Iterative Solvers
Iterative vs. Direct Solvers

- **Direct Solver/Method** – closed form strategy, e.g. quadratic/Cardano formula, Gaussian Elimination for LU factorization, Cholesky factorization, etc.
- **Iterative Solver/Method**
  - start with an initial guess $c^1$
  - use a recursive approach to improve that guess: $c^2$, $c^3$, $c^4$, ...
  - terminate based on a stopping criterion, e.g. when error is small $\|c^q - c^{exact}\| \leq \epsilon$

- A direct method can be used to obtain an initial guess
- Iterative methods are great for sparse matrices, as they often can ignore 0 entries
  - E.g. by formulating the method via the matrix’s action (multiplication) on a vector
- Direct solvers are more commonly used on dense matrices
- Iterative solvers are used for training Neural Networks!
Issues with Direct Methods

• (Recall) Quadratic formula loses precision, and can fail, when $-b \pm \sqrt{b^2 - 4ac}$ has catastrophic cancellation
  • The de-rationalized quadratic formula instead uses $-b \mp \sqrt{b^2 - 4ac}$
  • Using one formula for each root avoids catastrophic cancellation

• Cardano’s formula for the roots of a cubic equation suffers from similar issues, but there is no straightforward fix
  • The computed roots too often have unacceptably high error
  • To highlight why one might need accurate cubic roots, consider collision detection...
In order to detect interactions between objects in video games, objects were assigned a hit box. Anything inside an object’s hit box can potentially interact with (i.e. hit) it.
Better Hit Boxes

- These evolved over time to more complicated shapes in both 2D and 3D
  - e.g. spheres, ellipsoids, capsules, etc.
- Anything inside any of an object’s hit boxes can potentially interact with it
Accurate Collision Detection

• More complex objects are often modeled by a triangulated surface mesh
• The interior can be filled with tetrahedra, or approximated with other objects
• Anything inside any of an object’s interior structures can potentially interact with it
Objects Without Interiors

• Very thin objects, such as cloth/shells, do not have an interior region
• One cannot use the same concept of inside to detect potential interactions
Continuous Collision Detection (CCD)

• Model the time varying trajectories of surface triangle vertices to see if/when they collide with each other
• Doesn’t depend on the existence of an interior region
• There are two cases to consider: (1) Point-Face, (2) Edge-Edge
Continuous Collision Detection (CCD)

• In both cases, the 4 relevant points need to become **coplanar** in order to (potentially) collide

• Once deemed coplanar, a second check determines whether: the lone point is inside the triangle (for Point-Face) or the two edges intersect (for Edge-Edge)
Continuous Collision Detection (CCD)

• Consider time $t_o$ to time $t_f$ and assume that the points have constant velocities during that time interval: $V_i(t_o)$ for $i = 1, 2, 3, 4$

• The time evolving positions are: $X_i(t) = X_i(t_o) + V_i(t_o)(t - t_o)$ for $t \in [t_o, t_f]$

• Although their paths are (generally) curved, considering piecewise linear increments is sufficient for preventing self-intersecting states
Continuous Collision Detection (CCD)

- Coplanarity occurs when $X_4(t) - X_1(t)$, $X_3(t) - X_1(t)$, and $X_2(t) - X_1(t)$ are not a basis for $R^3$, which can be checked by making them the columns of a 3x3 matrix and setting the determinant to zero (obtaining a cubic equation in $t$)

- Find the first root of this cubic equation in the interval $[t_o, t_f]$

- Cubic equation solvers are so error prone that collisions are (very) often missed, and the cloth/shell ends up in a spurious self-intersecting state

- A very carefully devised/implemented iterative solver for cubic equations was able to detect all collisions:
  - It requires double precision (and fails too often in single precision)
  - See Bridson et al. “Robust Treatment of Collisions, Contact, and Friction for Cloth Animation” (2002)
Residual and Solution Error

• When solving $A c = b$, a current guess $c^q$ has residual $r^q = b - A c^q$

• The residual measures the errors in the equations, not the error in the solution

• The error in the solution $e^q = c^q - c^{\text{exact}}$ relates to the residual via:

$$r^q = b - A c^q = A c^{\text{exact}} - A c^q = A(c^{\text{exact}} - c^q) = -A e^q$$

• That is, the residual is the solution error transformed into the space that $b$ lives in (the range of $A$)
1D example

- Consider a simple size $1 \times 1$ matrix, i.e. $[a]c = b$ with exact solution $c = \frac{b}{a}$
- Since $r^q = -ae^q$, smaller $a$ values lead to deceivingly small residuals even when the error is large
Diagonalizing the Residual/Error Equation

