Gradient-based Methods for Optimization. Part I.

Prof. Nathan L. Gibson

Department of Mathematics

Oregon State University

Applied Math and Computation Seminar
October 21, 2011

Outline

- Unconstrained Optimization
- Newton's Method
    - Inexact Newton
    - Quasi-Newton
- Nonlinear Least Squares
- Gauss-Newton Method
- Steepest Descent Method
- Levenberg-Marquardt Method

Unconstrained Optimization

- Minimize function f of N variables
- I.e., find *local minimizer* x^* such that

$$f(x^*) \leq f(x) \text{ for all } x \text{ near } x^*$$

- Different from *constrained optimization*

$$f(x^*) \leq f(x) \text{ for all } x \in U \text{ near } x^*$$

- Different from *global minimizer*

$$f(x^*) \leq f(x) \text{ for all } x \text{ (possibly in } U\text{)}$$

Sample Problem

Parameter Identification
Consider

$$u'' + cu' + ku = 0; u(0) = u_0; u'(0) = 0 \tag{1}$$

where u represents the motion of an unforced harmonic oscillator (e.g., spring). We may assume u_0 is known, and data $\{u_j\}_{j=1}^{M}$ is given for some times t_j on the interval $[0, T]$.
Now we can state a *parameter identification* problem to be: find $x = [c, k]^T$ such that the solution $u(t)$ to (1) using parameters x is (as close as possible to) u_j when evaluated at times t_j.

Objective Function

Consider the following formulation of the Parameter Identification problem: Find $x{=}[c,k]^T$ such that the following objective function is minimized:

$$f(x) = \frac{1}{2}\sum_{j=1}^{M} |u(t_j;x) - u_j|^2 \,.$$

This is an example of a *nonlinear least squares problem*.

Iterative Methods

An iterative method for minimizing a function $f(x)$ usually has the following parts:

- Choose an initial iterate x_0
- For $k = 0, 1, \dots$
  - If x_k optimal, **stop**.
  - Determine a search direction d and a step size λ
  - Set $x_{k+1} = x_k + \lambda d$

Convergence Rates

The sequence $\{x_k\}_{k=1}^{\infty}$ is said to converge to x^* with rate p and rate constant C if

$$\lim_{k\to\infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^p} = C.$$

- **Linear**: $p = 1$ and $0 < C < 1$, such that error decreases.
- **Quadratic**: $p = 2$, doubles correct digits per iteration.
- **Superlinear**: If $p = 1$, $C = 0$. Faster than linear. Includes quadractic convergence, but also intermediate rates.

Necessary Conditions

Theorem

Let f be twice continuously differentiable, and let x^* be a local minimizer of f. Then

$$\nabla f(x^*) = 0 \tag{2}$$

and the Hessian *of f, $\nabla^2 f(x^*)$, is* positive semidefinite*.*

Recall A *positive semidefinite* means

$$x^T A x \geq 0 \quad \forall x \in \mathbb{R}^N.$$

Equation (2) is called the *first-order necessary condition*.

Hessian

Let $f : \mathbb{R}^N \rightarrow \mathbb{R}$ be twice continuously differentiable ($\mathcal{C}^2$), then

- The **gradient** of f is

$$\nabla f = \left[\frac{\partial f}{\partial x_1}, \cdots, \frac{\partial f}{\partial x_N} \right]^T$$

- The **Hessian** of f is

$$\nabla^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_N \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_N^2} \end{bmatrix}$$

Sufficient Conditions

Theorem

Let f be twice continuously differentiable in a neighborhood of x^*, and let

$$\nabla f(x^*) = 0$$

and the Hessian *of f, $\nabla^2 f(x^*)$, be* positive semidefinite*. Then x^* is a local minimizer of f.*

Note: second derivative information is required to be certain, for instance, if $f(x) = x^3$.

