

# Numerical Optimization

Lecture Notes #13 — Practical Newton Methods  
Inexact Newton Steps / Linesearch Newton Methods

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**
  - Nonlinear Conjugate Gradient Methods
- 2 **Practical Newton Methods**
  - Introduction
  - Newton-CG — “Inexact Newton Methods”
- 3 **Line Search Newton-CG Method**
  - Dealing with Negative Curvature
- 4 **Modified Newton’s Method**

## Quick Recap: Nonlinear Conjugate Gradient Methods

Extension of the linear CG to work for non-linear (optimization) problems.

In the first pass (Fletcher-Reeves' Algorithm), we simply replaced all instances of the residual  $\bar{\mathbf{r}}_k$  by the gradient of the objective  $\nabla f(\bar{\mathbf{x}}_k)$ , and the step length  $\alpha_k$  is calculated by a linesearch.

We looked at some modifications, and arrived at the Polak-Ribière PR+CG algorithm, where the  $\beta$  of Fletcher-Reeves is modified

$$\beta_{k+1}^{\text{FR}} = \frac{\nabla f_{k+1}^T \nabla f_{k+1}}{\nabla f_k^T \nabla f_k} \quad \rightarrow \quad \beta_{k+1}^{\text{PR}} = \frac{\nabla f_{k+1}^T (\nabla f_{k+1} - \nabla f_k)}{\nabla f_k^T \nabla f_k}$$

and the final  $\beta$  is  $\beta_{k+1}^+ = \max(\beta_{k+1}^{\text{PR}}, 0)$ .

Finally, periodic restarting, when

$$\frac{\nabla f_k^T \nabla f_{k-1}}{\nabla f_k^T \nabla f_k} \geq \nu \sim 0.1$$

was introduced in order to ensure good convergence.

## Practical Newton Methods: Introduction

We know (e.g. from LECTURE #5) that Newton's method has great local convergence properties. Once we get close to the minimizer  $\bar{\mathbf{x}}^*$  convergence is **quadratic**.

This convergence requires that we start “close enough” to  $\bar{\mathbf{x}}^*$  — in regions far away, where the objective is not convex, all bets are off and the behavior can be quite erratic; we cannot guarantee convergence at all!

### Our present goal:

⇒ To design a Newton-based method which is robust and efficient in “all” cases.

## The Newton Step

We get the Newton step from the symmetric  $n \times n$  linear system

$$\text{The Newton Direction: } \nabla^2 f(\bar{\mathbf{x}}_k) \bar{\mathbf{p}}_k^N = -\nabla f(\bar{\mathbf{x}}_k)$$

For **global convergence** the Newton direction must be a **descent direction**, this is true if the Hessian ( $\nabla^2 f(\bar{\mathbf{x}}_k)$ ) is **positive definite**.

If the Hessian is not positive definite, the Newton direction may be an **ascent direction** and/or extremely long (division by almost zero).

We look at two approaches: The first uses the conjugate gradient method, and gives us the **“Newton-CG”** methods for both line-search and trust-region methods; the second strategy involves modifying the Hessian so that it becomes *“sufficiently positive definite,”* yielding the **“modified Newton method.”**

## Computational Cost

As always, we want to keep the computational cost down.

In Newton-CG, this is accomplished by terminating the computation before an exact solution to

$$\nabla^2 f(\bar{\mathbf{x}}_k) \bar{\mathbf{p}}_k^N = -\nabla f(\bar{\mathbf{x}}_k),$$

has been found. Thus we get an **approximation**  $\bar{\mathbf{p}}_k \approx \bar{\mathbf{p}}_k^N$ , hence the name **“inexact Newton methods.”**

We would like to exploit any special **sparsity structure** in the Hessian in order to solve the linear problem as efficiently as possible.

For now, we assume we **have** access to the Hessian in **analytical** form. We will cover this final issue soon.

