Theme 3: Soft Soils and Nonlinear Equations

Martin Berggren

November 17, 2009

Nonlinear equations

- Many computational problems are inherently nonlinear
- *Ex.*: the Euler equation of gas dynamics (the aerodynamic problem discussed in the introduction) become, after discretization, a very large system of nonlinear equations in the pressures, densities, and velocities at the nodes
- Linear systems of equations: can in principle be solved by hand. Need computers for large problems
- Nonlinear equations: can rarely be solved "by hand". There is no "formula" for the solution in general! Numerical methods are needed, even in the scalar case.

Linear vs. nonlinear

- Linear systems: has a unique solution if the matrix is nonsingular
- Nonlinear equations: usually difficult to verify before computation whether a solution exists. There are often more than one solution.

Nonlinear systems

A system of n equations in n unknowns:

$$\begin{aligned} f_1(x_1,\ldots,x_n) &= 0 \\ f_2(x_1,\ldots,x_n) &= 0 \\ &\vdots \\ f_n(x_1,\ldots,x_n) &= 0 \end{aligned} \quad .$$

Can be written in vector form as

$$\mathbf{f}(\mathbf{x}) = \mathbf{0},$$

where

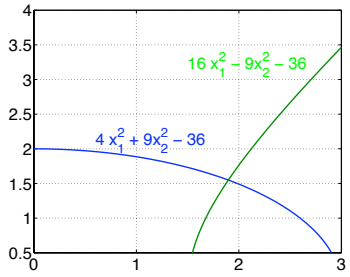
$$\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}, \qquad \mathbf{f} = \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix}.$$

Nonlinear systems, example

Example, p. 94 in book:

$$4x_1^2 + 9x_2^2 - 36 = 0$$
$$16x_1^2 - 9x_2^2 - 36 = 0$$

to be solved for $x_1 \geq 0$, $x_2 \geq 0$.



Unusual example: can be solved by hand!

Nonlinear systems

Of central importance for nonlinear systems: the **Jacobian matrix J** with components

$$J_{ij} = \frac{\partial f_i}{\partial x_j}$$

$$\mathsf{J} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & & & \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}$$

For previous example:

$$\mathsf{J} = \begin{pmatrix} 8x_1 & 18x_2 \\ 32x_1 & -18x_2 \end{pmatrix}$$

Direct vs. iterative methods

Numerical methods for equation solving are either **direct** or **iterative**

- Direct methods: the exact solution (up to roundoff) is found after a fixed, known-in-advance number of arithmetic operations. *Ex.:* Gaussian elimination for solving linear systems of equations
- Iterative methods: the algorithm produces a sequence of vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots$ that are better and better approximation of the (unknown) exact solution $\mathbf{x}$. Need to terminate algorithm when $\|\mathbf{x}_k - \mathbf{x}\|$ small enough; thus, the algorithms introduces additional errors besides rounding.
- Nonlinear equations *require* in general iterative solution methods
- Iterative methods become necessary also for very large linear system, when direct methods become too time and memory consuming

Nonlinear solvers in Matlab

- `fzero` finds a zero of a continuous function of one variable
- No solver for **systems** of nonlinear equations in "basic" Matlab
- `fsolve` solves systems of nonlinear equations using Newton's method, the prime iterative method for small to medium nonlinear systems. Available in Matlab's *optimization toolbox*, which is sold separately (available in the lab computers).

Iterative methods

Rule of thumb:
nonlinear equations or very large linear equations ⇒ iterative methods

A general iterative method:

```
x = starting guess
while stopping criterion not satisfied
    x = new guess
end
```


Iterative methods: convergence rate

- For efficiency, important *how fast* $\mathbf{x}_k$ approaches the exact solution $\mathbf{x}$: the **convergence rate**
- *Ex.:* assume that $\|\mathbf{x}_{k+1} - \mathbf{x}\| = 0.7\|\mathbf{x}_k - \mathbf{x}\|$ for each k
  - Then $\mathbf{x}_0, \mathbf{x}_1, \ldots$ converges **linearly** to $\mathbf{x}$ with **rate constant** 0.7
  - At each step, the error reduces with 30 % compared to previous step
  - $\|\mathbf{x}_k - \mathbf{x}\| = 0.7\|\mathbf{x}_{k-1} - \mathbf{x}\| = 0.7^2\|\mathbf{x}_{k-2} - \mathbf{x}\| = \cdots = 0.7^k\|\mathbf{x}_0 - \mathbf{x}\|$
  - The error never becomes zero!