Quadratic Objective Functions

Suppose

$$f(x) = \frac{1}{2}x^T Hx - x^T b$$

then we have that

$$\nabla^2 f(x) = H$$

and if H is symmetric (assume it is)

$$\nabla f(x) = Hx - b.$$

Therefore, if H is positive definite, then the unique minimizer x^* is the solution to

$$Hx^* = b.$$

Newton's Method

Newton's Method solves for the minimizer of the *local quadratic model* of f about the current iterate x_k given by

$$m_k(x) = f(x_k) + \nabla f(x_k)^T(x - x_k) + \frac{1}{2}(x - x_k)^T\nabla^2 f(x_k)(x - x_k).$$

If $\nabla^2 f(x_k)$ is positive definite, then the minimizer x_{k+1} of m_k is the unique solution to

$$0 = \nabla m_k(x) = \nabla f(x_k) + \nabla^2 f(x_k)(x - x_k). \tag{3}$$

Newton Step

The solution to (3) is computed by solving

$$\nabla^2 f(x_k) s_k = -\nabla f(x_k)$$

for the Newton Step s_k^N. Then the Newton update is defined by

$$x_{k+1} = x_k + s_k^N.$$

Note: the step s_k^N has both direction and length. Variants of Newton's Method modify one or both of these.

Standard Assumptions

Assume that f and x^* satisfy the following

1. Let f be twice continuously differentiable and Lipschitz continuous with constant γ

$$\|\nabla^2 f(x) - \nabla^2 f(y)\| \leq \gamma \|x - y\|.$$

2. $\nabla f(x^*) = 0.$
3. $\nabla^2 f(x^*)$ is *positive definite*.

Convergence Rate

Theorem

Let the Standard Assumptions hold. Then there exists a $\delta > 0$ *such that if* $x_0 \in \mathcal{B}_\delta(x^*)$, *the Newton iteration converges quadratically to* x^*.

- I.e., $\|e_{k+1}\| \leq K\|e_k\|^2$.
- If x_0 is not close enough, Hessian may not be positive definite.
- If you start close enough, you stay close enough.

Problems (and solutions)

- Need derivatives
  - Use finite difference approximations
- Needs solution of linear system at each iteration
  - Use iterative linear solver like CG (Inexact Newton)
- Hessians are expensive to find (and solve/factor)
  - Use chord (factor once) or Shamanskii
  - Use Quasi-Newton (update H_k to get H_{k+1})
  - Use Gauss-Newton (first order approximate Hessian)

Nonlinear Least Squares

Recall,

$$f(x) = \frac{1}{2} \sum_{j=1}^{M} \left| u(t_j; x) - u_j \right|^2 .$$

Then for $x = [c, k]^T$

$$\nabla f(x) = \begin{bmatrix} \sum_{j=1}^{M} \frac{\partial u(t_j;x)}{\partial c} \left(u(t_j; x) - u_j \right) \\ \sum_{j=1}^{M} \frac{\partial u(t_j;x)}{\partial k} \left(u(t_j; x) - u_j \right) \end{bmatrix} = R'(x)^T R(x)$$

where $R(x) = [u(t_1; x) - u_1, \ldots, u(t_M; x) - u_M]^T$ is called the *residual* and $R'_{ij}(x) = \frac{\partial R_i(x)}{\partial x_j}$.

Approximate Hessian

In terms of the residual R, the Hessian of f becomes

$$\nabla^2 f(x) = R'(x)^T R'(x) + R''(x)R(x)$$

where $R''(x)R(x) = \sum_{j=1}^{M} r_j(x)\nabla^2 r_j(x)$ and $r_j(x)$ is the jth element of the vector $R(x)$.

The second order term requires the computation of M Hessians, each size $N \times N$. However, if we happen to be solving a *zero residual problem*, this second order term goes to zero. One can argue that for *small residual problems* (and good initial iterates) the second order term is neglibible.

