Conjugate Gradient Methods

Saurav Samantaray

Department of Mathematics

Indian Institute of Technology Madras

March 1, 2024

Conjugate Gradient Methods

- They are among the most useful techniques for solving large linear systems of equations.
- They can be adapted to solve non-linear optimisation problems.
- The linear conjugate gradient method is an alternative to Gaussian elimination that is well suited for solving large scale problems.
- Linear conjugate gradient method was proposed by Hestenes and Stiefel in 1950.
- A Key feature of these algorithms is, they require no matrix storage and are faster than the steepest descent method.

Linear Conjugate Gradient Method

The linear conjugate gradient method is an iterative method for solving linear system of equations

$$Ax = b \tag{1}$$

where A is an $n \times n$ symmetric positive definite matrix.

- The above problem of solving a linear system of equations can be equivalently stated as a minimisation problem:

$$\min_x \phi(x) := \frac{1}{2}x^T Ax - b^T x \tag{2}$$

Remark

Both (1) and (2) have the same unique solution.

Linear Conjugate Gradient Method

- The equivalence of both the problems allows us to view conjugate gradient methods either as an algorithm for solving linear systems or as a technique for minimising convex quadratic functions.
- The residual r of the linear system (1) is defined as:

$$r(x) := Ax - b \tag{3}$$

- Note that the gradient of ϕ is:

$$\nabla\phi = r(x) \tag{4}$$

- In particular at $x = x_k$

$$r_k = r(x_k) = Ax_k - b$$

Conjugate Direction Methods

- Generates a set of vectors with a property known as conjugacy.
- The vectors are manufactured, in a very economical fashion.

Conjugacy

A set of non-zero vectors $\{p_0, p_1, \cdots, p_l\}$ is said to be conjugate with respect to the symmetric, positive definite matrix A if

$$p_i^T A p_j = 0 \qquad \text{for,} \quad i \neq j \tag{5}$$

- Any set of vectors satisfying this property is also linearly independent.

Conjugate Direction Methods

- The objective function $\phi(.)$ can be minimised in n steps by successively minimising it along the individual directions in a conjugate set.
- Let $x_0 \in \mathbb{R}^n$ and a set of conjugate directions $\{p_0, p_1, \cdots, p_{n-1}\}$, the sequence of iterates is generated as:

$$x_{k+1} = x_k + \alpha_k p_k \tag{6}$$

- Where α_k is the one-dimensional minimiser of the quadratic function $\phi(.)$ along $x_k + \alpha p_k$, and can be obtained explicitly as:

$$\alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k} \tag{7}$$

Convergence of Conjugate Direction Methods

Theorem

For any $x_0 \in \mathbb{R}^n$ the sequence $\{x_k\}$ generated by the conjugate direction algorithm converges to the solution x^* of the linear system (1) in at most n steps.

Sketch of the Proof:

- Since the directions $\{p_i\}$ are linearly independent, they must span the whole space $\mathbb{R}^n$.
- Therefore, the difference between x_0 and the solution x^* can be written in the following way:

$$x^* - x_0 = \sigma_0 p_0 + \sigma_1 p_1 + \ldots + \sigma_{n-1} p_{n-1},$$

for some choice of scalars σ_k.

Convergence of Conjugate Direction Methods

- By premultiplying this expression by $p_k^T A$ and using the conjugacy property, we obtain:

$$\sigma_k = \frac{p_k^T A(x^* - x_0)}{p_k^T A p_k} \tag{8}$$

- We now establish the result by showing that these coefficients σ_k coincide with the step lengths α_k.
- If x_k is generated by the conjugate direction algorithm, we have

$$x_k = x_0 + \alpha_0 p_0 + \alpha_1 p_1 + \ldots + \alpha_{k-1} p_{k-1}.$$

- By premultiplying this expression by $p_k^T A$ and using the conjugacy property, we have that

$$p_k^T A(x_k - x_0) = 0,$$

Convergence of Conjugate Direction Methods

- Therefore,

$$p_k^T A(x^* - x_0) = p_k^T A(x^* - x_k) = p_k^T (b - Ax_k) = -p_k^T r_k$$

- By comparing the above relation with (7) and (8), we find that $\sigma_k = \alpha_k$, giving the result.

