Conjugate Gradient:

An Iterative Descent Method
The Plan

• Review Iterative Descent

• Conjugate Gradient
Iterative Descent is an unconstrained optimization process

\[ x^{(k+1)} = x^{(k)} + \alpha \Delta x \]

- \( \Delta x \) is the descent direction
- \( \alpha > 0 \) is the step size along the descent direction
- Initial point \( x^0 \) needs to be chosen
Review : Descent Direction

- Gradient Descent: 1st-order approximation
  \[ \Delta x = -\nabla f(x) \]

- Newton’s Method: 2nd-order approximation
  \[ \Delta x = -\nabla^2 f(x)^{-1} \nabla f(x) \]
Review : Calculating $\alpha$

- Line search methods for calculating $\alpha$
  - Exact
  - Backtracking
  - Others
Review : Convergence

• Gradient Descent: Linear rate

• Newton’s Method: Quadratic near $x^*$
  – if certain conditions are met
  – otherwise, linear rate
Conjugate Gradient (CG)

• Is an alternative Iterative Descent algorithm
CG: Focus

• To more easily explain CG, I will focus on minimizing a quadratic system

\[ f(x) = \frac{1}{2}x^TAx - b^Tx + c \]
CG: Assumptions

• A is
  – Square
  – Symmetric
  – Positive Definite

• Theorem:
  - \( x^* \) minimizes \( f(x) = \frac{1}{2}x^TAx - b^Tx + c \)
    iff
  - \( x^* \) solves \( Ax = b \)

• Will focus on solving \( Ax = b \)
CG: Philosophy

- Avoid taking steps in the same direction as Gradient Descent can.
- Coordinate axes as search direction
  - only works if already know answer!
- CG, instead of orthogonality, uses A-orthogonality.
CG: A-orthogonality

• For descent directions, the CG method uses an A-orthogonal set of nonzero search direction vectors \( \{v^{(1)}, \ldots, v^{(n)}\} \)

\[
<v^{(i)}, Av^{(j)}> = 0 \text{ if } i \neq j
\]

• Each search direction will be evaluated once and a step size of just the right length will be applied in order to line up with \(x^*\)
CG: A-orthogonality

• Note:
  – With respect to a symmetric, positive definite $A$, A-orthogonal vectors are linear independent
  – So, they can be used as basis vectors for components of the algorithm, such as errors and residuals
• With regard to an A-orthogonal basis vector, eliminate the component of the error \((e^{(k)} = x^{(k)} - x^*)\) associated with that basis vector.

• The new error, \(e^{(k+1)}\), must be A-orthogonal to all A-orthogonal basis vectors considered so far.
CG: Algorithm

1. Choose initial point $x^{(0)}$
2. Generate first search direction $v^{(1)}$
3. For $k = 1, 2, \ldots, n$
   1. Generate (not search for) step size $t_k$
   2. $x^{(k)} = x^{(k-1)} + t_k \ v^{(k)}$ ($t_k$ is the step size)
   3. Generate next $A$-orthogonal search direction $v^{(k+1)}$
4. $x^{(n)}$ is the solution to $Ax = b$
CG: Algorithm

$r^{(0)} = b - Ax^{(0)}; \quad v^{(1)} = r^{(0)}$

for $k = 1, 2, \cdots, n$

\% generate new $x$

$$t_k = \frac{<r^{(k-1)}, r^{(k-1)}>}{<v^{(k)}, Av^{(k)}>}$$

$x^{(k)} = x^{(k-1)} + t_k v^{(k)}$

\% generate new search direction $v$

$$r^{(k)} = r^{(k-1)} - t_k Av^{(k)}$$

$$s_k = \frac{<r^{(k)}, r^{(k)}>}{<r^{(k-1)}, r^{(k-1)}>},$$

$$v^{(k+1)} = r^{(k)} + s_k v^{(k)}$$
CG: Notes

• Generating A-orthogonal vectors as we iterate, and don’t have to keep old ones around. Important for storage on large problems.

• Convergence is quicker than $O(n)$ if there are duplicated eigenvalues.
CG: Notes

• For a well-conditioned matrix $A$, a good approximation can sometimes be reached in fewer steps (e.g. $\sqrt{n}$ )

• Good approximation method for solving large sparse systems of $Ax = b$ with nonzero entries occurring in predictable patterns
CG: Reality Intrudes

• Accumulated roundoff error causes the residual, $r^{(i)} = b - Ax^{(i)}$, to gradually lose accuracy.

• Cancellation error causes the search vectors to lose $A$-orthogonality.

• If $A$ is ill-conditioned, CG is highly subject to rounding errors.
  – can precondition the matrix to ameliorate this
CG: Nonlinear \( f(x) \)

- CG can be used if the gradient \( \nabla f(x) \) can be computed.

- Use of CG in this case is guided by the idea that near the solution point every problem is approximately quadratic. So, instead of approximating a solution to the actual problem, instead approximate a solution to an approximating problem.
CG: Nonlinear $f(x)$

- Convergence behavior is similar to that for the pure quadratic situation.

- Changes to CG algorithm
  - recursive formula for residual $r$ can’t be used
  - more complicated to compute the step size $t_k$
  - multiple ways to compute $A$-orthogonal descent vectors

- Not guaranteed to converge to global minimum if $f$ has many local minima
CG: Nonlinear f(x)

- Doesn’t have same convergence guarantees as linear CG
  - Because CG can only generate \( n \) A-orthogonal vectors in \( n \)-space, it makes sense to restart CG every \( n \) iterations

- Preconditioning can be used to speed up convergence
CG: Roundup

• CG typically performs better than gradient descent, but not as well as Newton’s method

• CG avoids Newton’s information requirements associated with the evaluation, storage, and inversion of the Hessian (or at least a solution of a corresponding system of equations)
References

• Boyd & Vandenberghe, *Convex Optimization*
• Chong & Zak, *An Introduction to Optimization*, 1st edition
• Shewchuck, “An Introduction to the Conjugate Gradient Method Without the Agonizing Pain,” 1 ¼ edition, online
• Luenberger, *Linear and Nonlinear Programming*, 3rd edition