# Numerical Optimization

## Lecture Notes #12
## Conjugate Gradient Methods — Nonlinear CG

Fall 2019

# Outline

# 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 $A\bar{\mathbf{x}} = \bar{\mathbf{b}}$, 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 $M\bar{\mathbf{y}} = \bar{\mathbf{r}}$ 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 $A$, $\bar{\mathbf{b}}$ and $\bar{\mathbf{x}}_0$:

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

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

$$\begin{aligned} \alpha_k &= \frac{\bar{\mathbf{r}}_k^T \bar{\mathbf{r}}_k}{\bar{\mathbf{p}}_k^T A \bar{\mathbf{p}}_k}, \\ \bar{\mathbf{x}}_{k+1} &= \bar{\mathbf{x}}_k + \alpha_k \bar{\mathbf{p}}_k \\ \bar{\mathbf{r}}_{k+1} &= \bar{\mathbf{r}}_k + \alpha_k \mathbf{A}\bar{\mathbf{p}}_k \\ \beta_{k+1} &= \frac{\bar{\mathbf{r}}_{k+1}^T \bar{\mathbf{r}}_{k+1}}{\bar{\mathbf{r}}_k^T \bar{\mathbf{r}}_k}, \\ \bar{\mathbf{p}}_{k+1} &= -\bar{\mathbf{r}}_{k+1} + \beta_{k+1} \bar{\mathbf{p}}_k \\ k &= k + 1 \end{aligned}$$

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

Keep numerator for next step!

end-while

# Preconditioned CG Algorithm (a.k.a. "PCG(M)")

## Algorithm: PCG

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

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

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

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

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

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

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

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

Save the numerator for next step!

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

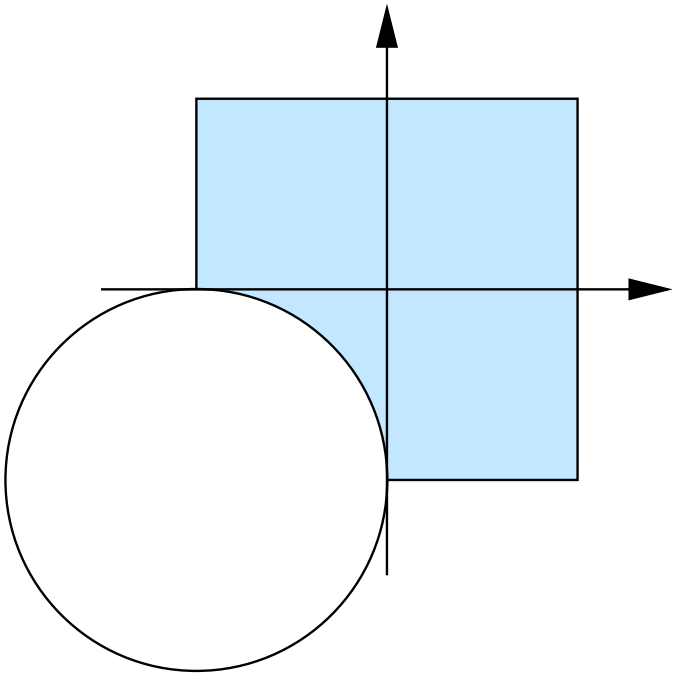
$$k = k + 1$$

end-while

# Example: CG vs. PCG(M) Performance



**Problem:** Solve $\nabla^2 u(x,y) = f(x,y)$ in the domain



$$D = \{(x,y) \,:\, -1 \le x,y \le 1\} - \{(x,y) \,:\, (x+1)^2 + (y+1)^2 < 1\}$$

Set $u(x,y) = 0$ on $\Gamma = \partial D$ (Dirichlet Boundary Conditions).