Remark

If the matrix A is diagonal, the contours of the function $\phi(.)$ are ellipses whose axes are aligned with the co-ordinate directions $e_1, e_2, \cdots, e_n$.

Convergence of Conjugate Direction Methods

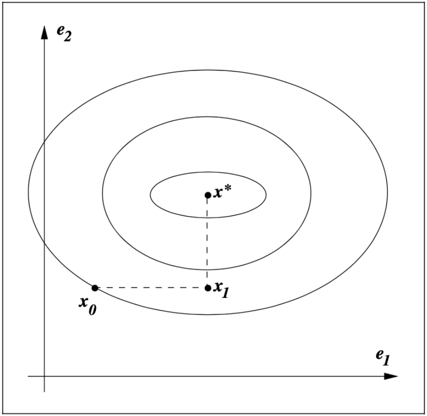
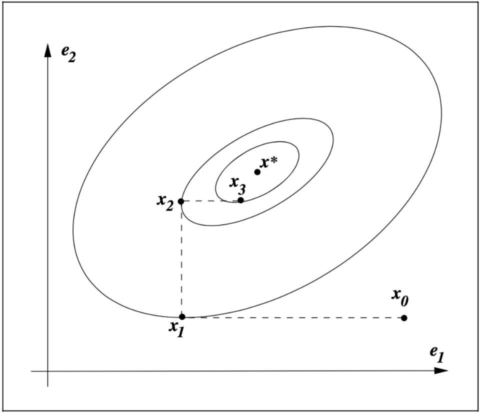


Figure: Successive minimizations along the coordinate directions find the minimizer of a quadratic with a diagonal Hessian in n iterations.

Convergence of Conjugate Direction Methods

- Find the minimiser of this function by performing one-dimensional minimisations along the coordinate directions $e_1, e_2, \cdots, e_n$ in turn.
- When A is not diagonal, its contours are still elliptical, but they are usually no longer aligned with the coordinate directions.
- Successive minimization along these directions in turn no longer leads to the solution in n iterations.

Convergence of Conjugate Direction Methods



Convergence of Conjugate Direction Methods

- The nice behaviour of Figure 1 can be obtained if we transform the problem to make A diagonal and then minimize along the coordinate directions.
- We transform the problem by defining new variables $\hat{x}$ as:

$$\hat{x} = S^{-1}x \tag{9}$$

- S is the $n \times n$ matrix defined by

$$S = [p_0, p_1, \ldots, p_{n-1}]$$

- The quadratic ϕ defined by (2) now becomes:

$$\hat{\phi}(\hat{x}) := \phi(S\hat{x}) = \frac{1}{2}\hat{x}^T(S^TAS)\hat{x} - (S^Tb)^T\hat{x}. \tag{10}$$

Convergence of Conjugate Direction Methods

- By conjugacy property (5), the matrix S^TAS is diagonal.
- The minimising value of $\hat{\phi}$ can be found by performing n one-dimensional minimisations along the coordinate directions of $\hat{x}$.
- The coordinate search strategy applied to $\hat{\phi}$ is equivalent to the conjugate direction algorithm (6)-(7).
- The conjugate direction algorithm terminates in at most n steps.

Convergence of Conjugate Direction Methods

- When the Hessian matrix is diagonal, each coordinate minimisation correctly determines one of the components of the solution x^* .
- After k one-dimensional minimisations, the quadratic has been minimized on the subspace spanned by $e_1, e_2, \dots, e_k$.
- The following theorem proves this result for the general case in which the Hessian of the quadratic is not necessarily diagonal.

