

# Numerical Optimization

## Lecture Notes #14

### Practical Newton Methods — Hessian Modifications

Peter Blomgren,  
(blomgren.peter@gmail.com)

Department of Mathematics and Statistics  
Dynamical Systems Group  
Computational Sciences Research Center  
San Diego State University  
San Diego, CA 92182-7720

<http://terminus.sdsu.edu/>

Fall 2018



# Outline

- 1 Recap
  - Robust Inexact Newton Methods
- 2 Hessian Modifications
  - Eigenvalue Modification
  - $\mathbf{B} = \mathbf{A} + \tau \mathbf{I}$
  - Gershgorin Modification

## Quick Recap: Building Robust Inexact Newton Methods

We looked at combining a modified version of the linear CG-solver (or preferably a PCG(M)-solver) with a line-search algorithm to produce an almost “unbreakable” approximate Newton method.

The modification to the CG-solver comprise of an additional termination criterion for the case where the local Hessian ( $\nabla^2 f(\bar{\mathbf{x}}_k)$ ) is not positive definite, and we get a CG-internal search direction for which  $\bar{\mathbf{p}}^T \nabla^2 f(\bar{\mathbf{x}}_k) \bar{\mathbf{p}} \leq 0$ , i.e the search takes into a part of space with negative curvature.

The worst we do (in a particular iteration) is to take a steepest descent step.

*Potential Outstanding Problem:*  $\bar{\mathbf{p}}^T \nabla^2 f(\bar{\mathbf{x}}_k) \bar{\mathbf{p}}$  small and positive  
 $\rightsquigarrow$  long step.

## Quick Recap: Building Robust Inexact Newton Methods

We also discussed how to specify the **forcing sequence**  $\{\eta^{(k)}\}$  for the tolerance termination criterion ( $\|\bar{\mathbf{r}}_k\| \leq \eta^{(k)} \|\nabla f(\bar{\mathbf{x}}_k)\|$ ) so that the overall convergence rate of the resulting algorithm is quadratic (when  $B_k = \nabla^2 f(x_k)$ ) or super-linear (when  $B_k \approx \nabla^2 f(x_k)$ ).

We also hinted at a different approach to dealing with non-positive definite Hessians in the direct-linear-solver-framework — a modification of the Hessian ( $\nabla^2 f(\bar{\mathbf{x}}_k) + E_k$ ) so that the resulting matrix is sufficiently positive definite; today we take a closer look at this approach.

## Hessian Modifications

Old Default Project: `modelhess()`

We look at modifying the Hessian matrix  $\nabla^2 f(\bar{\mathbf{x}}_k)$  by either explicitly or implicitly adding a matrix  $E_k$  (usually a multiple of the identity matrix) so that the resulting matrix

$$B_k = \nabla^2 f(\bar{\mathbf{x}}_k) + E_k$$

is **sufficiently positive definite** (all the eigenvalues of  $B_k$  are bounded away from zero.)

There are a number of different approaches, we look at a few...

- Eigenvalue Modification
- Direct and Indirect modification of the Hessian

## Eigenvalue Modification

Since  $\nabla^2 f(\bar{\mathbf{x}}_k)$  is symmetric we can always find an orthonormal matrix  $Q_k$  and a diagonal matrix  $\Lambda_k = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$  so that (dropping the subscripts  $k$ )

$$\nabla^2 f(\bar{\mathbf{x}}) = Q \Lambda Q^T = \sum_{i=1}^n \lambda_i \bar{\mathbf{q}}_i \bar{\mathbf{q}}_i^T.$$

For simplicity of argument, let us assume  $Q = I$  (we can get to this scenario by an appropriate change of variables.)

**Example:**

$$\nabla f(\bar{\mathbf{x}}) = \begin{bmatrix} 1 \\ -3 \\ 2 \end{bmatrix}, \quad \nabla^2 f(\bar{\mathbf{x}}) = \text{diag}(10, 3, -1) \Rightarrow \bar{\mathbf{p}}^N = \begin{bmatrix} -0.1 \\ 1 \\ 2 \end{bmatrix}$$

and  $\nabla f(\bar{\mathbf{x}})^T \bar{\mathbf{p}}^N = 0.90$ , hence  $\bar{\mathbf{p}}^N$  is **not a descent direction**.  
(continued...)

