Analytical study of the Least Squares Quasi-Newton method for interaction problems

Thesis · January 2009

1 author:

Rob Haelterman
Royal Military Academy
60 PUBLICATIONS **281** CITATIONS

Universiteit Gent
Faculteit Ingenieurswetenschappen
Vakgroep Mechanica van Stroming, Warmte en Verbranding

Analytische studie van de Least Squares Quasi-Newton methode voor interactieproblemen

Analytical study of the Least Squares Quasi-Newton method for interaction problems

Robby Haelterman

Proefschrift tot het bekomen van de graad van
Doctor in de Ingenieurswetenschappen
Academiejaar 2008-2009

Universiteit Gent
Faculteit Ingenieurswetenschappen
Vakgroep Mechanica van Stroming, Warmte en Verbranding

Promotoren: Prof. Dr. Ir. Jan Vierendeels
Prof. Dr. Dirk Van Heule

Universiteit Gent
Faculteit Ingenieurswetenschappen

Vakgroep Mechanica van Stroming, Warmte en Verbranding
St.-Pietersnieuwstraat 41, B-9000 Gent, België

Tel.: +32-9-264.32.97
Fax.: +32-9-264.35.86

Proefschrift tot het behalen van de graad van
Doctor in de Ingenieurswetenschappen
Academiejaar 2008-2009

Dankwoord

Elke thesis is grotendeels een individuele calvarietocht, al duurt de ene al wat langer dan de andere.

Hoewel de eerste en tweede statie al bijna geheel aan ons geheugen onttrokken zijn, kan dit niet gezegd worden van de derde, en later de zevende en negende statie, die permanente herinneringen zijn gaan vormen.

In dit klein stukje van de schriftelijke neerslag van deze lijdensweg staan we even stil bij de personen die daar waren ten tijde van de vijfde statie.

Zij hebben, al dan niet achter de schermen, hun eigen onmisba(a)re steen(tje) bijgedragen.

In de eerste plaats wil ik mijn promotoren Jan Vierendeels en Dirk Van Heule danken. Jan voor het leveren van een extreem boeiend onderwerp, met mogelijkheden waarover het laatste woord nog lang niet gezegd is en Dirk voor het leveren van zijn eigen nuchtere, doch boeiende kijk op deze materie.

Verdere dank gaat uit naar professor Van hamme voor de interesse in het onderwerp, en zijn uitermate constructieve commentaar en suggesties.

Een doctoraat heeft natuurlijk ook een financieel kostenplaatje, waar Kolonel Isabelle De Leeneer in het verhaal komt. Niet dat zij alles uit eigen zak betaalde, maar ze zorgde wel voor de fondsen die mij door de KMS ter beschikking werden gesteld, en natuurlijk voor het spreekwoordelijke schouderklopje op gepaste tijden.

Constructieve criticus Joris Degroote had een onmisbare invloed door zijn visies, constante verbetering en fine-tuning.

Verder zijn er natuurlijk de collegae Steve en Peter voor “tips and tricks” op het gebied van Latex en Matlab, Helena voor het nalezen van dit werk en Patrick voor de technische ondersteuning in soms moeilijke omstandigheden.

Al mijn vrienden wordt nederig vergiffenis gevraagd voor de complete verwaarlozing die hun ten deel viel de laatste jaren. (Ik kan het allemaal uitleggen...)

Mijn moeder verdient dan weer een dikke knuffel voor morele en logistieke steun; zij is één van de belangrijkste slachtoffers van wat in de hierna volgende pagina’s is neergepend.

Een dankwoord is natuurlijk niet compleet zonder een “Last but not Least”, die in dat geval uitgaat naar mijn levensgezellin Claire, die altijd in de eerste linie stond als mijn humeur het weer eens begaf. Ik hoop dat ze snel gaat kunnen wennen aan de tijd die ik nu aan en met haar ga kunnen besteden.

Gent, maart 2009
Robby Haelterman

Table of Contents

Dankwoord **i**

List of symbols **x**

English summary **xv**

Nederlandse samenvatting **xix**

1 Introduction and problem-statement **1**

1.1 Problem-statement 2

1.1.1 Non-linear systems of equations 2

1.1.2 Linear systems of equations 4

1.1.2.1 Example: monolithic approach 5

1.1.2.2 Example: partitioned approach 6

1.1.3 Series of related coupled systems 7

1.2 Solution methods 8

1.2.1 Quasi-Newton methods 8

1.2.2 Iterative Substructuring Method 9

1.2.3 Fixed-point iteration with stationary or dynamic relaxation and vector extrapolation 11

1.3 Outline of the study 12

2 Introductory definitions and theorems **15**

2.1 Definitions 15

2.2 Conventions 20

2.3 General theorems 21

3 Solution methods for a single system of linear equations **27**

3.1 Legacy solvers: Richardson, Jacobi, Gauss-Seidel and SOR 27

3.1.1 Richardson iteration 28

3.1.2 Jacobi iteration 28

3.1.3 Gauss-Seidel iteration 28

3.1.4 Successive Over-Relaxation (SOR) 29

3.2 Krylov subspace methods 29

3.2.1 The Full Orthogonalization Method 30

3.2.2 The Generalized Minimal Residual Method (GMRes) . . . 30

3.2.2.1 Elementary implementation of GMRes 31
3.2.2.2 Adaptation of GMRes to our framework 32
3.2.3 Other Krylov subspace methods 34
3.2.3.1 The Lanczos Method, Conjugate Gradient Method (CG), CGNR, LSQR, CGLS, CGNE, OrthoRes and GENCG 35
3.2.3.2 The Generalized Conjugate Residual Method (GCR), Axelsson’s method, OrthoDir, OrthoMin, GENCR and MinRes 35
3.2.3.3 Bi-Conjugate Gradients (Bi-CG), Conjugate Gradient Squared (CGS), Bi-CGStab, QMR and TFQMR 36
3.2.4 Orthogonalization procedures 36

4 Solution methods for a single system of non-linear equations 37
4.1 Newton’s method 37
4.2 Quasi-Newton methods 38
4.3 Least change secant updates 41
4.4 Rank-one update quasi-Newton methods 41
4.4.1 Broyden’s first or “good” method 42
4.4.2 Broyden’s second or “bad” method 44
4.4.3 Column-Updating Method 46
4.4.4 Inverse Column-Updating Method 46
4.4.5 Symmetric Rank-One update (SR1) 47
4.4.6 Pearson’s Method 48
4.4.7 McCormick’s method 48
4.4.8 The Eirola-Nevanlinna method 48
4.5 Rank-two update quasi-Newton methods 50
4.5.1 Powell symmetric Broyden (PSB) method 50
4.5.2 Davidon-Fletcher-Powell (DFP) method 51
4.5.3 Broyden-Fletcher-Goldfarb-Shanno (BFGS) method . . . 51
4.5.4 Greenstadt’s method 51
4.6 Quasi-Newton methods preserving the structure of a matrix 51

5 Solution methods for two systems of non-linear equations 53
5.1 Interface quasi-Newton methods (IQN and IQN-I) 54
5.2 Interface quasi-Newton method with Composed Jacobian (IQN-C) 55
5.3 Interface block quasi-Newton method (IBQN) 55
5.4 Construction of the approximate Jacobians for two systems of non-linear equations 58
5.4.1 Broyden’s good and bad method for the Interface quasi-Newton approach 58
5.4.2 Column-Updating and Inverse Column-Updating method for the Interface quasi-Newton approach 59

6 Least Squares Jacobian 61
6.1 Construction of the Least Squares Jacobian 61
6.2 Orthogonalization and re-arrangement of input–ouptut modes . . . 64
6.3 Applying the Least Squares approximate Jacobian to quasi-Newton type methods 66
6.3.1 IQN-LS 66
6.3.2 IQN-ILS 68
6.3.3 IQN-CLS 69
6.3.4 IBQN-LS 70

7 Properties of IQN-LS, IQN-ILS, IQN-CLS and IBQN-LS for non-linear mappings 71
7.1 Re-writing IQN-LS and IQN-ILS with a Rank-One Update formula 72
7.1.1 IQN-LS 72
7.1.2 IQN-ILS 77
7.1.3 IQN-CLS and IBQN-LS 78
7.1.4 Conclusions 80
7.2 The generalized secant property 82
7.2.1 IQN-LS 82
7.2.2 IQN-ILS 84
7.2.3 IQN-CLS and IBQN-LS 85
7.2.4 Comment 86
7.3 The Least Change Secant Update property 86
7.3.1 IQN-LS 90
7.3.2 IQN-ILS 90
7.3.3 IQN-CLS and IBQN-LS 91
7.4 Equivalence between the different Least Squares methods 91
7.4.1 Equivalence between IQN-LS and IQN-CLS 91
7.4.2 Equivalence between IQN-CLS and IBQN-LS when F is an affine mapping 95
7.5 Avoiding singularity 98
7.5.1 Avoiding singularity for the original formulation (§6.3.1) . 98
7.5.2 Avoiding singularity for the rank-one update formulation . 99

8 Properties of IQN-LS, IQN-ILS, IQN-CLS and IBQN-LS for linear mappings 101
8.1 Non-singularity and convergence of the approximate Jacobian for IQN-LS and IQN-ILS 102
8.2 Comparison between IQN-LS, IQN-ILS and GMRes 105
8.2.1 Writing GMRes as a Quasi-Newton method 106
8.2.2 Krylov subspaces for IQN-LS 108
8.2.3 Krylov subspaces for IQN-ILS 114
8.2.4 Further discussion 119
8.3 The effect of step-length parameters 119

8.4 Modifying IQN-LS and IQN-ILS to make them algebraically identical to GMRes 121

8.4.1 Re-writing the quasi-Newton algorithm for matrix-vector products 121

8.4.2 Optimal step-length 123

8.4.3 Multiple parameters 124

9 Improving the Jacobian for discretized time-dependent and grid-based problems 127

9.1 Recovery of data from previous time-levels 128

9.1.1 Recovery methods based on input-output pairs 128

9.1.2 Recovery methods based on the rank-one update formulation 131

9.2 Creating a better initial approximate Jacobian from a coarser grid . 133

10 Numerical experiments with non-affine operators 135

10.1 One-dimensional flow in a flexible tube 135

10.1.1 Preliminary remark about notations 135

10.1.2 Analytical description of the problem 136

10.1.3 Discretizing the equations 138

10.1.4 Non-dimensionalizing the equations 140

10.1.5 Fourier error analysis 141

10.1.6 Results with the quasi-Newton solvers 144

10.1.6.1 No re-use of data from previous time-steps . . . 148

10.1.6.2 Re-use of data from previous time-steps 152

10.1.6.3 Initial Jacobian from a coarser grid 158

10.2 One-dimensional heat equation 161

10.2.1 Analytical description of the problem 161

10.2.2 Results with the quasi-Newton solvers 162

10.2.2.1 No re-use of data from previous time-steps . . . 163

10.2.2.2 Re-use of data from previous time-steps 165

10.3 Conclusion 170

11 Numerical experiments with linear operators 171

11.1 Test matrix from the MATRIX MARKET REPOSITORY 172

11.2 One-dimensional advection-diffusion equation 172

11.3 Two-dimensional advection-diffusion equation 174

11.4 Conclusion 175

12 Conclusions 183

A Matlab Source code 185

A.1 Main loop 185

A.2 first_init_temp.m 188

A.3 second_init_temp.m 189

A.4 third_init_temp.m 189

A.5 data_reset.m 191
A.6 create_extrapolated_Jacobian.m 191
A.7 IQN solver: QN_solver_IQN_.m 193
A.8 IQN-C solver: QN_solver_IQNC.m 195
A.9 IBQN solver: QN_solver_IBQN.m 198
A.10 IQN-I solver: QN_solver_IQNI.m 202

Bibliography **205**

List of Symbols

b	General system vector (“right hand side”)
b_F, b_H, etc.	System vector for F, H, etc. when affine ($F(g) = A_F g - b_F$, $H(p) = A_H p - b_H$)
c	(For FSI problem) wave speed
c_{mk}	(For FSI problem) Moens-Korteweg wave speed
c_s, d_s	Defining vector for quasi-Newton method in rank-one update form
e_s	Error for p_s, i.e. $p_s - p^*$
g	One of the variables in the interaction problem
g	(For heat problem) set of coefficients (ρ, C and k)
g^*	Solution of the interaction problem
g^*	(For FSI problem) solution for the cross-section in the discretized, non-dimensional equations
g_s	Value for g at the s-th iteration
$\mathfrak{g}$	(For FSI problem) continuous cross-section variable
g_t	(For FSI problem) discretized non-dimensional cross-section variable (vector) at time-step t
$g_{s,t}$	(For FSI problem) discretized non-dimensional cross-section variable (vector) at iteration s and time-step t
$\hat{g}_s$	(For FSI problem) error for the cross-section in Fourier analysis (vector)
$[\hat{g}_s]_i$	(For FSI problem) i-th component of $\hat{g}_s$
$[\tilde{g}_s]^l$	(For FSI problem) amplitude of l-th Fourier mode for the cross-section at iteration s
h	(For FSI problem) thickness of the tube wall
i	(For FSI problem) nodal index
$\imath_j$, ℓ_j	j-th vector of the canonical basis
$\jmath$	$= \sqrt{-1}$
k	(For heat problem) thermal conductivity
m	Length of the vector g
n	Length of the vector p
$\tilde{\mathfrak{n}}$	$= \max\{0, s - n\}$
p	One of the variables in the interaction problem
p^*	Solution of the interaction problem
p^*	(For FSI problem) solution for the pressure in the discretized, non-dimensional equations
p_s	Value for p at the s-th iteration
$[p]_i$	The i-th element of vector p
$\mathfrak{p}$	(For FSI problem) continuous (kinematic) pressure variable
$\tilde{\mathfrak{p}}$	(For FSI problem) continuous pressure variable
$\mathfrak{p}_o$	(For FSI problem) reference (kinematic) pressure
$\tilde{\mathfrak{p}}_o$	(For FSI problem) reference pressure

p_t	(For FSI problem) discretized non-dimensional pressure variable (vector) at time-step t
$p_{s,t}$	(For FSI problem) discretized non-dimensional pressure variable (vector) at iteration s and time-step t
$\hat{p}_s$	(For FSI problem) error for the pressure in Fourier analysis (vector)
$[\hat{p}_s]_i$	(For FSI problem) i-th component of $\hat{p}_s$
$[\tilde{p}_s]^l$	(For FSI problem) amplitude of l-th Fourier mode for the pressure at iteration s
q_s	(In algorithm) $A_K\Delta_s$
$\mathfrak{r}$	(For FSI problem) inner radius of tube
r_s	Residual of p_s, i.e. $K(p_s)$
t	(For FSI and heat problem) time variable
$\mathfrak{u}$	(For FSI problem) continuous velocity variable
$\mathfrak{u}_o$	(For FSI problem) reference velocity
u_o	(For FSI problem) reference velocity / CFL number
u_t	(For FSI problem) discretized non-dimensional velocity variable (vector) at time-step t
$u_{s,t}$	(For FSI problem) discretized non-dimensional velocity variable (vector) at iteration s and time-step t
$\hat{u}_s$	(For FSI problem) error for the velocity in Fourier analysis (vector)
$[\hat{u}_s]_i$	(For FSI problem) i-th component of $\hat{u}_s$
$[\tilde{u}_s]^l$	(For FSI problem) amplitude of l-th Fourier mode for the velocity at iteration s
u^*	(For FSI problem) solution for the velocity in the discretized, non-dimensional equations
v, w, x, y, z	General vectors
x	(For FSI and heat problem) space variable
α	(For FSI problem) pressure stabilization coefficient
β	(For FSI problem) coefficient
δg_s, δH_s, etc.	$= g_{s+1} - g_s$, $= H(p_{s+1}) - H(p_s)$ etc.
θ_s, $\theta_{s,k}$	Iteration parameter
ϑ_l	(For FSI problem) Fourier mode
κ	(For FSI problem) dimensionless structural stiffness
μ_l	(For FSI problem) amplification factor of l-th Fourier mode
ν	(For heat problem) coefficient
ϖ_l	(For FSI problem) spatial pulsation of l-th Fourier mode
ϕ	(For FSI problem) angle in the cross-sectional plane
ρ	(For FSI and heat problem) density
σ_o	(For FSI problem) reference stress
$\sigma_{\phi\phi}$	(For FSI problem) stress in the circumferential direction
τ	(For FSI problem) dimensionless time-step
ω_s, $\omega_{i,s}$	Iteration parameters

A	General system matrix
A_F, A_H, etc.	System matrix for F, H, etc. when affine ($F(g) = A_F g - b_F$, $H(p) = A_H p - b_H$)
$\mathbb{A}$	Set of "interpolating" matrices
$\mathring{\mathbb{A}}$	"Interpolating" matrix
C	(For heat problem) heat capacity
D_F, D_H, etc.	Domain of F, H, etc.
E	(For FSI problem) Young's modulus
F	One of the functions in the interaction problem
F_t	One of the functions in the interaction problem; time dependent
$F'(g_s)$	Jacobian of F evaluated at g_s
$\hat{F}'_s$	Approximate Jacobian of F at iteration s
FSI	Fluid-Structure Interaction
G_t	(For FSI problem) discretized cross-section variable (vector) at time-step t
$[G_t]_i$	(For FSI problem) i-th component of G_t
H	Function: $H = F \circ S$, or alternatively $H = S \circ F$
$H'(p_s)$	Jacobian of H evaluated at p_s
$\hat{H}'_s$	Approximate Jacobian of H at iteration s
I	Identity matrix
$\mathcal{I}$	Identity function
$\mathfrak{I}_c^f$	Prolongation matrix from coarse to fine grid
K	Function: $K = H - \mathcal{I}$
$K'(p_s)$	Jacobian of K evaluated at p_s
$\hat{K}'_s$	Approximate Jacobian of K at iteration s
$\mathcal{K}_s\{M; v\}$	Krylov subspace of dimension s generated by M and v
L	(For FSI and heat problem) length of the domain
$\bar{L}_s$, $\bar{L}_s^{pH}$ etc.	s-th column of $\mathcal{L}_s$, $\mathcal{L}_s^{pH}$, etc.
$\mathcal{L}_s$, $\mathcal{L}_s^{pH}$, etc.	Matrix of rank s containing an orthogonal basis for $\mathcal{R}(V_s)$, $\mathcal{R}(V_s^{pH})$, etc.
$\mathbf{L}_{s,t}^{pH}$, etc.	Matrix of rank s containing an orthogonal basis for $\mathcal{R}(\mathbf{V}_{s,t}^{pH})$, etc.
M	Function; inverse of K
$\mathcal{N}(\cdot)$	Null space (or Kernel)
$\mathbb{N}$	Set of natural numbers
O	Zero matrix
$\mathcal{O}_s^r$	Residual operator
P_t	(For FSI problem) discretized pressure variable (vector) at time-step t
$[P_t]_i$	(For FSI problem) i-th component of P_t
$[P]_{in}$, $[P]_{out}$	(For FSI problem) discretized pressure at inlet, resp. outlet
$\mathcal{P}_k$	Set of real polynomials of degree k with zero constant term
$[Q]_{ij}$	The element on the i-th row and j-th column of matrix Q

$\mathcal{R}(\cdot)$	Range
$\mathbb{R}$	Set of real numbers
S	One of the functions in the interaction problem
$S'(p_s)$	Jacobian of S evaluated at p_s
S_t	One of the functions in the interaction problem; time dependent
$\hat{S}'_s$	Approximate Jacobian of S at iteration s
T	(For heat problem) temperature
Tr($\cdot$)	Trace
U_t	(For FSI problem) discretized velocity variable (vector) at time-step t
$[U_t]_i$	(For FSI problem) i-th component of U_t
$[U]_{in}$, $[U]_{out}$	(For FSI problem) discretized velocity at inlet, resp. outlet
U_o	(For FSI problem) initial flow velocity
V_s, V_s^{pH}, etc.	Matrix with the “input modes” at iteration s
$\mathbf{V}_{s,t}^{pH}$, etc.	Matrix with the “input modes” at iteration s and time step t; also including input modes from previous time-steps
$\mathcal{V}$	Vector space
W_s, W_s^{pH}, etc.	Matrix with the “output modes” at iteration s
$\mathbf{W}_{s,t}^{pH}$, etc.	Matrix with the “output modes” at iteration s and time step t; also including output modes from previous time-steps
$\mathcal{Y}$, $\mathcal{Y}_s$, $\mathcal{Z}$, $\mathcal{Z}_s$	Vector spaces
Δ_s	(In algorithm) $(\hat{K}'_s)^{-1}r_s$ or $\hat{M}'_s r_s$
Δt	(For FSI and heat problem) time-step
Δx	(For FSI and heat problem) mesh size
Δx_i^s, Δp_i^s, etc.	Input or output mode
Θ_s	$= [\theta_{s,o}\ \theta_{s,1}\ \ldots\ \theta_{s,s}]^T$; set of iteration parameters
Φ	General function
$\Vert \cdot \Vert$	Vector or matrix norm
$\Vert \cdot \Vert_2$	Euclidean norm
$\Vert \cdot \Vert_{Fr}$	Frobenius norm
$\oplus$	Direct sum of two vector subspaces
$(\cdot)^\perp$	Orthogonal complement of a vector subspace.
$\langle \cdot, \cdot \rangle$	Scalar product
$\vec{\subset}$	(Is) vector subspace (of)
$(\cdot)^T$	Transpose of matrix Q
$(\cdot)^+$	Moore-Penrose generalized inverse of matrix a matrix, i.e. $((\cdot)^T(\cdot))^{-1}(\cdot)^T$

English summary

Often in nature different systems interact, like fluids and structures, heat and electricity, populations of species, etc. It is our aim in this thesis to find, describe and analyze solution methods to solve the equations resulting from the mathematical models describing those interacting systems. Even if powerful solvers often already exist for problems in a single physical domain (e.g. structural or fluid problems), the development of similar tools for multi-physics problems is still ongoing. When the interaction (or coupling) between the two systems is strong, many methods still fail or are computationally very expensive.

Approaches for solving these multi-physics problems can be broadly put in two categories: monolithic or partitioned. While we are not claiming that the partitioned approach is panacea for all coupled problems, we will only focus our attention in this thesis on studying methods to solve (strongly) coupled problems with a partitioned approach in which each of the physical problems is solved with a specialized code that we consider to be a black box solver and of which the Jacobian is unknown. We also assume that calling these black boxes is the most expensive part of any algorithm, so that performance is judged by the number of times these are called. In 2005 Vierendeels presented a new coupling procedure for this partitioned approach in a fluid-structure interaction context, based on sensitivity analysis of the important displacement and pressure modes which are detected during the iteration process. This approach only uses input-output couples of the solvers (one for the fluid problem and one for the structural problem). In this thesis we will focus on establishing the properties of this method and show that it can be interpreted as a block quasi-Newton method with approximate Jacobians based on a least squares formulation. We also establish and investigate other algorithms that exploit the original idea but use a single approximate Jacobian.

The main focus in this thesis lies on establishing the algebraic properties of the methods under investigation and not so much on the best implementation from a numerical point of view.

The work is organized as follows.

After an introductory chapter (chapter 1), where we give a historical overview and introduce the established terminology, we give, in chapter 2, the most useful definitions. Lemmas and theorems that are used in later chapters are established and proven in chapter 2.

In chapter 3 an overview is given of existing linear solvers with the main focus on

Krylov methods in general and GMRes in particular.
In chapter 4 a similar overview is given of existing non-linear solvers with a focus on quasi-Newton methods in general and those using a rank-one update form in particular. Methods of particular interest are Broyden's "good" and " bad" method, the Column-Updating Method and Inverse Column-Updating Method.
In chapter 5 we extend the idea of the quasi-Newton method to interacting systems of non-linear equations and formulate four approaches based on the original quasi-Newton method. These are the Interface Quasi-Newton (IQN) method, Interface Quasi-Newton method with Inverse Jacobian (IQN-I), Interface Block Quasi-Newton method (IBQN) and Interface Quasi-Newton method with Composed Jacobian (IQN-C). The construction of the Jacobian can be done based on existing methods, or with the Least Squares approach which is investigated in detail and forms the main topic of this thesis.
In chapter 6 we formalize the method for constructing approximate Least Squares Jacobians proposed by Vierendeels and co-workers and apply it to the quasi- Newton methods established in the previous chapter. This method is based on available input-output pairs of a function, which will allow us to approximate the Jacobian of that function. Basic properties of this construction are established and proven, like the possibility to re-arrange the input data; this allows us to orthogonalize the input vectors and, hence, improve the conditioning of the matrices involved in the construction. The resulting methods with this particular construction of the approximate Jacobian will be called IQN-LS, IQN-ILS, IBQN-LS and IQN-CLS.
In chapters 7 and 8, which form the main body of this work, we establish and prove the properties of the quasi-Newton methods with Least Squares Jacobian(s) for non-linear and linear systems respectively. In chapter 7 we re-write the original formulation of the method of Vierendeels and co-workers, and its newly established variants, in a rank-one update formulation. We remark that the method bears some similarities to the Broyden methods. We further establish that these Least Squares method satisfy a generalized secant property and a Least Change Secant Update property; these properties formally bring the Least Squares method within the framework of other well-established quasi-Newton methods. Further in chapter 7, the equivalence between the four methods is investigated in detail; it is shown that IQN-LS and IQN-CLS are algebraically identical; IQN-LS and IBQN-LS are only identical if one of the black box solvers represents an affine mapping. Finally, attention is given to the possibility of singularities and methods to deal with them.
Chapter 8 includes a detailed comparison between GMRes and two of the new quasi-Newton methods when applied to linear systems. It is shown that for affine mappings all of the Least Squares quasi-Newton methods find the exact solution after at most $n + 1$ matrix-vector products, in exact arithmetic (n being the size of the solution-vector); singularities cannot occur before the solution has been reached. It is proven that IQN-LS, IQN-ILS and GMRes share the same Krylov search subspace, but not the subspace of constraints. It is also shown that under certain hypotheses the quasi-Newton Least Squares method can be transformed to a method that is mathematically identical to GMRes without the need for extra

matrix-vector products.

In chapter 9 we discuss various possibilities to enhance the Jacobian for problems originating from (time-dependent) ordinary and partial differential equations. For the Least Squares method in the original formulation this is based on adding input-output couples from previous time-steps. For quasi-Newton methods in a rank-one update form this is either based on the re-use of the final Jacobian of the previous time-step or on the construction of an initial Jacobian on a coarser grid.

In chapters 10 and 11 the algorithms discussed in this work are compared against each other on two non-linear and three linear problems respectively. The non-linear problems are one-dimensional flow in a flexible tube and the one-dimensional heat equation with temperature-dependent coefficients. These tests show the potential of the Least Squares methods and the proposed methods to re-use data from previous time-steps. The linear tests are taken from the literature and are meant to illustrate the possible adaptation of IQN-LS and IQN-ILS to a form that is equivalent to GMRes, as shown in chapter 8.

Finally, we finish by drawing the most important conclusions.

Nederlandse samenvatting –Summary in Dutch–

In de natuur komt het vaak voor dat systemen met elkaar interageren, zoals fluida en structuren, hitte en electriciteit, bevolkingsgroepen, etc. In deze thesis is het onze bedoeling om oplossingsmethoden te vinden en te beschrijven die toelaten de vergelijkingen op te lossen die voortkomen uit de wiskundige modellen die deze interagerende systemen beschrijven. Zelfs al bestaan er reeds krachtige oplossingstechnieken voor problemen in een enkel fysisch domein (bv. structuur- of fluidumproblemen), dan blijft de ontwikkeling van gelijkaardige middelen voor multi-fysica problemen nog steeds een domein van actief onderzoek. Wanneer de interactie (of koppeling) tussen de systemen sterk is, falen nog steeds veel methoden of vragen ze erg veel rekentijd.

Voor de aanpak van dergelijke multi-fysica problemen kan een gebruiker meestal kiezen uit monolitische of gepartitioneerde methoden. Ook al beweren we niet dat de gepartitioneerde aanpak een zaligmakende oplossing is, toch zullen we ons in deze thesis enkel toespitsen op dit soort methoden om (sterk) gekoppelde problemen op te lossen, waarbij we er vanuit gaan dat elk deelprobleem wordt opgelost met een beschikbare oplossingsmethode die beschouwd wordt als een zwarte doos en waarvan bijgevolg de Jacobiaan niet gekend is. We gaan er verder ook van uit dat het aanroepen van de oplossingsmethode voor een deelprobleem rekenkundig het duurste deel is van elk koppelingsalgoritme; bijgevolg meten we de performantie van een methode aan de hand van het aantal keer dat deze worden aangeroepen. In 2005 presenteerde Vierendeels een nieuw koppelingsalgoritme voor de gepartitioneerde oplossing in de context van vloeistof-structuur interactie, gebaseerd op de analyse van de gevoeligheden van de belangrijkste verplaatsings- en drukmodes, die tijdens de iteraties werden waargenomen. Deze aanpak gebruikt enkel invoer-uitvoer koppels van de oplossingsmethoden (die voor het fluidum en die voor de structuur) en bouwt voor elk een Jacobiaan. In deze thesis spitsen we onze aandacht toe op het formuleren van de eigenschappen van deze methode en tonen dat ze kan geïnterpreteerd worden als een blok quasi-Newton methode met een benaderde Jacobiaan gebaseerd op het principe van de kleinste kwadraten. We formuleren en onderzoeken ook andere algoritmes die het oorspronkelijke idee exploiteren, maar die één enkele Jacobiaan gebruiken. De aandacht in deze thesis gaat vooral uit naar de algebraïsche eigenschappen van de onderzochte methoden, en niet zozeer naar de optimale numerieke implementatie.

Het werk is als volgt georganiseerd.

Na een inleidend hoofdstuk (hoofdstuk 1), waar een historisch overzicht en de geijkte terminologie worden gegeven, vindt men de belangrijkste definities geformuleerd in hoofdstuk 2. Lemma's en theorema's die in latere hoofdstukken zullen worden gebruikt, worden vermeld en bewezen.

In hoofdstuk 3 vindt men een overzicht van bestaande oplossingstechnieken voor lineaire problemen, met particuliere aandacht voor Krylovmethoden, meer bepaald GMRes.

In hoofdstuk 4 wordt een gelijkaardig overzicht gegeven voor niet-lineaire problemen. Dit betreft quasi-Newton methoden in het algemeen en deze met een rang-één aanpassing in het bijzonder. Specifieke methoden die behandeld worden zijn Broyden's "goede" en "slechte" methode, de Kolom-Aanpassingsmethode (*Eng: Column-Updating Method*) en de Inverse Kolom-Aanpassingsmethode (*Eng: Inverse Column-Updating Method*).

In hoofdstuk 5 breiden we het idee van de quasi-Newton methode uit naar interagerende systemen van niet-lineaire vergelijkingen. We formuleren vier methoden gebaseerd op de oorspronkelijke quasi-Newton methode. Deze zijn de Raakvlak Quasi-Newton Methode (*Eng. afk. IQN*), Raakvlak Quasi-Newton Methode met Inverse Jacobiaan (*Eng. afk. IQN-I*), Raakvlak Blok Quasi-Newton Methode (*Eng. afk. IBQN*) en Raakvlak Quasi-Newton Methode met Samengestelde Jacobiaan (*Eng. afk. IQN-C*). De constructie van de Jacobiaan kan gebeuren aan de hand van bestaande methoden of met de Kleinste Kwadraten methode die in detail wordt bestudeerd en de hoofdmoot uitmaakt van deze thesis. In hoofdstuk 6 formaliseren we de methode voor de constructie van de Kleinste Kwadraten Jacobiaan, zoals voorgesteld door Vierendeels en medewerkers, en passen we ze toe op de quasi-Newton methoden uit het vorige hoofdstuk. Deze methode is gebaseerd op invoer-uitvoer koppels van een functie, die ons zullen toelaten een benaderde Jacobiaan van die functie op te bouwen. De voornaamste eigenschappen van deze methode worden gegeven en bewezen, zoals de mogelijkheid de invoergegevens te herschikken. Dit laat ons toe de invoervectors te orthogonalizeren en zo de conditionering van de betrokken matrices te verbeteren. De resulterende methoden met deze Jacobiaan krijgen de Engelse afkortingen IQN-LS, IQN-ILS, IBQN-LS en IQN-CLS mee.

In hoofdstukken 7 en 8, die de kern van dit werk vormen, staven we de belangrijkste eigenschappen van de quasi-Newton methoden met Kleinste Kwadraten Jacobi(a)n(en), respectievelijk voor niet-lineaire en lineaire systemen. In hoofdstuk 7 herschrijven we de oorspronkelijke methode van Vierendeels en medewerkers, en de nieuwe varianten, in een rang-één aanpassing formulering. We merken op dat de methoden gelijkenissen vertonen met de Broyden methoden. We staven verder dat de Kleinste Kwadraten methode een veralgemeende secant eigenschap en een Kleinste Wijziging Secant Aanpassing (*Eng. Least Change Secant Update*) eigenschap vertoont. Deze eigenschappen brengen de Kleinste Kwadraten methode formeel binnen het kader van een type quasi-Newton methode dat reeds in het verleden uitgebreid werd onderzocht. Verder in hoofdstuk 7 wordt de gelijkwaardigheid tussen de vier methoden onderzocht; er wordt getoond dat IQN-LS en IQN-CLS algebraïsch identiek zijn en dat IQN-LS en IBQN-LS identiek zijn op voorwaarde dat een van beide zwarte doos functies een affiene transformatie voorstelt. Finaal wordt er aandacht besteed aan de mogelijkheid van singulariteiten en oplossingen ervoor.

Hoofdstuk 8 bevat een gedetailleerde vergelijking tussen GMRes en twee van de nieuwe quasi-Newton methoden wanneer ze worden toegepast op lineaire systemen. Er wordt aangetoond dat voor affiene transformaties alle Kleinste Kwadraten quasi-Newton methoden de exacte oplossing vinden na maximaal $n + 1$ matrix-vector producten, op numerieke fouten na (n zijnde de dimensie van de oplossingsvector); singulariteiten kunnen niet voorkomen alvorens de oplossing is bereikt. Er wordt aangetoond dat IQN-LS, IQN-ILS en GMRes een Krylov zoek-deelruimte delen, maar niet de deelruimte van beperkingen. Er wordt ook aangetoond dat onder bepaalde hypothesen de quasi-Newton Kleinse Kwadraten methode omgevormd kan worden tot een methode die wiskundig identiek is aan GMRes zonder de nood voor extra matrix-vector producten.

In hoofdstuk 9 bespreken we verschillende mogelijkheden om de initiële Jacobiaan te verbeteren wanneer het op te lossen probleem afkomstig is van een (mogelijks tijdsafhankelijke) gewone of partiële differentiaalvergelijking. Voor de Kleinste Kwadraten methode in de originele formulering is dit gebaseerd op het toevoegen van invoer-uitvoer koppels van vorige tijd-stappen. Voor quasi-Newton methodes in rang-één aanpassing formulering wordt dit gedaan door ofwel de laatste Jacobiaan van de vorige tijd-stap te nemen of door een initiële Jacobiaan op te bouwen op een groffer rooster.

In hoofdstukken 10 en 11 worden de algoritmes die we eerder bestudeerden met elkaar vergeleken op respectievelijk twee niet-lineaire en drie lineaire problemen. De niet-lineaire problemen zijn de één-dimensionale stroming in een flexibele buis en de één-dimensionale hittevergelijking met temperatuursafhankelijke coëfficiënten. Deze tests tonen het potentieel aan van de Kleinste Kwadraten methoden en de voorgestelde methoden om gegevens van vorige tijd-stappen te herbruiken. De lineaire tests werden genomen uit de literatuur en zijn bedoeld om de aanpassing van IQN-LS en IQN-ILS naar een vorm equivalent met GMRes te illustreren, zoals opgesteld in hoofdstuk 8.

Uiteindelijk eindigen we met de belangrijkste conclusies.

1
Introduction and problem-statement

Often in nature different systems interact, like fluids and structures, heat and electricity, populations of species, etc. It is our aim to find, describe and analyze solution methods to solve the equations resulting from the mathematical models describing those interacting systems.

From the growing number of conferences, publications and software releases it is clear that in silico simulations of these kinds of coupled systems are becoming ever more important in the engineering community. Examples can be found in

- aeronautics; e.g. [66, 68–71, 71, 123, 177, 211];
- bio-medical science; e.g. [13, 17, 35, 59, 61, 89, 110, 185, 224, 227];
- civil engineering; e.g. [21, 64, 79, 115, 120, 176, 187, 188, 239, 240];

to name but a few.

Often powerful solvers already exist for problems in a single physical domain (e.g. structural or fluid problems). Even so, development of similar tools for multi-physics problems is still ongoing and the paths followed to obtain such a solver can be broadly put in one of the following categories:

- *Monolithic or simultaneous solution*: the whole problem is treated as a monolithic entity and solved simultaneously with a specialized *ad hoc* solver.

- *Partitioned solution*: the physical components are treated as isolated entities that are solved separately. Interaction is modeled as forcing terms and/or boundary conditions.

The relative merits of these methods are very problem dependent.
The advantage of the monolithic approach is the enhanced stability [156]. This comes at a cost, however, as specialized software has to be written for each type of interaction problem, which can result in very large systems. Furthermore, it can be inappropriate to use the same basic formulation for both types of problems. It also forces the user to treat non-linearities in the same way for all components. Still, the monolithic approach has shown to be a very popular method, e.g. [14, 17, 23, 111, 186, 213].

The partitioned approach allows for the use of available specialized solvers for each physical component (structure, fluid, ...), on the condition that the coupling effects can be treated efficiently. The latter is often feasible for problems where the systems only weakly interact. Strongly coupled problems, on the other hand, still pose a real challenge. Many articles can be found on partitioned methods in the literature, e.g. [21, 46, 73, 150, 151, 153, 174, 177, 225].

We will give a simple example of both approaches in §1.1.2 after which we will solely focus on the partitioned approach, as the main aim of this work is to study the properties of a partitioned method first proposed by Vierendeels and coworkers in 2007 [227]. We are not concerned with the solution process of the constituent physical problems as these are assumed to be handled by specialized solvers which we assume to be *black box* operations of which no specific details can be modified or even assessed.

1.1 Problem-statement

1.1.1 Non-linear systems of equations

In general we are interested in non-linear coupled problems that can be mathematically stated in the following form:

$$F(g) = p \qquad (1.1a)$$

$$S(p) = g, \qquad (1.1b)$$

where

$$F : D_F \subset \mathbb{R}^{m\times 1} \to \mathbb{R}^{n\times 1} : g \mapsto F(g)$$

and

$$S : D_S \subset \mathbb{R}^{n\times 1} \to \mathbb{R}^{m\times 1} : p \mapsto S(p).$$

Each equation describes (the discretized equations of) a physical problem that is spatially decomposed. (E.g. $F(g) = p$ could give the pressure p on the wall of a flexible tube for a given geometry g, while $S(p) = g$ could give the deformed geometry of that same wall under influence of the pressure exerted on it by the fluid.)

We limit p and g to values on the interface between the two physical problems. In this way the physically decoupled nature of the problem is exploited. This approach can be regarded as a special case of *heterogeneous domain decomposition methods* [54] and limits the number of variables the coupling technique will be dealing with, even though the black box solvers that give $F(g)$ and $S(p)$ might use a substantially higher number of internal variables; for instance in the case of a fluid-structure interaction problem where the pressure is passed from the fluid to the structure, the fluid velocity is an internal variable for the flow solver as are all nodal values of the pressure that are not on the interface.

Alternatively, (1.1) can be written as the fixed point problem

$$F(S(p)) = H(p) \quad = \quad p \tag{1.2}$$

or the root-finding problem

$$H(p) - p = K(p) \quad = \quad 0. \tag{1.3}$$

Using (1.2) or (1.3) means that we have actually lumped both systems F and S together into one system (either H or K), which in general has a lower number of variables than the sum of the number of variables of both constituent systems F and S.

We assume that F, S, H and K satisfy the following hypotheses, which are typical when working with Newton and quasi-Newton type methods [166]:

Hypothesis 1.1. *F, resp. S, H, K, is continuously differentiable in an open set D_F, resp. D_S, D_H, D_K.*

Hypothesis 1.2. *$K(p) = 0$ has one solution p^* in D_K.*

Hypothesis 1.3. *$(K'(p))^{-1}$ exists and is continuous in an open set containing p^*.*

We assume the operations $F(g)$ and $S(p)$ (and hence $H(p)$ and $K(p)$) are black box systems, representing the propriety solvers, with a high computational cost and of which nothing is known about the Jacobian; neither do we make assumptions about this Jacobian like sparseness, symmetry, etc. For that reason we count the performance of a method by the number of times $F(g)$ or $S(p)$ are executed, a process we will call a *function call*. Requirements like actual cpu-time or storage are not taken into consideration.

Remark 1.1. *While we write the equations in (1.1) in explicit form, this is only for convenience; any form is usable as long as for a given value of g (resp. p) a corresponding value of p (resp. g) can be computed that satisfies equation (1.1a) (resp. (1.1b)).*

Remark 1.2. *Some of the solution methods that we will present in this thesis will be specifically aimed at solving (1.3) irrespective of the fact that it stems from a coupled problem or not.*

Remark 1.3. *We could have used (and sometimes will use)*

$$S(F(g)) \quad = \quad g \tag{1.4}$$

instead of (1.2). The choice between both can depend on

- *practical implementation issues due to the solvers used;*
- *the relative sizes of n and m. If $n < m$, resp. $n > m$, the use of (1.2), resp. (1.4), will result in a problem that is defined on a space with the lowest dimension.*

1.1.2 Linear systems of equations

Besides being easier to analyze, linear problems are interesting in their own right. They also give a valuable insight into the behavior and properties of weakly non-linear systems. Moreover, when we are sufficiently close to a zero of a non-linear function, its behavior is mainly linear.
When F, S, H and/or K represent affine mappings then we will write

$$F(g) \quad = \quad A_F g - b_F \tag{1.5a}$$
$$S(p) \quad = \quad A_S p - b_S \tag{1.5b}$$
$$H(p) \quad = \quad A_H p - b_H \tag{1.5c}$$
$$K(p) \quad = \quad A_K p - b_K, \tag{1.5d}$$

where $A_F \in \mathbb{R}^{n\times m}$, $A_S \in \mathbb{R}^{m\times n}$, $A_H, A_K \in \mathbb{R}^{n\times n}$, $b_F, b_H, b_K \in \mathbb{R}^{n\times 1}$, $b_S \in \mathbb{R}^{m\times 1}$. We assume that A_F, A_S, A_H and A_K are non-singular.
Obviously the following relations hold:

$$\begin{aligned} A_H &= A_F A_S \\ b_H &= A_F b_S + b_F \\ A_K &= A_H - I \\ b_K &= b_H. \end{aligned}$$

Just as in §1.1.1 we assume that we have no prior knowledge of A_F $(A_S, \ldots)$ or b_F $(b_S, \ldots\,)$ whatsoever, but are able to form either $A_F x$ $(A_S y, \ldots)$ or $A_F x - b_F$ $(A_S y - b_S, \ldots)$ for all $x \in \mathbb{R}^{m\times 1}$ $(y \in \mathbb{R}^{n\times 1})$[1]. We also assume that it is impossible to form either $A_F^T x$ $(A_S^T y, \ldots)$ or $A_F^T x - b_F$ $(A_S^T y - b_S, \ldots)$.
We count the performance of a method by the number of times a matrix-vector product is computed, which we assume to be the dominant cost of any solution method.

The simplicity of the linear case also allows us to give an elementary example of the difference between the monolithic and partitioned approaches.

1.1.2.1 Example: monolithic approach

If the coupled problem is derived from the discretization of a set of partial differential equations, we can choose between two different techniques in the monolithic approach. The first is to discretize the ensemble of all partial differential equations as a whole. This will, in general, result in a system of equations with a number of variables that is vastly superior to the sum of the number of variables contained in p and g (equation (1.1)); i.e. all the internal variables of both physical domains have to be treated by the solver. The advantage is that no further constraints are imposed on the use of the best discretization method or the solution method for the resulting system.

Alternatively, the domain decomposition technique could be used, resulting in a set of discretized equations using only the variables on the interface between the physical domains, as done in (1.1). We illustrate this for affine mappings, for which (1.1) can then be written as

[1] Most often it will be $A_F x - b_F$, etc., as this is the affine equivalent of $F(x)$, etc; we tacitly assume that this is the case, unless otherwise stated.

$$\begin{aligned} A_F g - b_F &= p & (1.7a)\\ A_S p - b_S &= g. & (1.7b)\end{aligned}$$

If we write (1.7) as an aggregated system

$$Az = b, \qquad (1.8)$$

with

$$A = \begin{bmatrix} I & -A_F \\ -A_S & I \end{bmatrix}, z = \begin{bmatrix} p \\ g \end{bmatrix}, b = \begin{bmatrix} -b_F \\ -b_S \end{bmatrix},$$

then (1.8) can be solved with any of the well-known linear solvers, like Richardson, GMRes,... As we do not explicitly exploit the components A_F and A_S of A or limit ourselves to a solution process based solely on function calls $F(g)$ and $S(p)$, this is an example of a monolithic solution method.

We will discuss neither of these approaches any further as the focus of our study lies in the study of partitioned methods.

1.1.2.2 Example: partitioned approach

Alternatively, we could keep the segregated nature of (1.1) and use a separate solver for each constituent equation. This forms the basis of the partitioned approach.

While we will discuss the partitioned approach in more details in the following chapters, we give an illustrative example applied to linear systems.

One way to solve (1.8) is

$$\begin{bmatrix} I & O \\ O & I \end{bmatrix} \begin{bmatrix} p_{s+1} \\ g_{s+1} \end{bmatrix} = \begin{bmatrix} O & A_F \\ A_S & O \end{bmatrix} \begin{bmatrix} p_s \\ g_s \end{bmatrix} - \begin{bmatrix} b_F \\ b_S \end{bmatrix}. \qquad (1.9)$$

(I represents the identity matrix and O the zero matrix.) We see that in (1.9) the following equations ensue: $p_{s+1} = A_F g_s - b_F (= F(g_s))$ and $g_{s+1} = A_S p_s - b_S (= S(p_s))$. This shows the possible use of solvers that yield $F(g)$ and $S(p)$.

Note that the resulting method is equal to the block Jacobi approach applied to (1.8), which shows the possible similarities between both approaches. The block Gauss-Seidel method would give

$$\begin{bmatrix} I & O \\ -A_S & I \end{bmatrix} \begin{bmatrix} p \\ g \end{bmatrix}_{s+1} = \begin{bmatrix} O & A_F \\ O & O \end{bmatrix} \begin{bmatrix} p \\ g \end{bmatrix}_{s} - \begin{bmatrix} b_F \\ b_S \end{bmatrix}, \tag{1.10}$$

i.e. $S(p_s) = g_s$ and $F(g_s) = p_{s+1}$.

Equation (1.9), resp. (1.10), also gives a clear illustration of the conditional stability of this method, as the convergence of the iteration is governed by the spectral radius of $\begin{bmatrix} O & A_F \\ A_S & O \end{bmatrix}$, resp. $\begin{bmatrix} I & O \\ -A_S & I \end{bmatrix}^{-1} \begin{bmatrix} O & A_F \\ O & O \end{bmatrix}$, which should be inferior to unity and which will mainly depend on some measure of the "size" of A_F and A_S, i.e. the coupling matrices.

1.1.3 Series of related coupled systems

When (1.1) is derived from a physical problem, it often represents the equations obtained after discretizing the continuous equations in time and space, and thus only represents the evolution over one time-step. This is an example of how we could be presented with a series of related problems.

In this context we can write (1.1) as

$$F_{t+1}(g, p_t, g_t) = p \tag{1.11a}$$

$$S_{t+1}(p, p_t, g_t) = g, \tag{1.11b}$$

where the subscript $t + 1$ ($t = 0, 1, \dots$) denotes the time-level at which the problem is solved. The solution of (1.11) will give the values of p and g at that time-level (p_{t+1}, resp. g_{t+1}); the extra arguments p_t and g_t are added to show that the solution at the next time-level depends on the values at the previous time-level[2].
In what follows we will almost always simply write $F(g)$ and $S(p)$ and assume it is clear from the context that this either describes an isolated problem or a problem solved over one time-step. The only time we will use the subscript is in chapter 9 when we will use data from various time-steps.

In time-dependent problems, two coupling approaches can be distinguished: *weak coupling* and *strong coupling*. Weak coupling means that (1.11) is only solved approximately, while in the strong coupling method (1.11) is solved up to convergence (see §1.2.2 for further details).

[2] It is possible that it depends on more than one of the previous time-levels.

1.2 Solution methods

As the focus of this study lies on partitioned methods, and one specific partitioned method in particular, we will no longer treat monolithic methods from hereon and assume we are solving a single coupled problem in either the form of (1.1) or (1.3), unless otherwise noted.
In this section we give a brief outline of three possible solution methods.

1.2.1 Quasi-Newton methods

A classical solution method for the nonlinear problem $K(p) = 0$ is Newton's method. However, as we have assumed we don't have access to the Jacobians of F, S, H or K, we will resort to quasi-Newton methods, which we apply either to the single equation

$$K(p) = 0$$

or the system

$$\begin{aligned} F(g) &= p \\ S(p) &= g. \end{aligned}$$

Classical quasi-Newton methods [49] replace the Jacobian of the well-known Newton method for the non-linear equation $K(p) = 0$ by an approximation. The way this approximation is constructed differentiates the particular methods. (See §4.2 for more details.)

In this thesis we will also consider slightly different approaches that use approximate Jacobians of both S and F in

$$\begin{aligned} F(g) &= p \\ S(p) &= g \end{aligned}$$

based on ideas first formulated by Vierendeels and coworkers in [227]. Whatever the way we construct the Jacobian(s), we will distinguish four solution methods: *Interface*[3] *Quasi-Newton, Interface Quasi-Newton with Composed Jacobian, Interface Block Quasi-Newton* and *Interface Quasi-Newton with Inverse Jacobian*. We will go into more detail on these methods in chapter 4.

Quasi-Newton methods have been used intensively for solving linear and non-linear systems and for minimization problems [145]. Their main attraction is

[3] "Interface" refers to the fact that we only use values on the interface between the two coupled problems. This term will be dropped from the name of the method if the equations do not originate from an interaction problem on the interface.

that they avoid the cumbersome computation of derivatives for the Jacobians. Recently, interest in quasi-Newton methods has waned, as automatic differentiation has become available [58, 126] except for a recent algorithm by Eirola and Nevanlinna [63, 85] and the research performed by Deuflhard (e.g. [55, 57]) and Brown [22].

We are mainly interested in quasi-Newton methods because

- we do not have access to the Jacobian as we are working with black box systems, which also makes automatic differentiation impossible;

- the cost of a function evaluation is sufficiently high so that numerical differentiation becomes prohibitive. For this reason we will judge performance of the method by the number of function evaluations it needs for convergence.

We will also extensively study the proposed methods when applied to linear systems of equations. Studying quasi-Newton methods for linear problems is not only important because many problems are linear or nearly linear, but also because the properties of a method in the linear case often define the local convergence behavior of the method in the non-linear case. This can be understood by observing that close to a solution of $K(p) = 0$ where the Jacobian is non-singular, the linear approximation of $K(p)$ tends to be dominant. Hence, the generated iteration sequence tends to behave like in the linear case. This is the main reason why the local convergence of Newton's method is quadratic [166].

If the problem is time-dependent, solving a single time-step with the quasi-Newton method until convergence is reached, represents a strong coupling technique, as we assume the exact solution is obtained.

In the following paragraphs we will briefly sketch other well-known strongly coupled partitioned methods.

1.2.2 Iterative Substructuring Method

The fixed point method applied to (1.3) gives the *iterative substructuring method* (ISM) [3, 37, 135, 161, 183]. The method is given below in algorithm 1.2.1.

Algorithm 1.2.1 (Iterative substructuring method - ISM)**.**

```
1. Startup:
   a. Take an initial value $p_o$.
   b. Set $s = 0$.
2. Loop until sufficiently converged:
   a. Compute $g_s = S(p_s)$.
   b. Compute $p_{s+1} = F(g_s)$.
   d. Set $s = s + 1$.
```

If the problem is time-dependent, solving a single time-step with the ISM represents a strong coupling technique.

As the lines 2.a and 2.b of algorithm 1.2.1 can be re-written as $p_{s+1} = H(p_s)$ it is clear that the iterative substructuring method can be seen as a fixed point iteration applied to equation (1.2)[4]. ISM can also be interpreted as a preconditioner which compresses the coupled problem onto a smaller subspace with a better distribution of the eigenvalues [156].

The main drawback of this method is its conditional stability which has been widely studied, e.g. [34, 53, 127, 135, 156–158, 165, 236], and it is well-known that the approach fails for problems where the interaction between the two solvers is strong [156, 219].

The conditional stability is not difficult to see from the following example. We have $p_{s+1} = H(p_s)$, which, if all mappings are affine, translates to $p_{s+1} = A_H p_s - b_H$. As $A_H = A_F A_S$ it is thus clear that convergence will depend on the spectral radius of A_H and hence on A_F and A_S. (Also see the examples in §1.1.2 and §10.1.5.)

Remark 1.4. *Doing only a single iteration of the ISM for a time-dependent problem results in one of the best known weak coupling techniques, called the staggered solution method [67, 79, 158, 170, 175, 176, 252]. The method obviously only gives good results when the interaction between the two systems is not too strong, as the solution does not necessarily satisfy (1.11).*

[4] Other names can be given to this type of iteration: nonlinear Richardson iteration, Picard iteration or method of successive substitution [127].

1.2.3 Fixed-point iteration with stationary or dynamic relaxation and vector extrapolation

An improvement over ISM is the the *fixed-point iteration with dynamic relaxation*, which in its most general form is given in algorithm 1.2.2.

Algorithm 1.2.2 (Fixed-point iteration with dynamic relaxation)**.**

```
1. Startup:
   a. Take an initial value $p_o$.
   b. Set $s = 0$.
2. Loop until sufficiently converged:
   a. Compute $g_s = S(p_s)$.
   b. Compute $p_{s+1} = (1 - \omega_s)p_s + \omega_s F(g_s)$, with $\omega_s$ a suitably chosen relaxation factor.
   d. Set $s = s + 1$.
```

Another way to write the resulting iteration is $p_{s+1} = p_s + \omega_s K(p_s)$.
Fixed-point iteration with stationary relaxation is obtained by simply setting $\omega_s = \omega_o$ for $s = 1, 2, \ldots$.

This approach falls under the general label of *(vector) acceleration techniques*, about which a vast literature is available, e.g. [87, 119, 120, 129, 158, 235].
Perhaps the best known is Aitken's δ^2 method studied by Aitken [2] and Lubkin [137]. (See also [118, 129, 157, 158], and variants, e.g. [138].)
In this method ω_s is defined recursively by

$$\omega_{s+1} = -\omega_s \frac{\langle K(p_{s-1}), K(p_s) - K(p_{s-1})\rangle}{\langle K(p_s) - K(p_{s-1}), K(p_s) - K(p_{s-1})\rangle}. \tag{1.12}$$

Generalized versions of Aitken's method have been developed by Wynn [241–244], and Graves-Morris [95]. All of these are part of what is called the *Lozenge algorithm family* [15, 16, 109].
Other acceleration techniques have been proposed: the Minimal Polynomial Extrapolation (MPE) of Cabay and Jackson [30]; the Modified Minimal Polynomial Extrapolation (MMPE) of Sidi et al. [203], Brezinski [20] and Pugachev [182]; the Reduced Rank Extrapolation (RRE) of Eddy and Mešina [62, 154].
Polynomial extrapolation methods are closely related to Krylov methods [191, 192, 194, 203, 238].
A survey of these method can be found in [121, 207].
In this work we will not go into details about these methods.

1.3 Outline of the study

While we are not claiming that the partitioned approach is panacea for all coupled problems, we will only focus our attention in this thesis on studying methods to solve (strongly) coupled problems with a partitioned approach in which each of the physical problems is solved with a specialized code that we consider to be a black box and of which the Jacobian is unknown. We also assume that calling these black boxes is the most expensive part of any algorithm, so that performance is judged by the number of times these are called.

In 2007 Vierendeels [227] presented a new coupling procedure for this partitioned approach in a fluid-structure interaction context, based on sensitivity analysis of the important displacement and pressure modes which are detected during the iteration process. This approach only uses input-output couples of the solvers (one for the fluid problem and one for the structural problem).
In this thesis we will focus on establishing the properties of this method and show that it can be interpreted as a block quasi-Newton method with approximate Jacobians based on a least squares formulation. We also establish and investigate other algorithms that exploit the original idea but use a single approximate Jacobian.
These methods fall within a well-established framework of quasi-Newton methods that can be written in a rank-one update form and belong to the family of Least Change Secant Update quasi-Newton methods.
We establish the relationship between the variants and other existing quasi-Newton methods.
When applied to affine operators, the method shares a Krylov search subspace with GMRes and, with a mild assumption on the nature of the black box solvers, can be modified to be analytically identical to GMRes.
We stress that throughout this work we will mainly be concerned about the behavior of the algorithms in exact arithmetic, more than about implementation issues and/or numerical stability.

This work is organized as follows.
In chapter 2 we start with formulating the most useful definitions. Lemmas and theorems that are used in later chapters are given.
In chapter 3 an overview is given of existing linear solvers with the main focus on Krylov methods in general and GMRes in particular.
In chapter 4 a similar overview is given of existing non-linear solvers with a focus on quasi-Newton methods in general and those using a rank-one update form in particular.
In chapter 5 we extend the idea of the quasi-Newton method to interacting systems of non-linear equations.

In chapter 6 we formalize the method for constructing approximate Least Squares Jacobians proposed by Vierendeels and coworkers [227] and apply it to the quasi-Newton methods established in the previous chapter.

In chapters 7 and 8, which form the main body of this work, we establish and prove the properties of the quasi-Newton methods with Least Squares Jacobian(s) for non-linear and linear systems respectively. Chapter 8 also includes a detailed comparison between GMRes and two of the new quasi-Newton methods when applied to linear systems.

In chapter 9 we discuss various possibilities to enhance the Jacobian for problems originating from (time-dependent) ordinary and partial differential equations.

In chapters 10 and 11 the algorithms discussed in this work are compared against each other on non-linear and linear problems respectively.

Finally, we finish by drawing the most important conclusions.

2 Introductory definitions and theorems

In this chapter we give the definitions that are most commonly encountered in this work. We also establish and prove a number of theorems and lemmas upon which later properties are based. Theorems and proofs that were found in the literature are stated as such.
Most of the properties apply to the single system of (non-)linear equations as given in (1.3).

2.1 Definitions

We will use p^* and g^* for the exact solution of (1.1) and/or (1.3); when using an iterative process to solve the equation we will use $e_s = p_s - p^*$ for the error at the s-th iterate. (We do not use an error measure for the iterates of g). The residual of this iterate for (1.3) will be defined by $r_s = K(p_s)$, which equals $(A_H - I)e_s = A_K e_s$ if H is an affine operator, i.e. $H(p) = A_H p - b_H$.
We will also use the notation $\delta p_s = p_{s+1} - p_s (= \delta e_s = e_{s+1} - e_s)$ and $\delta g_s = g_{s+1} - g_s$.

Definition 2.1. *A "**natural**" (or "**induced**") matrix norm is a matrix norm induced* *by a vector norm in the following manner (with $M \in \mathbb{R}^{n\times m}, x \in \mathbb{R}^{m\times 1}$):*

$$\|M\| = \sup_{x\neq 0} \frac{\|Mx\|}{\|x\|} \tag{2.1}$$

or equivalently

$$\|M\| = \sup_{\|x\|=1} \|Mx\|, \tag{2.2}$$

or

$$\|M\| = \sup_{\|x\|\leq 1} \|Mx\|. \tag{2.3}$$

Definition 2.2. *The Frobenius norm of a matrix is defined in the following manner (with $M \in \mathbb{R}^{n\times m}$):*

$$\|M\|_{Fr} = \sqrt{Tr(MM^T)} = \sqrt{\left(\sum_{i=1}^{n}\sum_{j=1}^{m}([M]_{ij})^2\right)}, \tag{2.4}$$

where Tr denotes the trace of a matrix and $[M]_{ij}$ is the element on the i-th row and j-th column of M.

Note that the Frobenius norm is not a natural matrix norm, although it is compatible with the Euclidean norm.

Definition 2.3. *Any $n+1$ vectors $x_o, x_1, \ldots, x_n \in \mathbb{R}^{n\times 1}$ are in "**general position**" if the vectors $x_n - x_j$ ($j = 0, \ldots, n-1$) are linearly independent.*

Definition 2.4. *The "Moore-Penrose generalized matrix inverse" or "pseudo-inverse"[1] $Q^+ \in \mathbb{R}^{n\times m}$ [18,173] for a real matrix $Q \in \mathbb{R}^{m\times n}$ is uniquely defined by the following four properties:*

1. *$QQ^+Q = Q$;*
2. *$Q^+QQ^+ = Q^+$;*
3. *$(QQ^+)^T = QQ^+$;*
4. *$(Q^+Q)^T = Q^+Q$.*

[1] This is also sometimes called the "**general reciprocal**" and written as Q^I, e.g. [113].

Remark 2.1. *When Q is full rank the pseudo-inverse can be computed easily.*

- *For $m > n$ we have $Q^+ = (Q^TQ)^{-1}Q^T$. It will be mainly this case we will be interested in.*
- *For $m < n$ we have $Q^+ = Q^T(QQ^T)^{-1}$.*

Remark 2.2. *If Q is not full rank we can use the SVD decomposition of Q to compute its pseudo-inverse. We can write the singular value decomposition[2] of Q as $Q = LSR^T$, where the singular values are given by σ_i $(i = 1, \ldots, s)$. According to the conventions of the singular value decomposition we have $[S]_{ii} = \sigma_i$ and $[S]_{ij} = 0$ when $i \neq j$. Both L and R are orthogonal matrices.*
Then we have that $Q^+ = RS^+L^T$, where $S^+ \in \mathbb{R}^{n\times m}$ is defined by $[S^+]_{ii} = \sigma_i^{-1}$ and $[S^+]_{ij} = 0$ when $i \neq j$. If a certain singular value is zero, which happens when Q is not full rank, then its inverse in S^+ is replaced by zero.

Remark 2.3. *We also have that $x = Q^+y$ is the least squares solution to the problem $Qx = y$.*

Definition 2.5. *We define the "**range**" $\mathcal{R}$ of $M \in \mathbb{R}^{n\times m}$ as*

$$\mathcal{R}(M) \quad = \quad \{Mx \,|\, x \in \mathbb{R}^{m\times 1}\}. \tag{2.5}$$

Definition 2.6. *We define the "**null space**" or "**kernel**" $\mathcal{N}$ of $M \in \mathbb{R}^{n\times m}$ as*

$$\mathcal{N}(M) \quad = \quad \{x \in \mathbb{R}^{m\times 1} \,|\, Mx = 0\}. \tag{2.6}$$

Definition 2.7. *Let $\mathcal{V}$ be a subspace of the vector space $\mathbb{R}^{n\times 1}$. The "**orthogonal complement**" of $\mathcal{V}$ is the set of vectors which are orthogonal to all elements of $\mathcal{V}$. We write this as $(\mathcal{V})^\perp$.*

Definition 2.8. *A vector $v \in \mathbb{C}^{n\times 1} \setminus \{0\}$ is a "**generalized eigenvector**" of $A \in \mathbb{R}^{n\times n}$ corresponding to the eigenvalue $\lambda \in \mathbb{C}$ if $\exists k \in \mathbb{N} \setminus \{0\}$ such that $(A - \lambda I)^k x = 0$ [10].*

Some properties of eigenvalues and generalized eigenvectors are given below [10, 195, 230].

[2] Different conventions exist for the singular value decomposition (SVD). We will use the one where $L \in \mathbb{R}^{m\times m}, S \in \mathbb{R}^{m\times n}, R \in \mathbb{R}^{n\times n}$ and where the singular values are ordered in a non-increasing way.

- The multiplicity of an eigenvalue λ of A is defined to be the dimension of the subspace spanned by the generalized eigenvectors corresponding to λ and is equal to the multiplicity of λ as a root of the characteristic polynomial, i.e. its algebraic multiplicity.
- The geometric multiplicity of an eigenvalue λ is the dimension of the subspace of (ordinary) eigenvectors of λ.
- The geometric multiplicity of an eigenvalue is smaller than or equal to its algebraic multiplicity.

Definition 2.9. *Let $u, v \in \mathbb{R}^{n\times 1}$. We define a "**rank-one matrix**" by uv^T.*

Definition 2.10. *A mapping $K : D_K \subset \mathbb{R}^{n\times 1} \to \mathbb{R}^{n\times 1}$ is called "**affine**" on D_K if there exists $A \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^{n\times 1}$ such that $K(x) = Ax - b, \forall x \in D_K$.*

Definition 2.11. *If $P^2 = P$ ($P \in \mathbb{R}^{n\times n}$) we say that P is a "**projection matrix**".*

Some properties of a projection matrix are given below [193].

- P defines a projection onto $\mathcal{R}(P)$ parallel to $\mathcal{N}(P)$.
- If P is a projection matrix, then so is $(I - P)$.
- $\mathcal{N}(P) = \mathcal{R}(I - P)$.
- $\mathcal{N}(P) \oplus \mathcal{R}(P) = \mathbb{R}^{n\times 1}$ where $\oplus$ denoted the direct sum of two subspaces.
- Let P be of rank s, $\{v_1, \ldots, v_s\}$ a basis for $\mathcal{R}(P)$ and $\{w_1, \ldots, w_s\}$ a basis for $(\mathcal{N}(P))^\perp$.
  If $V = [v_1| \ldots |v_s], W = [w_1| \ldots |w_s]$, then
  $$P = V(W^T V)^{-1} W^T.$$

Definition 2.12. *If P is a projection matrix such that $\mathcal{R}(P) = (\mathcal{N}(P))^\perp$, then we say that P is an "**orthogonal projection matrix**". If a projection matrix is not orthogonal, we say it is oblique.*

In [193] it is shown that a real projection matrix is orthogonal if and only if it is symmetric.
Note that the term *orthogonal projection matrix* is not to be confused with *orthogonal matrix* which is a matrix for which $P^T P = I$.

Definition 2.13. *[193] Let $\mathcal{Y}_s$ and $\mathcal{Z}_s$ be two subspaces of $\mathbb{R}^{n\times 1}$ of dimension s ($s \leq n$).*
*A "**projection method**" for solving the linear system given in (1.3) (i.e. $K(p) = A_K p - b_K = 0$) is a method in which an approximate solution $\tilde{p}$ is found in an affine subspace $p_o + \mathcal{Y}_s$ (where p_o is an initial guess) and in which a Petrov-Galerkin condition is imposed:*

$$\tilde{r} = A_K \tilde{p} - b \perp \mathcal{Z}_s. \tag{2.7}$$

*We call $p_o + \mathcal{Y}_s$ the "**search subspace**" and $\mathcal{Z}_s$ the "**subspace of constraints**".*
*If $\mathcal{Y}_s = \mathcal{Z}_s$ we say the projection method is "**orthogonal**"[3], otherwise we say it is "**oblique**"[4].*

If $\{v_1, \ldots, v_s\}$ is a basis for $\mathcal{Z}_s$, $\{w_1, \ldots, w_s\}$ a basis for $\mathcal{Y}_s$ and
$V = [v_1|\ldots|v_s], W = [w_1|\ldots|w_s]$, then a projection method defined above (definition 2.13) will result in the approximation $\tilde{p}$ given by

$$\tilde{p} = p_o - W(V^T A_K W)^{-1} V^T r_o, \tag{2.8}$$

if $V^T A_K W$ is non-singular [193].
$V^T A_K W$ is guaranteed to be non-singular if one of the following conditions holds [193]:

1. A_K is positive-definite and $\mathcal{Z}_s = \mathcal{Y}_s$;
2. A_K is non-singular and $\mathcal{Z}_s = A_K \mathcal{Y}_s$, where $A_K \mathcal{Y}_s = \{A_K x | x \in \mathcal{Y}_s\}$.

Definition 2.14. *We define a "**Krylov subspace of dimension** s" generated by $M \in \mathbb{R}^{n\times n}$ and $v \in \mathbb{R}^{n\times 1}$ as*

$$\mathcal{K}_s\{M; v\} = \textit{span}\{v, Mv, M^2 v, \ldots, M^{s-1} v\}. \tag{2.9}$$

If there is no confusion possible we will refer to $\mathcal{K}_s\{M; v\}$ as $\mathcal{K}_s$ for short.
$\mathcal{K}_s\{M; v\}$ is the subspace of all vectors $x \in \mathbb{R}^{n\times 1}$ that can be written as $x = q_{s-1}(M)v$ where $q_{s-1}(M)$ is a polynomial in M of degree $s-1$ or less.

[3] This is also called the "Ritz-Galerkin approach" [222].

[4] This is also called the "Petrov-Galerkin approach" [222].

Definition 2.15. *[193] A "**Krylov subspace method**" is a projection method (definition 2.13) where for the s-th iterate p_s we have $\mathcal{Y}_s = \mathcal{K}_s\{A_K; r_o\}$ when solving the linear system $A_K p = b_K$.*

It follows that p_s will be an approximation of $A_K^{-1} b_K$ such that
$p_s = p_o + q_{s-1}(A_K) r_o$ where $q_{s-1}(A_K)$ is a polynomial in A_K of degree $s-1$ or less.
The choice of $\mathcal{Z}_s$ defines the type of Krylov method within the broader class.

Definition 2.16. *Let $\{x_s\}_{s\in\mathbb{N}}$ be a sequence with x_s and $x^* \in \mathbb{R}^{n\times 1}$. We say that the sequence $\{x_s\}$ converges towards x^* with q-order $\alpha > 1$ if*

$$\exists C, N > 0, \forall s > N : \|x_{s+1} - x^*\| \leq C\|x_s - x^*\|^\alpha, \tag{2.10}$$

for a given norm $\|\cdot\|$ in $\mathbb{R}^{n\times 1}$.

Definition 2.17. *Let $\{x_s\}_{s\in\mathbb{N}}$ be a sequence with x_s and $x^* \in \mathbb{R}^{n\times 1}$. We say that the sequence $\{x_s\}$ converges superlinearly*[5] *towards x^* if*

$$\lim_{s\to\infty} \frac{\|x_{s+1} - x^*\|}{\|x_s - x^*\|} = 0 \tag{2.11}$$

for a given norm $\|\cdot\|$ in $\mathbb{R}^{n\times 1}$.

Definition 2.18. *Let $f : \Omega \subset \mathbb{R}^{n\times 1} \to \mathbb{R}^{m\times 1}$. f is Lipschitz continuous on Ω if $\exists C > 0$ (the Lipschitz constant) such that*

$$\forall p_1, p_2 \in \Omega : \|f(p_1) - f(p_2)\| \leq C\|p_1 - p_2\|. \tag{2.12}$$

If $C < 1$ and $f : \Omega \to \Omega$ then f is called a contraction mapping with respect to the chosen norm.