Gauss-Newton Method

The equation defining the Newton step

$$\nabla^2 f(x_k) s_k = -\nabla f(x_k)$$

becomes

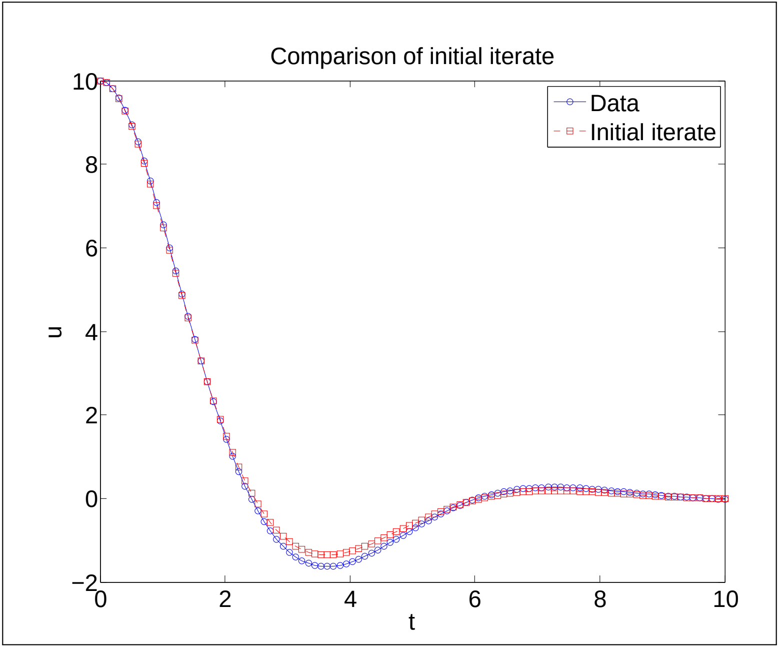
$$\begin{aligned} R'(x_k)^T R'(x_k) s_k &= -\nabla f(x_k) \\ &= -R'(x_k)^T R(x_k). \end{aligned}$$

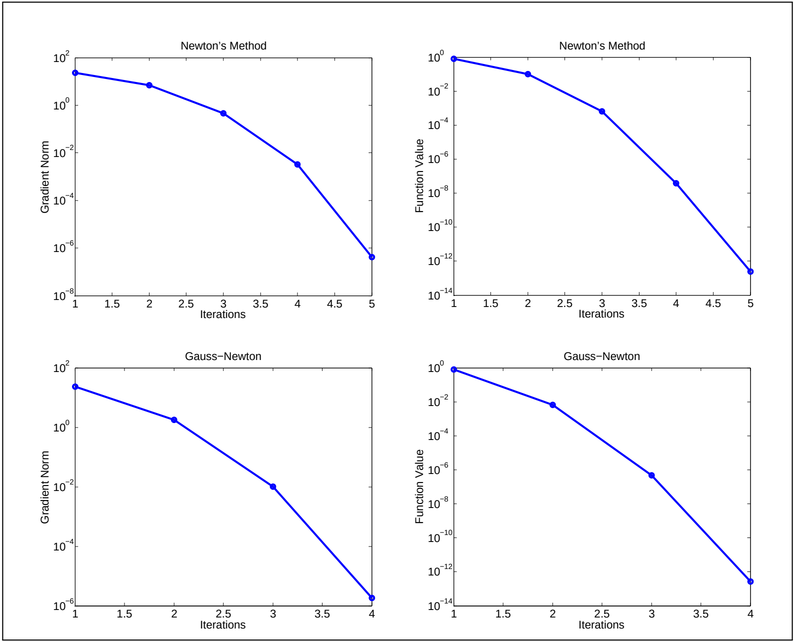
We define the Gauss-Newton step as the solution s_k^{GN} to this equation.

You can expect close to *quadratic* convergence for small residual problems. Otherwise, not even *linear* is guaranteed.

