Chapter 1

Jacobian-Free Newton-Krylov Solver

A Jacobian-Free Newton-Krylov (JFNK) solver is a general name for an algorithm that solves a system of nonlinear equations. As implied by the name, these solvers are characterized by two main features: lack of a need to form the exact Jacobian and Newton-like updates obtained from a Krylov method. The need for a Jacobian and an explanation of Newton updates will be discussed first. An overview of Krylov methods and the more specific Generalized Minimal Residual (GMRES) method will follow. The section will conclude with an approximation that allow the pieces to fit together and form a JFNK solver.

1.1   Newton's Method

Systems of coupled, nonlinear equations are pervasive across all disciplines of science and engineering. They are also one of the most important and difficult class of problems to analyze and solve. While there are numerous analytical and numerical techniques for solving such problems, Newton's method is the singular option that will be explored here.

1.1.1 Problem Statement

The problem under consideration is defined as follows [1]: given an $n \times 1$ vector of nonlinear equations $r(x)$, determine an $n \times 1$ vector x_n such that

$$r(x_n) = 0. \tag{1.1}$$

Because each row of the equation vector is sought to be identically zero in this definition, the problem is sometimes referred to as a multi-dimensional root-finding problem. A different but similar statement can be formed by viewing the solution as the result of a minimization procedure: given an $n \times 1$ vector of nonlinear equations $r(x)$, determine an $n \times 1$ vector x_n such that

$$x_n = \underset{x}{\text{ArgMin}} \, \|r(x)\| \tag{1.2}$$

in any valid norm of $r(x)$, where ArgMin_x is a function that returns the argument which minimizes $\|r(x)\|$ over the domain of x. The minimization problem is equivalent to the root-finding problem as long as $r(x)$ admits a root in the domain of x. If it does not, as in $r(x) = x^2 + 1$ where $x \in \mathbb{R}$, the minimization problem still has a well-defined solution. However, because many engineering problems posed in the manner of equation 1.1 arise from a physical balance of left- and right-hand sides involving x, a root most often exists within its domain.

Either problem is still difficult to solve in practice. Even for a modest value of n, because the function $r(x)$ is taken to be nonlinear, a closed-form solution to either variation may be impractical, if not impossible, to find. Therefore, numerical techniques like Newton's method are frequently used to solve the problem approximately.

1.1.2 Newton Update

In an effort to satisfy equation 1.1, Newton's method forms a solution from a recursive series of linear approximations. The method begins with some initial value x_0. Assuming $r(x)$ has a root in the domain of x, if $\|r(x_0)\|$ is not sufficiently close to zero, there may be a direction δx_0 in which the norm decreases, called a *descent direction*. Such a direction can be found by considering the function's Taylor series about x_0 evaluated at $x_0 + \delta x_0$:

$$r(x_0 + \delta x_0) = r(x_0) + J_r(x_0)\delta x_0 + c(x_0, \delta x_0) \tag{1.3}$$

where $J_r(x_0)$ is an $n \times n$ Jacobian matrix and $c(x_0, \delta x_0)$ is an $n \times 1$ vector of higher-order corrections that ensure the equality [2]. Near the expansion point, the higher-order corrections are small relative to the linear term and may be neglected. Further, to both drive $r(x)$ toward zero and yield a solvable system, the function value $r(x_0 + \delta x_0)$ is taken to be zero. This procedure can be thought of as assuming the direction and magnitude of δx_0 leads to the exact solution of the problem. The resulting linear system used to solve for the descent direction is

$$J_r(x_0)\delta x_0 = -r(x_0). \tag{1.4}$$

The above procedure yields a locally linear system whose solution is the descent direction δx_0 as long as the Jacobian is non-singular. Any search direction calculated from the above formula is called a *Newton direction*. Of course, since the higher-order corrections were ignored in attaining the linear system, this δx_0 will unlikely lead to an acceptable solution. However, if δx_0 leads to a new point with a norm smaller than that of the initial guess, it is arguably a better point at which to form the Taylor expansion.

From the new point $x_1 = x_0 + \delta x_0$, another linear system can be formed, a new descent direction calculated, and the process repeated. As such, the k-th new point from the current

