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
1 Recap
   - Robust Inexact Newton Methods

2 Hessian Modifications
   - Eigenvalue Modification
   - $B = A + \tau I$
   - 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 ($\nabla^2 f(\bar{x}_k)$) is not positive definite, and we get a CG-internal search direction for which $\bar{p}^T \nabla^2 f(\bar{x}_k) \bar{p} \leq 0$, 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:* $\bar{p}^T \nabla^2 f(\bar{x}_k) \bar{p}$ small and positive $\leadsto$ long step.
We also discussed how to specify the forcing sequence \( \{ \eta^{(k)} \} \) for the tolerance termination criterion \( \| \bar{r}_k \| \leq \eta^{(k)} \| \nabla f(\bar{x}_k) \| \) 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 \( \nabla^2 f(\bar{x}_k) + E_k \) so that the resulting matrix is sufficiently positive definite; today we take a closer look at this approach.
We look at modifying the Hessian matrix $\nabla^2 f(\bar{x}_k)$ by either explicitly or implicitly adding a matrix $E_k$ (usually a multiple of the identity matrix) so that the resulting matrix

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

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

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

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

**Example:**

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

and $\nabla f(\bar{x})^T \bar{p}^N = 0.90$, hence $\bar{p}^N$ is **not a descent direction**. (continued...)
**Idea #1:** Replace negative eigenvalues by some positive number $\delta$, e.g. $\delta = \sqrt{\varepsilon_{\text{mach}}}$

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

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

We now have

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

We notice that $\bar{p}$ is approximately parallel to $\bar{q}_3$, and huge...
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 $B_k = \text{diag}(|\lambda_1|, |\lambda_2|, \ldots, |\lambda_n|)$, and in our example we get

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

This seems to work?!?

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

$$\lambda_1 < \lambda_2 < \lambda_3, \quad \text{but} \quad |\lambda_2| < |\lambda_1| < |\lambda_3|.$$
Let’s reconsider Idea #1, what went wrong? When we solve $B \bar{p} = -\nabla f(\bar{x})$ we get

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

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.
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:

$$B = A + \Delta A, \quad \text{where} \quad \Delta A = Q \text{diag}(\tau_i) Q^T, \quad \tau_i = \begin{cases} 0 & \lambda_i \geq \delta \\ \delta - \lambda_i & \lambda_i < \delta \end{cases}$$
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” — \( Q \text{diag} (\tau_i) Q^T \) — 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.
In adding a multiple of the identity matrix, we would like to identify a scalar $\tau$ so that

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

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

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

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

\[ |\lambda_i| \leq \|A\|_F, \quad \forall i = 1, 2, \ldots, n \]

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

**Algorithm**

\[
\begin{align*}
\beta &= \|A\|_F, \quad k=0 \\
\text{if( min}(a_{ii}) > 0 ) \{ \tau_0 = 0 \} \quad \text{else} \{ \tau_0 = \beta/2 \} \quad \text{endif} \\
\text{while( } k < \text{maxiter) } \\
\quad \text{ATTEMPT (Incomplete) Cholesky Factorization} \\
\quad LL^T &= A + \tau_k I \\
\quad \text{if( successful\_factorization ), return(L) } \\
\quad \text{else, } \tau_{k+1} &= \max(2\tau_k, \beta/2) \\
\quad \text{endif} \\
\end{align*}
\]
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**

```plaintext
for j = 1:n
    $c_{jj} = a_{jj} - \sum_{s=1}^{j-1} d_s l_j^2$
    $d_j = c_{jj}$  --- The diagonal entries in $D$ (must be $\geq \delta$)
    for i = (j+1):n
        $c_{ij} = a_{ij} - \sum_{s=1}^{j-1} d_s l_i l_j$
        $l_{ij} = c_{ij} / d_j$  --- We don’t want $l_{ij}$ to be too large
    end
end
```
If we want to require that the matrix $LDL^T$ is sufficiently positive definite, we simply modify the elements $d_j$:

$$d_j = c_{jj} \quad \rightarrow \quad d_j = \max(c_{jj}, \delta)$$

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

$$d_j = c_{jj} \quad \rightarrow \quad d_j = \max \left( c_{jj}, \delta, \left( \frac{\theta_j}{\beta} \right)^2 \right)$$

we have

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

<table>
<thead>
<tr>
<th>choldecomp()</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Implementation</strong></td>
</tr>
<tr>
<td>maxoffl</td>
</tr>
<tr>
<td>minl</td>
</tr>
<tr>
<td>maxadd</td>
</tr>
</tbody>
</table>
Theorem (Gershgorin’s circle theorem) tells us where the eigenvalues of a matrix are located:

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

Now given a matrix $A$, let $b_1$ be the smallest value which makes $A + b_1 I$ positive definite from the Gershgorin circle theorem.

Let $b_2 = \max\text{add}$ from choldecomp(), and let $\mu = \min(b_1, b_2)$. Now, $A + \mu I$ is guaranteed to be positive definite.

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

\[ A = \begin{bmatrix} 1 & 1/2 & 1/5 \\ 1/2 & 2 & 1/3 \\ 1/5 & 1/3 & 3/2 \end{bmatrix}, \quad \lambda(A) = \{ 0.7875, 1.3363, 2.3762 \} \]
Gershgorin’s Circle Theorem: Illustration

Gershgorin’s Theorem for \texttt{hilb}(5)

Gershgorin’s Theorem for \texttt{hilb}(8)

Gershgorin’s Theorem for \texttt{hilb}(12)

Gershgorin’s Theorem for \texttt{hilb}(20)
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)
Recap

Hessian Modifications

Eigenvalue Modification
\[ B = A + \tau I \]

Gershgorin Modification

Next...

- Practical Newton Methods: Trust-Region Newton Methods
- Calculating Derivatives: Finite Differencing & Automatic Differentiation
- Quasi-Newton Methods...
Matrix norm
- Euclidean, 10
- Frobenius, 10

Theorem
- Gershgorin's circle theorem, 17