Chapter 6

Strongly coupled methods for coupled fields

This chapter presents the most common strongly coupled/implicit methods employed to solve coupled field problems. This presentation seeks to provide a literature overview of the available approaches.

6.1 Equations to be solved

For the sake of clarity, the discretized equations of the thermo-mechanical problem at the next time instant, $n+1$e are recovered here

$$\mathbf{M}\ddot{\mathbf{u}}_{n+1} + \mathbf{f}_u^{\text{int}}(\theta_{n+1}, \mathbf{u}_{n+1}) - \mathbf{f}_{u,n+1}^{\text{ext}} = \mathbf{0}, \tag{6.1}$$

$$\mathbf{C}\dot{\theta}_{n+1} + \mathbf{f}_\theta^{\text{int}}(\theta_{n+1}, \mathbf{u}_{n+1}) - \mathbf{f}_{\theta,n+1}^{\text{ext}} = \mathbf{0}. \tag{6.2}$$

The complete definition of the material incremental discretized thermo-mechanical initial boundary value problem can be found in Chapter 4.

As only partitioned approaches are considered, the thermal and mechanical problems are solved separately, i.e., Equation (6.1) is solved considering a fixed temperature, and Equation (6.2) is solved assuming a fixed configuration. To ease the discussion, consider the existence of two functions $\mathcal{U}_{n+1}$ and $\mathcal{T}_{n+1}$ that represent these solution procedures at timestep $n+1$. These so-called mechanical and thermal solvers satisfy

$$\mathcal{U} : \mathscr{K}_{\theta,n+1} \to \mathscr{K}_{u,n+1}, \quad \mathbf{u} = \mathcal{U}_{n+1}(\theta), \tag{6.3}$$

$$\mathcal{T} : \mathscr{K}_{u,n+1} \to \mathscr{K}_{\theta,n+1}, \quad \theta = \mathcal{T}_{n+1}(\mathbf{u}). \tag{6.4}$$

See Chapter 4 for detailed information on them. In the following, the subscripts on the solvers will be dropped to avoid clutter.

The goal now is to consider functions, built from $\mathcal{U}$ and $\mathcal{T}$, whose roots are also the solutions to the thermo-mechanical problem (Equations (6.1) and (6.2)). Several examples can be provided. The most appropriate for the current use case are presented in what follows. They can be found in Uekermann (2016) in the context of fluid-structure interaction (FSI).

Consider the residues defined as,

$$\mathcal{R}_{\mathrm{J}}\colon \mathscr{K}_{u,n+1} \times \mathscr{K}_{\theta,n+1} \to K_{u,n+1} \times \mathscr{K}_{\theta,n+1}, \quad \mathcal{R}_{\mathrm{J}}(\mathbf{u},\theta) = \left\{ \begin{array}{c} \mathbf{u} - \mathcal{U}(\boldsymbol{\theta}) \\ \theta - \mathcal{T}(\mathbf{u}) \end{array} \right\}, \tag{6.5}$$

and

$$\mathcal{R}_{\mathrm{GS}}\colon \mathscr{K}_{\theta,n+1} \to \mathscr{K}_{\theta,n+1}, \quad \mathcal{R}_{\mathrm{GS}}(\theta) = \theta - \mathcal{T} \circ \mathcal{U}(\theta), \tag{6.6}$$

or

$$\mathcal{R}^{*}_{\mathrm{GS}}\colon \mathscr{K}_{u,n+1} \to \mathscr{K}_{u,n+1}, \quad \mathcal{R}^{*}_{\mathrm{GS}}(\mathbf{u}) = \mathbf{u} - \mathcal{U} \circ \mathcal{T}(\mathbf{u}), \tag{6.7}$$

where the subscript "J" stands for Jacobi and the subscript "GS" for Gauss-Seidel. The reason for this choice of subscripts is made clear in Section 6.3.1.

Since the methods described below for the solution of nonlinear systems of equations apply to both functions $\mathcal{R}_{\mathrm{J}}$ and $\mathcal{R}_{\mathrm{GS}}$, a general function denoted as $\mathcal{R}$, whose variable is $\mathbf{x}$, is considered instead. As already stated, the solution for the thermo-mechanical problem (Equations (6.1) and (6.2)) can be abstracted as the solution of

$$\mathcal{R}(\mathbf{x}) = 0. \tag{6.8}$$

To obtain simpler expressions in what follows, consider also the function

$$\mathcal{S}(\mathbf{x}) = \mathbf{x} - \mathcal{R}(\mathbf{x}), \tag{6.9}$$

whose fixed point is the solution to the nonlinear system of equation in Equation (6.8).

6.2 A classification scheme for iterative methods

Most methods available for the solution of systems of nonlinear equations, such as the one in Equation (6.8), are iterative methods. They can be more precisely defined lettting $\mathbf{x}^k, \mathbf{x}^{k-1}, \ldots$, whose superscripts correspond to the loop of the iteration method, be approximants to $\mathbf{x}_{n+1}$, whose subscript concerns the timestep

To better understand the landscape of available methods to solve nonlinear systems of equations, the iteration functions are classified according to the information they require following the classification scheme by Traub (1982). Let $\mathbf{x}^{k+1}$ be determined uniquely by information obtained at $\mathbf{x}^k, \mathbf{x}^{k-1}, \ldots$, including the derivatives of any order of $\mathcal{R}$. Let the function that maps $\mathbf{x}^k, \mathbf{x}^{k-1}, \ldots$ into $\mathbf{x}^{k+1}$ be called ϕ. Thus

$$\mathbf{x}^{k+1} = \phi\Big(\mathbf{x}^k, \mathcal{R}(\mathbf{x}^k), J_{\mathcal{R}}(\mathbf{x}^k), \ldots\Big), \tag{6.10}$$

where ϕ is called an iteration function, and $J_{\mathcal{R}}$ is the Jacobian of $\mathcal{R}$. To prevent clutering $\mathbf{x}^k$ will stand for its value as well as for the values of $\mathcal{R}(\mathbf{x}^k)$, $J_{\mathcal{R}}(\mathbf{x}^k)$ and further derivatives of higher order. Then ϕ is called a *one-point iteration function*. Most iteration functions that have been used for root-finding are one-point iteration functions. The most commonly known examples are the fixed point schemes and Newton's iteration method.

Next, let $\mathbf{x}^{k+1}$ be determined by new information at $\mathbf{x}^k$ and reused information at $\mathbf{x}^{k-1}, \ldots$ Thus

$$\mathbf{x}^{k+1} = \phi\left(\mathbf{x}^{k};\mathbf{x}^{k-1},\ldots\right). \tag{6.11}$$

Then ϕ is called a *one-point iteration function with memory*. The semicolon in Equation (6.11) separates the point at which new data are used from the points at which old data are reused. The secant iteration function is the best-known example of a one-point iteration function with memory.

Let $\mathbf{x}^{k+1}$ be determined by new information at $\mathbf{x}^{k}, \omega_1\left(\mathbf{x}^{k}\right), \ldots, \omega_i\left(\mathbf{x}^{k}\right)$, $i \geq 1$, where ω_i denote operations on $\mathbf{x}^k$. No old information is reused. Thus

$$\mathbf{x}^{k+1} = \phi\left[\mathbf{x}^{k},\omega_1\left(\mathbf{x}^{k}\right),\ldots,\omega_i\left(\mathbf{x}^{k}\right)\right]. \tag{6.12}$$

Then ϕ is called a *multipoint iterative function*. Such methods include the Aitken-Steffson method.

Finally, let $\mathbf{z}_j$ represent the quantities $\mathbf{x}^{j},\omega_1\left(\mathbf{x}^{j}\right),\ldots,\omega_i\left(\mathbf{x}^{j}\right)$, $i \geq 1$. Let

$$\mathbf{x}^{k+1} = \phi\left(\mathbf{z}^{k};\mathbf{z}^{k-1},\ldots\right). \tag{6.13}$$

Then ϕ is called a *multipoint iterative function with memory*. The semicolon in Equation (6.13) separates the points at which new data are used from the points at which old data are reused.

In the present work, the criteria used for the choice of the iterative method used fit roughly into the ones provided by Fang and Saad (2009) for problems in the context the electronic structure problems. They are

1. The dimensionality of the problem is large.
2. $\mathcal{R}(\mathbf{x})$ is continuously differentiable, but the analytic form of its derivative is not readily available, or it is costly to compute.
3. The cost of evaluating $\mathcal{R}(\mathbf{x})$ is computationally high.
4. The problem is noisy. In other words, the evaluated function values of $\mathcal{R}(\mathbf{x})$ usually contain errors.

Thus, the methods chosen must minimize the number of calls to $\mathcal{R}$, as it is expensive to compute. The amount of information saved from previous iterations must also be judiciously chosen as the problem's dimensionality is large, leading to memory limitations. Finally, the analytical form of the derivative $\mathcal{R}$ is also not available. Thus methods that use it must be discarded.

6.2.1 Predictor

Iterative procedures are considered to solve the thermo-mechanical problem at a given timestep $n+1$. As the first value approximating $\mathbf{x}_{n+1}$, one can employ the converged value of the previous timestep, $\mathbf{x}_n$. However, a very efficient way to increase the chances of stability and reduce computation time is to predict the optimal initial values at the beginning of every time step (Erbts and Düster, 2012; Erbts et al., 2015; Wendt et al., 2015). The prediction of the new solution by polynomial extrapolation is based on the converged solution of the last two or three timesteps. This method is based on polynomial vector extrapolation, which is relatively easy to implement, and the extra computational input is negligible.

The maximum polynomial under consideration is of the order two, i.e., the new solution is extrapolated from the results from the last three time steps. The predictors $\mathbf{x}^*$ for the order $p = 1$ and $p = 2$ polynomials read:

$$p = 1: \quad \mathbf{x}^*_{n+1} = 2\mathbf{x}_n - \mathbf{x}_{n-1}, \tag{6.14}$$
$$p = 2: \quad \mathbf{x}^*_{n+1} = 3\mathbf{x}_n - 3\mathbf{x}_{n-1} + \mathbf{x}_{n-2}. \tag{6.15}$$

6.2.2 Global Approaches

Following Dennis and Schnabel (1996), the terms "global," as in "global method" or "globally convergent algorithm," are here used to denote a method that is designed to converge to a local minimizer of a nonlinear functional or some solution of a system of nonlinear equations, from almost any starting point. The methods presented in this chapter do not qualify as global methods since if the initial trial is not close enough to the solution, they will not converge. There are, however, approaches to mitigate this problem. The ideas presented below apply with particular relevance to the Newton method (see Section 6.3.2) and related procedures. Their exposition follows Dennis and Schnabel (1996) where more details can be found.

Consider that the iterative solution method determines $\Delta\mathbf{x}^k$ in

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \Delta\mathbf{x}^k. \tag{6.16}$$

The two global approaches here considered both come into action after $\Delta\mathbf{x}^k$ has been computed by some appropriate method (see from Section 6.3 on). At this point, one decides whether to accept the step $\Delta\mathbf{x}^k$ or to choose $\mathbf{x}^{k+1}$ by a global strategy.

A solution to the system of equations (6.8) clearly also satisfies

$$r(\mathbf{x}) = 0, \quad \text{where } r \equiv \frac{1}{2}\|\mathcal{R}\|_2^2 : \mathbb{R}^n \to \mathbb{R}, \tag{6.17}$$

so the problem can be regarded as an unconstrained minimization problem, with caveat that local minimizers of r may not be the solution to the system of equations (6.8).

The basic idea of a global method for unconstrained minimization is geometrically obvious: take steps that lead "downhill" for the function r. More precisely, one chooses a direction $\mathbf{p}$ from the current point $\mathbf{x}^k$ in which r decreases initially, and a new point $\mathbf{x}^{k+1}$ in this direction from $\mathbf{x}^k$ such that $r(\mathbf{x}^{k-1}) < r(\mathbf{x}^k)$. Such a direction is called a descent direction.

An important question to ask is, "What is a descent direction for problem (6.17)?". It is any direction $\mathbf{p}$ for which $\nabla r(\mathbf{x}^k)^T\mathbf{p} < 0$, where

$$\nabla r\left(\mathbf{x}^k\right) = J_{\mathcal{R}}\left(\mathbf{x}^k\right)^T \mathcal{R}\left(\mathbf{x}^k\right), \tag{6.18}$$

where $J_{\mathcal{R}}(\mathbf{x}^k)$ is the Jacobian matrix of $\mathcal{R}$ at $\mathbf{x}^k$. Therefore, the steepest-descent direction for (6.17) is along $-J_{\mathcal{R}}(\mathbf{x}^k)^T\mathcal{R}(\mathbf{x}^k)$.

The Newton step for the update equation (6.16) is (see Section 6.3.2)

$$\Delta \mathbf{x}_N^k = -J_{\mathcal{R}}\left(\mathbf{x}^k\right)^{-1} \mathcal{R}\left(\mathbf{x}^k\right), \tag{6.19}$$

and it is a descent direction, since

$$\nabla r\left(\mathbf{x}^k\right)^T \Delta \mathbf{x}_N^k = -\mathcal{R}\left(\mathbf{x}^k\right)^T J_{\mathcal{R}}\left(\mathbf{x}^k\right) J_{\mathcal{R}}\left(\mathbf{x}^k\right)^{-1} \mathcal{R}\left(\mathbf{x}^k\right) = -\mathcal{R}\left(\mathbf{x}^k\right)^T \mathcal{R}\left(\mathbf{x}^k\right) < 0 \tag{6.20}$$

as long as $\mathcal{R}\left(\mathbf{x}^k\right) \neq \mathbf{0}$. Hence, the appropriateness of these methods to the Newton method and related methods.

Since the Newton step yields a root of

$$M^k\left(\mathbf{x}^k + \Delta \mathbf{x}^k\right) = \mathcal{R}\left(\mathbf{x}^k\right) + J_{\mathcal{R}}\left(\mathbf{x}^k\right) \Delta \mathbf{x}^k, \tag{6.21}$$

it also goes to a minimum of the quadratic function

$$\begin{aligned} \hat{m}^k\left(\mathbf{x}^k + \Delta \mathbf{x}^k\right) &\equiv \frac{1}{2} M^k\left(\mathbf{x}^k + \Delta \mathbf{x}^k\right)^T M^k\left(\mathbf{x}^k + \Delta \mathbf{x}^k\right) \\ &= \frac{1}{2} \mathcal{R}\left(\mathbf{x}^k\right)^T \mathcal{R}\left(\mathbf{x}^k\right) + \left(J_{\mathcal{R}}\left(\mathbf{x}^k\right)^T \mathcal{R}\left(\mathbf{x}^k\right)\right)^T \Delta \mathbf{x}^k \\ &\qquad + \frac{1}{2} \Delta \mathbf{x}^{k\,T} \left(J_{\mathcal{R}}\left(\mathbf{x}^k\right)^T J_{\mathcal{R}}\left(\mathbf{x}^k\right)\right) \Delta \mathbf{x}^k, \end{aligned} \tag{6.22}$$

because $\hat{m}^k\left(\mathbf{x}^k + \Delta \mathbf{x}^k\right) \geq 0$ for all $\Delta \mathbf{x}^k$ and $\hat{m}^k\left(\mathbf{x}^k + \Delta \mathbf{x}_N^k\right) = 0$. Therefore, $\Delta \mathbf{x}_N^k$ is a descent direction for $\hat{m}^k$, and since the gradients at $\mathbf{x}^k$ of $\hat{m}^k$ and r are the same, it is also a descent direction for r.

