Gradient-based Methods for Optimization

Prof. Nathan L. Gibson

Department of Mathematics

Oregon State University

Applied Math and Computation Seminar
February 23, 2018

Outline

- Unconstrained Optimization
- Nonlinear Least Squares
- Newton's Method
  - Inexact Newton
  - Quasi-Newton
- Gauss-Newton Method
- Steepest Descent Method
- Levenberg-Marquardt Method
- Line Search (Armijo Rule)
  - Damped Gauss-Newton
- Trust Region

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*.
(Technically an ODE constrained optimization problem.)

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*.
(Technically an ODE constrained optimization problem.)

Recall: the linear least squares problem is

$$\min_{x\in\mathbb{R}^N} \frac{1}{2}\|Ax-b\|_2^2. \tag{2}$$

where

$$\frac{1}{2}\|Ax-b\|_2^2 = \frac{1}{2}x^T(A^TA)x - (A^Tb)^Tx + \frac{1}{2}\|b\|_2^2 \tag{3}$$

Iterative Methods

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

- Choose an initial iterate x_0
- For $k = 0, 1, \ldots$
    - If x_k is (close enough to) 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.

Gradient and Hessian

Let $f : \mathbb{R}^N \to \mathbb{R}$ 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

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

Necessary Conditions

Theorem

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

$$\nabla f(x^*) = 0 \tag{4}$$

and the Hessian *of f,*

$$\nabla^2 f(x^*), \ \textit{is} \text{ positive semidefinite.} \tag{5}$$

Recall A *positive semidefinite* means

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

Equation (4) is called the *first-order necessary condition*, including (6) we have the *second-order neccessary conditions*.

Sufficient Conditions

Strict positivity of the second derivative allows one to be certain that there exists a minimum, for instance, consider $f(x) = x^3$ vs $f(x) = x^4$.

Theorem

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

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

and the Hessian *of f,*

$$\nabla^2 f(x^*), \text{ \textit{is} positive definite.} \tag{6}$$

Then x^* is a local minimizer of f.

These are the *second-order sufficient conditions*.

Quadratic Objective Functions

Suppose

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

Quadratic Objective Functions

Suppose

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

(for example, from linear least squares

$$\frac{1}{2}x^T(A^T A)x - (A^T b)^T x + \frac{1}{2}\|b\|_2^2 \tag{7}$$

with $H = A^T A$, $g = (A^T b)$, and ignoring $\|b\|_2^2$),

Quadratic Objective Functions

Suppose

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

(for example, from linear least squares

$$\frac{1}{2}x^T(A^T A)x - (A^T b)^T x + \frac{1}{2}\|b\|_2^2 \tag{7}$$

with $H = A^T A$, $g = (A^T b)$, and ignoring $\|b\|_2^2$), then we have that

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

(if H is symmetric; WLOG assume it is), and

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

Quadratic Objective Functions

Suppose

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

(for example, from linear least squares

$$\frac{1}{2}x^T(A^T A)x - (A^T b)^T x + \frac{1}{2}\|b\|_2^2 \tag{7}$$

with $H = A^T A$, $g = (A^T b)$, and ignoring $\|b\|_2^2$), then we have that

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

(if H is symmetric; WLOG assume it is), and

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

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

$$Hx^* = g.$$

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

$$0 = \nabla m_k(x) = \nabla f(x_k) + \nabla^2 f(x_k)(x - x_k). \tag{8}$$

Newton Step

The solution to (8) 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., $\|e_{k+1}\| \le K\|e_k\|^2$.
- 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 (with Implicit Filtering)
  - Or automatic differentiation
- Need solution of linear system at each iteration
  - Use iterative linear solver like CG
    (Inexact Newton)
- Hessians are expensive to compute (and solve/factor)
  - Use chord (factor once) or Shamanskii (refresh occassionally)
  - Use Quasi-Newton (low rank update of H_k to get H_{k+1},
    and its inverse)
  - Use Gauss-Newton (first order approximation of Hessian)

Nonlinear Least Squares

Recall,

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

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.

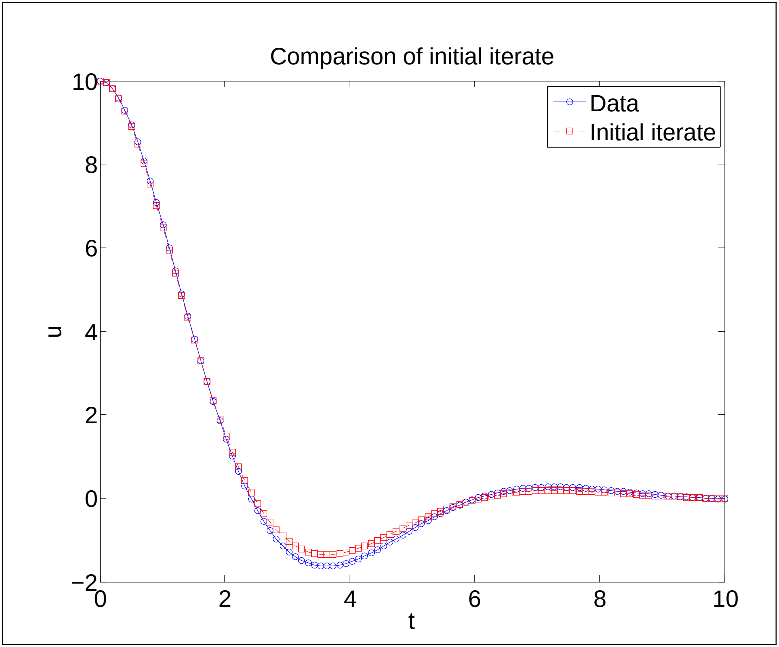
You can expect close to *quadratic* convergence for small residual problems. Otherwise, not even *linear* is guaranteed.

