Computational Linear Algebra: Ax=b
Part V: Conjugate Gradient

David Shuman

March 1, 2022

Many figures in these slides are from "An Introduction to the Conjugate Gradient Method Without the Agonizing Pain, Edition $1\frac{1}{4}$" by Jonathan Richard Shewchuk of Carnegie Mellon University. I've posted a link to that technical report on Moodle.

The Conjugate Gradient (CG) Method

- Conjugate gradient iteration (CG) is a method of solving $Ax = b$ for matrices A that are symmetric positive definite
- Solves $Ax = b$ amazingly quickly (especially if the eigenvalues of A are clustered)
- Discovered by Magnus Hestenes and Eduard Steifel in 1952
- See the original paper and the anniversary celebration
- Now a mainstay in scientific computation (the "algorithm of choice" for large positive definite matrices)
- It is very very cool!

Quadratic Forms

Definition

For an $n \times n$ matrix A and a vector $\mathrm{b} \in \mathbb{R}^n$, the quadratic form associated with $A\mathrm{x} = \mathrm{b}$ is the function

$$f(\mathrm{x}) = \frac{1}{2}\mathrm{x}^T A\mathrm{x} - \mathrm{x}^T \mathrm{b}$$

where $x = (x_1, x_2, \ldots, x_n)^T$ is a vector of variables.

- $f(\mathrm{x})$ is a quadratic function in the variables $\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_n$

Quadratic Forms

Definition

For an $n \times n$ matrix A and a vector $\mathrm{b} \in \mathbb{R}^n$, the quadratic form associated with $A\mathrm{x} = \mathrm{b}$ is the function

$$f(\mathrm{x}) = \frac{1}{2}\mathrm{x}^T A\mathrm{x} - \mathrm{x}^T \mathrm{b}$$

where $x = (x_1, x_2, \ldots, x_n)^T$ is a vector of variables.

- $f(\mathrm{x})$ is a quadratic function in the variables $\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_n$
- Compute it for the following matrices
  - $A = (3)$ and $\mathrm{b} = (12)$
  - $A = \begin{pmatrix} 3 & 2 \\ 2 & 6 \end{pmatrix}$ and $\mathrm{b} = \begin{pmatrix} 2 \\ -8 \end{pmatrix}$

Solving $Ax = b$ as a minimization problem

Example

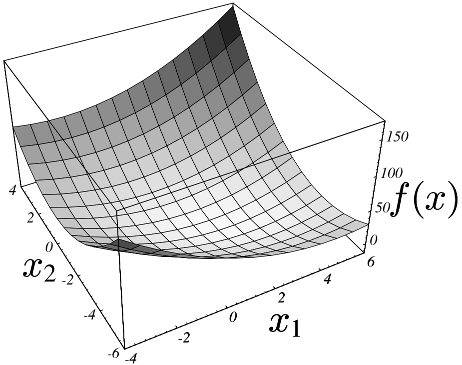
- Let $A = \begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix}$ (pos. def.) and $b = \begin{bmatrix} 2 \\ -8 \end{bmatrix}$ and solve $Ax = b$
- Compute the quadratic form: $f(x) = \frac{1}{2}x^T Ax - x^T b$

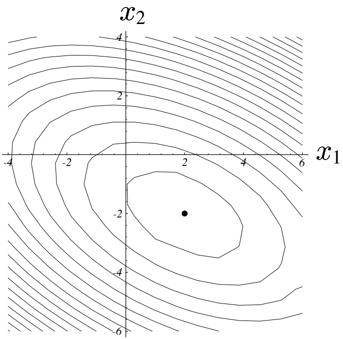
$$
\begin{aligned}
f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) &= \frac{1}{2}\begin{bmatrix} x_1 & x_2 \end{bmatrix}\begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} x_1 & x_2 \end{bmatrix}\begin{bmatrix} 2 \\ -8 \end{bmatrix} \\
&= \frac{1}{2}(3x_1^2 + 4x_1x_2 + 6x_2^2) - (2x_1 - 8x_2) \\
&= \frac{3}{2}x_1^2 - 2x_1 + 2x_1x_2 + 8x_2 + 3x_2^2
\end{aligned}
$$

