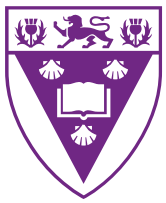

RHODES UNIVERSITY

Where leaders learn

KalCal: A Novel Calibration Framework for Radio Interferometry using the Kalman Filter and Smoother

---

Author:
Brian Welman

Supervisor:
Dr Landman Bester

Co-supervisor(s):
Dr Jonathan Kenyon & Prof Oleg Smirnov

A thesis submitted in fulfilment of the requirements
for the degree of Master of Science

in the

Centre for Radio Astronomy Techniques and Technologies
Department of Physics and Electronics

of

Rhodes University

The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF.

September, 2024

Abstract

Calibration in radio interferometry is essential for correcting measurement errors. Traditional methods employ maximum likelihood techniques and non-linear least squares solvers but face challenges due to the data volumes and increased noise sensitivity of contemporary instruments such as MeerKAT. A common approach for mitigating these issues is using "solution intervals", which helps manage the data volume and reduces overfitting. However, inappropriate interval sizes can degrade calibration quality, and determining optimal sizes is challenging, often relying on brute-force methods.

This study introduces *Kalman Filtering and Smoothing in Calibration* (KalCal), a new framework for calibration that combines the Kalman Filter, Kalman Smoother, and the energy function: the negative logarithm of the Bayesian evidence. KalCal offers Bayesian-optimal solutions as probability densities and models calibration effects with lower computational requirements than iterative approaches. Unlike traditional methods, which require all the data for a particular solution to be in memory simultaneously, KalCal's recursive computations only need a single pass through the data with appropriate prior information. The energy function provides the means for KalCal to determine this prior information.

Theoretical contributions include additions to complex optimisation literature and the "Kalman-Woodbury Identity" that reformulates the traditional Kalman Filter. A Python implementation of the KalCal framework was benchmarked against solution intervals as implemented in the QuartiCal package. Simulations show KalCal matching solution intervals in high *Signal-to-Noise Ratio* (SNR) scenarios and surpassing them in low SNR conditions. Moreover, the energy function produced minima that coincide with KalCal's *Mean Square Error* (MSE) on the true gain signal. This result is significant as the MSE is unavailable in real applications. Further research is needed to assess the computational feasibility and intricacies of KalCal.

Declaration of Authorship

I, Brian Welman, declare that this thesis titled, "KalCal: A Novel Calibration Framework for Radio Interferometry using the Kalman Filter and Smoother" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Brian Welman

Date: September, 2024

Acknowledgements

This thesis is the culmination of the combined efforts and support of numerous individuals and institutions, to whom I owe a profound debt of gratitude. I extend my sincere thanks to Rhodes University, SARAO, and NRF for their financial support and the academic opportunities they have provided. The RATT research group deserves special mention; my fellow RATTs have been instrumental in this journey, enriching it with invaluable knowledge, insightful discussions, and memorable social gatherings. I am profoundly grateful for the learning and camaraderie experienced. I am indebted to Dr. Ulrich Sob, whose expert guidance with my simulated experiment, calibration, and radio interferometry has been indispensable. Similarly, Cyndie Russeeawon's unwavering support and assistance, both theoretically and practically, have been invaluable, not only professionally but also in fostering a deep friendship. To Prof. Oleg Smirnov, I express my deepest gratitude for the opportunity to undertake this project under RATT. Your patience, wisdom, belief in my abilities, and kindness, especially during challenging times, have been a beacon of support. Dr. Jonathan Kenyon's contributions to my thesis have been immeasurable. From programming assistance to meticulous reviews of my academic writing, the dedication and effort you have invested have been priceless. Finally, a massive thank you to Dr. Landman Bester. Although we debated many aspects of the work, I am immensely thankful for your experience, knowledge, guidance, and patience, which, in conjunction with Dr. Kenyon, have been crucial to the fruition and refinement of this work. I am highly grateful for your supervision of this work and for empowering me with the confidence and critical thinking skills essential for navigating the complexities of this field.

To my family, spread from Bermuda to England, Mozambique, and Graaff-Reinet, the love and support you have extended have been a cornerstone of my journey. Nigel and Nita Hallowes, your invaluable support, especially in the latter stages, whether through family assistance, babysitting, or the simple act of preparing a meal, has been a tremendous source of strength. You have my deepest thanks and love. To Mom, your endless love, kindness, and encouragement are highly cherished. The sacrifices you've made for me to have this opportunity are immeasurable; for that, my love and gratitude know no bounds. To Dad, thank you for the meaningful heart-to-hearts about my work and life. Your unwavering belief in me and your motivational spirit during tough times have been foundational. Your love and sacrifices have paved the way for this achievement, and for that, I am eternally grateful. To my dearest Freya, Thor, Minos, Bastet, and Hera, your unconditional love and comforting presence have been a source of joy and solace. I love all of you so much. Finally and most importantly, I want to thank my wife, Lola and my son, Björn. To my son, I am so lucky to be your dad and to watch you grow up. I feel so privileged that after work, I got to spend the rest of my time with you. Even as you grow up, I will never forget the healing moments we had together, especially when watching rally racing videos or dinosaur documentaries or going to the arcade or skate-park. I hope this papa makes you proud, and I love you so much. To my beautiful wife, you are the number one reason why I was able to finish this thesis. You sacrificed so much to allow me to finish this work, even when times were tough. When

I was down and weak, you lifted me up and kept me going. You picked up where I could not. You made me hundreds of cups of amazing coffee and tea. You provided me so many opportunities to complete my work. I remember everything you did for me, and I will always be indebted to you for this. I hope I have made you proud and that you can share the infinite love and kindness you afforded me when you are in need. You truly are the most amazing person ever. Thank you, my love.

To my beautiful wife, Lola*,*

and my beloved son, Björn*,*

for their endless love, support, and inspiration.

"*There is nothing I cannot teach myself.*"

Contents

Abstract ii

Declaration of Authorship iii

Acknowledgements iv

List of Abbreviations and Acronyms xiii

List of Symbols xvi

List of Online Sources xx

1 Introduction 1
- 1.1 Content Structure . . . 4
- 1.2 Radio Interferometry . . . 5
  - 1.2.1 Interferometers . . . 5
  - 1.2.2 Visibilities . . . 7
  - 1.2.3 Calibration . . . 7
  - 1.2.4 Imaging . . . 11
- 1.3 Research Problem . . . 13
  - 1.3.1 Considerations for Solution Intervals . . . 13
  - 1.3.2 Non-linear Kalman Filters for Calibration . . . 17
  - 1.3.3 Improving upon the Filtering Approach . . . 19

2 Kalman Filters and Smoothers 23
- 2.1 Introduction . . . 23
  - 2.1.1 History . . . 24
  - 2.1.2 What is filtering and smoothing? . . . 25
  - 2.1.3 Uniqueness of the KFS . . . 26
- 2.2 Probability and Statistics Fundamentals . . . 27
  - 2.2.1 Probability Theory . . . 27
  - 2.2.2 Measurable Statistics . . . 29
  - 2.2.3 Complex Statistics . . . 31
  - 2.2.4 Gaussian Densities . . . 33
  - 2.2.5 Stochastic Processes . . . 34
  - 2.2.6 Inference and Estimation . . . 38
  - 2.2.7 The Bayesian Perspective . . . 40

2.3 Bayesian Filtering and Smoothing . . . 42
2.3.1 Bayesian Filtering Solution . . . 42
2.3.2 Kalman Filter . . . 45
2.3.3 Bayesian Smoothing Solution . . . 48
2.3.4 Kalman Smoother . . . 50
2.4 Extensions to KFS . . . 52
2.4.1 Wirtinger Calculus . . . 52
2.4.2 Non-linear signals . . . 54
2.4.3 Energy function . . . 57
2.4.4 The Kalman-Woodbury Identity . . . 59

3 KalCal **65**
3.1 Deriving from the RIME . . . 65
3.1.1 Calibration Problem . . . 65
3.1.2 Forming a Measurement Equation . . . 66
3.1.3 Evolution of Gains . . . 68
3.1.4 Augmenting the models . . . 69
3.2 Kalman Filtering and Smoothing in Calibration . . . 71
3.2.1 KalCal-Full . . . 71
3.2.2 KalCal-Diag . . . 74
3.3 Extensions to KalCal . . . 77
3.3.1 Whitening Transform . . . 77
3.3.2 Large Inverse Problem and the Kalman-Woodbury Identity . . . 79
3.3.3 Diagonal Approximation . . . 80
3.3.4 Energy function on Gains . . . 82

4 Application **85**
4.1 Simulation Setup . . . 85
4.1.1 Problem Contextualisation . . . 85
4.1.2 Simulated Observation Details and Sky Model . . . 87
4.1.3 Gain Signal . . . 91
4.1.4 Visibilities . . . 93
4.1.5 Calibration runs . . . 97
4.1.6 Evaluation Criteria . . . 97
4.2 Results and Analysis . . . 99
4.2.1 Validation of Simulation . . . 100
4.2.2 KalCal Analysis . . . 104
4.2.3 Component Solutions Analysis . . . 107
4.2.4 Energy Function Analysis . . . 111

5 Conclusion **114**
5.1 Summary of Key Findings . . . 114
5.1.1 The KalCal Approach . . . 114

5.1.2 Simulated Calibration Experiment 115
5.1.3 Kalman Smoother and Energy Function 115
5.2 Limitations and Future Research 116

A Additional Mathematics Definitions and Derivations **118**
A.1 Linear Algebra 118
A.2 Wirtinger Calculus 124
A.3 Taylor Series 125
A.4 Statistics and Probability 126

B Additional Figures and Tables **136**

Bibliography **153**

List of Figures

1.1 The Galactic Centre Imaged by MeerKAT (2022) . . . 2
1.2 Two-element Interferometer . . . 8
1.3 Visualising Solution Intervals . . . 15
1.4 Example Solution Interval Parameter Space . . . 17

4.1 The *uv*-coverage of Simulated Observation . . . 88
4.2 Source Position Density Process . . . 90
4.3 Simulated Sky Model . . . 92
4.4 Amplitude and Phase of Simulated Gains . . . 94
4.5 Real and Imaginary Parts of Simulated Gains . . . 95
4.6 MSE Results for QuartiCal in High *Signal-to-Noise Ratio* (SNR) . . . 100
4.7 RMS Results for QuartiCal in High SNR . . . 101
4.8 MSE Results for QuartiCal in Low SNR . . . 102
4.9 RMS Results for QuartiCal in Low SNR . . . 103
4.10 MSE Results in High SNR . . . 105
4.11 RMS Results in High SNR . . . 105
4.12 MSE Results in Low SNR . . . 106
4.13 RMS Results in Low SNR . . . 106
4.14 MSE Results for KalCal-Full Components in High SNR . . . 108
4.15 MSE Results for KalCal-Full Components in Low SNR . . . 109
4.16 MSE Results for KalCal-Diag Components in High SNR . . . 110
4.17 MSE Results for KalCal-Diag Components in Low SNR . . . 110

B.1 RMS Results for KalCal-Diag Components in High SNR . . . 138
B.2 RMS Results for KalCal-Diag Components in Low SNR . . . 138
B.3 RMS Results for KalCal-Full Components in High SNR . . . 139
B.4 RMS Results for KalCal-Full Components in Low SNR . . . 139
B.5 QuartiCal Real Part Estimates in High SNR . . . 141
B.6 QuartiCal Imaginary Part Estimates in High SNR . . . 142
B.7 QuartiCal Real Part Estimates in Low SNR . . . 143
B.8 QuartiCal Imaginary Part Estimates in Low SNR . . . 144
B.9 Real Part KalCal-Diag Estimates in High SNR . . . 145
B.10 Imaginary Part KalCal-Diag Estimates in High SNR . . . 146
B.11 Real Part KalCal-Diag Estimates in Low SNR . . . 147
B.12 Imaginary Part KalCal-Diag Estimates in Low SNR . . . 148

B.13 Real Part KalCal-Full Estimates in High SNR 149
B.14 Imaginary Part KalCal-Full Estimates in High SNR 150
B.15 Real Part KalCal-Full Estimates in Low SNR 151
B.16 Imaginary Part KalCal-Full Estimates in Low SNR 152

List of Tables

4.1 Calibration Scenario Codes 98

B.1 MSE Optimum Values in High SNR 137
B.2 MSE Optimum Values in Low SNR 137
B.3 RMS Optimum Values in High SNR 140
B.4 RMS Optimum Values in Low SNR 140

List of Abbreviations and Acronyms

AEKF	*Augmented Extended Kalman Filter*.
AEKFS	*Augmented Extended Kalman Filter and Smoother*.
AEKS	*Augmented Extended Kalman Smoother*.
AKF	*Augmented Kalman Filter*.
AKFS	*Augmented Kalman Filter and Smoother*.
AKS	*Augmented Kalman Smoother*.
ASKAP	*Australian Square Kilometer Array Pathfinder*.
BFS	*Bayesian Filtering and Smoothing*.
CASA	*Common Astronomy Software Applications*.
DDE	*Direction-Dependent Effect*.
DEC	*Declination*.
DFT	*Discrete Fourier Transform*.
DIE	*Direction-Independent Effect*.
EKF	*Extended Kalman Filter*.
EKFS	*Extended Kalman Filter and Smoother*.
EKS	*Extended Kalman Smoother*.
EKS	*Unscented Kalman Smoother*.
EM	*electromagnetic*.
EnKF	*Ensemble Kalman Filter*.
FFT	*Fast Fourier Transform*.
FOV	*Field-Of-View*.
IEKF	*Iterated Extended Kalman Filter*.
JTRF2014	*Jet Propulsion Laboratory Terrestrial Reference Frame 2014*.

KalCal	*Kalman Filtering and Smoothing in Calibration.*
KalCal-Full	*Pure KalCal.*
KalCal-Diag	*Diagonally Approximated KalCal.*
KF	*Kalman Filter.*
KFS	*Kalman Filter and Smoother.*
KP	*Kalman Predictor.*
KS	*Kalman Smoother.*
LOFAR	*Low Frequency Array.*
MAP	*Maximum A Posterior.*
MeerKAT	*Meer Karoo Array Telescope.*
ML	*Maximum Likelihood.*
MLE	*Maximum Likelihood Estimate.*
MMSE	*Minimum Mean Square Error.*
MP	*Modelled Flux Percentage.*
MSE	*Mean Square Error.*
NLL	*Negative Log-Likelihood.*
PSF	*Point Spread Function.*
RA	*Right Ascension.*
RIME	*Radio Interferometric Measurement Equation.*
RMS	*Root Mean Square Error.*
RTS Smoother	*Rauch-Tung-Striebel Smoother.*
SARAO	*South African Radio Astronomy Observatory.*
SBD	*Sky Brightness Distribution.*
SKA	*Square Kilometer Array.*
SMW	*Sherman-Morrison-Woodbury Identity.*
SNR	*Signal-to-Noise Ratio.*
UKF	*Unscented Kalman Filter.*
UKFS	*Unscented Kalman Filter and Smoother.*
vCZ	*van Cittert-Zerneke Theorem.*
VLBI	*Very Long Baseline Interferometer.*

WEKF	*Widely-Linear Extended Kalman Filter*.
WEKFS	*Widely-Linear Extended Kalman Filter and Smoother*.
WEKS	*Widely-Linear Extended Kalman Smoother*.
WKF	*Widely-Linear Kalman Filter*.
WKFS	*Widely-Linear Kalman Filter and Smoother*.
WKS	*Widely-Linear Kalman Smoother*.

List of Symbols

$x,\ X,\ x(t),\ X(t)$	Scalar values and functions.
$\mathbf{x},\ \mathbf{x}(t)$	Vector values and functions.
$\mathbf{X},\ \mathbf{X}(t)$	Matrix values and functions.
$\mathbb{X} = \{1, 2, 3\}$	Unordered Set.
$\mathbb{Y} = (1, 2, 3)$	Ordered Set.
Hz	Hertz Unit, a measure of frequency (s^{-1}).
Jy	Jansky Unit, a measure of spectral flux density ($10^{-23}\ \mathrm{Wm}^{-2}\mathrm{Hz}^{-1}$).
N_{ant}	Number of antennas used in a radio interferometer.
N_{bl}	Number of baselines used in a radio interferometer, depending on the value of N_{ant}.
N_{time}	Number of time steps for a scan, depending on the scan length and integration time used.
$\mathbb{A}$	Ordered set of Antenna Indices of length N_{ant}.
$\mathbb{B}$	Ordered set of Baseline Indices of length N_{bl}, constructed in upper-triangle format.
$\mathbb{T}$	Ordered set of Time Steps of length N_{time}.
θ_{res}	Angular Resolution.
c	Speed of Light.
λ	Wavelength.
lm	Direction Cosines of the Sky Plane.
uv	Direction Cosines of the Antenna Plane.
$\mathcal{I}(l, m, n)$	True Sky Brightness Distribution.
$\mathcal{I}_d(l, m, n)$	Apparent Sky Brightness Distribution.
$\mathcal{I}_b(l, m, n)$	Point Spread Function Distribution.
$\mathcal{V}(u, v, w)$	Visibility Function Distribution.
$\star$	Convolution Operator.

$(\,\cdot\,)^T$	Transpose Operator.
$(\,\cdot\,)^*$	Conjugate Operator.
$(\,\cdot\,)^H$	Adjoint Operator, Hermitian Transpose, or Conjugate Transpose.
$\mathcal{L}(\mathbf{e})$	Loss Function.
$\mathbf{V}_{pq}$	Visibility Matrix for baseline (p, q).
$\mathbf{J}_p,\ \mathbf{J}_q$	Jones Matrix for a given antenna index.
$\mathbf{B}$	Brightness Matrix for point source.
$\langle\,\cdot\,\rangle$	Averaging Operator.
$\mathbf{G}_p,\ \mathbf{G}_q$	G-Jones Matrix.
$\mathbf{E}_p,\ \mathbf{E}_q$	E-Jones Matrix.
$\phi_{pq}(l, m)$	Scalar phase function that details the geometric delay between antennas p and q.
$\mathbf{X}_{pq}$	Sky Coherency Matrix.
$(\,\hat{\cdot}\,)$	Estimate of Term.
$\mathbf{C}_{pq}$	Corrected Visibility Matrix.
Δ_t	Time (Solution) Interval Size.
Δ_ν	Frequency (Solution) Interval Size.
$[\, i \mid i \in \mathbb{N}\,]$	Vector-builder Notation.
$p(\mathbf{x})$	Probability Density Function of $\mathbf{x}$.
$p(\mathbf{x}, \mathbf{y})$	Joint Density Function of $\mathbf{x}$ and $\mathbf{y}$.
$p(\mathbf{x} \mid \mathbf{y})$	Conditional Density Function of $\mathbf{x}$ given $\mathbf{y}$.
$\mathrm{E}[x]$, $\mathrm{E}[\mathbf{x}]$	Expectation Operator.
$m_x,\ \mathbf{m}_\mathbf{x}$	Expectation values and vectors.
$\mathrm{Cov}[x, y],\ \mathrm{Cov}[\mathbf{x}, \mathbf{y}]$	Covariance Operator.
$\sigma_{xy},\ \mathbf{P}_\mathbf{xy},\ \mathbf{P}_\mathbf{x}$	Covariance values and matrices.
$\mathrm{Var}[\mathbf{x}]$	Variance Operator.
σ_x^2	Variance.
$\left[\, [x_k]_{ij} \;\middle|\; (x_k \in \mathbb{X})((i,j)_k \in \mathbb{N}^2) \,\right]$	Matrix-builder Notation.
$\widetilde{\mathbf{P}}_\mathbf{xy},\ \widetilde{\mathbf{P}}_\mathbf{x}$	Pseudo-Covariance Matrices.
$\widetilde{\mathbf{A}}$	Wirtinger Matrix.
$\underline{\mathbf{x}},\ \underline{\mathbf{X}}$	Augmented Vectors and Matrices.

$\mathcal{CN}(\mathbf{m}_\mathbf{x}, \mathbf{P}_\mathrm{x})$	Proper Complex Gaussian Density.
$\det(\,\cdot\,)$	Determinant Operator.
$\mathcal{CN}(\mathbf{m}_\mathbf{x}, \mathbf{P}_\mathrm{m}, \widetilde{\mathbf{P}}_\mathrm{m})$	General Complex Gaussian Density.
$\mathcal{CN}(\underline{\mathbf{m}}_\mathbf{x}, \underline{\mathbf{P}}_\mathrm{m})$	Augmented Complex Gaussian Density.
$\mathcal{CN}(\mathbf{x} \mid \mathbf{m}_{\mathbf{x}\|\mathbf{y}}, \mathbf{P}_{\mathbf{x}\|\mathbf{y}})$	Conditional (Proper) Complex Gaussian Density of $\mathbf{x}$ given $\mathbf{y}$.
$\mathbf{x}(t)$	Signal Function of Hidden Signal.
$\mathbf{x}_{1:N} = (\mathbf{x}_1, ..., \mathbf{x}_N)$	Stochastic Process.
$\sigma_Q,\ \mathbf{q}_k,\ \mathbf{Q}_k$	Process Noise Values, Vectors and Matrices.
$\mathcal{CP}(\mathbf{m}(t), \mathbf{P}(t))$	Complex Gaussian Process defined by a mean and covariance function.
$\mathbf{x}_k$	State Vector at time k.
$\mathbf{f}(\mathbf{x}_k)$	Evolution Function on state $\mathbf{x}_k$.
$\mathbf{y}(t)$	Signal Function of Measurable Signal.
$\sigma_R,\ \mathbf{r}_k,\ \mathbf{R}_k$	Measurement Noise Values, Vectors and Matrices.
$\mathbf{y}_k$	Measurement Vector at time k.
$\mathbf{h}(\mathbf{x}_k)$	Measurement Function on state $\mathbf{x}_k$.
$\hat{\mathbf{x}}$	Estimate of vector $\mathbf{x}$.
$\mathbf{e}$	Error or Residual Vector.
$\mathbf{S}$	Error or Residual Covariance Matrix.
$p(\mathbf{x}_k \mid \mathbf{x}_{k-1})$	Evolution Density from time $k-1$ to k.
$p(\mathbf{y}_k \mid \mathbf{x}_k)$	Measurement Density at time k.
$p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1})$	Prediction Density at time k.
$p(\mathbf{x}_k \mid \mathbf{y}_{1:k})$	Filtering Density at time k.
$\mathbf{F}_k$	Evolution matrix at time k.
$\mathbf{H}_k$	Measurement matrix at time k.
$\mathbf{m}_{p,k},\ \mathbf{P}_{p,k}$	Predictor Mean and Covariance at time k.
$\mathbf{m}_{f,k},\ \mathbf{P}_{f,k}$	Filter Mean and Covariance at time k.
$\mathbf{K}_k$	Kalman Gain at time k.
$p(\mathbf{x}_k \mid \mathbf{y}_{1:N})$	Smoothing Density at time k.
$\mathbf{m}_{s,k},\ \mathbf{P}_{s,k}$	Smoother Mean and Covariance at time k.

$\mathbf{W}_k$	Smoothing (Kalman) Gain at time k.
$\frac{\partial}{\partial \mathbf{x}}$, $\frac{\partial}{\partial \mathbf{x}^*}$	Wirtinger Derivatives.
φ_k	Energy Function at time k.
$\mathbf{U}_k$	$J^H J$-computation at time k.
$\mathbf{z}_k$	$J^H r$-computation at time k.
$\mathbf{v}$	Visibility Vector.
$\mathbf{g}$	Gains Vector.
$\text{chol}(\,\cdot\,)$	Cholesky Decomposition Operator.
$\text{diag}(\,\cdot\,)$	Diagonal Operator.
$\odot$	Hadamard (Element-wise) Product.
$\oslash$	Hadamard (Element-wise) Division.
$(\,\cdot\,)^{(k)}$	Hadamard (Element-wise) Power.
$(\,\cdot\,)^{(-1)}$	Hadamard (Element-wise) Inverse.
N_{src}	Number of Sources.
$\mathbf{a}(t)$	Gain Amplitude Function.
$\phi(t)$	Gain Phase Function.
$\mathbf{P}_{\text{SE}}$	Squared Exponential Covariance Matrix.
$\mathcal{GP}(\mathbf{m}(t), \mathbf{P}(t))$	Real Gaussian Process defined by a mean and covariance function.
$\Re(\mathbf{x})$	Real Part of $\mathbf{x}$.
$\Im(\mathbf{x})$	Imaginary Part of $\mathbf{x}$.
$\lvert\,\cdot\,\rvert$	Absolute Value Operator.
$\lVert\,\cdot\,\rVert$	Complex Norm Operator.

List of Online Sources

codex-africanus	Common Radio Astronomy Software and Tools. https://github.com/ratt-ru/codex-africanus.
CubiCal	CubiCal Calibration Software. *Kenyon* et al. *[1]*. https://github.com/ratt-ru/CubiCal.
Dask	An open-source Python library for parallel computing. https://www.dask.org/.
ducc0	Distinctly Useful Code Collection (DUCC). https://github.com/mreineck/ducc.
killMS	killMS Calibration Software. *Tasse [2]*. https://github.com/saopicc/killMS.
python3	Python Programming Language. https://docs.python.org/3/.
QuartiCal	QuartiCal Calibration Software. *Kenyon, Perkins, and Smirnov [3]*. https://github.com/ratt-ru/QuartiCal.
SAGECal	A fast, memory efficient, GPU accelerated radio interferometric calibration program. *Kazemi* et al. *[4]*. https://sagecal.sourceforge.net/.
simms	Measurement Set Simulator. https://github.com/ratt-ru/simms.
StEFCal	Statistically Efficient and Fast Calibration. *Salvini and Wijnholds [5]*.

Chapter 1

Introduction

The 2018 launch of the *Meer Karoo Array Telescope* (MeerKAT) instrument in South Africa marked a significant milestone in modern radio astronomy. MeerKAT is a *radio telescope array*, or *radio interferometer*, an array of antennas functioning as a single radio telescope with higher *angular resolution* [6]. Using radio interferometry techniques, MeerKAT has been able to demonstrate remarkable observational abilities by transforming radio signals received by the interferometer into images of the celestial sky [7]. By providing unprecedented insights into the centre of our galaxy (see fig. 1.1), astronomers around the world have been equipped with the means to deepen their understanding of the universe we inhabit [8], [9]. The advent of large interferometers such as MeerKAT, *Low Frequency Array* (LOFAR) and the upcoming *Square Kilometer Array* (SKA) has spurred a paradigm shift in radio interferometry, ushering in a new era of *Big Data*.[1] The enhanced sensitivity of these instruments has led to a surge in data volume and detail. This, in turn, drives ongoing advances in research, algorithms, and computational resources to improve image quality [11]–[13].

This paradigm shift is particularly visible in the *calibration* subfield. There are numerous ways in which the received signal can be corrupted, thus negatively impacting the quality of the resulting images. These corruptions, traditionally termed as *gains*, can be caused by *instrumental effects*, e.g. corruptions induced by electronic systems, or *atmospheric effects*, e.g. ionospheric phase changes. These gains are corrected by performing calibration to enhance image quality and, consequently, the scientific knowledge derived from these data products [11], [15], [16]. In general, the process of calibration is two-fold. The first step is to find and model the relationship between the measurement of an instrument and the actual value of the subject of the measurement. The second step is to use this model to correct the measurements to ensure they are as close to the truth as possible [17]. The *computational complexity*[2] of calibration, in general, increases with larger arrays due to the larger volume of measurement data. Furthermore, calibration becomes more complicated as the sensitivity of the instrument increases. As a result, this requires researching and developing better modelling and estimation methods for calibration to handle these obstacles. Thus, it is an

[1] *Big Data*: An all-encompassing term referring to the need or ability to analyse large amounts of information and gain valuable insights not attainable using traditional methods and smaller datasets [10].

[2] *Computational Complexity*: A measure of the difficulty of solving a particular problem computationally. This analysis looks at the time and storage space required to find the solution, such as the number of calculations and the amount of memory needed to store the data [18].

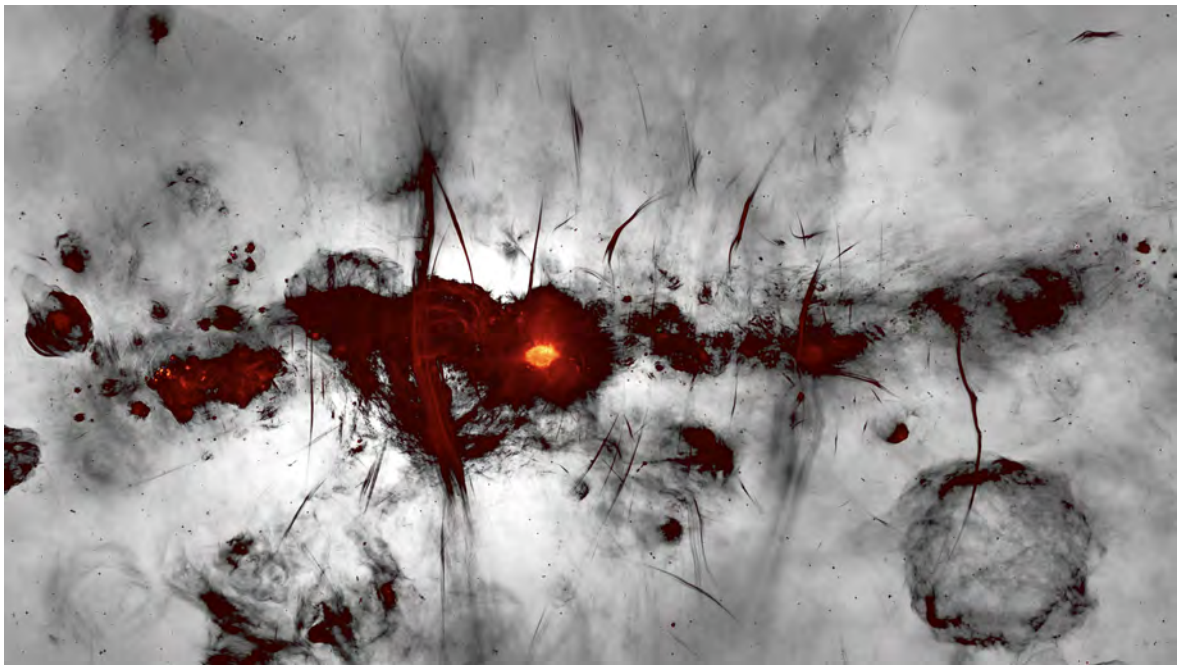

FIGURE 1.1: One of the newest MeerKAT images of the Milky Way's centre, showcasing the instrument's capability to produce high-resolution images and providing astronomers with detailed insights into intricate astronomical objects such as the heart of our galaxy. Credit: South African Radio Astronomy Observatory - SARAO [14].

active research area in calibration to create algorithms that are also suitable in the era of Big Data (see [1], [3], [4], [11], [12], [19]).

Despite many successes in the field, several challenges persist, particularly with the use of *solution intervals* in calibration. As noted by Sob *et al.* [20], this ad hoc *regularisation*[3] technique has its complications. Specifically, it approximates non-linear gain signals across time and frequency as piecewise-constant functions over equally spaced intervals of a particular width known as a *solution interval*. This is advantageous because it regularises the estimated gains by reducing the influence of noise and diminishes the computational strain by decreasing the number of parameters to estimate, i.e. one parameter is estimated per solution interval. However, Sob *et al.* [20] discusses the disadvantages of this strategy. Finding optimal choices for solution interval sizes is non-trivial and computationally expensive. If an optimal solution interval is found, it will not necessarily produce the best possible calibration results due to the underlying piecewise-constant approximations involved. Furthermore, there are certain conditions in which the use of such a technique can be detrimental, e.g. the occurrence of "flux suppression" effects caused by image artefacts or "ghost sources" [20], [22]. Sob *et al.* emphasised that although solution intervals have advantages in reducing the effects of noise and computational cost, no alternative has yet been developed that offers these benefits without the accompanying drawbacks. Given these challenges, Sob *et al.* [20] proposed alternative calibration approaches, including the *Kalman Filter* (KF) method introduced into calibration by Tasse [11].

[3] *Regularisation*: The incorporation of a penalty term into estimation processes to improve generalisation and stabilise the solutions across a broader range of statistical conditions [21].

This statistical estimation algorithm utilises *Bayesian inference* techniques: it incorporates additional known or *prior* information into the probabilistic model to regularise the subsequent estimates derived from measurements. Moreover, it computes these estimates recursively along a given axis, such as the time or frequency axis. Uniquely, the KF generates solutions at each step using current measurement information and the estimate of the previous step. Theoretically, the estimation algorithm passes through the data only once. Thus, the KF can use the power of Bayesian statistics to generate accurate and robust estimates while reducing the computational complexity of creating them [11], [23], [24]. Building on the work of Tasse [11], who developed a calibration algorithm using the KF to estimate and correct non-linear direction-dependent gains, this research aims to refine their approach as an alternative to the problematic solution intervals strategy. This involves addressing the limitations of the KF in the calibration context. The KF is ideal for real-time or *online* estimation problems where future measurement information is unavailable. Calibration is traditionally an *off-line* estimation problem where all measurements over time would be available. This means the KF is less suitable for calibration because earlier filter estimates are not informed by all measurements. Therefore, the approach requires additional regularisation to improve the KF's solutions. Although Tasse [11] dealt with this issue by incorporating a form of "Tikhonov" regularisation, this research aims to explore the use of the *Rauch-Tung-Striebel Smoother* (RTS Smoother), or *Kalman Smoother* (KS), to address this limitation of the KF. The RTS Smoother was created to solve this problem by filling in the gaps of missing measurement information left behind by the KF to formulate estimates that consider all measurements. It is designed to work in tandem with the KF to regularise or *smooth* the solutions and enhance the quality of the estimates. Furthermore, the RTS Smoother is often less computationally intensive because it exploits the calculations already computed by the KF [24], [25]. Consequently, the *Kalman Filter and Smoother* (KFS) is a promising alternative to more traditional calibration methods.

When looking at the utilisation of the KFS for practical purposes, it is essential to take into account the selection of suitable *prior information*, such as initial estimates, noise assumptions, and state-transition functions, which adds an extra layer of computational difficulty.[4] Without automated methods to determine these optimal priors, calibration via KFS could prove cumbersome and result in the same issues experienced with solution intervals. While Tasse [11] successfully overcame this obstacle through various additions to the KF, this study explores an alternative approach discussed by Särkkä [24]. Specifically, this research will investigate the computation of the recursive form of the *marginal likelihood* or *Bayesian evidence* called the *energy function*. The Bayesian evidence measures the model's fit to the observed data; consequently, it can be used to optimise the initial model setup, which includes selecting appropriate priors. Moreover, the energy function uses terms that are computed during the evaluation of the KF. Therefore, incorporating the energy function into the KFS framework could allow real-time adjustments of prior terms, potentially amplifying the utility

[4] *Prior information* or *Priors*: Refers to pre-existing knowledge or assumptions about the probabilistic model before taking any observations into account [23], [24].

of this estimation approach. Moreover, this capability opens the door to "intelligent" calibration algorithms that autonomously select optimal priors. This is particularly valuable in calibration, where determining such information can be difficult and computationally expensive. Therefore, in conjunction with research on the KFS approach, another focal point is to investigate whether the energy function can provide such optimality information. Together, these research components culminate in a new calibration framework, dubbed *Kalman Filtering and Smoothing in Calibration* (KalCal), which aims to mitigate the challenges associated with solution intervals and provide improved calibration results.

1.1 Content Structure

The research structure addresses the topics needed to develop the KalCal algorithm and the simulated experiment to evaluate it against the objectives mentioned above.

- **Chapter 1, Introduction** - This chapter develops the context of this research, providing a solid literary foundation. It introduces radio interferometry, the *Radio Interferometric Measurement Equation* (RIME), and calibration. Additionally, it discusses the research problem, focusing on concerns with the solution intervals technique, the application of the *Kalman Filter* (KF) in calibration, and enhancements through the *Kalman Smoother* (KS) and energy function.

- **Chapter 2, Kalman Filters and Smoothers** - This chapter lays out the fundamental theory behind Bayesian Filtering and Smoothing, leading up to the *Kalman Filter and Smoother* (KFS). It begins with a brief history of the KFS, followed by an overview of probability and statistics basics. The chapter then delves into *Bayesian Filtering and Smoothing* (BFS) theory, deriving the KFS, and concludes with extensions to the KFS theory.

- **Chapter 3, KalCal** - Using the RIME framework, this chapter describes the specific calibration problem addressed in this research. It then develops and implements the Kalman Filter Smoother calibration algorithm framework, *Kalman Filtering and Smoothing in Calibration* (KalCal). The chapter starts with framing the calibration problem in terms of RIME. It presents the KalCal-Full and KalCal-Diag variants of KalCal, concluding with a discussion on the extensions included in the algorithms.

- **Chapter 4, Application** - This chapter involves creating a simulated calibration experiment to evaluate the KalCal algorithms, as constructed in the previous chapter. It discusses the setup of the simulation and the criteria for evaluation. Following this, it provides an in-depth analysis of the results and outcomes of the experiment to address the associated research questions.

- **Chapter 5, Conclusion** - The final chapter synthesises the work presented in the dissertation and the results from the application chapter. It offers a summary of key findings, discusses the limitations of the research, and suggests potential avenues for future research.

To support the main content within this manuscript without unnecessary complications, two appendices are included:

- **Appendix A, Additional Mathematics Definitions and Derivations** - Contains all mathematical definitions, theorems, lemmas, properties and remarks that support the various derivations and discussion within the dissertation, especially for chapters 2 and 3. Moreover, various proofs of the presented algorithms are placed here, too.
- **Appendix B, Additional Figures and Tables** - Consists of additional figures and tables created during the calibration experiment in chapter 4. They offer additional support to the discussion in §4.2.

1.2 Radio Interferometry

The following section provides an introduction to the subject of radio interferometry. The discussion will focus on the steps from capturing radio signals with interferometers to reconstructing the corresponding image of the sky. It will highlight the topics that relate the most to this research.

1.2.1 Interferometers

Radio astronomy involves studying *electromagnetic* (EM) radiation within the radio band emitted from celestial objects called *radio sources* [26]. These emissions, known as *radio signals*, span a frequency range of 10 MHz to 1 THz. Observing within this band of frequencies offers astronomers a distinctive view of celestial sources. Specifically, radio astronomy enables the study of unique physical processes and attributes that are primarily or exclusively evident at these radio wavelengths. Furthermore, radio signals have a wider observing window from Earth compared to other EM signals, further enriching the information available to analyse and learn from these celestial objects [27].

There are a variety of mechanisms by which radio signals can be emitted from astronomical sources. Some examples are thermal processes, such as blackbody radiation, e.g. the Sun, or non-thermal processes, such as synchrotron emission, e.g. pulsars and Cygnus A [26], [27]. High *angular resolution* is often required to achieve certain science goals. This denotes a telescope's ability to differentiate between small features within the celestial body under observation. A telescope's resolution, symbolised as θ_{res} [26], is approximately computed using the formula

$$\theta_{\text{res}} \approx \frac{\lambda}{D}. \tag{1.1}$$

Here, λ represents the *wavelength* of the observed EM signal, and D is the *diameter* of the telescope's dish or *aperture*. Given the relatively longer wavelengths of radio waves compared to optical, infrared or ultraviolet radiation, achieving high resolution requires radio telescopes to have extremely large apertures; however, such a requirement poses challenges in terms of impractical cost, material availability, and engineering expertise [26].

Radio interferometers are designed to overcome this obstacle by emulating a large aperture telescope with multiple smaller telescopes through the process called *aperture synthesis*. This technique combines the signals received from the different antennas, taking into account the staggered reception of the signal due to spatial separation of the antennas, to construct a high-resolution image of the sky [27]. However, the completeness of this construction depends on the interferometer's sampling capability of what are known as *spatial frequencies* during the observation. If the image of the sky is fixed on a two-dimensional coordinate system called the *lm-plane*, then a spatial frequency within this plane represents the rate of change of structure and detail of the source within the image at a specific scale. The terms l and m are called *direction cosines* and, conventionally, l represents the East-West component and m the North-South component of the plane [28]. Combining this information with sinusoid functions defined on the lm-plane across different spatial frequencies allows the reconstruction of the image through the *Fourier transform*, the mathematical operation that relates the waveforms to their constituent sinusoid functions [17].[5] Therefore, to improve the accuracy of this reconstruction, the interferometer must sample a sufficient distribution of spatial frequencies [28].

In an interferometer with some fixed, arbitrarily chosen *phase centre*,[6] the positions of all antennas uniquely determine the spatial frequencies that the interferometer can sample. Specifically, the spatial frequencies sampled by the interferometer are determined by the relative distance between all pairwise combinations of antennas, called *baselines* [29]. These baselines are assigned coordinates on a plane known as the *uv-plane*, the Fourier dual of the lm-plane defined above. Thus, the array's configuration in the uv-plane directly influences how effectively the instrument samples the sky. This sampling response is called the *instantaneous uv-coverage*, or simply *uv-coverage*, of the instrument [26]. This coverage is enhanced by making multiple observations over time. Specifically, if the lm-plane is fixed, then from the perspective of this sky frame, the antennas on Earth would move over time, and individual baselines would trace out ellipses that fill in the uv-plane in a process known as *Earth rotation synthesis*. Hence, the uv-coverage improves over time. Finally, since the antennas' positions are defined by wavelength units, the length of baselines is inversely proportional to spatial frequencies. That is, with shorter (longer) baselines, larger (smaller) spatial frequencies can be captured, and broader (finer) details and structures can be discerned. Therefore, by optimising the antenna layout to maximise uv-coverage over time, frequency, and baseline lengths, the reconstruction, that is, the quality of the original image, is improved [26], [27].

From the above facts, an interferometer's resolution is determined by the lengths of its baselines. The highest resolution it can achieve depends on the length of its longest baseline. Thus, interferometers can achieve better θ_{res} than single aperture telescopes by including longer baselines in their layout. Consequently, they overcome the limitations of single-dish radio telescopes at higher resolutions, demonstrating why interferometers are essential for viewing within the radio spectrum.

[5] In practical cases, the *Fast Fourier Transform* (FFT), a discretised and optimised form of the Fourier transform, is used [27].

[6] *Phase Centre* or *Field Centre*: A reference point in the sky from which relative phase measurements can be made. It is typically chosen as the position of the target source or nearby bright source [28], [29].

1.2.2 Visibilities

Each antenna captures the voltage induced across its receivers by the radio signal at precise time and frequency intervals to map the spatial frequencies of the sky. As mentioned above, staggered reception of the signal introduces a time delay when the antennas record the same signal. This introduces a delay in the measured voltage phase that is corrected for. Once phase-corrected, these voltages are correlated and averaged with those from other antennas to form complex-valued *visibilities*, the Fourier component defined on the uv-plane [27]. These visibilities are the fundamental measurement data of interferometers due to their relationship to the *Sky Brightness Distribution* (SBD).[7] This distribution describes the random nature of a given variable, e.g. sky brightness, through some probabilistic model [30]. Let $\mathcal{I}$ be the *true* SBD defined on the lm-plane, i.e. it captures the randomness perfectly. In the ideal case, all corresponding visibilities can be observed and represented as the *visibility* function $\mathcal{V}$ defined over the uv-plane. In that case, the relationship between the visibilities and the true SBD is described in the following two-dimensional Fourier transform [27], [29]:

$$\mathcal{V}(u,v) = \iint \mathcal{I}(l,m)e^{-2\pi i(ul+vm)}\,\mathrm{d}l\mathrm{d}m. \tag{1.2}$$

This is a form of the famous *van Cittert-Zerneke Theorem* (vCZ) on which radio interferometry is based. Equation (1.2) represents the ideal scenario, but a radio interferometer can only measure a noisy and incomplete version of $\mathcal{V}$ due to sampling limitations. The SBD obtained from incomplete visibilities is known as the *dirty* SBD and will be denoted as $\mathcal{I}_d$. The interferometer can observe the dirty image directly without additional prior information, and more accurate reconstructions can be obtained by incorporating additional prior information. This way, an attempt can be made to reconstruct the true SBD from the measured visibilities. A typical representation of the discussion thus far is given in fig. 1.2.

1.2.3 Calibration

A critical stage in recovering the true SBD is calibrating and correcting the measured visibilities for radio signal corruptions. To re-iterate, calibration in radio interferometry is the steps involved in correcting the visibilities for corruptions incorporated during the signal's propagation from the source to the receiver. By correcting these measured visibilities, the accuracy of the visibility density improves and subsequently, so does the recovery of the true sky. Equation (1.2) forms the basis of the imaging problem; however, it does not account for the corruption of radio signals. For this, the succinct *Radio Interferometric Measurement Equation* (RIME) framework is conventionally used to describe and derive solutions to calibrate for these signal alterations. Created by Hamaker, Bregman, and Sault [31] and further developed by Smirnov [15], [16], the RIME is a reformulation of eq. (1.2) using *Jones calculus* to quickly and directly incorporate models of these signal corruptions. Once the visibilities have been corrected, the RIME reduces to the vCZ relation, and the imaging step

[7] In truth, the "sky brightness distribution" in this context represents a *probability density function* and not necessarily a *probabilistic distribution function* (see chapter 2). However, the naming convention is kept to be consistent with terminology.

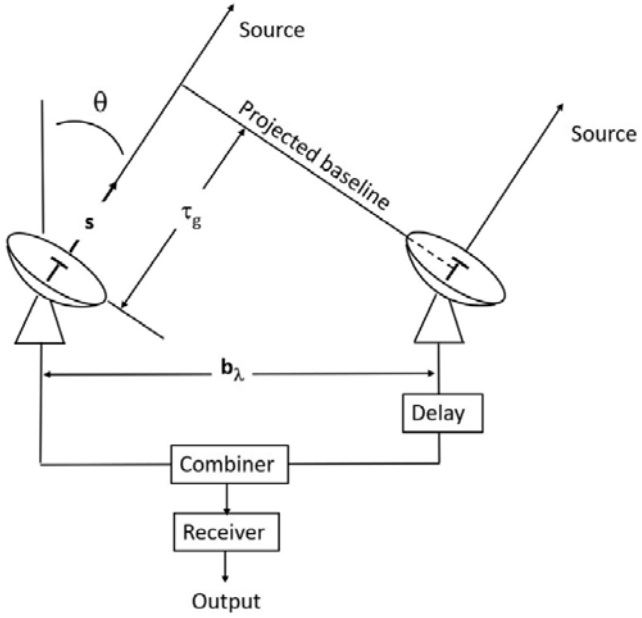

FIGURE 1.2: A conceptual diagram of a classic two-element interferometer by Burke, Graham-Smith, and Wilkinson [27]. The diagram consists of an antenna pair separated by a distance $\mathbf{b}_\lambda$. They point at a target source in the direction of $\mathbf{s}$ and observe the same radio emission at some angle θ. A delay τ_g is experienced since the same wavefront of the emission reaches the two antennas at different points in time. The voltages are captured, the delay is corrected, and the results are combined. The general output of this is the visibilities measured by the baseline pair.

can commence. The RIME will be used for the mathematical basis of calibration within this research. For this, a few indexing conventions that will be used throughout the investigation must be introduced:

- *Number of time steps*, N_{time} - The total number of time steps or time bins within a given observation. It is assumed to be equally spaced.
- *Number of antennas*, N_{ant} - The number of antennas in the radio interferometer.
- *Number of baselines*, N_{bl} - The number of unique baselines within the radio interferometer.
- *Time set*, or *time frame*, $\mathbb{T} = (1, ..., N_{\text{time}})$ - An ordered set of time steps indices based on N_{time}.
- *Antenna set*, $\mathbb{A} = (1, ..., N_{\text{ant}})$ - An ordered set of antenna indices based on N_{ant}.
- *Baseline set*, $\mathbb{B} = \Big((p, q) \;\Big|\; (p, q) \in \mathbb{A}^2 : \; p < q \Big)$ - An ordered set of baselines in *upper triangle* form that represents all uniquely combined antennas. For *lower-triangle* form, flip the conditional inequality.[8]

[8] The terms "upper triangle" and "lower triangle" refer to the element indices associated with the upper and lower triangles of a $N_{\text{ant}} \times N_{\text{ant}}$ matrix.

The theory presented in the remainder of this chapter is based on Smirnov [15], [16] and Hamaker, Bregman, and Sault [31]. For a specific point in time and frequency, represent the resulting radio signal from a point source at the phase centre as the complex vector $\mathbf{e} \in \mathbb{C}^2$ and the corresponding visibility for the baseline $(p, q) \in \mathbb{B}$ as the 2×2 matrix $\mathbf{V}_{pq}$. The former and latter terms are known as the *signal* vector and *visibility* matrix, respectively. If all signal corruptions experienced by $\mathbf{e}$ along the paths to antennas p and q are expressed as 2×2 matrices denoted $\mathbf{J}_p$ and $\mathbf{J}_q$, respectively, then the RIME to represent this relationship between $\mathbf{e}$ and $\mathbf{V}_{pq}$ with corruption effects can be written as

$$\mathbf{V}_{pq} = \mathbf{J}_p \mathbf{B} \mathbf{J}_q^H. \tag{1.3}$$

The matrix $\mathbf{B}$ is called the *brightness* matrix and is expressed as $\mathbf{B} = 2\langle \mathbf{e}\mathbf{e}^H \rangle$, where $\langle \cdot \rangle$ is an *averaging* operator in time and frequency. It represents the sky brightness distribution of the point source. Conventionally, the matrices $\mathbf{J}_p$ and $\mathbf{J}_q$ are called *Jones* matrices due to the Jones calculus that RIME is based on. The convenience of this formulation is that corruptions can be isolated to linear 2×2 transforms of the original radio signal. In addition, the total corruption effect experienced by $\mathbf{e}$ along a distinct path can be represented as a product of individual Jones matrices, where each Jones matrix represents a certain corruption, and the order of the matrices indicates the order of occurrence. This is called a *Jones chain* and is presented as $\mathbf{J} = \prod_{i=1}^{n} \mathbf{J}_i$ if the signal encountered $n > 1$ corruptions. This allows for the formulation of the RIME that depends on the calibration needs of the user. If the gains must be modelled as explicit physical phenomena, individual physically motivated or *physical* Jones terms are used. An example would be those that account for the effects on phase due to physical interactions with the ionosphere. Otherwise, if the origin of individual effects is unimportant, the corruption effects can be summarised as a single Jones matrix equivalent to the total product of a Jones chain of each effect. This is known as a *phenomenological* Jones term.

Equation (1.3) can be extended to represent the continuous sky distribution in all directions instead of the direction of discrete sources. This is done by projecting the signal paths defined over a unit sphere centred at the baseline onto a lm-plane also tangential and perpendicular to the phase centre. This is called the *full sky* RIME [15] and is written as follows for baseline $(p, q) \in \mathbb{B}$:

$$\mathbf{V}_{pq} = \iint \mathbf{J}_p(l, m) \mathbf{B}(l, m) e^{-2\pi i \phi_{pq}(l,m)} \mathbf{J}_q^H(l, m) \, \frac{\mathrm{d}l \mathrm{d}m}{n}, \tag{1.4}$$

where $n = \sqrt{1 - l^2 - m^2}$ and

$$\phi_{pq}(l, m) = (u_p - u_q)l + (v_p - v_q)m + (w_p - w_q)(n - 1). \tag{1.5}$$

Now, the brightness matrix becomes a matrix function that describes the brightness in all directions defined over lm-plane. The scalar function ϕ_{pq}, called the *phase delay*, describes the phase terms that account for the geometric time delay induced by the staggered reception of radio signals between antennas p and q. The phase delay is often represented as an

individual Jones term called the *K-Jones* denoted as $\mathbf{K}_p$ and $\mathbf{K}_q$ for each antenna p and q, respectively; however, for this discussion, the scalar function representation is used. In addition, the Jones matrices are defined as functions over the lm-plane as $\mathbf{J}_p(l,m)$ and $\mathbf{J}_q(l,m)$ to represent the gains corruptions applied to the signal in the same direction for antennas p and q, respectively. Note, eq. (1.4) is a more realistic representation of the idealised eq. (1.2). From this point, specific calibration problems can be formed. In the context of this research, the phenomenological form of the RIME will be used. The functions $\mathbf{J}_p(l,m)$ and $\mathbf{J}_q(l,m)$ that represent Jones chains will be decomposed into two categories of effects: a *Direction-Independent Effect* (DIE) and a *Direction-Dependent Effect* (DDE). The significant difference is whether the corruption effects vary in direction, that is, in terms of (l,m). An example of a DIE is corruptions induced by hardware imperfections or temperature fluctuations. For the latter, the previously mentioned ionospheric effects are a well-known DDE where the modification of a signal's phase depends on the point on the ionosphere it travelled through.

The phenomenological Jones term used to capture DIEs is known as the *G-Jones* and can be written for antenna $p \in \mathbb{A}$ as

$$\mathbf{G}_p = \begin{bmatrix} g_x & 0 \\ 0 & g_y \end{bmatrix}, \tag{1.6}$$

if a linear antenna feed configuration is used. Equivalent expressions exist for circular feed configurations. Here, $g_x, g_y \in \mathbb{C}$ are complex scalars called *antenna gains*, or *gains*, and determine the direction-invariant corruption effects per feed-axis for a given receiver feed setup, e.g. linear or circular feeds. Since $\mathbf{G}_p$ is constant on the lm-plane, it represents a single matrix. The equivalent DDE term is called the *E-Jones* and includes the DDEs in a given calibration situation. It is a matrix function defined over the lm-plane, denoted as $\mathbf{E}_p(l,m)$ for the same antenna p, and described similarly to eq. (1.6) but represents the corruptions associated within a given direction. Using these definitions, the full-sky RIME in eq. (1.4) can be rewritten in terms of phenomenological G-Jones and E-Jones. The resulting expression is called the *general full-sky* RIME [15] and is expressed for the baseline $(p,q) \in \mathbb{B}$ as

$$\mathbf{V}_{pq} = \mathbf{G}_p \left(\iint \mathbf{E}_p(l,m)\mathbf{B}(l,m)e^{-2\pi i \phi_{pq}(l,m)} \mathbf{E}_q^H(l,m) \, \frac{\mathrm{d}l\mathrm{d}m}{n} \right) \mathbf{G}_q^H, \tag{1.7}$$

where $\phi_{pq}(l,m)$ is defined as before. Additionally, the terms $\mathbf{G}_p$ and $\mathbf{G}_q$ fall outside the integral since they are constant over direction. For this research, DDEs will not be considered to reduce the complexity of the RIME model, and thus eq. (1.7) can be re-expressed as

$$\mathbf{V}_{pq} = \mathbf{G}_p \mathbf{X}_{pq} \mathbf{G}_q^H, \tag{1.8}$$

where

$$\mathbf{X}_{pq} = \iint \mathbf{B}(l,m) e^{-2\pi i \phi_{pq}(l,m)} \, \frac{\mathrm{d}l\mathrm{d}m}{n}. \tag{1.9}$$

The new matrix $\mathbf{X}_{pq}$ is called the *sky coherency* matrix, representing the visibility that would be measured in the absence of the corrupting gains in all directions of the lm-plane.

From eq. (1.8), it is trivial to see that the goal of calibration is to solve for the G-Jones terms for all $p, q \in \mathbb{A}$. If the estimations of these terms are represented as $\hat{\mathbf{G}}_p$ and $\hat{\mathbf{G}}_q$, respectively, where $\hat{(\cdot)}$ denotes an estimate of a given term, then the effects of the gains can be undone as follows:

$$\mathbf{C}_{pq} = \hat{\mathbf{G}}_p^{-1} \mathbf{V}_{pq} \hat{\mathbf{G}}_q^{-1}. \tag{1.10}$$

This is called the *correction* step, and the term $\mathbf{C}_{pq}$ is known as the *corrected* visibility matrix. By performing this step, subsequent images created from this corrected data will represent a version of the sky that is more consistent with the true sky. The degree of truth derived from these visibilities depends on the accuracy of the estimated gains terms and the amount of noise included with the measured visibility. The visibility data's inherent SNR is a crucial factor influencing the accuracy of this correction step. The SNR measures how well the desired signal stands out against background noise incorporated within the measurements. That is, the signal of interest consists of more power than the noise. Within a high SNR regime, the measured visibilities have a higher-quality with respect to gain and noise errors. Therefore, the derived calibration solutions are accurately resolved and capture the majority of these errors. The result would be high-quality calibrated visibilities. However, with low SNR, these solutions are likely to be less accurate, as the signal would be weak relative to the noise, leading to poor quality calibrated visibilities. The level of noise changes between different calibration scenarios, and thus, a robust calibration algorithm that can produce accurate results across a range of calibration situations is desirable.

1.2.4 Imaging

After calculating the corrected visibilities, the next goal is to recover the true SBD and, hence, the image of the true sky from these visibilities. However, due to the incompleteness of the uv-coverage, the interferometer necessitates several assumptions to achieve this [27]. As discussed at the end of §1.2.2, the now corrected visibility density can only recover the dirty SBD $\mathcal{I}_d$. This represents an approximate $\mathcal{I}$ and can be expressed mathematically as

$$\mathcal{I}_d = \mathcal{I} \star \mathcal{I}_b, \tag{1.11}$$

where $\star$ is the *convolution* operator. The term $\mathcal{I}_b$ is called the *Point Spread Function* (PSF) and represents the interferometer's response to a theoretical point-like source, called a *point source*, i.e. it captures the interferometer's sampling characteristics. Thus, the dirty SBD is deconvolved to remove the contributions of the PSF to recover the true SBD.[9] The term $\mathcal{I}_b$ is determined separately, given the characteristics of the interferometer and the observation itself.

[9] Strictly speaking, the vCZ in eq. (1.2) is only a true two-dimensional Fourier transform if the antennas are restricted to lie in the uv-plane. This means that the dirty image is only approximately equal to the model image convolved by the PSF, but these technicalities are not relevant to this research.

Recovering an image from measured visibilities using eq. (1.2) is known as an *inverse problem*. The inverse problem would be considered *well-posed*[10] only if the interferometer had a perfect uv-coverage, i.e. coverage that spanned the entire uv-plane. Hence, recovering an image is an *ill-posed*[11] inverse problem, and certain constraints or assumptions must be imposed to solve for a unique solution consistent with the true sky [27]. This step would be a form of regularisation and is easiest to describe considering a *measurement* model.[12] Combine the discretised versions of the vCZ in eq. (1.2) and the relationship in eq. (1.11). For this combined representation, let $\mathcal{I}$ be mapped to a real $m \times n$ grid of pixels. The output of this mapping is an image denoted as $\mathbf{Z} \in \mathbb{R}^{m \times n}$ and becomes the image vector $\mathbf{z} \in \mathbb{R}^{nm}$ when flattened. Then, using this mapping, a realistic version of the vCZ in eq. (1.2) that accounts for noise present in the observation can be expressed as

$$\mathbf{v} = \mathbf{D}\mathbf{z} + \boldsymbol{\epsilon}, \tag{1.12}$$

where $\mathbf{v} \in \mathbb{C}^d$ represents the corresponding visibilities as a vector and the measurement operator $\mathbf{D} \in \mathbb{C}^{d \times nm}$ is the approximate Fourier transform represented as a linear operator, sometimes called the *degridding* operator [27], [29].[13] The scalar value d is the product of the visibility dimensions and is calculated as $d = N_{\text{bl}} N_{\text{time}}$. Note this assumes that only the antenna, baseline and time axes of the visibilities are considered.

The term $\boldsymbol{\epsilon}$ encodes the noise present within the visibilities and is assumed to be *normal* or *Gaussian* with zero mean and covariance $\boldsymbol{\Sigma}$. Note that an in-depth discussion of these statistical quantities is deferred until chapter 2. For now, this assumption on the noise specifies the *likelihood* for the problem, i.e. the plausibility of realising the measurement model in eq. (1.12) as the truth given the information provided by the measured visibilities. Hence, recovering $\mathbf{z}$ can be treated as a *statistical estimation problem*, an estimation problem that includes randomness or uncertainty within its terms. Since the noise is Gaussian, the *quadratic loss function* can be applied: the least squares loss function weighted by the measurement noise [24], i.e. the negative logarithm applied to the resulting Gaussian likelihood function. Let the quadratic loss function associated with eq. (1.12) be denoted as $\mathcal{L}(\mathbf{e})$ where $\mathbf{e} = \mathbf{v} - \mathbf{D}\mathbf{z}$ is the *error* between the observed visibilities $\mathbf{v}$ and the visibilities, $\mathbf{D}\mathbf{z}$, degridded from the image vector, i.e. the quadratic loss of the image vector given the observed visibilities. The resulting loss function can be expressed as

$$\mathcal{L}(\mathbf{e}) \propto \mathbf{e}^H \boldsymbol{\Sigma}^{-1} \mathbf{e} = (\mathbf{v} - \mathbf{D}\mathbf{z})^H \boldsymbol{\Sigma}^{-1} (\mathbf{v} - \mathbf{D}\mathbf{z}), \tag{1.13}$$

where $(\,\cdot\,)^H$ is the *Hermitian transpose* or *adjoint* operator. Therefore, solving for $\mathbf{z}$ requires minimising $\mathcal{L}$ with respect to $\mathbf{z}$. This is known as a *quadratic problem*, and the usual way to solve it requires differentiating with respect to the quantity of interest and setting the result to zero [33]. Applying these steps to eq. (1.13) yields

[10] *Well-Posed*: The problem has a unique solution [21].
[11] Opposite of *well-posed*.
[12] *Measurement Model*: The mathematical model that accounts for noise in observations [32].
[13] *Degridding*: Derived from the adjoint operation called *gridding* that maps visibilities to a discretised grid, i.e. the image of the sky, through the approximated Fourier transform [27].

$$\mathbf{D}^H\boldsymbol{\Sigma}^{-1}\mathbf{D}\mathbf{z} = \mathbf{D}^H\boldsymbol{\Sigma}^{-1}\mathbf{v}. \tag{1.14}$$

Some key insights related to traditional imaging can be obtained from eq. (1.14). The expression's right-hand side is usually called the (naturally weighted) *dirty image*, $\mathbf{Z}_d$, derived from $\mathcal{I}_d$. The change in notation indicates the discretisation of the SBDs, e.g. the two-dimensional matrix $\mathbf{Z}_d$ is the dirty SBD $\mathcal{I}_d$ after gridding. It is the measured visibilities weighted by the inverse covariance of the measurement noise and projected back into image space through the adjoint of $\mathbf{D}$, also known as the *gridding* operator [27], [29]. The operator to the left of the equality in eq. (1.14), $\mathbf{D}^H\boldsymbol{\Sigma}^{-1}\mathbf{D}$, can be interpreted using the approximate Fourier nature of $\mathbf{D}$. If $\mathbf{D}$ is a Fourier transform, this operator represents a convolution with the PSF where the PSF is computed as $\mathbf{Z}_b = \mathbf{D}^H\text{diag}(\boldsymbol{\Sigma}^{-1})$ where diag($\cdot$) is the *diagonal* operator (see definition A.1.13). In this context, the diagonal operator applied to the inverse covariance matrix returns the diagonal of the resulting matrix expression as a vector. Thus, the classical notion of imaging emerges that the dirty image is the true image convolved by the image of the PSF [29]. However, since the problem is ill-posed, no unique optimal estimate could be derived from the information described by $\mathcal{L}(\mathbf{z} \mid \mathbf{v})$ and additional regularisation is needed to recover the image [27]. Conventionally, the *CLEAN* algorithm [34] is used to deconvolve the effects of the PSF and obtain an image estimate (see Offringa *et al.* [35] as an example). For this research, only partial imaging elements are required in chapter 4; however, discussion of the deconvolution steps is beyond the scope of the current research.

1.3 Research Problem

This section builds on the context of the beginning of this chapter and elaborates on the specific problems this research aims to address.

1.3.1 Considerations for Solution Intervals

Once a calibration problem is defined using the RIME, the next task is to model and estimate the underlying gain terms. This is traditionally framed as a statistical estimation problem, which involves finding values that minimise a predefined loss function, often in the presence of random variables like measurement noise (see §1.2.4 for an example). Assuming that eq. (1.8) represents a measurement model with Gaussian noise, the quadratic loss function is typically used to derive an appropriate *solver*. A solver refers to the algorithm used to perform the estimation. The form and complexity of solvers can be adjusted to suit both the problem's requirements and the computational limitations. They aim to accurately calculate the solution using the loss function while keeping computational costs reasonable [36]. However, calibration problems are often non-linear and, with the emergence of Big Data within radio interferometry, require solving for large numbers of parameters. Therefore, for a solver to be successful within calibration, it should yield accurate results in the presence of noise without excessive computational overhead [11], [13], [37].

A conventional method in calibration employs *Maximum Likelihood* (ML) estimates derived from a least squares model, often grounded in the assumption of Gaussian noise [37].[14] The ML estimate represents the most probable realisation of the desired values based on the measurements and the assumed level of noise [23]. To navigate the non-linearity inherent in the RIME, popular non-linear solvers like the "Levenberg-Marquardt" and the "Gauss-Newton" methods are commonly employed [37]. Other example techniques that have been used within calibration are the "Expectation-Maximisation" or "Alternating-Direction Implicit" methods [4], [5]. These solvers typically approximate non-linear relationships through statistical or numerical methods, like the Taylor series expansion or statistical moment-matching [24], [38]. There are several known implementations of the above, such as SAGECal [4], StEFCal [12], CubiCal [1], and QuartiCal [3]. It is well known that these solvers carry high computational complexity, especially within calibration. Various mathematical, statistical, and computational optimisations are necessary to make these methods viable (see [1], [4], [11], [12], [38]). In addition, ML estimates are suboptimal in the low SNR regime, leading to *overfitting*[15] and suboptimal solutions. Therefore, some form of regularisation is typically required to constrain solutions and improve SNR [20], [37].

Solution intervals are a popular technique to reduce computational complexity and to regularise solutions from overfitting noise. This technique partitions the gain signal over time and frequency into discrete $\Delta_t \times \Delta_\nu$ batches, where $\Delta_t \in \mathbb{N}$ and $\Delta_\nu \in \mathbb{N}$ are the widths along the time and frequency axes, respectively. Within each batch, a solver is tasked with finding a constant ML estimate, reducing the problem's dimensionality based on the intervals' size. Larger interval sizes serve to average out outliers in the data, thereby reducing the risk of overfitting noise and improving the quality of the estimated gains [1], [37]. The noise is often assumed to follow a probability model where the mean is zero. Therefore, by averaging the noise, it will tend to zero, and SNR inherently improves. In terms of computations, several assumptions are applied to improve computational efficiency. Treating each solution interval batch as an independent estimation problem allows for a highly parallelisable environment. This allows an "embarrassingly parallel" calibration algorithm to be distributed across parallel computing frameworks that can scale with the Big Data volumes of data with enough computing resources [1], [3]. This makes solution intervals an attractive regularisation technique for modern data-intensive calibration algorithms, as exemplified by the QuartiCal software. An example visualisation of the solution intervals technique is provided in fig. 1.3.