Figure 1.1: The first few Newton updates of equation 1.14 with a starting guess of $x_0 = 1$. Solid lines, aside from the residual's, indicate tangents, and dashed lines indicate update locations from the roots of the tangents.
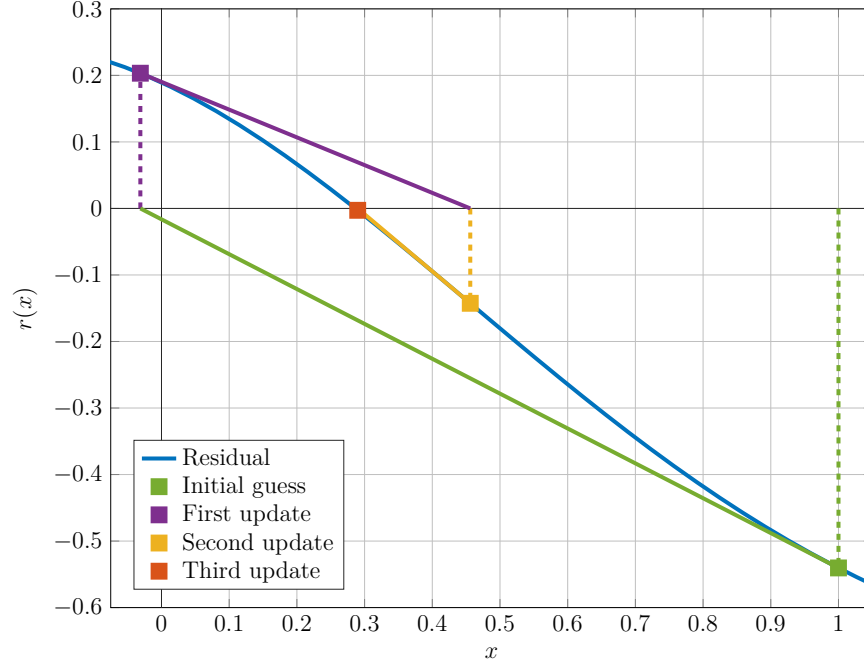
expansion location x_{k-1} moving in the Newton direction δx_{k-1} is

$$x_k = x_{k-1} - J_r^{-1}(x_{k-1})r(x_{k-1}) = x_{k-1} + \delta x_{k-1} \tag{1.5}$$

As long as each successive point from the locally linear system yields a norm smaller than its predecessor, the repetition is forming a sequence of points that is approaching a solution to the nonlinear problem. Moreover, if each successive point decreases the residual's norm, the norms of the descent directions and the higher-order correction terms are decreasing as well. So if the sequence of points is convergent to a solution, the linear approximation becomes increasingly valid as the Newton step decreases. Setting aside algorithmic flow, stopping criteria, and other details, this update procedure is Newton's method.

As an example of the above procedure, the residual function is taken to be

$$r(x) = \text{Exp}\left[-\left(x + \frac{1}{4}\right)^2\right] - \frac{3}{4} \tag{1.6}$$

While the function admits a simple analytical solution, Newton's method may be applied to find one of the solutions. Graphical results of applying the method to the residual with a starting guess of 1 are shown in figure 1.1. As can be seen, each successive iteration of the method results in a decrease of the norm and a decrease of the length of the Newton update. It is noted that because this residual is nonlinear, it does have another solution which is not found in this example because the initial guess is further from it than the one found. In accordance with the discussion above, Newton's method finds *a solution* of the problem but not all of the solutions.

1.1.3   Line Search

While the solution of equation 1.4 is a descent direction and figure 1.1 shows smooth convergence with full Newton steps, reduction of the norm is only guaranteed close to the point of expansion where preclusion of the higher-order terms in the Taylor expansion is valid. As such, the magnitude of δx_{k-1} may result in an x_k with a higher norm due to stepping outside the neighborhood of descent. To remedy this problem, implementations of Newton's method often introduce a step fraction α such that equation 1.5 becomes

$$x_k(\alpha) = x_{k-1} + \alpha \delta x_{k-1}. \tag{1.7}$$

Using this formulation, after solving for the descent direction δx_{k-1} as normal, the scaling parameter is calculated from

$$\alpha = \underset{\alpha^*}{\text{ArgMin}} \, \|r(x_k(\alpha^*))\| \tag{1.8}$$

This problem is referred to as a *line search* and is solved by finding the scalar value α which minimizes the norm of the residual along the Newton direction [1]. Because the linear assumption is valid near the expansion point, the minimization often results in an α bounded between 0 and 1, meaning that the Newton direction is often underrelaxed. However, if the norm of the function exhibits positive convexity along the entire descent direction, an overrelaxtion of the step fraction may be possible and can lead to faster convergence.

While finding the exact minimum of a residual along the search direction is an admirable goal in theory, it can be computationally disadvantageous. Therefore, instead of strictly adhering to equation 1.8, which is called an *exact line search*, less stringent criteria that guarantee acceptable advancement of the solution are often employed in practice. These criteria form the basis for an *inexact line search*. The most obvious criterion to impose would be a reduction in the residual's norm after the step:

$$\|r(x_{k-1} + \alpha\,\delta\!x_{k-1})\|_2 < \|r(x_{k-1})\|_2 \tag{1.9}$$

Another option is one often employed in routines for finding extrema: the *Armijo Rule*. This criterion requires the step to generate a residual value that falls below a relaxed tangent line of the norm of the residual [3]:

$$\|r(x_{k-1} + \alpha\,\delta\!x_{k-1})\|_2 < \|r(x_{k-1})\|_2 + \alpha\,\beta\,\left.\frac{\partial\|r(x_{k-1})\|_2}{\partial\alpha}\right|_{\alpha=0}. \tag{1.10}$$

The relaxation parameter β is often taken to be small such that α is not restricted to small values for steep residuals and that the criterion is similar to the former, pure reduction one for modestly decreasing residuals. The derivative along the step fraction α is

$$\left.\frac{\partial\|r(x_{k-1})\|_2}{\partial\alpha}\right|_{\alpha=0} = \frac{r(x_{k-1})^{\mathsf{T}} J_r(x_{k-1})\delta\!x_{k-1}}{\|r(x_{k-1})\|_2} = -\|r(x_{k-1})\|_2. \tag{1.11}$$

