6 Nonlinear Algebraic Systems

We saw earlier that the implementation of implicit IVP solvers often requires the solution of nonlinear systems of algebraic equations. Nonlinear systems also come up in the solution of boundary values problems of ODEs and PDEs (see later sections). We now briefly discuss some available techniques for the solution of a system of equations

$$G(\boldsymbol{z}) = \boldsymbol{g}(\boldsymbol{z}) - \boldsymbol{z} = \boldsymbol{0} \tag{53}$$

since essentially any implicit method can be transformed into this form.

Example For the implicit trapezoidal (AM2) method we saw earlier that

$$\boldsymbol{g}(\boldsymbol{z}) = \boldsymbol{y}_n + \frac{h}{2}\boldsymbol{f}(t_n, \boldsymbol{y}_n) + \frac{h}{2}\boldsymbol{f}(t_{n+1}, \boldsymbol{z}).$$

Example If we were to implement the three-step Adams-Moulton formula

$$\boldsymbol{y}_{n+3} = \boldsymbol{y}_{n+2} + \frac{h}{12}\left[5\boldsymbol{f}(t_{n+3}, \boldsymbol{y}_{n+3}) + 8\boldsymbol{f}(t_{n+2}, \boldsymbol{y}_{n+2}) - \boldsymbol{f}(t_{n+1}, \boldsymbol{y}_{n+1})\right]$$

as a stand-alone method (instead of in the context of predictor-corrector methods), then we would have to find a root of $G(\boldsymbol{z}) = \boldsymbol{0}$, where

$$\boldsymbol{g}(\boldsymbol{z}) = \boldsymbol{y}_{n+2} + \frac{h}{12}\left[8\boldsymbol{f}(t_{n+2}, \boldsymbol{y}_{n+2}) - \boldsymbol{f}(t_{n+1}, \boldsymbol{y}_{n+1})\right] + \frac{5h}{12}\boldsymbol{f}(t_{n+3}, \boldsymbol{z}).$$

6.1 Functional Iteration

The functional iteration approach was already discusses in Section 1 in the context of the implicit trapezoidal rule. We simply iterate

$$\boldsymbol{z}^{[i+1]} = \boldsymbol{g}(\boldsymbol{z}^{[i]}), \qquad i = 0, 1, 2, \ldots$$

with a good initial value $\boldsymbol{z}^{[0]}$. As mentioned earlier, convergence is guaranteed by the *Banach fixed-point theorem* provided the norm of the Jacobian of $\boldsymbol{g}$ is small enough, i.e.,

$$\left\|\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{z}}\right\| < 1.$$

Remark

1. Functional iteration works rather well as long as the ODE is not stiff. For stiff equations we should use either a Newton-Raphson or modified Newton-Raphson method (see below).

2. Even for non-stiff problems it may happen that functional iteration only converges for very small stepsizes h. Again, it is better to use a Newton-Raphson method.

6.2 Newton-Raphson Iteration

In order to see how Newton-Raphson iteration applies to our standard nonlinear problem (53) we expand about an arbitrary point $\boldsymbol{z}^{[i]}$, i.e.,

$$G(\boldsymbol{z}) = \boldsymbol{g}(\boldsymbol{z}) - \boldsymbol{z} = \boldsymbol{0} \Longleftrightarrow \boldsymbol{z} = \boldsymbol{g}(\boldsymbol{z}) = \boldsymbol{g}\left(\boldsymbol{z}^{[i]} + (\boldsymbol{z} - \boldsymbol{z}^{[i]})\right)$$

or — using a Taylor expansion —

$$\boldsymbol{z} = \boldsymbol{g}(\boldsymbol{z}^{[i]}) + (\boldsymbol{z} - \boldsymbol{z}^{[i]})\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{z}}(\boldsymbol{z}^{[i]}) + \mathcal{O}\left(\|\boldsymbol{z} - \boldsymbol{z}^{[i]}\|^2\right).$$

If we now proceed as in the derivation of the standard Newton method and drop the second-order terms then we end up with

$$\boldsymbol{z} - \boldsymbol{z}^{[i]} \approx \boldsymbol{g}(\boldsymbol{z}^{[i]}) + (\boldsymbol{z} - \boldsymbol{z}^{[i]})\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{z}}(\boldsymbol{z}^{[i]}) - \boldsymbol{z}^{[i]}$$

or

$$\left(I - \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{z}}(\boldsymbol{z}^{[i]})\right)(\boldsymbol{z} - \boldsymbol{z}^{[i]}) \approx \boldsymbol{g}(\boldsymbol{z}^{[i]}) - \boldsymbol{z}^{[i]}.$$

