Chapter 9: Nonlinear Equations

Outline

Outline

Nonlinear Equations

- Nonlinear equations:

$$r(x) = 0$$

  where $r(x) = (r_1(x), \ldots, r_n(x))^T$, $r_i : \mathbb{R}^n \to \mathbb{R}$.

- The techniques for solving nonlinear equations overlap in their motivation, analysis, and implementation with optimization techniques discussed in earlier chapters.
- There are particularly close connections with the nonlinear least-squares problem:

$$\min \quad \sum_{i=1}^{n} r_i^2(x).$$

- The differences are that in nonlinear equations, the number of equations equals the number of variable, and that we expect all equations to be satisfied at the solution, rather than just minimizing the sum of squares. This point is important because the nonlinear equations may represent physical or economic constraints such as conservation laws or consistency principles, which must hold exactly in order for the solution to be meaningful.

Newton's Method for Nonlinear Equations

Theorem 1

Suppose that $r : \mathbb{R}^n \to \mathbb{R}^n$ is continuously differentiable in some convex open set D and that x and $x + p$ are vectors in D. We then have that

$$r(x+p) = r(x) + \int_0^1 J(x+tp)p\,dt. \tag{1.1}$$

We can define a linear model $M_k(p)$ of $r(x_k + p)$ by approximating the second term on the right-hand-side of (1.1) by $J(x)p$, and writing

$$M_k(p) := r(x_k) + J(x_k)p$$

Newton's method:

$$M_k(p_k) = 0 \quad \Rightarrow \quad p_k = -J(x_k)^{-1} r(x_k).$$

Algorithm 9.1 (Newton's Method for Nonlinear Equations)

```
Choose $x_0$;
for  $k = 0, 1, 2, \ldots$
    Calculate a solution $p_k$ to the Newton equations
        $J(x_k)p_k = -r(x_k);$    (1.2)
    $x_{k+1} = x_k + p_k$;
end(for)
```

Newton's Method for Nonlinear Equations

Theorem 2

Suppose that r is continuously differentiable in a convex open set $\mathcal{D} \subset \mathbb{R}^n$. Let $x^* \in \mathcal{D}$ be a nondegenerate solution of $r(x) = 0$, and let $\{x_k\}$ be the sequence of iterates generated by Algorithm 9.1. Then when $x_k \in \mathcal{D}$ is sufficiently close to x^*, we have

$$x_{k+1} - x^* = o(\|x_k - x^*\|),$$

indicating local Q-superlinear convergence. When r is Lipschitz continuously differentiable near x^*, we have for all x_k sufficiently close to x^* that

$$x_{k+1} - x^* = O(\|x_k - x^*\|^2),$$

indicating local Q-quadratic convergence.

Newton's Method for Nonlinear Equations

When the iterate x_k is close to a nondegenerate root x^*, Newton's method can be quadratically convergent. Potential shortcomings of the method include the following.

- When $J(x_k)$ is singular, the Newton step may not even be defined.
- First-derivative information (the Jacobian matrix J) may be difficult to obtain.
- It may be too expensive to find and calculate the Newton step p_k exactly when n is large.
- The root x^* may be degenerate, that is, $J(x^*)$ may be singular. Consider $r(x) = x^2$, which has a single degenerate root at $x^* = 0$. Algorithm 9.1, when started from any nonzero x_0, generates the sequence of iterates

$$x_k = \frac{1}{2^k} x_0,$$

which converges to the solution 0, but only at a linear rate.

Inexact Newton Methods

Inexact Newton methods use search directions p_k that satisfy the condition

$$\|r_k + J_k p_k\| \le \eta_k \|r_k\|, \qquad \text{for some } \eta_k \in [0, \eta], \tag{1.3}$$

where $\eta \in [0, 1)$ is a constant and $\{\eta_k\}$ are referred as the forcing sequence.

Algorithm 9.2 Inexact Newton for Nonlinear Equations

```
Given $\eta \in [0,1)$; Choose $x_0$;
for $k = 0, 1, 2, \ldots$
    Choose forcing parameter $\eta_k \in [0, \eta]$;
    Find a vector $p_k$ that satisfies (1.3);
    $x_{k+1} \leftarrow x_k + p_k$;
end(for)
```

Inexact Newton Methods

- Under mild conditions, $\{x_k\}$ generated by inexact Newton method converges to x^*
  - Q-linearly, if η is sufficiently small;
  - Q-superlinearly, if $\eta_k \to 0$;
  - Q-quadratically, if $\eta_k = O(\|r_k\|)$.
- The most important methods in this class make use of iterative techniques for solving linear systems of the form $Jp = -r$, such as GMRES or ther Krylov-space methods. These methods typically require us to perform a matrix-vector multiplication of the form Jd for some d at each iteration, and to store a number of work vectors of length n.