## Eigenvalue Modification

2 of 6

**Idea#1:** Replace negative eigenvalues by some positive number  $\delta$ , e.g.  
 $\delta = \sqrt{\epsilon^{\text{mach}}}$

In 32-bit double precision (and Matlab)  $\epsilon^{\text{mach}} \approx 10^{-16}$ , so  $\delta = 10^{-8}$  seems like a reasonable choice(?) We can express the Hessian modification as

$$B_k = \sum_{i=1}^2 \lambda_i \bar{\mathbf{q}}_i \bar{\mathbf{q}}_i^T + \delta \bar{\mathbf{q}}_3 \bar{\mathbf{q}}_3^T \quad \left[ = \sum_{i=1}^n \max(\lambda_i, \delta) \bar{\mathbf{q}}_i \bar{\mathbf{q}}_i^T \right]$$

We now have

$$B_k = \text{diag}(10, 3, 10^{-8}) \Rightarrow \bar{\mathbf{p}} \approx \begin{bmatrix} -0.1 \\ 1 \\ -200,000,000 \end{bmatrix}$$

We notice that  $\bar{\mathbf{p}}$  is approximately parallel to  $\bar{\mathbf{q}}_3$ , and **huge**...

## Eigenvalue Modification

The long step length violates the spirit of Newton's method — recall that the quadratic convergence properties come from a **local** argument with the Taylor expansion.

**Idea#2:** Replace negative eigenvalues by  $-\lambda_i$

Now  $B_k = \text{diag}(|\lambda_1|, |\lambda_2|, \dots, |\lambda_n|)$ , and in our example we get

$$\bar{\mathbf{p}} = \begin{bmatrix} -0.1 \\ 1 \\ -2 \end{bmatrix}, \quad \nabla f(\bar{\mathbf{x}})^T \bar{\mathbf{p}} = -7.1, \quad \text{descent direction!}$$

This seems to work?!?

It may reorder the eigenvalues (and thus the “importance” / ordering of subspaces), *i.e.*

$$\lambda_1 < \lambda_2 < \lambda_3, \quad \text{but} \quad |\lambda_2| < |\lambda_1| < |\lambda_3|.$$



## Eigenvalue Modification

Let's reconsider Idea#1, what went wrong? When we solve  $B\bar{\mathbf{p}} = -\nabla f(\bar{\mathbf{x}})$  we get

$$\bar{\mathbf{p}} = -B^{-1}\nabla f(\bar{\mathbf{x}}) = -\sum_{i=1}^2 \frac{1}{\lambda_i} \bar{\mathbf{q}}_i (\bar{\mathbf{q}}_i^T \nabla f(\bar{\mathbf{x}})) - \frac{1}{\delta} \bar{\mathbf{q}}_3 (\bar{\mathbf{q}}_3^T \nabla f(\bar{\mathbf{x}})),$$

it's clearly the right-most term that makes us violate the spirit of Newton's method.

We could simply just drop this term (*i.e.* ignore the subspace corresponding to negative eigenvalues), or

Select  $\delta$  so that we ensure that the step length is not excessive (**trust-region** flavor!).

**Bad news:** There is no accepted “best” way of modifying the Hessian in this manner.

## Eigenvalue Modification

If we for a moment “forget” about the issue of selecting  $\delta$  so that the step length is reasonable, we can ask the question “**what is the smallest change to  $A$ , which gives us an positive definite matrix  $B$ ?**”

The answer depends on how we measure... Two standard measures are the **Frobenius norm**  $\|A\|_F$ , and the **Euclidean norm**  $\|A\|$

$$\|A\|_F^2 = \sum_{i,j} a_{ij}^2, \quad \|A\| = \max_{\|\bar{x}\|=1} \bar{x}^T A \bar{x} = \max |\text{eig}(A)|.$$

If we use the Frobenius norm, the smallest change is of the type “**change negative eigenvalues to small positive ones:**”

$$B = A + \Delta A, \text{ where } \Delta A = Q \text{diag}(\tau_i) Q^T, \tau_i = \begin{cases} 0 & \lambda_i \geq \delta \\ \delta - \lambda_i & \lambda_i < \delta \end{cases}$$

## Eigenvalue Modification

If, on the other hand, we use the Euclidean norm the smallest change includes a multiple of the identity matrix, *i.e.* **“shift the eigenvalue spectrum, so all eigenvalues are positive:”**

$$B = A + \Delta A, \quad \text{where} \quad \Delta A = \tau I, \quad \tau = \max(0, \delta - \lambda_{\min}(A))$$

We recognize this type of modification to  $A$  from our discussion on **“Nearly exact solutions to the subproblem”** for trust-region methods (Lecture #9)...