The latter reduction comes from equation 1.4 and succinctly shows that the Newton direction is a descent direction since the value of the derivative is always negative. Inserting the value

of the derivative into equation 1.10 shows that the Armijo rule is stricter than equation 1.9 because the reduction requirement is below unity if α is non-zero. However, the criterion is still a simple reduction requirement with a free parameter β that can be used to control the overall strictness:

$$\|r(x_{k-1} + \alpha \, \delta x_{k-1})\|_2 < (1 - \alpha \, \beta)\|r(x_{k-1})\|_2. \tag{1.12}$$

It is noted that the simplification of the Armijo rule to this form does not arise in its optimization uses because those routines solve a linear system involving the Hessian and gradient of the residual's norm. For the rest of this work, the criterion for acceptance of the step fraction α will be one of strict reduction with a multiplier of unity.

There are a number of algorithms that could be used to perform the line search to meet the imposed criterion. The methods vary in their complexity, run-time, and robustness. The following is a small, non-exhaustive list of line search methods:

- *Proportional back-tracking*: iteratively multiply the current step fraction by some factor less then one until a reduction in the norm is found. While extremely simple, the method only requires evaluation of the residual and is guaranteed to converge since α will eventually be reduced into the local area of descent.

- *Interpolation back-tracking*: use any known values of the residual or Jacobian to form a polynomial whose minimum can be easily found and, if necessary or desired, iterate using the new information. Since the calculated residual is assumed to be higher than the expansion point's and the initial derivative is downward, at least one minimum of the polynomial will exist within the interval. This procedure is typically applied iteratively with a quadratic or a cubic polynomial since their minima are readily computed. As long as all of the residuals used are higher than the expansion point's, the interval over

which the polynomial is constructed will shrink and eventually be reduced into the local area of descent.

- *Golden Section Search*: starting from two bounding points and two interior points of an interval, find the new interval which guarantees a minimum is contained within itself, sample a new interior point, and iterate until convergence. The "golden" part of the name comes from the interior points being chosen such that they are a relative distance of one over the golden ratio (~ 0.618) from the interval bounds. This method is most useful for exact line searches since an exact minimum in the relaxation regime of α may potentially be exactly found, as opposed to the previous back-tracking methods. However, the search is only guaranteed to find the true minimum if the residual is unimodal (i.e., only has one minimum) within the initial interval and only converges linearly to a minimum. As such, either the proportional and interpolation back-tracking methods are most frequently used. This method is mentioned for an example of a non-back-tracking algorithm.

The method implemented in this work is proportional back-tracking with plans to add interpolation back-tracking in the future.

As an example to motivate the use of line searches, consider the following vector-valued function:

$$f(x;\mu) = \begin{bmatrix} (2\mu)^{-1} \, \mathrm{Exp}\Big[-\mu(x_1 + x_2)^2\Big] \\ (x_1^2 + 1)^{-1} \, \mathrm{Cosh}[x_2] \end{bmatrix}, \tag{1.13}$$

where $x = [x_1, x_2]^{\mathsf{T}}$ and the scalar parameter μ is used to adjust the function's convexity near the origin. The associated residual is taken to be

$$r(x;\mu) = f(x;\mu) - f([0,0]^{\mathsf{T}};\mu), \tag{1.14}$$

Figure 1.2: Residual value of equation 1.14 as a function of α for several values of μ.
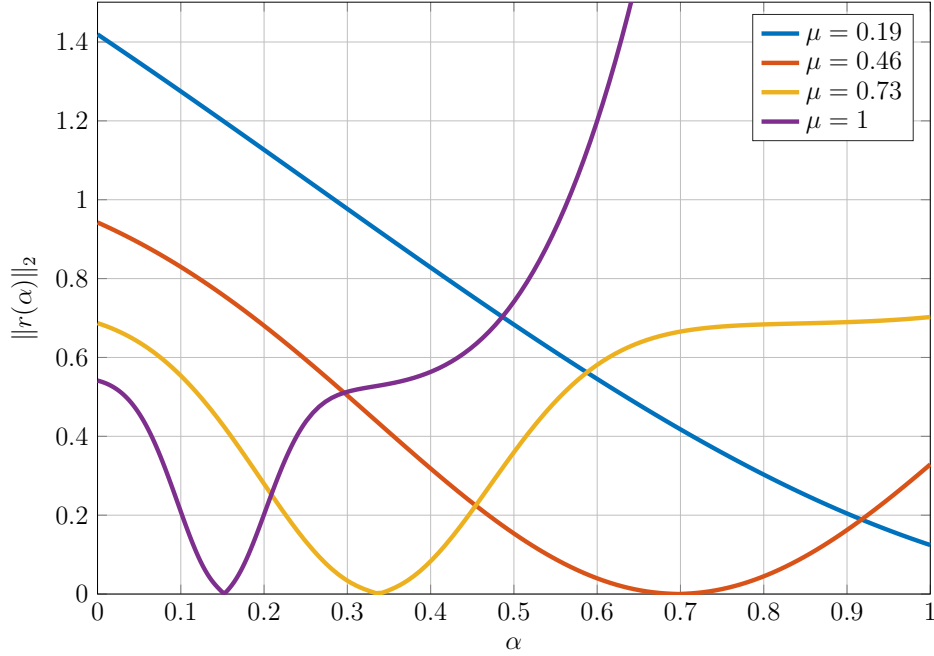
whose solution is then $x_n = [0,0]^\top$ by construction. Figure 1.2 shows the residual norms

for the first Newton step from the initial point $x_0 = [1,1]^\top$ for several values of μ. For the

lowest value of μ, the norm is decreasing along the entire length of the Newton step while

the residual quickly explodes to over twenty-times its initial value for the highest μ. The

intermediate parameters exhibit mild features of the extremes: one that reduces the norm at

the full Newton step but not as much as possible and the other results in a slight increase in

the norm but does showcase a promising minimum near a step fraction of 0.33.