Numerical Example

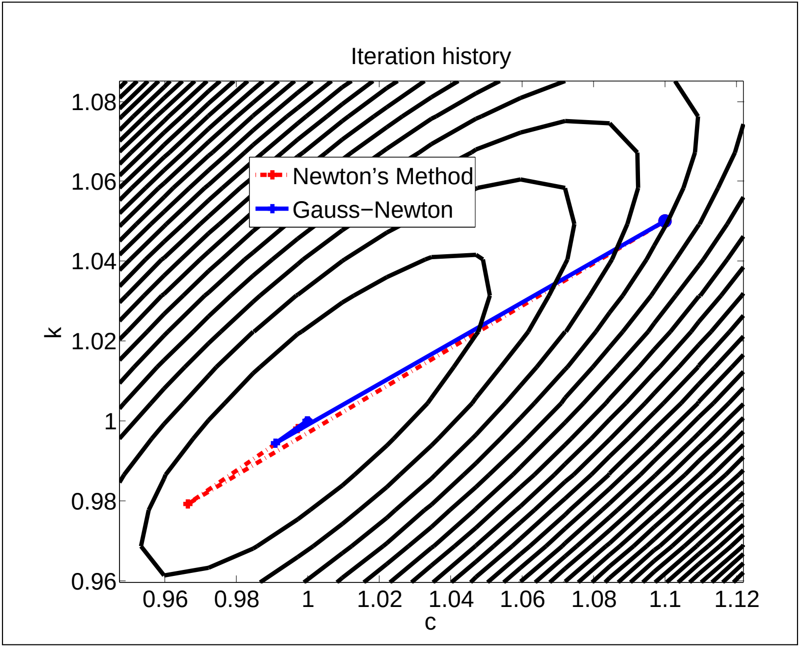
- Recall
$$u'' + cu' + ku = 0; u(0) = u_0; u'(0) = 0.$$
- Let the true parameters be $x^* = [c, k]^T = [1, 1]^T$. Assume we have $M = 100$ data u_j from equally spaced time points on $[0, 10]$.
- We will use the initial iterate $x_0 = [1.1, 1.05]^T$ with Newton's Method and Gauss-Newton.
- We compute gradients with forward differences, analytical 2×2 matrix inverse, and use `ode15s` for time stepping the ODE.

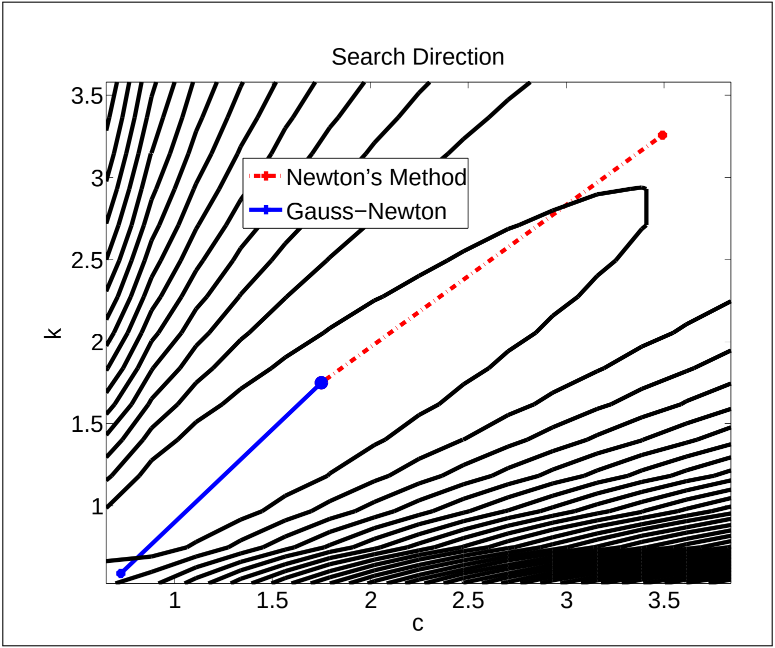

Comparison of initial iterate
Data
Initial iterate
u
t
10
8
6
4
2
0
−2
0
2
4
6
8
10