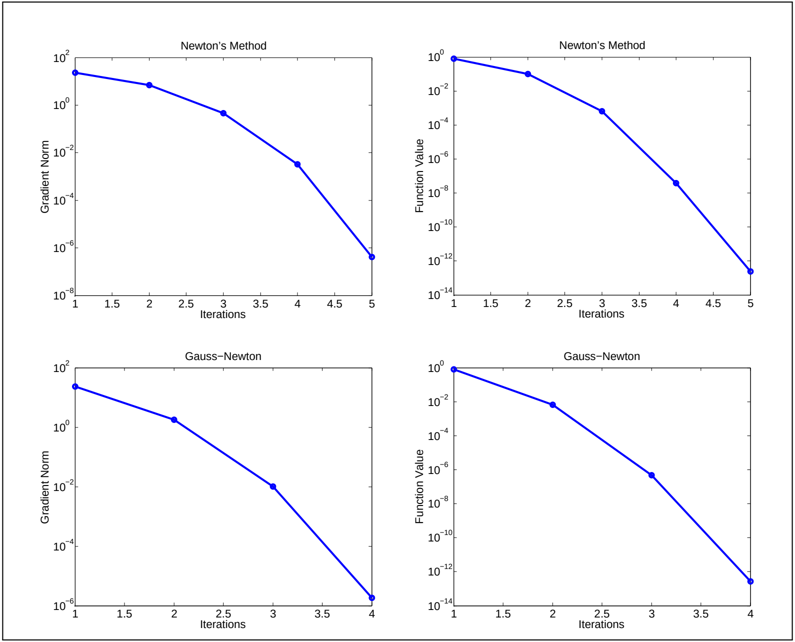
Numerical Example

- Recall

$$u'' + cu' + ku = 0; u(0) = u_0; u'(0) = 0.$$

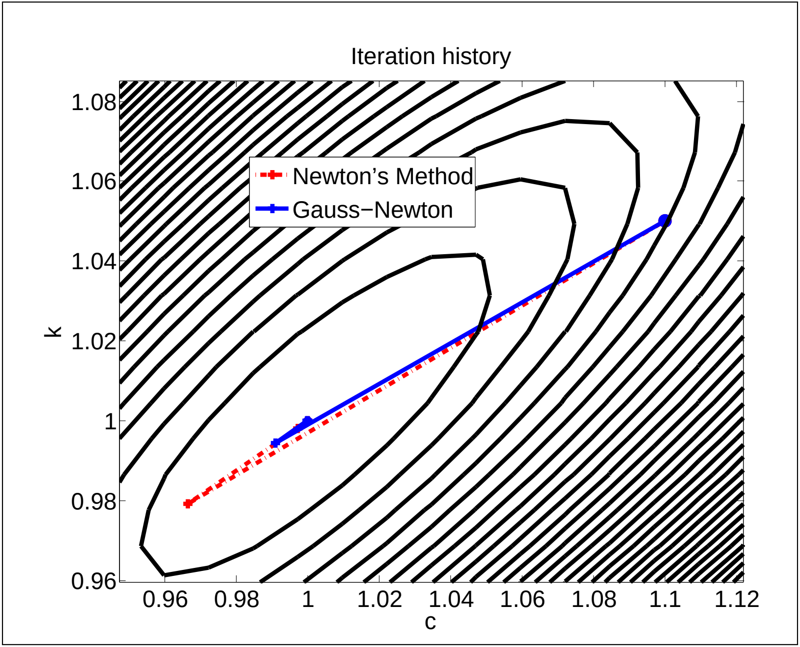
- Let the true parameters be $x^* = [c, k]^T = [1, 1]^T$. Assume we have $M = 100$ data u_j from equally spaced time points on $[0, 10]$.
- We will use the initial iterate $x_0 = [1.1, 1.05]^T$ with Newton's Method and Gauss-Newton.
- We compute gradients with forward differences, analytical 2×2 matrix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

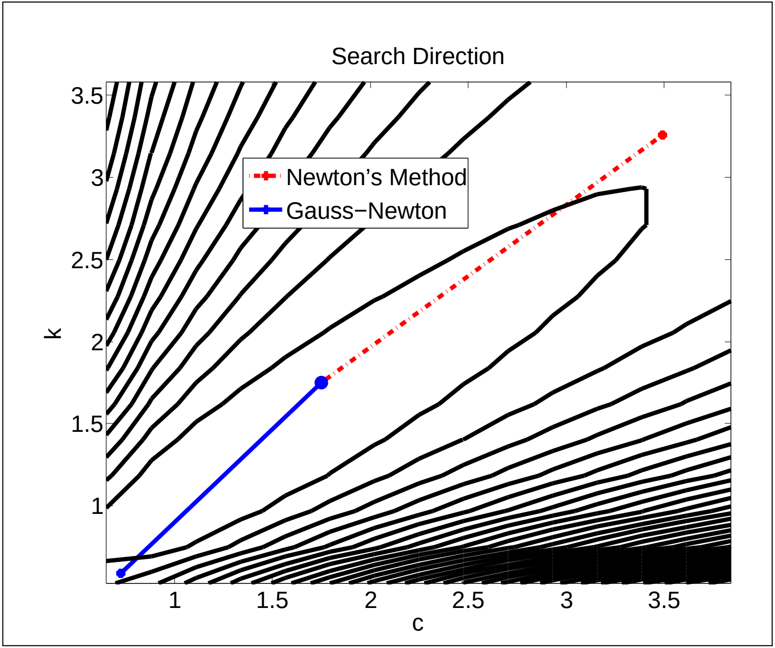

Comparison of initial iterate
Data
Initial iterate
u
t
10
8
6
4
2
0
−2
0
2
4
6
8
10

