



THE UNIVERSITY OF
CHICAGO

Lecture 6

STAT 310 Mihai Anitescu

5.4 CONJUGATE GRADIENT CONVERGENCE/ PRECONDITIONING

Consequences of using a Krylov space:

matrix polynomial formulation

- Iteration in Krylov Space

$$\begin{aligned}x_{k+1} &= x_0 + \alpha_0 p_0 + \cdots + \alpha_k p_k \\ &= x_0 + \gamma_0 r_0 + \gamma_1 A r_0 + \cdots + \gamma_k A^k r_0,\end{aligned}$$

- Matrix Polynomial

$$P_k^*(A) = \gamma_0 I + \gamma_1 A + \cdots + \gamma_k A^k,$$

- Iteration as a matrix Polynomial

$$x_{k+1} = x_0 + P_k^*(A)r_0.$$

Error in A-space

- Error in A-norm

$$\|z\|_A^2 = z^T A z. \quad \frac{1}{2} \|x - x^*\|_A^2 = \frac{1}{2} (x - x^*)^T A (x - x^*) = \phi(x) - \phi(x^*).$$

- So what is the conjugate gradient method computing?

$$x_{k+1} = x_0 + P_k^*(A)r_0. \quad \min_{P_k} \|x_0 + P_k(A)r_0 - x^*\|_A.$$

- Another form of the error

$$x_{k+1} - x^* = x_0 + P_k^*(A)r_0 - x^* = [I + P_k^*(A)A](x_0 - x^*).$$

The calculation in eigenvalue space

Let $0 < \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$ be the eigenvalues of A , and let v_1, v_2, \dots, v_n be the corresponding orthonormal eigenvectors, so that

$$A = \sum_{i=1}^n \lambda_i v_i v_i^T.$$

$$\|x_{k+1} - x^*\|_A^2 = \min_{P_k} \sum_{i=1}^n \lambda_i [1 + \lambda_i P_k(\lambda_i)]^2 \xi_i^2.$$

$$\begin{aligned} \|x_{k+1} - x^*\|_A^2 &\leq \min_{P_k} \max_{1 \leq i \leq n} [1 + \lambda_i P_k(\lambda_i)]^2 \left(\sum_{j=1}^n \lambda_j \xi_j^2 \right) \\ &= \min_{P_k} \max_{1 \leq i \leq n} [1 + \lambda_i P_k(\lambda_i)]^2 \|x_0 - x^*\|_A^2, \end{aligned}$$

Consequences for Convergence

- Linear Convergence Rate Estimate: $\min_{P_k} \max_{1 \leq i \leq n} [1 + \lambda_i P_k(\lambda_i)]^2.$
- Consequences:

Theorem 5.4.

If A has only r distinct eigenvalues, then the CG iteration will terminate at the solution in at most r iterations.

Theorem 5.5.

If A has eigenvalues $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$, we have that

$$\|x_{k+1} - x^*\|_A^2 \leq \left(\frac{\lambda_{n-k} - \lambda_1}{\lambda_{n-k} + \lambda_1} \right)^2 \|x_0 - x^*\|_A^2.$$

- Note: finite termination in n steps.

Acceleration of Conjugate Gradient

- Rescaling of the problem

$$\hat{x} = Cx.$$

- The modified objective function

$$\hat{\phi}(\hat{x}) = \frac{1}{2}\hat{x}^T (C^{-T}AC^{-1})\hat{x} - (C^{-T}b)^T \hat{x}.$$

- Equivalent linear system.

$$(C^{-T}AC^{-1})\hat{x} = C^{-T}b,$$

How to find a preconditioner?

- Idea (from Theorem 5.5). Compute a C such that the eigenvalues are “clustered”, then convergence is fast. For example

$$C^{-T}AC \approx I; \quad \text{or} \quad C \approx L^T; \quad A = LL^T$$

- Preconditioners must be easy to factorize or invert.
- Example preconditioners:
 - Incomplete Cholesky (use sparsity pattern of A)
 - Symmetric Successive overrelaxation
 - Multigrid.