2.2 Conventions

All matrix norms that are used are natural matrix norms, unless otherwise stated.
$\langle\cdot,\cdot\rangle$ denotes the standard scalar product between vectors defined as follows:

$$\forall x, y \in \mathbb{R}^{n\times 1} : \langle x, y\rangle = x^T y.$$

[5] These definitions are based on "q-superlinearity" as opposed to "r-superlinearity" which is a weaker type of convergence rate [52]; as we only use this type of convergence criteria, we will simply use the term "superlinear", etc.

Orthogonality of vectors or vector spaces will be expressed with respect to this scalar product unless noted otherwise.
We use the symbol $\vec{\subset}$ to denote a vector subspace.

2.3 General theorems

Theorem 2.3.1. *[Fundamental Theorem of Linear Algebra[6]]*
Let $Q \in \mathbb{R}^{n\times m}$, then $\mathcal{N}(Q) = (\mathcal{R}(Q^T))^\perp$.

For a proof of this theorem we refer to [212].

Lemma 2.3.1. $\forall u, v \in \mathbb{R}^{n\times 1} : \det(I + uv^T) = 1 + \langle u, v\rangle$.

Proof. Let $P = I + uv^T$.
For $u = 0$ or $v = 0$ the proof is trivial.
If $u, v \neq 0$ and $\langle u, v\rangle \neq 0$, then any vector orthogonal to v is a right eigenvector of P (corresponding to an eigenvalue 1) and any multiple of u is also a right eigenvector of P (corresponding to an eigenvalue $1 + \langle u, v\rangle \neq 0$). As there are $n - 1$ vectors orthogonal to v, and as the algebraic multiplicity of an eigenvalue is larger than or equal to its geometric multiplicity, we see that the algebraic multiplicity of the eigenvalue 1 is at least $n-1$. As there is another eigenvalue different from 1, the algebraic multiplicity of the eigenvalue 1 must be equal to $n - 1$. As the determinant of a matrix equals the product of its eigenvalues, we have that $\det P = 1 + \langle u, v\rangle$.
If $u, v \neq 0$ and $\langle u, v\rangle = 0$ then $1 + \langle u, v\rangle = 1$. But then

$$(P - I)^2 = (uv^T)^2 = uv^Tuv^T = 0.$$

So the space of generalized eigenvectors corresponding to the eigenvalue 1 has dimension n. Hence the algebraic multiplicity of the eigenvalue 1 is n and
$\det P = 1$. □

Remark We like to point out that the property given in lemma 2.3.1 can be found in most reference works on linear algebra but that this is the first time, to our knowledge, that a complete proof is given that also takes into account the case for $u, v \neq 0$ and $\langle u, v\rangle = 0$.

[6] Also known as Fredholm's Theorem.

Lemma 2.3.2. *Let $V \in \mathbb{R}^{n\times s}$ be a matrix with rank r, then*

$$VV^+ = \mathcal{L}_r \mathcal{L}_r^T, \tag{2.13}$$

with $\mathcal{L}_r = [\bar{L}_1 | \bar{L}_2 | \ \ldots \ |\bar{L}_r] \in \mathbb{R}^{n\times r}$, where $\bar{L}_k$ is the k-th left (normalized) singular vector of V [7].

Proof. Let $V = LSR^T$ be the singular value decomposition of V. Then

$$\begin{aligned} VV^+ &= (LSR^T)(RS^+L^T) \\ &= LSS^+L^T. \end{aligned}$$

Because of the special structure of S and S^+ (see definition 2.4) we have $SS^+ = I_{n,r}$, where $I_{n,r} = \text{diag}(\underbrace{1, 1, \ldots, 1}_{r \text{ times}}, \underbrace{0, 0, \ldots, 0}_{n-r \text{ times}})$. Hence

$$VV^+ = LI_{n,r}L^T.$$

We can further reduce this expression by introducing $\mathcal{L}_r = [\bar{L}_1 | \bar{L}_2 | \ \ldots \ |\bar{L}_r]$, where $\bar{L}_k$ is the k-th left singular vector of V:

$$VV^+ = \mathcal{L}_r \mathcal{L}_r^T,$$

which completes the proof. □

Note that $VV^+ = \mathcal{L}\mathcal{L}^T$ is an orthogonal projection matrix on the range of V (definition 2.11 and 2.12).

Lemma 2.3.3. *Let $V \in \mathbb{R}^{n\times s}$ be a matrix with rank r and $M \in \mathbb{R}^{m\times n}$, then, using the notation of lemma 2.3.2,*

$$\mathcal{N}(M\mathcal{L}\mathcal{L}^T) \supset (\mathcal{R}(V))^\perp \tag{2.14a}$$

$$\mathcal{N}(M(\mathcal{L}\mathcal{L}^T - I)) \supset \mathcal{R}(V). \tag{2.14b}$$

If M has rank n, then

$$\mathcal{N}(M\mathcal{L}\mathcal{L}^T) = (\mathcal{R}(V))^\perp \tag{2.15a}$$

$$\mathcal{N}(M(\mathcal{L}\mathcal{L}^T - I)) = \mathcal{R}(V). \tag{2.15b}$$

[7] The first r singular vectors of V form an orthonormal basis for the range of V

Proof. (2.14) is an immediate consequence of lemma 2.3.2 as
$\forall x \in \mathcal{R}(V) : \mathcal{L}\mathcal{L}^T x = x$ and $\forall y \in (\mathcal{R}(V))^\perp : \mathcal{L}\mathcal{L}^T y = 0$.
To prove (2.15b) we note that from $Mx = 0$ it follows that $x = 0$ if the rank of M is n . From $M(\mathcal{L}\mathcal{L}^T - I)x = 0$ it follows that $\mathcal{L}\mathcal{L}^T x = x \in \mathcal{R}(V)$.
We now prove (2.15a). If $M\mathcal{L}\mathcal{L}^T x = 0$ then it follows that $\mathcal{L}\mathcal{L}^T x = 0$, such that $\forall y \in \mathbb{R}^{s\times 1}$: $0 = \langle \mathcal{L}\mathcal{L}^T x, Vy\rangle = \langle V^T V V^+ x, y\rangle$.
Using the properties of the SVD decomposition of the Moore-Penrose generalized inverse (definition 2.4) we obtain
$0 = \langle V^T V V^+ x, y\rangle = \langle V^T x, y\rangle = \langle x, Vy\rangle$
and hence $x \in (\mathcal{R}(V))^\perp$. □

Lemma 2.3.4. *If $\mathcal{A}\vec{\subset}\mathcal{B}\vec{\subset}\mathbb{R}^{n\times 1}$ then $\|MP_\mathcal{A}\| \leq \|MP_\mathcal{B}\|$, where $P_\mathcal{A}$, resp. $P_\mathcal{B}$, is an orthogonal projection matrix on $\mathcal{A}$, resp. $\mathcal{B}$ and $M \in \mathbb{R}^{n\times n}$.*

Proof. Let $z \in \mathbb{R}^{n\times 1}$, with $\|z\| \leq 1$. Then we have $\|P_\mathcal{A} z\| \leq 1$
and $P_\mathcal{A} z = P_\mathcal{B} P_\mathcal{A} z$. Hence

$$\|MP_\mathcal{A} z\| = \|MP_\mathcal{B}(P_\mathcal{A} z)\| \leq \sup_{x\in\mathbb{R}^{n\times 1};\|x\|\leq 1} \|MP_\mathcal{B} x\| = \|MP_\mathcal{B}\|.$$

From this it follows that

$$\|MP_\mathcal{A}\| = \sup_{z\in\mathbb{R}^{n\times 1};\|z\|\leq 1} \|MP_\mathcal{A} z\| \leq \|MP_\mathcal{B}\|.$$

□

Theorem 2.3.2. *Let $T_1, T_2 \in \mathbb{R}^{n\times n}$. Let $\{X_s\}_{s\in[1,n]}$ be an arbitrary sequence of vectors ($X_s \in \mathbb{R}^{n\times 1}$) that are linearly independent, $\{V_s\}_{s\in[1,n]}$ be a sequence defined by $V_s = [X_1|\ X_2|\ \ldots |\ X_s]$ and let $\{Q_s\}_{s\in[1,n]}$ be a sequence defined by $Q_s = T_1 V_s V_s^+ - T_2$;*
then

$$\|Q_{s+1} - (T_1 - T_2)\| \leq \|Q_s - (T_1 - T_2)\|, \tag{2.16}$$

for $s = 1, 2, \ldots$, and

$$Q_n = T_1 - T_2. \tag{2.17}$$

Proof. We first note that the rank of V_s is s and that for $s = n$ we will have

$$Q_n = T_1 V_n V_n^+ - T_2 = T_1 - T_2, \tag{2.18}$$

and hence $\|Q_n - (T_1 - T_2)\| = 0$, which proves (2.17).
For $s < n$ lemma 2.3.2 gives us

$$Q_s = T_1 \mathcal{L}_s \mathcal{L}_s^T - T_2, \tag{2.19}$$

where $\mathcal{L}_s = [\bar{L}_1 | \bar{L}_2 | \ \ldots \ | \bar{L}_s]$, with $\{\bar{L}_k\}_{k=1}^s$ an orthonormal basis for the range of V_s.
Hence

$$\begin{aligned} Q_s - (T_1 - T_2) &= T_1(\mathcal{L}_s \mathcal{L}_s^T - I) && (2.20)\\ Q_{s+1} - (T_1 - T_2) &= T_1(\mathcal{L}_{s+1} \mathcal{L}_{s+1}^T - I). && (2.21) \end{aligned}$$

Introducing P_s, resp. P_{s+1}, as the orthogonal projection matrix on $(\mathcal{R}(V_s))^\perp$, resp. $(\mathcal{R}(V_{s+1}))^\perp$, we obtain

$$\begin{aligned} Q_s - (T_1 - T_2) &= T_1 P_s && (2.22)\\ Q_{s+1} - (T_1 - T_2) &= T_1 P_{s+1}. && (2.23) \end{aligned}$$

As $(\mathcal{R}(V_{s+1}))^{\perp} \vec{\subset} (\mathcal{R}(V_s))^\perp$ we can use lemma 2.3.4 to obtain

$$\begin{aligned} \|T_1 P_{s+1}\| &\leq \|T_1 P_s\| && (2.24)\\ \|Q_{s+1} - (T_1 - T_2)\| &\leq \|Q_s - (T_1 - T_2)\|, && (2.25) \end{aligned}$$

which completes our proof. □

Theorem 2.3.3. *Sherman-Morrison Theorem [201]*
Let $Q \in \mathbb{R}^{n \times n}$ be non-singular, and let $u, v \in \mathbb{R}^{n \times 1}$ be vectors such that $v^T Q^{-1} u \neq -1$, then $Q + uv^T$ is non-singular and

$$(Q + uv^T)^{-1} = Q^{-1} - \frac{Q^{-1} u v^T Q^{-1}}{1 + v^T Q^{-1} u}. \tag{2.26}$$

Proof. The non-singularity of $Q + uv^T$ follows from lemma 2.3.1: $\det(Q + uv^T) = \det(Q) \det(I + (Q^{-1} uv^T)$; equation (2.26) is easily verified by multiplying both sides by $Q + uv^T$. □

This theorem is also sometimes called the Sherman-Morrison-Woodbury theorem [52], although strictly speaking this is a generalized theorem given in the following form [113].

Theorem 2.3.4. *Sherman-Morrison-Woodbury Theorem*
Let $Q \in \mathbb{R}^{n \times n}$ and $S \in \mathbb{R}^{m \times m}$ be non-singular, and let $U, V \in \mathbb{R}^{n \times m}$ ($m \leq n$), then $(Q - USV^T)$ is non-singular and there exists a non-singular matrix $W \in \mathbb{R}^{m \times m}$ such that

$$(Q - USV^T)^{-1} \quad = \quad Q^{-1} - Q^{-1}UWV^TQ^{-1}, \tag{2.27}$$

where W is defined implicitly by

$$W^{-1} + S^{-1} \quad = \quad V^TQ^{-1}U. \tag{2.28}$$

Theorem 2.3.5. *Assume we are solving $A_Kp - b_k = 0$ with $b_K \in \mathbb{R}^{n \times 1}$ and $A_K \in \mathbb{R}^{n \times n}$ non-singular. If the minimal polynomial of A_K has degree δ, then the solution of the system is contained in the subspace $\mathcal{K}_\delta(A; b_K)$.*

A proof of this theorem is an immediate consequence of the definition of the minimal polynomial [117].

This theorem explains the main strength of Krylov methods in that (in exact arithmetic) the solution will be found in at most n iterations, and possibly less, depending on the degree of the minimal polynomial of the system matrix.

Theorem 2.3.6. *Assume we are solving $A_Kp - b_K = 0$, with $A_K \in \mathbb{R}^{n \times n}$ an arbitrary matrix; assume that we have $\mathcal{Z}_s = A_K\mathcal{Y}_s$, defined as in definition 2.13. Then a vector $\tilde{p}$ is the result of an oblique projection method onto $\mathcal{Y}_s$ orthogonally to $\mathcal{Z}_s$ with the starting iterate p_o if and only if it minimizes the Euclidean norm of the residual vector $\tilde{r} = A_K\tilde{p} - b_K$ over $p \in p_o + \mathcal{Y}_s$.*

For a proof of this theorem we refer to [193].

Theorem 2.3.7. *Let $A_K \in \mathbb{R}^{n \times n}$ be an arbitrary matrix. Let $\tilde{p}$ be the approximate solution to $A_Kp - b_K = 0$, obtained from a projection process onto $\mathcal{Y}_s$ orthogonally to $\mathcal{Z}_s = A_K\mathcal{Y}_s$ ($\mathcal{Y}_s$ and $\mathcal{Z}_s$ defined as in definition 2.13) and let $\tilde{r} = A_K\tilde{p} - b_K$ be the associated residual vector. Then*

$$\tilde{r} \quad = \quad (I - P)r_o, \tag{2.29}$$

where P denotes the orthogonal projection matrix onto the subspace $\mathcal{Z}_s$.

For a proof of this theorem we refer to [193].
The class of methods respecting theorem 2.3.7 is called “residual projection methods”.

3
Solution methods for a single system of linear equations

While, in general, we are interested in solving the non-linear system of equations (1.1), a substantial part of the analytical study of the available algorithms will be performed assuming that either $F(g)$, $S(p)$ and/or $H(p)$ are affine operators. We therefore include a small overview of the most important linear solvers known in the literature.
In this chapter the system of equations we are trying to solve is

$$Ap \quad = \quad b, \tag{3.1}$$

where $p, b \in \mathbb{R}^{n \times 1}$, $A \in \mathbb{R}^{n \times n}$. (We will drop the subscript "K" when no confusion is possible for ease of reading.)

Readers with a good knowledge of this matter can skip this chapter.

3.1 Legacy solvers: Richardson, Jacobi, Gauss-Seidel and SOR

As no introduction about linear solvers can be complete without mentioning the most elementary of them, we briefly summarize four of the best-known linear solvers, even though they are no longer competitive with current Krylov subspace methods.

For more details we refer to any of the handbooks on numerical algebra like [72], [98], [101], [127], [226] and [247], to name but a few.

3.1.1 Richardson iteration

In the Richardson method the iteration can be written in one of the following forms:

$$p_{s+1} = (I - \omega_s A)p_s + \omega_s b \tag{3.2a}$$

$$p_{s+1} = p_s - \omega_s r_s, \tag{3.2b}$$

where $\omega_s \in \mathbb{R}_o$ is a relaxation parameter, I is the identity matrix and r_s is the residual of p_s defined as $Ap_s - b$.

If ω_s varies from iteration to iteration we speak of a non-stationary Richardson iteration; if $\omega_s = \omega_o$ is fixed, then we speak of a stationary Richardson iteration.

Note that the Richardson iteration is essentially identical to the fixed point iteration with dynamic relaxation (algorithm 1.2.2).

3.1.2 Jacobi iteration

In the Jacobi method the iteration can be written in one of the following forms:

$$p_{s+1} = D^{-1}(D - A)p_s + D^{-1}b \tag{3.3a}$$

$$p_{s+1} = p_s - D^{-1}r_s, \tag{3.3b}$$

where D is defined by

$$[D]_{ij} = \begin{cases} 0 & \text{for} \quad i \neq j \\ [A]_{ij} & \text{for} \quad i = j \end{cases} . \tag{3.4}$$

Often an under-relaxation parameter ω_s is added:

$p_{s+1} = p_s - \omega_s D^{-1} r_s$ for $s = 0, 1, \ldots$.

3.1.3 Gauss-Seidel iteration

In the forward Gauss-Seidel method the iteration can be written in one of the following forms:

$$p_{s+1} = (D + L)^{-1}(D + L - A)p_s + (D + L)^{-1}b \tag{3.5a}$$

$$p_{s+1} = p_s - (D + L)^{-1}r_s, \tag{3.5b}$$

where L is defined by

$$[L]_{ij} = \begin{cases} 0 & \text{for} \quad i \geq j \\ [A]_{ij} & \text{for} \quad i < j \end{cases} . \tag{3.6}$$

In the backward Gauss-Seidel method this becomes:

$$p_{s+1} = (D+U)^{-1}(D+U-A)p_s + (D+U)^{-1}b \tag{3.7a}$$

$$p_{s+1} = p_s - (D+U)^{-1}r_s, \tag{3.7b}$$

where U is defined by

$$[U]_{ij} = \begin{cases} 0 & \text{for} \quad i \leq j \\ [A]_{ij} & \text{for} \quad i > j \end{cases} . \tag{3.8}$$

3.1.4 Successive Over-Relaxation (SOR)

The forward SOR method is a parametrized version of the forward Gauss-Seidel method, given by:

$$p_{s+1} = (D+\omega L)^{-1}((1-\omega)D - \omega U)p_s + \omega(D+\omega L)^{-1}b, \tag{3.9}$$

where $\omega \in \mathbb{R}$ is a relaxation parameter.
By analogy one can also start from the backward Gauss-Seidel method to obtain the backward SOR method:

$$p_{s+1} = (D+\omega U)^{-1}((1-\omega)D - \omega L)p_s + \omega(D+\omega U)^{-1}b. \tag{3.10}$$

3.2 Krylov subspace methods

Krylov subspace methods[1] [131] have been defined in definitions 2.13-2.15. They have been developed as linear solvers for systems of the form of equation (3.1). The choice of the search subspace $p_o + \mathcal{Y}_s$ and subspace of constraints $\mathcal{Z}_s$ defines the particular method within this class.
In these methods the system matrix does not need to be stored or formed; only a routine for matrix-vector products Ax is needed $\forall x \in \mathbb{R}^{n\times 1}$. For that reason these methods are also known as "*matrix-free*" methods.
We will briefly describe some of the best known Krylov methods in the following paragraphs and mention their applicability to our framework. For further reading we refer to the excellent book by Van der Vorst [222].

[1] Also known as "Krylov projection methods" and "Krylov methods" for short.

3.2.1 The Full Orthogonalization Method

The “Full Orthogonalization Method” or FOM[2], is a Krylov subspace method defined by $\mathcal{Y}_s = \mathcal{Z}_s = \mathcal{K}_s(A; r_o)$; it is thus an orhogonal projection method.
It is intended for general non-Hermitian matrices A and, in exact arithmetic, converges in at most n steps [193].
Variants of the method include “restarted FOM” where the method is stopped after a number of iterations and the iterate at that point is then used as an initial iterate for a new iteration loop.
Another variant is the “Incomplete Orthogonalization Method” or IOM where only a limited number of the last basis vectors of the search subspace and subspace of constraints are kept.

3.2.2 The Generalized Minimal Residual Method (GMRes)

GMRes [192,193,222], which is a generalization of the MinRes algorithm of Paige and Saunders [167] and based on the Arnoldi orthogonalization process [7,190], is perhaps the best known Krylov subspace method for general non-Hermitian matrices A. It is defined by $\mathcal{Y}_s = \mathcal{K}_s(A; r_o)$ and $\mathcal{Z}_s = A\mathcal{K}_s(A; r_o)$ when solving (3.1); it is thus an oblique projection method (definition 2.13) and a residual projection method (see theorem 2.3.7).
As $p_s \in p_o + \mathcal{K}_s(A; r_o)$ implies $r_s \in r_o + A\mathcal{K}_s(A; r_o)$, and as $r_s \perp A\mathcal{K}_s(A; r_o)$, it results that p_s is the unique value in the search subspace that minimizes $\|r_s\|_2$ (which denotes the Euclidean norm of r_s).
Hence, another way to look at GMRes is the following: we construct

$$p_s = p_o + \sum_{i=1}^{s} \omega_i A^{i-1} r_o \tag{3.11}$$

with coefficients $\{\omega_i\}_{i=1}^{i=s}$ chosen such that

$$\|r_s\|_2 = \|r_o + \sum_{i=1}^{s} \omega_i A^i r_o\|_2 \tag{3.12}$$

is minimal.

Well known properties of GMRes [98, 222] are:

[2] Sometimes called “Arnoldi method” [7, 193], although this term more correctly refers to the orthogonalization procedure contained in the FOM.

- GMRes generates a residual $r_s \in r_o + A\mathcal{K}_s\{A; r_o\}$, at which point it has used $s + 1$ function calls, corresponding to matrix-vector products in the linear case[3].
- GMRes will converges in n iterations or less (in exact arithmetic) corresponding to $n + 1$ function calls (or less).
- The convergence of GMRes (measured by the Euclidean norm of the residual) is strictly non-increasing. This does not necessarily mean the residual decreases as it can stagnate for maximally $n - 1$ iterations (see [96] for more details).
- In exact arithmetic the method will not encounter any singularities (cfr. definition 2.13 [193]).
- When used in Newton iterations and if 0 is the initial guess, GMRes offers descent directions for minimizing KK^T [23]) and monotone errors [31].

These properties all suppose that the operations are performed in exact arithmetic. For the behavior of GMRes in the context of finite precision arithmetic see [5, 60, 97, 98]. Other properties can be found in [127, 222] and references therein.
We can use GMRes to solve (1.3) when it represents a linear problem. As we are only dealing with variables on the interface between the two constituent problems S and F, the term *Interface GMRes* is sometimes used in this context.

Variants of GMRes include "restarted GMRes" or GMResR[4] where the method is stopped after a number of iterations and the iterate at that point is then used as an initial iterate for a new iteration loop. This variant does not keep the property of convergence in at most n steps [193].
Another variant is "quasi-GMRes" (or "truncated GMRes") where only a limited number of the last basis vectors of the search subspace and subspace of constraints are kept.
As recent developments of GMRes we mention GMBack, MinPert [33, 124, 125] and "*simpler* GMRes" [233].
A variant working with affine operators will be described in §3.2.2.2.

3.2.2.1 Elementary implementation of GMRes

In the GMRes method it is assumed that for every $p \in \mathbb{R}^{n\times 1}$ we are able to form Ap, which differs from what we have assumed in chapter 1, i.e. we normally

[3] We recall that the reason for counting function calls is that we have assumed that these constitute the dominant computational cost of any algorithm.

[4] Also called GMRES(m) [127,192,222] where m indicates the number of iterations before a restart is performed; Van der Vorst uses "GMRESR" for a particular case of preconditioned GMRes.

assume we can form $H(p)$, $K(p)$ etc., which corresponds to $Ap - b$.
In its most basic form GMRes can be written as in algorithm 3.2.1.

Algorithm 3.2.1. *GMRes method*

```
1. Take a starting value $p_o$;
   $r_o = Ap_o - b$;
   $s = 1$.
2. Loop until sufficiently converged:
   a. Compute $A^s r_o$.
   b. Find coefficients $\{\bar{\omega}_i\}_{i=1}^{i=s}$ that minimize $\|r_s\|_2$ with $r_s = r_o + \sum_{i=1}^{s} \omega_i A^i r_o$.
   c. Optionally: $p_s = p_o + \sum_{i=1}^{s} \bar{\omega}_i A^{i-1} r_o$.
   d. Set $s = s + 1$.
```

Note that the computation of p_s in 2.c is optional, and is normally only performed when the algorithm has sufficiently converged.

Most of the analysis of GMRes that will be done in this work takes an analytical point of view, meaning that we will not use GMRes in the form that is the most numerically stable as can be found in most textbooks on Krylov methods, e.g. [12, 98, 127, 193, 222, 232].
The most common modification is an orthogonalization of the basis vectors of the Krylov search subspace; this orthogonalization can be any of those discussed in §3.2.4.

3.2.2.2 Adaptation of GMRes to our framework

When we want to solve (1.3) when $H(p)$ is an affine (black box) operator, we are only able to form $H(p) = A_H p - b_H \; \forall p \in \mathbb{R}^{n \times 1}$ without explicitly knowing b_H, according to our framework as outlined in chapter 1.
Most formulations of GMRes however assume that the user has knowledge of the right hand side b_H $(= b_K)$ and can compute the matrix-vector product $(A_H - I)p = A_K p$ for any given $p \in \mathbb{R}^{n \times 1}$ as outlined above.
Within our self-imposed constraints constructing $r_o = A_H p_o - b_H - p_o$ poses no problem, but finding $A_K^i r_o$ $(i = 0, \ldots, s)$ does. In [218] an algorithm was proposed that circumvents this problem. The idea behind it is the following: the

exact solution of $A_K p = b_K$ would be given by

$$p^* \quad = \quad p_o \underbrace{-A_K^{-1} r_o}_{\delta p}, \tag{3.13}$$

which could be seen as a single (exact) Newton iteration. Re-arranging the terms we get

$$r_o + A_K \delta p \quad = \quad 0. \tag{3.14}$$

As A_K is not available, we approximate δp by $\widehat{\delta p}_s \in \text{span}\{\delta_1, \delta_2, \dots, \delta_s\}$ ($s = 1, 2, \dots$), where $\delta_s = d_s - d_o$, $d_o = p_o$ and $d_{i+1} = H(d_i)$. p^* is thus approximated by the iterate $p_s = p_o + \widehat{\delta p}_s$ to which corresponds the residual r_s which we can write as

$$\begin{aligned} r_o + A_K \sum_{i=1}^{s} \omega_i \delta_i \quad &= \quad r_o + A_K \sum_{i=1}^{s} \omega_i \left(d_i - d_o\right) \\ &= \quad r_o + \sum_{i=1}^{s} \omega_i \left(A_K d_i - A_K d_o\right) \\ &= \quad r_o + \sum_{i=1}^{s} \omega_i \underbrace{\left(\left(d_{i+1} - d_i\right) - r_o\right)}_{\rho_i}. \end{aligned}$$

According to (3.14) r_s should ideally be zero; coefficients $\bar{\omega}_1, \dots, \bar{\omega}_s$ are thus computed that minimize $\|r_o + \sum_{i=1}^{s} \omega_i \rho_i\|_2$.

It is easy to show that

$$\rho_j = A_K^j r_o - \sum_{k=1}^{j-1} \binom{j}{k} (-1)^{k+j} \rho_k \tag{3.15}$$

and thus that $r_o + \text{span}\{\rho_1, \dots, \rho_s\} = r_o + A_K \ \text{span}\{r_o, A_K r_o, \dots, A_K^{s-1} r_o\}$, which shows that the resulting method is indeed analytically identical to the classical implementation of GMRes.

The resulting algorithm is given in 3.2.2.

Algorithm 3.2.2. *GMRes - affine variant [218]*

```
1. Take a starting value $p_o$;
   set $d_o = p_o$;
   compute $d_1 = H(d_o) = A_H d_o - b_H$;
   $r_o = d_1 - d_o$;
   $s = 1$.
2. Loop until sufficiently converged:
   a. Compute $d_{s+1} = H(d_s) = A_H d_s - b_H$.
   b. $\delta_s = d_s - d_o$.
   c. $\rho_s = (d_{s+1} - d_s) - r_o$.
   d. Find coefficients $\{\bar{\omega}_i\}_{i=1}^{i=s}$ that minimize $\|r_s\|_2$ with $r_s = r_o + \sum_{i=1}^{s} \omega_i \rho_i$.
   e. Optionally: $p_s = p_o + \sum_{i=1}^{s} \bar{\omega}_i \delta_i$.
   f. Set $s = s + 1$.
```

Note that the computation of p_s in 2.e is optional, and is normally only performed when the algorithm has sufficiently converged.

Our experience with this variant of GMRes is that it exhibits poor numerical stability. In [218] a remedy was proposed in which the δ_s $(s = 1, 2, \dots)$ were orthogonalized. As, in our view, it is mainly the conditioning of the ρ_s $(s = 1, 2, \dots)$ which causes the instability in 2.d of algorithm 3.2.1, we noted only a slight improvement with this modification as it does not guarantee orthogonality (or indeed a better conditioning) of ρ_s $(s = 1, 2, \dots)$.
As we have not found a better alternative in the literature, we therefore propose to use standard GMRes (e.g. [193, 222]) after finding $b_K (= b_H)$ with a function call $K(0)$. If the initial iterate is $p_o = 0$ then this invokes no additional cost; otherwise an extra function call needs to be spent.

Remark 3.1. *In chapter 8 (§8.2.1) we will show that GMRes can also be written as a quasi-Newton method.*

3.2.3 Other Krylov subspace methods

We only briefly outline some of the other Krylov subspace methods, either because they cannot immediately be applied to the framework as stated in chapter 1 or because they are variants of the Krylov subspace methods given above that are algebraically identical to GMRes.

3.2.3.1 The Lanczos Method, Conjugate Gradient Method (CG), CGNR, LSQR, CGLS, CGNE, OrthoRes and GENCG

The Lanczos, resp. Conjugate Gradient, method [112, 134, 222] can be viewed as the Full Orthogonalization Method applied to (3.1) when A is symmetric, resp. **s**ymmetric **p**ositive **d**efinite (spd). This allows to simplify the algorithm [193].
In our framework we do not assume this condition for A.

An extension of CG is the CGNR ("Conjugate Gradient on the Normal equations to minimize the Residual") and CGNE ("Conjugate Gradient on the Normal equations to minimize the Error") methods. In CGNR [65, 81, 112, 163] (also known as LSQR [168] or CGLS [222]) the original problem $Ap = b$ is transformed to $A^T Ap = A^T b$ and as $A^T A$ is a spd matrix CG can then be applied. Similarly, in CGNE [39, 81, 163] the CG method is applied to $AA^T y = b$ with $x = A^T y$.
The disadvantages of CGNR, LSQR, CGLS and CGNE are that two matrix vector product are needed per iteration: one with A and one with A^T. Not only is this expensive in our framework, but we also assume that a matrix-vector product with A^T is unavailable.
Methods that are analytically equivalent to FOM are OrthoRes [122] and the Generalized Conjugate Gradient Method (GENCG or GCG) [38, 192, 222, 237].

For further reading about these methods we refer to [93], [101] and [222].

Remark An interesting discussion about the classical argument that CGNE, LSQR, CGLS and CGNR cannot be competitive because the squaring of the condition number of A by either $A^T A$ or AA^T (and other arguments against these methods) can be found in [98].

3.2.3.2 The Generalized Conjugate Residual Method (GCR), Axelsson's method, OrthoDir, OrthoMin, GENCR and MinRes

The Generalized Conjugate Residual method [65] is mathematically equivalent to GMRes, but with double the amount of storage required and 50% more arithmetic operations per step compared to GMRes [193].
Axelsson's method [8], Orthodir [122], Orthomin [229] and GENCR [65] are four other Krylov methods that are analytically identical to GMRes but differ in their implementation.
MinRes [167] is the resulting method when GMRes is applied to systems with a symmetric matrix.

3.2.3.3 Bi-Conjugate Gradients (Bi-CG), Conjugate Gradient Squared (CGS), Bi-CGStab, QMR and TFQMR

Other methods for general, non-symmetric systems have been developed, all of which need two matrix-vector products[5].
It is well documented that these methods can be faster in terms of iterations than GMRes, but at a cost of a higher number of matrix-vector products, which makes them uneconomical in our framework [98, 127], and will not be discussed further for that reason.

For further reading we refer to [29, 77, 82, 83, 98, 127, 134, 208, 220, 222].

3.2.4 Orthogonalization procedures

A key ingredient of Krylov methods is the construction of a basis for the Krylov subspaces. As the conditioning of the basis of the search subspace has a major impact on the numerical stability of the algorithms, and as the vectors r_o, Ar_o, $A^2 r_o, \dots$ point more and more in the direction of the dominant eigenvector, an orthogonalization procedure is almost invariably added to the method [193, 222]. We will briefly discuss several orthogonalization procedures in this paragraph.

The best-known orthogonalization procedure is the Gram-Schmidt (GS) method. In its original form it is not numerically stable [193] and is therefore better replaced by the Modified Gram-Schmidt method (MGS), although both are algebraically identical.
These methods are typically implemented in the Arnoldi method, in which the (modified) Gram-Schmidt method is applied to the basis-vectors of $\mathcal{K}_s(A, r_o)$, storing the scalar products of the GS method in an upper Hessenberg matrix.
Even with the MGS method numerical instabilities can develop. This can happen when the new basis vector has only a very small component that is orthogonal to the previous basis vectors. A solution, that is known to work well, is to re-orthogonalize a basis-vector if this occurs [193].
An even more stable algorithm, albeit at a higher cost, is the Householder-Arnoldi method which uses reflection matrices, as proposed by Walker [234].

The typical implementation of GMRes (and most other Krylov methods) also includes an orthogonalization process, as found in [193] or [222] for instance.

[5] Some of the methods require a matrix-vector product with A^T.

4
Solution methods for a single system of non-linear equations

In this chapter we give an overview of the most commonly encountered quasi-Newton solvers.

4.1 Newton's method

Probably the best known solution method for (systems of) non-linear equations is Newton's method [164], also known as Newton-Raphson's method[1] [19, 33, 166, 184], which for the solution of (1.3) is given by:

$$p_{s+1} = p_s - (K'(p_s))^{-1} K(p_s), \tag{4.1}$$

where $K'(p_s)$ represents the Jacobian of K evaluated at p_s.
The hypotheses 1.1, 1.2 and 1.3 in §1.1.1 ensure that there exists an open set D which contains the solution p^* such that for any initial $p_o \in D$ the Newton iterates are well-defined, remain in D and converge superlinearly to p^* as specified by the Newton attraction theorem [166]. If $K'(p)$ is also Lipschitz continuous for all p

[1] Actually, Newton only developed the method for single equations; it was Simpson that extended it to systems of equations [248]. Raphson was responsible for the practical implementation of the algorithm as we know it today [184].

close enough to p^* then the convergence is quadratic. (For more details see [166].)

The main disadvantages of the method are that it requires a good initial guess p_o and the possible cost of computing the Jacobian. If the latter is not immediately available it can be replaced by a finite difference approximation (requiring n function evaluations in general) or it can be kept constant during a number of iterations. In the context of this study we assume that this Jacobian is indeed unavailable and that it is too expensive to compute it by finite differences. For that reason we turn our attention to quasi-Newton methods in the next section.

Note that in actual computations the inverse of a matrix is almost never explicitly computed; for instance (4.1) is most often replaced by

$$\begin{aligned} K'(p_s)\delta &= -K(p_s) & (4.2) \\ p_{s+1} &= p_s + \delta. & (4.3) \end{aligned}$$

The solution of (4.2) can be obtained with any linear solver. If this linear solver is a Krylov method, the resulting method is called a Newton-Krylov (e.g. Newton-GMRes) method, based on the naming convention in [166] and [200].
If (4.2) is only solved approximately by the linear solver the term “*Inexact*” or “*Truncated*” Newton method is also used.
The disadvantage of Newton-Krylov methods for non-linear problems lies in the number of function evaluations needed. For these methods one function evaluation is needed for each outer and inner iteration[2] while for a quasi-Newton method one function evaluation is needed per outer evaluation. (There is no inner iteration.)

4.2 Quasi-Newton methods

When the Jacobian of K is unavailable, or too expensive to compute, we can replace the true Jacobian of Newton’s method by an approximation, resulting in what is called a *quasi-Newton method* (QN method). While, strictly speaking, any approximation would result in a quasi-Newton method (e.g. the chord method, Jacobi iteration etc.), the term is generally reserved for specific methods. Other names have been used in the literature like: *variable metric method*, *variance method*, *modification method*, and *secant update method*. We will reserve the designator *secant method* for a specific subclass of quasi-Newton methods which we will discuss below.

[2] By “outer iteration”, or “Newton step”, we mean equation (4.3), while by “inner iteration’ we mean an iteration needed to solve (4.2).

A particular new development that can be considered to be a quasi-Newton method is that where the Jacobian of a set of equations, describing a physical phenomenon, is approximated by the Jacobian of a simpler, though related, physical phenomenon (e.g. [88, 89]).

Historically, most quasi-Newton methods have been developed to solve non-linear equations resulting from a minimization problem and thus require a symmetric (possibly positive-definite) Jacobian; we will only briefly mention these methods in this chapter, as we have not made this assumption about the Jacobian for the problems we are trying to solve (see chapter §1).

Quasi-Newton methods can take two forms and are either defined by

$$p_{s+1} = p_s - (\hat{K}'_s)^{-1} K(p_s) \tag{4.4}$$

or

$$p_{s+1} = p_s - \hat{M}'_s K(p_s), \tag{4.5}$$

($s = 0, 1, 2, \dots$) where $\{\hat{K}'_s\}_{s \in \mathbb{N}}$ is a sequence of approximations to $K'(p_s)$ and $\{\hat{M}'_s\}_{s \in \mathbb{N}}$ is a sequence of approximations to $(K'(p_s))^{-1}$.
Sometimes an iteration parameter is added to (4.4) and (4.5); this parameter can be a fixed value, or based on line-searches. We refrain from the latter as a line-search invokes a high number of function calls, which we have considered very expensive. For that reason, we will only use a fixed under-relaxation parameter in the first time-step to avoid excessive divergence of the iterations, which would hamper further convergence (see chapter 10). Further discussion about the use of relaxation parameters in linear problems can be found in §8.3.

Again, the inverse of the approximate Jacobian is rarely explicitly computed[3]. As for the Newton method a linear system is solved instead.

If $K(p)$ represents an affine mapping ($K(p) = A_K p - b_K$) then the resulting error ($e_{s+1} = p_{s+1} - p^*$), resp. residual ($r_{s+1} = K(p_{s+1})$), for equation (4.4) is

$$\begin{aligned} e_{s+1} &= e_s - (\hat{K}'_s)^{-1} A_K e_s && (4.6) \\ r_{s+1} &= r_s - A_K (\hat{K}'_s)^{-1} r_s, && (4.7) \end{aligned}$$

[3] When, in the remainder of this study, we write a matrix inverse, we will always tacitly assume that this represents a concise notation for the corresponding linear system to be solved.

while for (4.5) it is

$$\begin{aligned} e_{s+1} &= e_s - \hat{M}_s' A_K e_s && (4.8)\\ r_{s+1} &= r_s - A_K \hat{M}_s' r_s. && (4.9) \end{aligned}$$

For the quasi-Newton methods we are interested in, these approximations are constructed in a way that the *secant equation*[4]

$$\hat{K}_s'(p_s - p_{s-1}) = K(p_s) - K(p_{s-1}), \qquad (4.10)$$

or

$$p_s - p_{s-1} = \hat{M}_s'(K(p_s) - K(p_{s-1})), \qquad (4.11)$$

is respected for all $s = 1, 2, \ldots$. We call methods that respect the secant equation *secant methods*, following the nomenclature of [52]. The origins of these secant equations are to be found in a first order Taylor expansion of K:

$$\begin{aligned} K(p_s) &\approx K(p_{s-1}) + K'(p_s)(p_s - p_{s-1}) && (4.12a)\\ K(p_s) &\approx K(p_{s-1}) + \hat{K}_s'(p_s - p_{s-1}). && (4.12b) \end{aligned}$$

In one dimension, equation (4.12b) uniquely defines $\hat{K}_s'$ (or $\hat{M}_s'$), but in more than one dimension it leaves an infinite choice of approximate Jacobians, as (4.12b) represents an underdetermined system of equations, thus allowing for different methods, some of which we will discuss in the following paragraphs.

Remark 4.1. *While the Richardson, Jacobi, Gauss-Seidel and SOR methods are rarely counted among the class of quasi-Newton methods, (3.2b), (3.3b), (3.5b), (3.7b), (3.9) and (3.10) can be seen to correspond to the quasi-Newton formula (4.4) with an approximate Jacobian* $\hat{K}_s'$ *equal to resp.* $(\tau_s I)^{-1}$, D, $D + L$, $D + U$, $\omega^{-1}(D - \omega L)$ *and* $\omega^{-1}(D - \omega U)$ *when applied to linear systems.*
Similarly, the Iterative Substructuring Method (§1.2.2) and "Fixed-point iteration with dynamic relaxation" (§1.2.3) can be written as a quasi-Newton method with an approximate Jacobian equal to $-I$ *and* $-\omega_s I$ *respectively.*

Remark 4.2. *The study of the convergence behavior of quasi-Newton methods is still a field of ongoing research (e.g. [32]).*

[4] The secant equation is sometimes called *the fundamental equation of quasi-Newton methods.*

4.3 Least change secant updates

Most quasi-Newton methods respect the underlying idea that the approximate Jacobian must not change "too much" from one iteration to another, which translates into an update using a rank-one or rank-two matrix or a matrix with minimal norm, but still satisfying the secant equation.

Definition 4.1. *Least Change Secant Update (LCSU)*
We define a Least Change Secant Update (LCSU) method as a secant method where, of all possible new approximations of the Jacobian taken from a given set, the difference between the new and old approximation is the smallest in some norm, i.e. $\hat{K}'_{s+1}$*, resp.* $\hat{M}'_{s+1}$*, is the solution of*

$$\min\{\|\hat{K}' - \hat{K}'_s\|, \hat{K}' \in \mathcal{Q}_K\},$$

resp.

$$\min\{\|\hat{M}' - \hat{M}'_s\|, \hat{M}' \in \mathcal{Q}_M\},$$

with the choice of matrix norm[5] *and* $\mathcal{Q}$ *to be specified [50].*

The most important properties of LCSU methods can be found in [50–52,140,141]. One of the most important conclusions from these studies is that LCSU algorithms are well-defined, converge to a solution while the rate of convergence is superlinear [28, 140–142].

Remark 4.3. *For ease of reading in the remainder of this chapter, we have chosen not to add a subscript or other indication to the approximate Jacobian referring to the method used. It should be clear from the context which method this approximation refers to.*

4.4 Rank-one update quasi-Newton methods

Rank-one update quasi-Newton methods are characterized by the fact that the difference between $\hat{K}'_{s+1}$ and $\hat{K}'_s$ (or between $\hat{M}'_{s+1}$ and $\hat{M}'_s$) is given by a rank-one matrix (definition 2.9), i.e.

[5] For most known methods this is the Frobenius norm.

$$\exists u, v \in \mathbb{R}^{n\times 1} : \hat{K}'_{s+1} = \hat{K}'_s + uv^T \tag{4.13}$$

or

$$\exists u, v \in \mathbb{R}^{n\times 1} : \hat{M}'_{s+1} = \hat{M}'_s + uv^T. \tag{4.14}$$

(4.13) and (4.14) are called rank-one update formulae. If wanted, (4.13), resp. (4.14), can be transformed to an update for $(\hat{K}'_s)^{-1}$, resp. $(\hat{M}'_s)^{-1}$, by applying the Sherman-Morrison theorem (theorem 2.3.3).

In the majority of existing quasi-Newton methods the rank-one update has a particular form:

$$\hat{K}'_{s+1} = \hat{K}'_s + \frac{(\delta K_s - \hat{K}'_s \delta p_s) c_s^T}{\langle c_s, \delta p_s \rangle}, \tag{4.15}$$

resp.

$$\hat{M}'_{s+1} = \hat{M}'_s + \frac{(\delta p_s - \hat{M}'_s \delta K_s) d_s^T}{\langle d_s, \delta K_s \rangle}, \tag{4.16}$$

where $\delta p_s = p_{s+1} - p_s$ and $\delta K_s = K(p_{s+1}) - K(p_s)$. The use of either (4.15) or (4.16) and the choice of the vector c_s or d_s then defines the particular method. We will discuss some of the best known methods in the remainder of this section.

We will also need the following definition.

Definition 4.2. *Let $V \in \mathbb{R}^{\mu_1 \times \mu_2}, W \in \mathbb{R}^{\mu_3 \times \mu_2}$, $\mu_2 \leq \mu_1$ and V be of rank μ_2. Define the set of "interpolating matrices" between V and W as*
$\mathbb{A}(V, W) = \{A \in \mathbb{R}^{\mu_3 \times \mu_1}; W = AV\}$.

4.4.1 Broyden's first or "good" method

Broyden's first or good method[6] (also abbreviated as "BG") [24, 25, 49, 50] is a quasi-Newton method that uses equations (4.4) and (4.15). It is part of the family

[6] Most often simply called *Broyden's method.*

of LCSU methods [50,84], where the approximate Jacobian $\hat{K}'_{s+1}$ is chosen as the solution of the following minimization problem:

$$\min\{\|\hat{K}' - \hat{K}'_s\|_{Fr}, \hat{K}' \in \mathbb{A}(\delta p_s, \delta K_s)\}. \tag{4.17}$$

In other words, it gives a new approximate Jacobian that is closest to the previous one in the Frobenius norm and that satisfies the secant equation.
The solution of (4.17) leads to the following rank-one update:

$$\begin{aligned} \hat{K}'_{s+1} &= \hat{K}'_s + \frac{(\delta K_s - \hat{K}'_s \delta p_s)\delta p_s^T}{\langle \delta p_s, \delta p_s \rangle} && (4.18) \\ &= \hat{K}'_s + \frac{K(p_{s+1})\delta p_s^T}{\langle \delta p_s, \delta p_s \rangle}. && (4.19) \end{aligned}$$

This means that, using the form of equation (4.15), we have $c_s = \delta p_s$.

The methods starts from an educated guess $\hat{K}'_o$.

The following property is an immediate consequence of (4.17).

Theorem 4.4.1. *Let Q be an arbitrary matrix in $\mathbb{A}(\delta p_s, \delta K_s)$. If $\hat{K}'_{s+1}$ and $\hat{K}'_s$ are defined by Broyden's good update, then*

$$\|\hat{K}'_{s+1} - Q\|_{Fr} \leq \|\hat{K}'_s - Q\|_{Fr}. \tag{4.20}$$

Proof. Since Q lies in the affine subspace $\mathbb{A}(\delta p_s, \delta K_s)$ and since by construction the matrix $\hat{K}'_{s+1}$ is the orthogonal projection of $\hat{K}'_s$ onto this subspace we have

$$\begin{aligned} \|\hat{K}'_s - Q\|^2_{Fr} &= \|\hat{K}'_{s+1} - \hat{K}'_s\|^2_{Fr} + \|\hat{K}'_{s+1} - Q\|^2_{Fr} && (4.21) \\ \|\hat{K}'_s - Q\|^2_{Fr} &\geq \|\hat{K}'_{s+1} - Q\|^2_{Fr}. && (4.22) \end{aligned}$$

□

The above proof can also be found in [49], but as later theorems are based on this theorem we have copied it here for clarity.
In the linear case we have $K(p) = A_K p - b_k$ and hence $K'(p_s) = A_K$ with $A_K \in \mathbb{A}(\delta p_s, \delta K_s), \forall s$. As a consequence of theorem 4.4.1 we then have that the sequence of approximate Jacobians $\{\hat{K}'_s\}_{s\in\mathbb{N}}$ will converge to the true Jacobian A_K in a monotone way.

Interpreting Broyden's good method differently, we could say that

- $\hat{K}'_{s+1}$ is the projection w.r.t. the Frobenius norm of $\hat{K}'_s$ onto $\mathbb{A}(\delta p_s, \delta K_s)$ of matrices that satisfy the secant equation at iteration $s+1$;
- no change occurs between $\hat{K}'_{s+1}$ and $\hat{K}'_s$ on the orthogonal complement of δp_s, i.e. $(\hat{K}'_{s+1} - \hat{K}'_s)z = 0$ if $\langle z, \delta p_s \rangle = 0$.

We also have the following properties of this method:

1. This method can also be directly applied to $(\hat{K}'_s)^{-1}$, using the Sherman-Morrison theorem:

$$(\hat{K}'_{s+1})^{-1} = (\hat{K}'_s)^{-1} - \frac{(\hat{K}'_s)^{-1} K(p_{s+1}) \delta p_s^T (\hat{K}'_s)^{-1}}{\langle \delta p_s, (\hat{K}'_s)^{-1} \delta K_s \rangle} \tag{4.23}$$

   if $\langle \delta p_s, (\hat{K}'_s)^{-1} \delta K_s \rangle \neq 0$.

2. For linear problems, the method is known to show superlinear convergence [127] and it needs at most $2n$ iteration to reach p^* (Gay's theorem [86]).
3. No guarantee can be given that the approximate Jacobians are non-singular.
4. Convergence is not monotone.
5. Broyden's good method does not preserve the semi-positive definite structure of the Jacobian.

4.4.2 Broyden's second or "bad" method

Broyden's second or bad method (also abbreviated as "BB" [24] is a quasi-Newton method that uses equations (4.5) and (4.16). It is also part of the family of LCSU methods [50, 84], where the approximate Jacobian $\hat{M}'_{s+1}$ is chosen as the solution of the following minimization problem:

$$\min\{\|\hat{M}' - \hat{M}'_s\|_{Fr}, \hat{M}' \in \mathbb{A}(\delta K_s, \delta p_s)\}; \tag{4.24}$$

i.e. it gives a new approximation of the inverse of the Jacobian that is closest to the previous one in the Frobenius norm and that satisfies the secant equation.
The solution of (4.24) leads to the following rank- one update

$$\hat{M}'_{s+1} = \hat{M}'_s + \frac{(\delta p_s - \hat{M}'_s \delta K_s) \delta K_s^T}{\langle \delta K_s, \delta K_s \rangle}. \tag{4.25}$$

This means that, using the form of equation (4.16), we have $d_s = \delta K_s$.

The methods starts from an educated guess $\hat{M}'_o$.

The following property is an immediate consequence of (4.24) [49]:

Theorem 4.4.2. *Let Q be an arbitrary matrix in $\mathbb{A}(\delta K_s, \delta p_s)$. If $\hat{M}'_{s+1}$ and $\hat{M}'_s$ are defined by Broyden's bad update, then*

$$\|\hat{M}'_{s+1} - Q\|_{Fr} \leq \|\hat{M}'_s - Q\|_{Fr}. \tag{4.26}$$

The proof of this theorem is analogous to that of theorem 4.4.1. In a similar way we can conclude that in the linear case the sequence $\{\hat{M}'_s\}_{s\in\mathbb{N}}$ will converge to A_K^{-1} in a monotone way.

Interpreting Broyden's bad method differently, we could say that

- $\hat{M}'_{s+1}$ is the projection w.r.t. the Frobenius norm of $\hat{M}'_s$ onto $\mathbb{A}(\delta K_s, \delta p_s)$ of matrices that satisfy the secant equation at iteration $s+1$;

- no change occurs between $\hat{M}'_{s+1}$ and $\hat{M}'_s$ on the orthogonal complement of δK_s, i.e. $(\hat{M}'_{s+1} - \hat{M}'_s)z = 0$ if $\langle z, \delta K_s\rangle = 0$.

Broyden himself [24] admitted that this formulation of his algorithm didn't function properly[7]. The reasons for the "good" or "bad" behavior are not well understood, and it is quite possible that in some instances the bad method outperforms the good method. It is believed, however, that the good method is better whenever the Jacobian $\hat{K}'_s$ of Broyden's good method "underestimates" the true Jacobian (see [145] for more details and other differences).
We also have the following properties of this method:

1. For linear problems, the method is known to show superlinear convergence [127] and it needs at most $2n$ iteration to reach p^* (Gay's theorem [86]).

2. No guarantee can be given that the approximate Jacobians are non-singular.

3. Convergence is not monotone.

4. Broyden's bad method does not preserve the semi-positive definite structure of the Jacobian.

[7] This is the reason the method is called "bad"

4.4.3 Column-Updating Method

The Column-Updating method (CUM) is a quasi-Newton method that was introduced by Martinez [139,144,145]. It uses equations (4.4) and (4.15). The rank-one update of this method is such that the column of the approximate Jacobian corresponding to the largest coordinate of the latest increment ($\delta p_s = p_{s+1} - p_s$) is replaced in order to satisfy the secant equation (4.10) at each iteration.

The resulting method to update the approximate Jacobian is defined as follows:

$$\hat{K}'_{s+1} = \hat{K}'_s + \frac{(\delta K_s - \hat{K}'_s \delta p_s)\imath^T_{j_{K,s}}}{\langle \imath_{j_{K,s}}, \delta p_s \rangle}, \tag{4.27}$$

where $\imath_{j_{K,s}}$ is chosen such that

$$j_{K,s} = \quad \text{Argmax}\{|\langle \imath_j, \delta p_s \rangle|; j = 1, \ldots, n\}. \tag{4.28}$$

($\{\imath_j ; j = 1, \ldots, n\}$ is the canonical (orthonormal) basis for $\mathbb{R}^{n\times 1}$.)
This can be viewed as a rank-one update, where only the $j_{K,s}$-th column of the approximate Jacobian is altered.
It also means that, using the form of equation (4.15), we have $c_s = \imath_{j_{K,s}}$.

The methods starts from an educated guess $\hat{K}'_o$.

The properties of this method have been investigated in [94, 139, 144]. It has to be noted that this method does not belong to the family of the LCSU methods, but it satisfies the hypotheses of Gay's theorem [86] such that finite convergence is reached in at most $2n$ iterations.

Remark 4.4. *This method can also be directly applied to* $(\hat{K}'_s)^{-1}$*, using the Sherman-Morrison theorem if* $\langle \imath_{j_{K,s}}, (\hat{K}'_s)^{-1}\delta K_s \rangle \neq 0$.

4.4.4 Inverse Column-Updating Method

The Inverse Column-Updating method (ICUM) is a quasi-Newton method that was introduced by Martinez and Zambaldi [136, 143]. It uses equations (4.5) and (4.16). The rank-one update of this method is such that the column of the approximation of the inverse of the Jacobian corresponding to the largest coordinate of $\delta K_s = K(p_{s+1}) - K(p_s)$ is replaced in order to satisfy the secant equation (4.11) at each iteration.

The resulting method to update the approximate Jacobian is defined as follows:

$$\hat{M}'_{s+1} = \hat{M}'_s + \frac{(\delta p_s - \hat{M}'_s \delta K_s)\imath^T_{j_{M,s}}}{\langle \imath_{j_{M,s}}, \delta K_s \rangle}, \tag{4.29}$$

where $\imath_{j_{M,s}}$ is chosen such that

$$j_{M,s} = \text{Argmax}\{|\langle \imath_j, \delta K_s \rangle|; j = 1, \ldots, n\}. \tag{4.30}$$

This can be viewed as a rank-one update, where only the $j_{M,s}$-th column of the approximate inverse Jacobian is altered.
It also means that, using the form of equation (4.16), we have $d_s = \imath_{j_{M,s}}$.

The methods starts from an educated guess $\hat{M}'_o$.

The properties of this method have been investigated in [136]. The method does not belong to the family of the LCSU methods, but it satisfies the hypotheses of Gay's theorem [86] such that finite convergence is reached in at most $2n$ iterations.

4.4.5 Symmetric Rank-One update (SR1)

The symmetric rank-one (SR1) update method of Davidon [42] and Murtagh and Sargent [162] uses equations (4.4) and (4.15) with the following update formula:

$$\begin{aligned} \hat{K}'_{s+1} &= \hat{K}'_s + \frac{(\delta K_s - \hat{K}'_s \delta p_s)(\delta K_s - \hat{K}'_s \delta p_s)^T}{\langle \delta K_s - \hat{K}'_s \delta p_s, \delta p_s \rangle} && (4.31) \\ &= \hat{K}'_s + \frac{K(p_{s+1})K(p_{s+1})^T}{\langle K(p_{s+1}), \delta p_s \rangle}. && (4.32) \end{aligned}$$

Thus, using the form of equation (4.15), we have $c_s = \delta K_s - \hat{K}'_s \delta p_s$.

As the rank-one update in (4.32) is symmetric, this method is only to be used when the Jacobian is symmetric too. For that reason it will not be discussed further.

4.4.6 Pearson's Method

Pearson's method [172] uses equations (4.4) and (4.15) with the following update formula:

$$\hat{K}'_{s+1} = \hat{K}'_s + \frac{K(p_{s+1})\delta K_s^T}{\langle \delta K_s, \delta p_s \rangle}. \tag{4.33}$$

Thus, using the form of equation (4.15), we have $c_s = \delta K_s$.

Pearson's method is only valid when $K'(p^*)$ is symmetric positive definite (spd) and needs to start from an initial guess that is spd. For that reason it will not be discussed further.

4.4.7 McCormick's method

McCormick's method [152,172] uses equations (4.5) and (4.16) with the following update formula:

$$\hat{M}'_{s+1} \quad = \quad \hat{M}'_s + \frac{(\delta p_s - \hat{M}'_s \delta K_s)\delta p_s^T}{\langle \delta p_s, \delta K_s \rangle}. \tag{4.34}$$

Thus, using the form of equation (4.16), we have $d_s = \delta p_s$.

Mc Cormick's method is only valid when $K'(p^*)$ is spd and needs to start from an initial guess that is spd. For that reason it will not be discussed further.

4.4.8 The Eirola-Nevanlinna method

In 1989 Eirola and Nevanlinna [63] proposed a quasi-Newton method to solve linear systems[8] of the type of equation (3.1), where the approximation $\hat{M}'_s$ to the inverse of the Jacobian was updated with a rank-one matrix (equation (4.14)).
For this algorithm we need to define the "residual operator" $\mathcal{O}_s^r$ as

$$\mathcal{O}_s^r = I - A_K \hat{M}'_s, \tag{4.35}$$

such that $r_{s+1} = \mathcal{O}_s^r r_s$ (see equation (4.9)).
The argument to arrive at this algorithm starts from the requirement that the residual operator for quasi-Newton methods must respect the following relationship:

[8] As it was proposed as a quasi-Newton method it is included in this chapter and not in the chapter on linear solvers, although no reference to its use on non-linear problems was found so far.

$$\mathcal{O}^r_{s+1} = (I - e_s e_s^T)\mathcal{O}^r_s \tag{4.36}$$

for some unitary vector $e_s \in \mathbb{R}^{n\times 1}$, which can be interpreted as a projection of $\mathcal{O}^r_s$ onto the orthogonal complement of span$\{e_s\}$.
If we write the linear problem to be solved as $A_K p = b_K$ and if we require that $r_{s+1} = \mathcal{O}^r_{s+1} r_s = 0$, then the resulting update would be defined by
$u_s = A_K^{-1}\mathcal{O}^r_s r_s$ and $v_s = \frac{(\mathcal{O}^r_s)^T A_K u_s}{\|A_K u_s\|^2}$.
Unfortunately, A_K^{-1} is not known, and hence $\hat{M}'_s$ is used as a proxy. This results in the following algorithm, in its basic form:

Algorithm 4.4.1. *Eirola-Nevanlinna method [63]*

```
1. Startup. Take a starting value $p_o$ and $\hat{M}'_{-1}$.
   Set $s = 0$.
2. Loop until sufficiently converged:
   a. Compute $r_s = K(p_s) = A_K p_s - b_K$.
   b. $u_s = \hat{M}'_s(I - A_K\hat{M}'_s)r_s$.
   c. $v_s = \frac{(I-(A_K\hat{M}'_s)^T)A_K u_s}{\|A_K u_s\|^2}$.
   e. $\hat{M}'_s = \hat{M}'_{s-1} + u_s v_s^T$.
   e. Quasi-Newton step: $p_{s+1} = p_s - \hat{M}'_s r_s$.
   d. Set $s = s + 1$.
```

Its main properties are, if $\hat{M}'_s$ does not become singular [63]:

- $\mathcal{O}^r_{s+1}$ is the orthogonal projection onto the subspace spanned by $\{A_K u_o, \ldots, A_K u_s\}$;
- $\|\mathcal{O}^r_{s+1}\|_{Fr} < \|\mathcal{O}^r_s\|_{Fr}$;
- the singular values of $\mathcal{O}^r_s$ do not increase;
- the method gives the solution in at most n steps;
- the algorithm is invariant under unitary transformation of coordinates.

The main drawbacks of this method, in its original form are that:

- two multiplications by A_K are needed per iteration, which in our self-imposed conditions are very costly;
- we need to be able to compute $A_K x$ for a given $x \in \mathbb{R}^{n\times 1}$, which is not always immediately available in our framework, as we often assume we can only compute $A_K x - b_K$ for a given $x \in \mathbb{R}^{n\times 1}$ although it might be possible that a way can be found to work around this;

- we must either form $A_K \hat{M}'_s$ and transpose it, which is costly, or be able to compute $A_K^T x$ for a given $x \in \mathbb{R}^{n\times 1}$, which we can't. In [63] an alternative formulation is given that seemingly avoids this requirement, on condition that c_o is given.

As far as is known, the first drawback cannot be avoided, and while numerical tests with the algorithm show that it has indeed very nice convergence properties, when counted as a function of the iteration count, its overall performance is quite poor when measured against the number of function calls (matrix-vector multiplications). For that reason we have not investigated this method beyond some elementary test-cases. Furthermore it can be shown that, when measured against the number of function calls, the residual of the Eirolla-Nevanlinna method cannot be smaller in the Euclidean norm than that of GMRes [231].
It is unknown how this algorithm behaves when used in the context of non-linear problems.
Further developments of this algorithm can be found in [85].

4.5 Rank-two update quasi-Newton methods

Other quasi-Newton methods use rank-two updates, i.e. the difference between two consecutive approximations is a matrix of rank two. The reason for this approach is to preserve the symmetrical structure of the approximate Jacobian. It is thus not surprising that these methods are not meant as non-linear solvers but used to solve minimization problems. We will only touch upon the best-known very briefly; for a good survey of these and other methods we refer to [245, 250].

4.5.1 Powell symmetric Broyden (PSB) method

The PSB method [181] is defined by the following rank-two update:

$$\begin{aligned} \hat{K}'_{s+1} &= \hat{K}'_s + \frac{K(p_{s+1})\delta p_s^T + \delta p_s K(p_{s+1})^T}{\langle \delta p_s, \delta p_s \rangle} \\ &\quad - \frac{(K(p_{s+1})^T \delta p_s)\delta p_s \delta p_s^T}{\langle \delta p_s, \delta p_s \rangle^2}. \end{aligned} \tag{4.37}$$

The PSB method is only valid when $K'(p^*)$ is spd and needs to start from an initial guess that is spd.

4.5.2 Davidon-Fletcher-Powell (DFP) method

The Davidon-Fletcher-Powell method [41,75] is defined by the following rank-two update:

$$\hat{K}'_{s+1} \quad = \quad \hat{K}'_s + \frac{K(p_s)\delta K_s^T + \delta K_s K(p_s)^T}{\langle \delta K_s, \delta p_s \rangle} - \frac{(K(p_s)^T \delta p_s)\delta K_s \delta K_s^T}{\langle \delta K_s, \delta p_s \rangle^2}. \tag{4.38}$$

The DFP method is only valid when $K'(p^*)$ is spd and needs to start from an initial guess that is spd.

4.5.3 Broyden-Fletcher-Goldfarb-Shanno (BFGS) method

The Broyden-Fletcher-Goldfarb-Shanno method [26,76,90,198] is defined by the following rank-two update:

$$\hat{K}'_{s+1} = \hat{K}'_s - \frac{(\hat{K}'_s \delta p_s)\delta p_s^T (\hat{K}'_s)^T}{\langle \delta p_s, \hat{K}'_s \delta p_s \rangle} + \frac{\delta K_s \delta K_s^T}{\langle \delta p_s, \delta K_s \rangle}. \tag{4.39}$$

The BFGS method is only valid when $K'(p^*)$ is spd and needs to start from an initial guess that is spd.

4.5.4 Greenstadt's method

Greenstadt's method [99] is defined by the following rank-two update:

$$\begin{aligned} \hat{M}'_{s+1,G} \quad = \quad & \hat{M}'_{s,G} - \frac{\hat{M}'_{s,G} K(p_{s+1})\delta K_s^T + \delta K_s (\hat{M}'_{s,G} K(p_{s+1}))^T}{\langle \delta K_s, \delta K_s \rangle} \\ & + \frac{(\delta K_s^T \hat{M}'_{s,G} K(p_{s+1}))\delta K_s \delta K_s^T}{\langle \delta K_s, \delta K_s \rangle^2}. \end{aligned} \tag{4.40}$$

4.6 Quasi-Newton methods preserving the structure of a matrix

Some quasi-Newton methods have specifically been developed for problems where the Jacobian has a certain structure and impose this structure on the approximate

Jacobian. For sparse matrices Schubert [27, 147, 196] has developed the Schubert or sparse Broyden algorithm.

Symmetric secant updates for sparse matrices have been developed by Marwil [146], Shanno [199] and Toint [214–216]. An overview of sparse quasi-Newton methods can be found in [132]

We will not go into further detail about these methods.

5

Solution methods for two systems of non-linear equations

As stated in chapter 1 we are interested in solving problems where two systems of non-linear equations interact via their interface (equation (1.1)):

$$F(g) = p \tag{5.1a}$$

$$S(p) = g. \tag{5.1b}$$

One way to do this is to pass to equation (1.3):

$$F(S(p)) - p = H(p) - p = K(p) = 0, \tag{5.2}$$

which transforms (5.1) into a single system of non-linear equations. At that point the solvers of chapter 4 can be put to use. If the single system is obtained in this manner we will indicate it in the name of the non-linear solver: *Interface Newton method*, *Interface quasi-Newton method* (IQN), etc.

As we have already stated that we assume that the true Jacobian is unavailable, we will only focus on quasi-Newton type methods[1].

Apart from methods that work on the single system of non-linear equations of equation (5.2) we will also propose other approaches that take into account the two constituent systems and build an approximate Jacobian for each of them [104].

[1] All the methods proposed in this chapter can be easily transformed to Newton type methods by replacing the approximate Jacobians with their real values.

5.1 Interface quasi-Newton methods (IQN and IQN-I)

The *Interface quasi-Newton* (IQN) method, resp. *Interface quasi-Newton* method with *Inverse Jacobian* (IQN-I), is identical to the ordinary quasi-Newton method of chapter 4 (equation (4.4), resp. (4.5)) applied to (5.2). The only distinction between the algorithms in this section (algorithms 5.1.1 and 5.1.2) and those in chapter 4 is the origin of the equation (5.2).

Algorithm 5.1.1 (IQN)**.**

```
1. Startup:
   a. Take an initial value $p_o$.
   b. Compute $p_1 = (1-\omega)p_o + \omega H(p_o)$.
   c. Set $s = 1$.
2. Loop until sufficiently converged:
   a. Compute $K(p_s)$.
   b. Construct the approximate Jacobian $\hat{K}'_s$.
   c. Quasi-Newton step: $p_{s+1} = p_s - (\hat{K}'_s)^{-1} K(p_s)$.
   d. Set $s = s + 1$.
```

In this algorithm (and the following) ω represents a relaxation parameter, which we apply to avoid excessive initial divergence. (For further discussion, see §8.3.) The actual construction of the approximate Jacobian $\hat{K}'_s$ can be based on those in chapter 4 (see §5.4 below for more details) or given by the Least Squares quasi-Newton method specified in chapter 6.

As already shown in chapter 4 we can also approximate the inverse of the Jacobian. If we choose to do so, we will use the term *Interface quasi-Newton* method with *Inverse Jacobian* (IQN-I) as given in algorithm 5.1.2.

Algorithm 5.1.2 (IQN-I)**.**

```
As algorithm 5.1.1 but
                    2.b. Construct the approximate inverse Jacobian $\hat{M}'_s$.
                    2.c. Quasi-Newton step: $p_{s+1} = p_s - \hat{M}'_s K(p_s)$.
```

Again, the actual construction of the approximate inverse Jacobian $\hat{M}'_s$ can be based on those chosen from chapter 4 or given by the Least Squares quasi-Newton method given in chapter 6.

5.2 Interface quasi-Newton method with Composed Jacobian (IQN-C)

We recall that $K(p) = H(p) - p = F(S(p)) - p$ and that as such

$$K'(p_s) \quad = \quad F'(S(p_s)) \cdot S'(p_s) - I. \tag{5.3}$$

We could therefore replace $F'(S(p_s))$ and $S'(p_s)$ by their own approximate Jacobian and write the approximation to $K'(p_s)$ as

$$\hat{K}'_s \quad = \quad \hat{F}'_s \hat{S}'_s - I. \tag{5.4}$$

The resulting method is called *Interface quasi-Newton* method with *Composed Jacobian* (IQN-C) and can be described as in algorithm 5.2.1.

```
Algorithm 5.2.1 (IQN-C).

1. Startup:
   a. Take an initial value $p_o$.
   b. Compute $g_o = S(p_o)$ and $p_1 = (1-\omega)p_o + \omega F(g_o)$.
   c. Set $s = 1$.
2. Loop until sufficiently converged:
   a. Compute $g_s = S(p_s)$.
   b. Construct the approximate Jacobian $\hat{S}'_s$.
   c. Compute $H(p_s) = F(g_s)$.
   d. Construct the approximate Jacobian $\hat{F}'_s$.
   e. Quasi-Newton step: $p_{s+1} = p_s - (\hat{F}'_s \hat{S}'_s - I)^{-1}(H(p_s) - p_s)$.
   f. Set $s = s + 1$.
```

The actual construction of the approximate Jacobians $\hat{F}'_s$ and $\hat{S}'_s$ will be specified in §5.4 or, for the Least Squares quasi-Newton methods, in chapter 6.

5.3 Interface block quasi-Newton method (IBQN)

In [227] another approach was proposed to couple (5.1) with two approximate Jacobians. In this approach we look at each equation in (5.1) separately and write a block Newton method in which we insert the approximate Jacobians.