Solving $Ax = \mathrm{b}$ as a minimization problem

Example

- Let $A = \begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix}$ and $b = \begin{bmatrix} 2 \\ -8 \end{bmatrix}$ and solve $Ax = \mathrm{b}$.
- $f(\mathrm{x}) = \frac{1}{2}\mathrm{x}^T A\mathrm{x} - \mathrm{x}^T \mathrm{b} = \frac{3}{2}x_1^2 - 2x_1 + 2x_1x_2 + 8x_2 + 3x_2^2$
- A plot of the quadratic form $f(\mathrm{x})$ and its contour plot:





The Gradient of the Quadratic Form

- The gradient of a function $f : \mathbb{R}^n \to \mathbb{R}$ is $f'(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(x) \\ \frac{\partial f}{\partial x_2}(x) \\ \vdots \\ \frac{\partial f}{\partial x_n}(x) \end{bmatrix}$

The Gradient of the Quadratic Form

- The gradient of a function $f : \mathbb{R}^n \to \mathbb{R}$ is $f'(\mathrm{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(\mathrm{x}) \\ \frac{\partial f}{\partial x_2}(\mathrm{x}) \\ \vdots \\ \frac{\partial f}{\partial x_n}(\mathrm{x}) \end{bmatrix}$
- For the quadratic form we get $f'(\mathrm{x}) = \frac{1}{2}A^T\mathrm{x} + \frac{1}{2}A\mathrm{x} - \mathrm{b}$

The Gradient of the Quadratic Form

- The gradient of a function $f : \mathbb{R}^n \to \mathbb{R}$ is $f'(\mathrm{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(\mathrm{x}) \\ \frac{\partial f}{\partial x_2}(\mathrm{x}) \\ \vdots \\ \frac{\partial f}{\partial x_n}(\mathrm{x}) \end{bmatrix}$
- For the quadratic form we get $f'(\mathrm{x}) = \frac{1}{2}A^T\mathrm{x} + \frac{1}{2}A\mathrm{x} - \mathrm{b}$
- When A is symmetric, this reduces to $f'(\mathrm{x}) = A\mathrm{x} - \mathrm{b}$

The Gradient of the Quadratic Form

- The gradient of a function $f : \mathbb{R}^n \to \mathbb{R}$ is $f'(\mathrm{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(\mathrm{x}) \\ \frac{\partial f}{\partial x_2}(\mathrm{x}) \\ \vdots \\ \frac{\partial f}{\partial x_n}(\mathrm{x}) \end{bmatrix}$
- For the quadratic form we get $f'(\mathrm{x}) = \frac{1}{2}A^T\mathrm{x} + \frac{1}{2}A\mathrm{x} - \mathrm{b}$
- When A is symmetric, this reduces to $f'(\mathrm{x}) = A\mathrm{x} - \mathrm{b}$
- Thus, the solution to $A\mathrm{x} = \mathrm{b}$ is a critical point of $f(\mathrm{x})$

The Gradient of the Quadratic Form

- The gradient of a function $f : \mathbb{R}^n \to \mathbb{R}$ is $f'(\mathrm{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(\mathrm{x}) \\ \frac{\partial f}{\partial x_2}(\mathrm{x}) \\ \vdots \\ \frac{\partial f}{\partial x_n}(\mathrm{x}) \end{bmatrix}$
- For the quadratic form we get $f'(\mathrm{x}) = \frac{1}{2}A^T\mathrm{x} + \frac{1}{2}A\mathrm{x} - \mathrm{b}$
- When A is symmetric, this reduces to $f'(\mathrm{x}) = A\mathrm{x} - \mathrm{b}$
- Thus, the solution to $A\mathrm{x} = \mathrm{b}$ is a critical point of $f(\mathrm{x})$
- When A is symmetric positive definite, it is a minimum