Even for this simple example, some form of line search would be required to ensure convergence

across all values of μ. The two highest values are prime candidates for one of the back-tracking

schemes since they have multiple extrema. The second smallest value is a candidate for a

golden section search since it is unimodal with a better optimum than the full step at around

0.75. Finally, the lowest μ value can actually be over-relaxed to a value around 1.3. However,

these observations and determinations can only be made since the true residual was also

---

Algorithm 1: Nonlinear solve with Newton's Method

---

1 function `newtonSolve`$(x_0, r(x), J(x), \varepsilon, \tau)$

2 $x_{k-1} = x_0$ % Initialization

3 $r_{k-1} = r(x_{k-1})$

4   **while** $\|r_{k-1}\|_2 > \varepsilon$ **do**   % begin linear solves

5   **Solve** $J(x_{k-1})\, \delta\!x_{k-1} = -r_{k-1}$

6 $x_k = x_{k-1} + \delta\!x_{k-1}$

7 $r_k = r(x_k)$

8   **if** $\|r(x_k)\|_2 > \|r(x_{k-1})\|_2$ % back-track if needed

9 $\alpha = \tau$

10   **while** $\|r(x_k)\|_2 > \|r(x_{k-1})\|_2$ **do**

11 $x_k = x_{k-1} + \alpha\, \delta\!x_{k-1}$

12 $r_k = r(x_k)$

13 $\alpha = \tau\, \alpha$

14   **end**

15   **end**

16 $x_{k-1} = x_k$

17 $r_{k-1} = r_k$

18   **end**

19   **Return** x_k

20 end

---

observed. In practice, a back-tracking algorithm is employed only if the residual does not reduce enough and the Newton iteration is continued automatically.

1.1.4   Algorithm

An outline of the nonlinear solving algorithm is given in algorithm 1. The function introduces two more parameters that have been implied but not fully addressed since they are independent of the mathematics. The first parameter is the *absolute residual tolerance* ε. The Newton

iteration is considered complete when the residual's norm falls below ε. If the system is scaled to around an order of one, common default values of ε fall in the range of 10^{-6} to 10^{-8}. The second parameter is the constant used in the proportional back-tracker τ. The value of τ is arbitrary but a good value has been found to be 0.5.

The motivation and derivation of Newton's method for solving non-linear equations is now complete. The next item to be explained is how to efficiently solve the linear system, as shown on line 5 of algorithm 1, and how the manner in which the system is solved will allow the preclusion of calculating and inverting the Jacobian.

1.2 Krylov Methods

Krylov methods are a class of techniques used to solve linear systems of the form

$$Ax = b \tag{1.15}$$

where x and b are $n \times 1$ vectors and A is an $n \times n$ nonsingular matrix [4]. Different methods place other limitations on A, but a required condition for all methods is that the coefficient matrix A be invertible. The exact solution to equation 1.15 is simply

$$x = A^{-1}b. \tag{1.16}$$

Although trivially , the exact inversion of A can be computationally and memory intensive if n is large. The computational cost of inverting A is only exacerbated when solving nonlinear problems with a Newton-like scheme since several, if not many, linear solves are required to make one nonlinear advancement.

Krylov methods aim to provide an approximate solution to equation 1.15 without ever forming

the inverse of A explicitly. To explain the procedure used, first, the solution will be written in an inverse-free form. Then, a general explanation of Krylov subspaces and methods will be given. The section will conclude with the particular Krylov method used for this work: GMRES.

1.2.1   Inverse-free Form

Let x_n represent the exact solution to equation 1.15 and x_0 represent an initial guess to the solution. Assuming the problem is solvable, there exists some δx such that $x_n = x_0 + \delta x$. Therefore, the residual vector for the initial guess from the solution is

$$r_0 = b - Ax_0 \tag{1.17}$$

Isolating b on the left-hand side and multiplying through by A^{-1} then gives

$$A^{-1}b = x_0 + A^{-1}r_0 \tag{1.18}$$

Substituting this into equation 1.16 gives

$$x_n = x_0 + A^{-1}r_0. \tag{1.19}$$

The burden of calculating A's inverse is still present. To avoid this burden, we use a result of the Cayley-Hamilton theorem: the inverse of a square $n \times n$ matrix can be expressed as a linear combination of the matrix's powers from 0 to $n - 1$ [5]. To wit, given the proper weights c_i, the equality $A^{-1} = \sum c_i A^i$ is satisfied and equation 1.19 can be re-written to

$$x_n = x_0 + \sum_{i=0}^{n-1} c_i A^i r_0. \tag{1.20}$$

The true solution to the problem is now in a form that incorporates some initial guess and is free of matrix inversions. However, the weights c_i, which are required for the equality to

hold, are based on A's characteristic polynomial. Calculating these weights exactly are, as in the explicit calculation of the inverse, difficult to efficiently compute for large n and an approximation must be made for tractability.