Preconditioned conjugate gradient

Algorithm 5.3 (Preconditioned CG).

Given x_0 , preconditioner M ;

Set $r_0 \leftarrow Ax_0 - b$;

Solve $My_0 = r_0$ for y_0 ;

Set $p_0 = -y_0$, $k \leftarrow 0$;

while $r_k \neq 0$

$$\alpha_k \leftarrow \frac{r_k^T y_k}{p_k^T A p_k};$$

$$x_{k+1} \leftarrow x_k + \alpha_k p_k;$$

$$r_{k+1} \leftarrow r_k + \alpha_k A p_k;$$

Solve $My_{k+1} = r_{k+1}$;

$$\beta_{k+1} \leftarrow \frac{r_{k+1}^T y_{k+1}}{r_k^T y_k};$$

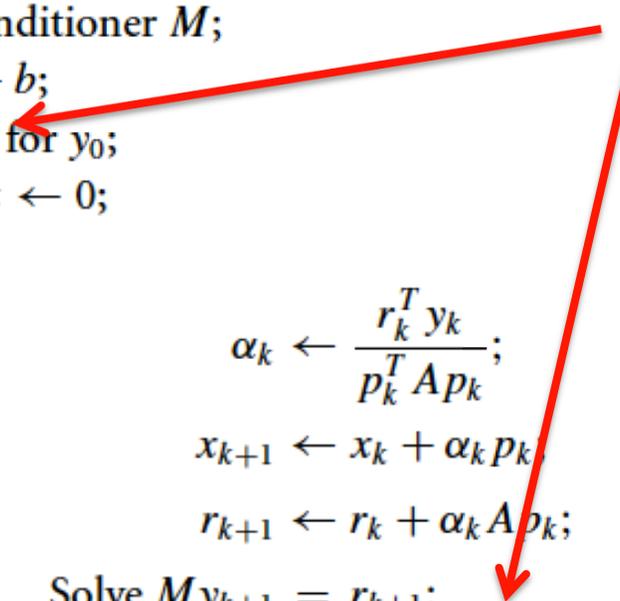
$$p_{k+1} \leftarrow -y_{k+1} + \beta_{k+1} p_k;$$

$$k \leftarrow k + 1;$$

end (while)

Preconditioner action

$$M = C^T C$$



5.5 NONLINEAR CONJUGATE GRADIENT

The Fletcher-Reeves method

Algorithm 5.4 (FR).

Given x_0 ;

Evaluate $f_0 = f(x_0)$, $\nabla f_0 = \nabla f(x_0)$;

Set $p_0 \leftarrow -\nabla f_0$, $k \leftarrow 0$;

while $\nabla f_k \neq 0$

 Compute α_k and set $x_{k+1} = x_k + \alpha_k p_k$;

 Evaluate ∇f_{k+1} ;

$$\beta_{k+1}^{\text{FR}} \leftarrow \frac{\nabla f_{k+1}^T \nabla f_{k+1}}{\nabla f_k^T \nabla f_k};$$

$$p_{k+1} \leftarrow -\nabla f_{k+1} + \beta_{k+1}^{\text{FR}} p_k;$$

$$k \leftarrow k + 1;$$

end (while)

Actual line search

$$\alpha_k \leftarrow \frac{r_k^T r_k}{p_k^T A p_k};$$

$$x_{k+1} \leftarrow x_k + \alpha_k p_k;$$

$$r_{k+1} \leftarrow r_k + \alpha_k A p_k;$$

$$\beta_{k+1} \leftarrow \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k};$$

$$p_{k+1} \leftarrow -r_{k+1} + \beta_{k+1} p_k;$$

$$k \leftarrow k + 1;$$

$$r_k = \nabla f(x_k)$$

The line search method

- Use a search that satisfies the Wolfe Conditions.

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f_k^T p_k,$$

$$|\nabla f(x_k + \alpha_k p_k)^T p_k| \leq -c_2 \nabla f_k^T p_k,$$

- Use the parameters

$$0 < c_1 < c_2 < \frac{1}{2}.$$

Convergence of Fletcher-Reeves

Lemma 5.6.