The above development motivates how the global methods to be described are applied, i.e., they are applied to the quadratic model $\hat{m}^k(\mathbf{x})$. Since $\nabla^2 \hat{m}^k(\mathbf{x}^k) = J_{\mathcal{R}}(\mathbf{x}^k)^T J_{\mathcal{R}}(\mathbf{x}^k)$, this model is positive definite as long as $J_{\mathcal{R}}(\mathbf{x}^k)$ is nonsingular, which is consistent with the fact that $\mathbf{x}^k + \Delta \mathbf{x}_N^k$ is the unique root of $M^k(\mathbf{x})$ and thus the unique minimizer of $\hat{m}^k(\mathbf{x})$ in this case. Thus, the model $\hat{m}^k(\mathbf{x})$ has the attractive properties that its minimizer is the Newton point for the original problem, and that all its descent directions are descent directions for $r(\mathbf{x})$ because $\nabla \hat{m}^k(\mathbf{x}^k) = \nabla r(\mathbf{x}^k)$. Therefore methods based on this model, by going downhill and trying to minimize $\hat{m}^k(\mathbf{x})$, will combine Newton's method for nonlinear equations with global methods for an associated minimization problem.

If the Jacobian of $\mathcal{R}$ is not available and its estimate is of poor quality, the global procedure may be compromised (Kelley, 2003). However, these procedures may be unnecessary in the present use case since the initial trial is probably close enough to the solution, even without accounting for the improvements coming from more carefully chosen initial shots through predictors (see Section 6.2.1). Fang and Saad (2009) also employ a simple restarting procedure instead of a global convergence strategy. If in two consecutive values of $\mathcal{R}$, $\mathcal{R}_{\text{old}}$ and $\mathcal{R}_{\text{new}}$, $\|\mathcal{R}_{\text{new}}\|$ is much larger than $\|\mathcal{R}_{\text{old}}\|$, the solution procedure is restarted, with the new initial trial values corresponding to $\mathcal{R}_{\text{old}}$. They suggest r between 0.1 and 0.3, with $\|\mathcal{R}_{\text{old}}\| < r\|\mathcal{R}_{\text{new}}\|$ leading to a restart. In their opinion, global approaches such as those suggested below are too expensive when the evaluation of $\mathcal{R}$ is also costly to compute.

Line search The line search approach is based on the traditional idea of backtracking along the Newton direction if a complete Newton step is unsatisfactory. More precisely given a descent direction $\mathbf{p}^k$, a step in that direction is taken as

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \lambda^k \mathbf{p}^k, \tag{6.23}$$

for some $\lambda^k > 0$ that makes $\mathbf{x}^{k+1}$ an acceptable iterate. The common procedure is to first try $\lambda_k = 1$ and only if this fails backtrack in a systematic way along the direction defined by that step. See Dennis and Schnabel (1996) for a full discussion on the choice of λ_k.

Trust region algorithms The trust region algorithm is based on estimating the region in which the local model, underlying Newton's method, can be trusted to represent the function adequately and taking a step to approximately minimize the model in this region. It drops the assumption that the step must be in the Newton direction. $\hat{m}^k(\mathbf{x}^k + \Delta\mathbf{x}^k)$ is approximately minimized subject to $\|\mathbf{x}^k\|_2 \leq \delta^k$. If $\delta^k \geq \left\| J_{\mathcal{R}}(\mathbf{x}^k)^{-1}\mathcal{R}(\mathbf{x}^k)\right\|_2$, then the step attempted is the Newton step. Otherwise, for the locally constrained optimal step, it is

$$\Delta\mathbf{x}^k = -\left(J_{\mathcal{R}}\left(\mathbf{x}^k\right)^T J_{\mathcal{R}}\left(\mathbf{x}^k\right) + \mu^k\mathbf{I}\right)^{-1} J_{\mathcal{R}}\left(\mathbf{x}^k\right)^T \mathcal{R}\left(\mathbf{x}^k\right), \tag{6.24}$$

for μ^k such that $\|\Delta\mathbf{x}^k\|_2 \cong \delta^k$. For the details on the choice of δ^k see Dennis and Schnabel (1996).

6.2.3 Convergence criteria

For an iterative method to be useful, there must be reasonable criteria to determine its convergence. The iteration residual is defined as

$$\mathbf{r}^k = \mathcal{R}(\mathbf{x}^k), \tag{6.25}$$

and if it is equal to zero then $\mathbf{x}$ is the solution to the system of nonlinear equations, i.e.,

$$\mathbf{r} = \mathcal{R}(\mathbf{x}) = \mathbf{0}, \tag{6.26}$$

and hence, a reasonable convergence measure for the iteration procedure.

The discrete l^2-norm can be used to obtain a scalar representative of the vectorial residual $\mathbf{r}^k = \left(r^{k,1}, \ldots, r^{k,n_{\text{unknown}}}\right)^T$ as

$$\left\|\mathbf{r}^k\right\|_{L^2} = \sqrt{\sum_i \left(r^{k,i}\right)^2}. \tag{6.27}$$

Directly using (6.27) yields an absolute convergence criterion

$$\left\|\boldsymbol{r}^k\right\|_{l^2} < \epsilon_{\text{abs}}. \tag{6.28}$$

with $\epsilon_{\text{abs}} > 0$ as an absolute convergence tolerance, with convergence being achieved when the above condition is satisfied. However, since the absolute value of the $r^{k,i}$'s can change by orders of magnitude during one simulation, an absolute measure is not appropriate in all situations. A relative measure solves this problem by setting the residual in relation with the current coupling iterate values as

$$\frac{\left\|\mathbf{r}^k\right\|_{l^2}}{\left\|\mathbf{x}^k\right\|_{l^2}} < \epsilon_{\text{rel}}. \tag{6.29}$$

A relative convergence measure can fail to work correctly when the coupling iterate values are close to zero, and rounding errors occur. Thus, a combination of absolute and relative measures, where the absolute measure takes care of close to zero cases, and the relative handles all other cases, is often a good choice.

6.3 One-point iteration function

6.3.1 Fixed-point approaches

The application of the fixed-point method to obtain the roots of $\mathcal{R}$ yields

$$\mathbf{x}^{k+1} = \mathcal{S}(\mathbf{x}^k) = \mathbf{x}^k - \mathcal{R}(\mathbf{x}^k). \tag{6.30}$$

See Figure 6.1 for its geometric interpretation in one dimension.

If the particular functions defined on Equations (6.5) and (6.6) are used, one finds the two basic Schwarz procedures commonly employed in strongly coupled solution procedures. They are the additive or block Jacobi and the parallel Scharwz or Gauss-Seidel procedures. The names originate from domain decomposition, and justify the subscripts employed in Equations (6.5) and (6.6).

6.3.1.1 Block Jacobi or Schwarz additive

Applying the fixed-point approach to $\mathcal{R}_{\text{J}}$ (Equation (6.5)), yields

$$\left\{\mathbf{u}^{k+1}, \theta^{k+1}\right\}^T = \mathcal{S}_{\text{J}}(\mathbf{u}^k, \theta^k) \tag{6.31}$$

$$= \left\{\mathbf{u}^k, \theta^k\right\}^T - \mathcal{R}_{\text{J}}(\mathbf{u}^k, \theta^k), \tag{6.32}$$

It is the same as solving both the mechanical (Equation (6.1)) and the thermal problem (Equation (6.2)) in parallel. Such a procedure is said to be Schwarz additive or block Jacobi, refering to the similarities with the procedure for the solution of linear systems of equations with the same name i.e.,

$$\mathbf{u}^{k+1} = \mathcal{U}(\theta^k), \tag{6.33}$$

$$\theta^{k+1} = \mathcal{T}(\mathbf{u}^k). \tag{6.34}$$

Box 6.1 shows the pseudo-code for the block Jacobi approach.

6.3.1.2 Block Gauss-Seidel or Schwarz multiplicative

Applying the fixed-point approach to $\mathcal{R}_{\text{GS}}$ (Equation (6.5)), yields

$$\theta^{k+1} = \mathcal{S}_{\text{GS}}(\theta^k) = \theta^k - \mathcal{R}_{\text{GS}}(\theta^k). \tag{6.35}$$

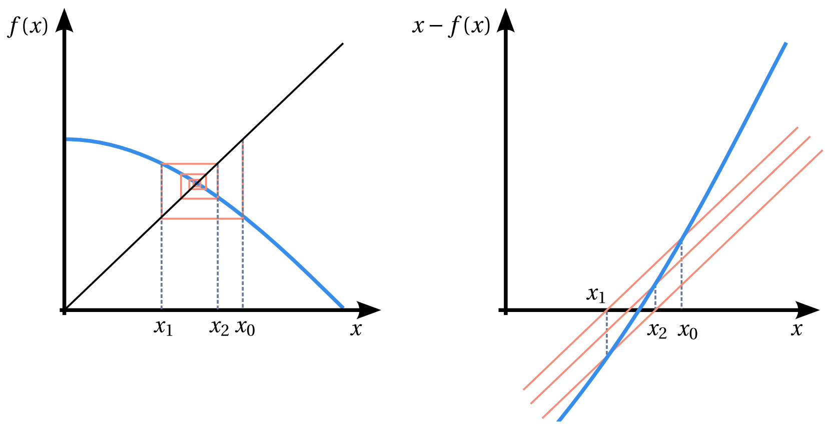

Figure 6.1: Geometric interpretation of the fixed-point iteration method in one dimension. The fixed-point of f is sought, which is equivalent to the root of $x - f(x)$.

Thus, the fields are solved sequentially, where the output of the first solver is the input of the second solver. This the solution procedure is said to be Scharwz multiplicative or block Gauss-Seidel.

$$\mathbf{u}^{k+1} = \mathcal{U}(\theta^k), \tag{6.36}$$

$$\theta^{k+1} = \mathcal{T}(\mathbf{u}^{k+1}). \tag{6.37}$$

One of the fields must be chosen as the first, and this may be crucial for the stability and convergence rate of the approach (Joosten et al., 2009). Here, the focus is on the sequence coinciding with the isothermic split, i.e., first, the mechanical problem is solved at a fixed temperature. Then the thermal problem is solved at a fixed configuration.

Box 6.2 shows the pseudo-code for the block Gauss-Seidel approach.

6.3.2 Newton's method

The Newton-Raphson or Newton scheme is a very popular iterative solution procedure for nonlinear systems of equations, which under appropriate conditions converges quadratically (Dennis and Schnabel, 1996; Kelley, 2003). It can be applied to Equation (6.8) yielding

$$J_{\mathcal{R}}(\mathbf{x}^k)\Delta\mathbf{x}^k = -\mathcal{R}(\mathbf{x}^k), \tag{6.38}$$

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \Delta\mathbf{x}^k. \tag{6.39}$$

See Figure 6.2 for its geometric interpretation in one dimension.

In particular, using $\mathcal{R}_{\mathrm{J}}$, a few simplifications can be obtained. To ease the explanation, consider, a thermal residual operator $\mathcal{R}_u(\mathbf{u},\theta)$ and a mechanical residual operator $\mathcal{R}_\theta(\mathbf{u},\theta)$ defined to be the first and second components in the definition of

Box 6.1: Additive Schwarz procedure, also called block Jacobi, for one timestep.

(i) $\mathbf{u}^0 = \mathbf{u}_n$

(ii) $\theta^0 = \theta_n$

(iii) Set fixed-point counter to zero: $k = 0$

(iv) Enter the fixed-point loop

  (1) Solve the mechanical problem at fixed temeperature θ^k: $\mathbf{u}^{k+1} = \mathcal{U}(\theta^k)$

  (2) Solve the thermal problem at a fixed configuration $\mathbf{u}^k$: $\theta^{k+1} = \mathcal{T}(\mathbf{u}^k)$

  (3) If the desired accuracy has not been reached, update $k = k+1$ and go to step (1).

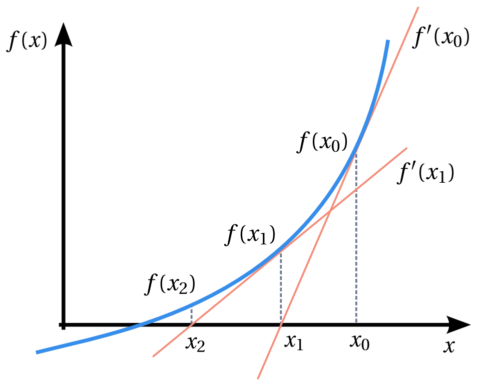

Figure 6.2: Geometric interpretation of the Newton method in one dimension for an example function f, whose derivative is denoted by f'.

$\mathcal{R}_\mathrm{J}$ (Equation (6.6)). Written in full

$$\mathcal{R}_u(\mathbf{u},\theta) = \mathbf{u} - \mathcal{U}(\theta) = 0, \tag{6.40}$$

$$\mathcal{R}_\theta(\mathbf{u},\theta) = \theta - \mathcal{T}(\mathbf{u}) = 0, \tag{6.41}$$

From this, a block Newton iteration can be written as

$$\begin{bmatrix} J_{\mathcal{R}_u}\left(\mathbf{u}^k,\theta^k\right) \\ J_{\mathcal{R}_\theta}\left(\mathbf{u}^k,\theta^k\right) \end{bmatrix} \begin{Bmatrix} \Delta\mathbf{u}^k \\ \Delta\theta^k \end{Bmatrix} = -\begin{Bmatrix} \mathcal{R}_u\left(\mathbf{u}^k,\boldsymbol{\theta}^k\right) \\ \mathcal{R}_\theta\left(\mathbf{u}^k,\boldsymbol{\theta}^k\right) \end{Bmatrix}, \tag{6.42}$$

and the update of the iteration variables reads

Box 6.2: Multiplicative Schwarz procedure, also called block Gauss-Seidel, for one timestep.

