Lectures on Scientific Computing

Lise-Marie Imbert-Gerard

10/2/18

1 Software

1.1 Programming for Performance

To program for proper performance:

- Design test
- Trade offs: Speed/clarity/numerical stability
- Timing your code. Matlab has a built in code profiler which can help find computational bottlenecks

2 Chapter 4: Nonlinear Equations

Reference for this chapter: Nocedal-Wright "Numerical Optimization" Chapter 11 (Link on Prof. I-G's Website)

We wish to examine the problem: For $r : \mathbb{R}^n \to \mathbb{R}^n$ smooth, find $x_* \in \mathbb{R}^n$ such that $r(x_*) = 0$. We say that x_* is a solution or a root of the equation $r(x) = 0$.

2.1 Example

$$r(x) = \begin{bmatrix} x_2^2 - 1 \\ sin(x_1) - x_2 \end{bmatrix}$$
$$x_* = \begin{bmatrix} \frac{3\pi}{2} \\ -1 \end{bmatrix}$$
$$x_* = \begin{bmatrix} \frac{\pi}{2} \\ 1 \end{bmatrix}$$

Hypothesis: r is continuously differentiable on $mathcalD \in \mathbb{R}^n$. $J(x)$, the Jacobian of r exists and is continuous on $\mathcal{D}$, the domain. x_* is a degenerate solution if $J(x_*) = 0$ and $r(x_*)$. It is non-degenerate otherwise.

2.2 Local Algorithms

For many of these methods, we start with an iterative method and attempt to find a convergence if it exists.

For local algorithms, they converge if your initial case is close enough to the actual solution (like Newton's Method).

2.2.1 Newton's Method

Given some iterate x_k, you find the value of $r(x_k)$ and find the tangent. Then you replace the function with its tangent meaning x_{k+1} =the tangent at $r(x_k)$.

Theorem:

Suppose x and p are in $\mathcal{D}$. Then $r(x+p) = r(x) + \int_0^1 J(x+tp)pdt$.

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

Definition:

Newton's Method, in its pure form, chooses step p_k to be such that $M_k(p_k) = 0 \rightarrow p_k = -J(x_k)^{-1}r(x_k)$

Algorithm:

Choose x_0 for $x \in \mathbb{N}$. Calculate the solution p_k of $J(x_k)p_k = -r(x_k)$. Then

$$x_{k+1} = x_k + p_k$$

Remark: If x_0 is far from x_*, the algorithm may behave erratically. Also note that Newton's method is dependent on the Jacobian and calculating derivatives may be complicated. If n is large this method may also be expensive since calculating p_k will be expensive. If $J(x_*)$ is singular you can also run into problems.

1D Example:

$$\begin{aligned} r(x) &= x^2 \\ J(x) &= 2x \\ \forall x_0 &\neq 0 \\ \therefore x_k &= \frac{1}{2^k}x_0 \end{aligned}$$

This converges only linearly, which is a problem for Newton's method.

Theorem:

Let $\mathcal{D}$ be a convex open set. Let $x_* \in \mathcal{D}$ be a non-degenerate solution of $r(x) = 0$ and x_k be the sequence generated by the Newton's Method Algorithm.

- When $x_k \in \mathcal{D}$ is sufficiently close to x_*, we have $x_{k+1} - x_* = \mathcal{O}(||x_k - x_*||)$ indicating local superlinear Q Convergence
- When R is Lipschitz continuously differentiable in the neighborhood of x_* then for all x_k sufficiently close to x_* you will have local Q-quadratic convergence. i.e. $x_{k+1} - x_* = \mathcal{O}(||x_k - x_*||^2)$

As a reminder, the Lipschitz hypothesis is $||J(x_1 - J(x_0))|| \leq \beta_L ||x_0 - x_1|| \forall x_1, x_0 \in \mathcal{D}$.

Proof:

$$
\begin{aligned}
r(x_k) = r(x_k) r(x_*) &= \int_0^1 J(x_k + t(x_* - x_k))(x_k - x_*) dt \\
&= J(x_k)(x_k - x_*) + \int_0^1 (J(x_k + t(x_* - x_k)) - J(x_k))(x_k - x_*) dt
\end{aligned}
$$