Although solution intervals offer a robust approach to overcoming specific challenges within calibration, they are not without drawbacks. Notable issues are the difficulty in determining optimal interval sizes for different calibration scenarios, the challenges in calibrating for *complex gain* terms and limits on the computational efficiency of the method [20]. In this context, complex gains refer to complex-valued gain terms that encode both the real amplitude and phase. Calibration algorithms can either estimate the complex-valued gain exactly, i.e. "complex gains", or parameterise the problem to solve for comprising component values,

[14] The noise in calibration is frequently modelled as Gaussian, justifying the use of a quadratic loss function.

[15] *Overfitting*: When an estimate fits the data too much, resulting in poor generalisation and possibly inaccurate values in the presence of noise [39].

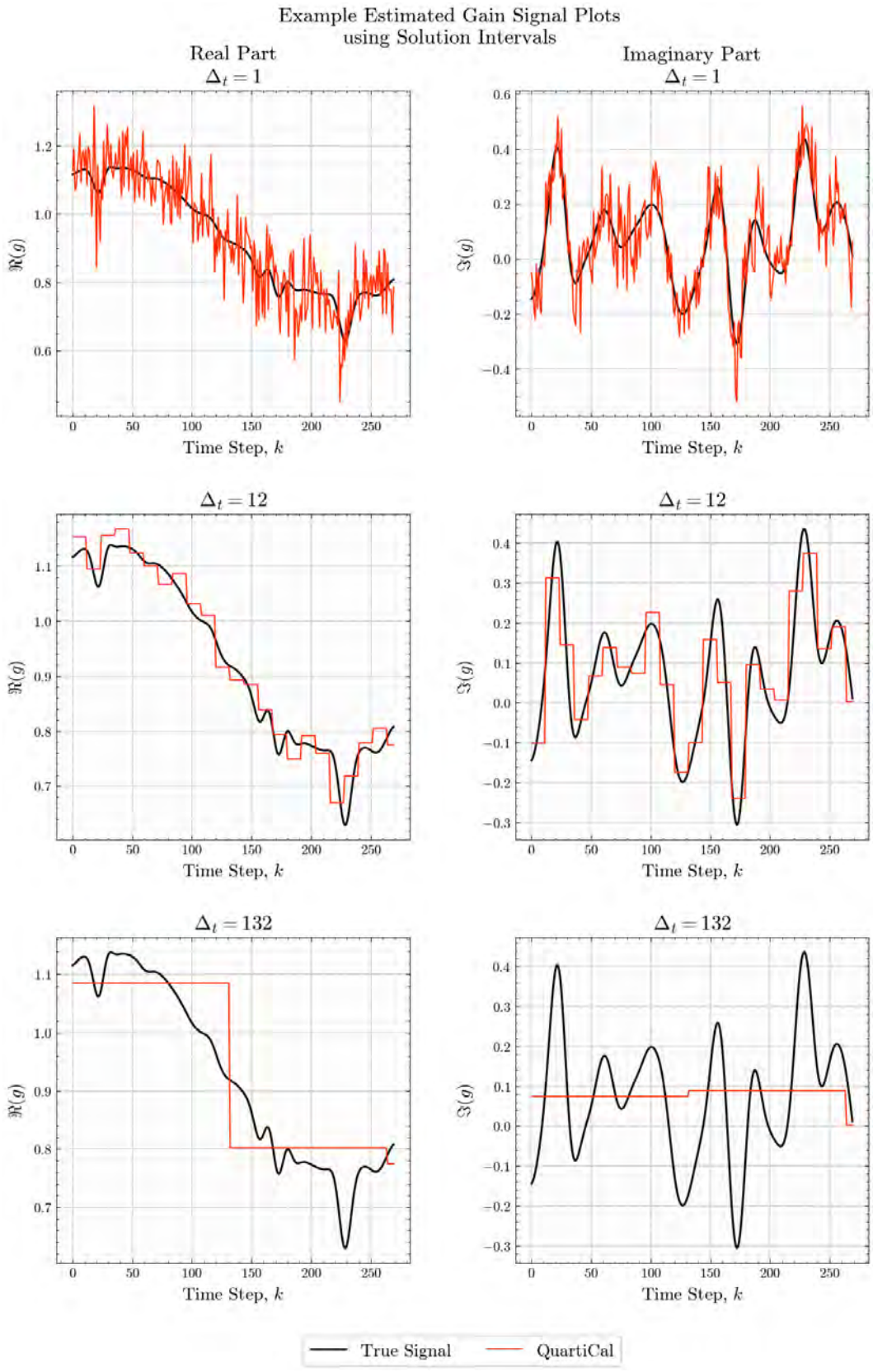

FIGURE 1.3: The estimated solutions (red) of the real (*left*) and imaginary part (*right*) of a random complex signal (black) using the solution intervals technique. Each subplot is titled with the solution interval size used. (*Top row*) A smaller Δ_t that results in fitting more noise, and therefore, the solution is erratic. (*Bottom row*) Larger Δ_t that averages out the signal, leading to flatter solutions that underfit. (*Middle row*) A balance between these extremes at $\Delta_t = 12$ mitigates the overfitting of noise and avoids underfitting the true signal.

like amplitude and phase, separately [16]. Sob *et al.* [20] provides an extensive overview of the associated risks and the appropriate use of solution intervals within calibration. One well-documented issue is the phenomenon of "flux suppression", where sources within the reconstructed image appear fainter than the actual source brightness due to calibration solutions diminishing the genuine signal. In fact, it is even possible to create "ghost sources" if calibration is not performed carefully [22], [40], [41]. Although this effect is more prominent with incomplete sky models, flux suppression can occur throughout the image if the SNR is sufficiently low. This degrades the calibration solutions and the scientific results derived from them.

Even though adaptations have allowed for parallelising across solution intervals to enhance computational efficiency and solution quality, several obstacles remain. Solution intervals are often employed alongside ML solvers [1], [37]. These ML-based approaches require holding all data for a given solution interval in memory simultaneously, increasing the strain on data transfer, storage and computing services. This becomes problematic to impossible when particularly large intervals are required. Coupled with the iterative nature of these algorithms, specific estimation problems may become intractable. Some technologies have been applied successfully to handle these requirements, such as Dask with dynamic task scheduling and lazy evaluation within QuartiCal [1], [3]; however, this research investigates other solvers to address these concerns.

Another obstacle with solution intervals is determining optimal sizes. The nature of the associated parameter spaces is erratic and non-smooth, which complicates the development of guided strategies for selecting optimal interval sizes and makes such endeavours computationally expensive [20]. See fig. 1.4 as an example of the solution interval's parameter space. The risks of solution intervals are exacerbated when gains change rapidly over time and frequency. Smaller intervals are necessary to capture this behaviour accurately but risk overfitting noise, particularly in low SNR regimes. In contrast, larger intervals can average out noise and improve SNR, but they may underfit the signal, leading to inaccurate estimates. Therefore, striking a balance on interval size is essential to achieve optimal calibration results. However, this balance changes for different calibration scenarios because the characteristics of the desired signal change. Hence, without an automated process to calculate these optimal sizes, finding an optimal calibration run with solution intervals will always be difficult, expensive, and cumbersome [20]. No practical method has been found or developed at the time of writing to find optimal solution intervals automatically. Sob *et al.* has explored a brute-force algorithm to address this problem. Unfortunately, the authors point out that it is computationally inefficient and relies on a heuristic, i.e. an optimal solution interval is not guaranteed. Additionally, using solution intervals becomes problematic when calibrating for complex-valued gains because of the differing characteristics of the amplitudes and phases. Therefore, parameterisation of these components is required to effectively model and estimate the gain signal, further adding to the computational complexity of using solution intervals [20]. Despite these challenges, as Sob *et al.* notes, there is currently no alternative that matches solution intervals' success in reducing overfitting, regularising solutions, improving SNR and, with certain assumptions, reducing computational complexity. This research aims

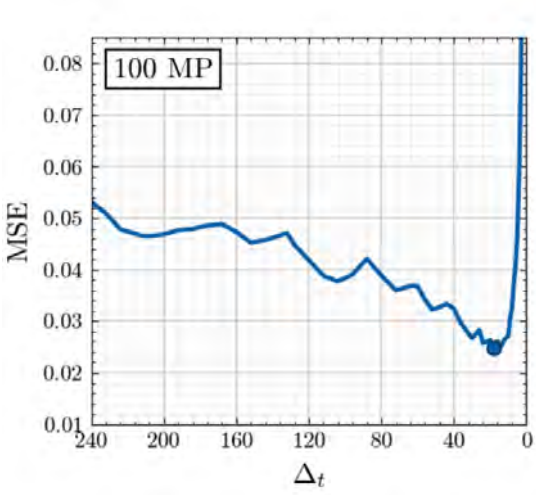

FIGURE 1.4: The above is a plot taken from the simulated experiment in chapter 4. It is the *Mean Square Error* (MSE) (*blue*), a measure of solution accuracy against the true gains versus different choices of time interval size Δ_t for the solution interval solver. This example plot demonstrates the non-smooth nature of solution interval solutions and why they are not ideal for optimisation methods. Note, this result assumes all sky brightness is modelled perfectly, and the x-axis is reversed for reasons that will be detailed in chapter 4.

to address these gaps by introducing a new calibration framework as a potential alternative to solution intervals.

1.3.2 Non-linear Kalman Filters for Calibration

Sob *et al.* [20] outlines several possible alternative approaches that could potentially address the challenges associated with solution intervals. Among these, the KF utilised by Tasse [11] stands out. The KF is a recursive estimation algorithm grounded in Bayesian inference techniques. Specifically, it includes an additional built-in prior signal model to regularise its estimates. This prior model, often called the *dynamic* or *evolution* model, describes the known dynamics of this signal outside the context of the measurement model. By combining the information between the evolution and measurement model, an improved and better-informed model called the *posterior* model can be derived. This enables the KF to regularise its solutions against overfitting noise and generalise its ability to handle various estimation conditions. Moreover, the KF's recursive nature allows it to generate its solutions in single-step increments, typically along the time axis. Unlike the ML approach, the KF's solutions are Gaussian posterior models that describe the statistical behaviour of the underlying signals given the data observed. The ML approach only describes point estimates, whereas the KF provides the models to infer what the estimate should be and offers a probabilistic framework for understanding the underlying signal model. This is achieved by matching and calculating mean and covariance parameters of desired Gaussian posterior models. These parameters are obtained by combining those from the preceding recursive step and the measurement from the current recursive step. Overall, the KF offers a potentially effective and robust alternative, depending on the selection of signal models and prior information [20], [23], [24].

Tasse [11] uses the *Unscented Kalman Filter* (UKF), a variant of the KF designed to handle non-linearities through recursive moment-matching approximations. Formulating the algorithm around the optimisation techniques in [38], the algorithms presented in Tasse [11] utilise physical Jones terms to perform calibration on DDEs related to LOFAR such as ionospheric phase interactions, and pointing errors. Furthermore, the signal models were extended to utilise "pseudo-images" instead of visibilities to reduce the measurement models' non-linearity. These *pseudo-images* are one-dimensional images created by applying the Fourier transform on visibilities along the frequency axis. To further enhance its robustness and efficiency, the algorithm includes several other optimisations:

- *Sherman-Morrison-Woodbury Identity* (SMW) - Also called the *matrix inversion lemma*, this lemma reduces the computational cost of the large matrix inversion step contained in the UKF by switching the subject of the inversion to one with smaller dimension sizes. This initial step is required for the *Kalman-Woodbury Identity* that will be discussed in §2.4.4.

- *Dual Evolution function* - A two-step evolution function to assist in finding and tracking the signal early on. The evolution function describes how the desired gains would change or "evolve" over time.

- *Adaptive Process Noise Step* - A per-step scaling of the constant process noise term based on residual data from the past. Given the previous evolution function, the process noise determines the stochastic force that continuously perturbs the state, known as the driving noise of the gain signal.

- *Tikhonov Regularisation* - Improves the algorithm's robustness such that the accuracy of solutions is also improved over a wider variety of calibration scenarios.

Together, these optimisations and the UKF address challenges such as *ill-conditioning*,[16] solution accuracy and computational costs. After testing on simulated data, Tasse concluded that there is a potential to use UKF and similar filtering algorithms to improve radio interferometric calibration, with particular emphasis on addressing the challenges mentioned above. However, Tasse concluded with the note that this approach had not yet been tested with real data, and there were plans to publish a future paper that would cover this. At the time of this study, the only follow-up work found was [2]. This paper discusses the software killMS, which implemented the resulting algorithm in [11]. Furthermore, killMS now also contains a phenomenological calibration algorithm to expand its capability as a calibration suite. The software has been successful in calibrating for DDEs. Its use has been cited in various astronomical and scientific articles associated with developments and results from LOFAR, *Australian Square Kilometer Array Pathfinder* (ASKAP) and MeerKAT (see [43]–[46]). Needless to say, the work of Tasse can be considered significant and a promising start for this research. This approach is further supported by the research of Herring, Davis, and Shapiro

[16] *Ill-conditioned*: A problem or computation that can be modelled as a system of linear equations where a slight change in inputs results in large changes in outputs. This often indicates that solutions are hard to find [42].

[47], Soja *et al.* [48], and Nilsson *et al.* [49] which developed *Very Long Baseline Interferometer* (VLBI) calibration techniques using the KF. These studies show that the KF can be just as effective, if not more so, than the conventional non-linear least squares techniques.

Referencing the previous discussion of solution intervals, implementing the KF could provide potential solutions to the previously mentioned issues. Integrating a prior model during estimations could enhance calibration solutions' accuracy and reduce the effect of noise. Additionally, the recursive design of the KF offers computational advantages over solution intervals and its batch ML solvers, as it does not require all data to be loaded into memory and only performs one pass over the measurement data. This reduces memory requirements and dimensionality, resulting in improved computational efficiency. However, despite the potential of the KF approach, it does come with certain limitations, which will be discussed in the following section.

1.3.3 Improving upon the Filtering Approach

Despite its many advantages, the KF has inherent limitations due to its recursive design. The algorithm only considers past and current measurements at each time step, leaving out potentially vital future information. This incompleteness leads to less informed and, subsequently, less accurate estimates. In contrast, traditional iterative estimation techniques do not suffer from this problem since all observations are considered when calculating each ML estimate. Furthermore, the full regularisation effect of the KF's prior model is not realised since its estimates are suboptimal from a Bayesian perspective. This may make the KF ideal for on-line methods where future measurements are unavailable; however, this approach is not ideal for off-line methods. Although the optimality of solutions improves as the KF progresses over time, earlier solutions in time for the KF are less informed than those later, where the last solution is the only output that captures all available measurement information. As a result, calibration solutions derived from the KF can be potentially suboptimal, especially in lower SNR conditions [24], [50].

Various adaptations have been explored to address the inherent limitations of the KF. Tasse [11] incorporated techniques such as the dual evolution function and Tikhonov regularisation that proved effective. In addition, Nilsson *et al.* [49] approaches this issue by adapting the "two filter" method. In this setup, the KF is run twice: the first forward in time and the second backwards in time, but starting with the last estimate of the previous KF run as its initial prior information [24]. Nilsson *et al.* notes the suboptimality of their solutions due to the dependence on the first estimates of the KF. Furthermore, Särkkä [24] mention how, although viable, this two-filter technique can often be a non-trivial and computationally expensive task to accomplish, especially when the underlying model relationships are non-linear. For this research, a different approach will be investigated to improve the limitations of the KF.

If the recursion property of the KF is desired while enhancing its capabilities, the RTS Smoother, also known as the KS, is a suitable extension [24], [25]. Developed by Rauch, Striebel, and Tung, the KS is designed to work in conjunction with the KF using the same

probabilistic models. After the KF produces its estimates, the KS begins its backward recursion to complete the solutions, initialised with the final estimate of the KF. Then, at each step, the filter estimates are corrected or *smoothed*, using the smoothed estimate calculated from the step ahead in time. The outcome of this KS run is estimates that include all the measurement information across the entire time frame. The advantage of this process is that there is no need to repeat measurement-model-related calculations. Instead, it uses information from later estimates of the KF to fill in the gaps of more recent estimates. It is called a *smoother* because it is considered a Bayesian smoothing solution, i.e. it is derived from a Bayesian model that has been regularised through a prior model. Using a smoother, the solutions are less likely to fit noise, resulting in "smoother" estimated signals [24].

In the off-line calibration context, the KS would be an ideal addition alongside the KF. It improves upon the limitations of the KF and provides the complete regularisation effect of the prior model upon the solutions. Hence, estimates will be more accurate and robust. Furthermore, the computational complexity of the KS is lower than the KF since it exploits the measurement information discerned by the KF. This combination of regularising the solutions with minimal increase in complexity makes it a strong candidate to solve the previously discussed calibration issues. This is because the same process of including priors into iterative estimation algorithms or techniques is often non-trivial to design and implement. For example, an estimate for a given solution interval in the time-frequency plane is informed only from the data within that batch. Therefore, smoothing between solution intervals would require each interval to communicate statistical information between independent batches. Although beneficial and possible, it is a difficult task to complete and introduces further challenges to overcome from a software implementation perspective. In comparison, achieving such levels of regularisation with the KFS equates to ensuring that the KS is run over the KF results and an appropriate and effective prior model of the gains signal is used.

To the writer's knowledge, the KS has not been used in the context of radio interferometric calibration. However, there is evidence that demonstrates improvement on the results of the KF for other, similar problems. A sample of such papers is presented below:

- "Position estimation using extended Kalman Filter and RTS-smoother in a GPS receiver" by Wang [51] - The author presents an alternative approach to estimating position in GPS receiver software that compares the non-linear form of the KF called the *Extended Kalman Filter* (EKF). This algorithm is similar to the KF, but linearises non-linear functions through statistical or Taylor series linearisation [24]. The results of these inclusions showed how the EKF outperformed the traditional least squares algorithm that was used in terms of accuracy and estimation stability. Furthermore, by including RTS Smoother, i.e. KS, with the EKF, the author notes the significant improvement in accuracy and precision of position estimation.

- "Resolving the data asynchronicity in high-speed atomic force microscopy measurement via the Kalman Smoother" by Kubo *et al.* [52] - Researchers developed an algorithm around KFS to correct for issues related to time differences, asynchronicity, and data acquisition steps when creating "movies" derived from frames created using *high-speed*

atomic force microscopy (HS-AFM). This type of microscopy aims to observe the mechanics of biomolecules at atomic resolutions by creating these videos. A central point discussed was how using the KS for these movies drastically improved the quality by reducing the distortion and noise within each frame.

- "JTRF2014, the JPL Kalman filter and smoother realisation of the International Terrestrial Reference System" by Abbondanza *et al.* [53], and "KALREF - A Kalman filter and time series approach to the International Terrestrial Reference Frame realisation" by Wu *et al.* [54] - *Jet Propulsion Laboratory Terrestrial Reference Frame 2014* (JTRF2014) algorithm implemented in KALREF to calculate station coordinates by incorporating several avenues of information into a KFS algorithm. It is the first case in this list where a KS was used to smooth the output of the KF. The results showed significantly improved results and the possibility of reducing the computation time from weeks to days.

Given the above evidence, investigating the KFS as a phenomenological RIME calibration algorithm would be worthwhile to address the concerns when using the solution interval technique. At the same time, this investigation will use the *Bayesian evidence*, whose unnormalised negative log-likelihood is referred to as the *energy function*, to determine whether it is possible to determine optimal calibration parameters automatically. Bayesian evidence is the marginal likelihood function within the Bayesian estimation scheme and describes how well a statistical model describes the observed data within the estimation problem [23], [24]. While not essential for deriving a Bayesian estimator, it is valuable for constructing parameter models that facilitate optimal hyperparameter selection. In the case of calibration, there are several benefits to discerning this information. This is particularly beneficial in calibration scenarios where useful accuracy metrics, like the *Mean Square Error* (MSE) between the actual gain signal and the estimated gain signal, may not be readily available [20]. The Bayesian evidence can potentially enable calibration algorithms to automate optimal parameter selection strategies by representing information not readily available to the solution intervals approach. The only disadvantage is that it is computationally expensive to calculate and, for iterative estimation algorithms, can only be calculated once the algorithm is complete. Therefore, its inclusion is also considered when considering such approaches [24].

Särkkä [24] highlights a unique feature the KFS has access to the energy function. Specifically, Särkkä demonstrates that the Bayesian-formulated KFS already computes the required terms to construct the energy function. This can be done recursively during the algorithm's run-time, making the calculation more efficient. The implication is that hyperparameters for the KFS could be adaptively tuned during the algorithm's run-time, improving robustness and accuracy. Additionally, it can be used between estimation runs to improve the quality of solutions with each iteration. Several existing methods exploit this energy function in various applications [24]. This research aims to investigate the utility and implementation of the energy function in the context of the KFS calibration algorithm. Without a mechanism for selecting appropriate priors, the KFS would face challenges similar to those seen with solution intervals. If implemented successfully, the energy function could provide the KFS with

self-tuning capabilities that automatically derive optimal calibration strategies. This could make the KFS a strong competitor to solution intervals techniques. Therefore, this research will develop the KalCal framework to investigate whether these statements are valid. This marks the end of the research introduction. The following chapter will explore the theory of the KFS and how they work.

Chapter 2

Kalman Filters and Smoothers

Estimation is central in statistics and is crucial in various fields, such as signal processing, stochastic optimal control systems, telecommunications, tracking, and biological modelling [55]. This chapter aims to introduce the mathematical foundations of statistical estimation, focusing on *Bayesian Filtering and Smoothing* (BFS) and the *Kalman Filter and Smoother* (KFS) within the context of this research. Specifically, provide a strong understanding of the KFS from a Bayesian perspective, discuss how the *Kalman Smoother* (KS) improves upon the *Kalman Filter* (KF), and explore the integration of the energy function within the KFS. Note the context and content within this chapter primarily focus on those relevant to calibration and upcoming chapter 3. For an in-depth theoretical analysis and discussion, see Jazwinski [23], Särkkä [24], Dini and Mandic [56], and Dang and Scharf [57]. This chapter begins by exploring the history and foundational context of the KFS, leading to a detailed presentation of its derivation and functionalities. Furthermore, additional extensions and optimisations are made to the KFS necessary for developing a practical calibration algorithm framework.

> **Note:** Readers familiar with *Bayesian Filtering and Smoothing* (BFS) and KFS may proceed directly to §2.4, "Extensions to KFS". For those new to these topics, this chapter begins with a historical context, moving through the essential theories and culminating in the presentation of the KFS.[1]

2.1 Introduction

The KFS and its place in modern society is a fascinating story. It has seen significant successes in technology, telecommunications, aeronautics, and engineering for over 60 years in practical and scientific endeavours. Present-day examples of its implementation can be found in self-driving cars, speech recognition, financial modelling, computer vision systems, and more [24]. Even though today it is most commonly associated with BFS theory, the original derivation of the KFS does not utilise Bayesian techniques despite the logical equivalences. Instead, it was created using a combination of mathematical topics used in control systems at the time. This section will briefly discuss this to provide the core perspective and aim behind the concept of KFSs and why they are helpful estimation algorithms.

[1] The content in this chapter reflects the author's learning journey through the complexities of BFS and KFS, aiming to provide a comprehensive and accessible knowledge base for researchers who wish to understand complex signal estimation using Bayesian statistics and KFS theory.

2.1.1 History

The journey of estimation has its provenance linked to the renowned mathematician and physicist *Karl Friedrich Gauss*. He pioneered the popular least squares method in his astronomical studies to predict and describe celestial mechanics of planets and comets via telescopic observations [23], [24], [55]. Following two centuries of additional research and development, *Dr. Rudolf E. Kalman*, a Hungarian control systems researcher and mathematician, published a paper in 1960 titled "A New Approach to Linear Filtering and Prediction Problems" [58]. For its time, Kalman's work revolutionised the standard approach to estimation problems. Built on influential predecessors like Wiener, Kolmogorov, Bode, Shannon, Pugachev, and others, he introduced the *Kalman Filter* (KF), a recursive discrete estimation algorithm that utilised unique concepts and ideas that solved the *filtering* form of the so-called *Wiener Problem*. This is a class of estimation problems presented by Wiener [59] where a hidden signal must be estimated from a related measurable but noisy signal. It has different variants but can be categorised into the following forms: *prediction*, *filtering*, and *smoothing*. This is discussed in detail in §2.1.2. At its conception, it had a tremendous impact on science and technology, especially in aviation and network engineering, cementing its legacy and relevance even today. In fact, for the KF and his influential contributions to optimal control systems and filtering theory, he received the "National Medal of Science", the highest award in the United States of America for scientific achievement [60], [61].

Despite the ingenuity and benefits of the KF, it is limited to a particular class of estimation problems. The KF is ideal for real-time or on-line recursive estimation systems where future observations are restricted to the present, i.e. the *filtering problem*.[2] That means the older the estimate, the less informed the calculated estimate is from newer measurements. For off-line estimation problems, where all measurements are available, like radio interferometric calibration, the KF would most likely produce less accurate solutions because not all estimates are equally informed. This is known as the *smoothing problem*[3] and it sparked the development of algorithms called *smoothers*.[4] Shortly after Kalman published his work in 1965, the *Rauch-Tung-Striebel Smoother* (RTS Smoother), also known as the *Kalman Smoother* (KS),[5] was created and published in the paper "Maximum likelihood estimates of linear dynamic systems" [25] by *Dr. Harry E. Rauch, Dr. F. Tung, and Dr. C. T. Striebel*. Rauch, Striebel, and Tung also lay out a recursive discrete algorithm, utilising similar concepts as in Kalman's paper [58], to solve the smoothing issue of the KF. The most significant difference between the KS and other smoother algorithms is that it is designed to be coupled with and leverage the KF. Specifically, the KS improves older KF estimates by propagating measurement information from newer estimates backwards through time. This allowed the authors to improve the filter's solutions in an off-line context drastically and provide more regularisation in the

[2] The *filtering* form of the Wiener Problem.

[3] The *smoothing* form of the Wiener Problem.

[4] Note, the filtering and smoothing problems are often associated with *Bayesian filtering* and *smoothing*, respectively. However, within this context, they are distinct concepts to avoid confusion, i.e. BFS provides unique solutions to the more generalised filtering and smoothing problems.

[5] Note the independent creators of the aforementioned algorithms are wholly acknowledged. However, from this point onwards, it will be easier to refer to the RTS Smoother as the KS to avoid any confusion.

final estimates without extracting any new information from the measurements or repeating calculations [25].[6] Together, the KFS form a succinct and robust estimation algorithm framework flexible enough to cover a range of essential applications.

The discovery of the KFS was impactful in the rise of the BFS field, which can be described as formulating algorithmic solutions to the various forms of the Wiener problem using the Bayesian approach. In fact, the KFS can viewed as an exact linear Gaussian BFS solution. Therefore, it will be natural to study the KFS within this chapter via this lens. With the development and improvement of BFS and computing abilities, new extensions and scientific use cases have emerged to solve various estimation problems. One of the most important considerations was the incorporation of non-linear problems, which led to many flavours of the KFS to address a variety of scenarios. Some examples of this are the *Extended Kalman Filter and Smoother* (EKFS), *Iterated-Extended KFS*, *Unscented KFS*, and the *Ensemble KFS* to name a few [24], [32]. This research investigates the EKFS due to its similarities with current calibration techniques and optimisations. There is ongoing research and development with KFS and BFS in various academic and technological fields, like this research that aims to discern its potential role and influence within radio interferometry.

2.1.2 What is filtering and smoothing?

Until now, the terms "filtering" and "smoothing" have appeared within a few different contexts with minimal description. Its meaning is often conflated with descriptions in many technological, scientific, and practical discussions. This chapter will describe it via the perspective of BFS, for which Saberi, Stoorvogel, and Sannuti [55] and Grewal and Andrews [61] provide a suitable differentiation. Simply, filtering and smoothing can be viewed as a collection of tools and techniques that cover the following theoretical and practical functions:

- Perform instantaneous or delayed estimation of states via measurements.
- Separate the signals into desired signals and noise.
- Solve the inversion problem, i.e. solve for system inputs from measured outputs.
- Learn how to track, estimate and predict a given signal.

This covers the practical side of this theory; however, a formal definition is provided. Wiener [59] and Kalman [58] define the *Wiener problem*, also referred to as the *filtering problem*, as follows:

> *Let $f(t)$ and $g(t)$ be two complex time series, where $f(t)$ represents the "message", and $g(t)$ represents the "disturbance" accompanying the message. The [Wiener] problem is then to find a linear operator that can be applied to $f(t) + g(t)$ such that we can make the best approximation for the term $f(t + \alpha)$.*

[6] Although the exact motivations for the development of the KS by Rauch, Striebel, and Tung is unknown, the research was published under *Lockheed Missiles and Space Company* (see [25]). This was a unit of the infamous *Lockheed Martin*, an American defence and arms technology company [62]. Therefore, since the work was published during the 1960s, one can speculate that motives were linked to developing projectile research and technologies, e.g. missiles and rockets, during the Cold War.

Wiener's problem takes multiple forms that are determined by the value of α and can be categorised by completing the question, "Estimate the value of $f(t+\alpha)$ if ... " [23]–[25], [58], [59]:

- *"... only past information is known about $f(t)$, i.e. $\alpha > 0$?"*: This is called the *prediction form* of the Wiener problem, which is also known as the *prediction problem*.
- *"... we know past and current information about $f(t)$, i.e. $\alpha = 0$?"*: This is called the *filtering form* of the Wiener problem, which is also known as the *filtering problem*.
- *"... we know past, current and future information about $f(t)$, i.e. $\alpha < 0$?"*: This is called the *smoothing form* of the Wiener problem, which is also known as the *smoothing problem*.

It must be noted that the original definitions harbour ambiguity that is not Wiener's doing but is likely the result of decades of differing terminologies across many professions and academic subjects. The above labels will be used for consistency in the rest of this dissertation. Additionally, any algorithms that are solutions to the prediction, filtering, or smoothing problems are thus called *predictors*, *filters* and *smoothers*, respectively. This is directly related to the nature of the previously mentioned algorithms in BFS theory.

2.1.3 Uniqueness of the KFS

As introduced in §2.1.2, the Wiener problem forms the root that branches out into the problems of predicting, filtering, and smoothing. Stepping within these subproblems, one could further branch into whether it is continuous, discrete, or both. To clarify, the desired and measured signals are not restricted to being both continuous or discrete and can exist as a combination of either [23]. Furthermore, the description and severity of the perturbations incorporated into the measurements also affect the type of solution derived. This is one reason why Wiener's definition of the problem is widespread: it is succinct and is excellent for tailoring theory to the specificity of estimation requirements. However, Wiener's solution to the problem called the *Wiener Filter*, was limited as he only solved the linear, *stationary*[7] version of the problem. Thus, in the years following his publication, most published solutions aimed to build upon the Wiener filter, often reaching out to other mathematical sciences to fill the regions not covered by Wiener's approach. However, credit for finding the solution to the discrete, linear, stationary or *non-stationary*[8] filtering problem is given to Kalman with his KF. As Kalman [58] writes, the KF is designed around the concept of state spaces, difference equations, and projections, which during Kalman's time as a researcher, was still considered a new and modern approach to describing the Wiener problem. Still, it formed the perfect piece to fit into Kalman's puzzle [60].

State spaces can be viewed as parameterised functions over time that represent the set of all states the signal can be in [23]. The state space can be defined over a domain of

[7] *Stationary*: A random signal where the statistics are the same, and the signal function is linear, regardless of your frame of reference along an axis, e.g. time [23].

[8] Opposite of a *stationary* signal.

random variables such that the system state has a degree of randomness at a given time. Within the same state space and between state spaces, unique functional relationships can be represented as recursive formulae, called *stochastic difference equations*. They can be viewed as modelling the slight changes between states while moving along the time indices [23], [58]. Compounding state spaces with inference techniques that are now understood to be equivalent to Bayesian inference, Kalman created the highly robust, well-defined, and powerful KF algorithm. Kalman's paper did not directly use Bayesian techniques as described before, but the solutions are logically equivalent. Rauch, Tung and Striebel manifested their smoother with the same mathematical concepts and frameworks as established in Kalman's paper but formulated both the filtering and smoothing solutions with the more well-known ML estimate method [25]. This allowed the formation of an optional but neat attachment to the already beneficial KF to allow it to smooth less informed estimates. These works formed the foundational layer for other statisticians, engineers, control system experts, and scientists to develop similar algorithms with even more advanced techniques to solve other complex issues, such as non-linear signals, larger dimension sizes, adaptive algorithms, or even creating fast real-time estimation algorithms.

2.2 Probability and Statistics Fundamentals

The KFS is popular for signal estimation algorithms because it has a strong, well-developed probabilistic and mathematical foundation. This is achieved using Bayes' revolutionary work in probabilistic modelling and estimation. To describe why that is, this section will present the fundamental and elementary theory of Bayesian statistics and the relevant estimation techniques needed to derive the KFS in §2.3. Furthermore, it is given in the context of signal processing in the complex plane, i.e. complex probability theory and statistics.[9] This section is based on the comprehensive works of Jazwinski [23], Särkkä [24], Schreier and Scharf [33], and Bishop [39], and so only the most important points will be referenced. Furthermore, to handle the multitude of notation standards throughout this research, the notation style used in this work is defined in definition A.1.1.

2.2.1 Probability Theory

Uncovering mathematical patterns within a signal often initially leads to analytic and deterministic strategies. However, in situations where the signal's underlying dynamics are elusive or overly complex, these methods might fall short. Instead, embracing a notion of "randomness" can often allow for more practical and effective modelling and estimation in scenarios where these aforementioned strategies struggle to capture the signal's intricacies. The theory of probability aims to provide the mathematical toolkit to describe this randomness and allows the extraction of vital information despite the nature of the signal itself. A *random variable*, denoted x, represents a value that can vary due to chance or randomness. It is associated with a *probability density* or *density function*, represented as $p(x)$, which gives

[9] If applicable, the real-plane counterparts are discussed by [23], [24].

the likelihood of each possible value it could be in the *sample space*.[10] This relationship is often denoted by the notation $x \sim p(x)$, which means that x is a random variable of the density $p(x)$. This means x has a likelihood described by $p(x)$ of taking a value in its sample space. These random variables are considered complex values in signal processing, and the sample space is defined over the complex plane. Complex numbers are used in signal processing because they allow for a compact representation of signals that encapsulate amplitude and phase information. Additionally, it enhances the intuitive understanding of these signals and the mathematical manipulations applied to them [32], [33]. Moreover, these concepts are extended to a system of multiple variables that is more useful for this research, i.e. the probability of vectors. In that case, these random variables are replaced with *random* vectors denoted as

$$\mathbf{x} = [\ x_i \mid i \in (1, ..., n)\ ]^T = [x_1, ..., x_n]^T, \tag{2.1}$$

where x_i is a random variable and the i^{th} element of $\mathbf{x}$.[11] Additionally, $(\ \cdot\ )^T$ denotes the *transpose* operator. These vectors are drawn from multi-dimensional sample spaces governed by $p_n(\mathbf{x})$, where $n \in \mathbb{N}$ denotes the *cardinality*[12] of the complex vector space. The scalar index n may be omitted if the dimensionality of the modelled random vector is already known, that is, if it is stated that $\mathbf{x} \in \mathbb{C}^n$. The convention in the future will be to list it for the first appearance of a probability density and omit it from that point onwards.

Let $\mathbf{x} \in \mathbb{C}^n$ and $\mathbf{y} \in \mathbb{C}^m$ be random vectors for some $n, m \in \mathbb{N}$. The probability of observing a specific outcome of a random variable or vector, known as a *realisation*, for both $\mathbf{x}$ and $\mathbf{y}$ is determined by the density $p_{n+m}(\mathbf{x}, \mathbf{y})$. This is called a *joint* density function of random vectors $\mathbf{x}$ and $\mathbf{y}$ and describes the probability of observing both $\mathbf{x}$ *and* $\mathbf{y}$. Here, the positional re-ordering of random vectors in the joint density argument does not affect the probability, i.e. $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{y}, \mathbf{x})$. The densities $p_n(\mathbf{x})$ and $p_m(\mathbf{y})$ in this context are known as *marginal* densities. This is because any given marginal density of one random vector can be retrieved through *marginalisation* of a joint density, which involves integrating the joint density over the other random vector. It is used to calculate the probability densities of a specific random vector without the context of the other. This step can be written as

$$p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{y})\ \mathrm{d}\mathbf{y}. \tag{2.2}$$

If there is no relationship between these random vectors, then they are considered *independent* of each other, and the joint density becomes

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x})p(\mathbf{y}). \tag{2.3}$$

[10] *Sample space*: The collection of all values that random variable could be [23].

[11] The notation used in eq. (2.1) to construct the vector is called *vector-builder notation* designed to demonstrate how vectors and matrices are constructed from sets and indices recipes using the same style as *set builder notation*. It closely matches the steps used to implement these objects in software and assists in the translation from mathematics to programmable code. This notation and reasoning are fully explained in definition A.1.2 and remarks A.1.3 and A.1.4.

[12] *Cardinality*: In this context, it is the number of dimensions formed within the vector space [17].

If this condition does not otherwise hold, they are considered *dependent*, i.e. some underlying relationship exists between the random vectors. Signals are based on time and frequency axes, meaning there can be a sequence of observations of random vectors. A different probability density is required to capture this sequential relationship between vectors. If $\mathbf{y}$ is realised before $\mathbf{x}$, then the probability density of $\mathbf{x}$ given that $\mathbf{y}$ has been realised is denoted as $p_n(\mathbf{x} \mid \mathbf{y})$ and called a *conditional* density. This density captures information relating one random vector's realisation to another's probability density. If $\mathbf{x}$ and $\mathbf{y}$ are independent, then

$$p(\mathbf{x} \mid \mathbf{y}) = p(\mathbf{x}). \tag{2.4}$$

The above entails that regardless of what $\mathbf{y}$ is, it does not affect the probability model of $\mathbf{x}$. Continuing, the following relationship exists between the marginal, joint, and conditional densities:

$$p(\mathbf{x} \mid \mathbf{y}) = \frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{y})}. \tag{2.5}$$

When rearranged and extended, it forms the useful *chain rule* of probability densities written as

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x} \mid \mathbf{y})p(\mathbf{y}) = p(\mathbf{y} \mid \mathbf{x})p(\mathbf{x}), \tag{2.6}$$

which will be used throughout the rest of this chapter. By rearranging the terms in the latter two expressions in eq. (2.6), the result is *Bayes' Theorem* defined as

$$p(\mathbf{x} \mid \mathbf{y}) = \frac{p(\mathbf{y} \mid \mathbf{x})p(\mathbf{x})}{p(\mathbf{y})}, \tag{2.7}$$

and forms the fundamental foundation of Bayesian statistics and subsequently BFS [23], [24]. It will be discussed in detail in §2.2.7 when all the mathematical theory is provided.

2.2.2 Measurable Statistics

To infer information from probabilistic models, there needs to be an understanding of their dynamics. A set of tools is required to measure particular attributes of these density functions directly related to the signal data. The most common and intuitive measure is the *expectation*[13] of a random vector. If $\mathbf{x} \in \mathbb{C}^n$ is a random vector with density $p(\mathbf{x})$, then the *expectation* vector, *mean* vector or *expectation* of $\mathbf{x}$ is denoted as

$$\mathbf{m}_\mathbf{x} = \mathrm{E}\left[\mathbf{x}\right] = \left[\, \mathrm{E}\left[x_i\right] \mid i \in (1, ..., n) \,\right]^T = \Big[\mathrm{E}\left[x_1\right], ..., \mathrm{E}\left[x_n\right]\Big]^T \tag{2.8}$$

where the *expectation* operator, $\mathrm{E}\left[\,\cdot\,\right]$, is defined as

$$\mathrm{E}\left[x\right] = \int_{-\infty}^{\infty} x p(x)\, \mathrm{d}x. \tag{2.9}$$

[13] Also known as the *mean*, *average*, or *first moment*.

The expectation measures the average realisation for a given random vector in its sample space. If there exists another random vector $\mathbf{y} \in \mathbb{C}^m$ that is related to $\mathbf{x}$ with the linear relation $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b}$, where $\mathbf{A} \in \mathbb{C}^{m\times n}$ and $\mathbf{b} \in \mathbb{C}^m$, then $\mathbf{m}_\mathbf{y} = \mathrm{E}[\mathbf{y}]$ is equivalent to

$$\mathbf{m}_\mathbf{y} = \mathbf{A}\mathbf{m}_\mathbf{x} + \mathbf{b}. \tag{2.10}$$

This statistic only provides partial amounts of information on its own. It cannot discern the data spread around the mean and, therefore, does not know how sparse the different realisations are within the sample space. Furthermore, the expectation does not entirely encapsulate information encoded within relationships between variables. For this, consider the *second moment*, or the *covariance* matrix or *covariance* between random vectors. Using the same vectors as above, the covariance matrix between $\mathbf{x}$ and $\mathbf{y}$ is defined as

$$\mathbf{P}_\mathbf{xy} = \mathrm{Cov}[\mathbf{x},\mathbf{y}] = \left[\left[\mathrm{Cov}\left[x_i, y_j\right]\right]_{ij} \,\middle|\, (i,j) \in (1,...,n)\times(1,...,m) \right], \tag{2.11}$$

where the *covariance* operator, $\mathrm{Cov}[\,\cdot\,,\,\cdot\,]$, is defined as

$$\mathrm{Cov}[x,y] = \mathrm{E}\left[(x - m_x)(y - m_y)^*\right], \tag{2.12}$$

where $m_x = \mathrm{E}[x]$ and $m_y = \mathrm{E}[y]$ are scalar means. The symbol $(\,\cdot\,)^*$ represents the *complex conjugate* operator.[14][15] The covariance measures how changes in one variable affect changes in another and can be used to discern relationships between random variables and vectors. To represent eq. (2.11) only in terms of random vectors, the relation,

$$\mathrm{Cov}[\mathbf{x},\mathbf{y}] = \mathrm{E}\left[(\mathbf{x} - \mathbf{m}_\mathbf{x})(\mathbf{y} - \mathbf{m}_\mathbf{y})^H\right], \tag{2.13}$$

can be used where $(\,\cdot\,)^H$ represents the *Hermitian transpose* or *adjoint* operator.[16] The term $\mathbf{P}_\mathbf{xy}$, called a *cross-covariance* matrix, is a *Hermitian*[17] *positive semidefinite*[18] matrix that describes the joint variability between elements of $\mathbf{x}$ and $\mathbf{y}$. If $\mathbf{x}$ and $\mathbf{y}$ are independent of each other, then intuitively, $\mathbf{P}_\mathbf{xy} = \mathbf{0}$. The cross-covariance matrix does not contain information on the variability between elements within a single vector. The statistic for this is called the *auto-covariance* matrix. Using the predefined operators within the cross-covariance matrix, the auto-covariance matrix is defined as

$$\mathbf{P}_\mathbf{x} = \mathrm{Cov}[\mathbf{x},\mathbf{x}] = \left[\left[\mathrm{Cov}\left[x_i, x_j\right]\right]_{ij} \,\middle|\, (i,j) \in (1,...,n)\times(1,...,n) \right]. \tag{2.14}$$

The elements of $\mathbf{P}_\mathbf{x}$ indicate the variability between the elements within $\mathbf{x}$. It is given a distinct operator, called the *variance* operator, defined as $\mathrm{Var}[\mathbf{x}] = \mathrm{Cov}[\mathbf{x},\mathbf{x}]$. However,

[14] When applied to a vector or matrix, the scalar conjugate operator is applied to each element. With that noted, the notation will be used interchangeably between scalars and multi-variable terms.

[15] This is a unique use of the "vector-builder notation" used to construct matrices dubbed the *matrix-builder notation* and is explained in remark A.1.3.

[16] *Adjoint operator*: The complex-valued counterpart of the transpose operator $(\,\cdot\,)^T$ in the real plane. If $\mathbf{A} \in \mathbb{C}^{n\times n}$, then $\mathbf{A}^H = (\mathbf{A}^*)^T = (\mathbf{A}^T)^*$ [42].

[17] *Hermitian*: The matrix $\mathbf{A} \in \mathbb{C}^{n\times n}$ is *Hermitian* iff $\mathbf{A}^H = \mathbf{A}$ [42].

[18] *Positive Semidefinite*: The matrix $\mathbf{A} \in \mathbb{C}^{n\times n}$ is *positive semidefinite* iff, for all non-zero $\mathbf{x} \in \mathbb{C}^n$, $\mathbf{x}^H\mathbf{A}\mathbf{x} \geq 0$ [42].

throughout this dissertation, the label "covariance matrix" will be associated with both types. Each can be discerned by whether the matrices are square, if they comprise two random vectors or one, or unless explicitly stated. A random vector's variance indicates how the data is spread around the mean. If the variance is small, then the data are clustered around the mean, and thus, the expectation is a good representation of the underlying data. However, if the variance is large, then the data are spread out around the mean, and thus, the mean does not represent the underlying data well. Together, they provide a way to understand the statistical behaviour within a given probability model. If this information is coupled with the covariance of another vector, this knowledge can be extended to other probability models. Thus, signals' intrinsic statistical behaviour and relationships with others can be analysed and determined.

2.2.3 Complex Statistics

Before moving into the probabilistic modelling of signals, a few components of complex statistics must be covered. As mentioned above, much information can be discerned from the covariance matrices and mean vectors from the random signals. However, since it is much more convenient to work with signals as functions of complex values, probability theory and statistics require the same adjustment. Specifically, by including the imaginary axis, random terms now carry interrelated information in their real and imaginary parts. This leads to complex-valued probability densities, functions, means, and covariances for the full complex plane. Furthermore, there now exists the possibility of relationships between conjugate pairs. A new covariance matrix called the *pseudo-covariance* matrix defines the relationship between a complex random vector and its conjugate. If $\mathbf{x} \in \mathbb{C}^n$ for some $n \in \mathbb{N}$, then the pseudo-covariance matrix of $\mathbf{x}$ is

$$\widetilde{\mathbf{P}}_{\mathbf{x}} = \mathrm{Cov}\left[\mathbf{x}, \mathbf{x}^*\right] = \left[\left[\mathrm{Cov}\left[x_i, x_j^*\right]\right]_{ij} \,\middle|\, (i,j) \in (1,...,n) \times (1,...,n) \right], \tag{2.15}$$

where $(\,\cdot\,)^*$ is the conjugate operator. The matrix $\widetilde{\mathbf{P}}_{\mathbf{x}} \in \mathbb{C}^{n \times n}$ is a *symmetric*[19] matrix in which each element describes the relationship between a complex random variable and the conjugate of another complex random variable. If there is no relationship between complex random vectors and their conjugates, i.e. $\widetilde{\mathbf{P}}_{\mathbf{x}} = \mathbf{0}$, then the random vector is called a *proper* complex random vector. Otherwise, the random vector is called an *improper* complex random vector. The degree to how interrelated conjugate pairs are is called *impropriety*. Proper random vectors are often easier to model since the number of relationships encoded within their probability densities is much lower than those of improper random vectors. The pseudo-covariance can be vital for modelling complex-valued systems, as it reveals potentially important hidden details that the standard covariance term would not capture. For example, suppose two antennas within an interferometer are relatively close during observations. In that case, an effect called *cross-coupling* can occur where an unintended interaction between the instruments and their received signals generates correlated noise within the measurements. Usually, instrument noise is assumed to be uncorrelated and proper; however, cross-coupling

[19] *Symmetric*: The matrix $\mathbf{A} \in \mathbb{C}^{n \times n}$ is *symmetric* iff $\mathbf{A}^T = \mathbf{A}$ [42].

can invalidate this assumption by introducing impropriety, leading to biased and inaccurate visibilities if not accounted for during calibration [28].

Their complex statistics can be succinctly represented as a *widely linear* model [33]. A widely-linear model is similar to a linear model in the real plane; however, it contains additional terms to account for any information in the complex plane. Let $\mathbf{y} \in \mathbb{C}^m$ be another complex random vector related to $\mathbf{x}$ through the relation

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \widetilde{\mathbf{A}}\mathbf{x}^* + \boldsymbol{\epsilon}, \tag{2.16}$$

where $\mathbf{A} \in \mathbb{C}^{m\times n}$ is some linear transform that maps $\mathbf{x}$ to its contribution in $\mathbf{y}$, $\widetilde{\mathbf{A}} \in \mathbb{C}^{m\times n}$ is the linear transform that maps $\mathbf{x}^*$ to its contribution in $\mathbf{y}$, called a *conjugate* transform or *Wirtinger* transform,[20] and $\boldsymbol{\epsilon} \in \mathbb{C}^m$ is the noise term that encompasses the total noise present between each mapping. This definition demonstrates that complex random vectors can contribute more than is intuitively understood. Equation (2.16) mitigates any possibility of ignoring the conjugate term by completely defining all dependencies between complex random vectors. Furthermore, eq. (2.16) reduces to a linear relationship if $\mathbf{y}$ does not depend on $\mathbf{x}^*$. Note that this relationship can still exist regardless of signal impropriety. If the complex random vector is proper, then $\mathbf{x}$ and $\mathbf{x}^*$ are independent. If it is improper, then the vectors are dependent. For this scenario, the noise and its impropriety are independent of the state terms. Therefore, due to the additive property of the noise, the term $\boldsymbol{\epsilon}$ is sufficient to encapsulate the noise information.

Now, widely-linear models are a great tool for demonstrating the underlying statistical relationship in complex random vectors. However, constructing theories and devising estimation procedures based on them can be complicated and unintuitive due to the additional terms within the linear model. Luckily, there exists a succinct way to simplify this necessary model. Define a new random vector $\underline{\mathbf{x}} \in \mathbb{C}^{2n}$ as the random vector $\mathbf{x}$ stacked on its conjugate $\mathbf{x}^*$. This "stacking" procedure is written as follows:

$$\underline{\mathbf{x}} = \begin{bmatrix} \mathbf{x} \\ \mathbf{x}^* \end{bmatrix}. \tag{2.17}$$

This new random vector is called an *augmented* random vector and aims to treat the conjugate random vector as part of the whole probability density. By doing this, the subsequent joint probability densities and related terms naturally incorporate the widely-linear relationship and any pseudo-covariance terms needed without inserting them explicitly. Using the augmented random vector, the resulting augmented covariance matrix is

$$\underline{\mathbf{P}}_{\mathbf{x}} = \begin{bmatrix} \mathbf{P}_{\mathbf{x}} & \widetilde{\mathbf{P}}_{\mathbf{x}} \\ \widetilde{\mathbf{P}}^*_{\mathbf{x}} & \mathbf{P}^*_{\mathbf{x}} \end{bmatrix}, \tag{2.18}$$

[20] The distinct relationship between generalised complex probability models and Wirtinger calculus, the generalisation of complex-valued calculus, is elaborated in §2.4.1. Therefore, although not strictly required, it is preferable to use the latter term to avoid any confusion with the ideas of complex conjugates, as they are not directly related in this context.

where $\underline{\mathbf{P}}_{\mathbf{x}} \in \mathbb{C}^{2n\times 2n}$ and is Hermitian. The notation $\underline{\cdot}$ indicates whether a matrix or vector is *augmented*. Augmented matrices are represented as 2×2 *block* matrices, i.e. comprised of four sub-matrices. The benefits of this augmented model are that all statistical calculations and derivations are semantically similar to *strictly-linear* models, i.e. real-valued linear models, and only require manual augmentation of vectors and matrices. Additionally and most importantly, an augmented linear model is logically equivalent to widely-linear models [33], i.e. if a widely-linear model is defined, e.g. eq. (2.16), it can be rewritten in terms of augmented terms as

$$\underline{\mathbf{y}} = \underline{\mathbf{A}}\underline{\mathbf{x}} + \underline{\boldsymbol{\epsilon}}, \tag{2.19}$$

where $\underline{\mathbf{y}} \in \mathbb{C}^{2m}$ is another augmented complex vector, $\underline{\boldsymbol{\epsilon}} \in \mathbb{C}^{2m}$ is an augmented complex vector that represents the noise term in the system, and $\underline{\mathbf{A}} \in \mathbb{C}^{2m\times 2n}$ is an *augmented* matrix constructed, using $\mathbf{A}$ and $\widetilde{\mathbf{A}}$ defined in eq. (2.16), as

$$\underline{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & \widetilde{\mathbf{A}} \\ \widetilde{\mathbf{A}}^* & \mathbf{A}^* \end{bmatrix}. \tag{2.20}$$

A disadvantage of this formalism is that the number of parameters to model and estimate doubles, increasing the computational complexity of the approach. Thus, an augmented model is not always appropriate if the aim is to be computationally frugal. Balance must be struck to meet the needs of the algorithm, but regardless of the approach, deriving from a widely linear model or an augmented model results in the same derivations and estimation algorithms that capture all statistical behaviour within the full complex plane.

2.2.4 Gaussian Densities

For complex random vectors, the mean, covariance, and pseudo-covariance can be used to derive the intrinsic behaviours of signals to model as a probability density. One way to do this in statistics of real variables is to model these signals with *Gaussian* densities. A Gaussian density is a continuous bell-shaped probability density parameterised by two arguments: a mean vector and a covariance matrix. Multiple practical mathematical methods for modelling and estimation arise from this symmetric density. For complex variables, the definition is extended. If $\mathbf{x} \in \mathbf{C}^n$ is a proper Gaussian complex random vector, then the resulting complex Gaussian density, $\mathcal{CN}_n$, that governs this term can be expressed as

$$\mathcal{CN}_n(\mathbf{m}_{\mathbf{x}}, \mathbf{P}_{\mathbf{x}}) = \frac{1}{\pi^n}\det(\mathbf{P}_{\mathbf{x}})^{-1}\exp\left[-(\mathbf{x}-\mathbf{m}_{\mathbf{x}})^H\mathbf{P}_{\mathbf{x}}^{-1}(\mathbf{x}-\mathbf{m}_{\mathbf{x}})\right], \tag{2.21}$$

where $\det(\,\cdot\,)$ is the complex matrix determinant and $\mathbf{m}_{\mathbf{x}} \in \mathbb{C}^n$ and $\mathbf{P}_{\mathbf{x}} \in \mathbb{C}^{n\times n}$ are the mean and covariance of $\mathbf{x}$, respectively. To show that $\mathbf{x}$ is a Gaussian complex random vector, the notation $\mathbf{x} \sim \mathcal{CN}_n(\mathbf{m}_{\mathbf{x}}, \mathbf{P}_{\mathbf{x}})$, where n denotes the number of dimensions, is used. Note this density models the complex random vector $\mathbf{x}$ and its conjugate $\mathbf{x}^*$, but $\mathbf{x}$ and $\mathbf{x}^*$ are independent. However, if the same complex random vector $\mathbf{x}$ is improper, it is easier to re-write the density in augmented form as

$$\mathcal{CN}_{2n}(\underline{\mathbf{m}}_{\mathbf{x}}, \underline{\mathbf{P}}_{\mathbf{x}}) = \frac{1}{\pi^n} \det(\underline{\mathbf{P}}_{\mathbf{x}})^{-\frac{1}{2}} \exp\left[-\frac{1}{2}(\underline{\mathbf{x}} - \underline{\mathbf{m}}_{\mathbf{x}})^H \underline{\mathbf{P}}_{\mathbf{x}}^{-1}(\underline{\mathbf{x}} - \underline{\mathbf{m}}_{\mathbf{x}})\right], \tag{2.22}$$

where $\underline{\mathbf{m}}_{\mathbf{x}} \in \mathbb{C}^{2n}$ is the augmented mean vector and $\underline{\mathbf{P}}_{\mathbf{x}} \in \mathbb{C}^{2n\times 2n}$ is the augmented covariance matrix. The augmented model works because eq. (2.22) can be seen as a joint density between $\mathbf{x}$ and $\mathbf{x}^*$. Therefore, eq. (2.22) naturally accommodates the inclusion of the important pseudo-covariance within. To denote this relationship between the improper complex random vector and its density in eq. (2.22), write $\underline{\mathbf{x}} \sim \mathcal{CN}_{2n}(\underline{\mathbf{m}}_{\mathbf{x}}, \underline{\mathbf{P}}_{\mathbf{x}})$. Additionally, if the widely linear model formalism is explicitly used, then, without defining the density function, the complex random vector and Gaussian density relationship will be denoted as $\mathbf{x} \sim \mathcal{CN}_n(\mathbf{m}_{\mathbf{x}}, \mathbf{P}_{\mathbf{x}}, \widetilde{\mathbf{P}}_{\mathbf{x}})$, where $\mathbf{P}_{\mathbf{x}}$ is the covariance and $\widetilde{\mathbf{P}}_{\mathbf{x}}$ is the pseudo-covariance.

Finally, if the probability density is conditional, then descriptions remain the same, but in terms of their equivalent conditional means, covariances and pseudo-covariances. However, an additional piece of notation is introduced above to show this distinct scenario. Namely, if the density of $\mathbf{x}$, given some other complex random vector $\mathbf{y} \in \mathbb{C}^m$ for some $m \in \mathbb{N}$, is Gaussian, then, using the augmented Gaussian as an example, it will be denoted as $\underline{\mathbf{x}} \mid \underline{\mathbf{y}} \sim \mathcal{CN}_n(\underline{\mathbf{x}} \mid \underline{\mathbf{m}}_{\mathbf{x}|\mathbf{y}}, \underline{\mathbf{P}}_{\mathbf{x}|\mathbf{y}})$, where $\underline{\mathbf{m}}_{\mathbf{x}|\mathbf{y}}$ is the augmented conditional mean and $\underline{\mathbf{P}}_{\mathbf{x}|\mathbf{y}}$ is the augmented conditional covariance. These augmented arguments can be calculated as

$$\underline{\mathbf{m}}_{\mathbf{x}|\mathbf{y}} = \underline{\mathbf{m}}_{\mathbf{x}} + \underline{\mathbf{P}}_{\mathbf{xy}}\underline{\mathbf{P}}_{\mathbf{y}}^{-1}(\underline{\mathbf{y}} - \underline{\mathbf{m}}_{\mathbf{y}}), \tag{2.23}$$

$$\underline{\mathbf{P}}_{\mathbf{x}|\mathbf{y}} = \underline{\mathbf{P}}_{\mathbf{x}} - \underline{\mathbf{P}}_{\mathbf{xy}}\underline{\mathbf{P}}_{\mathbf{y}}^{-1}\underline{\mathbf{P}}_{\mathbf{xy}}^H. \tag{2.24}$$

The new terms listed in eqs. (2.23) and (2.24) are the augmented cross-covariance matrix $\underline{\mathbf{P}}_{\mathbf{xy}} = \mathrm{Cov}\left[\underline{\mathbf{x}}, \underline{\mathbf{y}}\right] \in \mathbb{C}^{2n\times 2m}$, the augmented auto-covariance matrix $\underline{\mathbf{P}}_{\mathbf{y}} = \mathrm{Var}\left[\underline{\mathbf{y}}\right] \in \mathbb{C}^{2m\times 2m}$ and the augmented mean vector $\underline{\mathbf{m}}_{\mathbf{y}} = \mathrm{E}\left[\underline{\mathbf{y}}\right] \in \mathbb{C}^m$. To distinguish between eqs. (2.21) and (2.22), the latter will be called the *general* Gaussian density and the former will be called the *proper* Gaussian density. Equation (2.22) is also called a "general" complex Gaussian density because it can be applied to both proper and improper random vectors as long as the augmented covariance matrix is appropriately constructed. Going forward, the proper Gaussian will be used to present mathematical concepts and derivations to follow for simplicity and brevity. However, the methods discussed in §2.4 will carry over to the general Gaussian density.

2.2.5 Stochastic Processes

As defined in §2.1.3, a stochastic process models a signal using difference equations that indicate subtle changes between the random physical states of a signal over the time axis. To represent this mathematically, the signal is defined over an ordered set or sequence of possible random states that belong to a state space that describes the signal over time. A state space can be considered a sample space for a signal's physical states. Due to this randomness, probability theory can be applied to model the behaviour of this signal. Let the set $\mathbb{T}$ represent the time axis and generate a complex signal by some function $\mathbf{x}(t) : \mathbb{T} \longrightarrow \mathbb{C}^n$,

where $n \in \mathbb{N}$. For some $k \in \mathbb{T}$, the output $\mathbf{x}(k)$ represents the actual state of the signal at time k. To introduce randomness, let $\mathbf{q}_k \in \mathbb{C}^n$ be an independent *white noise* term representing the random behaviour of the signal present at time k, called the *process* noise vector. Add this process noise term to the signal function to get the aforementioned defined random state:

$$\mathbf{x}_k = \mathbf{x}(k) + \mathbf{q}_k. \tag{2.25}$$

White noise is a random state governed by a zero-mean Gaussian process, called a *white noise* process. When Gaussians can model all states within a stochastic process, it is called a *Gaussian* process. For a white noise process, each Gaussian has an expectation of zero and has a unique diagonal covariance matrix. Furthermore, each white noise term is independent or uncorrelated with other elements within the white noise vector and other white noise terms at different points in time. Suppose that each state's mean vectors and covariance matrices are generated using the functions $\mathbf{m}(t)$ and $\mathbf{P}(t)$, respectively. Then, a Gaussian process can be denoted as $\mathcal{CP}_n(\mathbf{m}(t), \mathbf{P}(t))$ over $\mathbb{T}$. The covariance function $\mathbf{P}(t)$ is also referred to as a *kernel* and, with the expectation function, completely defines the characteristics of the Gaussian process.

The process noise vector in eq. (2.25) is governed by $\mathcal{CP}_n(\mathbf{0}, \mathbf{Q}(t))$ where $\mathbf{Q}(t) : \mathbb{T} \longrightarrow \mathbb{R}^{n \times n}$ is a function that outputs unique diagonal covariance matrices of process noise covariance per state element. That is, for each $x \in \mathbf{x}_t$ for some $t \in \mathbb{T}$, there exists a corresponding complex variance denoted $\sigma_Q^2 \in \mathbf{Q}(t)$. Here, the term σ_Q is referred to as the *process noise* and it is the uncertainty associated with state element x. Moving forward, for simplicity, noise terms will be represented as samples from individual Gaussian densities within the Gaussian process. Specifically, for any time k in $\mathbb{T}$, the noise term $\mathbf{q}_k$ is drawn from a complex Gaussian distribution: $\mathbf{q}_k \sim \mathcal{CN}_n(\mathbf{0}, \mathbf{Q}_k)$, where $\mathbf{Q}_k$ is an output of the kernel $\mathbf{Q}(t)$ at $t = k$. The kernel models the noise as white, i.e. each output of $\mathbf{Q}(t)$ is uncorrelated in time. This entire signal can be considered the "randomness" component in the original signal of interest. The definition of the random component here is a special case of the generalised version. Particularly, this version of it is called "independent additive" noise because it is not dependent on the function that generates the signal or the state-to-state functions. There exist methods to handle such non-additive noise cases, but for simplicity, the former case is used [23], [24]. The process noise σ_Q models the variability and uncertainty into the evolution of the signal's true states over time. Understanding its behaviour allows for understanding the dynamics of the desired signal. The ordered set of all states generated from eq. (2.25) over $\mathbb{T}$ is the state space of the signal, denoted as $(\mathbf{x}_t \mid t \in \mathbb{T})$ and represents the entire stochastic process.[21] For a given relative time frame $\mathbb{T}_N = (1, ..., N) \subset \mathbb{T}$ and $N \in \mathbb{N}$, an observed random signal can be described as

$$(\mathbf{x}_t \mid t \in \mathbb{T}_N) = (\mathbf{x}_1, ..., \mathbf{x}_N). \tag{2.26}$$

[21] Stochastic processes are often written as $(\mathbf{x}_t)_{t \in \mathbb{T}}$; however, the current notation is used to be consistent with the style used within this dissertation.

The terms "time frame" and "time axis" will be used interchangeably, chosen based on the specific needs and clarity of the discussion in which they appear. For brevity, ordered sets like eq. (2.26) are written using the shorthand notation $\mathbf{x}_{1:N}$ to avoid possible confusion with conditional densities. A given probability density models each random state in the stochastic process. Namely, for $k \in \mathbb{T}$, the density function of $\mathbf{x}_k$ is $p(\mathbf{x}_k)$. For BFS theory, the modelling perspective is focused on localised changes between sequential sample states of the signal rather than considering the entire sampled signal. To rewrite these densities using this idea, stochastic difference equations are required. For any time $k \in \mathbb{T}_N$, define a recursive vector function $\mathbf{f}$, called the *dynamics* or *evolution* function, that describes the evolution from the state $\mathbf{x}_{k-1}$ to $\mathbf{x}_k$. With the evolution function and the state formalism, the expression in eq. (2.25) can be re-imagined as

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}) + \mathbf{q}_{k-1}. \tag{2.27}$$

This is a *stochastic difference equation* and is also known as a *state space* model.[22] It is named as such because it can be re-expressed as $\mathbf{x}_k - f(\mathbf{x}_{k-1}) = \mathbf{q}_{k-1}$, i.e. the difference between the random state and the evolution of the previous state is the added randomness. Since the stochastic difference equation is defined inside a state space with the evolution function, it is called an *evolution* model since it describes the evolution of the signal over time. It is also referred to as the *dynamics* model, but the former is used as it is more intuitive to understand [24].

Say there exists another separate signal $\mathbf{y}(t) \in \mathbb{C}^m$ defined over the same time axis with a separate state space $(\mathbf{y}_t \mid t \in \mathbb{T})$ for some $m \in \mathbb{N}$. Define another function $\mathbf{h}$, called the *measurement* function, which describes, for some $k \in \mathbb{T}_N$, how the state $\mathbf{x}_k$ relates to state $\mathbf{y}_k$. At time $k \in \mathbb{T}_N$, let the noise term of the random state $\mathbf{y}_k$, called the *measurement* noise vector, be $\mathbf{r}_k \sim \mathcal{CN}_m(\mathbf{0}, \mathbf{R}_k)$, where $\mathbf{R}_k \in \mathbb{R}^{m \times m}$ is the noise covariance matrix governed by the *measurement noise* denoted σ_R. This functional relationship also forms a stochastic difference equation called the *measurement* model, written as

$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{r}_k. \tag{2.28}$$

The reason for its name is that if $\mathbf{y}_k$ is measured, then the difference between $\mathbf{y}_k$ and the result of $\mathbf{h}(\mathbf{x}_k)$ is the random noise $\mathbf{r}_k$ included in the measurement of $\mathbf{y}_k$. The measurement noise σ_R models the uncertainty in the measurements and the imperfections in the observation process. From eqs. (2.27) and (2.28), an unknown and hidden signal, $\mathbf{x}(t)$, can be modelled using measurements of $\mathbf{y}(t)$ and the relationship between the hidden signal and the measurement signal. Furthermore, this is done recursively and at a localised level using state-space models rather than modelling entire portions of the signal at a time like batch estimation techniques.