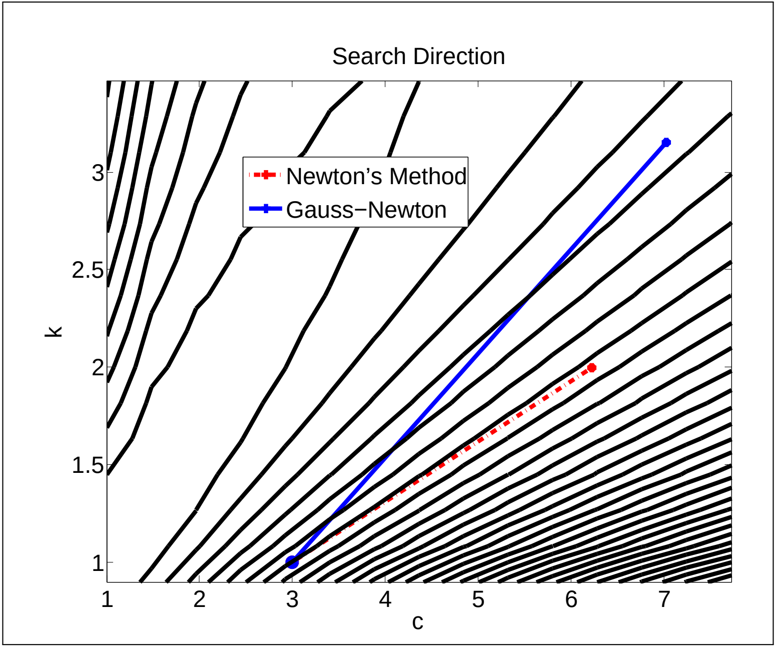

Newton's Method
Gradient Norm
Iterations
10^{2}
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
1
1.5
2
2.5
3
3.5
4
4.5
5
Newton's Method
Function Value
Iterations
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
10^{-12}
10^{-14}
Gauss−Newton
Gradient Norm
10^{-6}
Gauss−Newton
Function Value

	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

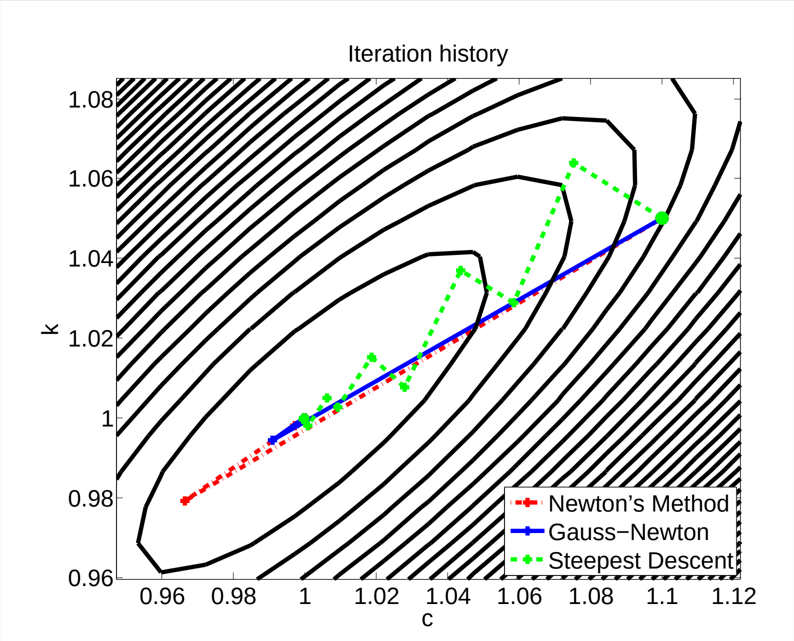

Search Direction
Newton's Method
Gauss−Newton
k
c
1
1.5
2
2.5
3
1
2
3
4
5
6
7

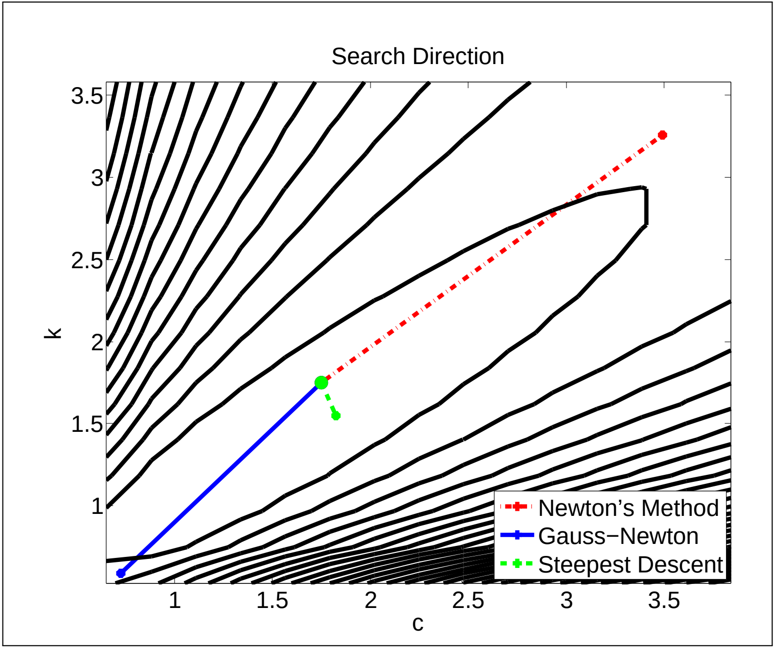
Global Convergence

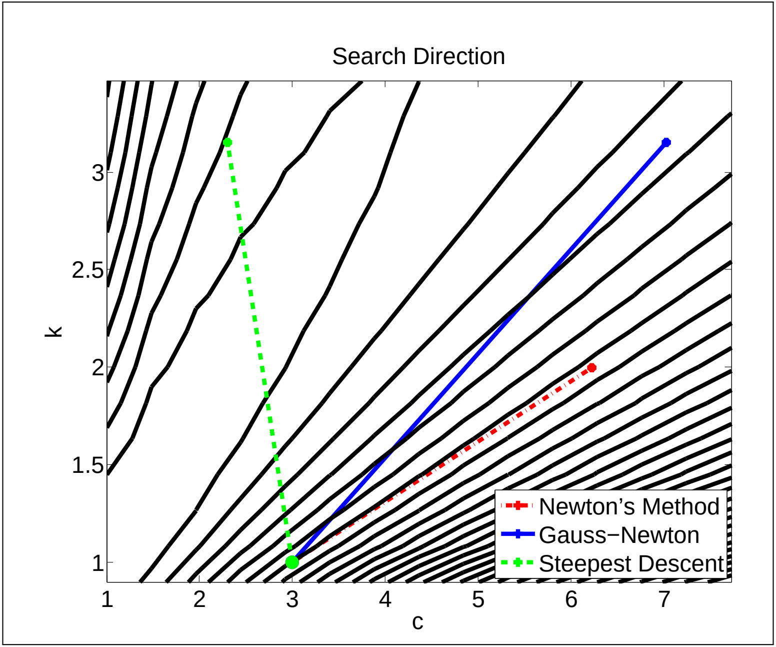
- Newton (or Gauss-Newton) direction may not be a descent direction (if Hessian not positive definite).
- Thus Newton (or any Newton-based method) may fail to decrease f if x_0 not close enough. Not *globally convergent*.
- Globally convergent methods ensure (sufficient) decrease in f.
- The *steepest 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_k.
- Since the steepest descent direction is always a descent direction, a λ_k can be found to ensure (sufficient) decrease, thus the method is guaranteed to converge to a local minima regardless of how far away it starts (global convergence).