# Example: CG vs. PCG(M) Performance



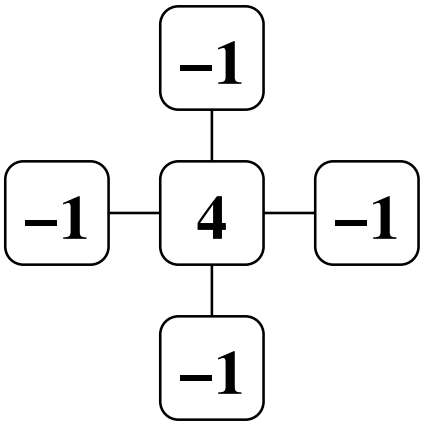



**Figure:** We discretize $\nabla^2 u(x, y)$ by the standard 5-point finite difference approximation of the Laplacian.

We study the numerical solution of the resulting linear system $A\bar{\mathbf{u}} = \bar{\mathbf{f}}$ 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



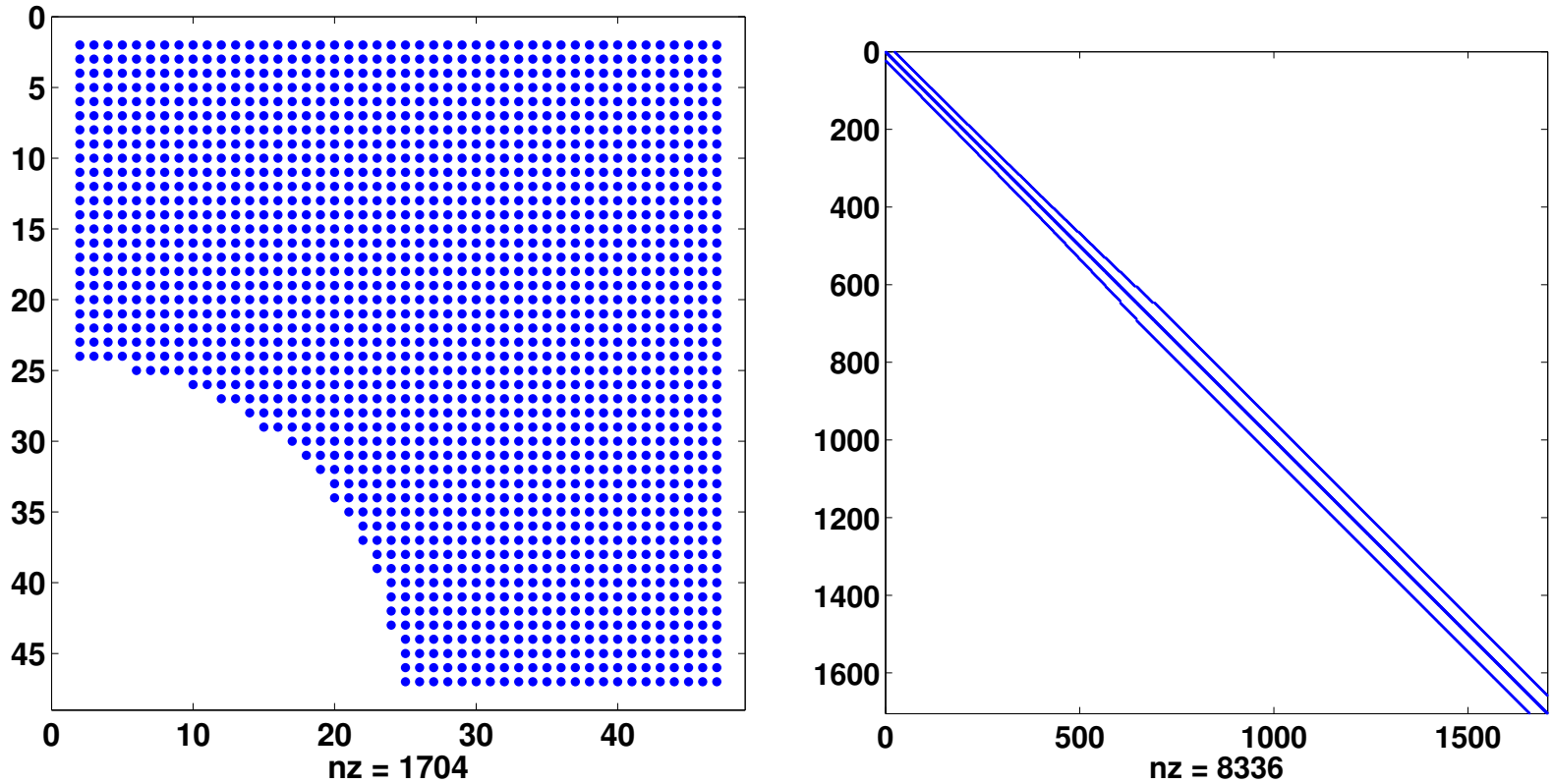



**Figure:** We discretize $\nabla^2 u(x, y)$ by the standard 5-point finite difference approximation of the Laplacian on the numerical domain (illustrated on the left with a $48 \times 48$ grid), the