The final piece needed to derive BFS theory and the KFS is the concept of a *Markov-chain* or *Markov* process. Simply stated, a Markov process is a stochastic process that satisfies the

[22] The time index of the noise term has changed to reflect the recursive nature of this new expression. However, the semantics has not been altered.

"Markov property of states" or simply the *Markov* property. Using the above signal functions $\mathbf{x}(t)$ and $\mathbf{y}(t)$, the former signal is a Markov process if for any $k \in \mathbb{T}$:

- The state $\mathbf{x}_k$ is independent of the previous states of $\mathbf{x}(t)$ and $\mathbf{y}(t)$ up to time $k-1$, except for $\mathbf{x}_{k-1}$,
- And, the state $\mathbf{x}_{k-1}$ is independent of future states of $\mathbf{x}(t)$ and $\mathbf{y}(t)$ until time k, except for $\mathbf{x}_k$.

A Markov process is necessary for formulating the recursive estimation solution since, with the evolution and measurement model, it restricts the dependence of any given state to those in the previous time step. Aside from the mathematical motivations, Markov processes also reflect the physical reality of many systems where future states depend primarily on the current state. Using the newly defined evolution and measurement model relationships, it would be appropriate to think that the state $\mathbf{x}_k$ would be modelled by some conditional density $p(\mathbf{x}_k \mid \mathbf{x}_{1:k-1}, \mathbf{y}_{1:k-1})$ because of the recursive nature of eqs. (2.27) and (2.28). Similarly, the state $\mathbf{x}_{k-1}$ would be modelled by another conditional density of future states if looking through the inverted equivalents of the evolution and measurement functions, i.e. $p(\mathbf{x}_{k-1} \mid \mathbf{x}_{k:N}, \mathbf{y}_{k:N})$. Generally, probability densities are difficult to reform as recursive conditional relationships. However, if the signal from which these states are derived is a Markov process, then, using the conditional independence identity mentioned in eq. (2.4), these densities reduce to

$$p(\mathbf{x}_k \mid \mathbf{x}_{1:k-1}, \mathbf{y}_{1:k-1}) = p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) \quad \text{and} \quad p(\mathbf{x}_{k-1} \mid \mathbf{x}_{k:N}, \mathbf{y}_{k:N}) = p(\mathbf{x}_{k-1} \mid \mathbf{x}_k). \tag{2.29}$$

Even without the Markov property, the batch-like joint density of a sampled signal in the time frame $\mathbb{T}_N$ can be re-expressed as a product of recursive conditional densities by repeatedly applying the chain rule (see eq. (2.6)) to get

$$p(\mathbf{x}_{1:N}) = p(\mathbf{x}_1) \cdot \prod_{i=2}^{N} p(\mathbf{x}_i \mid \mathbf{x}_{1:i-1}). \tag{2.30}$$

The role of the Markov property is to restrict the recursive dependence on past states. Using the result in eq. (2.29), the eq. (2.30) is reformulated as

$$p(\mathbf{x}_{1:N}) = p(\mathbf{x}_1) \cdot \prod_{i=2}^{N} p(\mathbf{x}_i \mid \mathbf{x}_{i-1}). \tag{2.31}$$

This localisation of the recursive conditional densities simplifies the modelling and estimation of stochastic processes, essential for practical algorithms like the KFS. This is one of many benefits of Markov processes, but eq. (2.31) describes the purpose and idea to assist in formulating recursive probabilistic solutions. For now, the details of stochastic processes will continue in §2.3. The next step is to discuss estimations of states in the context of these densities.

2.2.6 Inference and Estimation

Once probability densities are constructed for a desired signal, the subsequent step involves inferring the signal states based on these models or underlying assumptions about model parameters. *Inference* involves drawing conclusions about the signal based on experimental evidence. [33]. With appropriate probability models, it provides a comprehensive understanding of the signal and its underlying properties. To simplify, inference offers a complete picture of a signal's behaviour that can answer various questions, such as the likelihood of certain signal characteristics, the range of potential state outcomes, and the degree of confidence in predictions. However, performing full inference can be computationally intensive, particularly when dealing with large state dimensionality, as it involves working with comprehensive probability densities that integrate over extensive parameter spaces. *Estimation*, a subset of inference, offers a more practical and computationally feasible approach. The choice between full inference and estimation depends on the specific requirements of the statistical problem, the computational resources available and the required precision for decision-making. Depending on these requirements, estimation entails finding the most representative value or "best guess" for a model parameter or state, known as *point estimates* [24]. These point estimates are derived via a *minimisation problem*. Consider the desired model parameter or state, denoted as $\mathbf{x} \in \mathbb{C}^n$. To retrieve a point estimate of $\mathbf{x}$, denoted as $\hat{\mathbf{x}}$, solve the following via minimisation:

$$\hat{\mathbf{x}} = \underset{\hat{\mathbf{x}} \in \mathbb{C}^n}{\operatorname{argmin}} \, L(\mathbf{e}), \tag{2.32}$$

where

$$\mathbf{e} = \mathbf{y} - \mathbf{h}(\hat{\mathbf{x}}) \tag{2.33}$$

In this context, $\mathbf{y} \in \mathbb{C}^m$ represents observed data, and $\mathbf{h} : \mathbb{C}^n \to \mathbb{C}^m$ is the *measurement* operator or *function* that describes the relationship between $\mathbf{x}$ and $\mathbf{y}$. The term $\mathbf{e} \in \mathbb{C}^m$ is called the *error* or *residual* vector and quantifies the discrepancy between the actual data $\mathbf{y}$ and the estimate of the data $\mathbf{h}(\hat{\mathbf{x}})$ given some estimate $\hat{\mathbf{x}}$ of $\mathbf{x}$. Finally, $L : \mathbb{C}^m \to [0, \infty)$ is a non-negative convex function called the *loss function*. This function details how to score the accuracy of the choice of $\hat{\mathbf{x}}$ via the error vector based on the requirements of the estimation problem. Therefore, the "best" point estimate $\hat{\mathbf{x}}$ of $\mathbf{x}$ under the given conditions minimises the chosen L.

The selection of a loss function depends upon the specific requirements and characteristics of the estimation problem. There exist multiple choices for L that are suitable for various subject fields and scenarios of estimation, which is further discussed by Jazwinski [23, p. 146-147]. However, when using probabilistic models, a common approach is to define the loss function as

$$L(\mathbf{e}) = -\ln p(\mathbf{y} \mid \mathbf{x}). \tag{2.34}$$

This approach is referred to as the *Negative Log-Likelihood* (NLL), where $p(\mathbf{y} \mid \mathbf{x})$ is known as the *likelihood* or *measurement* density. A point estimate derived from the NLL corresponds to the most probable realisation of the measurement density, i.e. the *mode* of $p(\mathbf{y} \mid \mathbf{x})$. Note, this point estimate coincides with an estimate derived from the *Maximum Likelihood* (ML) approach, which is the same procedure, except the goal is to maximise the (positive) log-likelihood instead. This point estimate is called the *Maximum Likelihood Estimate* (MLE) and represents the point estimate that best describes the observed data. This is the traditional method used within calibration for several reasons: it is effective and can be applied to linear or non-linear problems or Gaussian or non-Gaussian densities. Particularly, when assuming the noise can be modelled by (proper) Gaussian densities, also known as *Gaussian* noise, the ML whittles down to a quadratic problem where

$$p(\mathbf{y} \mid \mathbf{x}) = \mathcal{CN}_m(\mathbf{y} \mid \mathbf{h}(\mathbf{x}), \mathbf{R}). \tag{2.35}$$

The term $\mathbf{R}$ represents the *measurement* covariance matrix, discussed in detail in §2.2.5. For now, know it governs the measurement noise incorporated within the observations of $\mathbf{y}$. The NLL of eq. (2.35) using eq. (2.21) results in the following quadratic loss function:

$$L(\mathbf{e}) \propto \mathbf{e}^H \mathbf{R}^{-1} \mathbf{e} = (\mathbf{y} - \mathbf{h}(\mathbf{x}))^H \mathbf{R}^{-1} (\mathbf{y} - \mathbf{h}(\mathbf{x})) \tag{2.36}$$

The omitted constant term $-\ln(\pi^m \det(\mathbf{R}))$ is not necessary for minimisation since it does not depend on $\mathbf{x}$; however, it is essential to acknowledge since it will appear within §2.4.3. The same loss function is discussed within §1.2.4, except the measurement function is approximated as some linear transform. From eq. (2.36), several solutions can be derived by assuming some measurement model through $\mathbf{h}$ and differentiating L with respect to $\mathbf{x}$. Note, eq. (2.36) coincides with the least squares approach, and therefore, with Gaussian densities, the MLE coincides with the well-known *Minimum Mean Square Error* (MMSE), i.e. the mode and mean are equivalent. In calibration, the measurement model, i.e. the RIME, is non-linear; thus, extensions are applied to make eq. (2.36) viable for non-linear estimation problems. The primary discussion of non-linear estimation is deferred to sections 2.4.1 and 2.4.2, but know that a traditional approach in calibration is to linearise the non-linear model as an approximate of the equivalent linear case. When combined with methods like Gauss-Newton and Levenberg-Marquardt, non-linear estimations are addressed using iterative batch solvers to enhance computational feasibility. Specifically, the relevant terms are divided into batches in memory and repeatedly solved independently till L is sufficiently minimised. As a result, this approach is effective and efficient for most estimation tasks because it lends itself well to parallel and distributed computing techniques [1], [37]. To learn more, Kenyon *et al.* [1] and Sob *et al.* [37] provide an insightful discussion on the nature of these ML solvers within calibration.

Although pragmatic, the ML approach is susceptible to overfitting the noise within the measurements, especially when the magnitude of the noise is high. Therefore, regularisation is needed to reduce this effect and improve estimate accuracy. Including some regularisation alongside the ML formulation is often cumbersome and requires imposing certain assumptions

on the probability models or the loss function. This is why the solution intervals technique is utilised in conjunction with this ML estimation approach in traditional calibration: it reduces overfitting of noise *and* is a relatively simple form of regularisation to incorporate [1], [20], [37]. Furthermore, the MLE can only assist with describing full probability densities when the models are Gaussian where the mode and mean are equivalent. However, this description is incomplete without a corresponding covariance (or even pseudo-covariance), which adds computational complexity to the process. Moreover, this only applies in the case of Gaussian densities. Consequently, the construction and inference of probability densities from MLEs is often impractical. Following on from that point, when considering signal systems at Big Data scales, like within calibration, the computational complexity of solving for these estimates rapidly becomes high. This is caused by the need to hold all the terms in memory and perform multiple expensive matrix operations. Consequently, several approximations, assumptions and high-performance computing strategies are required to make estimating feasible, e.g. see [1], [3], [11], [38].

Consequently, while the MLE serves as a practical point estimate for identifying specific model parameters or states, it becomes impractical for comprehensive inference. In addition, incorporating regularisation within the ML loss function is not seamless. Although implementing solution intervals alongside the ML improves estimate quality, inference cannot be achieved without additional computational steps. The above section establishes the concepts of inference and estimation and demonstrates the limitations of the ML approach, especially within the context of calibration and in comparison to Bayesian forms of inference and estimation.

2.2.7 The Bayesian Perspective

The philosophy of Bayesian reasoning can be determined straight from Bayes' Theorem in eq. (2.7). Probabilistic information is about a given random vector that can be improved by including prior information about unknown variables within the model and updating this information given related observations. These unknown variables can be the initial model of the random vector, assumptions about noise within the models, and, to an extent, the evolution and measurement functions (see §2.2.5). Then, an improved probabilistic model can be determined using a combination of conditional densities between the unknown hidden signal and a related measurable noisy signal. Below, each probability density within eq. (2.7) is described in more depth to enhance this explanation [24]:

- *Prior density*, $p(\mathbf{x})$ - Also known as the *a priori* knowledge of $\mathbf{x}$, it describes what is known about the marginal density of $\mathbf{x}$ before looking at how it statistically relates to the measurement $\mathbf{y}$.
- *Likelihood* or *Measurement density*, $p(\mathbf{y} \mid \mathbf{x})$ - The *likelihood* model of measurement $\mathbf{y}$ given information of $\mathbf{x}$, i.e. how is $\mathbf{y}$ affected by observing $\mathbf{x}$.

- *Posterior density*, $p(\mathbf{x} \mid \mathbf{y})$ - Also known as the *a posteriori* knowledge of $\mathbf{x}$ given the measurement $\mathbf{y}$, it describes the prior model updated based on the information given by the measurement density and observations of $\mathbf{y}$.

- *Marginal likelihood* or *Evidence*, $p(\mathbf{y})$ - Indicates the probability of observing a measurement $\mathbf{y}$ after marginalising over the signal states and assuming a specific measurement model. From the perspective of Bayes theorem, it is only a normalisation constant that ensures the posterior probability density integrates to unity. However, it can be seen as the amount of support or *evidence* for observing the measurements under the model assumptions.

The Bayesian approach to modelling uses prior knowledge *(prior density)*. It refines it with observations *(measurement density)* to produce an updated model *(posterior density)* that can be used for signal modelling purposes. This is the overarching strategy for Bayesian probabilistic modelling and, more importantly, BFS.

In the context of inference, *Bayesian inference* applies to the posterior density, offering a dynamic method that refines understanding based on new data. Bayesian inference is often seen as an advanced form of inference because it integrates prior information, leading to more informed and contextually relevant conclusions. Moreover, the prior density inherently introduces a form of regularisation into the probability models. This inherent feature of Bayesian regularisation counteracts overfitting, improves generalisation, and allows for more sophisticated regularisation techniques. These advanced regularisation methods in Bayesian frameworks can adaptively conform to the complexities of the data and underlying processes, often leading to more nuanced and contextually rich signal models. Unlike the ML approaches, where regularisation often demands manual adjustments and expert input, Bayesian methods allow for seamless integration of prior knowledge and assumptions by selecting an appropriate prior model. Overall, Bayesian inference offers a comprehensive and adaptable modelling approach that effectively balances prior knowledge against new data insights.

Like traditional inference, full Bayesian inference is often computationally impractical. With *Bayesian estimation*, point estimates are derived with respect to the posterior density, differing from the ML approach that focuses on the measurement density. Given the same terms discussed in §2.2.6, a typical general loss function for *Bayesian estimation* is expressed as

$$L(\mathbf{e}) = -\ln p(\mathbf{x} \mid \mathbf{y}). \tag{2.37}$$

Point estimates derived from eq. (2.37) are known as *Maximum A Posterior* (MAP). It represents an enhanced estimate that considers both prior information *and* measurement information. To calculate eq. (2.37), the prior density must be determined and subsequently incorporated alongside the measurement and evidence density. This adds additional computational requirements and why the ML approach is often more computationally appropriate. When determining MAPs, the computational complexity can be reduced by re-expressing eq. (2.37), after applying Bayes' theorem (eq. (2.7)), to get

$$L(\mathbf{e}) \propto -\ln\left(p(\mathbf{y} \mid \mathbf{x})p(\mathbf{x})\right). \tag{2.38}$$

The evidence density is omitted since it does not depend on $\mathbf{x}$. This would be the standard approach to determining MAPs from posterior densities. Additionally, this could be applied to the BFS solutions to arrive at appropriate point estimates similar to those derived from ML approaches. However, the KFS is not designed around point estimates, i.e. it does not explicitly minimise some loss function. Instead, the KFS determines the parameters needed to describe the total Gaussian densities. Consequently, estimation is not required, and full inference can be applied, demonstrating the powerful benefits of the KFS algorithms compared to methods like purely estimating MAPs and MLEs via a loss function. Interestingly, the KFS is based on Gaussian densities, meaning the mean and mode coincide. As a result, the means calculated by the KFS are, in fact, MAPs of their respective posterior densities, similar to the MLE and MMSE discussed in §2.2.6. Note, this is purely a relationship that arises with the design of eq. (2.37) with Gaussian densities. Although the MAP approach is not explicitly used within the KFS, it is crucial to understand in the context of the *energy function* in §2.4.3.

2.3 Bayesian Filtering and Smoothing

Within this section of chapter 2, the BFS and KFS are derived and defined. This is done using the concepts of stochastic processes in §2.2.5 and the Bayesian approach discussed in §2.2.7. Unlike the steps taken by Kalman [58] or Rauch, Striebel, and Tung [25], the KFS will be constructed as a special case of the BFS solutions following Särkkä [24]. As a reminder, this section presents the theory assuming that complex random vectors are proper. The corresponding improper versions are given in §2.4.

2.3.1 Bayesian Filtering Solution

Bayesian filtering emerges as a succinct probabilistic solution to a generalised filtering problem, as discussed in §2.1.2, by leveraging the recursive nature of Markov processes coupled with Bayesian modelling and inference techniques. The Bayesian approach allows for managing uncertainties in the system, e.g. measurement noise, and integrates prior knowledge to enhance existing model information. The result is a dynamic estimation process that incrementally and continuously assimilates new measurements to refine the predicted states. The Bayesian filtering solution distinguishes itself from similar static techniques by continuously adapting and improving models. This is what lends itself so well to on-line estimation problems. By considering a suitable prior model, the solution provides inherent regularisation against noise for posterior densities, reducing the risk of overfitting and improving model accuracy. Furthermore, it is formulated using probability densities, which form the basis for deriving estimation algorithms like the KF and facilitates the creation of a specialised "Bayesian prediction solution" as a by-product.

To formalise the Bayesian filtering solution, the associated filtering problem is defined. Consider a stochastic process $\mathbf{x}_{1:k}$ that describes the evolution of states for an unknown hidden signal $\mathbf{x}(t) \in \mathbb{C}^n$ up until time $k \leq N$ in the time frame $\mathbb{T}_N$ where $n, N \in \mathbb{N}$. Sequential measurements $\mathbf{y}_{1:k}$ are collected from an associated, noisy and measurable signal $\mathbf{y}(t) \in \mathbb{C}^m$ where $m \in \mathbb{N}$. This data collection process can be either real-time or off-line, with the method chosen having no impact on the overall approach. For a Bayesian approach, the central task is to construct the posterior density $p(\mathbf{x}_k \mid \mathbf{y}_{1:k})$, i.e. the probability of current state $\mathbf{x}_k$ given the measurements $\mathbf{y}_{1:k}$. This is called the *filtering* density at time k and represents the ideal model to infer the current state, given the prescribed assumptions imposed on the models. The next step is to re-express the filtering density through Bayes' theorem (eq. (2.7)) and rewrite it in a recursive form. For this, assume $\mathbf{x}_{1:N}$ is a Markov process. In addition, for $k \in \mathbb{T}_N$, each measurement within $\mathbf{y}_{1:k}$ is assumed to be *conditionally independent* of each other and the previous states $\mathbf{x}_{1:k-1}$. This assumption can be written as $p(\mathbf{y}_k \mid \mathbf{x}_{1:k}, \mathbf{y}_{1:k-1}) = p(\mathbf{y}_k \mid \mathbf{x}_k)$, and is known as the *conditional independence of measurements* property. Assuming this property means that each measurement is influenced only by the state at that moment and is independent of past states or measurements. Together with the Markov property, they allow these large joint probability densities to be decomposed into recursive time steps of smaller densities. The time axis is decomposed for this discussion as it is more intuitive and common. However, this formulation can be applied to any axis within the system, e.g. frequency.

With the recursive formulation, stochastic difference equations are naturally introduced to describe the statistical behaviour between recursive states and measurements. For the Bayesian filtering solution, the evolution density $p(\mathbf{x}_k \mid \mathbf{x}_{k-1})$ is needed to describe the behaviour of the hidden signal recursively, and the measurement density $p(\mathbf{y}_k \mid \mathbf{x}_k)$ is required to relate the states with the measurements (see §2.2.5). Given this setup, the Bayesian filtering solution for each state at each time $k \in \mathbb{T}_N$ can be derived and displayed via the following theorem.

Theorem 2.3.1 (Bayesian Filtering Solution). *Särkkä [24].* Given a Markov process $\mathbf{x}_{1:N}$ and a set of conditionally independent measurements $\mathbf{y}_{1:N}$, the Bayesian filtering solution recursively computes the *filtering* density $p(\mathbf{x}_k \mid \mathbf{y}_{1:k})$ forward in time at each step $k \in \mathbb{T}_N$, as

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) = \frac{1}{z_k} p(\mathbf{y}_k \mid \mathbf{x}_k) p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}), \tag{2.39}$$

where z_k is calculated as

$$z_k = \int p(\mathbf{y}_k \mid \mathbf{x}_k) p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}) \, \mathrm{d}\mathbf{x}_k, \tag{2.40}$$

where the *prediction* density is given by

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}) = \int p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) p(\mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1}) \, \mathrm{d}\mathbf{x}_{k-1}. \tag{2.41}$$

The recursion is initiated with some prior density $p(\mathbf{x}_0)$. A detailed proof is given in proof A.4.6.

Theorem 2.3.1 can be easily explained by elaborating the role of each density within:

- Current *filtering* density, $p(\mathbf{x}_k \mid \mathbf{y}_{1:k})$ - The filtering solution density for the current time step k.
- Current *measurement* density, $p(\mathbf{y}_k \mid \mathbf{x}_k)$ - The conditional probability model of the current measurement $\mathbf{y}_k$ given the current state $\mathbf{x}_k$. It describes the measurement relationship between $\mathbf{x}_k$ and $\mathbf{y}_k$, i.e. the measurement model.
- Current *prediction* density, $p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1})$ - The prediction solution density for the current time step k with past measurements $\mathbf{y}_{1:k-1}$.
- Current *evolution* density, $p(\mathbf{x}_k \mid \mathbf{x}_{k-1})$ - The conditional probability model of the current state $\mathbf{x}_k$ given the previous state $\mathbf{x}_{k-1}$. It describes the evolution between consecutive states, $\mathbf{x}_k$ and $\mathbf{x}_{k-1}$, i.e. the evolution model.
- Previous *filtering* density, $p(\mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1})$ - The filtering density acquired for time step $k-1$.
- Current *evidence*, z_k - The scalar normalising term to ensure the filtering solution remains a valid probability density. It also describes how well the current models explain the observed data $\mathbf{y}_{1:k}$.

The Bayesian filtering solution can be separated into two steps: a *prediction* and *update* step. The prediction step is constructing the prediction density $p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1})$ using eq. (2.41). Without the current measurement $\mathbf{y}_k$, the best solution is to compute it by considering the previous filtering density $p(\mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1})$ propagated through evolution density $p(\mathbf{x}_k \mid \mathbf{x}_{k-1})$ and realising the previous state $\mathbf{x}_{k-1}$. That is, assuming the previous state can be described by its own filtering density, the information within can be leveraged for the current state by using the knowledge of how the stochastic process transitions from the previous state to the current state via the evolution density. The result is a prediction of the current state before including the current measurement. The update step is also known as the *correction* step, because it determines how good the prediction solution, $p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1})$, describes the current measurement $\mathbf{y}_k$ through the measurement density, $p(\mathbf{y}_k \mid \mathbf{x}_k)$, and corrects the prediction accordingly. In essence, the prediction made during the prediction step is updated with the current measurement information to formulate the filtering solution at a given point in time.

To find a Bayesian filtering solution for each state, begin with some external prior density $p(\mathbf{x}_0)$ as the previous filtering density for the first prediction step, and then apply the above two-step procedure recursively for $k \in \mathbb{T}_N$. This is called a *forward recursion* algorithm because it constructs each density solution recursively using the previous density solution in time. With the prediction and update steps, the Bayesian filtering solution can continuously make predictions of new states, adapt to new data and enhance its accuracy over time. When

considering algorithms that utilise MAPs instead of complete densities, the evidence density is not required because it does not depend on the current state (see §2.2.7). It is only needed if a measure of the performance of the assumptions applied to the probability densities, e.g. uncertainty of noise or appropriateness of initial priors, is required. If not required, whether calculating MAPs or not, eq. (2.39) is often simplified to

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) \propto p(\mathbf{y}_k \mid \mathbf{x}_k)p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}), \tag{2.42}$$

like the loss function defined in eq. (2.38). This is one of the most common forms of the Bayesian filtering solution and forms the foundation to derive the KF in the next section.

2.3.2 Kalman Filter

Building upon the Bayesian filtering solution, the KF emerges as the definitive solution for linear state-space models with Gaussian noise. Unlike standard MAP estimation techniques, which primarily focus on point estimates, the KF excels in determining the complete set of parameters for the Gaussian density, i.e. the mean and the covariance. In the linear case, the evolution and measurement functions in eqs. (2.27) and (2.28) devolve into linear transforms, and the Markov processes are now Gaussian processes. Consequently, they are now labelled as *Gauss-Markov* processes and are essential for deriving the KF. At any discrete time $k \in \mathbb{T}_N$, eqs. (2.27) and (2.28) are expressed as the linear expressions

$$\mathbf{x}_k = \mathbf{F}_{k-1}\mathbf{x}_{k-1} + \mathbf{q}_{k-1}, \qquad \mathbf{q}_{k-1} \sim \mathcal{CN}_n(\mathbf{0}, \mathbf{Q}_{k-1}), \tag{2.43}$$

$$\mathbf{y}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{r}_k, \qquad \mathbf{r}_k \sim \mathcal{CN}_m(\mathbf{0}, \mathbf{R}_k). \tag{2.44}$$

The new terms within these linear state-space models are the *evolution* matrix, $\mathbf{F}_{k-1} \in \mathbb{C}^{n\times n}$, which describes the mapping of the previous state to the current state, and *measurement* matrix, $\mathbf{H}_k \in \mathbb{C}^{m\times n}$, which describes the mapping of the current state to the measurement. Since $\mathbf{x}_{1:N}$ is a Gauss-Markov process, a Gaussian density governs each state, i.e. $\mathbf{x}_k \sim \mathcal{CN}_n(\mathbf{m}_k, \mathbf{P}_k)$ where $\mathbf{m}_k \in \mathbb{C}^n$ is the expectation of $\mathbf{x}_k$ and $\mathbf{P}_k \in \mathbb{C}^{n\times n}$ is the covariance. For clarity, only the time indices will be used for the states' mean vectors and covariance matrices, assuming all terms henceforth are derived from the $\mathbf{x}(t)$ signal. For now, assume the Gaussian densities are proper. The same logic extends to the general Gaussian densities, but an additional pseudo-covariance term is included. Due to lemmas A.4.2 and A.4.3, the measurement signal $\mathbf{y}(t)$ is also a Gaussian process, and all conditional densities possible between states of $\mathbf{x}_{1:N}$ and $\mathbf{y}_{1:N}$ are subsequently Gaussian. As a result, the critical densities in the Bayesian filtering solution in eq. (2.42), such as the prediction and filtering density, can be modelled as Gaussian, i.e. the filtering density at time $k \in \mathbb{T}_N$ is modelled as

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) = \mathcal{CN}(\mathbf{m}_{f,k}, \mathbf{P}_{f,k}), \tag{2.45}$$

where $\mathbf{m}_{f,k} \in \mathbb{C}^n$ is the mean and $\mathbf{P}_{f,k} \in \mathbb{C}^{n\times n}$ is the covariance. The subscript f denotes "filter" to indicate its association with the filtering density. Similarly, the prediction density is modelled as

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}) = \mathcal{CN}(\mathbf{m}_{p,k}, \mathbf{P}_{p,k}), \tag{2.46}$$

where $\mathbf{m}_{p,k} \in \mathbb{C}^n$ is the mean and $\mathbf{P}_{p,k} \in \mathbb{C}^{n\times n}$ is the covariance. Here, the subscript p denotes "predictor" since it is associated with the prediction density. For this research, an explicit label "predictor" is given for prediction density solutions to accentuate its place within the KFS framework. However, this is not usually done since it is often seen as a trivial component as described by Jazwinski [23], Särkkä [24], and Saberi, Stoorvogel, and Sannuti [55]. Given this setup of densities, the desired density parameter quantities can be derived by computing and matching these quantities between Gaussian densities. This follows a similar process used to arrive at the NLL in eq. (2.34) and the MAP loss function in eq. (2.38). Therefore, for each state $\mathbf{x}_k \in \mathbf{x}_{1:N}$, the goal is first to compute, using the previous step's results, the *predictor* mean $\mathbf{m}_{p,k}$ with *predictor* covariance $\mathbf{P}_{p,k}$ to acquire the prediction density. Second, with the prediction density parameters and the newest measurement $\mathbf{y}_k$, compute the *filter* mean $\mathbf{m}_{f,k}$ with *filter* covariance $\mathbf{P}_{f,k}$ to retrieve the desired filtering density. Thus, the KF is simply the set of expressions that compute these steps, starting with some initial filter mean $\mathbf{m}_{f,0}$ and covariance $\mathbf{P}_{f,0}$, for each state in $\mathbf{x}_{1:N}$. This is succinctly summarised in the following algorithm.

Algorithm 2.3.2 (Kalman Filter). *Särkkä [24].* For a given time frame $\mathbb{T}_N$ with measurements $\mathbf{y}_{1:N}$, and prior filter mean $\mathbf{m}_{f,0}$ and covariance $\mathbf{P}_{f,0}$, the *Kalman Filter* (KF) acquires the Gaussian densities for each state in $\mathbf{x}_{1:N}$ by recursively computing the predictor means $\mathbf{m}_{p,1:N}$ and covariances $\mathbf{P}_{p,1:N}$ followed by the filter means $\mathbf{m}_{f,1:N}$ and covariances $\mathbf{P}_{f,1:N}$ via forward recursion with the following steps:

1.) **Prediction Step**:

$$\mathbf{m}_{p,k} = \mathbf{F}_{k-1}\mathbf{m}_{f,k-1}, \tag{2.47}$$
$$\mathbf{P}_{p,k} = \mathbf{F}_{k-1}\mathbf{P}_{f,k-1}\mathbf{F}_{k-1}^H + \mathbf{Q}_{k-1}, \tag{2.48}$$

2.) **Update Step**:

$$\mathbf{e}_k = \mathbf{y}_k - \mathbf{H}_k \mathbf{m}_{p,k}, \tag{2.49}$$
$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{p,k} \mathbf{H}_k^H + \mathbf{R}_k, \tag{2.50}$$
$$\mathbf{K}_k = \mathbf{P}_{p,k} \mathbf{H}_k^H \mathbf{S}_k^{-1}, \tag{2.51}$$
$$\mathbf{m}_{f,k} = \mathbf{m}_{p,k} + \mathbf{K}_k \mathbf{e}_k, \tag{2.52}$$
$$\mathbf{P}_{f,k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{p,k}. \tag{2.53}$$

The forward recursion over $\mathbb{T}_N$ is initiated with $\mathbf{m}_{f,0}$ and $\mathbf{P}_{f,0}$. A detailed proof is given in proof A.4.7.

The terms within the KF in algorithm 2.3.2 are:

- Previous *filter* mean, $\mathbf{m}_{f,k-1} \in \mathbb{C}^n$ - The mean of the filtering density for time step $k-1$.
- Previous *filter* covariance, $\mathbf{P}_{f,k-1} \in \mathbb{C}^{n \times n}$ - The covariance of filtering density associated with $\mathbf{m}_{f,k-1}$.
- Previous *evolution* matrix, $\mathbf{F}_{k-1} \in \mathbb{C}^{n \times n}$ - The linear transform that describes the movement from state $\mathbf{x}_{k-1}$ to $\mathbf{x}_k$.
- Previous *process noise* covariance, $\mathbf{Q}_{k-1} \in \mathbb{C}^{n \times n}$ - The driving noise for the linear evolution model, particularly for steps $k-1$ to k.
- Current *predictor* mean, $\mathbf{m}_{p,k} \in \mathbb{C}^n$ - The mean of the prediction density derived by propagating the previous filter mean $\mathbf{m}_{f,k-1}$ through the linear evolution model.
- Current *predictor* covariance, $\mathbf{P}_{p,k} \in \mathbb{C}^{n \times n}$ - The covariance of prediction density associated with $\mathbf{m}_{p,k}$.
- Current *measurement* matrix, $\mathbf{H}_k \in \mathbb{C}^{m \times n}$ - The linear transform that describes the mapping of the current state $\mathbf{x}_k$ to measurement $\mathbf{y}_k$ for time step k.
- Current *predictor residual* or *residual*, $\mathbf{e}_k \in \mathbb{C}^m$ - The error between the predictor mean $\mathbf{m}_{p,k}$ and the current mean $\mathbf{y}_k$ calculated in the measurement's state space. It is also known as the "predictor error" or simply "error".
- Current *measurement noise* covariance, $\mathbf{R}_k \in \mathbb{C}^{m \times m}$ - Noise present during observation of $\mathbf{y}_k$ according to the linear measurement model, particularly for step k.
- Current *predictor residual* or *residual* covariance, $\mathbf{S}_k \in \mathbb{C}^{m \times m}$ - The covariance matrix associated with the residual $\mathbf{e}_k$.
- Current *Kalman gain*, $\mathbf{K}_k \in \mathbb{C}^{n \times m}$ - The update term which balances the confidence in the predictor terms $\mathbf{m}_{p,k}$ and $\mathbf{P}_{p,k}$ versus the confidence in the measurement information to correct them appropriately to form the filter terms.

- Current *filter* mean, $\mathbf{m}_{f,k} \in \mathbb{C}^n$ - The mean of the filtering density for time step k calculated by adjusting the predictor mean $\mathbf{m}_{p,k}$ to best explain the current measurement $\mathbf{y}_k$ using the Kalman gain $\mathbf{K}_k$.
- Current *filter* covariance, $\mathbf{P}_{f,k} \in \mathbb{C}^{n \times n}$ - The covariance of filtering density associated with $\mathbf{m}_{f,k}$.

In summary, the KF algorithm is broken down into two distinct steps: the prediction step calculates the predictor mean and covariance by pushing the previous filtering results forward in time through the linear evolution model. The update step then evaluates the accuracy of predictor results given the newest measurement while considering the mean and noise uncertainties. Then, it appropriately adjusts these results to produce the filter mean and covariance. The KF is computationally efficient since it can determine the next state's Gaussian density parameters as new measurements come in without recomputing previous density parameters. Furthermore, it does not require holding the entire time frame in computing memory to attain the necessary parameters. The KF on its own completely solves linear signals with Gaussian noise for real-time problems, i.e. is optimal for linear signals for given assumptions on the problem. However, the KF would be suboptimal if applied directly to calibration.

Firstly, as mentioned in §2.2.6, the RIME is a non-linear expression and, therefore, the KF's linearity premises breakdown. In addition, the concept of impropriety has not been addressed, which requires adjusting state-space models and considering how the pseudo-covariance would be calculated alongside the other Gaussian parameters. Hence, a few mathematical and numerical extensions are required to rectify this. This is done in §2.4. Secondly, calibration is an off-line statistical estimation problem, i.e. a smoothing problem. Therefore, all state densities or estimates within the time frame should consider all measurements. This is essential when capturing the signal's behaviour over the entire context, leading to better-quality calibration solutions. Due to the KF's design, the only state probability density that satisfies this requirement is the last filter terms generated. This is because the time frame is finite, which means the resulting filtering and smoothing densities for the last state $\mathbf{x}_N$ are equivalent. Hence, to have a practical calibration framework that uses the KF, older filter solutions should be "smoothed" by incorporating missing measurement information from the future. This leads to the recursive form of the *Bayesian smoothing solution* and subsequently the KS.

2.3.3 Bayesian Smoothing Solution

Although the KF in algorithm 2.3.2 effectively addresses filtering problems, it is limited when applied to smoothing problems. The Bayesian filtering solution discussed in §2.3.1 only incorporates current and past measurements, leading to suboptimal solutions when considering future measurements in its scope. This limitation worsens the older the filtering solution is from the present time step [24]. The natural approach would be to derive a separate smoother algorithm using a similarly derived smoothing solution that incorporates all measurements in the time frame. For Bayesian inference, this is called the *Bayesian smoothing solution* and

is associated with *any* batch-like or recursive techniques that generate the aforementioned smoothed solutions. Going forward, "Bayesian smoothing solution" will be associated with the recursive Bayesian approach to solving the smoothing problem as its other forms are not discussed within this research. To avoid creating an independent smoothing algorithm, Rauch, Striebel, and Tung [25] provided a means to leverage the KF to formulate the needed smoothing algorithm. Rauch, Striebel, and Tung [25] introduced the *Rauch-Tung-Striebel Smoother* (RTS Smoother) (*Kalman Smoother* (KS)) that generates a smoothing solution by propagating future measurement information in the KF's results backwards in time to re-inform earlier filtering terms. Särkkä [24] provides a generalisation of the same concepts and steps to form a Bayesian smoothing solution that extends the Bayesian filtering solution. Like the KF, the KS is seen as the Bayesian smoothing solution under linear Gaussian conditions designed to be coupled with the KF.

A key advantage of this formulation is its computational efficiency over other smoothing algorithms, as it avoids recalculating densities and estimates related to the measurement model already computed by the Bayesian filtering solution. This mitigates adding computational complexity to the solutions while producing more informed solutions appropriate for off-line problems. Again, the Bayesian approach is formulating the problem around an appropriate posterior density. In a given time k in the time frame $\mathbb{T}_N$, the Bayesian smoothing solution constructs, for each state $\mathbf{x}_k$, a *smoothing* density given all measurements $\mathbf{y}_{1:N}$ written as $p(\mathbf{x}_k \mid \mathbf{y}_{1:N})$. This construction involves the future prediction density $p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:k})$, the future smoothing density $p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:N})$, and the current filtering density $p(\mathbf{x}_k \mid \mathbf{y}_{1:k})$, as shown within theorem 2.3.3.

Theorem 2.3.3 (Bayesian Smoothing Solution). *Särkkä [24].* After completing the forward-recursion of the Bayesian filtering solution in theorem 2.3.1, the Bayesian smoothing solution recursively computes the *smoothing* density $p(\mathbf{x}_k \mid \mathbf{y}_{1:N})$ backwards in time at each step $k \in \mathbb{T}_N$ (except $k = 0$), as

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:N}) = p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) \cdot \int \frac{p(\mathbf{x}_{k+1} \mid \mathbf{x}_k) p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:N})}{p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:k})} \, \mathrm{d}\mathbf{x}_{k+1}, \tag{2.54}$$

where the prediction density $p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:k})$ and filtering density $p(\mathbf{x}_k \mid \mathbf{y}_{1:k})$ is from the Bayesian filtering solution (see theorem 2.3.1). The recursion is initiated with the last filtering density $p(\mathbf{x}_N \mid \mathbf{y}_{1:N})$. A detailed proof is given in proof A.4.8.

The above is the recursive Bayesian smoothing solution in terms of prediction, filtering and smoothing densities. The breakdown of the components in eq. (2.54) is given below:

- Current *filtering* density, $p(\mathbf{x}_k \mid \mathbf{y}_{1:k})$ - The filtering density of the current state $\mathbf{x}_k$ determined by the Bayesian filtering solution for time step k.
- Future *predictor* density, $p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:k})$ - The prediction density of the future state $\mathbf{x}_{k+1}$ determined by the Bayesian filtering solution for time step k.

- Future *evolution* density, $p(\mathbf{x}_{k+1} \mid \mathbf{x}_k)$ - The evolution density that describes the evolution of the current state $\mathbf{x}_k$ to the future state $\mathbf{x}_{k+1}$.
- Future *smoothing* density, $p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:N})$ - The smoothing solution density calculated for future state $\mathbf{x}_{k+1}$.
- Current *smoothing* density, $p(\mathbf{x}_k \mid \mathbf{y}_{1:N})$ - The smoothing density of the current state $\mathbf{x}_k$.

The Bayesian smoothing solution is predicated on the assumption that the smoothing solution of the future state, i.e. $\mathbf{x}_{k+1}$, is known. At time step N, i.e. the last time step, the filtering density is equivalent to the smoothing density. Therefore, the backwards recursion is initialised with the last filtering density and works backwards in time, incorporating the future measurement information into a current smoothing density using future smoothing solutions coupled with the current filtering solution. Along with exploiting the Bayesian filtering solution, the Bayesian smoothing solution can also leverage the prediction densities calculated in §2.3.1 and thus forms a commonality that can be leveraged computationally. This is the predominant reason for emphasising the prediction density within this text. These points will be discussed further in chapter 3. The Bayesian filtering and smoothing solutions form the *Bayesian Filtering and Smoothing* (BFS) framework for solving statistical estimation problems using recursive Bayesian techniques. It is stressed here that the Bayesian smoothing solution is not limited to this form, but for the requirements of deriving the KS, it is presented as such. This approach is also probabilistically generalised for the same reasons given in §2.3.1 for the Bayesian filtering solution.

2.3.4 Kalman Smoother

As mentioned before, the KS was designed to work given the results of the KF under the assumption that the stochastic processes are linear and their noise is modelled as Gaussian (see eqs. (2.43) and (2.44)). As a result, the required smoothing density for state $\mathbf{x}_k$ in the stochastic process $\mathbf{x}_{1:N}$ given all measurements $\mathbf{y}_{1:k}$ can be expressed as

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:N}) = \mathcal{CN}_n(\mathbf{m}_{s,k}, \mathbf{P}_{s,k}), \tag{2.55}$$

where $\mathbf{m}_{s,k}$ is the mean and $\mathbf{P}_{s,k}$ is the covariance. The subscript s denotes "smoother" to refer to terms related to the smoothing density. As discussed in §2.3.2, the goal is to retrieve the Gaussian density parameters $\mathbf{m}_{s,k}$ and $\mathbf{P}_{s,k}$ to describe the smoothing density. However, this is now done using the Bayesian smoothing solution in theorem 2.3.3 as a guide. Under these conditions, the Bayesian smoothing solution reformulates as the RTS Smoother (KS) that naturally couples with the KF to form the *Kalman Filter and Smoother* (KFS) framework.

Algorithm 2.3.4 (Rauch-Tung-Striebel (Kalman) Smoother). *Särkkä [24].* Run the *Kalman Filter* (KF) to acquire the prediction and filtering density parameters (see algorithm 2.3.2).

The *Kalman Smoother* (KS) acquires the Gaussian densities for each state in $\mathbf{x}_{1:N}$ by recursively refining the current filtering densities with residual information between the future prediction and smoothing densities propagated backwards in time. The result is the smoother means $\mathbf{m}_{s,1:N}$ and covariances $\mathbf{P}_{s,1:N}$ and each is calculated at each $k \in \mathbb{T}_N$ as follows:

3.) **Smoothing Step**:

$$\mathbf{W}_k = \mathbf{P}_{f,k}\mathbf{F}_k^H \mathbf{P}_{p,k+1}^{-1}, \tag{2.56}$$

$$\mathbf{m}_{s,k} = \mathbf{m}_{f,k} + \mathbf{W}_k(\mathbf{m}_{s,k+1} - \mathbf{m}_{p,k+1}), \tag{2.57}$$

$$\mathbf{P}_{s,k} = \mathbf{P}_{f,k} + \mathbf{W}_k(\mathbf{P}_{s,k+1} - \mathbf{P}_{p,k+1})\mathbf{W}_k^H. \tag{2.58}$$

The backward recursion over $\mathbb{T}_N$ is initiated with $\mathbf{m}_{f,N}$ and $\mathbf{P}_{f,N}$. A detailed proof is given in proof A.4.9.

The prediction step within algorithm 2.3.2 can be used in algorithm 2.3.4 to produce the predictor terms if not kept after KF run. The final step in the KFS framework is the *smoothing step* that smooths the updated filter terms. The terms found within the KS in algorithm 2.3.4 are:

- Current *filter* mean, $\mathbf{m}_{f,k} \in \mathbb{C}^n$ - The mean of the filtering density for time step k.
- Current *filter* covariance, $\mathbf{P}_{f,k} \in \mathbb{C}^{n\times n}$ - The covariance of the filtering density associated with $\mathbf{m}_{f,k}$.
- Current *evolution* matrix, $\mathbf{F}_k \in \mathbb{C}^{n\times n}$ - The linear transform that describes the movement from state $\mathbf{x}_k$ to $\mathbf{x}_{k+1}$.
- Future *predictor* mean, $\mathbf{m}_{p,k+1} \in \mathbb{C}^n$ - The mean of the prediction density for time step $k+1$.
- Future *predictor* covariance, $\mathbf{P}_{p,k+1} \in \mathbb{C}^{n\times n}$ - The covariance of the prediction density associated with $\mathbf{m}_{p,k+1}$.
- Current *Smoothing gain*, $\mathbf{W}_k \in \mathbb{C}^{n\times n}$ - A term similar to the Kalman gain in eq. (2.51). It balances the confidence in the filtering density against the prediction density that describes how information best transitions from the future state $\mathbf{x}_{k+1}$ to the current state $\mathbf{x}_k$.
- Future *smoother* mean, $\mathbf{m}_{s,k+1} \in \mathbb{C}^n$ - The mean of the smoothing density for time step $k+1$.
- Future *smoother* covariance, $\mathbf{P}_{s,k+1} \in \mathbb{C}^{n\times n}$ - The covariance of the smoothing density associated with $\mathbf{m}_{s,k+1}$.

- Current *smoother* mean, $\mathbf{m}_{s,k} \in \mathbb{C}^n$ - The mean of the smoothing density for time step k that is calculated by updating the filter mean $\mathbf{m}_{f,k}$ in light of the future smoother mean $\mathbf{m}_{s,k+1}$.
- Current *smoother* covariance, $\mathbf{P}_{s,k} \in \mathbb{C}^{n \times n}$ - The covariance of the smoothing density associated with $\mathbf{m}_{s,k}$.

The output densities of the KS solve the linear Gaussian smoothing problem constrained by the prior information of the model in terms of parameters and noise. Therefore, by running the KS once the KF is complete, i.e. a KFS run, this recursive Bayesian approach to inference and estimation is extended to include off-line problems like calibration. Given the noise and state-space model assumptions, the KFS solves the exact solution for these signal estimation problems. Consequently, the KFS is only beneficial if the correct priors are chosen. This is especially relevant within calibration where choosing priors is vital for improving calibration solution quality and enhancing SNR. Within the BFS context, there exists recursive methods that can be adapted to provide such capabilities to the KFS and are further discussed within §2.4.3. To use the KFS within calibration, a few extensions are required to the algorithms to accommodate for non-linearity and impropriety of the complex signals of interest.

2.4 Extensions to KFS

The KFS presented in §2.3 is a perfect choice in systems where the governing evolution and measurement state-space models are linear, signals are proper, and the corresponding stochastic processes are Gaussian processes. However, for many real-world scenarios, these assumptions are not enough for some signals. In the case of calibration, various types of estimation problems can have signals that range from linear to non-linear, proper to improper, Gaussian to non-Gaussian, and small to large number of parameters to solve. Luckily, there exist numerous techniques and theoretical extensions to the KFS as a means to address these issues. This section provides the extensions required to derive KalCal for the calibration problem to be described in chapter 3.

2.4.1 Wirtinger Calculus

The content of this section extends the linear approximation process that will be defined in §2.4.2 for complex functions. Whether the model is augmented or widely linear, both encounter the same issue regarding non-linear extensions using linear approximation techniques. That is, traditional calculus can only assist if the state functions are holomorphic, i.e. complex differentiable, and break down if they are non-holomorphic [33], [63].[23] This can become an issue within signal estimation when considering improper complex signals. Specific signals encode information between the real and imaginary axes that holomorphic functions cannot capture, and therefore, statistical estimation is often limited to proper complex signals. This incomplete statistical information on signals can potentially lead to suboptimal estimation solutions, e.g. less accurate calibration solutions. Smirnov and Tasse [38] avoided the

[23] Specifically, the "Cauchy-Riemann Conditions" are satisfied for the given function [63].

problem within calibration by describing the calculus surrounding the RIME using *Wirtinger calculus*, also known as "$\mathbb{CR}$-calculus" according to Kreutz-Delgado [63]. Specifically, for any complex-valued function, Wirtinger calculus considers the complex variable and its conjugate as individual arguments of this function. The result is that the mathematics now resembles partial differential equations, where the second differential is based on the conjugate term. These complex differential operators are defined as

$$\frac{\partial}{\partial \mathbf{x}} = \frac{1}{2}\left(\frac{\partial}{\partial \mathbf{u}} - i\frac{\partial}{\partial \mathbf{v}}\right) \quad \text{and} \quad \frac{\partial}{\partial \mathbf{x}^*} = \frac{1}{2}\left(\frac{\partial}{\partial \mathbf{u}} + i\frac{\partial}{\partial \mathbf{v}}\right), \tag{2.59}$$

where $\mathbf{x} = \mathbf{u} + i\mathbf{v} \in \mathbb{C}^n$ is a complex vector with conjugate $\mathbf{x}^* = \mathbf{u} - i\mathbf{v} \in \mathbb{C}^n$, given that $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$ and $n \in \mathbb{N}$. These are known as the *Wirtinger derivatives*, and their properties are defined in lemma A.2.4 [33], [63].

The benefit of Wirtinger calculus is that it is invariant to holomorphy.[24][25] That is, it can be applied to holomorphic and non-holomorphic functions. A consequence of this definition is that a holomorphic function is zero when differentiated by the conjugate term and non-zero for non-holomorphic functions. In addition, the Cauchy-Riemann equations (see definition A.2.2) for a function to be holomorphic can be re-expressed using the Wirtinger derivatives. Let $\mathbf{f}(\mathbf{z})$ be a holomorphic complex function, i.e. it satisfies the Cauchy-Riemann equations, then the following is true:

$$\frac{\mathrm{d}\mathbf{f}}{\mathrm{d}\mathbf{z}^*} = \mathbf{0}. \tag{2.60}$$

This is labelled as the *Wirtinger-Cauchy-Riemann equations* for holomorphy and is logically equivalent to the Cauchy-Riemann equations [33]. A complete definition is provided in definition A.2.3.

The intuition of Wirtinger calculus can be seen using the Wirtinger derivatives: $\partial/\partial\mathbf{z}$ provides the information contained in the holomorphic component of the function, and $\partial/\partial\mathbf{z}^*$ provides the information contained in the non-holomorphic part [33]. Therefore, if the function is holomorphic, then $\partial/\partial\mathbf{z}^*$ would provide no information, i.e. $\partial/\partial\mathbf{z}^* = 0$. If the function is non-holomorphic, t, then $\partial/\partial\mathbf{z}^*$ would provide some information, i.e. $\partial/\partial\mathbf{z}^* \neq 0$. This is reflected by the Wirtinger-Cauchy-Riemann equations in eq. (2.60). Wirtinger calculus perfectly fits the theory of widely linear models within complex statistics. Both the complex vector and its conjugate are taken into account as distinct arguments, and the resulting complex vector can be seen as a combination of the complex inputs for a multi-variable function [56], [57]. This relationship will be more evident in defining the linear approximation process for complex functions.

[24] *Holomorphism*: A complex function is *holomorphic* if it satisfies the Cauchy-Riemann equations in definition A.2.2. If not, then it is considered *non-holomorphic* [33].

[25] Holomorphism can be seen as a stronger description of complex differentiability and possibly a subset of "Wirtinger" differentiability. Traditional laws of differentiability still constrain the Wirtinger derivatives but are not as restricted by holomorphism [63].

2.4.2 Non-linear signals

When the signal within an estimation problem exhibits non-linearity, the functions governing its generation or state-transition functions become non-linear. This is particularly relevant to the evolution $\mathbf{f}$ and measurement $\mathbf{h}$ functions (see eqs. (2.27) and (2.28), respectively) where either one or both are non-linear. The linearity condition is core to the KFS to ensure probability densities derived throughout remain Gaussian. If these state-transition functions are indeed non-linear, then conditional Gaussian lemmas no longer apply (see lemmas A.4.2 and A.4.3) and several steps in deriving the KFS become invalid (see eqs. (A.49), (A.57), (A.68) and (A.78)). Consequently, the KFS will not work for non-linear signals and solutions are inherently suboptimal [24], [50]. One approach to overcome this obstacle is via the *unscented transform* used by the *Unscented Kalman Filter and Smoother* (UKFS). To summarise, the desired probability density is computed by projecting several known samples, called *sigma points*, from a state with a known Gaussian density through the non-linear state-transition function. The objective is to analyse how non-linearity affects these sigma points, thereby determining the parameters of the desired probability density. Simply stated, the UKFS solves the non-linearity issue by approximating the desired density while ensuring the state-transition functions remain intact. Compared to other non-linear flavours of KFS, the UKFS is often better at handling non-linearity, especially when it is severe, i.e. high-variability in time and signal state. However, it can also be computationally expensive since it requires calculating several sigma points and applying the state-transition function to them. The choice depends on the requirements of the estimation problem [24], [50]. The UKF is used by [11] to account for the non-linearity in the RIME with known success. For this research, a different approach to this obstacle is employed. For more information on the UKFS, see Tasse [11] and Särkkä [24], [50].

A standard approach within calibration that will be used within this research is *linearisation* [37], [38]. This technique, crucial for simplifying the complex non-linear interactions in signal processing, involves approximating a non-linear function as a linear one near a specific input value. Therefore, the linear relationship can be maintained for the KFS by linearising state-transition functions at particular points in time during its derivation. This form of the KFS is called the *Extended Kalman Filter and Smoother* (EKFS) and is a standard estimation framework for tackling the majority of non-linear estimation problems [24], [57]. The performance of the EKFS at estimation and inference depends on the accuracy of this linearisation. If the function varies rapidly across time and value, then the EKFS does not perform as well since the approximation might not be as accurate. In this scenario, the UKFS would be more appropriate because it does not suffer from linearisation errors as much. However, the EKFS is more than appropriate for most cases and is often easier to compute. This is especially true for calibration. The computations involved in this approximation are common within calibration software, e.g. QuartiCal; therefore, the EKFS can be easily integrated into existing software systems. Hence, the EKFS is chosen for the design of KalCal.

A common method for linearising a non-linear function is expanding it via a *Taylor series expansion* to the first-order, i.e. up until the first derivative term, for a specific point in the function's domain. The result is a linear approximation of the non-linear function at that

given point and is completely described in definition A.3.1. Since linearisation requires the calculation of derivatives, the methods are also extended to handle complex-valued functions. As discussed in §2.4.1, complex differentiability can be handled safely by utilising Wirtinger derivatives (eq. (2.59)) in place of traditional complex derivatives. As a result, the Taylor series expansion now contains two sub-series of derivatives for each Wirtinger derivative, i.e. a linear approximation for the function argument and another for its conjugate. This "Wirtinger-based" linearisation can be written for some non-linear function $\mathbf{f} : \mathbb{C}^n \to \mathbb{C}^m$ of $\mathbf{x} \in \mathbb{C}^n$ around $\mathbf{x} = \mathbf{a}$ as follows:

$$\mathbf{f}(\mathbf{x}) \simeq \mathbf{f}(\mathbf{a}) + \mathbf{F}(\mathbf{x} - \mathbf{a}) + \widetilde{\mathbf{F}}(\mathbf{x}^* - \mathbf{a}^*), \tag{2.61}$$

where $\mathbf{F} \in \mathbb{C}^{m \times n}$ is the common *Jacobian* evaluated at $\mathbf{x} = a$ and written as

$$\begin{aligned} \mathbf{F} &= \frac{\mathrm{d}\mathbf{f}}{\mathrm{d}\mathbf{x}}(\mathbf{a}), \\ &= \left[\left[\frac{\partial f_i}{\partial x_j}(a_j) \right]_{ij} \middle| \, (f_i \in \mathbf{f})(x_j \in \mathbf{x})(a_j \in \mathbf{a}) \right], \\ &= \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(a_1) & \cdots & \frac{\partial f_1}{\partial x_n}(a_n) \\ \vdots & \ldots & \vdots \\ \frac{\partial f_m}{\partial x_1}(a_1) & \cdots & \frac{\partial f_m}{\partial x_n}(a_n) \end{bmatrix}, \end{aligned} \tag{2.62}$$

and $\widetilde{\mathbf{F}} \in \mathbb{C}^{m \times n}$ is the *Wirtinger Jacobian* evaluated at $\mathbf{a}$. It is written as

$$\begin{aligned} \widetilde{\mathbf{F}} &= \frac{\mathrm{d}\mathbf{f}}{\mathrm{d}\mathbf{x}^*}(\mathbf{a}), \\ &= \left[\left[\frac{\partial f_i}{\partial x_j^*}(a_j) \right]_{ij} \middle| \, (f_i \in \mathbf{f})(x_j \in \mathbf{x})(a_j \in \mathbf{a}) \right], \\ &= \begin{bmatrix} \frac{\partial f_1}{\partial x_1^*}(a_1) & \cdots & \frac{\partial f_1}{\partial x_n^*}(a_n) \\ \vdots & \ldots & \vdots \\ \frac{\partial f_m}{\partial x_1^*}(a_1) & \cdots & \frac{\partial f_m}{\partial x_n^*}(a_n) \end{bmatrix}. \end{aligned} \tag{2.63}$$

Equality is achieved in eq. (2.61) if $\mathbf{f}$ is linear in both $\mathbf{x}$ and $\mathbf{x}^*$, i.e. when $\mathbf{f}$ is widely linear (see §2.2.3) [56]. Equation (2.61) reduces to a traditional first-order Taylor expansion if $\mathbf{f}$ is holomorphic. Furthermore, the expression in eq. (2.61) is inherently a widely-linear model, and for this reason, it is appropriate to denote this Wirtinger-based linear approximation as a *widely-linear approximation* [33], [38], [56], [57], [63]. To avoid confusion with similarly named processes within the BFS subject field, it will simply be referred to as *augmented linearisation*.[26]

[26] Widely-linear approximation is a known approach utilised within calibration (see Smirnov and Tasse [38] and Tasse [64]); however, it is not given this name since the concept of widely-linear models is not explicitly

Referring back to calibration, the traditional goal is to find a MLE given a certain Gaussian density using iterative techniques like the Gauss-Newton and Levenberg-Marquardt (see §2.2.6). This is a point estimate and, as such, does not require further discussion to apply linearisation. Consequently, the non-linear measurement functions can be directly approximated via augmented linearisation in eq. (2.61) through minimising the NLL loss function [37]. However, the KFS aims to describe the entire density projected through the non-linear function and not simply a point estimate, similar to the UKFS. As a result, the approximation cannot be directly incorporated like the ML non-linear solvers. Therefore, further considerations of linearisation are required for the KFS. Moreover, these considerations will be based on general complex Gaussian densities instead of proper ones. This is done because complex-plane statistics require further details and steps than real-plane statistics, and the general complex Gaussian will demonstrate this. First, let $\mathbf{x}$ be governed by a general complex Gaussian, i.e. $\mathbf{x} \sim \mathcal{CN}_n(\mathbf{m}, \mathbf{P}, \widetilde{\mathbf{P}})$ where $\mathbf{m} = \mathrm{E}[\mathbf{x}] \in \mathbb{C}^n$ is the mean, $\mathbf{P} = \mathrm{Cov}[\mathbf{x}, \mathbf{x}] \in \mathbb{C}^{n \times n}$ is the covariance, and $\widetilde{\mathbf{P}} = \mathrm{Cov}\left[\mathbf{x}, \mathbf{x}^*\right] \in \mathbb{C}^{n \times n}$ is the pseudo-covariance. To achieve the best linear approximation of $\mathbf{f}(\mathbf{x})$, it should be expanded around its most likely value, the mean $\mathbf{m}$. This is simply substituting $\mathbf{a}$ with $\mathbf{m}$ in eq. (2.61). Similar to the real-plane approach by Särkkä [24], let $\mathbf{x}$ be re-expressed as

$$\mathbf{x} = \mathbf{m} + \delta\mathbf{x}, \tag{2.64}$$

where $\delta\mathbf{x} \sim \mathcal{CN}_n(\mathbf{0}, \mathbf{P}, \widetilde{\mathbf{P}})$ is the (improper) deviation of $\mathbf{x}$ from the mean $\mathbf{m}$ in the complex plane. Equation (2.64) is distinct from the real case because it incorporates additional deviation information according to the pseudo-covariance $\widetilde{\mathbf{P}}$. Substituting eq. (2.64) into the previous linearisation and simplifying results in

$$\mathbf{f}(\mathbf{x}) = \mathbf{f}(\mathbf{m} + \delta\mathbf{x}) \simeq \mathbf{f}(\mathbf{m}) + \mathbf{F}\delta\mathbf{x} + \widetilde{\mathbf{F}}\delta\mathbf{x}^*, \tag{2.65}$$

where $\mathbf{F}$ and $\widetilde{\mathbf{F}}$ are evaluated with respect to $\mathbf{m}$. This probabilistic form of linearisation projects the known mean, deviation and conjugate deviation into the desired state space using the augmented-linearised transforms of $\mathbf{F}$ and $\widetilde{\mathbf{F}}$. Equation (2.65) is quite verbose, but it demonstrates the importance and nuance of utilising probability models in the complex plane. Now, using eq. (2.65), the desired Gaussian density can be approximated by determining its various parameters via this expression [24], [57]. By applying the expectation and covariance operators on $\mathbf{f}(\mathbf{x})$ as approximated in eq. (2.65), the mean is calculated as

$$\mathrm{E}[\mathbf{f}(\mathbf{x})] \simeq \mathbf{f}(\mathbf{m}). \tag{2.66}$$

Additionally, the covariance is calculated as

$$\mathrm{Cov}[\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x})] \simeq \mathbf{F}\mathbf{P}\mathbf{F}^H + \widetilde{\mathbf{F}}\widetilde{\mathbf{P}}^*\mathbf{F}^H + \mathbf{F}\widetilde{\mathbf{P}}\widetilde{\mathbf{F}}^H + \widetilde{\mathbf{F}}\mathbf{P}^*\widetilde{\mathbf{F}}^H, \tag{2.67}$$

and the pseudo-covariance is calculated as

used, despite the equivalent result. At the time of writing, this text is the first association of widely-linear models with this method within this context.

$$\text{Cov}\left[\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x})^*\right] \simeq \mathbf{F}\mathbf{P}\mathbf{F}^T + \widetilde{\mathbf{F}}\widetilde{\mathbf{P}}^*\mathbf{F}^T + \mathbf{F}\widetilde{\mathbf{P}}\widetilde{\mathbf{F}}^T + \widetilde{\mathbf{F}}\mathbf{P}^*\widetilde{\mathbf{F}}^T. \tag{2.68}$$

The above can be used to approximate the desired general Gaussian density. If $\mathbf{x}$ were a proper complex vector, only the terms including the $\widetilde{\mathbf{P}}$ would disappear. Note, if $\mathbf{x}$ is indeed proper, it does not mean the desired general Gaussian density will be because eq. (2.68) would still be non-zero. Incorporating these steps during the derivation of the KFS where a non-linear state-transition appears, regardless of the impropriety of $\mathbf{x}$, allows for approximating the necessary Gaussian, i.e. the linear Gaussian conditions are approximated. The complete derivation is discussed by Särkkä [24], Dini and Mandic [56], and Dang and Scharf [57].

This discussion demonstrates why the widely-linear approach is excellent for defining complex-valued relationships but is cumbersome and verbose when used to derive solutions. This verbosity only escalates when deriving the full EKFS (see algorithm 2 in [57]). For this reason, the more standard approach would be to switch to the augmented model formalism after defining the widely-linear relationships since it mimics real-plane linearisation (see algorithm 2 in [56]). However, the above content is vital when discussing approximations within the KalCal framework in chapter 3; thus, it is detailed here for convenience. For now, the EKFS derived using the widely-linear formalism will be called the *Widely-Linear Extended Kalman Filter and Smoother* (WEKFS), and the one derived using the augmented formalism will be called the *Augmented Extended Kalman Filter and Smoother* (AEKFS). The WEKFS and AEKFS are mathematically equivalent algorithms with different representations of complex-valued states and covariances and the choice between the two depends on the specific requirements and constraints of the estimation problem being considered. The WEKFS naturally deals with dimension sizes that are half that of AEKFS; however, this does not guarantee computational efficiency. The WEKFS contains far more matrix operations than the AEKFS. As a result, with a suitably large state size, the widely-linear form may have an equivalent or even higher computational complexity than the AEKFS. This is discussed in more detail in chapter 3. This completes all theoretical extensions needed to allow for the EKFS to be applied to calibration.

2.4.3 Energy function

In the §1.3.3, it was noted that utilising the KFS for calibration would be more problematic to use than NLL methods coupled with solution intervals if the issue of parameter tuning was not addressed. To re-iterate, the KFS requires several priors to perform the necessary computation steps. These include initial filter mean and covariance estimates, assumptions on state-space model noise or state-transition functions. Consequently, the KFS requires far more tuning than traditional solvers used in calibration and the solution intervals technique. Moreover, this research aimed to address an issue with solution intervals: input parameter optimisation. Thus, to make the KFS a viable contender against the NLL method coupled with solution intervals, a means of estimating appropriate priors should be established alongside the resulting calibration algorithm. The energy function, as discussed in §1.3.3, is what

will be investigated as a potential solution to this issue. This section will demonstrate how it is derived and how it can be used to determine the KFS's priors.

Following the discussion provided by Särkkä [24], the performance of a Bayesian-derived model with respect to the model parameter choices. It is derived by including these unknown parameters in the state-space models such that the probability densities can be described as

$$p(\mathbf{x}_{0:N}, \boldsymbol{\theta} \mid \mathbf{y}_{1:N}) = \frac{p(\mathbf{y}_{1:N} \mid \mathbf{x}_{0:N}, \boldsymbol{\theta}) p(\mathbf{x}_0 \mid \boldsymbol{\theta}) p(\boldsymbol{\theta})}{p(\mathbf{y}_{1:N})}, \tag{2.69}$$

where $\boldsymbol{\theta} \in \mathbb{C}^d$ is a *parameter* vector consisting of $d \in \mathbb{N}$ parameters [24]. Since the states are modelled by the prediction, filtering and smoothing densities, they can be integrated out, and this relationship is reduced to

$$p(\boldsymbol{\theta} \mid \mathbf{y}_{1:N}) \propto p(\mathbf{y}_{1:N} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta}). \tag{2.70}$$

Equation (2.70) describes the parameter density given the data observed. The density $p(\mathbf{y}_{1:N} \mid \boldsymbol{\theta})$ is the evidence density, also known as the *Bayesian evidence*, that describes how well the parameter choices explain the measurements, and $p(\boldsymbol{\theta})$ represents the prior knowledge of the parameters. It can infer the most likely model parameters for the given estimation problem. Referring back to §2.2.6, these model parameters can be estimated through the MAP approach without realising the total parameter density. In fact, it can be determined precisely for linear models and approximately for non-linear models. Therefore, the loss function can be defined by applying the negative logarithm to eq. (2.70) to get a result similar to eq. (2.37). This loss function can be expressed as

$$\varphi_N(\boldsymbol{\theta}) = -\ln p(\mathbf{y}_{1:N} \mid \boldsymbol{\theta}) - \ln p(\boldsymbol{\theta}), \tag{2.71}$$

where φ_N is called the *energy function*. The subscript N indicates that it considers all time steps up to and including N. As a consequence, eq. (2.70) can be re-imagined using φ_N as follows [24]:

$$p(\boldsymbol{\theta} \mid \mathbf{y}_{1:N}) \propto e^{-\varphi_N(\boldsymbol{\theta})}. \tag{2.72}$$

For Bayesian inference, eq. (2.71) can be used to determine any model parameters, regardless of whether the density is Gaussian or not and whether they are defined over the real or complex plane. However, this additional computation step is conducted independently of the state modelling and estimation process. Moreover, with MAP estimation, the evidence density is not constructed. Therefore, the evidence density would need to be calculated even before the parameter estimation could be conducted. Finally, this form of parameter estimation is not available to ML state estimation algorithms since it is not based on Bayesian reasoning. However, under Markov processes, eq. (2.71) can be re-contextualised to a more pragmatic form for BFS methods.