(i) $\mathbf{u}^0 = \mathbf{u}_n$

(ii) $\theta^0 = \theta_n$

(iii) Set fixed-point counter to zero: $k = 0$

(iv) Enter the fixed-point loop

   (1) Solve the mechanical problem at fixed temeperature θ^k: $\mathbf{u}^{k+1} = \mathcal{U}(\theta^k)$

   (2) Solve the thermal problem at a fixed configuration $\mathbf{u}^{k+1}$: $\theta^{k+1} = \mathcal{T}(\mathbf{u}^{k+1})$

   (3) If the desired accuracy has not been reached, update $k = k+1$ and go to step (1).

$$\left\{ \begin{array}{c} \mathbf{u}^{k+1} \\ \boldsymbol{\theta}^{k+1} \end{array} \right\} = \left\{ \begin{array}{c} \mathbf{u}^{k} \\ \boldsymbol{\theta}^{k} \end{array} \right\} + \left\{ \begin{array}{c} \Delta\mathbf{u}^{k} \\ \Delta\boldsymbol{\theta}^{k} \end{array} \right\}. \tag{6.43}$$

The system of equations in Equation (6.42) can be further simplified following Degroote (2010) considering the definitions of the mechanical and thermal residuals and taking their derivatives. It yields

$$\left[\begin{array}{cc} \mathbf{I} & -J_{\mathcal{U}}\left(\theta^k\right) \\ -J_{\mathcal{T}}\left(\mathbf{u}^k\right) & \mathbf{I} \end{array} \right] \left\{ \begin{array}{c} \Delta\mathbf{u}^k \\ \Delta\theta^k \end{array} \right\} = - \left\{ \begin{array}{c} \mathcal{R}_u\left(\mathbf{u}^k, \boldsymbol{\theta}^k\right) \\ \mathcal{R}_\theta\left(\mathbf{u}^k, \boldsymbol{\theta}^k\right) \end{array} \right\}, \tag{6.44}$$

Solving for $\Delta\mathbf{u}^k$ and $\Delta\theta^k$, one finds

$$\left(\mathbf{I} + J_{\mathcal{U}}(\theta^k) J_{\mathcal{T}}(\mathbf{u}^k)\right) \Delta\mathbf{u}^k = -\mathcal{R}_u(\mathbf{u}^k, \theta^k) + J_{\mathcal{U}}(\theta^k)\mathcal{R}_\theta(\mathbf{u}^k, \theta^k), \tag{6.45}$$

$$\left(\mathbf{I} + J_{\mathcal{T}}(\mathbf{u}^k) J_{\mathcal{U}}(\theta^k)\right) \Delta\theta^k = -\mathcal{R}_\theta(\theta^k, \mathbf{u}^k) + J_\theta(\mathbf{u}^k)\mathcal{R}_u(\mathbf{u}^k, \theta^k). \tag{6.46}$$

Thus, the Jacobians now needed are $J_{\mathcal{U}}$ and $J_{\mathcal{T}}$. See Section 6.4.1 for the practical application of this.

Every iteration of the Newton scheme involves at least one invocation of the thermal and mechanical solvers when computing $\mathcal{R}\left(\mathbf{u}^k\right)$ or both $\mathcal{R}_u\left(\mathbf{u}^k, \boldsymbol{\theta}^k\right)$ and $\mathcal{R}_\theta\left(\mathbf{u}^k, \boldsymbol{\theta}^k\right)$. The critical point for black-box equation coupling is how to obtain the derivative information in the Jacobi matrices. In different ways, some of the methods presented next find approximations for the required Jacobian times vector products.

6.3.3 Constant Underrelaxation

One of the most straightforward ways to stabilize an iterative method is to use constant underrelaxation (Gatzhammer, 2014). The relaxation is performed as follows

$$\mathbf{x}^{k+1} = (1-\omega)\mathbf{x}^k + \omega\left(\mathbf{x}^k - \mathcal{R}(\mathbf{x}^k)\right) = \mathbf{x}^k - \omega\mathcal{R}(\mathbf{x}^k), \tag{6.47}$$

where ω is the relaxation factor chosen in the range $0 < \omega < 1$, which corresponds to an underrelaxation, to achieve a stabilizing effect.

Applying to Equation (6.5)

$$\left\{\begin{array}{c} \mathbf{u}^{k+1} \\ \theta^{k+1} \end{array}\right\} = (1-\omega)\left\{\begin{array}{c} \mathbf{u}^{k} \\ \theta^{k} \end{array}\right\} + \omega\left\{\begin{array}{c} \mathcal{U}(\theta^k) \\ \mathcal{T}(\mathbf{u}^k) \end{array}\right\} \tag{6.48}$$

Applying to Equation (6.6)

$$\theta^{k+1} = (1-\omega)\theta^k + \omega\mathcal{T}\circ\mathcal{U}(\theta^k). \tag{6.49}$$

Constant underrelaxation works well if ω is close to 1 but leads to a slow convergence if ω has to be chosen close to 0. Thus, the constant underrelaxation method creates unmanageable computational costs for severe instabilities. The optimal ω is not necessary the largest stable one (Gatzhammer, 2014) and has to be set empirically. In what follows, alternative methods are discussed to decrease the number of iterations necessary while maintaining stability.

Box 6.3: Constant underrelaxation applied to the block Gauss-Seidel scheme.

(i) $\theta^0 = \theta^p_{n+1}$

(ii) Set fixed-point counter to zero: $k = 0$

(iii) Enter the fixed-point loop

  (1) Solve the mechanical problem at fixed temeperature θ^k: $\mathbf{u}^{k+1} = \mathcal{U}(\theta^k)$

  (2) Solve the thermal problem at a fixed configuration $\mathbf{u}^{k+1}$: $\theta^{k+1} = \mathcal{T}(\mathbf{u}^{k+1})$

  (3) Compute θ^{k+1} using constant relaxation (Equation (6.47))

  (4) If the desired accuracy has not been reached, update $k = k+1$ and go to step (1).

6.4 One-point iteration function with memory

6.4.0.1 Aitken relaxation

The so-called Aitken Δ^2 relaxation method was introduced by Irons and Tuck (1969) as a modified Aitken Δ^2 that does not require the computation of the function twice per iteration as in the original method. It has been widely used in the context of FSI (Irons and Tuck, 1969; Küttler and Wall, 2008; Joosten et al., 2009; Küttler and Wall, 2009; Erbts et al., 2015; Wendt et al., 2015). It has also been used in the context of thermo-mechanics by Danowski et al. (2013).

In the one-dimensional case, this method resembles the secant method applied to the fixed point problem, which can be used to solve nonlinear equations without differentiation. Calling it an Aitken method is perhaps a misnomer since, in the Aitken-Steffensen method, the function values are computed twice per iteration (see Section 6.5.3). It is more closely related to secant methods, reusing values from previous iterations. This version of Aitken's Δ^2 method provides a dynamic under relaxation, which can be used to improve the convergence/stability properties of the coupling algorithm.

Assume that f is the function whose fixed point is sought. The linear interpolation between two points already known of the function, $(a, f(a))$ and $(b, f(b)$ is

$$y = \frac{f(b) - f(a)}{b - a}(x - a) + f(a). \tag{6.50}$$

The fixed point of this approximation is

$$c = \frac{f(b) - f(a)}{b - a}(c - a) + f(a). \tag{6.51}$$

Thus, after rearranging,

$$c = \frac{af(b) - bf(a)}{\big(a - f(a)\big) - \big(b - f(b)\big)} \tag{6.52}$$

This can be rewritten as

$$c = \big(1 - \omega_b\big)\, b + \omega_b f(b) \quad \text{with } \omega_b = \frac{a - b}{\big(a - f(a)\big) - \big(b - f(b)\big)} \tag{6.53}$$

Anticipating the next iteration step,

$$d = \big(1 - \omega_c\big)\, c + \omega_c f(c) \quad \text{with } \omega_c = \frac{c - b}{\big(b - f(b)\big) - \big(c - f(c)\big)} \tag{6.54}$$

a convenient expression for updating the relaxation factor may be found, i.e.

$$\omega_c = -\omega_b \frac{f(b) - b}{(c - f(c)) - (f(b) - b)}. \tag{6.55}$$

See Figure 6.3 for its geometric interpretation in one dimension.

Now, for the vector case, the next step is to work out the solution to the current iteration from the outcome of the previous iteration $\mathbf{x}^k$ plus a new increment $\Delta\mathbf{x}^k$

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \Delta\mathbf{x}^k. \tag{6.56}$$

The increment reads

$$\Delta\mathbf{x}^k = \omega^k \left(\mathcal{S}(\mathbf{x}^{(k)}) - \mathbf{x}^{(k)}\right) = -\omega^k \mathcal{R}(\mathbf{x}^k). \tag{6.57}$$

with ω^k being the relaxation coefficient. This coefficient is updated in every iteration cycle as a function of two previous residuals

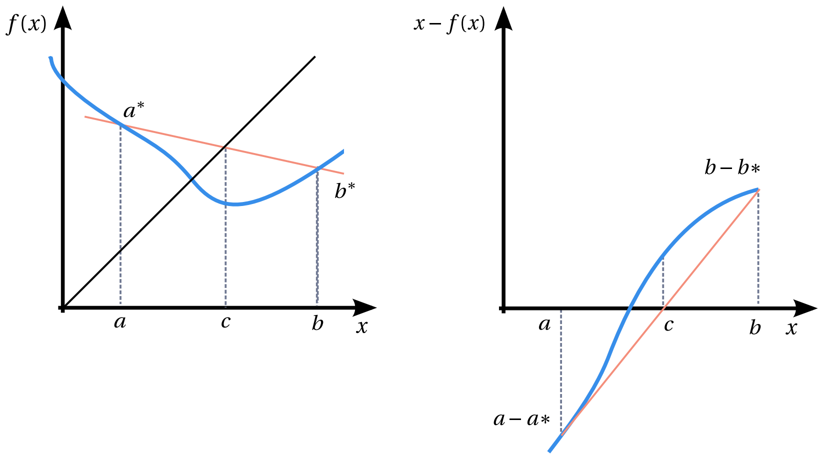

Figure 6.3: Geometric interpretation of the Aitken relaxation in one dimension for an example function f and corresponding interpreation as the secant method.

$$\omega^k = -\omega^{k-1} \frac{\left(\mathbf{r}^{(k)} - \mathbf{r}^{(k-1)}\right)^{\mathrm{T}} \mathbf{r}^{(k-1)}}{\left(\mathbf{r}^{(k)} - \mathbf{r}^{(k-1)}\right)^2}. \tag{6.58}$$

Check signs!!!

Comparing with Equations (6.38) and (6.39), ω^k can be, in a sense, regarded as an approximation to the inverse of the Jacobian. Dynamic relaxation is also easy to implement, and the additional computational input is acceptable since only inner vector products must be performed. See Box 6.4 for the pseudocode.

6.4.1 Multi-secant methods

The following exposition follows closely Fang and Saad (2009). In quasi-Newton methods the Jacobian is updated in each iteration using a rank-one update. Standard quasi-Newton methods require the updated J_{k+1} to satisfy the following secant condition

$$J_{\mathcal{R}}^{k+1} \Delta\mathbf{x}^k = \Delta\mathcal{R}^k, \tag{6.59}$$

where $\Delta\mathcal{R}^k \equiv \mathcal{R}\left(\mathbf{x}^{k+1}\right) - \mathcal{R}\left(\mathbf{x}^k\right)$. Furthermore, another common requirement is the following so-called no-change condition

$$J_{\mathcal{R}}^{k+1}\mathbf{q} = J_{\mathcal{R}}^{k}\mathbf{q} \quad \forall \mathbf{q} \text{ such that } \mathbf{q}^{\mathrm{T}}\Delta\mathbf{x}^k = 0, \tag{6.60}$$

which stipulates that there be no new information from $J_{\mathcal{R}}^{k}$ to $J_{\mathcal{R}}^{k+1}$ along any direction $\mathbf{q}$ orthogonal to $\Delta\mathbf{x}^k$.

Broyden (1965) developed a method satisfying both secant condition (Equation (6.59)) and the no-change condition (Equation (6.60)). By simply imposing these conditions he arrived at the update formula

Box 6.4: Aitken relaxation for one timestep.

```
(i) Set nonlinear counter to zero: $k = 0$
(ii) $\mathbf{x}^k = \mathbf{x}^p_{n+1}$
(iii) Enter the nonlinear loop
    (1) Compute $\mathcal{R}(\mathbf{x}^k)$, which implies the solution of the mechanical and
        the thermal problems, $\mathcal{U}$ and $\mathcal{T}$, respectively.
    (2) if $k = 0$:
        • Compute $\mathbf{x}^{k+1}$ using constant relaxation (Equation (6.47))
    (3) else:
        • Compute $\mathbf{x}^{k+1}$ using Aitken relaxation relaxation
          (Equations (6.57) and (6.58))
        • Save the current residue $\mathbf{r}^k = \mathcal{R}^k$.
    (4) If the desired accuracy has not been reached, update $k = k+1$ and go
        to step (1).
```

$$J_{\mathcal{R}}^{k+1} = J_{\mathcal{R}}^{k} + \left(\Delta\mathcal{R}^k - J_{\mathcal{R}}^{k}\Delta\mathbf{x}^k\right)\frac{\Delta\mathbf{x}^{k^{\mathrm{T}}}}{\Delta\mathbf{x}^{k^{\mathrm{T}}}\Delta\mathbf{x}^k}. \tag{6.61}$$

Matrix $J_{\mathcal{R}}^{k+1}$ in Equation (6.61) is the unique matrix satisfying both conditions (6.59) and (6.60). The Broyden update can also be obtained by minimizing $E\left(J_{\mathcal{R}}^{k+1}\right) = \left\|J_{\mathcal{R}}^{k+1} - J_{\mathcal{R}}^{k}\right\|_F^2$ with respect to terms of $J_{\mathcal{R}}^{k+1}$, subject to the secant condition (6.59).

It may seem at first that Broyden's first method can be expensive since computing the quasi-Newton step $\Delta\mathbf{x}^k$ requires solving a linear system at each iteration. However, note that, typically, the approximate Jacobian is a small rank modification of a diagonal matrix (or a matrix that is easy to invert); hence, the cost to obtain this solution is not too high as long as the number of steps is not too large.