Both constant-diagonal —  $\tau I$  — and “Frobenius-style” —  $\mathbf{Q} \text{diag}(\tau_i) \mathbf{Q}^T$  — modifications are used in production software. Generally they do not rely on an exact spectral decomposition (full computation of the eigenvalues) of the Hessian, but use a cousin of Gaussian Elimination (usually the Cholesky factorization) which allows introduction of modifications indirectly.

$$\mathbf{B} = \mathbf{A} + \tau \mathbf{I}$$

In adding a multiple of the identity matrix, we would like to identify a scalar  $\tau$  so that

$$\tau = \max \left( 0, \delta - \lambda_{\min}(A) \right).$$

Usually we do not have access to  $\lambda_{\min}(A)$ , so we have to use some clever heuristic to get an estimate and generate

$$\begin{cases} \tau = 0 & \text{if } \lambda_{\min}(A) \geq \delta \\ \tau \geq \delta - \lambda_{\min}(A) & \text{if } \lambda_{\min}(A) < \delta \end{cases}$$

It is important not to select a value of  $\tau$  that is unnecessarily large, since this biases the direction toward the steepest descent direction.

$$\mathbf{B} = \mathbf{A} + \tau \mathbf{I}$$

The following algorithm uses the fact that

$$|\lambda_i| \leq \|A\|_F, \quad \forall i = 1, 2, \dots, n$$

it is quite expensive since a new factorization is attempted in each loop, further the generated  $\tau$  may be unnecessarily large.

### Algorithm

```
 $\beta = \|A\|_F, k=0$   
if(  $\min(a_{ii}) > 0$  ) {  $\tau_0 = 0$  } else {  $\tau_0 = \beta/2$  } endif  
while(  $k < \text{maxiter}$  )  
  ATTEMPT (Incomplete) Cholesky Factorization  
   $LL^T = A + \tau_k I$   
  if( successful_factorization ), return(L)  
  else,  $\tau_{k+1} = \max(2\tau_k, \beta/2)$   
  endif  
end(while)
```



$\mathbf{B} = \mathbf{A} + \text{diag}(\bar{\mathbf{d}}^{\text{add}})$  — Breaking Cholesky

It is more efficient to let the Cholesky factorization routine directly modify the matrix  $A$  so that the factorization succeeds.

**What can go wrong in Cholesky factorization?**

We look at the Cholesky factorization in  $LDL^T$ -form — set  $M = LD^{1/2}$  to get to  $MM^T$  form.

**Algorithm: Cholesky Factorization,  $LDL^T$ -form**

```

for j = 1:n
   $c_{jj} = a_{jj} - \sum_{s=1}^{j-1} d_s l_{js}^2$ 
   $\mathbf{d}_j = \mathbf{c}_{jj}$       --- The diagonal entries in  $D$  (must be  $\geq \delta$ )
  for i = (j+1):n
     $c_{ij} = a_{ij} - \sum_{s=1}^{j-1} d_s l_{is} l_{js}$ 
     $l_{ij} = c_{ij} / \mathbf{d}_j$       --- We don't want  $l_{ij}$  to be too large
  end
end
end

```

$\mathbf{B} = \mathbf{A} + \text{diag}(\bar{\mathbf{d}}^{\text{add}})$  — Modifying Cholesky

If we want to require that the matrix  $LDL^T$  is sufficiently positive definite, we simply modify the elements  $d_j$ :

$$\mathbf{d}_j = \mathbf{c}_{jj} \quad \rightarrow \quad \mathbf{d}_j = \max(\mathbf{c}_{jj}, \delta)$$

Usually, we also want to have a bound on the size of the off-diagonal entries of  $M = LD^{1/2}$ , i.e.  $|m_{ij}| \leq \beta$  ( $i > j$ ), we set

$$\theta_j = \max_{j < i \leq n} |c_{ij}|$$

and let

$$d_j = \mathbf{c}_{jj} \quad \rightarrow \quad d_j = \max\left(\mathbf{c}_{jj}, \delta, \left[\frac{\theta_j}{\beta}\right]^2\right)$$

we have

$$|m_{ij}| = |l_{ij} \sqrt{d_j}| = \frac{|c_{ij}|}{\sqrt{d_j}} \leq \frac{|c_{ij}| \beta}{\theta_j} \leq \beta.$$

$\mathbf{B} = \mathbf{A} + \text{diag}(\bar{\mathbf{d}}^{\text{add}})$  — Modifying Cholesky

Finally, we throw in an absolute value on the  $c_{jj}$  term for good measure, and come up with

$$d_j = \max \left( |c_{jj}|, \delta, \left[ \frac{\theta_j}{\beta} \right]^2 \right), \quad d_j^{\text{add}} = d_j - c_{jj}$$