The method starts from the block Newton method for the system of equations (5.1), written as

$$\begin{aligned} F(g) - p &= 0 && \text{(5.5a)} \\ S(p) - g &= 0. && \text{(5.5b)} \end{aligned}$$

We define the block Newton method as

$$\begin{bmatrix} F'(g_s) & -I_n \\ -I_m & S'(p_s) \end{bmatrix} \begin{bmatrix} \delta g_s \\ \delta p_s \end{bmatrix} = - \begin{bmatrix} F(g_s) - p_s \\ S(p_s) - g_s \end{bmatrix}, \tag{5.6}$$

where $\delta g_s = g_{s+1} - g_s$ and $\delta p_s = p_{s+1} - p_s$.
Replacing the exact Jacobians by approximate Jacobians, we obtain

$$\begin{bmatrix} \hat{F}'_s & -I_n \\ -I_m & \hat{S}'_s \end{bmatrix} \begin{bmatrix} \delta g_s \\ \delta p_s \end{bmatrix} = - \begin{bmatrix} F(g_s) - p_s \\ S(p_s) - g_s \end{bmatrix}. \tag{5.7}$$

Solving (5.7) for p_{s+1} and g_{s+1} we obtain

$$\begin{aligned} \hat{F}'_s \cdot (g_{s+1} - g_s) &= -F(g_s) + p_{s+1} && \text{(5.8a)} \\ \hat{S}'_s \cdot (p_{s+1} - p_s) &= -S(p_s) + g_{s+1}; && \text{(5.8b)} \end{aligned}$$

Re-arranging both equations gives us

$$\begin{aligned} \hat{F}'_s \cdot g_{s+1} - p_{s+1} &= -F(g_s) + \hat{F}'_s g_s && \text{(5.9a)} \\ \hat{S}'_s \cdot p_{s+1} - g_{s+1} &= -S(p_s) + \hat{S}'_s p_s; && \text{(5.9b)} \end{aligned}$$

Multiplying (5.9a) to the left by $\hat{S}'_s$ and (5.9b) by $\hat{F}'_s$ we obtain

$$\begin{aligned} \hat{S}'_s \hat{F}'_s \cdot g_{s+1} - \hat{S}'_s \cdot p_{s+1} &= \hat{S}'_s \left(-F(g_s) + \hat{F}'_s g_s \right) && \text{(5.10a)} \\ \hat{F}'_s \hat{S}'_s \cdot p_{s+1} - \hat{F}'_s \cdot g_{s+1} &= \hat{F}'_s \left(-S(p_s) + \hat{S}'_s p_s \right). && \text{(5.10b)} \end{aligned}$$

Inserting (5.9a) in (5.10b) and (5.9b) in (5.10a) we get

$$\begin{aligned} \hat{S}'_s \hat{F}'_s \cdot g_{s+1} - g_{s+1} + S(p_s) - \hat{S}'_s p_s &= \hat{S}'_s \left(-F(g_s) + \hat{F}'_s g_s \right) && \text{(5.11a)} \\ \hat{F}'_s \hat{S}'_s \cdot p_{s+1} - p_{s+1} + F(g_s) - \hat{F}'_s g_s &= \hat{F}'_s \left(-S(p_s) + \hat{S}'_s p_s \right); && \text{(5.11b)} \end{aligned}$$

$$\begin{aligned} \left(\hat{S}'_s \hat{F}'_s - I \right) \cdot g_{s+1} &= -S(p_s) + \hat{S}'_s \left(-F(g_s) + \hat{F}'_s g_s + p_s \right) && \text{(5.12a)} \\ \left(\hat{F}'_s \hat{S}'_s - I \right) \cdot p_{s+1} &= -F(g_s) + \hat{F}'_s \left(-S(p_s) + \hat{S}'_s p_s + g_s \right). && \text{(5.12b)} \end{aligned}$$

Re-arranging the terms we finally obtain

$$\begin{aligned} g_{s+1} &= (I - \hat{S}'_s \hat{F}'_s)^{-1} \Big(S(p_s) + \hat{S}'_s \cdot (F(g_s) - \hat{F}'_s \cdot g_s - p_s) \Big) & \text{(5.13a)} \\ p_{s+1} &= (I - \hat{F}'_s \hat{S}'_s)^{-1} \Big(F(g_s) + \hat{F}'_s \cdot (S(p_s) - \hat{S}'_s \cdot p_s - g_s) \Big) . & \text{(5.13b)} \end{aligned}$$

Up till now, we have assumed that we are solving the equations for g_{s+1} and p_{s+1} in parallel (i.e. a block Jacobi approach). We could also solve one of the equations first and afterwards use the available information to update the second equation (i.e. a block Gauss-Seidel approach).
If we solve the equation for p_{s+1} first we obtain

$$\begin{aligned} g_{s+1} &= (I - \hat{S}'_{s+1} \hat{F}'_s)^{-1} \Big(S(p_{s+1}) + \hat{S}'_{s+1} \cdot (F(g_s) - \hat{F}'_s \cdot g_s - p_{s+1}) \Big) \\ p_{s+1} &= (I - \hat{F}'_s \hat{S}'_s)^{-1} \Big(F(g_s) + \hat{F}'_s \cdot (S(p_s) - \hat{S}'_s \cdot p_s - g_s) \Big) . \end{aligned}$$

The resulting method can be found in algorithm 5.3.1 and is called *Interface block quasi-Newton* method (IBQN).

Algorithm 5.3.1 (IBQN)**.**

```
1. Startup:
   a. Take an initial value $p_o$.
   b. Compute $g_o = S(p_o)$ and $p_1 = (1-\omega)p_o + \omega F(g_o)$.
   c. Set $g_1 = S(p_1)$.
   d. Construct the approximate Jacobian $\hat{S}'_1$.
   e. Set $s = 1$.
2. Loop until sufficiently converged:
   a. Compute $F(g_s)$.
   b. Construct the approximate Jacobian $\hat{F}'_s$.
   c. Set $p_{s+1} = (I - \hat{F}'_s \hat{S}'_s)^{-1} \Big( F(g_s) + \hat{F}'_s \cdot (S(p_s) - \hat{S}'_s \cdot p_s - g_s) \Big)$.
   d. Compute $S(p_{s+1})$.
   e. Construct the approximate Jacobian $\hat{S}'_{s+1}$.
   f. Set $g_{s+1} = (I - \hat{S}'_{s+1} \hat{F}'_s)^{-1} \Big( S(p_{s+1}) + \hat{S}'_{s+1} \cdot (F(g_s) - \hat{F}'_s \cdot g_s - p_{s+1}) \Big)$.
   g. Set $s = s + 1$.
```

The actual construction of the approximate Jacobians $\hat{F}'_s$ and $\hat{S}'_s$ will be specified in §5.4 or, for the Least Squares quasi-Newton methods, in chapter 6.

5.4 Construction of the approximate Jacobians for two systems of non-linear equations

For the construction of the approximate Jacobians for S, F, H and/or K we can base ourselves on existing methods as described in section 4.2.
In this section we will only discuss methods based on existing techniques. Another construction, the Least Squares approximate Jacobian, which will form the main topic of our study, will be discussed in chapter 6 and following.

5.4.1 Broyden's good and bad method for the Interface quasi-Newton approach

If the mapping K in (5.2) is derived from two interacting systems as in (5.1) then Broyden's good method (§4.4.1) can be applied straightaway [103, 104], in which case we call the resulting method the *Interface quasi-Newton method with "Broyden good" Jacobian* or IQN-BG (cfr. algorithm 5.1.1).
Similarly, Broydens's bad method (§4.4.2) can be applied, which we then call *Interface quasi-Newton method with "Broyden bad" Jacobian* or IQN-BB (cfr. algorithm 5.1.2).
We point out that Broyden's bad method falls under the label of Interface quasi-Newton methods with Inverse Jacobian (§5.1), so we might as well have used the label IQN-IB for that method[2].
We recall that the only difference with Broyden's good and bad method of §4.4.1 and §4.4.2 is the origin of the equation to be solved.

We propose to extend the idea of Broyden's good method to solution methods using 2 Jacobians, cfr. algorithms 5.2.1 (IQN-C) and 5.3.1 (IBQN), and construct $\hat{S}'_s$ and $\hat{F}'_s$ in a similar way to (4.18):

$$\hat{S}'_{s+1} = \hat{S}'_s + \frac{(\delta S_s - \hat{S}'_s \delta p_s)\delta p_s^T}{\langle \delta p_s, \delta p_s \rangle} \tag{5.15}$$

$$\hat{F}'_{s+1} = \hat{F}'_s + \frac{(\delta F_s - \hat{F}'_s \delta g_s)\delta g_s^T}{\langle \delta g_s, \delta g_s \rangle}, \tag{5.16}$$

where $\delta p_s = p_{s+1} - p_s$, $\delta g_s = g_{s+1} - g_s$, $\delta S_s = S(p_{s+1}) - S(p_s)$, and $\delta F_s = F(g_{s+1}) - F(g_s)$.

The methods start from an educated guess $\hat{S}'_{o,BG}$ and $\hat{F}'_{o,BG}$.

[2] When doing so, the label IQN-B would be more logical for the Broyden good method.

5.4.2 Column-Updating and Inverse Column-Updating method for the Interface quasi-Newton approach

Just as for Broyden's method we can apply the Column-Updating method (§4.4.3) to (5.2) if it is derived from (5.1). In that case we call the resulting method the *Interface quasi-Newton method with "CUM" Jacobian* or IQN-CUM (cfr. algorithm 5.1.1).
Similarly, the Inverse Column-Updating method (§4.4.4) can be applied, which we then call *Interface quasi-Newton method with "Inverse CUM" Jacobian* or IQN-ICUM (cfr. algorithm 5.1.2).
We recall that the only difference with CUM and ICUM of §4.4.3 and §4.4.4 is the origin of the equation to be solved.

As in §5.4.1 we extend the idea of the Column-Updating Method to solution methods using 2 Jacobians, cfr. algorithms 5.2.1 (IQN-C) and 5.3.1 (IBQN), and construct $\hat{S}'_s$ and $\hat{F}'_s$ in a similar way to (4.27):

$$\hat{S}'_{s+1} = \hat{S}'_s + \frac{(\delta S_s - \hat{S}'_s \delta p_s)\imath^T_{j_{S,s}}}{\langle \imath_{j_{S,s}}, \delta p_s \rangle} \tag{5.17}$$

$$\hat{F}'_{s+1} = \hat{F}'_s + \frac{(\delta F_s - \hat{F}'_s \delta g_s)\ell^T_{j_{F,s}}}{\langle \ell_{j_{F,s}}, \delta g_s \rangle}, \tag{5.18}$$

and where $\imath_{j_{S,s}}$, resp. $\ell_{j_{F,s}}$ is chosen such that

$$j_{S,s} = \text{Argmax}\{|\langle \imath_j, \delta p_s \rangle|; j = 1, \ldots, n\} \tag{5.19a}$$

$$j_{F,s} = \text{Argmax}\{|\langle \ell_j, \delta g_s \rangle|; j = 1, \ldots, m\}, \tag{5.19b}$$

where $\{\imath_j; j = 1, \ldots, n\}$ is the canonical (orthonormal) basis for $\mathbb{R}^{n\times 1}$ and $\{\ell_j; j = 1, \ldots, m\}$ is the canonical (orthonormal) basis for $\mathbb{R}^{m\times 1}$.

The methods start from an educated guess $\hat{S}'_{o,CUM}$ and $\hat{F}'_{o,CUM}$.

6

Least Squares Jacobian

6.1 Construction of the Least Squares Jacobian

In this chapter we show how an approximate Jacobian for a given vector-valued function Φ can be constructed using a Least Squares approach based on known input-output pairs.

Let $\Phi : \mathbb{R}^{n\times 1} \longrightarrow \mathbb{R}^{m\times 1} : x \mapsto \Phi(x)$.
Assume that for Φ we have $s+1$ $(s \leq n)$ *input-output pairs* $(x_i, \Phi(x_i))$ $(i = 0, \ldots, s)$[1] at our disposal. This allows us to construct s *input modes* $\Delta x_i^s = x_s - x_i$ $(i = 0, \ldots, s-1)$ and an equal number of *output modes* $\Delta\Phi_i^s = \Phi(x_s) - \Phi(x_i)$ $(i = 0, \ldots, s-1)$. We assume $x_o, x_1, \ldots, x_s$ are in general position (definition 2.3).
We define $V_s^{x\Phi} = [\Delta x_{s-1}^s \ldots \Delta x_o^s]$ and $W_s^{x\Phi} = [\Delta\Phi_{s-1}^s \ldots \Delta\Phi_o^s]$[2].

When we have a new input x_{s+1}, of which the output is not known, but want to make an approximation of $\Phi(x_{s+1})$, we write $x_{s+1} - x_s$ as a linear combination

[1] These input-output pairs can be created in an ad hoc manner or taken from computations that arise in an iterative process.

[2] Strictly speaking $V_s^{x\Phi}$ does not depend on Φ; it is included in the notation, however, for consistency with $W_s^{x\Phi}$ and to emphasize that it will be used in the construction of the approximate Jacobian of Φ.

of the input modes plus a rest-term ε:

$$x_{s+1} - x_s = \sum_{k=0}^{s-1} \alpha_k \Delta x_k^s + \varepsilon, \tag{6.1}$$

where α_k ($k = 0, \ldots, s-1$) represents the coordinates of $x_{s+1} - x_s$ with respect to the input modes and ε the part of $x_{s+1} - x_s$ that lies outside $\mathcal{R}(V_s^{x\Phi})$. If we use $\alpha = [\alpha_{s-1} \ldots \alpha_o]^T$, then $x_{s+1} - x_s = V_s^{x\Phi}\alpha + \varepsilon$.
If we want ε to be minimal in the Euclidean norm[3], we impose $\varepsilon \perp \mathcal{R}(V_s^{x\Phi})$ with respect to the standard scalar product.
This leads to

$$\begin{aligned} (V_s^{x\Phi})^T\varepsilon &= (V_s^{x\Phi})^T((x_{s+1} - x_s) - V_s^{x\Phi}\alpha) = 0 && (6.2)\\ \alpha &= ((V_s^{x\Phi})^T V_s^{x\Phi})^{-1}(V_s^{x\Phi})^T(x_{s+1} - x_s) && (6.3)\\ &= (V_s^{x\Phi})^+(x_{s+1} - x_s), && (6.4) \end{aligned}$$

where $(V_s^{x\Phi})^+$ represents the Moore-Penrose generalized inverse of $V_s^{x\Phi}$, i.e.

$$(V_s^{x\Phi})^+ = ((V_s^{x\Phi})^T V_s^{x\Phi})^{-1}(V_s^{x\Phi})^T. \tag{6.5}$$

We now make a prediction on the output by writing the same linear combination with respect to the output modes:

$$\begin{aligned} \Delta\Phi_s = \Phi(x_{s+1}) - \Phi(x_s) &\approx W_s^{x\Phi}\alpha = W_s^{x\Phi}(V_s^{x\Phi})^+(x_{s+1} - x_s) && (6.6)\\ \Phi(x_{s+1}) &\approx \Phi(x_s) + \underbrace{W_s^{x\Phi}(V_s^{x\Phi})^+}_{\hat{\Phi}'_s}(x_{s+1} - x_s). && (6.7) \end{aligned}$$

We see that the expression $W_s^{x\Phi}(V_s^{x\Phi})^+$ thus fulfills the role of the approximate Jacobian of Φ with respect to x. We will call this approximation the Least Squares (LS) approximate Jacobian $\hat{\Phi}'_s$ of the true Jacobian $\Phi'(x_s)$:

$$\boxed{\begin{aligned} \hat{\Phi}'_s &= W_s^{x\Phi}((V_s^{x\Phi})^T V_s^{x\Phi})^{-1}(V_s^{x\Phi})^T && (6.8a)\\ &= W_s^{x\Phi}(V_s^{x\Phi})^+. && (6.8b) \end{aligned}}$$

If $s = n$ then (6.8) becomes $\hat{\Phi}'_s = W_s^{x\Phi}(V_s^{x\Phi})^{-1}$.

[3] In other words: we want a least squares approximation of $x_{s+1} - x_s$.

To generalize the construction for $s > n$ we use the following formulae:

$$\begin{aligned}
\Delta x_i^s &= x_s - x_i \ \ (i = \tilde{\mathfrak{n}}, \ldots, s-1) && \text{(6.9a)}\\
V_s^{x\Phi} &= [\Delta x_{s-1}^s \ \Delta x_{s-2}^s \ \ldots \ \Delta x_{\tilde{\mathfrak{n}}}^s] \in \mathbb{R}^{n \times \min\{s,n\}} && \text{(6.9b)}\\
\Delta \Phi_i^s &= \Phi(x_s) - \Phi(x_i) \ \ (i = \tilde{\mathfrak{n}}, \ldots, s-1) && \text{(6.9c)}\\
W_s^{x\Phi} &= [\Delta \Phi_{s-1}^s \ \Delta \Phi_{s-2}^s \ \ldots \ \Delta \Phi_{\tilde{\mathfrak{n}}}^s] \in \mathbb{R}^{m \times \min\{s,n\}}, && \text{(6.9d)}
\end{aligned}$$

where $\tilde{\mathfrak{n}} = \max\{0, s-n\}$.

Note that the Jacobian that we construct in this manner (equation (6.8)) is a matrix that "interpolates" between the columns of $V_s^{x\Phi}$ and those of $W_s^{x\Phi}$; such a matrix is not unique as long as the rank of $V_s^{x\Phi}$ is inferior to n.
To formalize this notion for use in later chapters, we use definition 4.2.

Remark 6.1. *The above description is a generalization of a method first described in [227]. Originally this construction was called "Reduced Order Model" (ROM), but as the "reduced" aspect only refers to the use of interface variables and to the low number of input-output modes used, we have changed the name to "Interface quasi-Newton Method with Least Squares Jacobian."*

Remark 6.2. *Another way to interpret this construction is to look at it as an affine approximation to Φ:*

$$\Phi(x) \approx \hat{\Phi}(x) = \Phi(x_s) + \hat{\Phi}'_s(x - x_s), \tag{6.10}$$

such that

$$\Phi(x) = \hat{\Phi}(x) \quad \textit{for } x \in \{x_{\tilde{\mathfrak{n}}},\ x_{\tilde{\mathfrak{n}}+1},\ \ldots\ , x_s\}. \tag{6.11}$$

This approach only leads to a unique value of $\hat{\Phi}'_s$ if $s \geq n$ and if $\{x_{\tilde{\mathfrak{n}}},\ x_{\tilde{\mathfrak{n}}+1},\ \ldots\ , x_s\}$ are in general position.

Remark 6.3. *If Φ is an affine mapping, i.e. $\Phi(x) = Ax - b$, then (6.8) becomes $\hat{\Phi}'_s = AV_s^{x\Phi}(V_s^{x\Phi})^+$, which, according to lemma 2.3.2, corresponds to $\hat{\Phi}'_s = A\mathcal{L}_s^{x\Phi}(\mathcal{L}_s^{x\Phi})^T$ where $\mathcal{L}_s^{x\Phi} = [\bar{L}_1^{x\Phi}|\bar{L}_2^{x\Phi}| \ \ldots \ |\bar{L}_s^{x\Phi}]$, with $\{\bar{L}_k^{x\Phi}\}_{k=1}^s$ an orthonormal basis for the range of $V_s^{x\Phi}$.*

Remark 6.4. *Note that the approximation of $\Delta\Phi(x_{s+1})$ in equation (6.6) not necessarily represents the least squares approximation of $\Delta\Phi(x_{s+1})$ in $\mathcal{R}(W_s^{x\Phi})$. This is easy to illustrate in the case where Φ is an affine mapping: $\Phi(x) = Ax - b$. Let $\Delta\Phi_s = W_s^{x\Phi}(V_s^{x\Phi})^+\Delta x_s + \eta$ (see equation (6.6)).*

If $W_s^{x\Phi}(V_s^{x\Phi})^+\Delta x_s$ were to be a least square approximation to $\Delta\Phi_s$ in $\mathcal{R}(W_s^{x\Phi})$, then we would need that $\eta\perp\mathcal{R}(W_s^{x\Phi})$, or in other words

$$(W_s^{x\Phi})^T \cdot \eta = (W_s^{x\Phi})^T \cdot \left(\Delta\Phi_s - W_s^{x\Phi}(V_s^{x\Phi})^+\Delta x_s\right) = 0. \tag{6.12}$$

As for an affine mapping we have that $W_s^{x\Phi} = AV_s^{x\Phi}$, this leads to

$$(V_s^{x\Phi})^T A^T A \left(I - V_s^{x\Phi}\left[(V_s^{x\Phi})^T V_s^{x\Phi}\right]^{-1}(V_s^{x\Phi})^T\right)\Delta x_s = 0, \tag{6.13}$$

which is only satisfied if either

- *$s = n$;*
- *$\exists\kappa\in\mathbb{R} : A^T A = \kappa I$.*

6.2 Orthogonalization and re-arrangement of input–ouptut modes

As the following theorem will show, we can re-arrange the columns of $V_s^{x\Phi}$ without changing the algorithm. This means we can orthogonalize the columns of $V_s^{x\Phi}$ which improves the condition number of the matrix $(V_s^{x\Phi})^T V_s^{x\Phi}$ that needs to be inverted in the construction of the approximate Jacobian of the Least Squares methods (cfr. equation (6.8).

Theorem 6.2.1. *Consider the approximate Jacobian $\hat{\Phi}'_s$ constructed in equation (6.8). If we replace $V_s^{x\Phi}$ by $V_s^{x\Phi}\mathcal{T}$ and $W_s^{x\Phi}$ by $W_s^{x\Phi}\mathcal{T}$, where $\mathcal{T}$ is a non-singular matrix $\in\mathbb{R}^{s\times s}$, the resulting Jacobian remains the same.*

Proof. Let

$$\hat{\Phi}'_s \quad = \quad W_s^{x\Phi}((V_s^{x\Phi})^T V_s^{x\Phi})^{-1}(V_s^{x\Phi})^T \tag{6.14}$$

and

$$\hat{\Phi}'_{s,\mathcal{T}} \quad = \quad W_s^{x\Phi}\mathcal{T}((V_s^{x\Phi}\mathcal{T})^T V_s^{x\Phi}\mathcal{T})^{-1}(V_s^{x\Phi}\mathcal{T})^T \tag{6.15}$$

then

$$\begin{aligned}
\hat{\Phi}'_{s,\mathcal{T}} &= W_s^{x\Phi}\mathcal{T}(\mathcal{T}^T(V_s^{x\Phi})^T V_s^{x\Phi}\mathcal{T})^{-1}\mathcal{T}^T(V_s^{x\Phi})^T && (6.16)\\
&= W_s^{x\Phi}\mathcal{T}T^{-1}((V_s^{x\Phi})^T V_s^{x\Phi})^{-1}(\mathcal{T}^T)^{-1}\mathcal{T}^T(V_s^{x\Phi})^T && (6.17)\\
&= W_s^{x\Phi}((V_s^{x\Phi})^T V_s^{x\Phi})^{-1}(V_s^{x\Phi})^T && (6.18)\\
&= \hat{\Phi}'_s && (6.19)
\end{aligned}$$

which proves the assertion. □

Not only will this theorem allow us to orthogonalize the columns of $V_s^{x\Phi}$, it also allows the use of any linear combination of the columns as long as the rank is maintained; in other words, as long as $\mathcal{R}(V_s^{x\Phi})$ is not altered. This means, for instance, that we could have used

$$\begin{aligned}
\Delta x_i^s &= x_{i+1} - x_i \;\; (i = \tilde{\mathfrak{n}}, \ldots, s-1) && \text{(6.20a)}\\
V_s^{x\Phi} &= [\Delta x_{\tilde{\mathfrak{n}}}^s \; \Delta x_{\tilde{\mathfrak{n}}+1}^s \; \ldots \; \Delta x_{s-1}^s] \in \mathbb{R}^{n\times\min\{s,n\}} && \text{(6.20b)}\\
\Delta \Phi_i^s &= \Phi(x_{i+1}) - \Phi(x_i) \;\; (i = \tilde{\mathfrak{n}}, \ldots, s-1) && \text{(6.20c)}\\
W_s^{x\Phi} &= [\Delta \Phi_{\tilde{\mathfrak{n}}}^s \; \Delta \Phi_{\tilde{\mathfrak{n}}+1}^s \; \ldots \; \Delta \Phi_{s\tilde{-}1}^s] \in \mathbb{R}^{m\times\min\{s,n\}}, && \text{(6.20d)}
\end{aligned}$$

instead of the conventions in (6.9).
To see this it suffices to multiply $V_s^{x\Phi}$ and $W_s^{x\Phi}$ in (6.9) by

$$\mathcal{T} = \begin{bmatrix} 0 & 0 & \cdots & \cdots & -1 & 1 \\ \vdots & \vdots & \iddots & \iddots & \vdots & \vdots \\ 0 & -1 & 1 & \cdots & 0 & 0 \\ -1 & 1 & 0 & \cdots & 0 & 0 \\ 1 & 0 & 0 & \cdots & 0 & 0 \end{bmatrix}$$

to obtain the expressions in (6.20). (Note that $\mathcal{T}$ is a non-singular matrix.)

If we use (6.20) then we have, for $s \leq n$, that $V_s^{x\Phi} = [V_{s-1}^{x\Phi} \mid \Delta x_{s-1}^s]$ and $W_s^{x\Phi} = [W_{s-1}^{x\Phi} \mid \Delta x_{s-1}^s]$. This property will be useful for analyzing the algorithms.

Remark 6.5. *If $V_s^{x\Phi}$ is given by (6.9) or (6.20) then $\mathcal{R}(V_s^{x\Phi}) = \mathcal{R}(V_s^{x\Phi}\mathcal{T})$ for $s = 0, 1, \ldots,$ if $\mathcal{T}$ is a non-singular matrix $\in \mathbb{R}^{s\times s}$.*
The proof of this property is straightforward.

Theorem 6.2.2. *For $s \leq n$: $\mathbb{A}(V_s^{x\Phi}, W_s^{x\Phi}) \subset \mathbb{A}(V_{s-1}^{x\Phi}, W_{s-1}^{x\Phi})$.*

Proof. As we have seen in theorem 6.2.1 we can re-arrange the columns of $V_s^{x\Phi}$ and $W_s^{x\Phi}$ without changing the construction of the approximate Jacobian. In this proof we will use the form given in (6.20).
$\forall \mathring{A} \in \mathbb{A}(V_{s-1}^{x\Phi}, W_{s-1}^{x\Phi})$ we have that

$$\mathring{A} V_{s-1}^{x\Phi} = W_{s-1}^{x\Phi} \tag{6.21}$$

or, alternatively,

$$\mathring{A} \Delta x_i^s = \Delta \Phi_i^s \quad \text{for } i = \tilde{\mathfrak{n}}, \ldots, s-2. \tag{6.22}$$

$\forall \mathring{A} \in \mathbb{A}(V_s^{x\Phi}, W_s^{x\Phi})$ we have that

$$\mathring{A}\Delta x_i^s \quad = \quad \Delta\Phi_i^s \quad \text{for } i = \tilde{\mathfrak{n}}, \ldots, s-1. \tag{6.23}$$

We can conclude that $\forall \mathring{A} \in \mathbb{A}(V_s^{x\Phi}, W_s^{x\Phi}) : \mathring{A} \in \mathbb{A}(V_{s-1}^{x\Phi}, W_{s-1}^{x\Phi})$, which proves the theorem. □

Theorem 6.2.3. *Consider $V_s^{x\Phi}$ given by (6.9) or (6.20). Then $\mathcal{R}(V_{s-1}^{x\Phi}) \subset \mathcal{R}(V_s^{x\Phi})$ for $s \leq n$.*

Proof. We will prove the theorem for the formulation (6.20).
For that formulation the proof is straightforward as $V_s^{x\Phi} = [V_{s-1}^{x\Phi} \mid \Delta x_{s-1}^s]$.
With remark 6.5 in mind we see that the same holds for the formulation of (6.9). □

Remark 6.6. *Orthogonalization can be performed with any available method (cfr. §3.2.4).*

Remark 6.7. *From the construction of $\hat{\Phi}'_s$ it is obvious that $\hat{\Phi}'_s \in \mathbb{A}(V_s^{x\Phi}, W_s^{x\Phi})$, and if Φ is an affine mapping ($\Phi(x) = Ax - b$), then $A \in \mathbb{A}(V_s^{x\Phi}, W_s^{x\Phi})$, $\forall s \in \mathbb{N}$.*

6.3 Applying the Least Squares approximate Jacobian to quasi-Newton type methods

When solving (1.3), i.e. $K(p) = 0$, we can apply the approximate Jacobian of §6.1 to the quasi-Newton type methods of chapters 4 and 5 as we will show in this section.

6.3.1 IQN-LS

When we apply the Least Squares approximate Jacobian construction of §6.1 to the Interface quasi-Newton (IQN) method of §5.1 (algorithm 5.1.1) we obtain the IQN-LS method.

We first point out that the method described in §6.1 will result in a Jacobian of rank s ($\leq n$), which clearly poses a problem if we want to use it straightaway to

approximate the Jacobian of K in a quasi-Newton method, as we need to invert this matrix (equation (4.4): $p_{s+1} = p_s - (\hat{K}'_s)^{-1} K(p_s)$). We therefore set $\hat{K}'_s = \hat{H}'_s - I$, based on the fact that $K = H - \mathcal{I}$, where we define $\mathcal{I}$ as the identity function $\mathcal{I} : \mathbb{R}^{n\times 1} \to \mathbb{R}^{n\times 1} : x \mapsto \mathcal{I}(x) = x$ and where $\hat{H}'_s$ is the approximate Jacobian for H,
We then construct $\hat{H}'_s$ according to the least squares formulation of §6.1:

$$\hat{H}'_s \quad = \quad W^{pH}_s (V^{pH}_s)^+ \tag{6.24}$$

and thus

$$\hat{K}'_s \quad = \quad W^{pH}_s (V^{pH}_s)^+ - I, \tag{6.25}$$

where

$$\begin{aligned}
\Delta p^s_i &= p_s - p_i \;\; (i = \tilde{\mathfrak{n}}, \dots, s-1) && \text{(6.26a)}\\
V^{pH}_s &= [\Delta p^s_{s-1}\; \Delta p^s_{s-2}\; \dots\; \Delta p^s_{\tilde{\mathfrak{n}}}] \in \mathbb{R}^{n\times \min\{s,n\}} && \text{(6.26b)}\\
\Delta H^s_i &= H(p_s) - H(p_i) \;\; (i = \tilde{\mathfrak{n}}, \dots, s-1) && \text{(6.26c)}\\
W^{pH}_s &= [\Delta H^s_{s-1}\; \Delta H^s_{s-2}\; \dots\; \Delta H^s_{\tilde{\mathfrak{n}}}] \in \mathbb{R}^{n\times \min\{s,n\}}. && \text{(6.26d)}
\end{aligned}$$

In chapter 7 we will show that, if we construct $\hat{K}'_s$ in this manner, it will not become singular for affine mappings before the solution has been reached, which justifies our choice.

Note that (6.24) can only be used for $s \geq 1$, because $\hat{H}'_s$ cannot be constructed earlier. Often $p_1 = H(p_o)$ will be used, which can be seen as setting $\hat{K}'_o = -I$ in equation (4.4) or setting $\hat{H}'_s$ equal to zero, which is as good as any guess, as we don't know anything about the Jacobian a priori.
Alternatively, we could use under-relaxation, as already mentioned in algorithm 5.1.1: $p_1 = (1-\omega)p_o + \omega H(p_o)$ ($\omega \in \mathbb{R}_o$).
We will go into more detail about the use of a relaxation factor in chapter 8 (§8.3).

Remark 6.8. *Note that for an affine mapping $H(p) = A_H p - b_H$ we have*

$$\hat{K}'_s = A_H V^{pH}_s (V^{pH}_s)^+ - I. \tag{6.27}$$

Remark 6.9. *Note that in (6.26) $\Delta p^s_{s-1} = p_s - p_{s-1} = \delta p_{s-1}$. We will use both notations, where δp_{s-1} is used to denote the difference of two consecutive iterates and Δp^s_{s-1} for an input-mode. As we have seen in §6.2, input modes can be defined in various ways; hence the distinction in notation.*

6.3.2 IQN-ILS

When we apply the Least Squares approximate Jacobian construction of §6.1 to the Interface quasi-Newton method with Inverse Jacobian (IQN-I) of §5.1 (algorithm 5.1.2) we obtain the IQN-ILS method.

If we define $M : \mathbb{R}^{n\times 1} \to \mathbb{R}^{n\times 1} : w \mapsto M(w)$ such that $M^{-1} = K$ then we have that $(K'(M(w)))^{-1} = M'(w)$. A logical choice for the Jacobian to be approximated by the least squares approach would be that of M by means of $W_s^{wM}(V_s^{wM})^+$. Unfortunately, just as in §6.3.1, the Jacobian that we obtain in this manner would be singular for $s < n$, which means that in the quasi-Newton step of equation (4.5) we would have that $p_{s+1} - p_s$ lies in the range of this singular Jacobian, which is equal to the range of W_s^{wM}. As $\mathcal{R}(W_s^{wM})$ equals span$\{p_s - p_i\}_{i=0}^{s-1}$ (for $s \leq n$) it follows that all consecutive iterates will be linear combinations of $p_o, p_1, \ldots, p_s$. Unless p^* lies in the subspace spanned by these vectors, there is no hope to achieve convergence.

To avoid singularity of $\hat{M}'_s$ we will use the procedure described below, which is similar to the one in §6.3.1. Just as for IQN-LS we will show in chapter 7 that the approximate inverse Jacobian $\hat{M}'_s$ obtained in this manner will not become singular for affine mappings before the solution has been reached, which justifies our choice.

Let $G(w) = H(K^{-1}(w))$, then

$$\begin{aligned}
G(w) - \mathcal{I}(w) &= H(K^{-1}(w)) - \mathcal{I}(w) && (6.28)\\
&= H(K^{-1}(w)) - K(K^{-1}(w)) && (6.29)\\
&= (H - K)(K^{-1}(w)) && (6.30)\\
&= \mathcal{I}(K^{-1}(w)) && (6.31)\\
&= K^{-1}(w). && (6.32)
\end{aligned}$$

It follows that $(K^{-1})'(w) = G'(w) - I = (K'(K^{-1}(w)))^{-1}$.
Hence, to approximate $(K'(K^{-1}(w)))^{-1}$ we can use the approximation of $G'(w)$, using the same technique described in §6.1. We obtain

$$\begin{aligned}
\Delta w_i^s &= w_s - w_i \ \ (i = \tilde{\mathfrak{n}}, \ldots, s-1) && (6.33a)\\
V_s^{wG} &= [\Delta w_{s-1}^s \ \Delta w_{s-2}^s \ \ldots \ \Delta w_{\tilde{\mathfrak{n}}}^s] \in \mathbb{R}^{n\times \min\{s,n\}} && (6.33b)\\
\Delta G_i^s &= G(w_s) - G(w_i) \ \ (i = \tilde{\mathfrak{n}}, \ldots, s-1) && (6.33c)\\
V_s^{wG} &= [\Delta G_{s-1}^s \ \Delta G_{s-2}^s \ \ldots \ \Delta G_{\tilde{\mathfrak{n}}}^s] \in \mathbb{R}^{n\times \min\{s,n\}}. && (6.33d)
\end{aligned}$$

$$\begin{aligned} \hat{G}'_s &= W_s^{wG}(V_s^{wG})^+ && (6.34)\\ \widehat{(K^{-1})}'_s &= W_s^{wG}(V_s^{wG})^+ - I. && (6.35) \end{aligned}$$

Setting $K^{-1}(w_i) = p_i$ for $i = 0, 1, \ldots$ (i.e. $w_i = K(p_i)$ and $G(w_i) = H(K^{-1}(w_i)) = H(p_i)$) and modifying the notation accordingly we get

$$\widehat{(K^{-1})}'_s = \hat{M}'_s \quad = \quad W_s^{KH}(V_s^{KH})^+ - I, \qquad (6.36)$$

where

$$\begin{aligned} \Delta K_i^s &= K(p_s) - K(p_i) \;\; (i = \tilde{\mathfrak{n}}, \ldots, s-1) && (6.37a)\\ V_s^{KH} &= [\Delta K_{s-1}^s \; \Delta K_{s-2}^s \; \ldots \; \Delta K_{\tilde{\mathfrak{n}}}^s] \in \mathbb{R}^{n \times \min\{s,n\}} && (6.37b)\\ \Delta H_i^s &= H(p_s) - H(p_i) \;\; (i = \tilde{\mathfrak{n}}, \ldots, s-1) && (6.37c)\\ W_s^{KH} &= [\Delta H_{s-1}^s \; \Delta H_{s-2}^s \; \ldots \; \Delta H_{\tilde{\mathfrak{n}}}^s] \in \mathbb{R}^{n \times \min\{s,n\}}. && (6.37d) \end{aligned}$$

Remark 6.10. *Note that for affine mappings $H(p) = A_H p - b_H$, $K(p) = A_K p - b_K$ we have*

$$\begin{aligned} \hat{M}'_s &= A_H(A_K)^{-1}V_s^{KH}(V_s^{KH})^+ - I && (6.38)\\ &= (A_K^{-1} + I)V_s^{KH}(V_s^{KH})^+ - I. && (6.39) \end{aligned}$$

6.3.3 IQN-CLS

When we apply the Least Squares approximate Jacobian construction of §6.1 to the Interface quasi-Newton method with Composed Jacobian (IQN-C) of §5.2 (algorithm 5.2.1) we obtain the IQN-CLS method.
To do this we construct $\hat{F}'_s$ and $\hat{S}'_s$ as follows:

$$\begin{aligned} \hat{F}'_s &= W_s^{gF}(V_s^{gF})^+ && (6.40a)\\ \hat{S}'_s &= W_s^{pS}(V_s^{pS})^+, && (6.40b) \end{aligned}$$

where

$$\begin{aligned}
\Delta g_i^s &= g_s - g_i \ \ (i = \tilde{\mathfrak{n}}, \dots, s-1) && \text{(6.41a)}\\
V_s^{gF} &= [\Delta g_{s-1}^s \ \Delta g_{s-2}^s \ \dots \ \Delta g_{\tilde{\mathfrak{n}}}^s] \in \mathbb{R}^{m \times \min\{s,n\}} && \text{(6.41b)}\\
\Delta F_i^s &= F(g_s) - F(g_i) \ \ (i = \tilde{\mathfrak{n}}, \dots, s-1) && \text{(6.41c)}\\
W_s^{gF} &= [\Delta F_{s-1}^s \ \Delta F_{s-2}^s \ \dots \ \Delta F_{\tilde{\mathfrak{n}}}^s] \in \mathbb{R}^{n \times \min\{s,n\}} && \text{(6.41d)}\\
\Delta p_i^s &= p_s - p_i \ \ (i = \tilde{\mathfrak{n}}, \dots, s-1) && \text{(6.41e)}\\
V_s^{pS} &= [\Delta p_{s-1}^s \ \Delta p_{s-2}^s \ \dots \ \Delta p_{\tilde{\mathfrak{n}}}^s] \in \mathbb{R}^{n \times \min\{s,n\}} && \text{(6.41f)}\\
\Delta S_i^s &= S(p_s) - S(p_i) \ \ (i = \tilde{\mathfrak{n}}, \dots, s-1) && \text{(6.41g)}\\
W_s^{pS} &= [\Delta S_{s-1}^s \ \Delta S_{s-2}^s \ \dots \ \Delta S_{\tilde{\mathfrak{n}}}^s] \in \mathbb{R}^{m \times \min\{s,n\}}. && \text{(6.41h)}
\end{aligned}$$

6.3.4 IBQN-LS

When we apply the Least Squares approximate Jacobian construction of §6.1 to the Interface Block quasi-Newton method (IBQN) of §5.3 (algorithm 5.3.1) we obtain the IBQN-LS method.

$\hat{F}_s'$ and $\hat{S}_s'$ are constructed as in §6.3.3.

7

Properties of IQN-LS, IQN-ILS, IQN-CLS and IBQN-LS for non-linear mappings

In this chapter we will establish some important properties of the four Least Squares algorithms discussed in chapter 6 (IQN-LS, IQN-ILS, IQN-CLS and IBQN-LS), when the mappings K, H, S and F are non-linear.

This chapter is broadly organized as follows.
In §7.1 we re-write the construction of the Least Squares approximate Jacobian in a Rank-One Update form; we discuss the generalized secant property for the Least Squares quasi-Newton methods in §7.2 and the Least Change Secant Update property for IQN-LS and IQN-ILS in §7.3. In §7.4.1 we will show that IQN-LS is algebraically equivalent to IQN-CLS; if F is an affine mapping then both are also equivalent to IBQN-LS, as shown in §7.4.2. Finally, in §7.5, we tackle the problem of possible singularities in the construction of the approximate Jacobian.

In this chapter we will often use the following notation (as used in lemma 2.3.2): $\mathcal{L}_s^{**}$, which represents a matrix containing, in its columns, an orthonormal basis for the range of V_s^{**}. The "wildcard" superscript " ** " will be replaced with the appropriate letters as first used in chapter 6; e.g. for the approximate Jacobian of H this becomes V_s^{pH} and $\mathcal{L}_s^{pH}$, for that of S we write V_s^{gS} and $\mathcal{L}_s^{gS}$, etc.
For the theoretical analysis we will assume that the orthonormal bases are con-

structed such that $\mathcal{L}^{**}_{s+1} = [\mathcal{L}^{**}_s|\bar{L}^{**}_{s+1}]$ ($s < n$) where $\bar{L}^{**}_{s+1}$ is the newly added basis vector for the range of V^{**}_{s+1}. We recall that, according to theorem 6.2.3, $\mathcal{R}(V^{x\Phi}_s) \subset \mathcal{R}(V^{x\Phi}_{s+1})$ for $s < n$.
Note that for this convention we have that

$$\mathcal{L}^{**}_{s+1}(\mathcal{L}^{**}_{s+1})^T = \mathcal{L}^{**}_s(\mathcal{L}^{**}_s)^T + \bar{L}^{**}_{s+1}(\bar{L}^{**}_{s+1})^T. \tag{7.1}$$

Unless otherwise stated we assume that no singularities occur, i.e. all matrices that need to be inverted are of full rank[1]. As we will show later, this assumption can be guaranteed when F, S, H and/or K are affine mappings (§8.1).

7.1 Re-writing IQN-LS and IQN-ILS with a Rank-One Update formula

In this section we will show that the approximate Jacobian of the IQN-LS, resp. IQN-ILS, method ($\hat{K}'_{s+1}$, resp. $\hat{M}'_{s+1}$) can be obtained by applying a rank-one update to $\hat{K}'_s$, resp. $\hat{M}'_s$ (cfr. §4.4) [107].
For IQN-CLS and IBQN-LS a similar procedure is available.
We like to stress that these rank-one formulations do not change the actual algorithms algebraically, although they can influence them numerically and can reduce the computational cost of the actual implementation.

7.1.1 IQN-LS

Theorem 7.1.1. *Suppose that $\hat{K}'_s$ is constructed as in the IQN-LS method (§6.3.1), then $\hat{K}'_{s+1}$ is linked to $\hat{K}'_s$ by the following expression (for $s < n$):*

$$\forall \mathring{A}_H \in \mathbb{A}(V^{pH}_{s+1}, W^{pH}_{s+1}) : \hat{K}'_{s+1} \quad = \quad \hat{K}'_s + \mathring{A}_H \bar{L}^{pH}_{s+1}(\bar{L}^{pH}_{s+1})^T, \tag{7.2}$$

where $\bar{L}^{pH}_{s+1}$ is the $(s+1)$-th column of $\mathcal{L}^{pH}_{s+1}$ (which is a matrix containing in its columns an orthonormal basis for the range of V^{pH}_{s+1})[2].

Proof. $\forall \mathring{A}_H \in \mathbb{A}(V^{pH}_{s+1}, W^{pH}_{s+1}) \subset \mathbb{A}(V^{pH}_s, W^{pH}_s)$ (theorem 6.2.2) we have that $W^{pH}_s = \mathring{A}_H V^{pH}_s$ and $W^{pH}_{s+1} = \mathring{A}_H V^{pH}_{s+1}$. It follows that, according to lemma

[1] We point out that inverses are rarely computed in real applications, but replaced by the computation of the solution of the corresponding linear system.

[2] See definition 4.2 for the meaning of $\mathbb{A}(V^{pH}_{s+1}, W^{pH}_{s+1})$.

2.3.2, we have

$$\begin{aligned}\hat{K}'_{s+1} - \hat{K}'_s &= (\mathring{A}_H \mathcal{L}^{pH}_{s+1}(\mathcal{L}^{pH}_{s+1})^T - I) - (\mathring{A}_H \mathcal{L}^{pH}_s (\mathcal{L}^{pH}_s)^T - I) \quad (7.3)\\ &= \mathring{A}_H(\mathcal{L}^{pH}_{s+1}(\mathcal{L}^{pH}_{s+1})^T - \mathcal{L}^{pH}_s(\mathcal{L}^{pH}_s)^T). \quad (7.4)\end{aligned}$$

Using equation (7.1) we get

$$\hat{K}'_{s+1} - \hat{K}'_s = \mathring{A}_H \bar{L}^{pH}_{s+1}(\bar{L}^{pH}_{s+1})^T, \quad (7.5)$$

which completes our proof. □

Corollary 7.1.1. *Suppose that (for $s < n$) $\hat{K}'_s$ is constructed as in the IQN-LS method (§6.3.1) and that, $\forall \mathring{A}_H \in \mathbb{A}(V^{pH}_{s+1}, W^{pH}_{s+1})$, we have*

$$(\bar{L}^{pH}_{s+1})^T (\hat{K}'_s)^{-1} \mathring{A}_H \bar{L}^{pH}_{s+1} \neq -1, \quad (7.6)$$

then $\hat{K}'_{s+1}$ is non-singular and

$$(\hat{K}'_{s+1})^{-1} = (\hat{K}'_s)^{-1} - \frac{(\hat{K}'_s)^{-1}\mathring{A}_H \bar{L}^{pH}_{s+1}(\bar{L}^{pH}_{s+1})^T(\hat{K}'_s)^{-1}}{1 + (\bar{L}^{pH}_{s+1})^T(\hat{K}'_s)^{-1}\mathring{A}_H\bar{L}^{pH}_{s+1}}, \quad (7.7)$$

where $\bar{L}^{pH}_{s+1}$ is the $(s+1)$-th column of $\mathcal{L}^{pH}_{s+1}$.

Proof. To prove this theorem we apply theorem 2.3.3 (Sherman-Morrison theorem) where we put $Q = \hat{K}'_s, u = \mathring{A}_H \bar{L}_{s+1}, v = \bar{L}_{s+1}$. Then according to theorem 7.1.1 we have $\hat{K}'_{s+1} = Q + uv^T$ and (7.7) follows. □

The results from theorem 7.1.1 and corollary 7.1.1 imply that we can use equation (7.2) to update $\hat{K}'_s$ and equation (7.7) to update $(\hat{K}'_s)^{-1}$, if (7.6) is satisfied.
Note that $\mathring{A}_H \bar{L}^{pH}_{s+1}(\bar{L}^{pH}_{s+1})^T$ is a matrix of rank 1, and that as a consequence (7.2) can be considered as a rank-one update applied to $\hat{K}'_s$ to obtain $\hat{K}'_{s+1}$ (definition 2.9 and equation (4.13)). Based on (7.2) and (7.7) we can now conclude that we are able to form $\hat{K}'_{s+1}$ and $(\hat{K}'_{s+1})^{-1}$ using only matrix-vector and scalar products if we are able to compute $\bar{L}^{pH}_{s+1}$ and $\mathring{A}_H \bar{L}^{pH}_{s+1}$ from the available data.

We show that this is indeed possible and that we do not need the actual knowledge of a matrix $\mathring{A}_H \in \mathbb{A}(V^{pH}_{s+1}, W^{pH}_{s+1})$ to compute the rank-one update.

The new basis vector $\bar{L}^{pH}_{s+1}$ can be computed using

$$\bar{L}^{pH}_{s+1} = \frac{\delta p_s - \mathcal{L}^{pH}_s (\mathcal{L}^{pH}_s)^T \delta p_s}{\|\delta p_s - \mathcal{L}^{pH}_s (\mathcal{L}^{pH}_s)^T \delta p_s\|}, \tag{7.8}$$

which can be done based on the available data ($\delta p_s = p_{s+1} - p_s$), while for the computation of $\mathring{A}_H \bar{L}^{pH}_{s+1}$ we can use the following relationships.

$$\begin{aligned}
\mathring{A}_H \bar{L}^{pH}_{s+1} &= \mathring{A}_H \frac{\delta p_s - \mathcal{L}^{pH}_s (\mathcal{L}^{pH}_s)^T \delta p_s}{\|\delta p_s - \mathcal{L}^{pH}_s (\mathcal{L}^{pH}_s)^T \delta p_s\|} \\
&= \frac{\mathring{A}_H \delta p_s - \mathring{A}_H \mathcal{L}^{pH}_s (\mathcal{L}^{pH}_s)^T \delta p_s}{\|\delta p_s - \mathcal{L}^{pH}_s (\mathcal{L}^{pH}_s)^T \delta p_s\|} \\
&= \frac{\delta H_s - \mathring{A}_H V^{pH}_s (V^{pH}_s)^T \delta p_s}{\|\delta p_s - \mathcal{L}^{pH}_s (\mathcal{L}^{pH}_s)^T \delta p_s\|} \\
&= \frac{\delta H_s - \mathring{A}_H V^{pH}_s (V^{pH}_s)^T \delta p_s - \delta p_s + \delta p_s}{\|\delta p_s - \mathcal{L}^{pH}_s (\mathcal{L}^{pH}_s)^T \delta p_s\|} \\
&= \frac{\delta H_s - \delta p_s - (\mathring{A}_H V^{pH}_s (V^{pH}_s)^T - I) \delta p_s}{\|\delta p_s - \mathcal{L}^{pH}_s (\mathcal{L}^{pH}_s)^T \delta p_s\|} \\
&= \frac{\delta K_s - \hat{K}'_s \delta p_s}{\|\delta p_s - \mathcal{L}^{pH}_s (\mathcal{L}^{pH}_s)^T \delta p_s\|} \\
&= \frac{K(p_{s+1}) - K(p_s) - \hat{K}'_s \delta p_s}{\|\delta p_s - \mathcal{L}^{pH}_s (\mathcal{L}^{pH}_s)^T \delta p_s\|},
\end{aligned} \tag{7.9}$$

where $\delta K_s = K(p_{s+1}) - K(p_s)$. Note that all the terms in (7.9) are available from the algorithm.

We can write the resulting update (for $s < n$) as

$$\hat{K}'_{s+1} = \hat{K}'_s + \frac{(\delta K_s - \hat{K}'_s \delta p_s)((I - \mathcal{L}^{pH}_s (\mathcal{L}^{pH}_s)^T) \delta p_s)^T}{\langle (I - \mathcal{L}^{pH}_s (\mathcal{L}^{pH}_s)^T) \delta p_s, (I - \mathcal{L}^{pH}_s (\mathcal{L}^{pH}_s)^T) \delta p_s \rangle}. \tag{7.10}$$

As $(I - \mathcal{L}^{pH}_s (\mathcal{L}^{pH}_s)^T)^T (I - \mathcal{L}^{pH}_s (\mathcal{L}^{pH}_s)^T) = I - \mathcal{L}^{pH}_s (\mathcal{L}^{pH}_s)^T$ this simplifies to

$$\hat{K}'_{s+1} \quad = \quad \hat{K}'_s + \frac{(\delta K_s - \hat{K}'_s \delta p_s)((I - \mathcal{L}^{pH}_s (\mathcal{L}^{pH}_s)^T) \delta p_s)^T}{\langle \delta p_s, (I - \mathcal{L}^{pH}_s (\mathcal{L}^{pH}_s)^T) \delta p_s \rangle}. \tag{7.11}$$

Using the fact that $p_{+1} = p_s - (\hat{K}'_s)^{-1} K(p_s)$ we can write equation (7.11) as

$$\hat{K}'_{s+1} \quad = \quad \hat{K}'_s + \frac{K(p_{s+1})((I - \mathcal{L}^{pH}_s (\mathcal{L}^{pH}_s)^T) \delta p_s)^T}{\langle \delta p_s, (I - \mathcal{L}^{pH}_s (\mathcal{L}^{pH}_s)^T) \delta p_s \rangle}. \tag{7.12}$$

Remark 7.1. *We will go into more details about possible singularities in* §*7.5.*

Remark 7.2. *If H is an affine mapping ($H(p) = A_H p - b_H$) then an obvious choice for $\mathring{A}_H$ is A_H.*

For the update from $\hat{K}'_n$ to $\hat{K}'_{n+1}$ matters are slightly more involved.
For the derivation we will use the formulation given in (6.20), i.e. $\hat{K}'_n$ is formed with V_n^{pH} and W_n^{pH} as follows.

$$\begin{aligned} V_n^{pH} &= [p_1 - p_o|\, p_2 - p_1|\, \ldots\, |p_n - p_{n-1}] \in \mathbb{R}^{n\times n} \\ W_n^{pH} &= [H(p_1) - H(p_o)|\, H(p_2) - H(p_1)|\, \ldots\, |H(p_n) - H(p_{n-1})] \in \mathbb{R}^{m\times n}; \end{aligned}$$

$$\hat{K}'_n = W_n^{pH}(V_n^{pH})^+ - I.$$

$\hat{K}'_{n+1}$ is formed with V_{n+1}^{pH} and W_{n+1}^{pH} as follows

$$\begin{aligned} V_{n+1}^{pH} &= [p_2 - p_1|\, p_3 - p_2|\, \ldots\, |p_{n+1} - p_n] \in \mathbb{R}^{n\times n} \\ W_{n+1}^{pH} &= [H(p_2) - H(p_1)|\, H(p_3) - H(p_2)|\, \ldots\, |H(p_{n+1}) - H(p_n)] \in \mathbb{R}^{m\times n}; \end{aligned}$$

$$\hat{K}'_{n+1} = W_{n+1}^{pH}(V_{n+1}^{pH})^+ - I.$$

Additionally, we introduce the following matrices:

$$\begin{aligned} \tilde{V} &= [p_2 - p_1|\, p_3 - p_2|\, \ldots\, |p_n - p_{n-1}] \in \mathbb{R}^{n\times (n-1)} \\ \tilde{W} &= [H(p_2) - H(p_1)|\, H(p_3) - H(p_2)|\, \ldots\, |H(p_n) - H(p_{n-1})] \in \mathbb{R}^{m\times (n-1)}; \end{aligned}$$

$$\tilde{K}' = \tilde{W}(\tilde{V})^+ - I.$$

(We have chosen to drop most superscripts from the newly introduced entities in this section, as we believe this does not hamper understanding; these will only appear in this section.)

It is clear from these conventions that

$$V_n^{pH} = [p_1 - p_o|\tilde{V}] \tag{7.16}$$
$$V_{n+1}^{pH} = [\tilde{V}|p_{n+1} - p_n]. \tag{7.17}$$

As such, using theorem 6.2.3, we see that $\mathcal{R}(\tilde{V}) \subset \mathcal{R}(V_n^{pH})$ and $\mathcal{R}(\tilde{V}) \subset \mathcal{R}(V_{n+1}^{pH})$. We will be using $\mathcal{L}_n^{pH}$, resp. $\mathcal{L}_{n+1}^{pH}$, $\tilde{\mathcal{L}}$, for the matrix containing in its columns an orthonormal basis for $\mathcal{R}(V_n^{pH})$, resp. $\mathcal{R}(V_{n+1}^{pH})$, $\mathcal{R}(\tilde{V})$. Contrary to previous conventions we will construct the bases such that

$$\begin{aligned}\mathcal{L}_n^{pH} &= [\bar{L}_1^{pH}|\tilde{\mathcal{L}}] && (7.18)\\ \mathcal{L}_{n+1}^{pH} &= [\tilde{\mathcal{L}}|\bar{L}_{n+1}^{pH}]. && (7.19)\end{aligned}$$

By analogy with the results of theorem 7.1.1 we arrive at

$$\begin{aligned}\hat{K}_n' &= \tilde{K}' + \mathring{A}_{H,n}\bar{L}_1^{pH}(\bar{L}_1^{pH})^T && (7.20)\\ \hat{K}_{n+1}' &= \tilde{K}' + \mathring{A}_{H,n+1}\bar{L}_{n+1}^{pH}\bar{(}L_{n+1}^{pH})^T, && (7.21)\end{aligned}$$

where $\bar{L}_1^{pH} = (I - \tilde{\mathcal{L}}\tilde{\mathcal{L}}^T)\delta p_o$, $\bar{L}_{n+1}^{pH} = (I - \tilde{\mathcal{L}}\tilde{\mathcal{L}}^T)\delta p_n$. $\mathring{A}_{H,n}$ is any matrix in $\mathbb{A}(V_n^{pH}, W_n^{pH})$ and $\mathring{A}_{H,n+1}$ is any matrix in $\mathbb{A}(V_{n+1}^{pH}, W_{n+1}^{pH})$.
From this we see that

$$\hat{K}_{n+1}' - \hat{K}_n' \quad = \quad \mathring{A}_{H,n+1}\bar{L}_{n+1}^{pH}(\bar{L}_{n+1}^{pH})^T - \mathring{A}_{H,n}\bar{L}_1^{pH}(\bar{L}_1^{pH})^T. \qquad (7.22)$$

Pointing out that $\bar{L}_1^{pH} = \bar{L}_{n+1}^{pH}$ because $(\mathcal{R}(\tilde{V}))^\perp$ is only a subspace of dimension 1, we obtain

$$\hat{K}_{n+1}' - \hat{K}_n' \quad = \quad (\mathring{A}_{H,n+1} - \mathring{A}_{H,n})\bar{L}_1^{pH}(\bar{L}_1^{pH})^T, \qquad (7.23)$$

which represents a rank-one update. In general $\mathring{A}_{H,n} \neq \mathring{A}_{H,n+1}$, unless H is an affine mapping. (In that case it is clear that the update would be the zero matrix.) We will show in chapter 8 (§8.1), that this is a logical consequence for affine mappings, as in that instance convergence will have been reached.

By analogy with (7.11), and using

$$\begin{aligned}\hat{K}_n' &= \tilde{K}' + \frac{(\delta K_o - \tilde{K}'\delta p_o)((I - \tilde{\mathcal{L}}\tilde{\mathcal{L}}^T)\delta p_o)^T}{\langle \delta p_o, (I - \tilde{\mathcal{L}}\tilde{\mathcal{L}}^T)\delta p_o\rangle} && (7.24)\\ \hat{K}_{n+1}' &= \tilde{K}' + \frac{(\delta K_n - \tilde{K}'\delta p_n)((I - \tilde{\mathcal{L}}\tilde{\mathcal{L}}^T)\delta p_n)^T}{\langle \delta p_n, (I - \tilde{\mathcal{L}}\tilde{\mathcal{L}}^T)\delta p_n\rangle}, && (7.25)\end{aligned}$$

(7.23) can be re-written as

$$\begin{aligned}\hat{K}_{n+1}' = \hat{K}_n' \quad &+ \quad \frac{(\delta K_n - \tilde{K}'\delta p_n)((I - \tilde{\mathcal{L}}\tilde{\mathcal{L}}^T)\delta p_n)^T}{\langle \delta p_n, (I - \tilde{\mathcal{L}}\tilde{\mathcal{L}}^T)\delta p_n\rangle}\\ &- \quad \frac{(\delta K_o - \tilde{K}'\delta p_o)((I - \tilde{\mathcal{L}}\tilde{\mathcal{L}}^T)\delta p_o)^T}{\langle \delta p_o, (I - \tilde{\mathcal{L}}\tilde{\mathcal{L}}^T)\delta p_o\rangle}, && (7.26)\end{aligned}$$

which equals

$$\hat{K}'_{n+1} \quad = \quad \hat{K}'_n + \Big((\delta K_n - \tilde{K}'\delta p_n) - (\delta K_o - \tilde{K}'\delta p_o)\Big)\frac{((I-\tilde{\mathcal{L}}\tilde{\mathcal{L}}^T)\delta p_n)^T}{\langle \delta p_n, (I-\tilde{\mathcal{L}}\tilde{\mathcal{L}}^T)\delta p_n\rangle}. \tag{7.27}$$

Unfortunately (7.27) cannot be computed because $\tilde{K}$ is never explicitly formed. However, we can show that (7.27) can be re-arranged to a form that is very similar to (7.12):

$$\hat{K}'_{n+1} = \hat{K}'_n + \frac{(\delta K_n - \hat{K}'_n\delta p_n)((I-\tilde{\mathcal{L}}\tilde{\mathcal{L}}^T)\delta p_n)^T}{\langle \delta p_n, (I-\tilde{\mathcal{L}}\tilde{\mathcal{L}}^T)\delta p_n\rangle}. \tag{7.28}$$

To show this equality we replace $\hat{K}'_n$ in (7.28) by (7.24). Re-arranging the terms then yields (7.27).

7.1.2 IQN-ILS

Theorem 7.1.2. *Suppose that $\hat{M}'_s$ is constructed as in the IQN-ILS method (§6.3.2), then $\hat{M}'_{s+1}$ is linked to $\hat{M}'_s$ by the following expression (for $s < n$):*

$$\forall \mathring{A}_M \in \mathbb{A}(V^{KH}_{s+1}, W^{KH}_{s+1}) : \hat{M}'_{s+1} \quad = \quad \hat{M}'_s + \mathring{A}_M \bar{L}^{KH}_{s+1}(\bar{L}^{KH}_{s+1})^T, \tag{7.29}$$

where $\bar{L}^{KH}_{s+1}$ is the $(s+1)$-th column of $\mathcal{L}^{KH}_{s+1}$.

Proof. $\forall \mathring{A}_M \in \mathbb{A}(V^{KH}_{s+1}, W^{KH}_{s+1}) \subset \mathbb{A}(V^{KH}_s, W^{KH}_s)$ (theorem 6.2.2) we have that $W^{KH}_s = \mathring{A}_M V^{KH}_s$ and $W^{KH}_{s+1} = \mathring{A}_M V^{KH}_{s+1}$. It follows that, according to lemma 2.3.2, we have

$$\hat{M}'_{s+1} - \hat{M}'_s \quad = \quad \big(\mathring{A}_M \mathcal{L}^{KH}_{s+1}(\mathcal{L}^{KH}_{s+1})^T - I\big) - \big(\mathring{A}_M \mathcal{L}^{KH}_s(\mathcal{L}^{KH}_s)^T - I\big) \tag{7.30}$$

$$= \quad \mathring{A}_M(\mathcal{L}^{KH}_{s+1}(\mathcal{L}^{KH}_{s+1})^T - \mathcal{L}^{KH}_s(\mathcal{L}^{KH}_s)^T). \tag{7.31}$$

Using equation (7.1) we get

$$\hat{M}'_{s+1} - \hat{M}'_s \quad = \quad \mathring{A}_M \bar{L}^{KH}_{s+1}(\bar{L}^{KH}_{s+1})^T, \tag{7.32}$$

which completes our proof. □

Theorem 7.1.2 implies that we can use (7.29) to update $\hat{M}'_s$. Note that $\mathring{A}_M \bar{L}^{KH}_{s+1} (\bar{L}^{KH}_{s+1})^T$ is a matrix of rank 1, and that as a consequence (7.29) can be considered as a rank-one update applied to $\hat{M}'_s$ to obtain $\hat{M}'_{s+1}$ (definition 2.9 and equation (4.14)).
Just as with the matrix $\hat{K}'_s$ of the IQN-LS method we have to find a convenient way to compute $\bar{L}^{KH}_{s+1}$ and $\mathring{A}_M \bar{L}^{KH}_{s+1}$ from the available data.
The new basis vector $\bar{L}^{KH}_{s+1}$ can be computed using

$$\bar{L}^{KH}_{s+1} = \frac{(I - \mathcal{L}^{KH}_s (\mathcal{L}^{KH}_s)^T)\delta K_s}{\|(I - \mathcal{L}^{KH}_s (\mathcal{L}^{KH}_s)^T)\delta K_s\|}, \tag{7.33}$$

while $\mathring{A}_M \bar{L}^{KH}_{s+1}$ can be computed, in similar fashion to §7.1.1, as

$$\mathring{A}_M \bar{L}^{KH}_{s+1} = \frac{\delta p_s - \hat{M}'_s \delta K_s}{\|(I - \mathcal{L}^{KH}_s (\mathcal{L}^{KH}_s)^T)\delta K_s\|}. \tag{7.34}$$

As a result, we can write the rank-one update as

$$\begin{aligned} \hat{M}'_{s+1} &= \hat{M}'_s + \frac{(\delta p_s - \hat{M}'_s \delta K_s)\left((I - \mathcal{L}^{KH}_s (\mathcal{L}^{KH}_s)^T)\delta K_s\right)^T}{\langle \delta K_s, (I - \mathcal{L}^{KH}_s (\mathcal{L}^{KH}_s)^T)\delta K_s \rangle} && (7.35) \\ &= \hat{M}'_s - \frac{\hat{M}'_s K(p_{s+1})\left((I - \mathcal{L}^{KH}_s (\mathcal{L}^{KH}_s)^T)\delta K_s\right)^T}{\langle \delta K_s, (I - \mathcal{L}^{KH}_s (\mathcal{L}^{KH}_s)^T)\delta K_s \rangle}. && (7.36) \end{aligned}$$

Remark 7.3. *We will go into more details about possible singularities in §7.5.*

Remark 7.4. *If H is an affine mapping ($H(p) = A_H p - b_H$) then an obvious choice for $\mathring{A}_M$ is $A_H(A_H - I)^{-1} = A_K^{-1} + I$. (This follows from the fact that $H\delta p_i = A_H \delta p_i = A_H A_K^{-1} \delta K_i = A_H (A_H - I)^{-1} \delta K_i$, for $i = 1, \ldots, s$.)*

Remark 7.5. *A corollary similar to corollary 7.1.1 can be written, even though no actual need exists to know the inverse of $\hat{M}'_{s+1}$.*

Remark 7.6. *Similar results as for IQN-LS are found for the update from $\hat{M}'_n$ to $\hat{M}'_{n+1}$*

7.1.3 IQN-CLS and IBQN-LS

In the IQN-CLS and IBQN-LS method we use two approximate Jacobians, which each can be obtained with a rank-one update formula, as given below.

$$\hat{S}'_{s+1} = \hat{S}'_s + \frac{(\delta S_s - \hat{S}'_s \delta p_s)((I - \mathcal{L}_s^{pS}(\mathcal{L}_s^{pS})^T)\delta p_s)^T}{\langle \delta p_s, (I - \mathcal{L}_s^{pS}(\mathcal{L}_s^{pS})^T)\delta p_s \rangle} \qquad (7.37)$$

$$\hat{F}'_{s+1} = \hat{F}'_s + \frac{(\delta F_s - \hat{F}'_s \delta g_s)((I - \mathcal{L}_s^{gF}(\mathcal{L}_s^{gF})^T)\delta g_s)^T}{\langle \delta g_s, (I - \mathcal{L}_s^{gF}(\mathcal{L}_s^{gF})^T)\delta g_s \rangle}, \qquad (7.38)$$

where $\delta S_s = S(p_{s+1}) - S(p_s)$, $\delta F_s = F(g_{s+1}) - F(g_s)$.

We note that we will actually need the inverse of $\hat{F}'_{s+1}\hat{S}'_{s+1} - I$ for the IQN-CLS method and the inverses of $I - \hat{F}'_{s+1}\hat{S}'_{s+1}$ and $I - \hat{S}'_{s+1}\hat{F}'_s$ for the IBQN-LS method (algorithms 5.2.1 and 5.3.1 respectively). In order to be able to apply the Sherman-Morrison theorem to these expressions, we would like to write these as a rank-one update with respect to the previously known values.

For IQN-CLS the updates to be computed are the following:

$$\hat{F}'_s\hat{S}'_s - I \quad \rightarrow \quad \hat{F}'_{s+1}\hat{S}'_{s+1} - I.$$

If we use $\Delta\hat{F}$, resp. $\Delta\hat{S}$ for the respective rank-one updates of $\hat{F}'_s$ and $\hat{S}'_s$ (equations (7.37) and (7.38)) we see that

$$\begin{aligned} \hat{F}'_{s+1}\hat{S}'_{s+1} - I &= (\hat{F}'_s + \Delta\hat{F})(\hat{S}'_s + \Delta\hat{S}) - I \\ &= \hat{F}'_s\hat{S}'_s - I + \underbrace{\hat{F}'_s\Delta\hat{S} + \Delta\hat{F}\hat{S}'_s + \Delta\hat{F}\Delta\hat{S}}_{(*)}. \end{aligned}$$

The part marked with $(*)$ is thus the update of the whole expression. The three terms of $(*)$ are each rank-one matrices. In general, this does not imply that their sum, i.e $(*)$, is a rank-one matrix. This means that we cannot use the Sherman-Morrison theorem straight away, but would need to use the theorem three times, once for each rank-one matrix.

However, as we will show in §7.4, IQN-LS and IQN-CLS are algebraically identical, and thus

$(*) = (\hat{F}'_{s+1}\hat{S}'_{s+1} - I) - (\hat{F}'_s\hat{S}'_s - I) = \hat{F}'_s\Delta\hat{S} + \Delta\hat{F}\hat{S}'_s + \Delta\hat{F}\Delta\hat{S}$

is indeed a rank-one matrix.

We point out that this conclusion is generally not applicable to the Jacobians of the other quasi-Newton methods that were summarized in chapter 4; see §7.4 for more details.

For IBQN-LS the updates to be computed are the following:

$$\begin{aligned} I - \hat{S}'_s\hat{F}'_s \quad &\rightarrow \quad I - \hat{S}'_{s+1}\hat{F}'_s \\ I - \hat{F}'_s\hat{S}'_{s+1} \quad &\rightarrow \quad I - \hat{F}'_{s+1}\hat{S}'_{s+1}. \end{aligned}$$

Using the same approach as for the IQN-CLS method we obtain:

$$\begin{aligned} I - \hat{S}'_{s+1}\hat{F}'_s &= I - \hat{S}'_s\hat{F}'_s \underbrace{-\Delta\hat{S}\hat{F}'_s}_{(**)} \\ I - \hat{F}'_{s+1}\hat{S}'_{s+1} &= I - \hat{F}'_s\hat{S}'_{s+1} \underbrace{-\Delta\hat{F}\hat{S}'_s}_{(***)}, \end{aligned}$$

where the parts marked with $(**)$ and $(***)$ are rank-one matrices. As such, the Sherman-Morrison theorem can be applied if required.

Remark 7.7. *Similar results as for IQN-LS are found for the update from $\hat{S}'_n$ and $\hat{F}'_n$ to $\hat{S}'_{n+1}$ and $\hat{F}'_{n+1}$.*

7.1.4 Conclusions

From the results in this section, we see that IQN-LS corresponds to the rank-one update form of equation (4.15), i.e.

$$\hat{K}'_{s+1} = \hat{K}'_s + \frac{(\delta K_s - \hat{K}'_s\delta p_s)c_s^T}{\langle c_s, \delta p_s\rangle},$$

with $c_s = (I - \mathcal{L}_s^{pH}(\mathcal{L}_s^{pH})^T)\delta p_s$, while IQN-ILS corresponds to the form of equation (4.16), i.e.

$$\hat{M}'_{s+1} = \hat{M}'_s + \frac{(\delta p_s - \hat{M}'_s\delta K_s)d_s^T}{\langle d_s, \delta K_s\rangle},$$

with $d_s = (I - \mathcal{L}_s^{KH}(\mathcal{L}_s^{KH})^T)\delta K_s$.
Furthermore, we see that the vector v in equation (4.13), resp. (4.14), respects the condition that $\langle v, \delta p_{s-1}\rangle = 0$, resp. $\langle v, \delta K_{s-1}\rangle = 0$, for IQN-LS, resp. IQN-ILS. Convergence properties of rank-one update secant methods that respect this condition have been well-studied [145].

We also see, from (7.12), that we have that

$$\forall z \perp (I - \mathcal{L}_s^{pH}(\mathcal{L}_s^{pH})^T)\delta p_s \quad : \quad \hat{K}'_{s+1}z = \hat{K}'_s z; \tag{7.39}$$

i.e. the approximate Jacobian only changes in the direction of the newly added orthogonal basis-vector $(I - \mathcal{L}_s^{pH}(\mathcal{L}_s^{pH})^T)\delta p_s$ of $\mathcal{R}(V_{s+1}^{pH})$.
For every direction $z \perp \mathcal{R}(V_s^{pH})$ we have $\hat{K}'_s z = -z$. The latter can be interpreted as setting the approximate Jacobian of H equal to zero in every direction of which

no information is available.