Suppose that Algorithm 5.4 is implemented with a step length α_k that satisfies the strong Wolfe conditions (5.43) with $0 < c_2 < \frac{1}{2}$. Then the method generates descent directions p_k that satisfy the following inequalities:

$$-\frac{1}{1-c_2} \leq \frac{\nabla f_k^T p_k}{\|\nabla f_k\|^2} \leq \frac{2c_2-1}{1-c_2}, \quad \text{for all } k = 0, 1, \dots \quad (5.53)$$

Assumptions 5.1.

- (i) The level set $\mathcal{L} := \{x \mid f(x) \leq f(x_0)\}$ is bounded;
- (ii) In some open neighborhood \mathcal{N} of \mathcal{L} , the objective function f is Lipschitz continuously differentiable.

Fixed fraction of steepest descent

Theorem 5.7 (Al-Baali [3]).

Suppose that Assumptions 5.1 hold, and that Algorithm 5.4 is implemented with a line search that satisfies the strong Wolfe conditions (5.43), with $0 < c_1 < c_2 < \frac{1}{2}$. Then

$$\liminf_{k \rightarrow \infty} \|\nabla f_k\| = 0. \quad (5.63)$$



THE UNIVERSITY OF
CHICAGO

Section 6

Very Large Scale Methods

6.1 MATRIX-FREE METHODS CONVERGENCE FRAMEWORK

Matrix Free Methods

- Parallel computing: avoid factorization and return only matrix-vector products (and not matrices) .

$$d \rightarrow \boxed{\text{Model}} \rightarrow B * d$$

$$d, x_k \rightarrow \boxed{\text{Model}} \rightarrow \nabla_{xx}^2 f(x_k) * x$$

- The last version in particular can be particularly efficiently carried out with Automatic Differentiation.
- Most common algorithms in optimization: Krylov Algorithms (Lanczos, modified CG).
- But I must deal with **early termination** (I will not wait n steps) and **indefinite matrices**.

Framework for Early termination: Inexact Newton Methods

- We modify the original Newton method:

$$\nabla^2 f_k p_k^N = -\nabla f_k.$$

$$\nabla^2 f_k p_k \approx -\nabla f_k$$

- The residual:

$$r_k = \nabla^2 f_k p_k + \nabla f_k,$$

- CG loop termination rule

$$\|r_k\| \leq \eta_k \|\nabla f_k\|, \quad \eta_k \rightarrow 0$$



Forcing sequence

Convergence Result

- Main Result:

Theorem 7.1.

Suppose that $\nabla^2 f(x)$ exists and is continuous in a neighborhood of a minimizer x^ , with $\nabla^2 f(x^*)$ is positive definite. Consider the iteration $x_{k+1} = x_k + p_k$ where p_k satisfies (7.3), and assume that $\eta_k \leq \eta$ for some constant $\eta \in [0, 1)$. Then, if the starting point x_0 is sufficiently near x^* , the sequence $\{x_k\}$ converges to x^* and satisfies*

$$\|\nabla^2 f(x^*)(x_{k+1} - x^*)\| \leq \hat{\eta} \|\nabla^2 f(x^*)(x_k - x^*)\|, \quad (7.4)$$

for some constant $\hat{\eta}$ with $\eta < \hat{\eta} < 1$.

- Note: if sequence is forced, superlinear convergence will occur!

6.2 KRYLOV-TYPE METHODS FOR NONLINEAR UNCONSTRAINED OPTIMIZATION

Main Concern:

- How do I deal with indefiniteness of the matrices, since CG works only for positive definite matrices?

Line-Search CG

Algorithm 7.1 (Line Search Newton–CG).