An alternative is Broyden's second method that approximates the inverse Jacobian instead of the Jacobian itself. $G_{\mathcal{R}}^{k}$ is used to denote the estimated inverse Jacobian at the k th iteration. The secant condition (Equation (6.59)) now reads

$$G_{\mathcal{R}}^{k+1}\Delta\mathcal{R}^k = \Delta\mathbf{x}^k \tag{6.62}$$

By minimizing $E\left(G_{\mathcal{R}}^{k+1}\right) = \left\|G_{\mathcal{R}}^{k+1} - G_{\mathcal{R}}^{k}\right\|_F^2$ with respect to $G_{\mathcal{R}}^{k+1}$ subject to Equation (6.62), the following update formula is found for the inverse Jacobian

$$G_{\mathcal{R}}^{k+1} = G_{\mathcal{R}}^{k} + \left(\Delta\mathbf{x}_k - G_{\mathcal{R}}^{k}\Delta\mathcal{R}^k\right)\frac{\Delta\mathcal{R}^{k^{\mathrm{T}}}}{\Delta\mathcal{R}^{k^{\mathrm{T}}}\Delta\mathcal{R}^k} \tag{6.63}$$

which is also the only update satisfying both the secant condition (Equation (6.62)) and the no-change condition for the inverse Jacobian

$$G_{\mathcal{R}}^{k}\mathbf{q} = G_{\mathcal{R}}^{k+1}\mathbf{q} \quad \forall\mathbf{q} \text{ such that } \mathbf{q}^{\mathrm{T}}\Delta\mathcal{R}^{k} = 0. \tag{6.64}$$

The update formula in Equation (6.61) can also be obtained in terms of $G_{\mathcal{R}}^{k} \equiv J_{\mathcal{R}}^{k\,-1}$ by applying the Sherman-Morrison formula

$$G_{\mathcal{R}}^{k+1} = G_{\mathcal{R}}^{k} + \left(\Delta\mathbf{x}^{k} - G_{\mathcal{R}}^{k}\Delta\mathcal{R}^{k}\right)\frac{\Delta\mathbf{x}^{k^{\mathrm{T}}}G_{\mathcal{R}}^{k}}{\Delta\mathbf{x}^{k^{\mathrm{T}}}G_{\mathcal{R}}^{k}\Delta\mathcal{R}^{k}} \tag{6.65}$$

This shows, as was explained earlier, that to solve the Jacobian system associated with Broyden's first approach can be reduced to a set of update operations that are not more costly than those required by the second update. Note, however, that the above formula requires the inverse of the initial Jacobian.

From Equation (6.61) and Equation (6.63) it is possible to define Broyden's family of updates, in which an update formula takes the general form

$$G_{\mathcal{R}}^{k+1} = G_{\mathcal{R}}^{k} + \left(\Delta\mathbf{x}^{k} - G_{\mathcal{R}}^{k}\Delta\mathcal{R}^{k}\right)\mathbf{v}_{k}^{\mathrm{T}} \tag{6.66}$$

where $\mathbf{v}_{k}^{\mathrm{T}}\Delta\mathcal{R}^{k} = 1$ so that the secant condition (6.59) holds. Note that the secant condition (6.62) is equivalent to condition (6.59). Some authors called Broyden's first method Broyden's good update and Broyden's second method as Broyden's bad update. These are two particular members of Broyden's family.

6.4.1.1 Generalized Broyden

The multi-secant methods provide an approximation to the Jacobian in Equation (6.38) or Equation (6.42) using information from previous iterations. A generalized Broyden's method with a flexible rank update on the inverse Jacobian, satisfying a set of m secant equations

$$G_{\mathcal{R}}^{k}\Delta\mathcal{R}^{i} = \Delta\mathbf{x}^{i} \quad \text{for } i = k-m,\ldots,k-1 \tag{6.67}$$

where it is assumed $\Delta\mathcal{R}^{k-m},\ldots,\Delta\mathcal{R}^{k-1}$ are linearly independent and $m \leqslant n$ can also be described. Aggregating Equations (6.67) in matrix form, they can be rewriten it as

$$G_{\mathcal{R}}^{k}\mathscr{R}^{k} = \mathscr{X}^{k}. \tag{6.68}$$

where

$$\mathscr{R}^{k} = \left[\Delta\mathcal{R}^{k-m}\cdots\Delta\mathcal{R}^{k-1}\right], \quad \mathscr{X}^{k} = \left[\Delta\mathbf{x}^{k-m}\cdots\Delta\mathbf{x}^{k-1}\right] \in \mathbb{R}^{n\times m} \tag{6.69}$$

The no-change condition corresponding to (6.60) is

$$\left(G_{\mathcal{R}}^{k} - G_{\mathcal{R}}^{k-m}\right)\mathbf{q} = 0 \tag{6.70}$$

for all $\mathbf{q}$ orthogonal to the subspace spanned by $\Delta\mathcal{R}^{k-m},\ldots,\Delta\mathcal{R}^{k-1}$, the columns of $\mathscr{R}^{k}$. In the end, this yields

$$G_{\mathcal{R}}^{k} = G_{\mathcal{R}}^{k-m} + \left(\mathscr{X}^{k} - G_{\mathcal{R}}^{k-m}\mathscr{R}^{k}\right)\left(\mathscr{R}^{k^{\mathrm{T}}}\mathscr{R}^{k}\right)^{-1}\mathscr{R}^{k^{\mathrm{T}}} \tag{6.71}$$

a rank-m update formula. Note that $\operatorname{rank}\left(\mathscr{R}^k\right) = m$. The update formula for $\mathbf{x}^{k+1}$ is

$$\mathbf{x}^{k+1} = \mathbf{x}^k - G_{\mathcal{R}}^k \mathcal{R}^k \tag{6.72}$$

$$= \mathbf{x}^k - G_{\mathcal{R}}^{k-m} \mathcal{R}^k - \left(\mathscr{X}^k - G_{\mathcal{R}}^{k-m} \mathscr{R}^k\right) \left(\mathscr{R}^{k\mathrm{T}} \mathscr{R}^k\right)^{-1} \mathscr{R}^{k\mathrm{T}} \mathcal{R}^k \tag{6.73}$$

$$= \mathbf{x}^k - G_{\mathcal{R}}^{k-m} \mathcal{R}^k - \left(\mathscr{X}^k - G_{\mathcal{R}}^{k-m} \mathscr{R}^k\right) \gamma_k \tag{6.74}$$

where the column vector γ_k is obtained by solving the normal equations $\left(\mathscr{R}^{k\mathrm{T}} \mathscr{R}^k\right) \gamma_k = \mathscr{R}^{k\mathrm{T}} \mathcal{R}^k$, which is equivalent to solving the least squares problem

$$\min_{\gamma} \left\| \mathscr{R}^k \gamma - \mathcal{R}^k \right\|_2 . \tag{6.75}$$

Note that in Equation (6.74), if $\mathscr{R}^k$ is square and of full rank, then for any $G_{\mathcal{R}}^{k-m}$,

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \mathscr{X}^k \mathscr{R}^{k^{-1}} \mathcal{R}^k , \tag{6.76}$$

the same form as that in the standard secant method.

6.4.1.2 Anderson mixing

The Anderson mixing scheme [5] takes the latest m steps into account to obtain a better approximation to $\mathbf{x}_{n+1}$ without evaluating $\mathcal{R}$ again. Consider

$$\bar{\mathbf{x}}^k = \mathbf{x}^k - \sum_{i=k-m}^{k-1} \gamma_i^k \Delta \mathbf{x}^i = \mathbf{x}_k - \mathscr{X}^k \gamma^k , \tag{6.77}$$

$$\bar{\mathcal{R}}^k = \mathcal{R}^k - \sum_{i=k-m}^{k-1} \gamma_i^k \Delta \mathcal{R}^i = \mathcal{R}^k - \mathscr{R}^k \gamma^k , \tag{6.78}$$

where $\Delta \mathbf{x}^i = \mathbf{x}^{i+1} - \mathbf{x}^i$ and $\Delta \mathcal{R}^i = \mathcal{R}^{i+1} - \mathcal{R}^i$, $\mathscr{X}^k = \left[\Delta \mathbf{x}^{k-m} \cdots \Delta \mathbf{x}^{k-1}\right]$, $\mathscr{R}^k = \left[\Delta \mathcal{R}^{k-m} \cdots \Delta \mathcal{R}^{k-1}\right]$, and $\gamma^k = \left[\gamma_{k-m}^k \cdots \gamma_{k-1}^k\right]^{\mathrm{T}}$. Expressing the equations in the form $\bar{\mathbf{x}}^k = \sum_{j=k-m}^k w_j \mathbf{x}^j$ and $\bar{\mathcal{R}}^k = \sum_{j=k-m}^k w_j \mathcal{R}^j$, it is found that $\sum_{j=k-m}^k w_j = 1$. In other words, $\bar{\mathbf{x}}_k$ and $\bar{\mathcal{R}}_k$ are weighted averages of $\mathbf{x}_{k-m}, \ldots, \mathbf{x}_k$ and $\mathcal{R}^{k-m}, \ldots, \mathcal{R}^k$, respectively. The arguments $\gamma^k = \left[\gamma_{k-m}^k \cdots \gamma_{k-1}^k\right]^{\mathrm{T}}$ are determined by minimizing

$$E\left(\gamma^k\right) = \left\langle \bar{\mathcal{R}}^k, \bar{\mathcal{R}}^k \right\rangle = \left\| \mathcal{R}^k - \mathscr{R}^k \gamma^k \right\|_2^2 \tag{6.79}$$

whose solution can, but should not in practice, be obtained by solving the normal equations

$$\left(\mathscr{R}^{k\mathrm{T}} \mathscr{R}^k\right) \gamma^k = \mathscr{R}^{k\mathrm{T}} \mathcal{R}^k . \tag{6.80}$$

Combining Equations (6.77), (6.78), and (6.80), one obtains

$$\mathbf{x}^{k+1} = \bar{\mathbf{x}}^k + \beta \bar{\mathcal{R}}^k \tag{6.81}$$

$$= \mathbf{x}^k + \beta \mathcal{R}^k - \left(\mathscr{X}^k + \beta \mathscr{R}^k\right) \gamma^k \tag{6.82}$$

$$= \mathbf{x}^k + \beta \mathcal{R}^k - \left(\mathscr{X}^k + \beta \mathscr{R}^k\right) \left(\mathscr{R}^{k\mathrm{T}} \mathscr{R}^k\right)^{-1} \mathscr{R}^{k\mathrm{T}} \mathcal{R}^k \tag{6.83}$$

where β is the preset mixing parameter and $\mathscr{R}^{k^{\mathrm{T}}}\mathscr{R}^k$ is assumed to be nonsingular. In particular, if no previous iterate is taken into account (i.e. $m = 0$), then Equation (6.83) reads

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \beta\mathcal{R}^k \tag{6.84}$$

This scheme is referred to as simple mixing and underrelaxation if $0 < \beta < 1$ (see Section 6.3.3). The update formula (6.83) is the same as (6.74) by setting $G_{\mathcal{R}}^{k-m} = -\beta\mathbf{I}$. In this respect Anderson mixing implicitly forms an approximate inverse Jacobian $G_{\mathcal{R}}^k$ that minimizes $\left\|G_{\mathcal{R}}^k + \beta\mathbf{I}\right\|_F$ subject to (6.68). In the context of mixing, generalized Broyden's second method is equivalent to Anderson mixing. Note that if $\mathscr{R}^k$ is square and nonsingular, then Equation (6.83) matches the formula of the standard secant method.

6.4.1.3 Generalized Broyden's family

Now we can write down the generalized Broyden family, in which an update algorithm is in the form

$$G_{\mathcal{R}}^k = G_{\mathcal{R}}^{k-m} + \left(\mathscr{X}^k - G_{\mathcal{R}}^{k-m}\mathscr{R}^k\right)V^{k^{\mathrm{T}}} \tag{6.85}$$

where $V^{k^{\mathrm{T}}}\mathscr{R}^k = I$ so that the secant condition $G_{\mathcal{R}}^k\mathscr{R}^k = \mathscr{X}^k$ holds. The two optimal choices of $V^{k^T} = M^{k^{-1}}N^{k^T}$ are

$$M^k = \mathscr{R}^{k^T}\mathscr{R}^k, \quad N^{k^T} = \mathscr{R}^{k^T}, \tag{6.86}$$

minimizaing $\left\|G_{\mathcal{R}}^k - G_{\mathcal{R}}^{k-m}\right\|_F$ and

$$M^k = \mathscr{X}^{k^T}G_{\mathcal{R}}^k\mathscr{R}^k, \quad N^{k^T} = \mathscr{X}^{k^T}G_{\mathcal{R}}^k, \tag{6.87}$$

minimizaing $\left\|J_{\mathcal{R}}^k - J_{\mathcal{R}}^{k-m}\right\|_F$. This last choice yields as the approximation for the Jacobian

$$J_{\mathcal{R}}^k = J_{\mathcal{R}}^{k-m} + \left(\mathscr{R}^k - J_{\mathcal{R}}^{k-m}\mathscr{X}^k\right)\left(\mathscr{X}^{k^{\mathrm{T}}}\mathscr{X}^k\right)^{-1}\mathscr{X}^{k^{\mathrm{T}}}, \tag{6.88}$$

after applying the Woodbury formula. The first choice is said to correspond to a Type-II update and the second to a Type-I update (Fang and Saad, 2009).

6.4.1.4 Anderson's family

The udpate formula for Anderson's family can be found from Equation (6.85) using as the approximation to the previous Jacobian the identity matrix multiplied by a constant β, i.e.,

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \beta\mathcal{R}^k - (\mathscr{X}^k + \beta\mathscr{R}^k)\mathbf{V}^{k^T}\mathcal{R}^k. \tag{6.89}$$

The two choices for $\mathbf{V}^k$ remain the same, replacing $G_{\mathcal{R}}^{k-m}$ by $-\beta\mathbf{I}$. They now minimize $\|G_{\mathcal{R}}^k + \beta\mathbf{I}\|$ and $\|J_{\mathcal{R}}^k + (1/\beta)\mathbf{I}\|$.