This now motivates the iterative method

$$\boldsymbol{z}^{[i+1]} = \boldsymbol{z}^{[i]} - \left(I - \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{z}}(\boldsymbol{z}^{[i]})\right)^{-1}\left[\boldsymbol{z}^{[i]} - \boldsymbol{g}(\boldsymbol{z}^{[i]})\right], \qquad i = 0, 1, \ldots, \tag{54}$$

which is known as the *Newton-Raphson* method.

Note the similarity with the standard (scalar) Newton method

$$z^{[i+1]} = z^{[i]} - \frac{g(z^{[i]}) - z^{[i]}}{g'(z^{[i]}) - 1}, \qquad i = 0, 1, \ldots,$$

which corresponds to (using $G(z) = g(z) - z$)

$$z^{[i+1]} = z^{[i]} - \frac{G(z^{[i]})}{G'(z^{[i]})}, \qquad i = 0, 1, \ldots,$$

— the well-known formula for finding a zero of G.

Remark This is essentially the same as in the textbook. There, however, the function $\boldsymbol{g}$ is represented in the form $h\boldsymbol{g}(\boldsymbol{z}) + \boldsymbol{\beta}$.

As with the standard Newton iteration, one can show that — for a "good" starting value $\boldsymbol{z}^{[0]}$ — the iteration converges *quadratically*, i.e., the error satisfies

$$\|\boldsymbol{z} - \boldsymbol{z}^{[i+1]}\| \le c\|\boldsymbol{z} - \boldsymbol{z}^{[i]}\|^2,$$

where $\boldsymbol{z}$ is the exact root of $G(\boldsymbol{z}) = \boldsymbol{g}(\boldsymbol{z}) - \boldsymbol{z}$. In particular, if G is linear, then Newton-Raphson iteration converges in a single step (cf. the linear boundary value problems in the next section).

Example Solve

$$\begin{aligned} x^2 + y^2 &= 4 \\ xy &= 1 \end{aligned}$$

which corresponds to finding the intersection points of a circle and a hyperbola in the plane. Here

$$G(\boldsymbol{z}) = G(x,y) = \begin{bmatrix} G_1(x,y) \\ G_2(x,y) \end{bmatrix} = \begin{bmatrix} x^2 + y^2 - 4 \\ xy - 1 \end{bmatrix}$$

and

$$\frac{\partial G}{\partial \boldsymbol{z}} = J(x,y) = \begin{bmatrix} \frac{\partial G_1}{\partial x} & \frac{\partial G_1}{\partial y} \\ \frac{\partial G_2}{\partial x} & \frac{\partial G_2}{\partial y} \end{bmatrix} (x,y) = \begin{bmatrix} 2x & 2y \\ y & x \end{bmatrix}.$$

This example is illustrated in the Matlab script `run_newtonmv.m`.

Remark 1. It is apparent from formula (54) that implementation of the Newton-Raphson method is rather expensive since we need to compute and evaluate the entire Jacobian matrix $\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{z}}(\boldsymbol{z}^{[i]})$ at every iteration.

2. Moreover, each Newton-Raphson iteration requires the *inverse* of the Jacobian. This, however, corresponds to the solution of a system of linear equations — another expensive task.

In order to avoid both of these heavy computational burdens a *modified Newton-Raphson* method has been proposed.

6.3 Modified Newton-Raphson Iteration

In the *modified Newton-Raphson* method we approximate the Jacobian (which really should change in each iteration, cf. (54) by a *fixed matrix*, e.g., $\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{z}}(\boldsymbol{z}^{[0]})$. Now, we no longer need to re-compute the Jacobian in every iteration, nor do we need to solve the linear system (invert the Jacobian) in every iteration. However, quadratic convergence is lost.

Remark Other, so-called *quasi-Newton methods* exist, that are analogous to the scalar *secant method* (and thus avoid computation of the jacobian). These methods are more cheaper to use than Newton-Raphson iteration. One such method is *Broyden's method.* It can be found it some textbooks on numerical analysis, e.g., Kincaid and Cheney's "Numerical Analysis: Mathematics of Scientific Computing". A discussion of that method goes beyond the scope of this course.