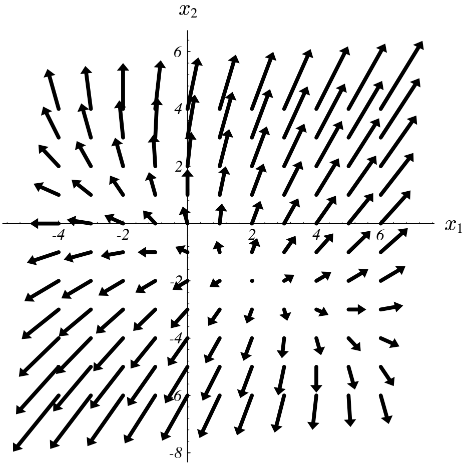
The Gradient of the Quadratic Form

- The gradient of a function $f : \mathbb{R}^n \to \mathbb{R}$ is $f'(\mathrm{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(\mathrm{x}) \\ \frac{\partial f}{\partial x_2}(\mathrm{x}) \\ \vdots \\ \frac{\partial f}{\partial x_n}(\mathrm{x}) \end{bmatrix}$
- For the quadratic form we get $f'(\mathrm{x}) = \frac{1}{2}A^T\mathrm{x} + \frac{1}{2}A\mathrm{x} - \mathrm{b}$
- When A is symmetric, this reduces to $f'(\mathrm{x}) = A\mathrm{x} - \mathrm{b}$
- Thus, the solution to $A\mathrm{x} = \mathrm{b}$ is a critical point of $f(\mathrm{x})$
- When A is symmetric positive definite, it is a minimum
- Can solve $A\mathrm{x} = \mathrm{b}$ by minimizing $f(\mathrm{x})$

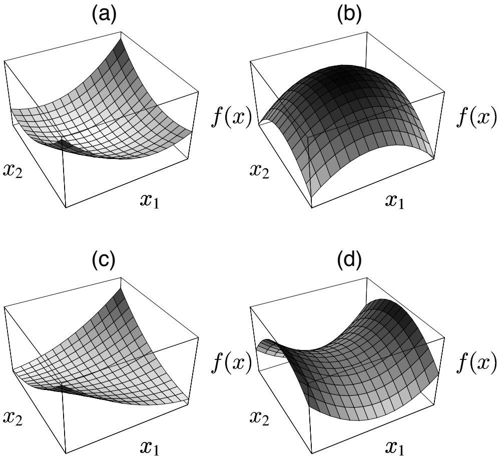
The Gradient of the Quadratic Form

Example

For our example, here's the gradient vector field:



The Role of the Positive Definiteness



(a) Positive definite. (b) Negative definite. (c) Singular. (d) Indefinite.

Gradient Descent

- The exact solution x_* to $A\mathrm{x} = \mathrm{b}$ is the minimum of $f(\mathrm{x})$

Gradient Descent

- The exact solution x_* to $A\mathrm{x} = \mathrm{b}$ is the minimum of $f(\mathrm{x})$
- For a given guess x_k the residual $\mathrm{r}_k = \mathrm{b} - A\mathrm{x}_k$ equals the negative gradient $-f'(\mathrm{x})$
- That is, the residual points downhill towards the minimum and thus towards the solution