6.4.1.5 The Broyden-like class

Both the generalized Broyden's family and Anderson's family can be understood as methods in the Broyden-like class as described in Fang and Saad (2009). Ssuppose the latest m iterates are available, which are denoted by $\mathbf{x}^1,\dots,\mathbf{x}^m$. Let $\Delta\mathbf{x}^i = \mathbf{x}^{i+1} - \mathbf{x}^i$ for $i = 1,\dots,m-1$. Partition $\Delta\mathbf{x}^1,\dots,\Delta\mathbf{x}^{m-1}$ into p groups,

$$\mathscr{X}^1 = \left[\Delta\mathbf{x}^1,\dots,\Delta\mathbf{x}^{z_1}\right], \tag{6.90}$$

$$\mathscr{X}^2 = \left[\Delta\mathbf{x}^{z_1+1},\dots,\Delta\mathbf{x}^{z_2}\right], \tag{6.91}$$

$$\vdots \tag{6.92}$$

$$\mathscr{X}^p = \left[\Delta\mathbf{x}^{z_{k-1}+1},\dots,\Delta\mathbf{x}^{z_k}\right], \tag{6.93}$$

where z_i is the index of the last entry in the i th group for $i = 1,\dots,p$; $z_0 = 0$ and $z_p = m-1$. Also partition $\Delta\mathcal{R}^1,\dots,\Delta\mathcal{R}^{m-1}$ into $\mathscr{R}^1,\dots,\mathscr{R}^p$ accordingly, where $\Delta\mathcal{R}^i = \mathcal{R}^{i+1} - \mathcal{R}^i$ with $\mathcal{R}^i = \mathcal{R}(\mathbf{x}^i)$. The sizes of the groups for $i = 1,\dots,k$ are denote by $s_i := z_i - z_{i-1}$. Note that the indexing here is different from the previous sections. The inverse of the Jacobian is iteratively approximated at the (z_i+1)st iterate for $i = 1,\dots,p$ as

$$G_{\mathcal{R}}^{i+1} = G_{\mathcal{R}}^i + \left(\mathscr{X}^i - G_{\mathcal{R}}^i\mathscr{R}^i\right)\mathbf{V}^{i^T}, \tag{6.94}$$

where $\mathbf{V}^{i^T}\mathscr{R}^i = \mathbf{I}$ for the secant condition. The update follows the formula of the generalized Broyden family. In the context of mixing, the base case is

$$G_{\mathcal{R}}^1 = -\beta\mathbf{I}, \tag{6.95}$$

where β is the mixing parameter. The next iterate is set as

$$\mathbf{x}^{m+1} = \mathbf{x}^m - G_{\mathcal{R}}^{k+1}\mathcal{R}^m. \tag{6.96}$$

The choice of V_i satisfying $V_i^{\mathrm{T}}\mathscr{F}_i = I$ is performed as described in Section 6.4.1.3.

6.4.2 Practical considerations

The application of Broyden's method as described so far is unfeasible for the problems considered in this document, i.e., thermo-mechanical coupled problems with many unknowns. So far, the descriptions considered of the Broyden-like class methods require one to keep the large $G_{\mathcal{R}}^i$ matrices of size $n_{\text{unknowns}} \times n_{\text{unknowns}}$ in memory, in addition to the previous iterates. This is a significant drawback. However, Fang and Saad (2009) present a more memory efficient way of implementing these methods. Let

$$E^i = \mathscr{X}^i - G_{\mathcal{R}}^i\mathscr{R}^i. \tag{6.97}$$

Substituting Equation (6.97) into Equation (6.94) one obtains

$$G_{\mathcal{R}}^i = G_{\mathcal{R}}^{i-1} + E^{i-1}V^{i-1^T}, \tag{6.98}$$

$$= G_{\mathcal{R}}^{i-2} + E^{i-2}V^{i-2^T} + E^{i-1}V^{i-1^T}, \tag{6.99}$$

$$\vdots \tag{6.100}$$

$$= G_{\mathcal{R}}^1 + \sum_{j=1}^{i-1} E^j V^{j^{\mathrm{T}}}, \tag{6.101}$$

for $i = 2,\dots,p+1$. Matrices $G^i_{\mathcal{R}}$ need not be explicitly stored. $G^i_{\mathcal{R}}$ is needed only to compute $G^i_{\mathcal{R}}\mathscr{R}^i$ in Equation (6.97) and $G^{p+1}_{\mathcal{R}}\mathcal{R}^m$ in Equation (6.96), and also for V^i if it depends on $G^i_{\mathcal{R}}$. Substituting Equation (6.101) into Equation (6.97) obtains

$$E^i = \mathscr{X}^i - G^1_{\mathcal{R}}\mathscr{R}^i - \sum_{j=1}^{i-1} E^j\left({V^j}^T\mathscr{R}^i\right), \qquad (6.102)$$

for $i = 1,\dots,p$. The computation is economic for large-scale problems with $n \gg m$. The next iterate $\mathbf{x}^{m+1}$ in (32) can also be computed in a similar manner

$$\mathbf{x}^{m+1} = \mathbf{x}^m - G^{p+1}_{\mathcal{R}}\mathcal{R}^m = \mathbf{x}^m - G^1_{\mathcal{R}}\mathcal{R}^m - \sum_{j=1}^{k} E^j\left({V^j}^T\mathcal{R}^m\right). \qquad (6.103)$$

Using Type-II update, the computation of V^i is straightforward from $\mathscr{R}^i$. On the other hand, Type-I update involves $G^i_{\mathcal{R}}$ to compute V^i. Thus

$${N^i}^T = {\mathscr{X}^i}^T G^i_{\mathcal{R}} = G^1_{\mathcal{R}}{\mathscr{X}^i}^T + \sum_{j=1}^{i-1}\left({\mathscr{X}^i}^{\mathrm{T}}E^j\right){V^j}^T. \qquad (6.104)$$

After obtaining N^i, we compute $M^i = {N^i}^T\mathscr{R}^i$ and then ${V^i}^T = {M^i}^{-1}{N^i}^T$ for $i = 1,\dots,p$.

Looking at Equations (6.102), (6.103) and (6.104), one still needs the initial approximation to the inverse of the Jacobian, $G^1_{\mathcal{R}}$, whose size is $n_{\text{unknown}} \times n_{\text{unknown}}$. In Fang and Saad (2009) the approach adopted was to follow the idea of Anderson's mixing and set

$$G^1_{\mathcal{R}} = -\beta\mathbf{I}, \qquad (6.105)$$

drastically improving the memory requirements, as only one scalar parameter, β, needs to be saved. Also, Kelley (2003), assumes in his implementation of Broyden's method, an initial approximation to $G^1_{\mathcal{R}}$ equal to the identity matrix. Information about $G^1_{\mathcal{R}}$ is applied in the preconditioning of the system instead.

To compute V^i, Fang and Saad (2009) suggest a QR decomposition with pivoting. Be it for a Type-II update, where one needs to solve a least-squares problem, or for a Type-I update, one needs to invert a generally non-symmetric matrix. This approach leads to better numerical stability when the matrices to be inverted are singular or ill-conditioned, compared with solving the normal equations. The QR decomposition has a computational cost of $\mathcal{O}(n^3_{\text{unknown}})$ algebraic operations (Dennis and Schnabel, 1996).

If the size of the groups $s_1, s_2, \dots$ are fixed from one Newton iteration to the next, so the E^i and V^i matrices remain the same from one iteration to the next, the computation effort to compute them is saved from one iteration to the next. Here only constant $s = s_1 = s_2 = \cdots$ is considered, where $s = \infty$ corresponds to Anderson's mixing, where all previous iterates available are considered.

One question remains. How to proceed when the available memory runs out?. According to Kelley (2003), as is often the case with GMRES, the iteration can be restarted if there is no more room to store the vectors. A different approach, called limited memory in the optimization literature, is to replace the oldest of the stored steps with the most recent one.

Thus, appplying the method as suggest by Fang and Saad (2009), leads to a storage need of

1. Two column vectors of size n_{unknown} for $\mathbf{x}^m$ and $\mathcal{R}^m$.
2. An $n_{\text{unknown}} \times (m-1)$ matrix for $\mathscr{X}^1, \dots, \mathscr{X}^k$ (shared with $E^1, \dots, E^k$).
3. An $n_{\text{unknown}} \times (m-1)$ matrix for $\mathscr{R}^1, \dots, \mathscr{R}^k$ (shared with $V^1, \dots, V^k$).
4. For Type-I update we also store the last group N^k, since its computation involves $G^k_{\mathcal{R}}$.

and for each nonlinear iteration the computational cost is $\mathcal{O}(n^3_{\text{unknown}})$. For $m = 1$, V^i can be directly computed without needing to invert matrices, and the cost comes down to $\mathcal{O}(n_{\text{unknown}})$ per nonlinear iteration. Also, when k is different from any z_i a group is being complete, one can either use a shortened group or reuse the approximation to the Jacobian without using the new information. This save computational effort and those iteration cost only $\mathcal{O}(n_{\text{unknown}})$. See in Box 6.5 the pseudocode for the Broyden-like family with restart.

Kelley (2003) presents for the Broyden method an implementation that halves the memory requirement relative to the one present in Fang and Saad (2009). It is based on the Type-I update and to deduce it consider Equation (6.61) and Sherman-Morrison formula

$$\left(J_{\mathcal{R}} + \mathbf{u}\mathbf{v}^T\right)^{-1} = \left(\mathbf{I} - \frac{(G_{\mathcal{R}}\mathbf{u})\,\mathbf{v}^T}{1 + \mathbf{v}^T G_{\mathcal{R}}\mathbf{u}}\right) G_{\mathcal{R}}, \tag{6.106}$$

where as before $G_{\mathcal{R}} \equiv J^{-1}_{\mathcal{R}}$. One can rewrite Equation (6.61) as

$$J^{k+1}_{\mathcal{R}} = J^k_{\mathcal{R}} + \mathbf{u}^k {\mathbf{v}^k}^T, \tag{6.107}$$

where

$$\mathbf{u}^k = \left(\Delta\mathcal{R}^k - J^k_{\mathcal{R}}\Delta\mathbf{x}^k\right) / \left\|\Delta\mathbf{x}^k\right\| \text{ and } \mathbf{v}^k = \Delta\mathbf{x}^k / \left\|\Delta\mathbf{x}^k\right\|. \tag{6.108}$$

Then, keeping in mind that $J^1_{\mathcal{R}} = \mathbf{I}$,

$$G^{k+1}_{\mathcal{R}} = \left(\mathbf{I} - \mathbf{w}^k {\mathbf{v}^k}^T\right)\left(\mathbf{I} - \mathbf{w}_{k-1} {\mathbf{v}^{k-1}}^T\right)\cdots\left(\mathbf{I} - \mathbf{w}^1 {\mathbf{v}^1}^T\right) G^1_{\mathcal{R}}, \tag{6.109}$$

$$= \prod_{j=0}^{k}\left(\mathbf{I} - \mathbf{w}^j {\mathbf{v}^j}^T\right), \tag{6.110}$$

where, for $k \geq 0$,

$$\mathbf{w}^k = \frac{G^k_{\mathcal{R}}\mathbf{u}^k}{1 + {\mathbf{v}^k}^T G^k_{\mathcal{R}}\mathbf{u}^k}. \tag{6.111}$$

So, to apply $G^{k+1}_{\mathcal{R}}$ to a vector $\mathbf{p}$, the is cost of $O(n_{\text{unknown}}k)$ floating point operations and storage of the $2k$ vectors $\left\{\mathbf{w}^j\right\}_{j=1}^k$ and $\left\{\Delta\mathbf{x}^j\right\}_{j=1}^k$. The storage can be halved with a trick (see Kelley (2003) for details)

$$\Delta\mathbf{x}^k = -G^{k+1}_{\mathcal{R}}\mathcal{R}^k, \tag{6.112}$$

$$= -\left(\mathbf{I} - \frac{\mathbf{w}^k {\Delta\mathbf{x}^k}^T}{\left\|\Delta\mathbf{x}^k\right\|}\right) G^k_{\mathcal{R}}\mathcal{R}^k, \tag{6.113}$$

$$= -\frac{G^k_{\mathcal{R}} R^k}{1 + {\Delta\mathbf{x}^k}^T G^k_{\mathcal{R}}\mathcal{R}^k / \|\Delta\mathbf{x}^k\|^2}. \tag{6.114}$$

According to Kelley (2003), the Sherman-Morrison approach is more efficient, in terms of both time and storage, than dense matrix approaches proposed elsewhere. For example, the approach presented in Dennis and Schnabel (1996) has a $\mathcal{O}(n_{\text{unknown}})$ cost per nonlinear iteration and requires one to keep in memory the QR decomposition of the previous approximation to the Jacobian. However, the dense matrix approach can detect ill-conditioning in the approximate Jacobians. Bounded deterioration implies that the Broyden matrices will be well-conditioned if the data is sufficiently good, and superlinear convergence suggests that only a few iterates will be needed.

In the context of FSI The multi-secant quasi-Newton methods have been used in the context of FSI, although not always presented as such (Haelterman et al., 2009; Gatzhammer, 2014; Uekermann, 2016; Scheufele, 2018). Vierendeels et al. (2007) and Degroote et al. (2008) consider the system of equations (6.45) and (6.46), where recall that an estimate for the Jacobians $J_{\mathcal{U}}$ and $J_{\mathcal{T}}$ are needed. The authors achieve this by using linear reduced-order models for the fluid solver and the structure solver. These are set up from solver input and output deltas or sensitivities during the coupling iterations. The resulting method for two black-box solvers is called interface block quasi-Newton method with least-squares approximation (IBQN-LS) in Degroote (2010).

This approach can be understood in the framework of the multi-secant quasi-Newton methods presented above and originating in Fang and Saad (2009) as follows. If one looks at $\beta\mathbf{I} - (\mathscr{X}^k + \beta\mathscr{R}^k)\mathbf{V}^{k^T}$ in Equation (6.89) as, in a sense, an approximation to the inverse of the Jacobian (compare with Equation (6.85)). The corresponding Jacobian is given by

$$J^k_{\mathcal{R}} = \alpha\mathbf{I} + (\mathscr{R}^k - \alpha\mathbf{I}\mathscr{X}^k)(\mathscr{X}^{k^T}\mathscr{X}^k)^{-1}\mathscr{X}^{k^T}, \tag{6.115}$$

where $\alpha = 1/\beta$. If one sets $\alpha = 0$, the approximation to the Jacobian obtained is

$$J^k_{\mathcal{R}} = \mathscr{R}^k(\mathscr{X}^{k^T}\mathscr{X}^k)^{-1}\mathscr{X}^{k^T}. \tag{6.116}$$

This corresponds to the linear reduced order models in Vierendeels et al. (2007), where $\mathcal{R}$ is replaced by the functions corresponding to the fluid and structure solvers. If the functions considered are instead the mechanical and thermal solvers, this method can easily be applied to the thermomechanical problem. The block $(\mathscr{X}^{k^T}\mathscr{X}^k)^{-1}\mathscr{X}^{k^T}$ can be understood as being a part of a least-squares solution, the so-called normal equations, i.e., the equations whose solution also solve the minimization problem

$$\arg\min_{\tilde{\gamma}} \|\Delta\mathbf{x} - \mathscr{X}^k\tilde{\gamma}\|_2. \tag{6.117}$$

As such one can avoid the use of the normal equations and employ more numerically stable and efficient methods such economy size QR-decomposition. In addition, Vierendeels et al. (2007) solve the system of equation (6.45) and (6.46) in a Gauss-Seidel manner, using always the most recent values available to estimate the Jacobians.