Expanding Subspace Minimization

$$r_{k+1} = r_k + \alpha_k A p_k \tag{11}$$

Theorem(Expanding Subspace Minimization)

Let $x_0 \in \mathbb{R}^n$ be any starting point and suppose that the sequence $\{x_k\}$ is generated by the conjugate direction algorithm (6)-(7). Then

$$r_k^T p_i = 0, \qquad \text{for} \quad i = 0, 1, \ldots, k-1, \tag{12}$$

and x_k is the minimiser of $\phi(x) = \frac{1}{2}x^T Ax - b^T x$ over the set

$$\{x | x = x_0 + \text{span}\{p_0, p_1, \cdots, p_{k-1}\}\} \tag{13}$$

- That is, the method minimizes ϕ piece-wise, one direction at a time.
- The current residual r_k is orthogonal to all previous search direction.

How to obtain conjugate directions??

- The discussion applies to a conjugate direction method (6)-(7) based on any choice of the conjugate direction set $\{p_0, p_1, \ldots, p_{n-1}\}$.
- There are many ways to choose the set of conjugate directions.
- The eigenvectors $\{v_1, v_2, \ldots, v_n\}$ of A are mutually orthogonal as well as conjugate with respect to A.
- For large-scale applications computation of the complete set of eigenvectors requires an excessive amount of computation.
- One could modify the Gram–Schmidt orthogonalisation process to produce a set of conjugate directions rather than a set of orthogonal directions.
- The Gram–Schmidt approach is also expensive, since it requires us to store the entire direction set.

Conjugate Gradient Method

- The conjugate gradient method is a conjugate direction method with a very special property.
- In generating its set of conjugate vectors, it can compute a new vector p_k by using only the previous vector p_{k-1}.
- Does not need to know all the previous elements $p_0, p_1, \ldots, p_{k-2}$ of the conjugate set, p_k is automatically conjugate to these vectors.
- Requires little storage and computation.

Conjugate Gradient Method

- The direction p_k is chosen to be a linear combination of the negative residual $-r_k$ and the previous direction p_{k-1}:

$$p_k = -r_k + \beta_k p_{k-1} \tag{14}$$

- The scalar β_k is to be determined by the requirement that p_{k-1} and p_k must be conjugate with respect to A.
- Note that we want to impose $p_{k-1}^T A p_K = 0$ (the conjugacy condition).
- By pre-multiplying (14) by $p_{k-1}^T A$ and using the above imposition, we have:

$$\beta_k = \frac{r_k^T A p_{k-1}}{p_{k-1}^T A p_{k-1}} \tag{15}$$

Conjugate Gradient Method

- We choose the first search direction p_0 to be the steepest descent direction at the initial point x_0.
- We perform successive one-dimensional minimisations along each of the search directions generated.

Algorithm (CG–Preliminary Version)

```
Given $x_0$;
Set $r_0 \longleftarrow Ax_0 - b(= \nabla\phi(x_0))$, $p_0 \longleftarrow -r_0$, $k \longleftarrow 0$;
while $(r_k \neq 0)$:
    $\alpha_k \longleftarrow -\frac{r_k^T p_k}{p_k^T A p_K}$;
    $x_{k+1} \longleftarrow x_k + \alpha_k p_k$;
    $r_{k+1} \longleftarrow Ax_{k+1} - b$;
    $\beta_{k+1} \longleftarrow \frac{r_{k+1}^T A p_k}{p_k^T A p_k}$;
    $p_{k+1} \longleftarrow -r_{k+1} + \beta_{k+1} p_k$;
    $k \longleftarrow k + 1$
end(while)
```

The algorithm Works

- We present a more efficient version later.