## Inexact Newton Steps

If we settle for an inexact (approximate) solution of

$$\nabla^2 f(\bar{\mathbf{x}}_k) \bar{\mathbf{p}}_k^N = -\nabla f(\bar{\mathbf{x}}_k),$$

then we need a measure of how close our approximate solution  $\bar{\mathbf{p}}_k$  is to the exact Newton direction... Another use of the **residual**

$$\bar{\mathbf{r}}_k = \nabla^2 f(\bar{\mathbf{x}}_k) \bar{\mathbf{p}}_k + \nabla f(\bar{\mathbf{x}}_k).$$

Usually we do not want the termination condition for the inexact solution to depend on the size of  $f$ , hence are interested in the **relative size** of the residual, and say that an approximate solution is good enough when

**Termination criterion:**  $\|\bar{\mathbf{r}}_k\| \leq \eta_k \|\nabla f(\bar{\mathbf{x}}_k)\|, \quad \eta_k \in (0, 1)$

The sequence  $\{\eta_k\}$  is known as a **forcing sequence**.

## The Forcing Sequence: Convergence

1 of 3

Selection of the forcing sequence greatly impacts how fast the overall algorithm converges, as illustrated in the following results:

### Theorem

Suppose that the gradient  $\nabla f(\bar{\mathbf{x}})$  is continuously differentiable in a neighborhood  $\mathcal{N}$  of a minimizer  $\bar{\mathbf{x}}^*$ , and assume that the Hessian  $\nabla^2 f(\bar{\mathbf{x}}^*)$  is positive definite. Consider the iteration  $\bar{\mathbf{x}}_{k+1} = \bar{\mathbf{x}}_k + \bar{\mathbf{p}}_k$  where  $\bar{\mathbf{r}}_k(\bar{\mathbf{p}}_k)$  satisfies

$$\|\bar{\mathbf{r}}_k\| \leq \eta_k \|\nabla f(\bar{\mathbf{x}}_k)\|, \quad \eta_k \in (0, 1)$$

and assume that  $\eta_k \leq \eta$  for some  $\eta \in [0, 1)$ . Then, if the starting point  $\bar{\mathbf{x}}_0$  is sufficiently near  $\bar{\mathbf{x}}^*$ , the sequence  $\{\bar{\mathbf{x}}_k\}$  converges to  $\bar{\mathbf{x}}^*$  *linearly*. That is, for all sufficiently large  $k$  we have

$$\|\bar{\mathbf{x}}_{k+1} - \bar{\mathbf{x}}^*\| \leq c \|\bar{\mathbf{x}}_k - \bar{\mathbf{x}}^*\|$$

for some constant  $c \in (0, 1)$ .





## The Forcing Sequence: Convergence

2 of 3

The preceding theorem is neither very exciting, nor very useful (on its own).

The restriction on the forcing sequence is very mild, we are basically just requiring that we make *some* progress in solving the linear system

$$\nabla^2 f(\bar{\mathbf{x}}_k) \bar{\mathbf{p}}_k^N = -\nabla f(\bar{\mathbf{x}}_k).$$

Likewise, the result — **linear convergence** — is good news, but hardly anything that causes us to throw a party!

However, by **carefully selecting** the forcing sequence we get a slightly more exciting result...



## The Forcing Sequence: Convergence

3 of 3

### Theorem

Suppose that the conditions of the previous theorem hold, and assume that the iterates  $\{\bar{\mathbf{x}}_k\}$  generated by the inexact Newton method converge to  $\bar{\mathbf{x}}^*$ . Then the rate of convergence is **superlinear** if  $\eta_k \rightarrow 0$ , and **quadratic** if  $\eta_k = \mathcal{O}(\|\nabla f(\bar{\mathbf{x}}_k)\|)$ .

Now we know exactly how hard we have to work at solving the linear systems in order to achieve certain convergence rates, e.g.

$$\eta_k = \min\left(10^{-3}, \sqrt{\|\nabla f(\bar{\mathbf{x}}_k)\|}\right) \quad \text{Superlinear Convergence}$$

$$\eta_k = \min\left(10^{-3}, \|\nabla f(\bar{\mathbf{x}}_k)\|\right) \quad \text{Quadratic Convergence}$$

**Note:** These results are still **local** — we still have to figure out how to make our algorithms work if not started “close” to  $\bar{\mathbf{x}}^*$ .



We now have the pieces necessary to build **robust Newton methods** with good performance characteristics: first on the menu — the line search Newton-CG method:

### Getting the search direction $\bar{\mathbf{p}}_k$ :

We apply the linear Conjugate Gradient (CG) method to the Newton equations

$$\nabla^2 f(\bar{\mathbf{x}}_k) \bar{\mathbf{p}}_k^N = -\nabla f(\bar{\mathbf{x}}_k),$$

and require that the solution satisfies a termination test of the type

$$\|\bar{\mathbf{r}}_k\| \leq \eta_k \|\nabla f(\bar{\mathbf{x}}_k)\|, \quad \eta_k \in (0, 1).$$

However, if the Hessian is not positive definite this may break...