As a result of the recursive definitions of the BFS solutions, a recursive version of eq. (2.72) can be derived. Note for this research, the $\boldsymbol{\theta}$ argument is omitted to reduce clutter. However, the primary parameter being analysed is the evolution model's process noise, σ_Q (see). The

specific details of this choice are provided in §3.3.4. For now, view the energy function as a continuous scalar function that describes how well σ_Q explains the observations seen. Since the energy function is a form of loss function, the value of σ_Q that minimises the energy function would be the MAP estimate for the optimal process noise, i.e. best explains the observed measurements. To make eq. (2.72) applicable for the recursive design of BFS and the KFS framework, the recursive form can be used:

$$\varphi_k = \varphi_{k-1} - \ln p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}, \boldsymbol{\theta}), \tag{2.73}$$

The above is initiated with $\varphi_0 = -\ln p(\boldsymbol{\theta})$ [24]. Compared to the definition in eq. (2.71), the energy function in Equation (2.73) is computed at each time step using the evidence density up till time k and the energy function computed in the previous time step. A prominent difference is that eq. (2.73) can be computed alongside the Bayesian filtering solution at each time step. As a result, there is space at each prediction and update step to compute the energy function value and predict priors through an adaptive priors selection scheme. In addition, the energy function is computed alongside the algorithm, and as a result, when the filter run is complete, so will the computation of the total energy function. However, this is only beneficial if the resulting recursive calculations become tractable enough not to hinder the performance of the filter algorithm. Assuming that it is tractable, the energy function in terms of the KFS algorithms for proper complex signals is

$$\varphi_k = \varphi_{k-1} + \frac{1}{2}\ln\det(\pi\mathbf{S}_k) + \frac{1}{2}\mathbf{e}_k^H\mathbf{S}_k^{-1}\mathbf{e}_k, \tag{2.74}$$

where $\mathbf{e}_k$ is the residual vector in eq. (2.49) and $\mathbf{S}_k$ is the residual covariance in eq. (2.50).[27] For the non-linear extensions, this function can be applied since the AEKFS and WEKFS are approximate forms of the KFS. This is one point where the augmented model formalism is easier to use since if the signal was improper, then the only adjustments to eq. (2.74) is to augment the various terms and change the constant π to 2π. The equivalent expression is highly complex to derive and understand for the widely-linear model formalism since it explicitly accounts for all the relationships arising from improper signals. These recursive energy function calculations could provide the potential for developing feasible parameter optimisation techniques, especially in calibration. This is further discussed in §3.3.4 and analysed against experiment results in §4.2.4.

2.4.4 The Kalman-Woodbury Identity

During the research and development of KalCal, an interesting fact was derived regarding the KF. For reasons detailed in chapter 3, it is essential to include the *Sherman-Morrison-Woodbury Identity* (SMW) to handle the resulting inverse within algorithm 2.3.2. Specifically, this is the inversion of the residual covariance matrix $\mathbf{S}_k$. This step is based on the same one applied by Tasse [11] and further discussed by Smirnov and Tasse [38]. In summary, the goal was to change the context in which the inverse is determined to reduce the computational

[27] For real random vectors, the only change in eq. (2.74) is that the adjoint operators become transposes and the constant π becomes 2π [24].

complexity of this calculation, i.e. inversion in state space instead of measurement space, since the former's dimension is smaller than that of the latter. To begin, for matrices $\mathbf{A} \in \mathbb{C}^{n\times n}$, $\mathbf{B} \in \mathbb{C}^{n\times m}$, $\mathbf{C} \in \mathbb{C}^{m\times m}$ and $\mathbf{D} \in \mathbb{C}^{m\times n}$ where $n, m \in \mathbb{N}$, the SMW [42] is defined as

$$(\mathbf{A}+\mathbf{BCD})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}\left(\mathbf{C}^{-1}+\mathbf{DA}^{-1}\mathbf{B}\right)^{-1}\mathbf{DA}^{-1}. \tag{2.75}$$

In this generalised form, employing this step is only advantageous if m is much smaller than n. This is true within calibration and is further discussed within chapter 3. Instead, this section aims to present the intriguing mathematical result that arises when combining the KF with the SMW and then simplifying the resulting equations. From now on, the time index is omitted in this section for clarity; however, all derivations presented will be valid for all $k \in \mathbb{T}_N$ within the KFS. Also, assume that the prediction step has been computed and the predictor results are available. To start, consider the residual covariance matrix given in eq. (2.50), that is

$$\mathbf{S} = \mathbf{HP}_p\mathbf{H}^H + \mathbf{R}, \tag{2.76}$$

where $\mathbf{H}$ is the measurement matrix, $\mathbf{P}_p$ is the predictor covariance, and $\mathbf{R}$ is the measurement noise covariance. Calculating the Kalman gain $\mathbf{K}$ in eq. (2.51) requires determining the following:

$$\mathbf{S}^{-1} = \left(\mathbf{HP}_p\mathbf{H}^H + \mathbf{R}\right)^{-1}. \tag{2.77}$$

If $\mathbf{A} = \mathbf{R}$, $\mathbf{B} = \mathbf{H}$, $\mathbf{C} = \mathbf{P}_p$ and $\mathbf{D} = \mathbf{H}^H$, then applying the SMW in eq. (2.75) onto eq. (2.77) results in

$$\mathbf{S}^{-1} = \mathbf{R}^{-1} - \mathbf{R}^{-1}\mathbf{H}\left(\mathbf{P}_p^{-1} + \mathbf{H}^H\mathbf{RH}\right)^{-1}\mathbf{H}^H\mathbf{R}^{-1}. \tag{2.78}$$

This is the result that Tasse [11] determined. Next, substitute the result in eq. (2.78) into the Kalman gain in eq. (2.51), and then substitute that result into the filter covariance step in eq. (2.53) to get

$$\mathbf{P}_f = \mathbf{P}_p - \mathbf{P}_p\mathbf{H}^H\mathbf{R}^{-1}\mathbf{HP}_p + \mathbf{P}_p\mathbf{H}^H\mathbf{R}^{-1}\mathbf{H}(\mathbf{P}_p^{-1} + \mathbf{H}^H\mathbf{R}^{-1}\mathbf{H})^{-1}\mathbf{H}^H\mathbf{R}^{-1}\mathbf{HP}_p. \tag{2.79}$$

Now, the above can be simplified. Start by letting and substituting in $\mathbf{U} = \mathbf{H}^H\mathbf{R}^{-1}\mathbf{H}$ to reduce clutter within the expression. Then, apply the following steps to simplify it:

$$
\begin{aligned}
\mathbf{P}_f &= \mathbf{P}_p - \mathbf{P}_p\mathbf{U}\mathbf{P}_p + \mathbf{P}_p\mathbf{U}\left(\mathbf{P}_p^{-1} + \mathbf{U}\right)^{-1}\mathbf{U}\mathbf{P}_p, \\
&= \left(\mathbf{I} - \left(\mathbf{P}_p - \mathbf{P}_p\mathbf{U}\left(\mathbf{P}_p^{-1} + \mathbf{U}\right)^{-1}\right)\mathbf{U}\right)\mathbf{P}_p, \\
&= \left(\mathbf{I} - \left(\mathbf{P}_p\left(\mathbf{P}_p^{-1} + \mathbf{U}\right) - \mathbf{P}_p\mathbf{U}\right)\left(\mathbf{P}_p^{-1} + \mathbf{U}\right)^{-1}\mathbf{U}\right)\mathbf{P}_p, \\
&= \left(\mathbf{I} - \left(\mathbf{P}_p\mathbf{P}_p^{-1} + \mathbf{P}_p\mathbf{U} - \mathbf{P}_p\mathbf{U}\right)\left(\mathbf{P}_p^{-1} + \mathbf{U}\right)^{-1}\mathbf{U}\right)\mathbf{P}_p, \\
&= \left(\mathbf{I} - \left(\mathbf{P}_p^{-1} + \mathbf{U}\right)^{-1}\mathbf{U}\right)\mathbf{P}_p, \\
&= \left(\left(\mathbf{P}_p^{-1} + \mathbf{U}\right)^{-1}\left(\mathbf{P}_p^{-1} + \mathbf{U}\right) - \left(\mathbf{P}_p^{-1} + \mathbf{U}\right)^{-1}\mathbf{U}\right)\mathbf{P}_p, \\
&= \left(\mathbf{P}_p^{-1} + \mathbf{U}\right)^{-1}\left(\mathbf{P}_p^{-1} + \mathbf{U} - \mathbf{U}\right)\mathbf{P}_p, \\
&= \left(\mathbf{P}_p^{-1} + \mathbf{U}\right)^{-1}\mathbf{P}_p^{-1}\mathbf{P}_p, \\
&= \left(\mathbf{P}_p^{-1} + \mathbf{U}\right)^{-1}.
\end{aligned} \tag{2.80}
$$

Substitute back in $\mathbf{U} = \mathbf{H}^H\mathbf{R}\mathbf{H}$ to get the final result

$$
\mathbf{P}_f = \left(\mathbf{P}_p^{-1} + \mathbf{H}^H\mathbf{R}\mathbf{H}\right)^{-1}. \tag{2.81}
$$

This is the filter covariance step given the SMW. It is logically equivalent to the original but now is computed predominantly in the state space instead of the measurement space. The next step is to address the filter mean. Again, using eq. (2.78) within the Kalman gain in eq. (2.51), substitute the resulting expression into eq. (2.52) and expand it to arrive at

$$
\mathbf{m}_f = \mathbf{m}_p + \mathbf{P}_p\mathbf{H}^H\mathbf{R}^{-1}\mathbf{e} - \mathbf{P}_p\mathbf{H}^H\mathbf{R}^{-1}\mathbf{H}\left(\mathbf{P}_p^{-1} + \mathbf{H}^H\mathbf{R}^{-1}\mathbf{H}\right)^{-1}\mathbf{H}^H\mathbf{R}^{-1}\mathbf{e}. \tag{2.82}
$$

Again, the time indices are ignored. Let $\mathbf{U} = \mathbf{H}^H\mathbf{R}\mathbf{H}$ and $\mathbf{z} = \mathbf{H}^H\mathbf{R}^{-1}\mathbf{r}$. Then eq. (2.82) can be rewritten and simplified:

$$\begin{aligned}
\mathbf{m}_f &= \mathbf{m}_p + \mathbf{P}_p\mathbf{z} - \mathbf{P}_p\mathbf{U}\left(\mathbf{P}_p^{-1} + \mathbf{U}\right)^{-1}\mathbf{z}, \\
&= \mathbf{m}_p + \mathbf{P}_p\left(\mathbf{I} - \mathbf{U}\left(\mathbf{P}_p^{-1} + \mathbf{U}\right)^{-1}\right)\mathbf{z}, \\
&= \mathbf{m}_p + \mathbf{P}_p\left(\left(\mathbf{P}_p^{-1} + \mathbf{U}\right)\left(\mathbf{P}_p^{-1} + \mathbf{U}\right)^{-1} - \mathbf{U}\left(\mathbf{P}_p^{-1} + \mathbf{U}\right)^{-1}\right)\mathbf{z}, \\
&= \mathbf{m}_p + \mathbf{P}_p\left(\mathbf{P}_p^{-1} + \mathbf{U} - \mathbf{U}\right)\left(\mathbf{P}_p^{-1} + \mathbf{U}\right)^{-1}\mathbf{z}, \\
&= \mathbf{m}_p + \mathbf{P}_p\mathbf{P}_p^{-1}\left(\mathbf{P}_p^{-1} + \mathbf{U}\right)^{-1}\mathbf{z}, \\
&= \mathbf{m}_p + \left(\mathbf{P}_p^{-1} + \mathbf{U}\right)^{-1}\mathbf{z}. \qquad (2.83)
\end{aligned}$$

Substitute back in the placeholders $\mathbf{U}$ and $\mathbf{z}$, and the result in eq. (2.81) to arrive at the final expression

$$\mathbf{m}_f = \mathbf{m}_p + \mathbf{P}_p\mathbf{H}^H\mathbf{R}^{-1}\mathbf{e}. \qquad (2.84)$$

This is the filter mean step given the SMW. Similar to eq. (2.81), it is logically equivalent to the original and computed in the state space. To summarise these results, the KF rewritten using the results in eqs. (2.81) and (2.84) is presented below.

Algorithm 2.4.1 (Kalman-Woodbury Identity). Given the initial setup of the KF given in algorithm 2.3.2, apply the *Sherman-Morrison-Woodbury Identity* (SMW) to the inversion step in eq. (2.51) and simplify to re-express the KF algorithm as:

1.) **Prediction Step (Original)**:

$$\mathbf{m}_{p,k} = \mathbf{F}_{k-1}\mathbf{m}_{f,k-1}, \qquad (2.85)$$
$$\mathbf{P}_{p,k} = \mathbf{F}_{k-1}\mathbf{P}_{f,k-1}\mathbf{F}_{k-1}^H + \mathbf{Q}_{k-1}, \qquad (2.86)$$

2.) **Update Step (New)**:

$$\mathbf{e}_k = \mathbf{y}_k - \mathbf{H}_k\mathbf{m}_{p,k}, \qquad (2.87)$$
$$\mathbf{P}_{f,k} = \left(\mathbf{P}_{p,k}^{-1} + \mathbf{H}_k^H\mathbf{R}_k^{-1}\mathbf{H}_k\right)^{-1}, \qquad (2.88)$$
$$\mathbf{m}_{f,k} = \mathbf{m}_{p,k} + \mathbf{P}_{f,k}\mathbf{H}_k^H\mathbf{R}_k^{-1}\mathbf{e}_k. \qquad (2.89)$$
$$(2.90)$$

The above is dubbed the *Kalman-Woodbury Identity*. It is mathematically equivalent to the KF but calculated from a different perspective. Notably, most update terms are

not calculated within the measurement space. Instead, they are calculated within the state space. Moreover, the algorithm does not require the Kalman gain, an often computationally expensive term to calculate. Although not directly evaluated, it is believed that the same algorithm can be created during the derivation of the KF by introducing the SMW into eqs. (A.60) to (A.62).

The obvious benefit of this formulation is that it can potentially reduce the computational complexity of the KF if the size of the state space is smaller than the size of the measurement space. Additionally, several terms can be re-used to avoid unnecessary calculations. First, the filter covariance is pre-computed in eq. (2.88) and used to update the filter estimate in eq. (2.89). Second, the KS requires the calculation of the inverse predictor covariance $\mathbf{P}_{p,k}^{-1}$ at each time step to formulate the smoothing gain in eq. (2.56). This term is computed within eq. (2.88) and thus can be stored for the KS to avoid repeating the inversion calculation, reducing its computational complexity. Third, the energy function in eq. (2.71) requires the calculation of $\mathbf{S}_k$ and $\mathbf{S}_k^{-1}$. These calculations are discussed in detail in chapter 3. For now, assume that the goal is only to find $\mathbf{S}_k^{-1}$. The Kalman-Woodbury identity form of the KF does not explicitly calculate this term. However, using eq. (2.78), it can be determined by noting the result in eq. (2.88):

$$\mathbf{S}^{-1} = \mathbf{R}^{-1} - \mathbf{R}^{-1}\mathbf{H}\mathbf{P}_f\mathbf{H}^H\mathbf{R}^{-1}. \tag{2.91}$$

It can be computed using the newly generated filter covariance term. Another benefit of this form is that it provides a more intuitive understanding of how the KF works. It describes precisely how the prior model can regularise the solution by adding the predictor terms within the creation of the filter mean and covariance. The predictor mean is adjusted according to the predictor residual with uncertainty incorporated through the filter covariance, created by combining the uncertainty in state and measurement space. The result is the filter mean. Further research is required to understand its significance altogether. Most importantly, does it pose additional *numerical stability*[28] issues, and how do errors induced by linearisation affect the results? Despite a thorough literature search, the origin of the Kalman-Woodbury identity and the re-expression of the KF with it has yet to be identified. Nevertheless, it is a relatively straightforward task to derive the identity using introductory linear algebra; hence, it is unlikely it is a unique discovery. Moreover, credit is given to Tasse for introducing the SMW into the calibration, and their result is logically equivalent to the final result presented here. The primary outcome from this discussion is that it significantly impacts KFS-based calibration algorithms, where the measurement space's size completely dominates the state space's size. Note this result applies to the *Augmented Kalman Filter* (AKF) and *Augmented Extended Kalman Filter* (AEKF) since the derivation would be the same through the augmented formalism. In addition, it also applies to the *Widely-Linear Kalman Filter* (WKF) and *Widely-Linear Extended Kalman Filter* (WEKF) since the widely-linear formalism is equivalent to the augmented formalism. Further optimisations are discussed

[28] *Numerical stability*: A phenomenon in which the approximation errors resulting from finite arithmetic precision in calculations become so large that it reduces the accuracy of the calculations or, in some cases, makes it impossible to calculate the result [65].

in chapter 3, and additional theoretical exploration is planned for future research. This concludes the theory of KFS and this chapter. The next chapter uses the ideas and concepts presented here to develop the phenomenological calibration framework, KalCal.

Chapter 3

KalCal

With the mathematical foundation in chapter 2, developing and implementing a prototype calibration algorithm using the KFS is now possible. As mentioned in chapter 1, this prototype will be based on the RIME formalism using phenomenological Jones terms to solve for scalar, time-only antenna gains. This approach is called KalCal, which is an abbreviation of ***Kal****man filtering and smoothing in* ***cal****ibration*. KalCal will be presented in two forms: *Diagonally Approximated KalCal* (KalCal-Diag) and *Pure KalCal* (KalCal-Full). The former is an optimised and approximated version that balances accuracy with reducing computational complexity. The latter represents a complete and unaltered form of the calibration algorithm that focuses on accuracy and validates the fundamental theory. These algorithms will be tested alongside QuartiCal in chapter 4 to determine the success of this approach against the solution intervals technique. To begin, the calibration problem for this research must be defined.

3.1 Deriving from the RIME

To construct KalCal, the calibration problem it is aimed to solve needs to be defined such that the appropriate measurement and evolution models can be applied. The following sections detail this setup to develop the KalCal algorithms using the RIME.

3.1.1 Calibration Problem

The KalCal algorithms will be developed around the direction-independent RIME given in eq. (1.8). To re-iterate, the RIME per baseline $(p, q) \in \mathbb{B}$ is

$$\mathbf{V}_{pq} = \mathbf{G}_p \mathbf{X}_{pq} \mathbf{G}_q^H, \tag{3.1}$$

where $\mathbf{X}_{pq} \in \mathbb{C}^{2\times2}$ is the sky coherency matrix (see eq. (1.9) for a complete definition). The objective is to estimate $\mathbf{G}_p, \mathbf{G}_q \in \mathbb{C}^{2\times2}$ to correct the noisy corrupted visibilities $\mathbf{V}_{pq} \in \mathbb{C}^{2\times2}$ to acquire the corrected visibilities $\mathbf{C}_{pq} \in \mathbb{C}^{2\times2}$ via the correction step in eq. (1.10). For this discussion, the correction step is unnecessary since the focus is on estimating the G-Jones terms. Assume that the observations only consider the intensity of the signal and ignore polarisation information. Additionally, the feed configuration is assumed to be linear. This results in a diagonal brightness matrix and two elements along the diagonal. This is often denoted the *diagonal* case, and the resulting calibration problem can be solved by treating

each diagonal entry as an independent scalar estimation problem. The estimation procedure is equivalent for each entry because the only difference between both estimation problems is the observed values and statistics. Furthermore, it is assumed that the observation is restricted to a single channel and no direction axes are considered, i.e. the E-Jones terms are set to be unity. Finally, auto-correlations, formed from individual antennas forming a baseline with themselves, i.e. baselines where $p = q$, are ignored for this calibration problem. Specifically, the baselines considered are defined in baseline index set $\mathbb{B} = \left((p,q) \,\middle|\, (p,q) \in \mathbb{A}^2 : p < q \right)$ for a given antenna index set $\mathbb{A} = (1, ..., N_{\text{ant}})$. Taking all the assumptions into account, the resulting RIME, for $(p,q) \in \mathbb{B}$ and $t \in \mathbb{T}$, where $\mathbb{T} = (1, ..., N_{\text{time}})$, is the frequency- and direction-independent scalar expression

$$v_{pq,t} = g_{p,t} x_{pq,t} g^*_{q,t} + r_{pq,t}, \tag{3.2}$$

which is comprised of the following terms:

- *Predicted* visibility, $x_{pq,t} \in \mathbb{C}$ - The visibility calculated from a given sky model as the prior of the true visibility.
- *Measured* visibility, $v_{pq,t} \in \mathbb{C}$ - The visibility observed at baseline pq at time t.
- *Gain terms*, $g_{p,t}, g_{q,t} \in \mathbb{C}$ - Gains for the antennas indexed using index pairs given by the associated baseline pq.
- *Measurement noise*, $r_{pq,t} \sim \mathcal{CN}(0, \sigma_R^2)$ - The complex white noise added during observation, governed by the variance of the measurement noise σ_R^2. Note that the noise statistics are assumed to be equivalent (or constant) across all baselines and overall time.

Given the calibration problem given in eq. (3.2), the objective of KalCal is then to estimate $\forall (p,q) \in \mathbb{B}$ over $\mathbb{T}$, the gains terms, $\hat{g}_{p,t}$ and $\hat{g}_{q,t}$, and correct the measured visibilities, $v_{pq,t}$, to obtain the corrected visibilities as

$$c_{pq,t} = \frac{v_{pq,t}}{\hat{g}_{p,t} \hat{g}^*_{q,t}}. \tag{3.3}$$

This will be the extent of the calibration problem for this research, and additional investigations will be left for future research. With that, the KalCal algorithms can be derived for this calibration scenario.

3.1.2 Forming a Measurement Equation

To create the resulting KalCal algorithms, an evolution and measurement model is required to describe the various stochastic processes. To start, the gains signal per antenna $p \in \mathbb{A}$ can be represented as the stochastic process $g_{p,1:N_{\text{time}}}$ and is assumed to be a Gauss-Markov process. Furthermore, for all $(p,q) \in \mathbb{B}$, the measured visibilities in the stochastic process $v_{pq,1:N_{\text{time}}}$ are assumed to satisfy the conditional independence of measurements property (see §2.2.5). Following the content of sections 2.2.5 and 2.3, the state-space models must be described to

create the KFS algorithms. This section aims to define the measurement state-space model in eq. (2.28) in the context of the calibration problem in eq. (3.2). The first step is to identify a measurement function. For $(p, q) \in \mathbb{B}$, define the scalar measurement function h_{pq} on both antenna gains as

$$h_{pq}(g_p, g_q) = g_p x_{pq} g_q^*. \tag{3.4}$$

The above will be used to construct the vector form of the required function. The time indices have been omitted for ease of notation since all terms in the measurement model depend on the current time step. Substitute this new function into eq. (3.2) to redefine the calibration problem as

$$v_{pq} = h_{pq}(g_p, g_q) + r_{pq}. \tag{3.5}$$

This is the scalar measurement model that describes the corruption of predicted visibility x_{pq} by antenna gains g_p and g_q over the baseline (p, q). The next step is to use this expression to construct the multi-variable form of the measurement model. The vectors that will be described are defined using the vector-builder notation that is defined in definition A.1.2. Moreover, matrices are described using the matrix-builder notation in remark A.1.3. Define a new vector function $\mathbf{h}$ that allows eq. (3.5) to be written for all baselines $\mathbb{B}$ to create the multi-variable measurement model

$$\mathbf{v} = \mathbf{h}(\mathbf{g}) + \mathbf{r}, \tag{3.6}$$

where each term is defined as:

- *Measurement* vector, $\mathbf{v} \in \mathbb{C}^{N_{\text{bl}}}$ - A vector comprised of noisy corrupted visibilities overall baseline pairs, excluding conjugate equivalents:[1]

$$\mathbf{v} = \left[\, v_{pq} \;\middle|\; (p, q) \in \mathbb{B} \,\right]^T = \left[v_{12}, ..., v_{(N_{\text{ant}}-1)N_{\text{ant}}}\right]^T. \tag{3.7}$$

- *Gains* vector, $\mathbf{g} \in \mathbb{C}^{N_{\text{ant}}}$ - A vector of antenna gains arranged by ascending antenna index:

$$\mathbf{g} = \left[\, g_a \mid a \in \mathbb{A} \,\right]^T = \left[g_1, ..., g_{N_{\text{ant}}}\right]^T. \tag{3.8}$$

- *Measurement* function, $\mathbf{h} : \mathbb{C}^{N_{\text{ant}}} \to \mathbb{C}^{N_{\text{bl}}}$ - Describes the corruption of all predicted visibilities given the gains vector $\mathbf{g}$:

$$\begin{aligned} \mathbf{h}(\mathbf{g}) &= \left[\, h_{pq}(g_p, g_q) \;\middle|\; (p, q) \in \mathbb{B} :\; g_p, g_q \in \mathbf{g} \,\right]^T \\ &= \left[h_{12}(g_1, g_2), ..., h_{(N_{\text{ant}}-1)N_{\text{ant}}}(g_{N_{\text{ant}}-1}, g_{N_{\text{ant}}})\right]^T. \end{aligned} \tag{3.9}$$

[1] The conjugate equivalent of visibility v_{pq} is $v_{qp} = v_{pq}^*$.

- *Measurement noise* vector, $\mathbf{r} \sim \mathcal{CN}_{N_{\text{bl}}}(\mathbf{0}, \mathbf{R})$, $\mathbf{R} \in \mathbb{R}^{N_{\text{bl}} \times N_{\text{bl}}}$ - External white noise added during the observation, governed by the diagonal measurement covariance $\mathbf{R}$ defined as:

$$\mathbf{R} = \text{diag}\left(\left[\, \sigma_R^2 \;\middle|\; (p,q) \in \mathbb{B} \,\right]\right) = \sigma_R^2 \mathbf{I} \tag{3.10}$$

The diagonal operator, diag($\cdot$), lays the vector along the diagonal. See definition A.1.13 for a full definition. $\mathbf{I} \in \mathbb{R}^{N_{\text{bl}} \times N_{\text{bl}}}$ is the identity matrix in visibility space. The measurement model is non-linear since each row in measurement function $\mathbf{h}$ depends on two different elements in the gains vector $\mathbf{g}$; hence, the original KFS cannot be used. Consequently, it is not widely linear, so the measurement model cannot be augmented. Thus, the widely-linear approximation discussed in §2.4.2 will be used to extend it. This approximation will be evaluated in §3.1.4. That leaves only the evolution model to be defined.

3.1.3 Evolution of Gains

In conjunction with the measurement model, a suitable model to describe the evolution of the gains over time is required. For this, the random walk was chosen. This approach is sufficient to demonstrate the benefits of incorporating a prior model in signal estimation tasks without specialising the model. It is one of the simplest models to use and can be used as a benchmark to compare against future, more sophisticated models. The evolution model of a gain signal over time defined as a random walk according to eq. (2.27) can be expressed as

$$g_{p,t} = g_{p,t-1} + q_{p,t-1}, \tag{3.11}$$

where, for $p \in \mathbb{A}$ and $t \in \mathbb{T}$, the terms $g_{p,t-1}$ and $g_{p,t}$ are consecutive scalar gain terms adjusted by process noise $q_{p,t-1} \sim \mathcal{CN}(0, \sigma_Q^2)$. The process noise is also assumed to have the same statistics over all baselines and in time, and thus can be described entirely by process variance σ_Q^2. In eq. (3.11), the evolution function is the identity matrix and therefore the scalar evolution model is linear. To obtain the multi-variable system, the expression in eq. (3.11) is stacked over all the antennas in $\mathbb{A}$ to get

$$\mathbf{g}_t = \mathbf{g}_{t-1} + \mathbf{q}_{t-1}, \tag{3.12}$$

where, for $t \in \mathbb{T}$, the terms are:

- Current *gains* vector, $\mathbf{g}_t \in \mathbb{C}^{N_{\text{ant}}}$ - The gains vector at the present time step.
- Previous *gains* vector, $\mathbf{g}_{t-1} \in \mathbb{C}^{N_{\text{ant}}}$ - The gains vector from the past time step.
- *Process noise* vector, $\mathbf{q}_{t-1} \sim \mathcal{CN}_{N_{\text{ant}}}(\mathbf{0}, \mathbf{Q}_{t-1})$ - The white noise that drives the random walk over time, with step sizes controlled by the process noise covariance $\mathbf{Q}_{t-1}$:

$$\mathbf{Q}_{t-1} = \text{diag}\left(\left[\, \sigma_Q^2 \;\middle|\; p \in \mathbb{A} \,\right]\right) = \sigma_Q^2 \mathbf{I}. \tag{3.13}$$

The term $\mathbf{I} \in \mathbb{R}^{N_{\text{ant}} \times N_{\text{ant}}}$ is the identity matrix in gain space. For the contents of the gains vectors, see eq. (3.8). In the multi-variable case, linearity is preserved; thus, no non-linear extensions are required. Additionally, the evolution model is strictly linear instead of widely linear, simplifying the steps incorporating the evolution density. This completes the state-space model requirements to establish the KalCal algorithms.

3.1.4 Augmenting the models

The final component before presenting the KalCal algorithms is to take the measurement and evolution state-space models in eqs. (3.6) and (3.12), respectively, and augment them to account for improper complex signals. Here, the two forms of KalCal will diverge. The KalCal-Full algorithm will be constructed using the augmented formalism. In contrast, KalCal-Diag will be constructed using an approximated widely-linear form derived from the augmented form of KalCal-Full. As discussed in §2.2.3, the augmented model provides mathematical completeness but may not always be computationally practical. While the algorithms for widely-linear models can be complex to implement, the number of parameters to estimate remains unchanged, unlike the augmented form where it is doubled. However, augmenting the models can simplify the mathematical derivation process. Therefore, it is best to augment the models for derivation, and then convert to a widely-linear form to avoid computational inefficiencies. Using the stacking of vectors in eq. (2.17) and matrices in eq. (2.20), the augmented measurement model becomes

$$\underline{\mathbf{v}} = \underline{\mathbf{h}}(\underline{\mathbf{g}}) + \underline{\mathbf{r}}, \tag{3.14}$$

where $\underline{\mathbf{v}} \in \mathbb{C}^{2N_{\text{bl}}}$ is the augmented visibility vector, $\underline{\mathbf{g}} \in \mathbb{C}^{2N_{\text{ant}}}$ is the augmented gains vector and $\underline{\mathbf{r}} \sim \mathcal{CN}_{2N_{\text{bl}}}(\mathbf{0}, \underline{\mathbf{R}})$ is the augmented measurement noise vector derived from augmented measurement noise covariance matrix $\underline{\mathbf{R}} \in \mathbb{C}^{2N_{\text{bl}} \times 2N_{\text{bl}}}$. This covariance matrix is expressed as

$$\underline{\mathbf{R}} = \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}^* \end{bmatrix} = \begin{bmatrix} \sigma_R^2 \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \sigma_R^2 \mathbf{I} \end{bmatrix}. \tag{3.15}$$

Even though the evolution model is strictly linear, to assist with derivations, it is easier to stack it by default. The resulting augmented evolution model is

$$\underline{\mathbf{g}}_t = \underline{\mathbf{g}}_{t-1} + \underline{\mathbf{q}}_{t-1}, \tag{3.16}$$

where $\underline{\mathbf{g}}_{t-1}, \underline{\mathbf{g}}_t \in \mathbb{C}^{2N_{\text{ant}}}$ are the previous and current augmented gains vectors, respectively, and $\underline{\mathbf{q}} \sim \mathcal{CN}_{2N_{\text{ant}}}(\mathbf{0}, \underline{\mathbf{Q}}_{t-1})$ is the augmented process noise vector derived from augmented process noise covariance matrix $\underline{\mathbf{Q}} \in \mathbb{C}^{2N_{\text{ant}} \times 2N_{\text{ant}}}$ that is constructed similarly to eq. (3.15), but using $\mathbf{Q}$ instead of $\mathbf{R}$. The next aspect to handle is the non-linearity and the non-holomorphic nature of $\mathbf{h}$ and now $\underline{\mathbf{h}}$. The measurement function $\mathbf{h}$ is non-holomorphic since $\partial \mathbf{h} / \partial \mathbf{g}^*$ would be non-zero; hence widely-linear approximation is needed (see eq. (2.60)). The steps are equivalent to those discussed in §2.4.2. If the linearisation is applied to eq. (3.14)

but in the augmented form, then the resulting Jacobian is identical to those defined by Smirnov and Tasse [38]. As a result, it can be used in place of the measurement equation without further consideration. However, as discussed in §2.4.2, extra attention is required for the EKFS.

During the derivation of the KFS using eqs. (3.14) and (3.16), the augmented linearisation steps are applied when matching Gaussian densities. To summarise the exact steps, let $\underline{\mathbf{g}} \sim \mathcal{C}_{2N_{\text{ant}}}(\underline{\mathbf{m}}, \underline{\mathbf{P}})$ where $\underline{\mathbf{m}}$ is the augmented mean and $\underline{\mathbf{P}}$ is the augmented covariance. Note that the Gaussian density will be considered as a general Gaussian density, i.e. pseudo-covariance is non-zero. Similarly, consider an augmented deviation $\delta\underline{\mathbf{g}} \sim \mathcal{CN}_{2N_{\text{ant}}}(\mathbf{0}, \underline{\mathbf{P}})$ that encapsulates variability across the whole complex plane. Furthermore, consider the following:

$$\underline{\mathbf{g}} = \underline{\mathbf{m}} + \delta\underline{\mathbf{g}}. \tag{3.17}$$

Given all of the above, the widely linearisation of $\underline{\mathbf{h}}(\underline{\mathbf{g}})$ around the mean $\underline{\mathbf{g}}$ is determined using the augmented form of eq. (2.61). By substituting in eq. (3.17), the result is an augmented form of eq. (2.65) written as

$$\underline{\mathbf{h}}(\underline{\mathbf{g}}) \approx \underline{\mathbf{h}}(\underline{\mathbf{m}}) + \underline{\mathbf{J}}\delta\underline{\mathbf{g}}. \tag{3.18}$$

The augmented matrix $\underline{\mathbf{J}} \in \mathbb{C}^{2N_{\text{bl}} \times 2N_{\text{ant}}}$ is the augmented Jacobian and is constructed as[2]

$$\underline{\mathbf{J}} = \begin{bmatrix} \mathbf{J} & \widetilde{\mathbf{J}} \\ \widetilde{\mathbf{J}}^* & \mathbf{J}^* \end{bmatrix}. \tag{3.19}$$

The terms in the block matrix are the Jacobian $\mathbf{J} \in \mathbb{C}^{N_{\text{bl}} \times N_{\text{ant}}}$ and Wirtinger Jacobian $\widetilde{\mathbf{J}} \in \mathbb{C}^{N_{\text{bl}} \times N_{\text{ant}}}$ evaluated with respect to $\mathbf{m}$. In this case, the block Jacobians are calculated first with respect to $\mathbf{m}$, the standard form of the augmented mean $\underline{\mathbf{m}}$, and then the augmented Jacobian is assembled. However, performing this evaluation with respect to augmented mean $\underline{\mathbf{m}}$ only after finding the whole augmented Jacobian is also valid. The outcome will remain the same. The former approach is chosen to reduce the required computational resources involved. These Jacobians are constructed as follows:[3]

$$\mathbf{J} = \frac{\partial \mathbf{h}}{\partial \mathbf{g}}(\mathbf{m}) = \left[[x_{pq}\hat{g}_q^*]_{ip} \;\middle|\; (p,q)_i \in \mathbb{B} : \begin{cases} x_{pq} \in \mathbf{x}, \\ \hat{g}_q \in \mathbf{m}. \end{cases} \right], \tag{3.20}$$

$$\widetilde{\mathbf{J}} = \frac{\partial \mathbf{h}}{\partial \mathbf{g}^*}(\mathbf{m}) = \left[[\hat{g}_p x_{pq}]_{iq} \;\middle|\; (p,q)_i \in \mathbb{B} : \begin{cases} x_{pq} \in \mathbf{x}, \\ \hat{g}_p \in \mathbf{m}. \end{cases} \right]. \tag{3.21}$$

[2] A critical fact here is that no factor of half is included in the Jacobian, unlike how it is traditionally handled in calibration. This is justified by the same steps done by Dini and Mandic [56] and Dang and Scharf [57].

[3] The term $(p,q)_i$ is written to show both the baseline tuple (p,q) and the index position of the baseline i in $\mathbb{B}$. It is similar to the output of the built-in `enumerate` function in the python3 programming language, e.g. "`for i, (p, q) in enumerate(B)`" (source).

The additional term $\mathbf{x} \in \mathbb{C}^{N_{\text{bl}}}$ used above is the *predicted visibilities* vector where each predicted visibility is stacked according to the order of the baseline index set $\mathbb{B}$. The constructions of eqs. (3.20) and (3.21) are equivalent to the standard calibration Jacobian constructions detailed by Smirnov and Tasse [38]. From eq. (3.18), the Gaussian densities determined via the augmented measurement model can be approximated. For the approximated Gaussian, the augmented mean is calculated as

$$\mathrm{E}\left[\underline{\mathbf{h}}(\underline{\mathbf{g}})\right] \approx \underline{\mathbf{h}}(\underline{\mathbf{m}}), \tag{3.22}$$

and the augmented covariance is calculated as

$$\mathrm{Cov}\left[\underline{\mathbf{h}}(\underline{\mathbf{g}}), \underline{\mathbf{h}}(\underline{\mathbf{g}})\right] \approx \underline{\mathbf{JPJ}}^{H}. \tag{3.23}$$

The above approximations are logically equivalent to those detailed in §2.4.2, but are presented in the augmented form and utilise a different non-linear function. With this result and the augmented evolution model, the AEKFS for KalCal-Full can be derived. The WEKFS can then be derived for KalCal-Diag from the resulting AEKFS; however, this will not be necessary yet since several approximations and assumptions are required before this is done. The following section will state the KalCal algorithms. Following this are the various extensions employed to arrive at these algorithms.

3.2 Kalman Filtering and Smoothing in Calibration

Based on the details given in chapter 2 and the setup for the calibration problem in §3.1, this section will directly state the widely-linear and augmented EKFS required for the KalCal-Diag and KalCal-Full algorithms, respectively. The algorithms will be presented with already-included modifications, improvements, and simplifications. Furthermore, implementations of the energy function described in §2.4.3 will be provided for each algorithm, considering the available mathematical terms. The explanations of these alterations will be discussed in §3.3. The following subsections are direct recipes for constructing the KalCal algorithms.

3.2.1 KalCal-Full

This section is devoted to the KFS algorithms that make up KalCal-Full. In particular, the focus will be on presenting the algorithm that will solve the calibration problem in §3.1.1 using the augmented formalism. Furthermore, no approximations are applied to the algorithm, and the only optimisation introduced is the whitening of the noise in §3.3.1 and the Kalman-Woodbury identity discussed in §3.3.2. Even though the prior information will be assumed to follow proper Gaussians, the calculated covariances and expectations will consider all correlations that can exist over time, antenna and baseline axes, and between conjugate pairs. This calibration algorithm is called KalCal-Full as it considers the "full" covariance matrices. The primary goal will be to validate whether the principal KalCal algorithm works as a calibration algorithm and whether the theory is valid. However, it also serves to test

the theoretical advantages of including complete covariance terms, as noted by Schreier and Scharf [33], Dini and Mandic [56], and Dang and Scharf [57].

Specifically, for estimates based around least-square approaches, e.g. the MLEs of quadratic problems, the accuracy of an estimate derived from a widely-linear (or augmented) model that includes pseudo-covariances is equivalent or better than the corresponding linear model that does not. This accuracy gain depends on the degree of impropriety of the underlying state-space models and their noise assumptions. Generally, as the impropriety increases, the accuracy of the widely-linear model estimate increases in linear and non-linear cases compared to the standard linear model estimate [56]. See Dini and Mandic [56] and Dang and Scharf [57] for a comprehensive analysis of this fact. The goal of the KFS is to generate Gaussian densities. However, in the linear Gaussian case, the mean is equivalent to the MAP (see §2.2.7). Therefore, the accuracy of the KFS can be directly measured against ML methods used in calibration. As a result, the gains impropriety assumptions can be assessed by measuring the impact of the pseudo-covariance on solution accuracy. Hence, it would be an important result to analyse in the context of calibration and the viability of KalCal. The following algorithms are the aforementioned AEKFS derived from eqs. (3.14) and (3.17).

Algorithm 3.2.1 (KalCal-Full Predictor and Filter). Given the prior augmented filter mean $\underline{\mathbf{m}}_{f,0} \in \mathbb{C}^{2N_{\text{ant}}}$ and corresponding prior augmented filter covariance $\underline{\mathbf{P}}_{f,0} \in \mathbb{C}^{2N_{\text{ant}} \times 2N_{\text{ant}}}$, the following steps, for $k \in \mathbb{T}$, defines the forward recursive AEKF of KalCal-Full:

1.) **Prediction Step (Full-Predictor)**:

$$\underline{\mathbf{m}}_{p,k} = \underline{\mathbf{m}}_{f,k-1}, \tag{3.24}$$
$$\underline{\mathbf{P}}_{p,k} = \underline{\mathbf{P}}_{f,k-1} + \underline{\mathbf{Q}}, \tag{3.25}$$

2.) **Update Step (Full-Filter)**:

$$\underline{\mathbf{e}}_k = \underline{\mathbf{v}}_k - \underline{\mathbf{h}}(\underline{\mathbf{m}}_{p,k}), \tag{3.26}$$
$$\underline{\mathbf{z}}_k = \underline{\mathbf{J}}_k^H \underline{\mathbf{e}}_k, \tag{3.27}$$
$$\underline{\mathbf{U}}_k = \underline{\mathbf{J}}_k^H \underline{\mathbf{J}}_k, \tag{3.28}$$
$$\underline{\mathbf{P}}_{f,k} = (\underline{\mathbf{P}}_{p,k}^{-1} + \underline{\mathbf{U}}_k)^{-1}, \tag{3.29}$$
$$\underline{\mathbf{m}}_{f,k} = \underline{\mathbf{m}}_{p,k} + \underline{\mathbf{P}}_{f,k}\underline{\mathbf{z}}_k. \tag{3.30}$$

This algorithm's augmented term construction can be found in sections 2.2.3 and 3.1.4.

Each step is given an identifier to reference it specifically during the analysis in chapter 4. The augmented measurement matrix in $\underline{\mathbf{J}}_k$ is evaluated with respect to the augmented predictor mean $\underline{\mathbf{m}}_{p,k}$ according to sections 2.4.2 and 3.1.4. The update step uses the

Kalman-Woodbury identity in §2.4.4. Note that $\mathbf{z}_k$ and $\mathbf{U}_k$ are equivalent to the $J^H r$- and $J^H J$-computation, respectively, which commonly appear in NLL algorithms, like those within calibration algorithms [38]. Hence, this algorithm can easily adapt to existing software and calibration techniques with minimal effort. The last optimisation used is the whitening transform introduced in §3.3.1 that allows the augmented measurement vectors and augmented measurement matrices to absorb the augmented measurement covariance matrix, leaving an identity in its place. No adjustments are required for the *Augmented Extended Kalman Smoother* (AEKS) and can be stated as is.

Algorithm 3.2.2 (KalCal-Full Smoother). Let the AEKF in algorithm 3.2.1 run and generate the augmented predictor means, $\underline{\mathbf{m}}_{p,1:N_{\text{time}}}$, with covariances, $\underline{\mathbf{P}}_{p,1:N_{\text{time}}}$ (including their inverses), and augmented filter means, $\underline{\mathbf{m}}_{f,1:N_{\text{time}}}$ with covariances, $\underline{\mathbf{P}}_{f,1:N_{\text{time}}}$. Using the last filter mean $\underline{\mathbf{m}}_{f,N_{\text{time}}}$ and covariance $\underline{\mathbf{P}}_{f,N_{\text{time}}}$ as the start point, the following steps, for $k \in \mathbb{T}$, defines the backward recursive AEKS of KalCal-Full:[4]

3.) **Smoothing Step (Full-Smoother):**

$$\underline{\mathbf{W}}_k = \underline{\mathbf{P}}_{f,k}\underline{\mathbf{P}}_{p,k+1}^{-1}, \tag{3.31}$$

$$\underline{\mathbf{m}}_{s,k} = \underline{\mathbf{m}}_{f,k} + \underline{\mathbf{W}}_k(\underline{\mathbf{m}}_{s,k+1} - \underline{\mathbf{m}}_{p,k+1}), \tag{3.32}$$

$$\underline{\mathbf{P}}_{s,k} = \underline{\mathbf{P}}_{f,k} + \underline{\mathbf{W}}_k(\underline{\mathbf{P}}_{s,k+1} - \underline{\mathbf{P}}_{p,k+1})\underline{\mathbf{W}}_k^H, \tag{3.33}$$

The construction of augmented terms within this algorithm can be found in sections 2.2.3 and 3.1.4.

The resulting smoother algorithm heavily re-uses the calculations made by algorithm 3.2.2. In addition to filter terms, it re-uses predictor terms and the inverse of predictor covariance matrices. This effectively eliminates the need for a prediction step within the smoother algorithm. Furthermore, the only inversion steps are constrained to the Full-Filter. This recycling of terms is done to avoid additional computations and reduce the overall complexity of the joint AEKFS algorithm. This comes at the cost of increased storage requirements to hold the additional solutions, but is worthwhile for the efficiency it brings in computation time. With algorithms 3.2.1 and 3.2.2, the AEKFS is complete and therefore can be implemented as the KalCal-Full algorithm. The final component to incorporate is the equivalent energy function discussed in §2.4.3 and detailed in §3.3.4 using the terms generated in algorithm 3.2.1.

[4] Note, the prior filter mean and covariance are not generally included in the smoothing step since the problem is defined over $\mathbb{T}$, i.e. $k = 0 \in \mathbb{T}$. Instead, they can be improved via optimisation techniques, like those derived from the energy function calculations.

Algorithm 3.2.3 (KalCal-Full Energy Function). While running the AEKF in algorithm 3.2.1 and assuming $\varphi_0 = 0$, the energy function φ_k can be calculated after each time step $k \in \mathbb{T}$ using the calculated terms in the AEKF with the expression below:

$$\underline{\mathbf{S}}_k^{-1} = \mathbf{I} - \underline{\mathbf{J}}_k \underline{\mathbf{P}}_{f,k} \underline{\mathbf{J}}_k^H, \tag{3.34}$$

$$\underline{\mathbf{L}}_k = \text{chol}(\underline{\mathbf{S}}_k^{-1}), \tag{3.35}$$

$$\underline{\mathbf{l}}_k = \text{diag}(\underline{\mathbf{L}}_k), \tag{3.36}$$

$$\varphi_k = \varphi_{k-1} - \frac{1}{2} \sum_{l \in \underline{\mathbf{l}}_k} \ln(l) + \frac{1}{2} \underline{\mathbf{e}}_k^H \underline{\mathbf{S}}_k^{-1} \underline{\mathbf{e}}_k. \tag{3.37}$$

The operator chol($\cdot$) is *Cholesky decomposition* operator and diag($\cdot$) is the *diagonal* operator in definition A.1.13.

This energy function is referred to as *Full-Energy* in chapter 4. Therefore, the whole KalCal-Full framework consists of the *Full-Predictor*, *Full-Filter*, *Full-Smoother* and *Full-Energy*. The energy function is calculated at the end of each Full-Filter step to re-use the filter covariance and residual vector terms. Including the Cholesky decomposition merely aids in the speed and stability of the computation of the energy function. The meaning of the energy function does not change. This completes the necessary KalCal-Full calibration algorithm to be tested in chapter 4.

3.2.2 KalCal-Diag

Following from the AEKFS, it is clear to see the various disadvantages of the augmented formalism with no numerical or statistical approximations in the context of computational resources. Namely, by introducing the augmented vectors, the number of gains terms to estimate doubles due to the stacking of the vectors, even though the upper half is the conjugate of the lower half, and the Jacobians can be evaluated with respect to the upper half exclusively (see eqs. (3.20) and (3.21)). Furthermore, the various augmented matrices go from containing N_{ant}^2, $N_{\text{ant}} N_{\text{bl}}$, and N_{bl}^2 elements to $4N_{\text{ant}}^2$, $4N_{\text{ant}} N_{\text{bl}}$ and $4N_{\text{bl}}^2$, respectively. That means each matrix scales in storage size by a factor of four. For matrix operations such as matrix-to-matrix multiplication and matrix inversion, the number of calculations required roughly scales by a factor of eight using the augmented formalism [38]. There are ways around this issue, such as exploiting the sparse and repetitive structure of the vectors and matrices to focus only on calculations of non-zero elements. Additionally, sophisticated software implementations use specialised high-performance libraries and tools that leverage the power of hardware to alleviate the problem [1].

However, a common and valuable optimisation within calibration is the *diagonal approximation* by Smirnov and Tasse [38]. Its implementation defines the KalCal-Diag algorithm. This approximation allows the resulting KFS to be described entirely by vectors, which are

known to be computationally efficient compared to matrices. The details of its implementation are discussed within §3.3.3. Along with the diagonal approximation, the same adjustments made for KalCal-Full, such as the Kalman-Woodbury identity and the whitening transform, are also applied for KalCal-Diag. The aim of KalCal-Diag is to validate whether the KalCal algorithm works as a practical calibration algorithm by using the diagonal approximation. Moreover, it will also test the widely-linear formalism, as described in §2.2.3 and [57], to see if the algorithm's computational complexity can be improved compared to the augmented formalism of KalCal-Full. The resulting algorithms below are the approximated WEKFS derived from the AEKFS described in §3.2.1.

Algorithm 3.2.4 (KalCal-Diag Predictor and Filter). Given the prior filter mean $\mathbf{m}_{f,0} \in \mathbb{C}^{N_{\text{ant}}}$ and corresponding covariance vector $\mathbf{p}_0 \in \mathbb{R}^{N_{\text{ant}}}$, the following steps, for $k \in \mathbb{T}$, defines the forward recursive, diagonally approximated WEKF of KalCal-Diag:

1.) **Prediction Step (Diag-Predictor)**:

$$\mathbf{m}_{p,k} = \mathbf{m}_{f,k-1}, \tag{3.38}$$
$$\mathbf{p}_{p,k} = \mathbf{p}_{f,k-1} + \mathbf{q}, \tag{3.39}$$

2.) **Update Step (Diag-Filter)**:

$$\mathbf{e}_k = \mathbf{v}_k - \mathbf{h}(\mathbf{m}_{p,k}), \tag{3.40}$$
$$\mathbf{z}_k = \mathbf{J}_k^H \mathbf{e}_k + \widetilde{\mathbf{J}}_k^H \mathbf{e}_k^*, \tag{3.41}$$
$$\mathbf{u}_k = \operatorname{diag}(\mathbf{J}_k^H \mathbf{J}_k) + \operatorname{diag}(\widetilde{\mathbf{J}}_k^H \widetilde{\mathbf{J}}_k), \tag{3.42}$$
$$\mathbf{p}_{f,k} = (\mathbf{p}_{p,k}^{(-1)} + \mathbf{u}_k)^{(-1)}, \tag{3.43}$$
$$\mathbf{m}_{f,k} = \mathbf{m}_{p,k} + \mathbf{p}_{f,k} \odot \mathbf{z}_k. \tag{3.44}$$

The description and construction of the above terms can be found in sections 3.1.4 and 3.3.3. Additionally, the algorithm contains the:

- *Hadamard* inverse, $(\,\cdot\,)^{(-1)}$ - the element-wise inversion or reciprocal of a given vector (see remark A.1.8).
- *Hadamard* product, $\odot$ - the element-wise multiplication of two vectors (see definition A.1.5).

Again, each step is given a unique identifier to reference during the analysis in chapter 4. This algorithm exploits the optimisation properties derived through the various diagonal approximations. Furthermore, it includes several assumptions that allow it to be represented in

this succinct, widely-linear form. Specifically, the gain terms are assumed to be uncorrelated and the gain signal is considered a proper complex signal. The result is that all calculations can be transformed into vector operations that are much faster than matrix operations and require far less storage and memory. This is further improved because it does not rely on the augmented formalism. Similarly to algorithm 3.2.1, the calculations of $\mathbf{z}_k$ and $\mathbf{u}_k$ are well-studied and implemented processes within calibration software [1], [38]. Therefore, algorithm 3.2.4 can also be easily integrated with existing calibration techniques and software. The rest of the adjustments and constructions are thoroughly detailed in §3.3.

The sacrifice of such optimisations is the reduction of the accuracy of the predictor and filter terms and their resulting densities. The only concern of introducing such an approximation is that, unlike iterative-based calibration solutions, the WEKF cannot simply increase the number of estimation epochs to converge to a solution, as suggested by Smirnov and Tasse [38], since each estimation is not calculated via some descent-based algorithm. There exists a KF that can potentially similarly address this issue called the *Iterated Extended Kalman Filter* (IEKF); however, it is left for future investigations. For now, the degree to which the diagonal approximation affects the KFS algorithm will be analysed to assess its use with this approach. The corresponding *Widely-Linear Extended Kalman Smoother* (WEKS) that couples with algorithm 3.2.4 derived from the AEKS in §3.2.1 is given below.

Algorithm 3.2.5 (KalCal-Diag Smoother). Let the WEKF in algorithm 3.2.4 run and generate the predictor means $\mathbf{m}_{p,1:N_{\text{time}}}$, with covariance vectors $\mathbf{p}_{p,1:N_{\text{time}}}$ (including their inverses), and filter means $\mathbf{m}_{f,1:N_{\text{time}}}$, with covariance vectors $\mathbf{p}_{f,1:N_{\text{time}}}$. Using the last filter mean $\mathbf{m}_{f,N_{\text{time}}}$ and covariance vector $\mathbf{p}_{f,N_{\text{time}}}$ as the start point, the following steps, for $k \in \mathbb{T}$, defines the backward recursive, diagonally approximated WEKS of KalCal-Diag:[5]

3.) **Smoothing Step (Diag-Smoother):**

$$\mathbf{w}_k = \mathbf{p}_k \oslash \mathbf{p}_{f,k+1}, \tag{3.45}$$

$$\mathbf{m}_{s,k} = \mathbf{m}_k + \mathbf{w}_k \odot (\mathbf{m}_{s,k+1} - \mathbf{m}_{f,k+1}), \tag{3.46}$$

$$\mathbf{p}_{s,k} = \mathbf{p}_k + \mathbf{w}_k^{(2)} \odot (\mathbf{p}_{s,k+1} - \mathbf{p}_{f,k+1}), \tag{3.47}$$

In addition to the operators in algorithm 3.2.4, this algorithm includes the:

- *Hadamard* division $\oslash$ - the element-wise division of one vector with elements of another (see definition A.1.6).
- n^{th} *Hadamard* power $(\,\cdot\,)^{(n)}$ - the element-wise power operator where each element is raised to the power of n (see definition A.1.7).

[5] Similar to the algorithm 3.2.2, the prior filter mean and covariance are not smoothed as they are not part of $\mathbb{T}$.

Again, algorithm 3.2.5 re-uses computations already calculated within algorithm 3.2.4 to further reduce the computational complexity of the calibration algorithm. The last requirement is the incorporation of the equivalent energy function as discussed in §2.4.3 and detailed in §3.3.4 using the terms generated by algorithm 3.2.4.

Algorithm 3.2.6 (KalCal-Diag Energy Function). While running the WEKF in algorithm 3.2.4 and assuming $\varphi_0 = 0$, the energy function φ_k can be calculated after each time step $k \in \mathbb{T}$ using the calculated terms in the WEKF with the expression below:

$$\varphi_k = \varphi_{k-1} - \sum_{s \in \mathbf{s}_k^{-1}} \ln(s) + \left(\mathbf{e}_k \odot \mathbf{s}_k^{-1}\right)^H \mathbf{e}_k, \tag{3.48}$$

where the approximate inverse error covariance vector $\mathbf{s}^{-1}$ is calculated explicitly as

$$\mathbf{s}_k^{-1} = \left[1 - \left(\frac{\sigma_{f,p}}{\sigma_R} |\hat{g}_q x_{pq}|^2 + \frac{\sigma_{f,q}}{\sigma_R} |\hat{g}_p x_{pq}|^2 \right) \middle| (p,q) \in \mathbb{B} : \begin{cases} \hat{g}_p, \hat{g}_q \in \mathbf{m}_{p,k} \\ \sigma_{f,p}, \sigma_{f,q} \in \mathbf{p}_{f,k} \\ x_{pq} \in \mathbf{x}_k \\ \sigma_R^2 \in \mathbf{R}_k \end{cases} \right]^T. \tag{3.49}$$

In addition to the operators in algorithms 3.2.4 and 3.2.5, this algorithm includes the *absolute value* operator, $|\cdot|$.

This energy function is referred to as *Diag-Energy* in chapter 4. Therefore, the whole KalCal-Diag framework consists of the *Diag-Predictor*, *Diag-Filter*, *Diag-Smoother* and *Diag-Energy*. Note the contents of these algorithms depend heavily on the content that will be described in §3.3. Algorithm 3.2.6 is also calculated at the end of each Diag-Filter step. These approximated algorithms form the complete KalCal-Diag to be used in chapter 4.

3.3 Extensions to KalCal

With the previous presentations of the KalCal-Full and KalCal-Diag algorithms, this section delves into the optimisations and approximations that were used to improve the performance of the KalCal algorithms in terms of computational complexity and estimation accuracy.

3.3.1 Whitening Transform

This is a common approach used within statistical estimation problems and in calibration. The goal of the *whitening transform* is to explicitly remove the need to construct a measurement covariance matrix within quadratic loss problems. A whitening transform is a linear transform that, when applied to a complex random vector, changes its covariance matrix to an identity matrix, i.e. elements within the vector are uncorrelated and each has unity

variance. The result is a *whitened* measurement density that replaces the role of the measurement density. The complete steps to create the whitened measurement density are provided in lemma A.4.5. This section outlines the outcomes of whitening the measurement noise in the KalCal algorithms. For both KalCal-Full and KalCal-Diag, the measurement density to be whitened is based on the corresponding measurement model. To understand how this result is useful, consider KalCal-Full as the example. The standard measurement density for KalCal-Full is $p(\underline{\mathbf{v}}_k \mid \underline{\mathbf{g}}_k)$ at time $k \in \mathbb{T}$. If $\underline{\mathbf{v}}'_k$ is the corresponding *whitened augmented visibility vector*, then the whitened measurement density is

$$p(\underline{\mathbf{v}}'_k \mid \underline{\mathbf{g}}_k) = \mathcal{CN}_{2N_{\text{bl}}}\left(\underline{\mathbf{v}}_k \mid \underline{\mathbf{R}}_k^{-\frac{1}{2}}\underline{\mathbf{h}}(\underline{\mathbf{g}}_k), \mathbf{I}\right), \tag{3.50}$$

where $\mathbf{I} \in \mathbb{C}^{2N_{\text{bl}} \times 2N_{\text{bl}}}$ is the identity matrix. Now, the effect this has on the derived algorithms can be observed by examining the NLL of eq. (3.50):

$$-\ln p(\underline{\mathbf{v}}'_k \mid \underline{\mathbf{g}}_k) \propto \left(\underline{\mathbf{R}}^{-\frac{1}{2}}\underline{\mathbf{v}}_k - \underline{\mathbf{R}}_k^{-\frac{1}{2}}\underline{\mathbf{h}}(\underline{\mathbf{g}}_k)\right)^H \left(\underline{\mathbf{R}}^{-\frac{1}{2}}\underline{\mathbf{v}}_k - \underline{\mathbf{R}}_k^{-\frac{1}{2}}\underline{\mathbf{h}}(\underline{\mathbf{g}}_k)\right). \tag{3.51}$$

Within eq. (3.51), the original visibility is scaled by $\underline{\mathbf{R}}^{-\frac{1}{2}}$ to get the whitened visibility vector and the covariance term has vanished. Therefore, using the whitened measurement density instead of the standard measurement density applies the same scaling in the KFS algorithms. The subsequent steps added to the KalCal algorithms, for both versions, to incorporate this whitening process are the following changes to how the terms found within §3.2 are calculated:

- *Whitened measurement* covariance, $\mathbf{R}_k \leftarrow \mathbf{I} \in \mathbb{R}^{N_{\text{bl}} \times N_{\text{bl}}}$ - The measurement covariance is now the identity matrix matching the dimensions of the measurement model.
- *Whitened measurement* vector, $\mathbf{v}_k \leftarrow \mathbf{R}_k^{-\frac{1}{2}}\mathbf{v}_k$ - The measurement or visibility vector that has been whitened with respect to measurement noise, i.e. $\underline{\mathbf{v}}_k$ is now $\underline{\mathbf{v}}'_k$.
- *Whitened measurement* Jacobians, $\mathbf{J}_k \leftarrow \mathbf{R}_k^{-\frac{1}{2}}\mathbf{J}_k$ and $\widetilde{\mathbf{J}}_k \leftarrow \mathbf{R}_k^{-\frac{1}{2}}\widetilde{\mathbf{J}}_k$ - Similar to the whitened measurement vector, but with respect to the Jacobian and Wirtinger Jacobian.

These re-assignments can be directly substituted into the WEKF algorithm without requiring adjustment. For the AEKF, the measurement noise is considered proper in this case since $\underline{\mathbf{R}}_k$ is diagonal. Therefore, the same transform is applied to the conjugate measurement model when constructing the augmented whitened terms, i.e. the Wirtinger Jacobians are also scaled. If the same assignments are made where each matrix and vector appears in the augmented terms, nothing changes for the AEKF. An advantage of this whitening transform is that the calculations of $\mathbf{z}$ and $\mathbf{U}$ discussed in §2.4.4 now form the respective $J^H r$- and $J^H J$-computations commonly used within calibration techniques and software. Furthermore, it allows additional optimisations to be integrated into the KalCal algorithms.

3.3.2 Large Inverse Problem and the Kalman-Woodbury Identity

As Tasse [11] and Smirnov and Tasse [38] point out, inverting large matrices is an expensive computation for calibration algorithms. For an invertible $n \times n$ matrix where $n \in \mathbb{N}$, the computation time scales cubically, i.e. matrix inversion calculates at $\mathcal{O}(n^3)$ time.[6] For multi-dimensional situations with large axis lengths, this operation can quickly become crippling. For KalCal, the same issue creeps in during the inversion of the error covariance matrix. Without optimisations and the augmented formalism, the computation would require $\mathcal{O}(N_b^3)$ or, in terms of antenna count, $\mathcal{O}(N_a^6)$ calculations to invert the matrix. This relationship is established using $N_{\text{bl}} = N_{\text{ant}}(N_{\text{ant}} - 1)/2$. Therefore, the augmented formalism scales the required number of calculations by a factor of eight [38]. Furthermore, the inversion of the matrix is sometimes ill-conditioned, which leads to issues of numerical instability of the computed inverse. This can degrade the accuracy of solutions or even cause the algorithm to fail. Hence, it is clear that even in this small calibration scenario, the computation of the large inverse is still a costly problem, and, without optimisation, the KFS approach would scale poorly with the growing data sizes [11].

Tasse [11] demonstrated the use of the SMW to handle this issue within radio interferometry, and, more specifically, within the KF in killMS. Using eq. (2.77) as a guide, the calculation of the augmented error covariance matrix can be rewritten using eq. (2.75) as

$$\underline{\mathbf{S}}_k^{-1} = \mathbf{I} - \underline{\mathbf{J}}_k \left(\underline{\mathbf{P}}_{p,k}^{-1} + \underline{\mathbf{J}}_k^H \underline{\mathbf{J}}_k \right)^{-1} \underline{\mathbf{J}}_k. \tag{3.52}$$

Equation (3.52) includes the whitening transform discussed in §3.3.1 [11]. The reason why this change is so beneficial is that the computation changes from a single $\mathcal{O}(N_{\text{ant}}^6)$ inversion to two $\mathcal{O}(N_{\text{ant}}^2)$ inversions. This improves the conditioning of the calculations and, especially for large arrays like MeerKAT where $N_{\text{ant}} = 64$, it scales much better. Given the success of its implementation within killMS, it is reasonable to assume that its inclusion within KalCal will yield similar success. When implementing this optimisation within the KalCal algorithms, the Kalman-Woodbury identity was uncovered (see §2.4.4). Essentially, based on the work of Tasse and the phenomenological Jones constructions of Smirnov and Tasse [38], the SMW applied to the update step of KalCal-Full can significantly simplify the step to the following:

$$\underline{\mathbf{P}}_{f,k} = \left(\underline{\mathbf{P}}_{p,k}^{-1} + \underline{\mathbf{J}}_k^H \underline{\mathbf{J}}_k \right)^{-1}, \tag{3.53}$$

$$\underline{\mathbf{m}}_{f,k} = \underline{\mathbf{m}}_{p,k} + \underline{\mathbf{P}}_{f,k} \underline{\mathbf{J}}_k^H \underline{\mathbf{e}}_k. \tag{3.54}$$

The significance of eqs. (3.53) and (3.54) is that the complexity of KalCal-Full is greatly reduced due to the Kalman-Woodbury identity. In this form, computing these two filter terms is still computationally expensive; however, it has been expressed in a way that uses well-studied and optimised $J^H r$- and $J^H J$-computations (see Smirnov and Tasse [38]). This improves its compatibility with existing calibration techniques and software, enhancing its

[6] *Big-O Notation*: An approximate indication of the number of computations required for a given data size parameter n. It is a useful measure when analysing computational complexity to determine how algorithms scale in computational time as data sizes increase [18], [65].

appropriateness as a new calibration approach and ability to scale with data size. Hence, within algorithm 3.2.1, the terms $\underline{\mathbf{J}}_k^H \underline{\mathbf{J}}_k$ and $\underline{\mathbf{J}}_k^H \underline{\mathbf{e}}_k$ are calculated separately as $\underline{\mathbf{U}}_k$ and $\underline{\mathbf{z}}_k$, respectively, to emphasise this fact.