Similarly, for IQN-ILS we see, from equation (7.36), that we have that

$$\forall z \perp (I - \mathcal{L}_s^{KH}(\mathcal{L}_s^{KH})^T)\delta K_s \quad : \quad \hat{M}'_{s+1}z = \hat{M}'_s z; \tag{7.40}$$

i.e. the approximate Jacobian only changes in the direction of the newly added orthogonal basis-vector $(I - \mathcal{L}_s^{KH}(\mathcal{L}_s^{KH})^T)\delta K_s$ of $\mathcal{R}(V_{s+1}^{KH})$.
For every direction $z \perp \mathcal{R}(V_s^{KH})$ we have $\hat{M}'_s z = -z$. The latter can again be interpreted as setting the approximate Jacobian of H equal to zero in every direction of which no information is available.

These results show that IQN-LS and IQN-ILS possess a number of similarities with resp. Broyden's good and Broyden's bad method (§4.4.1 and §4.4.2).
For Broyden's good method we have

$$\forall z \perp \delta p_s \quad : \quad \hat{K}'_{s+1}z = \hat{K}'_s z. \tag{7.41}$$

This closely links IQN-LS with Broyden's good method. The difference being that Broyden does not change the Jacobian in the orthogonal complement of the last direction δp_s, while we do not change the Jacobian in the orthogonal complement of the new direction δp_s after orthogonalizing it w.r.t. all previous directions. This means that we keep our Jacobian the same for all previously visited directions, while Broyden doesn't. This can be both an advantage and a disadvantage. The advantage being that we keep information about more directions intact. On the other hand, we are "locked-in" by our previous directions, which may no longer be accurate, while Broyden does not have this restriction.
One of the most important advantages of our method over Broyden's is the fact that in exact arithmetic IQN-LS converges in $n+1$ iterations for affine mappings, while Broyden's good method only converge in $2n$ iterations. (See §8.1 for more details.) It is believed that this advantage is carried over to weakly non-linear problems.

For Broyden's bad method we have

$$\forall z \perp \delta K_s \quad : \quad \hat{M}'_{s+1}z = \hat{M}'_s z. \tag{7.42}$$

The relationship between Broyden's bad method and IQN-ILS is analogous to the one between Broyden's good method and IQN-LS.
Similar conclusions hold for IQN-CLS versus IQN-CBG and IBQN-LS and IBQN-BG.

7.2 The generalized secant property

In this section we will show that the IQN-LS and IQN-ILS algorithms not only respect the secant equation given in (4.10) and (4.11), but a *generalized secant equation*[3] (of order σ), which we define as

$$\hat{K}'_s(p_{s-j+1} - p_{s-j}) = K(p_{s-j+1}) - K(p_{s-j}), \tag{7.43}$$

resp.

$$p_{s-j+1} - p_{s-j} = \hat{M}'_s(K(p_{s-j+1}) - K(p_{s-j})), \tag{7.44}$$

for $j = 1, 2, \ldots, \sigma$, ($\sigma \leq \min(s, n)$).

7.2.1 IQN-LS

Theorem 7.2.1. *The IQN-LS method (§6.3.1) is a generalized secant method of order* $\min\{s, n\}$ *(equation (7.43)).*

Proof. From the construction of the Jacobian in §6.3.1 we know that there exists a non-singular matrix $\mathcal{T} \in \mathbb{R}^{s\times s}$ such that

$$\begin{aligned}
\tilde{V}_s^{pH} &= [p_{\tilde{n}+1} - p_{\tilde{n}}|p_{\tilde{n}} - p_{\tilde{n}-1}|\ldots|p_s - p_{s-1}] \\
&= V_s^{pH}\mathcal{T} \\
\tilde{W}_s^{pH} &= [H(p_{\tilde{n}+1}) - H(p_{\tilde{n}})|H(p_{\tilde{n}}) - H(p_{\tilde{n}-1})|\ldots|H(p_s) - H(p_{s-1})] \\
&= W_s^{pH}\mathcal{T}.
\end{aligned}$$

(See equation (6.20).)

Then, according to theorem 6.2.1, we have that

$$\hat{K}'_s = \tilde{W}_s^{pH}((\tilde{V}_s^{pH})^T\tilde{V}_s^{pH})^{-1}(\tilde{V}_s^{pH})^T - I.$$

It follows that

$$\begin{aligned}
\hat{K}'_s[p_s - p_{s-1}|p_{s-1} - p_{s-2}|\ldots|p_{\tilde{n}+1} - p_{\tilde{n}}] &= \hat{K}'_s\tilde{V}_s^{pH} \\
&= \tilde{W}_s^{pH}(\tilde{V}_s^{pH})^+\tilde{V}_s^{pH} - \tilde{V}_s^{pH} \\
&= \tilde{W}_s^{pH} - \tilde{V}_s^{pH} \\
&= [K(p_s) - K(p_{s-1})|K(p_{s-1}) - K(p_{s-2})|\ldots \\
&\quad \ldots|K(p_{\tilde{n}+1}) - K(p_{\tilde{n}})],
\end{aligned}$$

[3] Also known as *extended secant condition* [246].

which proves (7.43). □

Corollary 7.2.1. *If K is an affine mapping, then for the IQN-LS method (§6.3.1) the following property holds. $\forall x, y \in \mathbb{R}^{n\times 1}$ such that $x - y \in \mathcal{R}(V_s^{pH})$ we have that*

$$\hat{K}_s'(x-y) = K(x) - K(y). \tag{7.45}$$

Proof. The proof is an immediate consequence of lemmas 2.3.2 and 2.3.3.
As $\hat{K}_s' = A_H \mathcal{L}_s^{pH}(\mathcal{L}_s^{pH})^T - I$ and as $\mathcal{L}_s^{pH}(\mathcal{L}_s^{pH})^T$ is an orthogonal projector on $\mathcal{R}(V_s^{pH})$ we have

$\forall x, y \in \mathbb{R}^{n\times 1} : x - y \in \mathcal{R}(V_s^{pH}) :$

$$\begin{aligned}
\hat{K}_s'(x-y) &= A_H(x-y) - (x-y) && (7.46)\\
&= ((A_H - I)x - b_K) - ((A_H - I)y - b_K) && (7.47)\\
&= K(x) - K(y), && (7.48)
\end{aligned}$$

which proves (7.45). □

Corollary 7.2.2. *For the IQN-LS method (§6.3.1) the following expression holds:*

$$K(p_{s+1}) = (I - \hat{K}_{s+1}'(\hat{K}_s')^{-1})K(p_s). \tag{7.49}$$

Proof. We have that $p_{s+1} = p_s - (\hat{K}_s')^{-1}K(p_s)$ (from the quasi-Newton iteration (4.4)) and (from theorem 7.2.1) $K(p_{s+1}) = K(p_s) + \hat{K}_{s+1}'(p_{s+1} - p_s)$. From this (7.49) follows.

□

Corollary 7.2.3. *For the IQN-LS method (§6.3.1) the following property holds (for $s < n$):*

$$\forall \mathring{A}_H \in \mathbb{A}(V_{s+1}^{pH}, W_{s+1}^{pH}) : K(p_{s+1}) = \mathring{A}_H \bar{L}_{s+1}^{pH}(\bar{L}_{s+1}^{pH})^T \delta p_s, \tag{7.50}$$

with $\delta p_s = p_{s+1} - p_s$ and $\bar{L}_{s+1}^{pH}$ the $(s+1)$-th column of $\mathcal{L}_{s+1}^{pH}$.

Proof. From theorem 7.1.1 we have $\forall \mathring{A}_H \in \mathbb{A}(V_{s+1}^{pH}, W_{s+1}^{pH})$:

$$\hat{K}'_{s+1}(\hat{K}'_s)^{-1} \quad = \quad I + \mathring{A}_H \bar{L}_{s+1}^{pH}(\bar{L}_{s+1}^{pH})^T(\hat{K}'_s)^{-1}. \tag{7.51}$$

Inserting this in (7.49) gives

$$\begin{aligned} K(p_{s+1}) \quad &= \quad -\mathring{A}_H \bar{L}_{s+1}^{pH}(\bar{L}_{s+1}^{pH})^T(\hat{K}'_s)^{-1}K(p_s) && (7.52) \\ &= \quad \mathring{A}_H \bar{L}_{s+1}^{pH}(\bar{L}_{s+1}^{pH})^T \delta p_s, && (7.53) \end{aligned}$$

which completes our proof. □

Remark 7.8. *From the arguments in theorem 7.2.1 it follows that IQN-LS also respects a generalized secant equation, if it were written as*

$$\hat{K}'_s(p_s - p_{s-j}) = K(p_s) - K(p_{s-j}), \tag{7.54}$$

for $j = 1, 2, \ldots, \sigma$, *(*$\sigma \leq \min(s, n)$*).*

Remark 7.9. *Corollary 7.2.2 can be extended for* $s \geq n$ *by using the arguments in* §*7.1.*

7.2.2 IQN-ILS

Theorem 7.2.2. *The IQN-ILS method (*§*6.3.2) is a generalized secant method of order* $\min(s, n)$ *(equation (7.44)).*

The proof is analogous to the one of theorem 7.2.1.

Corollary 7.2.4. *If* K *is an affine mapping, then for the IQN-ILS method (*§*6.3.2) the following property holds.* $\forall x, y \in \mathbb{R}^{n \times 1}$ *such that* $x - y \in \mathcal{R}(V_s^{KH})$ *we have that*

$$x - y = \hat{M}'_s(K(x) - K(y)). \tag{7.55}$$

The proof is analogous to the one of corollary 7.2.1.

Corollary 7.2.5. *For the IQN-ILS method (§6.3.2) the following expression holds:*

$$K(p_{s+1}) = (I - (\hat{M}'_{s+1})^{-1}\hat{M}'_s)K(p_s). \tag{7.56}$$

The proof is analogous to the one of corollary 7.2.2.

Corollary 7.2.6. *For the IQN-ILS method (§6.3.2) the following expression holds (for $s < n$):*

$$\forall \mathring{A}_M \in \mathbb{A}(V^{KH}_{s+1}, W^{KH}_{s+1}) : \delta p_{s+1} = -\mathring{A}_M \bar{L}^{KH}_{s+1}(\bar{L}^{KH}_{s+1})^T K(p_s), \tag{7.57}$$

where $\delta p_{s+1} = p_{s+2} - p_{s+1}$.

The proof is analogous to the one of corollary 7.2.3.

7.2.3 IQN-CLS and IBQN-LS

The notion of the (generalized) secant property can be extended to the approximate Jacobians $\hat{S}'_s$ and $\hat{F}'_s$ used in IQN-CLS and IBQN-LS:

$$\hat{S}'_s(p_{s-j+1} - p_{s-j}) = S(p_{s-j+1}) - S(p_{s-j}) \tag{7.58}$$

$$\hat{F}'_s(g_{s-j+1} - g_{s-j}) = F(g_{s-j+1}) - F(g_{s-j}), \tag{7.59}$$

for $j = 1, 2, \ldots, \sigma$, ($\sigma \leq \min(s, n)$).

Using the same methods as for IQN-LS and IQN-ILS in the previous sections, one can prove that the Least Squares Jacobians $\hat{S}'_s$ and $\hat{F}'_s$ respect a generalized secant property of order $\min(s, n)$[4].
For IQN-CLS the approximate Jacobian for K is $\hat{F}'_s\hat{S}'_s - I$. By using the fact that it is algebraically identical to IQN-LS (which we will show in §7.4.1) it is obvious that this Jacobian also respects (7.43).

[4] Using this terminology, we can say that the Jacobians of IQN-CBG, IQN-CCUM, IBQN-BG and IBQN-CUM respect a generalized secant property of order 1.

7.2.4 Comment

We would like to point out that other methods that respect the generalized secant equation (7.43), resp (7.44), of order $\min\{s,n\}$ can be constructed that differ from IQN-LS, resp. IQN-ILS, as this condition only provides $\min\{s,n\} \times n$ equations for the n^2 unknowns of the approximate Jacobian and is thus under-determined for $s < n$.

7.3 The Least Change Secant Update property

Lemma 7.3.1. *Let* $\alpha \in \mathbb{R}$, $x, v, w_o, w_1, \ldots, w_k \in \mathbb{R}^{\mu\times 1}$ *(*$k < \mu$*), with* $v \notin$ *span*$\{w_o, w_1, \ldots, w_k\}$*; if*

$$\langle v, x\rangle \quad = \alpha \quad , \tag{7.60}$$

$$\langle w_j, x\rangle \quad = 0 \quad \textit{for } j = 0, \ldots, k, \tag{7.61}$$

then the unique solution to $\min \|x\|$ *is*

$$x_{\min} = \frac{\alpha(I-\mathcal{P})v}{\langle v, (I-\mathcal{P})v\rangle}, \tag{7.62}$$

where $\mathcal{P}$ *is an orthogonal projector on the span of* $\{w_o, \ldots, w_k\}$.

Proof. Let the dimension of $\mathcal{W}$=span$\{w_0, \ldots, w_k\}$ be ν, and let $\{e_1, \ldots, e_\nu\}$ be an orthonormal basis for $\mathcal{W}$. We complete this basis with $\{e_{\nu+1}, \ldots, e_\mu\}$ to obtain an orthonormal basis for $\mathbb{R}^{\mu\times 1}$; in other words $\{e_{\nu+1}, \ldots, e_\mu\}$ is an orthonormal basis for $\mathcal{W}^\perp$.
As $\langle w_j, x\rangle = 0$ ($j = 0, 1, \ldots, k$), we have $x \in \mathcal{W}^\perp$. Hence, we can then write

$$x \quad = \quad \sum_{i=\nu+1}^{\mu} x_i e_i \tag{7.63}$$

and

$$v \quad = \quad \sum_{i=1}^{\mu} v_i e_i. \tag{7.64}$$

The condition $\langle v, x\rangle = \alpha$ then becomes $\displaystyle\sum_{i=\nu+1}^{\mu} v_i x_i = \alpha$. So, in order to find $x \in \mathbb{R}^{n\times 1}$ that satisfies (7.60) and (7.61) and which minimizes $\|x\|$, we have to find $x_{\nu+1},\dots,x_\mu$ which minimizes the function

$$f(x_{\nu+1},\dots,x_\mu)=\sum_{i=\nu+1}^{\mu} x_i^2, \tag{7.65}$$

with the additional constraint

$$g(x_{\nu+1},\dots,x_\mu)=\sum_{i=\nu+1}^{\mu} v_i x_i-\alpha=0. \tag{7.66}$$

The solution of (7.65-7.66) with the method of Lagrange yields

$$x_i = \frac{\alpha v_i}{\sum_{i=\nu+1}^{\mu} v_i^2}, \tag{7.67}$$

and hence

$$\begin{aligned} x_{\min} &= \sum_{i=\nu+1}^{\mu}\frac{\alpha v_i e_i}{\sum_{i=\nu+1}^{\mu} v_i^2} && (7.68)\\ &= \frac{\alpha(I-\mathcal{P})v}{\|(I-\mathcal{P})v\|^2} && (7.69)\\ &= \frac{\alpha(I-\mathcal{P})v}{\langle v,(I-\mathcal{P})v\rangle}. && (7.70)\end{aligned}$$

□

We also present an alternative proof for this lemma [223].

Proof. For $\alpha=0$ the proof is trivial ($x_{\min}=0$).
Assume $\alpha\neq 0$ and let $\mathcal{W}$=span$\{w_0,\dots,w_k\}$. Then there are unique vectors $v_1\in\mathcal{W}$ and $v_2\in\mathcal{W}^\perp$ such that $v=v_1+v_2$.
From the condition $\langle x,w_i\rangle=0$ ($i=0,1\dots,k$) we have $x\in\mathcal{W}^\perp$ and as a consequence $\alpha=\langle x,v\rangle=\langle x,v_2\rangle$.
As $x\in\mathcal{W}^\perp$, we are also able to write $x=x_1+x_2$ where $x_1=\beta v_2$ ($\beta\in\mathbb{R}$) and $x_2\in(\mathcal{W}+\text{span } v_2)^\perp$. As $\langle x_2,v_2\rangle=0$ it follows that

$$\alpha=\langle x,v_2\rangle=\langle x_1,v_2\rangle=\beta\langle v_2,v_2\rangle \tag{7.71}$$

and hence $\beta=\frac{\alpha}{\langle v_2,v_2\rangle}$.
The vector $x=\frac{\alpha}{\langle v_2,v_2\rangle}v_2+x_2$, with arbitrary $x_2\in\text{span}(v_2)^\perp$ satisfies (7.60) and

(7.61).
To minimize $\|x\|^2 = \|x_1\|^2 + \|x_2\|^2$ it suffices to take $x_2 = 0$. It follows that

$$x = x_1 = \frac{\alpha}{\langle v_2, v_2\rangle} v_2 \tag{7.72}$$

$$= \frac{\alpha(I-\mathcal{P})v}{\langle v, (I-\mathcal{P})v\rangle}. \tag{7.73}$$

□

The following proof is a generalization of the one in [50].

Theorem 7.3.1. *Let $\Upsilon \in \mathbb{R}^{n\times n}$, $x_{k_{\min}}, \ldots, x_{k_{\max}}, y_{k_{\min}}, \ldots, y_{k_{\max}} \in \mathbb{R}^{n\times 1}$ and $x_{k_{\min}}, \ldots, x_{k_{\max}} \neq 0$ ($k_{\max} - k_{\min} \leq n-1$).*
Then the unique solution to

$$\min_{\Upsilon_* \in \mathbb{A}\left(X_{k_{\min}}^{k_{\max}}, Y_{k_{\min}}^{k_{\max}}\right)} \|\Upsilon_* - \Upsilon\|_{Fr} \text{ with } \Upsilon \in \mathbb{A}\left(X_{k_{\min}}^{k_{\max}-1}, Y_{k_{\min}}^{k_{\max}-1}\right), \tag{7.74}$$

where $X_i^j = [x_i|x_{i+1}|\ldots|x_j]$ and $Y_i^j = [y_i|y_{i+1}|\ldots|y_j]$, is

$$\Upsilon_+ = \Upsilon + \frac{(y_{k_{\max}} - \Upsilon x_{k_{\max}})\left((I-\mathcal{P})x_{k_{\max}}\right)^T}{\langle x_{k_{\max}}, (I-\mathcal{P})x_{k_{\max}}\rangle}, \tag{7.75}$$

where $\mathcal{P}$ is an orthogonal projector on the span of $\{x_{k_{\min}}, \ldots, x_{k_{\max}-1}\}$.

Proof. We first define $\Delta_\Upsilon = \Upsilon_* - \Upsilon$.

We have for $j = k_{\min}, \ldots k_{\max-1}$:

$$\Upsilon_* x_j = \Upsilon x_j + \Delta_\Upsilon x_j \tag{7.76}$$

$$y_j = y_j + \Delta_\Upsilon x_j \tag{7.77}$$

$$\Delta_\Upsilon x_j = 0. \tag{7.78}$$

For $j = k_{\max}$ we have

$$\Upsilon_* x_{k_{\max}} = \Upsilon x_{k_{\max}} + \Delta_\Upsilon x_{k_{\max}} = y_{k_{\max}}. \tag{7.79}$$

We can thus restate the problem as

$$\min_{\Delta_\Upsilon \in \mathbb{R}^{n\times n}} \|\Delta_\Upsilon\|_{Fr} \tag{7.80}$$

subject to

$$\begin{aligned} \Delta_\Upsilon x_{k_{\max}} &= \quad y_{k_{\max}} - \Upsilon x_{k_{\max}} && \qquad (7.81) \\ \Delta_\Upsilon x_j &= \qquad 0 \qquad \text{for } j = k_{\min}, \ldots, k_{\max}-1. && \qquad (7.82) \end{aligned}$$

If we write $[\Delta_\Upsilon]_i$ for the i-th row of Δ_Υ then the original problem becomes n disjoint problems ($i = 1, \ldots, n$)

$$\min_{[\Delta_\Upsilon]_i \in \mathbb{R}^{1\times n}} \|[\Delta_\Upsilon]_i\|_{Fr} \tag{7.83}$$

subject to

$$\begin{aligned} \langle [\Delta_\Upsilon]_i^T, x_{k_{\max}} \rangle &= \quad [y_{k_{\max}} - \Upsilon x_{k_{\max}}]_i && \qquad (7.84) \\ \langle [\Delta_\Upsilon]_i^T, x_j \rangle &= \qquad 0 \qquad \text{for } j = k_{\min}, \ldots, k_{\max}-1 && \qquad (7.85) \end{aligned}$$

Finally, according to lemma 7.3.1 we have ($i = 1, \ldots, n$)

$$\Delta_\Upsilon \quad = \quad \frac{(y_{k_{\max}} - \Upsilon x_{k_{\max}})\left((I-\mathcal{P})x_{k_{\max}}\right)^T}{\langle x_{k_{\max}}, (I-\mathcal{P})x_{k_{\max}}\rangle}, \tag{7.86}$$

and thus

$$\Delta_\Upsilon \quad = \quad \frac{(y_{k_{\max}} - \Upsilon x_{k_{\max}})\left((I-\mathcal{P})x_{k_{\max}}\right)^T}{\langle x_{k_{\max}}, (I-\mathcal{P})x_{k_{\max}}\rangle}, \tag{7.87}$$

which completes our proof. □

7.3.1 IQN-LS

Using theorem 7.3.1 and equation (7.12) we see that for IQN-LS we have that $\hat{K}'_{s+1}$ is the solution of

$$\begin{aligned} &\min_{K_* \in \mathbb{A}\left([\delta p]^s_{\max\{0,s-n+1\}},[\delta K]^s_{\max\{0,s-n+1\}}\right)} \|K_* - \hat{K}'_s\|_{Fr}, \\ &\text{with } \hat{K}'_s \in \mathbb{A}\left([\delta p]^{s-1}_{\max\{0,s-n+1\}}, [\delta K]^{s-1}_{\max\{0,s-n+1\}}\right), \end{aligned} \tag{7.88}$$

where $[\delta p]^j_i = [\delta p_i|\delta p_{i+1}|\dots|\delta p_j]$ and $[\delta K]^j_i = [\delta K_i|\delta K_{i+1}|\dots|\delta K_j]$;
i.e. of all possible rank-one updates applied to $\hat{K}'_s$ that respect the generalized secant equation (7.43) of order $\min\{s,n\}$ the update used in the IQN-LS method results in a value of $\|\hat{K}'_{s+1} - \hat{K}'_s\|_{Fr}$ that is minimal. The method is thus part of the LCSU family (§4.3).
We can compare this with Broyden's good method, where $\hat{K}'_{s+1}$ is solution of

$$\min_{K_* \in \mathbb{A}(\delta p_s, \delta K_s)} \|K_* - \hat{K}'_s\|_{Fr}.$$

This finding re-iterates the notion that IQN-LS respects a generalized secant condition whereas Broyden's good method only respects the ordinary secant condition (i.e. a generalized secant condition of order 1).

Similarly to theorem 4.4.1 we have the following property:

Theorem 7.3.2. *Let Q be an arbitrary matrix in $\mathbb{A}\left([\delta p]^s_{\max\{0,s-n+1\}}, [\delta K]^s_{\max\{0,s-n+1\}}\right)$. If $\hat{K}'_{s+1}$ and $\hat{K}'_s$ are defined by the IQN-LS update, then*

$$\|\hat{K}'_{s+1} - Q\|_{Fr} \leq \|\hat{K}'_s - Q\|_{Fr}. \tag{7.89}$$

7.3.2 IQN-ILS

Using theorem 7.3.1 and equation (7.36) we see that for IQN-ILS we have that

$$\begin{aligned} &\min_{M_* \in \mathbb{A}\left([\delta K]^s_{\max\{0,s-n+1\}},[\delta p]^s_{\max\{0,s-n+1\}}\right)} \|M_* - \hat{M}'_s\|_{Fr}, \\ &\text{with } \hat{M}'_s \in \mathbb{A}\left([\delta K]^{s-1}_{\max\{0,s-n+1\}}, [\delta p]^{s-1}_{\max\{0,s-n+1\}}\right); \end{aligned} \tag{7.90}$$

i.e. of all possible rank-one updates applied to $\hat{M}'_s$ that respect the generalized secant equation (7.44) of order $\min\{s,n\}$ the update used in the IQN-ILS method

results in a value of $\|\hat{M}'_{s+1} - \hat{M}'_s\|_{Fr}$ that is minimal. The method is thus part of the LCSU family (§4.3).
For Broyden's bad method $\hat{M}'_{s+1}$ is solution of

$$\min_{M_* \in \mathbb{A}(\delta K_s, \delta p_s)} \|M_* - \hat{M}'_s\|_{Fr}.$$

This finding re-iterates the notion that IQN-ILS respects a generalized secant condition whereas Broyden's bad method only respects the ordinary secant condition.

Similarly to theorem 4.4.2 we have the following property:

Theorem 7.3.3. *Let Q be an arbitrary matrix in $\mathbb{A}\left([\delta K]^s_{\max\{0,s-n+1\}}, [\delta p]^s_{\max\{0,s-n+1\}}\right)$. If $\hat{M}'_{s+1}$ and $\hat{M}'_s$ are defined by the IQN-ILS update, then*

$$\|\hat{M}'_{s+1} - Q\|_{Fr} \leq \|\hat{M}'_s - Q\|_{Fr}. \tag{7.91}$$

7.3.3 IQN-CLS and IBQN-LS

Similarly to IQN-LS and IQN-ILS we see that the approximate Jacobians $\hat{F}'_s$ and $\hat{S}'_s$ of the IQN-CLS and IBQN-LS methods respect a Least Change Secant Update principle.

7.4 Equivalence between the different Least Squares methods

7.4.1 Equivalence between IQN-LS and IQN-CLS

In this section we show that IQN-LS and IQN-CLS are algebraically identical, although we will keep both methods to see if they behave the same numerically.
To see this we note that for IQN-LS the approximate Jacobian is constructed as

$$\hat{K}'_s \quad = \quad W^{pH}_s[(V^{pH}_s)^T V^{pH}_s]^{-1}(V^{pH}_s)^T - I,$$

while for IQN-CLS this is

$$\begin{aligned}\hat{K}'_s &= \hat{F}'_s\hat{S}'_s - I\\ &= \left(W^{gF}_s[(V^{gF}_s)^T V^{gF}_s]^{-1}(V^{gF}_s)^T\right)\left(W^{pS}_s[(V^{pS}_s)^T V^{pS}_s]^{-1}(V^{pS}_s)^T\right) - I.\end{aligned}$$

Noting that, for this scheme, $g_s = S(p_s)$, we get $W^{pS}_s = V^{gF}_s$ and thus

$$\hat{K}'_s = \hat{F}'_s\hat{S}'_s - I \quad = \quad W^{gF}_s[(V^{pS}_s)^T V^{pS}_s]^{-1}(V^{pS}_s)^T - I.$$

As $H(p_s) = F(g_s)$ this also gives $W_s^{gF} = W_s^{pH}$; as we also have $V_s^{pS} = V_s^{pH}$ this shows the equivalence between IQN-LS and IQN-CLS.

Another way to prove this equivalence is based on the following theorem.

Theorem 7.4.1. *Consider the following algorithms:*

- *the IQN method (§5.1) with a rank-one update of the Jacobian given by (for* $s = 0, 1 \ldots$*)*

$$\hat{K}'_{s+1,IQN} = \hat{K}'_s + \frac{(\delta K_s - \hat{K}'_s \delta p_s) v_s^T}{\langle \delta p_s, v_s \rangle}, \tag{7.92}$$

with $v_s \in \mathbb{R}^{n \times 1}$*;*

- *the IQN-C method (§5.2) with a rank-one update of the Jacobians given by*

$$\hat{F}'_{s+1} = \hat{F}'_s + \frac{(\delta F_s - \hat{F}'_s \delta g_s) w_s^T}{\langle \delta g_s, w_s \rangle} \tag{7.93a}$$

$$\hat{S}'_{s+1} = \hat{S}'_s + \frac{(\delta S_s - \hat{S}'_s \delta p_s) z_s^T}{\langle \delta p_s, z_s \rangle}, \tag{7.93b}$$

and

$$\hat{K}'_{s+1,IQN-C} = \hat{F}'_{s+1} \hat{S}'_{s+1} - I, \tag{7.94}$$

with $w_s \in \mathbb{R}^{m \times 1}$ *and* $z_s \in \mathbb{R}^{n \times 1}$*.*

A sufficient condition so that the IQN-C method produces the same approximate Jacobian for K *as the IQN method (and hence are algebraically identical methods) is*

1. $v_o = z_o$*, for* $\hat{K}'_1$ *(i.e. for* $s = 0$*);*
2. $v_s = z_s$*, and* $w_s^T \hat{S}'_s = 0$ *for* $\hat{K}'_{s+1}$ *(*$s \geq 1$*).*

Proof. This proof will be done by induction.
For $s = 0$ we have, from chapter 5 that $\hat{F}'_o$ and $\hat{S}'_o$ are zero matrices and $\hat{K}'_o = -I$. Equations (7.93-7.94) then become

$$\begin{aligned} \hat{K}'_{1,IQN-C} &= \left(\hat{F}'_o + \frac{(\delta F_o - \hat{F}'_o \delta g_o) w_o^T}{\langle \delta g_o, w_o \rangle} \right) \left(\hat{S}'_o + \frac{(\delta S_o - \hat{S}'_o \delta p_o) z_o^T}{\langle \delta p_o, z_o \rangle} \right) - I \\ &= \frac{\delta F_o w_o^T \delta S_o z_o^T}{\langle \delta g_o, w_o \rangle \langle \delta p_o, z_o \rangle} - I. \end{aligned}$$

As for IQN-C we have that $\delta F_s = \delta H_s$ and $\delta g_s = \delta S_s$ ($s = 0, 1\ldots$), and as $\langle \delta g_o, w_o\rangle = w_o^T \delta g_o$,this becomes

$$\hat{K}'_{1,IQN-C} \quad = \quad \frac{\delta H_o z_o^T}{\langle \delta p_o, z_o\rangle} - I. \tag{7.95}$$

For IQN we have, from equation (7.92),

$$\begin{aligned}\hat{K}'_{1,IQN} &= \hat{K}'_o + \frac{(\delta K_o - \hat{K}'_o \delta p_o) v_o^T}{\langle \delta p_o, v_o\rangle} \\ &= -I + \frac{(\delta K_o + \delta p_o) v_o^T}{\langle \delta p_o, v_o\rangle}.\end{aligned}$$

As $\delta K_s = \delta H_s - \delta p_s$ for $s = 0, 1, \ldots,$ we obtain

$$\begin{aligned}\hat{K}'_{1,IQN} &= -I + \frac{(\delta H_o - \delta p_o + \delta p_o) v_o^T}{\langle \delta p_o, v_o\rangle} \\ &= \frac{\delta H_o v_o^T}{\langle \delta p_o, v_o\rangle} - I. \end{aligned} \tag{7.96}$$

Hence we see that (7.95) equals (7.96) if $z_o = v_o$, i.e. $\hat{K}'_{1,IQN} = \hat{K}'_{1,IQN-C}$ if $z_o = v_o$.

For $s \geq 1$ we start from the assumption that $\hat{K}'_{s,IQN-C} = \hat{K}'_{s,IQN} = \hat{K}'_s$ and prove that $\hat{K}'_{s+1,IQN-C} = \hat{K}'_{s+1,IQN} = \hat{K}'_{s+1}$ follows.
Introducing $\Delta\hat{K}'_{IQN-C} = \hat{K}'_{s+1,IQN-C} - \hat{K}'_s$) we obtain for IQN-C, from equations (7.93-7.94), that

$$\begin{aligned}&\Delta\hat{K}'_{IQN-C} \\ = &\left(\hat{F}'_s + \frac{(\delta F_s - \hat{F}'_s \delta g_s) w_s^T}{\langle \delta g_s, w_s\rangle}\right)\left(\hat{S}'_s + \frac{(\delta S_s - \hat{S}'_s \delta p_s) z_s^T}{\langle \delta p_s, z_s\rangle}\right) - \hat{F}'_s\hat{S}'_s \\ = &\frac{(\delta F_s - \hat{F}'_s \delta g_s) w_s^T}{\langle \delta g_s, w_s\rangle}\hat{S}'_s + \hat{F}'_s \frac{(\delta S_s - \hat{S}'_s \delta p_s) z_s^T}{\langle \delta p_s, z_s\rangle} \\ &+ \frac{(\delta F_s - \hat{F}'_s \delta g_s) w_s^T}{\langle \delta g_s, w_s\rangle}\frac{(\delta S_s - \hat{S}'_s \delta p_s) z_s^T}{\langle \delta p_s, z_s\rangle}.\end{aligned}$$

We recall that we are proving that $v_s = z_s$ and $w_s^T \hat{S}'_s = 0$ are sufficient conditions for the equivalence. Hence, using $w_s^T \hat{S}'_s = 0$ and as $\delta g_s = \delta S_s$, the expression for $\Delta\hat{K}'_{IQN-C}$ becomes

$$\Delta\hat{K}'_{IQN-C} \quad = \quad \hat{F}'_s \frac{(\delta g_s - \hat{S}'_s \delta p_s) z_s^T}{\langle \delta p_s, z_s\rangle} + \frac{(\delta F_s - \hat{F}'_s \delta g_s) w_s^T \delta g_s z_s^T}{\langle \delta g_s, w_s\rangle\langle \delta p_s, z_s\rangle}.$$

Using $\langle \delta g_s, w_s\rangle = w_s^T \delta g_s$ this expression can be simplified to

$$\begin{aligned}\Delta \hat{K}'_{IQN-C} &= \hat{F}'_s \frac{(\delta g_s - \hat{S}'_s \delta p_s) z_s^T}{\langle \delta p_s, z_s\rangle} + \frac{\delta F_s z_s^T - \hat{F}'_s \delta g_s z_s^T}{\langle \delta p_s, z_s\rangle} \\ &= \left(-\frac{\hat{F}'_s \hat{S}'_s \delta p_s}{\langle \delta p_s, z_s\rangle} + \frac{\delta F_s}{\langle \delta p_s, z_s\rangle} \right) z_s^T .\end{aligned}$$

As $\hat{F}'_s \hat{S}'_s = \hat{K}'_s + I$ and $\delta F_s = \delta H_s$, we obtain

$$\begin{aligned}\Delta \hat{K}'_{IQN-C} &= \frac{-(\hat{K}'_s + I)\delta p_s + \delta H_s}{\langle \delta p_s, z_s\rangle} z_s^T && (7.97) \\ &= \frac{\left(\delta H_s - \delta p_s - \hat{K}'_s \delta p_s\right) z_s^T}{\langle \delta p_s, z_s\rangle} . && (7.98)\end{aligned}$$

For IQN we obtain

$$\begin{aligned}\Delta \hat{K}'_{IQN} &= \hat{K}'_{s+1,IQN} - \hat{K}'_s \\ &= \frac{(\delta K_s - \hat{K}'_s \delta p_s) v_s^T}{\langle \delta p_s, v_s\rangle} .\end{aligned}$$

As $\delta K_s = \delta H_s - \delta p_s$, we finally obtain

$$\Delta \hat{K}'_{IQN} = \frac{\left(\delta H_s - \delta p_s - \hat{K}'_s \delta p_s\right) v_s^T}{\langle \delta p_s, v_s\rangle} . \qquad (7.99)$$

We see that (7.99) equals (7.98) if $z_s = v_s$ and $w_s^T \hat{S}'_s = 0$, i.e. $\hat{K}'_{s+1,IQN} = \hat{K}'_{s+1,IQN-C}$ if $z_s = v_s$ and $w_s^T \hat{S}'_s = 0$.

We conclude that, under the assumptions of the theorem, IQN-C and IQN are algebraically identical. □

Corollary 7.4.1. *IQN-LS and IQN-CLS are algebraically identical methods.*

Proof. Note that for IQN-LS and IQN-CLS in theorem 7.4.1 we have
$v_s = (I - \mathcal{L}_s^{pH} (\mathcal{L}_s^{pH})^T) \delta p_s = (I - \mathcal{L}_s^{pS} (\mathcal{L}_s^{pS})^T) \delta p_s = z_s$
and
$w_s = (I - \mathcal{L}_s^{gF} (\mathcal{L}_s^{gF})^T) \delta g_s$.
As $\hat{S}'_s = W_s^{pS} ((V_s^{pS})^T V_s^{pS})^{-1} (V_s^{pS})^T$, and for IQN-C that $W_s^{pS} = V_s^{gF}$, we have that

$$w_s^T \hat{S}'_s = w_s^T (V_s^{gF} ((V_s^{pS})^T V_s^{pS})^{-1} (V_s^{pS})^T). \qquad (7.100)$$

As, by construction, w_s is orthogonal to every column of V_s^{gF} we have that $w_s^T \hat{S}_s' = 0$, and that, hence, IQN-LS and IQN-CLS are identical. □

Worthy of mention in the proof of corollary 7.4.1 is that no condition $z_s^T \hat{F}_s' = 0$ needs to be imposed for the equivalence to hold.

Remark Although theorem 7.4.1 is only a sufficient condition for the theorem to hold, a simple example suffices to show that IQN-BG and IQN-CBG, resp. IQN-CUM and IQN-CCUM, which do not satisfy this sufficient condition, are not identical.

7.4.2 Equivalence between IQN-CLS and IBQN-LS when F is an affine mapping

In this section we show that IQN-CLS (and hence IQN-LS) is algebraically identical to IBQN-LS if F is an affine mapping. As always, we assume that no singularities occur in either algorithm.

We recall that the iterations for the IBQN-LS algorithm are

$$\begin{aligned} p_{s+1} &= (I - \hat{F}_s' \hat{S}_s')^{-1} \Big(F(g_s) + \hat{F}_s'(S(p_s) - \hat{S}_s' p_s - g_s) \Big) && (7.101) \\ g_{s+1} &= (I - \hat{S}_{s+1}' \hat{F}_s')^{-1} \Big(S(p_{s+1}) + \hat{S}_{s+1}'(F(g_s) - \hat{F}_s' g_s - p_{s+1}) \Big) . && (7.102) \end{aligned}$$

If F is an affine operator ($F(p) = A_F p - b_F$) we will show that both algorithms are equivalent[5] in the sense that they give the same iteration values for p and the same approximate Jacobians $\hat{S}_s'$ and $\hat{F}_s'$, but different iteration values for g. The proof of the equivalence will be done by induction.
We will use subscripts “C” for values related to IQN-CLS and “B” for those related to IBQN-LS.

- $s = 0, 1$.
  We first remark that $p_o, g_o = S(p_o), p_1 = F(g_o)$ and $g_1 = S(p_1)$ are identical for both IBQN-LS and IQN-CLS. This is the startup phase, where we have assumed we do not use an initial relaxation parameter. It follows that $\hat{S}_1'$ and $\hat{F}_1'$ are also identical as the input-output pairs are identical.

[5] No such requirement is imposed on S.

- $s = 2$.

  We now prove that p_2 for IBQN-LS, which we will call $p_{2,B}$, is identical to p_2 for IQN-CLS, which we will call $p_{2,C}$. We have

$$p_{2,B} = (I - \hat{F}'_1\hat{S}'_1)^{-1}(F(g_1) + \hat{F}'_1 S(p_1) - \hat{F}'_1\hat{S}'_1 p_1 - \hat{F}'_1 g_1). \tag{7.103}$$

As $g_1 = S(p_1)$ this becomes

$$p_{2,B} = (I - \hat{F}'_1\hat{S}'_1)^{-1}(F(g_1) - \hat{F}'_1\hat{S}'_1 p_1). \tag{7.104}$$

For IQN-CLS we have

$$p_{2,C} = p_1 - (\hat{F}'_1\hat{S}'_1 - i)^{-1}(F(g_1) - p_1) \tag{7.105}$$

$$p_{2,C} = p_1 + (I - \hat{F}'_1\hat{S}'_1)^{-1}(F(g_1) - (p_1 - \hat{F}'_1\hat{S}'_1 p_1) - \hat{F}'_1\hat{S}'_1 p_1) \tag{7.106}$$

$$p_{2,C} = (I - \hat{F}'_1\hat{S}'_1)^{-1}(F(g_1) - \hat{F}'_1\hat{S}'_1 p_1), \tag{7.107}$$

and hence $p_{2,B} = p_{2,C} = p_2$ and $\hat{S}'_2$ is identical for both algorithms.
We now look into $g_{2,B}$. From (7.102) we obtain

$$g_{2,B} = S(p_2) + \hat{S}'_2(F(g_1) - \hat{F}'_1 g_1 - p_2 + \hat{F}'_1 g_{2,B}), \tag{7.108}$$

while $g_{2,C} = S(p_2)$.
We know from lemma 2.3.2 that

$$\forall \mathring{A}_S \in \mathbb{A}(V_2^{pS}, W_2^{pS}) : \hat{S}'_2 = \mathring{A}_S \mathcal{L}_2^{pS}(\mathcal{L}_2^{pS})^T. \tag{7.109}$$

As $\mathcal{L}_2^{pS}(\mathcal{L}_2^{pS})^T$ is an orthogonal projector on the range of $\mathcal{L}_2^{pS}$, which is the space spanned by $(p_2 - p_1)$ and $(p_2 - p_O)$, we have that $\forall x \in \mathbb{R}^{n\times 1}$, $\exists \alpha, \beta \in \mathbb{R}$:

$$\mathcal{L}_2^{pS}(\mathcal{L}_2^{pS})^T x = \alpha(p_2 - p_1) + \beta(p_2 - p_o) = V_2^{pS}[\alpha\ \beta]^T, \tag{7.110}$$

and as a consequence

$$\hat{S}'_2 x = \mathring{A}_S V_2^{pS}[\alpha\ \beta]^T = W_2^{pS}[\alpha\ \beta]^T = \alpha(g_{2,C} - g_1) + \beta(g_{2,C} - g_o), \tag{7.111}$$

because $S(p_i) = g_{i,C}$ ($i = 0, 1, 2$).
There thus exist coefficients γ_1, γ_2 that allow us to write (7.108) as

$$g_{2,B} = S(p_2) + \gamma_1(g_{2,C} - g_o) + \gamma_2(g_{2,C} - g_1) \tag{7.112}$$

$$g_{2,B} = g_{2,C} + \gamma_1(g_{2,C} - g_o) + \gamma_2(g_{2,C} - g_1). \tag{7.113}$$

We are now able to compare $\hat{F}'_{2,B}$ and $\hat{F}'_{2,C}$. By applying lemma 2.3.2 to both we can write

$$\forall \mathring{A}_{F,C} \in \mathbb{A}(V^{gF}_{2,C}, W^{gF}_{2,C}) \quad : \quad \hat{F}'_{2,C} = \mathring{A}_{F,C}\mathcal{L}^{gF}_{2,C}(\mathcal{L}^{gF}_{2,C})^T \quad (7.114)$$
$$\forall \mathring{A}_{F,B} \in \mathbb{A}(V^{gF}_{2,B}, W^{gF}_{2,B}) \quad : \quad \hat{F}'_{2,B} = \mathring{A}_{F,B}\mathcal{L}^{gF}_{2,B}(\mathcal{L}^{gF}_{2,B})^T. \quad (7.115)$$

From (7.113) we get

$$g_{2,B} - g_o = (\gamma_1 + 1)(g_{2,C} - g_o) + \gamma_2(g_{2,C} - g_1) \quad (7.116a)$$
$$g_{2,B} - g_1 = \gamma_1(g_{2,C} - g_o) + (\gamma_2 + 1)(g_{2,C} - g_1). \quad (7.116b)$$

We thus see that $\exists \mathcal{T} \in \mathbb{R}^{2\times 2} : V^{gF}_{2,B} = V^{gF}_{2,C}\mathcal{T}$, with $\mathcal{T}$ non-singular if both IQN-CLS and IBQN-CLS are non-singular.
If F is affine, then we also have $W^{gF}_{2,B} = W^{gF}_{2,C}\mathcal{T}$ and thus, using theorem 6.2.1 we have that $\hat{F}'_{2,C} = \hat{F}'_{2,B}$.

- $s > 2$.
  To complete the proof of the equivalence between IQN-CLS and IBQN-LS we assume that p_s and $\hat{S}'_s$ are identical for both algorithms and that from this it follows that p_{s+1} and $\hat{S}'_{s+1}$ are identical as well.
  Under the above assumption we can show that $\hat{F}'_s$ is also identical for both schemes, in analogy to the procedure in equations (7.108)-(7.116) used for $\hat{F}'_2$. From equation (7.102) we have

$$g_{s,B} = S(p_s) + \hat{S}'_{s,B}(F(g_{s-1,B}) - \hat{F}'_{s-1}g_{s-1,B} - p_s + \hat{F}'_{s-1}g_{s,B}). \quad (7.117)$$

Knowing that $\hat{S}'_{s,B} = \hat{S}'_{s,C} = \hat{S}'_s$ we see that, in the manner of equation (7.113), γ_i ($i = 1, \dots, s$) can be found such that $g_{s,B}$ can be written as

$$g_{s,B} = g_{s,C} + \sum_{i=0}^{s} \gamma_i(g_{s,C} - g_{i,C}). \quad (7.118)$$

In an analogous way as in the first part of this argument we thus have $\mathbb{A}(V^{gF}_{s,C}, W^{gF}_{s,C}) = \mathbb{A}(V^{gF}_{s,B}, W^{gF}_{s,B})$ and $\hat{F}'_{s,B} = \hat{F}'_{s,C} = \hat{F}'_s$ follows. From this last equation we can also derive that $\forall \mathring{A}_F \in \mathbb{A}(V^{gF}_{s,C}, W^{gF}_{s,C})$ we have

$$\hat{F}'_s(g_{s,B} - g_{s,C}) = \mathring{A}_F\mathcal{L}^{gF}_{s,C}(\mathcal{L}^{gF}_{s,C})^T \sum_{i=0}^{s} \gamma_i(g_{s,C} - g_{i,C}) \quad (7.119)$$
$$= \mathring{A}_F \sum_{i=0}^{s} \gamma_i(g_{s,C} - g_{i,C}). \quad (7.120)$$

As we have assumed that F is affine we have

$$\hat{F}_s'(g_{s,B} - g_{s,C}) \quad = \quad F\left(\sum_{i=0}^{s} \gamma_i (g_{s,C} - g_{i,C})\right) \tag{7.121}$$

$$\hat{F}_s' g_{s,B} - \hat{F}_s' g_{s,C} \quad = \quad F(g_{s,B} - g_{s,C}) = F(g_{s,B}) - F(g_{s,C}). \tag{7.122}$$

Then we have for IBQN-LS, using equation (7.101),

$$p_{s+1,B} \quad = \quad (I - \hat{F}_s'\hat{S}_s')^{-1}(F(g_{s,B}) + \hat{F}_s' g_{s,C} - \hat{F}_s'\hat{S}_s' p_s - \hat{F}_s' g_{s,B}) \tag{7.123}$$

and using (7.122)

$$p_{s+1,B} \quad = \quad (I - \hat{F}_s'\hat{S}_s')^{-1}(F(g_{s,B}) + \hat{F}_s' g_{s,C} - \hat{F}_s'\hat{S}_s' p_s - \hat{F}_s' g_{s,B}) \tag{7.124}$$

$$p_{s+1,B} \quad = \quad (I - \hat{F}_s'\hat{S}_s')^{-1}(F(g_{s,C}) - \hat{F}_s'\hat{S}_s' p_s). \tag{7.125}$$

For algorithm IQN-CLS we also have

$$p_{s+1,C} \quad = \quad (I - \hat{F}_s'\hat{S}_s')^{-1}(F(g_{s,C}) - \hat{F}_s'\hat{S}_s' p_s), \tag{7.126}$$

which shows the equivalence.

7.5 Avoiding singularity

In this section we will discuss two type of singularities of IQN-LS in the rank-one update form and one for the original form of (§6.3.1). For the other three Least Squares quasi-Newton methods similar arguments hold.

7.5.1 Avoiding singularity for the original formulation (§6.3.1)

Although we will show in §8.1 that singularities in the construction of the approximate (inverse) Jacobian cannot occur when the mappings are affine and when working in exact arithmetic, it is quite possible that the columns of V_s^{pH}, V_s^{pS}, etc. become linearly dependent of one another when the mappings are non-linear. When this happens, the construction of the approximate Jacobian will fail, as the Moore-Penrose generalized inverse can no longer be constructed using equation

(6.5). We will solve this problem by applying "*QR-filtering*". We will illustrate the principle on V_s^{pH}.

Note that the procedure below is a reworked version of an idea first used in [45].

We apply the "economy size" QR-decomposition to V_s^{pH}:

$$V_s^{pH} = Q_s R_s, \tag{7.127}$$

where $Q_s \in \mathbb{R}^{n \times s}$ such that $Q^T Q = I_s$ and $R_s \in \mathbb{R}^{s \times s}$ is an upper triangular matrix[6].

This corresponds to a Gram-Schmidt orthogonalization process applied to the columns of V_s^{pH} starting from the first column.

When, during the orthogonalization process, but before the normalization step, an orthogonalized column of V_s^{pH} is smaller than 10^{-8} times the norm of the original column of V_s^{pH} (i.e. the column which we are orthogonalizing), then this indicates that the column was (nearly) linearly dependent of the previous columns. We then discard this column of V_s^{pH}, and the corresponding column of W_s^{pH}, and restart the orthogonalization procedure. A column is also discarded if, after orthogonalization, its norm is smaller than 10^{-15} times the largest norm of the previously orthogonalized vectors. We do this until all orthogonalized vectors are sufficiently large.

As the most recent input and output modes are leftmost in V_s^{pH} and W_s^{pH} this means that we will discard the oldest conflicting modes.

Note that after filtering we have obtained

$$\left(\mathbf{V}_s^{pH}\right)^+ = \left(\left(\mathbf{V}_s^{pH}\right)^T \mathbf{V}_s^{pH}\right)^{-1} \left(\mathbf{V}_s^{pH}\right)^T = R_s^{-1} Q_s^T. \tag{7.128}$$

7.5.2 Avoiding singularity for the rank-one update formulation

A first type of singularity can occur when $\bar{L}_{s+1}^{pH}$ in (7.2) is zero. For this to happen $(I - \mathcal{L}_s^{pH} (\mathcal{L}_s^{pH})^T) \delta p_s$ needs to be zero, which means that $\delta p_s \in \mathcal{R}(V_s^{pH})$.

Both in the rank-one update formula (7.12) and in the formulation (§6.3.1) this would lead to a singularity of $\hat{K}'_{s+1}$.

In the rank-one update formulation we cannot just use the procedure described in §7.5.1 as we do not store V_s^{pH} and W_s^{pH}. (If we did, we could apply the QR-filtering.)

We therefore propose to keep $\hat{K}'_{s+1} = \hat{K}'_s$ in these circumstances.

[6] For ordinary QR-decomposition we have $Q_s \in \mathbb{R}^{n \times n}$ and $R_s \in \mathbb{R}^{n \times s}$.

A second type of singularity has already been mentioned in §7.1, i.e. the occurrence where

$$(\bar{L}^{pH}_{s+1})^T(\hat{K}'_s)^{-1}\mathring{A}_H\bar{L}^{pH}_{s+1} = -1, \tag{7.129}$$

for $\mathring{A}_H \in \mathbb{A}(V^{pH}_s, W^{pH}_s)$, in which case the approximate Jacobian $\hat{K}'_{s+1}$ for IQN-LS becomes singular.

Again we could propose to keep $\hat{K}'_{s+1} = \hat{K}'_s$ if this occurs. Unfortunately, doing so would cause the loss of the secant property as in general $\hat{K}'_s(p_{s+1} - p_s) \neq K(p_{s+1}) - K(p_s)$.
To satisfy the secant property as closely as possible, we could use a convex average

$$\hat{K}'_{s+1} \quad = \quad (1-\theta_s)\hat{K}'_s + \theta_s\left(\hat{K}'_s + \mathring{A}_H\bar{L}^{pH}_{s+1}(\bar{L}^{pH}_{s+1})^T\right), \tag{7.130}$$

where θ_s is chosen $\in [0,1]$. We choose the value closest to unity that keeps the absolute value of the determinant $\det(\hat{K}'_{s+1})$ above a certain threshold σ (say $0.001\det(\hat{K}'_s)$). To help us choose θ_s, we turn to lemma 2.3.1 to obtain

$$\hat{K}'_{s+1} \quad = \quad \hat{K}'_s\left(I + \theta_s(\hat{K}'_s)^{-1}\mathring{A}_H\bar{L}^{pH}_{s+1}(\bar{L}^{pH}_{s+1})^T\right) \tag{7.131}$$

$$\det(\hat{K}'_{s+1}) \quad = \quad \det(\hat{K}'_s)\left(1 + \theta_s\langle(\hat{K}'_s)^{-1}\mathring{A}_H\bar{L}^{pH}_{s+1}, \bar{L}^{pH}_{s+1}\rangle\right). \tag{7.132}$$

For the update of the inverse approximate Jacobian this becomes:

$$(\hat{K}'_{s+1})^{-1} \quad = \quad (1-\theta_s)(\hat{K}'_s)^{-1} + \theta_s\left((\hat{K}'_s) - \frac{(\hat{K}'_s)^{-1}\mathring{A}_H\bar{L}^{pH}_{s+1}(\bar{L}^{pH}_{s+1})^T(\hat{K}'_s)^{-1}}{1 + \theta_s(\bar{L}^{pH}_{s+1})^T(\hat{K}'_s)^{-1}\mathring{A}_H\bar{L}^{pH}_{s+1}}\right). \tag{7.133}$$

Note that a similar modification is not possible for the first type of singularity as $\bar{L}^{pH}_{s+1}$ does not exist.

Tests have shown that the modifications for both types of singularities are adequate.

8

Properties of IQN-LS, IQN-ILS, IQN-CLS and IBQN-LS for linear mappings

In this section we assume K, H, F and/or S are affine mappings, i.e. $K(p) = A_K p - b_K$, etc., with A_K etc. non-singular. We study the properties of the four Least Squares quasi-Newton methods described in chapter 6 under this assumption. We recall that, as shown in chapter 7, under this assumption IQN-LS, IQN-CLS and IBQN-LS are algebraically identical.

This chapter is organized as follows.
In §8.1 we show that the approximate Jacobians for IQN-LS and IQN-ILS exhibit monotone convergence in the affine case, and that for affine mappings singularities cannot occur before the solution has been reached (in exact arithmetic).
In §8.2 we establish the relationship between IQN-LS and IQN-ILS on one hand and Krylov methods on the other and write GMRes as a quasi-Newton method; we observe similarities between the three methods; finally in §8.3 we discuss the possibilities of using step-length parameters for the Least Squares quasi-Newton methods.
In §8.4 we modify the IQN-LS and IQN-ILS methods to make them algebraically identical to GMRes.

Most of the material in this chapter can be found in [106] and [108].

8.1 Non-singularity and convergence of the approximate Jacobian for IQN-LS and IQN-ILS

We show in the next theorems that, in exact arithmetic, the approximate Jacobian for the IQN-LS and IQN-ILS methods never become singular before the solution has been reached, if the mappings are affine.
These results can then be extended to IQN-CLS and IBQN-LS which are equivalent, under the above assumptions, as shown in §7.4.1 and §7.4.2.

Theorem 8.1.1. *Let K be an affine mapping. Consider the IQN-LS method (§6.3.1). Let $\breve{s}$ be the first value for which $\delta p_{\breve{s}+1}$ ($= p_{\breve{s}+2} - p_{\breve{s}+1}$) is linearly dependent on $\delta p_o, \delta p_1, \ldots, \delta p_{\breve{s}}$, then $p_{\breve{s}+2}$ is the solution of $K(p) = 0$.*

Proof. If $\delta p_{\breve{s}+1}$ is linearly dependent on $\delta p_o, \delta p_1, \ldots, \delta p_{\breve{s}}$ then

$$\delta p_{\breve{s}+1} = \sum_{j=0}^{\breve{s}} k_j \delta p_j \tag{8.1}$$

and

$$\hat{K}'_{\breve{s}+1} \delta p_{\breve{s}+1} = \sum_{j=0}^{\breve{s}} k_j \hat{K}'_{\breve{s}+1} \delta p_j . \tag{8.2}$$

Then we have, according to theorem 7.2.1, that

$$\begin{aligned} \hat{K}'_{\breve{s}+1} \delta p_{\breve{s}+1} &= \sum_{j=0}^{\breve{s}} k_j A_K \delta p_j & (8.3) \\ &= A_K \delta p_{\breve{s}+1} . & (8.4) \end{aligned}$$

We then have

$$\begin{aligned} K(p_{\breve{s}+2}) &= K(p_{\breve{s}+1}) + K(\delta p_{\breve{s}+1}) & (8.5) \\ &= K(p_{\breve{s}+1}) + A_K \delta p_{\breve{s}+1} & (8.6) \\ &= K(p_{\breve{s}+1}) + \hat{K}'_{\breve{s}+1} \delta p_{\breve{s}+1} . & (8.7) \end{aligned}$$

Because $p_{s+2} = p_{s+1} - (\hat{K}'_{s+1})^{-1} K(p_{s+1})$ we finally have

$$K(p_{\breve{s}+2}) = A_K p_{\breve{s}+2} - b = 0, \tag{8.8}$$

which shows that $p_{\breve{s}+2}$ is the solution of $K(p) = 0$ (equation (1.3)). □

Theorem 8.1.2. *Let K be an affine mapping and let A_K be non-singular. Consider the IQN-ILS method (§6.3.2). Let $\breve{s}$ be the first value for which $\delta K_{\breve{s}+1}$ ($= K_{\breve{s}+2} - K_{\breve{s}+1}$) is linearly dependent on $\delta K_o, \delta K_1, \dots, \delta K_{\breve{s}}$, then $p_{\breve{s}+2}$ is the solution of $K(p) = 0$.*

Proof. If $\delta K_{\breve{s}+1}$ is linearly dependent on $\delta K_o, \delta K_1, \dots, \delta K_{\breve{s}}$ then

$$\delta K_{\breve{s}+1} = \sum_{j=0}^{\breve{s}} k_j \delta K_j \tag{8.9}$$

and

$$\hat{M}'_{\breve{s}+1} \delta K_{\breve{s}+1} = \sum_{j=0}^{\breve{s}} k_j \hat{M}'_{\breve{s}+1} \delta K_j. \tag{8.10}$$

Then we have, according to theorem 7.2.2, that

$$\begin{aligned} \hat{M}'_{\breve{s}+1} \delta K_{\breve{s}+1} &= \sum_{j=0}^{\breve{s}} k_j A_K^{-1} \delta K_j & (8.11) \\ &= A_K^{-1} \delta K_{\breve{s}+1}. & (8.12) \end{aligned}$$

Also

$$\begin{aligned} p_{\breve{s}+2} &= p_{\breve{s}+1} - \hat{M}'_{\breve{s}+1} K(p_{\breve{s}+1}) & (8.13) \\ K(p_{\breve{s}+2}) &= K(p_{\breve{s}+1}) - A_K \hat{M}'_{\breve{s}+1} K(p_{\breve{s}+1}). & (8.14) \end{aligned}$$

We also have

$$K(p_{\breve{s}+2}) = K(p_{\breve{s}+1}) + \delta K_{\breve{s}+1}. \tag{8.15}$$

Using (8.12) in (8.15) we obtain

$$K(p_{\breve{s}+2}) = K(p_{\breve{s}+1}) + A_K \hat{M}'_{\breve{s}+1} \delta K_{\breve{s}+1}. \tag{8.16}$$

Combining (8.16) and (8.14) we get

$$\begin{aligned} A_K \hat{M}'_{\breve{s}+1} \delta K_{\breve{s}+1} &= -A_K \hat{M}'_{\breve{s}+1} K(p_{\breve{s}+1}) & (8.17) \\ A_K \hat{M}'_{\breve{s}+1} K(p_{\breve{s}+2}) &= 0, & (8.18) \end{aligned}$$

and as $\hat{M}'_{\breve{s}+1}$ and A_K are non-singular it follows that $K(p_{\breve{s}+2}) = 0$ and that $p_{\breve{s}+2}$ is the solution of $K(p) = 0$ (equation (1.3)). □

Theorem 8.1.3. *If K is an affine mapping, then the approximate Jacobian $\hat{K}'_s$ of the IQN-LS method (§6.3.1) converges to the true Jacobian A_K in a monotone way, i.e.*

$$\|\hat{K}'_{s+1} - A_K\| \leq \|\hat{K}'_s - A_K\| \tag{8.19}$$

($s = 0, 1, 2, \ldots$).

Proof. For the proof of this theorem we apply theorem 2.3.2 in a straightforward manner.
As, for an affine mapping, we have(equation (6.27))

$$\hat{K}'_s = A_H V_s^{pH}[(V_s^{pH})^T V_s^{pH}]^{-1}(V_s^{pH})^T - I, \tag{8.20}$$

and $A_K = A_H - I$, it suffices to replace Q_s in theorem 2.3.2 by $\hat{K}'_s$, T_1 by A_H and T_2 by I, to prove equation (8.19). □

Theorem 8.1.4. *If K is an affine mapping, then the approximation $\hat{M}'_s$ of the inverse of the Jacobian for the IQN-ILS method (§6.3.2) converges to the true inverse Jacobian A_K^{-1} in a monotone way, i.e.*

$$\|\hat{M}'_{s+1} - A_K^{-1}\| \leq \|\hat{M}'_s - A_K^{-1}\| \tag{8.21}$$

($s = 0, 1, 2, \ldots$).

Proof. For the proof of this theorem we apply theorem 2.3.2 in a straightforward manner.
As, for an affine mapping, we have
$\hat{M}'_s = A_H(A_K)^{-1}W_s^{KH}[(V_s^{KH})^T V_s^{KH}]^{-1}(V_s^{KH})^T - I$ (equation (6.39)),
and $A_H A_K^{-1} - I = A_K^{-1}$, it suffices to replace Q_s in theorem 2.3.2 by $\hat{M}'_s$, T_1 by $A_H A_K^{-1}$ and T_2 by I, to prove equation (8.21). □

As a general conclusion we can see that $\hat{K}'_n$, resp. $\hat{M}'_n$, will correspond to the exact Jacobian, resp. inverse of the exact Jacobian (theorem 2.3.2: equation (2.17)) and that hence p_{n+1} will be the exact solution for both IQN-LS and IQN-ILS, when the operators are affine.
This compares very favorable against Broyden's good and bad methods, the Column-Updating method and the Inverse Column-Updating method which, according to Gay's theorem, converge in at most $2n$ iterations for affine operators (cfr. §4.4.1, §4.4.2, §4.4.3 and §4.4.4). For that reason, we will only investigate properties for

which $s \leq n$ further in this chapter.
Unfortunately monotone convergence of the approximate (inverse) Jacobians does not necessarily imply monotone convergence of the iterates[1]. As for IQN-ILS $\hat{M}'_s$ converges towards A_K^{-1} in a monotone manner and as it is the inverse of A_K that is needed in the iteration (equations (4.1) and (4.4)) this shows that IQN-ILS is more likely to exhibit monotone convergence than IQN-LS.
It is believed that this advantage in convergence speed is carried over to weakly non-linear problems.

As IQN-CLS is algebraically equivalent to IQN-LS and equivalent to IBQN-LS for affine operators, these conclusions extend to these two methods.

We would also like to point out that respecting the generalized secant property of order $\min(s, n)$ (equations (7.43) or (7.44)), i.e. for all previous iterates, is clearly a good idea if K is an affine mapping as judged from theorems 8.1.3 and 8.1.4. However, this matter should be investigated in more detail for non-linear systems, as secant properties obtained with points that are far from the actual solution might not be representative of the actual tangent hyperplane and thus might actually hamper convergence.

8.2 Comparison between IQN-LS, IQN-ILS and GMRes

At first sight Newton-GMRes and the various quasi-Newton methods have little in common. Newton-GMRes is an inexact Newton method, meaning that the linear system (4.2) in the exact Newton equation (4.1) is only solved approximately. This is different from quasi-Newton methods where an approximation of the Jacobian is used in the Newton equation, but the resulting system solved exactly.
Nevertheless, Cătinaş [32] has shown in an abstract framework that inexact Newton methods are equivalent to quasi-Newton methods. In this section we will show the relationship between the Least Squares quasi-Newton methods and Newton-GMRes in more detail.
We also note that for the affine mappings we are studying here, Newton-GMRes actually becomes “plain” GMRes if we solve the first Newton step exactly by GMRes. We will make this assumption further on in this section.

[1] An illustration of this can be found in chapter 11.

8.2.1 Writing GMRes as a Quasi-Newton method

We will show how GMRes can be interpreted as a quasi-Newton method when applied to the linear equation $A_K p - b_K = 0$. We will start our derivation from the elementary formulation given in §3.2.2.1.
(We recall that in the GMRes method it is assumed that for every $x \in \mathbb{R}^{n\times 1}$ we are able to form $A_K x$, which does not correspond to our notion of "function call", defined as $K(x) = A_K x - b_K$.)

As we have seen in §3.2.2, the GMRes method constructs

$$p_s = p_o + [r_o \; A_K r_o \ldots A_K^{s-1} r_o][\bar{\omega}_{1,s} \; \bar{\omega}_{2,s} \ldots \bar{\omega}_{s,s}]^T \tag{8.22}$$

such that

$$r_s = r_o + [A_K r_o \; A_K^2 r_o \ldots A_K^s r_o][\bar{\omega}_{1,s} \; \bar{\omega}_{2,s} \ldots \bar{\omega}_{s,s}]^T \tag{8.23}$$

is minimal in the Euclidean norm (theorem 2.3.6). As a result we have (theorem 2.3.7):

$$r_s = \left(I - V_s^{GM} \left((V_s^{GM})^T V_s^{GM}\right)^{-1} (V_s^{GM})^T\right) r_o \tag{8.24}$$

and

$$p_s = p_o - W_s^{GM} \left((V_s^{GM})^T V_s^{GM}\right)^{-1} (V_s^{GM})^T r_o, \tag{8.25}$$

where $V_s^{GM} = [A_K r_o | \; A_K^2 r_o | \ldots | A_K^s r_o]$ and
$W_s^{GM} = [r_o | \; A_K r_o | \ldots | A_K^{s-1} r_o] = A_K^{-1} V_s^{GM}$.
Using lemma 2.3.2 we see that

$$V_s^{GM} \left((V_s^{GM})^T V_s^{GM}\right)^{-1} (V_s^{GM})^T = \mathcal{L}_s^{GM} (\mathcal{L}_s^{GM})^T \tag{8.26}$$

and

$$W_s^{GM} \left((V_s^{GM})^T V_s^{GM}\right)^{-1} (V_s^{GM})^T = A_K^{-1} \mathcal{L}_s^{GM} (\mathcal{L}_s^{GM})^T, \tag{8.27}$$

where the columns of $\mathcal{L}_s^{GM}$ form an orthonormal basis for the range of V_s^{GM} as shown in lemma 2.3.2.
Using theorem 2.3.2, with $T_1 = A_K^{-1}$ and T_2 equal to the zero matrix, we see that for $s \to n$ (8.27) converges monotonically to A_K^{-1}; similarly, (8.26) converges monotonically to I.

GMRes can thus be written as

$$\begin{aligned} p_{s+1} &= p_o - A_K^{-1}\mathcal{L}_s^{GM}(\mathcal{L}_s^{GM})^T r_o \\ A_K(p_{s+1}-p_o) &= -\mathcal{L}_s^{GM}(\mathcal{L}_s^{GM})^T r_o \\ A_K(p_{s+1}-p_o) - A_K(p_s-p_o) &= -\mathcal{L}_s^{GM}(\mathcal{L}_s^{GM})^T r_o - A_K(p_s-p_o) \\ A_K(p_{s+1}-p_s) &= -\mathcal{L}_s^{GM}(\mathcal{L}_s^{GM})^T r_o - A_K(e_s-e_o) \\ &= -\mathcal{L}_s^{GM}(\mathcal{L}_s^{GM})^T r_o - (r_s-r_o). \end{aligned}$$

As $(r_s - r_o) \in \mathcal{R}(V_s^{GM})$ and hence $\mathcal{L}_s^{GM}(\mathcal{L}_s^{GM})^T(r_s-r_o) = (r_s-r_o)$ this becomes

$$A_K(p_{s+1}-p_s) = -\mathcal{L}_s^{GM}(\mathcal{L}_s^{GM})^T r_s, \tag{8.28}$$

and we can thus write GMRes as

$$p_{s+1} = p_s - \underbrace{A_K^{-1}\mathcal{L}_s^{GM}(\mathcal{L}_s^{GM})^T}_{\hat{M}_s'} r_s, \tag{8.29}$$

where $\hat{M}_s'$ can be seen as an approximation to A_K^{-1}. (8.29) corresponds to the form of (4.5) and hence GMRes can be considered as a quasi-Newton method applied to (1.3).

Also note that (8.25) corresponds to (2.8) and that the approximate Jacobian is guaranteed to be non-singular due to the properties of a projection method (definition 2.13).

Theorem 8.2.1. *GMRes, applied to the linear system $K(p) = A_K p - b_K = 0$, is a generalized secant method of order $\min(s,n)$ (equation (7.44)).*

Proof. For GMRes we have that $p_s - p_o \in \mathcal{R}(W_s^{GM})$ (equation (8.25)), for $s = 0, 1, \ldots$. As from theorem 6.2.3 we have that $\mathcal{R}(W_j^{GM}) \subset \mathcal{R}(W_s^{GM})$, for $j = 0, 1, \ldots, s$, we have that $p_j - p_o \in \mathcal{R}(W_j^{GM}) \subset \mathcal{R}(W_s^{GM})$, and hence that $p_s - p_j \in \mathcal{R}(W_s^{GM})$.

It follows that $A_K(p_s - p_j) \in \mathcal{R}(V_s^{GM})$, i.e.

$$\mathcal{L}_s^{GM}(\mathcal{L}_s^{GM})^T A_K(p_s-p_j) = A_K(p_s-p_j), \tag{8.30}$$

from which we obtain that

$$A_K^{-1}\mathcal{L}_s^{GM}(\mathcal{L}_s^{GM})^T A_K(p_s-p_j) = p_s - p_j \tag{8.31}$$

We know that, for GMRes in the form of equation (8.29), the approximate inverse Jacobian can be written as $\hat{M}_s' = A_K^{-1}\mathcal{L}_s^{GM}(\mathcal{L}_s^{GM})^T$. We finally obtain

$$\hat{M}'_s(K(p_s) - K(p_j)) \quad = \quad p_s - p_j, \tag{8.32}$$

which completes our proof. □

Theorem 8.2.2. *Suppose that $\hat{M}'_s$ is constructed as in the GMRes method, applied to the linear problem $K(p) = A_K p - b_K = 0$. Then $\hat{M}'_{s+1}$ is linked to $\hat{M}'_s$ by the following expression (for $s < n$):*

$$\hat{M}'_{s+1} = \hat{M}'_s + A_K^{-1} L_{s+1}^{GM} (L_{s+1}^{GM})^T. \tag{8.33}$$

where $\bar{L}_{s+1}^{GM}$ is the $(s+1)$-th column of $\mathcal{L}_{s+1}^{GM}$.