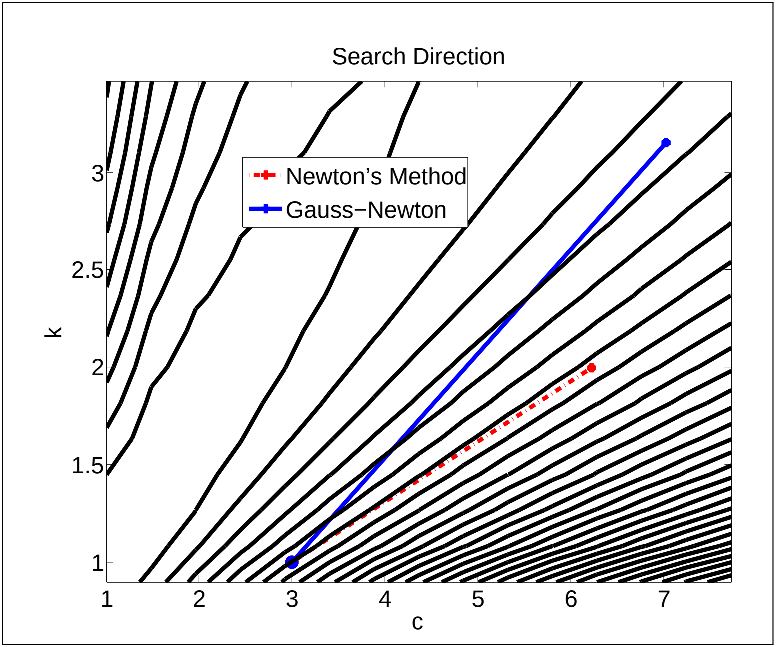

Newton's Method
Gradient Norm
10^{2}
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
1
1.5
2
2.5
3
3.5
4
4.5
5
Iterations
Newton's Method
Function Value
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
10^{-12}
10^{-14}
1
1.5
2
2.5
3
3.5
4
4.5
5
Iterations
Gauss−Newton
Gradient Norm
10^{2}
10^{0}
10^{-2}
10^{-4}
10^{-6}
1
1.5
2
2.5
3
3.5
4
Iterations
Gauss−Newton
Function Value
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
10^{-12}
10^{-14}
1
1.5
2
2.5
3
3.5
4
Iterations

	Newton		Gauss-Newton	
k	$\|\|\nabla f(x_k)\|\|$	$f(x_k)$	$\|\|\nabla f(x_k)\|\|$	$f(x_k)$
0	2.330e+01	7.881e-01	2.330e+01	7.881e-01
1	6.852e+00	9.817e-02	1.767e+00	6.748e-03
2	4.577e-01	6.573e-04	1.016e-02	4.656e-07
3	3.242e-03	3.852e-08	1.844e-06	2.626e-13
4	4.213e-07	2.471e-13		

Table: Parameter identification problem, locally convergent iterations. CPU time Newton: 3.4s, Gauss-Newton: 1s.


Iteration history
Newton's Method
Gauss−Newton
k
c
1.08
1.06
1.04
1.02
1
0.98
0.96
0.96
0.98
1
1.02
1.04
1.06
1.08
1.1
1.12


Search Direction
Newton's Method
Gauss−Newton
k
c
3.5
3
2.5
2
1.5
1
1
1.5
2
2.5
3
3.5