Theorem

Suppose that the kth iterate generated by the conjugate gradient method is not the solution point x^*. The following four properties hold:

$$
\begin{aligned}
r_k^T r_i &= 0, \qquad \text{, for } i = 0, 1, \ldots, k-1, && (16)\\
\operatorname{span}\{r_0, r_1, \ldots, r_k\} &= \operatorname{span}\{r_0, Ar_0, \ldots, A^k r_0\}, && (17)\\
\operatorname{span}\{p_0, p_1, \ldots, p_k\} &= \operatorname{span}\{r_0, Ar_0, \ldots, A^k r_0\}, && (18)\\
p_k^T A p_i &= 0, \qquad \text{for } i = 0, 1, \ldots, k-1. && (19)
\end{aligned}
$$

Therefore the sequence $\{x_k\}$ converges to x^* in at most n steps.

A More Efficient Form of Conjugate Gradient Method

- A slightly more economical version of the CG method can be derived using the results of the previous theorems.
- First we can use the definition of p_{k+1} i.e.

$$p_k = -r_k + \beta_k p_{k-1}$$

and the orthogonality of the residual with the (previous) conjugate directions

$$r_k^T p_i = 0, \quad \text{for } i = 0, 1, \ldots, k-1$$

- Now consider α_k as

$$\begin{aligned} \alpha_k &= -\frac{r_k^T p_k}{p_k^T A p_k} \\ &= -\frac{r_k^T(-r_k + \beta_k p_{k-1})}{p_k^T A p_k} \\ \implies \alpha_k &= \frac{r_k^T r_k}{p_k^T A p_k} \end{aligned}$$

A More Efficient Form of Conjugate Gradient Method

- Second from $r_{k+1} = r_k + \alpha_k A p_k$ we get

$$Ap_k = \frac{1}{\alpha_k}(r_{k+1} - r_k)$$

- Note that β_{k+1} is given by

,

$$\beta_{k+1} = \frac{r_{k+1}^T A p_k}{p_k^T A p_k}$$

- Concentrate on the denominator

$$\begin{aligned} p_k^T A p_k &= \frac{p_k^T}{\alpha_k}(r_{k+1} - r_k) \\ &= -\frac{1}{\alpha_k} p_k^T r_k \qquad (p_k^T r_{k+1} = 0) \\ &= -\frac{1}{\alpha_k}(-r_k^T + \beta_k p_{k-1}^T) r_k \quad (p_k = -r_k + \beta_k p_{k-1}) \\ &= \frac{1}{\alpha_k} r_k^T r_k \qquad (p_{k-1}^T r_k = 0) \end{aligned}$$

A More Efficient Form of Conjugate Gradient Method

- Now the numerator of β_{k+1}

$$\begin{aligned} r_{k+1}^T A p_k &= r_{k+1}^T (\frac{1}{\alpha_k}(r_{k+1} - r_k)) \\ &= \frac{r_{k+1}^T r_{k+1}}{\alpha_k} - \frac{1}{\alpha_k} r_{k+1}^T r_k \\ &= \frac{r_{k+1}^T r_{k+1}}{\alpha_k} - \frac{1}{\alpha_k} r_{k+1}^T (-p_k + \beta_k p_{k-1}) \\ &= \frac{r_{k+1}^T r_{k+1}}{\alpha_k} \qquad (r_{k+1}^T p_k = 0),\ (r_{k+1}^T p_{k-1} = 0) \end{aligned}$$

- Therefore finally we have:

$$\beta_{k+1} = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k} \tag{20}$$

Updated CG algorithm

Algorithm (Refined Version)

```
Given $x_0$;
Set $r_0 \longleftarrow Ax_0 - b(= \nabla\phi(x_0))$, $p_0 \longleftarrow -r_0$, $k \longleftarrow 0$;
while $(r_k \neq 0)$:
    $\alpha_k \longleftarrow \frac{r_k^T r_k}{p_k^T A p_K}$;
    $x_{k+1} \longleftarrow x_k + \alpha_k p_k$;
    $r_{k+1} \longleftarrow r_k + \alpha_k A p_k$;
    $\beta_{k+1} \longleftarrow \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$;
    $p_{k+1} \longleftarrow -r_{k+1} + \beta_{k+1} p_k$;
    $k \longleftarrow k + 1$
end(while)
```

Computation in CG Method

The major computational tasks to be performed at each step are:

- computation of the matrix–vector product Ap_k,
- calculation of the inner products $p_k^T(Ap_k)$ and,
- $r_{k+1}^T r_{k+1}$,
- and calculation of three vector sums.