Gradient Descent

- The exact solution x_* to $Ax = b$ is the minimum of $f(x)$
- For a given guess x_k the residual $r_k = b - Ax_k$ equals the negative gradient $-f'(x)$
- That is, the residual points downhill towards the minimum and thus towards the solution
- Gradient descent method:
  - Guess an approximate solution x_k
  - Compute the residual

    $$r_k = b - Ax_k,$$

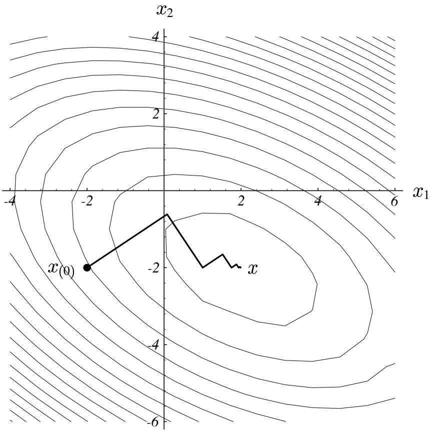
    which points towards the exact solution
  - Take a small step in the direction of the residual

    $$x_{k+1} = x_k + \alpha_k r_k,$$

    where $\alpha_k = \frac{r_k^T r_k}{r_k^T A r_k}$
  - Repeat

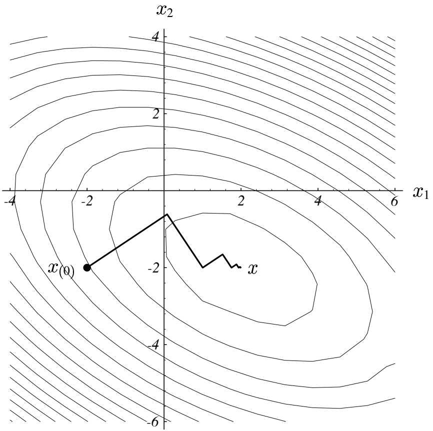
Gradient Descent

For our example, gradient descent converges to the solution $x_* = \begin{bmatrix} 2 \\ -2 \end{bmatrix}$:



Gradient Descent

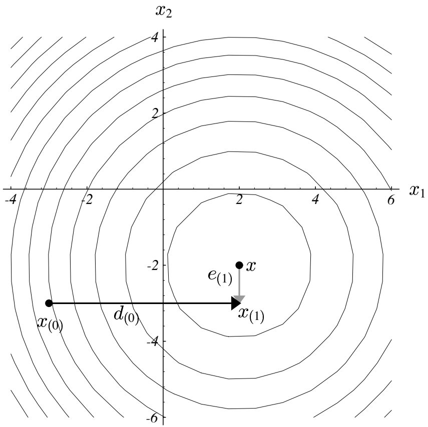
For our example, gradient descent converges to the solution $x_* = \begin{bmatrix} 2 \\ -2 \end{bmatrix}$:



- One issue: gradient descent often takes steps in the same directions as earlier steps
- Would be nicer to get the length of the step right the first time

Method of Orthogonal Directions

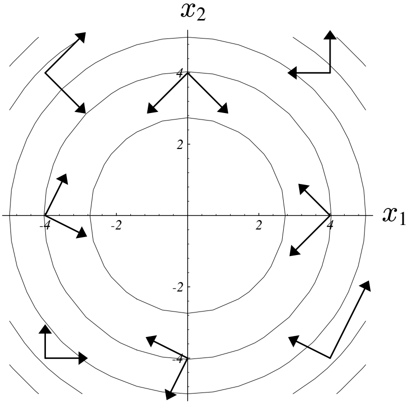
Idea: get one component right each step

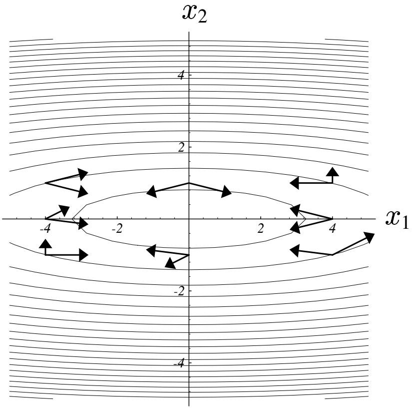


Would require an oracle to know the solution in order to guarantee that the new error is orthogonal to the current descent direction

A-Orthogonality

- Idea: choose the search directions to be A-orthogonal instead of orthogonal
- Two vectors y and z called *A-orthogonal* (also called *A-conjugate*) if $y^\top A z = 0$