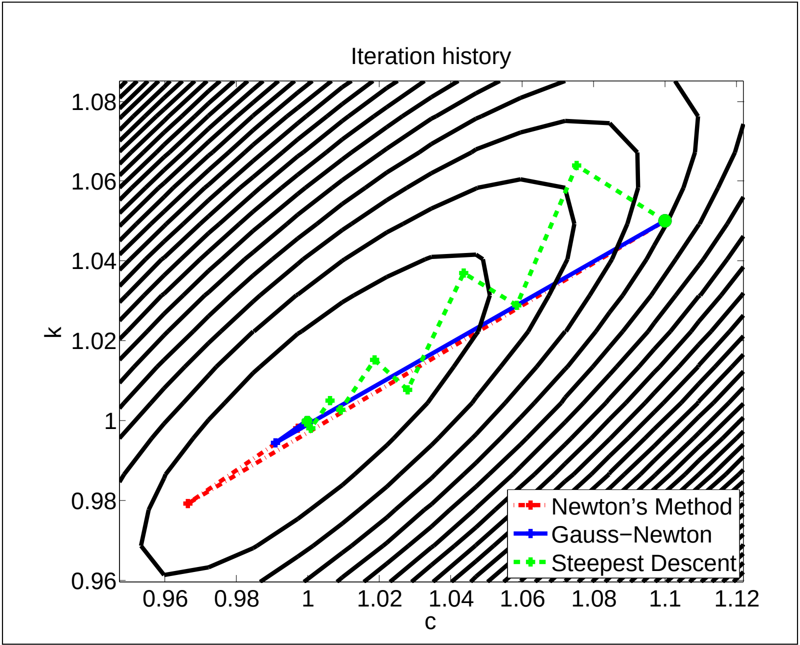

Search Direction
Newton's Method
Gauss−Newton
k
3
2.5
2
1.5
1
c
1
2
3
4
5
6
7

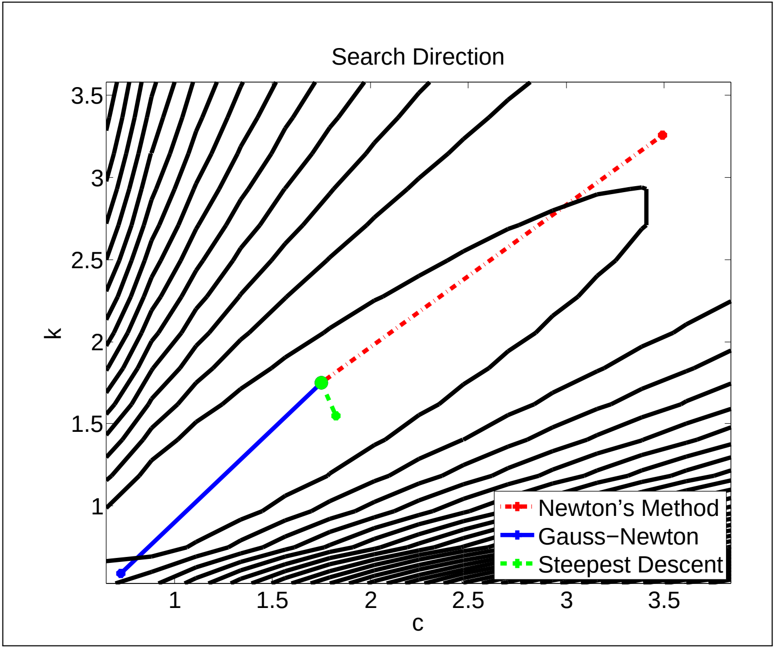
Global Convergence

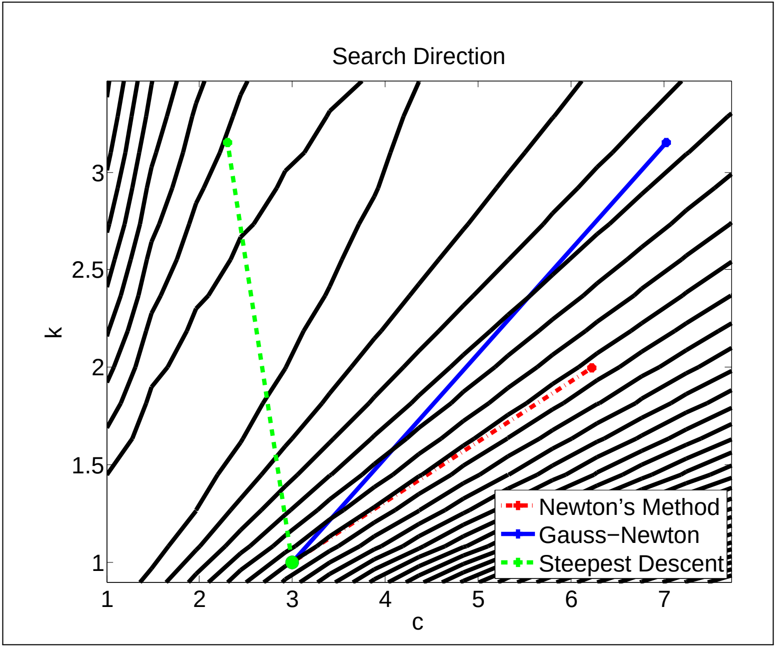
- Newton direction may not be a descent direction (if Hessian not positive definite).
- Thus Newton (or any Newton-based method) may increase f if x_0 not close enough. Not *globally convergent*.
- Globally convergent methods ensure (sufficient) decrease in f.
- The *steepest descent* direction is always a descent direction.