The proof of this theorem is analogous to that of theorems 7.1.1 and 7.1.2.

8.2.2 Krylov subspaces for IQN-LS

In this section we apply the Quasi-Newton Least Squares method to a single linear system $A_K p - b_K = 0$. We will show that the iterates of this method share the same Krylov search subspace as those of GMRes, but not the subspace of constraints.

Theorem 8.2.3. *Consider the IQN-LS method (§6.3.1). Assume that $\hat{K}'_s$ is non-singular[2] and that K is an affine mapping. Then the following relations hold:*

$$\forall x \in \mathcal{R}(V_j^{pH}) : (\hat{K}'_s)^{-1} A_K x = x, \tag{8.34}$$

($j = 0, 1, \ldots, s$), and

$$\begin{aligned} &\forall j \in \{0, 1, \ldots, s\} : \\ (I - (\hat{K}'_s)^{-1} A_K)(e_s - e_j) \quad &= \quad 0 && \text{(8.35a)} \\ e_{s+1} \quad &= \quad e_j - (\hat{K}'_s)^{-1} A_K e_j && \text{(8.35b)} \\ r_{s+1} \quad &= \quad r_j - A_K (\hat{K}'_s)^{-1} r_j && \text{(8.35c)} \\ p_{s+1} \quad &= \quad p_j - (\hat{K}'_s)^{-1} K(p_j). && \text{(8.35d)} \end{aligned}$$

[2] As shown in theorem 8.1.1, this assumption is always satisfied for affine mappings.

Proof. As, from theorem 6.2.3, $\forall j \in \{0,1,\ldots,s\} : \mathcal{R}(V_j^{pH}) \subset \mathcal{R}(V_s^{pH})$, lemma 2.3.3 allows us to write $\forall x \in \mathcal{R}(V_j^{pH}),\ j = 0,1,\ldots,s$:

$$
\begin{aligned}
(A_H \mathcal{L}_s^{pH}(\mathcal{L}_s^{pH})^T - I)x &= (A_H - I)x && (8.36)\\
x &= (A_H \mathcal{L}_s^{pH}(\mathcal{L}_s^{pH})^T - I)^{-1}(A_H - I)x && (8.37)\\
x &= (\hat{K}_s')^{-1} A_K x, && (8.38)
\end{aligned}
$$

which proves (8.34). It follows that

$$
(I - (\hat{K}_s')^{-1} A_K)x = 0. \qquad (8.39)
$$

We can conclude that $\mathcal{R}(V_s^{pH})$ is part of the null space of $(I - (\hat{K}_s')^{-1}(A_H - I))$ (it equals the null-space if A_H is non-singular).
As $e_s - e_j = p_s - p_j$ we have $(e_s - e_j) \in \mathcal{R}(V_s^{pH})$ from the definition of V_s^{pH}. By replacing x in (8.39) by $(e_s - e_j)$ we obtain (8.35a). Equations (8.35b), (8.35c) and (8.35d) follow immediately. □

Note that this theorem can be extended to non-linear mappings by replacing A_H with $\mathring{A}_H \in \mathbb{A}(V_s^{pH}, W_s^{pH})$.

From this theorem we see that the previous iterates only contribute to the solution process by creating a better approximate Jacobian $\hat{K}_s'$. This can be seen from equation (8.35d) which shows that we can use any previous iterate for the construction of the new iterate as long as the most recent approximate Jacobian is used.

Corollary 8.2.1. *Consider the IQN-LS method (§6.3.1). Assume that $\hat{K}_s'$ is non-singular, that K (or H) is an affine mapping and that the iterations start from an initial guess $p_o = 0$. Then the quasi-Newton iteration can also be written as*

$$
p_{s+1} = (A_H \mathcal{L}_s^{pH}(\mathcal{L}_s^{pH})^T - I)^{-1} b_K \qquad (s \geq 1), \qquad (8.40)
$$

and the error e_{s+1} is given by

$$
e_{s+1} = \left((A_H \mathcal{L}_s^{pH}(\mathcal{L}_s^{pH})^T - I)^{-1} - (A_H - I)^{-1}\right) b_K \qquad (s \geq 1). \qquad (8.41)
$$

Proof. If we start from $p_o = 0$, then $e_o = -A_K^{-1} b_K$, as the exact solution p^* is given by $p^* = A_K^{-1} b_K$. From equation (8.35b) we get

$$\begin{aligned} e_{s+1} &= -(I - (\hat{K}_s')^{-1} A_K) A_K^{-1} b_K && (8.42) \\ &= ((\hat{K}_s')^{-1} - A_K^{-1}) b_K. && (8.43) \end{aligned}$$

As $\hat{K}_s' = A_H \mathcal{L}_s^{pH} (\mathcal{L}_s^{pH})^T - I$ and $A_H - I = A_K$, equation (8.41) has been proven.

As $p_{s+1} = p^* + e_{s+1}$ we get

$$p_{s+1} = (A_H - I)^{-1} b_K + ((\hat{K}_s')^{-1} - (A_H - I)^{-1}) b_K, \qquad (8.44)$$

from which (8.40) follows. □

Note that (8.41) can also be written, from (8.43) as

$$e_{s+1} = \left((\hat{K}_s')^{-1} - (K'(p_s))^{-1} \right) b_K, \qquad (8.45)$$

where $(K'(p_s))^{-1} = A_K^{-1}$.

Corollary 8.2.2. *Consider the IQN-LS method (§6.3.1). Assume that $\hat{K}_s'$ is non-singular and that K is an affine mapping. Then $e_1 = A_H e_o$, $r_1 = A_H r_o$ and for $s \geq 1$, there exists $\{\gamma_{1,s+1}, \gamma_{2,s+1}, \ldots, \gamma_{s,s+1}\} \subset \mathbb{R}$, such that*

$$\begin{aligned} e_{s+1} &= A_H e_o + A_H \sum_{i=1}^{s} \gamma_{i,s+1} (e_i - e_o) && (8.46) \\ r_{s+1} &= A_H r_o + A_H \sum_{i=1}^{s} \gamma_{i,s+1} (r_i - r_o). && (8.47) \end{aligned}$$

Proof. From theorem 8.2.3 it follows that

$$\begin{aligned} e_{s+1} &= e_o - (\hat{K}_s')^{-1} (A_H - I) e_o \\ (A_H \mathcal{L}_s^{pH} (\mathcal{L}_s^{pH})^T - I)(e_{s+1} - e_o) &= -(A_H - I) e_o \\ e_{s+1} &= A_H \mathcal{L}_s^{pH} (\mathcal{L}_s^{pH})^T (e_{s+1} - e_o) + A_H e_o. \end{aligned}$$

As $\mathcal{L}_s^{pH} (\mathcal{L}_s^{pH})^T$ is a projection operator on the span of $\{e_1 - e_o, e_2 - e_o, \ldots, e_s - e_o\}$, the latter expression can be written as

$$e_{s+1} = A_H e_o + A_H \sum_{i=1}^{s} \gamma_{i,s+1} (e_i - e_o), \qquad (8.48)$$

which proves (8.46).
To prove (8.47) we start from (8.46), and re-arrange it as

$$\begin{aligned}
e_{s+1} &= A_K e_o + e_o + A_K \sum_{i=1}^{s} \gamma_{i,s+1}(e_i - e_o) + \sum_{i=1}^{s} \gamma_{i,s+1}(e_i - e_o) \\
&= r_o + e_o + \sum_{i=1}^{s} \gamma_{i,s+1}(r_i - r_o) + \sum_{i=1}^{s} \gamma_{i,s+1}(e_i - e_o) \\
A_K e_{s+1} = r_{s+1} &= A_K r_o + r_o + A_K \sum_{i=1}^{s} \gamma_{i,s+1}(r_i - r_o) + \sum_{i=1}^{s} \gamma_{i,s+1}(r_i - r_o) \\
&= A_H r_o + A_H \sum_{i=1}^{s} \gamma_{i,s+1}(r_i - r_o),
\end{aligned}$$

which completes our proof. □

We will now show that IQN-LS shows some similarities with Krylov subspace methods (definition 2.15) as at the s-th iterate we have $p_s \in \mathcal{Y}_s$, with $\mathcal{Y}_s = \mathcal{K}_s\{A_K; r_o\}$, which is the same search subspace as GMRes; on the other hand we have $r_s \perp \mathcal{Z}_{s-1} = (A_H^T)^{-1}\mathcal{K}_{s-1}\{A_K; r_o\}$, which is different from GMRes where $r_s \perp A_K \mathcal{K}_s\{A_K; r_o\}$.

Theorem 8.2.4. *Consider the IQN-LS method (§6.3.1) and assume K is an affine mapping. Assume $\hat{K}'_{s-1}$ is non-singular. Then we have that*

$$e_s \in e_o + \mathcal{K}_s\{A_K; r_o\} \tag{8.49a}$$
$$p_s \in p_o + \mathcal{K}_s\{A_K; r_o\} \tag{8.49b}$$
$$r_s \in r_o + A_K \mathcal{K}_s\{A_K; r_o\}. \tag{8.49c}$$

Proof. Let $\mathcal{P}_k = \{q(x) \in \mathbb{R}[x] : q(x) = \sum_{i=1}^{k} \kappa_i x^i\}$, i.e. the space of real polynomials of degree k, or lower, with zero constant. $q(A_K)$ represents a polynomial in A_K, i.e $q(A_K) = \sum_{i=1}^{k} \kappa_i A_K^i$ for $q(x) = \sum_{i=1}^{k} \kappa_i x^i$.
We first note that $\mathcal{P}_k$ over $\mathbb{R}$ is a vector-space of dimension k, and that as such

$$\forall t_1(x), t_2(x) \in \mathcal{P}_k, \forall \alpha_1, \alpha_2 \in \mathbb{R} : \alpha_1 t_1(x) + \alpha_2 t_2(x) \in \mathcal{P}_k,$$

and that $\forall l \leq k : \mathcal{P}_l \subset \mathcal{P}_k$.
We will now give our proof by induction.

We know that

$$e_1 \quad = \quad A_H e_o = e_o + (A_H - I)e_o = e_o + q_1(A_K)e_o, \tag{8.50}$$

where $q_1 \in \mathcal{P}_1$ and $A_K = A_H - I$. We know (from corollary 8.2.2) that

$$\begin{aligned} e_2 &= A_H \sum_{i=1}^{1} \gamma_{i,2}(e_i - e_o) + A_H e_o & (8.51)\\ &= A_H \gamma_{1,2}(e_1 - e_o) + A_H e_o & (8.52)\\ &= A_H \gamma_{1,2} e_1 - A_H \gamma_{1,2} e_o + A_H e_o & (8.53) \end{aligned}$$

As $e_1 = A_H e_o$ we obtain

$$\begin{aligned} e_2 &= \gamma_{1,2} A_H A_H e_o - A_H \gamma_{1,2} e_o + A_H e_o & (8.54)\\ &= \gamma_{1,2} A_H (A_H - I)e_o + A_H e_o & (8.55)\\ &= \gamma_{1,2} A_H A_K e_o + \gamma_{1,2}(A_K e_o - A_K e_o) + A_H e_o + (e_o - e_o) & (8.56)\\ &= \gamma_{1,2} A_K A_K e_o + \gamma_{1,2} A_K e_o + A_K e_o + e_o & (8.57)\\ &= \gamma_{1,2} A_K^2 e_o + (\gamma_{1,2} + 1) A_K e_o + e_o & (8.58)\\ &= e_o + q_2(A_K)e_o, & (8.59) \end{aligned}$$

where $q_2 \in \mathcal{P}_2$. We now prove that, if we have

$$e_k \quad = \quad e_o + q_k(A_K)e_o, \tag{8.60}$$

for $k = 1, 2, \dots s-1$, where $q_k \in \mathcal{P}_k$, it follows that

$$e_s \quad = \quad e_o + q_s(A_K)e_o, \tag{8.61}$$

where $q_s \in \mathcal{P}_s$.

We have (from corollary 8.2.2) that

$$\begin{aligned} e_s &= A_H e_o + A_H \sum_{k=1}^{s-1} \gamma_{k,s}(e_k - e_o) & (8.62)\\ &= A_H e_o + A_H \sum_{k=1}^{s-1} \gamma_{k,s}(q_k(A_K)e_o). & (8.63) \end{aligned}$$

Knowing that $\forall k \leq s-1 : q_k \in \mathcal{P}_k \Rightarrow q_k \in \mathcal{P}_{s-1}$ and since $\mathcal{P}_{s-1}$ is a vector-space over $\mathbb{R}$, we can write

$$e_s \quad = \quad A_H \tilde{q}_{s-1}(A_K)e_o + A_H e_o, \tag{8.64}$$

where $\tilde{q}_{s-1} \in \mathcal{P}_{s-1}$. We also have that

$$e_s = (A_K)\tilde{q}_{s-1}(A_K)e_o + \tilde{q}_{s-1}(A_K)e_o + (A_K)e_o + e_o. \tag{8.65}$$

As $\forall q(x) \in \mathcal{P}_k : xq(x) \in \mathcal{P}_{k+1}$ and as $x \in \mathcal{P}_{k+1}$ we can finally write

$$e_s = e_o + q_s(A_K)e_o, \tag{8.66}$$

where $q_s \in \mathcal{P}_s$.
Thus $e_{s+1} \in e_o + \text{span}\{A_K e_o, A_K^2 e_o, A_K^3 e_o, \ldots, A_K^s e_o\}$.
Noting that
$\text{span}\{A_K e_o, A_K^2 e_o, A_K^3 e_o, \ldots, A_K^s e_o\} = \text{span}\{r_o, A_K r_o, A_K^2 r_o, \ldots, A_K^{s-1} r_o\}$
we have proven (8.49a).
Equations (8.49b) and (8.49c) follow immediately. □

Theorem 8.2.5. *Consider the IQN-LS method (§6.3.1) and assume K is an affine mapping. Assume $\hat{K}'_{s-1}$ is non-singular. Then we have that $r_s \perp (A_H^T)^{-1}\mathcal{K}_{s-1}\{A_K; r_o\}$.*

Proof. From corollary 7.2.3 we know that $r_s = A_H \bar{L}_s^{pH}(\bar{L}_s^{pH})^T \delta p_{s-1}$.
If we write $(\bar{L}_s^{pH})^T \delta p_{s-1} = \kappa \in \mathbb{R}$ then we have $r_s = \kappa A_H \bar{L}_s^{pH}$.
As $\forall y \in \mathcal{R}(V_{s-1}^{pH}) : \langle \bar{L}_s^{pH}, y \rangle = 0$, it follows that

$$\forall y \in \mathcal{R}(V_{s-1}^{pH}) : \langle r_s, (A_H^T)^{-1} y \rangle = 0. \tag{8.67}$$

From the definition of V_s^{pH} and equation (8.49b), we see that $\mathcal{R}(V_{s-1}^{pH}) = \mathcal{K}_{s-1}\{A_K; r_o\}$ if V_{s-1}^{pH} is of rank $s-1$.
r_s is thus orthogonal to $(A_H^T)^{-1}\mathcal{K}_{s-1}\{A_K; r_o\}$. □

Remark 8.1. *As r_s is only orthogonal to an $s-1$-dimensional subspace, it is not a Krylov subspace method in the classical sense.*

8.2.3 Krylov subspaces for IQN-ILS

We will show that the iterates of this method share the same Krylov search subspace as those of GMRes (and hence of IQN-LS), but not the subspace of constraints.

Theorem 8.2.6. *Consider the IQN-ILS method (§6.3.2). Assume that $\hat{M}'_s$ is nonsingular*[3] *and that K is an affine mapping. Then the following relations hold:*

$$\begin{aligned}\forall j \in \{0,1,\ldots,s\} :&\\ (I - A_K\hat{M}'_s)(r_s - r_j) &= 0 && \text{(8.68a)}\\ e_{s+1} &= e_j - \hat{M}'_s A_K e_j && \text{(8.68b)}\\ r_{s+1} &= r_j - A_K\hat{M}'_s r_j && \text{(8.68c)}\\ p_{s+1} &= p_j - \hat{M}'_s K(p_j). && \text{(8.68d)}\end{aligned}$$

Proof. (The proof is similar to the one in theorem 8.2.3.)
As, from theorem 6.2.3, $\forall j \in \{0,1,\ldots,s\} : \mathcal{R}(V_j^{KH}) \subset \mathcal{R}(V_s^{KH})$, lemma 2.3.3 allows us to write, using equation (8.34), $\forall x \in \mathcal{R}(V_j^{KH}),\ j = 0,1,\ldots,s$:

$$\begin{aligned}\hat{M}'_s x &= A_K^{-1}x && \text{(8.69)}\\ (I - A_K\hat{M}'_s)x &= 0. && \text{(8.70)}\end{aligned}$$

As $r_s - r_j = K(p_s) - K(p_j)$ we have $(r_s - r_j) \in \mathcal{R}(V_s^{KH})$ from the definition of V_s^{KH} and the theorem follows. □

From this theorem we see that the previous iterates only contribute to the solution process by creating a better approximate inverse Jacobian $\hat{M}'_s$, just as for the IQN-LS method.

Corollary 8.2.3. *Consider the IQN-ILS method (§6.3.2). Assume that $\hat{M}'_s$ is nonsingular, that K (or H) is an affine mapping and that the iterations start from an initial guess $p_o = 0$. Then the quasi-Newton iteration can also be written as*

$$p_{s+1} = ((A_K^{-1} + I)\mathcal{L}_s^{KH}(\mathcal{L}_s^{KH})^T - I)b_K \qquad (s \geq 1), \qquad \text{(8.71)}$$

[3] As shown in theorem 8.1.2, this assumption is always satisfied for linear systems.

and the error e_{s+1} is given by

$$e_{s+1} = (A_K^{-1} + I)\left(\mathcal{L}_s^{KH}(\mathcal{L}_s^{KH})^T - I\right) b_K \qquad (s \geq 1). \tag{8.72}$$

The proof is analogous to the one in corollary 8.2.1.

Note that (8.72) can also be written as

$$e_{s+1} = \left(\hat{M}_s' - (K'(p_s))^{-1}\right) b_K, \tag{8.73}$$

where $(K'(p_s))^{-1} = A_K^{-1}$.
Comparing this with $e_{s+1} = \left((\hat{K}_s')^{-1} - (K'(p_s))^{-1}\right) b_K$ for IQN-LS and taken into account that $\hat{M}_s'$ converges in a monotone manner towards A_K^{-1} for IQN-ILS, whereas $(\hat{K}_s')^{-1}$ for IQN-LS does not, would indicate that IQN-ILS would most likely exhibit better convergence.

Corollary 8.2.4. *Consider the IQN-ILS method (§6.3.2). Assume that $\hat{M}_s'$ is non-singular and that K is an affine mapping. Then $e_1 = A_H e_o$, $r_1 = A_H r_o$ and for $s \geq 1$, there exists $\{\gamma_{1,s+1}, \gamma_{2,s+1}, \ldots, \gamma_{s,s+1}\} \subset \mathbb{R}$, such that*

$$r_{s+1} \;=\; A_H r_o + A_H \sum_{i=1}^{s} \gamma_{i,s+1}(r_i - r_o) \tag{8.74}$$

$$e_{s+1} \;=\; A_H e_o + A_H \sum_{i=1}^{s} \gamma_{i,s+1}(e_i - e_o). \tag{8.75}$$

Proof. From theorem 8.2.6 it follows that

$$\begin{aligned}
r_{s+1} &= r_o - A_K \hat{M}_s' r_o && (8.76)\\
&= r_o - A_K\left[(A_K + I)A_K^{-1}\mathcal{L}_s^{KH}(\mathcal{L}_s^{KH})^T - I\right] r_o && (8.77)\\
&= r_o - (\mathcal{L}_s^{KH}(\mathcal{L}_s^{KH})^T + A_K(I - \mathcal{L}_s^{KH}(\mathcal{L}_s^{KH})^T))r_o && (8.78)\\
&= (A_K + I)(I - \mathcal{L}_s^{KH}(\mathcal{L}_s^{KH})^T)r_o && (8.79)\\
&= A_H r_o - A_H \mathcal{L}_s^{KH}(\mathcal{L}_s^{KH})^T r_o. && (8.80)
\end{aligned}$$

As $\mathcal{L}_s^{KH}(\mathcal{L}_s^{KH})^T$ is a projection operator on span$\{r_1 - r_o, r_2 - r_o, \ldots, r_s - r_o\}$, the latter expression can be written as

$$r_{s+1} \;=\; A_H r_o + A_H \sum_{i=1}^{s} \gamma_{i,s+1}(r_i - r_o). \tag{8.81}$$

To prove (8.75) it suffices to multiply both sides of (8.81) by A_K^{-1} and re-arrange the terms as follows:

$$\begin{aligned} r_{s+1} &= (A_K+I)r_o + (A_K+I)\sum_{i=1}^{s}\gamma_{i,s+1}(r_i-r_o) \\ A_K^{-1}r_{s+1} = e_{s+1} &= A_K^{-1}(A_K+I)r_o + A_K^{-1}(A_K+I)\sum_{i=1}^{s}\gamma_{i,s+1}(r_i-r_o) \\ &= (I+A_K^{-1})r_o + (I+A_K^{-1})\sum_{i=1}^{s}\gamma_{i,s+1}(r_i-r_o) \\ &= (I+A_K^{-1})A_K e_o + (I+A_K^{-1})A_K\sum_{i=1}^{s}\gamma_{i,s+1}(e_i-e_o) \\ &= A_H e_o + A_H\sum_{i=1}^{s}\gamma_{i,s+1}(e_i-e_o), \end{aligned}$$

which completes our proof. □

We will now show that IQN-ILS shows some similarities with Krylov subspace methods (definition 2.15) as at the s-th iterate we have $p_s \in \mathcal{Y}_s = \mathcal{K}_s\{A_K; r_o\}$, which is the same search subspace as GMRes; on the other hand we have $r_s \perp \mathcal{Z}_{s-1}$, where $\mathcal{Z}_{s+1} = (A_H^T)^{-1}A_K\mathcal{K}_{s-1}\{A_K; r_o\}$, which is different from GMRes and IQN-LS.

Theorem 8.2.7. *Consider the IQN-ILS method (§6.3.2) and assume K is an affine mapping. Assume $\hat{M}'_{s-1}$ is non-singular. Then we have that*

$$\begin{aligned} e_s &\in e_o + \mathcal{K}_s\{A_K; r_o\} && \text{(8.82a)} \\ p_s &\in p_o + \mathcal{K}_s\{A_K; r_o\} && \text{(8.82b)} \\ r_s &\in r_o + A_K\mathcal{K}_s\{A_K; r_o\}. && \text{(8.82c)} \end{aligned}$$

Proof. Let $\mathcal{P}_k = \{q(x) \in \mathbb{R}[x] : q(x) = \sum_{i=1}^{k}\kappa_i x^i\}$, i.e. the space of real polynomials of degree k, or lower, with zero constant. $q(A_K)$ represents a polynomial in A_K, i.e $q(A_K) = \sum_{i=1}^{k}\kappa_i A_K^i$ for $q(x) = \sum_{i=1}^{k}\kappa_i x^i$.
We first note that $\mathcal{P}_k$ over $\mathbb{R}$ is a vector-space of dimension k, and that as such

$$\forall t_1(x), t_2(x) \in \mathcal{P}_k, \forall\alpha_1,\alpha_2 \in \mathbb{R} : \alpha_1 t_1(x) + \alpha_2 t_2(x) \in \mathcal{P}_k,$$

and that $\forall l \leq k : \mathcal{P}_l \subset \mathcal{P}_k$.
We will now give our proof by induction.

We know that

$$e_1 = A_H e_o = e_o + (A_H - I)e_o = e_o + q_1(A_K)e_o, \qquad (8.83)$$

where $q_1 \in \mathcal{P}_1$ and $A_K = A_H - I$. We know (from corollary 8.2.4) that

$$\begin{aligned} e_2 &= A_H \sum_{i=1}^{1} \gamma_{i,2}(e_i - e_o) + A_H e_o && (8.84)\\ &= A_H \gamma_{1,2}(e_1 - e_o) + A_H e_o && (8.85)\\ &= \gamma_{1,2}(A_H - I)^2 e_o + (1 + \gamma_{1,2})(A_H - I)e_o + e_o && (8.86)\\ &= e_o + q_2(A_K)e_o, && (8.87) \end{aligned}$$

where $q_2 \in \mathcal{P}_2$. (The derivation is similar to the one in theorem 8.2.4.) We now prove that, if we have

$$e_k = e_o + q_k(A_K)e_o, \qquad (8.88)$$

for $k = 1, 2, \ldots s - 1$, where $q_k \in \mathcal{P}_k$, it follows that

$$e_s = e_o + q_s(A_K)e_o, \qquad (8.89)$$

where $q_s \in \mathcal{P}_s$.

We have (from corollary 8.2.4) that

$$\begin{aligned} e_s &= A_H e_o + A_H \sum_{k=1}^{s-1} \gamma_{k,s}(e_k - e_o) && (8.90)\\ &= A_H e_o + A_H \sum_{k=1}^{s-1} \gamma_{k,s}(q_k(A_K)e_o). && (8.91) \end{aligned}$$

Knowing that $\forall k \leq s - 1 : q_k \in \mathcal{P}_k \Rightarrow q_k \in \mathcal{P}_{s-1}$ and since $\mathcal{P}_{s-1}$ is a vector-space over $\mathbb{R}$, we can write

$$e_s = A_H \tilde{q}_{s-1}(A_K)e_o + A_H e_o, \qquad (8.92)$$

where $\tilde{q}_{s-1} \in \mathcal{P}_{s-1}$. We also have that

$$e_s = (A_H - I)\tilde{q}_{s-1}(A_K)e_o + \tilde{q}_{s-1}(A_K)e_o + (A_K)e_o + e_o. \qquad (8.93)$$

As $\forall q(x) \in \mathcal{P}_k : xq(x) \in \mathcal{P}_{k+1}$ and as $x \in \mathcal{P}_{k+1}$ we can finally write

$$e_s = e_o + q_s(A_K)e_o, \tag{8.94}$$

where $q_s \in \mathcal{P}_s$.
Thus $e_{s+1} \in e_o + \text{span}\{A_K e_o, A_K^2 e_o, A_K^3 e_o, \dots, A_K^s e_o\}$.
Noting that
$\text{span}\{A_K e_o, A_K^2 e_o, A_K^3 e_o, \dots, A_K^s e_o\} = \text{span}\{r_o, A_K r_o, A_K^2 r_o, \dots, A_K^{s-1} r_o\}$ we have proven (8.82a).
Equations (8.82b) and (8.82c) follow immediately. □

Theorem 8.2.8. *Consider the IQN-ILS method (§6.3.2) and assume K is an affine mapping. Assume $\hat{M}'_{s-1}$ is non-singular. Then we have that $r_s \perp (A_H^T)^{-1} A_K \mathcal{K}_{s-1}\{A_K; r_o\}$.*

Proof. From (4.9) we know that

$$r_s = r_{s-1} - A_K \hat{M}'_{s-1} r_{s-1}, \tag{8.95}$$

and hence

$$\begin{aligned} r_s &= r_{s-1} - A_K(A_H A_K^{-1} \mathcal{L}_{s-1}^{KH} (\mathcal{L}_{s-1}^{KH})^T - I) r_{s-1} && (8.96)\\ &= r_{s-1} - ((I + A_K)\mathcal{L}_{s-1}^{KH} (\mathcal{L}_{s-1}^{KH})^T - A_K) r_{s-1} && (8.97)\\ &= (I + A_K)(I - \mathcal{L}_{s-1}^{KH} (\mathcal{L}_{s-1}^{KH})^T) r_{s-1} && (8.98)\\ &= A_H(I - \mathcal{L}_{s-1}^{KH} (\mathcal{L}_{s-1}^{KH})^T) r_{s-1}. && (8.99) \end{aligned}$$

Hence $r_s = A_H z$, with $z = (I - \mathcal{L}_{s-1}^{KH} (\mathcal{L}_{s-1}^{KH})^T) r_{s-1} \in (\mathcal{R}(V_{s-1}^{KH}))^\perp$.
As $\forall x \in (\mathcal{R}(V_{s-1}^{KH}))^\perp, \forall y \in \mathcal{R}(V_{s-1}^{KH}) : \langle x, y \rangle = 0$, it follows that

$$\forall y \in \mathcal{R}(V_{s-1}^{KH}) : \langle r_s, (A_H^T)^{-1} y \rangle = 0. \tag{8.100}$$

From the definition of V_s^{KH} and equation (8.82c), we see that
$\mathcal{R}(V_{s-1}^{KH}) = A_K \mathcal{K}_{s-1}\{A_K; r_o\}$ if V_{s-1}^{KH} is of rank $s-1$.
r_s is thus orthogonal to $(A_H^T)^{-1} A_K \mathcal{K}_{s-1}\{A_K; r_o\}$. □

Remark 8.2. *As r_s is only orthogonal to an $s-1$-dimensional subspace, it is not a Krylov subspace method in the classical sense.*

8.2.4 Further discussion

While IQN-LS and IQN-ILS share the same Krylov search subspace for the iterates we need to stress some subtle differences.

First of all, IQN-LS and IQN-ILS were developed based on "function calls" which in the case of affine mappings correspond to $H(p) = A_H p - b_H = K(p) + p$, while GMRes has been developed based on matrix vector products $A_K p$. Variants of GMRes exist that work with function calls $H(p)$ or $K(p)$ instead of matrix-vector products (e.g. §3.2.2.2 [218]), but these exhibit poor numerical stability. Experience has shown that the most stable way to use GMRes based on function calls is to compute $b_K (= b_H)$ explicitly, thus requiring an extra function call.

As we have shown, p_{n+1} (or an earlier iterate) will be the solution of the system for both IQN-LS and IQN-ILS (starting from p_o), at which point $n+1$ function calls will have been spent. For GMRes p_n (or an earlier iterate [222]) will be the solution. Nevertheless, at that point we will also have used $n+1$ function calls, as the residual $r_n = K(p_n)$ needs to be computed[4]. In IQN-LS and IQN-ILS, however, we have no knowledge of the final residual r_{n+1}, even though we are sure that, in exact arithmetic, it is zero. If we were to compute it, for convergence verification when working in finite precision, then an extra function call would need to be spent.

Combining this last argument with the one before we see that, when we base our methods on function calls, GMRes needs at most $n+2$ function calls (for a stable version) and so do IQN-LS and IQN-ILS (if we want to verify the final residual).

8.3 The effect of step-length parameters

We could consider modifying the basic quasi-Newton iteration (4.4), resp. (4.5), with a step-length parameter $\omega_s \in \mathbb{R}$:

$$p_{s+1} = p_s - \omega_s (\hat{K}_s')^{-1} r_s, \tag{8.101}$$

resp.

$$p_{s+1} = p_s - \omega_s \hat{M}_s' r_s. \tag{8.102}$$

The resulting error and residual equations are

$$\begin{aligned} e_{s+1} &= e_s - \omega_s (\hat{K}_s')^{-1} A_K e_s && (8.103) \\ r_{s+1} &= r_s - \omega_s A_K (\hat{K}_s')^{-1} r_s, && (8.104) \end{aligned}$$

[4] It is this variable that is optimized.

resp.

$$
\begin{aligned}
e_{s+1} &= e_s - \omega_s \hat{M}'_s A_K e_s && (8.105)\\
r_{s+1} &= r_s - \omega_s A_K \hat{M}'_s r_s. && (8.106)
\end{aligned}
$$

We now show that the effect of the step-length parameter is limited to the current iterate when K is an affine mapping.

Theorem 8.3.1. *Let K be an affine mapping. Consider the IQN-LS method (§6.3.1), using a relaxation factor as in equation (8.101), then the choice of $\omega_s \in \mathbb{R} \setminus \{0\}$ is irrelevant for the value of p_{s+2}*[5].

Proof. We have, by posing $e_s = e_{s,\|} + e_{s,\perp}$, where $e_{s,\|} \in \mathcal{R}(V_s^{pH})$, $e_{s,\perp} \in (\mathcal{R}(V_s^{pH}))^{\perp}$ and using theorem 8.2.3, that

$$
\begin{aligned}
e_{s+1} &= e_s - \omega_s (\hat{K}'_s)^{-1} A_K e_s && (8.107)\\
&= e_{s,\|} + e_{s,\perp} - \omega_s (\hat{K}'_s)^{-1} A_K e_{s,\perp} - \omega_s e_{s,\|} && (8.108)\\
&= (1-\omega_s) e_{s,\|} + e_{s,\perp} - \omega_s (\hat{K}'_s)^{-1} A_K e_{s,\perp} && (8.109)\\
&= (1-\omega_s) e_{s,\|} + (1-\omega_s) e_{s,\perp} - \omega_s (\hat{K}'_s)^{-1} A_H e_{s,\perp} && (8.110)\\
&= \underbrace{(1-\omega_s)\left(e_{s,\|} + e_{s,\perp} + (\hat{K}'_s)^{-1} A_H e_{s,\perp}\right)}_{(*)} - \underbrace{(\hat{K}'_s)^{-1} A_H e_{s,\perp}}_{(**)}. && (8.111)
\end{aligned}
$$

The new column vector added to update V_s^{pH} to V_{s+1}^{pH} will depend on $\delta p_s = p_{s+1} - p_s = \delta e_s = e_{s+1} - e_s$:

$$
\delta e_s = -\omega_s \left(e_{s,\|} + e_{s,\perp} + (\hat{K}'_s)^{-1} A_H e_{s,\perp}\right). \qquad (8.112)
$$

We can thus conclude that the direction of δe_s is independent of ω_s. We see that e_{s+1} has a part that is parallel to δe_s (part $(*)$ in equation (8.111)) and a remaining part that is independent of ω_s (part $(**)$).
$(*)$ will be eliminated completely at the next iteration, according to theorem 8.2.3, as it lies in $\mathcal{R}(V_{s+1}^{pH})$. We can thus conclude that ω_s will have no effect at the next iteration. □

This theorem shows that, for linear problems, line-searches, which are often part of a (quasi-) Newton method, do not improve the long-term convergence of our

[5] The value of p_{s+1} will be affected however.

algorithm, but can improve the instantaneous convergence. While classical line-searches [6,91,202] require supplemental function evaluations (which we consider to be very expensive), we will discuss a cheap and easy alternative in §8.4.
This does not mean that the use of a relaxation parameter might not be beneficial in the non-linear case. Nevertheless, methods that determine the relaxation parameter based on extra function calls might not be economical due to the high cost of these function calls. For that reason we will only use a fixed relaxation parameter in the first iteration, where it is meant to avoid excessive initial divergence that might impair the later convergence.

Remark 8.3. *Theorem 8.3.1 extends to p_1 by setting $\hat{K}_o' = -I$.*

Remark 8.4. *Similar properties hold for IQN-ILS (equation (8.102)) and by analogy with IQN-LS also for IQN-CLS and IBQN-LS.*

8.4 Modifying IQN-LS and IQN-ILS to make them algebraically identical to GMRes

8.4.1 Re-writing the quasi-Newton algorithm for matrix-vector products

As we have shown in §8.2, IQN-LS and IQN-ILS (and by extension IQN-CLS and IBQN-LS) share the same Krylov search subspace for the iterates if the operators are affine. This means that, theoretically, we can modify the Least Squares quasi-Newton methods to make them algebraically identical to GMRes, if we can find a suitable linear combination of the basis vectors of the search subspace.
In this section we will show that this is indeed possible when we can form $A_K x$, $\forall x \in \mathbb{R}^{n\times 1}$, as opposed to $K(x)$, for those quasi-Newton methods. We will also show that this can be done without the need for supplementary matrix-vector products (which are the equivalent of function calls in this context).
(In chapter 11 we will verify this claim with the help of numerical experiments.)

To do this, we first re-write the general quasi-Newton method given in (8.101), resp. and (8.102), as in algorithm 8.4.1, resp. 8.4.2, based on ideas in [55].

Algorithm 8.4.1 (Alternative form 1 of the IQN algorithms with approximate Jacobian)**.**

```
1. Startup.
   Take a starting value $p_o$;
   compute $r_o = A_K p_o - b_K$;
   choose $\hat{K}'_o$.
   Set $s = 0$.
2. Loop until sufficiently converged
   a. $\Delta_s = (\hat{K}'_s)^{-1} r_s$
   b. $p_{s+1} = p_s - \omega_s \Delta_s$
   c. $q_s = A_K \Delta_s$
   d. $r_{s+1} = r_s - \omega_s q_s$
   e. $\hat{K}'_{s+1} = \hat{K}'_s + \frac{(q_s - r_s) c_s^T}{\langle \Delta_s, c_s \rangle}$
   f. Set $s = s + 1$.
```

Algorithm 8.4.2 (Alternative form 1 of the IQN algorithms with approximate inverse Jacobian)**.**

```
1. Startup.
   Take a starting value $p_o$;
   compute $r_o = A_K p_o - b_K$;
   choose $\hat{M}'_o$.
   Set $s = 0$.
2. Loop until sufficiently converged
   a. $\Delta_s = \hat{M}'_s r_s$
   b. $p_{s+1} = p_s - \omega_s \Delta_s$
   c. $q_s = A_K \Delta_s$
   d. $r_{s+1} = r_s - \omega_s q_s$
   e. $\hat{M}'_{s+1} = \hat{M}'_s - \frac{\hat{M}'_s (q_s - r_s) d_s^T}{\langle q_s, d_s \rangle}$
   f. Set $s = s + 1$.
```

(We assume $\omega_s \neq 0$ with $s = 0, 1, 2, \ldots.$)
The definitions of c_s and d_s in these algorithms are the same as those used in §4.4 and define the specific quasi-Newton method.

For IQN-LS we use the following in algorithm 8.4.1:

- $\hat{K}'_o = -I$.
- If $s = 0$: $c_s = \bar{L}^{pH}_{s+1} = \frac{\Delta_s}{\|\Delta_s\|}$;

else: $c_s = \bar{L}_{s+1}^{pH} = \frac{(I-\mathcal{L}_s^{pH}(\mathcal{L}_s^{pH})^T)\Delta_s}{\|(I-\mathcal{L}_s^{pH}(\mathcal{L}_s^{pH})^T)\Delta_s\|}$.

- If $s = 0$: $\mathcal{L}_{s+1}^{pH} = \bar{L}_{s+1}^{pH}$;
  else: $\mathcal{L}_{s+1}^{pH} = [\mathcal{L}_s^{pH}|\bar{L}_{s+1}^{pH}]$.

For IQN-LS we use the following in algorithm 8.4.2:

- $\hat{M}'_o = -I$.
- If $s = 0$: $d_s = \bar{L}_{s+1}^{KH} = \frac{q_s}{\|q_s\|}$;
  else: $d_s = \bar{L}_{s+1}^{KH} = \frac{(I-\mathcal{L}_s^{KH}(\mathcal{L}_s^{KH})^T)q_s}{\|(I-\mathcal{L}_s^{KH}(\mathcal{L}_s^{KH})^T)q_s\|}$.
- If $s = 0$: $\mathcal{L}_{s+1}^{KH} = \bar{L}_{s+1}^{KH}$;
  else: $\mathcal{L}_{s+1}^{KH} = [\mathcal{L}_s^{KH}|\bar{L}_{s+1}^{KH}]$.

We recall that using the parameter $\omega_s(\neq 0)$ does not change the search subspace and the long-term convergence, but can improve the instantaneous convergence of the algorithm (cfr. §8.3).
Setting $\omega_s = 1$ in both algorithms yields the standard IQN-LS and IQN-ILS methods in rank-one update form.

8.4.2 Optimal step-length

If we use the formulation of the IQN-LS method given in algorithm 8.4.1 we can find the value of ω_s that minimizes r_s in the Euclidean norm. To do so, we impose

$$r_{s+1} \quad \perp \quad q_s, \tag{8.113}$$

with $r_{s+1} = r_s - \omega_s q_s$. This leads to

$$\begin{aligned} \langle r_{s+1}, q_s\rangle = \langle r_s - \omega_s q_s, q_s\rangle &= 0 \\ \omega_s &= \frac{\langle r_s, q_s\rangle}{\langle q_s, q_s\rangle}. \end{aligned} \tag{8.114}$$

For IQN-ILS we obtain the same expression.

If we compare the resulting variant of IQN-LS and IQN-ILS with GMRes, we see that both still share the same Krylov spaces for p_s and r_s. When using the optimal step-length parameter in (8.114) IQN-LS and IQN-ILS search for the smallest residual r_s in $r_o + A_K\mathcal{K}_s\{A_K; r_o\}$, but only along the direction q_{s-1}. Even though this will improve convergence (in exact arithmetic), it will in general result in a larger residual than for GMRes, as the latter searches for the smallest residual in $r_o + A_K\mathcal{K}_s\{A_K; r_o\}$, but in all directions contained in that subspace.

8.4.3 Multiple parameters

In §8.4.1 and §8.4.2 we have only added a single iteration parameter to the quasi-Newton methods. While an optimal value for that parameter will result in better convergence[6], it does not allow us to obtain algorithms that are algebraically equivalent to GMRes.

If we want to obtain an algorithm that is algebraically equivalent to GMRes, we should add multiple parameters, as done in algorithm 8.4.3.

Algorithm 8.4.3 (Alternative form 2 of the IQN algorithms)**.**

As algorithms 8.4.1 and 8.4.2 , but with

2. b. $p_{s+1} = p_s - \sum_{i=o}^{s} \theta_{s,i}\Delta_i$

2. d. $r_{s+1} = r_s - \sum_{i=o}^{s} \theta_{s,i}q_i$

(We assume $\theta_{s,s} \neq 0$ with $s = 0, 1, 2, \ldots.$)

We now show, in the following theorem, that adding these parameters does not change the search subspace (i.e. the subspace in which the iterates are found).

Theorem 8.4.1. *The IQN-LS method implemented as in algorithm 8.4.3 retains the same search subspace as the unmodified algorithm (which can be obtained by setting $\theta_{s,s} = 1$ and $\theta_{s,i\neq s} = 0$, for $s = 0, 1, \ldots$).*

Proof. We will proof this theorem recursively, based on the results of theorem 8.3.1.

We see that

$$p_1 = p_o - \theta_{o,o}(\hat{K}'_o)^{-1}K(p_o) \tag{8.115}$$

$$\delta p_o = -\theta_{o,o}(\hat{K}'_o)^{-1}K(p_o) \tag{8.116}$$

corresponds to the form of theorem 8.3.1 by setting $\omega_o = \theta_{o,o}$.

For p_2 we write

$$p_2 = p_1 - \theta_{1,o}(\hat{K}'_o)^{-1}K(p_o) - \theta_{1,1}(\hat{K}'_1)^{-1}K(p_1) \tag{8.117}$$

$$p_2 = \underbrace{p_1 - \theta_{1,1}(\hat{K}'_1)^{-1}K(p_1)}_{(*)} + \frac{\theta_{1,o}}{\theta_{o,o}}\delta p_o. \tag{8.118}$$

[6] As always, in exact arithmetic.

We can apply theorem 8.3.1 to the part marked $(*)$ by setting $\omega_1 = \theta_{1,1}$. We obtain

$$e_2 = \underbrace{(1-\theta_{1,1})(e_1 + (\hat{K}_1')^{-1}A_H e_{1,\perp})}_{(**)} - \underbrace{(\hat{K}_1')A_H e_{1,\perp}}_{(****)} + \underbrace{\frac{\theta_{1,o}}{\theta_{o,o}}\delta e_o}_{(***)} \tag{8.119}$$

$$\delta e_1 = -\theta_{1,1}(e_1 + (\hat{K}_1')^{-1}A_H e_{1,\perp}) + \frac{\theta_{1,o}}{\theta_{o,o}}\delta e_o. \tag{8.120}$$

We thus see that e_2 has a part that is parallel to δe_1 (marked $(**)$), a part parallel to δe_o (marked $(***)$), and a part that is independent of the iteration parameters (marked $(****)$). As $(**)$ and $(***)$ will be eliminated in the next iteration, we see that there is no long term effect of the iteration parameters.
We also see that δe_o and δe_1 are linear combinations of the values of δe_o and δe_1 that would be obtained by setting $\theta_{s,s} = 1$ and $\theta_{s,i\neq s} = 0$, for $s = 0, 1, \ldots$ (i.e. the unmodified algorithm). This means that the subspace spanned by δe_o and δe_1 is not altered.
Applying this reasoning recursively proves the theorem. □

Remark 8.5. *A similar proof holds for the IQN-ILS algorithm.*

We now compute the optimal values of the parameters, i.e. those hat minimize the residual in the Euclidean norm. We define $\Theta_s = [\theta_{s,o}\ \theta_{s,1}\ \ldots\ \theta_{s,s}]^T$ and impose

$$r_{s+1} \perp q_i \tag{8.121}$$

($i = 0, 1, \ldots s$). By analogy with (8.114), this leads to

$$\Theta_s = (Q^TQ)^{-1}Q^Tr_s, \tag{8.122}$$

where $Q = [q_o\,|q_1\,|\ldots\,|q_s]$.

As $\{\Delta_i\}_{i=o}^s$ span the same Krylov subspace as the unmodified algorithm (cfr. theorem 8.4.3), they form a basis for the Krylov subspace $\mathcal{K}_s\{A_K; r_o\}$. As A_K is assumed to be non-singular, it follows that $\{q_i\}_{i=o}^s$ form a basis for the Krylov subspace $A_K\mathcal{K}_s\{A_K; r_o\}$ to which r_{s+1} is now orthogonal. It follows that the IQN-LS and IQN-ILS method, implemented as in algorithm 8.4.3, are algebraically identical to GMRes (cfr §3.2.2).

Note that the modifications to obtain algorithms 8.4.1, 8.4.2 and 8.4.3 do not demand extra matrix-vector products, and only conceptually differ from the original formulation in their ability to form A_Kx for all $x \in \mathbb{R}^{n\times 1}$ as opposed to $A_Kx - b_K$.

9
Improving the Jacobian for discretized time-dependent and grid-based problems

Sometimes we are presented with a series of related problems, for instance time-related problems; this was briefly mentioned in §1.1.3.
We will write

$$
\begin{aligned}
F_t(g\,; p_{t-1}, g_{t-1}) &= p && (9.1a)\\
S_t(p\,; p_{t-1}, g_{t-1}) &= g, && (9.1b)
\end{aligned}
$$

or

$$
K_t(p\,; p_{t-1}, g_{t-1}) = 0, \qquad (9.2)
$$

($t = 1, 2, \ldots$) for this type of problem. (The subscript denotes the problem within the series, i.e. most often the time-level.)
By this we mean: “solve (9.1) for p and g, with known values of g_{t-1} and p_{t-1}”, resp. “solve (9.2) for p with known values of g_{t-1} and p_{t-1}”. p_{t-1} and g_{t-1} are the values of p and g at the previous time-level; if exact values are not available we will use the final values of the iterative process used to solve the problem at the

previous time-level.
The solution of (9.1) or (9.2) will give the values of p and/or g at the new time-level (p_t, resp. g_t).
When we solve these equations iteratively, the iterates at iteration s for time-level t will be written as $p_{s,t}$ and $g_{s,t}$, etc.

While there are many problems of different origins that can be written in this form, we assume that from here on we are dealing with discretized time-dependent ordinary or partial differential equations, and that the subscript t denotes the time-level for which we are solving the resulting algebraic system.

9.1 Recovery of data from previous time-levels

9.1.1 Recovery methods based on input-output pairs

The following method can be applied to all of the Least Squares quasi-Newton methods when using the formulation of §6.3 (i.e. when not in rank-one update form). We will use IQN-LS as an example.

If we assume that the input-output pairs of previous time-levels are representative enough for the current time-level, we might think of enhancing the Jacobian by adding these to the formulation of §6.3. For the proposed method we might re-use the data as follows.

$$\mathbf{V}_{s,t}^{pH} = [V_{s,t}^{pH} \mid V_{final,t-1}^{pH} \mid \ldots \mid V_{final,t-\tau}^{pH}]$$

$$\mathbf{W}_{s,t}^{pH} = [W_{s,t}^{pH} \mid W_{final,t-1}^{pH} \mid \ldots \mid W_{final,t-\tau}^{pH}],$$

where $V_{s,t}^{pH}$ and $W_{s,t}^{pH}$ are constructed at the current time-level t and current iteration s as in (6.26), $V_{final,t-i}^{pH}$ and $W_{final,t-i}^{pH}$ ($i = 1, \ldots, \tau$) are the input and output matrices constructed as in (6.26) at the end of the iteration process at time-level $t - i$ and τ is a parameter that determines how many time-levels are kept.

The Jacobian at iteration s of time-level t is then constructed as in (6.25):

$$\hat{K}'_{s,t} \quad = \quad \mathbf{W}_{s,t}^{pH}(\mathbf{V}_{s,t}^{pH})^{+} - I. \tag{9.3}$$

Experience has shown that creating input-output pairs by computing the difference over different time-levels (e.g. $p_{o,t} - p_{final,t-1}$) is not a good idea. One reason being that the difference taken over different time-levels is partially a time-derivative,

which is not present in the Jacobian we are approximating.
We start the first iteration of a new time-level ($t > 1$) by computing
$p_{1,t} = p_{o,t} - (\hat{K}'_{final,t-1})^{-1} K_t(p_{o,t})$,
where $\hat{K}'_{final,t-1}$ is the approximate Jacobian at the last iteration of the previous time-level; this means we set $\hat{K}'_{o,t} = K'_{final,t-1}$. (For the first time-level we implicitly used $\hat{K}'_{o,1} = -I$, possibly combined with under-relaxation.) A first input-output pair for the new time-level can then be computed based on $p_{1,t} - p_{o,t}$ and $H_t(p_{1,t}) - H_t(p_{o,t})$.
$p_{o,t}$ is obtained by linear extrapolation based on $p_{final,t-1}$ and $p_{final,t-2}$ (if available), which are the last iterates at time step $t-1$ and $t-2$ respectively[1].

We will now establish some properties that result from this construction. The first is an extension of theorem 7.1.1, showing that the method can still be written in rank-one update form.

Theorem 9.1.1. *Suppose that $\hat{K}'_{s,t}$ is constructed as in (9.3), then $\hat{K}'_{s+1,t}$ is linked to $\hat{K}'_{s,t}$ by the following expression (for $s < n$):*

$$\forall \mathring{A}_H \in \mathbb{A}(\mathbf{V}^{pH}_{s+1,t}, \mathbf{W}^{pH}_{s+1,t}) : \hat{K}'_{s+1,t} = \hat{K}'_{s,t} + \mathring{A}_H \bar{L}^{pH}_{s+1,t} (\bar{L}^{pH}_{s+1,t})^T, \qquad (9.4)$$

where $\bar{L}^{pH}_{s+1,t}$ is the $(s+1)$-th column of $\mathbf{L}^{pH}_{s+1,t}$ and $\mathbf{L}^{pH}_{s+1,t}$ is a matrix of which the columns form an orthonormal basis for $\mathbf{V}^{pH}_{s+1,t}$.

The proof is similar to the one of theorem 7.1.1.

Just as in §7.1.1 we can re-write (9.4) so that the rank-one update can be computed from available data.
The new basis vector $\bar{L}^{pH}_{s+1,t}$ can be computed using

$$\bar{L}^{pH}_{s+1,t} = \frac{\delta p_{s,t} - \mathbf{L}^{pH}_{s,t} (\mathbf{L}^{pH}_{s,t})^T \delta p_{s,t}}{\|\delta p_{s,t} - \mathbf{L}^{pH}_{s,t} (\mathbf{L}^{pH}_{s,t})^T \delta p_{s,t}\|}, \qquad (9.5)$$

which can be done based on the available data ($\delta p_{s,t} = p_{s+1,t} - p_{s,t}$), while for the computation of $\mathring{A}_H \bar{L}^{pH}_{s+1,t}$ we can use the following relationships.

[1] We assume that at the last iteration convergence has been reached.

$$
\begin{aligned}
\mathring{A}_H \bar{L}^{pH}_{s+1,t} &= \mathring{A}_H \frac{\delta p_{s,t} - \mathbf{L}^{pH}_{s,t}(\mathbf{L}^{pH}_{s,t})^T \delta p_{s,t}}{\|\delta p_{s,t} - \mathbf{L}^{pH}_{s,t}(\mathbf{L}^{pH}_{s,t})^T \delta p_{s,t}\|} \\
&= \frac{\mathring{A}_H \delta p_{s,t} - \mathring{A}_H \mathbf{L}^{pH}_{s,t}(\mathbf{L}^{pH}_{s,t})^T \delta p_{s,t}}{\|\delta p_{s,t} - \mathbf{L}^{pH}_{s,t}(\mathbf{L}^{pH}_{s,t})^T \delta p_{s,t}\|} \\
&= \frac{\delta H_{s,t} - \mathring{A}_H \mathbf{V}^{pH}_{s,t}(\mathbf{V}^{pH}_{s,t})^T \delta p_{s,t}}{\|\delta p_{s,t} - \mathbf{L}^{pH}_{s,t}(\mathbf{L}^{pH}_{s,t})^T \delta p_{s,t}\|} \\
&= \frac{\delta H_{s,t} - \mathring{A}_H \mathbf{V}^{pH}_{s,t}(\mathbf{V}^{pH}_{s,t})^T \delta p_{s,t} - \delta p_{s,t} + \delta p_{s,t}}{\|\delta p_{s,t} - \mathbf{L}^{pH}_{s,t}(\mathbf{L}^{pH}_{s,t})^T \delta p_{s,t}\|} \\
&= \frac{\delta H_{s,t} + \delta p_{s,t} - (\mathring{A}_H \mathbf{V}^{pH}_{s,t}(\mathbf{V}^{pH}_{s,t})^T - I) \delta p_{s,t}}{\|\delta p_{s,t} - \mathbf{L}^{pH}_{s,t}(\mathbf{L}^{pH}_{s,t})^T \delta p_{s,t}\|} \\
&= \frac{\delta K_{s,t} - \hat{K}'_{s,t} \delta p_{s,t}}{\|\delta p_{s,t} - \mathbf{L}^{pH}_{s,t}(\mathbf{L}^{pH}_{s,t})^T \delta p_{s,t}\|} \\
&= \frac{K_t(p_{s+1,t}) - K_t(p_{s,t}) - \hat{K}'_{s,t} \delta p_{s,t}}{\|\delta p_{s,t} - \mathbf{L}^{pH}_{s,t}(\mathbf{L}^{pH}_{s,t})^T \delta p_{s,t}\|},
\end{aligned}
$$

where $\delta K_{s,t} = K_t(p_{s+1,t}) - K_t(p_{s,t})$.

We can write the resulting update as follows.

$$
\hat{K}'_{s+1,t} = \hat{K}'_{s,t} + \frac{(\delta K_{s,t} - \hat{K}'_{s,t} \delta p_{s,t})((I - \mathbf{L}^{pH}_{s,t}(\mathbf{L}^{pH}_{s,t})^T) \delta p_{s,t})^T}{\langle \delta p_{s,t}, (I - \mathbf{L}^{pH}_{s,t}(\mathbf{L}^{pH}_{s,t}))^T) \delta p_{s,t} \rangle}, \quad (9.6)
$$

which can be simplified to

$$
\hat{K}'_{s+1,t} = \hat{K}'_{s,t} + \frac{K_t(p_{s+1,t})((I - \mathbf{L}^{pH}_{s,t}(\mathbf{L}^{pH}_{s,t})^T) \delta p_{s,t})^T}{\langle \delta p_{s,t}, (I - \mathbf{L}^{pH}_{s,t}(\mathbf{L}^{pH}_{s,t})^T) \delta p_{s,t} \rangle}. \quad (9.7)
$$

Equation (9.7) shows that the Jacobian does not change in the orthogonal complement of the space spanned by $(I - \mathbf{L}^{pH}_{s,t}(\mathbf{L}^{pH}_{s,t})^T) \delta p_{s,t}$, where $(I - \mathbf{L}^{pH}_{s,t}(\mathbf{L}^{pH}_{s,t})^T) \delta p_{s,t}$ is the direction of the component of $\delta p_{s,t}$ that is orthogonal to all previously visited directions (including those of previous time-levels). In other words

$$
\hat{K}'_{s+1,t} z = \hat{K}'_{s,t} z \ \text{ when } z \perp (I - \mathbf{L}^{pH}_{s,t}(\mathbf{L}^{pH}_{s,t})^T) \delta p_{s,t} \quad (9.8)
$$

$(I - \mathbf{L}^{pH}_{s,t}(\mathbf{L}^{pH}_{s,t})^T) \delta p_{s,t}$

Some caution is needed when using this method.

- The choice of the parameter τ is difficult, as it is not always clear a priori how many time-levels can be kept, i.e. how long old data will be representative for the problem at the current time-level.
- If we add a column to $\mathbf{V}^{pH}_{s,t}$ which is linearly dependent on other columns, then the method will break down. This problem can be solved by *QR-filtering* to $\mathbf{V}^{pH}_{s,t}$, as explained in §7.5.1.

Remark 9.1. *When the operators are affine, e.g.* $H_t(p) = A_{H,t}p - b_{H,t}$ *etc., then in general we will have* $A_{H,j} \notin \mathbb{A}(\mathbf{V}^{pH}_{s,t}, \mathbf{W}^{pH}_{s,t})$*, for* $j = 1, 2, \ldots, t$*, even though* $A_{H,t} \in \mathbb{A}(V^{pH}_{s,t}, W^{pH}_{s,t})$.

Remark 9.2. *A similar approach with similar results exists for IQN-ILS, IQN-CLS and IBQN-LS.*

9.1.2 Recovery methods based on the rank-one update formulation

Another approach to re-using data is based on the rank-one update formulation established in §7.1. This can be applied to all quasi-Newton methods in rank-one update-form, e.g. the Least Squares methods, Broyden's methods, CUM and ICUM [105]. Again we will use IQN-LS as an example.

In this approach we will keep the final approximate Jacobian of the previous time-level $\hat{K}'_{final,t-1}$ and use it as $\hat{K}'_{o,t}$ (cfr. as in §9.1.1). If the Jacobian of the previous time-level is representative enough for the current time-level than this will result in a better initial Jacobian, which improves convergence.
This means that for the first iteration of the new time-level we have
$p_{1,t} = p_{o,t} - (\hat{K}'_{o,t})^{-1}K_t(p_{o,t}) = p_{o,t} - (\hat{K}'_{final,t-1})^{-1}K_t(p_{o,t})$.
Again, $p_{o,t}$ is obtained by linear extrapolation based on $p_{final,t-1}$ and $p_{final,t-2}$ (if available).
Starting with the input-output pairs $p_{1,t} - p_{o,t}$ and $H_t(p_{1,t}) - H_t(p_{o,t})$ we will apply a rank-one update to $\hat{K}'_o$ after the manner described in §7.1:

$$\begin{aligned}\hat{K}'_{s+1,t} &= \hat{K}'_{s,t} + \frac{(\delta K_{s,t} - \hat{K}'_{s,t}\delta p_{s,t})((I - \mathcal{L}^{pH}_{s,t}(\mathcal{L}^{pH}_{s,t})^T)\delta p_{s,t})^T}{\langle \delta p_{s,t}, (I - \mathcal{L}^{pH}_{s,t}(\mathcal{L}^{pH}_{s,t})^T)\delta p_{s,t}\rangle} && (9.9)\\ &= \hat{K}'_{s,t} + \frac{K_t(p_{s+1,t})((I - \mathcal{L}^{pH}_{s,t}(\mathcal{L}^{pH}_{s,t})^T)\delta p_{s,t})^T}{\langle \delta p_{s,t}, (I - \mathcal{L}^{pH}_{s+1,t}(\mathcal{L}^{pH}_{s+1,t})^T)\delta p_{s,t}\rangle}, && (9.10)\end{aligned}$$

where $\mathcal{L}^{pH}_{s,t}$ is a matrix of which the columns form an orthonormal basis for $V^{pH}_{s+1,t}$, although $V^{pH}_{s+1,t}$ is not explicitly constructed. We recall that $V^{pH}_{s+1,t}$ only contains input pairs of the current time-level as in (6.26).

Equations (9.7) and (9.10) show the main difference between the method in §9.1.1 and the method in this section.
From equation (9.7) we have concluded that the update only happens in the direction of the component of $\delta p_{s,t}$ that is orthogonal to all previous directions, including those of previous time-levels, while (9.10) teaches us that for this method the update happens in the direction of the component of $\delta p_{s,t}$ that is orthogonal to all previous directions of the current time-level only.
Thus for the method of this section the directions in the current time-level have a larger influence on the approximate Jacobian than in the method of section 9.1.1. Also, if the direction $\delta p_{s,t}$ has already occurred in a previous time-level then the old data will be simply over-written, while for the method in §9.1.1 a singularity would occur, which needs to be removed by QR-filtering.

A theorem similar to theorem 7.1.1 is no longer valid, i.e. we do not have that

$$\forall \mathring{A}_H \in \mathbb{A}(V^{pH}_{s+1,t}, W^{pH}_{s+1,t}) : \hat{K}'_{s+1,t} = \hat{K}'_{s,t} + \mathring{A}_H \bar{L}^{pH}_{s+1,t} (\bar{L}^{pH}_{s+1,t})^T, \qquad (9.11)$$

for $s < n$, where $\hat{K}'_{s,t}$ is constructed as in the IQN-LS method described in this section and where $\bar{L}^{pH}_{s+1,t}$ is the $(s+1)$-th column of $\mathcal{L}^{pH}_{s+1,t}$. ($\mathcal{L}^{pH}_{s+1,t}$ is a matrix of which the columns form an orthonormal basis for $V^{pH}_{s+1,t}$.)
This can be shown by a simple example.
Assume $K_t(p) = A_{K,t}p - b_{K,t}$, $H_t(p) = A_{H,t}p - b_{H,t}$, then we have

$$\begin{aligned}
\hat{K}'_{1,t} &= \hat{K}'_{o,t} + \frac{(\delta K_{o,t} - \hat{K}'_{o,t}\delta p_{o,t})(\delta p_{o,t})^T}{\langle \delta p_{o,t}, \delta p_{o,t}\rangle} \\
&= \hat{K}'_{o,t} + \frac{(A_{K,t} - \hat{K}'_{o,t})\delta p_{o,t}}{\|\delta p_{o,t}\|} \left(\frac{\delta p_{o,t}}{\|\delta p_{o,t}\|}\right)^T \\
&= \hat{K}'_{o,t} + (A_{K,t} - \hat{K}'_{o,t}) \bar{L}^{pH}_{1,t} (\bar{L}^{pH}_{1,t})^T.
\end{aligned}$$

As in general $\hat{K}'_{o,t} \neq -I$, and thus $A_{K,t} - \hat{K}'_{o,t} \neq A_{H,t}$ we have that
$\hat{K}'_{1,t} \neq \hat{K}'_{o,t} + A_{H,t}\bar{L}^{pH}_{1,t}(\bar{L}^{pH}_{1,t})^T$, with $A_{H,t} \in \mathbb{A}(V^{pH}_{s+1,t}, W^{pH}_{s+1,t})$.
Nevertheless, the (9.10) clearly shows that the method is still a rank-one update method.

Remark 9.3. *A similar approach with similar results exists for IQN-ILS, IQN-CLS and IBQN-LS.*

Remark 9.4. *The extension of this approach to Broyden's good and bad method, the Column-Updating Method and the Inverse Column-Updating method (chapter 4) is straightforward.*

9.2 Creating a better initial approximate Jacobian from a coarser grid

When the problem that needs to be solved is a discretized (partial) differential equation defined on a mesh, we can solve the same problem on a coarser mesh and use the input-output pairs on the coarse mesh to create an initial Jacobian on the fine mesh [105]. The method presented below can only be applied to the Least Squares methods. (Note that the problem to be solved not necessarily needs to be time-dependent.)

After solving the problem on the coarse grid, the input-output modes created during the solution process on that grid are prolongated to the fine grid and used to create the initial Jacobian on the fine grid after which the normal rank-one updates are performed. The prolongation is shown schematically in figure 9.1.
We illustrate this for the construction of $\hat{K}'_o$ in the IQN-LS method.

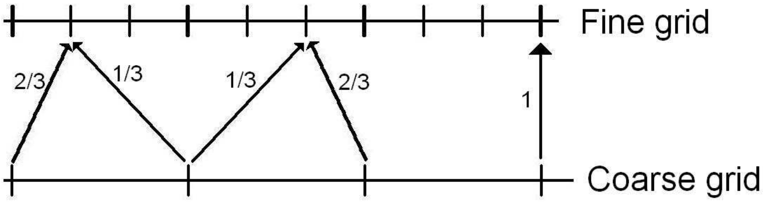

Figure 9.1: Schematic representation of prolongation from coarse to fine grid.

Let $\Im_c^f$ be the prolongation (interpolation) matrix used to go from the coarse to the fine grid. We construct the initial approximate Jacobian $\hat{K}'_o$ on the fine grid as

$$\hat{K}'_o = \Im_c^f W_{final}^{pH,c}[(\Im_c^f V_{final}^{pH,c})^T \Im_c^f V_{final}^{pH,c}]^{-1}(\Im_c^f V_{final}^{pH,c})^T - I, \tag{9.12}$$

where $V_{final}^{pH,c}$ and $W_{final}^{pH,c}$ are defined as in (6.9) at the end of the solution process on the coarse grid.

Afterwards the usual rank-one update is applied (cfr. §7.1).

The initial iterate on the fine grid can be obtained by prolongation of the final solution on the coarse grid or by extrapolating the solutions of the previous time-levels when the problem is time-dependent (just as in §9.1.1 and §9.1.2).

We recall that we are only considering values p and g on the interface between the domains of the interacting problems described by the functions F and S. The number of variables on the interface might be several orders lower than the (internal) variables involved in computing $F(g)$ or $S(p)$. Computations with a reduced number of interface variables will thus be substantially cheaper, hence the construction of the initial approximate Jacobian in this manner is relatively cheap.

The main difficulty with this method is the choice of the ratio of the number of variables on the coarse and fine grids, which is a trade-off between computational cost and accuracy.

Remark 9.5. *A similar approach with similar results exists for IQN-ILS, IQN-CLS and IBQN-LS.*

10

Numerical experiments with non-affine operators

We have so far mainly focused on the theoretical properties of the various quasi-Newton methods. In this chapter we will test these methods on two well-understood test-cases: one-dimensional flow in a flexible tube and the one-dimensional heat equation with variable coefficients.

In the former, the interaction is between the pressure and velocity of a fluid and the geometry of a structure. In the latter, the temperature interacts with density, heat capacity and thermal conductivity.

For the fluid-structure interaction problem we perform a detailed Fourier analysis and show the conditional stability of a simple fixed point iteration method.

Both problems are solved with approaches discussed in chapters 5, 6 and 9.

10.1 One-dimensional flow in a flexible tube

10.1.1 Preliminary remark about notations

In this section we will use continuous variables, discretized variables, non-dimensional discretized variables and Fourier coefficients. To avoid confusion we give a brief overview of the notation that will be used.

$\mathfrak{p}$, $\mathfrak{u}$, $\mathfrak{g}$	:	continuous variables: pressure, velocity, cross-sectional area;
P_{t+1}, P_t, etc.	:	discretized variables (vectors) at resp. time-step $t+1$ and t: pressure, etc.;
$[P_{t+1}]_i$, $[P_t]_i$, etc.	:	i-th component of discretized variables at resp. time-step $t+1$ and t: pressure, etc.;
$[P]_{in}$, $[P]_{out}$, etc.	:	discretized pressure at inlet, resp. outlet, etc.;
p_{t+1}, p_t, etc.	:	discretized non-dimensional variables (vectors) at resp. time-step $t+1$ and t: pressure, etc.;
$[p_{t+1}]_i$, $[p_t]_i$, etc.	:	i-th component of non-dimensional discretized variables at resp. time-step $t+1$ and t: pressure, etc.;
$p_{s,t+1}$, $p_{s,t}$, etc.	:	discretized non-dimensional variables (vectors) at resp. time-step $t+1$ and t and s-th iteration: pressure, etc.;
$\hat{p}_s$	:	error component in Fourier analysis (vector);
$[\hat{p}_s]_i$	:	i-th component of $\hat{p}_s$;
$[\tilde{p}_s]^l$	:	amplitude of l-th Fourier mode at iteration s;
p^*, u^*, g^*	:	solution for discretized, non-dimensional equation: pressure, velocity, cross-sectional area.

10.1.2 Analytical description of the problem

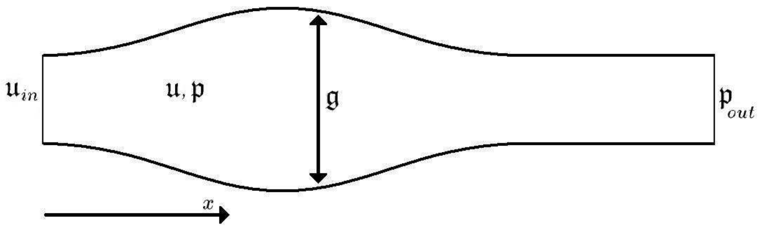

Figure 10.1: One-dimensional flow in a flexible tube.

This test-case describes one-dimensional unsteady flow in a flexible tube of length L[1].

The fluid is incompressible and inviscid and gravity is neglected. The governing equations are the conservation of mass and momentum, which can be written in

[1] This can be thought of as a much simplified model of pulsating flow in an artery.

conservative form as

$$\frac{\partial \mathfrak{g}}{\partial t} + \frac{\partial \mathfrak{g}\mathfrak{u}}{\partial x} = 0 \tag{10.1a}$$

$$\frac{\partial \mathfrak{g}\mathfrak{u}}{\partial t} + \frac{\partial \mathfrak{g}\mathfrak{u}^2}{\partial x} + \frac{1}{\rho}\left(\frac{\partial \mathfrak{g}\tilde{\mathfrak{p}}}{\partial x} - \tilde{\mathfrak{p}}\frac{\partial \mathfrak{g}}{\partial x}\right) = 0, \tag{10.1b}$$

with $\mathfrak{g}$ the cross-sectional area of the tube, $\mathfrak{u}$ the velocity along the axis of the tube and $\frac{\partial}{\partial t}$ the time derivative. x is the spatial coordinate, ρ is the density of the fluid and $\tilde{\mathfrak{p}}$ the pressure. We will write $\mathfrak{p} = \tilde{\mathfrak{p}}/\rho$ for the kinematic pressure.