An interface quasi-Newton method based on Equation (6.38) and Equation (6.39) is presented in Degroote (2010) for FSI. The method is called interface quasi-Newton with an approximation of the inverse of the interface Jacobian matrix by least squares (QIN-ILS). Its origin is the IBQN-LS method presented in Vierendeels et al. (2007), and it employs only one reduced-order model for the inverse of the overall interface Jacobian matrix of the Newton system (Equation (6.38)) applied to the right-hand side vector.

If in Equation (6.83), corresponding to Anderson's mixing, one sets $\beta = -1$, the update formula comes out to be

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \mathcal{R}^k - (\mathscr{X}^k - \mathscr{R}^k)\left(\mathscr{R}^{k^{\mathrm{T}}}\mathscr{R}^k\right)^{-1}\mathscr{R}^{k^{\mathrm{T}}}\mathcal{R}^k. \tag{6.118}$$

Using the definition for the fixed-point function $\mathcal{S}$, one finds

$$\mathbf{x}^{k+1} = \mathcal{S}^k - \mathscr{S}^k\left(\mathscr{R}^{k^{\mathrm{T}}}\mathscr{R}^k\right)^{-1}\mathscr{R}^{k^{\mathrm{T}}}\mathcal{R}^k, \tag{6.119}$$

or

$$\Delta\mathbf{x}^k = -\mathcal{R}^k - \mathscr{S}^k\left(\mathscr{R}^{k^{\mathrm{T}}}\mathscr{R}^k\right)^{-1}\mathscr{R}^{k^{\mathrm{T}}}\mathcal{R}^k, \tag{6.120}$$

When no delta columns are available yet, constant relaxation is used once to ensure stability.

Box 6.5: Broyden-like class methods for one timestep with restart.

(i) Set $\mathbf{x}^1 = \mathbf{x}^p_{n+1}$.

(ii) Evaluate $\mathcal{R}^1 = \mathcal{R}(\mathbf{x}^1)$, which implies the solution of the mechanical and the thermal problems, $\mathcal{U}$ and $\mathcal{T}$, respectively.

(iii) Compute $\mathbf{x}^2 = \mathbf{x}^1 + \beta R^1$.

(iv) Evaluate $\mathcal{R}^2 = \mathcal{R}(\mathbf{x}^2)$, which implies the solution of the mechanical and the thermal problems, $\mathcal{U}$ and $\mathcal{T}$, respectively.

(v) Intialize counters: $k = 2$ and $i = 1$

(vi) Enter the nonlinear loop

  (1) If the maximum number of previous iteration m has been reached restart all $\mathscr{X}^i$ and $\mathscr{R}^i$ and set $i = 1$.

  (2) Compute $\Delta\mathcal{R}^{k-1} = \mathcal{R}^k - \mathcal{R}^{k-1}$ and add it to $\mathscr{R}^i$.

  (3) Compute $\Delta\mathbf{x}^{k-1} = \mathbf{x}^k - \mathbf{x}^{k-1}$ and add it to $\mathscr{X}^i$.

  (4) If $s = \infty$ (Anderson mixng):

   - Compute ${V^i}^T$ according to the type of update chosen (Equation (6.87) and (6.86)).
   - Update according to $\mathbf{x}^{k+1} = \mathbf{x}^k + \beta\mathcal{R}^k - (\mathscr{X}^i + \beta\mathscr{R}^i){V^i}^T\mathcal{R}^k$.

  (5) else:

   - If $k = z_j + 1$ for any $j \geq 1$, compute E^i (Equation (6.102)) and ${V^i}^T$ according to the type of update chosen (Equation (6.87) and (6.86)). Save them on $\mathscr{X}^i$ and $\mathscr{R}^i$, respectively. Update $i = i + 1$.
   - Update according to $\mathbf{x}^{k+1} = \mathbf{x}^k + \beta\mathcal{R}^k - \sum_{j=1}^{i-1} E^j({V^j}^T\mathcal{R}^k)$.

  (6) Evaluate $\mathcal{R}^{k+1} = \mathcal{R}(\mathbf{x}^{k+1})$, which implies the solution of the mechanical and the thermal problems, $\mathcal{U}$ and $\mathcal{T}$, respectively.

  (7) If the desired accuracy has not been reached, update $k = k + 1$ and go to step (1).

6.5 Multipoint iteration functions

6.5.1 Finite-Difference Newton Method

This approach follows precisely the one described in Section 6.3.2 for the "standard" Newton method. The difference lies in the computation of the Jacobian. Here the Jacobian $J_{\mathcal{R}}(\mathbf{x})$ is approximated from a forward finite-difference, $J^h_{\mathcal{R}}(\mathbf{x})$, by columns. Following Kelley (2003). the jth column is

$$\left[J^h_{\mathcal{R}}(\mathbf{x})\right]_j = \begin{cases} \dfrac{\mathcal{R}(\mathbf{x}+h\sigma_j\mathbf{e}_j)-\mathcal{R}(\mathbf{x})}{\sigma_j h}, & x_j \neq 0 \\ \dfrac{\mathcal{R}(h\mathbf{e}_j)-\mathcal{R}(\mathbf{x})}{h}, & x_j = 0 \end{cases}, \tag{6.121}$$

where $\mathbf{e}_j$ is the unit vector in the j th coordinate direction. The difference increment h should be no smaller than the square root of the inaccuracy in $\mathcal{R}$ (Kelley, 2003). It should, however, be scaled. Rather than simply perturbing $\mathbf{x}$ by a difference increment h, roughly the square root of the error in $\mathcal{R}$, in each coordinate direction, the perturbation is multiplied to compute the j th column by

$$\sigma_j = \max(|(x)_j|, 1)\,\mathrm{sign}((x)_j), \tag{6.122}$$

with a view toward varying the correct fraction of the low-order bits in $(x)_j$. The sign function is defined as

$$\mathrm{sgn}(z) = \begin{cases} z/|z| & \text{if } z \neq 0 \\ 1 & \text{if } z = 0 \end{cases}. \tag{6.123}$$

While this scaling usually makes little difference, it can be crucial if $|(x)_j|$ is very large. Note that there is no adjustment if $|(x)_j|$ is very small because the error determines the lower limit on the size of the difference increment in $\mathcal{R}$. For example, if evaluations of $\mathcal{R}$ are accurate to 16 decimal digits, the difference increment should change roughly the last eight digits of x. (Kelley, 2003)

Each column of $J^h_{\mathcal{R}}(\mathbf{x})$ requires one new function evaluation and, therefore, a finite difference Jacobian costs n_{unknown} function evaluations. If the perturbation is appropriately chosen, the method converges quadratically when the function satisfies certain conditions, and the initial attempt is close enough to the solution (Dennis and Schnabel, 1996).

The Chord and Shamanskii Methods If the computational cost of a forward difference Jacobian is high, i.e., $\mathcal{R}$ is expensive and/, or n_{unknown} is significant. If an analytic Jacobian is not available, it is wise to amortize this cost over several nonlinear iterations. The chord method does precisely that. It differs from Newton's method in that the evaluation and factorization of the Jacobian are done only once for $J_{\mathcal{R}}(\mathbf{x}^0)$. The advantages of the chord method increase as n increases, since both the n function evaluations and the $O(n^3_{\text{unknown}})$ work (in the dense matrix case) in the matrix factorization are done only once. So, while the convergence is q-linear and more nonlinear iterations will be needed than for Newton's method, the overall cost of the solution will usually be much less. A middle ground is the Shamanskii method. Here the Jacobian factorization and matrix function evaluation is done after every m computations of the step (Kelley, 2003).

Since the present use-case, the number of unknowns n is very large, and the evaluation of the function $\mathcal{R}$ is also costly, approximating the Jacobian using a finite difference is not suitable, even utilizing the chord or Shamanskii methods.

6.5.2 Newton-Krylov methods

In the Newton-Krylov methods, the solution of the Newton system of equations in Equation (6.38) is achieved using Krylov methods, such as GMRES or BiCGSTAB. The Krylov iterative methods approximate the solution of a linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ using the Krylov subspace

$$\mathcal{K}_m = \operatorname{span}\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, \ldots, \mathbf{A}^{m-1}\mathbf{r}_0\}, \tag{6.124}$$

such that the mth iterate, $\mathbf{x}_m \in \mathcal{K}_m$, with $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$. The precise way the $\mathbf{x}_m$ is built is what distinguishes the different methods.

To produce the appropriate Krylov subspace, one needs the product $J_{\mathcal{R}}(\mathbf{x}^k)\mathbf{y}$ in Equation (6.38), for some vector $\mathbf{y}$. It is assumed that the Jacobian is not available, so it must be approximated. Also, it would be beneficial if the full Jacobian is neither computed in its entirety nor wholly stored in memory, i.e., a matrix-free method is desirable. As in Section 6.5.1, the Jacobian-vector product is easy to approximate with a forward difference directional derivative (Kelley, 2003). The forward difference directional derivative at $\mathbf{x}^k$ in the direction $\mathbf{q}$ is

$$J^h_{\mathcal{R}}(\mathbf{x}^k)\mathbf{q} = \begin{cases} \mathbf{0}, & \mathbf{q} = \mathbf{0} \\ \|\mathbf{q}\| \dfrac{\mathcal{R}(\mathbf{x}^k + \sigma(\mathbf{x}^k, \mathbf{q})h\mathbf{q}/\|\mathbf{q}\|) - \mathcal{R}(\mathbf{x}^k)}{\sigma(\mathbf{x}^k, \mathbf{q})h}, & \mathbf{q} \neq \mathbf{0}. \end{cases} \tag{6.125}$$

The scaling is important. $\mathbf{q}$ is scaled to be a unit vector and take a numerical directional derivative in the direction $\mathbf{q}/\|\mathbf{q}\|$. If h is roughly the square root of the error in $\mathcal{R}$, a difference increment in the forward difference is used to make sure that the appropriate low-order bits of $\mathbf{x}^k$ is perturbed. So h is multiplied by

$$\sigma(\mathbf{x}^k, \mathbf{q}) = \max(|\mathbf{x}^{k^T}\mathbf{q}|, \|\mathbf{q}\|)\operatorname{sign}(\mathbf{x}^{k^T}\mathbf{q})/\|\mathbf{q}\|. \tag{6.126}$$

Sidi (2017) describes two different Newton-Krylov methods, the Newton-Arnoldi and the Newton-GMRES. The goal is to solve Equation (6.38) and find $\Delta\mathbf{x}^k$. The Krylov iterates are denoted by $\Delta\mathbf{x}^*_m$.

The first phase of both algorithms is identical. They both use the Arnoldi-Gram-Schmidt process to produce an orthogonal basis for the Krylov subspace, $\{\mathbf{q}_1, \ldots, \mathbf{q}_m\}$ for some integer k. Consider the unitary matrix $\mathbf{Q}_m$

$$\mathbf{Q}_m = [\mathbf{q}_1 \cdots \mathbf{q}_m], \tag{6.127}$$

and the upper Hessenber matrix $\mathbf{H}_m$ and related matrix $\bar{\mathbf{H}}_m$

$$\mathbf{H}_m = \begin{bmatrix} h_{11} & h_{12} & \cdots & \cdots & h_{1m} \\ h_{21} & h_{22} & \cdots & \cdots & h_{2m} \\ & \ddots & \ddots & & \vdots \\ & & \ddots & \ddots & \vdots \\ & & & h_{m,m-1} & h_{mm} \end{bmatrix}, \quad \bar{\mathbf{H}}_m = \begin{bmatrix} h_{11} & h_{12} & \cdots & \cdots & h_{1m} \\ h_{21} & h_{22} & \cdots & \cdots & h_{2m} \\ & \ddots & \ddots & & \vdots \\ & & \ddots & \ddots & \vdots \\ & & & \ddots & h_{mm} \\ & & & & b_{m+1,m} \end{bmatrix}, \tag{6.128}$$

where the quantities h_{ij} are also obtained from the algorithm. $\mathbf{H}_m$ can be interpreted as the projection of $\mathbf{A}$ onto the Krylov subspace since $\mathbf{H}_m = \mathbf{Q}_m^T J_{\mathcal{R}}(\mathbf{x}^k)\mathbf{Q}_m$. See Box 6.6 for the pseudo-code of the Arnoldi-Gram-Schmidt process.

Box 6.6: Arnoldi process to orthonormalize the Krylov subspace

```
(i) Compute $\mathbf{r}_0 = -\mathcal{R}(\mathbf{x}^k)$ and set $\beta = \|\mathbf{r}_0\|$ and $\mathbf{q}_1 = \mathbf{r}_0/\beta$.
(ii) $j = 1$
(iii) $\mathbf{a}_{j+1}^{(1)} = J_{\mathcal{R}}(\mathbf{x}^k)\mathbf{q}_j$.
(iv) Compute $h_{ij} = (\mathbf{q}_i, \mathbf{a}_{j+1}^{(i)})$ and compute $\mathbf{a}_{j+1}^{(i+1)} = \mathbf{a}_{j+1}^{(i)} - h_{ij}\mathbf{q}_i$ for $i = 1, \ldots, j$.
(v) Compute $h_{j+1,j} = \|\mathbf{a}^{(j+1)})_{j+1}\|$ and set $\mathbf{q}_{j+1} = \mathbf{a}_{j+1}^{(j+1)}/h_{j+1,j}$.
(vi) $j = j + 1$
(vii) If $j < m - 1$ go to Step (iii)
```

After obtaining an orthogonal basis for the Krylov subspace $\mathcal{K}_m$, the Newton-Arnoldi projects the mth residual onto the Krylov subspace and sets it to zero, similar to a weigthed residual method, i.e.,

$$\mathbf{Q}_m \left(\mathcal{R}(\mathbf{x}^k) - J_{\mathcal{R}}(\mathbf{x}^k)\Delta\mathbf{x}_m^* \right) = \mathbf{0}. \tag{6.129}$$

In practice, one solves the equivalent linear system $\mathbf{H}_m\boldsymbol{\eta} = \beta\mathbf{e}_1$ for $\boldsymbol{\eta}$ and set $\Delta\mathbf{x}^k = \Delta\mathbf{x}_0^* + \mathbf{Q}_m\boldsymbol{\eta}$.