Conjugate Gradient: Details

The algorithm

x_0 = initial guess

$d_0 = b - Ax_0$ (initial direction)

$r_0 = d_0$ (initial residual)

for $k = 0, 1, 2, 3, \ldots, n-1$

if ($r_k = 0$) **stop**

$\alpha_k = (r_k^T r_k)/(d_k^T A d_k)$ new step length

$x_{k+1} = x_k + \alpha_k d_k$ take step

$r_{k+1} = r_k - \alpha_k A d_k$ new residual

$\beta_k = (r_{k+1}^T r_{k+1})/(r_k^T r_k)$

$d_{k+1} = r_{k+1} + \beta_k d_k$ new search direction

Conjugate Gradient: Details

The algorithm

x_0 = initial guess
$d_0 = b - Ax_0$ (initial direction)
$r_0 = d_0$ (initial residual)

for $k = 0, 1, 2, 3, \ldots, n-1$

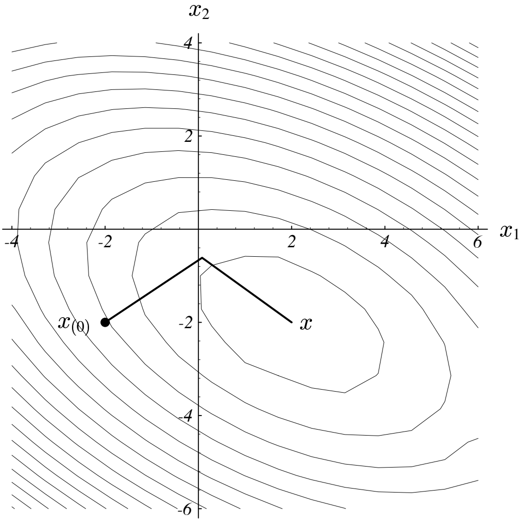
- if ($r_k = 0$) **stop**
- $\alpha_k = (r_k^T r_k)/(d_k^T A d_k)$ new step length
- $x_{k+1} = x_k + \alpha_k d_k$ take step
- $r_{k+1} = r_k - \alpha_k A d_k$ new residual
- $\beta_k = (r_{k+1}^T r_{k+1})/(r_k^T r_k)$
- $d_{k+1} = r_{k+1} + \beta_k d_k$ new search direction

Complexity Analysis

- Where is the bottleneck?
- What is the complexity?

Conjugate Gradient Example

Converges in two steps:



The Conjugate Gradient (CG) Method Summary

- A descent method that very cleverly optimizes the direction of descent and how far to step at each point
- It does so by making consecutive steps that are A-orthogonal to one another: $\langle \mathrm{d}_k, \mathrm{d}_{k+1} \rangle_A = \mathrm{d}_k^T A \mathrm{d}_{k+1} = 0$

The Conjugate Gradient (CG) Method Summary

- A descent method that very cleverly optimizes the direction of descent and how far to step at each point
- It does so by making consecutive steps that are A-orthogonal to one another: $\langle \mathrm{d}_k, \mathrm{d}_{k+1} \rangle_A = \mathrm{d}_k^T A \mathrm{d}_{k+1} = 0$
- In theory, it reaches the solution to $A\mathrm{x} = \mathrm{b}$ in at most n steps (i.e., it's a direct method)
  - The directions d_k always move closer to the solution
  - They are A-orthogonal and linearly independent
  - We are in $\mathbb{R}^n$ so the $(n+1)$st one must be 0