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
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


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

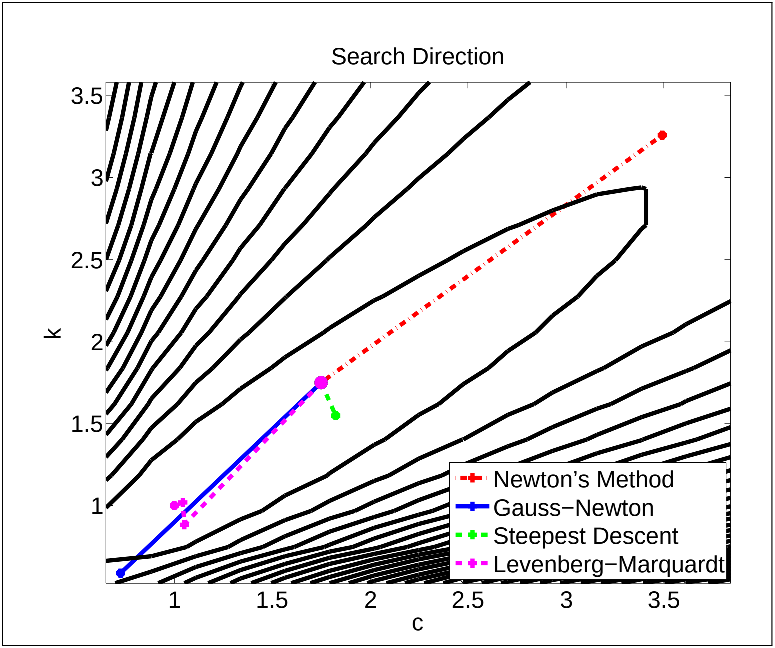
Recall the objective function

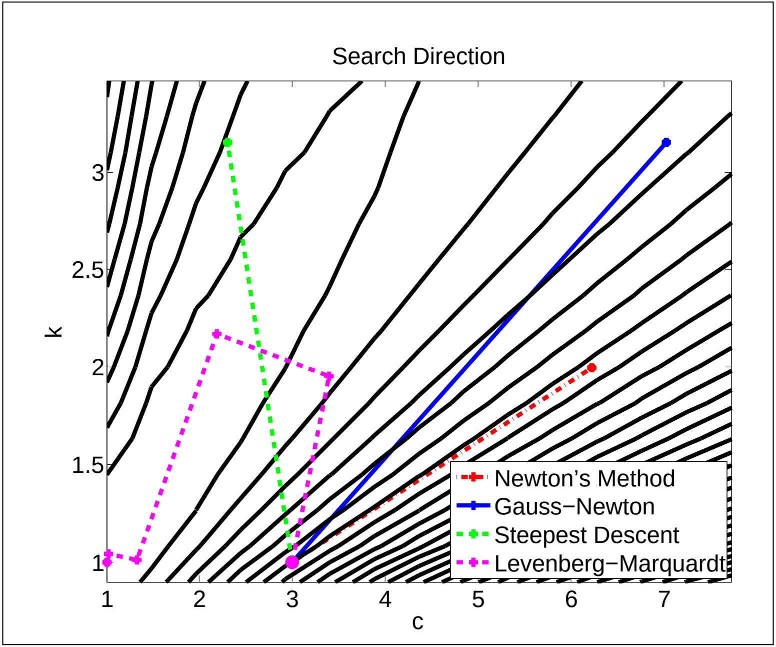
$$f(x) = \frac{1}{2} R(x)^T R(x)$$

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
c
1
2
3
4
5
6
7
1
1.5
2
2.5
3
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 of Methods

- 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:

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

$$0 = \nabla m_k(x) \implies H_k s_k = -\nabla f(x_k)$$

Globalization Outline

- Line Search (Armijo Rule)
    - Damped Gauss-Newton
    - LMA
- Levenberg-Marquardt Parameter
- Polynomial Models
- Trust Region
    - Changing TR Radius
    - Changing LM Parameter

Step Length

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 length*.
- We define the Steepest Descent update step to be $s_k^{SD} = \lambda_k d_k$ for some $\lambda_k > 0$.
- We would like to choose λ_k so that $f(x)$ *decreases sufficiently*.
- If we ask simply that

  $$f(x_{k+1}) < f(x_k)$$

  Steepest Descent might not converge (stagnation).

Predicted Reduction

Consider a linear model of $f(x)$

$$m_k(x) = f(x_k) + \nabla f(x_k)^T(x - x_k).$$

Then the *predicted reduction* using the Steepest Descent step ($x_{k+1} = x_k - \lambda_k \nabla f(x_k)$) is

$$pred = m_k(x_k) - m_k(x_{k+1}) = \lambda_k \|\nabla f(x_k)\|^2.$$

The *actual reduction* in f is

$$ared = f(x_k) - f(x_{k+1}).$$

Sufficient Decrease

We define a sufficient decrease to be when

$$\textit{ared} \geq \alpha \; \textit{pred},$$

where $\alpha \in (0,1)$ (e.g., 10^{-4} or so).
Note: $\alpha = 0$ is simple decrease.

Armijo Rule

We can define a strategy for determining the step length in terms of a sufficient decrease criteria as follows:
Let $\lambda = \beta^m$, where $\beta \in (0,1)$ (think $\frac{1}{2}$) and $m \geq 0$ is the smallest integer such that

$$\mathit{ared} > \alpha \, \mathit{pred},$$

where $\alpha \in (0,1)$.

Line Search

- The *Armijo Rule* is an example of a line search:
  Search on a ray from x_k in direction of locally decreasing f.
- Armijo procedure is to start with $m = 0$ then increment m until sufficient decrease is achieved, i.e., $\lambda = \beta^m = 1, \beta, \beta^2, \ldots$
- This approach is also called “backtracking” or performing “pullbacks”.
- For each m a new function evaluation is required.