## Line Search Newton-CG Method

2 of 5

If/When the Hessian is not positive definite we may enter a region of negative curvature; when we do, the CG iteration is terminated in order to guarantee that the generated  $\bar{\mathbf{p}}_k$  is a descent direction:

In search-direction-search, we set  $A = \nabla^2 f(\bar{\mathbf{x}}_k)$ ,  $\bar{\mathbf{b}} = -\nabla f(\bar{\mathbf{x}}_k)$  and then start the CG-iteration:

- (1) The starting point is set to  $\bar{\mathbf{x}}^{(0)} = 0$
- (2) If a **(CG-internal)** search direction  $\bar{\mathbf{p}}^{(i)}$  generated by the CG-iteration satisfies

$$\left[ \bar{\mathbf{p}}^{(i)} \right]^T A \left[ \bar{\mathbf{p}}^{(i)} \right] \leq 0, \quad \text{Negative curvature test}$$

then, if ( $i == 0$ ), set  $\bar{\mathbf{x}}^{(0)} = \bar{\mathbf{b}} = -\nabla f(\bar{\mathbf{x}}_k)$  [STEEPEST DESCENT] and return, otherwise stop immediately and return  $\bar{\mathbf{x}}^{(i)}$ .

- (3) The approximate Newton step  $\bar{\mathbf{p}}_k \stackrel{\text{def}}{=} \bar{\mathbf{x}}^{(i)}$ .



# Line Search Newton-CG Method

3 of 5

## Algorithm: "CG-core"

Given  $A$ ,  $\bar{\mathbf{b}}$ ,  $\eta^{(\kappa)}$ ,  $\bar{\mathbf{x}}_0^{\text{CG}}$ :  $\bar{\mathbf{r}}_0 = A\bar{\mathbf{x}}_0^{\text{CG}} - \bar{\mathbf{b}}$ ,  $\bar{\mathbf{p}}_0 = -\bar{\mathbf{r}}_0$ ,  $k = 0$

while (  $\|\bar{\mathbf{r}}_k\| > \eta^{(\kappa)}\|\bar{\mathbf{b}}\|$  )

$$\alpha_k = \frac{\bar{\mathbf{r}}_k^T \bar{\mathbf{r}}_k}{\bar{\mathbf{p}}_k^T A \bar{\mathbf{p}}_k},$$

Store the vector  $A\bar{\mathbf{p}}_k$   
 and the scalar  $\bar{\mathbf{r}}_k^T \bar{\mathbf{r}}_k$

if  $\bar{\mathbf{p}}_k^T A \bar{\mathbf{p}}_k \leq 0$ ,  $k > 0$  return( $\bar{\mathbf{x}}_k^{\text{CG}}$ )

if  $\bar{\mathbf{p}}_k^T A \bar{\mathbf{p}}_k \leq 0$ ,  $k = 0$  return( $\bar{\mathbf{p}}_0$ )

$$\bar{\mathbf{x}}_{k+1}^{\text{CG}} = \bar{\mathbf{x}}_k^{\text{CG}} + \alpha_k \bar{\mathbf{p}}_k$$

$$\bar{\mathbf{r}}_{k+1} = \bar{\mathbf{r}}_k + \alpha_k A \bar{\mathbf{p}}_k$$

$$\beta_{k+1} = \frac{\bar{\mathbf{r}}_{k+1}^T \bar{\mathbf{r}}_{k+1}}{\bar{\mathbf{r}}_k^T \bar{\mathbf{r}}_k},$$

Save numerator for next iteration!

$$\bar{\mathbf{p}}_{k+1} = -\bar{\mathbf{r}}_{k+1} + \beta_{k+1} \bar{\mathbf{p}}_k$$

end-while(  $k = k + 1$  )

- The  $\bar{\mathbf{p}}_k$ 's are CG-internal search directions, **not** to be confused with the search direction for the optimization algorithm!