The Conjugate Gradient (CG) Method Summary

- A descent method that very cleverly optimizes the direction of descent and how far to step at each point
- It does so by making consecutive steps that are A-orthogonal to one another: $\langle \mathrm{d}_k, \mathrm{d}_{k+1} \rangle_A = \mathrm{d}_k^T A \mathrm{d}_{k+1} = 0$
- In theory, it reaches the solution to $A\mathrm{x} = \mathrm{b}$ in at most n steps (i.e., it's a direct method)
  - The directions d_k always move closer to the solution
  - They are A-orthogonal and linearly independent
  - We are in $\mathbb{R}^n$ so the $(n+1)$st one must be 0
- In practice, due to rounding errors, directions are not actually A-orthogonal, and convergence can take far more than n steps or not happen at all
  - Discarded as a direct method in the 1960s
  - Renewed interest later in the 1970s as an iterative method with good approximations in much fewer than n steps
  - Computationally efficient if A is sparse ($\mathcal{O}(m)$)
  - Used with preconditioners (see accompanying activity)

More Details

A-Orthogonality and Choosing the Step Size

- How to pick the step size on the previous figure? We want

$$0 = \mathrm{d}_k^T \mathrm{e}_{k+1} = \mathrm{d}_k^T(\mathrm{e}_k + \alpha_k \mathrm{d}_k) \Rightarrow \alpha_k = -\frac{\mathrm{d}_k^T \mathrm{e}_k}{\mathrm{d}_k^T \mathrm{d}_k} = -\frac{\mathrm{d}_k^T(\mathrm{x}_* - \mathrm{x}_k)}{\mathrm{d}_k^T \mathrm{d}_k}$$

A-Orthogonality and Choosing the Step Size

- How to pick the step size on the previous figure? We want

$$0 = d_k^T e_{k+1} = d_k^T (e_k + \alpha_k d_k) \Rightarrow \alpha_k = -\frac{d_k^T e_k}{d_k^T d_k} = -\frac{d_k^T (x_* - x_k)}{d_k^T d_k}$$

- But we don't know x_*!

A-Orthogonality and Choosing the Step Size

- How to pick the step size on the previous figure? We want

$$0 = \mathrm{d}_k^T \mathrm{e}_{k+1} = \mathrm{d}_k^T(\mathrm{e}_k + \alpha_k \mathrm{d}_k) \Rightarrow \alpha_k = -\frac{\mathrm{d}_k^T \mathrm{e}_k}{\mathrm{d}_k^T \mathrm{d}_k} = -\frac{\mathrm{d}_k^T(\mathrm{x}_* - \mathrm{x}_k)}{\mathrm{d}_k^T \mathrm{d}_k}$$

- But we don't know x_*!
- Idea: choose the search directions to be *A*-orthogonal instead of orthogonal:

$$\mathrm{d}_i^T A d_k = 0$$

- Then to set the step size, we want:

$$0 = \mathrm{d}_k^T A \mathrm{e}_{k+1} = \mathrm{d}_k^T A(\mathrm{e}_k + \alpha_k \mathrm{d}_k)$$

$$\Rightarrow \alpha_k = -\frac{\mathrm{d}_k^T A \mathrm{e}_k}{\mathrm{d}_k^T A \mathrm{d}_k} = -\frac{\mathrm{d}_k^T(A\mathrm{x}_* - A\mathrm{x}_k)}{\mathrm{d}_k^T A \mathrm{d}_k} = -\frac{\mathrm{d}_k^T(\mathrm{b} - A\mathrm{x}_k)}{\mathrm{d}_k^T A \mathrm{d}_k} = -\frac{\mathrm{d}_k^T \mathrm{r}_k}{\mathrm{d}_k^T A \mathrm{d}_k}$$

- We can compute r_k, and therefore the step size!