Damped Gauss-Newton

- Armijo Rule applied to the Gauss-Newton step is called the *Damped Gauss-Newton Method*.
- Recall
$$d^{GN} = -\left(R'(x)^T R'(x)\right)^{-1} R'(x)^T R(x).$$
- Note that if $R'(x)$ has full column rank, then

$$0 > \nabla f(x)^T d^{GN} = \\ -\left(R'(x)^T R(x)\right)^T \left(R'(x)^T R'(x)\right)^{-1} R'(x)^T R(x)$$

so the GN direction is a descent direction.

Damped Gauss-Newton Step

Thus the step for Damped Gauss-Newton is

$$s^{DGN} = \beta^m d^{GN}$$

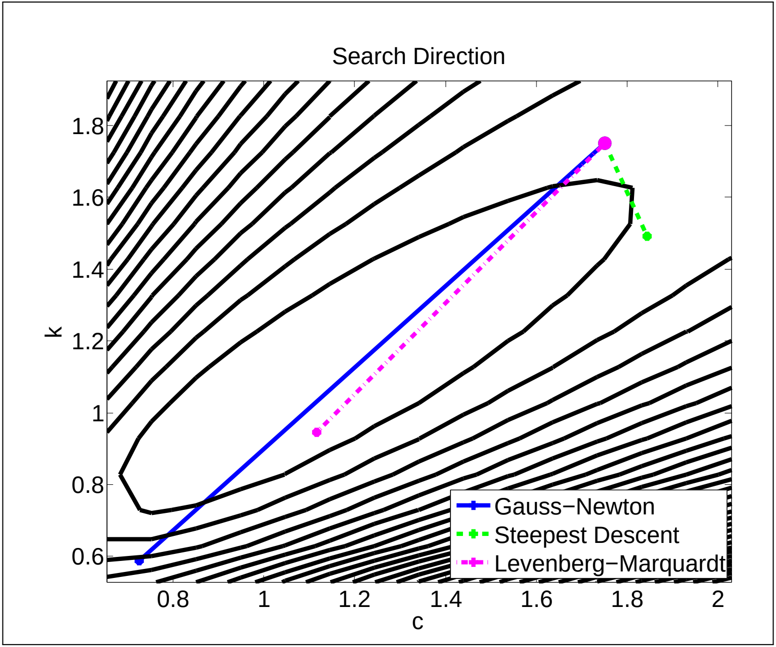
where $\beta \in (0, 1)$ and m is the smallest non-negative integer to guarantee sufficient decrease.

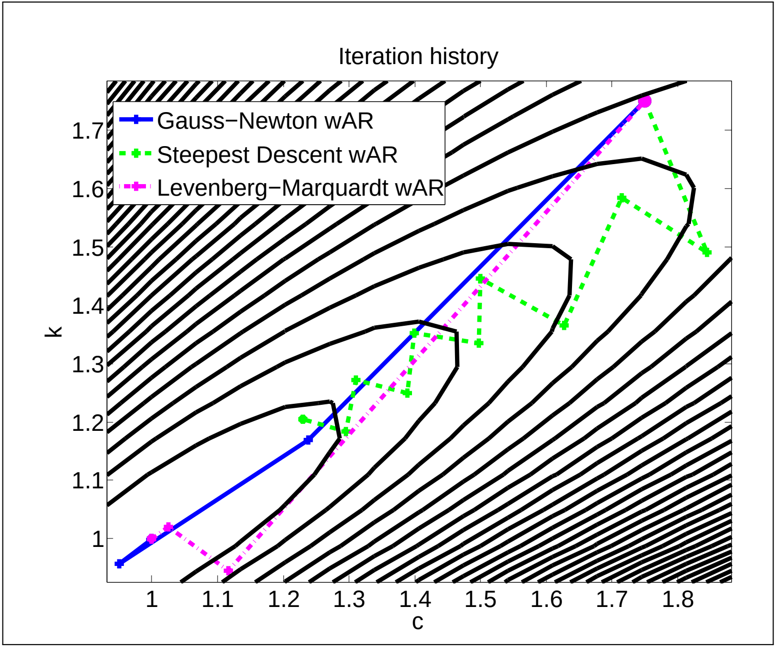
Levenberg-Marquardt-Armijo

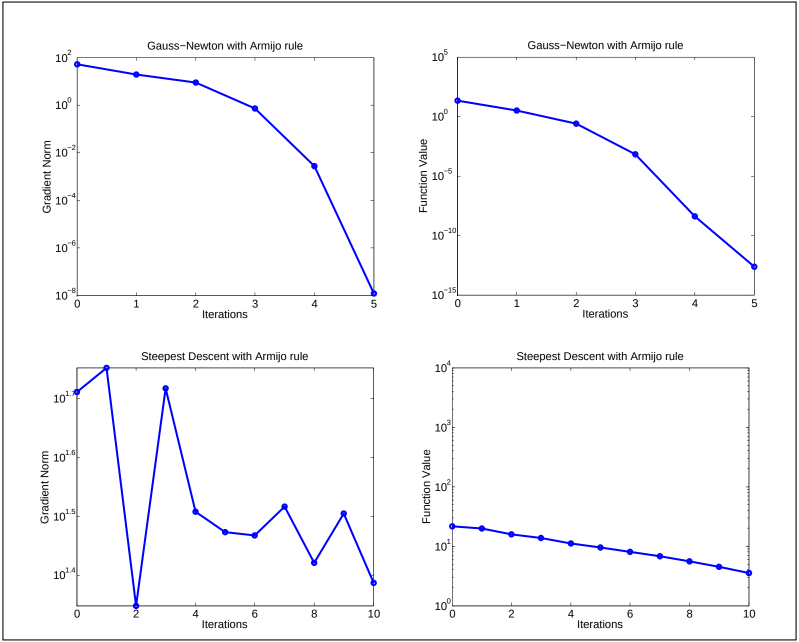
- If $R'(x)$ does not have full column rank, or if the matrix $R'(x)^T R'(x)$ may be ill-conditioned, you should be using Levenberg-Marquardt.
- The LM direction is a descent direction.
- Line search can be applied.
- Can show that if $\nu_k = O(\|R(x_k)\|)$ then LMA converges quadratically for (nice) zero residual problems.