1.2.2   Generic Krylov Methods

The set of vectors $\{A^i r_0\}$ present in equation 1.20 is called the Krylov subspace of A, denoted $\mathcal{K}_n(A, r_0)$, and is the basis of all Krylov methods [4]. The set is of dimension n and will hereafter be referred to as the full subspace. Typically, the methods to be discussed use a variable number m of lower dimensional spaces that will likewise be denoted $\mathcal{K}_m(A, r_0)$.

The approximate solution formed by Krylov methods is based on one simplification to equation 1.20: only form a subset of the full subspace; the idea being that the true solution in the full subspace may not be far from the subset. At every m-th iteration of a Krylov method, a new vector from the Krylov subspace is generated through the application of the matrix A to the previous vector and is then added to the current vector set $\mathcal{K}_m(A, r_0)$. An overdetermined problem from a linear combination of the vector set with m undetermined coefficients is then solved to create a new approximate solution. The set is grown and approximate solution updated until the norm of the residual is sufficiently low. The maximum number of basis vectors needed varies by problem, but it is guaranteed that the approximation will reach the true solution once the full subspace is formed since the number of degrees of freedom will equal the dimension of A.

In terms of linear algebra, let $Z_m = [z_1, z_2, ..., z_m]$ be an $n \times m$ matrix whose columns form a basis for $\mathcal{K}_m(A, r_0)$ and $w_m = [\omega_1, \omega_2, ..., \omega_m]^\mathsf{T}$ be an $m \times 1$ weight vector (analogous to the aforementioned c_i). Starting with $z_1 = r_0 / \|r_0\|$, at every m-th iteration, generate a new

vector in the Krylov subspace by applying A to the previous vector: $z_m = Az_{m-1}$. A new solution direction $\delta\!x_m$ is then found by calculating the weight vector w_m with z_m added into the Krylov expansion. Therefore, the m-th approximate solution is

$$x_m = x_0 + \delta\!x_m = x_0 + Z_m w_m \tag{1.21}$$

with the associated residual

$$r_m = r_0 - A\delta\!x_m = r_0 - AZ_m w_m. \tag{1.22}$$

Two side effects of iteratively growing the basis matrix Z_m are that the weights are no longer based on A's characteristic polynomial and the resulting system to solve for the weights is overdetermined as long as $m < n$. To remedy this problem, Krylov methods define an optimality condition that results in weights that attempt to reduce the residual every iteration.

Since Krylov methods are a family, a specific method arises from consideration of A's structure, choice of optimality condition, and choice of the basis vectors for Z_m [4]. GMRES will give concreteness to these three choices.

1.2.3 GMRES

First presented by Saad and Schultz in 1986, GMRES is a Krylov method designed to solve any linear system with an invertible matrix A [6]. Because GMRES does not require any other special structure, it is well-suited for generic, unsymmetrical problems. Of course, with the power to solve any invertible system comes a burden of mathematical and computational complexity when compared to more specialized Krylov methods.

The algorithm is developed with the goal of solving the minimization problem

$$\delta\!x_m = \underset{\delta\!x_m^*}{\mathrm{ArgMin}} \, \|r_0 - A\delta\!x_m^*\|_2^2. \tag{1.23}$$

Therefore, the m-th solution direction is the one which minimizes the associated residual r_m. If the solution direction has n free parameters, the minimization problem is equivalent to exactly solving the full linear system. However, if the direction is limited to $m < n$ free parameters, the problem is one of least squares. The least squares problem is the one to be solved because GMRES aims to solve the linear system without a full matrix inversion.

The least squares problem is solved by finding the m-th residual vector r_m that lies orthogonal to the A-combination of the current Krylov subspace:

$$r_m \perp A\mathcal{K}_m(A, r_0) \tag{1.24}$$

Noting that $\delta\!x_m \in \mathcal{K}_m(A, r_0)$, the requirement of orthogonality can be seen by considering the equality $r_0 = r_m + A\delta\!x_m$. If r_0 lies within the plane $A\mathcal{K}_m(A, r_0)$, it can be exactly found as a linear combination of the basis vectors Z_m, and r_m must be exactly 0 to honor the balance. If it lies outside of the plane, the closest $A\delta\!x_m$ can approach r_0 is its orthogonal projection onto the plane with r_m representing the components that lie outside the current subspace. Figure 1.3 shows a simple two-dimensional triangle analogue of this situation: assuming r_0 is