• “All matrices are diagonal matrices”
• And, diagonal matrices represent decoupled 1D scalar problems

• Using the SVD, \( r^q = -A e^q \) becomes \( (U^T r^q) = -\Sigma (V^T e^q) \) which is a decoupled set of diagonal equations

• Each decoupled equation has the form \( \hat{r}^q_k = -\sigma_k \hat{e}^q_k \) (seen on the previous slide)

• Small \( \sigma_k \) lead to deceivingly small residuals even when the error is large

• A small residual indicates a small error for larger singular values, but not for smaller singular values
Line Search

• Choose a search direction \( s^q \) and move some distance \( \alpha^q \) in that direction to update the current guess to the next guess: \( c^{q+1} = c^q + \alpha^q s^q \)
  • There are various strategies for choosing \( \alpha^q \), including the notion of safe sets that clamp its maximum magnitude
  • Subtract \( c^{exact} \) from both sides of this recursion to get \( e^{q+1} = e^q + \alpha^q s^q \)
  • Multiply through by \(-A\) to get \( r^{q+1} = r^q - \alpha^q As^q \)

• Optimally, one would follow \( s^q \) until all the error in that direction was eliminated
  • That is, until the remaining error is orthogonal to \( s^q \), i.e. \( e^{q+1} \cdot s^q = 0 \)
  • However, the error is unknown (otherwise, the solution would be known)

• Instead, follow \( s^q \) until the residual is orthogonal to \( s^q \), i.e. \( r^{q+1} \cdot s^q = 0 \)
  • Plugging in the recursion for \( r^{q+1} \) gives \( \alpha^q = \frac{s^q \cdot r^q}{s^q \cdot As^q} \)
Steepest Descent

• Steepest Descent chooses the steepest downhill direction as the search direction
  • That turns out to be the residual, i.e. choose $s^q = r^q$

• Iterate: $r^q = b - Ac^q$, $\alpha^q = \frac{r^q \cdot r^q}{r^q \cdot Ar^q}$, $c^{q+1} = c^q + \alpha^q r^q$, until $r^q$ is considered small enough

• Note: can replace $r^q = b - Ac^q$ with $r^q = r^{q-1} - \alpha^{q-1} Ar^{q-1}$
  • Since $Ar^{q-1}$ had already been computed to find $\alpha^{q-1}$, this eliminates one of the (possibly expensive) multiplications by $A$

• Drawback: Steepest Descent repeatedly searches in overlapping (non-orthogonal) directions, especially for higher condition number matrices (more on this later)
Conjugate Gradients (CG)

• A very efficient and robust method for SPD systems
• Converges (theoretically) in at most $n$-steps for an $nxn$ matrix
  • Theoretically, only need one step for each distinct eigenvalue
  • Almost converged when taking one step for each eigenvalue cluster
  • Thus, preconditioning makes a big difference (assuming it clusters eigenvalues)
• Motivation: choosing orthogonal search directions precludes repeatedly searching in overlapping directions (in contrast to Steepest Descent)
  • But, it is difficult to implement this orthogonality
• Instead: choose A-orthogonal search directions
  • Instead of $< s^q, s^\hat{q} > = 0$, choose $< s^q, s^\hat{q} >_A = 0$ for $q \neq \hat{q}$
Error Analysis for CG

• In the A-orthogonal basis of search directions, the initial error is $e^1 = \sum_{\hat{q}=1}^{n} \beta^{\hat{q}} s^{\hat{q}}$; so, $\langle s^{q}, e^1 \rangle_A = \beta^{q} \langle s^{q}, s^{q} \rangle_A$

• Error recursion gives $e^q = e^1 + \sum_{\hat{q}=1}^{q-1} \alpha^{\hat{q}} s^{\hat{q}}$; so, $\langle s^{q}, e^q \rangle_A = \langle s^{q}, e^1 \rangle_A$

• Progressing until $r^{q+1} \cdot s^{q} = 0$ gives $\alpha^q = \frac{s^{q} \cdot r^{q}}{s^{q} \cdot A s^{q}} = -\frac{\langle s^{q}, e^q \rangle_A}{\langle s^{q}, s^{q} \rangle_A} = -\beta^q$

• Thus, $e^1 = \sum_{\hat{q}=1}^{n} (-\alpha^{\hat{q}}) s^{\hat{q}}$ and $e^q = \sum_{\hat{q}=q}^{n} (-\alpha^{\hat{q}}) s^{\hat{q}}$

  • This proves that the error is indeed cancelled out in $n$ steps, i.e. $e^{q+1} = 0$