Definition

The sequence $\mathbf{x}_0, \mathbf{x}_1, \ldots$ convergences **linearly** to $\mathbf{x}$ if there is a $C < 1$ such that

$$\lim_{k\to\infty} \frac{\|\mathbf{x}_{k+1} - \mathbf{x}\|}{\|\mathbf{x}_k - \mathbf{x}\|} = C$$

Note: the definition concerns the *asymptotic* convergence rate

Iterative methods: convergence rate

- *Ex.:* assume that $\|\mathbf{x}_{k+1} - \mathbf{x}\| = 2\|\mathbf{x}_k - \mathbf{x}\|^2$ for each k, and assume that $\|\mathbf{x}_0 - \mathbf{x}\| = 0.7$.
- Then $\mathbf{x}_0, \mathbf{x}_1, \ldots$ converges **quadratically** to $\mathbf{x}$
- $\|\mathbf{x}_k - \mathbf{x}\| = 2\|\mathbf{x}_{k-1} - \mathbf{x}\|^2 = 2^2\|\mathbf{x}_{k-2} - \mathbf{x}\|^4 = \cdots = (2\|\mathbf{x}_0 - \mathbf{x}\|^2)^k$
- Need $\|\mathbf{x}_0 - \mathbf{x}\| < 1$ for convergence. Very quick convergence: roughly a doubling of the number of correct digits at each iteration!

Definition

The sequence $\mathbf{x}_0, \mathbf{x}_1, \ldots$ convergences **quadratically** to $\mathbf{x}$ if there is a $C > 0$ such that

$$\lim_{k\to\infty} \frac{\|\mathbf{x}_{k+1} - \mathbf{x}\|}{\|\mathbf{x}_k - \mathbf{x}\|^2} = C$$

Note: The size of the rate constant C does not matter in the definition; $\|\mathbf{x}_0 - \mathbf{x}\| < 1$ necessary for convergence.

Newton's method for nonlinear equations

n equations $f_i(\mathbf{x}) = 0$, $i = 1, \ldots, n$, in n unknowns $\mathbf{x} = (x_1, \ldots, x_n)$.

We assume that $\mathbf{f}$ is continuously differentiable.

Assume at iteration k, $f_i(\mathbf{x}_k) \neq 0$. Want to find a **step** $\mathbf{s}_k = (s_1^{(k)}, \ldots, s_n^{(k)})$ such that $f_i(\mathbf{x}_k + \mathbf{s}_k) \approx 0$ for all i. By Taylor expansion,

$$f_i(\mathbf{x}_k + \mathbf{s}) = f_i(\mathbf{x}_k) + \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j}(\mathbf{x}_k) s_j + \ldots$$

Idea: Choose $\mathbf{s}$ so that the first two terms to the right vanish for each i:

$$f_i(\mathbf{x}_k) + \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j}(\mathbf{x}_k) s_j^{(k)} = 0, \qquad i = 1, \ldots, n$$

Newton's method for nonlinear equations

The basic algorithm:

1. Choose starting vector $\mathbf{x}_0$
2. For $k = 0, 1, \ldots$
   2.1 Solve the linear system
   $$\mathbf{J}(\mathbf{x}_k)\mathbf{s}_k = -\mathbf{f}(\mathbf{x}_k)$$
   2.2 Set
   $$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{s}_k$$

Newton's method transforms the problem of solving *one nonlinear* system into a problem of solving a *sequence* of *linear* systems.