Further investigation is required into the impact of this result in calibration and how it affects the numerical aspects of its calculations. Moreover, the version implemented within KalCal-Diag is discussed in the next section. Also, by using the Kalman-Woodbury identity, certain obstacles are introduced when trying to calculate the equivalent energy functions. This is discussed within §3.3.4.

3.3.3 Diagonal Approximation

Despite the benefits of the Kalman-Woodbury identity to reduce the computational complexity of the KalCal algorithms, additional optimisations are required to make such an approach viable for the Big Data volumes associated with calibration. A successful technique within calibration to mitigate this problem is applying a diagonal approximation on specific terms [12], [38]. For KalCal, this takes two forms: (i) diagonal approximation of $J^H J$-computations, and (ii) diagonal approximation of covariance matrices. The first is motivated by the fact that the resulting $J^H J$ terms have been observed to be diagonally dominant [38]. For each row in the matrix, the magnitude of the diagonal element is greater than or equal to the sum of the magnitudes of the off-diagonal elements. This means that the influence on a variable is most substantial with its equation (*diagonal terms*) within the system of equations compared to the influence of other variables (*off-diagonal terms*) [42]. Hence, the term $J^H J$ can be approximated as a diagonal matrix without severe degradation of the information conveyed by this matrix. Several matrix-related operations on diagonal matrices are cheaper to compute and often more numerically stable.

The second approximation relates to the correlation information within the covariance matrices. In particular, assuming that individual antenna gains within $\mathbb{A}$ are independent, the resulting KFS covariance matrices related to the gain signal become diagonal. Using the Kalman-Woodbury identity update rules, this assumption only affects the covariances related to the gains estimate solutions. This action drops any information within off-diagonal terms, rendering the covariance matrices diagonal. To further enhance this approximation, the gain signal and visibility signal are assumed to be proper, i.e. pseudo-covariances between terms and their conjugates are zero. The effect of such an assumption will be examined within chapter 4, but a future in-depth analysis is required to fully understand the impact of such a choice, especially with more practical calibration scenarios. However, the primary reason for this assumption is to reduce the complexity of the widely-linear form. As demonstrated in §2.4.2, this formulation can get highly verbose and may not improve computational complexity. To exploit this formulation, the assumed impropriety of the gain and visibility signals removes the pseudo-covariance-related terms. As a result, only the standard covariance terms are considered, and the verbose structure of the widely-linear form is avoided. This can be easily verified by examining the resulting expressions in algorithm 2 in [57] when the pseudo-covariance terms are assumed to be zero. Another diagonal approximation is discussed in the next section since it is related to the energy function calculation.

With these diagonal approximations and related assumptions, KalCal-Full can be reduced and optimised to form the KalCal-Diag algorithm. The resulting KFS algorithms used are the WEKFS with pseudo-covariance terms ignored. The matrix operations are converted into vector operations to best leverage the underlying numerical software libraries. This is done by applying the diagonal operator diag(·) to the diagonal and the diagonally dominated matrices to convert them into vectors. Using lemma A.1.14 and remark A.1.16, the affected matrix operations are converted into *Hadamard* and vector operations. The Hadamard operation definitions are provided in definitions A.1.5 to A.1.7, remarks A.1.8 to A.1.10, and lemma A.1.11. For this research, when matrices are converted to vectors, they retain the same symbol, but now in lowercase, e.g. $\mathbf{x} = \text{diag}(\mathbf{X})$. To start, the prediction step or KalCal-Diag predictor then becomes

$$\mathbf{m}_{p,k} = \mathbf{m}_{f,k-1}, \tag{3.55}$$

$$\mathbf{p}_{p,k} = \mathbf{p}_{f,k-1} + \mathbf{q}_{k-1}, \tag{3.56}$$

where $\mathbf{p}_{p,k} = \text{diag}(\mathbf{P}_{p,k})$, $\mathbf{p}_{f,k-1} = \text{diag}(\mathbf{P}_{f,k-1})$, and $\mathbf{q}_{k-1} = \text{diag}(\mathbf{Q}_{k-1})$. The Kalman-Woodbury identity form in algorithm 2.4.1 is used for the update step. The resulting update step for KalCal-Diag, based on the above assumptions and the new update step in §3.3.2, is

$$\mathbf{e}_k = \mathbf{v}_k - \mathbf{J}_k \mathbf{m}_{p,k}, \tag{3.57}$$

$$\mathbf{z}_k = \mathbf{J}_k^H \mathbf{e}_k + \widetilde{\mathbf{J}}_k^H \mathbf{e}_k^*, \tag{3.58}$$

$$\mathbf{u}_k = \text{diag}(\mathbf{J}_k^H \mathbf{J}_k) + \text{diag}(\widetilde{\mathbf{J}}_k^H \widetilde{\mathbf{J}}_k), \tag{3.59}$$

$$\mathbf{p}_{f,k} = (\mathbf{p}_{p,k}^{(-1)} + \mathbf{u}_k)^{(-1)}, \tag{3.60}$$

$$\mathbf{m}_{f,k} = \mathbf{m}_{p,k} + \mathbf{p}_{f,k} \odot \mathbf{z}_k, \tag{3.61}$$

where $\mathbf{p}_{f,k} = \text{diag}(\mathbf{P}_{f,k})$, $(\ \cdot\ )^{(-1)}$ is the element-wise or *Hadamard* inverse (see remark A.1.8) and $\odot$ is the element-wise or *Hadamard* product (see definition A.1.5). The above prediction and update step is written in terms of vectors. To avoid Jacobian matrix constructions in eqs. (3.57) to (3.59), explicit calculations of the elements of $\mathbf{J}_k^H \mathbf{J}_k$ and $\widetilde{\mathbf{J}}_k^H \widetilde{\mathbf{J}}_k$ are done instead. That completes the KalCal-Diag filter, so the KalCal-Diag smoother can be addressed. No further optimisations are required for the smoothing step because of the simplicity of the chosen evolution model. Therefore, the smoothing step would be

$$\mathbf{w}_k = \mathbf{p}_{f,k} \oslash \mathbf{p}_{p,k+1}, \tag{3.62}$$

$$\mathbf{m}_{s,k} \approx \mathbf{m}_{f,k} + \mathbf{w}_k \odot \left(\mathbf{m}_{s,k+1} - \mathbf{m}_{p,k+1}\right), \tag{3.63}$$

$$\mathbf{p}_{s,k} \approx \mathbf{p}_{f,k} + \mathbf{w}_k^{(2)} \odot \left(\mathbf{p}_{s,k+1} - \mathbf{p}_{p,k+1}\right), \tag{3.64}$$

where $\mathbf{p}_{f,k} = \text{diag}(\mathbf{P}_{f,k})$, $\mathbf{p}_{p,k+1} = \text{diag}(\mathbf{P}_{p,k+1})$, $\mathbf{p}_{s,k} = \text{diag}(\mathbf{P}_{s,k})$, and $\mathbf{p}_{s,k+1} = \text{diag}(\mathbf{P}_{s,k+1})$.

Additionally, $\odot$ is the Hadamard product, $\oslash$ is the element-wise or *Hadamard* division (see definition A.1.6), $(\,\cdot\,)^{(2)}$ is the *Hadamard* power of two (see definition A.1.7). This smoothing step is required to complete the KalCal-Diag algorithm.

3.3.4 Energy function on Gains

The final component to complete the discussion of sections 3.2.1 and 3.2.2 is the calculation of the energy function defined in §2.4.3. If the energy function can be implemented and shown to correlate with the accuracy of the calibration algorithms, the KalCal algorithms would have access to a well-defined set of optimisation and hyperparameter selection algorithms. An energy function will be defined for each KalCal algorithm, and its behaviour will be analysed alongside them in chapter 4 to assess the validity and practicality of its use. The full description of the energy function can be found in §2.4.3. The energy functions per algorithm are defined in algorithms 3.2.3 and 3.2.6. This section delves into the derivation of these algorithms for the KalCal algorithms. The first issue to address is the inclusion of the Kalman-Woodbury identity. Although it greatly improves computational complexity in the context of this calibration problem, it does not explicitly calculate the error covariance matrix. To overcome this issue, the result in eq. (2.91) can be used to calculate the required term explicitly. With that, the energy function calculations can be derived. Starting with KalCal-Full, the equivalent augmented energy function calculation using eq. (2.73) as a template, can be written as

$$\varphi_k = \varphi_{k-1} + \frac{1}{2}\ln\det(2\pi\underline{\mathbf{S}}_k) + \frac{1}{2}\underline{\mathbf{e}}_k\underline{\mathbf{S}}_k^{-1}\underline{\mathbf{e}}_k. \tag{3.65}$$

The expression can be written entirely in terms of $\underline{\mathbf{S}}_k^{-1}$ by applying the *determinant-inversion* property in lemma A.1.12 to get

$$\varphi_k = \varphi_{k-1} - \frac{1}{2}\ln\det(2\pi\underline{\mathbf{S}}_k^{-1}) + \frac{1}{2}\underline{\mathbf{e}}_k\underline{\mathbf{S}}_k^{-1}\underline{\mathbf{e}}_k. \tag{3.66}$$

Next, the centre term can be further expanded by using the *determinant of constant* property in lemma A.1.12 to become

$$\varphi_k = \varphi_{k-1} + N_{\text{bl}}\ln(2\pi) - \frac{1}{2}\ln\det(\underline{\mathbf{S}}_k^{-1}) + \frac{1}{2}\underline{\mathbf{e}}_k\underline{\mathbf{S}}_k^{-1}\underline{\mathbf{e}}_k. \tag{3.67}$$

As detailed in §2.4.3, the energy function is used in a MAP estimation process, which involves minimising this function with respect to model parameters. Thus, only the trend of the energy function over time is required, and the constant term can be removed in eq. (3.67). Furthermore, it also avoids unnecessary computational problems because the final energy function at $k = N_{\text{time}}$ would contain the value $N_{\text{time}}N_{\text{bl}}\ln(2\pi) \approx 1.84N_{\text{time}}N_{\text{bl}}$. The final component that needs addressing is the determinant operation. For KalCal-Full, the underlying information of the error covariance matrix must be preserved, and thus, to assist with the computation time, the *Cholesky decomposition* process is used. This procedure, denoted $\text{chol}(\,\cdot\,)$, finds a lower-triangle matrix $\underline{\mathbf{L}}_k \in \mathbb{C}^{2N_{\text{bl}}\times 2N_{\text{bl}}}$ such that any matrix, e.g. $\underline{\mathbf{S}}_k^{-1}$, can be rewritten as $\underline{\mathbf{S}}_k^{-1} = \underline{\mathbf{L}}_k\underline{\mathbf{L}}_k^H$. If $\underline{\mathbf{L}}_k = \text{chol}(\underline{\mathbf{S}}_k^{-1})$ can be found, then the determinant of

the inverse covariance term would be equal to twice the determinant of $\underline{\mathbf{L}}_k$. Furthermore, the determinant of a lower-triangle matrix is equal to the product of squares of the diagonal elements, that is

$$\det(\underline{\mathbf{S}}_k^{-1}) = 2 \prod_{i=1}^{N_{\text{bl}}} [\underline{\mathbf{L}}_k]_{ii}^2 \tag{3.68}$$

If $\underline{\mathbf{l}}_k = \text{diag}(\underline{\mathbf{L}}_k)$ and noting the natural logarithm applied to the determinant, the final calculation of the energy function for the AEKF would be

$$\varphi_k = \varphi_{k-1} - \sum_{l \in \underline{\mathbf{l}}_k} \ln(l) + \frac{1}{2} \underline{\mathbf{e}}_k \underline{\mathbf{S}}_k^{-1} \underline{\mathbf{e}}_k. \tag{3.69}$$

This is the energy function calculation for KalCal-Full. To acquire the term $\underline{\mathbf{S}}_k^{-1}$, the expression in eq. (2.91) is used where the augmented Jacobian is entirely constructed to perform the calculation, i.e. no approximations are included. The computational complexity of this construction is not as great a concern since the goal of the implementation of KalCal-Full is to validate the theoretical framework of KalCal. However, the Cholesky decomposition assists with any numerical stability issues with the large error covariance matrix.

For KalCal-Diag, the calculation of the energy function is slightly different. Using eq. (2.91), the equivalent inverse error covariance vector would be

$$\mathbf{s}_k^{-1} = \mathbf{1} - \text{diag}\left(\mathbf{J}_k \mathbf{P}_{f,k} \mathbf{J}_k^H\right) - \text{diag}\left(\widetilde{\mathbf{J}}_k \mathbf{P}_{f,k} \widetilde{\mathbf{J}}_k^H\right), \tag{3.70}$$

where the pseudo-covariances are zero because the gain and visibility signals are assumed proper. This expression is calculated explicitly using the filter covariance vector $\mathbf{p}_{f,k}$ as

$$\mathbf{s}_k^{-1} = \left[1 - \left(\frac{\sigma_{f,p}}{\sigma_R} |\hat{g}_q x_{pq}|^2 + \frac{\sigma_{f,q}}{\sigma_R} |\hat{g}_p x_{pq}|^2 \right) \middle| (p,q) \in \mathbb{B} : \begin{cases} \hat{g}_p, \hat{g}_q \in \mathbf{m}_{p,k} \\ \sigma_{f,p}, \sigma_{f,q} \in \mathbf{p}_{f,k} \\ x_{pq} \in \mathbf{x}_k \\ \sigma_R^2 \in \mathbf{R}_k \end{cases} \right]^T. \tag{3.71}$$

This construction also includes the prediction mean, $\mathbf{m}_{p,k}$, predicted visibility vector, $\mathbf{x}_k$, and the absorbed measurement covariance, $\mathbf{R}_k$, from the whitening transform (see §3.3.1). The operator $|\cdot|$ is the absolute value operator. The same steps as in eqs. (3.65) to (3.67) are applied for the energy function of KalCal-Diag, but using the widely-linear form of eq. (2.73). Applying lemma A.1.14, the determinant of the diagonal $\mathbf{S}_k$ would be equal to the product of the diagonal elements. Using the same steps used to arrive at eq. (3.69) and noting that $\mathbf{s}_k^{-1} = \text{diag}(\mathbf{S}_k^{-1})$, the energy function for the WEKF is

$$\varphi_k = \varphi_{k-1} - \sum_{s \in \mathbf{s}_k^{-1}} \ln(s) + \left(\mathbf{e}_k \odot \mathbf{s}_k^{-1}\right)^H \mathbf{e}_k, \tag{3.72}$$

The energy function used for KalCal-Diag is expressed in terms of vector operations. It is much faster to compute than eq. (3.72) and more numerically stable. However, the approximations used may reduce the quality of information the energy function provides, which will be explored in chapter 4. The calculations of the energy function are discussed in sections 3.2.1 and 3.2.2, and their effectiveness will be evaluated in chapter 4. In particular, eq. (3.69) will assess the behaviour and validity of the energy function and eq. (3.72) will determine its practicality of such a calculation during calibration. This concludes the discussion of the KalCal algorithms and their extensions for the calibration problem defined in §3.1.1.

Chapter 4

Application

In the preceding chapter, two KalCal algorithms were developed to calibrate for time-only, antenna-based gains. These algorithms are KalCal-Full, which uses the AEKFS, and KalCal-Diag, which uses an approximated WEKFS. This fulfils the initial research goal of creating a calibration algorithm based on the theory of KFS. The next step is to evaluate the performance of these algorithms compared to solution intervals using QuartiCal. In addition, the viability of the KS and the energy function will also be assessed. This will be done within this chapter via a simulated calibration experiment based on the problem described in §3.1.1.

4.1 Simulation Setup

A simulated experiment was chosen to analyse the accuracy and robustness of the calibration algorithms in various settings. The following subsections describe the steps to construct and execute these calibration scenarios.

4.1.1 Problem Contextualisation

It is important to clarify the goals of this experiment, how it answers the proposed research questions and what the experiment does not account for. As discussed in §1.3.3, the goals are (i) to test whether the KF algorithm has its results improved by the KS within radio interferometric calibration. In addition, (ii) to verify if the energy function provides a proxy for the ground truth with respect to calibration solution accuracy. Then, overall, (iii) to conclude whether the KalCal approach, consisting of the KFS with the energy function, is a valid contender to address the issues of the solution intervals approach whilst maintaining the same performance. The first can be assessed by determining whether the accuracy and calculated uncertainty of the smoother's calibration solutions with respect to the simulated ground truth is improved overall compared to the filter's. Since the experiment is simulated, the ground truth gain signal is available, and therefore, the accuracy of each algorithm's solutions can be directly measured. The details of how this is done are discussed in §4.1.6. Likewise, for the third goal, the same approach will be applied to determine whether the KalCal solutions yield similar or improved calibration results against solution intervals. The simulation will be set up in such a way to best mimic physical conditions; however, a separate investigation should be conducted to investigate the applicability of KalCal to real data. Note that KalCal is derived around a simplified RIME that does not consider full visibility

correlations, e.g. all Stokes components, or DDEs, and for a single frequency channel. This is done deliberately to scrutinise the behaviour of the KFS within the results. Furthermore, this form of KalCal is designed around self-calibration and assumes sufficient external calibration has been applied. Again, future investigation is required to evaluate the KalCal framework outside of these constraints and in other calibration regimes. The specific details of this are given in §3.1.2.

In both comparisons, the evaluation of the generated solutions by the calibration algorithms will only be quantified in visibility space, i.e. there will be no measurement or analysis of the results within image space. This is done to reduce the complexity of the analysis within this dissertation and to focus on the core calibration algorithm aspects. Therefore, an improvement in gain calibration accuracy and uncertainty with respect to the true corrupting signal is the overall goal since the subsequently corrected visibilities will be closer to the truth. However, due to the often non-intuitive relationship between improving gain estimations and the overall improved image quality from corrected visibilities, a pragmatic relationship will be defined. Within astronomy, a common method for quantifying distortions and calibration errors within final image products is the *Strehl Ratio*, denoted S [26]. It is defined as

$$S \sim e^{-(2\pi\omega)^2}, \tag{4.1}$$

where ω is the *Root Mean Square Error* (RMS) of the wavefront in units of wavelength λ and $S \in [0, 1]$. It describes the correlation between the distortions in the wavefront of the signal compared to the quality of the final image. An ideal situation is when S is close to one which represents minimal errors in the final image. Conversely, calibration is poor if S is close to zero where the final image will contain severe image distortions and errors. The formal definition of the Strehl ratio deals with wavefront aberrations and is more appropriate to optical systems. For purposes of radio interferometry, the *Mahajan approximation* to the Strehl ratio is more practical. In a radio interferometer, $2\pi\omega$ directly relates to the RMS of the gain phase errors, i.e. the difference between the calibration solutions and the true phases, which will be denoted as RMS_ϕ. As such, eq. (4.1) can be re-written using the Mahajan approximation as

$$S \sim e^{-\text{RMS}_\phi^2}. \tag{4.2}$$

The visibilities are well-calibrated when $S \sim 1$, which is when $\text{RMS}_\phi \to 0$, i.e. when the accuracy of the calibrated phases are high. Furthermore, the visibilities are poorly-calibrated when $S \sim 0$, which is when $\text{RMS}_\phi \to \infty$, i.e. when the accuracy of the calibrated phases are low. For the experiment evaluations, the MSE is used to determine the accuracy of the resulting solutions. The MSE is proportional to RMS^2 of the gain solutions, which includes RMS_ϕ. Thus, if the MSE is minimised with respect to the gain solutions, then the overall accuracy of the solutions is high. Consequently, the accuracy of the gains solutions has a correlation to the improvement of the final image generated from the corresponding corrected visibilities via the relationship of the Strehl ratio to the MSE of the gain solutions. If the MSE = 0, i.e. perfectly estimated gains, then $S \sim 1$, i.e. well-calibrated system. If

the MSE is of the order of 0.01, then this correlates to well-calibrated system since $S \sim 0.99$. Therefore, if the measured MSE of the solutions equal to or less than this threshold, then they are already ideal [26].

The final goal of analysing the energy function also fits into this relationship. The MSE provides information not attainable in real calibration since it requires knowing the true underlying gain signal. This makes it difficult to optimise calibration algorithms; however, as discussed in §3.3.4, the energy function could potentially overcome this issue due to its correlation to the Bayesian evidence. If the energy function can be shown to display the same minimisation behaviour as the MSE, then minimising the energy function, which is possible to calculate in real calibration, equates to minimising the MSE of the true gains signal against calibration solutions. Since the minimisation of the MSE results in a higher Strehl ratio, then the minimisation of the energy function also results in a higher Strehl ratio. That is, by minimising the energy function during calibration by selecting appropriate priors, the Strehl ratio can be increased and image quality will improves. With this link between the evaluation of the results in this research and the Strehl ratio, the analysis will focus on the accuracy of the calibration solutions as the ultimate goal. Further details are provided in the following sections.

4.1.2 Simulated Observation Details and Sky Model

The MeerKAT telescope array was chosen for the experiment to simulate visibility measurements. All antennas in the interferometer will be used, resulting in $N_{\text{ant}} = 64$, and thus $N_{\text{bl}} = 2016$ total baselines and the simulated observing run was performed with a single channel at 1 GHz. Next, the emissions are assumed to be unpolarised and measured with a linear feed configuration, i.e. only XX and YY correlations are considered. Multi-channel, multi-correlation and direction-dependent calibration using KalCal is left for future research. As a result, the calibration process can be separated into two scalar calibrations for each component, also known as the diagonal case (see §3.1.1). The phase centre is set to[1] at RA = 11hr50min15sec and DEC = $-30^{\circ}27'43''$ (J2000). The scan duration is set to 1.5 hours with an integration time of 20 seconds, resulting in 270 independent observations on all antennas, i.e. $N_{\text{time}} = 270$. These time-related parameters are arbitrary choices meant to reduce the size of N_{time} and, therefore, the computational complexity of performing many calibration runs. Note, the experiment does not assess computational complexity, except between the KalCal algorithms, and there should be no impact on the calibration itself. It will affect uv-coverage (see fig. 4.1) and may affect the value of the SNR during imaging, but it will not change the meaning of the calibration experiment. An empty *Common Astronomy Software Applications* (CASA) measurement set was generated from the above scan parameters using the software package simms and then was populated with simulated visibilities.

The next step involves selecting a sky model to generate the associated visibilities. This process mimics simulation I in [20]: randomly generating an image of the sky containing multiple point sources with positions and flux values determined by probabilistic densities.

[1] The simulation is based on a suitable archival observation providing good uv-coverage. The actual coordinates are arbitrary for the purposes of our experiment.

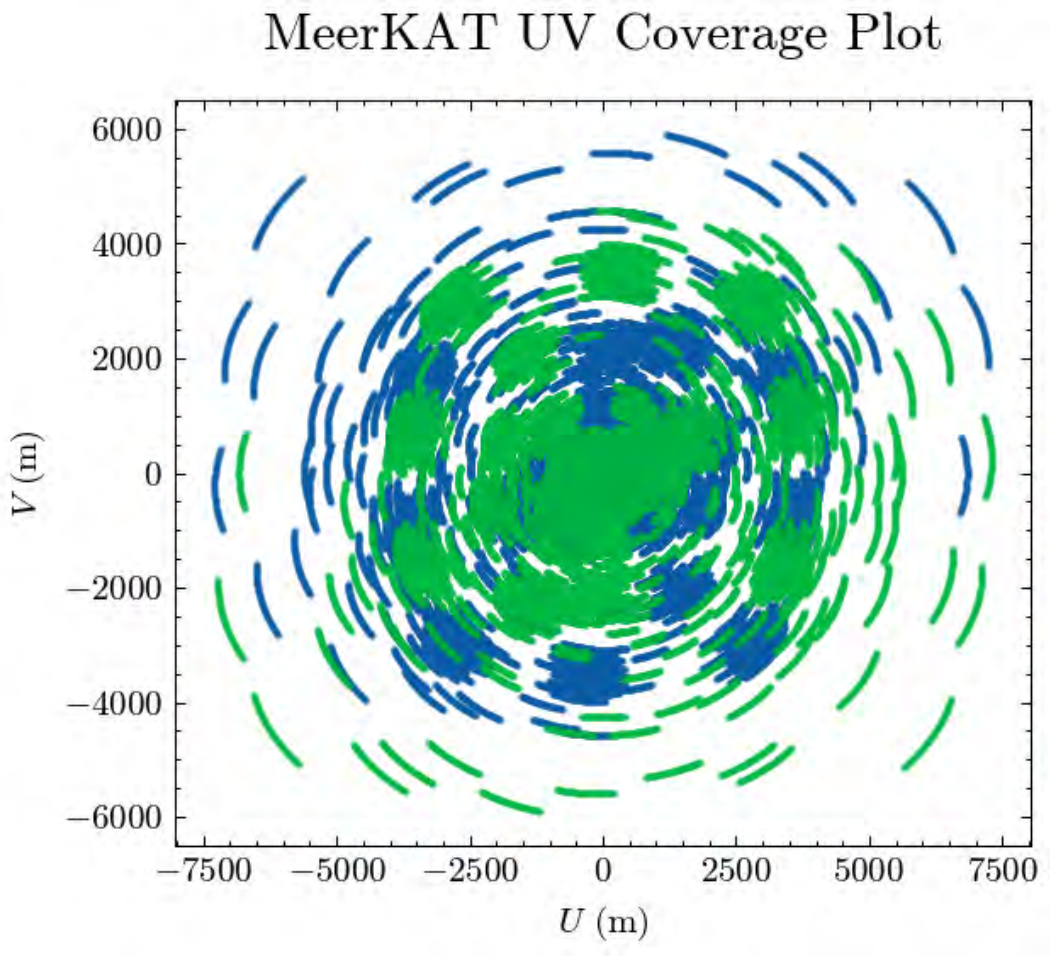

FIGURE 4.1: The total baseline *uv*-coverage of the simulated observation. The blue points signify the measurements taken throughout the scan, while the green points represent the corresponding conjugate baseline samples.

Several imaging parameters are required to generate these sky models. First, *Field-Of-View* (FOV) is set to 1°. As a simplification, the radio sources are treated as point-like. This is convenient as it allows them to be represented as single-pixel values within an image representation of the given sky where sources lie at pixel centres. Note, this is not a requirement for the KFS. Using the MeerKAT observation information above, an appropriate pixel size, Δ_c, and a grid shape, $N_x \times N_y$, are selected based on the *Nyquist sampling rate*[2] and what is computationally optimal for FFT calculations. The calculated pixel size is $\Delta_c \approx 3.82$ arcseconds with a square grid shape of $N_x \times N_y = 960 \times 960$.[3]

Second, sources are spread across the grid based on a source position density. This experiment randomly places the sources on the grid as pixels. To prevent two sources from being next to each other in the image, they are uniquely placed on a smaller grid of 80×80. Each of the 6400 cells on this grid corresponds to a 12×12 area in the original image. The pixel value is randomly placed within each block in a 8×8 grid in the centre of the 12×12 block. This process produces source pixels that appear randomly scattered within the original image, but at least 4 pixels exist between any two sources. This way, the issue of source confusion for purposes of flux measurement is avoided is avoided while preserving some degree of randomness. This process is visualised in fig. 4.2. Third, a source flux is assigned for each generated coordinate based on a given flux density. A typical density choice follows a *power law*[4] relationship. For this experiment, the flux will follow a *pareto* density and will be denoted as $\mathcal{P}(\alpha)$ where $\alpha \in \mathbb{R}$ is the exponent [20].

By selecting $\alpha = 4$ and taking a sufficiently large number of samples, which in this case was $N_{\text{src}} = 300$ sources, the results of this experiment were observed to be more reproducible for different choices of *random seeds*.[5] Once the samples were drawn and ordered, they were re-scaled so that the sum of the samples equalled a predefined total flux for the image. If x_i represents the i^{th} sample, let $s_x = \sum_{i=1}^{N_{\text{src}}} x_i$ be the sum of the sampled values, and let s_f be the desired total flux. The corresponding flux value f_i is calculated as

$$f_i = \frac{s_f}{s_x} x_i. \tag{4.3}$$

As a result, $s_f = \sum_{i=1}^{N_{\text{src}}} f_i$. For this experiment, the total flux was $s_f = 0.3$ Jy (300 mJy) which results in a peak flux of 1.07×10^{-2} Jy (10.7 mJy) and a minimum flux of 2.70×10^{-6} Jy (2.7 μJy). The average flux per source was ≈ 1 mJy. This final result is the complete simulated sky model that is assumed to be the truth for the experiment.

The percentage of flux modelled by the sky model will be defined as *Modelled Flux Percentage* (MP). Since the previous sky model represents the true sky, it would be considered a 100MP scenario, i.e. a complete sky model. This is what the interferometer observes at

[2] *Nyquist Sampling Rate*: The minimum sampling rate required to sample the largest frequency detectable by the interferometer, called the "Nyquist frequency". This rate is usually set at twice the Nyquist frequency [17], [26].

[3] These are calculated with a slightly under-sampled Nyquist rate and marginally larger grid shape to ensure no sources can be clipped by grid edges, regardless of how sources positions are distributed.

[4] *Power Law*: A proportionality between two variables whereby one variable is proportional to some power of the other [66].

[5] *Random seed*: The number used to initialise pseudo-random number generators in computers [67].

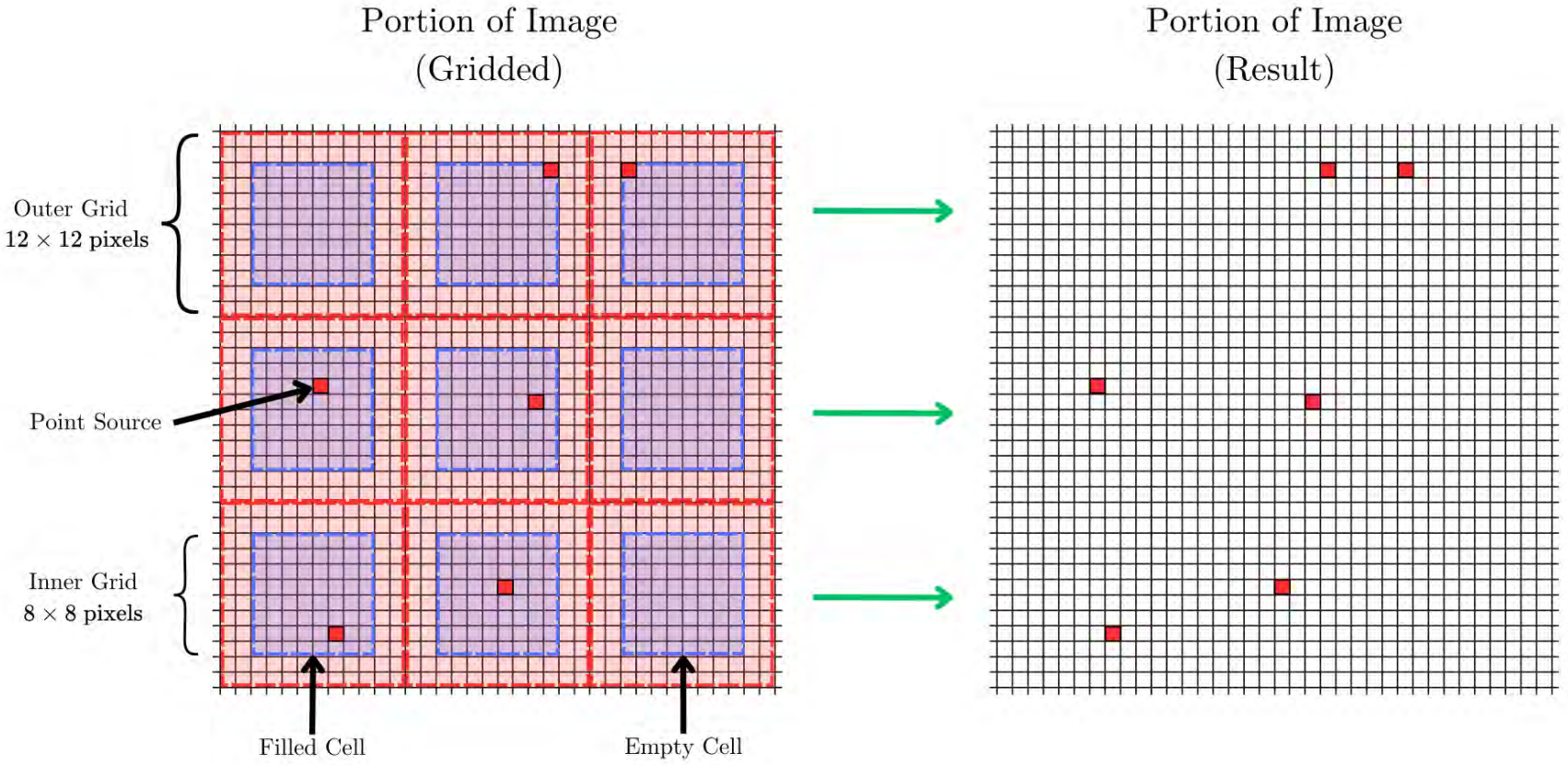

FIGURE 4.2: Visualisation of distributing source positions within the sky model. On the left, the image is divided into 12×12 adjacent grids (*red squares*), where each grid contains an 8×8 grid in its centre (*blue squares*). By only sampling one coordinate per blue grid in the gridded image, the chosen coordinates are guaranteed to be at least 4 pixels apart. The resulting image is on the right. No two sources share a blue grid; not all blue grids contain a source.

the given portion of the sky. Thus, to simulate noisy and corrupted observations from the same interferometer, the complete 100MP model is used to generate the true visibilities of the sources. Subsequently, these true visibilities are corrupted by the simulated gain signal and distorted with measurement noise sampled from a known Gaussian density. The result is a measured visibility signal that mimics the characteristics of real observations. At the same time, model visibilities are needed to perform calibration. However, in the calibration of real data, the quality of these model visibilities depends on the recovered sky model from external calibration using a bright calibrator source [27]. Therefore, the given observation conditions, like the SNR, and the residual calibration and deconvolution errors can decrease the accuracy of these derived model visibilities. Thus, to simulate a more realistic scenario calibration scenario, the above should be considered during model visibility generation.

In realistic observations, it is impossible to identify and categorise all the flux detected by the interferometer in a given portion of the sky before extensive calibration. Moreover, the SNR varies in strength. Generally, compared to the true sky, a certain percentage of flux remains unmodelled when creating the corresponding model visibilities [22]. Therefore, to create a more physical observation scenario, an 80MP sky model is chosen to generate the model visibilities to be used during calibration. The corresponding 80MP image was created by repeatedly removing the faintest source of the 100MP sky model until the remaining total flux was roughly 80% of the original total flux within the 100MP image. The sky models and images are depicted in fig. 4.3. The 80MP scenario adequately mimics the behaviour of real observations but, as an added benefit, allows for studying the effects of unmodelled flux on calibration solutions. The missing flux manifests as additional noise during calibration. If not accounted for, this can result in flux suppression and possibly calibration artefacts (see §1.3.1). Therefore, the robustness of a calibration algorithm against this issue is also a crucial component to analyse [20]. From these sky models, the necessary true and model visibilities were generated to be used within the experiment.

4.1.3 Gain Signal

This part of the simulation setup handles the creation of the gain signal that will be used to corrupt the simulated visibilities. It is this quantity that the calibration algorithms will attempt to model and estimate during calibration runs. Similar to §4.1.2, the creation of these gain signals is based on the same process within simulation I in [20]. It is difficult, if not impossible, to emulate a realistic gain signal representative of all observational scenarios. Consequently, this discussion is limited to an assumption that Gaussian processes can model the amplitude and phase component signals, the parameters of which are chosen to emulate their typical behaviour. If the gain signal for N_{ant} antennas is defined over time $t \in \mathbb{T}$ as the function

$$\mathbf{g}(t) = [\, g_i \mid i \in \mathbb{A} \,]^T = \left[g_1, ..., g_{N_{\text{ant}}}\right]^T, \tag{4.4}$$

then it can be decomposed as

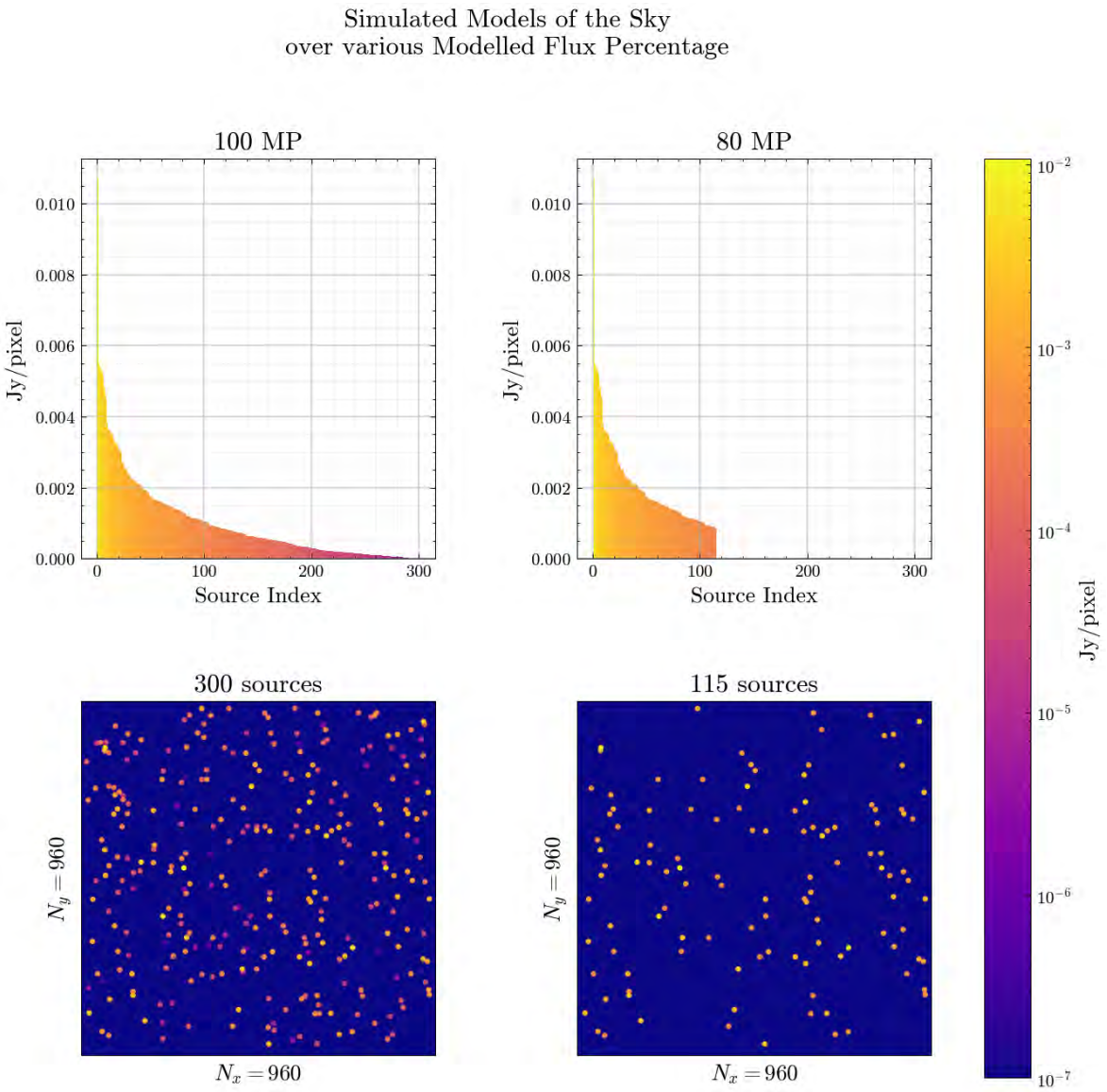

FIGURE 4.3: A representation of how the source flux and positions are distributed within the simulated sky models. The top row of the figure displays the density of sampled source fluxes that have been adjusted to a total constant flux for the experiment. The bottom row shows the sky model images with point sources placed so that no two sources are next to each other. The sources appear as enlarged points since the single pixels are not visible in the initial image. The source's colour indicates its flux value and is the same as in the density plot above. The left column is the complete or 100MP sky model, and the right column is the incomplete or 80MP sky model.

$$\mathbf{g}(t) = \mathbf{a}(t)e^{-i\boldsymbol{\phi}(t)}, \tag{4.5}$$

where $\mathbf{a}(t) : \mathbb{T} \longrightarrow \mathbb{R}^{N_{\text{ant}}}$ is the amplitude function and $\boldsymbol{\phi}(t) : \mathbb{T} \longrightarrow \mathbb{R}^{N_{\text{ant}}}$ is the phase function in radians. Generally, the amplitude signal varies slowly over time and at large scales, whereas the phase signal varies rapidly and on small scales [20]. As discussed within §1.3.1, the solution intervals approach requires separate calibration runs and, subsequently, different solution interval sizes for the amplitude and phase due to their differing signal dynamics. For KFS algorithms where the process noise kernel $\mathbf{Q}(t)$ is understood (see §2.2.5), both components behaviours can be modelled a single process noise kernel $\mathbf{Q}(t)$, i.e. a unique $\mathbf{Q}_k$ for each time step k. However, this is beyond the scope of this research. Instead, to best understand the dynamics of the energy function on the process noise, KalCal was created assuming $\mathbf{Q}_k$ is constant over $\mathbb{T}$ to simplify the analysis, i.e. it is stationary (see §2.1.3). Moreover, the gains are modelled and estimated as complex numbers, i.e. they encode the amplitude and phase. This allows for easy comparison between the various algorithms. The amplitude and phase functions will be generated and combined to simulate the true gain signal using eq. (4.5). The behaviours of the amplitude and phase can be mimicked by modelling each component as a real Gaussian process (see §2.2.5). Specifically, define each component's mean and covariance function over time and sample the corresponding signal for all $t \in \mathbb{T}$. For the amplitude signal, the mean function is set to unity; for the phase signal, the mean is set to zero. Both components will use the *squared-exponential* covariance function evaluated over $\mathbb{T}$ to get the covariance matrix

$$\mathbf{P}_{\text{SE}} = \left[\left[\sigma_{\text{SE}}^2 \exp\left(-\frac{(t_1 - t_2)^2}{2l^2} \right) \right]_{t_1,t_2} \middle| \, t_1, t_2 \in \mathbb{T} \right] \in \mathbb{R}^{N_{\text{time}} \times N_{\text{time}}}, \tag{4.6}$$

where l is the length scale and σ_{SE}^2 is the scale variance. Each element of $\mathbf{P}_{\text{SE}}$ encodes the variance, σ_{SE}^2, between two points in time based on the square distance between these points scaled by the value of l. If the mean over time is defined as $\mathbf{m} \in \mathbb{R}^{N_{\text{time}}}$, then a simulated signal can be created by sampling the real Gaussian density, $\mathcal{G}_{N_{\text{time}}}(\mathbf{m}, \mathbf{P}_{\text{SE}})$. For the amplitude signal, it is sampled from $\mathcal{G}_{N_{\text{time}}}(\mathbf{1}, \mathbf{P}_a)$ where $\mathbf{P}_a$ is eq. (4.6) with $l = 100$ and $\sigma_{\text{SE}}^2 = 0.2$. For the phase signal, it is sampled from $\mathcal{G}_{N_{\text{time}}}(\mathbf{0}, \mathbf{P}_\phi)$ where $\mathbf{P}_\phi$ is eq. (4.6) with $l = 10$ and $\sigma_{\text{SE}}^2 = 0.2$. For both component signals, this is repeated for each of the N_{ant} antennas and combined to form $\mathbf{a}(t)$ and $\boldsymbol{\phi}(t)$ for all $t \in \mathbb{T}$. Then, using eq. (4.5), the complex gain signal is produced using the newly created amplitude and phase signals. Samples of the resulting gain signal and its components are provided in figs. 4.4 and 4.5.

4.1.4 Visibilities

With the sky models generated in §4.1.2 and the gain signal generated in §4.1.3, various visibility data can be generated for the experiment. The first step is to create the *true* visibilities that represent the perfect measurements of the given sky. Since the 100MP sky model image is assumed to be the true sky, these true visibilities can be created by applying an FFT and a degridding (see §1.2.4) operation on the image given the observation and

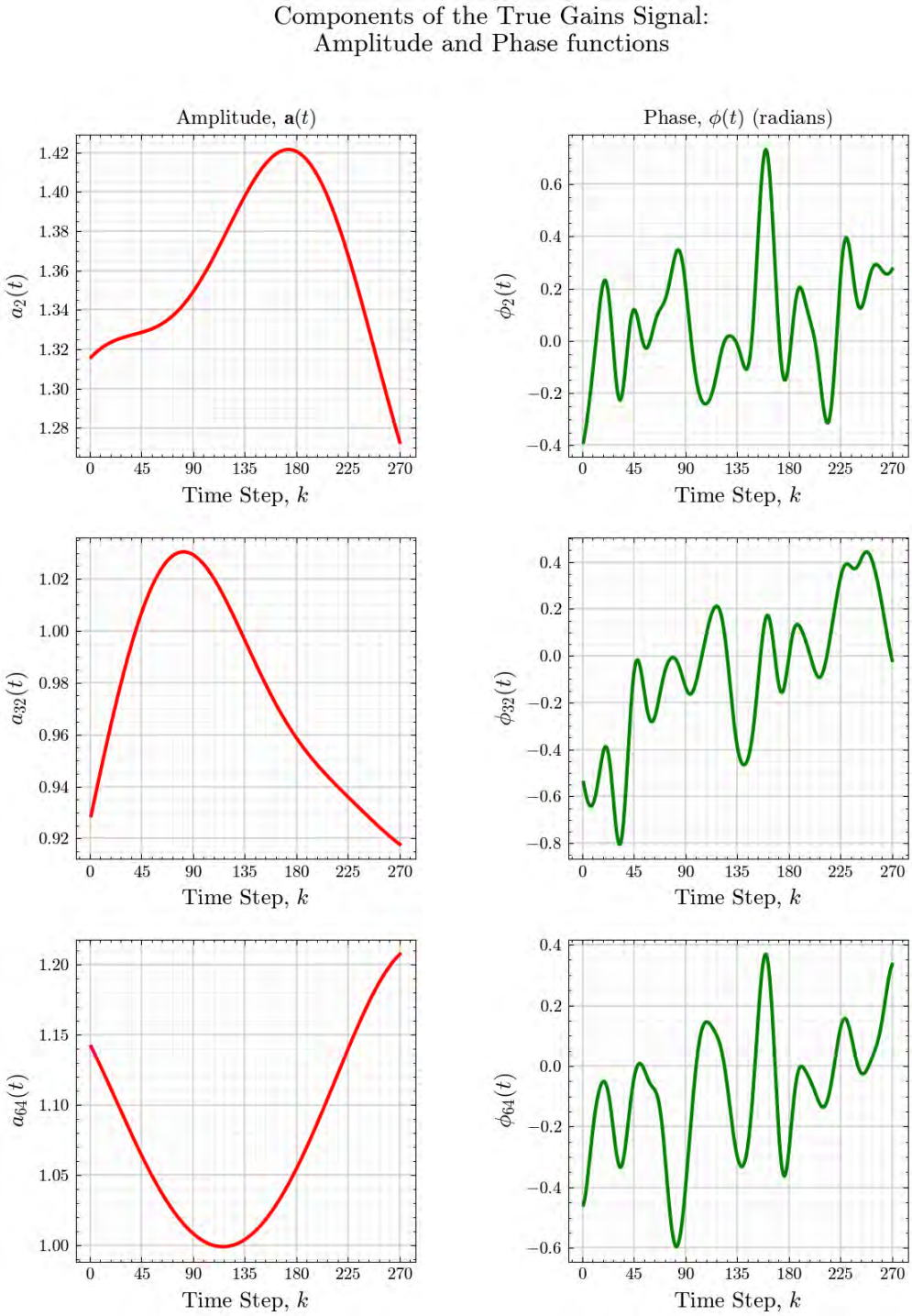

FIGURE 4.4: Amplitude and phase components of the simulated gain signal $\mathbf{g}(t)$ over time sampled from antennas indexed 1, 32 and 64. The left column represents the simulated amplitude signal $\mathbf{a}(t)$ (*solid red*) that has larger variations over longer scales centred around one. The right column represents the simulated phase signal $\boldsymbol{\phi}(t)$ in radians (*solid green*) that has smaller variations over shorter scales centred around zero. The combined signal $\mathbf{g}(t)$ can be found fig. 4.5.

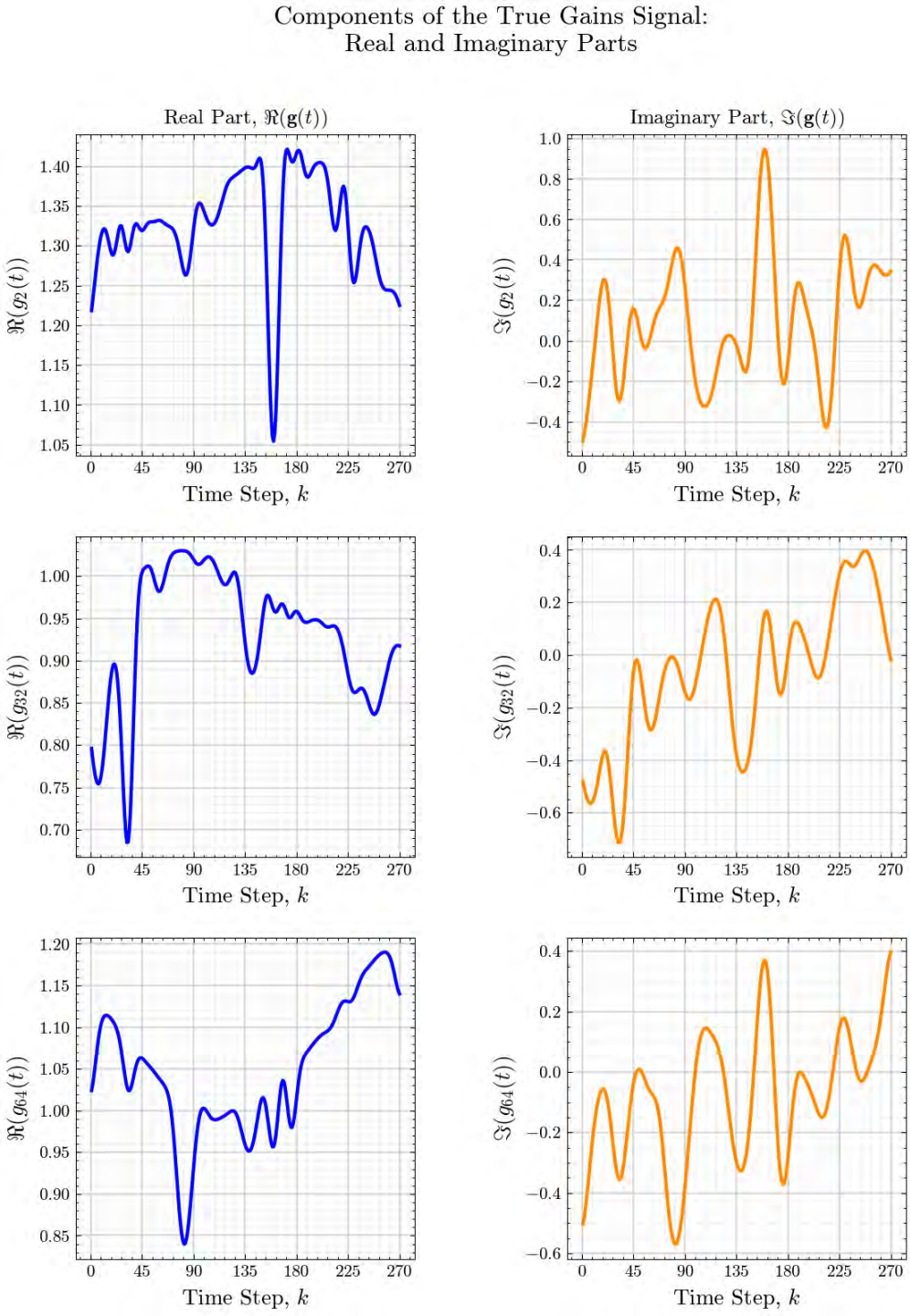

FIGURE 4.5: Real and imaginary parts of the simulated gain signal over time sampled from antennas indexed 1, 32 and 64. The left column represents the real part $\Re(\mathbf{g}(t))$ (*solid blue*) and the right column represents the imaginary part $\Im(\mathbf{g}(t))$ (*solid orange*). This gain signal is formed by combining the amplitude component $\mathbf{a}(t)$ and phase component $\boldsymbol{\phi}(t)$ fig. 4.4.

image details provided in §4.1.2.[6] Once the true visibilities are generated and saved in the measurement set, they are then corrupted using the simulated gain signal. For a single point in time, the corruption process consists of retrieving the pair of gain terms and the true visibility for a given baseline at that point in time. Then, using eq. (3.2), the true visibilities are corrupted using the gains. This is repeated for all baselines at all points in time. These visibilities will be referred to as the *clean* visibilities as they represent the measured visibilities without noise.

The last step is to apply noise to the clean visibilities. This is done by adding a sampled white Gaussian noise term to each corrupted visibility. This produces the *measured* visibilities that represent the noisy and corrupted measurements the radio interferometer would have recorded for a sky represented by the 100MP sky model image. Note that the final simulated terms are consistent with the RIME in eq. (1.8) and the measurement model created in eq. (3.5). For the experiment, two scenarios were established in terms of SNR to evaluate the accuracy and robustness of the calibration algorithms in different noise environments. This will assist in determining the effectiveness of the KalCal regularisation approach against solution intervals. Using eq. (3.5) as a guide, the two scenarios are labelled as the *high SNR* and *low SNR* environments.

Each is determined by the complex standard deviation of the measurement noise term within the visibility space, denoted as σ_R, within the measurement noise covariance $\mathbf{R}$ (see eq. (3.10)). For the high SNR environment, $\sigma_R = 0.015\sqrt{2}$ Jy ≈ 0.021 Jy (21 mJy) which, given the corresponding noisy measured visibilities and the actual generated noise terms, produces an SNR ≈ 3.95 dB, where dB is *decibels*. Note, the SNR is calculated here using the exact noise terms sampled from the measurement noise density and added to each visibility. For the low SNR environment, the measurement noise is scaled by four times with $\sigma_R = 0.06\sqrt{2}$ Jy ≈ 0.085 Jy (85 mJy) and has a corresponding SNR ≈ 0.39 dB. Note that both environments are not necessarily a realistic representation of appropriate levels of SNR in general and highly depend on the setup of a given observation. However, they are suitable to observe the behaviours of each calibration algorithm's ability to handle different noise levels. Furthermore, the only difference between each scenario is the value of σ_R. The sky models, gain signal, true visibilities and clean visibilities throughout are the same.

Once the measured visibilities are created per SNR scenario, the model visibilities are generated for each sky model. For the 100MP case, the original true visibilities are used as model visibilities, and for the 80MP case, the same process to derive the true visibilities is applied to the 80MP sky model image to produce the corresponding model visibilities. The last step is to produce the weights. These weights are required to construct the whitened visibilities components, such as the measurement vectors and Jacobians (see §3.3.1). For this experiment, it is assumed that the weights capture the measurement noise exactly, i.e. the σ_R is known. Furthermore, all weight terms are equal since the statistics are constant over all baselines and time. Thus, the weights per scenario are set as $w_{pq} = 1/\sigma_R$ for all $(p, q) \in \mathbb{B}$. This concludes all the simulated components required to perform calibration.

[6] This step was done using the `dirty2ms` function in the ducc0 package. The primary arguments of this function are the uvw-coordinates of the interferometer, the channel frequencies used, the angular pixel size, and the image of the sky.

4.1.5 Calibration runs

This section details the production of calibration solutions for the simulated experiment. The calibration runs were conducted using visibilities and weights described in the preceding sections. To understand the performance of the calibration algorithms with various input parameters, a one-dimensional grid or *line search* was performed on the primary input parameters while keeping all other parameters fixed. The result is a parameter function that maps the primary input parameter to the corresponding performance metric value. The calibration algorithms estimated the complex gains terms as established in §3.1. For QuartiCal, this line search is carried out on the time axis solution interval size, or *time-interval size* and is denoted as Δ_t. The solver was set to solve for a direction-independent G term using the `diag_complex` mode to match the conditions of the calibration problem set out in chapter 3. Expressly, QuartiCal was set to calibrate for the diagonal case. Furthermore, the convergence criteria was set to 10^{-7} and the iteration count to 100. This ensured that the estimation algorithms within were run sufficiently long enough to reach convergence with its iterative solvers. The choice of time-interval line search is inspired by simulations conducted in [20]. Specifically, the optimal solution interval sizes will tend to be smaller, given the observation parameters. Therefore, an irregular range of values is selected for this experiment best to appreciate the behaviour of the solutions as a function of solution interval size. Finally, The line search is broken into 50 points to search through the range between $\Delta_t = 1$ and $\Delta_t = 240$.

For the KalCal algorithms, the line search is conducted over the choice of process noise, σ_Q, for the evolution model. This value determines how constrained the recursive estimation algorithms are in determining appropriate solutions. That is, it controls how much the solution can deviate from the previous solution given the new information provided by the current measurement. Since the process noise is assumed constant over all antennas and time steps, and the measurement noise is assumed to be understood precisely, the line search will indicate how the KalCal results vary as a function of σ_Q. This is necessary to analyse the solutions against the energy function. For both KalCal-Full and KalCal-Diag, a logarithmic scale is applied to better appreciate the observed behaviours of the algorithms at different levels of σ_Q, especially at extremes. The logarithmic line searches are done over an evenly spread selection of 50 points from -3 to 0 (10^{-3} to 1). Noting the simulation setup, these calibration line search runs are performed for 100MP and 80MP sky models. In addition, the low and high SNR environments are run separately, totalling four separate calibration scenarios to examine and scrutinise these calibration algorithms. The exact means of doing this will be left for the next section. The naming convention in table 4.1 is used to distinguish between the four scenarios from this point onwards.

4.1.6 Evaluation Criteria

A few evaluation criteria must be established to quantify the behaviours of the KS and the energy function within this simulated calibration experiment. Referring back to the original research aims in §1.3, the objective is to determine whether the use of the KS can improve

Code	MP	SNR (dB)
H100	100	High (3.95)
L100	100	Low (0.39)
H80	80	High (3.95)
L80	80	Low (0.39)

TABLE 4.1: Scenario codes for different calibration scenarios in the simulated experiment based on the *Modelled Flux Percentage* (MP) and *Signal-to-Noise Ratio* (SNR) (in decibels, dB) present.

the quality of the KF calibration solutions and further regularise against extraneous measurement noise. Additionally, the behaviour of the energy function will be compared with the results of the KFS to determine if there exists a correlation between the minimum of the energy function and the optimal performance of the KalCal algorithms over σ_Q. If the KalCal framework can be shown to provide similar benefits to solution intervals while simultaneously providing a straightforward method for estimating optimal input parameters, like process noise, it would pave the way for the development of improved or "intelligent" calibration algorithms. The main aspects that will be examined within this experiment are the algorithms' solution accuracy, robustness, and input parameter sensitivity. Since the KalCal algorithms generate Gaussian densities as solutions, the comparison between gain solutions will be assessed against the means generated by their respective sub-algorithms. To a lesser extent, computational complexity is examined, but no in-depth analysis is provided. For measuring accuracy, the MSE is used. This well-known accuracy metric is helpful since it weights significant errors and provides a valuable indicator for the overall ability of the algorithm to estimate the underlying truth. The MSE is calculated using the solutions generated from the three calibration algorithms and the simulated gain signal. The smaller the MSE, the more accurate the solution is, indicating a better reconstruction of the underlying gain signal. It is calculated as follows:

$$\text{MSE} = \frac{1}{N_{\text{ant}} N_{\text{time}}} \sum_{p \in \mathbb{A}} \sum_{k \in \mathbb{T}} |g_{p,k} - \hat{g}_{p,k}|^2. \tag{4.7}$$

The above assumes, for time $k \in \mathbb{T}$ and antenna $p \in \mathbb{A}$, that $g_{p,t}$ is the true gains term and $\hat{g}_{p,k}$ is the corresponding calibration estimate of that term. The operator $|\cdot|$ is the absolute value operator. The corresponding curves are plotted by presenting the MSE of calibration algorithms against the line searches, and therefore, the optimality of different input parameters can be determined. "Optimality" in this context is simply the input parameter that minimises the metric under consideration. The associated plots will comprise most of the performance analysis for the calibration algorithms in the various scenarios. Specifically, by comparing the MSE curves over the four different calibration scenarios, the robustness and parameter sensitivity of the estimation algorithms can be analysed. Furthermore, the MSE curves are compared against the energy function results to determine if a correlation can be discerned between the two. To further enhance this analysis and validate the behaviour of

the signals, samples of the estimated gains are visualised for each algorithm. Secondly, in a real calibration scenario, eq. (4.7) cannot be calculated since the true underlying gains are unavailable. The next best option is to examine the *residual* visibilities given the estimated solutions, e.g. eqs. (3.26) and (3.40). Noting the measurement function given in eq. (3.9), the residual visibility at time $k \in \mathbb{T}$, $\mathbf{r}_k$, can be calculated as

$$\mathbf{r}_k = \mathbf{v}_k - \mathbf{h}(\mathbf{m}_k), \tag{4.8}$$

where $\mathbf{m}_k$ is the estimated gain vector from the calibration algorithm. Then, RMS on the visibility residuals over time is calculated as

$$\text{RMS} = \sqrt{\frac{1}{N_{\text{time}}} \sum_{k \in \mathbb{T}} \|\mathbf{r}_k\|^2}, \tag{4.9}$$

where $\| \cdot \|$ is the *complex-vector norm* operator. The RMS indicates how well calibration solutions minimise the residual visibilities. However, the RMS does not account for the overfitting of noise or unmodelled flux. Consequently, minimising the RMS will not necessarily correlate with minimising the MSE. Note that this value generally includes some form of uncertainty, e.g. measurement uncertainty (weights). However, since the measurement noise assumptions are the same across all baselines over time, there is no meaningful effect on the resulting RMS values. Consequent;y, the RMS values of the residual visibilities presented in §4.2 are unweighted. The resulting plots can be used to compare MSE of the gain solutions and the RMS of the residual visibilities, providing insight into the degree of overfitting and demonstrating the dangers of optimising solely with respect to the RMS. Brief mentions will be made of computationally related aspects where appropriate, specifically when discussing and comparing the different KalCal algorithms and their components. No such computational complexity analysis will be made between KalCal and QuartiCal since the experiment is not designed to make a fair and useful comparison. It will be a topic of future research.

4.2 Results and Analysis

Given the setup of the simulated calibration scenario, the simulated data products and the subsequent calibration runs, the results and analysis are presented below. When describing the results, QuartiCal and solution intervals will be used interchangeably. However, any critique and analysis are between KalCal and the solution intervals technique, not with the calibration software, QuartiCal, that implements it. Furthermore, KalCal-Full and KalCal-Diag produce the means and covariances for the prediction, filtering and smoothing densities. Any reference to their “estimates” will refer to the estimated means of their respective Gaussian densities. Unless stated otherwise, all KalCal results will be based on the outputs of the smoother. Larger and additional figures and tables are placed in Appendix B and are referenced in-text.

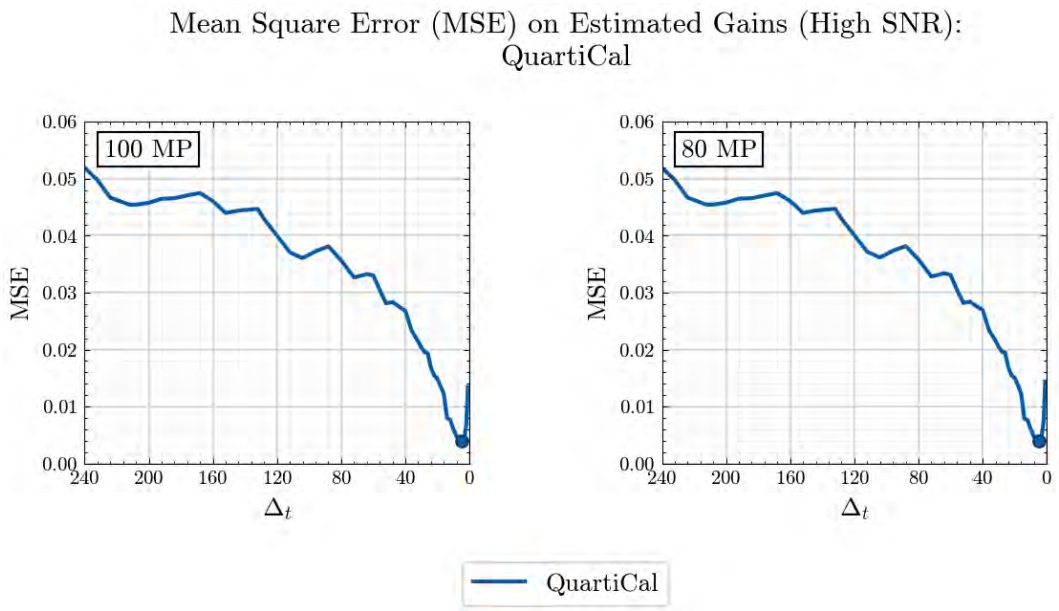

FIGURE 4.6: MSE of QuartiCal (*solid blue*) in high SNR scenarios H100 (*left*) and H80 (*right*). Dots on the curves represent the respective minima. Bottom x-axis is time-interval size Δ_t on a reversed linear scale.

4.2.1 Validation of Simulation

The initial phase of the analysis is to validate the simulated calibration experiment using QuartiCal. By examining QuartiCal's results and comparing it with the findings by Sob *et al.* [20], the validity of the experiment is discerned via the evaluation criteria in §4.1.6. Furthermore, QuartiCal will form a benchmark for determining KalCal's performance in this specific calibration problem. Refer to the MSE results in fig. 4.6. The H100 scenario does not suffer from issues experienced within practical situations, e.g. missing flux in sky models. Therefore, QuartiCal's primary task in H100 is to estimate the gain signal amidst moderate noise, the only obstacle to perfect reconstruction. In the H80 scenario, recovering an unbiased estimate of the gain signal is made more challenging by the incompleteness of the model. As a result, QuartiCal's robustness to model incompleteness is tested alongside the aspects already within H100. Figure 4.6 illustrates that both H100 and H80 scenarios exhibit MSE curves that share almost identical trends and optimal time-interval sizes. In both scenarios, a distinctive spike is observed in the MSE curve for smaller time-interval sizes, e.g. $\Delta_t = 1$. This trend indicates that solutions will fit the noise when Δ_t is small, which is expected. Examining the RMS results for the H100 scenario, shown in fig. 4.7, this point is supported since these solutions' corresponding RMS values are lower than the rest of the solutions. Note, these RMS results do not account for uncertainty and, therefore, are susceptible to overfitting. However, for H80, this is not the case. The noise induced by unmodelled flux would not be accounted for; therefore, marginal overfitting will occur for this portion of the total noise present. This results in higher RMS values for smaller Δ_t than the H100 scenario. The above points are visually substantiated in the second row of figs. B.5 and B.6.