• Aside: If $\tilde{q} < q$, then $s^{\tilde{q}} \cdot r^{q} = -\langle s^{\tilde{q}}, e^q \rangle_A = 0$; so, the residual is orthogonal to all previous search directions (not just the previous one)
Gram-Schmidt

• Orthogonalizes a set of vectors
• For each new vector, subtract its (weighted) dot product overlap with all prior vectors, making it orthogonal to them
• A-orthogonal Gram-Schmidt simply uses an A-weighted dot/inner product
• Given vector $\bar{S}^q$, subtract out the A-overlap with $s^1$ to $s^{q-1}$ so that the resulting vector $s^q$ has $< s^q, s^\hat{q} >_A = 0$ for $\hat{q} \in \{1,2,\ldots,q-1\}$
• That is, $s^q = \bar{S}^q - \sum_{\hat{q}=1}^{q-1} \frac{<\bar{S}^q,s^\hat{q}>_A}{<s^\hat{q},s^\hat{q}>_A} s^\hat{q}$ where the two non-normalized $s^\hat{q}$ both require division by their norm (and $< s^\hat{q}, s^\hat{q} >_A = \|s^\hat{q}\|^2_A$)
• Proof: $< s^q, s^\hat{q} >_A = < \bar{S}^q, s^\hat{q} >_A - \frac{<\bar{S}^q,s^\hat{q}>_A}{<s^\hat{q},s^\hat{q}>_A} < s^\hat{q}, s^\hat{q} >_A = 0$
Gram-Schmidt for CG

- Choose candidate search directions $\tilde{S}^q = r^q$, and make $A$-orthogonal via Gram-Schmidt
  
  That is, $s^q = r^q - \sum_{\tilde{q}=1}^{q-1} \frac{<r^q, s^{\tilde{q}}>_A}{<s^{\tilde{q}}, s^{\tilde{q}}>_A} s^{\tilde{q}}$

- Dot product with $r^{\tilde{q}}$ to get: $s^q \cdot r^{\tilde{q}} = r^q \cdot r^{\tilde{q}} - \sum_{\tilde{q}=1}^{q-1} \frac{<r^q, s^{\tilde{q}}>_A}{<s^{\tilde{q}}, s^{\tilde{q}}>_A} s^{\tilde{q}} \cdot r^{\tilde{q}}$
  
  - If $\tilde{q} > q$, then $0 = r^q \cdot r^{\tilde{q}} + 0$ implies that all the residuals are orthogonal
  
  - If $\tilde{q} = q$, then $s^q \cdot r^q = r^q \cdot r^q + 0$ implies $\alpha^q = \frac{r^q \cdot r^q}{<s^q, s^q>_A}$

- Dot product $r^q = r^{q-1} - \alpha^{q-1} A s^{q-1}$ with $r^{\tilde{q}}$ to get
  
  - $r^q \cdot r^q = r^{q-1} \cdot r^{q-1} - \alpha^{q-1} < r^{\tilde{q}}, s^{q-1} >_A$
  
  - If $\tilde{q} > q$, then $0 = 0 - \alpha^{q-1} < r^{\tilde{q}}, s^{q-1} >_A$ implies that only the last term in the sum is nonzero
  
  - If $\tilde{q} = q$, then $r^q \cdot r^q = 0 - \alpha^{q-1} < r^q, s^{q-1} >_A$ for the last term in the sum

- Finally, $s^q = r^q + \frac{r^q \cdot r^q}{\alpha^{q-1} <s^{q-1}, s^{q-1}>_A} s^{q-1} = r^q + \frac{r^q \cdot r^q}{r^{q-1} \cdot r^{q-1}} s^{q-1}$
Conjugate Gradients Method

• Start with: $s^1 = r^1 = b - Ac^1$

• Iterate:
  - $\alpha^q = \frac{r^q . r^q}{<s^q , s^q>_A}$
  - $c^{q+1} = c^q + \alpha^q s^q$ and $r^{q+1} = r^q - \alpha^q As^q$ (both as usual for line search)
  - $s^{q+1} = r^{q+1} + \frac{r^{q+1} . r^{q+1}}{r^q . r^q} s^q$

• Note: Gram-Schmidt drifts, making search directions less A-orthogonal over time; thus, occasionally throw out all search directions and start over with $s^1 = r^1 = b - Ac^1$
Non-Symmetric and/or Indefinite

• GMRES, MINRES, BiCGSTAB, etc...

• Generally speaking, iterative methods for non-symmetric and/or indefinite matrices are less stable, more error prone, and slower than CG on an SPD matrix