Broyden's Method

- Construct approximation to the Jacobian $J(x)$.
- At kth iteration, use the model

$$M_k(p) = r(x_k) + B_k p$$

- Setting $M_k(p) = 0$ yields $p_k = -B_k^{-1} r_k$.
- Update B_{k+1} to satisfy the secant equation:

$$y_k = B_{k+1} s_k,$$

  where $s_k = x_{k+1} - x_k$ and $y_k = r(x_{k+1}) - r(x_k)$.
- The most successful practical algorithm is Broyden's method, for which the update formula is

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k) s_k^T}{s_k^T s_k}. \tag{1.4}$$

- Under mild conditions, Broyden's method is Q-superlinearly convergent.

Broyden's Method

Algorithm 9.3 Broyden's Method for Nonlinear Equations

```
Choose $x_0$ and a nonsingular initial Jacobian approximation $B_0$;
for $k = 0, 1, 2, \ldots$
    Calculate a solution $p_k$ to the linear equations
        $$B_k p_k = -r(x_k);$$
    $x_{k+1} \leftarrow x_k + p_k$;
    $s_k \leftarrow x_{k+1} - x_k$; $y_k = r(x_{k+1}) - r(x_k)$;
    Obtain $B_{k+1}$ from the formula (1.4);
end(for)
```

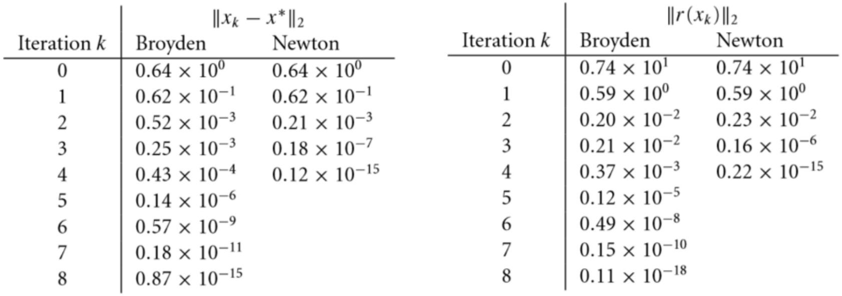
Comparison Between Newton's and Broyden's Method

Consider $r(x) = 0$ with r defined by

$$r(x) = \begin{bmatrix} (x_1 + 3)(x_2^3 - 7) + 18 \\ \sin(x_2 e^{x_1} - 1) \end{bmatrix}$$

It has a nondegenerate root at $x^* = (0, 1)^T$. We test both algorithms from the starting point $x_0 = (-0.5, 1.4)^T$, and use the exact Jacobian $J(x_0)$ as the initial Jacobian approximation B_0.

Comparison Between Newton's and Broyden's Method



Iteration k	$\|x_k - x^*\|_2$ Broyden	$\|x_k - x^*\|_2$ Newton
0	0.64×10^{0}	0.64×10^{0}
1	0.62×10^{-1}	0.62×10^{-1}
2	0.52×10^{-3}	0.21×10^{-3}
3	0.25×10^{-3}	0.18×10^{-7}
4	0.43×10^{-4}	0.12×10^{-15}
5	0.14×10^{-6}	
6	0.57×10^{-9}	
7	0.18×10^{-11}	
8	0.87×10^{-15}	

Iteration k	$\|r(x_k)\|_2$ Broyden	$\|r(x_k)\|_2$ Newton
0	0.74×10^{1}	0.74×10^{1}
1	0.59×10^{0}	0.59×10^{0}
2	0.20×10^{-2}	0.23×10^{-2}
3	0.21×10^{-2}	0.16×10^{-6}
4	0.37×10^{-3}	0.22×10^{-15}
5	0.12×10^{-5}	
6	0.49×10^{-8}	
7	0.15×10^{-10}	
8	0.11×10^{-18}	

Figure 1: Convergence of Iterates and Function Norms in Broyden and Newton Methods

Outline

Merit Functions

- The Newton and Broyden methods are local. Methods with unit step lengths can be guaranteed to converge when the starting point is close to the solution.
- Sometimes, components of the unknown or function vector or the Jacobian will blow up.
- Another kind of behavior is cycling, where the iterates move between distinct regions of the parameter space without approaching a root. For example, nonlinear equations with

  $$r(x) = -x^5 + x^3 + 4x$$

  It has five nondegenerate roots. When started from $x_0 = 1$, Newton's method produces a sequence of iterates that oscillates between 1 and -1.

Merit Functions

- They can be make more robust by using line-search and trust-region techniques, with better global convergence behavior.
- We need to define a *merit function*, which is a scalar-valued function of x that indicates whether a new iterate is better or worse than the current iterate, in the sense of making progress toward a root of r.
- The most widely used merit function is the sum of squares, defined by

$$f(x) = \frac{1}{2}\|r(x)\|^2$$

Any root x^* of r obviously has $f(x^*) = 0$, and since $f(x) \geq 0$ for all x, each root is a minimizer of f.