Given initial point x_0 ;

for $k = 0, 1, 2, \dots$

Define tolerance $\epsilon_k = \min(0.5, \sqrt{\|\nabla f_k\|}) \|\nabla f_k\|$;

Set $z_0 = 0, r_0 = \nabla f_k, d_0 = -r_0 = -\nabla f_k$;

for $j = 0, 1, 2, \dots$

if $d_j^T B_k d_j \leq 0$

if $j = 0$

return $p_k = -\nabla f_k$;

else

return $p_k = z_j$;

Set $\alpha_j = r_j^T r_j / d_j^T B_k d_j$;

Set $z_{j+1} = z_j + \alpha_j d_j$;

Set $r_{j+1} = r_j + \alpha_j B_k d_j$;

if $\|r_{j+1}\| < \epsilon_k$

return $p_k = z_{j+1}$;

Set $\beta_{j+1} = r_{j+1}^T r_{j+1} / r_j^T r_j$;

Set $d_{j+1} = -r_{j+1} + \beta_{j+1} d_j$;

end (for)

Set $x_{k+1} = x_k + \alpha_k p_k$, where α_k satisfies the Wolfe, Goldstein, or Armijo backtracking conditions (using $\alpha_k = 1$ if possible);

end

Note double iteration

- How come it works?
CG itself is a descent method !!!

$$\eta_k = \min(0.5, \sqrt{\|\nabla f_k\|})$$

CG-Trust Region (STEIHAUG)

$$\min_{p \in \mathbb{R}^n} m_k(p) \stackrel{\text{def}}{=} f_k + (\nabla f_k)^T p + \frac{1}{2} p^T B_k p \quad \text{subject to } \|p\| \leq \Delta_k,$$

Algorithm 7.2 (CG–Steihaug).

Given tolerance $\epsilon_k > 0$;

Set $z_0 = 0, r_0 = \nabla f_k, d_0 = -r_0 = -\nabla f_k$;

if $\|r_0\| < \epsilon_k$

 return $p_k = z_0 = 0$;

for $j = 0, 1, 2, \dots$

 if $d_j^T B_k d_j \leq 0$

 Find τ such that $p_k = z_j + \tau d_j$ minimizes $m_k(p_k)$ in (4.5)
 and satisfies $\|p_k\| = \Delta_k$;

 return p_k ;

 Set $\alpha_j = r_j^T r_j / d_j^T B_k d_j$;

 Set $z_{j+1} = z_j + \alpha_j d_j$;

 if $\|z_{j+1}\| \geq \Delta_k$

 Find $\tau \geq 0$ such that $p_k = z_j + \tau d_j$ satisfies $\|p_k\| = \Delta_k$;

 return p_k ;

 Set $r_{j+1} = r_j + \alpha_j B_k d_j$;

 if $\|r_{j+1}\| < \epsilon_k$

 return $p_k = z_{j+1}$;

 Set $\beta_{j+1} = r_{j+1}^T r_{j+1} / r_j^T r_j$;

 Set $d_{j+1} = -r_{j+1} + \beta_{j+1} d_j$;

end (for).

- Only inner loop.
- Define forcing sequence as LS-CG
- Iterate in x .

6.3 LANCZOS ALGORITHMS FOR UNCONSTRAINED OPTIMIZATION

A shortcoming of CG methods

- They accept even SMALL negative curvature foregoing more promising directions.
- Solution: try to approximate the spectrum of the Hessian, using the Lanczos algorithm

Conjugate Gradients and Lanczos Algorithm (Tremolet)

- The conjugate gradient algorithm minimizes a quadratic function with a symmetric positive-definite Hessian: $J(x) = \frac{1}{2}x^T Ax + b^T x + c$

□ The algorithm is:

$$x_{k+1} = x_k + \alpha_k d_k \quad \text{step to the line minimum}$$

$$g_{k+1} = g_k + \alpha_k A d_k \quad \text{recalculate the gradient}$$

$$d_{k+1} = -g_{k+1} + \beta_k d_k \quad \text{calculate a new direction}$$

where:

$$d_0 = -g_0 \quad \alpha_k = \frac{\langle g_k, g_k \rangle}{\langle d_k, A d_k \rangle} \quad \beta_k = \frac{\langle g_{k+1}, g_{k+1} \rangle}{\langle g_k, g_k \rangle}.$$