A-Orthogonality and Choosing the Step Size

- How to pick the step size on the previous figure? We want

$$0 = d_k^T e_{k+1} = d_k^T (e_k + \alpha_k d_k) \Rightarrow \alpha_k = -\frac{d_k^T e_k}{d_k^T d_k} = -\frac{d_k^T (x_* - x_k)}{d_k^T d_k}$$

- But we don't know x_*!
- Idea: choose the search directions to be A-orthogonal instead of orthogonal:

$$d_i^T A d_k = 0$$

- Then to set the step size, we want:

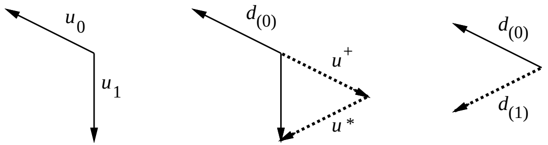
$$0 = d_k^T A e_{k+1} = d_k^T A (e_k + \alpha_k d_k)$$

$$\Rightarrow \alpha_k = -\frac{d_k^T A e_k}{d_k^T A d_k} = -\frac{d_k^T (A x_* - A x_k)}{d_k^T A d_k} = -\frac{d_k^T (b - A x_k)}{d_k^T A d_k} = -\frac{d_k^T r_k}{d_k^T A d_k}$$

- We can compute r_k, and therefore the step size!
- How do we choose the directions?

Choosing the Descent Direction

Use a conjugate Gram-Schmidt process:



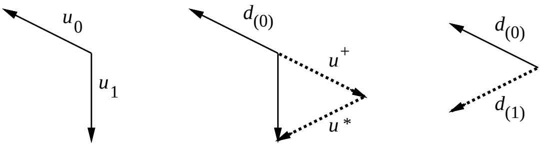
- Suppose you have n linearly independent (but not orthogonal) vectors $\mathsf{u}_0, \mathsf{u}_1, \ldots, \mathsf{u}_{n-1}$
- To construct d_k, take u_k and subtract out any components that are not A-orthogonal to the previous $k-1$ direction vectors:

$$\mathsf{d}_k = \mathsf{u}_k + \sum_{i=0}^{k-1} \beta_{ki} \mathsf{d}_i$$

where β's are chosen to satisfy $0 = \mathsf{d}_k^T A \mathsf{d}_j, \ \forall j < k$

Choosing the Descent Direction

Use a conjugate Gram-Schmidt process:



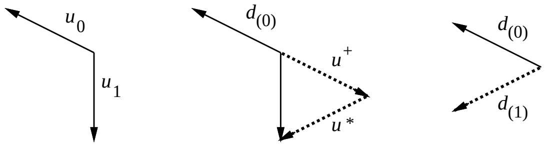
- Suppose you have n linearly independent (but not orthogonal) vectors $\mathsf{u}_0, \mathsf{u}_1, \ldots, \mathsf{u}_{n-1}$
- To construct d_k, take u_k and subtract out any components that are not A-orthogonal to the previous $k-1$ direction vectors:

$$\mathsf{d}_k = \mathsf{u}_k + \sum_{i=0}^{k-1} \beta_{ki} \mathsf{d}_i$$

  where β's are chosen to satisfy $0 = \mathsf{d}_k^T A \mathsf{d}_j, \ \forall j < k$
- Issue: with an arbitrary choice of u's, this takes $O(n^3)$

Choosing the Descent Direction (cont.)

Use a conjugate Gram-Schmidt process:



- Amazing solution: take $\mathrm{u}_{k+1} = \mathrm{r}_{k+1}$, which is already orthogonal to the previous residuals and all of the previous search directions except d_k. Thus, $\mathrm{d}_{k+1} = \mathrm{r}_{k+1} + \beta_k \mathrm{d}_k$, and we want

$$0 = \mathrm{d}_{k+1}^T A \mathrm{d}_k = \mathrm{r}_{k+1}^T A \mathrm{d}_k + \beta_k \mathrm{d}_k^T A \mathrm{d}_k$$

$$\Rightarrow \beta_k = -\frac{\mathrm{r}_{k+1}^T A \mathrm{d}_k}{\mathrm{d}_k^T A \mathrm{d}_k} = \frac{\mathrm{r}_{k+1}^T \mathrm{r}_{k+1}}{\mathrm{r}_k^T \mathrm{r}_k} \text{ (see Eq. 2.47, p. 125 of book)}$$

  - Complexity reduced (next slide)
  - Don't need to store old search vectors (hooray for memory!)