corresponding matrix $A$ is illustrated to the right; it has a tri-diagonal component, and two additional elements on every row — the bandwidth is not constant due to the cut-out in the domain.

# Example: CG vs. PCG(M) Performance



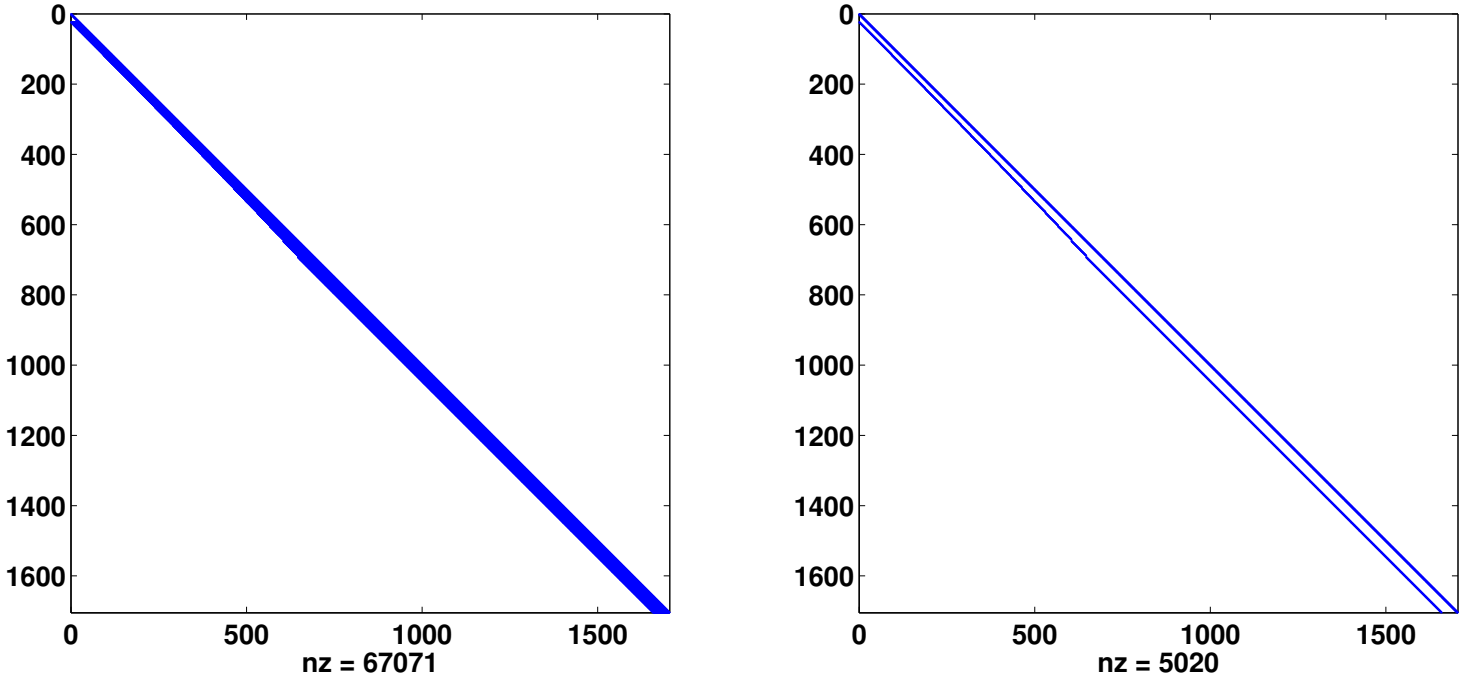



**Figure:** To the left we see the $L$ given by complete Cholesky factorization — we notice how the entire band fills in, we get a total of 67,071 non-zero entries. To the right we see the $\tilde{L}$ given by **incomplete Cholesky factorization** — here we only get 5,020 non-zero entries. ($A$ had 8,336 non-zero entries)