Remark

The CG method is recommended only for large problems; otherwise, Gaussian elimination or other factorization algorithms such as the singular value decomposition are to be preferred, since they are less sensitive to rounding errors.

Convergence of CG Method

- In exact arithmetic sense the conjugate gradient method will terminate at the solution in at most n iterations.
- When the distribution of the eigenvalues of A has certain favourable features, the algorithm will identify the solution in many fewer than n iterations.

Theorem

If A has only r distinct eigenvalues, then the CG iteration will terminate at the solution in at most r iterations.

Theorem

If A has eigenvalues $\lambda_1 \leq \lambda_2 \leq \ldots \lambda_n$, we have:

$$||x_{k+1} - x^*||_A^2 \leq \left(\frac{\lambda_{n-k} - \lambda_1}{\lambda_{n-k} + \lambda_1} \right) ||x_0 - x^*||_A^2. \qquad (21)$$

Clustered Eigenvalues

- The above theorem can be used to predict the behaviour of the CG method on specific problems.
- Suppose we have the situation plotted in the Figure where the eigenvalues of A consist of m large values,
- with the remaining $n - m$ smaller eigenvalues clustered around 1.

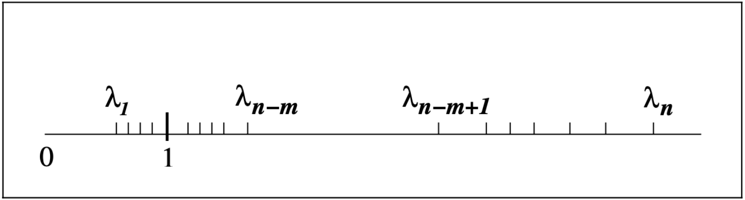


Figure Two clusters of eigenvalues.

Clustered Eigenvalues

- Define $\varepsilon = \lambda_{n-m} - \lambda_1$ the above theorem tells us that after $m+1$ steps of the CG method

$$||x_{m+1} - x^*||_A \approx \varepsilon ||x_0 - x^*||_A$$

- For a small value of ε, we conclude that the CG iterates will provide a good estimate of the solution after only $m+1$ steps.

Clustered VS Uniform

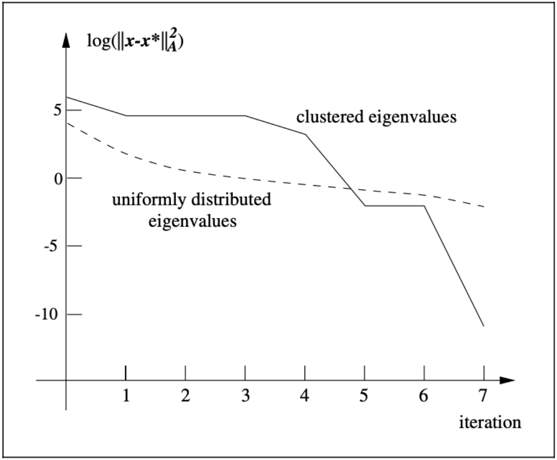


Figure Performance of the conjugate gradient method on (a) a problem in which five of the eigenvalues are large and the remainder are clustered near 1, and (b) a matrix with uniformly distributed eigenvalues.