Steepest Descent Method

- We define the *steepest descent direction* to be $d_k = -\nabla f(x_k)$. This defines a direction but not a step size.
- We define the Steepest Descent update step to be $s_k^{SD} = \lambda_k d_k$ for some $\lambda_k > 0$.
- We will talk later about ways of choosing λ.


Iteration history
1.08
1.06
1.04
1.02
1
0.98
0.96
k
0.96
0.98
1
1.02
1.04
1.06
1.08
1.1
1.12
c
Newton's Method
Gauss−Newton
Steepest Descent


Search Direction
3.5
3
2.5
2
1.5
1
k
1
1.5
2
2.5
3
3.5
c
Newton's Method
Gauss−Newton
Steepest Descent


Search Direction
k
3
2.5
2
1.5
1
c
1
2
3
4
5
6
7
Newton’s Method
Gauss−Newton
Steepest Descent

Steepest Descent Comments

- Steepest Descent direction is best direction *locally*.
- The negative gradient is perpendicular to level curves.
- Solving for s_k^{SD} is equivalent to assuming $\nabla^2 f(x_k) = I/\lambda_k$.
- In general you can only expect *linear* convergence.
- Would be good to combine global convergence property of Steepest Descent with *superlinear* convergence rate of Gauss-Newton.

Levenberg-Marquardt Method

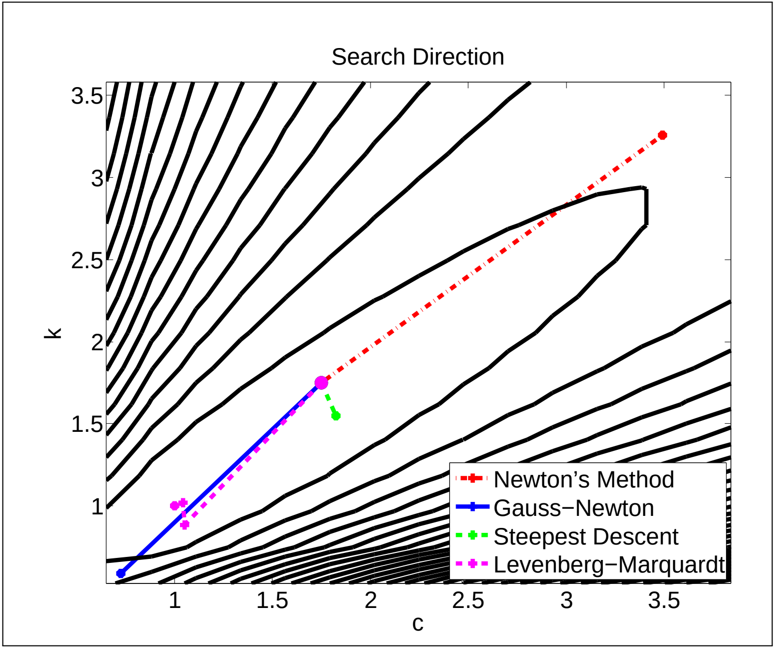
Recall the objective function

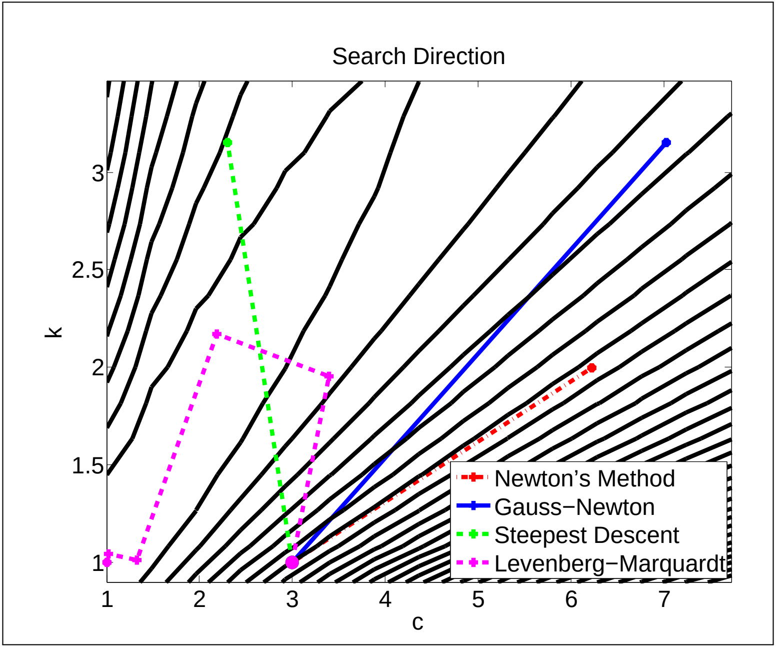
$$f(x) = \frac{1}{2} R(x)^T R(x)$$

where R is the residual. We define the Levenberg-Marquardt update step s_k^{LM} to be the solution of

$$\left(R'(x_k)^T R'(x_k) + \nu_k I \right) s_k = -R'(x_k)^T R(x_k)$$

where the *regularization parameter* ν_k is called the Levenberg-Marquardt parameter, and it is chosen such that the approximate Hessian $R'(x_k)^T R'(x_k) + \nu_k I$ is positive definite.


Search Direction
k
c
3.5
3
2.5
2
1.5
1
1
1.5
2
2.5
3
3.5
Newton’s Method
Gauss−Newton
Steepest Descent
Levenberg−Marquardt


Search Direction
k
3
2.5
2
1.5
1
c
1
2
3
4
5
6
7
Newton's Method
Gauss−Newton
Steepest Descent
Levenberg−Marquardt

Levenberg-Marquardt Notes

- Robust with respect to poor initial conditions and larger residual problems.
- Varying ν involves interpolation between GN direction ($\nu = 0$) and SD direction (large ν).
- See

```
doc lsqnonlin
```

for MATLAB instructions for LM and GN.

Levenberg-Marquardt Idea