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

# Example: CG vs. PCG(M) Performance



| $n_{\text{GRID}}$ | $48^2$ | $64^2$ |
|---|---|---|
| $A$ | $1704 \times 1704$ | $3094 \times 3094$ |
| `L = ichol(A)`, $LL^T \approx A$ | | |
| $\|A - LL^T\|_F / \|A\|_F$ | 0.0900 | 0.0907 |
| $\text{cond}(A)$ | 924.5193 | 1656.936 |
| $\text{cond}(L^{-1}AL^{-T})$ | 133.4733 | 238.4772 |
| `L = chol(tridiag(A))`, $LL^T \approx A$ | | |
| $\|A - LL^T\|_F / \|A\|_F$ | 0.3131 | 0.3139 |
| $\text{cond}(A)$ | 924.5193 | 1656.936 |
| $\text{cond}(L^{-1}AL^{-T})$ | 463.1769 | 829.6574 |

# Example: CG vs. PCG(M) Performance



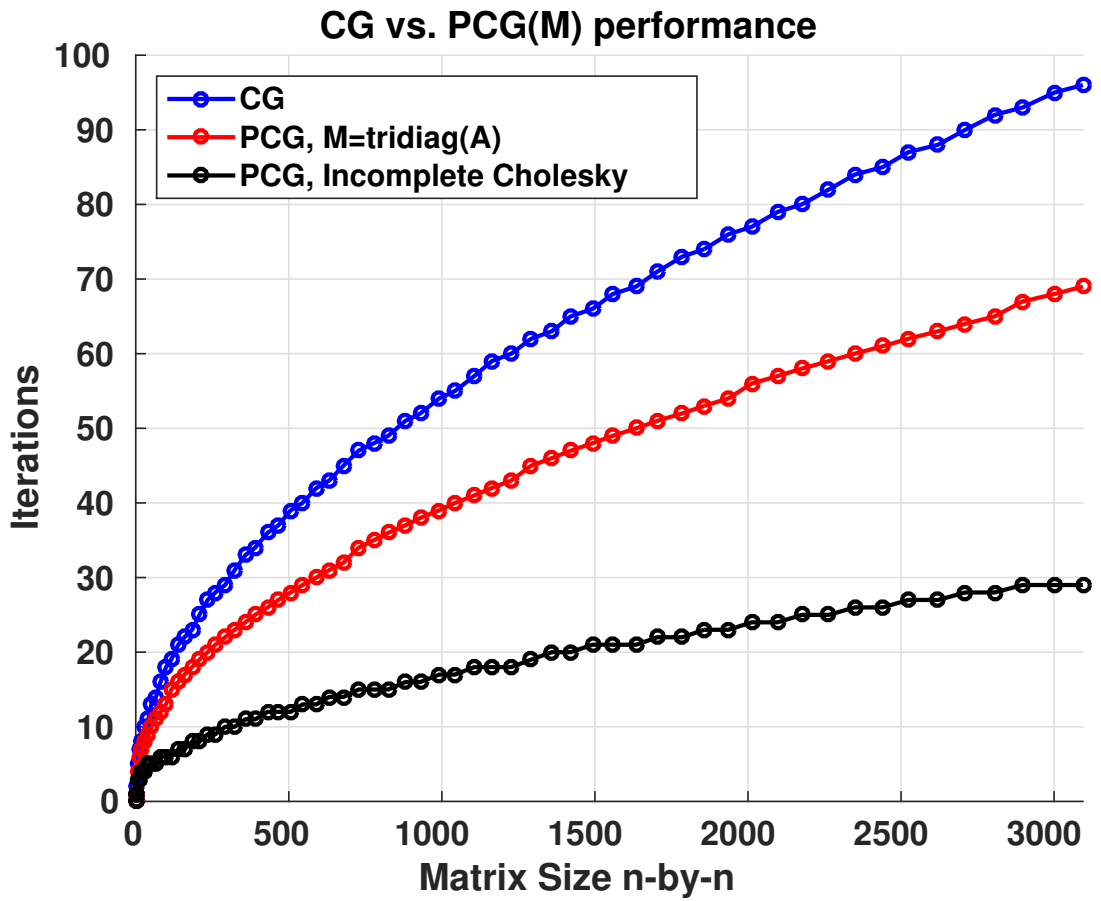



**Figure:** The performance of **CG** and **PCG(M)** on our test problem. The discretization of the square $[-1,1]^2$ ranges from $2 \times 2$ to $64 \times 64$, which gives us a matrix $A$ of dimensions ranging from $3 \times 3$ to $3094 \times 3094$. The stopping criteria was a relative reduction of $\|\bar{\mathbf{r}}\|$ by $10^{-6}$.