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 $x_0 = [3, 1]^T$ with Steepest Descent, Gauss-Newton and Levenberg-Marquardt methods using the Armijo Rule.


Search Direction
k
c
1.8
1.6
1.4
1.2
1
0.8
0.6
0.8
1
1.2
1.4
1.6
1.8
2
Gauss−Newton
Steepest Descent
Levenberg−Marquardt


Iteration history
Gauss−Newton wAR
Steepest Descent wAR
Levenberg−Marquardt wAR
1.7
1.6
1.5
1.4
1.3
1.2
1.1
1
k
1
1.1
1.2
1.3
1.4
1.5
1.6
1.7
1.8
c

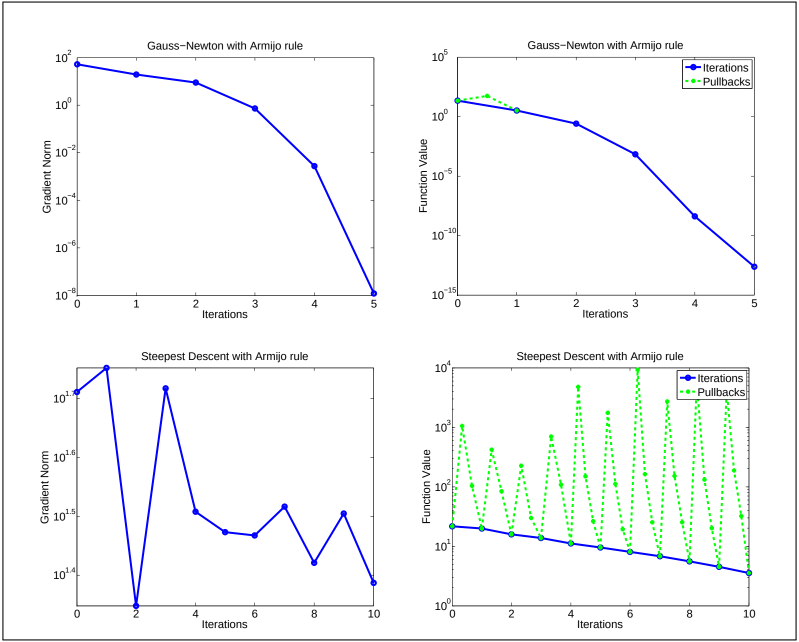


Gauss−Newton with Armijo rule

Gauss−Newton with Armijo rule

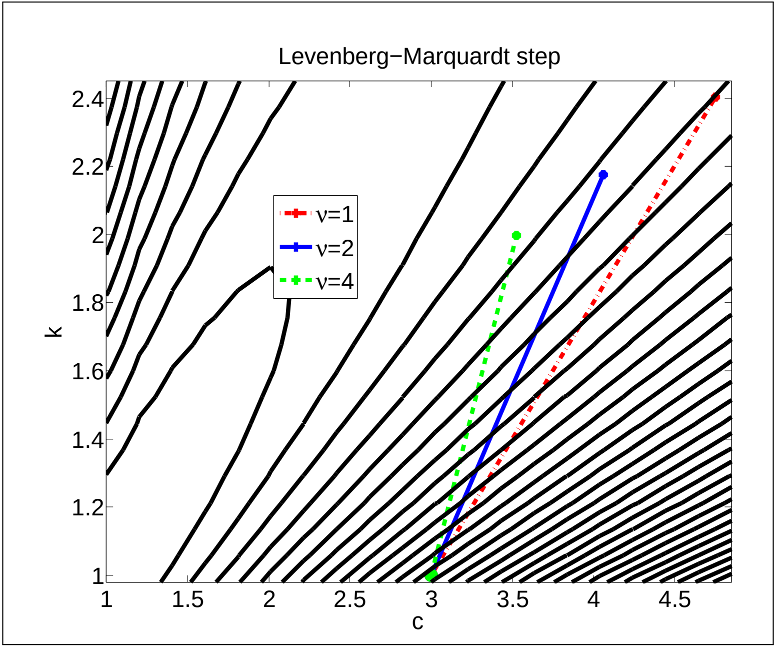
Steepest Descent with Armijo rule

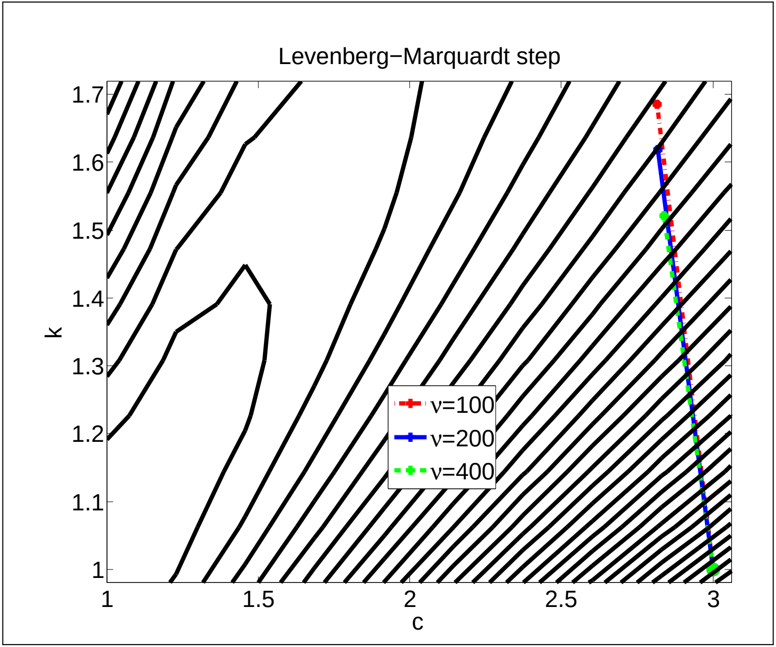
Steepest Descent with Armijo rule


Gauss−Newton with Armijo rule
Gradient Norm
10^{2}
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
0
1
2
3
4
5
Iterations
Gauss−Newton with Armijo rule
Function Value
Iterations
Pullbacks
10^{5}
10^{0}
10^{-5}
10^{-10}
10^{-15}
0
1
2
3
4
5
Iterations
Steepest Descent with Armijo rule
Gradient Norm
$10^{1.7}$
$10^{1.6}$
$10^{1.5}$
$10^{1.4}$
0
2
4
6
8
10
Iterations
Steepest Descent with Armijo rule
Function Value
Iterations
Pullbacks
10^{4}
10^{3}
10^{2}
10^{1}
10^{0}
0
2
4
6
8
10
Iterations

Word of Caution for LM

- Note that blindly increasing ν until a sufficient decrease criteria is satisfied is NOT a good idea (nor is it a line search).
- Changing ν changes direction as well as step length.
- Increasing ν does insure your direction is descending.
- But, increasing ν too much makes your step length small.


