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
  \[ f(x^*) \leq f(x) \text{ for all } x (\text{possibly in } U) \]
Parameter Identification

Consider

\[ u'' + cu' + ku = 0; \quad u(0) = u_0; \quad u'(0) = 0 \]  

(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 \( t_j \).
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 $k = 0, 1, \ldots$
  - If $x_k$ optimal, stop.
  - Determine a search direction $d$ and a step size $\lambda$
  - 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 quadratic convergence, but also intermediate rates.
**Theorem**

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

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

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

- The **gradient** of $f$ is

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

- 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 $f(x) = x^3$. 
Suppose

\[ f(x) = \frac{1}{2} x^T H x - x^T b \]

then we have that

\[ \nabla^2 f(x) = H \]

and if \( H \) is symmetric (assume it is)

\[ \nabla f(x) = H x - b. \]

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

\[ H x^* = b. \]
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

$$0 = \nabla m_k(x) = \nabla f(x_k) + \nabla^2 f(x_k)(x - x_k). \quad (3)$$
Newton’s Method

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 $\gamma$

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

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

3. $\nabla^2 f(x^*)$ is positive definite.
Newton’s Method

Convergence Rate

**Theorem**

Let the Standard Assumptions hold. Then there exists a $\delta > 0$ such that if $x_0 \in B_\delta(x^*)$, 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.
Newton’s Method

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)
Recall,

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

Then for \( x = [c, k]^T \)

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

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 $j$th 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 negligible.
Gauss-Newton Method

The equation defining the Newton step

\[ \nabla^2 f(x_k) s_k = -\nabla f(x_k) \]

becomes

\[ R'(x_k)^T R'(x_k) s_k = -\nabla f(x_k) = -R'(x_k)^T R(x_k). \]

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

You can expect close to \textit{quadratic} convergence for small residual problems. Otherwise, not even \textit{linear} is guaranteed.
Recall
\[ u'' + cu' + ku = 0; \quad u(0) = u_0; \quad u'(0) = 0. \]

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 \( 2 \times 2 \) matrix inverse, and use \( \text{ode15s} \) for time stepping the ODE.
Comparison of initial iterate

Data

Initial iterate

Prof. Gibson (OSU) Gradient-based Methods for Optimization AMC 2011 21 / 40
<table>
<thead>
<tr>
<th>$k$</th>
<th>$|\nabla f(x_k)|$</th>
<th>$f(x_k)$</th>
<th>$|\nabla f(x_k)|$</th>
<th>$f(x_k)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2.330e+01</td>
<td>7.881e-01</td>
<td>2.330e+01</td>
<td>7.881e-01</td>
</tr>
<tr>
<td>1</td>
<td>6.852e+00</td>
<td>9.817e-02</td>
<td>1.767e+00</td>
<td>6.748e-03</td>
</tr>
<tr>
<td>2</td>
<td>4.577e-01</td>
<td>6.573e-04</td>
<td>1.016e-02</td>
<td>4.656e-07</td>
</tr>
<tr>
<td>3</td>
<td>3.242e-03</td>
<td>3.852e-08</td>
<td>1.844e-06</td>
<td>2.626e-13</td>
</tr>
<tr>
<td>4</td>
<td>4.213e-07</td>
<td>2.471e-13</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

- Newton’s Method
- Gauss–Newton

Prof. Gibson (OSU)
Gradient-based Methods for Optimization
AMC 2011 24 / 40
Gauss-Newton Method

Search Direction

Newton’s Method
Gauss–Newton
Gauss-Newton Method

Search Direction

- **Newton’s Method**
- **Gauss–Newton**

Prof. Gibson (OSU)
Gradient-based Methods for Optimization
AMC 2011 26 / 40
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.
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 $s_{SD}^k = \lambda_k d_k$ for some $\lambda_k > 0$.

We will talk later about ways of choosing $\lambda$. 

Steepest Descent Method

Iteration history

Newton's Method
Gauss−Newton
Steepest Descent
Steepest Descent Method

Graph showing search direction with labeled axes and method comparisons:
- Newton’s Method
- Gauss–Newton
- Steepest Descent

Axes:
- k (vertical)
- c (horizontal)

Graph indicates the direction of search for different optimization methods.
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.
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* \( \nu_k \) 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.
Levenberg-Marquardt Method

Search Direction

- Newton’s Method
- Gauss–Newton
- Steepest Descent
- Levenberg–Marquardt

Prof. Gibson (OSU)  Gradient-based Methods for Optimization  AMC 2011  34 / 40
Levenberg-Marquardt Method

Search Direction

- Newton’s Method
- Gauss–Newton
- Steepest Descent
- Levenberg–Marquardt
Levenberg-Marquardt Notes

- Robust with respect to poor initial conditions and larger residual problems.
- Varying $\nu$ involves interpolation between GN direction ($\nu = 0$) and SD direction (large $\nu$).
- 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 $\nu$ to take more of a SD direction.
- As you get closer to minimizer, decrease $\nu$ to take more of a GN step.
  - For zero-residual problems, GN converges quadratically (if at all)
  - SD converges linearly (guaranteed)
Approximate Hessian may not be positive definite (or well-conditioned), increase $\nu$ to add regularity.

As you get closer to minimizer, Hessian will become positive definite (by Standard Assumptions). Decrease $\nu$, as less regularization is necessary.

Regularized problem is “nearby problem”, want to solve actual problem as soon as is feasible.
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:
\[ m_k^{LM}(x) = f(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{1}{2} (x - x_k)^T (R'(x_k)^T R'(x_k) + \nu_k I)(x - x_k) \]