Clustered VS Uniform

- The problem has has five large eigenvalues with all the smaller eigenvalues clustered between 0.95 and 1.05.
- The Figure compares this behaviour with that of CG on a problem in which the eigenvalues satisfy some random distribution.
- For the problem with clustered eigenvalues the Theorem predicts a sharp decrease in the error measure at iteration 6.
- Note, however, that this decrease was achieved one iteration earlier, illustrating the fact that Theorem gives only an upper bound, and that the rate of convergence can be faster.
- By contrast for the problem with randomly distributed eigenvalues (dashed line), the convergence rate is slower and more uniform.

Preconditioning

- We can aim to choose C such that the eigenvalues of $C^{-T}AC^{-1}$, are more favorable for the convergence theory discussed above.
- We can try to choose C such that the condition number of $C^{-T}AC^{-1}$ is much smaller than the original condition number of A.
- Besides being effective, a good pre-conditioner C should take little storage and allow an inexpensive solution of $Cx = \hat{x}$.
- Finding good pre-conditioners C depends on the problem (the structure of A).

Non-Linear Conjugate Gradient Methods

- CG method can be viewed as a minimization algorithm for the convex quadratic functions of a particular form.
- Can we adapt the approach to minimize general convex functions ??
- or even general non-linear functions f.

The Fletcher and Reeves Method

- Fletcher and Reeves modified the conjugate gradient method for non-linear functions by making two simple changes in the CG algorithm.
- To this end let us consider the updated CG algorithm.

Algorithm (Refined Version)

```
Given $x_0$;
Set $r_0 \longleftarrow Ax_0 - b(= \nabla\phi(x_0))$, $p_0 \longleftarrow -r_0$, $k \longleftarrow 0$;
while $(r_k \neq 0)$:
    $\alpha_k \longleftarrow \frac{r_k^T r_k}{p_k^T A p_K}$;
    $x_{k+1} \longleftarrow x_k + \alpha_k p_k$;
    $r_{k+1} \longleftarrow r_k + \alpha_k A p_k$;
    $\beta_{k+1} \longleftarrow \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$;
    $p_{k+1} \longleftarrow -r_{k+1} + \beta_{k+1} p_k$;
    $k \longleftarrow k + 1$
end(while)
```

The Fletcher and Reeves Method

- First, replace the formula for the step length α_k (which minimizes ϕ along the search direction p_k).
- We need to perform a line search that identifies an approximate minimum of the non-linear function f along p_k.
- Second, the residual r , which is simply the gradient of ϕ must be replaced by the gradient of the non-linear objective f.

Algorithm (FR)

Fletcher and Reeves Method

```
Given $x_0$;
Evaluate $f_0 = f(x_0)$, $\nabla f_0 = \nabla f(x_0)$;
Set $p_0 \longleftarrow -\nabla f_0$, $k \longleftarrow 0$;
while $\nabla f_k \neq 0$:
    Compute $\alpha_k$ and set $x_{k+1} = x_k + \alpha_k p_k$;
    Evaluate $\nabla f_k$;
        $\beta_{k+1}^{FR} \longleftarrow \frac{\nabla f_{k+1}^T \nabla f_{k+1}}{\nabla f_k^T \nabla f_k}$;
        $p_{k+1} \longleftarrow -\nabla f_{k+1} + \beta_{k+1}^{FR} p_k$;
        $k \longleftarrow k + 1$
end(while)
```

Uses no matrix operations, requires only f and ∇f.

Fletcher and Reeves Method

- To make the specification of the algorithm complete, we need to be more precise about the choice of line search parameter α_k.
- Because of the second term in the expression for p_k the search direction p_k may fail to be a descent direction unless α_k satisfies certain conditions.
- By taking the inner-product of p_k with the gradient vector ∇f_k we obtain:

$$\nabla f_k^T p_k = -||\nabla f_k||^2 + \beta_k^{FR} \nabla f_k^T p_{k-1} \tag{22}$$