□ Eliminate d_k to get the 3-term recurrence (Lanczos):

$$25 \quad A g_{k+1} = -\frac{\beta_k}{\alpha_k} g_k + \left(\frac{\beta_k}{\alpha_k} + \frac{1}{\alpha_{k+1}} \right) g_{k+1} - \frac{1}{\alpha_{k+1}} g_{k+2}$$

Lanczos Orthogonalization Procedure

- It orthogonalizes the Krylov space

$$(K2) \Rightarrow S_k = \{r_0, Ar_0, \dots, A^{k-1}r_0\} = K_k(A, r_0) \quad (K2)$$

- But the iteration works even if the matrix is NOT positive definite !!!

$$Ag_{k+1} = -\frac{\beta_k}{\alpha_k}g_k + \left(\frac{\beta_k}{\alpha_k} + \frac{1}{\alpha_{k+1}}\right)g_{k+1} - \frac{1}{\alpha_{k+1}}g_{k+2}$$

- The coefficients are found without needing d; by just eliminating $g_{(k+2)}$. **EXPAND**

Conjugate Gradients and Lanczos Algorithms

- Let Q_k be the matrix whose columns are $g_i/\|g_i\|$
- Then $AQ_k = Q_kT_k + g_k e_k^T$
 where T_k is tri-diagonal and $e_k^T = (0, \dots, 0, 1)$
- After N iterations, we get $Q_N^T A Q_N = T_N$.
- *i.e.* T_N has the same eigenvalues as A .
- Intermediate matrices have interleaving eigenvalues:

$$\lambda_{j-1}(T_k) \geq \lambda_j(T_{k+1}) \geq \lambda_j(T_k)$$
- Even for $k \ll N$, “the spectrum range” is well
27
 approximated.

Lanczos Iteration

```
 $q_0 = 0$   
 $\beta_0 = 0$   
 $x_0 =$  arbitrary nonzero starting vector  
 $q_1 = x_0 / \|x_0\|_2$   
for  $k = 1, 2, \dots$   
     $u_k = Aq_k$   
     $\alpha_k = q_k^H u_k$   
     $u_k = u_k - \beta_{k-1}q_{k-1} - \alpha_k q_k$   
     $\beta_k = \|u_k\|_2$   
    if  $\beta_k = 0$  then stop  
     $q_{k+1} = u_k / \beta_k$   
end
```

Unless it breaks down, produces orthogonal basis of Krylov space and a tridiagonal matrix similar to A .

Lanczos iteration

- α_k and β_k are diagonal and subdiagonal entries of symmetric tridiagonal matrix T_k
- If $\beta_k = 0$, then algorithm appears to break down, but in that case invariant subspace has already been identified (i.e., Ritz values and vectors are already exact at that point)

Lanczos iteration

- In principle, if Lanczos algorithm were run until $k = n$, resulting tridiagonal matrix would be orthogonally similar to A
- In practice, rounding error causes loss of orthogonality, invalidating this expectation
- Problem can be overcome by reorthogonalizing vectors as needed, but expense can be substantial
- Alternatively, can ignore problem, in which case algorithm still produces good eigenvalue approximations, but multiple copies of some eigenvalues may be generated

Tridiagonal System

$$\begin{bmatrix}
 d_1 & u_1 & & & & 0's \\
 l_1 & d_2 & u_2 & & & \\
 & l_2 & d_3 & u_3 & & \\
 & & \ddots & \ddots & \ddots & \\
 & & & l_{n-2} & d_{n-1} & u_{n-1} \\
 0's & & & & l_{n-1} & d_n
 \end{bmatrix}
 \begin{bmatrix}
 x_1 \\
 x_2 \\
 x_3 \\
 \vdots \\
 x_{n-1} \\
 x_n
 \end{bmatrix}
 =
 \begin{bmatrix}
 b_1 \\
 b_2 \\
 b_3 \\
 \vdots \\
 b_{n-1} \\
 b_n
 \end{bmatrix}$$