## Line Search Newton-CG Method

4 of 5

**Note:** If the CG-core algorithm encounters a **direction of negative curvature** in the first iteration, the steepest descent direction is used.

## Algorithm: Line Search Newton-CG Method

 HW#2 + HW#3<sup>+</sup>

Given  $\bar{\mathbf{x}}_0$ :  $k = 0$

while (  $\bar{\mathbf{x}}_k$  is not a minimum, e.g.  $\|\nabla f(\bar{\mathbf{x}}_k)\| \geq 10^{-6}$  )

$$\bar{\mathbf{p}}_k^{\text{N-CG}} = \text{CG-core}(A = \nabla^2 f(\bar{\mathbf{x}}_k), \bar{\mathbf{b}} = -\nabla f(\bar{\mathbf{x}}_k), \eta^{(\kappa)} = \eta_k, \bar{\mathbf{x}}_0^{\text{CG}} = \bar{\mathbf{0}})$$

$$\alpha_k^{\text{LS}} = \text{linesearch\_Strong\_Wolfe}(\bar{\mathbf{p}}_k^{\text{N-CG}}, \dots)$$

$$\bar{\mathbf{x}}_{k+1} = \bar{\mathbf{x}}_k + \alpha_k^{\text{LS}} \bar{\mathbf{p}}_k^{\text{N-CG}}$$

end-while (  $k = k + 1$  )

Where we specify  $\eta_k$  as discussed earlier, and the linesearch is such that  $\alpha_k$  satisfies the Wolfe, Strong Wolfe, Goldstein, or Armijo backtracking conditions.



**Comments:**

- Nothing is stopping us from basing this on the preconditioned version of CG, in fact that is probably the right thing to do (see other comments)!
- Line Search Newton-CG (LS-N-CG) is well suited for large problems.
- LS-N-CG has one minor weakness — If/When the Hessian is nearly singular, the Newton-CG direction can be excessively long resulting in many function evaluations in the linesearch.
- This weakness is greatly alleviated by preconditioning, *i.e.* implementing LS-N-PCG(M).



Sometimes it is desirable to use a direct linear algebra technique, *i.e.* an efficient cousin of Gaussian Elimination, to solve the Newton equations

$$\nabla^2 f(\bar{\mathbf{x}}_k) \bar{\mathbf{p}}_k^N = -\nabla f(\bar{\mathbf{x}}_k).$$

If/When the Hessian is not positive definite (or close to singular), it can be modified either before or **during** the solution process so that in effect we solve

$$\underbrace{[\nabla^2 f(\bar{\mathbf{x}}_k) + E_k]}_{\text{Sufficiently Positive Definite}} \bar{\mathbf{p}}_k^N = -\nabla f(\bar{\mathbf{x}}_k),$$

where the Hessian modification  $E_k$  is chosen so that the resulting matrix is sufficiently positive definite.





## Algorithm: Line Search Newton with Modification

```

Given  $\bar{x}_0$ :  $k = 0$ 
while (  $\bar{x}_k$  is not a minimum, e.g.  $\|\nabla f(\bar{x}_k)\| \geq 10^{-6}$  )
     $B_k = \text{factorize}(\nabla^2 f(\bar{x}_k) + E_k)$ 
     $\bar{p}_k^{\text{N-mod}} = -B_k^{-1} \nabla f(\bar{x}_k)$ 
     $\alpha_k^{\text{LS}} = \text{linesearch\_Strong\_Wolfe}(\bar{p}_k^{\text{N-mod}}, \dots)$ 
     $\bar{x}_{k+1} = \bar{x}_k + \alpha_k^{\text{LS}} \bar{p}_k^{\text{N-mod}}$ 
end-while (  $k = k + 1$  )
    
```

Where the linesearch is such that  $\alpha_k$  satisfies the Wolfe, Strong Wolfe, Goldstein, or Armijo backtracking conditions.

The **factorization** algorithm is such that  $E_k = 0$  if  $\nabla^2 f(\bar{x}_k)$  is sufficiently positive definite; otherwise chosen so that  $B_k = (\nabla^2 f(\bar{x}_k) + E_k)$  is sufficiently positive definite.

We save the details of Hessian modification for next lecture...

# Index

algorithm

CG-core, 13

linesearch Newton with modification, 17

linesearch Newton-CG, 14

forcing sequence  $\{\eta_k\}$ , 7