The Newton-GMRES attempts to minimize the norm of the mth residual, i.e.,

$$\Delta\mathbf{x}_m^* = \arg\min_{\mathbf{y}\in\mathcal{K}_m} \left\| \mathcal{R}(\mathbf{x}^k) - J_{\mathcal{R}}(\mathbf{x}^k)\mathbf{y} \right\|_2. \tag{6.130}$$

This is exectued in practice solving the linear least-squares problem $\|\beta\mathbf{e}_1 - \bar{\mathbf{H}}_m\boldsymbol{\eta}\|$ for $\boldsymbol{\eta}$ and set $\Delta\mathbf{x}^k = \Delta\mathbf{x}_0^* + \mathbf{Q}_m\boldsymbol{\eta}$.

Since the present use case includes many unknowns, it leads to memory concerns if the Krylov subspace is allowed to grow indefinitely. A restarted version where the maximum size of the Krylov space is restricted to m elements is preferred. Once this number is reached, the procedure is restarted. However, if m is small, the convergence can be poor.

In each nonlinear iteration of the Newton-Krylov, the number of iterations can be large, and each iteration requires an evaluation of the function. This can be a significant drawback when the intended use assumes that evaluating the function $\mathcal{R}$ is expensive. This problem is however mitigated by the fact the Newton system is only solved until it satisfies

$$\|J_{\mathcal{R}}(\mathbf{x}^k)\Delta x_m^* + \mathcal{R}(\mathbf{x}^k)\| \leq \eta\|\mathcal{R}(\mathbf{x}^k)\|, \tag{6.131}$$

where η is called the forcing term and it is chosen to avoid oversolving the Newton system (Equation (6.38). As a simple approach, Kelley (2003) suggests $\eta = 0.1$. However, he describes more sophisticated ways to choose this parameter. The smaller the forcing term η, the closer one gets to the "standard" Newton method. However, especially in the first nonlinear iterations, choosing a η that is too small leads to unnecessarily long computational times. The linear system is being solved with too much precision. See Box 6.7 for the pseudocode of the Newton-Arnoldi and Newton-GMRES.

Box 6.7: Timestep n of the Newton-Krylov methods with restart, Newton-Arnoldi and Newton-GMRES.

```
(i) $k = 0$
(ii) Enter the Newton loop
    (1) Compute $\mathbf{r}_0 = -\mathcal{R}(\mathbf{x}^k)$
    (2) Enter the Krylov loop
        (a) Set $\beta = \|\mathbf{r}_0\|$ and $\mathbf{q}_1 = \mathbf{r}_0/\beta$.
        (b) $j = 1$
        (c) $\mathbf{a}_{j+1}^{(1)} = J_{\mathcal{R}}(\mathbf{x}^k)\mathbf{q}_j$.
        (d) Compute $h_{ij} = (\mathbf{q}_i, \mathbf{a}_{j+1}^{(i)})$ and compute $\mathbf{a}_{j+1}^{(i+1)} = \mathbf{a}_{j+1}^{(i)} - h_{ij}\mathbf{q}_i$ for $i = 1, \dots, j$.
        (e) Compute $h_{j+1,j} = \|\mathbf{a}^{(j+1)})_{j+1}\|$ and set $\mathbf{q}_{j+1} = \mathbf{a}_{j+1}^{(j+1)}/h_{j+1,j}$.
        (f) If using the Arnoldi method:
            • Solve the linear system $\mathbf{H}_j\boldsymbol{\eta} = \beta\mathbf{e}_1$ for $\boldsymbol{\eta}$
            • set $\Delta\mathbf{x}_j^* = \Delta\mathbf{x}_0^* + \mathbf{Q}_j\boldsymbol{\eta}$.
        (g) Else if using the GMRES method:
            • Solve the linear least-squares problem $\|\beta\mathbf{e}_1 - \bar{\mathbf{H}}_j\boldsymbol{\eta}\|$ for $\boldsymbol{\eta}$.
            • Set $\Delta\mathbf{x}_j^* = \Delta\mathbf{x}_0^* + \mathbf{Q}_j\boldsymbol{\eta}$.
        (h) If $\|J_{\mathcal{R}}(\mathbf{x}^k)\Delta x_j^* + \mathcal{R}(\mathbf{x}^k)\| \leq \eta\|\mathcal{R}(\mathbf{x}^k)\|$ is not satisfied set $j = j+1$.
        (i) if $j > m$, restart the method going to Step (a). Else, go to Step (c).
    (3) $\mathbf{x}^{k+1} = \mathbf{x}^k + \Delta x^k$.
    (4) $k = k+1$
    (5) $\mathbf{r}_0 = -\mathcal{R}(\mathbf{x}^k)$
    (6) If convergence has not been reached, $\|\mathbf{r}_0\| > \epsilon_1$, go to Step (2)
```

6.5.3 Extrapolation tecniques with cycling

There is a vast literature on sequence acceleration/extrapolation methods (see Brezinski and Zaglia (2013) and Sidi (2017) for textbook treatments of this topic). One often deals with sequences and series in numerical analysis, applied mathematics, and engineering. They are produced by iterative methods, perturbation techniques, and approximation procedures depending on a parameter. Those sequences or series often converge so slowly that it is a severe drawback to their practical use. Convergence acceleration methods present a solution and have been studied for many years and applied to various situations. They are based on the very natural idea of extrapolation. In many cases, they lead to the solution of unsolvable problems otherwise. Sequences of vectors can also be considered, with their dimension being substantial. Such sequences arise, for example, in the solution by fixed-point iterative methods of systems of linear or nonlinear algebraic equations.

An example of a scalar acceleration method is first presented to fix ideas. Let (S_n) be a sequence of numbers that converges to S. This sequence can be transformed into another, denoted (T_n). For example, consider

$$T_n = \frac{S_n S_{n+2} - S_{n+1}^2}{S_{n+2} - 2S_{n+1} + S_n}, \quad n = 0, 1, \dots, \tag{6.132}$$

which corresponds to the Aitken Δ^2 process.

This expression can be obtained considering a transformation that would yield the limit of a geometric sequence from only three iterates, i.e., if one fits an exponential function

$$S + a\lambda^n, \tag{6.133}$$

the sequence transformation takes the horizontal asymptote of the exponential, S. For the geometrical interpretation of Aitken's Δ^2 method, see Figure 6.4.

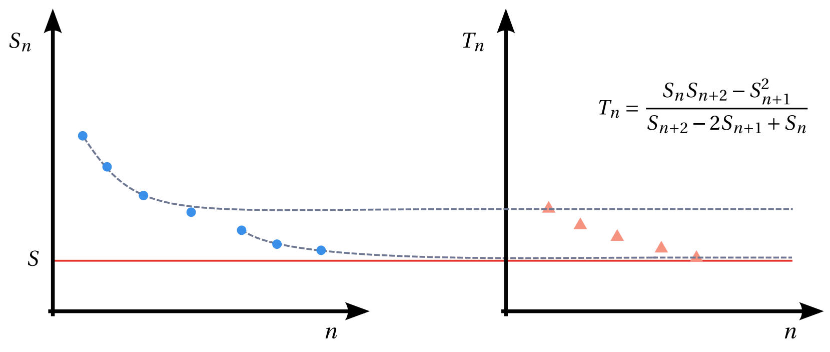
S_n
T_n
$T_n = \frac{S_n S_{n+2} - S_{n+1}^2}{S_{n+2} - 2S_{n+1} + S_n}$
S
n
n

Figure 6.4: Geometrical interpretation of Aitken's Δ^2 method.

One can also show that if (S_n) goes to its limit § at a rate strictly greater than 1[1], (T_n) does not have a better rate of convergence.

In practice, the sequence produced by Aitken's Δ^2 method tends to converge faster to the limit than (S_n) does. Very often, it is much cheaper to calculate (T_n), which involves only the calculation of differences, one multiplication, and one division, than to calculate many more terms of the sequence (S_n). Care must be taken, however, to avoid introducing errors due to insufficient precision when calculating the differences in the numerator and denominator of the expression.

There is, however, no universal sequence accelerator capable of accelerating all sequences. It is also the case that nonlinear transformations can even fail to converge or converge to a value other than the limit of the original sequence (Brezinski and Zaglia, 2013).

[1] (S_n), $n \in \mathbb{N}$ converges linearly to S if there exists a number $\mu \in (0,1)$ such that $\lim_{n\to\infty} \frac{|S_{n+1}-S|}{|S_n-S|} = \mu$.

According to Brezinski and Zaglia (2013), there is a very strong connection between sequence transformations and fixed point methods for solving $x = g(x)$, $g\colon \mathbb{R} \to \mathbb{R}$. The most well-known example of this connection is that between Aitken's Δ^2 process and Steffensen's method.

$$T_n = S_n - \frac{(S_{n+1} - S_n)^2}{S_{n+2} - 2S_{n+1} + S_n}, \quad n = 0, 1, \ldots \quad \text{for Aitken's process} \tag{6.134}$$

and

$$x_{n+1} = x_n - \frac{\left(g(x_n) - x_n\right)^2}{g\left(g(x_n)\right) - 2g(x_n) + x_n}, \quad n = 0, 1, \ldots \quad \text{for Steffensen's method.} \tag{6.135}$$

Turning to vector sequences and systems of nonlinear equations, let $F\colon (\mathbf{w}^k) \to (\mathbf{y}^k)$ be a vector extrapolation method defined by

$$\mathbf{y}^k = F\left(\mathbf{w}^k, \ldots, \mathbf{w}^{k+m}\right), \quad n = 0, 1, \ldots \tag{6.136}$$

For solving the fixed point problem $\mathbf{x} = \mathcal{S}(\mathbf{x})$ one can associate to it the iterative method

$$\mathbf{x}^{k+1} = F\left(\mathbf{x}^k, \mathcal{S}(\mathbf{x}^k), \ldots, \mathcal{S}^m(\mathbf{x}^k)\right), \quad n = 0, 1, \ldots \tag{6.137}$$

where $\mathcal{S}^{m+1}(\mathbf{x}) = \mathcal{S} \circ \mathcal{S}^m(\mathbf{x})$ and $\mathcal{S}^0(\mathbf{x}) = \mathbf{x}$. This approach is called full cycling or simply cycling. Conversely to any fixed point iteration of this form, one can associate a sequence transformation of the previous form. See Box 6.8 for the general algorithm, excluding the extrapolation method.

Box 6.8: Timestep n of vector extrapolation with cycling.

(i) Choose integers $n \geq 0$ and $k \leq 1$ and an initial vector $\mathbf{x}^k = \mathbf{x}_0^*$.

(ii) Compute $\mathbf{x}_1^*$, $\mathbf{x}_2^*$, $\ldots$, $\mathbf{x}_{n+k+1}^*$ via $\mathbf{x}_{m+1}^* = \mathcal{S}(\mathbf{x}_m^*)$.

(iii) Apply any of the four extrapolation methods, namely, MPE, RRE, MMPE, and SVD-MPE, to $\mathbf{x}_n^*$, $\mathbf{x}_{n+1}^*$, $\ldots$, $\mathbf{x}_{n+k+1}^*$, obtaining $\mathbf{s}_{k,n}$.

(iv) If $\mathbf{s}_{n,k}$ satisfies the accuracy test, stop.
Otherwise, set $\mathbf{x}_0^* = \mathbf{s}_{n,k}$ and go to Step (ii).

There is a variety of vector extrapolation methods, where the major two categories are polynomial methods and methods based on the ϵ-algorithm (Brezinski and Zaglia, 2013; Sidi, 2017). In this presentation, only the first category is considered since the second requires a relatively large number of function evaluations per iteration, making it unsuitable for the present use-case (Sidi, 2017).

Sidi (2017) presents four different polynomial extrapolation methods. They all attempt to exprress the limit of the vector sequence as a linear combination of the p previous iterates, as follows

$$\mathbf{s} \approx \mathbf{s}_{k,m} = \sum_{j=0}^{m} \gamma_j \mathbf{w}^{k+j}, \tag{6.138}$$

where $\mathbf{s}$ is the limit of the vector sequence. The methods to be presented next appear naturally when considering the vector sequence generated by

$$\mathbf{w}^{k+1} = \mathbf{T}\mathbf{w}^k + \mathbf{d}, \tag{6.139}$$

where $\mathbf{I} - \mathbf{T}$ is non-singular. It is tighly connected to the solution of linear systems of equations. Considering the minimal polynomial of $\mathbf{T}$ with respect to $\Delta\mathbf{w}^k = \mathbf{w}^{k+1} - \mathbf{w}^k$ and $\boldsymbol{\epsilon}^k = \mathbf{w}^k - \mathbf{s}$[2], $P(\lambda)$,

$$P(\lambda) = \sum_{j=0}^{i} c_j \lambda^j, \quad c_i = 1, \tag{6.140}$$

where i is the degree of the polynomial, the limit of the sequence can be found exactly as

$$\mathbf{s} = \frac{\sum_{j=0}^{i} c_j \mathbf{w}^{k+j}}{\sum_{j=0}^{i} c_j}. \tag{6.141}$$

This can be derived considering the definition of $P(\lambda)$, $P(\mathbf{T})\boldsymbol{\epsilon}^k = 0$. Therefore,

$$\mathbf{0} = P(\mathbf{T})\boldsymbol{\epsilon}^k = \sum_{j=0}^{i} c_j \mathbf{T}^j \boldsymbol{\epsilon}^j = \sum_{j=0}^{i} c_j \boldsymbol{\epsilon}^{k+j}. \tag{6.142}$$

and so

$$0 = \sum_{i=0}^{k} c_i \boldsymbol{\epsilon}_{n+i} = \sum_{i=0}^{k} c_i \boldsymbol{x}_{n+i} - \left(\sum_{i=0}^{k} c_i\right)\mathbf{s}. \tag{6.143}$$

Solving this for $\mathbf{s}$, one obtains the desired result, provided $\sum_{j=0}^{i} c_j \neq 0$.

The coefficients of $P(\lambda)$ can be computed considering

$$\mathscr{W}^{i-1}\mathbf{c}' = -\Delta\mathbf{w}_{k+i}, \quad \mathbf{c}' = \left[c_0, c_1, \ldots, c_{i-1}\right]^T, \tag{6.144}$$

where $\mathscr{W}^i = [\Delta\mathbf{w}^k, \ldots, \Delta\mathbf{w}^{k+i}]$, since from the defintion of $P(\lambda)$, one has

$$\mathbf{0} = P(\mathbf{T})\Delta\mathbf{w}^k = \sum_{j=0}^{i} c_i \mathbf{T}^j \Delta\mathbf{w}^k = \sum_{j=0}^{i} c_i \Delta\mathbf{w}^{k+j}. \tag{6.145}$$

The degree of $P(\lambda)$ can be as large as the dimension of $\mathbf{w}$. Hence, to be practical, the minimal polynomial extrapolation (MPE), the reduced rank extrapolation (RRE), the modified minimal extrapolation (MMPE), and the single-value decomposition, minimal polynomial extrapolation (SVD-MPE) all choose a polynomial of a lesser degree. The approximations corresponding to each extrapolation method are presented in what follows.