Levenberg−Marquardt step
ν=1
ν=2
ν=4
k
c
2.4
2.2
2
1.8
1.6
1.4
1.2
1
1
1.5
2
2.5
3
3.5
4
4.5


Levenberg−Marquardt step
k
c
1.7
1.6
1.5
1.4
1.3
1.2
1.1
1
1
1.5
2
2.5
3
ν=100
ν=200
ν=400

Line Search Improvements

Step length control with polynomial models

- If $\lambda = 1$ does not give sufficient decrease, use $f(x_k)$, $f(x_k + d)$ and $\nabla f(x_k)$ to build a quadratic model of

$$\xi(\lambda) = f(x_k + \lambda d)$$

- Compute the λ which minimizes model of ξ.
- If this fails, create cubic model.
- If this fails, switch back to Armijo.
- *Exact line search* is (usually) not worth the cost.

Trust Region Methods

- Let Δ be the radius of a ball about x_k inside which the quadratic model

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

  can be "trusted" to accurately represent $f(x)$.
- Δ is called the *trust region radius*.
- $\mathcal{T}(\Delta) = \{x | \, \|x - x_k\| \leq \Delta\}$ is called the *trust region*.

Trust Region Problem

- We compute a trial solution x_t, which may or may not become our next iterate.
- We define the trial solution in terms of a trial step $x_t = x_k + s_t$.
- The trial step is the (approximate) solution to the *trust region problem*

$$\min_{\|s\| \leq \Delta} m_k(x_k + s).$$

I.e., find the trial solution in the trust region which minimizes the quadratic model of f.

Changing Trust Region Radius

- Test the trial solution x_t using *predicted* and *actual* reductions.
- If $\mu = \mathit{ared}/\mathit{pred}$ too low, reject trial step and decrease trust region radius.
- If μ sufficiently high, we can accept the trial step, and possibly even increase the trust region radius (becoming more aggressive).

Exact Solution to TR Problem

Theorem

Let $g \in \mathbb{R}^N$ *and let* A *be a symmetric* $N \times N$ *matrix. Let*

$$m(s) = g^T s + s^T As/2.$$

Then a vector s *is a solution to*

$$\min_{\|s\| \leq \Delta} m(s)$$

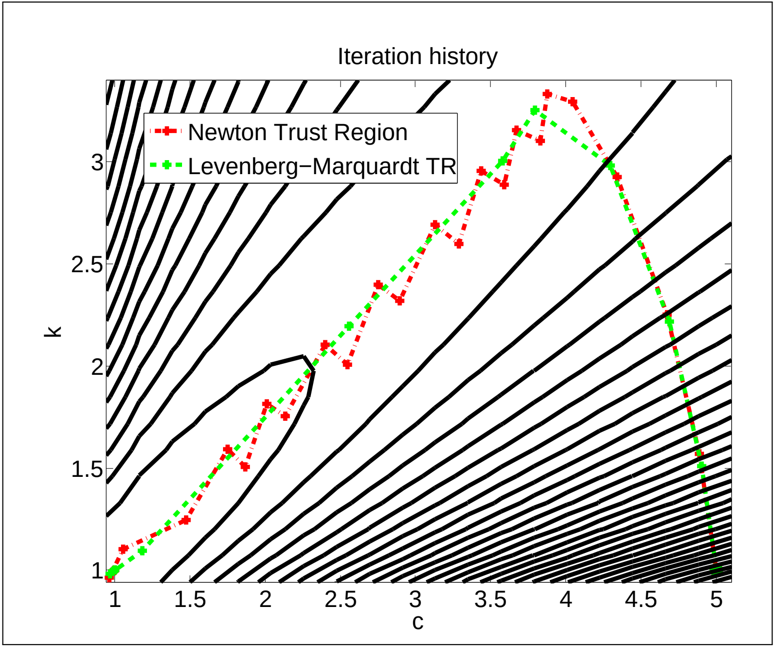
if and only if there is some $\nu \geq 0$ *such that*

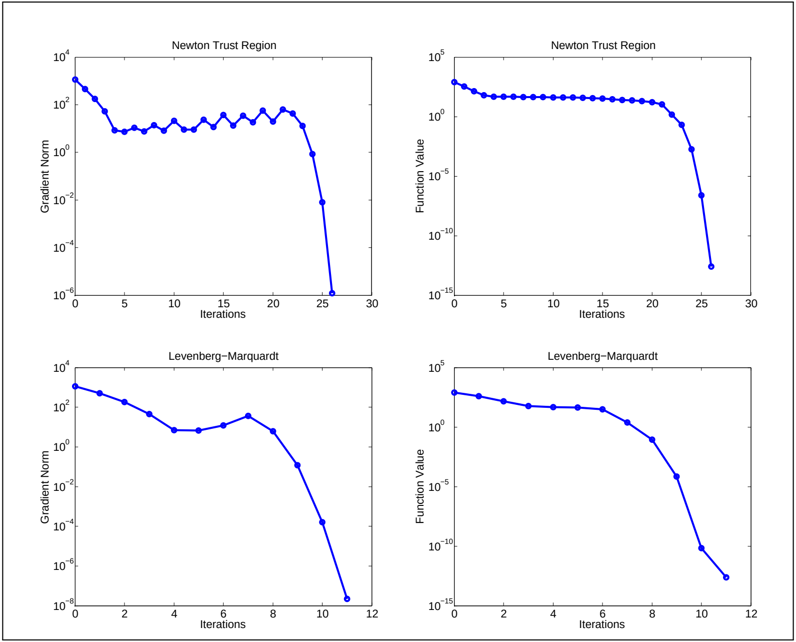
$$(A + \nu I)s = -g$$

and either $\nu = 0$ *or* $\|s\| = \Delta$*.*

LM as a TRM

- Instead of controlling Δ in response to $\mu = ared/pred$, adjust ν.
- Start with $\nu = \nu_0$ and compute $x_t = x_k + s^{LM}$.
- If $\mu = ared/pred$ too small, reject trial and *increase* ν. Recompute trial (only requires a linear solve).
- If μ sufficiently high, accept trial and possibly *decrease* ν (maybe to 0).
- Once trial accepted as an iterate, compute R, f, R', ∇f and test $\|\nabla f\|$ for termination.