In fig. 4.6, the optimal time-interval size is $\Delta_t = 5$ for both H100 and H80 scenarios, with corresponding MSE values of 0.00392 and 0.00395, respectively (see table B.1). This

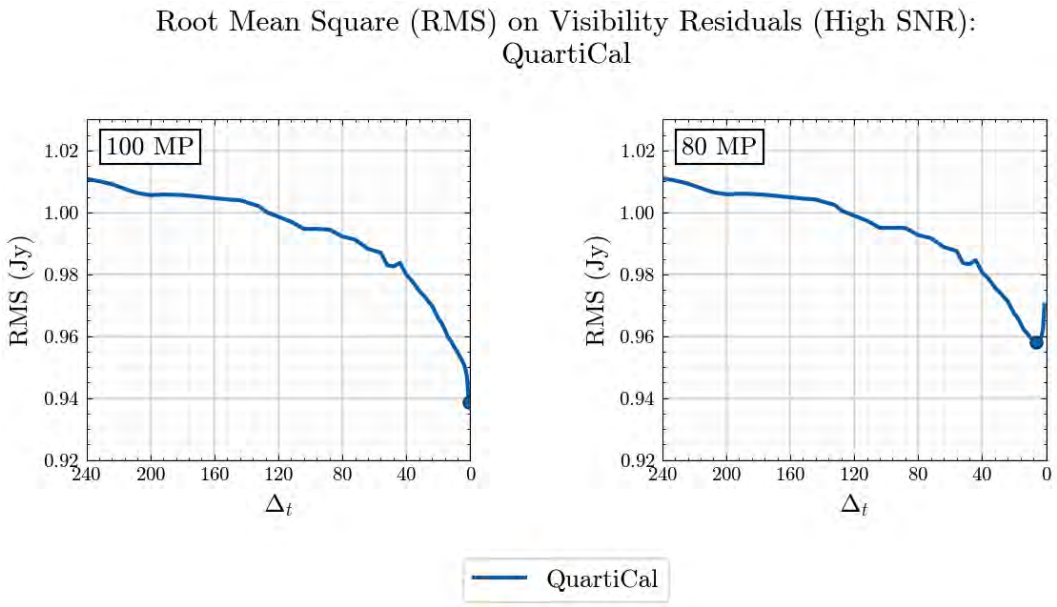

FIGURE 4.7: RMS of QuartiCal (*solid blue*) in high SNR scenarios H100 (*left*) and H80 (*right*). Units are in Jy. Dots on the curves represent the respective minima. Bottom x-axis is time-interval size Δ_t on a reversed linear scale.

aligns with the expectation that larger intervals are advantageous for noise averaging and regularising the solutions. However, due to the high SNR, a sufficiently small Δ_t can still capture gain variability effectively. This is illustrated in the first row of figs. B.5 and B.6. For larger Δ_t, a sharp initial increase in MSE is observed that transitions into a more gradual rise. This suggests that as Δ_t grows, the solutions cannot capture the gains' intrinsic variability as they increasingly focus on averaging out noise. As a consequence, the gain solutions will underfit the data. The solutions become near-constant as Δ_t tends to N_{time}. This trend is supported by the corresponding RMS values in fig. 4.7, which also increase in value as the interval sizes grow. These observations are also reflected within the third and fourth rows of figs. B.5 and B.6. The optimal RMS values are presented in table B.3.

The sensitivity of the MSE with respect to the choice of Δ_t is particularly significant in the high SNR scenarios. Smaller time-interval sizes can be used because the solution intervals can discern the gain signal amidst the noise. Moreover, the convexity of the MSE curves around the local minima is sharp, indicating a considerable sensitivity to time-interval sizes. Despite a distinct minimum, the non-smooth nature of the MSE curve could not be effectively used to optimise Δ_t towards this minimum MSE. In addition, this sharp convexity indicates that QuartiCal accuracy was sensitive to its input parameters. Moreover, the robustness against sky model incompleteness is evident when comparing the MSE curves for the H100 and H80 scenarios. There is minimal difference between their optimal MSE values; H80's optimum is only 0.8% larger than the H100 optimum, and the optimal Δ_t is equivalent. In summary, the calibration performance of QuartiCal in high SNR scenarios aligns well with the expectations associated with the solution interval approach.

In contrast to the high SNR scenarios, the MSE results for the low SNR scenarios, observed in fig. 4.8, present different behaviours. Here, QuartiCal faces the same obstacles as in H100

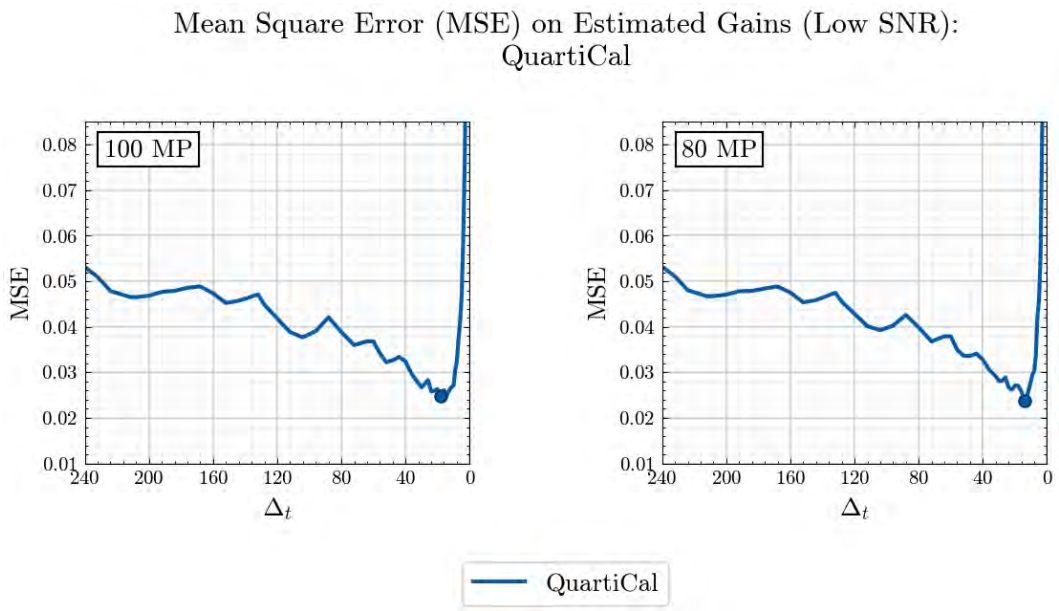

FIGURE 4.8: MSE of QuartiCal (*solid blue*) in high SNR scenarios L100 (*left*) and L80 (*right*). Dots on the curves represent the respective minima. Bottom x-axis is time-interval size Δ_t on a reversed linear scale.

and H80, but with higher noise. This uniquely challenges QuartiCal to maintain a small enough Δ_t to still capture gain variability while sufficiently averaging out the higher noise with larger Δ_t. Overall, the MSE curves are notably higher in the low SNR cases compared to the high SNR counterparts. This is expected since estimating the gain signal under a more prominent presence of noise leads to a decline in accuracy, i.e. lower MSE. According to these results, the drawbacks of using smaller solution intervals, like $\Delta_t = 1$, are more pronounced in these noisier conditions. Specifically, the algorithm struggles to differentiate signal from noise, compromising the estimates' accuracy and raising the MSE. This is most evident when comparing RMS values near to $\Delta_t = 1$ between H100 and L100 scenarios, as shown in figs. 4.7 and 4.9. These RMS values are significantly lower than other time-interval size solutions in L100 compared to H100. Moreover, the minimum RMS is less distinct in L100 than in H100. In the low SNR case, small solution intervals readily overfit the data, which is dominated by noise. This results in low RMS values but high MSE values. The same effect can be seen between the corresponding 80MP scenarios. This trend is visualised within the second rows of figs. B.7 and B.8.

Consistent with the elevated noise levels, optimal Δ_t values for QuartiCal shift toward larger time-interval sizes, as displayed in table B.2. Specifically, QuartiCal achieves highest accuracy in the L100 scenario at $\Delta_t = 18$ with an MSE $= 0.0247$, and in the L80 scenario at $\Delta_t = 14$ with an MSE $= 0.0238$. While the MSE curves are generally similar across both low SNR scenarios, minor differences are noticeable at $\Delta_t = 8, 16, 48$ and 104. This suggests that the unmodelled flux had a minor impact. Furthermore, the optimal MSE in the L100 scenario is 6.3 times greater than that in the H100 scenario. Similarly, the optimal MSE in the L80 scenario is about 6.01 times greater than in the H80 scenario. This emphasises the challenge of maintaining estimation accuracy at higher noise levels. Interestingly, the optimal

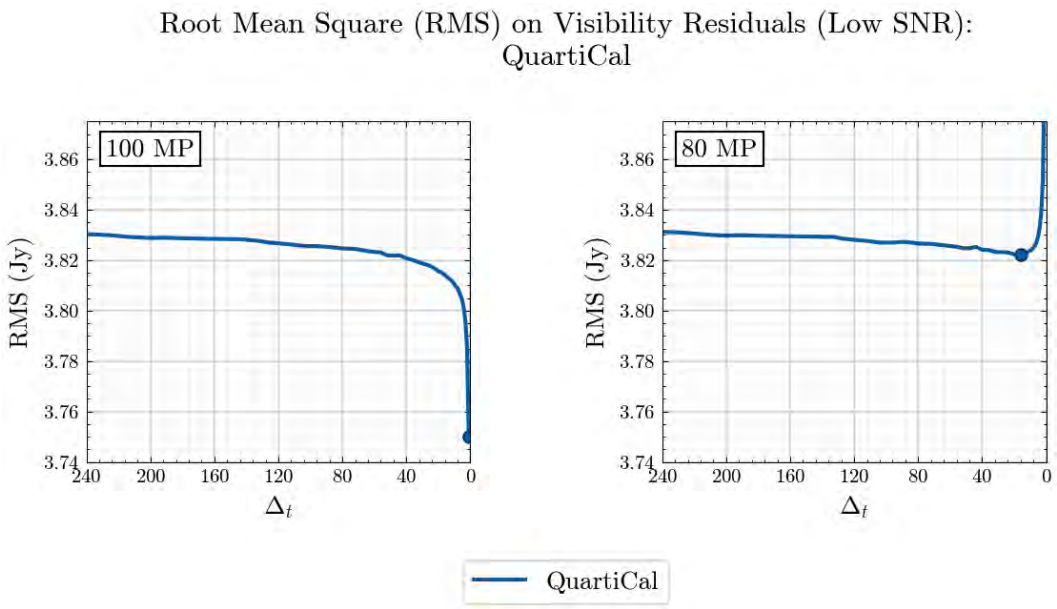

FIGURE 4.9: RMS of QuartiCal (*solid blue*) in low SNR scenarios L100 (*left*) and L80 (*right*). Units are in Jy. Dots on the curves represent the respective minima. Bottom x-axis is time-interval size Δ_t on a reversed linear scale.

MSE in the L80 scenario is 3.6% smaller than that of the L100 scenario. This is unexpected as it is assumed that calibration algorithms degrade in performance with an increase in model incompleteness. However, this could be attributed to the complex behaviour that emerges when observing the interactions between solution interval sizes, unmodelled flux and intrinsic variability of the gains. These optima are displayed in the first rows of figs. B.7 and B.8. When compared with the gain reconstructions of the high SNR optima, these solution curves are more erratic and do not track the underlying signal as effectively.

For solutions at larger Δ_t in fig. 4.8, the MSE increases at a slower rate compared to fig. 4.6. This reflects the challenge QuartiCal faces in balancing capturing gain variability while averaging out larger amounts of noise. For intervals close to $\Delta_t = N_{\text{time}}$, both high and low SNR scenarios produce similar MSE outcomes. When $\Delta_t = N_{\text{time}}$, the solution intervals reduce to solving a single average gain over the whole time axis. This similarity makes sense since both L100 and L80 are estimating the same underlying gains. This can be visualised by comparing the last rows of figs. B.5 and B.6 with those of figs. B.7 and B.8. As Δ_t decreases from this region in fig. 4.8, the accuracy deteriorates due to incorporating more noise into the solutions. This trend is corroborated within the RMS values in fig. 4.9, where larger time-interval solutions yield nearly identical RMS values. This indicates that the signal is too faint to determine as it is subsumed by the measurement noise, especially for smaller Δ_t. For a complete list of optimal RMS values in the low SNR scenarios, refer to table B.4. The resulting gain signals over various choices of Δ_t are presented within figs. B.7 and B.8.

Examining the parameter sensitivity within fig. 4.8, it is evident that QuartiCal is robust across the low SNR scenarios. The MSE curves tend to be more relaxed than the high SNR scenario; however, convexity around local minima is still relatively sharp for both L100 and L80. Again, although the MSE curve has a distinct minimum, its non-smooth nature makes it

challenging to determine an optimal value for the solution interval without resorting to brute force methods. Overall, these line search results and the sampled gain signal plots align with the expectations of solution intervals in low SNR scenarios. Therefore, coupled with the outcomes of the high SNR scenarios, the validity of this simulated calibration experiment is bolstered and should provide an appropriate environment to examine the results of the KalCal algorithms.

4.2.2 KalCal Analysis

The primary focus of this section is to assess the KalCal approach against the solution intervals technique. Specifically, using QuartiCal's result as a benchmark, analyse the results of KalCal-Full and KalCal-Diag. A detailed analysis of component estimates generated by each KalCal algorithm, e.g. predictor and filter estimates, is reserved for the next section. In a high SNR context, KalCal-Full's MSE results in fig. 4.10 obtain a minimum MSE at $\sigma_Q = 0.0687$ in both scenarios with MSE values of 0.00359 for H100, and 0.00489 for H80. These results indicate a comparable performance with QuartiCal, albeit with less robustness against model incompleteness. Furthermore, the RMS results in fig. 4.11 are indicative of the KFS's behaviour. When σ_Q is small, the prior model dominates, estimates underfit and RMS is high. Conversely, when σ_Q is large, the measurement model dominates, estimates overfit and RMS is low. Finally, if σ_Q is much larger, estimates are unconstrained and fail to track the underlying signal. Samples of the gain solutions can be visualised within figs. B.13 and B.14. The MSE curves display smoothness and are convex, in stark contrast with the solution intervals MSE curves. Overall, KalCal-Full has performed well compared to solution intervals within the high SNR regime.

Refer to fig. 4.12 for KalCal-Full's low SNR MSE results. Here, both L100 and L80 achieve their optimal MSE with a process noise of $\sigma_Q = 0.091$ with MSE = 0.0179 and MSE = 0.0198, respectively. Compared to QuartiCal, KalCal-Full provided better accuracy in the low SNR case. Again, only a minor effect attributed to model incompleteness is visible when comparing L100 and L80. The low SNR results in fig. 4.13 have the same properties as the high SNR results with one exception: there is more overfitting as σ_Q is much larger. This is because overfitting will only occur if process noise is large enough to capture these noise deviations. Samples of these gains signals can be found within figs. B.15 and B.16. KalCal-Full appears to be more robust to different calibration conditions compared to QuartiCal, especially in low SNR conditions. Furthermore, the MSE curves exhibit the same attributes as in the high SNR scenarios, i.e. they are smooth, and a distinct minimum exists. Given the above points, KalCal-Full has performed well compared to QuartiCal, suggesting the theoretical concepts behind KalCal are validated under simulated calibration conditions.

Shifting the focus to KalCal-Diag's MSE results, refer back to fig. 4.10. Interestingly, the resulting MSE curve is nearly identical to that of KalCal-Full in both high SNR scenarios. Moreover, KalCal algorithms share the same optimal process noise $\sigma_Q = 0.0687$ in both H100 and H80 with near-identical minimum MSEs. KalCal-Diag managed an MSE = 0.00369 in H100 and MSE = 0.00502 in H80. Therefore, KalCal-Diag also provides a minimum MSE similar to solution intervals in both high SNR scenarios, with solution intervals performing

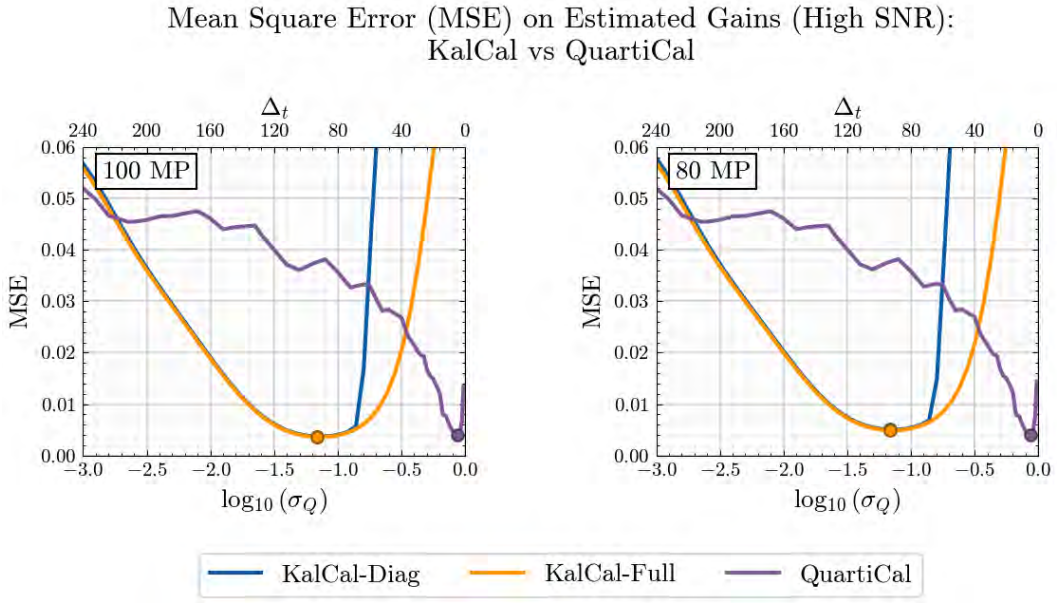

FIGURE 4.10: MSE comparison in high SNR scenarios H100 (*left*) and H80 (*right*) for QuartiCal (*solid violet*), KalCal-Diag (*solid blue*), and KalCal-Full (*solid orange*). Dots on the curves represent the respective minima for each algorithm. (*Bottom x-axis*) Process noise σ_Q for KalCal algorithms on a logarithmic scale. (*Top x-axis*) Time-interval size Δ_t for QuartiCal on a reversed linear scale.

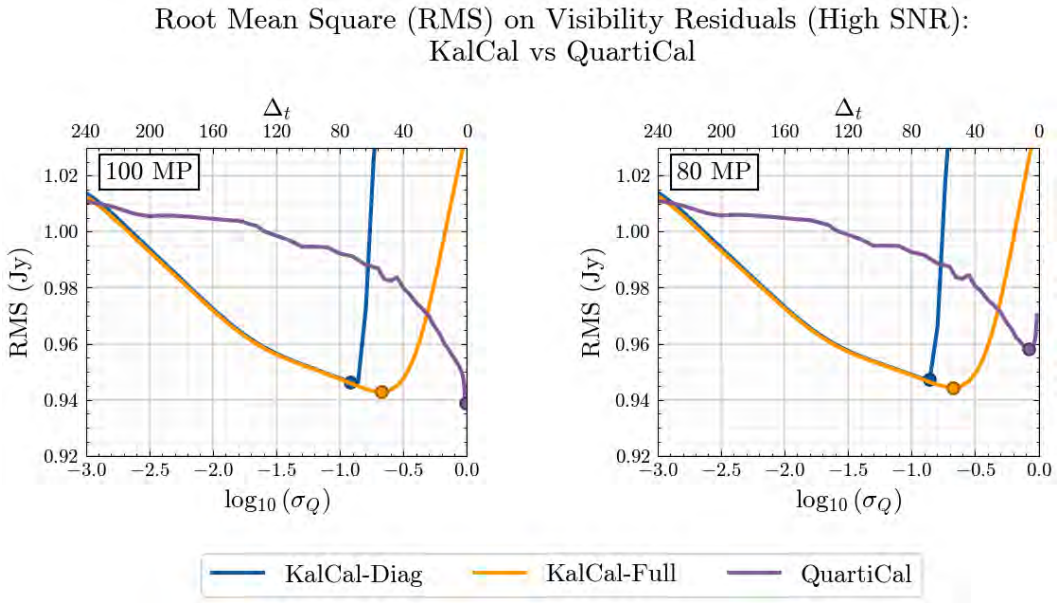

FIGURE 4.11: RMS comparison in high SNR scenarios H100 (*left*) and H80 (*right*) for QuartiCal (*solid violet*), KalCal-Diag (*solid blue*), and KalCal-Full (*solid orange*). Units are in Jy. Dots on the curves represent the respective minima. (*Bottom x-axis*) Process noise σ_Q for KalCal algorithms on a logarithmic scale. (*Top x-axis*) Time-interval size Δ_t for QuartiCal on a reversed linear scale.

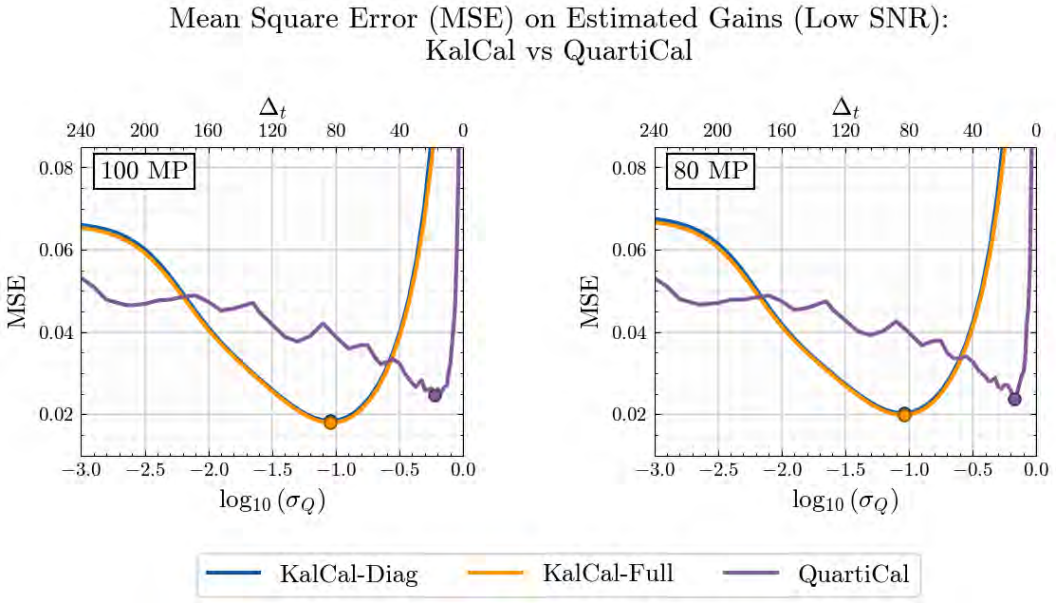

FIGURE 4.12: MSE comparison in low SNR scenarios L100 (*left*) and L80 (*right*) for QuartiCal (*solid violet*), KalCal-Diag (*solid blue*), and KalCal-Full (*solid orange*). Dots on the curves represent the respective minima for each algorithm. (*Bottom x-axis*) Process noise σ_Q for KalCal algorithms on a logarithmic scale. (*Top x-axis*) Time-interval size Δ_t for QuartiCal on a reversed linear scale.

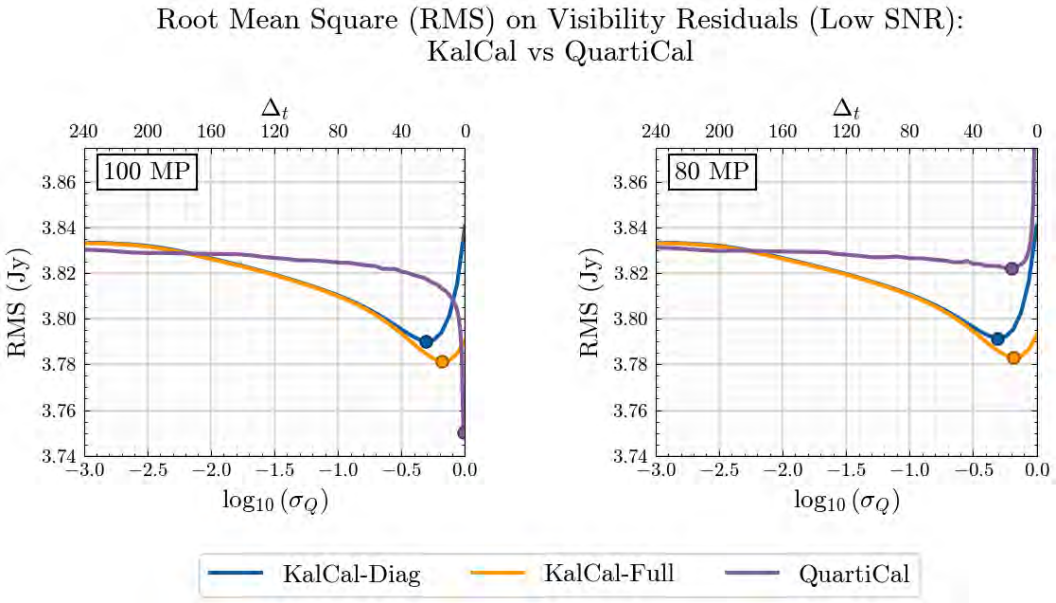

FIGURE 4.13: RMS comparison in low SNR scenarios L100 (*left*) and L80 (*right*) for QuartiCal (*solid violet*), KalCal-Diag (*solid blue*), and KalCal-Full (*solid orange*). Units are in Jy. Dots on the curves represent the respective minima. (*Bottom x-axis*) Process noise σ_Q for KalCal algorithms on a logarithmic scale. (*Top x-axis*) Time-interval size Δ_t for QuartiCal on a reversed linear scale.

marginally better in the H80 case. To visualise these gain solutions, see figs. B.9 and B.10. The similarities between the KalCal algorithm MSE results substantiate the approximation assumptions used to formulate KalCal-Diag, i.e. $J^H J$-computation is indeed diagonally dominant, and gain estimates are largely self-informed. The most notable feature of the KalCal-Diag results is the immediate spike in MSE after $\sigma_Q = 10^{-0.75} \approx 0.178$. The same feature is seen in the RMS results in fig. 4.11, and is best visualised in the last rows of figs. B.9 and B.10. Here, the solver stops tracking the signal and produces a non-physical estimated gain signal. This spike is indicative of numerical instability and the estimation algorithm breaking down. Further investigation is required to ascertain the root cause of the issue; however, the fact that this occurs for KalCal-Diag and not KalCal-Full suggests the problem lies in the approximations applied to the former. Despite this region of instability, the MSE and RMS results indicate that KalCal-Diag does indeed provide comparable performance to solution intervals as well. Moreover, KalCal-Diag is also robust in high SNR conditions, with only minor degradation in performance when the model is incomplete. Lastly, analysing the computation times per calibration run between the KalCal algorithms in their respective line searches, KalCal-Diag was shown to be, on average, a substantial three orders of magnitude faster than KalCal-Full. This shows that KalCal-Diag is pragmatic while maintaining appropriate accuracy.

The final aspect to assess is the behaviour of KalCal-Diag in low SNR conditions. Refer to KalCal-Diag's MSE results in fig. 4.12. KalCal-Diag achieves a minimum MSE with a MSE = 0.0184 for L100 and MSE = 0.0203 for L80 with the same process noise $\sigma_Q = 0.091$. Again, the MSE curves are near-identical to those of KalCal-Full, sharing the same optimal σ_Q and extremely similar minimum MSEs. The same pattern extends to the RMS results in fig. 4.13 and the gain signals displayed in figs. B.11 and B.12. Interestingly, the most notable "missing" feature is the spike in MSE observed in the high SNR scenarios. KalCal-Diag does not exhibit any numerical instability, and the algorithm does not break down, as justified in the last rows of figs. B.11 and B.12. Again, the details of this feature are left for future research. However, the lack of an unstable region within the low SNR results suggests the cause might also be related to the noise assumptions. Like KalCal-Full, KalCal-Diag demonstrates marginally better minimum MSEs compared to QuartiCal. Model incompleteness appears to impact KalCal-Diag's results, but the effect is insignificant. Finally, with the same computation time analysis presented earlier, it was found that, in these low SNR scenarios, KalCal-Diag was also three orders of magnitude faster than KalCal-Full. Therefore, given the above analysis of the KalCal algorithm results, both performed well compared to solution intervals. Except for the aforementioned instability, KalCal-Diag and KalCal-Full have been shown to behave similarly. However, KalCal-Diag is substantially faster and presents a more practical alternative to solution intervals.

4.2.3 Component Solutions Analysis

With the analysis of the KalCal algorithms, a deeper investigation is provided into the component estimates associated with each. The aim is to observe the effect of including the KS

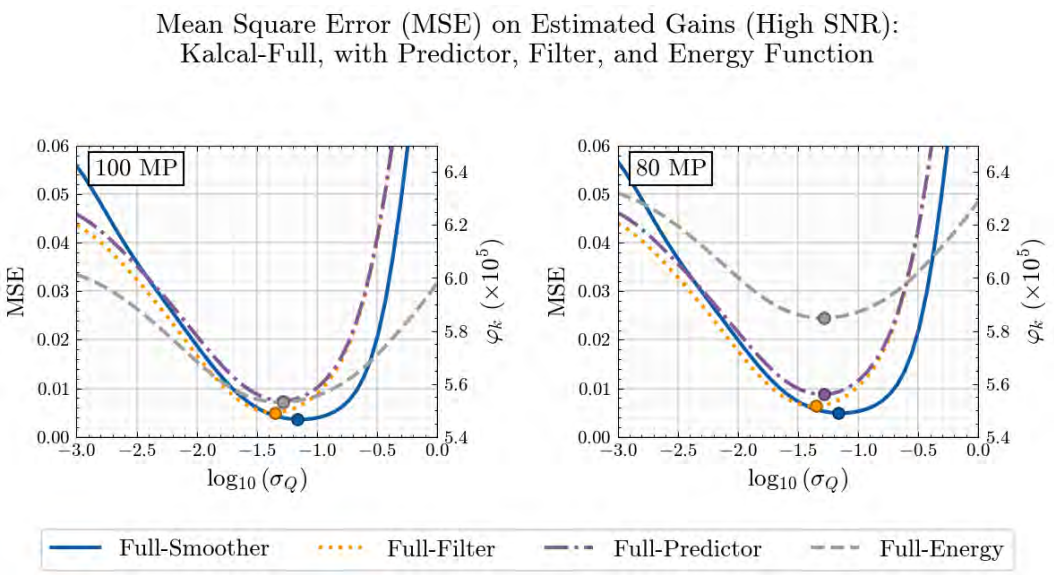

FIGURE 4.14: MSE results in the high SNR scenarios H100 (*left*) and H80 (*right*) for different components of KalCal-Full: Full-Predictor (*dot-dashed violet*), Full-Filter (*dotted orange*), Full-Smoother (*solid blue*), and Full-Energy (*dashed gray*). Dots on the curves represent the respective minima. Process noise σ_Q is represented on a logarithmic scale along the bottom x-axis. Energy function φ_k values are represented on the right y-axis.

alongside the KF and whether it improved gain estimates. The three components generated by each KalCal algorithm are: (i) predictor estimates that calculate the current gain given the previous filter estimate, (ii) filter estimates that update the current predictor estimates given the current measured visibility, and (iii) smoother estimates that represent the smoothed version of the filter results. The predictor results are not as important as the other estimates. Still, they provide additional information into the dynamics of the KFS process, e.g. the evolution model's effectiveness and the energy function's behaviour. The primary comparison is between the filter and smoother estimates. In the following plots, quantities will be prefixed with either "Diag" or "Full" to indicate the algorithm with which they are associated (see §3.2). Refer to both KalCal algorithm MSE results in high SNR conditions in figs. 4.14 and 4.16. For both algorithms, the filter and smoother estimates are highly similar. For smaller σ_Q, the filter marginally outperforms the smoother. This can be attributed to over-smoothing by the smoother, which underfits the gain signal and degrades accuracy. Examples of this can be seen in the second rows of figs. B.9, B.10, B.13 and B.14 where the output of the smoother is much flatter than the output of the filter.

For larger σ_Q, the smoother produces more accurate solutions than the filter. The difference is more significant for KalCal-Full than KalCal-Diag, and the trend only exists for KalCal-Diag up until the previously mentioned spike in MSE. The predictor solutions demonstrate that the filter estimates are reasonably accurate over time. This suggests that the chosen evolution model is appropriate, i.e. the filter doesn't dramatically alter the predictor estimates. Examining the minima, it is evident that all components provide similar minimum MSEs with respect to MSE with relatively similar choices of σ_Q. The smoother generates the best solutions for both algorithms in the high SNR scenario. This can be attributed to

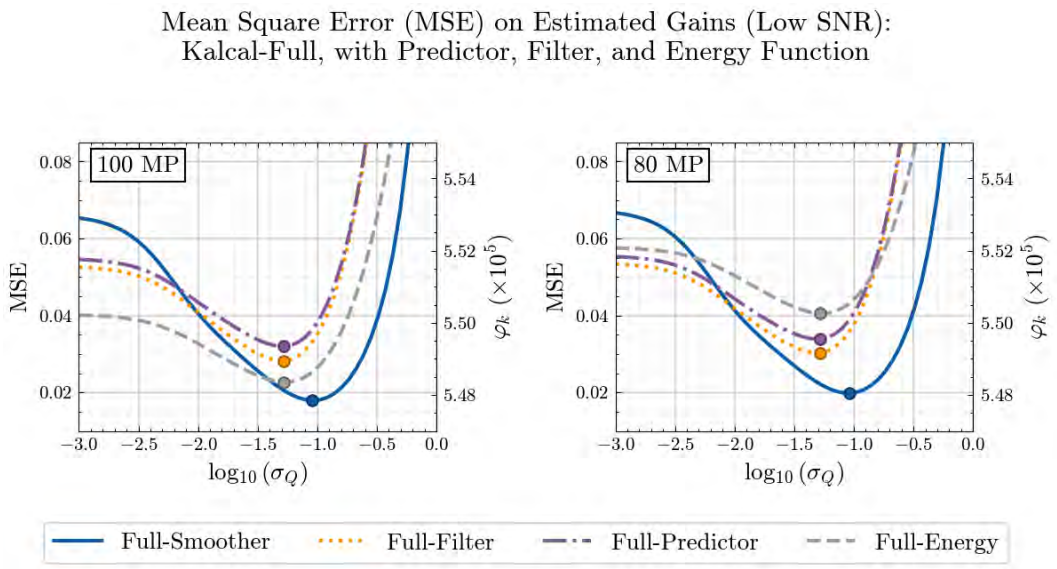

FIGURE 4.15: MSE results in the low SNR scenarios L100 (*left*) and L80 (*right*) for different components of KalCal-Full: Full-Predictor (*dot-dashed violet*), Full-Filter (*dotted orange*), Full-Smoother (*solid blue*), and Full-Energy (*dashed gray*). Dots on the curves represent the respective minima. Process noise σ_Q is represented on a logarithmic scale along the bottom x-axis. Energy function φ_k values are represented on the right y-axis.

including all the measurement information; each filter estimate only knows about the past. However, the difference between the filter and smoother's minimum MSE is marginal, suggesting that the KS only provided minor improvement in high SNR conditions. This makes sense because there is less noise, so there is less need to mitigate against overfitting. However, since the KS is computationally inexpensive, there is no reason not to use it, even in regimes where it offers little to no improvement. The corresponding RMS results for each KalCal algorithm's component estimates in high SNR scenarios are presented in figs. B.1 and B.3 and table B.3.

A different trend is observed in the low SNR conditions. The KalCal algorithms' component MSE results are found within figs. 4.15 and 4.17. For both KalCal-Full and KalCal-Diag, a notable difference is observed where the smoother produces much better results than the filter, except for smaller σ_Q where over-smoothing occurs. This is expected due to the increased level of noise present. The higher noise adjusts the measurement weights to suppress the influence of the measurement information. Therefore, updates to predictor estimates are not as informative as before, with better filter estimates only existing towards the end of the time frame. However, the smoother overcomes this issue by propagating this vital information backwards in time to inform older estimates and ultimately produce a superior solution. Note, with the increase in noise, the filter and smoother require larger process noise to find their respective minimum MSE.

From the smoother's perspective, it only needs to handle the discrepancies between the predictor and filter results. In the high SNR conditions, these differences were minimal and therefore, less smoothing was required, i.e. only a small process noise was required. However, in low SNR cases, the differences are much more significant and therefore, more regularisation

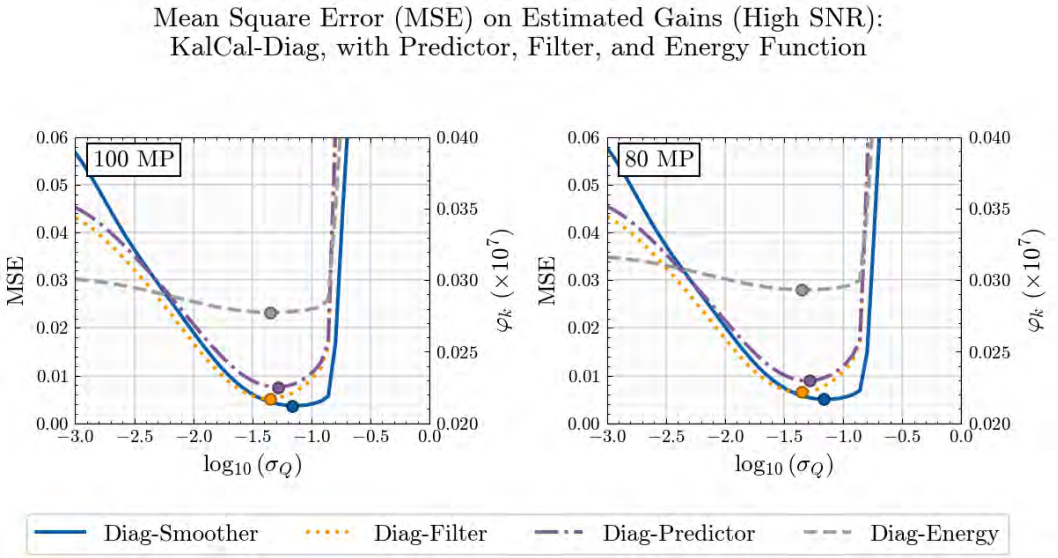

FIGURE 4.16: MSE results in the high SNR scenarios H100 (*left*) and H80 (*right*) for different components of KalCal-Diag: Diag-Predictor (*dot-dashed violet*), Diag-Filter (*dotted orange*), Diag-Smoother (*solid blue*), and Diag-Energy (*dashed gray*). Dots on the curves represent the respective minima. Process noise σ_Q is represented on a logarithmic scale along the bottom x-axis. Energy function φ_k values are represented on the right y-axis.

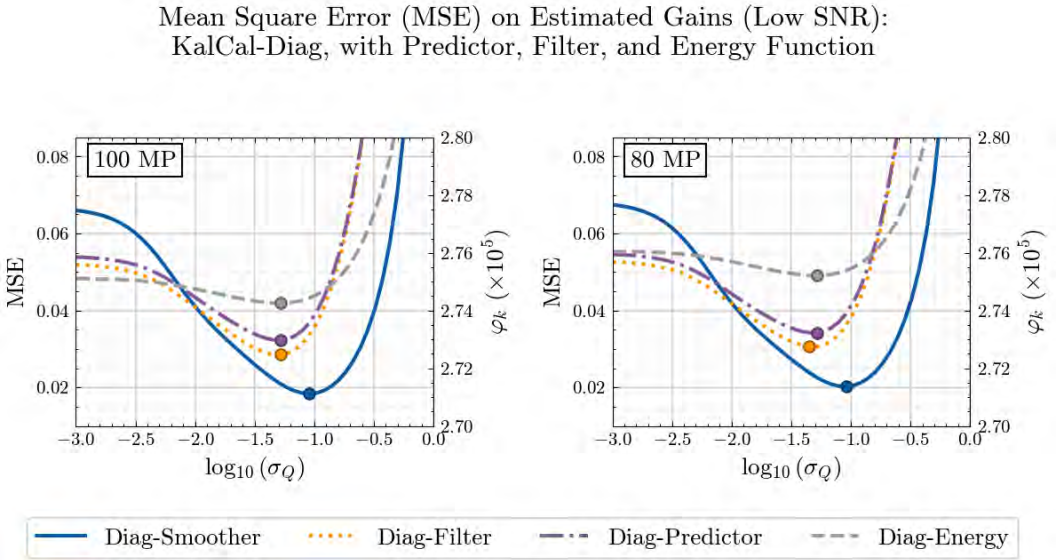

FIGURE 4.17: MSE results in the low SNR scenarios L100 (*left*) and L80 (*right*) for different components of KalCal-Diag: Diag-Predictor (*dot-dashed violet*), Diag-Filter (*dotted orange*), Diag-Smoother (*solid blue*), and Diag-Energy (*dashed gray*). Dots on the curves represent the respective minima. Process noise σ_Q is represented on a logarithmic scale along the bottom x-axis. Energy function φ_k values are represented on the right y-axis.

is needed, which the smoother can provide by considering all the measurements. Achieving this requires a more prominent process noise to address these more significant discrepancies between the predictor and filter. This demonstrates why the KS was more successful in the low SNR in terms of MSE because these discrepancies were larger compared to the high SNR case. To examine the related RMS results of the KalCal algorithms' component estimates in low SNR conditions, see figs. B.2 and B.4 and table B.4.

Based on the above analysis, it is clear that the KS did improve upon the KF in terms of calibration solution accuracy for both KalCal algorithms as expected. It only provides a marginal improvement in the high SNR cases, which makes sense because there is little need for smoothing. However, it vastly improved the filter's estimates in the low SNR cases. Since the KS is computationally inexpensive compared to the KF, it is highly suggested to incorporate this smoothing algorithm alongside the KF when developing KFS-based calibration algorithms. At worst, the smoother provides the exact solutions as the filter, and, at best, it provides substantially improved solutions. Moreover, there may be ways to improve these results by using different KFS variants or a more sophisticated state-space evolution model. This is left to future research.

4.2.4 Energy Function Analysis

The final aspect of this investigation is an analysis of the energy function as a proxy for MSE on gains. The energy function was already plotted against the MSE results in figs. 4.14 to 4.17. Note the specific values of energy function are not of particular interest. Instead, this section examines how closely the minimum of the energy function aligns with that of the MSE for the various components. For KalCal-Full, refer to figs. 4.14 and 4.15. The energy function minimum aligns exactly with the predictor minimum in the high SNR conditions. At the same time, the filter and smoother minima exist in small neighbourhoods around the energy function minimum. In low SNR conditions, the predictor and filter minima align exactly with the energy function. Again, the smoother minimum exists around the energy function minimum in a small neighbourhood. In both scenarios, the grouping of minima is relatively small, which indicates that the energy function correlates with the minimum MSE in both high and low SNR scenarios. This is a promising outcome because it demonstrates that the energy function does indeed track the optimal input parameters against the MSE of the solutions. Furthermore, the energy function tends to align exactly with the predictor minimum because the energy function is computed given the predictor residual and predictor residual covariance (see eqs. (2.49) and (2.50)). With this result, the theoretical expectations of the energy function, as discussed in sections 2.4.3 and 3.3.4, are validated. The energy function is a computationally tractable approach for deriving optimal input parameters, like the process noise, without requiring access to the ground truth, i.e. the accuracy of the solutions against the true underlying gain signal.

For KalCal-Diag, found in figs. 4.16 and 4.17, a similar outcome is observed. In high SNR conditions, KalCal-Diag's energy function minimum does not align exactly with any given minimum, like KalCal-Full, but rather the component estimates' MSE minima align in a small neighbourhood around the energy function. The spike in MSE is naturally also

seen within the energy function, but it does not impact the result significantly. For the low SNR results, the predictor minimum aligns exactly with the energy function minimum across both L100 and L80. The filter minimum aligns exactly for L100, but does not for L80 where it exists in a small region around the energy function minimum. The smoother does not align exactly in either case but exists in a small neighbourhood around the energy function's minimum. Several features can be ascertained from the above. First and most obvious, the predictor aligns exactly with the energy function due to their innate relationship. Second, the convexity of KalCal-Diag's energy function local minima is flatter compared to KalCal-Full across all scenarios. This may suggest that KalCal-Diag's energy function calculation provided in algorithm 3.2.6 may not incorporate enough information to generate a clear minimum like KalCal-Full. As it appears in figs. 4.16 and 4.17, noticing a change in energy function values would be harder to discern due to its flatter curve. However, this minimum is still distinct, so parameter optimisation will still be possible. Moreover, the perspective portrayed in these figures might be misleading because (i) the range of the energy function curve differs per KalCal algorithm, and (ii) the MSE spike in the high SNR scenarios could visually skew what is happening at a localised level. Therefore, it is suggested that further investigation is required to assess this trend. Nevertheless, it is highly promising outcome. Finally, the only case where unmodelled flux has had some impact on the energy function is KalCal-Diag's results in L100 and L80. Specifically, the filter MSE minimum aligns with the energy function minimum in L100, whereas it does not in L80. Note this does not significantly impact this experiment; however, its existence suggests some underlying relationship between model incompleteness and the energy function's minimum. If model incompleteness can alter the energy function's calculations, then it is a trend that should be noted in future research to mitigate against the associated risks.

With the above analysis of the energy function against the KalCal algorithms, the experiment indicates there is a strong relationship between minimising the KalCal algorithm MSE results and minimising the energy function. With only minor differences, the approximated energy function calculation provides the necessary model performance information during a calibration run. Therefore, there is potential to develop pragmatic, adaptable and automated parameter optimisation techniques that can extend into real calibration scenarios where ground truths like the MSE on estimated gains are unknown. Despite this positive result, additional research in numerical and computational optimisation of the energy functions is required. During the same computation time analysis between KalCal-Diag and KalCal-Full discussed in §4.2.2, the calculation of their respective energy functions was also briefly noted. From those results, it was found that the energy functions in their current forms in algorithms 3.2.3 and 3.2.6 double the computation time of their respective calibration algorithms. The time taken to calculate the prediction and update steps of the KF was equivalent to the time taken to calculate the energy function. Even with KalCal-Diag's approximations and noted speedup, the computation of its energy function can still be considered expensive. While implementing the energy function calculation for these experiments, the biggest computational obstacle to overcome was efficiently calculating eqs. (3.34) and (3.49). Based on these observations, a more in-depth investigation into the computational complexity of

KalCal as a whole is required since the energy function calculation depends heavily on the KFS. Although not utilised within this discussion, the energy function is also plotted against RMS of residual visibilities for each component algorithm. For these results, see figs. B.1 to B.4 and tables B.3 and B.4.

Chapter 5

Conclusion

With the theoretical discussion in chapters 2 and 3, and the results presented in chapter 4, this conclusion aims to summarise the key findings of this thesis. The initial research questions were: (i) Is the KalCal framework a viable approach for accurate and robust calibration for current and future calibration demands? (ii) Does the KS offer significant improvement over the KF in calibration? (iii) Does the energy function provide a potential avenue to enhance the KalCal approach with automated hyperparameter optimisation capabilities? Furthermore, the limitations of the investigation, interesting insights and future work are also discussed.

5.1 Summary of Key Findings

Below is a recount of the results throughout this research and how they relate to the aims of this investigation.

5.1.1 The KalCal Approach

The KalCal framework is based around the KFS estimation algorithms. Based on the initial work of Tasse [11], the goal was to assess if this approach could match the accuracy and robustness that is provided by solution intervals but without the associated drawbacks [20]. Given the general theory provided in chapter 2, the KalCal-Full and KalCal-Diag algorithms were developed to solve the for scalar, time-only antenna gains as defined in chapter 3. KalCal-Full uses the AEKFS that makes no approximations at the cost of being computationally expensive. While it is not a pragmatic solution, it serves to validate the theory behind KalCal and provides a point of comparison. KalCal-Diag is designed to be more practical by utilising diagonal approximations to reduce the algorithm's computational complexity. The resulting algorithm is based on the approximated WEKFS and entirely uses vector operations to minimise computational complexity. The output of these algorithms is Gaussian densities in the form of their means and covariances. Therefore, inference can be conducted directly from KalCal's results, a feature that calibration algorithms based on the ML approach do not have access to without additional computational expense. Based on Tasse [11]'s approach, an essential step in KalCal's development was the inclusion of the SMW to handle the large matrix inversion. By applying it to the KF, the problem can be treated entirely in the state space. However, the equations were further simplified, and this final result was dubbed the Kalman-Woodbury Identity. It applies to KF algorithms in general and significantly reduces

KalCal's computational complexity and improves its overall utility as a calibration algorithm (see §2.4.4). An energy function was derived for both KalCal-Full and KalCal-Diag. These energy functions, much like the Bayesian evidence, provide a means to discriminate between hyperparameter selections.

5.1.2 Simulated Calibration Experiment

In chapter 4, the KalCal algorithms were evaluated against the solution intervals technique in a calibration experiment on simulated data. Using similar simulations to Sob *et al.* [20], this experiment aimed to determine whether the KalCal algorithms outperform the solution interval approach coupled with ML solvers in terms of accuracy and robustness against measurement noise and unmodelled flux. To accomplish this, these algorithms were benchmarked against QuartiCal, a calibration software suite that utilises solution intervals. To determine the relative accuracy of the approaches, the MSE of the KalCal calibration solutions were compared against those of QuartiCal in several scenarios. These scenarios included high and low SNR regimes and varying levels of sky model completeness. This analysis was further enhanced by examining the RMS error on the residual visibilities. Finally, the sensitivity of the various algorithms to their input parameters was assessed.

The simulated experiment yielded several important results. The behaviour of QuartiCal's solutions was found to be consistent with the analysis provided by Sob *et al.* [20]. This was crucial in ensuring that comparisons between the algorithms were meaningful. KalCal-Full successfully recovered calibration solutions and demonstrated similar accuracy to QuartiCal in high SNR conditions and marginally better accuracy in low SNR conditions. Unfortunately, KalCal-Full was more susceptible to model incompleteness than the solution interval approach. KalCal-Diag demonstrated similar behaviour to KalCal-Full with nearly identical minimum MSE across all scenarios. The only cause for concern was a spike in the MSE observed at large process noise values in the high SNR case. This could suggest numerical instability and warrants further investigation into the approximations and assumptions made by KalCal-Diag. Based on these results, KalCal-Diag provides similar benefits to solution intervals while also being computationally feasible. Therefore, there is potential to utilise the KalCal approach as an alternative to solution intervals in future calibration software.

5.1.3 Kalman Smoother and Energy Function

Following the analysis of the KalCal algorithm results, a separate analysis was provided to investigate the impact of the KS on the KF in calibration. The individual component estimates, namely the predictor, filter and smoother estimates, were compared to make this assessment. For both KalCal-Full and KalCal-Diag in the high SNR scenarios, the KS only provided a minor improvement over the KF. This was expected due to the low noise level; therefore, the KF performed adequately, i.e. did not require much smoothing. A different trend was observed in the low SNR scenarios. For both KalCal algorithms, the KS was able to generate far better solutions than the KF. This behaviour is consistent with the theory as the quality of the KF solutions is coupled to the amount of data it has seen. In the case of higher

measurement noise, much more data must pass through the filter before it begins generating accurate solutions. The KS rectified this issue by propagating important measurement information discerned later in the time frame backwards, improving older filter estimates. These results motivate the inclusion of the KS in any KF-based radio interferometric calibration algorithm. This is entirely possible as (i) the KS has much lower computational complexity than the KF within these algorithms, (ii) regardless of the level of SNR, the KS always generated better estimates than the KF. The only concerns noted in the analysis were the instability of KalCal-Diag in high SNR conditions and the risk of over-smoothing solutions if the input process noise was too small.

The last component of this experiment was an analysis of the behaviour of the energy function. Both the non-approximate energy function of KalCal-Full and the approximate energy function of KalCal-Diag were considered. The former, while computationally impractical, serves to validate the theory, while the latter provides a computationally attractive alternative. For KalCal-Full across all scenarios, the MSE minima of the various component estimates appeared in a small grouping around the energy function minimum. The results suggested a clear relationship between minimising the MSE on the gains and minimising the energy function. In KalCal-Diag's case, a similar outcome was observed. Although not as distinct as KalCal-Full, the energy function minimum also existed in a small grouping with KalCal-Diag's various component estimates' minima across all calibration scenarios. This indicated that the energy function remains relatively consistent with the optimal input process noise even after approximation. These results are especially compelling as the MSE on the gains is not available in real calibration scenarios. The energy function provides a smooth, differentiable function that can be used to optimise KFS-based calibration algorithms with automatic hyperparameter tuning. Utilising this powerful result is an excellent topic for future research.

5.2 Limitations and Future Research

Several limitations were encountered during the course of this research, and several topics require further investigation. This research has not addressed the computational viability of KalCal for large-scale calibration problems. Theoretically, the KFS promises to be much faster and have a smaller memory footprint than iterative methods. However, its recursive design may hinder its ability to scale in parallel environments. Specifically, since each solution depends on states around it, the possibility of parallelism is drastically reduced. In terms of extending KalCal itself, several avenues exist to improve upon this framework. The most promising improvement to consider is the *Iterated Extended Kalman Filter* (IEKF) [23]. Similar to iterative methods, the *Iterated Extended Kalman Filter* (IEKF) repeats the update step within the EKF to converge to an improved solution. Moreover, it could also improve energy function calculations based on the same residual information. Another version to consider is the *Ensemble Kalman Filter* (EnKF), which can be viewed as a statistical approximation of the EKF [68]. It is designed to improve the computational complexity of the large-scale KFS algorithms by approximating the Gaussian densities through sampling and

only partially calculating the covariance matrices. Each sample is computed in parallel and combined to calculate the associated sample mean and covariance desired through specialised operations. It is claimed that this overall process boasts lower computational complexity than traditional KFS approaches if the number of samples is low enough.

The KalCal framework can be adapted to various situations and only requires choosing the desired RIME for the measurement model and an appropriate evolution model. For example, introducing a more sophisticated evolution model, like the dual-evolution function approach by Tasse [64], can improve the regularisation of gain solutions with minimal effort. Moreover, the RIME in KalCal can theoretically be designed to account for both the polarised case and direction-dependent calibration. This would be ideal for future research to validate whether the KalCal extends to other forms of the RIME. Another important aspect not covered in this research is calibrating with flagged data. The current implementation of KalCal assumes there is no missing data. Therefore, further research is required to extend KalCal to handle flags in calibration scenarios. The results of this research are constrained to the direction-independent form of the RIME to be used with self-calibration. However, since the solutions created are ideal for self-calibration, it would be beneficial to investigate whether, with an appropriate KalCal setup, the same approach would apply to transfer calibration, especially for its on-line calibration potential. In addition, the visibility and gain models used have not been tested over varying correlation and frequency axes, and the analysis of the experiment results is limited to the visibility space. Therefore, the next appropriate future step would be to develop KalCal for a more general RIME that addresses these missing axes and potentially DDEs. The impact it has on the imaging step should be tested and, most importantly, KalCal must be evaluated in a real calibration scenario to finalise its suitability for calibration within radio interferometry.

Appendix A

Additional Mathematics Definitions and Derivations

A.1 Linear Algebra

Definition A.1.1 (Notation Styles). For the mathematical contents of this research, the following notation style is used to discern between *scalars*, *vectors*, *matrices*, (ordered and unordered) *sets*, and *functions* thereof:

- Symbols of *sets* are uppercase and blackboard bold weighted, e.g. $\mathbb{R}$ and $\mathbb{A}$. *Ordered sets* are defined with parentheses, e.g. $\mathbb{A} = (1, ..., N_{\text{ant}})$, and *unordered sets* are defined with braces, e.g. $\mathbb{Z} = \{..., -2, -1, 0, 1, 2, ...\}$.
- *Scalars* and *scalar functions* are italicised, for example, $x \in \mathbb{R}$ and $f(t) : \mathbb{R} \longrightarrow \mathbb{R}$.
- *Vectors* and *vector functions* are lowercase and bold weighted, e.g. $\mathbf{x} \in \mathbb{R}^n$ or $\mathbf{x}(t) : \mathbb{R} \longrightarrow \mathbb{R}^n$, for some $n > 0$, and are displayed using square brackets.
- *Stochastic processes* are equivalent to *ordered sets* of *vectors* and is more intuitive to denote as $\mathbf{x}_{1:N} = (\mathbf{x}_1, ..., \mathbf{x}_N)$.
- *Matrices* and *matrix functions* are uppercase and bold weighted, e.g. $\mathbf{X} \in \mathbb{R}^{m \times n}$ or $\mathbf{X}(t) : \mathbb{R} \longrightarrow \mathbb{R}^{m \times n}$, for some $m, n \in \mathbb{N}$, and are displayed using square brackets.
- *Probability densities* are special scalar functions that are uniquely denoted in uppercase calligraphic font, e.g. $\mathcal{CN}(\mathbf{m}, \mathbf{P}) : \mathbb{C}^n \times \mathbb{R}^{n \times n} \longrightarrow \mathbb{R}$ for some $n > 0$.
- Any other special functions or values utilise similar font styles to scalar functions and values with normal or italicised weighting.
- *Additive* and *multiplicative identities* are not formatted and are implied to match the dimensions required of its placement, e.g. an n-dimensional augmented Gaussian density with zero-mean is $\mathcal{CN}_{2n}(\mathbf{0}, \underline{\mathbf{P}})$ where $\underline{\mathbf{P}} \in \mathbb{C}^{2n \times 2n}$ and $\mathbf{0}$ is a zero vector of length $2n$.

In addition, labels and indices of these terms will be placed as comma-separated subscripts in the order of priority. Where possible, similar indices and labels are grouped without

delimiters, e.g. baseline (p, q) is displayed as $\mathbf{X}_{pq}$. The superscript is left only for exponential terms to avoid confusion.

Definition A.1.2 (Vector-builder notation). Similar to *set-builder notation*, the *vector-builder notation* describes how to build a vector. If the vector $\mathbf{x}$ is built using the ordered set of values $\mathbb{X} = (x_1, ..., x_n)$, where $n \in \mathbb{N}$, then this construction can be denoted with the new vector-builder notation below:

$$\mathbf{x} = [\ x \mid x \in \mathbb{X}\ ]^T = [x_1, ..., x_n]^T. \tag{A.1}$$

The convention is that it is built as a row vector to ensure the rules comply with remark A.1.3. To build via an indexing scheme, say for some indices $\mathbb{T} = (1, ..., n)$ with a scalar function $x(t) = x_t$, then another example of vector-builder notation would be

$$\mathbf{x} = [\ x_t \mid t \in \mathbb{T}\ ]^T = [x_1, ..., x_n]^T. \tag{A.2}$$

The element indices of $\mathbf{x}$ will follow the ordering of the ordered set used to create the vector. In addition, any indices not covered by the indexing scheme are assumed to point to zero-valued elements. This way, no additional rules are required to handle zero-valued elements, and the construction process is more concise.

Remark A.1.3 (Matrix-builder). Extending on the vector-builder notation in definition A.1.2, the construction of matrices can also be described. Particularly, to build the matrix $\mathbf{X} \in \mathbb{C}^{m\times n}$ for some $n, m \in \mathbb{N}$, the vector-builder notation can be used over the following example ordered sets:

- Given $\mathbf{x}_{1:m} = (\mathbf{x}_1, ..., \mathbf{x}_m)$, where for $i \in (1, ..., m)$, each element is an n-dimensional vector that can be written as

$$\mathbf{x}_i = \left[\ x_{ij} \mid j \in (1, ..., n)\ \right]^T = \left[x_{i1}, ..., x_{in}\right]^T. \tag{A.3}$$

  So, using the vector-builder notation, the matrix $\mathbf{X} \in \mathbb{R}^{m\times n}$ constructed by concatenating each element of $\mathbf{x}_{1:m}$ as column vectors as follows:

$$\mathbf{X} = [\ \mathbf{x} \mid \mathbf{x} \in \mathbf{x}_{1:m}\ ] = \begin{bmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{bmatrix}. \tag{A.4}$$

  Here, the vectors form the individual columns of the matrix. However, constructing matrices by explicitly placing elements within them might often be easier. This can similarly be done in the following way using the same $\mathbf{x}_{1:m}$:

$$\mathbf{X} = \left[\, [x_{ij}]_{ij} \;\middle|\; \mathbf{x}_i \in \mathbf{x}_{1:m} : \; x_{ij} \in \mathbf{x}_i \, \right] = \begin{bmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{bmatrix}. \tag{A.5}$$

In this case, the subscripts of individual elements, such as i in $\mathbf{x}_i$ and j in x_{ij}, indicate their position in the parent collection. For example, $\mathbf{x}_i$ is the i^{th} vector of $\mathbf{x}_{1:m}$ and x_{ij} is the j^{th} element of $\mathbf{x}_i$.[1]

- Given l scalar values summarised in $\mathbb{X} = (x_1, ..., x_l)$, for some $0 < l \leq nm$ with the index function $f(l) = (n, m)$ that maps an element in $\mathbb{X}$ to a coordinate (n, m), a matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ can be populated by elements of $\mathbb{X}$ via the index function as follows:

$$\mathbf{X} = \left[\, [x_l]_{ij} \;\middle|\; x_l \in \mathbb{X} : \; f(l) = (i, j) \, \right]. \tag{A.6}$$

Recall that the construction order is given by the ordered sets used to construct them, and the element positions not touched are assumed zero, i.e. this form of vector-builder notation can be viewed as populating an empty or zero matrix.

Remark A.1.4. Using the *vector-builder notation* in definition A.1.2, the construction of vectors and matrices can be expressed in a way that is convenient for software implementations. This is especially helpful with matrices since this is how most calibration algorithm terms are constructed in this research, e.g. Jacobians. Furthermore, this building method only has to iterate once over the iterated set to construct these multi-dimensional objects and can be easily parallelised. In addition, there is no need to account for zero-valued elements to reduce the complexity of the construction. The programming language `python3` follows a similar formalism to builder-notations in general with their *iterator comprehensions* to dynamically build there iterator objects[2]. This is the primary reason for using the vector-builder notation within this research.

Definition A.1.5 (Hadamard Product). *Horn and Johnson [42].* Given two matrices $\mathbf{X}, \mathbf{Y} \in \mathbb{C}^{m \times n}$, where $n, m \in \mathbb{N}$, the *Hadamard product* between $\mathbf{X}$ and $\mathbf{Y}$ is defined as the element-wise product given below:

$$\mathbf{X} \odot \mathbf{Y} = \left[\, [X_{ij} Y_{ij}]_{ij} \;\middle|\; (X_{ij} \in \mathbf{X})(Y_{ij} \in \mathbf{Y}) \, \right] = \begin{bmatrix} X_{11}Y_{11} & \cdots & X_{1n}Y_{1n} \\ \vdots & \ddots & \vdots \\ X_{m1}Y_{m1} & \cdots & X_{mn}Y_{mn} \end{bmatrix}. \tag{A.7}$$

[1] This special subscript notation is logically equivalent to that of the `enumerate` function used within the `python3` programming language. See remark A.1.4 for insight into its inclusion.

[2] See `python3` list-comprehensions.

This operation is denoted with the $\odot$ symbol.

Definition A.1.6 (Hadamard Division). *Horn and Johnson [42].* Given two matrices $\mathbf{X}, \mathbf{Y} \in \mathbb{C}^{m\times n}$, where $n, m \in \mathbb{N}$, the *Hadamard division* between $\mathbf{X}$ and $\mathbf{Y}$ is defined as the element-wise division given below:

$$\mathbf{X} \oslash \mathbf{Y} = \left[\left[\frac{X_{ij}}{Y_{ij}} \right]_{ij} \middle| (X_{ij} \in \mathbf{X})(Y_{ij} \in \mathbf{Y}) \right] = \begin{bmatrix} \frac{X_{11}}{Y_{11}} & \cdots & \frac{X_{1n}}{Y_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{X_{m1}}{Y_{m1}} & \cdots & \frac{X_{mn}}{Y_{mn}} \end{bmatrix}. \tag{A.8}$$

This operation is denoted with the $\oslash$ symbol.

Definition A.1.7 (Hadamard Power). *Horn and Johnson [42].* Given the matrix $\mathbf{X} \in \mathbb{C}^{m\times n}$, where $n, m \in \mathbb{N}$, and, for some $k \in \mathbb{C}$, the k^{th} *Hadamard power* of $\mathbf{X}$ is the element-wise k^{th} power given below:

$$\mathbf{X}^{(k)} = \left[\left[X_{ij}^{k} \right]_{ij} \middle| X_{ij} \in \mathbf{X} \right] = \begin{bmatrix} X_{11}^{k} & \cdots & X_{1n}^{k} \\ \vdots & \ddots & \vdots \\ X_{m1}^{k} & \cdots & X_{mn}^{k} \end{bmatrix}. \tag{A.9}$$

This operation is denoted with the $(\,\cdot\,)^{(k)}$ symbol.

Remark A.1.8 (Hadamard Inverse). *Horn and Johnson [42].* For $k = -1$, the Hadamard power becomes the *Hadamard inverse*, i.e. an element-wise inverse. Given $\mathbf{X} \in \mathbb{C}^{m\times n}$ for $n, m \in \mathbb{N}$, this can be written as

$$\mathbf{X}^{(-1)} = \left[\left[X_{ij}^{-1} \right]_{ij} \middle| X_{ij} \in \mathbf{X} \right] = \begin{bmatrix} X_{11}^{-1} & \cdots & X_{1n}^{-1} \\ \vdots & \ddots & \vdots \\ X_{m1}^{-1} & \cdots & X_{mn}^{-1} \end{bmatrix}. \tag{A.10}$$

Remark A.1.9. For $\mathbf{X}, \mathbf{Y} \in \mathbb{C}^{m\times n}$ with $n, m \in \mathbb{N}$, then for any $a, b \in \mathbb{C}$, the following relationship exists:

$$\mathbf{X}^{(a)} \oslash \mathbf{Y}^{(b)} = \mathbf{X}^{(a)} \odot \mathbf{Y}^{(-b)}. \tag{A.11}$$

Remark A.1.10. *Horn and Johnson [42].* The Hadamard operators defined in definitions A.1.5 to A.1.7 and remarks A.1.8 and A.1.9 can be extended to vectors as well, given that the vectors have an equivalent number of components.