MPE Solve the overdetermined linear system $\mathscr{W}^{m-1}\mathbf{c}' = -\Delta\mathbf{w}_{k+m}$ in the least-squares sense for $\mathbf{c}' = \left[c_0, c_1, \ldots, c_{m-1}\right]^T$. This amounts to solving the optimization problem

$$\min_{c_0, c_1, \ldots, c_{p-1}} \left\| \sum_{j=0}^{m-1} c_j \Delta\mathbf{w}^{k+j} + \Delta\mathbf{w}^{k+m} \right\|_2 \tag{6.146}$$

which can also be expressed as

$$\min_{\mathbf{c}'} \left\| \mathscr{W}^{m-1}\mathbf{c}' + \mathbf{w}^{k+m} \right\|_2, \quad \mathbf{c}' = \left[c_0, c_1, \ldots, c_{m-1}\right]^T. \tag{6.147}$$

With $c_0, c_1, \ldots, c_{k-1}$ available, set $c_m = 1$ and compute $\gamma_q = c_q / \sum_{j=0}^{m} c_j, q = 0, 1, \ldots, m$, provided $\sum_{j=0}^{m} c_j \neq 0$.

[2]A polynomial $P(\lambda)$ is said to be minimal with respect to a vector $\mathbf{a}$, if $P(\mathbf{T})\mathbf{a} = 0$ and it of least degree.

RRE Solve the overdetermined linear system $\mathscr{W}^m \boldsymbol{\gamma} = 0$ in the least-squares sense, subject to the constraint $\sum_{j=0}^{m} \gamma_j = 1$. This amounts to solving the optimization problem

$$\min_{\gamma_0,\gamma_1,\ldots,\gamma_m} \left\| \sum_{j=0}^{m} \gamma_j \Delta \mathbf{w}^{k+j} \right\| \text{ subject to } \sum_{j=0}^{m} \gamma_j = 1 \tag{6.148}$$

which can also be expressed as

$$\min_{\boldsymbol{\gamma}} \left\| \mathscr{W}^m \boldsymbol{\gamma} \right\|_2 \quad \text{subject to } \sum_{j=0}^{m} \gamma_j = 1; \quad \boldsymbol{\gamma} = \left[\gamma_0, \gamma_1, \ldots, \gamma_m\right]^T. \tag{6.149}$$

MMPE Consider a set of m linearly independent vectors $\mathbf{q}_j$, $j = 1, \ldots, m$. Solve the linear system

$$\left(\mathbf{q}^j, \mathscr{W}^{m-1} \mathbf{c}'\right) = -\left(\mathbf{q}^j, \Delta \mathbf{w}^{k+m}\right), \quad j = 1, \ldots, m, \tag{6.150}$$

which can also be expressed as

$$\sum_{j=0}^{m-1} \left(\mathbf{q}_j, \Delta \mathbf{w}^{k+j}\right) \mathbf{c}_j = -\left(\mathbf{q}^j, \Delta \mathbf{w}^{k+p}\right), \quad j = 1, \ldots, m. \tag{6.151}$$

This is, in fact, a system of m linear equations for the m unknowns $c_0, c_1, \ldots, c_{m-1}$. With $c_0, c_1, \ldots, c_{m-1}$ available, set $c_m = 1$ and compute $\gamma_q = c_q / \sum_{j=0}^{m} c_j, i = 0, 1, \ldots, m$, provided $\sum_{j=0}^{m} c_j \neq 0$.

SVD-MPE Solve the standard l_2 constrained minimization problem

$$\min_{\mathbf{c}} \left\| \mathscr{W}^m \mathbf{c} \right\|_2 \quad \text{subject to } \|\mathbf{c}\|_2 = 1, \quad \mathbf{c} = \left[c_0, c_1, \ldots, c_m\right]^T. \tag{6.152}$$

The solution $\mathbf{c}$ is the right singular vector corresponding to the smallest singular value $\sigma_{\min}$ of $\mathscr{W}^m$, i.e., $\mathscr{W}^{m*} \mathscr{W}^m \mathbf{c} = \sigma_{\min}^2 \mathbf{c}$, $\|\mathbf{c}\|_2 = 1$. It is assumed that $\sigma_{\min}$ is simple so that $\mathbf{c}$ is unique up to a multiplicative constant ϕ, $|\phi| = 1$.

With $c_0, c_1, \ldots, c_m$ available, compute $\gamma_q = c_q / \sum_{j=0}^{m} c_j, q = 0, 1, \ldots, m$, provided $\sum_{j=0}^{m} c_j \neq 0$. The assumption that $\sigma_{\min}$ is simple guarantees the uniqueness of the γ_i.

When $m = 1$, MPE, RRE, MMPE, and SVD-MPE can be regarded as generalizations of the Aitken Δ^2-process to the vector case. Thus, when applied to the solution of a system of nonlinear equations using cycling

$$\mathbf{s}_{k,1} = \begin{cases} \mathbf{x}^k - \dfrac{\left(\Delta \mathbf{x}^k, \Delta \mathbf{x}^k\right)}{\left(\Delta \mathbf{x}^k, \Delta^2 \mathbf{x}^k\right)} \Delta \mathbf{x}^k & \text{for MPE,} \\ \mathbf{x}^k - \dfrac{\left(\Delta^2 \mathbf{x}^k, \Delta \mathbf{x}^k\right)}{\left(\Delta^2 \mathbf{x}^k, \Delta^2 \mathbf{x}^k\right)} \Delta \mathbf{x}^k & \text{for RRE,} \\ \mathbf{x}^k - \dfrac{\left(\mathbf{q}_1, \Delta \mathbf{x}^k\right)}{\left(\mathbf{q}_1, \Delta^2 \mathbf{x}^k\right)} \Delta \mathbf{x}^k & \text{for MMPE,} \\ \mathbf{x}^k - \dfrac{\left(\mathbf{g}_0, \Delta \mathbf{x}^k\right)}{\left(\mathbf{g}_0, \Delta^2 \mathbf{x}^k\right)} \Delta \mathbf{x}^k & \text{for SVD-MPE.} \end{cases} \tag{6.153}$$

Sidi (2017) also suggest cycling with frozen γ_i, where after some iterations the γ_i are frozen and reused henceforth. A parallel version of the full cycling procedure is also described.

Connection to Krylov subspace methods According to Sidi (2017), the so-called Krylov subspace methods are closely related to the vector extrapolation methods presented above. When the latter is applied to vector sequences obtained using fixed-point iterative methods to nonsingular linear systems of equations, they are mathematically equivalent. More precisely, the MPE and the RRE methods are mathematically equivalent to the methods of Arnoldi and generalized minimal residual (GMR).

However, Krylov subspace methods and extrapolation methods differ in their algorithmic aspects entirely: The only input of the former is a procedure that performs the matrix-vector multiplication without explicitly knowing the matrix coefficient matrix. The latter takes as their only input a vector sequence that results from a fixed-point iterative scheme without knowing the matrix coefficient matrix to know what the scheme is.

In Michler et al. (2005), a Krylov-subspace method is proposed in the context of FSI. However, as pointed out by Küttler and Wall (2009), the correct term for this approach should be instead a "Krylov-based vector extrapolation" method. The method proposed can be obtained by applying the RRE to the sequence of residuals computed as $\Delta\mathbf{r}_i^* = \mathbf{x}_i^* - \mathbf{x}^k$, where the subscript i concerns the internal loop of the vector extrapolation method, and whose limit is $\mathbf{0}$. Küttler and Wall (2009) argues that these residual differences have unfavorable numerical properties and should be avoided.

6.6 Multipoint iteration functions with memory

Multipoint iteration functions are rarer and are not thoroughly investigated in this exposition. One can mention the Airola-Nevanlinna family of methods (Fang and Saad, 2009) and Netwon-Krylov method that reuses the Krylov subspace from previous iterations (Sidi, 2017).

6.7 Conclusions

Table 6.1: Summary of the comparison between method for the solution methods of non-linear systems of equations. n here denotes the number of unknowns and m denotes depending on the context the number of previous iterates considered, the number of fixed point evaluations or the size of the Krylov subspace.

Method	Memory requirements	Nr function evaluations per iteration	Observations
Fixed-point iteration	2 $(n \times 1)$ vectors	1	•Often diverges. •Simplest method. •Memory efficient.
Underrelaxation	2 $(n \times 1)$ vectors	1	•Simple. •Improved stability over fixed-point. •Need to manually choose a relaxation parameter.
Aitken relaxation	3 $(n \times 1)$ vectors	1	•Very popular in FSI. •Dynamic relaxation. •Improved stability over fixed-point.
Broyden-like family (Fang and Saad, 2009)	2 $(n \times 1)$ vectors 2 $(n \times (m-1))$ matrices	1	•$\mathcal{O}(n^3)$ computation complexity for $m > 1$ (QR decomposition). •Low number of function evauations •Superlinear convergence when $m = 1$.
Broyden's method (Kelley, 2003)	Up to $(m+2)(n \times 1)$ matrices	1	•$\mathcal{O}(n)$ computation complexity. •Low storage. •Superlinear convergence.
Newton-Krylov	Up to $(m+1)$ $(n \times 1)$ vectors	$m+1^*$	•Large number of iterations possible. •Popular for the solution of systems of nonlinear equations. •Quadratic convergence under appropriate conditions.
Cycling with vector extrapolation	$(m+2)$ $(n \times 1)$ vectors	$m+1$	•Large number of function evaluations. •$\mathcal{O}(n^3)$ computational complexity (QR decomposition).

* The number of function evaluations in the Newton-Krylov methods will depend on how many iterations it will take for the inner loop to converge. There is a function evaluation per iteration of the inner loop.

Table 6.2: Summary of the update formulas for the solution methods of non-linear systems of equations.

Method	Update formula
Fixed-point	$\mathbf{x}^{k+1} = \mathbf{x}^k - \mathcal{R}^k$
Underrelaxation	$\mathbf{x}^{k+1} = \mathbf{x}^k - \omega\mathcal{R}^k$, $0 < \omega < 1$.
Aitken relaxation	$\mathbf{x}^{k+1} = \mathbf{x}^k - \omega^{(k)}\mathcal{R}^k$, $\omega^{(k)} = -\omega^{(k-1)} \frac{\left(\mathbf{r}^{(k)} - \mathbf{r}^{(k-1)}\right)^{\mathrm{T}} \mathbf{r}^{(k-1)}}{\left(\mathbf{r}^{(k)} - \mathbf{r}^{(k-1)}\right)^2}$.
Broyden's family	$\mathbf{x}^{k+1} = \mathbf{x}^k - \left(G_{\mathcal{R}}^{k-m} + \left(\mathscr{X}^k - G_{\mathcal{R}}^{k-m}\mathscr{R}^k\right)\mathbf{V}^{k^T}\right)\mathcal{R}^k$, $\mathbf{V}^{k^T} = \mathbf{M}^{k^{-1}}\mathbf{N}^{k^T}$, Type I: $M^k = \mathscr{X}^{k^T} G_{\mathcal{R}}^k \mathscr{R}^k, \quad N^{k^T} = \mathscr{X}^{k^T} G_{\mathcal{R}}^k$, Type II: $M^k = \mathscr{R}^{k^T}\mathscr{R}^k, \quad N^{k^T} = \mathscr{R}^{k^T}$ $\mathbf{V}^k$ is usually computed using QR-decomposition.
Anderson's family	$\mathbf{x}^{k+1} = \mathbf{x}^k - \left(-\beta\mathbf{I} + \left(\mathscr{X}^k + \beta\mathscr{R}^k\right)\mathbf{V}^{k^T}\right)\mathcal{R}^k$, $\mathbf{V}^{k^T} = \mathbf{M}^{k^{-1}}\mathbf{N}^{k^T}$, Type I: $M^k = \mathscr{X}^{k^T}\mathscr{R}^k, \quad N^{k^T} = \mathscr{X}^{k^T}$, Type II: $M^k = \mathscr{R}^{k^T}\mathscr{R}^k, \quad N^{k^T} = \mathscr{R}^{k^T}$ $\mathbf{V}^k$ is usually computed using QR-decomposition.
Newton-Krylov	$\mathbf{x}^{k+1} = \mathbf{x}^k + \Delta\mathbf{x}^k$ $J_{\mathcal{R}}(\mathbf{x}^k)\Delta x^k = -\mathcal{R}^k$ solved using a Krylov method to accuracy $\lVert J_{\mathcal{R}}(\mathbf{x}^k)\Delta x_m^* + \mathcal{R}(\mathbf{x}^k)\rVert \leq \eta\lVert\mathcal{R}(\mathbf{x}^k)\rVert$
Cycling with vector extrapolation	$\mathbf{x}^{k+1} = \sum_{j=0}^m \gamma_j \mathcal{S}^{i+j}(\mathbf{x}^k)$, $\gamma_q = c_q / \sum_j c_j, \quad q = 0, 1, \ldots, m$, **MPE**: $\mathbf{c}' = [c_0, c_1, \ldots, c_{m-1}]^T = \arg\min_{\bar{\mathbf{c}}} \left\lVert \mathscr{S}^{m-1}\bar{\mathbf{c}}' + \mathcal{S}^{i+m}(\mathbf{x}^k) \right\rVert_2$, $c_m = 1$, **RRE**: $\boldsymbol{\gamma} = [\gamma_0, \gamma_1, \ldots, \gamma_m]^T = \arg\min_{\bar{\boldsymbol{\gamma}}} \lVert \mathscr{S}^m \hat{\boldsymbol{\gamma}} \rVert_2$, Subject to $\sum_{j=0}^m \gamma_j = 1$. **MMPE**: $\left(\mathbf{q}^j, \mathscr{S}^{m-1}\mathbf{c}'\right) = -\left(\mathbf{q}^j, \mathcal{S}^{i+m}(\mathbf{x}^k)\right), \quad j = 1, \ldots, m$, for a set of m linearly independent vectors $\mathbf{q}_j$, $j = 1, \ldots, m$. $\mathbf{c}' = [c_0, c_1, \ldots, c_{m-1}]^T$. **SVD-MPE**: $\mathbf{c} = \arg\min_{\bar{\mathbf{c}}} \lVert \mathscr{S}^m \bar{\mathbf{c}} \rVert_2 \quad$ subject to $\lVert\bar{\mathbf{c}}\rVert_2 = 1$, $\mathbf{c} = \left[c_0, c_1, \ldots, c_m\right]^T$.