Iteration history
Newton Trust Region
Levenberg−Marquardt TR
k
c
1
1.5
2
2.5
3
3.5
4
4.5
5
1
1.5
2
2.5
3


Newton Trust Region
Gradient Norm
Iterations
10^4
10^2
10^0
10^{-2}
10^{-4}
10^{-6}
0
5
10
15
20
25
30
Newton Trust Region
Function Value
Iterations
10^5
10^0
10^{-5}
10^{-10}
10^{-15}
Levenberg−Marquardt
Gradient Norm
Iterations
10^{-6}
10^{-8}
0
2
4
6
8
10
12
Levenberg−Marquardt
Function Value
Iterations

Summary

- If Gauss-Newton fails, use Levenberg-Marquardt for low-residual nonlinear least squares problems.
  - Achieves global convergence expected of Steepest Descent, but limits to quadratically convergent method near minimizer.
- Use either a trust region or line search to ensure sufficient decrease.
  - Can use trust region with any method that uses quadratic model of f.
  - Can only use line search for descent directions.

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.

Consider $A \in \mathbb{R}^{M \times N}$ and $b \in \mathbb{R}^M$, we wish to find $x \in \mathbb{R}^N$ such that

$$Ax = b.$$

In the case when $M = N$ and A^{-1} exists, the unique solution is given by

$$x = A^{-1}b.$$

For all other cases, if A is full rank, a solution is given by

$$x = A^{+}b$$

where $A^{+} = (A^T A)^{-1} A^T$ is the (Moore-Penrose) psuedoinverse of A. This solution is known as the (linear) least squares solution because it minimizes the ℓ_2 distance between the range of A and the RHS b

$$x = argmin\|b - Ax\|_2.$$

Can also be written as the solution to the normal equation

$$A^T A x = A^T b.$$

Corollary: There exists a unique least squares solution to $Ax = b$ iff A has full rank.
However, there may be (numerical) problems if A is “close” to rank-deficient, i.e., $A^T A$ is close to singular.

Regularization

One can make $A^T A$ well-posed or better conditioned by adding on a well-conditioned matrix, e.g., $\alpha I, \alpha > 0$ (Tikhonov Regularization). Thus we may solve

$$(A^T A + \alpha I)x = A^T b$$

or equivalently

$$x = argmin\|b - Ax\|_2 + \alpha\|x\|_2$$

where we have added a penalty function.

Of course, now we are solving a different (nearby) problem; this is a trade-off between matching the data (b) and prefering a particular type of solution (e.g., minimum norm).

Linear Least Squares with Uncertainty

Consider solving

$$AX = B - N$$

where now X, B, N are random variables with $N \sim \mathcal{N}(\vec{0}, C_N)$ representing additive Gaussian white noise and we expect the solution X to behave $X \sim \mathcal{N}(\vec{0}, C_X)$ (prior distribution). For any given realization of B we wish to find the expected value of X under uncertainty governed by N.

Maximum Likelihood Estimator

The maximum likelihood estimator answers question: "which value of X is most likely to produce the measured data B?"

$$x_{MLE} = argmaxp(b|x) = argmaxlogp(b|x)$$

where

$$p(b|x) = c \exp\left(-\frac{1}{2}(b - Ax)^T C_N^{-1}(b - Ax)\right)$$

and

$$logp(b|x) = -\frac{1}{2}(b - Ax)^T C_N^{-1}(b - Ax) + \tilde{c}$$

The maximum occurs when

$$0 = \frac{d}{dx} logp(b|x) = A^T C_N^{-1}(b - Ax)$$

or

$$A^T C_N^{-1} Ax = A^T C_N^{-1} b.$$

Note that solution does not depend on assumed distribution for X (ignores prior). If we assume that the error i.i.d., $C_N = \sigma_N^2 I$, then

$$A^T Ax = A^T b$$

and we get exactly the normal equations. Thus if you use the least squares solution, you are assuming i.i.d, additive Gaussian white noise.

Weighted Linear Least Squares

If this is not a good assumption, don't use lsq. For instance, if $C_N = \gamma^2 \Gamma$, Γ spd, then x_{MLE} solves

$$A^T \Gamma^{-1} A x = A^T \Gamma^{-1} b$$

or

$$\min_x \|b - Ax\|_\Gamma$$

otherwise known as weighted least squares.

Maximum a Posteriori Estimator

MAP directly answers the question: "given observation b what is the most likely x?" Consider again

$$AX = B - N$$

with $N \sim \mathcal{N}(\vec{0}, C_N)$ and $X \sim \mathcal{N}(\vec{0}, C_X)$ (prior distribution). Applying Bayes' Law

$$p(x|b) = \frac{p(b|x)p(x)}{p(b)}$$

and taking logs on both sides gives

$$logp(x|z) = -\frac{1}{2}(b - Ax)^T C_N^{-1}(b - Ax) - \frac{1}{2}x^T C_X^{-1} x + \tilde{c}.$$

Differentiating wrt x implies x_{MAP} solves

$$(A^T C_N^{-1} A + C_x^{-1})x = A^T C_N^{-1} b.$$

Tikhonov Regularization (Again)

$$(A^T C_N^{-1} A + C_x^{-1})x = A^T C_N^{-1} b.$$

Assuming $C_N = \sigma_N^2 I$ and $C_X = \sigma_X^2 I$, then

$$\left(A^T A + \left(\frac{\sigma_N}{\sigma_X} \right)^2 I \right) x = A^T b$$

which are exactly the Tikhonov regularized normal equations with

$$\alpha = \left(\frac{\sigma_N}{\sigma_X} \right)^2$$

representing a signal-to-noise ratio (trade-off).