Outline

1. Linear Conjugate Gradient Methods
   - Recap
   - Conjugate Gradient Algorithms
   - The Effect of Preconditioning — CG vs. PCG(M)

2. Nonlinear Conjugate Gradient Methods
   - New Ideas... Fletcher-Reeves, etc...
   - Practical Considerations
   - Convergence

3. Projects
   - Separate Handouts, etc...
Quick Recap: Linear Conjugate Gradient Methods

We have introduced the Conjugate Gradient (CG) and Preconditioned Conjugate Gradient (PCG) methods for solution of the linear system $A\bar{x} = \bar{b}$, where $A$ is symmetric positive definite.

**Linear CG** is guaranteed to converge in $n$ iterations, but as we have seen, in many cases — eigenvalue clustering and/or $r < n$ distinct eigenvalues, convergence is much faster.

We briefly discussed preconditioning, where we use a simplified version $M \approx A$, and hope that $M^{-1}A \approx I$ has a favorable eigenvalue spectrum. We must be able to solve $M\bar{y} = \bar{r}$ fast.

**Today:**

(i) An example of CG vs. PCG performance.
(ii) Non-linear CG.
(iii) Projects!
The CG Algorithm (version 1.0, “Standard”)

Algorithm: Conjugate Gradient

Given $A$, $\bar{b}$ and $\bar{x}_0$:

$\bar{r}_0 = A\bar{x}_0 - \bar{b}$, $\bar{p}_0 = -\bar{r}_0$, $k = 0$

while ($\|r_k\| > 0$, or other stopping condition)

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

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

$\bar{x}_{k+1} = \bar{x}_k + \alpha_k \bar{p}_k$

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

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

Keep numerator for next step!

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

$k = k + 1$

end-while
Preconditioned CG Algorithm (a.k.a. “PCG(M)”)

Algorithm: PCG

Given $A$, $M = C^T C$, $\bar{b}$ and $\bar{x}_0$: compute $\bar{r}_0 = A\bar{x}_0 - \bar{b}$, $\bar{y}_0 = M^{-1}\bar{r}_0$, $\bar{p}_0 = -\bar{y}_0$, $k = 0$

while ( $\|r_k\| > 0$, or other stopping condition )

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

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

$$
\bar{x}_{k+1} = \bar{x}_k + \alpha_k \bar{p}_k
$$

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

$$
\bar{y}_{k+1} = M^{-1} \bar{r}_{k+1}
$$

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

Save the numerator for next step!

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

$$
k = k + 1
$$

end-while
Problem: Solve $\nabla^2 u(x, y) = f(x, y)$ in the domain $D = \{(x, y) : -1 \leq x, y \leq 1\} - \{(x, y) : (x+1)^2 + (y+1)^2 < 1\}$.

Set $u(x, y) = 0$ on $\Gamma = \partial D$ (Dirichlet Boundary Conditions).
We discretize $\nabla^2 u(x, y)$ by the standard 5-point finite difference approximation of the Laplacian.

We study the numerical solution of the resulting linear system $A\tilde{u} = \tilde{f}$ for varying discretizations of the square (from $2 \times 2$ to $64 \times 64$ grids.)

We look at CG, PCG($M$) with $M$ being the tri-diagonal preconditioner, and PCG($M$) with $M = \tilde{L}\tilde{L}^T$, where $\tilde{L}$ is given by the incomplete (zero fill-in) Cholesky factorization.
Example: CG vs. PCG(M) Performance

**Figure:** We discretize $\nabla^2 u(x, y)$ by the standard 5-point finite difference approximation of the Laplacian on the numerical domain (illustrated on the left with a $48 \times 48$ grid), the corresponding matrix $A$ is illustrated to the right; it has a tridiagonal component, and two additional elements on every row — the bandwidth is not constant due to the cut-out in the domain.
Figure: To the left we see the $L$ given by complete Cholesky factorization — we notice how the entire band fills in, we get a total of 67,071 non-zero entries. To the left we see the $\tilde{L}$ given by incomplete Cholesky factorization — here we only get 5,020 non-zero entries. ($A$ had 8,336 non-zero entries)

We will use the preconditioners $M = \tilde{L}\tilde{L}^T$, and $M = \text{tridiag}(A)$. 