Newton's method for nonlinear equations

- As above, let $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ be a continuously differentiable function
- Assume that there is a solution to the nonlinear system: $\mathbf{f}(\mathbf{x}_*) = \mathbf{0}$ for some $\mathbf{x}_* \in \mathbb{R}^n$

Theorem

The sequence of vectors $\mathbf{x}_0, \mathbf{x}_1, \ldots$ generated by Newton's method converges quadratically to $\mathbf{x}_*$ if

1. $\mathbf{J}(\mathbf{x}_*)$ *is nonsingular, and*
2. $\|\mathbf{x}_0 - \mathbf{x}_*\|$ *is small enough*

Properties of the basic Newton method

- Advantage: Can converge extremely quickly
- Limitations:
  1. The need to compute the Jacobian at each iteration
  2. The need to solve a linear system at each iteration
  3. The need to start close enough to the solution
  4. Iterations can break down if Jacobian becomes singular

Jacobian calculations

- A common situation: function values are obtained by using a commercial simulation software without access to the source code. The Jacobian is not available!
- The Jacobian can be *approximated* by **finite differences**:
$$J_{ij}(x_1, \ldots, x_n) = \frac{\partial f_i}{\partial x_j}(x_1, \ldots, x_n) \approx \frac{f_i(x_1, \ldots, x_j + h, \ldots, x_n) - f_i(x_1, \ldots, x_j, \ldots, x_n)}{h}$$
- h should not be too large (bad approximation of derivative) or too small (cancellation of significant digits). Good choices are problem dependent. Rule of thumb: $h \sim \sqrt{\epsilon_M}$.
- Computation of all elements in Jacobian matrix requires $n^2 + 1$ evaluation of $\mathbf{f}$. May be expensive for large problems is $\mathbf{f}$ is costly to compute

Other issues with Newton's method

- The need for starting close to he solution can be removed by using **globalization** techniques so that the method converges regardless of starting point. Globalization techniques modifies the Newton step length and/or direction when far away from the solution, typically for the first few iterates. (Limitation 3)
- There are also techniques do deal with singular Jacobians (Limitation 4)
- Common approach: start with a finite-difference approximation of the Jacobian, use **secant approximations**, based on function values $\mathbf{f}(\mathbf{x}_{k+1})$, $\mathbf{f}(\mathbf{x}_k)$ and the iterates $\mathbf{x}_{k+1}$, $\mathbf{x}_k$, to *update* approximations of the Jacobian (Limitation 1). Secant approximation reduces slightly the convergence rate (becomes "superlinear" instead of quadratic).

High-quality implementations of Newton's method are much more involved than the basic algorithm!

Fixed-point iterations

- A class of iterative methods that avoids solutions of linear systems (Limitation 2)
- Therefore of particular interest for very large problems
- Typically much slower than Newton's method
- Construction of a good fix-point method problem dependant

Fixed-point iterations

Idea:

- Rewrite $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ as the *fixed-point problem* $\mathbf{x} = \mathbf{g}(\mathbf{x})$
  - Can always be done, e. g. by $\mathbf{x} = \mathbf{x} - \mathbf{f}(\mathbf{x})$
- Define the iterative scheme

$$\mathbf{x}^{n+1} = \mathbf{g}(\mathbf{x}^n)$$

  and "hope" that the iterations converge to a fixed point of $\mathbf{g}$, constructed to be a solution to $\mathbf{f}(\mathbf{x}) = \mathbf{0}$.
- The iterations will converge if $\mathbf{g}$ is a **contraction mapping**; that is, if there is a $C < 1$ such that

$$\|\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{y})\| \leq C\|\mathbf{x} - \mathbf{y}\|$$

  for each $\mathbf{x}$, $\mathbf{y}$ in a (convex) subset of $\mathbb{R}^n$
- Thus, $\mathbf{g}$ is a contraction mapping if $\mathbf{g}(\mathbf{x})$ and $\mathbf{g}(\mathbf{y})$ are *strictly closer together* that $\mathbf{x}$ and $\mathbf{y}$.

Fixed-point iterations

- Not always easy to know whether a mapping is a contraction!
- A sufficient condition for a continuously differentiable $\mathbf{g}$ to be a contraction mapping is that

$$\|\mathbf{J}(\mathbf{x})\| < 1$$

  for each $\mathbf{x}$ in the region of interest, where $\mathbf{J}$ is the Jacobian of $\mathbf{g}$. (Note: matrix norm.)

Fixed-point iterations

- Simple, does not need solution of linear systems
- Sometimes the only resort for extremely large problem
- Construction of a contractive function $\mathbf{g}$ is highly problem dependent
- Robust versions of Newton's method often faster if applicable