Figure 1.3: Two-dimensional analogue of the n-dimensional triangle that motivates the optimality condition: the length of r_m is minimal when it is orthogonal to $A\delta\!x_m$.
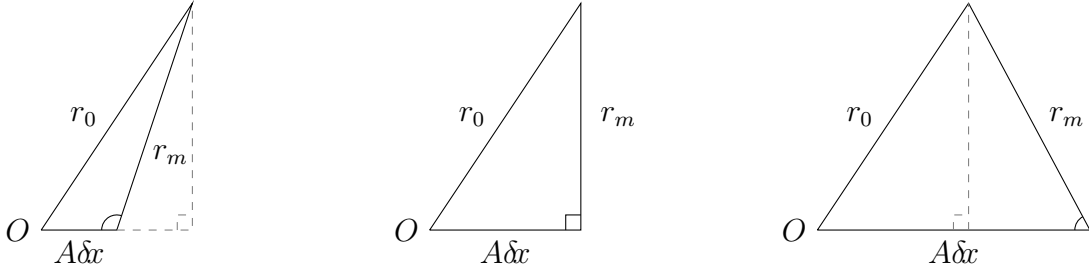
not parallel to $A\delta x_m$, r_m has its shortest, non-zero length when it is perpendicular to $A\delta x_m$.

The final choice to be made for the implementation of GMRES is that of the vectors used to form Z_m. The so-called classic GMRES of Saad and Schultz proposed an iteratively grown orthonormal basis for $\mathcal{K}_m(A, r_0)$ and used those vectors to form Z_m [6]. The choice of basis is natural since the solution lies in the Krylov subspace. However, this choice of basis leads to a least squares problem involving a Hessenberg matrix which itself requires a QR decomposition for an efficient solution. As such, a version called Simpler GMRES was later developed by Walker and Zhou that builds an orthonormal basis for $A\mathcal{K}_m(A, r_0)$ and uses those basis vectors to form Z_m [7]. The choice of vectors in Simpler GMRES leads directly to a QR factorization and is therefore simpler to implement; however, the choice was proven to result in an ill-conditioned R-factor and therefore be far less numerically stable than the classic method [8]. The stability problem was resolved by a method called Adaptive Simpler GMRES (ASGMRES) due to Jiránek and Rozložník. ASGMRES switches between the basis of Simpler GMRES and a basis formed from the sequence of residuals from the iterative solution [9]. Specifically, at the m-th iteration of the solution, given a Simpler GMRES basis vector q_{m-1} and residuals from two previous iterations r_{m-1}, r_{m-2}, the vector z_m is assigned thusly:

$$z_m = \begin{cases} r_{m-1}/\|r_{m-1}\|_2 & \text{if } \|r_{m-1}\|_2 < \nu\|r_{m-2}\|_2 \\ q_{m-1} & \text{otherwise} \end{cases}. \tag{1.25}$$

The reasoning behind switching between the two sets is that one is more stable if the residuals are decreasing sharply and the other is more stable if the residuals are decreasing slowly. The parameter ν, which is bounded between 0 and 1, is used to determine what decrease is sufficient enough to warrant the use of the residual vectors.

With the choice of basis vector for Z_m complete, the problem of solving the least squares

problem is all that remains. From equation 1.22, an equivalent form of equation 1.23 is

$$w_m = \underset{w_m^*}{\text{ArgMin}} \|r_0 - AZ_m w_m^*\|_2^2. \tag{1.26}$$

The QR factorization of the $n \times m$ coefficient matrix AZ_m has the form

$$AZ_m = \widetilde{Q}_m \widetilde{R}_m = \begin{bmatrix} Q_m & Q_{n-m} \end{bmatrix} \begin{bmatrix} R_m \\ 0 \end{bmatrix} = Q_m R_m \tag{1.27}$$

where $\widetilde{Q}_m$ is an $n \times n$ orthogonal matrix, $\widetilde{R}_m$ is an $n \times m$ full rank rectangle, Q_m is an $n \times m$ orthogonal rectangle, Q_{n-m} is an $n \times (n-m)$ orthogonal rectangle, and R_m is an $m \times m$ upper-triangular matrix. The matrices $\widetilde{Q}_m$ and $\widetilde{R}_m$ comprise what is called the full factorization, and Q_m and R_m comprise the reduced factorization.

The minimization problem can be reduced to a solvable, square system using the QR factorization. Three key properties of the orthogonal matrix $\widetilde{Q}_m$ aid in this reduction: the transpose $\widetilde{Q}_m^T$ is also the inverse, the transpose is also orthogonal, and orthogonal matrices do not change the 2-norm of a vector ($\|Qx\|_2 = \|x\|_2$) [10]. Introducing the full QR factorization into the minimization problem and using the properties of $\widetilde{Q}_m$, equation 1.26 has the equivalent form

$$w_m = \text{ArgMin} \left\| \widetilde{Q}_m^T r_0 - \widetilde{R}_m w_m \right\|_2^2. \tag{1.28}$$

The problem can be further simplified using the reduced factorization to yield

$$w_m = \text{ArgMin} \left\| \begin{bmatrix} Q_m^T r_0 - R_m w_m \\ Q_{n-m}^T r_0 \end{bmatrix} \right\|_2^2$$
$$= \text{ArgMin} \left(\|Q_m^T r_0 - R_m w_m\|_2^2 + \left\| Q_{n-m}^T r_0 \right\|_2^2 \right).$$

Since the quantity $\|Q_{n-m}^T r_0\|_2^2$ is independent of the weights, the minimization is achieved by solving the term involving the reduced factorization to render its norm zero. The weights are

therefore calculated from the square, linear system

$$R_m w_m = Q_m^T r_0. \tag{1.29}$$

With the weights now known, an approximate solution can be found and the residual checked for convergence against some tolerance.