Ex. 4: CG vs. PCG(M) Performance
### Example: CG vs. PCG(M) Performance

<table>
<thead>
<tr>
<th>$n_{\text{GRID}}$</th>
<th>$A$</th>
<th>$A$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$1704 \times 1704$</td>
<td>$3094 \times 3094$</td>
</tr>
</tbody>
</table>

$L = \text{ichol}(A), LL^T \approx A$

<table>
<thead>
<tr>
<th></th>
<th>$A - LL^T |_F / |A|_F$</th>
<th>$\text{cond}(A)$</th>
<th>$\text{cond}(L^{-1}AL^{-T})$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.0900</td>
<td>924.5193</td>
<td>133.4733</td>
</tr>
</tbody>
</table>

$L = \text{chol}(\text{tridiag}(A)), LL^T \approx A$

<table>
<thead>
<tr>
<th></th>
<th>$A - LL^T |_F / |A|_F$</th>
<th>$\text{cond}(A)$</th>
<th>$\text{cond}(L^{-1}AL^{-T})$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.3131</td>
<td>924.5193</td>
<td>463.1769</td>
</tr>
</tbody>
</table>

Peter Blomgren, ⟨blomgren.peter@gmail.com⟩

Nonlinear Conjugate Gradient Methods — (10/24)
Example: CG vs. PCG(M) Performance

Figure: The performance of CG and PCG(M) on our test problem. The discretization of the square \([-1,1]^2\) ranges from \(2 \times 2\) to \(64 \times 64\), which gives us a matrix \(A\) of dimensions ranging from \(3 \times 3\) to \(3094 \times 3094\). The stopping criteria was a relative reduction of \(\|\mathbf{r}\|\) by \(10^{-6}\).
Figure: The solution to our test problem on the 48×48 grid, with the right-hand-side \( f(x, y) = 1 \).
Nonlinear Conjugate Gradient Methods

We now turn our attention to making the CG methods useful for optimization problems (the non-linear situation).

The **Fletcher-Reeves** (CG-FR, published in 1964) extension requires two modifications to the CG algorithm:

1: The computation of the step length $\alpha_k$ is replaced by a line-search which minimizes the non-linear objective $f(\cdot)$ along the search direction $\bar{p}_k$.

2: The instances of the residual $\bar{r}$ (which are just $\nabla \Phi(\cdot)$ for the quadratic objective in standard CG) are replaced by the gradient of the non-linear objective $\nabla f(\cdot)$.

---

Algorithm: Fletcher-Reeves

Given $\bar{x}_0$:
Evaluate $f_0 = f(\bar{x}_0)$, $\nabla f_0 = \nabla f(\bar{x}_0)$.
Set $\bar{p}_0 = -\nabla f_0$, $k = 0$
while ($\|\nabla f_k\| > 0$, ...)
    $\alpha_k = \text{linesearch}(...)$
    $\bar{x}_{k+1} = \bar{x}_k + \alpha_k \bar{p}_k$
    $\nabla f_{k+1} = \text{Evaluate } \nabla f(\bar{x}_{k+1})$
    $\beta_{k+1}^{\text{FR}} = \frac{\nabla f_{k+1}^T \nabla f_{k+1}}{\nabla f_k^T \nabla f_k}$, \hspace{1cm} \text{Save numerator!}
    $\bar{p}_{k+1} = -\nabla f_{k+1} + \beta_{k+1}^{\text{FR}} \bar{p}_k$
    $k = k + 1$
end-while
Comments: The Fletcher-Reeves FR–CG Algorithm

**Sanity check:** If \( f(\bar{x}) \) is a strongly convex quadratic, and \( \alpha_k \) the exact minimizer, then FR–CG reduces to linear CG.

Each iteration requires evaluation of the objective function (for the line-search), and the gradient of the objective. — No Hessian evaluation, nor matrix operations are required. **Good** for large non-linear optimization problems.

If we require that \( \alpha_k \) satisfies the **strong Wolfe conditions**

\[
\begin{align*}
    f(\bar{x}_k + \alpha \bar{p}_k) & \leq f(\bar{x}_k) + c_1 \alpha \bar{p}_k^T \nabla f_k \\
|\bar{p}_k^T \nabla f(\bar{x}_k + \alpha \bar{p}_k)| & \leq c_2 |\bar{p}_k^T \nabla f_k|
\end{align*}
\]

where \( 0 < c_1 < c_2 < \frac{1}{2} \), then FR–CG converges globally.
The following modification to FR-CG was suggested by Polak-Ribière

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

when \( f \) is a strongly convex quadratic, and the line search is exact, the gradients are orthogonal and \( \beta_{k+1}^{\text{FR}} = \beta_{k+1}^{\text{PR}} \).