If the elastic wall of the tube has a constitutive law of the form $\mathfrak{g} = \mathfrak{g}(\mathfrak{p})$, with the cross-sectional area only a function of the local kinematic pressure[2] and if its inertia is neglected then (10.1) can be rewritten in the following form

$$\frac{\partial \mathfrak{p}}{\partial t} + \mathfrak{u}\frac{\partial \mathfrak{p}}{\partial x} + c^2\frac{\partial \mathfrak{u}}{\partial x} = 0 \tag{10.2a}$$

$$\frac{\partial \mathfrak{g}\mathfrak{u}}{\partial t} + \frac{\partial \mathfrak{g}\mathfrak{u}^2}{\partial x} + \frac{\partial \mathfrak{g}\mathfrak{p}}{\partial x} - \mathfrak{p}\frac{\partial \mathfrak{g}}{\partial x} = 0, \tag{10.2b}$$

where the wave speed c is defined by

$$c^2 = \frac{\mathfrak{g}}{\frac{\mathrm{d}\mathfrak{g}}{\mathrm{d}\mathfrak{p}}}. \tag{10.3}$$

The velocity at the inlet of the tube is imposed as

$$\mathfrak{u}(t) \quad = \quad \mathfrak{u}_o + \frac{\mathfrak{u}_o}{10}\sin^2(\pi t), \tag{10.4}$$

where $\mathfrak{u}_o$ is a reference velocity.

A non-reflecting boundary condition is prescribed at the outlet:

$$\frac{\partial \mathfrak{u}}{\partial t} = \frac{1}{c}\frac{\partial \mathfrak{p}}{\partial t}. \tag{10.5}$$

The behavior of the flexible tube wall is described with a Hookean constitutive relation. The structure model contains no mass, as the inertia of the tube wall is neglected with regards to that of the fluid.

An axisymmetric model is used in the coordinate system (x,$\mathfrak{r}$,ϕ), with $\mathfrak{r}$ the inner radius of the tube and ϕ the angle in the cross-sectional plane. The stress in the tube wall in the circumferential direction $\sigma_{\phi\phi}$ is approximated as

$$\sigma_{\phi\phi} = E\frac{\mathfrak{r} - \mathfrak{r}_o}{\mathfrak{r}_o} + \sigma_o, \tag{10.6}$$

[2] We will use a long-standing abuse of notation in that $\mathfrak{p}$ and $\mathfrak{g}$ will both refer to functions as to the corresponding pressure and geometrical variables. Similarly we will be writing both $\mathfrak{g}$ for the function $\mathfrak{g}_1 : \mathbb{R} \to \mathbb{R} : \mathfrak{p} \mapsto \mathfrak{g}_1(\mathfrak{p})$ and $\mathfrak{g}_2 : \mathbb{R}^2 \to \mathbb{R} : (x,t) \mapsto \mathfrak{g}_2(x,t)$; the latter meaning $\mathfrak{g}_1 \circ \mathfrak{p}$ with $\mathfrak{p} : \mathbb{R}^2 \to \mathbb{R} : (x,t) \mapsto \mathfrak{p}(x,t)$.

with E Young's modulus, $\mathfrak{r}_o$ the radius where $\sigma_{\phi\phi} = \sigma_o$ and σ_o a reference value. Other stress components are assumed to be zero. This model allows only radial motion of the tube wall.

Under the assumption that only pressure forces act on the fluid-structure interface, the force balance reads

$$\tilde{\mathfrak{p}}\mathfrak{r} = \sigma_{\phi\phi} h, \tag{10.7}$$

with h the thickness of the tube wall.

By substituting (10.6) and the definition of the kinematic pressure in (10.7), the following relation is obtained:

$$\mathfrak{r}\mathfrak{p} = \frac{Eh}{\rho \mathfrak{r}_o}(\mathfrak{r} - \mathfrak{r}_o) + \mathfrak{r}_o \mathfrak{p}_o, \tag{10.8}$$

with $\mathfrak{p}_o$ and $\tilde{\mathfrak{p}}_o$ defined by $\tilde{\mathfrak{p}}_o \mathfrak{r}_o = \sigma_o h$. This can be rewritten as

$$\mathfrak{g} = \mathfrak{g}_o \left(\frac{\mathfrak{p}_o - 2c_{mk}^2}{\mathfrak{p} - 2c_{mk}^2} \right)^2 \tag{10.9}$$

by using $\mathfrak{g} = \pi \mathfrak{r}^2$ (and $\mathfrak{g}_o = \pi \mathfrak{r}_o$) and by introducing the constant c_{mk} (the Moens-Korteweg wave speed), given by

$$c_{mk}^2 = \frac{Eh}{2\rho \mathfrak{r}_o}. \tag{10.10}$$

The wave speed according to definition (10.3) thus becomes

$$c^2 = c_{mk}^2 - \frac{\mathfrak{p}}{2}. \tag{10.11}$$

10.1.3 Discretizing the equations

The flow equations (10.1) are discretized on a one-dimensional equidistant mesh with n cells and mesh size Δx. The fluid velocity and pressure are stored in the mesh nodes. Central discretization of all terms in the continuity and momentum equations is used, except for the convective term in the momentum equation which is discretized with a first-order upwind scheme. The time discretization scheme is backward Euler and the time-step is indicated with Δt. The conservation of mass and momentum in a control volume around node i is expressed by the following system of equations:

$$\frac{\Delta x}{\Delta t}\left([G]_i - [G_t]_i\right) + [U]_{i+1/2}[G]_{i+1/2} - [U]_{i-1/2}[G]_{i-1/2} = 0 \tag{10.12a}$$

$$\begin{aligned}\frac{\Delta x}{\Delta t}\left([U]_i[G]_i - [U_t]_i[G_t]_i\right) + [U]_i[U]_{i+1/2}[G]_{i+1/2} - [U]_{i-1}[U]_{i-1/2}[G]_{i-1/2}\\ + [P]_{i+1/2}[G]_{i+1/2} - [P]_{i-1/2}[G]_{i-1/2} - [P]_i\left([G]_{i+1/2} - [G]_{i-1/2}\right) = 0,\end{aligned} \tag{10.12b}$$

for $[U]_i \geq 0$ ($i = 1, \ldots n$). The subscripts i, $i+1$ and $i-1$ indicate the mesh nodes ($i = 1, \ldots, n$)[3]. The subscript $i \pm 1/2$ signifies the values calculated at the cell interfaces, $[U]_{i-1/2} = 1/2([U]_{i-1} + [U]_i)$ and $[U]_{i+1/2} = 1/2([U]_i + [U]_{i+1})$, etc. The subscript t denotes the previous time-level; the subscript $t+1$ for the new time-level is omitted. A pressure stabilization term is added in the continuity equation (10.12a) to prohibit pressure wiggles due to central discretization of the pressure in the momentum equation (10.12b) :

$$\begin{aligned}\frac{\Delta x}{\Delta t}\left([G]_i - [G_t]_i\right) + [U]_{i+1/2}[G]_{i+1/2} - [U]_{i-1/2}[G]_{i-1/2}\\ - \alpha([P]_{i+1} - 2[P]_i + [P]_{i-1}) = 0\end{aligned} \tag{10.13a}$$

$$\begin{aligned}\frac{\Delta x}{\Delta t}\left([U]_i[G]_i - [U_t]_i[G_t]_i\right) + [U]_i[U]_{i+1/2}[G]_{i+1/2} - [U]_{i-1}[U]_{i-1/2}[G]_{i-1/2}\\ + \frac{1}{2}\left([G]_{i+1/2}([P]_{i+1} - [P]_i) + [G]_{i-1/2}([P]_i - [P]_{i-1})\right) = 0,\end{aligned} \tag{10.13b}$$

with $\alpha = \dfrac{G_o}{U_o + \frac{\Delta x}{\Delta t}}$. U_0 is the initial flow velocity.

The pressure at the inlet and the velocity at the outlet are linearly extrapolated from neighboring values as

$$[P]_{in} = 2[P]_1 - [P]_2 \tag{10.14a}$$

$$[U]_{out} = 2[U]_n - [U]_{n-1}. \tag{10.14b}$$

The velocity at the inlet is imposed and the pressure-condition at the outlet (equation (10.5)) is discretized as

$$[P]_{out} = 2\left(c_{mk}^2 - \left(\sqrt{c_{mk}^2 - \frac{[P_t]_{out}}{2}} - \frac{[U]_{out} - [U_t]_{out}}{4}\right)^2\right), \tag{10.15}$$

which takes into account the variation of c with $\mathfrak{p}$ given by (10.11) when integrating from time-level t to time-level $t+1$.

[3] The subscript "o" is used to indicate a reference value. No confusion should arise with nodal values which use a subscript "i" ($i = 1, \ldots, n$) as i cannot take the value 0. Hence, U_o should not be interpreted as he "0-th" node.

The geometrical discretization of the elastic problem is identical to that of the flow problem to avoid errors in the data transfer between the fluid and the structure:

$$[G]_i = G_o \left(\frac{P_o - 2c_{mk}^2}{[P]_i - 2c_{mk}^2} \right)^2 . \tag{10.16}$$

10.1.4 Non-dimensionalizing the equations

The non-dimensional parameters and nodal variables are defined as

$$\begin{array}{lll} [g]_i = \dfrac{[G]_i}{G_o} & [p]_i = \dfrac{[P]_i}{c_o^2} & \beta = \dfrac{\alpha c_o}{G_o} \\ [u]_i = \dfrac{[U]_i}{c_o} & u_o = \dfrac{U_o}{c_o} & D_o = \dfrac{\frac{\Delta x}{\Delta t}}{c_o}, \end{array}$$

for $i = 1, \ldots, n$, where u_o represents the Courant-Friedrichs-Lewy (CFL) number [44]. Note that $\beta = \frac{1}{uo+Do}$.

For (10.13a) the non-dimensionalized equation becomes (after division by $G_o c_o$), for $i = 1, \ldots, n$,

$$\begin{aligned} D_o \left([g]_i - [g_t]_i \right) + [u]_{i+1/2}[g]_{i+1/2} - [u]_{i-1/2}[g]_{i-1/2} & \\ -\beta([p]_{i+1} 2[p]_i - [p]_{i-1}) &= 0. \end{aligned} \tag{10.17}$$

For (10.13b) the non-dimensionalized equation becomes (after division by $G_o c_o^2$), for $i = 1, \ldots, n$,

$$\begin{aligned} D_o \left([u]_i [g]_i - [u_t]_i [g_t]_i \right) + [u]_i [u]_{i+1/2} [g]_{i+1/2} - [u]_{i-1} [u]_{i-1/2} [g]_{i-1/2} & \\ + \frac{1}{2} \big([g]_{i+1/2} ([p]_{i+1} - [p]_i) + [g]_{i-1/2} ([p]_i - [p]_{i-1}) \big) = 0. & \end{aligned} \tag{10.18}$$

For the right boundary condition (10.15) we obtain the following equation (after division by c_o^2), for $i = 1, \ldots, n$,

$$[p]_{out} = 2 \left(\frac{c_{mk}^2}{c_o^2} - \left(\sqrt{\frac{c_{mk}^2}{c_o^2} - \frac{[p_t]_{out}}{2}} - \frac{[u]_{out} - [u_t]_{out}}{4} \right)^2 \right).$$

As $\frac{c_{mk}^2}{c_o^2} = 1 + \frac{p_o}{2}$ (equation 10.11) this becomes

$$[p]_{out} = 2 + p_o - \left(\sqrt{2 + p_o - [p_t]_{out}} - [u]_{out} + [u_t]_{out} \right)^2 . \tag{10.19}$$

As $\mathfrak{p} = 2(c_{mk}^2 - c^2)$ and $\mathfrak{p}_o = 2(c_{mk}^2 - c_o^2)$ (equation 10.11), we obtain for the structural equation (10.16)

$$\begin{aligned}
[G]_i &= G_o \left(\frac{-2}{\frac{[P]_i}{c_o^2} - 2\frac{c_{mk}^2}{c_o^2}} \right)^2 \\
[g]_i &= g_o \left(\frac{-2}{[p]_i - 2\left(1 + \frac{p_o}{2}\right)} \right)^2 \\
[g]_i &= g_o \left(\frac{2}{2 + p_o - [p]_i} \right)^2 . \qquad (10.20)
\end{aligned}$$

10.1.5 Fourier error analysis

As already mentioned in §1.2.2, fixed-point iterations, like the Iterative Substructuring Method, are only conditionally stable.
We will illustrate this by a Fourier analysis of this one-dimensional fluid-structure interaction problem.

The Iterative Substructuring Method (algorithm 1.2.1) can be implemented as follows, based on the non-dimensional equations, with the first subscript indicating the coupling iteration and the second the time-level.

1. Solve the flow equations (10.17 - 10.18 - 10.19) for the velocity and pressure at time-level $t+1$ with a fixed geometry $g_{s,t+1}$; assign $u_{s+1,t+1}$ and $p_{s+1,t+1}$ to this solution.
   When we are only interested in the pressure, we can write this as $F_{t+1}(g_{s,t+1}, p_t, g_t) = p_{s+1,t+1}$, which corresponds to the definition of F_{t+1} used in (9.1a).

2. Compute the geometry at time-level $t+1$ from the structural equation (10.20) given the previously calculated pressure $p_{s+1,t+1}$; assign $g_{s+1,t+1}$ to this solution. We can write this as $S_{t+1}(p_{s+1,t+1}) = g_{s+1,t+1}$, which corresponds to the definition of S_{t+1} used in (9.1b); note that in this particular case S_{t+1} is no function of g_t or p_t.

3. Increase s and return to step 1 until convergence is obtained.

The stability of this simple iterative method is now investigated with Fourier analysis [44]. Every unknown in equations (10.17), (10.18) and (10.20) is written as the sum of the coupled solution (indicated with an asterisk) and the remaining error (indicated with a hat):

$$u_s = u^* + \hat{u}_s \tag{10.21a}$$
$$p_s = p^* + \hat{p}_s \tag{10.21b}$$
$$g_s = g^* + \hat{g}_s, \tag{10.21c}$$

where we have dropped the subscript $t+1$.
All non-linear combinations in the error terms are neglected and the equations satisfied by the coupled solution are subtracted from these equations. A constant velocity, pressure and section along the tube is chosen as a coupled solution:

$$u_i^* = u_o \tag{10.22a}$$
$$p_i^* = p_o \tag{10.22b}$$
$$g_i^* = g_o, \tag{10.22c}$$

for $i = 1, \dots, n$, with u_o the mean velocity and p_o and g_o defined previously. It is clear that (10.22) satisfies the equations for the coupled solution.
This results in the following equations for the error terms.

$$\begin{aligned} D_o[\hat{g}_s]_i + \frac{u_o}{2}\left([\hat{g}_s]_{i+1} - [\hat{g}_s]_{i-1}\right) + \frac{g_o}{2}\left([\hat{u}_{s+1}]_{i+1} - [\hat{u}_{s+1}]_{i-1}\right) \\ - \beta([\hat{p}_{s+1}]_{i+1} - 2[\hat{p}_{s+1}]_i + [\hat{p}_{s+1}]_{i-1}) = 0 \end{aligned} \tag{10.23a}$$

$$\begin{aligned} D_o\left(u_o[\hat{g}_s]_i + [\hat{u}_{s+1}]_i\right) + \frac{u_o^2}{2}\left([\hat{g}_s]_{i+1} - [\hat{g}_s]_{i-1}\right) \\ + u_o\left([\hat{u}_{s+1}]_{i+1/2} + [\hat{u}_{s+1}]_i - [\hat{u}_{s+1}]_{i-1/2} - [\hat{u}_{s+1}]_{i-1}\right) \\ + \frac{1}{2}\left([\hat{p}_{s+1}]_{i+1} - [\hat{p}_{s+1}]_{i-1}\right) = 0 \end{aligned} \tag{10.23b}$$

$$[\hat{g}_{s+1}]_i = [\hat{p}_{s+1}]_i. \tag{10.23c}$$

(Note that, according to our choice of non-dimensional parameters, $g_o = 1$.)
The error terms are expanded as an infinite sum of Fourier modes. As equations (10.23) are linear in the error terms, every Fourier mode with a given spatial pulsation ϖ_l $\left(\varpi_l = \frac{2\pi l}{n},\ l \in [-\frac{n}{2}, \frac{n}{2}]\right)$ can be studied separately[4]. The following substitutions are performed in equations (10.23):

$$[\hat{u}_s]_i \rightarrow [\tilde{u}_s]^l e^{\jmath \varpi_l i \Delta x} \tag{10.24a}$$
$$[\hat{p}_s]_i \rightarrow [\tilde{p}_s]^l e^{\jmath \varpi_l i \Delta x} \tag{10.24b}$$
$$[\hat{g}_s]_i \rightarrow [\tilde{g}_s]^l e^{\jmath \varpi_l i \Delta x}, \tag{10.24c}$$

[4] This definition of ϖ_l should not be confused with the relaxation parameter ω used in other chapters and is only used in this way in this section.

with $\jmath = \sqrt{-1}$ and l the index of the Fourier modes. With $\vartheta_l = \varpi_l \Delta x$, the following non-dimensional modal equations are obtained.

$$D_o[\tilde{g}_s]^l + u_o[\tilde{g}_s]^l \jmath \sin(\vartheta_l) + [\tilde{u}_{s+1}]^l \jmath \sin(\vartheta_l) - 2\beta[\tilde{p}_{s+1}]^l(\cos(\vartheta_l) - 1) = 0 \tag{10.25a}$$

$$\begin{aligned} D_o u_o[\tilde{g}_s]^l + D_o[\tilde{u}_{s+1}]^l + u_o^2[\tilde{g}_s]^l \jmath \sin(\vartheta_l) \\ + u_o[\tilde{u}_{s+1}]^l \left(\jmath \sin(\vartheta_l) + 1 - e^{-\jmath\vartheta_l}\right) + [\tilde{p}_{s+1}]^l \jmath \sin(\vartheta_l) = 0 \end{aligned} \tag{10.25b}$$

$$[\tilde{g}_{s+1}]^l = [\tilde{p}_{s+1}]^l. \tag{10.25c}$$

At every iteration the component of the error with spatial frequency ϖ_l for the cross-section and for the pressure is amplified by

$$\frac{[\tilde{g}_{s+1}]^l}{[\tilde{g}_s]^l} = \frac{[\tilde{p}_{s+1}]^l}{[\tilde{p}_s]^l} = 1 - \frac{u_o^2(1 - e^{-j\vartheta_l})j \sin\vartheta_l + b_1 D_o + b_2}{b_2}, \tag{10.26}$$

with $b_1 = D_o + u_o(1 + j\sin\vartheta_l - e^{-j\vartheta_l})$ and $b_2 = (\sin\vartheta_l)^2 - 2b_1\beta(\cos\vartheta_l - 1)$. This amplification factor is function of ϑ_l (and thus n), u_o and D_o only. In order to have a stable method the norm of this amplification factor should not be larger than one, i.e.

$$\mu_l = \left|\frac{[\tilde{g}_{s+1}]^l}{[\tilde{g}_s]^l}\right| = \left|\frac{[\tilde{p}_{s+1}]^l}{[\tilde{p}_s]^l}\right| \le 1, \tag{10.27}$$

for all $l \in [-\frac{n}{2}, \frac{n}{2}]$.

We introduce two new parameters:

$$\kappa = \frac{1}{u_o} = \frac{\sqrt{\frac{Eh}{2\rho r_o} - \frac{P_o}{2}}}{U_o} \quad \text{and} \quad \tau = \frac{U_o}{D_o n} = \frac{u_o \Delta t}{L}.$$

As we mainly vary E and Δt, these can be seen as a dimensionless structural stiffness and a dimensionless time-step respectively. The effect of the reference flow velocity U_o can be seen by modifying κ and τ such that $\kappa\tau$ remains constant. Approximate values for a human artery are $\kappa = 100$ and $\tau = 0.01$ [44].

Figures 10.2 and 10.3 show that the error amplification increases for decreasing ϑ_l, meaning that the lowest frequencies are those that are most unstable. The mode for $\vartheta_l = 0$ is always unstable. (Note that this mode is normally not present in actual

implementations due to the presence of boundary conditions.)
As can be seen in figure 10.2, the instability grows when the dimensionless stiffness κ decreases. For an infinitely stiff tube ($\kappa = \infty$ or $u_o = 0$) the amplification factor in equations (10.26) and (10.27) is zero for all Fourier modes, except $\vartheta_l = 0$. Figure 10.3 illustrates that a smaller value of τn (i.e. a smaller dimensionless timestep τ and/or a lower number of nodes n) increases the amount of instability.
While both τn and κ affect the stability of the iterations, the effect of κ is generally greater than that of τn. κ determines the vertical position of the curve while τn modifies both its shape and position. τn influences the stability significantly in the bottom graph of figure 10.3 where κ is small ($\kappa = 10$) but not in the top graph of figure 10.3. An increase of n by some factor has the same effect on the curve as an increase of τ with the same factor, i.e. on the ratio of unstable modes to total number of modes.
The main difference between an increase of τ and an increase of n lies in the fact that for the latter the total number of modes increases (the difference $\frac{2\pi}{n}$ between ϑ_l and ϑ_{l+1} decreases, for $l \in [-\frac{n}{2}, \frac{n}{2}]$), and hence that for a given ratio of unstable modes the total number of unstable modes will increase. In figure 10.4 we show this effect in detail. The influence of n is mainly felt for a flexible structure and a small time step. Thus, while reducing n will raise the relative number of modes that are unstable, it also reduces the total number of nodes; both effects counteract each-other.
Particular results are given below.

- For $\kappa = 1000$ and $\tau = 0.0001$: there is never more than 40% of the modes that are unstable.
- For $\kappa = 1000$ and $\tau = 0.01$: there is never more than 20% of the modes that are unstable.
- For $\kappa = 10$ and $\tau = 0.0001$: all frequencies are unstable as long as $n \leq 512$.
- For $\kappa = 10$ and $\tau = 0.001$: all frequencies are unstable as long as $n \leq 51$.
- For $\kappa = 10$ and $\tau = 0.01$: there is never more than 40% of the modes that is unstable.

(A more detailed study of this problem can be found in [44].)

10.1.6 Results with the quasi-Newton solvers

In this section we will solve the problem of one-dimensional flow in a flexible tube by means of the quasi-Newton methods described in chapter 5 and 6:

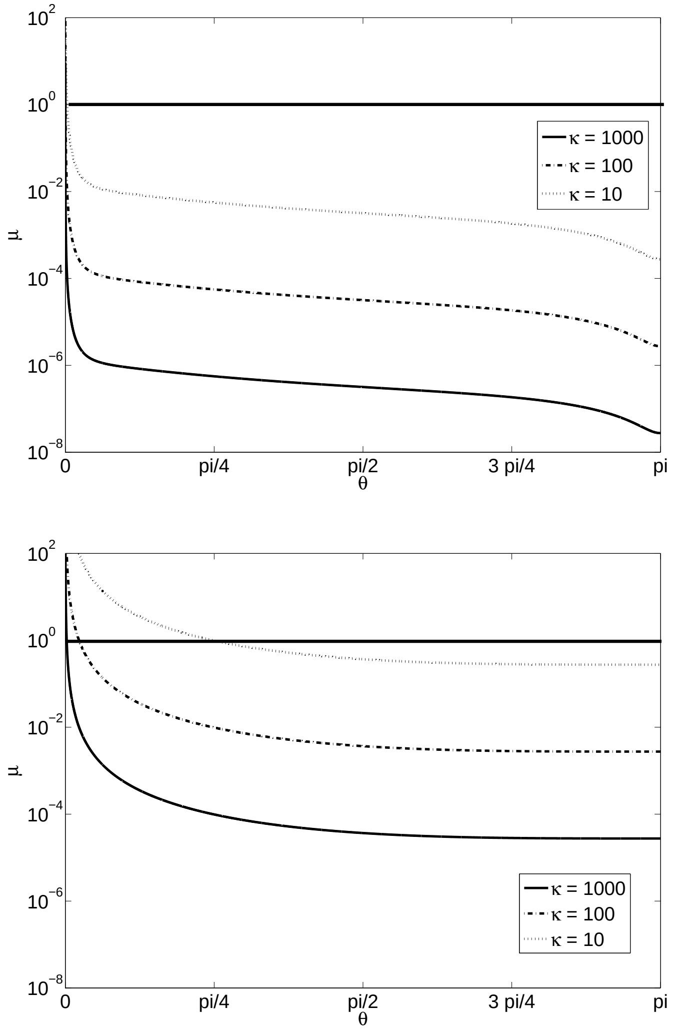

Figure 10.2: Error amplification for various values of κ and $\tau n = 10$ (top) or $\tau n = 0.1$ (bottom).

- IQN-BG, IQN-CBG, IBQN-BG, IQN-BB for the Broyden methods;
- IQN-CUM, IQN-CCUM, IBQN-CUM, IQN-ICUM for the (I)CUM methods;
- IQN-LS, IQN-CLS, IBQN-LS, IQN-ILS for the Least Squares methods, both in the original formulation and in rank-one update formulation.

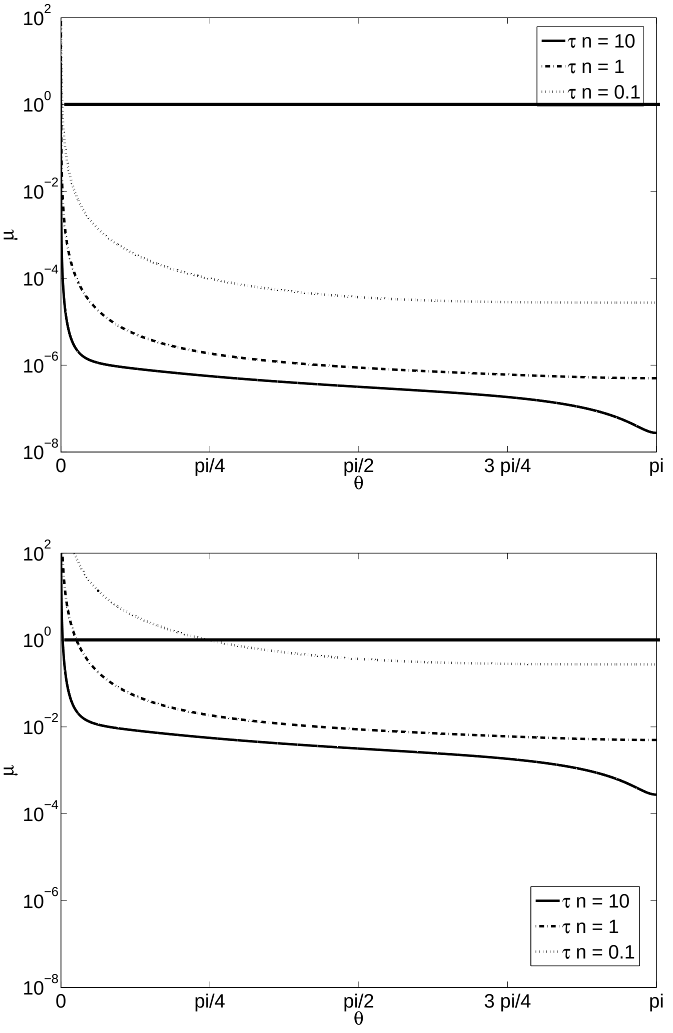

Figure 10.3: Error amplification for various values of τn *and* $\kappa = 1000$ *(top) or* $\kappa = 10$ *(bottom).*

For this time-dependent problem, the approaches from chapter 9 are used:

- Extrapolating the pressure to obtain an initial iterate and starting from a new Jacobian at every time-step.
- Extrapolating the pressure to obtain an initial iterate and adding input-output modes from previous time-steps to the original formulation (cfr. §9.1.1).

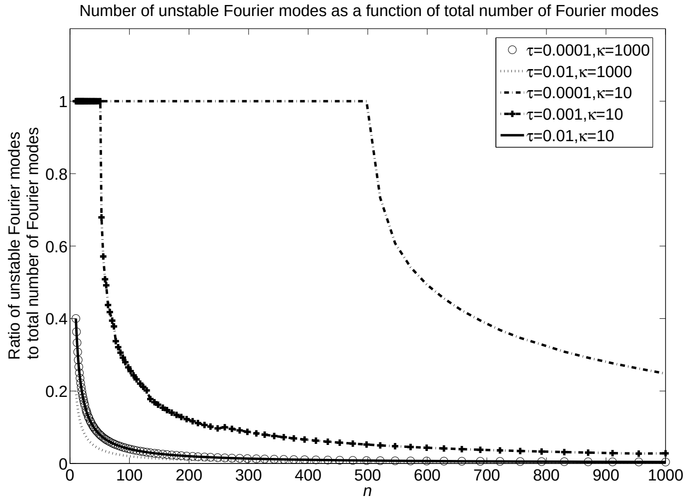

Figure 10.4: Ratio of number of unstable Fourier modes to total number of Fourier modes as a function of n for various values of κ and τ.

- Extrapolating the pressure to obtain an initial iterate and starting from the final approximate Jacobian of the previous time-step in a rank-one update formulation (cfr. §9.1.2).
- Extrapolating the pressure to obtain an initial iterate and creating an initial Jacobian based on data from a coarser grid (cfr. §9.2).
- Constructing an initial iterate based on a coarser grid and creating an initial Jacobian based on data from a coarser grid (cfr. §9.2).

We will use test-cases with $n = 100$ and $n = 1000$ nodes, except for the two-grid methods, where the fine grid will have 1000 nodes and the coarse grid 334 nodes. Values of τ ranging from 10^{-1} to 10^{-4} and values of κ ranging from 10 to 1000 will be used. (As shown in the Fourier study, more unstable modes will be present for lower values of both τ and κ, which will translate into the need for more coupling iterations.)
For the first iteration of the first time-step a relaxation factor ω is used, the value of which is given in the tables.
We define the relative residual (”Relres”) as $\frac{K_{t+1}(p_{s,t+1},p_t,g_t)}{K_{t+1}(p_{o,t+1},p_t,g_t)}$ (for IQN, IQN-C and IQN-I) or as $\frac{F_{t+1}(g_{s,t+1},p_t,g_t)-p_{s,t+1}}{F_{t+1}(g_{o,t+1},p_t,g_t)-p_{o,t+1}}$ (for IBQN) and use Relres $\leq 10^{-5}$ as a convergence criterium. The performance measure we use is the number of fluid-solver calls ($\mathcal{FC}$) at the fine grid and (if applicable) at the coarse grid. We break

off the iteration after 100 function calls if no convergence has been achieved at that point.
In tables 10.1-10.20 we give the number of iterations needed for the first time-step as well as the average over the first ten time-steps.

Remark 10.1. *All tests were performed using Matlab 7.0 on an Intel Xeon 3.40GHz dual-core processor.*

Remark 10.2. *The Matlab source code for these tests can be found in appendix A.*

10.1.6.1 No re-use of data from previous time-steps

As we can see in tables 10.1-10.8, there is little difference in the performance of the IQN, IQN-C, IBQN and IQN-I methods for the Least-Squares, Broyden and CUM methods, although the IBQN variants show a slightly lower numerical stability for low values of τ and κ.
We see that the Least Squares methods (tables 10.1, 10.2, 10.5 and 10.6) outperform the Broyden methods (tables 10.3 and 10.7) and CUM methods (tables 10.4 and 10.9), by a margin that grows as τ and κ become smaller. For the more difficult test-cases the gain can be of the order of 50%, while for the smallest values of τ and κ the Broyden and CUM methods fail. Also note that the CUM methods are slightly inferior to the Broyden methods.
Between the Least Squares methods in original formulation (tables 10.1 and 10.5) and in rank-one update formulation (tables 10.2 and 10.6) there is little difference.
Worthy of notice is that Broyden's "bad" method is not that much "worse" than Broyden's "good" method, although it is somewhat less stable. (The same comment can be made of IQN-ICUM with respect to IQN-CUM.)
We also remark that there is little difference between the values for $n = 100$ and $n = 1000$, unless for low values of τ and κ, which is in accordance with the findings of the Fourier analysis.

κ	τ	ω	IQN-LS	IQN-CLS	IBQN-LS	IQN-ILS
1000	10^{-1}	10^{-2}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
1000	10^{-2}	10^{-2}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
1000	10^{-3}	10^{-2}	4 - 4.0	4 - 4.0	4 - 4.0	4 - 4.0
1000	10^{-4}	10^{-3}	8 - 7.1	8 - 7.1	div	8 - 7.1
100	10^{-1}	10^{-2}	4 - 4.0	4 - 4.0	4 - 4.0	4 - 4.0
100	10^{-2}	10^{-2}	5 - 4.1	5 - 4.1	5 - 4.1	5 - 4.1
100	10^{-3}	10^{-2}	8 - 7.2	8 - 7.2	8 - 7.2	8 - 7.2
100	10^{-4}	10^{-3}	19 - 17.5	19 - 17.5	div	19 - 17.8
10	10^{-1}	10^{-2}	5 - 5.3	5 - 5.3	5 - 5.3	5 - 5.3
10	10^{-2}	10^{-4}	9 - 7.4	9 - 7.4	9 - 7.3	9 - 7.2
10	10^{-3}	10^{-5}	19 - 17.1	19 - 17.1	25 - 18.0	19 - 17.2
10	10^{-4}	10^{-6}	36 - 31.2	37 - 34.5	div	34 - 30.3

Table 10.1: $\mathcal{FC}$ required for convergence of the one-dimensional FSI problem if only the pressure is extrapolated over the time-steps and the Jacobian reset to $-I$ at every new time-step; values for the first time-step and average over the first 10 time-steps; solvers are of Least Squares quasi-Newton type in original (i.e. non rank-one update) formulation; $n = 100$; "div"= divergence or non-convergence after 100 function calls.

κ	τ	ω	IQN-LS (R1U)	IQN-CLS (R1U)	IBQN-LS (R1U)	IQN-ILS (R1U)
1000	10^{-1}	10^{-2}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
1000	10^{-2}	10^{-2}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
1000	10^{-3}	10^{-2}	4 - 4.0	4 - 4.0	4 - 4.0	4 - 4.0
1000	10^{-4}	10^{-3}	8 - 7.1	8 - 7.1	div	8 - 7.1
100	10^{-1}	10^{-2}	4 - 4.0	4 - 4.0	4 - 4.0	4 - 4.0
100	10^{-2}	10^{-2}	5 - 4.1	5 - 4.1	5 - 4.1	5 - 4.1
100	10^{-3}	10^{-2}	8 - 7.2	8 - 7.2	8 - 7.2	8 - 7.2
100	10^{-4}	10^{-3}	19 - 17.5	19 - 17.5	div	19 - 17.8
10	10^{-1}	10^{-2}	5 - 5.3	5 - 5.3	5 - 5.3	5 - 5.3
10	10^{-2}	10^{-4}	9 - 7.4	9 - 7.4	9 - 7.3	9 - 7.2
10	10^{-3}	10^{-5}	19 - 17.1	19 - 17.1	19 - 17.2	19 - 17.1
10	10^{-4}	10^{-6}	37 - 31.4	37 - 31.4	div	38 - 33.6

Table 10.2: $\mathcal{FC}$ required for convergence of the one-dimensional FSI problem if only the pressure is extrapolated over the time-steps and the Jacobian reset to $-I$ at every new time-step; values for the first time-step and average over the first 10 time-steps; solvers are of Least Squares quasi-Newton type in rank-one update formulation; $n = 100$; "div"= divergence or non-convergence after 100 function calls.

κ	τ	ω	IQN-BG	IQN-CBG	IBQN-BG	IQN-BB
1000	10^{-1}	10^{-2}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
1000	10^{-2}	10^{-2}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
1000	10^{-3}	10^{-2}	5 - 5.0	5 - 5.0	5 - 5.0	5 - 5.0
1000	10^{-4}	10^{-3}	9 - 9.0	10 - 9.0	div	11 - 9.4
100	10^{-1}	10^{-2}	4 - 4.0	4 - 4.0	4 - 4.0	4 - 4.0
100	10^{-2}	10^{-2}	5 - 5.0	5 - 5.0	5 - 5.0	5 - 5.0
100	10^{-3}	10^{-2}	9 - 8.9	10 - 9.0	9 - 8.9	11 - 9.4
100	10^{-4}	10^{-3}	37 - 34.8	39 - 36.3	div	75 - div
10	10^{-1}	10^{-2}	6 - 5.5	6 - 5.7	6 - 5.5	6 - 5.7
10	10^{-2}	10^{-4}	10 - 9.0	10 - 9.0	10 - 9.0	10 - 9.1
10	10^{-3}	10^{-5}	37 - 35.0	40 - 39.4	36 - 35.5	div
10	10^{-4}	10^{-6}	div	div	div	div

Table 10.3: $\mathcal{FC}$ required for convergence of the one-dimensional FSI problem if only the pressure is extrapolated over the time-steps and the Jacobian reset to $-I$ at every new time-step; values for the first time-step and average over the first 10 time-steps; solvers are of Broyden type; $n = 100$; "div"= divergence or non-convergence after 100 function calls.

κ	τ	ω	IQN-CUM	IQN-CCUM	IBQN-CUM	IQN-ICUM
1000	10^{-1}	10^{-2}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
1000	10^{-2}	10^{-2}	4 - 3.9	4 - 3.9	4 - 3.9	4 - 4.0
1000	10^{-3}	10^{-2}	6 - 5.1	6 - 5.1	6 - 5.1	6 - 5.1
1000	10^{-4}	10^{-3}	11 - 9.9	15 - 10.5	div	41 - 15.5
100	10^{-1}	10^{-2}	4 - 4.2	4 - 4.2	4 - 4.2	4 - 4.2
100	10^{-2}	10^{-2}	6 - 5.3	6 - 5.9	6 - 5.9	6 - 6.0
100	10^{-3}	10^{-2}	11 - 9.9	12 - 10.4	12 - 10.3	13 - 11.4
100	10^{-4}	10^{-3}	40 - 38.0	94 - 72.2	div	div
10	10^{-1}	10^{-2}	6 - 6.3	6 - 6.4	6 - 6.4	6 - 6.3
10	10^{-2}	10^{-4}	12 - 10.6	13 - 11.5	15 - 11.1	14 - 12.9
10	10^{-3}	10^{-5}	41 - 39.3	87 - 74.0	div	div
10	10^{-4}	10^{-6}	div	div	div	div

Table 10.4: $\mathcal{FC}$ required for convergence of the one-dimensional FSI problem if only the pressure is extrapolated over the time-steps and the Jacobian reset to $-I$ at every new time-step; values for the first time-step and average over the first 10 time-steps; solvers are of Column-Updating type; $n = 100$; "div"= divergence or non-convergence after 100 function calls.

κ	τ	ω	IQN-LS	IQN-CLS	IBQN-LS	IQN-ILS
1000	10^{-1}	10^{-2}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
1000	10^{-2}	10^{-2}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
1000	10^{-3}	10^{-2}	5 - 4.1	5 - 4.1	5 - 4.1	5 - 4.1
1000	10^{-4}	10^{-3}	8 - 7.0	8 - 7.0	div -	8 - 6.9
100	10^{-1}	10^{-2}	4 - 4.0	4 - 4.0	4 - 4.0	4 - 4.0
100	10^{-2}	10^{-2}	5 - 5.0	5 - 5.0	5 - 5.0	5 - 5.0
100	10^{-3}	10^{-2}	8 - 7.1	8 - 7.1	8 - 7.1	8 - 7.0
100	10^{-4}	10^{-3}	19 - 15.8	19 - 15.8	div	19 - 15.8
10	10^{-1}	10^{-2}	5 - 5.5	5 - 5.5	5 - 5.5	5 - 5.5
10	10^{-2}	10^{-4}	9 - 7.9	9 - 7.9	9 - 7.9	9 - 7.9
10	10^{-3}	10^{-5}	21 - 16.9	21 - 16.9	21 - 16.7	22 - 16.5
10	10^{-4}	10^{-6}	57 - 51.3	57 - 50.6	div	58 - 51.8

Table 10.5: $\mathcal{FC}$ required for convergence of the one-dimensional FSI problem if only the pressure is extrapolated over the time-steps and the Jacobian reset to $-I$ at every new time-step; values for the first time-step and average over the first 10 time-steps; solvers are of Least Squares quasi-Newton type in original (i.e. non rank-one update) formulation; $n = 1000$; "div"= divergence or non-convergence after 100 function calls.

κ	τ	ω	IQN-LS (R1U)	IQN-CLS (R1U)	IBQN-LS (R1U)	IQN-ILS (R1U)
1000	10^{-1}	10^{-2}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
1000	10^{-2}	10^{-2}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
1000	10^{-3}	10^{-2}	5 - 4.1	5 - 4.1	5 - 4.1	5 - 4.1
1000	10^{-4}	10^{-3}	8 - 7.0	8 - 7.0	div	8 - 6.9
100	10^{-1}	10^{-2}	4 - 4.0	4 - 4.0	4 - 4.0	4 - 4.0
100	10^{-2}	10^{-2}	5 - 5.0	5 - 5.0	5 - 5.0	5 - 5.0
100	10^{-3}	10^{-2}	8 - 7.1	8 - 7.1	8 - 7.1	8 - 7.0
100	10^{-4}	10^{-3}	19 - 15.8	19 - 15.8	div	19 - 15.8
10	10^{-1}	10^{-2}	5 - 5.5	5 - 5.5	5 - 5.5	5 - 5.5
10	10^{-2}	10^{-4}	9 - 7.9	9 - 7.9	9 - 7.9	9 - 7.9
10	10^{-3}	10^{-5}	21 - 16.9	21 - 16.9	22 - 16.6	22 - 16.5
10	10^{-4}	10^{-6}	57 - 51.3	57 - 51.3	div	59 - 53.2

Table 10.6: $\mathcal{FC}$ required for convergence of the one-dimensional FSI problem if only the pressure is extrapolated over the time-steps and the Jacobian reset to $-I$ at every new time-step; values for the first time-step and average over the first 10 time-steps; solvers are of Least Squares quasi-Newton type in rank-one update formulation; $n = 1000$; "div"= divergence or non-convergence after 100 function calls.

κ	τ	ω	IQN-BG	IQN-CBG	IBQN-BG	IQN-BB
1000	10^{-1}	10^{-2}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
1000	10^{-2}	10^{-2}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
1000	10^{-3}	10^{-2}	5 - 5.0	5 - 5.0	5 - 5.0	5 - 5.0
1000	10^{-4}	10^{-3}	9 - 8.7	9 - 8.7	div	9 - 8.7
100	10^{-1}	10^{-2}	4 - 4.1	4 - 4.1	4 - 4.1	4 - 4.1
100	10^{-2}	10^{-2}	5 - 5.0	5 - 5.0	5 - 5.0	5 - 5.0
100	10^{-3}	10^{-2}	10 - 9.1	10 - 9.1	10 - 9.1	10 - 9.1
100	10^{-4}	10^{-3}	36 - 34.9	38 - 36.4	div	div
10	10^{-1}	10^{-2}	6 - 5.6	6 - 5.7	6 - 5.6	6 - 5.7
10	10^{-2}	10^{-4}	11 - 9.7	11 - 9.8	11 - 9.7	12 - 10.7
10	10^{-3}	10^{-5}	38 - 36.0	42 - 43.6	39 - 38.6	div
10	10^{-4}	10^{-6}	div	div	div	div

Table 10.7: $\mathcal{FC}$ required for convergence of the one-dimensional FSI problem if only the pressure is extrapolated over the time-steps and the Jacobian reset to $-I$ at every new time-step; values for the first time-step and average over the first 10 time-steps; solvers are of Broyden type; $n = 1000$; ”div”= divergence or non-convergence after 100 function calls.

κ	τ	ω	IQN-CUM	IQN-CCUM	IBQN-CUM	IQN-ICUM
1000	10^{-1}	10^{-2}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
1000	10^{-2}	10^{-2}	4 - 4.0	4 - 4.0	4 - 4.0	4 - 4.0
1000	10^{-3}	10^{-2}	6 - 5.6	6 - 5.9	6 - 5.9	6 - 6.0
1000	10^{-4}	10^{-3}	11 - 10.1	12 - 10.3	div	14 - 11.6
100	10^{-1}	10^{-2}	4 - 4.2	4 - 4.2	4 - 4.2	4 - 4.2
100	10^{-2}	10^{-2}	6 - 6.0	6 - 6.0	6 - 6.0	6 - 6.0
100	10^{-3}	10^{-2}	11 - 10.4	14 - 11.2	16 - 11.6	14 - 12.1
100	10^{-4}	10^{-3}	41 - 37.4	75 - 67.6	div	div
10	10^{-1}	10^{-2}	6 - 6.4	6 - 6.5	6 - 6.4	6 - 6.4
10	10^{-2}	10^{-4}	13 - 11.3	14 - 12.4	15 - 12.6	15 - 14.0
10	10^{-3}	10^{-5}	44 - 44.0	div	div	div
10	10^{-4}	10^{-6}	div	div	div	div

Table 10.8: $\mathcal{FC}$ required for convergence of the one-dimensional FSI problem if only the pressure is extrapolated over the time-steps and the Jacobian reset to $-I$ at every new time-step; values for the first time-step and average over the first 10 time-steps; solvers are of Column-Updating type; $n = 1000$; ”div”= divergence or non-convergence after 100 function calls.

10.1.6.2 Re-use of data from previous time-steps

In this section we look into the data-recovery methods for the original formulation of the Least Squares methods (§9.1.1) and the data-recovery method for quasi-Newton methods in rank-one update methods (cfr. §9.1.2). The latter includes the

Least Squares method, the Broyden methods and the CUM methods.

For the original, non-rank-one update, form of the Least Squares algorithm we see that keeping the input-output pairs of the ten previous time-steps (when available) gives better results than only keeping those of the previous five time-steps (tables 10.10 and 10.12 versus tables 10.9 and 10.11). When keeping data from ten time-steps the gain (in the average number of iterations) with respect to the approach without re-use (§10.1.6.1) is in the order of 25% for the highest values of τ and κ and in the order of 80% for the lowest values of τ and κ. When using data from five time-steps these gains are 25% and 70% respectively. Also note that this average is only taken over 10 time-steps; as the number of iterates for the first time-step is the same with and without re-use, this first time-step will weigh rather heavily on the average. Also, the full potential of the re-use of data from the previous ten time-steps only comes in full force at the tenth time-step. To give a clearer indication of the potential gain, we point out that at the tenth time-step for $\tau = 10^{-4}$ and $\kappa = 10$ the number of iterations with re-use of the previous ten time-steps is about 85% lower than without re-use.

For the Least Squares methods in rank-one update formulation (tables 10.13 and 10.16) the gain is also substantial (up to 70%), although slightly inferior to that of the original formulation with the re-use of input-output pairs of the previous ten time-steps.

The Broyden and CUM methods also profit from the re-use of the Jacobian, but to a lesser extent than the Least Squares methods (tables 10.14, 10.15, 10.17 and 10.18). Gains up to 50% are recorded. The CUM methods are still slightly slower than the Broyden methods, and both are inferior to the Least Squares methods.

κ	τ	ω	IQN-LS	IQN-CLS	IBQN-LS	IQN-ILS
1000	10^{-1}	10^{-2}	3 - 2.2	3 - 2.2	3 - 2.2	3 - 3.1
1000	10^{-2}	10^{-2}	3 - 2.2	3 - 2.2	3 - 2.2	3 - 3.1
1000	10^{-3}	10^{-2}	4 - 2.4	4 - 2.4	4 - 2.4	4 - 3.3
1000	10^{-4}	10^{-3}	8 - 3.4	8 - 3.4	div	8 - 4.1
100	10^{-1}	10^{-2}	4 - 3.3	4 - 3.1	4 - 6.5	4 - 3.7
100	10^{-2}	10^{-2}	5 - 2.6	5 - 2.6	5 - 2.5	5 - 3.5
100	10^{-3}	10^{-2}	8 - 3.3	8 - 3.3	8 - 3.1	8 - 3.8
100	10^{-4}	10^{-3}	19 - 5.9	19 - 5.9	div	19 - 7.2
10	10^{-1}	10^{-2}	5 - 4.2	5 - 4.5	5 - div	5 - 5.3
10	10^{-2}	10^{-4}	9 - 3.6	9 - 3.7	9 - 3.7	9 - 5.4
10	10^{-3}	10^{-5}	19 - 5.8	19 - 5.8	25 - 6.6	19 - 7.9
10	10^{-4}	10^{-6}	36 - 11.9	37 - 11.3	div	34 - 12.5

Table 10.9: $\mathcal{FC}$ required for convergence of the one-dimensional FSI problem when the pressure is extrapolated over the time-steps and input-output modes of 5 previous time-steps (non-R1U formulation) are kept, when available; values for the first time-step and average over the first 10 time-steps; solvers are of Least Squares quasi-Newton type in original (i.e. non rank-one update) formulation; $n = 100$; ”div”= divergence or non-convergence after 100 function calls.

κ	τ	ω	IQN-LS	IQN-CLS	IBQN-LS	IQN-ILS
1000	10^{-1}	10^{-2}	3 - 2.2	3 - 2.2	3 - 2.2	3 - 3.0
1000	10^{-2}	10^{-2}	3 - 2.2	3 - 2.2	3 - 2.2	3 - 3.1
1000	10^{-3}	10^{-2}	4 - 2.4	4 - 2.4	4 - 2.4	4 - 3.3
1000	10^{-4}	10^{-3}	8 - 3.2	8 - 3.2	div	8 - 4.1
100	10^{-1}	10^{-2}	4 - 3.2	4 - 3.2	4 - 8.4	4 - 3.5
100	10^{-2}	10^{-2}	5 - 2.7	5 - 2.7	5 - 2.7	5 - 3.5
100	10^{-3}	10^{-2}	8 - 2.9	8 - 2.9	8 - 2.9	8 - 3.7
100	10^{-4}	10^{-3}	19 - 4.7	19 - 4.7	div	19 - 6.0
10	10^{-1}	10^{-2}	5 - 4.1	5 - 4.6	5 - div	5 - 5.2
10	10^{-2}	10^{-4}	9 - 3.6	9 - 3.6	9 - 3.5	9 - 5.6
10	10^{-3}	10^{-5}	19 - 4.0	19 - 4.0	25 - 4.9	19 - 6.0
10	10^{-4}	10^{-6}	36 - 8.1	37 - 7.8	div	34 - 10.6

Table 10.10: $\mathcal{FC}$ required for convergence of the one-dimensional FSI problem when the pressure is extrapolated over the time-steps and input-output modes of 10 previous time-steps (non-R1U formulation) are kept, when available; values for the first time-step and average over the first 10 time-steps; solvers are of Least Squares quasi-Newton type in original (i.e. non rank-one update) formulation; $n = 100$; ”div”= divergence or non-convergence after 100 function calls.

κ	τ	ω	IQN-LS	IQN-CLS	IBQN-LS	IQN-ILS
1000	10^{-1}	10^{-2}	3 - 2.2	3 - 2.2	3 - 2.2	3 - 3.1
1000	10^{-3}	10^{-2}	3 - 2.2	3 - 2.2	3 - 2.2	3 - 3.1
1000	10^{-2}	10^{-2}	5 - 2.4	5 - 2.4	5 - 2.4	5 - 3.4
1000	10^{-4}	10^{-3}	8 - 3.3	8 - 3.3	div	8 - 4.1
100	10^{-1}	10^{-2}	4 - 3.2	4 - 3.1	4 - 3.2	4 - 3.5
100	10^{-2}	10^{-2}	5 - 2.7	5 - 2.7	5 - 2.6	5 - 3.5
100	10^{-3}	10^{-2}	8 - 2.8	8 - 2.8	8 - 2.8	8 - 3.8
100	10^{-4}	10^{-3}	19 - 6.2	19 - 6.2	div	19 - 6.9
10	10^{-1}	10^{-2}	5 - 4.4	5 - 5.3	5 - div	5 - 5.0
10	10^{-2}	10^{-4}	9 - 3.6	9 - 3.6	9 - 3.6	9 - 5.5
10	10^{-3}	10^{-5}	21 - 5.9	21 - 5.9	21 - 6.0	22 - 8.2
10	10^{-4}	10^{-6}	57 - 17.1	57 - 16.8	div	58 - 21.4

Table 10.11: $\mathcal{FC}$ required for convergence of the one-dimensional FSI problem when the pressure is extrapolated over the time-steps and input-output modes of 5 previous time-steps (non-R1U formulation) are kept, when available; values for the first time-step and average over the first 10 time-steps; solvers are of Least Squares quasi-Newton type in original (i.e. non rank-one update) formulation; $n = 1000$; ”div”= divergence or non-convergence after 100 function calls.

κ	τ	ω	IQN-LS	IQN-CLS	IBQN-LS	IQN-ILS
1000	10^{-1}	10^{-2}	3 - 2.2	3 - 2.2	3 - 2.2	3 - 3.0
1000	10^{-2}	10^{-2}	3 - 2.2	3 - 2.2	3 - 2.2	3 - 3.1
1000	10^{-3}	10^{-2}	5 - 2.4	5 - 2.4	5 - 2.4	5 - 3.3
1000	10^{-4}	10^{-3}	8 - 3.1	8 - 3.1	div	8 - 4.0
100	10^{-1}	10^{-2}	4 - 3.1	4 - 3.1	4 - 3.1	4 - 3.4
100	10^{-2}	10^{-2}	5 - 2.7	5 - 2.7	5 - 2.8	5 - 3.5
100	10^{-3}	10^{-2}	8 - 2.8	8 - 2.8	8 - 2.8	8 - 3.8
100	10^{-4}	10^{-3}	19 - 5.0	19 - 5.1	div	19 - 5.3
10	10^{-1}	10^{-2}	5 - 4.3	5 - 5.2	5 - div	5 - 5.0
10	10^{-2}	10^{-4}	9 - 3.5	9 - 3.5	9 - 3.5	9 - 5.6
10	10^{-3}	10^{-5}	21 - 4.4	21 - 4.4	21 - 4.8	22 - 6.5
10	10^{-4}	10^{-6}	57 - 11.9	57 - 11.9	div	58 - 16.0

Table 10.12: $\mathcal{FC}$ required for convergence of the one-dimensional FSI problem when the pressure is extrapolated over the time-steps and input-output modes of 10 previous time-steps (non-R1U formulation) are kept, when available; values for the first time-step and average over the first 10 time-steps; solvers are of Least Squares quasi-Newton type in original (i.e. non rank-one update) formulation; $n = 1000$; ”div”= divergence or non-convergence after 100 function calls.

κ	τ	ω	IQN-LS (R1U)	IQN-CLS (R1U)	IBQN-LS (R1U)	IQN-ILS (R1U)
1000	10^{-1}	10^{-2}	3 - 2.6	3 - 2.6	3 - 2.7	3 - 2.6
1000	10^{-2}	10^{-2}	3 - 2.4	3 - 2.4	3 - 2.4	3 - 2.4
1000	10^{-3}	10^{-2}	4 - 2.7	4 - 2.7	4 - 2.7	4 - 2.7
1000	10^{-4}	10^{-3}	8 - 3.6	8 - 3.6	div	8 - 3.7
100	10^{-1}	10^{-2}	4 - 3.1	4 - 3.1	4 - 3.1	4 - 3.1
100	10^{-2}	10^{-2}	5 - 3.0	5 - 3.0	5 - 2.7	5 - 3.0
100	10^{-3}	10^{-2}	8 - 3.5	8 - 3.5	8 - 3.5	8 - 3.5
100	10^{-4}	10^{-3}	19 - 5.0	19 - 4.9	div	19 - 7.5
10	10^{-1}	10^{-2}	5 - 4.1	5 - 4.1	5 - 4.0	5 - 3.9
10	10^{-2}	10^{-4}	9 - 3.7	9 - 4.0	9 - 4.0	9 - 4.0
10	10^{-3}	10^{-5}	19 - 5.0	19 - 5.5	19 - 5.2	19 - 7.0
10	10^{-4}	10^{-6}	37 - 11.5	37 - 12.4	div	38 - 37.3

Table 10.13: $\mathcal{FC}$ required for convergence of the one-dimensional FSI problem when the pressure is extrapolated over the time-steps and the initial Jacobian for a new time-step is taken as the final Jacobian from the previous time-step; values for the first time-step and average over the first 10 time-steps; solvers are of Least Squares quasi-Newton type in rank-one update formulation; $n = 100$; "div"= divergence or non-convergence after 100 function calls.

κ	τ	ω	IQN-BG	IQN-CBG	IBQN-BG	IQN-BB
1000	10^{-1}	10^{-2}	3 - 2.6	3 - 2.6	3 - 2.6	3 - 2.6
1000	10^{-2}	10^{-2}	3 - 2.4	3 - 2.4	3 - 2.4	3 - 2.4
1000	10^{-3}	10^{-2}	5 - 2.8	5 - 2.8	5 - 3.0	5 - 2.8
1000	10^{-4}	10^{-3}	9 - 4.9	10 - 5.5	div	11 - 7.0
100	10^{-1}	10^{-2}	4 - 3.3	4 - 3.3	4 - 3.3	4 - 3.3
100	10^{-2}	10^{-2}	5 - 3.4	5 - 3.5	5 - 3.5	5 - 3.4
100	10^{-3}	10^{-2}	9 - 4.6	10 - 5.1	9 - 5.2	11 - 6.4
100	10^{-4}	10^{-3}	37 - 10.3	39 - 10.9	div	75 - div
10	10^{-1}	10^{-2}	6 - 4.9	6 - 5.0	6 - 5.0	6 - 5.0
10	10^{-2}	10^{-4}	10 - 5.1	10 - 5.5	10 - 5.3	10 - 6.2
10	10^{-3}	10^{-5}	37 - 10.5	40 - 10.8	36 - 10.4	div
10	10^{-4}	10^{-6}	div	div	div	div

Table 10.14: $\mathcal{FC}$ required for convergence of the one-dimensional FSI problem when the pressure is extrapolated over the time-steps and the initial Jacobian for a new time-step is taken as the final Jacobian from the previous time-step; values for the first time-step and average over the first 10 time-steps; solvers are of Broyden type; $n = 100$; "div"= divergence or non-convergence after 100 function calls.

κ	τ	ω	IQN-CUM	IQN-CCUM	IBQN-CUM	IQN-ICUM
1000	10^{-1}	10^{-2}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
1000	10^{-2}	10^{-2}	4 - 3.0	4 - 3.0	4 - 3.1	4 - 3.0
1000	10^{-3}	10^{-2}	6 - 4.3	6 - 4.5	6 - 4.3	6 - 4.5
1000	10^{-4}	10^{-3}	11 - 6.9	15 - 13.3	div	41 - 12.1
100	10^{-1}	10^{-2}	4 - 3.4	4 - 3.4	4 - 3.4	4 - 3.4
100	10^{-2}	10^{-2}	6 - 4.4	6 - 4.7	6 - 4.7	6 - 4.7
100	10^{-3}	10^{-2}	11 - 6.6	12 - 7.3	12 - 7.2	13 - 9.5
100	10^{-4}	10^{-3}	40 - 22.2	94 - div	div	div
10	10^{-1}	10^{-2}	6 - 5.1	6 - 5.0	6 - 5.2	6 - 5.3
10	10^{-2}	10^{-4}	12 - 7.0	13 - 7.7	15 - 8.0	14 - 12.7
10	10^{-3}	10^{-5}	41 - 17.1	87 - 34.1	div	div
10	10^{-4}	10^{-6}	div	div	div	div

Table 10.15: $\mathcal{FC}$ required for convergence of the one-dimensional FSI problem when the pressure is extrapolated over the time-steps and the initial Jacobian for a new time-step is taken as the final Jacobian from the previous time-step; values for the first time-step and average over the first 10 time-steps; solvers are of Column-Updating type; $n = 100$; "div"= divergence or non-convergence after 100 function calls.

κ	τ	ω	IQN-LS (R1U)	IQN-CLS (R1U)	IBQN-LS (R1U)	IQN-ILS (R1U)
1000	10^{-1}	10^{-2}	3 - 2.6	3 - 2.6	3 - 2.7	3 - 2.6
1000	10^{-2}	10^{-2}	3 - 2.4	3 - 2.4	3 - 2.4	3 - 2.4
1000	10^{-3}	10^{-2}	5 - 2.5	5 - 2.5	5 - 2.5	5 - 2.5
1000	10^{-4}	10^{-3}	8 - 3.6	8 - 3.6	div	8 - 3.6
100	10^{-1}	10^{-2}	4 - 3.0	4 - 3.0	4 - 3.1	4 - 3.0
100	10^{-2}	10^{-2}	5 - 2.9	5 - 2.9	5 - 2.7	5 - 2.9
100	10^{-3}	10^{-2}	8 - 3.4	8 - 3.4	8 - 3.4	8 - 3.4
100	10^{-4}	10^{-3}	19 - 4.5	19 - 4.5	div	19 - 5.9
10	10^{-1}	10^{-2}	5 - 4.1	5 - 4.0	5 - 4.2	5 - 4.0
10	10^{-2}	10^{-4}	9 - 3.8	9 - 3.9	9 - 3.9	9 - 4.4
10	10^{-3}	10^{-5}	21 - 5.2	21 - 5.3	22 - 5.2	22 - 7.0
10	10^{-4}	10^{-6}	57 - 11.2	57 - 11.7	div	59 - 21.5

Table 10.16: $\mathcal{FC}$ required for convergence of the one-dimensional FSI problem when the pressure is extrapolated over the time-steps and the initial Jacobian for a new time-step is taken as the final Jacobian from the previous time-step; values for the first time-step and average over the first 10 time-steps; solvers are of Least Squares quasi-Newton type in rank-one update formulation; $n = 1000$; "div"= divergence or non-convergence after 100 function calls.

κ	τ	ω	IQN-BG	IQN-CBG	IBQN-BG	IQN-BB
1000	10^{-1}	10^{-2}	3 - 2.6	3 - 2.6	3 - 2.6	3 - 2.6
1000	10^{-2}	10^{-2}	3 - 2.4	3 - 2.4	3 - 2.4	3 - 2.4
1000	10^{-3}	10^{-2}	5 - 2.9	5 - 3.0	5 - 3.5	5 - 3.4
1000	10^{-4}	10^{-3}	9 - 4.7	9 - 5.0	div	9 - 5.9
100	10^{-1}	10^{-2}	4 - 3.3	4 - 3.3	4 - 3.3	4 - 3.3
100	10^{-2}	10^{-2}	5 - 3.5	5 - 3.6	5 - 3.5	5 - 4.0
100	10^{-3}	10^{-2}	10 - 4.7	10 - 5.4	10 - 4.8	10 - 5.5
100	10^{-4}	10^{-3}	36 - 9.4	38 - 10.5	div	div
10	10^{-1}	10^{-2}	6 - 4.9	6 - 5.0	6 - 4.9	6 - 4.9
10	10^{-2}	10^{-4}	11 - 5.4	11 - 5.6	11 - 5.4	12 - 7.0
10	10^{-3}	10^{-5}	38 - 9.3	42 - 10.5	39 - 9.8	div
10	10^{-4}	10^{-6}	div	div	div	div

Table 10.17: $\mathcal{FC}$ required for convergence of the one-dimensional FSI problem when the pressure is extrapolated over the time-steps and the initial Jacobian for a new time-step is taken as the final Jacobian from the previous time-step; values for the first time-step and average over the first 10 time-steps; solvers are of Broyden type; $n = 1000$; "div"= divergence or non-convergence after 100 function calls.

κ	τ	ω	IQN-CUM	IQN-CCUM	IBQN-CUM	IQN-ICUM
1000	10^{-1}	10^{-2}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
1000	10^{-2}	10^{-2}	4 - 3.2	4 - 3.2	4 - 3.2	4 - 3.2
1000	10^{-3}	10^{-2}	6 - 4.4	6 - 3.8	6 - 4.7	6 - 4.4
1000	10^{-4}	10^{-3}	11 - 6.5	12 - 7.1	div	14 - 10.3
100	10^{-1}	10^{-2}	4 - 3.4	4 - 3.4	4 - 3.4	4 - 3.4
100	10^{-2}	10^{-2}	6 - 4.6	6 - 4.7	6 - 4.5	6 - 4.7
100	10^{-3}	10^{-2}	11 - 7.0	14 - 7.5	16 - 8.6	14 - 12.2
100	10^{-4}	10^{-3}	41 - 16.5	75 - div	div	div
10	10^{-1}	10^{-2}	6 - 5.1	6 - 5.3	6 - 5.2	6 - 5.2
10	10^{-2}	10^{-4}	13 - 7.7	14 - 8.5	15 - 8.6	15 - 14.9
10	10^{-3}	10^{-5}	44 - 17.3	div	div	div
10	10^{-4}	10^{-6}	div	div	div	div

Table 10.18: $\mathcal{FC}$ required for convergence of the one-dimensional FSI problem when the pressure is extrapolated over the time-steps and the initial Jacobian for a new time-step is taken as the final Jacobian from the previous time-step; values for the first time-step and average over the first 10 time-steps; solvers are of Column-Updating type; $n = 1000$; "div"= divergence or non-convergence after 100 function calls.

10.1.6.3 Initial Jacobian from a coarser grid

In this approach we use two grids as described in §9.2. The fine grid will have 1000 nodes and the coarse grid 334 nodes.

The validity of this approach is confirmed by the Fourier study in §10.1.5, which shows that the most unstable modes are those with the lowest frequency. It will be exactly those modes that we will be able to capture on the coarse grid.

Comparing these results with those of the previous sections is difficult, however, because

1. the initial residual can be different with this method, when the initial iterate is obtained from a coarser grid, instead of being extrapolated from previous time-steps (we recall that we have based our convergence criterium on the reduction of the relative residual);

2. it is not obvious to put an exact cost on the iterations on the fine grid, although it is roughly one third of the cost on the fine grid, based on the relative number of nodes on the fine and coarse grid.

From tables 10.19 and 10.20 we see that an initial iterate for the pressure obtained by extrapolating from the previous time-steps gives better results than when it is obtained from the coarser grid, even for this relatively high ratio between the number of nodes of the coarse and fine grid.

Compared with the results in tables 10.5 and 10.6 of §10.1.6.1, we see that the two-grid method with an extrapolated initial iterate gives a far lower value of the number of iterations for the first time-step for low values of τ and κ. The actual gain is largely off-set, however, when counting an iteration on the coarse grid for one third of an iteration on the fine grid. For high values of τ and κ we can even speak of a net loss.
Compared with the results in §10.1.6.2 (tables 10.11,10.12 and 10.16) there is no net gain to be obtained with this method, neither for low values of τ and κ or for high values.

κ	τ	ω	IQN-LS	IQN-CLS	IBQN-LS	IQN-ILS
1000	10^{-1}	10^{-2}	3 - 3.0 | 3 - 3.0	3 - 3.0 | 3 - 3.0	3 - 3.0| 3 - 3.0	3 - 3.0 | 3 - 3.0
1000	10^{-2}	10^{-2}	3 - 3.0 | 3 - 3.0	3 - 3.0 | 3 - 3.0	3 - 3.0 | 3 - 3.0	3 - 3.0 | 3 - 3.0
1000	10^{-3}	10^{-2}	4 - 4.0 | 4 - 4.0	4 - 4.0 | 4 - 4.0	4 - 4.0 | 4 - 4.0	4 - 4.0 | 4 - 4.0
1000	10^{-4}	10^{-3}	4 - 4.0 | 8 - 7.2	5 - 4.8 | 8 - 7.2	div	5 - 4.7 | 8 - 7.2
100	10^{-1}	10^{-2}	3 - 3.0 | 4 - 4.0	3 - 3.2 | 4 - 4.0	3 - 3.2 | 4 - 4.0	3 - 3.2 | 4 - 4.0
100	10^{-2}	10^{-2}	4 - 4.0 | 5 - 5.0	4 - 4.0 | 5 - 5.0	4 - 4.0| 5 - 5.0	4 - 4.0 | 5 - 5.0
100	10^{-3}	10^{-2}	5 - 4.3 | 8 - 7.1	6 - 5.2 | 8 - 7.1	6 - 5.2 | 8 - 7.1	6 - 5.2 | 8 - 7.1
100	10^{-4}	10^{-3}	7 - 7.8 | 20 - 20.4	7 - 8.3 | 20 - 20.4	div	7 - 8.1 | 20 - 20.8
10	10^{-1}	10^{-2}	4 - 3.7 | 5 - 5.5	4 - 4.2 | 5 - 5.5	4 - 4.2| 5 - 5.5	4 - 4.2 | 5 - 5.5
10	10^{-2}	10^{-4}	5 - 4.2 | 9 - 7.9	6 - 5.1 | 9 - 7.9	6 - 5.1 | 9 - 7.9	6 - 5.1 | 9 - 7.9
10	10^{-3}	10^{-5}	8 - 8.4 | 21 - 20.4	8 - 8.3 | 21 - 20.4	9 - 8.6 | 24 - 20.8	8 - 7.9 | 21 - 20.5
10	10^{-4}	10^{-6}	27 - 33.0 | 57 - 50.3	29 - 31.9 | 56 - 51.0	div	36 - 35.7 | 54 - 52.0

Table 10.19: $\mathcal{FC}$ required for convergence of the one-dimensional FSI problem when the pressure is extrapolated over the time-steps and the initial Jacobian for a new time-step is based on computations on a coarser grid at every time-step; values for the first time-step and average over the first 10 time-steps on fine and coarse grid respectively; solvers are of Least Squares type; 1000 nodes on fine grid and 334 nodes on coarse grid; "div"= divergence or non-convergence after 100 function calls.

κ	τ	ω	IQN-LS	IQN-CLS	IBQN-LS	IQN-ILS
1000	10^{-1}	10^{-2}	3 - 3.0 | 3 - 3.0	3 - 3.0 | 3 - 3.0	3 - 3.1 | 3 - 3.0	3 - 3.0 | 3 - 3.0
1000	10^{-2}	10^{-2}	3 - 3.0 | 3 - 3.0	3 - 3.0 | 3 - 3.0	3 - 3.0 | 3 - 3.0	3 - 3.0 | 3 - 3.0
1000	10^{-3}	10^{-2}	3 - 3.3 | 4 - 4.0	3 - 3.9 | 4 - 4.0	3 - 3.9 | 4 - 4.0	3 - 3.9 | 4 - 4.0
1000	10^{-4}	10^{-3}	div	div	div	div
100	10^{-1}	10^{-2}	3 - 3.0 | 4 - 3.8	4 - 3.7 | 4 - 3.8	4 - 3.6 | 4 - 3.8	4 - 3.7 | 4 - 3.8
100	10^{-2}	10^{-2}	3 - 3.8 | 5 - 4.9	4 - 4.0 | 5 - 4.9	4 - 4.0 | 5 - 4.9	4 - 4.0 | 5 - 4.9
100	10^{-3}	10^{-2}	5 - 4.3 | 8 - 7.3	5 - 5.0 | 8 - 7.3	5 - 5.0 | 8 - 7.3	5 - 5.1 | 8 - 7.1
100	10^{-4}	10^{-3}	8 - 10.7 | 20 - 17.7	8 - 10.9 | 20 - 17.8	div	8 - div | 20 - div
10	10^{-1}	10^{-2}	3 - 3.7 | 5 - 5.5	4 - 4.3 | 5 - 5.5	4 - 4.2 | 5 - 5.5	4 - 4.3 | 5 - 5.5
10	10^{-2}	10^{-4}	5 - 4.3 | 9 - 7.9	5 - 5.2 | 9 - 7.9	5 - 5.1 | 9 - 8.1	5 - 5.2 | 9 - 8.0
10	10^{-3}	10^{-5}	14 - 9.4 | 21 - 19.2	15 - 9.2 | 21 - 19.2	div	13 - 9.5 | 21 - 19.5
10	10^{-4}	10^{-6}	30 - 31.5 | 57 - 45.2	31 - 40.4 | 56 - 45.2	div	33 - 42.3 | 54 - 45.9

Table 10.20: $\mathcal{FC}$ required for convergence of the one-dimensional FSI problem when the initial value for the pressure and the initial Jacobian for a new time-step are based on computations on a coarser grid at every time-step; values for the first time-step and average over the first 10 time-steps on fine and coarse grid respectively; solvers are of Least Squares type; 1000 nodes on fine grid and 334 nodes on coarse grid; "div"= divergence or non-convergence after 100 function calls.

