THE UNIVERSITY OF CHICAGO

Section 6

Very Large Scale Methods

CHICAGO

6.1 MATRIX-FREE METHODS CONVERGENCE FRAMEWORK

Matrix Free Methods

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

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

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

- 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.

CHICAGO

Framework for Early termination: Inexact Newton Methods

- We modify the original Newton method:

$$\nabla^2 f_k p_k^{\mathrm{N}} = -\nabla f_k. \qquad \nabla^2 f_k p_k \approx -\nabla f_k$$

- The residual:

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

- CG loop termination rule

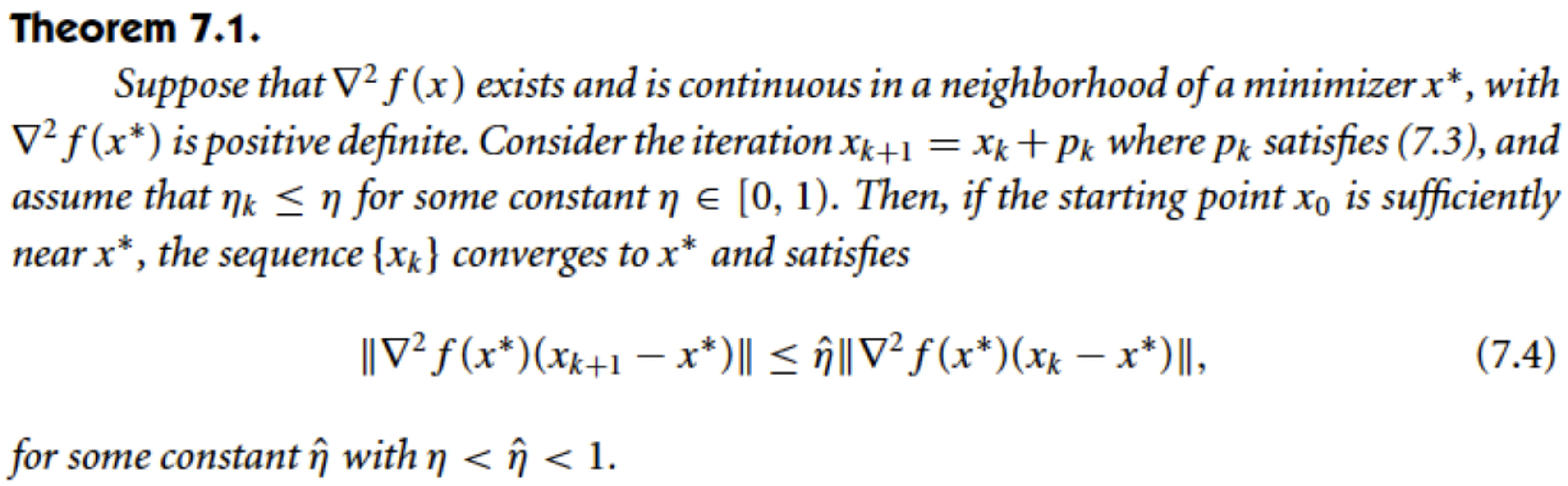
$$\|r_k\| \le \eta_k \|\nabla f_k\|, \qquad \eta_k \to 0$$

$$\boxed{\text{Forcing sequence}}$$

CHICAGO

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 $\eta_k \le \eta$ 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