# Example: CG vs. PCG(M) Performance



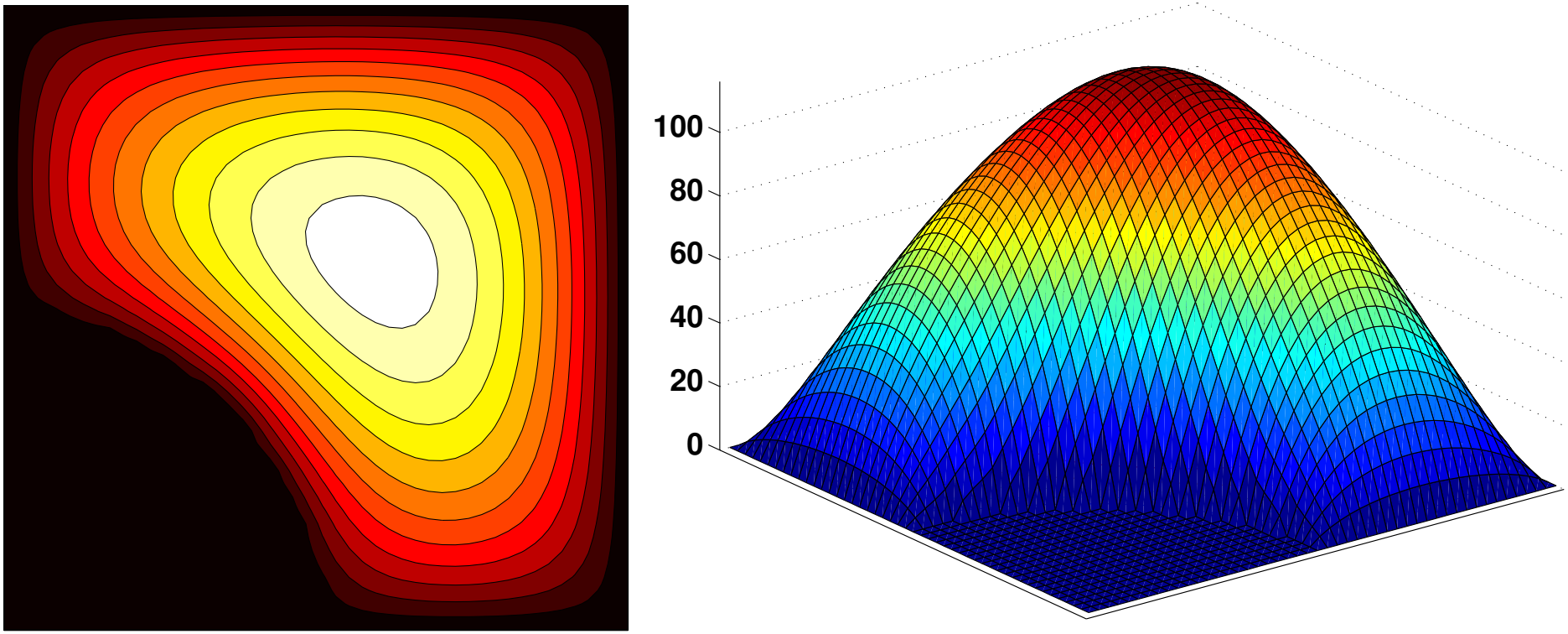



**Figure:** The solution to our test problem on the $48 \times 48$ grid, with the right-hand-side $f(x, y) = 1$.

# 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 $\bar{\mathbf{p}}_k$.

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

---

Fletcher, R., and Reeves, C. M. "Function minimization by conjugate gradients." *The computer journal*, 7, no. 2 (1964), 149-154.

# The Fletcher-Reeves FR-CG Algorithm

## Algorithm: Fletcher-Reeves

Given $\bar{\mathbf{x}}_0$:

Evaluate $f_0 = f(\bar{\mathbf{x}}_0)$, $\nabla f_0 = \nabla f(\bar{\mathbf{x}}_0)$.

Set $\bar{\mathbf{p}}_0 = -\nabla f_0$, $k = 0$

while ( $\|\nabla f_k\| > 0$, ... )

$$\begin{aligned}
\alpha_k &= \text{linesearch}(\ldots) \\
\bar{\mathbf{x}}_{k+1} &= \bar{\mathbf{x}}_k + \alpha_k \bar{\mathbf{p}}_k \\
\nabla f_{k+1} &= \text{Evaluate } \nabla f(\bar{\mathbf{x}}_{k+1}) \\
\beta_{k+1}^{\text{FR}} &= \frac{\nabla f_{k+1}^T \nabla f_{k+1}}{\nabla f_k^T \nabla f_k}, \qquad \text{Save numerator!} \\
\bar{\mathbf{p}}_{k+1} &= -\nabla f_{k+1} + \beta_{k+1}^{\text{FR}} \bar{\mathbf{p}}_k \\
k &= k+1
\end{aligned}$$

end-while

# Comments: The Fletcher-Reeves FR-CG Algorithm

**Sanity check:** If $f(\bar{\mathbf{x}})$ 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**

$$\begin{aligned} f(\bar{\mathbf{x}}_k + \alpha\bar{\mathbf{p}}_k) &\leq f(\bar{\mathbf{x}}_k) + c_1\alpha\bar{\mathbf{p}}_k^T\nabla f_k \\ |\bar{\mathbf{p}}_k^T\nabla f(\bar{\mathbf{x}}_k + \alpha\bar{\mathbf{p}}_k)| &\leq c_2|\bar{\mathbf{p}}_k^T\nabla f_k| \end{aligned}$$

where $0 < c_1 < c_2 < \frac{1}{2}$, then FR-CG converges globally.

# Variants: The Polak-Ribière (PR-CG) Method



The following modification to FR-CG was suggested by Polak-Ribière

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

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.

---

Polak, Elijah, and Gerard Ribiere. "Note sur la convergence de méthodes de directions conjugués." *Revue française d'informatique et de recherche opérationnelle.* Série rouge 3, no. 16 (1969): 35-43.

## Variants: The Polak-Ribière (PR-CG) Method



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

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

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+.

# Practical Considerations



If the line-search uses quadratic (or cubic) interpolation along the search direction $\bar{\mathbf{p}}_k$, 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 $\bar{\mathbf{x}}^*$ will (approximately) guarantee $n$-step convergence.

# Practical Considerations: Restarting Conditions



**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

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

in PR+ can be viewed as a restart-condition. This is not practical since these "restarts" are very infrequent — in practice $\beta_{k+1}^{\mathrm{PR}}$ 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 $\mathcal{L} = \{\bar{\mathbf{x}} \in \mathbb{R}^n : f(\bar{\mathbf{x}}) \leq f(\bar{\mathbf{x}}_0)\}$ is bounded.