10.2 One-dimensional heat equation

10.2.1 Analytical description of the problem

The heat equation in one dimension without heat source is given by:

$$\rho C \frac{\partial T}{\partial t} - \frac{\partial}{\partial x}\left(k \frac{\partial T}{\partial x} \right) = 0. \tag{10.28}$$

In this equation T is temperature, ρ is density, C is heat capacity, and k is thermal conductivity. The length L of the domain is set equal to 1000. The material we are working with is air. We assume we only have a linear solver to solve (10.28) with an imposed value of ρ, C and k that may vary in space.
If we want to take dependency of ρ, C and k on T into account, we need to do this outside the solver for (10.28). They can be obtained by the following interpolation polynomials (based on values between 0 and 300° C in [253]):

$$\begin{aligned} k &= 7.0277 \cdot 10^{-5} T + 2.4388 \cdot 10^{-2} && \text{(10.29a)} \\ C &= 4.3004 \cdot 10^{-7} T^2 + 1.1850 \cdot 10^{-5} T + 1.0048 && \text{(10.29b)} \\ \rho &= 5.3641 \cdot 10^{-6} T^2 - 3.7809 \cdot 10^{-3} T + 1.2781. && \text{(10.29c)} \end{aligned}$$

We use a finite differencing scheme on n equidistant nodes for the solver of (10.28), with spacing Δx, and an implicit time-discretization with time-step Δt. We assume that at the start of the test we have a uniform temperature of $T_o = 150°$ C. The right boundary is kept at T_o, while the left boundary condition is a function of time, defined as $T_L = T_o + \frac{T_o}{2} \sin\left(\frac{\pi t}{10}\right)$.
After discretization we obtain the n equations ($i = 1, \dots, n$):

$$\begin{aligned} \frac{2}{\nu}[\rho]_i[C]_i([T]_i - [T_t]_i) - ([k]_{i+1} + [k]_i)[T]_{i+1} \\ + ([k]_{i+1} + 2[k]_i + [k]_{i-1})[T]_i - ([k]_i + [k]_{i-1})[T]_{i-1} = 0, \end{aligned} \tag{10.30}$$

where $\nu = \frac{\Delta t}{\Delta x^2}$. The subscripts i, $i+1$ and $i-1$ indicate the mesh nodes. The subscript t denotes the previous time-level; the subscript $t+1$ for the new time-level is omitted. For ease of notation the same symbol is used for the continuous and discretized variable T, as the brackets and indices for the discretized variable clearly indicate the difference between continuous and discretized variables.
We will use g to denote the ensemble of discretized constants ρ, C and k expressed as functions of T; discretization is done using a one-to-one relationship with T.

We will define the heat and coefficient solver by the following conventions.

- $F_{t+1}(g_{s,t+1}, T_t) = T_{s+1,t+1}$: solve the heat equation (10.30) for the temperature at time-level $t+1$ with a fixed set of coefficients $g_{s,t+1}$; assign

$T_{s+1,t+1}$ to this solution.
This corresponds to the definition of F_{t+1} used in 9.1a; note that in this particular case F_{t+1} is no function of g_t.

- $S_{t+1}(T_{s+1,t+1}) = g_{s+1,t+1}$: compute the new coefficients at time-level $t+1$ from (10.29) given the previously calculated temperature $T_{s+1,t+1}$; assign $g_{s+1,t+1}$ to this solution.
  This corresponds to the definition of S_{t+1} used in (9.1b); note that in this particular case S_{t+1} is no function of g_t or T_t.

Remark Even though (10.30) is a linear equation in T, F_{t+1} is not linear, i.e. the relationship between g and T is not linear. This can be easily seen by the fact that a doubling of the values of g does not double the corresponding values of T.

10.2.2 Results with the quasi-Newton solvers

In this section we will solve the problem of the one-dimensional heat equation with variable coefficients by means of the quasi-Newton methods described in chapter 5 and 6:

- IQN-BG, IQN-CBG, IBQN-BG, IQN-BB for the Broyden methods;
- IQN-CUM, IQN-CCUM, IBQN-CUM, IQN-ICUM for the (I)CUM methods;
- IQN-LS, IQN-CLS, IBQN-LS, IQN-ILS for the Least Squares methods, both in the original formulation and in rank-one update formulation.

For this time-dependent problem, the following approaches from chapter 9 are used:

- Extrapolating the temperature to obtain an initial iterate and starting from a new Jacobian at every time-step.
- Extrapolating the temperature to obtain an initial iterate and adding input-output modes from previous time-steps to the original formulation (cfr. §9.1.1).
- Extrapolating the temperature to obtain an initial iterate and starting from the final approximate Jacobian of the previous time-step in a rank-one update formulation (cfr. §9.1.2).

We will use test-cases with $n = 100$ and $n = 1000$ nodes.
Values of Δt ranging from 10^{-7} to 1 will be used.

For the first iteration of the first time-step a relaxation factor $\omega = 0.1$ is used.
We define the relative residual (”Relres”) as $\frac{K_{t+1}(p_{s,t+1},T_t)}{K_{t+1}(p_{o,t+1},T_t)}$ (for IQN, IQN-C and IQN-I) or as $\frac{F_{t+1}(g_{s,t+1},T_t)-p_{s,t+1}}{F_{t+1}(g_{o,t+1},T_t)-p_{o,t+1}}$ (for IBQN) and use Relres $\leq 10^{-8}$ as a convergence criterium. The performance measure we use is the number of heat-solver calls ($\mathcal{F}C$). We break off the iteration after 100 function calls if no convergence has been achieved at that point.
In tables 10.21-10.32 we give the number of iterations needed for the first time-step as well as the average over the first 10 ten time-steps.

Remark All tests were performed using Matlab 7.3 on an Intel Xeon E520 2.50GHz quad-core processor.

10.2.2.1 No re-use of data from previous time-steps

This problem is clearly an easy problem where a low number of iterations is needed and most methods have a very similar performance.

As we can see in tables 10.21 and 10.23, all the Least Squares methods have the same performance, with the difference that for $n = 100$ and $\Delta t = 10^{-7}$ the original formulation diverges, while the rank-one update method converges in a low number of iterations. The same is true for $n = 1000$ and $\Delta t \in \{10^{-7}, 10^{-6}\}$.
All the Broyden and CUM methods also share the same performance (tables 10.22 and 10.24); while they are a little bit faster than the methods for the highest value of Δt when $n = 100$, they diverge for the smallest value ($\Delta t = 10^{-7}$). For $n = 1000$ their performance is identical to that of the Least Squares methods in the original formulation.
Note that convergence is actually faster for $n = 1000$ than for $n = 100$.

Δt	IQN -LS	IQN -CLS	IBQN -LS	IQN -ILS	IQN -LS (R1U)	IQN -CLS (R1U)	IBQN -LS (R1U)	IQN -ILS (R1U)
10^{-7}	3 -div	3 - div	3 - div	3 -div	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
10^{-6}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
10^{-5}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
10^{-4}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
10^{-3}	3 - 3.1	3 - 3.1	3 - 3.2	3 - 3.1	3 - 3.1	3 - 3.1	3 - 3.1	3 - 3.1
10^{-2}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
10^{-1}	4 - 4.0	4 - 4.0	4 - 4.0	4 - 4.0	4 - 4.0	4 - 4.0	4 - 4.0	4 - 4.0
1	6 - 5.1	6 - 5.1	6 - 5.1	6 - 5.1	6 - 5.1	6 - 5.1	6 - 5.1	6 - 5.1

Table 10.21: $\mathcal{FC}$ required for convergence of the one-dimensional heat equation if only the temperature is extrapolated over the time-steps and the Jacobian reset to $-I$ at every new time-step; values for the first time-step and average over the first 10 time-steps; solvers are of Least Squares type; $n = 100$; "div"= divergence or non-convergence after 100 function calls.

Δt	IQN -BG	IQN -CBG	IBQN -BG	IQN -BB	IQN -CUM	IQN -CCUM	IBQN -CUM	IQN -ICUM
10^{-7}	3 - div	3 - div	3 - div	3 - div	3 - div	3 - div	3 - div	3 - div
10^{-6}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
10^{-5}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
10^{-4}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
10^{-3}	3 - 3.1	3 - 3.1	3 - 3.1	3 - 3.1	3 - 3.1	3 - 3.1	3 - 3.1	3 - 3.1
10^{-2}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
10^{-1}	4 - 4.0	4 - 4.0	4 - 4.0	4 - 4.0	4 - 4.0	4 - 4.0	4 - 4.0	4 - 4.0
1	5 - 5.0	5 - 5.0	5 - 5.0	5 - 5.0	5 - 5.0	5 - 5.0	5 - 5.0	5 - 5.0

Table 10.22: $\mathcal{FC}$ required for convergence of the one-dimensional heat equation if only the temperature is extrapolated over the time-steps and the Jacobian reset to $-I$ at every new time-step; values for the first time-step and average over the first 10 time-steps; solvers are of Broyden and Column-Updating type; $n = 100$; "div"= divergence or non-convergence after 100 function calls.

Δt	IQN -LS	IQN -CLS	IBQN -LS	IQN -ILS	IQN -LS (R1U)	IQN -CLS (R1U)	IBQN -LS (R1U)	IQN -ILS (R1U)
10^{-7}	3 - div	3 - div	3 - div	3 - div	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
10^{-6}	3 - div	3 - div	3 - div	3 - div	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
10^{-5}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
10^{-4}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
10^{-3}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
10^{-2}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
10^{-1}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
1	4 - 4.0	4 - 4.0	4 - 4.0	4 - 4.0	4 - 4.0	4 - 4.0	4 - 4.0	4 - 4.0

Table 10.23: $\mathcal{FC}$ required for convergence of the one-dimensional heat equation if only the temperature is extrapolated over the time-steps and the Jacobian reset to $-I$ at every new time-step; values for the first time-step and average over the first 10 time-steps; solvers are of Least Squares type; $n = 1000$; "div"= divergence or non-convergence after 100 function calls.

Δt	IQN -BG	IQN -CBG	IBQN -BG	IQN -BB	IQN -CUM	IQN -CCUM	IBQN -CUM	IQN -ICUM
10^{-7}	3 - div	3 - div	3 - div	3 - div	3 - div	3 - div	3 - div	3 - div
10^{-6}	3 - div	3 - div	3 - div	3 - div	3 - div	3 - div	3 - div	3 - div
10^{-5}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
10^{-4}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
10^{-3}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
10^{-2}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
10^{-1}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
1	4 - 4.0	4 - 4.0	4 - 4.0	4 - 4.0	4 - 4.0	4 - 4.0	4 - 4.0	4 - 4.0

Table 10.24: $\mathcal{FC}$ required for convergence of the one-dimensional heat equation if only the temperature is extrapolated over the time-steps and the Jacobian reset to $-I$ at every new time-step; values for the first time-step and average over the first 10 time-steps; solvers are of Broyden and Column-Updating type; $n = 1000$; "div"= divergence or non-convergence after 100 function calls.

10.2.2.2 Re-use of data from previous time-steps

For the Least Squares methods in original formulation we see that the strategy to re-use data from previous time-steps, as discussed in chapter 9, only pays off for small values of Δt (tables 10.25-10.28). For large values of Δt the method is actually slower than for a method without re-use of data.
For $n = 100$ the re-use of data from 5 time-steps is identical for small values of τ and better for large values of Δt when compared with the re-use of data over 10 time-steps. The opposite is true for $n = 1000$, although by a very small margin.

For methods in rank-one update formulation we also see that a gain is only obtained for small values of Δt (tables 10.29-10.32). For large values the methods performs worse than without re-use. A notable exception is the IBQN-LS method which diverges for $\Delta t = 10^{-4}$ when $n = 100$ and for $\Delta t \in \{10^{-7}, 10^{-3}\}$ for $n = 1000$. The rank-one update methods with recovery perform somewhat better than the LS methods in original formulation with recovery of data.

For the Broyden and CUM methods (tables 10.30 and 10.32) we see that their performance is improved by the re-use of the Jacobian of previous time-steps for small values of Δt; for larger values the performance is identical or marginally better.
With respect to the Least Squares methods the Broyden and CUM methods have a slightly better performance.

Δt	IQN-LS	IQN-CLS	IBQN-LS	IQN-ILS
10^{-7}	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1
10^{-6}	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1
10^{-5}	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1
10^{-4}	3 - 2.1	3 - 2.1	3 - 2.2	3 - 2.1
10^{-3}	3 - 2.8	3 - 2.7	3 - 5.4	3 - 3.0
10^{-2}	3 - 2.8	3 - 2.7	3 - 3.1	3 - 3.1
10^{-1}	4 - 4.5	4 - 4.2	4 - 5.9	4 - 4.0
1	6 - 6.7	6 - 7.4	6 - 8.3	6 - 5.8

Table 10.25: $\mathcal{FC}$ required for convergence of the one-dimensional heat equation when the temperature is extrapolated over the time-steps and input-output modes of 5 previous time-steps (non-R1U formulation) are kept; values for the first time-step and average over the first 5 time-steps; solvers are of Least Squares quasi-Newton type in original (i.e. non rank-one update) formulation; $n = 100$; ”div”= divergence or non-convergence after 100 function calls.

Δt	IQN-LS	IQN-CLS	IBQN-LS	IQN-ILS
10^{-7}	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1
10^{-6}	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1
10^{-5}	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1
10^{-4}	3 - 2.1	3 - 2.1	3 - 2.6	3 - 2.1
10^{-3}	3 - 2.9	3 - 2.9	3 - 9.2	3 - 3.0
10^{-2}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
10^{-1}	4 - 4.2	4 - 4.5	4 - 5.8	4 - 4.1
1	6 - 6.8	6 - 7.6	6 - 11.5	6 - 5.9

Table 10.26: $\mathcal{FC}$ required for convergence of the one-dimensional heat equation when the temperature is extrapolated over the time-steps and input-output modes of 10 previous time-steps (non-R1U formulation) are kept; values for the first time-step and average over the first 10 time-steps; solvers are of Least Squares quasi-Newton type in original (i.e. non rank-one update) formulation; $n = 100$; "div"= divergence or non-convergence after 100 function calls.

Δt	IQN-LS	IQN-CLS	IBQN-LS	IQN-ILS
10^{-7}	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1
10^{-6}	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1
10^{-5}	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1
10^{-4}	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1
10^{-3}	3 - 2.1	3 - 2.1	3 - 2.4	3 - 2.1
10^{-2}	3 - 2.7	3 - 2.8	3 - 3.0	3 - 3.0
10^{-1}	3 - 3.0	3 - 3.0	3 - 3.1	3 - 3.1
1	4 - 4.1	4 - 4.2	5 - 7.1	4 - 4.1

Table 10.27: $\mathcal{FC}$ required for convergence of the one-dimensional heat equation when the temperature is extrapolated over the time-steps and input-output modes of 5 previous time-steps (non-R1U formulation) are kept; values for the first time-step and average over the first 10 time-steps; solvers are of Least Squares quasi-Newton type in original (i.e. non rank-one update) formulation; $n = 1000$; "div"= divergence or non-convergence after 100 function calls.

Δt	IQN-LS	IQN-CLS	IBQN-LS	IQN-ILS
10^{-7}	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1
10^{-6}	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1
10^{-5}	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1
10^{-4}	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1
10^{-3}	3 - 2.1	3 - 2.1	3 - 2.5	3 - 2.1
10^{-2}	3 - 2.7	3 - 3.0	3 - 6.1	3 - 3.0
10^{-1}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
1	4 - 4.1	4 - 4.3	5 - 6.1	4 - 4.1

Table 10.28: $\mathcal{FC}$ required for convergence of the one-dimensional heat equation when the temperature is extrapolated over the time-steps and input-output modes of 10 previous time-steps (non-R1U formulation) are kept; values for the first time-step and average over the first 10 time-steps; solvers are of Least Squares quasi-Newton type in original (i.e. non rank-one update) formulation; $n = 1000$; "div"= divergence or non-convergence after 100 function calls.

Δt	IQN-LS	IQN-CLS	IBQN-LS	IQN-ILS
10^{-7}	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1
10^{-6}	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1
10^{-5}	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1
10^{-4}	3 - 2.1	3 - 2.1	3 - div	3 - 2.1
10^{-3}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
10^{-2}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
10^{-1}	4 - 4.0	4 - 4.0	4 - 4.0	4 - 4.0
1	6 - 5.1	6 - 5.1	6 - 5.5	6 - 5.4

Table 10.29: $\mathcal{FC}$ required for convergence of the one-dimensional heat equation problem when the temperature is extrapolated over the time-steps and the initial Jacobian for a new time-step is taken as the final Jacobian from the previous time-step; values for the first time-step and average over the first 10 time-steps; solvers are of Least Squares type in rank-one update formulation; $n = 100$; "div"= divergence or non-convergence after 100 function calls.

Δt	IQN	IQN	IBQN	IQN	IQN	IQN	IBQN	IQN
	-BG	-CBG	-BG	-BB	-CUM	-CCUM	-CUM	-ICUM
10^{-7}	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1
10^{-6}	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1
10^{-5}	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1
10^{-4}	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.2	3 - 2.1
10^{-3}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
10^{-2}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
10^{-1}	4 - 4.0	4 - 4.0	4 - 4.0	4 - 4.0	4 - 4.0	4 - 4.0	4 - 4.0	4 - 4.0
1	5 - 5.0	5 - 5.0	5 - 5.0	5 - 5.0	5 - 5.0	5 - 5.0	5 - 5.0	5 - 5.0

Table 10.30: $\mathcal{FC}$ required for convergence of the one-dimensional heat equation problem when the temperature is extrapolated over the time-steps and the initial Jacobian for a new time-step is taken as the final Jacobian from the previous time-step; values for the first time-step and average over the first 10 time-steps; solvers are of Broyden and Column-Updating type; $n = 100$; "div"= divergence or non-convergence after 100 function calls.

Δt	IQN-LS	IQN-CLS	IBQN-LS	IQN-ILS
10^{-7}	3 - 2.1	3 - 2.1	div	3 - 2.1
10^{-6}	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1
10^{-5}	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1
10^{-4}	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1
10^{-3}	3 - 2.1	3 - 2.1	3 - div	3 - 2.1
10^{-2}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
10^{-1}	3 - 3.0	3 - 3.0	3 - 3.1	3 - 3.0
1	4 - 3.9	4 - 3.9	5 - 4.7	4 - 4.0

Table 10.31: $\mathcal{FC}$ required for convergence of the one-dimensional heat equation problem when the temperature is extrapolated over the time-steps and the initial Jacobian for a new time-step is taken as the final Jacobian from the previous time-step; values for the first time-step and average over the first 10 time-steps; solvers are of Least Squares type in rank-one update formulation; $n = 1000$; "div"= divergence or non-convergence after 100 function calls.

Δt	IQN -BG	IQN -CBG	IBQN -BG	IQN -BB	IQN -CUM	IQN -CCUM	IBQN -CUM	IQN -ICUM
10^{-7}	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1
10^{-6}	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1
10^{-5}	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1
10^{-4}	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.1
10^{-3}	3 - 2.1	3 - 2.1	3 - 2.1	3 - 2.4	3 - 2.1	3 - 2.1	3 - 2.5	3 - 2.1
10^{-2}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
10^{-1}	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0	3 - 3.0
1	4 - 3.9	4 - 3.9	4 - 3.9	4 - 4.0	4 - 3.9	4 - 3.9	4 - 3.9	4 - 4.0

Table 10.32: $\mathcal{FC}$ required for convergence of the one-dimensional heat equation problem when the temperature is extrapolated over the time-steps and the initial Jacobian for a new time-step is taken as the final Jacobian from the previous time-step; values for the first time-step and average over the first 10 time-steps; solvers are of Broyden and Column-Updating type; $n = 1000$; "div"= divergence or non-convergence after 100 function calls.

10.3 Conclusion

The tests in the previous paragraphs have shown that the quasi-Newton Least Squares methods give very good performance on the non-linear one-dimensional flexible tube problem, when compared to Broyden's methods and the Column-Updating methods. The gains increase when the problem becomes "harder", i.e. when more iterations are needed. Methods to further enhance the performance by re-using data from previous time-steps also show great potential to improve the performance.

On the one-dimensional heat equation with variable coefficients, which can be considered a fairly easy problem to solve, the gains are less outspoken, and Broyden's methods and the Column-Updating methods can even outperform the Least Squares methods.

11 Numerical experiments with linear operators

This chapter will serve as an illustration to the variants of the Least Squares methods discussed in §8.4, where we have shown that we can make IQN-LS and IQN-ILS[1] analytically identical to GMRes if we can form $A_K x$, $\forall x \in \mathbb{R}^{n\times 1}$.
We will present three test-cases found in the literature and compare the following methods [107].

- GMRes in the numerically stable form given by Barrett and coworkers [12];
- IQN-LS as given in algorithm 8.4.1 using $\omega_s = 1$ (i.e. analytically equivalent to the standard IQN-LS method);
- IQN-ILS as given in algorithm 8.4.2 using $\omega_s = 1$ (i.e. analytically equivalent to the standard IQN-ILS method);
- IQN-LS as given in algorithm 8.4.1 using the optimal value of ω_s given by (8.114);
- IQN-ILS as given in algorithm 8.4.2 using the optimal value of ω_s given by (8.114);
- IQN-LS as given in algorithm 8.4.3.;

[1] Technically, we would need to use “QN-LS” and “QN-ILS”, instead of “IQN-LS” and “IQN-ILS” as the problems presented in this chapter are defined by $K(p) = 0$ without any relation to interface problems. We will refrain from doing so to keep a uniform nomenclature.

- IQN-ILS as given in algorithm 8.4.3.

11.1 Test matrix from the MATRIX MARKET REPOSITORY

In this test we take for A_K a square non-spd matrix $\in \mathbb{R}^{32\times 32}$ from the MATRIX MARKET REPOSITORY [148] called IBM32.
For the vector b we choose $[b]_i = 1$ ($i = 1, \ldots, n$).
All iterations will start from $p_o = [0\ 0\ \ldots\ 0]^T$. As a convergence requirement we take a relative reduction of the residual $\frac{r_s}{r_o} \leq 10^{-5}$ and measure the number of matrix-vector products necessary to obtain convergence.

As seen in figure 11.1 the addition of an optimal value of the parameter ω_s improves the convergence of the IQN-LS method. Initially the same happens for IQN-ILS (figure 11.2) but the effect is short lived, and even turns into a slight worsening of the convergence at the end.
The second alternative form of both IQN-LS and IQN-ILS results in a convergence pattern that is identical to that of GMRes (as predicted by theory) until the very last iterations where numerical differences start to be felt; IQN-ILS having the smallest deviation with respect to GMRes.
Figure 11.3 shows that IQN-ILS has a more monotone convergence than IQN-LS when setting $\omega_s = 1$, which is also in accordance with theoretical findings. (We recall that for IQN-ILS $\hat{M}'_s$ converges in a monotone manner towards A_K^{-1} while for IQN-LS $\hat{K}'_s$ converges in a monotone manner towards A_K; for more details see §8.1.)
When using the optimal value of ω_s (figure 11.4) both IQN-LS and IQN-ILS are almost identical except for the last iterations. This similarity is even more pronounced for the second alternative form (figure 11.5). (Both are analytically identical for this formulation.)

11.2 One-dimensional advection-diffusion equation

In [55] the following ODE boundary value problem was proposed as a test-case:

$$-\frac{d^2u}{dx^2} + \beta\frac{du}{dx} = 0 \quad \text{on }]0,1[\tag{11.1a}$$

$$u(0) = 1 \tag{11.1b}$$

$$u(1) = 0. \tag{11.1c}$$

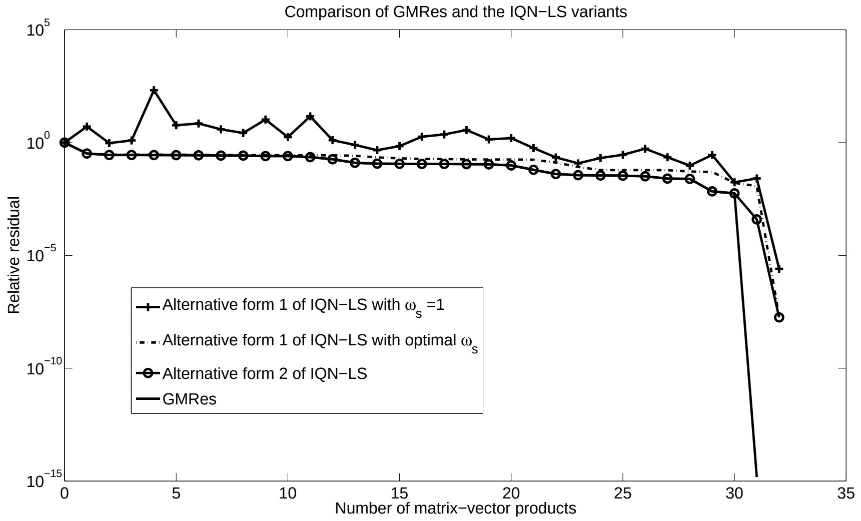

Figure 11.1: Convergence history of the different variants of IQN-LS and of GMRes for the IBM32 matrix test-case.

Equation (11.1) describes a one dimensional advection-diffusion problem and is discretized on a uniform grid with step-size $h = \frac{1}{n+1}$ using first-order finite difference upwind discretization for the advection term and second order central discretization for the diffusion term. This leads to a linear system that can be written as $K(p) = A_K p - b = 0$.

For β we take the value 10^{-1}; as in [55] we take $n = 50$. p_o is chosen as $p_o = [1\ 1\ \ldots\ 1]^T$.
The convergence criterion is a relative reduction of the residual of 10^{-5} and as a performance measure the number of matrix-vector products is used.

In figures 11.6, 11.7 and 11.8 we see that for this test-case neither IQN-LS nor IQN-ILS with $\omega_s = 1$ show monotone convergence. Using the optimal value of ω_s initially yields better convergence but results in a stagnation of the convergence (figures 11.6, 11.7 and 11.9). The cause was found in limit-cycle behavior, due to extremely small values of ω_s after about $\frac{2}{3}n$ iterations.
Figures 11.8 and 11.9 show that IQN-ILS exhibits better convergence performance than IQN-LS for $\omega_s = 1$ and for the optimal value of ω_s. No difference can be seen between IQN-LS and IQN-ILS in the second alternative form and GMRes (figure 11.10).

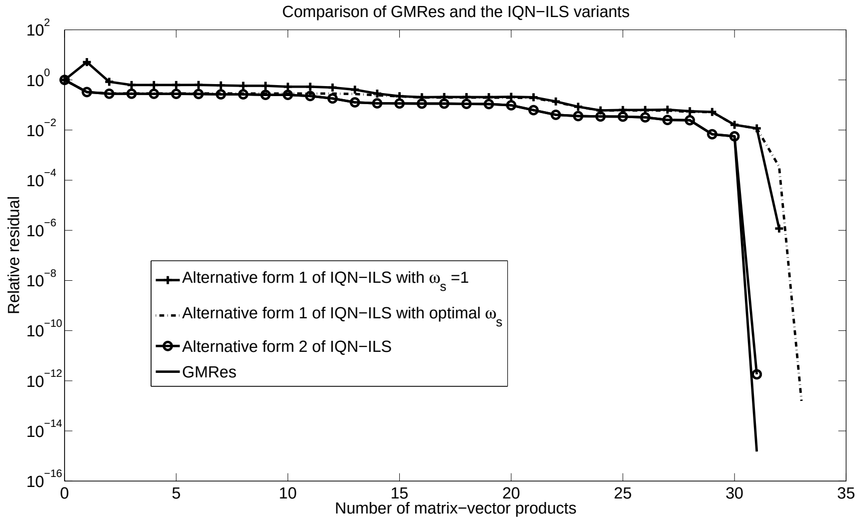

Figure 11.2: Convergence history of the different variants of IQN-ILS and of GMRes for the IBM32 matrix test-case.

11.3 Two-dimensional advection-diffusion equation

We propose the following PDE boundary as a test-case:

$$\vec{\lambda} \cdot \nabla u(x,y) - \nu \nabla^2 u(x,y) = f(x,y) \quad \text{on } \Omega =]-1,1[\times]-1,1[\tag{11.2a}$$

$$u(x,y) = 0 \quad \text{on } \partial\Omega. \tag{11.2b}$$

(11.2) describes a two-dimensional advection-diffusion problem and is discretized using a residual distribution scheme on an unstructured triangular mesh with 441 nodes (361 interior nodes) [43,102]. This leads to a linear system that can be written as $K(p) = A_K p - b = 0$.

We take $\vec{\lambda} = (1,1)$, $\nu = 0.1$, $f(x,y) = (x^2-1)(y^2-1)^2$. We start from $p_o = [0\ 0\ \dots\ 0]^T$.
The convergence criterion is a relative reduction of the residual of 10^{-5} and as a performance measure the number of matrix-vector products is used.

For this test-case IQN-LS with $\omega_s = 1$ shows very erratic convergence behavior and eventually diverges (figures 11.11 and 11.13), while for the same ω_s IQN-ILS converges in a monotone way (figures 11.12 and 11.13).

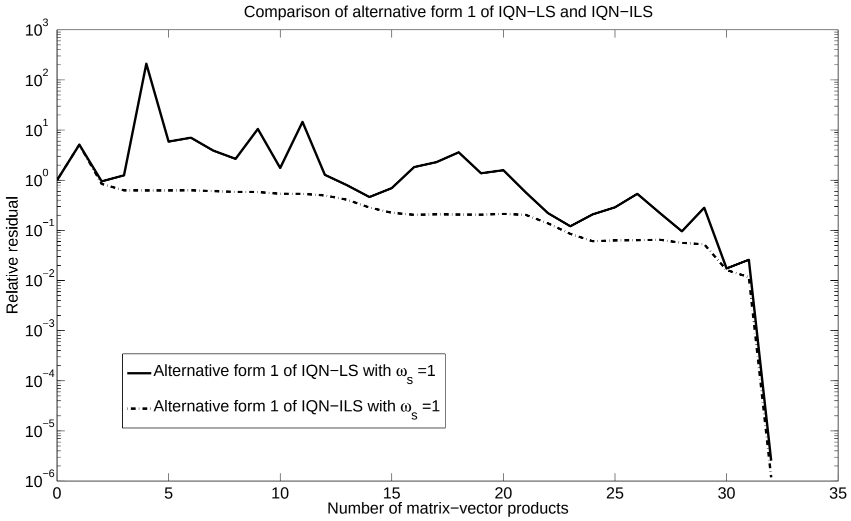

Figure 11.3: Convergence history of the first alternative form of IQN-LS and IQN-ILS with $\omega_s = 1$ for the IBM32 matrix test-case.

Both IQN-LS and IQN-ILS show good performance when using the first alternative form with the optimal value of ω_s, with IQN-ILS slightly outperforming IQN-LS (figures 11.11, 11.12 and 11.14).

For the second alternative form no difference between IQN-LS, IQN-ILS and GM-Res could be discerned (figures 11.11, 11.12 and 11.15).

11.4 Conclusion

This chapter served as an illustration for the theoretical findings in chapter 8. The tests that were chosen allow us to confirm that IQN-LS and IQN-ILS can be transformed to a method that is algebraically equivalent to GMRes by using multiple parameters but without incurring the cost of extra matrix-vector products. Another variant with a single parameter, as treated in chapter 8, shows only slight improvements, if at all, and suffers from poor numerical stability.

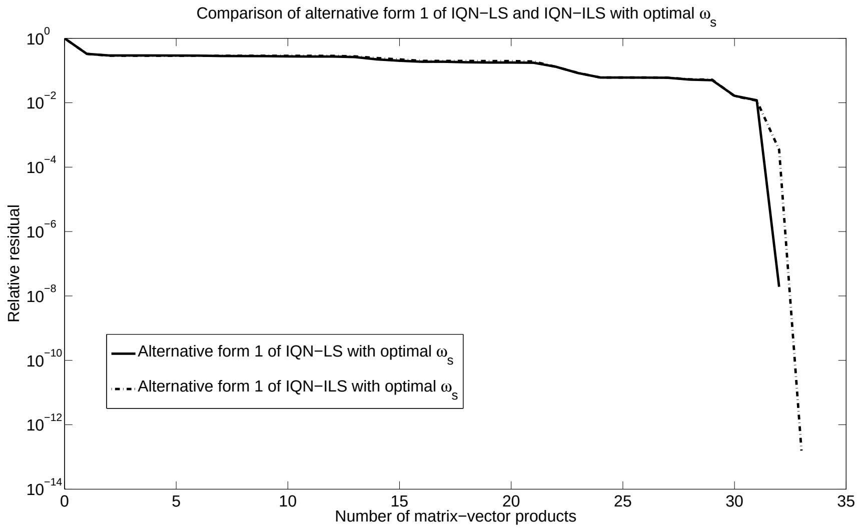

Figure 11.4: Convergence history of the first alternative form of IQN-LS and IQN-ILS with optimal ω_s for the IBM32 matrix test-case.

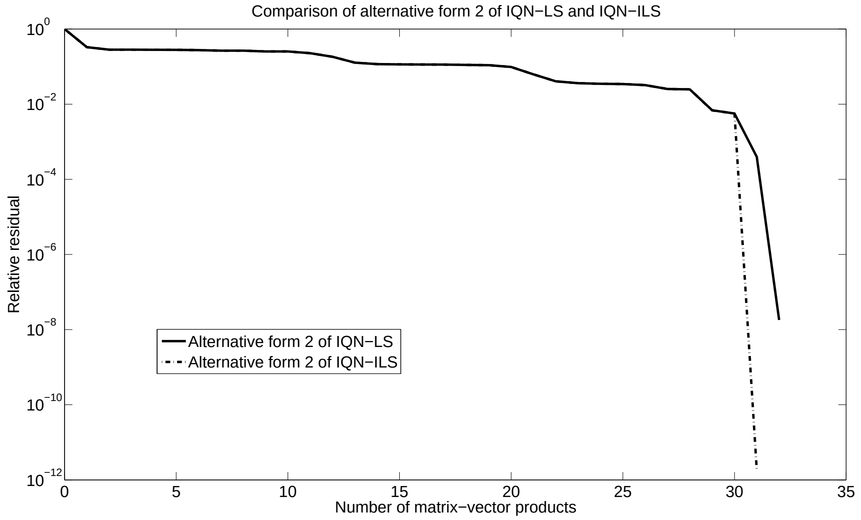

Figure 11.5: Convergence history of the second alternative form of IQN-LS and IQN-ILS for the IBM32 matrix test-case.

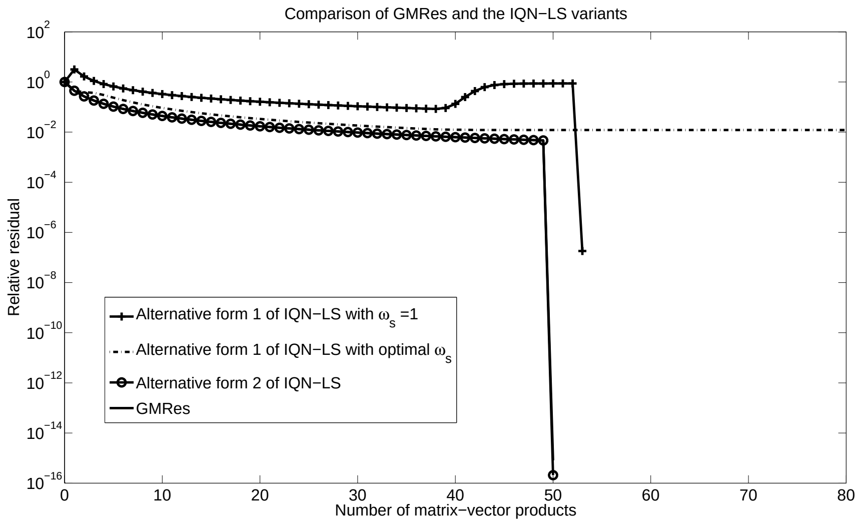

Figure 11.6: Convergence history of the different variants of IQN-LS and of GMRes for the one-dimensional advection-diffusion problem.

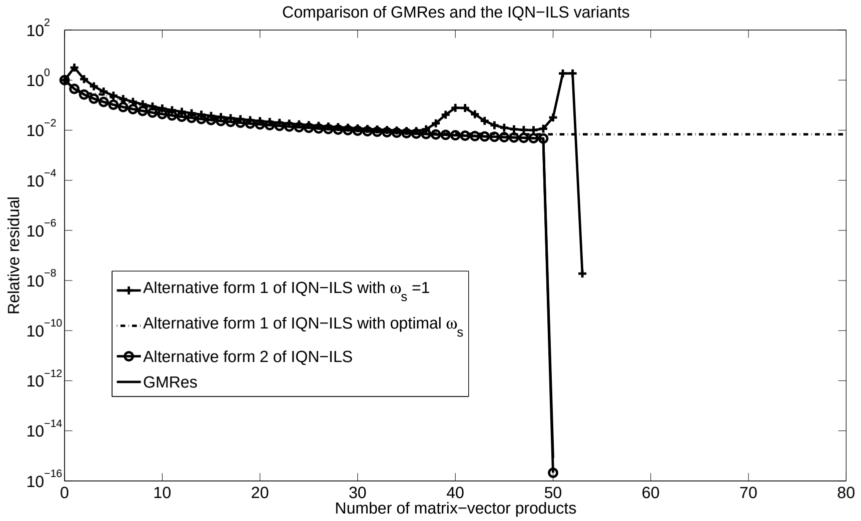

Figure 11.7: Convergence history of the different variants of IQN-ILS and of GMRes for the one-dimensional advection-diffusion problem.

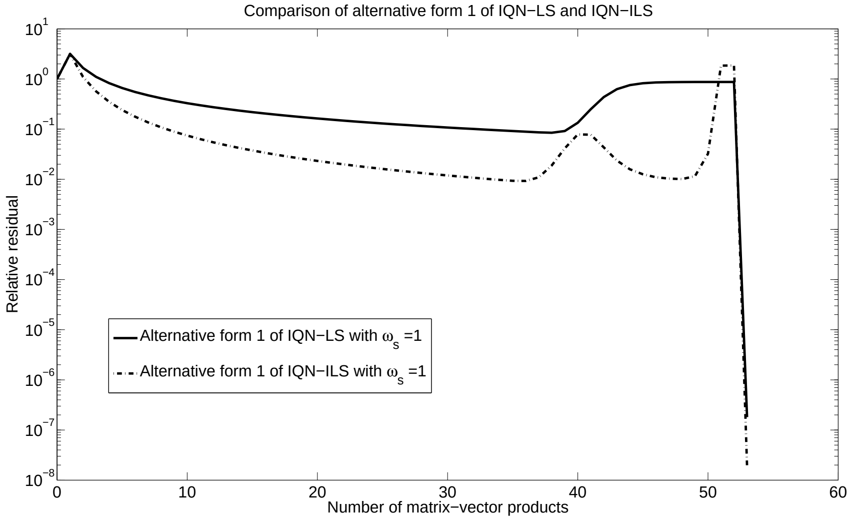

Figure 11.8: Convergence history of the first alternative form of IQN-LS and IQN-ILS with $\omega_s = 1$ for the one-dimensional advection-diffusion problem.

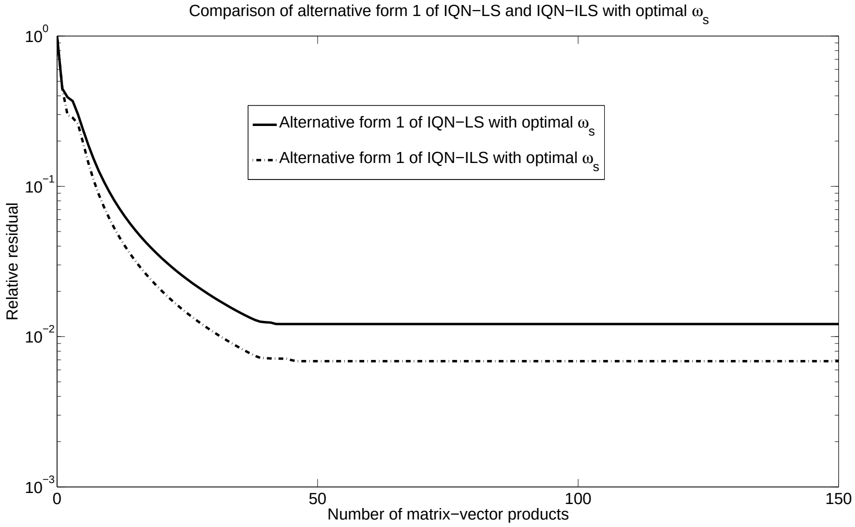

Figure 11.9: Convergence history of the first alternative form of IQN-LS and IQN-ILS with optimal ω_s for the one-dimensional advection-diffusion problem.

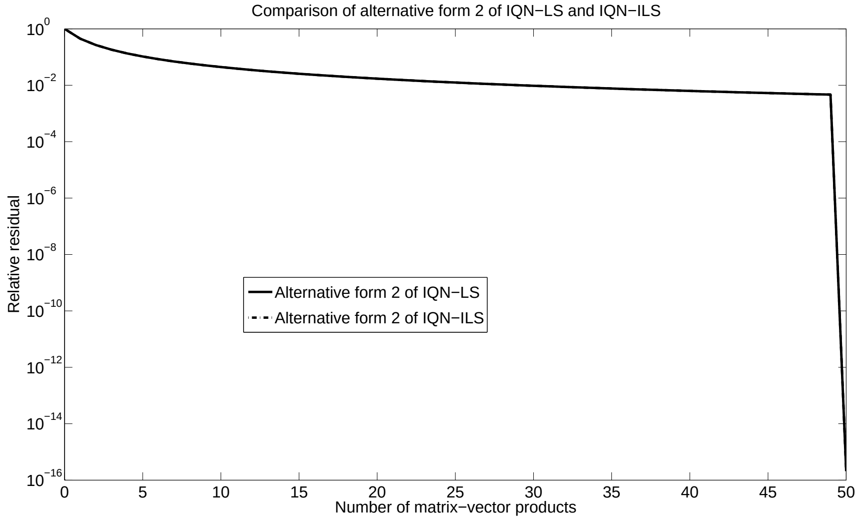

Figure 11.10: Convergence history of the second alternative form of IQN-LS and IQN-ILS for the one-dimensional advection-diffusion problem.

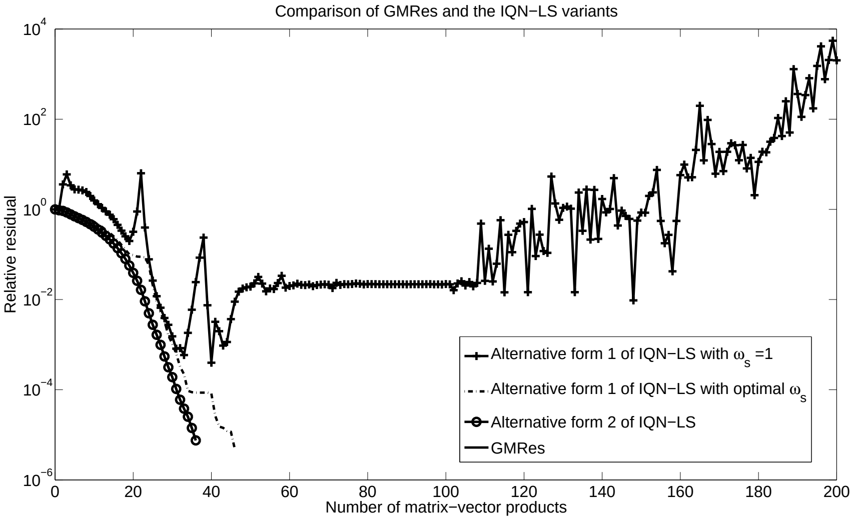

Figure 11.11: Convergence history of the different variants of IQN-LS and of GMRes for the two-dimensional advection-diffusion problem.

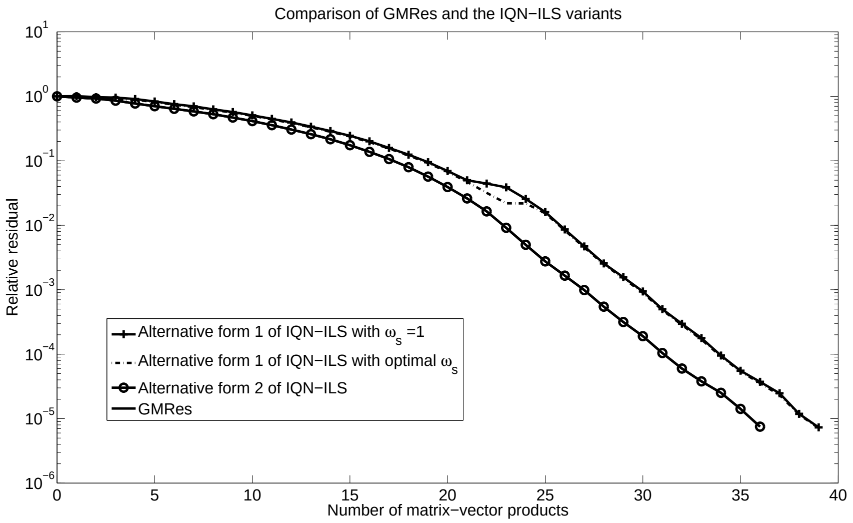

Figure 11.12: Convergence history of the different variants of IQN-ILS and of GMRes for the two-dimensional advection-diffusion problem.

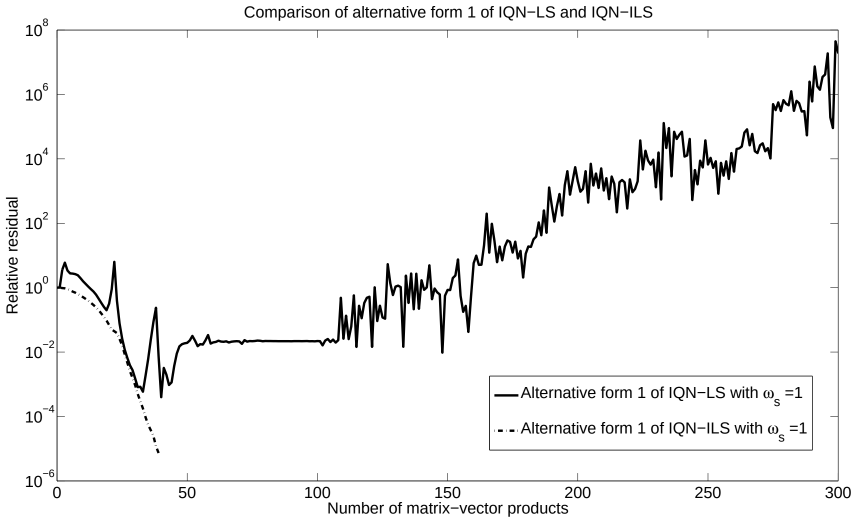

Figure 11.13: Convergence history of the first alternative form of IQN-LS and IQN-ILS with $\omega_s = 1$ for the two-dimensional advection-diffusion problem.

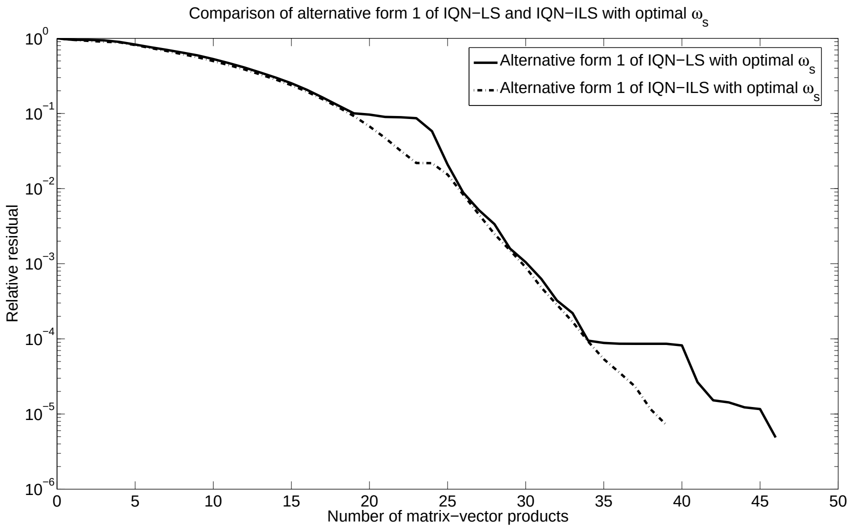

Figure 11.14: Convergence history of the first alternative form of IQN-LS and IQN-ILS with optimal ω_s for the two-dimensional advection-diffusion problem.

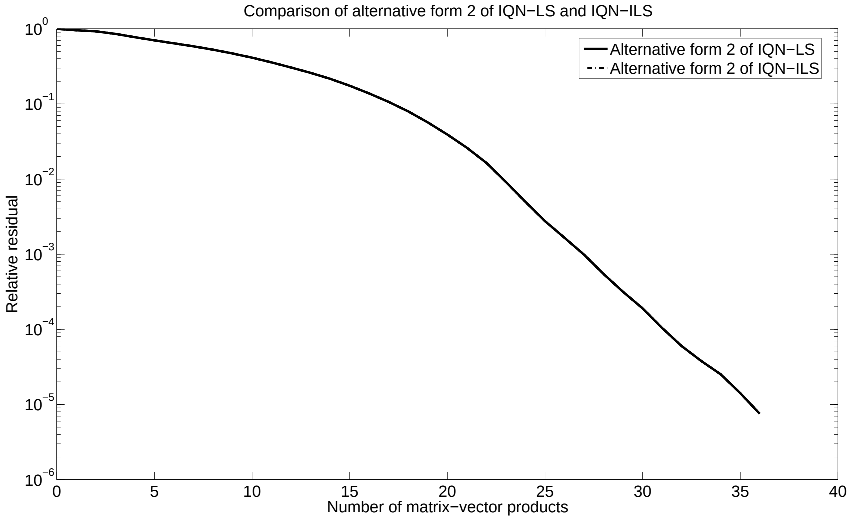

Figure 11.15: Convergence history of the second alternative form of IQN-LS and IQN-ILS for the two-dimensional advection-diffusion problem.

12
Conclusions

In this thesis we started from a partitioned coupling method for fluid-structure interaction problems proposed by Vierendeels earlier this decade. This method was based on the construction of approximate Jacobians of black-box solvers with a Least Squares approach. We formalized the method as being a block quasi-Newton method (Interface Block Quasi-Newton method with Least Squares Jacobian or IBQN-LS) and expanded the original idea of the Least Squares construction of the approximate Jacobian. Ensuing algorithms were IQN-LS (Interface Quasi-Newton method with Least Squares Jacobian), IQN-CLS (Interface Quasi-Newton method with Composed Least Squares Jacobian) and IQN-ILS (Interface Quasi-Newton method with Inverse Least Squares Jacobian).
The four methods were analyzed from a theoretical point of view and compared with existing quasi-Newton method that are applicable to the interface problem that was originally studied. These methods are Broyden's "good " and "bad" method, the Column-Updating method and the Inverse Column-Updating method. These are called IQN-BG, IQN-BB, IQN-CUM and IQN-ICUM respectively in this context. These methods are also adapted to "block" and "composed form" resulting in the new methods IQN-CBG, IBQN-BG, IQN-CCUM and IBQN-CUM. From the theoretical analysis it is concluded that Least Squares methods share similarities with their respective Broyden counterparts, after having re-written the former in rank-one update form. The methods also exhibit a generalized secant property and Least Change Secant Update property. For linear problems it is shown that convergence is guaranteed in at most $n + 1$ iterations (n being the dimension of the solution vector) without the possibility of singularities. This compares very

favorable to the Broyden and CUM methods which, for linear problems, converge in at most $2n$ iterations.

IQN-LS and IQN-ILS are also transformed, by the addition of multiple parameters, but without extra matrix-vector products, to a form that is algebraically equivalent to GMRes.

Approaches to further improve the performance of the Least Squares quasi-Newton methods for a series of (time-dependent) problems are given and tested.

From the numerical experiments we have learned that the quasi-Newton Least Squares methods give very good performance on the non-linear one-dimensional flexible tube problem, when compared to Broyden's methods and the Column-Updating methods. The gains increase when the problem becomes "harder", i.e. when more iterations are needed. Methods to further enhance the performance by re-using data from previous time-steps also show great potential to improve the performance.

On the one-dimensional heat equation with variable coefficients, which can be considered a fairly easy problem to solve, the gains are less outspoken, and Broyden's methods and the Column-Updating methods can even outperform the Least Squares methods.

Tests on linear problems, which served as an illustration for the theoretical findings in chapter 8, allow us to confirm that IQN-LS and IQN-ILS can be transformed to a method that is algebraically equivalent to GMRes by using multiple parameters but without incurring the cost of extra matrix-vector products. Another variant with a single parameter, as treated in chapter 8, shows only slight improvements, if at all, and suffers from poor numerical stability.

A
Matlab Source code

A.1 Main loop

```
clear all
close all
clc

global kappa
kappa=     % Fill in value
global tau
tau=       % Fill in value

% define grid: fine
gridsize=’f’;
% CALL SCRIPT-FILE
first_init_MGM
% CALL SCRIPT-FILE
second_init_MGM
% CALL SCRIPT-FILE
third_init_MGM

for timelabel=1:timesteps % define number of time-steps
    switch recovery
        % if recovery type is MG then we first create
        % an initial Jacobian from a coarser grid.
        case{MG1,MG2} % define case-designator for Multi-Grid

            % set grid to coarse
            gridsize=’c’;
            % change all variables that change when grid is changed
            second_init_MGM

            % create time-dependent BC for this grid
            global UL
            UL=Uo+Amplitude*Uo*(sin(pi*timelabel*tau2/PERIOD))ˆPOWER;

            % Restriction of initial variables to be used on
            % coarse grid
            Ginit_f=Ginit;
            Ginit=RestrictM*(Ginit);
            Gprev_f=Gprev;
            global Gprev
            Gprev=RestrictM*(Gprev);
            Pinit_f=Pinit;
            Pinit=RestrictN*(Pinit);
            Pprev_f=Pprev;
            global Pprev
            Pprev=RestrictN*(Pprev);
            Uinit_f=Uinit;
            Uinit=RestrictN*(Uinit);
            Uprev_f=Uprev;
            global Uprev
            Uprev=RestrictN*(Uprev);

            % CALL SCRIPT-FILE
            % resetting data for solver for start of each time-step
            data_reset

            % CALL SCRIPT-FILE
            % solver
            run(QN_solver_type)     % Define QN-method

            % Jacobian for fine grid built from fine grid
            if fcf<itmax
                 % CALL SCRIPT-FILE
                 create_extrapolated_Jacobian
            else
                break
            end

            switch recovery
                case{MG1}
                    % if recovery type MG1 then initial iterates are
                    % extrapolated from previous time-steps
                    gridsize=’f’;
                    Ginit=Ginit_f;
                    Pinit=Pinit_f;
                    global Gprev
                    Gprev=Gprev_f;
                    global Pprev
                    Pprev=Pprev_f;
                    Uinit=Uinit_f;
                    global Uprev
                    Uprev=Uprev_f;
                case{MG2}
                    % if recovery type MG2 then initial iterates
                    % are prolongated from coarse grid
                    gridsize=’f’;
                    Ginit=ProlongM*(G(:,end));
                    Gprev=Gprev_f;
                    Pinit=ProlongN*(P(:,end));
                    global Pprev
                    Pprev=Pprev_f
                    Uinit=ProlongN*(U(:,end));
                    global Uprev
                    Uprev=Uprev_f;
            end

            second_init_MGM
            % create time-dependent BC
            global UL
            UL=Uo+Amplitude*Uo*(sin(pi*timelabel*tau2/PERIOD))ˆPOWER;

            % CALL SCRIPT-FILE
```

```
            % resetting data for solver for start of each time-step
            data_reset
            % CALL SCRIPT-FILE
            % solver
            run(QN_solver_type) % Define QNmethod

        case{Basic,Addcolumns,PreviousJacR1U}
            global UL
            UL=Uo+Amplitude*Uo*(sin(pi*timelabel*tau2/PERIOD))^POWER;

            % CALL SCRIPT-FILE
            % resetting data for solver for start of each time-step
            data_reset
            % CALL SCRIPT-FILE
            % solver
            run(QN_solver_type) % Define QNmethod
    end


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % storage of solution will run one index ahead
    GFinal(:,timelabel+1)=G(:,end);
    PFinal(:,timelabel+1)=P(:,end);
    UFinal(:,timelabel+1)=U(:,end);
    % resetting data from previous time step
    global Gprev
    global Pprev
    global PRprev
    Gprev=G(:,end);
    Pprev=P(:,end);
    PRprev=P(N,end);
    global Uprev
    Uprev=U(:,end);

    % creating initial iterates for next loop
    switch recovery
        case{Basic,Addcolumns,PreviousJacR1U,MG1}
            % extrapolation
            if timelabel==1
                Ginit=2*GFinal(:,timelabel+1)-GFinal(:,timelabel);
                Pinit=2*PFinal(:,timelabel+1)-PFinal(:,timelabel);
                Uinit=2*UFinal(:,timelabel+1)-UFinal(:,timelabel);
            else
                Ginit=2.5*GFinal(:,timelabel+1)...
                -2*GFinal(:,timelabel)+0.5*GFinal(:,timelabel-1);
                Pinit=2.5*PFinal(:,timelabel+1)...
                -2*PFinal(:,timelabel)+0.5*PFinal(:,timelabel-1);
                Uinit=2.5*UFinal(:,timelabel+1)...
                -2*UFinal(:,timelabel)+0.5*UFinal(:,timelabel-1);
            end
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %     % recovery of Jacobian ?
    switch recovery
        case{Addcolumns}
            InitialjacobianK=DK;
            InitialjacobianM=DM;
            InitialjacobianS=DS;
            InitialjacobianF=DF;
            VV(1:size(V,1),1:size(V,2),timelabel)=V;
            WW(1:size(W,1),1:size(W,2),timelabel)=W;
            VVS(1:size(VS,1),1:size(VS,2),timelabel)=VS;
            WWS(1:size(WS,1),1:size(WS,2),timelabel)=WS;
            VVF(1:size(VF,1),1:size(VF,2),timelabel)=VF;
            WWF(1:size(WF,1),1:size(WF,2),timelabel)=WF;
        case{PreviousJacR1U}
```

```
            InitialjacobianK=DK;
            InitialjacobianM=DM;
            InitialjacobianS=DS;
            InitialjacobianF=DF;
            VV=[];
            WW=[];
            VVS=[];
            WWS=[];
            VVF=[];
            WWF=[];
        otherwise
            InitialjacobianK=-I;
            InitialjacobianM=-I;
            InitialjacobianS=zeros(M,N);
            InitialjacobianF=zeros(N,M);
            VVS=[];
            WWS=[];
            VVF=[];
            WWF=[];
    end

    if fcf==inf
        break
    end

    if fcf==itmax
        break
    end

    % END TIMELOOP
end
```

A.2 first_init_temp.m

```
Nf=                               % set value
contrac_fac =                     % set value
Nc=((Nf-1)/contrac_fac)+1;
Mc=                               % set value
Mf=                               % set value

% Number of time-steps to re-use
nnreuse=10;

% Max number of iterations allowed (outer loop per time-step)
itmax=100;

% Initializing time
timelabel=0;
% Set #timesteps#
timesteps= % set value

V=[];
W=[];
VS=[];
WS=[];
VF=[];
WF=[];
VV=zeros(Nf,itmax,timesteps);
WW=zeros(Nf,itmax,timesteps);
VVS=zeros(Nf,itmax,timesteps);
WWS=zeros(Mf,itmax,timesteps);
VVF=zeros(Mf,itmax,timesteps);
WWF=zeros(Nf,itmax,timesteps);
```

```
% Setting initial relaxation factor
% Set #omega#
omega=                 % set value

% setting parameters for fsolve
options=optimset('MaxFunEvals',10000,'MaxIter',10000,'TolFun',...
eps,'TolX',eps,'Display','off','Diagnostics','On');

% clear solver calls over time-steps on fine and coarse grid
FCF=[];
FCFc=[];

% Solution will contain u,p and g over time-steps
% clear converged solutions per time-step
PFinal=[];
GFinal=[];
UFinal=[];
```

A.3 second_init_temp.m

```
global N
if gridsize=='c'
    N=Nc;
elseif gridsize=='f'
    N=Nf;
end

global M
if gridsize=='c'
    M=Mc;
elseif gridsize=='f'
    M=Mf;
end

global I
I   = eye(N,N);
global O
O   = zeros(N,N);

global Uo
Uo  = 1/kappa;
global Do
Do  = Uo/(tau*N);
global Go
Go=1;
global Po
Po=0;

Amplitude=0.1;
POWER=2;
PERIOD=1;

relrestol=10^(-5);
QRthreshold=1e-8;
```

A.4 third_init_temp.m

```
% create restriction and prolongation matrices
RestrictN=zeros(Nc,Nf);

for k = 1:Nc
    RestrictN(k,1+contrac_fac*(k-1))=1;
```

```
end

ProlongN=zeros(Nf,Nc);

for k = 1:Nc
    for row=1:contrac_fac
        ProlongN(row+contrac_fac*(k-1),k)=...
            (contrac_fac+1-row)/contrac_fac;
        ProlongN(row+contrac_fac*(k-1),k+1)=...
            (-1+row)/contrac_fac;
    end
end

ProlongN=ProlongN(1:Nf,1:Nc);

RestrictM=zeros(Mc,Mf);

for k = 1:Mc
    RestrictM(k,1+contrac_fac*(k-1))=1;
end

ProlongM=zeros(Mf,Mc);

for k = 1:Mc
    for row=1:contrac_fac
        ProlongM(row+contrac_fac*(k-1),k)=...
            (contrac_fac+1-row)/contrac_fac;
        ProlongM(row+contrac_fac*(k-1),k+1)=...
            (-1+row)/contrac_fac;
    end
end
ProlongM=ProlongM(1:Mf,1:Mc);

% Set values of previous time-step
global Gprev
Gprev=Go*ones(N,1);
global Pprev
Pprev=Po*ones(N,1);
global PRprev
PRprev=Pprev(N);
global Uprev
Uprev=Uo*ones(N,1);

Ginit=Gprev;
Pinit=Pprev;
Uinit=Uprev;

% Solution will contain u,p and g over time-steps
PFinal(:,timelabel+1)=Pprev;
GFinal(:,timelabel+1)=Gprev;
UFinal(:,timelabel+1)=Uprev;

PFinalc(:,timelabel+1)=RestrictN*Pprev;
GFinalc(:,timelabel+1)=RestrictM*Gprev;
UFinalc(:,timelabel+1)=RestrictN*Uprev;

% Initialization of Jacobian & base for first time-step
InitialjacobianK=-I;
InitialjacobianM=-I;
InitialjacobianS=zeros(M,N);
InitialjacobianF=zeros(N,M);
InitialL=[];
InitialLS=[];
InitialLF=[];
```

A.5 data_reset.m

```
Residual=[];
Hp=[];
Fg=[];
Sp=[];
Kp=[];

d=[];

deltap=[];
deltaK=[];
deltaG=[];
deltaS=[];

fcs=0;
fcf=0;

G=[];
global G
G(:,1)=Ginit;
P=[];
global P
P(:,1)=Pinit;
U=[];
U(:,1)=Uinit;

% Initial orthogonal base
L =InitialL;

LS=InitialLS;
LF=InitialLF;

% Initial approx. Jacobian
if gridsize==’f’
    DM=InitialjacobianM;
    DK=InitialjacobianK;
    DF=InitialjacobianF;
    DS=InitialjacobianS;
elseif gridsize==’c’
    DM=-eye(Nc,Nc);
    DK=-eye(Nc,Nc);
    DF=zeros(Nc,Nc);
    DS=zeros(Nc,Nc);
end
```

A.6 create_extrapolated_Jacobian.m

```
if isempty(P)==0
    P_prol=ProlongN*P;
end

if isempty(G)==0
    G_prol=ProlongM*G;
end

if isempty(U)==0
    U_prol=ProlongN*U;
end

if isempty(Hp)==0
    Hp_prol=ProlongN*Hp;
end
```

```
if isempty(Kp)==0
    Kp_prol=ProlongN*Kp;
end

if isempty(Sp)==0
    Sp_prol=ProlongM*Sp;
end

if isempty(Fg)==0
    Fg_prol=ProlongN*Fg;
end


if QN_solver_type==’QN_solver_IQN_’
    Ptemp_prol =...
    P_prol(:,max(1,size( P_prol,2)-N):size( P_prol,2)-1);
    Fgtemp_prol=...
    Fg_prol(:,max(1,size(Fg_prol,2)-N):size(Fg_prol,2)-1);

    V_prol=( P_prol(:,s)*ones(1,size( Ptemp_prol,2)))- Ptemp_prol;
    W_prol=(Fg_prol(:,s)*ones(1,size(Fgtemp_prol,2)))-Fgtemp_prol;
    [ntemp mtemp] = size(W_prol);
    V_prol=V_prol(1:ntemp,1:mtemp);
    clear ntemp
    clear mtemp

    InitialjacobianK=W_prol*((V_prol’*V_prol)\V_prol’)-eye(Nf,Nf);

elseif QN_solver_type==’QN_solver_IQNI’
    Kptemp_prol=...
    Kp_prol(:,max(1,size(Kp_prol,2)-N+1):size(Kp_prol,2)-1);
    Fgtemp_prol=...
    Fg_prol(:,max(1,size(Fg_prol,2)-N+1):size(Fg_prol,2)-1);

    V_prol=(Kp_prol(:,s)*ones(1,size(Kptemp_prol,2)))-Kptemp_prol;
    W_prol=(Fg_prol(:,s)*ones(1,size(Fgtemp_prol,2)))-Fgtemp_prol;
    V_prol=V_prol(:,max(1,size(V_prol,2)-N+1):size(V_prol,2));
    W_prol=W_prol(:,max(1,size(W_prol,2)-N+1):size(W_prol,2));

    InitialjacobianM=W_prol*((V_prol’*V_prol)\V_prol’)-eye(Nf,Nf);

elseif QN_solver_type==’QN_solver_IQNC’
    Ptemp_prol =...
    P_prol(:,max(1,size( P_prol,2)-N+1):size( P_prol,2)-1);
    Sptemp_prol=...
    Sp_prol(:,max(1,size(Sp_prol,2)-N+1):size(Sp_prol,2)-1);
    Gtemp_prol =...
    G_prol(:,max(1,size( G_prol,2)-N+1):size( G_prol,2)-1);
    Fgtemp_prol=...
    Fg_prol(:,max(1,size(Fg_prol,2)-N+1):size(Fg_prol,2)-1);

    VS_prol=( P_prol(:,s)*ones(1,size( Ptemp_prol,2)))- Ptemp_prol;
    WS_prol=(Sp_prol(:,s)*ones(1,size(Sptemp_prol,2)))-Sptemp_prol;
    VS_prol=VS_prol(:,max(1,size(VS_prol,2)-N+1):size(VS_prol,2));
    WS_prol=WS_prol(:,max(1,size(WS_prol,2)-N+1):size(WS_prol,2));

    VF_prol=( G_prol(:,s)*ones(1,size( Gtemp_prol,2)))- Gtemp_prol;
    WF_prol=(Fg_prol(:,s)*ones(1,size(Fgtemp_prol,2)))-Fgtemp_prol;
    VF_prol=VF_prol(:,max(1,size(VF_prol,2)-N+1):size(VF_prol,2));
    WF_prol=WF_prol(:,max(1,size(WF_prol,2)-N+1):size(WF_prol,2));

    InitialjacobianS=WS_prol*((VS_prol’*VS_prol)\VS_prol’);
    InitialjacobianF=WF_prol*((VF_prol’*VF_prol)\VF_prol’);

elseif QN_solver_type==’QN_solver_IBQN’
    Ptemp_prol =...
    P_prol(:,max(1,size( P_prol,2)-N+1):size( P_prol,2)-1);
    Sptemp_prol=...
    Sp_prol(:,max(1,size(Sp_prol,2)-N+1):size(Sp_prol,2)-1);
    Gtemp_prol =...
    G_prol(:,max(1,size( G_prol,2)-N+1):size( G_prol,2)-1);
    Fgtemp_prol=...
    Fg_prol(:,max(1,size(Fg_prol,2)-N+1):size(Fg_prol,2)-1);

    VS_prol=( P_prol(:,s)*ones(1,size( Ptemp_prol,2)))-Ptemp_prol;
    WS_prol=(Sp_prol(:,s)*ones(1,size(Sptemp_prol,2)))-Sptemp_prol;
    VS_prol=VS_prol(:,max(1,size(VS_prol,2)-N+1):size(VS_prol,2));
    WS_prol=WS_prol(:,max(1,size(WS_prol,2)-N+1):size(WS_prol,2));

    VF_prol=( G_prol(:,s)*ones(1,size( Gtemp_prol,2)))-Gtemp_prol;
    WF_prol=(Fg_prol(:,s)*ones(1,size(Fgtemp_prol,2)))-Fgtemp_prol;
    VF_prol=VF_prol(:,max(1,size(VF_prol,2)-N+1):size(VF_prol,2));
    WF_prol=WF_prol(:,max(1,size(WF_prol,2)-N+1):size(WF_prol,2));

    InitialjacobianS=WS_prol*((VS_prol'*VS_prol)\VS_prol');
    InitialjacobianF=WF_prol*((VF_prol'*VF_prol)\VF_prol');
end

if recovery==MG2
    Ptemp=P(:,end);
    Pinit_temp=ProlongN*Ptemp;
    Gtemp=G(:,end);
    Ginit_temp=ProlongM*Gtemp;
    Utemp=U(:,end);
    Uinit_temp=ProlongN*Utemp;
end

GFinalc(:,timelabel+1)=G(:,end);
PFinalc(:,timelabel+1)=P(:,end);
UFinalc(:,timelabel+1)=U(:,end);
```