This exactly what the module `choldecomp()` in the old default project does! (With some modifications for computational efficiency — the algorithm generates the factorization directly in  $LL^T$ -form)

---

*Old Default Project*

---

<code>choldecomp()</code>	
Implementation	Theory (here)
<code>maxoffl</code>	$\beta$
<code>minl</code>	$\sqrt{\delta}$
<code>maxadd</code>	$\max(\text{diag}(\bar{\mathbf{d}}^{\text{add}}))$



## Gershgorin Modification

`choldecomp()` and `modelhess()`

## Theorem (Gershgorin's circle theorem)

tells us where the eigenvalues of a matrix are located:

$$|\lambda_i - a_{ii}| \leq \sum_{j \neq i} |a_{ij}|, \quad i = 1, \dots, n.$$

Now given a matrix  $A$ , let  $\mathbf{b}_1$  be the smallest value which makes  $A + \mathbf{b}_1 I$  positive definite from the Gershgorin circle theorem.

Let  $\mathbf{b}_2 = \text{maxadd}$  from `choldecomp()`, and let  $\mu = \min(\mathbf{b}_1, \mathbf{b}_2)$ . Now,  $A + \mu I$  is guaranteed to be positive definite.

---

*Old Default Project*

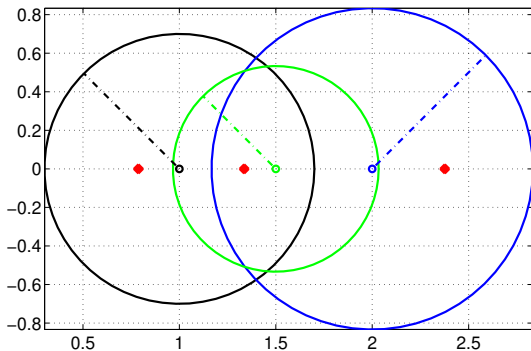
---

This is essentially `modelhess()`. In addition `modelhess()` returns the  $LL^T$ -decomposition of  $A + \mu I$ , and there are tests prior to the first call to `choldecomp()` which takes care of negative diagonal elements of  $A$  and large off-diagonal elements of  $A$ .

Note that `modelhess()` is similar to the algorithm on slide #13, but requires *at most* two calls to a Cholesky factorization algorithm.

## Gershgorin's Circle Theorem: Illustration

1 of 2

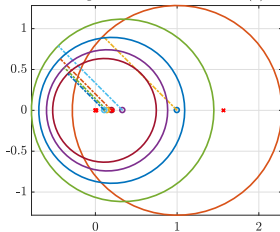


$$A = \begin{bmatrix} 1 & 1/2 & 1/5 \\ 1/2 & 2 & 1/3 \\ 1/5 & 1/3 & 3/2 \end{bmatrix}, \quad \lambda(A) = \{0.7875, 1.3363, 2.3762\}$$

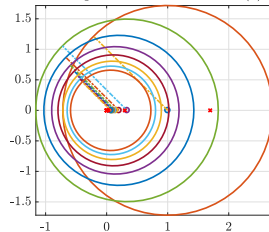
# Gershgorin's Circle Theorem: Illustration

2 of 2

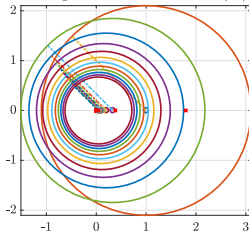
Gershgorin's Theorem for `hilb(5)`



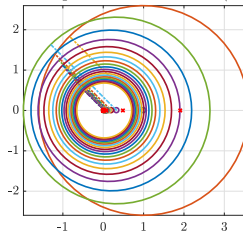
Gershgorin's Theorem for `hilb(8)`



Gershgorin's Theorem for `hilb(12)`



Gershgorin's Theorem for `hilb(20)`



## Project Expectation and Deliverables

Clarified

- Solve a larger optimization problem (see e.g. the “examples of past projects” handout from last time.
- You can look at different types of methods; performance for different test functions, etc... **BEST:** something relevant to your thesis project.
- **Deliverables:**
  - *Project Proposal* — 1 page, Due 11/16/2018
  - *Presentation* — 12–15 minutes, in-class (starting 12/10/2018)
  - *email* — presentation + code(s). (after presentation)

## Next...

- Practical Newton Methods: Trust-Region Newton Methods
- Calculating Derivatives: Finite Differencing & Automatic Differentiation
- Quasi-Newton Methods...

# Index

matrix norm

Euclidean, 10

Frobenius, 10

theorem

Gershgorin's circle theorem, 17