Lemma A.1.11 (Properties of Hadamard Operators). *Horn and Johnson [42].* Let $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \in \mathbb{C}^{m \times n}$ and $\alpha, \beta \in \mathbb{C}$. The Hadamard product defined in definition A.1.5 has the following properties:

- *Commutative*: $\mathbf{X} \odot \mathbf{Y} = \mathbf{Y} \odot \mathbf{X}$,
- *Associative*: $\mathbf{X} \odot (\mathbf{Y} \odot \mathbf{Z}) = \mathbf{X} \odot (\mathbf{Y} \odot \mathbf{Z})$,
- *Distributive* over matrix addition: $\mathbf{X} \odot (\mathbf{Y} + \mathbf{Z}) = (\mathbf{X} \odot \mathbf{Y}) + (\mathbf{X} \odot \mathbf{Z})$.

These properties carry over to the Hadamard division in definition A.1.6. The Hadamard power in definition A.1.7 only satisfies the first two properties.

Lemma A.1.12 (Properties of Determinants). *Horn and Johnson [42].* Given $\mathbf{X} \in \mathbb{C}^{n \times n}$ for $n \in \mathbb{N}$ and $\alpha \in \mathbb{R}$, then the *matrix determinant*, denoted as $\det(\,\cdot\,)$, has the following properties:

- $\det(\alpha \mathbf{X}) = \alpha^n \det(\mathbf{X})$,
- $\det(\mathbf{X}^{-1}) = [\det(\mathbf{X})]^{-1}$,
- If $\mathbf{X}$ is diagonal, then $\det(\mathbf{X}) = \operatorname{tr}(\mathbf{X})$.

The function $\operatorname{tr}(\,\cdot\,)$ represents the *trace* of a matrix.

Definition A.1.13 (Diagonal Operator). *Horn and Johnson [42].* Given the square matrix $\mathbf{X} \in \mathbb{C}^{n \times n}$ for $n \in \mathbb{N}$, the diagonal of $\mathbf{X}$, which shall be called a *diagonal vector* denoted $\mathbf{x}$, can be extracted by applying the *diagonal operator*, $\operatorname{diag}(\,\cdot\,)$, defined below on $\mathbf{X}$:

$$\mathbf{x} = \operatorname{diag}(\mathbf{X}) = [\, X_{ii} \mid X_{ii} \in \mathbf{X} \,] = \Big[X_{11}, ..., X_{nn}\Big]^T. \tag{A.12}$$

Suppose the diagonal operator is given a vector input $\mathbf{x} \in \mathbb{C}^n$. In that case, it outputs a $n \times n$ matrix, denoted as $\mathbf{D}$ here, with the vector laid along the diagonal to create a diagonal matrix, i.e., non-zero values on the diagonal only. This process can be written as follows:

$$\mathbf{D} = \operatorname{diag}(\mathbf{x}) = [\, [x_i]_{ii} \mid x_i \in \mathbf{x} \,] = \begin{bmatrix} x_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & x_n \end{bmatrix}. \tag{A.13}$$

Lemma A.1.14 (Properties of Diagonal Matrices). *Horn and Johnson [42].*
For $n \in \mathbb{N}$, the (complex) diagonal matrix $\mathbf{X} \in \mathbb{C}^{n \times n}$ has the following properties:

- *Symmetric* - the matrix is equivalent to its transpose:

$$\mathbf{X} = \mathbf{X}^T. \tag{A.14}$$

- *Commutative* under matrix multiplication - if $\mathbf{Y}$ is a matrix with the same shape as $\mathbf{X}$, then the non-commutativity of matrix multiplication is absolved between these matrices:

$$\mathbf{XY} = \mathbf{YX}. \tag{A.15}$$

- The *determinant* is the product of diagonal entries - to calculate the determinant of a diagonal matrix use the following expression:

$$\det(\mathbf{X}) = \prod_{i=1}^{n} X_{ii}. \tag{A.16}$$

- *Invertible* - if for all $X_{ii} \in \mathbf{X} : \ X_{ii} \neq 0$, then $\mathbf{X}$ is invertible and can be calculated as

$$\mathbf{X}^{-1} = \operatorname{diag}\left(\left[\frac{1}{X_{ii}} \;\middle|\; X_{ii} \in \mathbf{X} \right]\right) = \begin{bmatrix} \frac{1}{X_{11}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{1}{X_{nn}} \end{bmatrix}. \tag{A.17}$$

- *Matrix power is equivalent to the Hadamard power* - the k^{th} power of $\mathbf{X}$ is equivalent to the k^{th} Hadamard power in definition A.1.7, i.e., element-wise power, of the same matrix:

$$\mathbf{X}^k = \mathbf{X}^{(k)} = \operatorname{diag}\left(\left[X_{ii}^k \;\middle|\; X_{ii} \in \mathbf{X} \right]\right) = \begin{bmatrix} X_{11}^k & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & X_{nn}^k \end{bmatrix}. \tag{A.18}$$

- *Matrix inverse is equivalent to the Hadamard inverse* - if $\mathbf{X}$ is invertible, then inverting $\mathbf{X}$ is same as the Hadamard inverse in remark A.1.8, i.e., element-wise reciprocal, along the diagonal:

$$\mathbf{X}^-1 = \mathbf{X}^{(-1)} = \operatorname{diag}\left(\left[X_{ii}^{-1} \;\middle|\; X_{ii} \in \mathbf{X} \right]\right) = \begin{bmatrix} X_{11}^{-1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & X_{nn}^{-1} \end{bmatrix}. \tag{A.19}$$

Lemma A.1.15 (Properties of Diagonal Operator). *Horn and Johnson [42].* Given matrices $\mathbf{X}, \mathbf{Y} \in \mathbb{C}^{m\times n}$ for $n, m \in \mathbb{N}$ and constants $\alpha, \beta \in \mathbb{C}$, the diagonal operator defined in definition A.1.13, specifically the *matrix-to-vector* form, has the following properties:

- *Linear operator* over matrices:

$$\text{diag}(\alpha\mathbf{X} + \beta\mathbf{Y}) = \alpha\text{diag}(\mathbf{X}) + \beta\text{diag}(\mathbf{Y}),$$

- *Symmetric*:

$$\text{diag}(\mathbf{X}) = \text{diag}(\mathbf{X}^T),$$

- If $\mathbf{X}$ and $\mathbf{Y}$ are diagonal matrices, then *distributive* over matrix multiplication:

$$\text{diag}(\mathbf{XY}) = \text{diag}(\mathbf{X}) \odot \text{diag}(\mathbf{Y}).$$

Remark A.1.16. Given $\mathbf{A} \in \mathbb{C}^{n\times n}$ is a diagonal matrix and $\mathbf{v} \in \mathbb{C}^n$ with $n \in \mathbb{N}$, then

$$\mathbf{Av} = \text{diag}(\mathbf{A}) \odot \mathbf{v}. \tag{A.20}$$

A.2 Wirtinger Calculus

Definition A.2.1 (Holomorphic function). *Schreier and Scharf [33].* A complex-valued function $f(\mathbf{z})$ is considered *holomorphic* if it is *complex differentiable* in the neighbourhood of each $\mathbf{z} \in \mathbb{C}^n$ in its domain. If not, then the function is *non-holomorphic*. This property of a given function is known as *holomorphy*.

Definition A.2.2 (Cauchy-Riemann equations). *Schreier and Scharf [33].* If $\mathbf{z} = \mathbf{x} + i\mathbf{y}$, where $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, then the function $\mathbf{f}$ can be rewritten as $\mathbf{f}(\mathbf{x} + i\mathbf{y}) = \mathbf{u}(\mathbf{x}, \mathbf{y}) + i\mathbf{v}(\mathbf{x}, \mathbf{y})$, where $\mathbf{u}$ and $\mathbf{v}$ are real-valued functions. To test whether a complex function $\mathbf{f}$ is indeed holomorphic, they need to satisfy the conditions

$$\frac{\partial \mathbf{u}}{\partial \mathbf{x}} = \frac{\partial \mathbf{v}}{\partial \mathbf{y}} \quad \text{and} \quad \frac{\partial \mathbf{u}}{\partial \mathbf{y}} = -\frac{\partial \mathbf{v}}{\partial \mathbf{x}}, \tag{A.21}$$

which are called the *Cauchy-Riemann equations* or *conditions*.

Definition A.2.3 (Wirtinger Cauchy-Riemann Equations). *Schreier and Scharf [33].* Under the *Wirtinger derivatives*, as described in eq. (2.59), the Cauchy-Riemann equations can be represented with the following expression:

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \mathbf{0}. \tag{A.22}$$

If the above is true, $\mathbf{f}$ is *holomorphic*. That means $\mathbf{f}$ is only a function of $\mathbf{x}$ and does not depend on $\mathbf{x}^*$.

Lemma A.2.4 (Properties of Wirtinger derivatives). *Schreier and Scharf [33].* Let $\mathbf{g}$ be another function defined for $\mathbf{x}$. Then, from eq. (2.59), the following properties arise:

$$\begin{aligned}
\frac{\partial \mathbf{x}}{\partial \mathbf{x}} &= \mathbf{I} \quad \text{and} \quad \frac{\partial \mathbf{x}^*}{\partial \mathbf{x}^*} = \mathbf{I}, \\
\frac{\partial \mathbf{x}}{\partial \mathbf{x}^*} &= \mathbf{0} \quad \text{and} \quad \frac{\partial \mathbf{x}^*}{\partial \mathbf{x}} = \mathbf{0}, \\
\frac{\partial \mathbf{f}^*}{\partial \mathbf{x}^*} &= \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right)^* \quad \text{and} \quad \frac{\partial \mathbf{f}}{\partial \mathbf{x}^*} = \left(\frac{\partial \mathbf{f}^*}{\partial \mathbf{x}}\right)^*, \\
\frac{\partial (\mathbf{f} \circ \mathbf{g})}{\partial \mathbf{x}} &= \frac{\partial \mathbf{f}}{\partial \mathbf{g}}\frac{\partial \mathbf{g}}{\partial \mathbf{x}} + \frac{\partial \mathbf{f}}{\partial \mathbf{g}^*}\frac{\partial \mathbf{g}^*}{\partial \mathbf{x}} \quad \text{and} \quad \frac{\partial (\mathbf{f} \circ \mathbf{g})}{\partial \mathbf{x}^*} = \frac{\partial \mathbf{f}}{\partial \mathbf{g}}\frac{\partial \mathbf{g}}{\partial \mathbf{x}^*} + \frac{\partial \mathbf{f}}{\partial \mathbf{g}^*}\frac{\partial \mathbf{g}^*}{\partial \mathbf{x}^*},
\end{aligned} \tag{A.23}$$

A.3 Taylor Series

Definition A.3.1 (First-order Taylor Polynomial). *Kreutz-Delgado [63].* Given the (at least) once differentiable holomorphic function $\mathbf{f}(\mathbf{x}) \in \mathbb{C}^m$ where $\mathbf{x} \in \mathbb{C}^n$ and some constant $\mathbf{a} \in \mathbb{C}^n$, the *first-order Taylor polynomial* $\mathbf{p}_1(\mathbf{x})$ of $\mathbf{f}(\mathbf{x})$ centred around $\mathbf{x} = \mathbf{a}$ is

$$\mathbf{p}_1(\mathbf{x}) = \mathbf{f}(\mathbf{a}) + \mathbf{J}(\mathbf{x} - \mathbf{a}), \tag{A.24}$$

where $\mathbf{J}$ is called the *Jacobian* matrix expressed as

$$\begin{aligned}
\mathbf{J} &= \frac{\mathrm{d}\mathbf{f}}{\mathrm{d}\mathbf{x}}(\mathbf{a}), \\
&= \left[\left[\frac{\partial f_i}{\partial x_j}(a_j)\right]_{ij} \,\middle|\, (f_i \in \mathbf{f})(x_j \in \mathbf{x})(a_j \in \mathbf{a}) \right], \\
&= \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(a_1) & \cdots & \frac{\partial f_1}{\partial x_n}(a_n) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1}(a_1) & \cdots & \frac{\partial f_m}{\partial x_n}(a_n) \end{bmatrix},
\end{aligned} \tag{A.25}$$

and is evaluated at $\mathbf{x} = \mathbf{a}$.

Definition A.3.2 (Linear Approximation). *Kreutz-Delgado [63].* The linear approximation of the function $\mathbf{f}$ at $\mathbf{x} = \mathbf{a}$ is equivalent to the first-order Taylor polynomial (see definition A.3.1) of $\mathbf{f}$, that is

$$\mathbf{f}(\mathbf{x}) \simeq \mathbf{f}(\mathbf{a}) + \mathbf{J}(\mathbf{x} - \mathbf{a}), \tag{A.26}$$

where equality is reached if $\mathbf{f}$ is linear.

Definition A.3.3 (Wirtinger First-order Taylor Polynomial). *Schreier and Scharf [33].* Given the (at least) once differentiable function $\mathbf{f}(\mathbf{x}) \in \mathbb{C}^m$ where $\mathbf{x} \in \mathbb{C}^n$ and some constant $\mathbf{a} \in \mathbb{C}^n$, the *Wirtinger first-order Taylor polynomial* $\mathbf{p}_1(\mathbf{x})$ of $\mathbf{f}(\mathbf{x})$ centred around $\mathbf{x} = \mathbf{a}$ is

$$\mathbf{p}_1(\mathbf{x}) = \mathbf{f}(\mathbf{a}) + \mathbf{J}(\mathbf{x} - \mathbf{a}) + \widetilde{\mathbf{J}}(\mathbf{x}^* - \mathbf{a}^*), \tag{A.27}$$

where $\mathbf{J}$ and $\widetilde{\mathbf{J}}$ is called the *Jacobian* and *Wirtinger Jacobian*, respectively, where $\mathbf{J}$ is constructed using eq. (A.25), and $\widetilde{\mathbf{J}}$ is constructed as follows:

$$\begin{aligned}
\widetilde{\mathbf{J}} &= \frac{\mathrm{d}\mathbf{f}}{\mathrm{d}\mathbf{x}^*}(\mathbf{a}), \\
&= \left[\left[\frac{\partial f_i}{\partial x_j^*}(a_j) \right]_{ij} \,\middle|\, (f_i \in \mathbf{f})(x_j \in \mathbf{x})(a_j \in \mathbf{a}) \right], \\
&= \begin{bmatrix} \frac{\partial f_1}{\partial x_1^*}(a_1) & \cdots & \frac{\partial f_1}{\partial x_n^*}(a_n) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1^*}(a_1) & \cdots & \frac{\partial f_m}{\partial x_n^*}(a_n) \end{bmatrix}.
\end{aligned} \tag{A.28}$$

This form of the Taylor polynomial can be applied regardless of holomorphy. However, if $\mathbf{f}$ is holomorphic, the Wirtinger Jacobian is zero, and the final result is equivalent to definition A.3.1. It is often implied that these complex functions are mapped over $\mathbf{x}$ and its conjugate $\mathbf{x}^*$, and therefore, the conjugate terms are omitted in function definitions for brevity.

A.4 Statistics and Probability

Lemma A.4.1. *Särkkä [24].* Let $\mathbf{x} \sim \mathcal{C}_n(\mathbf{m}_\mathbf{x}, \mathbf{P}_\mathbf{x})$, where $\mathbf{m}_\mathbf{x} \in \mathbb{C}^n$ and $\mathbf{P}_\mathbf{x} \in \mathbb{R}^{n \times n}$, and let $\mathbf{y} \sim p(\mathbf{y})$, with $\mathrm{E}[\mathbf{y}] = \mathbf{m}_\mathbf{y} \in \mathbb{C}^m$ and $\mathrm{Var}[\mathbf{y}] = \mathbf{P}_\mathbf{y} \in \mathbb{C}^{m \times m}$. There exist two equivalent statements that can be made in this scenario:

- If $\mathbf{x}$ and $\mathbf{y}$ are related by the linear relationship $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b}$, for some $\mathbf{A} \in \mathbb{C}^{m \times n}$ and $\mathbf{b} \in \mathbb{C}^m$, then

  $$p(\mathbf{y}) = \mathcal{C}_m(\mathbf{m}_\mathbf{y}, \mathbf{P}_\mathbf{y}), \tag{A.29}$$

  that is, $p(\mathbf{y})$ is also a *Gaussian density.*

- If $p(\mathbf{y}) = \mathcal{C}(\mathbf{m}_\mathbf{y}, \mathbf{P}_\mathbf{y})$, i.e., it is Gaussian, and it is known there is a relationship between $\mathbf{x}$ and $\mathbf{y}$, then there exists $\mathbf{A}' \in \mathbb{C}^{m \times n}$ and $\mathbf{b}' \in \mathbb{C}^m$ such that

$$\mathbf{y} = \mathbf{A}'\mathbf{x} + \mathbf{b}', \tag{A.30}$$

that is, a linear relationship exists between these random vectors.

Lemma A.4.2. *Särkkä [24].* The conditional density of $\mathbf{y} \sim \mathcal{C}_m(\mathbf{m}_\mathbf{y}, \mathbf{P}_\mathbf{y})$ given $\mathbf{x} \sim \mathcal{C}_n(\mathbf{m}_\mathbf{x}, \mathbf{P}_\mathbf{x})$, with the linear relationship $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b}$ (see lemma A.4.1), is written as

$$\mathbf{y} \mid \mathbf{x} \sim \mathcal{C}_m(\mathbf{m}_{\mathbf{y}|\mathbf{x}}, \mathbf{P}_{\mathbf{y}|\mathbf{x}}) = \mathcal{C}_m(\mathbf{A}\mathbf{x} + \mathbf{b}, \mathbf{A}\mathbf{P}_\mathbf{x}\mathbf{A}^H). \tag{A.31}$$

Additionally, if the Gaussian densities of $\mathbf{x}$ and $\mathbf{y} \mid \mathbf{x}$ are multiplied, the resulting joint density between $\mathbf{x}$ and $\mathbf{y}$ is expressed as

$$\begin{aligned} p(\mathbf{x}, \mathbf{y}) &= \mathcal{C}_n(\mathbf{x} \mid \mathbf{m}_\mathbf{x}, \mathbf{P}_\mathbf{x}) \cdot \mathcal{C}_m(\mathbf{y} \mid \mathbf{A}\mathbf{x} + \mathbf{b}, \mathbf{A}\mathbf{P}_\mathbf{x}\mathbf{A}^H), \\ &= \mathcal{C}_{n+m}\left(\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \middle| \begin{bmatrix} \mathbf{m}_\mathbf{x} \\ \mathbf{A}\mathbf{m}_\mathbf{x} + \mathbf{b} \end{bmatrix}, \begin{bmatrix} \mathbf{P}_\mathbf{x} & \mathbf{P}_\mathbf{x}\mathbf{A}^H \\ \mathbf{A}\mathbf{P}_\mathbf{x} & \mathbf{A}\mathbf{P}_\mathbf{x}\mathbf{A}^H + \mathbf{P}_\mathbf{y} \end{bmatrix} \right), \end{aligned} \tag{A.32}$$

and the marginal density of $\mathbf{y}$, after integrating eq. (A.32) over $\mathbf{x}$, comes to

$$\mathbf{y} \sim \mathcal{C}_m(\mathbf{A}\mathbf{m}_\mathbf{x} + \mathbf{b}, \mathbf{A}\mathbf{P}_\mathbf{x}\mathbf{A}^H + \mathbf{P}_\mathbf{y}). \tag{A.33}$$

Lemma A.4.3. *Särkkä [24].* As a converse to eq. (A.31), if the joint density between $\mathbf{x}$ and $\mathbf{y}$ is

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \sim \mathcal{C}_{n+m}\left(\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}, \begin{bmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{C}^H & \mathbf{B} \end{bmatrix} \right), \tag{A.34}$$

then, the conditional and marginal densities from the above joint density are

- $\mathbf{x} \sim \mathcal{C}_n(\mathbf{a}, \mathbf{A})$,
- $\mathbf{y} \sim \mathcal{C}_m(\mathbf{b}, \mathbf{B})$,
- $\mathbf{x} \mid \mathbf{y} \sim \mathcal{C}_n(\mathbf{x} \mid \mathbf{a} + \mathbf{C}\mathbf{B}^{-1}(\mathbf{y} - \mathbf{b}), \mathbf{A} - \mathbf{C}\mathbf{B}^{-1}\mathbf{C}^H)$,
- $\mathbf{y} \mid \mathbf{x} \sim \mathcal{C}_m(\mathbf{y} \mid \mathbf{b} + \mathbf{C}^H\mathbf{A}^{-1}(\mathbf{x} - \mathbf{a}), \mathbf{B} - \mathbf{C}^H\mathbf{A}^{-1}\mathbf{C})$.

Lemma A.4.4 (Chapman-Kolmogorov Equation). *Särkkä [24].* Let $\mathbf{x}_{1:N}$ be a Markov process for some time frame $\mathbb{T}_N$ where $N \in \mathbb{N}$. The Chapman-Kolmogorov equation states that under these conditions, for $k \in \mathbb{T}_N$, the following relationship exists:

$$p(\mathbf{x}_k \mid \mathbf{x}_{k-2}) = \int p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) \cdot (\mathbf{x}_{k-1} \mid \mathbf{x}_{k-2}) \, \mathrm{d}\mathbf{x}_{k-1}. \tag{A.35}$$

If $\mathbf{x}_{1:N}$ is non-Markovian, then it is written as:

$$p(\mathbf{x}_k \mid \mathbf{x}_{k-2}) = \int p(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{x}_{k-2}) p(\mathbf{x}_{k-1} \mid \mathbf{x}_{k-2}) \, \mathrm{d}\mathbf{x}_{k-1}. \tag{A.36}$$

Lemma A.4.5 (Whitened Measurement Noise). *Schreier and Scharf [33].* Consider the following augmented complex non-linear relationship between $\underline{\mathbf{x}} \in \mathbb{C}^{2n}$ and $\underline{\mathbf{y}} \in \mathbb{C}^{2m}$:

$$\underline{\mathbf{y}} = \underline{\mathbf{f}}(\underline{\mathbf{x}}) + \underline{\mathbf{r}}, \quad \underline{\mathbf{r}} \sim \mathcal{CN}_{2m}(\mathbf{0}, \underline{\mathbf{R}}). \tag{A.37}$$

The function $\underline{\mathbf{f}} : \mathbb{C}^{2n} \to \mathbb{C}^2 m$ is non-linear measurement operator that describes how $\underline{\mathbf{x}}$ relates to $\underline{\mathbf{y}}$. The term $\underline{\mathbf{r}}$ is the measurement noise governed by measurement covariance $\underline{\mathbf{R}} \in \mathbb{C}^{2m}$. Equation (A.37) represented as a measurement density is

$$p(\underline{\mathbf{y}} \mid \underline{\mathbf{x}}) = \mathcal{CN}_{2m}(\underline{\mathbf{y}} \mid \underline{\mathbf{f}}(\underline{\mathbf{x}}), \underline{\mathbf{R}}). \tag{A.38}$$

Now, consider a different relationship for a new complex vector $\underline{\mathbf{y}}' \in \mathbb{C}^{2m}$:

$$\underline{\mathbf{y}}' = \underline{\mathbf{R}}^{-\frac{1}{2}} \underline{\mathbf{y}}. \tag{A.39}$$

To *whiten the measurement noise* in eq. (A.38), substitute eq. (A.37) into eq. (A.39), and determine the measurement density with respect to this new expression to get:

$$p(\underline{\mathbf{y}}' \mid \underline{\mathbf{x}}) = \mathcal{CN}_{2m}(\underline{\mathbf{y}}' \mid \underline{\mathbf{R}}^{-\frac{1}{2}} \underline{\mathbf{f}}(\underline{\mathbf{x}}), \mathbf{I}). \tag{A.40}$$

The term $\mathbf{I}$ is a $2m \times 2m$ identity matrix. Equation (A.40) is the *whitened measurement density.*

Proof A.4.6 (Bayesian Filtering Solution). *See theorem 2.3.1.* To begin, apply Bayes' theorem in eq. (2.7) to the filtering density $p(\mathbf{x}_k \mid \mathbf{y}_{1:k})$ to get

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) = \frac{p(\mathbf{y}_{1:k} \mid \mathbf{x}_k) p(\mathbf{x}_k)}{p(\mathbf{y}_{1:k})}. \tag{A.41}$$

Then, separate the current measurement $\mathbf{y}_k$ and prior measurements $\mathbf{y}_{1:k-1}$ by applying the chain rule identity in eq. (2.5) on the joint densities and re-order terms to get

$$\begin{aligned} p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) &= \frac{p(\mathbf{y}_k \mid \mathbf{x}_k, \mathbf{y}_{1:k-1}) p(\mathbf{y}_{1:k-1} \mid \mathbf{x}_k) p(\mathbf{x}_k)}{p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}) p(\mathbf{y}_{1:k-1})}, \\ &= \frac{p(\mathbf{y}_k \mid \mathbf{x}_k, \mathbf{y}_{1:k-1})}{p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1})} \cdot \frac{p(\mathbf{y}_{1:k-1} \mid \mathbf{x}_k) p(\mathbf{x}_k)}{p(\mathbf{y}_{1:k-1})}. \end{aligned} \tag{A.42}$$

Again, use Bayes' theorem in eq. (2.7) on the latter fraction in eq. (A.41) to get

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) = \frac{p(\mathbf{y}_k \mid \mathbf{x}_k, \mathbf{y}_{1:k-1}) p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1})}{p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1})}, \tag{A.43}$$

where the conditional density $p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1})$ is the prediction density. Given the conditional independence of measurements, the measurement density $p(\mathbf{y}_k \mid \mathbf{x}_k, \mathbf{y}_{1:k-1})$ simplifies such that

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) = \frac{p(\mathbf{y}_k \mid \mathbf{x}_k)p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1})}{p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1})}. \tag{A.44}$$

Let $z_k = p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1})$, i.e. the normalisation constant, that can be computed by applying the Chapman Kolmogorov equation in lemma A.4.4 with respect to the current state $\mathbf{x}_k$ which results in:

$$z_k = p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}) = \int p(\mathbf{y}_k \mid \mathbf{x}_k)p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}) \, \mathrm{d}\mathbf{x}_k. \tag{A.45}$$

In eqs. (A.44) and (A.45), the prediction density $p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1})$ is required. This can also be computed by using lemma A.4.4 again, but with respect to the previous state $\mathbf{x}_{k-1}$ and the density of the conditional density $p(\mathbf{x}_k \mid \mathbf{x}_{k-1})$, known as the evolution density. This is written as

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}) = \int p(\mathbf{x}_k \mid \mathbf{x}_{k-1})p(\mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1}) \, \mathrm{d}\mathbf{x}_{k-1}. \tag{A.46}$$

The above holds true because of the Markov property of states, i.e. $p(\mathbf{x}_k \mid \mathbf{x}_{1:k-1}, \mathbf{y}_{1:k-1}) = p(\mathbf{x}_k \mid \mathbf{x}_{k-1})$. Equation (A.46) is known as the prediction step. By incorporating the prediction step with eqs. (A.44) and (A.45), the theorem eqs. (2.39) and (2.40) are respectively formed as

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) = \frac{1}{z_k} p(\mathbf{y}_k \mid \mathbf{x}_k) \int p(\mathbf{x}_k \mid \mathbf{x}_{k-1})p(\mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1}) \, \mathrm{d}\mathbf{x}_{k-1}, \tag{A.47}$$

and

$$z_k = \int p(\mathbf{y}_k \mid \mathbf{x}_k) \left(\int p(\mathbf{x}_k \mid \mathbf{x}_{k-1})p(\mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1}) \, \mathrm{d}\mathbf{x}_{k-1} \right) \mathrm{d}\mathbf{x}_k. \tag{A.48}$$

Equation (A.47) is the generalised Bayesian solution to the filtering problem and concludes the proof.

□

Proof A.4.7 (Kalman Filter). *See algorithm 2.3.2.* The derivation of the KF begins by following the structure of the Bayesian filtering solution, specifically from eq. (2.42), to determine the parameters of the now Gaussian prediction and filtering densities. The evidence density is not required for this derivation, see §2.2.6. First, find the predictor mean and covariance using eq. (2.41). Consider the joint conditional density of $\mathbf{x}_k$ and $\mathbf{x}_{k-1}$ given $\mathbf{y}_{1:k-1}$ using the linear Gaussian models specified in eqs. (2.43) and (2.44). By incorporating the identities within lemma A.4.2 to get the following:

$$\begin{aligned} p(\mathbf{x}_{k-1}, \mathbf{x}_k \mid \mathbf{y}_{1:k-1}) &= p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) p(\mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1}), \\ &= \mathcal{CN}_n(\mathbf{x}_k \mid \mathbf{F}_{k-1}\mathbf{x}_{k-1}, \mathbf{Q}_{k-1}) \mathcal{CN}_n(\mathbf{x}_{k-1} \mid \mathbf{m}_{f,k-1}, \mathbf{P}_{f,k-1}), \\ &= \mathcal{CN}_n\left(\begin{bmatrix} \mathbf{x}_{k-1} \\ \mathbf{x}_k \end{bmatrix} \;\middle|\; \mathbf{m}'_k, \mathbf{P}'_k \right), \end{aligned} \tag{A.49}$$

where

$$\mathbf{m}'_k = \begin{bmatrix} \mathbf{m}_{f,k-1} \\ \mathbf{F}_{k-1}\mathbf{m}_{f,k-1} \end{bmatrix}, \tag{A.50}$$

$$\mathbf{P}'_k = \begin{bmatrix} \mathbf{P}_{f,k-1} & \mathbf{P}_{f,k-1}\mathbf{F}_{k-1}^H \\ \mathbf{F}_{k-1}\mathbf{P}_{f,k-1} & \mathbf{F}_{k-1}\mathbf{P}_{f,k-1}\mathbf{F}_{k-1}^H + \mathbf{Q}_{k-1} \end{bmatrix}. \tag{A.51}$$

To obtain the prediction density $p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1})$, marginalise eq. (A.49) over $\mathbf{x}_{k-1}$ to arrive at

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}) = \mathcal{CN}_n(\mathbf{m}_{p,k}, \mathbf{P}_{p,k}), \tag{A.52}$$

where

$$\mathbf{m}_{p,k} = \mathbf{F}_{k-1}\mathbf{m}_{f,k-1}, \tag{A.53}$$

$$\mathbf{P}_{p,k} = \mathbf{F}_{k-1}\mathbf{P}_{f,k-1}\mathbf{F}_{k-1}^H + \mathbf{Q}_{k-1}. \tag{A.54}$$

Equations (A.53) and (A.54) are known as the *prediction* step. It allows for formulating the update step that generates the filter mean and covariance. The filtering density in eq. (2.42) is required to achieve this. Integrate the current measurement $\mathbf{y}_k$ to update the Gaussian prediction density. Noting the identities within lemma A.4.2 again, the measurement density is a conditional Gaussian and thus can be expressed via the linear measurement model in eq. (2.44) as follows:

$$p(\mathbf{y}_k \mid \mathbf{x}_k) = \mathcal{CN}_n(\mathbf{y}_k \mid \mathbf{H}_k\mathbf{x}_k, \mathbf{R}_k). \tag{A.55}$$

Consider the joint density of $\mathbf{x}_k$ and $\mathbf{y}_k$. Applying the Markov and the conditional independence of measurement properties, this density can be rewritten as

$$p(\mathbf{x}_k, \mathbf{y}_k \mid \mathbf{y}_{1:k-1}) = p(\mathbf{y}_k \mid \mathbf{x}_k) p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}), \tag{A.56}$$

Substitute into eq. (A.56) the related densities in eq. (A.52) and eq. (A.55). Then, reintroduce the same derivation procedure for eq. (A.52) to produce

$$p(\mathbf{x}_k, \mathbf{y}_k \mid \mathbf{y}_{1:k-1}) = \mathcal{CN}_m(\mathbf{y}_k \mid \mathbf{H}_k\mathbf{x}_k, \mathbf{R}_k)\mathcal{CN}_n(\mathbf{x}_k \mid \mathbf{m}_{p,k}, \mathbf{P}_{p,k}),$$
$$= \mathcal{CN}_{n+m}\left(\begin{bmatrix}\mathbf{x}_k \\ \mathbf{y}_k\end{bmatrix} \;\middle|\; \mathbf{m}''_k, \mathbf{P}''_k\right), \tag{A.57}$$

where

$$\mathbf{m}''_k = \begin{bmatrix}\mathbf{m}_{p,k} \\ \mathbf{H}_k\mathbf{m}_{p,k}\end{bmatrix}, \tag{A.58}$$

$$\mathbf{P}''_k = \begin{bmatrix}\mathbf{P}_{p,k} & \mathbf{P}_{p,k}\mathbf{H}_k^H \\ \mathbf{H}_k\mathbf{P}_{p,k} & \mathbf{H}_k\mathbf{P}_{p,k}\mathbf{H}_k^H + \mathbf{R}_k\end{bmatrix}. \tag{A.59}$$

The final step is to apply lemma A.4.3 on eq. (A.57) to obtain the needed filtering density as

$$p(\mathbf{x}_k \mid \mathbf{y}_k, \mathbf{y}_{1:k-1}) = p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) = \mathcal{CN}_n(\mathbf{x}_k \mid \mathbf{m}_{f,k}, \mathbf{P}_{f,k}). \tag{A.60}$$

where

$$\mathbf{m}_{f,k} = \mathbf{m}_{p,k} + \mathbf{P}_{p,k}\mathbf{H}_k^H(\mathbf{H}_k\mathbf{P}_{p,k}\mathbf{H}_k^H + \mathbf{R}_k)^{-1}(\mathbf{y}_k - \mathbf{H}_k\mathbf{m}_{p,k}), \tag{A.61}$$
$$\mathbf{P}_{f,k} = \mathbf{P}_{p,k} - \mathbf{P}_{p,k}\mathbf{H}_k^H(\mathbf{H}_k\mathbf{P}_{p,k}\mathbf{H}_k^H + \mathbf{R}_k)^{-1}\mathbf{H}_k\mathbf{P}_{p,k}. \tag{A.62}$$

Equations (A.61) and (A.62) is the *update* step and generates the filter mean and covariance. All intermediate terms are contained within these expressions. Therefore, initialising the recursion with a prior filter mean $\mathbf{m}_{f,0}$ and covariance $\mathbf{P}_{f,0}$, the predictor and filter parameters can be generated for each state in $\mathbf{x}_{1:N}$ in a forward recursive manner. This forms the complete KF algorithm.

□

Proof A.4.8 (Bayesian Smoothing Solution). *See theorem 2.3.3.* After applying the Bayesian filtering solution for each state in the Markov process $\mathbf{x}_{1:N}$ to produce a corresponding prediction and filtering density, the recursive Bayesian smoothing solution can be applied. For the time frame $\mathbb{T}_N$, the smoothing density at time k is $p(\mathbf{x}_k \mid \mathbf{y}_{1:N})$. Apply the Chapman-Kolmogorov equation in eq. (A.36) for a non-Markov process to keep the condition on all measurements over the future state $\mathbf{x}_{k+1}$ to get

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:N}) = \int p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:N}) p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:N}) \, \mathrm{d}\mathbf{x}_{k+1},$$
$$= \int p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:k}) p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:N}) \, \mathrm{d}\mathbf{x}_{k+1}. \tag{A.63}$$

Here, the current smoothing density is written as the future smoothing density combined with a separate density $p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:k})$ that is generated by noting the independence of the current state with future measurements due to the Markov property. Apply Bayes' theorem in eq. (2.7) to this density to acquire

$$p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:k}) = \frac{p(\mathbf{x}_{k+1}, \mathbf{y}_{1:k} \mid \mathbf{x}_k) p(\mathbf{x}_k)}{p(\mathbf{x}_{k+1}, \mathbf{y}_{1:k})}. \tag{A.64}$$

Using the chain rule of probabilities, this becomes

$$p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:k}) = \frac{p(\mathbf{x}_{k+1} \mid \mathbf{x}_k, \mathbf{y}_{1:k}) p(\mathbf{y}_{1:k} \mid \mathbf{x}_k) p(\mathbf{x}_k)}{p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:k}) p(\mathbf{y}_{1:k})}. \tag{A.65}$$

The Markov property shows that $p(\mathbf{x}_{k+1} \mid \mathbf{x}_k, \mathbf{y}_{1:k}) = p(\mathbf{x}_{k+1} \mid \mathbf{x}_k)$, and with Bayes' theorem, $p(\mathbf{y}_{1:k} \mid \mathbf{x}_k) p(\mathbf{x}_k) = p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) p(\mathbf{y}_{1:k})$. Therefore, eq. (A.65) can be rewritten as

$$p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:k}) = \frac{p(\mathbf{x}_{k+1} \mid \mathbf{x}_k) p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) p(\mathbf{y}_{1:k})}{p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:k}) p(\mathbf{y}_{1:k})},$$
$$= \frac{p(\mathbf{x}_{k+1} \mid \mathbf{x}_k) p(\mathbf{x}_k \mid \mathbf{y}_{1:k})}{p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:k})}. \tag{A.66}$$

Here, the subject density can be constructed with the current filtering density pushed through the evolution density from the current state to the future state divided by the future prediction density. Substitute eq. (A.66) into eq. (A.63) to arrive at the Bayesian smoothing solution:

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:N}) = p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) \int \frac{p(\mathbf{x}_{k+1} \mid \mathbf{x}_k) p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:N})}{p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:k})} \, \mathrm{d}\mathbf{x}_{k+1}. \tag{A.67}$$

The filtering density falls outside the integral because it does not depend on $\mathbf{x}_{k+1}$. Moreover, the prediction density $p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:k})$ can be calculated with the same expression in eq. (A.46). Note, the filtering density $p(\mathbf{x}_N \mid \mathbf{y}_{1:N})$ is a smoothing density as well. Therefore, initiating the backwards recursion with the last filtering density as the initial smoothing density, each filtering density of each state in $\mathbf{x}_{1:N}$ can be adjusted to incorporate all measurement information, turning them into smoothing densities. This completes the proof.

□

Proof A.4.9 (Rauch-Tung-Striebel (Kalman) Smoother). *See algorithm 2.3.4.* Similar to proof A.4.7, the objective is to derive the parameters for the now Gaussian smoothing density, i.e. the expectation $\mathbf{m}_{s,k}$ and covariance $\mathbf{P}_{s,k}$, for each state in $\mathbf{x}_{1:N}$, given the results from

the KF in algorithm 2.3.2, using the Bayesian smoothing solution in theorem 2.3.3 as a guide. Consider the joint density of the current state $\mathbf{x}_k$ and the future state $\mathbf{x}_{k+1}$ given current and past measurements, $\mathbf{y}_{1:k}$. Then, carry out the same steps with the same assumptions in eqs. (A.49) to (A.51) with respect to this joint density to get

$$p(\mathbf{x}_k, \mathbf{x}_{k+1} \mid \mathbf{y}_{1:k}) = \mathcal{CN}_{2n}\left(\begin{bmatrix}\mathbf{x}_k \\ \mathbf{x}_{k+1}\end{bmatrix} \middle| \mathbf{m}', \mathbf{P}'\right), \tag{A.68}$$

where

$$\mathbf{m}' = \begin{bmatrix}\mathbf{m}_{f,k} \\ \mathbf{F}_k\mathbf{m}_{f,k}\end{bmatrix}, \tag{A.69}$$

$$\mathbf{P}' = \begin{bmatrix}\mathbf{P}_{f,k} & \mathbf{P}_{f,k}\mathbf{F}_k^H \\ \mathbf{F}_k\mathbf{P}_{f,k} & \mathbf{F}_k\mathbf{P}_{f,k}\mathbf{F}_k^H + \mathbf{Q}_k\end{bmatrix}. \tag{A.70}$$

From this point, because of the Markov property, it is known that

$$p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:N}) = p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:k}), \tag{A.71}$$

since $\mathbf{x}_k$ does not depend on future measurements. With lemma A.4.3, the conditional density in eq. (A.71) can be written as

$$p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:N}) = \mathcal{CN}_n(\mathbf{x}_k \mid \mathbf{m}'', \mathbf{P}''), \tag{A.72}$$

where

$$\mathbf{m}'' = \mathbf{m}_{f,k} + \mathbf{P}_{f,k}\mathbf{F}_k^H(\mathbf{F}_k\mathbf{P}_{f,k}\mathbf{F}_k^H + \mathbf{Q}_k)^{-1}(\mathbf{x}_{k+1} - \mathbf{F}_k\mathbf{m}_{f,k}), \tag{A.73}$$

$$\mathbf{P}'' = \mathbf{P}_{f,k} - \mathbf{P}_{f,k}\mathbf{F}_k^H(\mathbf{F}_k\mathbf{P}_{f,k}\mathbf{F}_k^H + \mathbf{Q}_k)^{-1}\mathbf{F}_k\mathbf{P}_{f,k}. \tag{A.74}$$

Now, introduce a new term called the *smoothing gain*[3] denoted as

$$\mathbf{W}_k = \mathbf{P}_{f,k}\mathbf{F}_k^H\left(\mathbf{F}_k\mathbf{P}_{f,k}\mathbf{F}_k^H + \mathbf{Q}_k\right)^{-1}, \tag{A.75}$$

that is a weight term that balances the influence of the confidence in the filtering solution of the current state $\mathbf{x}_k$ with the prediction solution of the future state $\mathbf{x}_{k+1}$. Using eq. (A.75), rewrite eq. (A.73) as

$$\mathbf{m}'' = \mathbf{m}_{f,k} + \mathbf{W}_k\left(\mathbf{x}_{k+1} - \mathbf{F}_k\mathbf{m}_{f,k}\right), \tag{A.76}$$

and eq. (A.74) as

[3] The conventional notation for the smoothing gain is $\mathbf{G}_k$. This notation is altered to avoid confusion with the Jones term discussed in §1.2.3.

$$\mathbf{P}'' = \mathbf{P}_{f,k} - \mathbf{W}_k \left(\mathbf{F}_k \mathbf{P}_{f,k} \mathbf{F}_k^H + \mathbf{Q}_k \right) \mathbf{W}_k^H. \tag{A.77}$$

Equations (A.76) and (A.77) updates the current filter mean and covariance given the difference between a realisation of the future state $\mathbf{x}_{k+1}$ and the predictor mean and covariance of the future state. This update is weighted according to the smoothing gain in eq. (A.75). Thus, the missing component is to realise the future state. Create another joint density of $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$, but given all measurements $\mathbf{y}_{1:N}$, written as $p(\mathbf{x}_k, \mathbf{x}_{k+1} \mid \mathbf{y}_{1:N})$, and apply Bayes' theorem in eq. (2.7). From here, noting the necessary points where the Markov property applies, use lemma A.4.2 to get

$$\begin{aligned} p(\mathbf{x}_k, \mathbf{x}_{k+1} \mid \mathbf{y}_{1:N}) &= p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:N}) p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:N}), \\ &= \mathcal{CN}_n(\mathbf{x}_k \mid \mathbf{m}'', \mathbf{P}'') \mathcal{CN}_n(\mathbf{x}_{k+1} \mid \mathbf{m}_{s,k+1}, \mathbf{P}_{s,k+1}), \\ &= \mathcal{CN}_{2n} \left(\begin{bmatrix} \mathbf{x}_k \\ \mathbf{x}_{k+1} \end{bmatrix} \middle| \mathbf{m}''', \mathbf{P}''' \right), \end{aligned} \tag{A.78}$$

where

$$\mathbf{m}''' = \begin{bmatrix} \mathbf{m}_{f,k} + \mathbf{W}_k(\mathbf{m}_{s,k+1} - \mathbf{F}_k \mathbf{m}_{f,k}) \\ \mathbf{m}_{s,k+1} \end{bmatrix}, \tag{A.79}$$

$$\mathbf{P}''' = \begin{bmatrix} \mathbf{P}_{s,k+1} & \mathbf{P}_{s,k+1} \mathbf{W}_k^H \\ \mathbf{W}_k \mathbf{P}_{s,k+1} & \mathbf{W}_k \mathbf{P}_{s,k+1} \mathbf{W}_k^H + \mathbf{P}'' \end{bmatrix}. \tag{A.80}$$

The terms $\mathbf{m}_{s,k+1}$ and $\mathbf{P}_{s,k+1}$ are the expectation and covariance for the smoothing density of the future state $\mathbf{x}_{k+1}$. Finally, using lemma A.4.3, marginalise the density in eq. (A.78) over $\mathbf{x}_{k+1}$ to retrieve the smoothing density for $\mathbf{x}_k$. The result is

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:N}) = \mathcal{CN}_n(\mathbf{x}_k \mid \mathbf{m}_{s,k}, \mathbf{P}_{s,k}), \tag{A.81}$$

where

$$\mathbf{m}_{s,k} = \mathbf{m}_{f,k} + \mathbf{W}_k(\mathbf{m}_{s,k+1} - \mathbf{F}_k \mathbf{m}_{f,k}), \tag{A.82}$$

$$\mathbf{P}_{s,k} = \mathbf{P}_{f,k} + \mathbf{W}_k(\mathbf{P}_{s,k+1} - \mathbf{F}_k \mathbf{P}_{f,k} \mathbf{F}_k^H - \mathbf{Q}_k) \mathbf{W}_k^H. \tag{A.83}$$

Noting that for the last state $\mathbf{x}_N$, the initial smoother results are $\mathbf{m}_{s,N} = \mathbf{m}_{f,N}$ and $\mathbf{P}_{s,N} = \mathbf{P}_{f,N}$ since the filtering and smoothing densities are equivalent. Therefore, using eqs. (A.82) and (A.83) and starting the backwards recursion with the last filter mean and covariance, the smoother means $\mathbf{m}_{s,1:N}$ and covariances $\mathbf{P}_{s,1:N}$ can be generated. This completes the KS algorithm.

□

Appendix B

Additional Figures and Tables

This appendix contains the additional figures and tables associated with the results and analysis presented in §4.2 of chapter 4. The results are given for each calibration scenario in table 4.1: H100, H80, L100, and L80.

		H100		H80	
Algorithm	**Parameter**	**Value**	**MSE**	**Value**	**MSE**
QuartiCal	Δ_t	5	0.00392	**5**	**0.00395**
Diag-Predictor	σ_Q	0.0518	0.00759	0.0518	0.00892
Diag-Filter	σ_Q	0.045	0.00518	0.045	0.00652
Diag-Smoother	σ_Q	0.0687^{*}	0.00369^{*}	0.0687^{*}	0.00502^{*}
Diag-Energy	σ_Q	0.045	(2.78×10^5)	0.045	(2.93×10^5)
Full-Predictor	σ_Q	0.0518	0.00741	0.0518	0.00872
Full-Filter	σ_Q	0.045	0.005	0.045	0.00633
Full-Smoother	σ_Q	$\mathbf{0.0687}^{**}$	$\mathbf{0.00359}^{**}$	0.0687^{**}	0.00489^{**}
Full-Energy	σ_Q	0.0518	5.53×10^5	0.0518	5.85×10^5

TABLE B.1: Optimal MSE values and corresponding input parameters for each calibration algorithm in the high SNR scenarios H100 and H80. The input parameters for QuartiCal are time-interval sizes Δ_t and for KalCal-Diag and KalCal-Full are process noise σ_Q. Diag-Energy and Full-Energy are their actual values wrapped in parentheses to emphasise this. Highlights indicate the best-performing algorithms and components: **bold** for the best overall algorithm, $(^{*})$ for the best within KalCal-Diag components, and $(^{**})$ for the best within KalCal-Full components.

		L100		L80	
Algorithm	**Parameter**	**Value**	**MSE**	**Value**	**MSE**
QuartiCal	Δ_t	18	0.0247	14	0.0238
Diag-Predictor	σ_Q	0.0518	0.0323	0.0518	0.0342
Diag-Filter	σ_Q	0.0518	0.0285	0.045	0.0306
Diag-Smoother	σ_Q	0.091^{*}	0.0184^{*}	0.091^{*}	0.0203^{*}
Diag-Energy	σ_Q	0.0518	(2.74×10^5)	0.0518	(2.75×10^5)
Full-Predictor	σ_Q	0.0518	0.032	0.0518	0.0339
Full-Filter	σ_Q	0.0518	0.0282	0.0518	0.0303
Full-Smoother	σ_Q	$\mathbf{0.091}^{**}$	$\mathbf{0.0179}^{**}$	$\mathbf{0.091}^{**}$	$\mathbf{0.0198}^{**}$
Full-Energy	σ_Q	0.0518	5.48×10^5	0.0518	5.5×10^5

TABLE B.2: Optimal MSE values and corresponding input parameters for each calibration algorithm in the low SNR scenarios L100 and L80. The input parameters for QuartiCal are time-interval sizes Δ_t and for KalCal-Diag and KalCal-Full are process noise σ_Q. Diag-Energy and Full-Energy are their actual values wrapped in parentheses to emphasise this. Highlights indicate the best-performing algorithms and components: **bold** for the best overall algorithm, $(^{*})$ for the best within KalCal-Diag components, and $(^{**})$ for the best within KalCal-Full components.

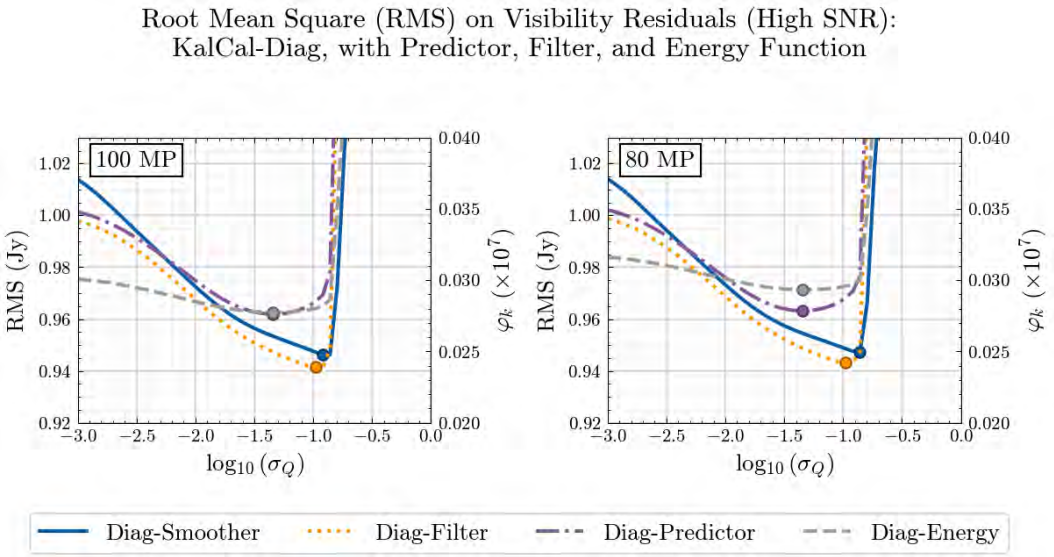

FIGURE B.1: RMS results in the high SNR scenarios L100 (*left*) and L80 (*right*) for different components of KalCal-Diag: Diag-Predictor (*dot-dashed violet*), Diag-Filter (*dotted orange*), Diag-Smoother (*solid blue*), and Diag-Energy (*dashed gray*). Units are in Jy. Dots on the curves represent the respective minimums. Process noise σ_Q is represented on a logarithmic scale along the bottom x-axis. Energy function φ_k values are represented on the right y-axis.

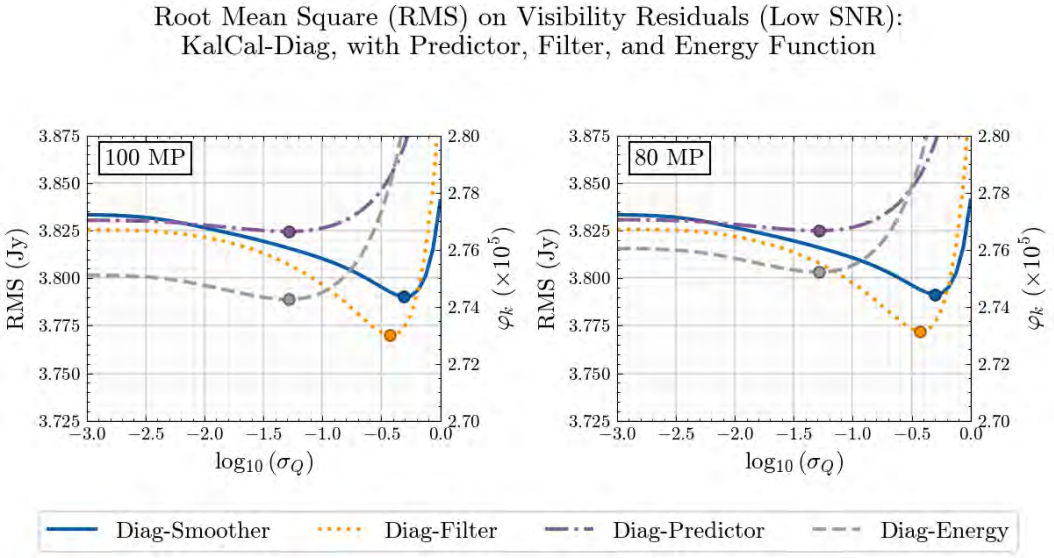

FIGURE B.2: RMS results in the low SNR scenarios H100 (*left*) and H80 (*right*) for different components of KalCal-Diag: Diag-Predictor (*dot-dashed violet*), Diag-Filter (*dotted orange*), Diag-Smoother (*solid blue*), and Diag-Energy (*dashed gray*). Units are in Jy. Dots on the curves represent the respective minimums. Process noise σ_Q is represented on a logarithmic scale along the bottom x-axis. Energy function φ_k values are represented on the right y-axis.

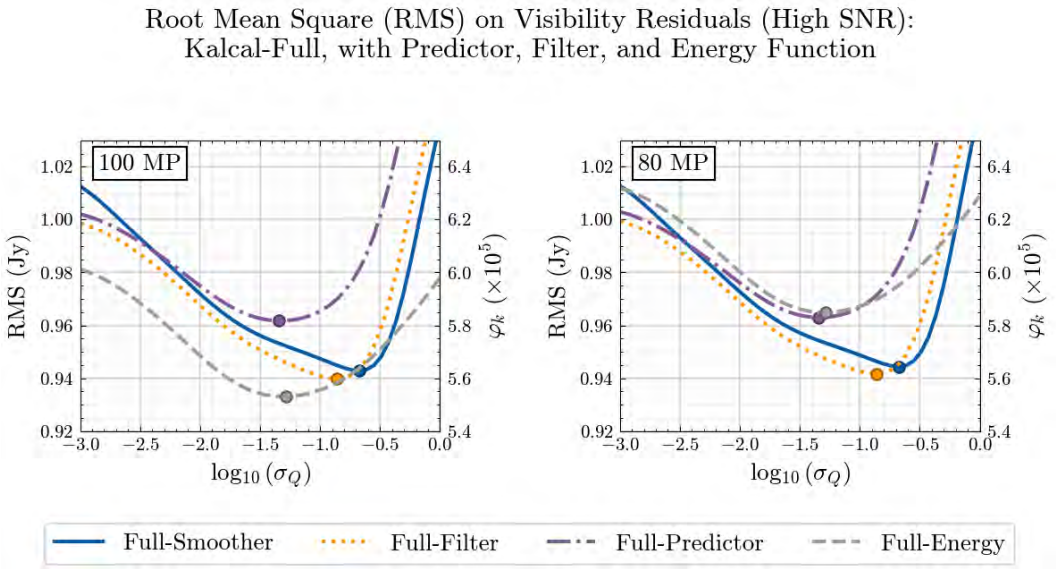

FIGURE B.3: RMS results in the high SNR scenarios H100 (*left*) and H80 (*right*) for different components of KalCal-Full: Full-Predictor (*dot-dashed violet*), Full-Filter (*dotted orange*), Full-Smoother (*solid blue*), and Full-Energy (*dashed gray*). Units are in Jy. Dots on the curves represent the respective minimums. Process noise σ_Q is represented on a logarithmic scale along the bottom x-axis. Energy function φ_k values are represented on the right y-axis.

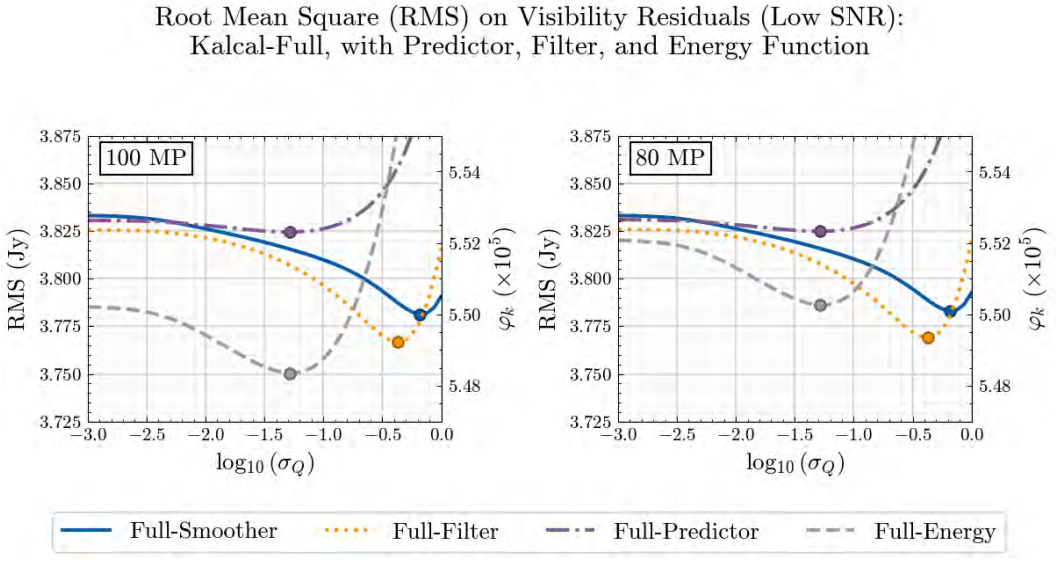

FIGURE B.4: RMS results in the low SNR scenarios L100 (*left*) and L80 (*right*) for different components of KalCal-Full: Full-Predictor (*dot-dashed violet*), Full-Filter (*dotted orange*), Full-Smoother (*solid blue*), and Full-Energy (*dashed gray*). Units are in Jy. Dots on the curves represent the respective minimums. Process noise σ_Q is represented on a logarithmic scale along the bottom x-axis. Energy function φ_k values are represented on the right y-axis.

		H100		H80	
Algorithm	**Parameter**	**Value**	**RMS**	**Value**	**RMS**
QuartiCal	Δ_t	**1**	**0.939**	6	0.986
Diag-Predictor	σ_Q	0.045	0.962	0.045	0.989
Diag-Filter	σ_Q	0.105^{*}	0.941^{*}	0.105^{*}	0.969^{*}
Diag-Smoother	σ_Q	0.121	0.946	0.139	0.973
Diag-Energy	σ_Q	0.045	(2.78×10^5)	0.045	(2.93×10^5)
Full-Predictor	σ_Q	0.045	0.962	0.045	0.989
Full-Filter	σ_Q	0.139^{**}	0.94^{**}	0.139^{**}	0.968^{**}
Full-Smoother	σ_Q	0.212	0.943	**0.212**	**0.97**
Full-Energy	σ_Q	0.0518	(5.53×10^5)	0.0518	(5.85×10^5)

TABLE B.3: Optimal RMS values and corresponding input parameters for each calibration algorithm in the high SNR scenarios H100 and H80. The input parameters for QuartiCal are time-interval sizes Δ_t and for KalCal-Diag and KalCal-Full are process noise σ_Q. Diag-Energy and Full-Energy are their actual values wrapped in parentheses to emphasise this. Highlights indicate the best-performing algorithms and components: **bold** for the best overall algorithm, $(^{*})$ for the best within KalCal-Diag components, and $(^{**})$ for the best within KalCal-Full components.

		L100		L80	
Algorithm	**Parameter**	**Value**	**RMS**	**Value**	**RMS**
QuartiCal	Δ_t	**1**	**3.75**	16	3.83
Diag-Predictor	σ_Q	0.0518	3.82	0.0518	3.83
Diag-Filter	σ_Q	0.373^{*}	3.77^{*}	0.373^{*}	3.78^{*}
Diag-Smoother	σ_Q	0.494	3.79	0.494	3.8
Diag-Energy	σ_Q	0.0518	(2.74×10^5)	0.0518	(2.75×10^5)
Full-Predictor	σ_Q	0.0518	3.82	0.0518	3.83
Full-Filter	σ_Q	0.429^{**}	3.77^{**}	0.429^{**}	3.77^{**}
Full-Smoother	σ_Q	0.655	3.78	**0.655**	**3.79**
Full-Energy	σ_Q	0.0518	(5.48×10^5)	0.0518	(5.5×10^5)

TABLE B.4: Optimal RMS values and corresponding input parameters for each calibration algorithm in the high SNR scenarios L100 and L80. The input parameters for QuartiCal are time-interval sizes Δ_t and for KalCal-Diag and KalCal-Full are process noise σ_Q. Diag-Energy and Full-Energy are their actual values wrapped in parentheses to emphasise this. Highlights indicate the best-performing algorithms and components: **bold** for the best overall algorithm, $(^{*})$ for the best within KalCal-Diag components, and $(^{**})$ for the best within KalCal-Full components.

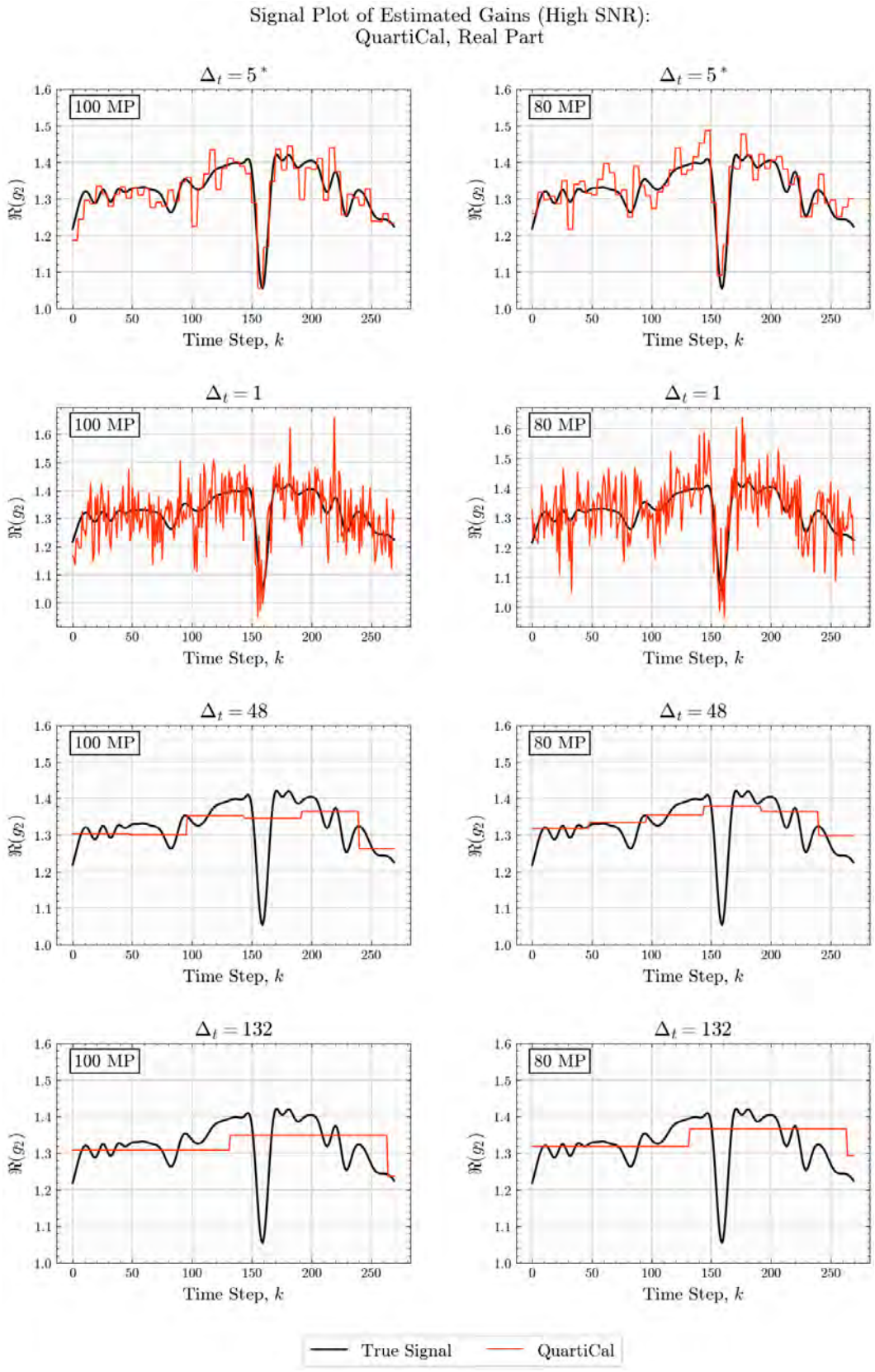

FIGURE B.5: Real part of gain estimates by QuartiCal (*solid red*) in high SNR scenarios H100 (*left column*) and H80 (*right column*) against true gain signal (*solid black*). Sampled different from time-interval size Δ_t solutions for a random antenna gain. (*Top row*) Optimal Δ_t based on minimum MSE indicated by (*).