A.7 IQN solver: QN_solver_IQN_.m

```
[Sp(:,1) fval flowflag]=fsolve(Sfun_gen,[G(:,1)],options);
fcs=fcs+1;

global G
G(:,1)=Sp(:,1);

[fg fval flowflag]=fsolve(Ffun_gen,[P(:,1);U(:,1)],options);
fcf=fcf+1;

Fg(:,1)=fg(1:N);
U(:,2) =fg(N+1:2*N);

Kp(:,1)=Fg(:,1)-P(:,1);
Residual(1)=norm(Kp(:,1));

switch recovery
    case{Basic}
        Ptemp=Fg(:,1);
        global P
        P(:,2)=omega*Ptemp+(1-omega)*P(:,1);
        clear Ptemp
    case{Addcolumns,PreviousJacR1U}
        if timelabel==1
            Ptemp=Fg(:,1);
            global P
            P(:,2)=omega*Ptemp+(1-omega)*P(:,1);
            clear Ptemp
        else
```

```
            global P
            P(:,2)=P(:,1)-DK\Kp(:,1);
        end
    case{MG1,MG2}
        global P
        if gridsize==’c’
            Ptemp=Fg(:,1);
            global P
            P(:,2)=omega*Ptemp+(1-omega)*P(:,1);
            clear Ptemp
        elseif gridsize==’f’
            P(:,2)=P(:,1)-DK\Kp(:,1);
        end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Main (outer) iteration loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for s=2:itmax
    [Sp(:,s) fval flowflag]=fsolve(Sfun_gen,[G(:,s-1)],options);
    fcs=fcs+1;

    global G
    G(:,s)=Sp(:,s);

    [fg fval flowflag]=fsolve(Ffun_gen,[P(:,s);U(:,s)],options);
    fcf=fcf+1;

    Fg(:,s)  =fg(1:N);
    U(:,s+1) =fg(N+1:2*N);

    Kp(:,s)=Fg(:,s)-P(:,s);
    Residual(s)=norm(Kp(:,s));

    if Residual(s)/Residual(1) < relrestol
        break
    end


    if R1U_type==’0’        % LS in basic formulation
        Ptemp = P(:,max(1,size( P,2)-N):size( P,2)-1);
        Fgtemp=Fg(:,max(1,size(Fg,2)-N):size(Fg,2)-1);

        V=(P(:,s)*ones(1,size(Ptemp,2)))-fliplr(Ptemp);
        W=(Fg(:,s)*ones(1,size(Fgtemp,2)))-fliplr(Fgtemp);

        VX=[];
        WX=[];

        if recovery==Addcolumns
            for kk=max(1,timelabel-nnreuse):timelabel-1
                VX=[VV(:,1:FCF(kk)-2,kk) VX];
                WX=[WW(:,1:FCF(kk)-2,kk) WX];
            end
        end

        VX=[V VX];
        WX=[W WX];
        [VX WX] =DoQR(VX,WX,QRthreshold);
        DK=WX*((VX’*VX)\VX’)-I;

    elseif R1U_type==’L’    % LS in R1U formulation
        deltap=P(:,s)-P(:,s-1);
        deltaK=(Kp(:,s))-(Kp(:,s-1));

        if min(size(L))<1
```

```
            Lnew =(deltap)  /norm(deltap);
            ALnew=(deltaK - DK*deltap)/norm(deltap );
        else
            Lnew =(deltap - L*L’*deltap)  /norm(deltap - L*L’*deltap);
            ALnew=(deltaK - DK*deltap)    /norm(deltap - L*L’*deltap);
        end

        R1U = ALnew*Lnew’;
        DK=DK+R1U;

        L=[L Lnew];
        L=L(:,max(1,size(L,2)-N+2):size(L,2));

        if max(max(isnan(L)))==1
            fcf=inf;
            break
        end

    elseif R1U_type==’B’    % R1U Broyden
        deltap=P(:,s)-P(:,s-1);
        deltaK=Kp(:,s)-Kp(:,s-1);

        R1U =(deltaK - DK*deltap)*deltap’/dot(deltap,deltap);
        DK =DK+R1U;

    elseif R1U_type==’C’    % R1U CUM
        deltap=P(:,s)-P(:,s-1);
        deltaK=Kp(:,s)-Kp(:,s-1);
        [aa kk]=max(abs(deltap));
        ej=zeros(N,1);
        ej(kk)=1;

        R1U =(deltaK - DK*deltap)*ej’/dot(ej,deltap);
        DK =DK+R1U;
    end

    global P
    P(:,s+1)=P(:,s)-DK\Kp(:,s);

    if max(max(isnan(P)))==1
        fcf=inf;
        fcs=inf;
        break
    end
end
```

A.8 IQN-C solver: QN_solver_IQNC.m

```
[Sp(:,1) fval flowflag]=fsolve(Sfun_gen,[G(:,1)],options);
fcs=fcs+1;

global G
G(:,1)=Sp(:,1);

[fg fval flowflag]=fsolve(Ffun_gen,[P(:,1);U(:,1)],options);
fcf=fcf+1;

Fg(:,1)=fg(1:N);
U(:,2) =fg(N+1:2*N);

Kp(:,1)=Fg(:,1)-P(:,1);
Residual(1)=norm(Kp(:,1));

switch recovery
```

```
    case{Basic}
        Ptemp=Fg(:,1);
        global P
        P(:,2)=omega*Ptemp+(1-omega)*P(:,1);
        clear Ptemp
    case{Addcolumns,PreviousJacR1U}
        if timelabel==1
            Ptemp=Fg(:,1);
            global P
            P(:,2)=omega*Ptemp+(1-omega)*P(:,1);
            clear Ptemp
        else
            global P
            P(:,2)=P(:,1)-DK\Kp(:,1);
        end
    case{MG1,MG2}
        global P
        if gridsize==’c’
            Ptemp=Fg(:,1);
            global P
            P(:,2)=omega*Ptemp+(1-omega)*P(:,1);
            clear Ptemp
        elseif gridsize==’f’
            P(:,2)=P(:,1)-DK\Kp(:,1);
        end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Main (outer) iteration loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for s=2:itmax
    [Sp(:,s) fval flowflag]=fsolve(Sfun_gen,[G(:,s-1)],options);
    fcs=fcs+1;

    global G
    G(:,s)=Sp(:,s);

    [fg fval flowflag]=fsolve(Ffun_gen,[P(:,s);U(:,s)],options);
    Fg(:,s)  =fg(1:N);
    U(:,s+1) =fg(N+1:2*N);

    Kp(:,s)=Fg(:,s)-P(:,s);
    Residual(s)=norm(Kp(:,s));

    if Residual(s)/Residual(1) < relrestol
        break
    end

    if R1U_type==’0’        % LS in basic formulation
        Ptemp = P(:,max(1,size( P,2)-N):size( P,2)-1);
        Sptemp=Sp(:,max(1,size(Sp,2)-N):size(Sp,2)-1);
        Gtemp = G(:,max(1,size( G,2)-N):size( G,2)-1);
        Fgtemp=Fg(:,max(1,size(Fg,2)-N):size(Fg,2)-1);

        VS=(P(:,s)*ones(1,size(Ptemp,2)))-fliplr(Ptemp);
        WS=(Sp(:,s)*ones(1,size(Sptemp,2)))-fliplr(Sptemp);
        VF=(G(:,s)*ones(1,size(Gtemp,2)))-fliplr(Gtemp);
        WF=(Fg(:,s)*ones(1,size(Fgtemp,2)))-fliplr(Fgtemp);

        VSX=[];
        WSX=[];
        VFX=[];
        WFX=[];

        if recovery==Addcolumns
            for kk=max(1,timelabel-nnreuse):timelabel-1
                VSX=[VVS(:,1:FCF(kk)-2,kk) VSX];
                WSX=[WWS(:,1:FCF(kk)-2,kk) WSX];
                VFX=[VVF(:,1:FCF(kk)-2,kk) VFX];
                WFX=[WWF(:,1:FCF(kk)-2,kk) WFX];
            end
        end
        VSX=[VS VSX];
        WSX=[WS WSX];
        VFX=[VF VFX];
        WFX=[WF WFX];

        [VSX WSX] =DoQR(VSX,WSX,QRthreshold);
        [VFX WFX] =DoQR(VFX,WFX,QRthreshold);

        DS=WSX*((VSX’*VSX)\VSX’);
        DF=WFX*((VFX’*VFX)\VFX’);
        DK=DF*DS-I;

    elseif R1U_type==’L’    % LS in R1U formulation
        deltap= P(:,s)-P(:,s-1);
        deltaS=Sp(:,s)-Sp(:,s-1);

        if min(size(LS))<1
            LSnew =(deltap)  /norm(deltap);
            ALSnew=(deltaS - DS*deltap)/norm(deltap);
        else
            LSnew =(deltap-LS*LS’*deltap)/norm(deltap-LS*LS’*deltap);
            ALSnew=(deltaS-DS*deltap)    /norm(deltap-LS*LS’*deltap);
        end

        R1US = ALSnew*LSnew’;
        DS=DS+R1US;

        deltaG=G(:,s)-G(:,s-1);
        deltaF=Fg(:,s)-Fg(:,s-1);

        if min(size(LF))<1
            LFnew =(deltaG)  /norm(deltaG);
            ALFnew=(deltaF - DF*deltaG)/norm(deltaG );
        else
            LFnew =(deltaG-LF*LF’*deltaG)/norm(deltaG-LF*LF’*deltaG);
            ALFnew=(deltaF-DF*deltaG)    /norm(deltaG-LF*LF’*deltaG);
        end

        R1UF = ALFnew*LFnew’;
        DF=DF+R1UF;
        DK=DF*DS-I;

        LS=[LS LSnew];
        LS=LS(:,max(1,size(LS,2)-N+2):size(LS,2));
        LF=[LF LFnew];
        LF=LF(:,max(1,size(LF,2)-N+2):size(LF,2));

        if max(max(isnan(LS)))==1
            fcf=inf;
            break
        end

        if max(max(isnan(LF)))==1
            fcf=inf;
            break
        end

    elseif R1U_type==’B’    % R1U Broyden
        deltap= P(:,s)-P(:,s-1);
        deltaS=Sp(:,s)-Sp(:,s-1);

        R1U = (deltaS - DS*deltap)*deltap’/dot(deltap,deltap);
```

```
        DS=DS+R1U;

        deltaG=G(:,s)-G(:,s-1);
        deltaF=Fg(:,s)-Fg(:,s-1);

        R1U =(deltaF - DF∗deltaG)∗deltaG’/dot(deltaG,deltaG);
        DF=DF+R1U;
        DK=DF∗DS-I;

    elseif R1U_type==’C’    % R1U CUM
        deltap= P(:,s)-P(:,s-1);
        deltaS=Sp(:,s)-Sp(:,s-1);
        [aa kk]=max(abs(deltap));
        ej=zeros(N,1);
        ej(kk)=1;

        R1U = (deltaS - DS∗deltap)∗ej’/dot(ej,deltap);
        DS=DS+R1U;

        deltaG=G(:,s)-G(:,s-1);
        deltaF=Fg(:,s)-Fg(:,s-1);
        [aa kk]=max(abs(deltaG));
        ej=zeros(M,1);
        ej(kk)=1;

        R1U =(deltaF - DF∗deltaG)∗ej’/dot(ej,deltaG);
        DF=DF+R1U;
        DK=DF∗DS-I;
    end

    global P
    P(:,s+1)=P(:,s)-DK\Kp(:,s);

    if max(max(isnan(P)))==1
        fcf=inf;
        fcs=inf;
        break
    end
end
```

A.9 IBQN solver: QN_solver_IBQN.m

```
[Sp(:,1) fval flowflag]=fsolve(Sfun_gen,[G(:,1)],options);
fcs=fcs+1;

global G
G(:,1)=Sp(:,1);

[fg fval flowflag]=fsolve(Ffun_gen,[P(:,1);U(:,1)],options);
fcf=fcf+1;

Fg(:,1)=fg(1:N);
U(:,2) =fg(N+1:2∗N);

Kp(:,1)=Fg(:,1)-P(:,1);
Residual(1)=norm(Kp(:,1));

switch recovery
    case{Basic}
        Ptemp=Fg(:,1);
        global P
        P(:,2)=omega∗Ptemp+(1-omega)∗P(:,1);
        clear Ptemp
    case{Addcolumns,PreviousJacR1U}
```

```
        if timelabel==1
            Ptemp=Fg(:,1);
            global P
            P(:,2)=omega*Ptemp+(1-omega)*P(:,1);
            clear Ptemp
        else
            global P
            P(:,2)=(eye(N,N)-DF*DS)\(Fg(:,1) +...
                   DF*(Sp(:,1) - DS*P(:,1) - G(:,1)));
        end
    case{MG1,MG2}
        global P
        if gridsize=='c'
            Ptemp=Fg(:,1);
            global P
            P(:,2)=omega*Ptemp+(1-omega)*P(:,1);
            clear Ptemp
        elseif gridsize=='f'
            P(:,2)=P(:,1)-DK\Kp(:,1);
        end
end

[Sp(:,2) fval flowflag]=fsolve(Sfun_gen,[G(:,1)],options);
fcs=fcs+1;

global G G(:,2)=Sp(:,2);

if R1U_type=='0'
    VS=P(:,2)-P(:,1);
    WS=Sp(:,2)-Sp(:,1);
    VSX=[];
    WSX=[];

    if recovery==Addcolumns
        for kk=max(1,timelabel-nnreuse):timelabel-1
            VSX=[VVS(:,1:FCF(kk)-1,kk) VSX];
            WSX=[WWS(:,1:FCF(kk)-1,kk) WSX];
        end
    end
    VSX=[VS VSX];
    WSX=[WS WSX];

    [VSX WSX] =DoQR(VSX,WSX,QRthreshold);
    DS=WSX*((VSX'*VSX)\VSX');

elseif R1U_type=='L'
    deltap=P(:,2)-P(:,1);
    deltaS=Sp(:,2)-Sp(:,1);

    LSnew =(deltap)  /norm(deltap);
    ALSnew=(deltaS - DS*deltap)/norm(deltap);

    R1US = ALSnew*LSnew';
    DS=DS+R1US;
    LS=[LS LSnew];

elseif R1U_type=='B'
    deltap=P(:,2)-P(:,1);
    deltaG=G(:,2)-G(:,1);

    R1US=(deltaG - DS*deltap)*deltap'/dot(deltap,deltap);
    DS=DS+R1US;

elseif R1U_type=='C'
    deltap=P(:,2)-P(:,1);
    deltaG=G(:,2)-G(:,1);
    [aa kk]=max(abs(deltap));
    ej=zeros(N,1);
    ej(kk)=1;
    R1US=(deltaG - DS*deltap)*ej'/dot(ej,deltap);
    DS=DS+R1US;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Main (outer) iteration loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for s=2:itmax
    [fg fval flowflag]=fsolve(Ffun_gen,[P(:,s);U(:,s)],options);
    fcf=fcf+1;

    Fg(:,s)  =fg(1:N);
    U(:,s+1) =fg(N+1:2*N);

    Kp(:,s)=Fg(:,s)-P(:,s);
    Residual(s)=norm(Kp(:,s));

    if Residual(s)/Residual(1) < relrestol
        break
    end

    if R1U_type=='0'
        Gtemp = G(:,max(1,size( G,2)-N):size( G,2)-1);
        Fgtemp=Fg(:,max(1,size(Fg,2)-N):size(Fg,2)-1);

        VF=(G(:,s)*ones(1,size(Gtemp,2)))-fliplr(Gtemp);
        WF=(Fg(:,s)*ones(1,size(Fgtemp,2)))-fliplr(Fgtemp);

        VFX=[];
        WFX=[];

        if recovery==Addcolumns
            for kk=max(1,timelabel-nnreuse):timelabel-1
                VFX=[VVF(:,1:FCF(kk)-2,kk) VFX];
                WFX=[WWF(:,1:FCF(kk)-2,kk) WFX];
            end
        end

        VFX=[VF VFX];
        WFX=[WF WFX];

        [VFX WFX] =DoQR(VFX,WFX,QRthreshold);

        DF=WFX*((VFX'*VFX)\VFX');

    elseif R1U_type=='L'
        deltaG=G(:,s)-G(:,s-1);
        deltaF=Fg(:,s)-Fg(:,s-1);

        if min(size(LF))<1
            LFnew =(deltaG)  /norm(deltaG);
            ALFnew=(deltaF - DF*deltaG)/norm(deltaG );
        else
            LFnew =(deltaG-LF*LF'*deltaG)/norm(deltaG-LF*LF'*deltaG);
            ALFnew=(deltaF-DF*deltaG)    /norm(deltaG-LF*LF'*deltaG);
        end

        R1UF = ALFnew*LFnew';
        DF=DF+R1UF;

        LF=[LF LFnew];
        LF=LF(:,max(1,size(LF,2)-N+2):size(LF,2));

        if max(max(isnan(LF)))==1
```

```
            fcf=inf;
            break
        end

    elseif R1U_type==’B’
        deltaG=G(:,s)-G(:,s-1);
        deltaF=Fg(:,s)-Fg(:,s-1);

        R1UF =(deltaF - DF*deltaG)*deltaG’/dot(deltaG,deltaG);
        DF=DF+R1UF;

    elseif R1U_type==’C’
        deltaG=G(:,s)-G(:,s-1);
        deltaF=Fg(:,s)-Fg(:,s-1);

        [aa kk]=max(abs(deltaG));
        ej=zeros(M,1);
        ej(kk)=1;
        R1UF=(deltaF - DF*deltaG)*ej’/dot(ej,deltaG);
        DF=DF+R1UF;
    end

    global P
    P(:,s+1)=(eye(N,N)-DF*DS)\(Fg(:,s) +...
             DF*(Sp(:,s) - DS*P(:,s) - G(:,s)));

    if max(max(isnan(P)))==1
        fcf=inf;
        fcs=inf;
        break
    end

    [Sp(:,s+1) fval flowflag]=fsolve(Sfun_gen,[G(:,s)],options);
    fcs=fcs+1;

    if R1U_type==’0’
        Ptemp = P(:,max(1,size( P,2)-N):size( P,2)-1);
        Sptemp=Sp(:,max(1,size(Sp,2)-N):size(Sp,2)-1);

        VS=(P(:,s+1)*ones(1,size(Ptemp,2)))-fliplr(Ptemp);
        WS=(Sp(:,s+1)*ones(1,size(Sptemp,2)))-fliplr(Sptemp);

        VSX=[];
        WSX=[];

        if recovery==Addcolumns
            for kk=max(1,timelabel-nnreuse):timelabel-1
                VSX=[VVS(:,1:FCF(kk)-1,kk) VSX];
                WSX=[WWS(:,1:FCF(kk)-1,kk) WSX];
            end
        end

        VSX=[VS VSX];
        WSX=[WS WSX];

        [VSX WSX] =DoQR(VSX,WSX,QRthreshold);

        DS=WSX*((VSX’*VSX)\VSX’);

    elseif R1U_type==’L’
        deltap= P(:,s+1)-P(:,s);
        deltaS=Sp(:,s+1)-Sp(:,s);

        if min(size(LS))<1
            LSnew =(deltap)  /norm(deltap);
            ALSnew=(deltaS - DS*deltap)/norm(deltap );
        else
```

```
            LSnew =(deltap-LS*LS’*deltap)/norm(deltap-LS*LS’*deltap);
            ALSnew=(deltaS-DS*deltap)    /norm(deltap-LS*LS’*deltap);
        end

        R1US = ALSnew*LSnew’;
        DS=DS+R1US;

        LS=[LS LSnew];
        LS=LS(:,max(1,size(LS,2)-N+2):size(LS,2));

        if max(max(isnan(LS)))==1
            fcf=inf;
            break
        end

    elseif R1U_type==’B’
        deltap=P(:,s+1)-P(:,s);
        deltaS=Sp(:,s+1)-Sp(:,s);

        R1US=(deltaS - DS*deltap)*deltap’/dot(deltap,deltap);
        DS=DS+R1US;

    elseif R1U_type==’C’
        deltap=P(:,s+1)-P(:,s);
        deltaS=Sp(:,s+1)-Sp(:,s);

        [aa kk]=max(abs(deltap));
        ej=zeros(N,1);
        ej(kk)=1;
        R1US=(deltaS - DS*deltap)*ej’/dot(ej,deltap);
        DS=DS+R1US;
    end

    global G
    G(:,s+1)=(eye(M,M)-DS*DF)\(Sp(:,s+1) +...
              DS*(Fg(:,s) - DF*G(:,s) - P(:,s+1)));

    if max(max(isnan(G)))==1
        fcf=inf;
        fcs=inf;
        break
    end
end
```

A.10 IQN-I solver: QN_solver_IQNI.m

```
[Sp(:,1) fval flowflag]=fsolve(Sfun_gen,[G(:,1)],options);
fcs=fcs+1;

global G G(:,1)=Sp(:,1);

[fg fval flowflag]=fsolve(Ffun_gen,[P(:,1);U(:,1)],options);
fcf=fcf+1;

Fg(:,1)=fg(1:N);
U(:,2) =fg(N+1:2*N);

Kp(:,1)=Fg(:,1)-P(:,1);
Residual(1)=norm(Kp(:,1));

switch recovery
    case{Basic}
        Ptemp=Fg(:,1);
        global P
        P(:,2)=omega*Ptemp+(1-omega)*P(:,1);
        clear Ptemp
    case{Addcolumns,PreviousJacR1U}
        if timelabel==1
            Ptemp=Fg(:,1);
            global P
            P(:,2)=omega*Ptemp+(1-omega)*P(:,1);
            clear Ptemp
        else
            global P
            P(:,2)=P(:,1)-DK\Kp(:,1);
        end
    case{MG1,MG2}
        global P
        if gridsize==’c’
            Ptemp=Fg(:,1);
            global P
            P(:,2)=omega*Ptemp+(1-omega)*P(:,1);
            clear Ptemp
        elseif gridsize==’f’
            P(:,2)=P(:,1)-DK\Kp(:,1);
        end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Main (outer) iteration loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for s=2:itmax
    [Sp(:,s) fval flowflag]=fsolve(Sfun_gen,[G(:,s-1)],options);
    fcs=fcs+1;

    global G
    G(:,s)=Sp(:,s);

    [fg fval flowflag]=fsolve(Ffun_gen,[P(:,s);U(:,s)],options);
    fcf=fcf+1;

    Fg(:,s)  =fg(1:N);
    U(:,s+1) =fg(N+1:2*N);

    Kp(:,s)=Fg(:,s)-P(:,s);
    Residual(s)=norm(Kp(:,s));

    if Residual(s)/Residual(1) < relrestol
        break
    end

    if R1U_type==’0’
        Kptemp=Kp(:,max(1,size(Kp,2)-N):size(Kp,2)-1);
        Fgtemp=Fg(:,max(1,size(Fg,2)-N):size(Fg,2)-1);

        V=(Kp(:,s)*ones(1,size(Kptemp,2)))-fliplr(Kptemp);
        W=(Fg(:,s)*ones(1,size(Fgtemp,2)))-fliplr(Fgtemp);

        VX=[];
        WX=[];

        if recovery==Addcolumns
            for kk=max(1,timelabel-nnreuse):timelabel-1
                VX=[VV(:,1:FCF(kk)-2,kk) VX];
                WX=[WW(:,1:FCF(kk)-2,kk) WX];
            end
        end

        VX=[V VX];
        WX=[W WX];
        [VX WX] =DoQR(VX,WX,QRthreshold);

        DM=WX*((VX’*VX)\VX’)-I;

    elseif R1U_type==’L’
        deltaK=Kp(:,s)-Kp(:,s-1);
        deltap=P(:,s)-P(:,s-1);

        if min(size(L))<1
            Lnew =(deltaK)  /norm(deltaK);
            ALnew=(deltap - DM*deltaK)/norm(deltaK );
        else
            Lnew =(deltaK-L*L’*deltaK)/norm(deltaK-L*L’*deltaK);
            ALnew=(deltap-DM*deltaK)  /norm(deltaK-L*L’*deltaK);
        end

        R1U = ALnew*Lnew’;
        DM=DM+R1U;

        L=[L Lnew];
        L=L(:,max(1,size(L,2)-N+2):size(L,2));

        if max(max(isnan(L)))==1
            fcf=inf;
            break
        end

    elseif R1U_type==’B’
        deltap=P(:,s)-P(:,s-1);
        deltaK=Kp(:,s)-Kp(:,s-1);

        R1U =((deltap - DM*deltaK)*deltaK’/dot(deltaK,deltaK));
        DM =DM+R1U;

    elseif R1U_type==’C’
        deltap=P(:,s)-P(:,s-1);
        deltaK=Kp(:,s)-Kp(:,s-1);
        [aa kk]=max(abs(deltaK));
        ej=zeros(N,1);
        ej(kk)=1;

        R1U =(deltap - DM*deltaK)*ej’/dot(ej,deltaK);
        DM =DM+R1U;
    end

    global P
    P(:,s+1)=P(:,s)-DM*Kp(:,s);

    if max(max(isnan(P)))==1
        fcf=inf;
        fcs=inf;
        break
    end
end
```

Bibliography

[1] J.P. Abbott, An efficient algorithm for the determination of certain bifurcation points. *J. Comput. Appl. Math.* **4**/1, pp. 19–27 (1978)

[2] A.C. Aitken, On Bernouilli's numerical solution of algebraic equations. Proc. Roy. Soc. Edinb. **46**, pp. 289–305 (1926)

[3] J.J. Alonso, A. Jameson, Fully-implicit time-marching aeroelastic solutions. Reno, NV, *AIAA Paper* **94–0056**, pp. 1–13 (1994)

[4] D. Anderson, Iterative procedures for nonlinear integral equations. *J. ACM* **12**, pp. 547–560 (1965)

[5] M. Arioli, C. Fassino, Roundoff error analysis of algorithms based on Krylov subspace methods. *BIT* **36**/2, pp. 189–206 (1996)

[6] L. Armijo, Minimization of functions having Lipschitz-continuous first partial derivatives. *Pacific J. Math.* **16**, pp. 1–3 (1966)

[7] W.E. Arnoldi, The principle of minimized iteration in the solution of the matrix eigenvalue problem. *Quart. Appl. Math.* **9**, pp. 17–29 (1951)

[8] O. Axelsson, Conjugate Gradient type methods for unsymmetric and inconsistent systems of linear equations. *Linear Algebra Appl.* **29**, pp. 1–16 (1980)

[9] O. Axelsson, Iterative solution methods. Cambridge University Press, New York, ISBN 0–521–44524–8 (1994)

[10] S. Axler, Linear algebra done right. 2nd edition, Springer, New York, ISBN 978–0–387–98258–8, (1996)

[11] R.E. Bank, T.F. Chan, PLTMGC: A multi-grid continuation program for parametrized nonlinear elliptic systems. *SIAM J. Sci. Statist. Comput.* **7**, pp. 540–559 (1986)

[12] R. Barret, M. Berry, T.F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, H.A. van der Vorst, Templates for the solution of linear systems: building blocks for iteration methods. SIAM, Philadeplhia, PA (1995)

[13] M. Bathe, R. Kamm, A fluid-structure interaction finite element analysis of pulsatile blood flow through a compliant stenotic artery. *Journal of Biomechanical Engineering* **121**, pp. 361–369 (1999)

[14] K.-J. Bathe, H. Zhang, Finite element developments for general fluid flows with structural interactions. *Internat. J. Numer. Methods Engrg.* **60**, pp. 213–232 (2004)

[15] F.L. Bauer, The quotient-difference algorithm and epsilon-algorithms. In: On numerical approximation. Univ. of Wisconsin Press, Madison, WI, pp. 361–370 (1959)

[16] F.L. Bauer, The g-algorithm. *J. Soc. Indust. Appl. Math.* **8**, pp. 1–17 (1960)

[17] Y. Bazilevs, V.M. Calo, Y. Zhang, T.J.R. Hughes, Isogeometric fluid-structure interaction analysis with applications to arterial blood flow. *Comput. Mech.* **38**/4–5, pp. 310–322 (2006)

[18] A. Ben-Israel, T.N.E. Greville, Generalized inverses: theory and applications. New York: Wiley (1977)

[19] N. Bićanić, K.H. Johnson, Who was '-Raphson'? *Internat. J. Numer. Methods Engrg.* **14**, pp. 148–152 (1979)

[20] C. Brezinski, Généralisation de la transformation de Shanks, de la table de Padé et de l'epsilon algorithm. *Calcolo* **12**, pp. 317–360 (1975)

[21] L. Broderick, J. Leonard, Nonlinear response of membranes to ocean waves using boundary and finite elements. *Ocean Engrg.* **22**, pp. 731–745 (1995)

[22] P.N. Brown, A local convergence theory for combined inexact-Newton/finite-difference projection methods. *SIAM J. Numer. Anal.* **24**/2, pp. 407–434 (1987)

[23] P.N. Brown, Y. Saad, Hybrid Krylov methods for nonlinear systems of equations. *SIAM J. Sci. Statist. Comput.* **11**/3, pp. 450-481 (1990)

[24] C.G. Broyden, A class of methods for solving nonlinear simultaneous equations. *Math. Comp.* **19**, pp. 577–593 (1965)

[25] C.G. Broyden, Quasi-Newton methods and their applications to function minimization. *Math. Comp.* **21**, pp. 368–381 (1967)

[26] C.G Broyden, The convergence of a class of double-rank minimization algorithms, Part 1 and 2. *J. Inst. Math. Applic.* **6**, pp. 76–90 and 222-231 (1970)

[27] C.G Broyden, The convergence of an algorithm for solving sparse non-linear systems. *Math. Comp.* **25**, pp. 285–294 (1971)

[28] C.G. Broyden, J.E. Dennis, J.J. Moré, On the local and superlinear convergence of quasi-Newton mehods. *J. Inst. Math. Appl.* **12**, pp. 223–246 (1973)

[29] A.M. Bruaset, A survey of preconditioned iterative methods. Pitman Research Notes in Mathematics Science **328**, Longman Scientific & Technical, Harlow, UK (1995)

[30] S. Cabay, L.W. Jackson, A polynomial extrapolation method for finding limits and antilimits of vector sequences. *SIAM J. Numer. Anal.* **13**, pp. 734–752 (1976)

[31] E. Catinas, Inexact perturbed Newton methods and applications to a class of Krylov solvers. *J. Optim. Theory Appl.* **108**/3, pp. 543–570 (2001)

[32] E. Cătinaş, The inexact, inexact perturbed and quasi-Newton methods are equivalent models. *Math. Comp.* **74**, pp. 291–301 (2005)

[33] E. Cătinaş, Methods of Newton and Newton-Krylov type. Risoprint, Cluj-Napoca, ISBN 978–973–751–533–9, (2007)

[34] P. Causin, J.-F. Gerbeau, F.Nobile, Added-mass effect in the design of partitioned algorithms for fluid-structure problems. *Tech. Rep.* **5084**, INRIA (2004)

[35] P. Causin, J.-F. Gerbeau, F. Nobile, Added-mass effect in the design of partitioned algorithms for fluid-structure problems. *Comput. Methods Appl. Mech. Engrg.* **194**/42-44, pp. 4506-4527 (2005)

[36] J.R. Cebral, R. Löhner, Fluid-structure coupling: extensions and improvements. *AIAA Paper* **97-0858** (1997)

[37] M. Cervera, R. Codina, M. Galindo, On the computational efficiency and implementation of block–iterative algorithms for nonlinear coupled problems. *Engrg. Comput.* **123**/6, pp. 4–30 (1996)

[38] P. Concus, G.H. Golub, A generalized Conjugate Gradient method for nonsymmetric systems of linear equations. *Technical Report* **STAN-CS-76-535**, Stanford University, Stanford, CA (1976)

[39] E.J. Craig, The N-step iteration procedures. *J. Math. Phys.* **34**, pp. 64–73 (1955)

[40] R.D. da Cunha, T.R. Hopkins, A comparison of acceleration techniques applied to the SOR method. In: A. Cuyt (Ed.), Proceedings on nonlinear numerical methods and rational approximation. pp. 247–260, Kluwer Academic Publishers Group , Dordrecht, The Netherlands (1994)

[41] W.C. Davidon, Variable metric methods for minimization. AEC Research and Development Report **ANL-5990** (Rev.) (1959)

[42] W.C. Davidon, Variance algorithms for minimization. *Comput. J.* **10**, pp. 406–410 (1968)

[43] H. Deconinck, K. Sermeus, R. Abgrall, Status of multidimensional upwind Residual Distribution Schemes and applications in aeronautics. *AIAA paper* **2000-2328** (2000)

[44] J. Degroote, P. Bruggeman, R. Haelterman, J. Vierendeels, Stability of a coupling technique for partitioned solvers in FSI applications. *Comput. & Structures* **86**, pp. 2224-2234 (2008)

[45] J. Degroote, K.-J. Bathe, J. Vierendeels, Performance of a new partitioned procedure versus a monolithic procedure in fluid–structure interaction. *Comput. & Structures*. In press; available online: doi:10.1016/j.compstruc.2008.11.013

[46] J. Degroote, P. Bruggeman, R. Haelterman, J. Vierendeels, Benchmark results of a rising bubble simulations with an interface tracking technique based on a partitioned fluid-structure interaction algorithm. *J. Comput. Appl. Math.*; accepted (2008)

[47] R.S. Dembo, S.C. Eisenstat, T. Steihaug, Inexact Newton methods. *SIAM J. Numer. Anal.* **19**, pp. 400–408 (1982)

[48] J.E. Dennis, J.J. Moré, A characterization of superlinear convergence and its application to quasi-Newton methods. *Math. Comp.*, **28**/126, pp. 549–560 (1974)

[49] J.E. Dennis, J.J. Moré, Quasi-Newton methods: motivation and theory. *SIAM Rev.* **19**, pp. 46–89 (1977)

[50] J.E. Dennis, R.B. Schnabel, Least Change Secant Updates for quasi-Newton methods. *SIAM Rev.* **21**, pp. 443–459 (1979)

[51] J.E. Dennis, H.F. Walker, Convergence theorems for Least-Change Secant Update methods. *SIAM J. Numer. Anal.* **18**, pp. 940–987 (1981)

[52] J.E. Dennis, R.B. Schnabel, Numerical methods for unconstrained optimization and nonlinear equations. Prentice-Hall, Englewood Cliffs, NJ (1983)

[53] S. Deparis, Numerical analysis of axisymmetric flows and methods for Fluid-Structure Interaction arising in blood flow simulation. PhD thesis, Ecole Polytechnique Fédrale de Lausanne (2004)

[54] S. Deparis, M. Discacciati, G. Fourestey, A. Quarteroni, Fluid-structure algorithms based on Steklov-Poincaré operators. *Comput. Methods Appl. Mech. Engrg.* **195**/41–43, pp. 5797–5812 (2006)

[55] P. Deuflhard, R. Freund, A. Walter, Fast secant methods for the iterative solution of large nonsymmetric linear systems. *IMPACT Comp. Sci. Eng.* **2**, pp. 244–276 (1990)

[56] P. Deuflhard, Global inexact Newton methods for very large scale nonlinear problems. *IMPACT Comp. Sci. Eng.* **3**, pp. 366–393 (1991)

[57] P. Deuflhard, Newton methods for nonlinear problems – affine invariance and adaptive algorithms, 2nd Ed. Springer, ISBN 978–3–540–21099–3 (2006)

[58] L.C.W. Dixon, Automatic differentiation and parallel processing in optimisation. *TR* **No.180**, The Hatfield Polytechnique, Hatfield, UK (1987)

[59] N. dos Santos, J.-F. Gerbeau, J.-F. Bourgat, Partitioned FSI strategy for simulations of a thin elastic valve. In: P. Wesseling, E. Oate, J. Priaux (Eds.), European Conference on Computational Fluid Dynamics ECCOMAS CFD 2006. ECCOMAS, Delft, The Netherlands, pp. 1–10 (2006)

[60] J. Drkošová, A. Greenbaum, M. Rozložník, Z. Strakoš, Numerical stability of GMRes. *BIT* **35**, pp. 309–330 (1995)

[61] K. Dumont, J. Vierendeels, R. Kaminsky, G. Van Nooten, P. Verdonck, D. Bluestein, Comparison of the hemodynamic and thrombogenic performance of two bileaflet mechanical heart valves using a CFD/FSI model. *ASME J. of Biomechanical Engineering* **129**/4, pp. 558–565 (2007)

[62] R.P. Eddy, Extrapolating to the limit of a vector sequence. In: P.C.C. Wang (Ed.), Information linkage between applied mathematics and industry. Academic Press, New York, pp. 387–396 (1979)

[63] T. Eirola, O. Nevanlinna, Accelerating with rank-one updates. *Linear Algebra Appl.* **121**, pp. 511–520 (1989)

[64] W. Elleithy, M. Tanaka, Interface relaxation algorithms for BEM-BEM coupling and FEM-BEM coupling. *Comput. Methods Appl. Mech. Engrg.* **192**, pp. 2977–2992 (2003)

[65] H.C. Elman, Iterative methods for large sparse nonsymmetric systems of linear equations. PhD Thesis, Computer Science Dept., Yale Univ., New Haven, CT (1982)

[66] C. Farhat, M. Lesoinne, P. LeTallec, Load and motion transfer algorithms for Fluid/Structure Interaction problems with non-matching discrete interfaces: Momentum and energy conservation, optimal discretization and application to aeroelasticity. *Comput. Methods Appl. Mech. Engrg.* **157**, pp. 95–114 (1998)

[67] C. Farhat, M. Lesoinne, Two efficient staggered algorithms for serial and parallel solution of three-dimensional nonlinear transient aeroelastic problems. *Comput. Methods Appl. Mech. Engrg.* **182**, pp. 499–515 (2000)

[68] C. Farhat, P. Geuzaine, C. Grandmont, The discrete geometric conservation law and the nonlinear stability of ALE schemes for the solution of flow problems on moving grids. *J. Comput. Phys.* **174**, pp. 669–694 (2001)

[69] C. Farhat, P. Geuzaine, G. Brown, Application of a three–field nonlinear fluid-structure formulation to the prediction of the aeroelastic parameters of an F-16 fighter. *Comput. & Fluids* **32**, pp. 3–29 (2003)

[70] C. Farhat, CFD-based nonlinear computational aeroelasticity. In: E. Stein, R. De Borst, T.J.R. Hughes (Eds.) Encyclopedia of computational mechanics, Vol. **3**, Chapter 13. Wiley, NY (2004)

[71] C. Farhat, K. van der Zee, P. Geuzaine, Provably second-order time-accurate loosely-coupled solution algorithms for transient nonlinear computational aeroelasticity. *Comput. Methods Appl. Mech. Engrg.* **195**, pp. 1973–2001 (2006)

[72] L.V. Fausett, Numerical methods, algorithms and applications. Prentice Hall, ISBN 0-13-031400-5 (2003)

[73] C.A. Felippa, K.C. Park, C. Farhat, Partitioned analysis of coupled mechanical systems. *Comput. Methods Appl. Mech. Engrg.* **190**, pp. 3247–3270 (2001)

[74] M.A. Fernández, M. Moubachir, A Newton method using exact jacobians for solving fluid-structure coupling. *Comput. & Structures* **83**, pp. 127–142 (2005)

[75] R. Fletcher, M.J.D. Powell, A rapidly convergent descent method for minimization. *Comput. J.* **6**, pp. 163–168 (1963)

[76] R. Fletcher, A new approach to variable metric algorithms. *Comput. J.* **13**, pp. 317–322 (1970)

[77] R. Fletcher, Conjugate Gradient methods for indefinite systems. In: G. Watson (Ed.) Numerical analysis Dundee 1975. Springer-Verlag, Berlin, New York, pp. 73–89 (1976)

[78] R. Fletcher, Practical methods of optimization, Vol. **1**: Unconstrained optimization. Wiley, New York (1980)

[79] G. Fourtesy, S. Piperno, A second-order time-accurate ALE Lagrange-Galerkin method applied to wind engineering and control of bridge profiles. *Comput. Methods Appl. Mech. Engrg.* **193**, pp. 4117–4137 (2004)

[80] F. Freudenstein, B. Roth, Numerical solutions of systems of nonlinear equations. *J. ACM* **10**, pp. 550–556 (1963)

[81] R.W. Freund, G.H. Golub, N.M. Nachtigal, Iterative solution of linear systems. *Acta Numerica* **1**, pp. 57–100 (1991)

[82] R.W. Freund, N.M. Nachtigal, QMR: a quasi-minimal residual method for non-Hermitian linear systems. *Numer. Math.* **60**, pp. 315–339 (1991)

[83] R.W. Freund, A transpose-free Quasi-Minimal Residual algorithm for non-Hermitian linear systems. *SIAM J. Sci. Comput.* **14**, pp. 470–482 (1993)

[84] A. Friedlander, M.A. Gomes-Ruggiero, D.N. Kozakevich, J.M. Martinez, S.A. dos Santos, Solving nonlinear systems of equations by means of quasi-Newton methods with a nonmonotone strategy. *Optim. Methods Softw.* **8**, pp. 25–51 (1997)

[85] W. Gai, J. Xue, Y. Qu, A new implementation of the EN method. *Appl. Math. Comput.* **98**, pp. 199–208 (1999)

[86] D.M. Gay, Some convergence properties of Broyden's method. *SIAM J. Numer. Anal.* **16**, pp. 623–630 (1979)

[87] M. El-Gebeily, W.M. Elleithy, H.J. Al-Gahtani, Convergence of the domain decomposition finite element-boundary element coupling methods. *Comput. Methods Appl. Mech. Engrg.* **191**, pp. 4851–4867 (2002)

[88] J.-F. Gerbeau, M. Vidrascu, A quasi-Newton algorithm based on a reduced model for fluid structure problems in blood flow. *Mathematical Modelling and Numerical Analysis* (M2AN) **37**, pp. 631–647 (2003)

[89] J.-F. Gerbeau, M. Vidrascu, P. Frey, Fluid-structure interaction in blood flows on geometries coming from medical imaging. *Comput. & Structures* **83**, pp. 155–165 (2005)

[90] D. Goldfarb, A family of varible metric methods. *Math. Comp.* **24**, pp. 23–26 (1970)

[91] A.A. Goldstein, Constructive Real Analysis. Harper and Row, New York (1967)

[92] G.H. Golub, C. Loan, Matrix Computations. Johns Hopkins Univ. Press, Baltimore, MD (1989)

[93] G.H. Golub, D.P. O'Leary, Some history of the conjugate gradient and Lanczos algorithms. *SIAM Rev.* **31**, pp. 50–102 (1989)

[94] M.A. Gomez-Ruggiero, J.M. Martinez, The Column-Updating Method for solving nonlinear equations in Hilbert space. *RAIRO Mathematical Modelling and Numerical Analysis* **26**, pp. 309–330 (1992)

[95] P.R. Graves–Morris, A review of Padé methods for the acceleration of convergence of a sequence of vectors. *Report* **No NA92-31**, Dept. of Mathematics, Univ. of Bradford (1992)

[96] A. Greenbaum, V. Pták, Z. Strakoš, Any nonincreasing convergence curve is possible for GMRes. *SIAM J. Matrix Anal. Appl.* **17**/3, pp. 465–469 (1996)

[97] A. Greenbaum, M. Rozložník, Z. Strakoš, Numerical behavior of the Modified Gram-Schmidt GMRes implementation. *BIT* **37**/3, pp. 706–719 (1997)

[98] A. Greenbaum, Iterative methods for solving linear systems. *Frontiers in Applied Mathematics*, SIAM, Philadelphia, PA, ISBN 0-89871-396-X (1997)

[99] J.L. Greenstadt, Variations on variable-metric methods. *Math. Comp.* **24**, pp. 1–22 (1970)

[100] W. Hackbush, Iterative solution of large sparse systems. New York, Springer Verlag, ISBN 0–387–94064–2 (1994)

[101] W. Hackbush, Iterative Lösung großer schwachbesetzer Gleichungssysteme. Teubner, Stuttgart (1991)

[102] R. Haelterman, Aggregation Multi-Grid for Residual Distribution Schemes. *Project Report* **2005-14**, Von Karman Institute for Fluid Dynamics (2005)

[103] R. Haelterman, J. Degroote, J. Vierendeels, D. Van Heule, Extending Broyden's method to interaction problems. *Rev. Anal. Numer. Theor. Approx.* **37/2**, pp. 169–180 (2008).

[104] R. Haelterman, J. Degroote, D. Van Heule, J. Vierendeels, Comparing Quasi-Newton Methods for Interaction Problems. *J. Comput. Appl. Math.*; submitted October 2008

[105] R. Haelterman, J. Degroote, D. Van Heule, J. Vierendeels, Comparison and analysis of four variants of the Quasi-Newton Least Squares method for fluid-structure interaction problems. *J. Comput. Appl. Math.*; submitted December 2008

[106] R. Haelterman, J. Degroote, D. Van Heule, J. Vierendeels, On the Similarities between the Quasi-Newton Least Squares Method and GMRes. *SIAM J. Sci. Comp.*; submitted December 2008

[107] R. Haelterman, J. Degroote, D. Van Heule, J. Vierendeels, The quasi-Newton Least Squares method: a new and fast secant method analyzed for linear systems. *SIAM J. Numer. Anal.*; accepted March 2009

[108] R. Haelterman, J. Degroote, D. Van Heule, J. Vierendeels, On the Similarities between the Quasi-Newton Inverse Least Squares Method and GMRes. *SIAM J. Numer. Anal.*; submitted February 2009

[109] R. Hamming, Numerical methods for scientists and engineers, 2nd Ed. McGraw–Hill, New York (1973)

[110] M. Heil, Stokes flow in an elastic tube – a large–displacement fluid–structure interaction problem. *Internat. J. Numer. Methods Fluids* **28**, pp. 243–265 (1998)

[111] M. Heil, An efficient solver for the fully coupled solution of large-displacement fluid–structure interaction problems. *Comput. Methods Appl. Mech. Engrg.* **193**, pp. 1–23 (2004)

[112] M.R. Hestenes, E. Stiefel, Methods of Conjugate Gradients for solving linear systems. *J. Res. Nat. Bur. Stand.* **49**, pp. 409–436 (1952)

[113] A.S. Householder, The theory of matrices in numerical analysis. Dover Publications, Mineola, NY (1964)

[114] J. Hron, S. Turek, A monolithic FEM/multigrid solver for ALE formulation of fluid structure interaction with application in biomechanics. In: H.–J. Bungartz, M. Schafer (Eds.), Fluid-Structure Interaction – modelling, simulation, optimisation. No. **53** in Lecture Notes in Computational Science and Engineering, Springer, Berlin, pp. 146–170, ISBN 3–540–34595–7 (2006)

[115] B. Hübner, E. Walhorn, D. Dinkler, Numerical investigation to bridge aeroelasticity. In: H.A. Mang, F.G. Rammerstofer, J. Eberhardsteiner (Eds.), Proceedings of the Fifth World Congress on Computational Mechanics (WCCM V). Vienna University of Technology, Austria (2002)
Available from <http://wccm.tuwien.ac.at/publications/Papers/fp81407.pdf>

[116] B. Hübner, E. Walhorn, D. Dinkler, A monolithic approach to fluid–structure interaction using space–time finite elements. *Comput. Methods Appl. Mech. Engrg.* **193**, pp. 2087–2104 (2004)

[117] I.C.F. Ipsen, C.D. Meyer, The idea behind Krylov methods. *Amer. Math. Monthly* **105**/10, pp. 889–899 (1988)

[118] B. Irons, R.C. Tuck, A version of the Aitken accelerator for computer implementation. *Internat. J. Numer. Methods Engrg.* **2**, pp. 275–277 (1969)

[119] H.Z. Jahromi, B.A. Izzudin, L. Zdravkovic, Coupling methods for modelling nonlinear soil-structure interaction. The 15th UK Conference of the Association of Computational Mechanics in Engineering, In: B.H.V. Toppin (Ed.), Paper No **78**, Stirlingshire, Scotland, Civil-Comp Press (2007)

[120] H.Z. Jahromi, B.A. Izzuddin, L. Zdravkovic, Partitioned analysis of nonlinear soil-structure interaction using iterative coupling. *Interaction and Multiscale Mechanics* **1**/1 pp. 33–51 (2007)

[121] K. Jbilou, H. Sadok, Vector extrapolation methods: applications and numerical comparison. *J. Comput. Appl. Math.* **122**, pp. 149–165 (2000)

[122] K.C. Jea, D.M. Young, Generalized conjugate gradient acceleration of nonsymmetrizable iterative methods. *Linear Algebra Appl.* **34**, pp. 159–194 (1980)

[123] V. Kalro, T.E. Tezduyar, A parallel 3D computational method for fluid-structure interactions in parachute systems. *Comput. Methods Appl. Mech. Engrg.* **190**, pp. 321–332 (2000)

[124] E. M. Kasenally, GMBACK: a generalised minimum backward error algorithm for nonsymmetric linear systems. *SIAM J. Sci. Comp.* **16**, pp. 698-719 (1995)

[125] E. M. Kasenally, V. Simoncini, Analysis of a minimum perturbation algorithm for nonsymmetric linear systems. *SIAM J. Numer. Anal.* **34**, pp. 48-66 (1997)

[126] G. Kedem, Automatic differentiation of computer programs. *ACM Trans. Math. Software* **6**, pp. 150–165 (1980)

[127] C.T. Kelley, Iterative methods for linear and nonlinear equations. *Frontiers Appl. Math.*, SIAM, Philadelphia (1995)

[128] I.M. Khabaza, An iterative least-squares method for solving large sparse matrices. *Comput. J.* **6**, pp. 202–206 (1963–1964)

[129] U. Küttler, W.A. Wall, Fixed-point fluid-structure interaction solvers with dynamic relaxation. *Comput. Mech.* **43**/1, pp. 61–72 (2008)

[130] S. Kim, R.P. Tewarson, A quasi-Newton method for solving non-linear algebraic equations. *Comput. Math. Appl.* **24**, pp. 93–97 (1992)

[131] A.N. Krylov, On the numerical solution of equations which in technical questions are determined by the frequency of small vibrations of material systems. *Izvestija AN SSSR* (News of Academy of Sciences of the USSR), *Otdel. mat. i estest. nauk* **VII**/4, pp. 491-539 (in Russian) (1931)

[132] C.-H. Lai, Comparing quasi-Newton method for solving sparse interface problem. *CWI Technical Report* **NM-R9303**, CWI, Amsterdam, the Netherlands (1993)

[133] B. Lam, On the convergence of a quasi-Newton method for sparse nonlinear systems. *Math. Comp.* **32**, pp. 447–451 (1978)

[134] C. Lanczos, Solution of systems of linear equations by minimized iterations. *J. Res. Natl. Bur. Stand.* **49**, pp. 33–53 (1952)

[135] P. Le Tallec, J. Mouro, Fluid structure interaction with large structural displacements. *Comput. Methods Appl. Mech. Engrg.* **10**, pp. 3039–3067 (2001)

[136] V.L.R. Lopes, J.M. Martinez, Convergence properties of the Inverse Column-Updating Method. *Optim. Methods Softw.* **6**, pp. 127–144 (1995)

[137] S. Lubkin, A method of summing infinite series. *J. Res. Nat. Bur. Stand.* **48**/3, pp. 228–254 (1952)

[138] A.J. MacLeod, Acceleration of vector sequences by multi-dimensional δ^2 methods. *Comm. in Appl. Num. Meth.* **2**, pp. 385–392 (1986)

[139] J.M. Martinez, A quasi-Newton method with modification of one column per iteration. *Computing* **33**, pp. 353–362 (1984)

[140] J.M. Martinez, Local convergence theory of inexact Newton methods based on structured least change updates. *Math. Comp.* **55**, pp. 143–168 (1990)

[141] J.M. Martinez, Fixed-point Quasi-Newton methods. *SIAM J. Numer. Anal.*, **29**, pp. 1413–1434 (1992)

[142] J.M. Martinez, On the relation between two local convergence theories of least change secant update methods. *Math. Comp.* **59**, pp. 457–481 (1992)

[143] J.M. Martinez, M.C. Zambaldi, An Inverse Column-Updating Method for solving large-scale nonlinear systems of equations. *Optim. Methods Softw.* **1**, pp. 129–140 (1992)

[144] J.M. Martinez, On the convergence of the column-updating method. *J. Comp. Appl. Math.* **12**/2, pp. 83–94 (1993)

[145] J.M. Martinez, Practical quasi-Newton method for solving nonlinear systems. *J. Comput. Appl. Math.* **124**, pp. 97–122 (2000)

[146] E. Marwil, Exploiting sparsity in Nweton-type methods. PhD Thesis, Cornell Apllied Math. (1978)

[147] E. Marwil, Convergence results for Schubert's method for solving sparse nonlinear equations. *SIAM J. Numer. Anal.* **16**, pp. 588–604 (1979)

[148] Matrix Market Repository. http://math.nist.gov/MatrixMarket/

[149] H.G. Matthies, J. Steindorf, Numerical efficiency of different partitioned methods for fluid–structure interaction. *Z. Angew. Math. Mech.* **80**/8, pp. 557–558 (2000)

[150] H.G. Matthies, J. Steindorf, Partitioned strong coupling algorithms for fluid-structure interaction. *Comput. & Structures* **81**, pp. 805–812 (2003)

[151] H. Matthies, R. Niekamp, J. Steindorf, Algorithms for strong coupling procedures. *Comput. Methods Appl. Mech. Engrg.* **195**, pp. 2028–2049 (2006)

[152] G.P. McCormick, J.D. Pearson, Variable metric methods and unconstrained optimization. In: R. Fletcher (Ed.), Optimization. Academic Press, London and New York (1969)

[153] R.B. Melville, S.A. Morton, Fully-implicit aeroelasticity on overset grid systems. *AIAA Paper* **98-0251** (1998)

[154] M. Mešina, Convergence acceleration for the iterative solution of the equations $X = AX + f$. *Comput. Methods Appl. Mech. Engrg.* **10**, pp. 165–173 (1977)

[155] C. Michler, E.H. Van Brummelen, R. De Borst, An interface Newton-Krylov solver for Fluid-Structure Interaction. *Internat. J. Numer. Methods Fluids* **47**/10–11, pp. 1189–1195 (2005)

[156] C. Michler, E.H. Van Brummelen, R. De Borst, Error-amplification analysis of subiteration-preconditioned GMRES for fluid-structure interaction. *Comput. Methods Appl. Mech. Engrg.* **195**, pp. 2124–2148 (2006)

[157] D.P. Mok, W.A. Wall, E. Ramm, Accelerated iterative substructuring schemes for instationary fluid-structure interaction. In: Bathe (Ed.), Computational fluid and solid mechanics, pp. 1325–1328. Elsevier (2001)

[158] D.P. Mok, W.A. Wall, Partitioned analysis schemes for the transient interaction of incompressible flows and nonlinear flexible structures. In: W.A. Wall, K.-U. Bletzinger, K. Schwezerhof (Eds.), Trends in computational structural mechanics. pp. 689–698 (2001)

[159] J.J. Moré, J.A. Trangenstein, On the global convergence of Broyden's method. *Math. Comput.* **30**/135, pp. 523–540 (1976)

[160] J.J. Moré, S.G. Garbow, K.E. Hillstrom, Testing unconstrained optimization software. *ACM Trans. Math. Software* **7**, pp. 17–41 (1981)

[161] S.A. Morton, R.B. Melville, M.R. Visbal, Accuracy and coupling issues of aeroelastic Navier-Stokes solutions on deforming meshes. *AIAA-paper* **97-1085**, pp. 252–262 (1997)

[162] B.A. Murtagh, R.W.H. Sargent, A constrained minimization method with quadratic convergence. In: R. Fletcher (Ed.), Optimization. Academic Press, London and New York (1970)

[163] N.M. Nachtigal, S.C. Reddy, L.N. Trefethen, How fast are nonsymmetric matrix iterations ? *SIAM J. Matrix Anal. Appl.* **13**, pp. 778–795 (1992)

[164] I. Newton, The mathematical papers of Isaac Newton (7 volumes). D.T. Whiteside (Ed.), Cambridge University Press, Cambridge (1967–1976)

[165] F. Nobile, Numerical approximation of fluid-structure interaction problems with application to haemodynamics. PhD thesis, Ecole Polytechnique Fédérale de Lausanne (2001)

[166] J.M. Ortega, W.C. Rheinboldt, Iterative solution of nonlinear equations in several variables. Classics in Applied Mathematics **30**, Society for Industrial and Applied Mathematics, Philadelphia, PA (2000)

[167] C.C. Paige, M.A. Saunders, Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.* **12**, pp. 617–624 (1975)

[168] C.C. Paige, M.A. Saunders, LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Softw.* **8**, pp. 43–71 (1982)

[169] K.C. Park, An improved stiffly stable method for direct integration of nonlinear structural dynamics equations. *J. Appl. Mech.* **42**, pp. 464–470 (1975)

[170] K.C. Park, C.A. Felippa, J.A. Deruntz, Stabilization of staggered solution procedures for fluid-structure interaction analysis. Proceedings of the Winter Annual Meeting, Atlanta, Ga, United States, pp. 95–124 (1977)

[171] K.C. Park, C.A. Felippa, Partitioned analysis of coupled systems. In: Belytschko T., Hughes T.J.R. (Eds), Computational Methods for Transient Analysis, North-Holland, Amsterdam, pp. 157–219 (1983)

[172] J.D. Pearson, Variable metric methos of minimization, *Comput. J.* **12**, pp. 171–178 (1969)

[173] R. Penrose, A generalized inverse for matrices. *Math. Proc. Cambridge Philos. Soc.* **51**, pp. 406–413 (1955)

[174] S. Piperno, C. Farhat, B. Larrouturou, Partitioned procedures for the transient solution of coupled aeroelastic problems - part I: model problem, theory and two-dimensional application. *Comput. Methods Appl. Mech. Engrg.* **124**, pp. 79–112 (1995)

[175] S. Piperno, Explicit/implicit fluid-structure staggered procedures with a structural predictor and fluid-subcycling for 2D inviscid aeroelastic simulations. *Internat. J. Numer. Methods Fluids* **25**, pp. 1207–1226 (1997)

[176] S. Piperno, Numerical simulation of aeroelastic instabilities of elementary bridge decks. *INRIA Tech. Rep.* **3549** (1998)

[177] S. Piperno, C. Farhat, Partitioned procedures for the transient solution of coupled aeroelastic problems - part II: energy transfer analysis and three-dimensional applications. *Comput. Methods Appl. Mech. Engrg.* **190**, pp. 3147–3170 (2001)

[178] M.J.D. Powell, An iterative method for finding stationary values of a function of several variables. *Comput. J.* **5**, pp. 147–151 (1962)

[179] M.J.D. Powell, An efficient method for finding the minimum of a function of several variables without calculating derivatives. *Comput. J.* **7**, pp. 155–162 (1964)

[180] M.J.D. Powell, A hybrid method for nonlinear equations. In: P. Rabinowitz (Ed.), Numerical methods for non-linear algebraic equations, Gordon and Breach, London (1970)

[181] M.J.D. Powell, A new algorithm for unconstrained optimization. *Report* **TP-393**, AERE, Harwell, England (1970)

[182] B.P. Pugachev, Acceleration of convergence of iterative processes and a method for solving systems of nonlinear equations. *USSR Comput. Math. Math. Phys.* **17** pp. 199–207 (1978)

[183] A. Quarteroni, A. Valli, Domain decomposition methods for partial differential equations. Oxford Science Publications, Oxford (1999)

[184] J. Raphson, Analysis aequationum universalis seu ad aequationes algebraicas resolvendas methodus generalis, et expedita, ex nova infinitarum serierum doctrina, deducta ac demonstrata. Original in British Library, London (1690)

[185] K. Riemslagh, J. Vierendeels, E. Dick, Coupling of a Navier-Stokes solver and an elastic boundary solver for unsteady problems. In: K. Papailiou, D. Tsahalis, J. Priaux, C. Hirsch, M. Pandolfi (Eds.), 4th European Computational Fluid Dynamics Conference. ECCOMAS, Athens, Greece, pp. 1040–1045 (1998)

[186] S.M. Rifai et al., Multiphysics simulation of flow-induced vibrations and aeroelasticity on parallel computing platforms. *Comput. Methods Appl. Mech. Engrg.* **174**, pp. 393–417 (1999)

[187] I. Robertson, S.J. Sherwin, P.W. Bearman, Prediction of flutter instability due to cross winds in the Second Forth Road Bridge. In: Computational Fluid Dynamics '01, Proceedings of the Eccomas CFD Conference 2001. Swansea, Wales, UK (2001)

[188] I. Robertson, S.J. Sherwin, P.W. Bearman, Flutter instability prediction techniques for bridge deck sections. *Internat. J. Numer. Methods Fluids* **43**, pp. 1239–1256 (2003)

[189] H.H. Rosenbrock, An automatic method for finding the greatest or least value of a function. *Comput. J.* **3**, pp. 175–184 (1960)

[190] Y. Saad, Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices. *Linear Algebra Appl.* **29**, pp. 323–346 (1980)

[191] Y. Saad, Krylov subspace methods for solving large unsymmetric linear systems. *Math. Comp.* **37**, pp. 105–126 (1981)

[192] Y. Saad, M.H. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.* **7**, pp. 856–869 (1986)

[193] Y. Saad, Iterative methods for sparse linear systems (2nd Ed). Society for Industrial and Applied Mathematics (SIAM), Philadelphia, ISBN 0–89871–534–2 (2003)

[194] H. Sadok, Méthodes de projections pour les systèmes linéaires et nonlinéaires. Habilitation Thesis, Université de Lille, France (1994)

[195] H. Schneider, G.P. Barker, Matrices and linear spaces, 2nd Edition. Dover Publications, New York (1989)

[196] L.K. Schubert, Modification of a quasi-Newton method for nonlinear equations with sparse Jacobian. *Math.Comp.* **24**, pp. 27–30 (1970)

[197] L.K. Schubert, An interval arithmetic aproach for the construction of an almost globally convergence method for the solution of the nonlinear Poisson equation on the unit square. *SIAM J. Sci. Statist. Comput.* **5**, pp. 427–452 (1984)

[198] D.F. Shanno, Conditioning of quasi-Newton methods for function minimization. *Math. Comp.* **24**, pp. 647–656 (1970)

[199] D.F. Shanno, On the variable metric methods for sparse Hessians. *Math. Comp.* **34**, pp. 499–514 (1980)

[200] A.H. Sherman, On Newton-iterative Methods for the solution of systems of nonlinear equations. *SIAM J. Numer. Anal.* **14**, pp. 755–774 (1978)

[201] A.H. Sherman, W.J. Morrison, Adjustment of an inverse matrix corersponding to changes in the elements of a given column or a given row of the original matrix. *Ann. Math. Statist.* **21**, pp. 124–127 (1950)

[202] Z.-J. Shi, Convergence of quasi-Newton method with new inexact line search. *J.Math.Anal.Appl.* **315**, pp. 120–131 (2006)

[203] A. Sidi, Extrapolation vs. projection methods for linear systems of equations. *J. Comput. Appl. Math.* **22**/1 pp. 71–88 (1988)

[204] A. Sidi, W.F. Ford, D.A. Smith, Acceleration of convergence of vector sequences. *SIAM J. Numer. Anal.* **23**/1, pp. 178–196 (1986)

[205] A. Sidi, Efficient implementation of minimal polynomial and reduced rank extrapolation methods. *J. Comput. Appl. Math.* **36**, pp. 305–337 (1991)

[206] A. Sidi, Practical extrapolation methods: theory and applications. Cambridge Monographs on Applied and Computational Mathematics **10**, Cambridge University Press, Cambridge, ISBN 0–521–66159–5 (2003)

[207] D.A. Smith, W.F. Ford, A. Sidi, Extrapolation methods for vector sequences. *SIAM Rev.* **29** pp. 199–233 (1987). Correction, *SIAM Rev.* **30** pp. 623–624 (1988)

[208] P. Sonneveld, CGS: a fast Lanczos type solver for nonsymmetric linear systems. *SIAM J. Sci. Stat. Comp.* **10**, pp. 36–52 (1989)

[209] T. Steihaug, Quasi-Newton methods for large scale nonlinear problems. Ph.D. Thesis, School of organization and management, Yale University, New Haven, CT. Also available as Series **49** (1981)

[210] E. Spedicato, Computational experience with quasi-Newton algorithms for minimization problems of moderately large size. *Report CISE–N–***175**, Segrate (Milano) (1975)

[211] K. Stein, T. Tezduyar, V. Kumar, S. Sathe, R. Benney, R. Charles, Numerical simulation of soft landing for clusters of cargo parachutes. In: P. Niettaanmäki, T. Rossi, K. Majava, O. Pieronneau (Eds.), European congres on computational methods in applied sciences and engineering ECCOMAS 2004. Jyväskylä, pp. 1–14 (2004)

[212] G. Strang, Linear algebra and its applications. 3rd ed. Orlando: Saunders (1988)

[213] T.E. Tezduyar, S. Sathe, K. Stein, Solution techniques for the fully-discretized equations in computation of fluid-structure interactions with the space-time formulations. *Comput. Methods Appl. Mech. Engrg.* **195**, pp. 5743–5753 (2006)

[214] Ph. L. Toint, On sparse and symmetric matrix updating subject to a linear equation. *Math. Comp.* **31**, pp. 954–961 (1977)

[215] Ph. L. Toint, Some numerical results using a sparse matrix updating formula in unconstrained optimizations. *Math. Comp.* **32**, pp. 839–851 (1978)

[216] Ph. L. Toint, A sparse quasi-Newton update derived variationally with a non-diagonally weighted Frobenius norm. *Math. Comp.* **37**, pp. 425–434 (1981)

[217] Ph. L. Toint, Numerical solution of large sets of algebraic non-linear equations. *Math. Comp.* **16**, pp. 175–189 (1986)

[218] E.H. Van Brummelen, C. Michler, R. De Borst, Interface-GMRES(R) acceleration of subiteration for fluid-structure-interaction problems. *Report* **DACS–05–001**, Delft Aerospace Computational Science (2005)

[219] E.H. Van Brummelen, R. De Borst, On the nonnormality of subiteration for a fluid-structure-interaction problem. *SIAM J. Sci. Comput.* **27**/2, pp. 599–621 (2005)

[220] H.A. van der Vorst, BI-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of non-symmetric linear systems. *SIAM J. Sci. Stat. Comput.* **13**, pp. 631–644 (1992)

[221] H.A. van der Vorst, C. Vuik, GMRESR: a family of nested GMRes methods. *Numer. Linear Algebra Appl.* **1**, pp. 369–386 (1994)

[222] H.A. van der Vorst, Iterative Krylov methods for large linear systems. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, New York, ISBN 0–521–81828–1 (2003)

[223] J. Van hamme, private communication (2008)

[224] R. van Loon, P. Anderson, F. van de Vosse, A fluid-structure interaction method with solid-rigid contact for heart valve dynamics. *J. Comput. Phys.* **217**, pp. 806–823 (2006)

[225] A. van Zuijlen, S. Bosscher, H. Bijl, Two level algorithms for partitioned fluid-structure interaction computations. *Comput. Methods Appl. Mech. Engrg.* **196**, pp. 1458–1470 (2007)

[226] R.S. Varga, Matrix iterative analysis. Prentice-hall, Englewood Cliffs, NJ (1962)

[227] J. Vierendeels, L. Lanoye, J. Degroote, P. Verdonck, Implicit coupling of partitioned fluid-structure interaction problems with reduced order models. *Comput. & Structures* **85**, pp. 970–976 (2007)

[228] J. Vierendeels, Implicit coupling of partitioned fluid-structure interaction solvers using reduced-order models. In: Bungartz, Schäfer, (Eds.), Fluid-Structure Interaction: modelling, simulation, optimisation. Series: Lecture Notes in Computational Science and Engineering; **53**, pp. 1–18. Springer, ISBN 3–540–34595–7 (2006)

[229] P.K.W. Vinsome, ORTHOMIN: an iterative method for solving sparse sets of simultaneous linear equations. In: Proc. Fourth Symposium on Reservoir Simulations, pp. 149–159 (1976)

[230] V. Voïévodine, Algèbre linéaire, Edition Mir, Moscou (1976)

[231] C. Vuik, H.A. van der Vorst, A comparison of some GMRes-like methods. *Linear Algebra Appl.* **160**, pp. 131–162 (1992)

[232] H.F. Walker, Implementations of the GMRes method. *Computer Physics Communications* **53**, North Holland, Amsterdam, pp. 311–320 (1989)

[233] H.F. Walker, L. Zhou, A simpler GMRes. *Num. Lin. Alg. Appl.* **1**, pp. 571–581 (1994)

[234] H.F. Walker, Implementation of the GMRes Method using Householder transformations. *SIAM J. Sci. Comput.* **18**, pp. 516–536 (1997)

[235] W.A. Wall, D.P. Mok, E. Ramm, Partitioned analysis approach of the transient coupled response of viscous fluids and flexible structures. In: W. Wunderlich (Ed.), Solids, structures and coupled problems in engineering. Proceedings of the European conference on computational mechanics, ECCM '99, Munich (1999)

[236] W.A. Wall, Fluid-Struktur-Interaktion mit stabilisierten Finiten Elementen. Ph.D.-Dissertation, Report **31**, Institute of Structural Mechanics, University of Stuttgart (1999)

[237] O. Widlund, A Lanczos mzethod for a class of nonsymmetric systems of linear equations. *SIAM J. Numer. Anal.* **15**, pp. 801–812 (1978)

[238] J.H. Wilkinson, The algebraic eigenvalue problem. Clarendon Press, Oxford, England (1965)

[239] K. Willcox, J. Paduano, J. Peraire, Low order aerodynamic models for aeroelastic control of turbomachines. In: 40th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials (SDM) Conference, St. Louis, MO, pp. 1–11 (1999)

[240] R. Wüchner, A. Kupzok, K.-U. Bletzinger, A framework for stabilized partitioned analysis of thin membrane–wind interaction. *Internat. J. Numer. Methods Fluids* **54**/6–8, pp. 945–963 (2007)

[241] P. Wynn, on a procrustean technique for the numerical transformation of slowly convergent sequences and series. *Proc. Cambridge Philos. Soc.* **52**, pp. 663–671 (1956)

[242] P. Wynn, On a device for computing the $e_m(S_n)$ transformation. *Math. Tables Aids Comput.* **10**, pp. 91–96 (1956)

[243] P. Wynn, The Epsilon Algorithm and operational formulas of numerical analysis. *Math. Comp.* **15**, pp. 151–158 (1961)

[244] P. Wynn, Acceleration techniques for iterated vector and matrix problems. *Math. Comp.* **16**/79, pp. 301–322 (1962)

[245] C. Xu, J. Zhang, A survey of quasi-Newton equations and quasi-Newton methods for optimization. *Ann. Oper. Res.* **103**, pp. 213–234 (2001)

[246] H. Yabe, H. Ogasawara, M. Yoshino, Local and superlinear convergence of quasi-Newton methods based on modified secant conditions. *J. Comput. Appl. Math.* **205** pp. 617–632 (2007)

[247] D.M. Young, Iterative solution of large linear systems. Dover Publications, New York, ISBN 0–486–42548–7 (1971)

[248] T. J. Ypma, Historical development of the Newton-Raphson method. *SIAM Rev.* **37**/4, pp. 531–551 (1995)

[249] Y. Yuan, A modified BFGS algorithm for unconstrained optimization. *IMA J. Numer. Anal.* **11** , pp. 325–332 (1991)

[250] J.Z. Zhang, N.Y. Deng, L.H. Chen, New quasi-Newton equation and related methods for unconstrained optimization. *J. Optim. Theory Appl.* **102**/1, pp. 147–167 (1994)

[251] F. Zhang, The Schur Complement and Its Applications. Springer, New York, ISBN 0387242716 (2005)

[252] O.C. Zienkiewicz, R.L. Taylor, Finite element method: solid and fluid mechanics: dynamics and nonlinearity, Vol. **2**. New York, McGraw-Hill (1991)

[253] http://www.engineeringtoolbox.com/air-properties-d_156.html, April 2008.