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

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

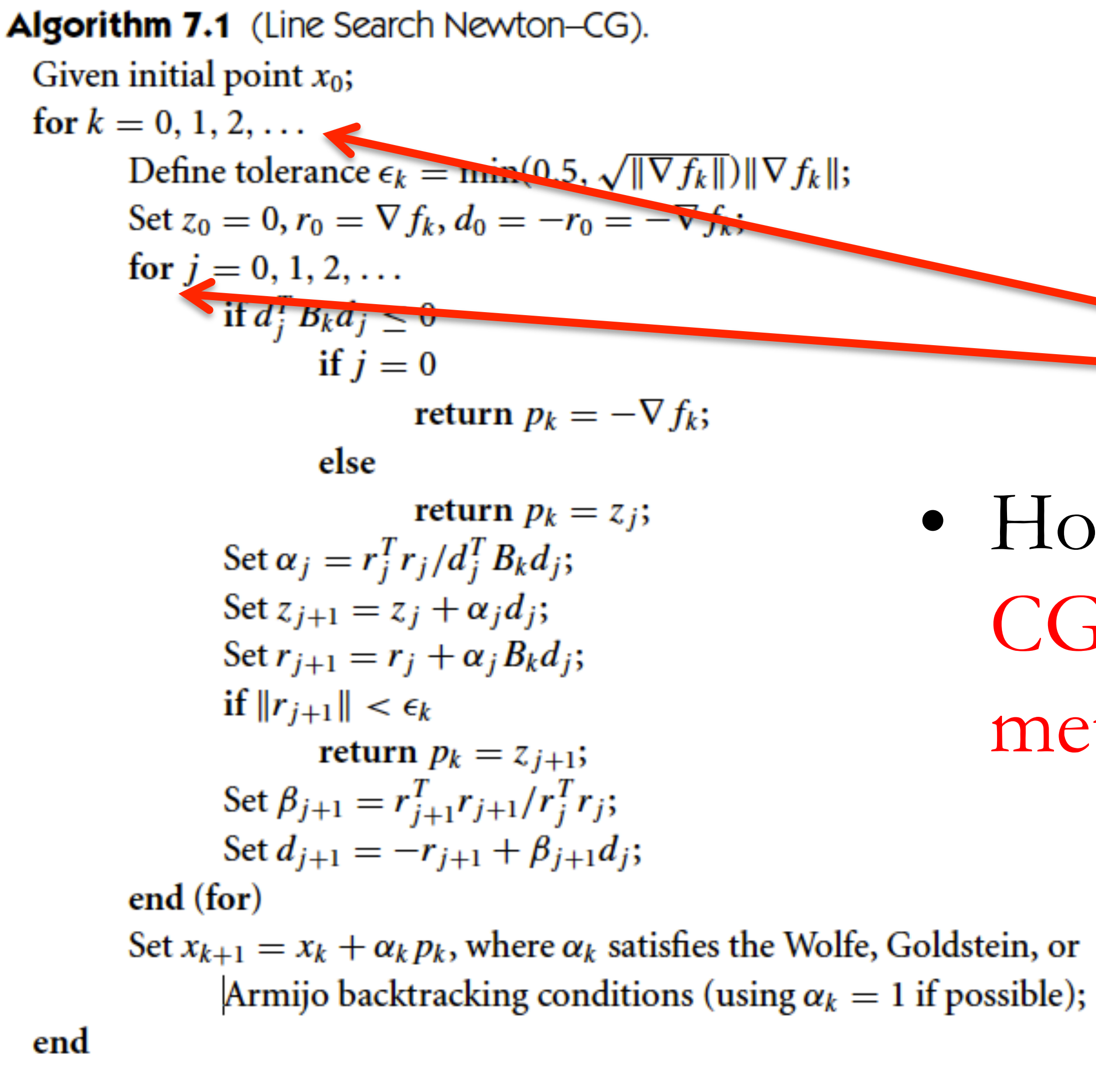
CHICAGO

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

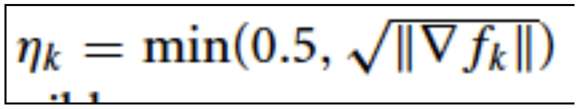


Algorithm 7.1 (Line Search Newton–CG).

```
Given initial point $x_0$;
for $k = 0, 1, 2, \ldots$
    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 $j = 0, 1, 2, \ldots$
        if $d_j^T B_k d_j \le 0$
            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 $\alpha_k$ 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 !!!



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

CHICAGO

CG-Trust Region (STEIHAUG)

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

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 $j = 0, 1, 2, \ldots$
    if $d_j^T B_k d_j \le 0$
        Find $\tau$ 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 $\|z_{j+1}\| \ge \Delta_k$
        Find $\tau \ge 0$ 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.

CHICAGO

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:

$$J(x) = \frac{1}{2}x^T A x + b^T x + c$$

❑ The algorithm is:

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

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

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

where:

$$d_0 = -g_0 \qquad \alpha_k = \frac{< g_k, g_k >}{< d_k, A d_k >} \qquad \beta_k = \frac{< g_{k+1}, g_{k+1} >}{< g_k, g_k >}.$$

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

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

CHICAGO

Lanczos Orthogonalization Procedure

- It orthogonalizes the Krylov space

$$\text{(K2)} \;\Rightarrow S_k = \left\{ r_0, Ar_0, \ldots, A^{k-1} r_0 \right\} = K_k\left(A, r_0\right) \quad (K2)$$

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

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

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

Conjugate Gradients and Lanczos Algorithms

- Let Q_k be the matrix whose columns are $g_i/\|g_i\|$.
- Then $AQ_k = Q_k T_k + g_k e_k^T$

  where T_k is tri-diagonal and $e_k^T = (0,\ldots,0,1)$
- 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 approximated.

Lanczos Iteration

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

CHICAGO

Lanczos iteration

- α_k and β_k are diagonal and subdiagonal entries of symmetric tridiagonal matrix $\boldsymbol{T}_k$
- 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 $\boldsymbol{A}$
- 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

CHICAGO

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).**
- **u = [u_1, u_2, …, u_{n-1}]**
- **d = [d_1, d_2, …, d_{n-1}, d_n]**
- **l = [l_1, l_2, …, l_{n-1}]**

CHICAGO

LU decomposition of Tridiagonal Matrix (Cholesky similar)

$$\begin{bmatrix} 1 & & & & & 0's \\ l1_1 & 1 & & & & \\ & l1_2 & 1 & & & \\ & & \ddots & \ddots & & \\ & & & l1_{n-2} & 1 & \\ 0's & & & & l1_{n-1} & 1 \end{bmatrix} \times \begin{bmatrix} d1_1 & u1_1 & & & & 0's \\ & d1_2 & u1_2 & & & \\ & & d1_3 & u1_3 & & \\ & & & \ddots & \ddots & \\ & & & & d1_{n-1} & u1_{n-1} \\ 0's & & & & & d1_n \end{bmatrix} =$$

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

Thomas Algorithm:

$$l1_{n-1} = l_{n-1} / d1_{n-1}$$
$$d1_n = d_n - (l_{n-1} / d1_{n-1}) * u_{n-1}$$

CHICAGO

Using Lanczos for Optimization

- Solve trust-region (inner loop):

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

- 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)