While calculating the weights and explicitly evaluating the residual every iteration is possible, that procedure will slow down the solution as R_m becomes large. A way around this problem is to recall that r_m lies perpendicular to $A\mathcal{K}_m(A, r_0)$ and the columns of $Q_m = [q_1, q_2, \ldots, q_m]$ form an orthonormal basis for $A\mathcal{K}_m(A, r_0)$ [9]. As such, r_m can be calculated by an orthogonal projection using Q_m:

$$r_m = (I - Q_m Q_m^\top) r_0 = (I - q_m q_m^\top) r_{m-1} = r_{m-1} - (q_m^\top r_{m-1}) q_m. \tag{1.30}$$

This formula allows for the calculation of the m-th residual without explicitly calculating w_m and will therefore increase the efficiency of the implementation.

1.2.4   Algorithm

A full outline of an implementation of ASGMRES is given in algorithm 2. The absolute convergence tolerance ε and residual decrease ratio ν are two additional inputs to the required pieces of the linear system itself. A full explanation of the QR factorization as shown on lines 4 and 15 is omitted for brevity; however, this work makes use of a QR implementation based on Householder reflectors as described in [11]. The final solution of the linear system for the weights will also be omitted, but since R_m is upper-triangular, the solution is simply one of back-substitution.

Algorithm 2: Solve a linear system using Adaptive Simple GMRES

---

1   **function** asgmres(x_0, A, b, ε, ν)

2     $r_0 = b - Ax_0$ % Initialization

3     $z_1 = r_0/\|r_0\|_2$

4   **UpdateQR** $AZ_1 = Q_1 R_1$

5     $r_1 = r_0 - \left(q_1^\mathsf{T} r_0\right) q_1$

6     $m = 2$

7     $r_{m-2} = r_0$

8     $r_{m-1} = r_1$

9   **while** $\|r_{m-1}\|_2 > \varepsilon$ **do**   % begin building the Krylov solution

10   **if** $\|r_{m-1}\| < \nu\|r_{m-2}\|$ % choose the next basis vector

11         $z_m = r_{m-1}/\|r_{m-1}\|_2$

12   **else**

13         $z_m = q_{m-1}$

14   **end**

15   **UpdateQR** $AZ_m = Q_m R_m$

16       $r_{m-2} = r_{m-1}$

17       $r_{m-1} = r_{m-1} - \left(q_m^\mathsf{T} r_{m-1}\right) q_m$

18       $m = m + 1$

19   **end**

20   **Solve** $R_m\, w_m = Q_m^\mathsf{T} r_{m-1}$

21     $x_m = x_0 + Z_m w_m$

22   **Return** x_m

23 **end**

1.3   Jacobian-Free Augmentation

With both Newton's method and ASGMRES explained, a Jacobian-free non-linear solver can now be discussed. Using the ASGMRES implementation discussed to solve the linear system for the Newton direction, the QR factorization in ASGMRES becomes $J_r(x_k)Z_m = Q_m R_m$. This is the only manner in which the Jacobian affects ASGMRES; it only appears as a sequence of matrix-vector products on the left-hand side of the QR factorization:

$$J_r(x_k)Z_m = [\,J_r(x_k)z_1\,,\,J_r(x_k)z_2\,,\,...\,,\,J_r(x_k)z_m\,] \tag{1.31}$$

Because each row of the Jacobian is a gradient of the associated function from $r(x_k)$, the matrix-vector product results in a dot product of the vector z_m with all of the gradients which produces a vector of directional derivative values. Therefore, the resulting product is in some sense a directional derivative and may be approximated by a finite difference:

$$J_r(x_k)z_m \approx \frac{r(x_k + \sigma\,z_m) - r(x_k)}{\sigma}, \tag{1.32}$$

where σ is a small number. Formally, the approximation is a finite application of the Gâteaux derivative [12] and can be seen as valid through a multivariate Taylor expansion of $r(x_k + \sigma\,z_m)$ about x_k

$$r(x_k + \sigma\,z_m) = r(x_k) + J_r(x_k)\sigma\,z_m + c(x_k, \sigma\,z_m). \tag{1.33}$$

Substituting this expansion into equation 1.32 sans the higher-order terms yields the first-order finite difference approximation. Therefore, the m-th column of the left-hand side of the QR factorization can be approximately computed without explicit presence of the Jacobian matrix. And because everything is being solved approximately to a tolerance anyways, one more approximation, especially one that greatly reduces the computational complexity of the solution, is not detrimental to the overall solution.