- If iterate is not close enough to minimizer so that GN does not give a descent direction, increase ν to take more of a SD direction.
- As you get closer to minimizer, decrease ν to take more of a GN step.
  - For zero-residual problems, GN converges quadratically (if at all)
  - SD converges linearly (guaranteed)

LM Alternative Perspective

- Approximate Hessian may not be positive definite (or well-conditioned), increase ν to add regularity.
- As you get closer to minimizer, Hessian will become positive definite (by Standard Assumptions). Decrease ν, as less regularization is necessary.
- Regularized problem is "nearby problem", want to solve actual problem as soon as is feasible.

Summary

- Taylor series with remainder:
$$f(x) = f(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{1}{2}(x - x_k)^T \nabla^2 f(\xi)(x - x_k)$$

- Newton:
$$m_k^N(x) = f(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{1}{2}(x - x_k)^T \nabla^2 f(x_k)(x - x_k)$$

- Gauss-Newton:
$$m_k^{GN}(x) = f(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{1}{2}(x - x_k)^T R'(x_k)^T R'(x_k)(x - x_k)$$

- Steepest Descent:
$$m_k^{SD}(x) = f(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{1}{2}(x - x_k)^T \frac{1}{\lambda_k} I(x - x_k)$$

- Levenberg-Marquardt:
$$m_k^{LM}(x) = f(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{1}{2}(x - x_k)^T \left(R'(x_k)^T R'(x_k) + \nu_k I\right)(x - x_k)$$

References

1. Levenberg, K., “A Method for the Solution of Certain Problems in Least-Squares”, Quarterly Applied Math. 2, pp. 164-168, 1944.
2. Marquardt, D., “An Algorithm for Least-Squares Estimation of Nonlinear Parameters”, SIAM Journal Applied Math., Vol. 11, pp. 431-441, 1963.
3. Moré, J. J., “The Levenberg-Marquardt Algorithm: Implementation and Theory”, Numerical Analysis, ed. G. A. Watson, Lecture Notes in Mathematics 630, Springer Verlag, 1977.
4. Kelley, C. T., “Iterative Methods for Optimization”, Frontiers in Applied Mathematics 18, SIAM, 1999. http://www4.ncsu.edu/∼ctk/matlab_darts.html.
5. Wadbro, E., “Additional Lecture Material”, Optimization 1 / MN1, Uppsala Universitet, http://www.it.uu.se/edu/course/homepage/opt1/ht07/.