On general non-linear objectives, an inexact line-searches PR-CG tends to be more robust and more efficient than FR-CG.

One problem: The strong Wolfe conditions do not guarantee that $\bar{p}_k$ is always descent direction for PR–CG. In order to fix this, $\beta$ is defined to be

$$\beta_{k+1}^+ = \max(\beta_{k+1}^{PR}, 0)$$

the resulting algorithm is known as PR+.

There are a number of other choices for $\beta$ in the literature, but they are not (in general) more efficient than Polak-Ribière PR–CG/PR+. 
If the line-search uses quadratic (or cubic) interpolation along the search direction $\bar{p}_k$, then if/when $f(\cdot)$ is a strictly convex quadratic, the step lengths $\alpha_k$ will be the exact 1D-minimizers $\Rightarrow$ the non-linear algorithm reduces to linear CG. [This is Highly Desirable!]

**Restarting:** CG gets its favorable convergence properties from the conjugacy of the search directions near the optimum. If we start “far” from the optimum, the algorithm does not necessarily gain anything from maintaining this conjugacy.

Therefore, we should periodically restart the algorithm, by setting $\beta = 0$ (i.e. taking a steepest-descent step).

The $n$-step convergence is only guaranteed when we start with a steepest-descent step, and the model is quadratic. Hence a restart close to $\bar{x}^*$ will (approximately) guarantee $n$-step convergence.

Peter Blomgren, $\langle$blomgren.peter@gmail.com$\rangle$
Restarting conditions: The most common condition is based on the fact that for the strictly quadratic objective, the residuals are orthogonal. Hence, when two consecutive residuals are “far” from orthogonal

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

a restart is triggered.

The formula

$$\beta_{k+1}^+ = \max(\beta_{k+1}^{PR}, 0)$$

in PR+ can be viewed as a restart-condition. This is not practical since these “restarts” are very infrequent — in practice $\beta_{k+1}^{PR}$ is positive most of the time.
Nonlinear CG: Global Convergence

**Linear CG:** Global convergence properties well understood, and optimal.

**Nonlinear CG:** Convergence properties not so well understood, except in special cases. The behavior is sometimes surprising and bizarre!

We look at some results, under the following non-restrictive assumptions

**Assumptions:**

(i) The level set \( L = \{ \bar{x} \in \mathbb{R}^n : f(\bar{x}) \leq f(\bar{x}_0) \} \) is bounded.

(ii) In some neighborhood \( \mathcal{N} \) of \( L \), the objective function \( f \) is Lipschitz continuously differentiable, i.e. there exists a constant \( L > 0 \) such that

\[
\| \nabla f(\bar{x}) - \nabla f(\bar{y}) \| \leq L \| \bar{x} - \bar{y} \|, \quad \forall \bar{x}, \bar{y} \in \mathcal{N}
\]
Global Convergence: FR-CG

Theorem

Suppose that the assumptions hold, and that FR-CG is implemented with a line search which satisfies the strong Wolfe conditions, with $0 < c_1 < c_2 < \frac{1}{2}$. Then

$$\lim_{k \to \infty} \inf \|\nabla f_k\| = 0.$$  

This does not say that the limit of the sequence of gradients $\{\nabla f_k\}$ is zero; but it does tell us that at least the sequence is not bounded away from zero.

If, however, we restart the algorithm every $n$ steps, we get $n$-step quadratic convergence:

$$\|\bar{x}_{k+n} - \bar{x}^*\| = O(\|\bar{x}_k - \bar{x}^*\|^2).$$
Global Convergence: PR–CG

In practice PR–CG performs better than FR–CG, but we cannot prove a theorem like the one for FR–CG on the previous slide.

The following surprising result can be shown:

Theorem

Consider the Polak-Ribiere PR–CG method with an ideal line search. There exists a twice continuously differentiable objective function \( f : \mathbb{R}^3 \to \mathbb{R} \) and a starting point \( \bar{x}_0 \in \mathbb{R}^3 \) such that the sequence of gradients \( \{\|\nabla f_k\|\} \) is bounded away from zero.

The modification (PR+)

\[
\beta_{k+1}^+ = \max(\beta_{k+1}^{PR}, 0)
\]

fixes this strange behavior, and it is possible to show global convergence for PR+. 