Knoll and Keyes provide a much wider and in-depth discussion of JFNK methods in [13], but only two of the items they presented will be mentioned. First, they mention a centered, second-order finite difference scheme may be used in place of equation 1.32 but note it has not gained much popularity, and indeed, any standard differencing scheme may be used as the approximation as long as the residual is defined on the stencil. The apparent aversion to such schemes may stem from the need to evaluate the residual function two or more times per loop instead of the single evaluation for the first-order scheme and due to little gain in effectiveness since σ must be adjusted according to the order of the method to balance finite difference truncation error and floating-point round-off error. The second item a discussion of a good choice of σ. If the system is scaled to around unity, a choice of around 10^{-8} is standard for finite differences since smaller values succumb to the limitation 64-bit floating-point precision. If the system is not scaled that nicely, a number of recommendations are given, such as

$$\sigma = \frac{\sqrt{(1 + \|x_k\|_2)\, \sigma_{\mathrm{mach}}}}{\|z_m\|_2}, \tag{1.34}$$

where σ_{mach} is the machine epsilon. If the norm of the expansion point is particularly large, this is the value of σ used and 10^{-8} is used for norms close to unity.

A modified ASGMRES algorithm is presented in algorithm 3. The non-linear residual function is denoted as r_{NL} in order to clearly distinguish it from the residuals of the linear solve r_m. Also, the first-order finite difference is denoted

$$Dr_{\mathrm{NL}}(z_m;\, x_k, \sigma) = \frac{r(x_k + \sigma\, z_m) - r(x_k)}{\sigma} \tag{1.35}$$

both for cleanliness and to emphasize that the first-order approximation is only a function of the Krylov vector with the expansion point and difference step size acting as a parameters. The only major change from algorithm 2 is instead of performing the matrix-matrix product AZ_m, a new matrix L is introduced that holds the matrix-vector columns generated from the

Dr_{NL} function. Also, since the Jacobian doesn't explicitly exist, the true initial linear residual $r_0 = -r_{\text{NL}}(x_K) - J_r(x_k)\delta x_k$ cannot be calculated, so the guess value is taken to be the zero vector.

Algorithm 3: Solve for a Newton update using ASGMRES with no Jacobian

---

1 **function** jfgmres(x_k, r_{NL}, ε, ν, σ)

2     $r_0 = -r_{\text{NL}}(x_k)$ % Initialization

3     $z_1 = r_0/\|r_0\|_2$

4     $L = Dr_{\text{NL}}(z_1; x_k, \sigma)$ % Store the first approximate column of $J_r(x_k)z_1$

5   **UpdateQR** $L = Q_1 R_1$

6     $r_1 = r_0 - (q_1^{\mathsf{T}} r_0)\, q_1$

7     $m = 2$

8     $r_{m-2} = r_0$

9     $r_{m-1} = r_1$

10   **while** $\|r_{m-1}\|_2 > \varepsilon$ **do**   % begin building the Krylov solution

11   **if** $\|r_{m-1}\| < \nu\|r_{m-2}\|$ % choose the next basis vector

12             $z_m = r_{m-1}/\|r_{m-1}\|_2$

13   **else**

14             $z_m = q_{m-1}$

15   **end**

16         $L = [L,\, Dr_{\text{NL}}(z_1; x_k, \sigma)]$ % augment L with the new column

17   **UpdateQR** $L = Q_m R_m$

18         $r_{m-2} = r_{m-1}$

19         $r_{m-1} = r_{m-1} - (q_m^{\mathsf{T}} r_{m-1})\, q_m$

20         $m = m + 1$

21   **end**

22   **Solve** $R_m\, w_m = Q_m^{\mathsf{T}} r_{m-1}$

23     $x_m = x_0 + Z_m w_m$

24   **Return** x_m

25 **end**

Bibliography

[1] C. Kelley. *Solving Nonlinear Equations with Newton's Method.* Fundamentals of Algorithms. Society for Industrial and Applied Mathematics, January 2003.

[2] Kenneth Eriksson, Donald Estep, and Claes Johnson. *Applied Mathematics: Body and Soul: Calculus in Several Dimensions.* Springer Science & Business Media, October 2003.

[3] Larry Armijo. Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of Mathematics*, 16(1):1–3, 1966.

[4] Y. Saad. *Iterative Methods for Sparse Linear Systems.* Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics, January 2003.

[5] G. Shanker Rao. *Mathematical Methods.* I. K. International Pvt Ltd, January 2009.

[6] Y. Saad and M. Schultz. GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, July 1986.

[7] Homer F. Walker and Lu Zhou. A simpler GMRES. *Numerical Linear Algebra with Applications*, 1(6):571–581, November 1994.

[8] P. Jiránek, M. Rozložník, and M. Gutknecht. How to Make Simpler GMRES and GCR More Stable. *SIAM Journal on Matrix Analysis and Applications*, 30(4):1483–1499, December 2008.

[9] Pavel Jiránek and Miroslav Rozložník. Adaptive version of Simpler GMRES. *Numerical Algorithms*, 53(1):93–112, July 2009.

[10] Christodoulos A. Floudas and Panos M. Pardalos. *Encyclopedia of Optimization*. Springer Science & Business Media, September 2008.

[11] Biswa Nath Datta. *Numerical Linear Algebra and Applications, Second Edition*. SIAM, 2010.

[12] Mariano Giaquinta and Stefan Hildebrandt. *Calculus of Variations I: The Lagrangian Formalism*. Springer Science & Business Media, 1996.

[13] D. A. Knoll and D. E. Keyes. Jacobian-free Newton–Krylov methods: a survey of approaches and applications. *Journal of Computational Physics*, 193(2):357–397, January 2004.