- If the line search is exact, so that α_{k-1} is a local minimiser of f along the direction p_{k-1}, we have

$$\nabla f_k^T p_{k-1} = 0$$

Fletcher and Reeves Method

- If the line search is not exact, second term in (22) may dominate the first term, and we may have $\nabla f_k^T p_k > 0$,
- implying that p_k is actually a direction of ascent.
- We can avoid this situation by requiring the step length α_k to satisfy the strong Wolfe conditions:

$$f(x_k + \alpha_k p_k) \le f(x_k) + c_1 \alpha_k \nabla f_k^T p_k, \tag{23}$$

$$|\nabla f(x_k + \alpha_k p_k)^T p_k| \le -c_2 \nabla f_k^T p_k, \tag{24}$$

  where $0 < c_1 < c_2 < \frac{1}{2}$.
- It can be shown that the second of the conditions is what ensures that p_k is indeed a descent direction.

Other Variants of FR Method

Polak-Ribière

$$\beta_{k+1}^{PR} = \frac{\nabla f_{k+1}^{T} (\nabla f_{k+1} - \nabla f_k)}{||f_k||^2}$$

- It is identical to the algorithm FR when f is strongly convex quadratic function and the line search is exact;
- as $r_k^T r_i = 0$ for $i = 0, 1, \ldots, k-1$, the gradients are mutually orthogonal, rendering

$$\beta_{k+1}^{FR} = \beta_{k+1}^{PR}$$

Other Variants of FR Method

- PR is the variant of choice in practice.
- For non-linear functions in general, with inexact l.s., PR is empirically more robust and efficient than FR.
- Yet, the strong Wolfe conditions don't guarantee that p_k is a descent direction.
- PR needs a good l.s. to do well.

Other Variants of FR Method

- **Hestenes-Stiefel**:

$$\beta_{k+1}^{HS} = \frac{\nabla f_{k+1}^T(\nabla f_{k+1} - \nabla f_k)}{(\nabla f_{k+1} - \nabla f_k)^T p_k}$$

- **other variant** :

$$\beta_{k+1}^{other} = \frac{||\nabla f_{k+1}||^2}{(\nabla f_{k+1} - \nabla f_k)^T p_k}$$

- β_{k+1} can be defined in other ways that also generalise the quadratic case: for quadratic functions with a p.d. Hessian and exact linear search we have:

$$\beta_{k+1}^{FR} = \beta_{k+1}^{PR} = \beta_{k+1}^{HS} = \beta_{k+1}^{other}$$

  for linear CG (since the successive gradients are mutually perpendicular)

Restarts

- Restarting the iteration every n steps.
- by setting $\beta_k = 0$, i.e., taking a steepest descent step.
- periodically refreshes the algorithm and works well in practice.
- It leads to n-step quadratic convergence:

$$\frac{||x_{k+n} - x^*||}{||x_k - x^*||^2} \leq M$$

.

- intuitively because near the minimum, f is approx. quadratic and so after a restart we will have (approximately) the linear CG method.
- which requires $p_0 =$ steepest descent.

- For large n restarts may never occur, since an approximate solution may be found in less than n steps.
- nonlinear CG method are sometimes implemented without restarts,
- or else they include strategies for restarting that are based on considerations other than iteration counts.
- The most popular restart strategy makes use of the observation $r_k^T r_i = 0$, which is that the gradients are mutually orthogonal when f is a quadratic function.
- A restart is performed whenever two consecutive gradients are far from orthogonal, as measured by the test

$$\frac{|\nabla f_k^T \nabla f_{k-1}|}{||\nabla f_k||^2} \geq \mu$$

where a typical value for the parameter μ is 0.1.

Global convergence

- With restarts and the strong Wolfe conditions, the algorithms (FR, PR) have global convergence since they include as a subsequence the steepest descent method (which is globally convergent with the Wolfe conditions).
- Without restarts:
  - FR has global convergence with the strong Wolfe conditions.
  - PR does not have global convergence, even though in practice it is better.
- In general, the theory on the rate of convergence of CG is complex and assumes exact l.s.