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

$$\|\nabla f(\bar{\mathbf{x}}) - \nabla f(\bar{\mathbf{y}})\| \leq L\|\bar{\mathbf{x}} - \bar{\mathbf{y}}\|, \quad \forall \bar{\mathbf{x}}, \bar{\mathbf{y}} \in \mathcal{N}$$

# 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*

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

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:

$$\|\bar{\mathbf{x}}_{k+n} - \bar{\mathbf{x}}^*\| = \mathcal{O}(\|\bar{\mathbf{x}}_k - \bar{\mathbf{x}}^*\|^2).$$

## 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:

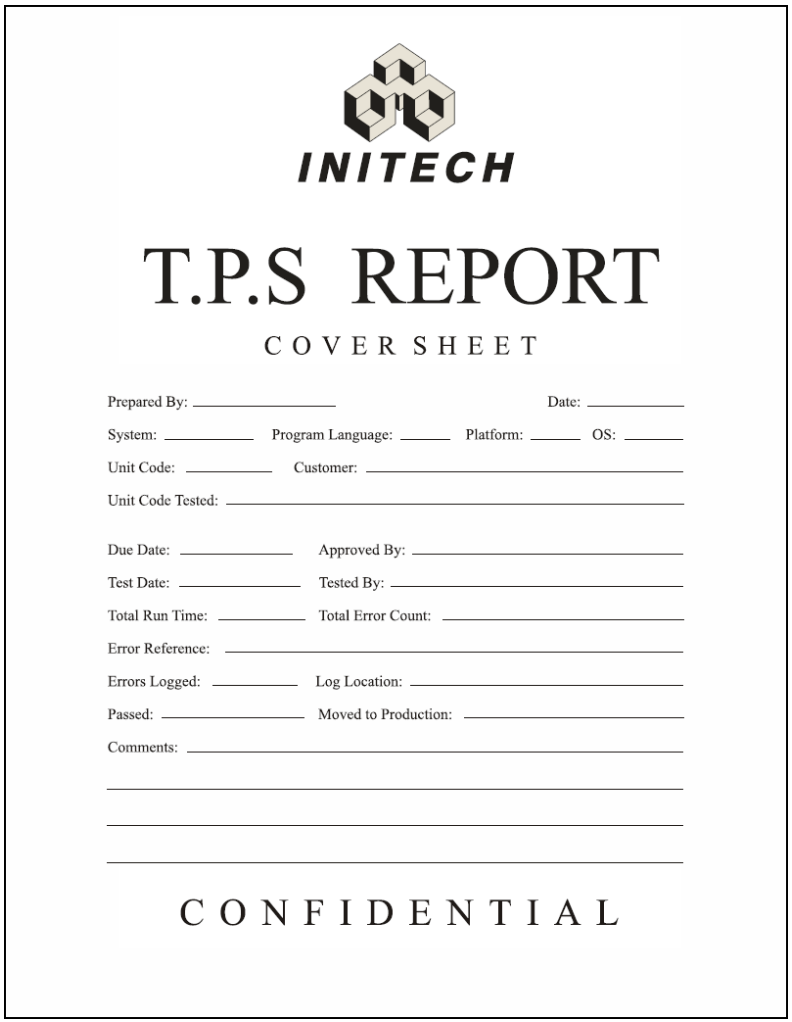
**Theorem**

*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* $\bar{\mathbf{x}}_0 \in \mathbb{R}^3$ *such that the sequence of gradients* $\{\|\nabla f_k\|\}$ *is* **bounded away from zero**.

The modification (PR+)

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

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



INITECH

# T.P.S REPORT

COVER SHEET

Prepared By: _____ Date: _____

System: _____ Program Language: _____ Platform: _____ OS: _____

Unit Code: _____ Customer: _____

Unit Code Tested: _____

Due Date: _____ Approved By: _____

Test Date: _____ Tested By: _____

Total Run Time: _____ Total Error Count: _____

Error Reference: _____

Errors Logged: _____ Log Location: _____

Passed: _____ Moved to Production: _____

Comments: _____

CONFIDENTIAL

# Index