Merit Function

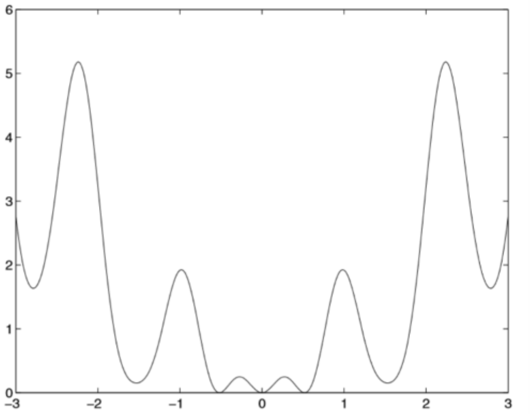


Figure 2: Plot of $\frac{1}{2}[\sin(5x) - x]^2$

Merit Functions

- Local minimizers of the merit function f are not roots of r if f is strictly positive at the point in question. Actually, since

$$\nabla f(x^*) = J(x^*)^T r(x^*) = 0,$$

  we have $r(x^*) \neq 0$ only if $J(x^*)$ is singular.

- Still, this sum-of-squares merit function has been used successfully in many applications and implemented in a number of software packages.

- Other merit functions are also used in practice, such as l_1 norm merit function:

$$f_1(x) = \|r(x)\|_1 = \sum_{i=1}^{m} |r_i(x)|.$$

Line Search Methods

- Global convergence properties can be obtained by applying the line search approach to the sum-of-squares merit function $f(x) = \frac{1}{2}\|r(x)\|^2$.
- When it is well defined, the Newton's step satisfying

$$J(x_k)p_k = -r_k$$

is a descent direction of f whenever $r_k \neq 0$, since

$$p_k^T \nabla f(x_k) = p_k^T J_k^T r_k = -\|r_k\|^2 < 0.$$

- Then step length α_k can be chosen by performing line search, and the iterates are defined by

$$x_{k+1} = x_k + \alpha_k p_k, \quad k = 0, 1, 2, \ldots \tag{2.1}$$

Modification

- To prevent the near-singularity of J_k, we define p_k to be
$$p_k = -(J_k^T J_k + \lambda_k I)^{-1} J_k^T r_k \tag{2.2}$$
- This approach is analogous to the classical Levenberg-Marquardt algorithm.
- p_k can be obtained by solving the linear least-squares problem
$$\min_p \frac{1}{2} \left\| \begin{bmatrix} J \\ \sqrt{\lambda} I \end{bmatrix} p + \begin{bmatrix} r \\ 0 \end{bmatrix} \right\|^2 .$$

Algorithm 9.4 Line Search Newton-like Method

```
Initialization;
for  $k = 0, 1, 2, \ldots$
    Calculate a Newton-like step from (1.2) (or (2.2) if $J_k$ appears to be near-singular);
    Find $\alpha_k$ satisfying the Wolfe conditions;
    $x_{k+1} \leftarrow x_k + \alpha_k p_k$;
endfor
```

Trust-Region Methods

- Use the merit function $f(x) = \frac{1}{2}\|r(x)\|^2$.
- Define the trust-region subproblem

$$\begin{aligned} \min \quad & m_k(p) = \tfrac{1}{2}\|J_k p + r_k\|^2 && \text{(2.3a)} \\ s.t. \quad & \|p\| \le \Delta_k. && \text{(2.3b)} \end{aligned}$$

- Define the reduction ratio

$$\rho_k = \frac{\|r(x_k)\|^2 - \|r(x_k + p_k)\|^2}{\|r(x_k)\|^2 - \|r(x_k) + J_k p_k\|^2}$$

Outline

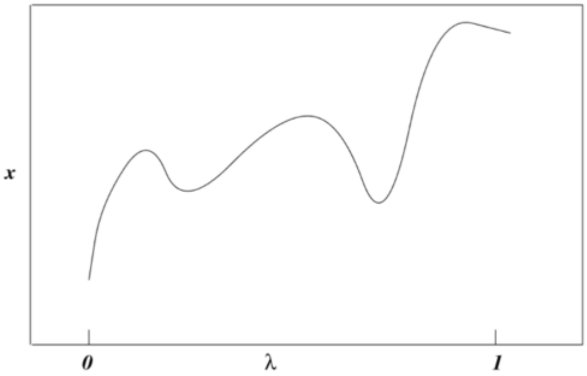
Motivation

- Newton-based methods all suffer from one shortcoming: they are in danger of converging to a local minimum of the merit function rather that is not a solution of $r(x) = 0$.
- Rather than dealing with $r(x) = 0$ directly, we set up an "easy" system of equations for which the solution is obvious. We then gradually transform the easy system into the original system $r(x)$, and follow the solution as it moves from the solution of the easy problem to the solution of the original problem.
- *Homotopy* map $H(x, \lambda)$ is defined as

$$H(x, \lambda) = \lambda r(x) + (1 - \lambda)(x - a).$$

- Gradually increasing the parameter λ from 0 to 1 to solve $r(x) = 0$ eventually.

Illustration



Outline

Notes and References

Nonlinear differential equations and integral equations are a rich source of nonlinear equations. When formulated as finite-dimensional nonlinear equations,the unknown vector x is a discrete approximation to the (infinite-dimensional) solution.

Thanks for your attention!