- **Tridiagonal matrices are EXTREMELY easy to factorize, solve with, and find eigenvalues of (if symmetric).**
- $\mathbf{u} = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_{n-1}]$
- $\mathbf{d} = [\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_{n-1}, \mathbf{d}_n]$
- $\mathbf{l} = [\mathbf{l}_1, \mathbf{l}_2, \dots, \mathbf{l}_{n-1}]$

LU decomposition of Tridiagonal Matrix (Cholesky similar)

$$\begin{bmatrix} 1 & & & & & & & & 0's \\ l_1 & 1 & & & & & & & \\ & l_2 & 1 & & & & & & \\ & & \ddots & \ddots & & & & & \\ & & & l_{n-2} & 1 & & & & \\ 0's & & & & l_{n-1} & 1 & & & \end{bmatrix} \times \begin{bmatrix} d_1 & u_1 & & & & & & & 0's \\ & d_2 & u_2 & & & & & & \\ & & d_3 & u_3 & & & & & \\ & & & \ddots & \ddots & & & & \\ & & & & d_{n-1} & u_{n-1} & & & \\ 0's & & & & & & d_n & & \end{bmatrix} = \begin{bmatrix} d_1 & u_1 & & & & & & & 0's \\ l_1 & d_2 & u_2 & & & & & & \\ & l_2 & d_3 & u_3 & & & & & \\ & & \ddots & \ddots & \ddots & & & & \\ & & & l_{n-2} & d_{n-1} & u_{n-1} & & & \\ 0's & & & & l_{n-1} & d_n & & & \end{bmatrix}$$

Thomas Algorithm:

$$\begin{aligned}
 l_{n-1} &= l_{n-1} / d_{n-1} \\
 d_n &= d_n - (l_{n-1} / d_{n-1}) * u_{n-1}
 \end{aligned}$$

Using Lanczos for Optimization

- Solve trust-region (inner loop):

$$\min_{w \in \mathbb{R}^j} f_k + e_1^T Q_j (\nabla f_k) e_1^T w + \frac{1}{2} w^T T_j w \quad \text{subject to } \|w\| \leq \Delta_k,$$

- Note, however, that you must store ALL vectors.
- But you will not truncate a promising direction just before it gives a negative inner product.
- Iteration continues, until a similar stopping test is reached (i.e residual=gradient is small)



THE UNIVERSITY OF
CHICAGO

Section 7: Special Optimization Problem Classes

Mihai Anitescu, Stat 310

Special Problem Classes

- Nonlinear Least Squares
- Nonlinear Equations
- The latter is *ALSO* affected by the divergence of classical Newton, so needs to be “globalized”

7.1 NONLINEAR LEAST SQUARES

Framework

- Problem:
- Vector of residuals
- Jacobian
- Derivatives:

$$\min_{x \in \mathbb{R}^n} f(x) = \frac{1}{2} \sum_{j=1}^m r_j^2(x),$$

$$r(x) = (r_1(x), r_2(x), \dots, r_m(x))^T.$$

$$\begin{aligned} \nabla f(x) &= \sum_{j=1}^m r_j(x) \nabla r_j(x) = J(x)^T r(x), \\ \nabla^2 f(x) &= \sum_{j=1}^m \nabla r_j(x) \nabla r_j(x)^T + \sum_{j=1}^m r_j(x) \nabla^2 r_j(x) \\ &= J(x)^T J(x) + \sum_{j=1}^m r_j(x) \nabla^2 r_j(x). \end{aligned}$$

$$J(x) = \left[\frac{\partial r_j}{\partial x_i} \right]_{\substack{j=1,2,\dots,m \\ i=1,2,\dots,n}} = \begin{bmatrix} \nabla r_1(x)^T \\ \nabla r_2(x)^T \\ \vdots \\ \nabla r_m(x)^T \end{bmatrix},$$

Gauss-Newton

- Use only the Jacobian to generate the search direction.

$$\nabla^2 f_k \approx J_k^T J_k \qquad J_k^T J_k p_k^{\text{GN}} = -J_k^T r_k.$$

- This can be done with QR factorization since it is a LINEAR least squares

$$\min_p \frac{1}{2} \|J_k p + r_k\|^2.$$

- Followed by a line search since iteration matrix is positive (semi)-definite.
- But what if it is singular or close to? The iteration can be poor (numerical instability)

Levenberg-Marquardt: Use a trust-region approach !