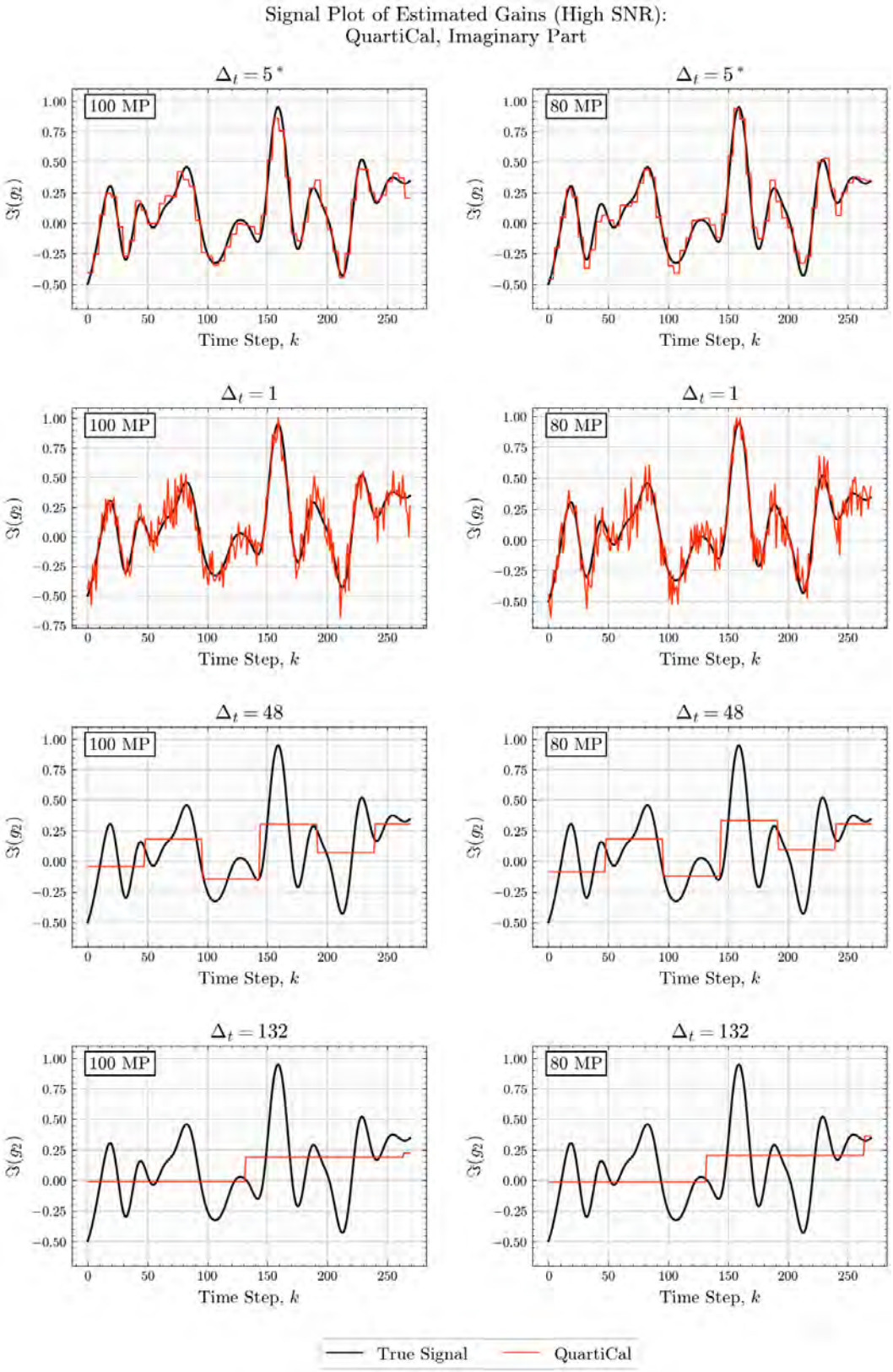

FIGURE B.6: Imaginary part of gain estimates by QuartiCal (*solid red*) in high SNR scenarios H100 (*left column*) and H80 (*right column*) against true gain signal (*solid black*). Sampled from different time-interval size Δ_t solutions for a random antenna gain. (*Top row*) Optimal Δ_t based on minimum MSE indicated by (*).

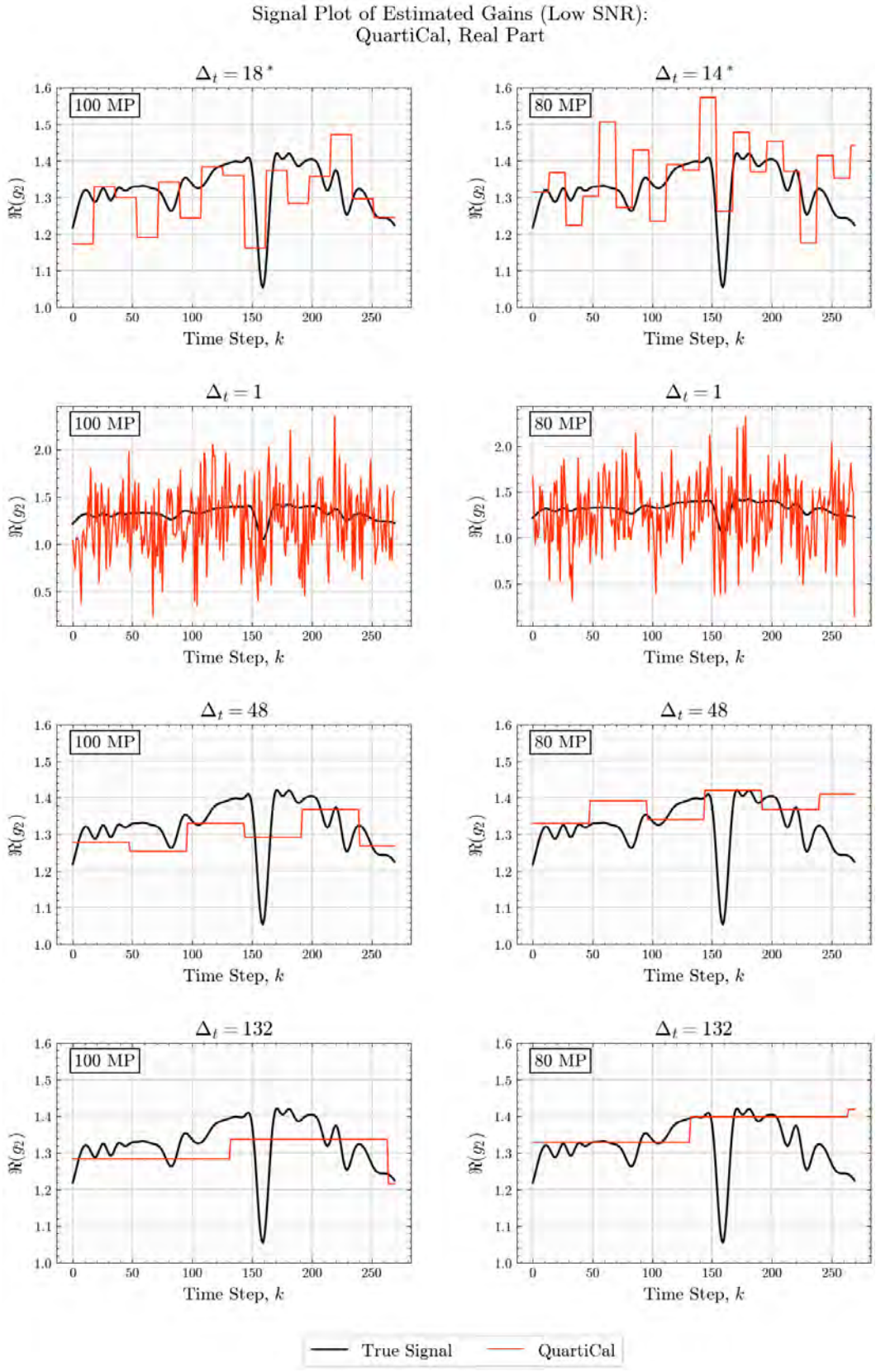

FIGURE B.7: Real part of gain estimates by QuartiCal (*solid red*) in low SNR scenarios L100 (*left column*) and L80 (*right column*) against true gain signal (*solid black*). Sampled from different time-interval size Δ_t solutions for a random antenna gain. (*Top row*) Optimal Δ_t based on minimum MSE indicated by (*).

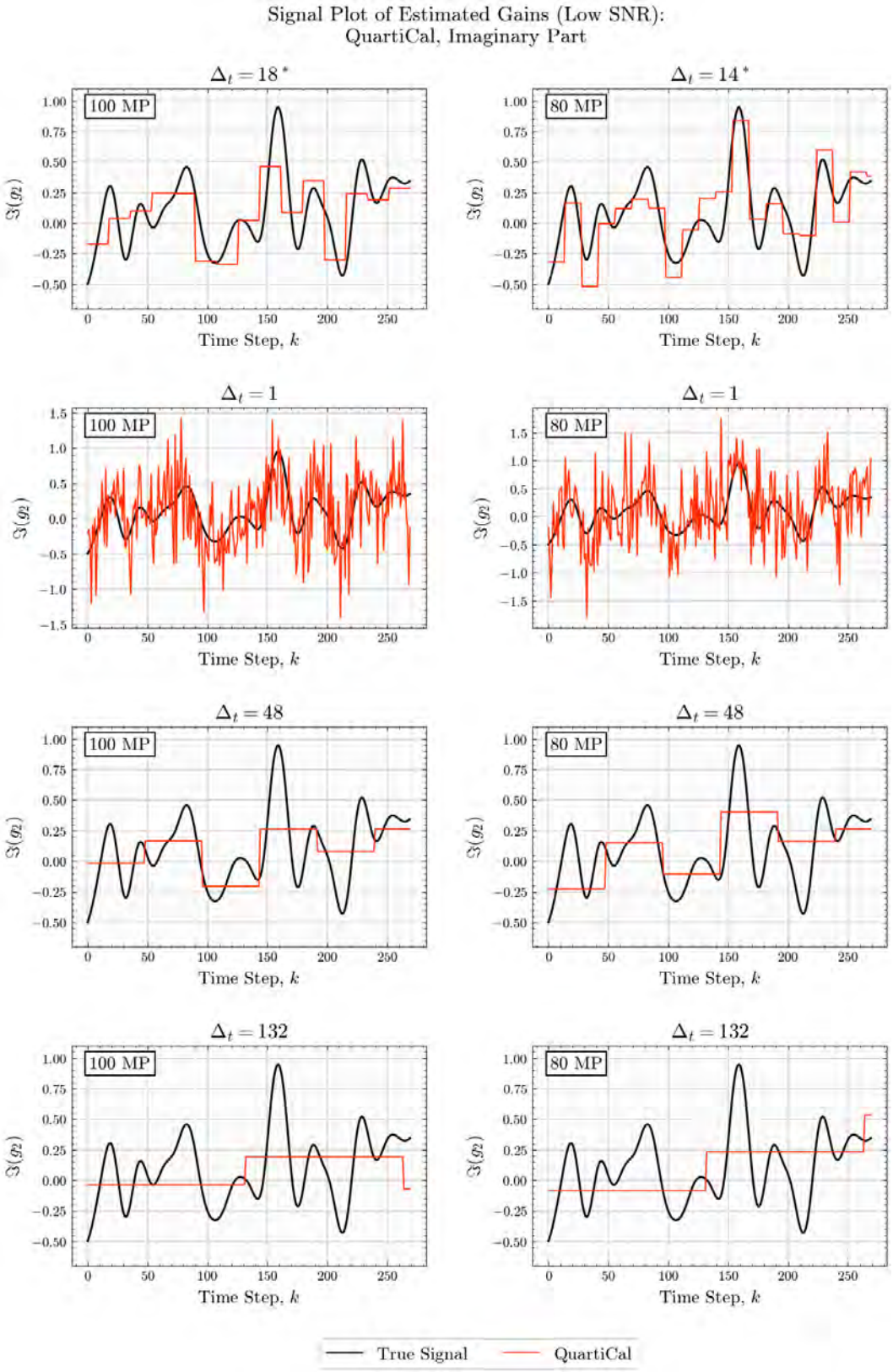

FIGURE B.8: Imaginary part of gain estimates by QuartiCal (*solid red*) in low SNR scenarios L100 (*left column*) and L80 (*right column*) against true gain signal (*solid black*). Sampled from different time-interval size Δ_t solutions for a random antenna gain. (*Top row*) Optimal Δ_t based on minimum MSE indicated by (*).

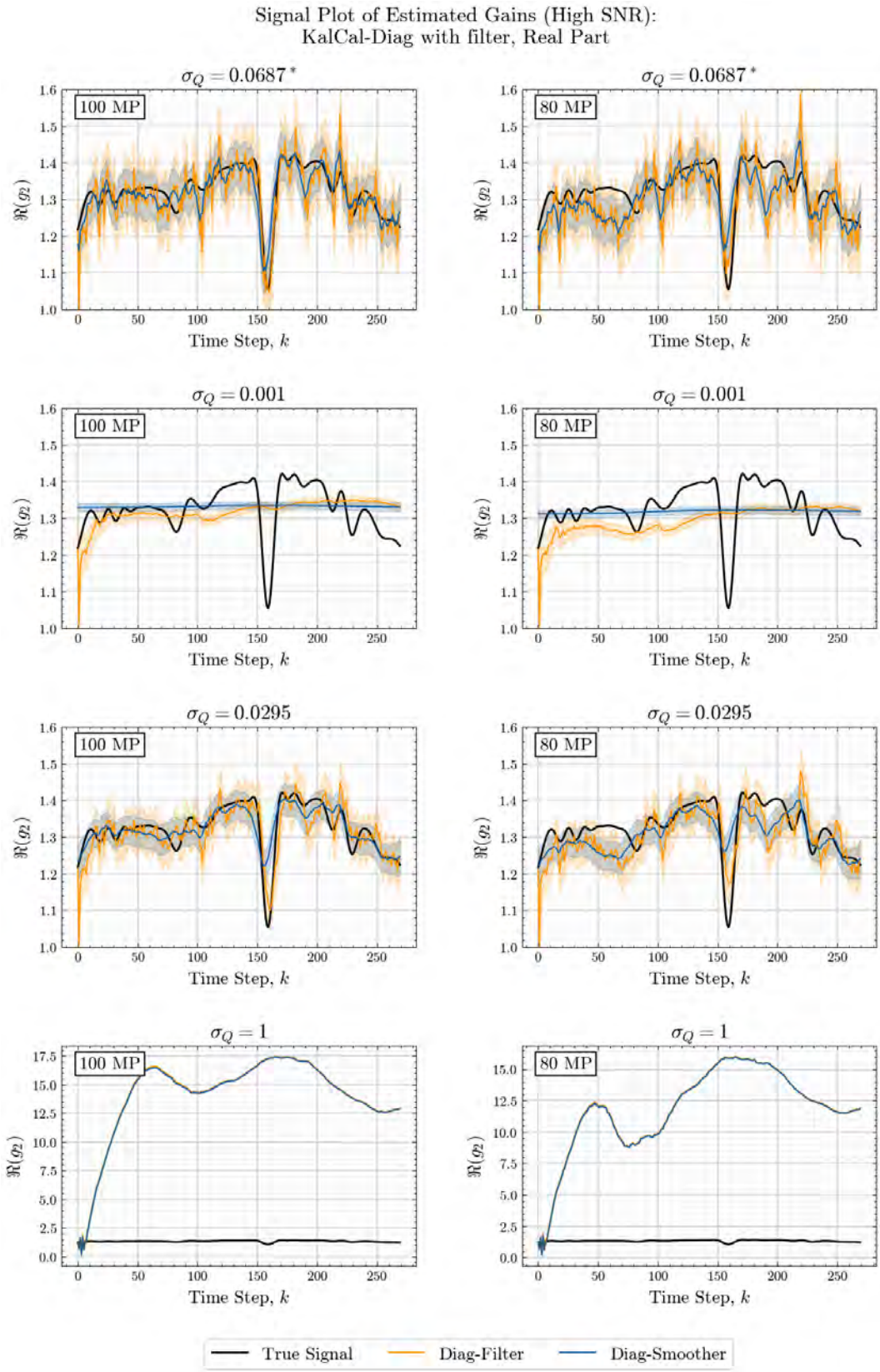

FIGURE B.9: Real part of gain estimates by KalCal-Diag components: Diag-Filter (*solid orange*) and Diag-Smoother (*solid blue*) in high SNR scenarios H100 (*left column*) and H80 (*right column*) against true gain signal (*solid black*). Shaded regions represent the corresponding uncertainty from calculated covariances. Sampled from different process noise σ_Q solutions for a random antenna gain. (*Top row*) Optimal σ_Q based on minimum MSE indicated by (*).

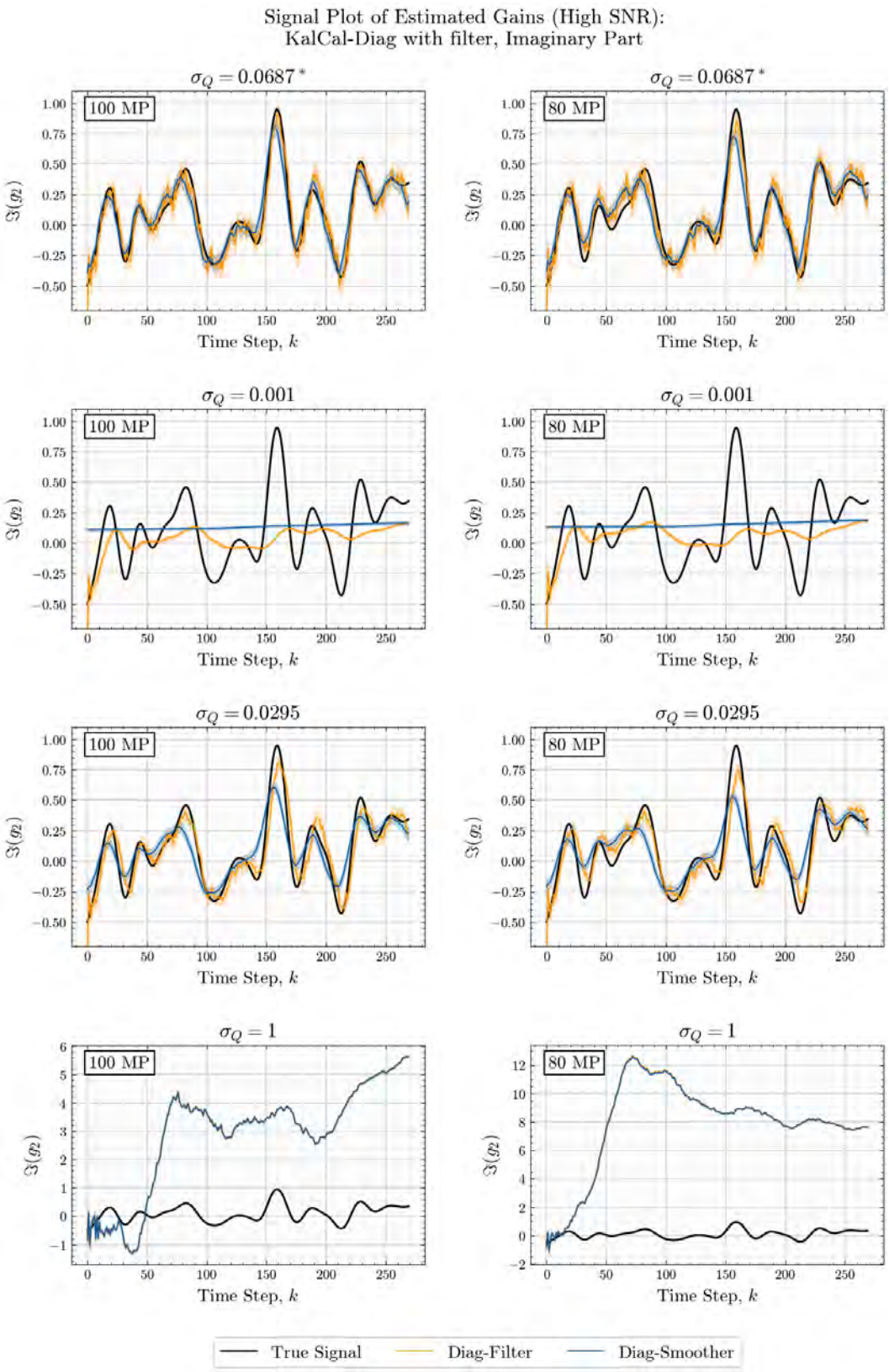

FIGURE B.10: Imaginary part of gain estimates by KalCal-Diag components: Diag-Filter (*solid orange*) and Diag-Smoother (*solid blue*) in high SNR scenarios H100 (*left column*) and H80 (*right column*) against true gain signal (*solid black*). Shaded regions represent the corresponding uncertainty from calculated covariances. Sampled from different process noise σ_Q solutions for a random antenna gain. (*Top row*) Optimal σ_Q based on minimum MSE indicated by $(^*)$.

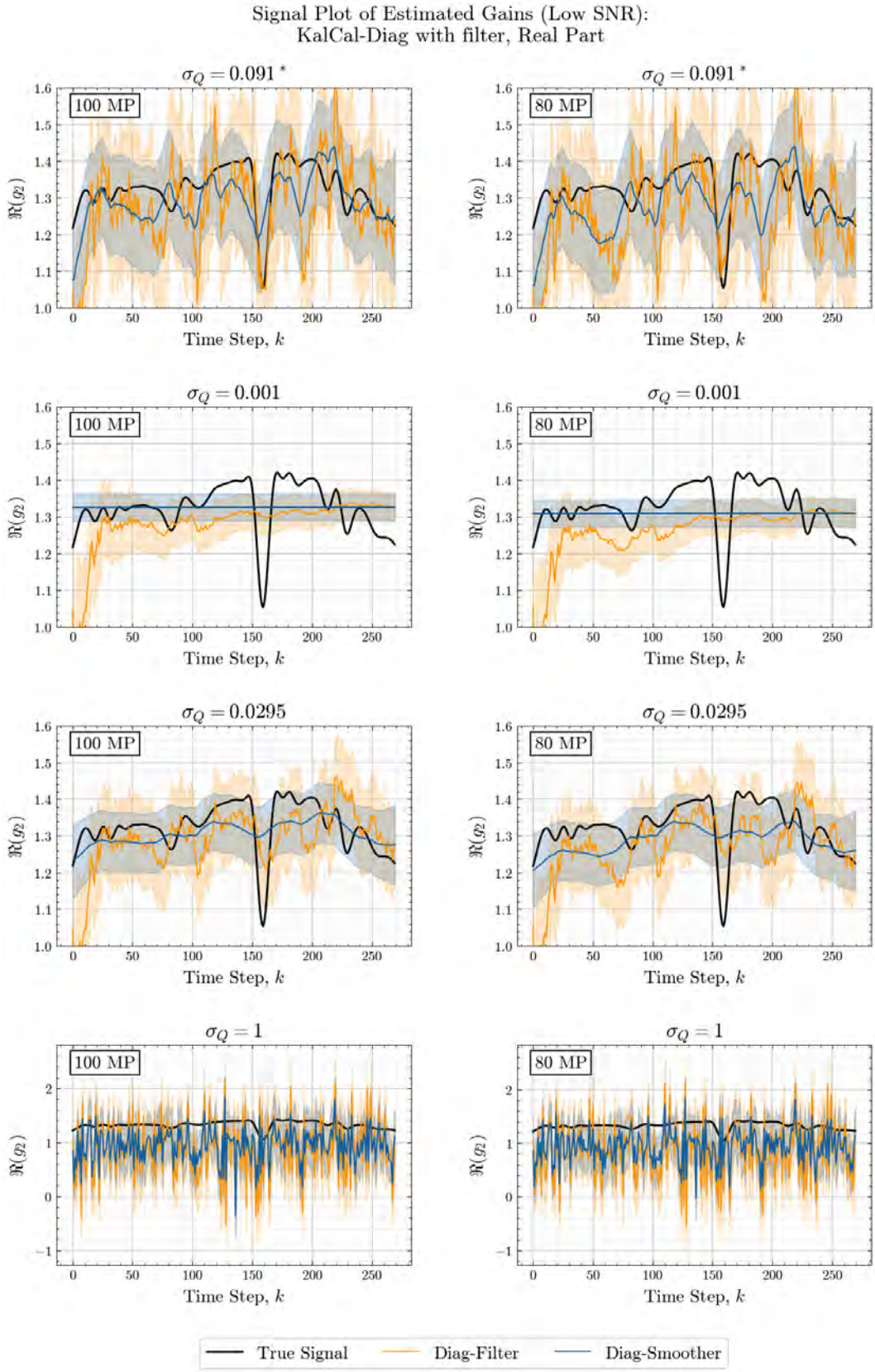

FIGURE B.11: Real part of gain estimates by KalCal-Diag components: Diag-Filter (*solid orange*) and Diag-Smoother (*solid blue*) in low SNR scenarios L100 (*left column*) and L80 (*right column*) against true gain signal (*solid black*). Shaded regions represent the corresponding uncertainty from calculated covariances. Sampled from different process noise σ_Q solutions for a random antenna gain. (*Top row*) Optimal σ_Q based on minimum MSE indicated by (*).

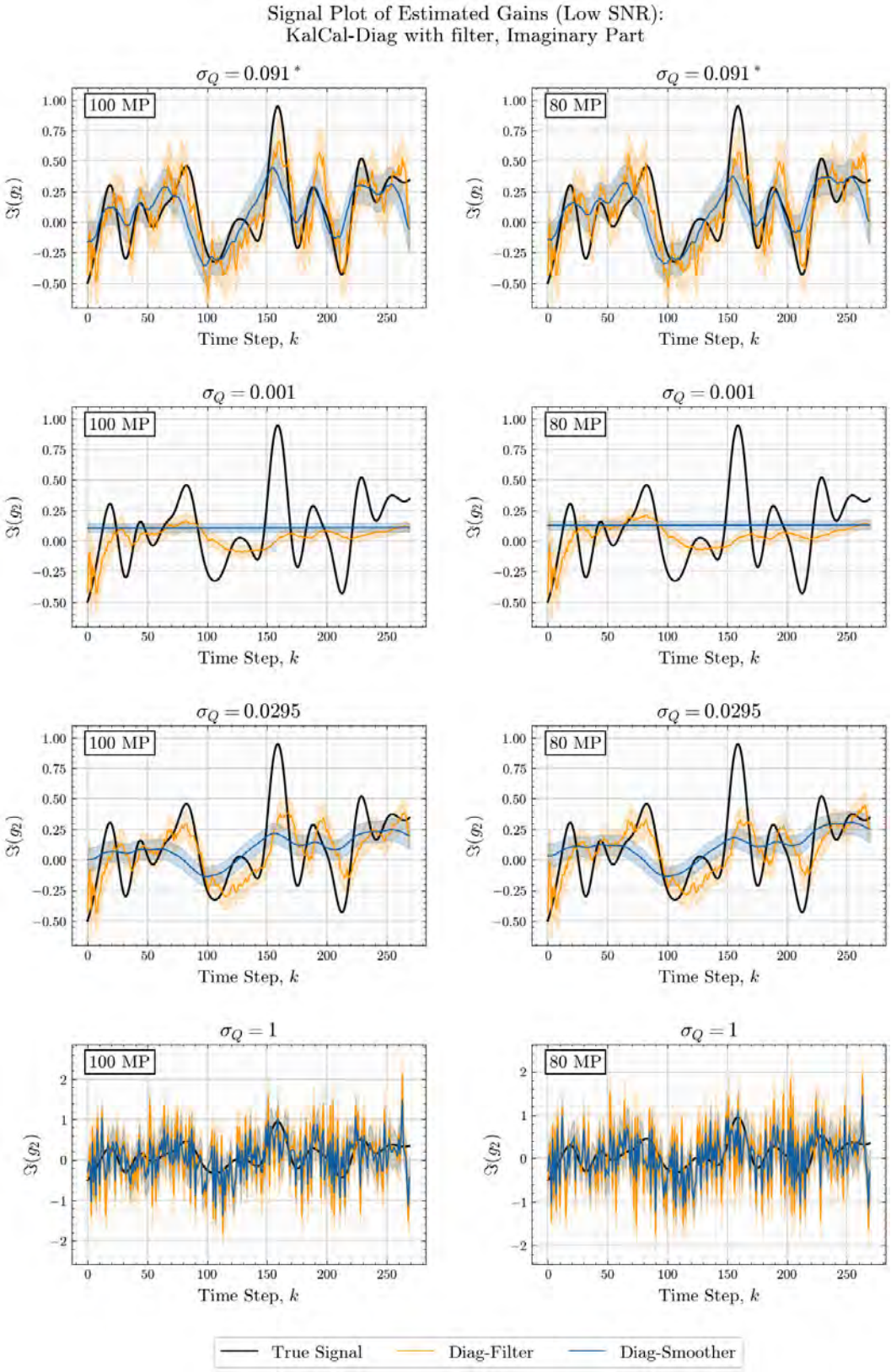

FIGURE B.12: Imaginary part of gain estimates by KalCal-Diag components: Diag-Filter (*solid orange*) and Diag-Smoother (*solid blue*) in low SNR scenarios L100 (*left column*) and L80 (*right column*) against true gain signal (*solid black*). Shaded regions represent the corresponding uncertainty from calculated covariances. Sampled from different process noise σ_Q solutions for a random antenna gain. (*Top row*) Optimal σ_Q based on minimum MSE indicated by (*).

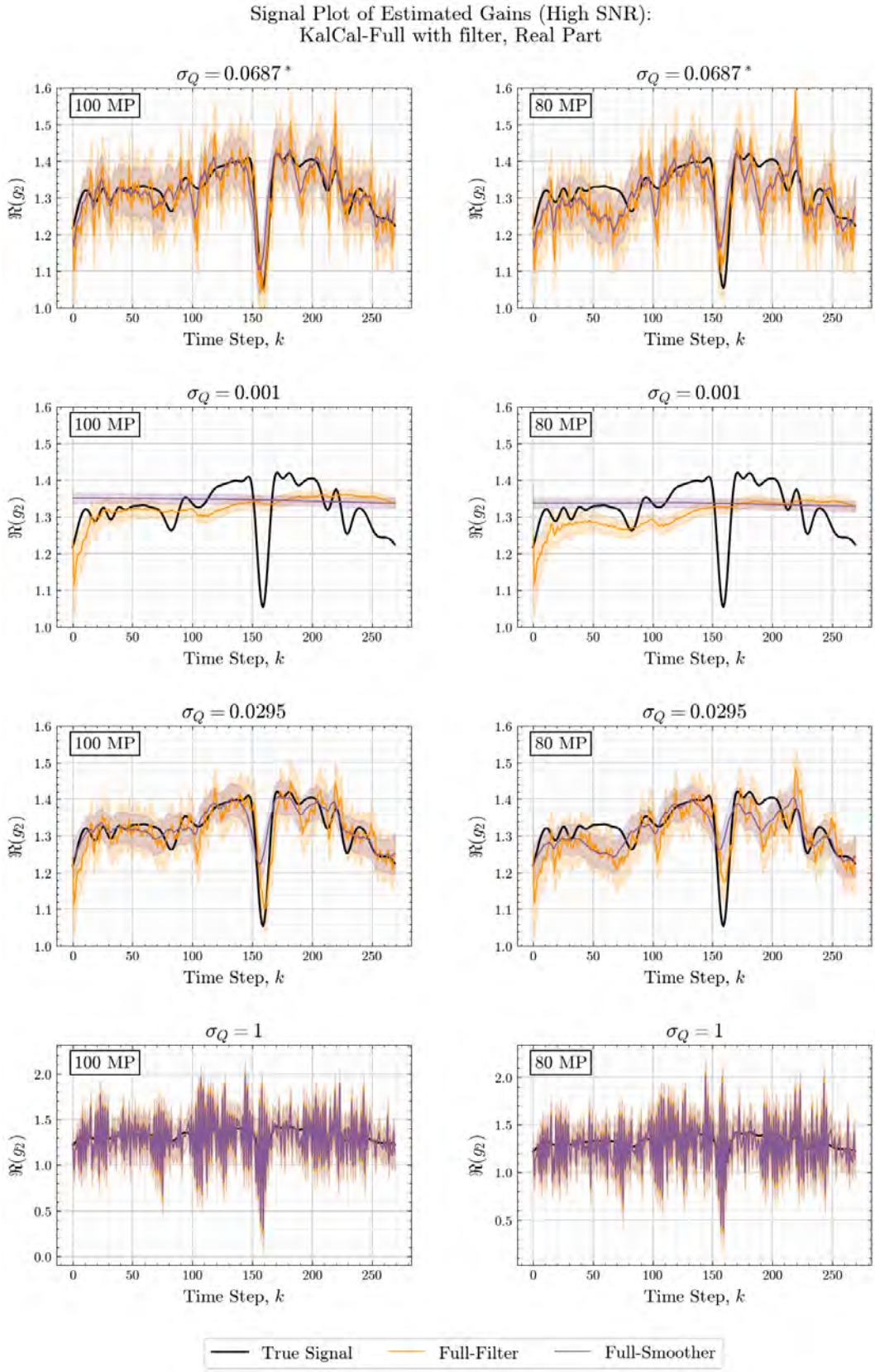

FIGURE B.13: Real part of gain estimates by KalCal-Full components: Full-Filter (*solid orange*) and Full-Smoother (*solid violet*) in high SNR scenarios H100 (*left column*) and H80 (*right column*) against true gain signal (*solid black*). Shaded regions represent the corresponding uncertainty from calculated covariances. Sampled from different process noise σ_Q solutions for a random antenna gain. (*Top row*) Optimal σ_Q based on minimum MSE indicated by (*).

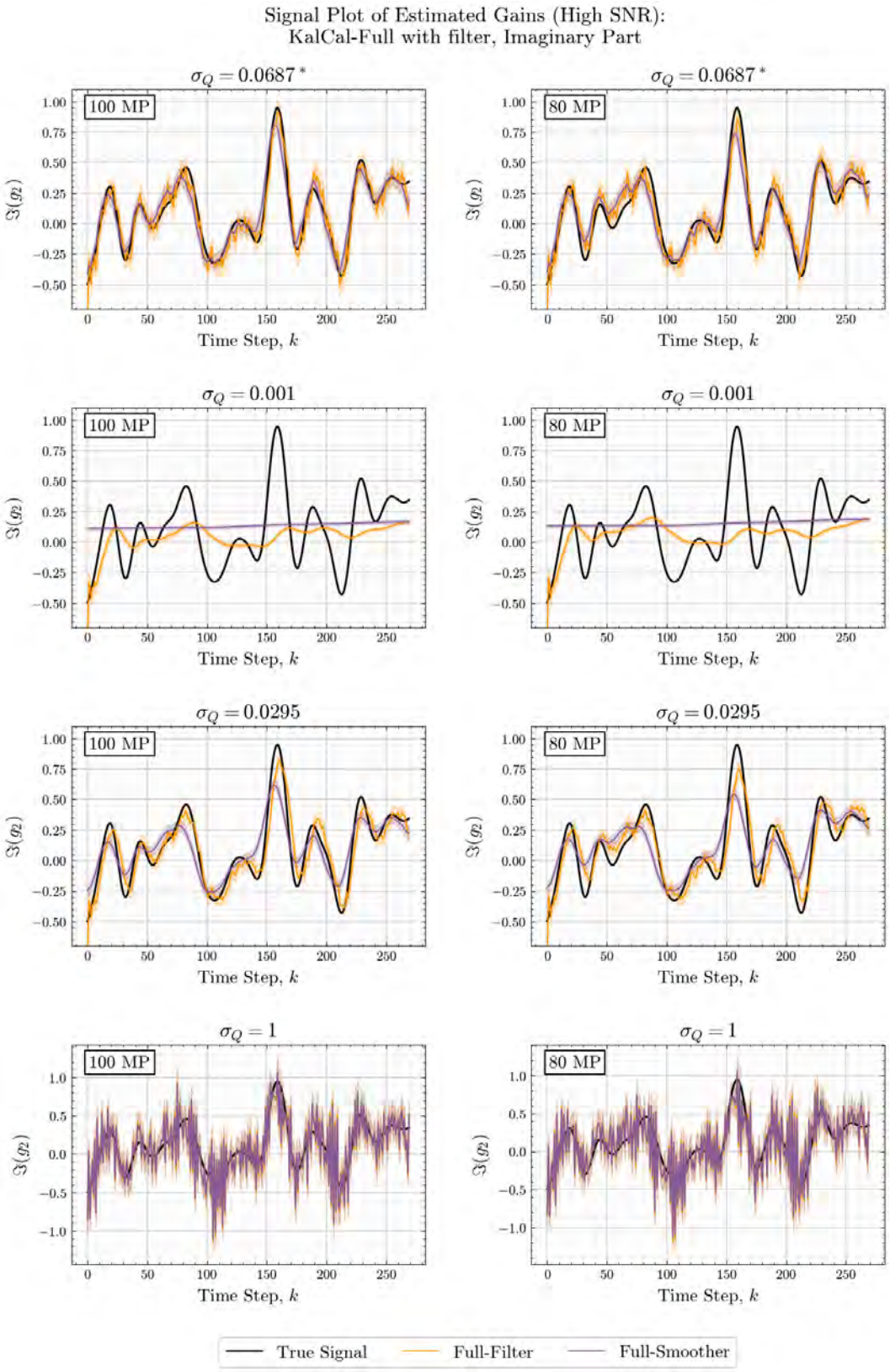

FIGURE B.14: Imaginary part of gain estimates by KalCal-Full components: Full-Filter (*solid orange*) and Full-Smoother (*solid violet*) in high SNR scenarios H100 (*left column*) and H80 (*right column*) against true gain signal (*solid black*). Shaded regions represent the corresponding uncertainty from calculated covariances. Sampled from different process noise σ_Q solutions for a random antenna gain. (*Top row*) Optimal σ_Q based on minimum MSE indicated by (*).

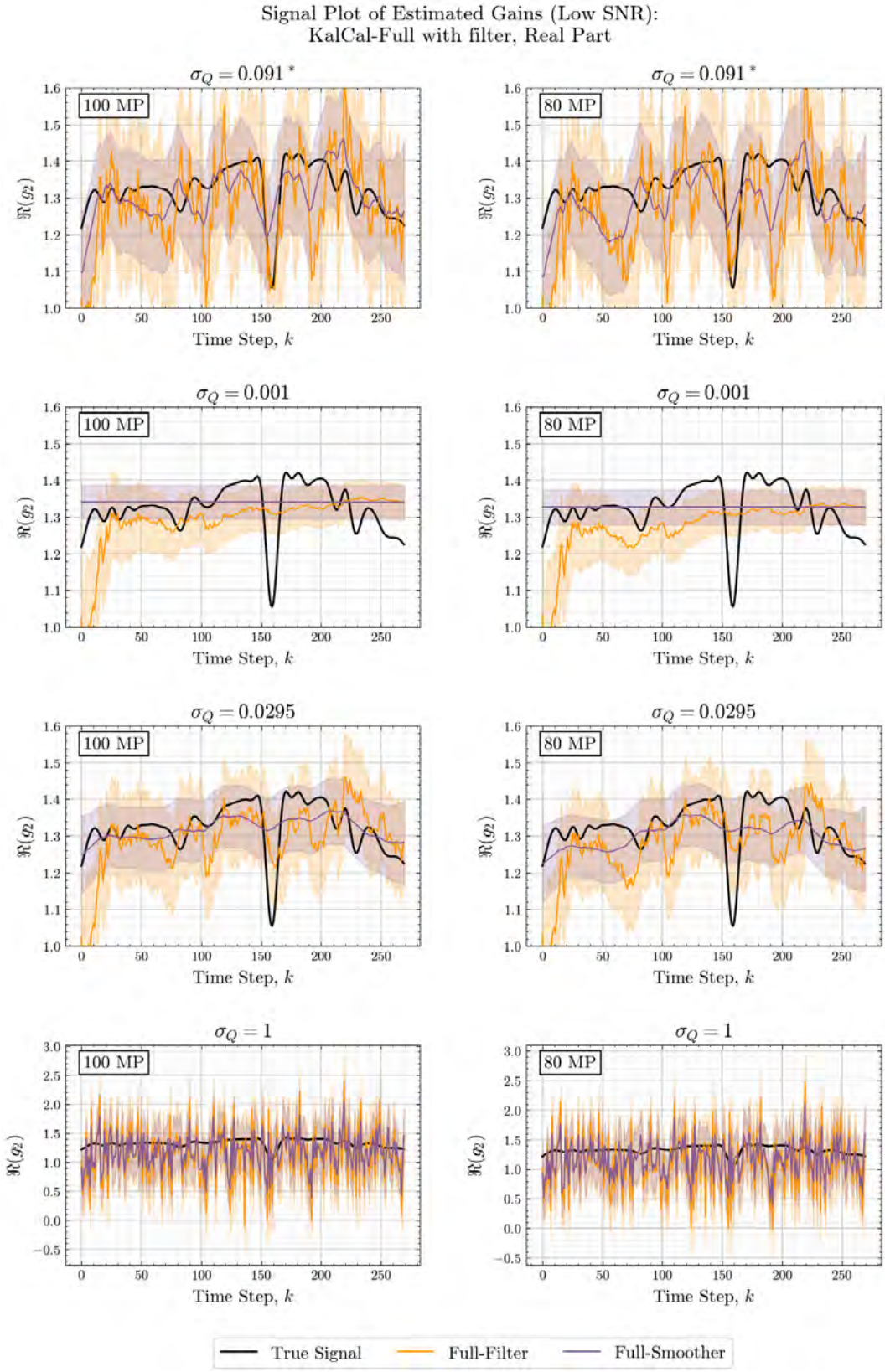

FIGURE B.15: Real part of gain estimates by KalCal-Full components: Full-Filter (*solid orange*) and Full-Smoother (*solid violet*) in low SNR scenarios L100 (*left column*) and H80 (*right column*) against true gain signal (*solid black*). Shaded regions represent the corresponding uncertainty from calculated covariances. Sampled from different process noise σ_Q solutions for a random antenna gain. (*Top row*) Optimal σ_Q based on minimum MSE indicated by (*).

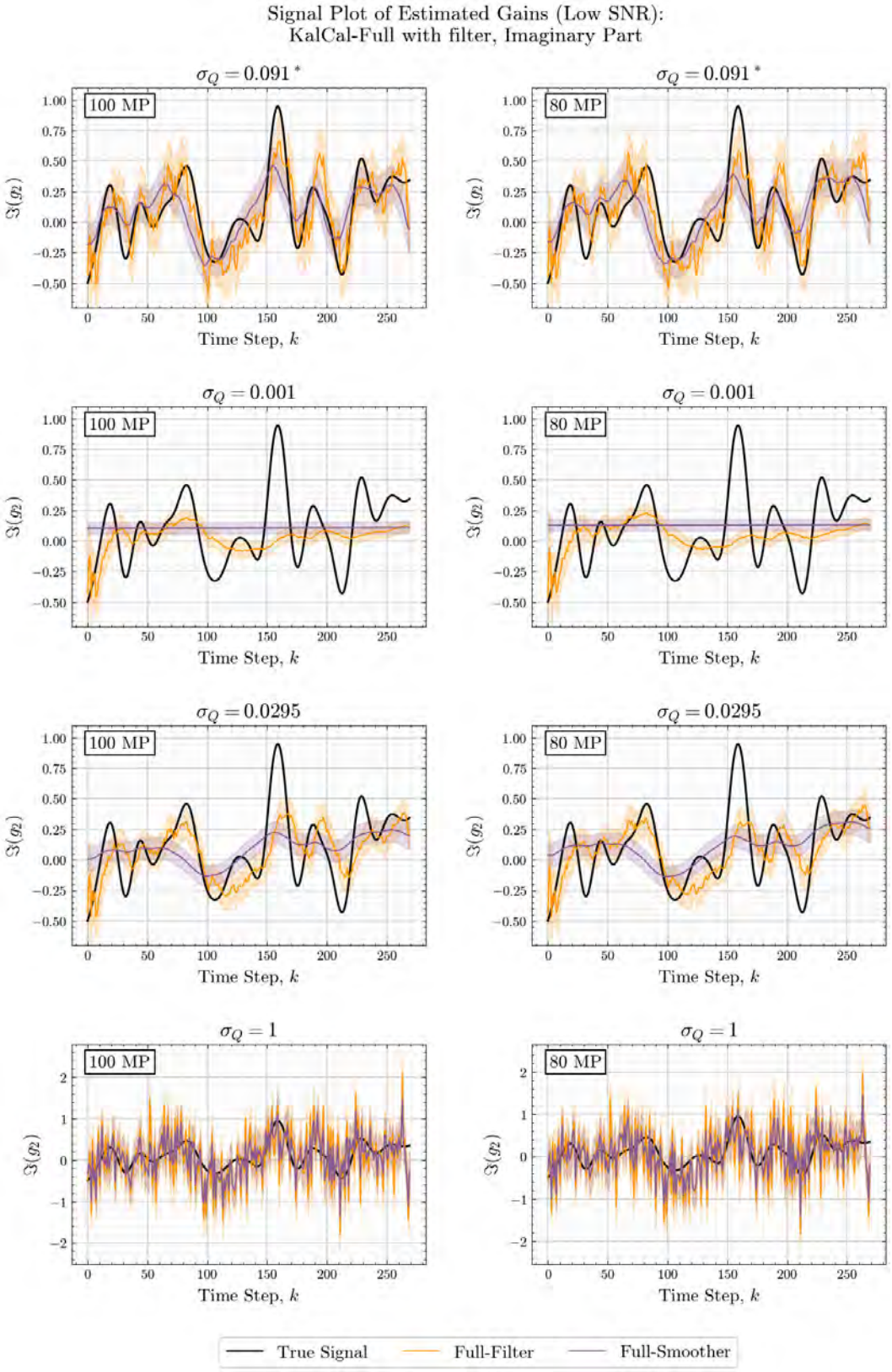

FIGURE B.16: Imaginary part of gain estimates by KalCal-Full components: Full-Filter (*solid orange*) and Full-Smoother (*solid violet*) in low SNR scenarios L100 (*left column*) and H80 (*right column*) against true gain signal (*solid black*). Shaded regions represent the corresponding uncertainty from calculated covariances. Sampled from different process noise σ_Q solutions for a random antenna gain. (*Top row*) Optimal σ_Q based on minimum MSE indicated by (*).

Bibliography

[1] J. S. Kenyon, O. M. Smirnov, T. L. Grobler, and S. J. Perkins, "cubical – fast radio interferometric calibration suite exploiting complex optimization," *Monthly Notices of the Royal Astronomical Society*, vol. 478, no. 2, pp. 2399–2415, May 2018, ISSN: 0035-8711. DOI: 10.1093/mnras/sty1221. eprint: https://academic.oup.com/mnras/article-pdf/478/2/2399/25060220/sty1221.pdf. [Online]. Available: https://doi.org/10.1093/mnras/sty1221.

[2] C. Tasse, "killMS: Direction-dependent radio interferometric calibration package," *Astrophysics Source Code Library*, ascl:2305.005, May 1, 2023. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2023ascl.soft05005T.

[3] J. S. Kenyon, S. Perkins, and O. Smirnov, "QuartiCal - Embarrassingly Parallel Calibration of Radio Interferometer Data Using Numba and Dask," vol. 532, p. 349, Jul. 1, 2022. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2022ASPC..532..349K.

[4] S. Kazemi, S. Yatawatta, S. Zaroubi, P. Lampropoulos, A. G. de Bruyn, L. V. E. Koopmans, *et al.*, "Radio interferometric calibration using the SAGE algorithm," *Monthly Notices of the Royal Astronomical Society*, vol. 414, no. 2, pp. 1656–1666, Jun. 2011, ISSN: 0035-8711. DOI: 10.1111/j.1365-2966.2011.18506.x. eprint: https://academic.oup.com/mnras/article-pdf/414/2/1656/3014218/mnras0414-1656.pdf. [Online]. Available: https://doi.org/10.1111/j.1365-2966.2011.18506.x.

[5] S. Salvini and S. J. Wijnholds, "StEFCal — An Alternating Direction Implicit method for fast full polarization array calibration," in *2014 XXXIth URSI General Assembly and Scientific Symposium (URSI GASS)*, Aug. 2014, pp. 1–4. DOI: 10.1109/URSIGASS.2014.6930038.

[6] I. Ridpath, Ed., *A Dictionary of Astronomy*, Third edition. Oxford: Oxford University Press, 2018, ISBN: 978-0-19-185119-3.

[7] E. J. Dasch and S. J. O'Meara, *A Dictionary of Space Exploration*, 3 ed. Oxford: Oxford University Press, 2018, ISBN: 978-0-19-184276-4.

[8] South African Radio Astronomy Observatory - SARAO. "MeerKAT radio telescope inaugurated in South Africa – reveals clearest view yet of center of the Milky Way." (), [Online]. Available: https://www.sarao.ac.za/media-releases/meerkat-radio-telescope-inaugurated-in-south-africa-reveals-clearest-view-yet-of-center-of-the-milky-way/ (visited on 01/25/2023).

[9] A. Morris. "Mysterious dashes revealed in Milky Way's center," NORTHWESTERN Now. (Jun. 2, 2023), [Online]. Available: https://news.northwestern.edu/stories/2023/06/mysterious-dashes-revealed-in-milky-ways-center/ (visited on 06/13/2023).

[10] I. M. Dunham, "Big data: A revolution that will transform how we live, work, and think," *The AAG Review of Books*, vol. 3, no. 1, pp. 19–21, 2015. DOI: 10.1080/2325548X.2015.985533. eprint: https://doi.org/10.1080/2325548X.2015.985533. [Online]. Available: https://doi.org/10.1080/2325548X.2015.985533.

[11] C. Tasse, "Nonlinear Kalman filters for calibration in radio interferometry," *Astronomy and Astrophysics*, vol. 566, A127, A127, Jun. 2014. DOI: 10.1051/0004-6361/201423503. arXiv: 1403.6308 [astro-ph.IM].

[12] S. Salvini and S. J. Wijnholds, "Fast gain calibration in radio astronomy using alternating direction implicit methods: Analysis and applications," *Astronomy & Astrophysics*, vol. 571, A97, Nov. 1, 2014, ISSN: 0004-6361, 1432-0746. DOI: 10.1051/0004-6361/201424487.

[13] X. Cai, L. Pratley, and J. D. McEwen, "Online radio interferometric imaging: Assimilating and discarding visibilities on arrival," *Monthly Notices of the Royal Astronomical Society*, vol. 485, no. 4, pp. 4559–4572, Jun. 1, 2019, ISSN: 0035-8711. DOI: 10.1093/mnras/stz704.

[14] South African Radio Astronomy Observatory - SARAO. "New MeerKAT radio image reveals complex heart of the Milky Way." (), [Online]. Available: https://www.sarao.ac.za/media-releases/new-meerkat-radio-image-reveals-complex-heart-of-the-milky-way/ (visited on 09/01/2023).

[15] O. M. Smirnov, "Revisiting the radio interferometer measurement equation: II. Calibration and direction-dependent effects," *Astronomy and Astrophysics*, vol. 527, no. 14, 2011, ISSN: 00046361. DOI: 10.1051/0004-6361/201116434.

[16] O. M. Smirnov, "Revisiting the radio interferometer measurement equation: I. A full-sky Jones formalism," *Astronomy and Astrophysics*, vol. 527, no. 14, 2011, ISSN: 00046361. DOI: 10.1051/0004-6361/201016082.

[17] A. Butterfield and J. E. Szymanski, *A Dictionary of Electronics and Electrical Engineering* (Oxford Quick Reference), 5th edition. New York, NY: Oxford University Press, 2018, 712 pp., ISBN: 978-0-19-872572-5.

[18] J. Daintith, E. Wright, and O. U. Press, Eds., *A Dictionary of Computing* (Oxford Paperback Reference), 6th ed. Oxford ; New York: Oxford University Press, 2008, 567 pp., ISBN: 978-0-19-923400-4.

[19] H. Spreeuw, S. Yatawatta, B. Van Werkhoven, and F. Diblen, "Scaling performance of the SAGECal calibration package: from LOFAR to SKA," *arXiv e-prints*, arXiv:2010.16209,

arXiv:2010.16209, Oct. 2020. DOI: 10.48550/arXiv.2010.16209. arXiv: 2010.16209 [astro-ph.IM].

[20] U. M. Sob, H. L. Bester, O. M. Smirnov, J. S. Kenyon, and C. Russeeawon, "Solution intervals considered harmful: on the optimality of radio interferometric gain solutions," *Monthly Notices of the Royal Astronomical Society*, vol. 504, no. 2, pp. 1714–1732, Apr. 2021, ISSN: 0035-8711. DOI: 10.1093/mnras/stab928. eprint: https://academic.oup.com/mnras/article-pdf/504/2/1714/37412581/stab928.pdf. [Online]. Available: https://doi.org/10.1093/mnras/stab928.

[21] A. Neumaier, "Solving Ill-Conditioned and Singular Linear Systems: A Tutorial on Regularization," *SIAM Review*, vol. 40, no. 3, pp. 636–666, Jan. 1998, ISSN: 0036-1445. DOI: 10.1137/S0036144597321909.

[22] T. L. Grobler, C. D. Nunhokee, O. M. Smirnov, A. J. van Zyl, and A. G. de Bruyn, "Calibration artefacts in radio interferometry – I. Ghost sources in Westerbork Synthesis Radio Telescope data," *Monthly Notices of the Royal Astronomical Society*, vol. 439, no. 4, pp. 4030–4047, Mar. 2014, ISSN: 0035-8711. DOI: 10.1093/mnras/stu268. eprint: https://academic.oup.com/mnras/article-pdf/439/4/4030/3996018/stu268.pdf. [Online]. Available: https://doi.org/10.1093/mnras/stu268.

[23] A. H. Jazwinski, *Stochastic Processes and Filtering Theory*. San Diego, CA: Elsevier, 1970, vol. 64, 376 pp., ISBN: 978-0-12-381550-7. DOI: 10.1016/S0076-5392(09)X6022-4.

[24] S. Särkkä, *Bayesian Filtering and Smoothing*. Cambridge: Cambridge University Press, 2013, 1-232, ISBN: 978-1-139-34420-3. DOI: 10.1017/CBO9781139344203.

[25] H. E. Rauch, C. T. Striebel, and F. Tung, "Maximum likelihood estimates of linear dynamic systems," *AIAA Journal*, vol. 3, no. 8, pp. 1445–1450, Aug. 1965. DOI: 10.2514/3.3166.

[26] J. J. Condon and M. R. Scott, *Essential Radio Astronomy*. Princeton: Princeton University Press, 2016, ISBN: 978-0-691-13779-7.

[27] B. F. Burke, F. Graham-Smith, and P. N. Wilkinson, *An Introduction to Radio Astronomy*. Cambridge: Cambridge University Press, Jul. 31, 2019, 540 pp., ISBN: 978-1-316-98750-6. DOI: 10.1017/9781316987506.

[28] A. R. Thompson, J. M. Moran, and J. Swenson George W., *Interferometry and Synthesis in Radio Astronomy, 3rd Edition*. 2017. DOI: 10.1007/978-3-319-44431-4.

[29] J. D. Monnier and R. J. Allen, "Radio and optical interferometry: Basic observing techniques and data analysis," in *Planets, Stars and Stellar Systems: Volume 2: Astronomical Techniques, Software, and Data*, T. D. Oswalt and H. E. Bond, Eds. Dordrecht: Springer Netherlands, 2013, pp. 325–373, ISBN: 978-94-007-5618-2. DOI: 10.1007/978-94-007-5618-2_7. [Online]. Available: https://doi.org/10.1007/978-94-007-5618-2_7.

[30] R. Rennie, Ed., *A Dictionary of Physics* (Oxford Quick Reference), Eighth edition. Oxford ; New York, NY: Oxford University Press, 2019, 696 pp., ISBN: 978-0-19-882147-2.

[31] J. P. Hamaker, J. D. Bregman, and R. J. Sault, "Understanding radio polarimetry. I. Mathematical foundations," *Astronomy and Astrophysics Supplement Series*, vol. 117, no. 1, pp. 137–147, May 1996, ISSN: 0365-0138, 1286-4846. DOI: 10.1051/aas:1996146.

[32] J. V. Candy, *Bayesian Signal Processing: Classical, Modern, and Particle Filtering Methods*. Hoboken, NJ: John Wiley & Sons, Jul. 22, 2016, 445 pp., ISBN: 978-1-119-12545-7. DOI: 10.1002/9781119125495.

[33] P. J. Schreier and L. L. Scharf, *Statistical Signal Processing of Complex-Valued Data: The Theory of Improper and Noncircular Signals*. Cambridge: Cambridge University Press, 2010, ISBN: 978-0-511-67772-4.

[34] J. A. Högbom, "Aperture Synthesis with a Non-Regular Distribution of Interferometer Baselines," *Astronomy and Astrophysics Supplement Series*, vol. 15, p. 417, Jun. 1, 1974, ISSN: 0365-01380004-6361. [Online]. Available: https://ui.adsabs.harvard.edu/abs/1974A&AS...15..417H.

[35] A. R. Offringa, B. McKinley, N. Hurley-Walker, F. H. Briggs, R. B. Wayth, D. L. Kaplan, *et al.*, "wsclean: an implementation of a fast, generic wide-field imager for radio astronomy," *Monthly Notices of the Royal Astronomical Society*, vol. 444, no. 1, pp. 606–619, Aug. 2014, ISSN: 0035-8711. DOI: 10.1093/mnras/stu1368. eprint: https://academic.oup.com/mnras/article-pdf/444/1/606/18501772/stu1368.pdf. [Online]. Available: https://doi.org/10.1093/mnras/stu1368.

[36] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, Mar. 8, 2004, ISBN: 9780511804441. DOI: 10.1017/CBO9780511804441.

[37] U. M. Sob, H. L. Bester, O. M. Smirnov, J. S. Kenyon, and T. L. Grobler, "Radio interferometric calibration using a complex Student's t-distribution and Wirtinger derivatives," *Monthly Notices of the Royal Astronomical Society*, vol. 491, no. 1, pp. 1026–1042, Oct. 2019, ISSN: 0035-8711. DOI: 10.1093/mnras/stz3037. eprint: https://academic.oup.com/mnras/article-pdf/491/1/1026/31101959/stz3037.pdf. [Online]. Available: https://doi.org/10.1093/mnras/stz3037.

[38] O. M. Smirnov and C. Tasse, "Radio interferometric gain calibration as a complex optimization problem," *Monthly Notices of the Royal Astronomical Society*, vol. 449, no. 3, pp. 2668–2684, May 2015. DOI: 10.1093/mnras/stv418. arXiv: 1502.06974 [astro-ph.IM].

[39] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006, ISBN: 0387310738. DOI: 10.5555/1162264.

[40] S. J. Wijnholds, T. L. Grobler, and O. M. Smirnov, "Calibration artefacts in radio interferometry - II. Ghost patterns for irregular arrays," *Monthly Notices of the Royal Astronomical Society*, vol. 457, no. 3, pp. 2331–2354, Apr. 2016. DOI: 10.1093/mnras/stw118.

[41] T. L. Grobler, A. J. Stewart, S. J. Wijnholds, J. S. Kenyon, and O. M. Smirnov, "Calibration artefacts in radio interferometry – III. Phase-only calibration and primary beam correction," *Monthly Notices of the Royal Astronomical Society*, vol. 461, no. 3, pp. 2975–2992, Jun. 2016, ISSN: 0035-8711. DOI: 10.1093/mnras/stw1437. eprint: https://academic.oup.com/mnras/article-pdf/461/3/2975/8107666/stw1437.pdf. [Online]. Available: https://doi.org/10.1093/mnras/stw1437.

[42] R. A. Horn and C. R. Johnson, *Matrix Analysis*, 2nd ed. Cambridge ; New York: Cambridge University Press, 2012, 643 pp., ISBN: 978-0-521-83940-2.

[43] A. G. Wilber, M. Johnston-Hollitt, S. W. Duchesne, C. Tasse, H. Akamatsu, H. Intema, *et al.*, "ASKAP reveals giant radio halos in two merging SPT galaxy clusters," *Publications of the Astronomical Society of Australia*, vol. 37, e040, e040, Oct. 2020. DOI: 10.1017/pasa.2020.34.

[44] M. Brienza, T. W. Shimwell, F. de Gasperin, I. Bikmaev, A. Bonafede, A. Botteon, *et al.*, "A snapshot of the oldest active galactic nuclei feedback phases," *Nature Astronomy*, vol. 5, pp. 1261–1267, Dec. 2021. DOI: 10.1038/s41550-021-01491-0. arXiv: 2110.09189 [astro-ph.HE].

[45] V. Parekh, R. Kincaid, B. Hugo, A. Ramaila, and N. Oozeer, "Third-generation calibrations for meerkat observation," *Galaxies*, vol. 9, no. 4, 2021, ISSN: 2075-4434. DOI: 10.3390/galaxies9040090. [Online]. Available: https://www.mdpi.com/2075-4434/9/4/90.

[46] A. Botteon, R. J. van Weeren, G. Brunetti, F. Vazza, T. W. Shimwell, M. Brüggen, *et al.*, "Magnetic fields and relativistic electrons fill entire galaxy cluster," *Science Advances*, vol. 8, no. 44, eabq7623, 2022. DOI: 10.1126/sciadv.abq7623. eprint: https://www.science.org/doi/pdf/10.1126/sciadv.abq7623. [Online]. Available: https://www.science.org/doi/abs/10.1126/sciadv.abq7623.

[47] T. A. Herring, J. L. Davis, and I. I. Shapiro, "Geodesy by radio interferometry: The application of Kalman Filtering to the analysis of very long baseline interferometry data," *Journal of Geophysical Research*, vol. 95, no. B8, p. 12 561, 1990, ISSN: 0148-0227. DOI: 10.1029/JB095iB08p12561.

[48] B. Soja, T. Nilsson, M. Karbon, F. Zus, G. Dick, Z. Deng, *et al.*, "Tropospheric delay determination by Kalman filtering VLBI data," *Earth, Planets and Space*, vol. 67, 144, p. 144, Sep. 2015. DOI: 10.1186/s40623-015-0293-0.

[49] T. Nilsson, B. Soja, M. Karbon, R. Heinkelmann, and H. Schuh, "Application of Kalman filtering in VLBI data analysis," *Earth, Planets and Space*, vol. 67, no. 1, p. 136, Aug. 25, 2015, ISSN: 1880-5981. DOI: 10.1186/s40623-015-0307-y.

[50] S. Särkkä, "Unscented Rauch–Tung–Striebel Smoother," *IEEE Transactions on Automatic Control*, vol. 53, no. 3, pp. 845–849, Apr. 2008, ISSN: 1558-2523. DOI: 10.1109/TAC.2008.919531.

[51] Y. Wang, "Position estimation using extended Kalman Filter and RTS-smoother in a GPS receiver," pp. 1718–1721, Oct. 2012. DOI: 10.1109/CISP.2012.6469979.

[52] S. Kubo, S. Kato, K. Nakamura, N. Kodera, and S. Takada, "Resolving the data asynchronicity in high-speed atomic force microscopy measurement via the Kalman Smoother," *Scientific Reports*, vol. 10, no. 1, p. 18 393, 1 Oct. 27, 2020, ISSN: 2045-2322. DOI: 10.1038/s41598-020-75463-1.

[53] C. Abbondanza, T. M. Chin, R. S. Gross, M. B. Heflin, J. W. Parker, B. S. Soja, *et al.*, "JTRF2014, the JPL Kalman filter and smoother realization of the International Terrestrial Reference System: JTRF2014," *Journal of Geophysical Research: Solid Earth*, vol. 122, no. 10, pp. 8474–8510, Oct. 2017, ISSN: 21699313. DOI: 10.1002/2017JB014360.

[54] X. Wu, C. Abbondanza, Z. Altamimi, T. M. Chin, X. Collilieux, R. S. Gross, *et al.*, "KALREF—A Kalman filter and time series approach to the International Terrestrial Reference Frame realization," *Journal of Geophysical Research (Solid Earth)*, vol. 120, no. 5, pp. 3775–3802, May 2015. DOI: 10.1002/2014JB011622.

[55] A. Saberi, A. Stoorvogel, and P. Sannuti, *Filtering theory : with applications to fault detection, isolation, and estimation* (Systems and control : foundations and applications), English. Switzerland: Birkhäuser Verlag, 2007, ISBN: 0-8176-4301-X. DOI: 10.1007/978-0-8176-4564-9.

[56] D. H. Dini and D. P. Mandic, "Class of Widely Linear Complex Kalman Filters," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 5, pp. 775–786, May 2012, ISSN: 2162-237X, 2162-2388. DOI: 10.1109/TNNLS.2012.2189893.

[57] W. Dang and L. L. Scharf, "Extensions to the Theory of Widely Linear Complex Kalman Filtering," *IEEE Transactions on Signal Processing*, vol. 60, no. 12, pp. 6669–6674, Dec. 2012. DOI: 10.1109/TSP.2012.2214213. arXiv: 1105.5432 [cs.SY].

[58] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Fluids Engineering, Transactions of the ASME*, vol. 82, no. 1, pp. 35–45, 1960, ISSN: 1528901X. DOI: 10.1115/1.3662552.

[59] N. Wiener, *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. The MIT Press, 1964, ISBN: 978-0-262-73005-1.

[60] V. Kučera, "Rudolf E. Kalman: Life and Works," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 631–636, 2017, ISSN: 24058963. DOI: 10.1016/j.ifacol.2017.08.070.

[61] M. S. Grewal and A. P. Andrews, *Kalman Filtering: Theory and Practice Using MATLAB*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2015, vol. 148, 619 pp., ISBN: 978-1-118-85121-0.

[62] T. C. for Land Use Interpretation. "Lockheed martin sunnyvale." (), [Online]. Available: https://clui.org/ludb/site/lockheed-martin-sunnyvale (visited on 12/15/2023).

[63] K. Kreutz-Delgado, "The Complex Gradient Operator and the CR-Calculus," Jun. 25, 2009. DOI: 10.48550/arXiv.0906.4835. arXiv: 0906.4835 [math], preprint.

[64] C. Tasse, "Applying Wirtinger derivatives to the radio interferometry calibration problem," *arXiv e-prints*, arXiv:1410.8706, arXiv:1410.8706, Oct. 2014. DOI: 10.48550/arXiv.1410.8706. arXiv: 1410.8706 [astro-ph.IM].

[65] R. Earl and J. Nicholson, *The Concise Oxford Dictionary of Mathematics* (Oxford Quick Reference), Sixth edition. New York: Oxford University Press, 2021, 492 pp., ISBN: 978-0-19-884535-5.

[66] S. Mayhew, *A Dictionary of Geography* (Oxford Quick Reference), Fifth edition. Oxford: Oxford University Press, 2015, 546 pp., ISBN: 978-0-19-968085-6.

[67] G. J. G. Upton and I. Cook, *A dictionary of statistics* (Oxford paperback reference), Third edition. Oxford: Oxford University Press, 2014, 488 pp., ISBN: 978-0-19-967918-8.

[68] P. L. Houtekamer and H. L. Mitchell, "Data assimilation using an ensemble kalman filter technique," *Monthly Weather Review*, vol. 126, p. 796, Jan. 1, 1998, ADS Bibcode: 1998MWRv..126..796H, ISSN: 0027-0644. DOI: 10.1175/1520-0493(1998)126<0796:DAUAEK>2.0.CO;2. [Online]. Available: https://ui.adsabs.harvard.edu/abs/1998MWRv..126..796H.