- Trust-region problem: $\min_p \frac{1}{2} \|J_k p + r_k\|^2, \quad \text{subject to } \|p\| \leq \Delta_k,$
- Computational Foundation: $m_k(p) = \frac{1}{2} \|r_k\|^2 + p^T J_k^T r_k + \frac{1}{2} p^T J_k^T J_k p.$

Lemma 10.2.

The vector p^{LM} is a solution of the trust-region subproblem

$$\min_p \|Jp + r\|^2, \quad \text{subject to } \|p\| \leq \Delta,$$

if and only if p^{LM} is feasible and there is a scalar $\lambda \geq 0$ such that

$$\begin{aligned} (J^T J + \lambda I) p^{\text{LM}} &= -J^T r, \\ \lambda(\Delta - \|p^{\text{LM}}\|) &= 0. \end{aligned}$$

LM: Can still USE QR !!

- Trust region acts like regularization
- I do not need to form the matrix product or factorize it, equivalent to a linear least squares.
- Solve same way as other TR (replacing Cholesky with QR and adding TR management)

$$(J^T J + \lambda I) p = -J^T r.$$

$$\min_p \frac{1}{2} \left\| \begin{bmatrix} J \\ \sqrt{\lambda} I \end{bmatrix} p + \begin{bmatrix} r \\ 0 \end{bmatrix} \right\|^2.$$


Nonlinear Equations

- Formulation:
- Newton's Method ...
which may not converge.

$$r : \mathbb{R}^n \rightarrow \mathbb{R}^n \quad r(x) = 0,$$

$$r(x) = \begin{bmatrix} r_1(x) \\ r_2(x) \\ \vdots \\ r_n(x) \end{bmatrix}.$$

Algorithm 11.1 (Newton's Method for Nonlinear Equations).

Choose x_0 ;

for $k = 0, 1, 2, \dots$

 Calculate a solution p_k to the Newton equations

$$J(x_k)p_k = -r(x_k);$$

$x_{k+1} \leftarrow x_k + p_k$;

end (for)

Trust-region for nonlinear equations

- Model:
$$m_k(p) = \frac{1}{2} \|r_k + J_k p\|_2^2 = f_k + p^T J_k^T r_k + \frac{1}{2} p^T J_k^T J_k p.$$
- Trust-region problem:
$$\min_p m_k(p), \quad \text{subject to } \|p\| \leq \Delta_k,$$
- Reduction ratio:
- Can use dogleg ! (with positive definite B)
$$\rho_k = \frac{\|r(x_k)\|^2 - \|r(x_k + p_k)\|^2}{\|r(x_k)\|^2 - \|r(x_k) + J(x_k)p_k\|^2}.$$

TR algorithm for nonlinear equations

Algorithm 11.5 (Trust-Region Method for Nonlinear Equations).

Given $\bar{\Delta} > 0$, $\Delta_0 \in (0, \bar{\Delta})$, and $\eta \in [0, \frac{1}{4}]$:

for $k = 0, 1, 2, \dots$

 Calculate p_k as an (approximate) solution of (11.46);

 Evaluate ρ_k from (11.47);

 if $\rho_k < \frac{1}{4}$

$$\Delta_{k+1} = \frac{1}{4} \|p_k\|;$$

 else

 if $\rho_k > \frac{3}{4}$ and $\|p_k\| = \Delta_k$

$$\Delta_{k+1} = \min(2\Delta_k, \bar{\Delta});$$

 else

$$\Delta_{k+1} = \Delta_k;$$

 end (if)

end (if)

if $\rho_k > \eta$

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

else

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

end (if)

end (for).