We study some iterative methods in solving systems of linear equations. We begin with some direct methods such as the Jacobi iterations and Gauss-Siedel iterations, and see its convergence analysis by spectral arguments. Then we study the commonly used Krylov subspace methods, and derive the famous conjugate gradient method. We will study the elegant convergence analysis relating the spectrum and polynomial approximation in next week.

---

**Project Proposal**

Due date: April 3, in class

A 1-2 page project proposal, including proposed title, a brief outline of the content, some key references.

The basic requirement of the project is to write a survey of a topic of your choice that is related to this course (and ideally to your research). Original research is not a requirement, but is of course welcome. Originality, however, is highly encouraged (e.g. make some connections between different research topics, identify a potential new application of spectral techniques, simplify some existing proofs, present several results in a unified way, etc.). I also expect the project report is more mathematically oriented.

The project proposal is counted 20% of the course project. I will give you some suggestions after reading your project proposal. You can also ask me for suggestions or comments or any other questions before submitting the project proposal.

---

**Solving Linear Equations**

Our interest in the coming lectures is in solving systems of linear equations. It is a basic primitive for solving other problems. Of course, we can use Gaussian elimination to solve any non-linear system in $O(n^3)$ arithmetic operations. We are interested in algorithms that perform considerably better in special but useful settings. These algorithms are usually based on iteratively compute better and better approximate solutions. They are very effective in practice and are very elegant in theory. In the coming two weeks we will present some general iterative methods and analyze their performance. In the last three
weeks we will present some general iterative methods and analyse their performance. In the last three weeks, we will study the special case of solving Laplacian systems. We will study the recent developments of an almost linear time solver for Laplacian systems, show the beautiful connections between solving linear equations and (spectral) graph theory, and see some surprising applications of these results.

Iterative Methods 1]

To begin, let's consider some direct iterative methods and analyse their performance.

Consider the linear system:

\[ 9x_1 + x_2 + x_3 = b_1 \]
\[ 2x_1 + 10x_2 + 3x_3 = b_2 \]
\[ 3x_1 + 4x_2 + 11x_3 = b_3 \]

Let \( x^{(0)} = [x_1^{(0)}, x_2^{(0)}, x_3^{(0)}]^T \) be an initial guess of the true solution \( x \). Then define an iteration sequence:

\[ x_1^{(m)} = \frac{1}{9} (b_1 - x_2^{(m)} - x_3^{(m)}) \]
\[ x_2^{(m)} = \frac{1}{10} (b_2 - 2x_1^{(m)} - 3x_3^{(m)}) \]
\[ x_3^{(m)} = \frac{1}{11} (b_3 - 3x_1^{(m)} - 4x_2^{(m)}) \]

This is called the Jacobi iteration method or the method of simultaneous replacements.

Let \( e^{(k)} = ||x - x^{(k)}|| \) be the error in the \( k \)-th iteration.

You can check that the error decreases as the number of iteration increases.

There is a natural modification of the above method, using the most updated values:

\[ x_1^{(m+1)} = \frac{1}{9} (b_1 - x_2^{(m)} - x_3^{(m)}) \]
\[ x_2^{(m+1)} = \frac{1}{10} (b_2 - 2x_1^{(m)} - 3x_3^{(m)}) \]
\[ x_3^{(m+1)} = \frac{1}{11} (b_3 - 3x_1^{(m)} - 4x_2^{(m)}) \]

This is called the Gauss-Seidel method or the method of successive replacements.

Convergence Analysis

The above methods can be explained in a common framework.

Rewrite the linear system \( Ax = b \) as \( Nx = b + Px \) where \( A = N + P \) is a "splitting" of \( A \).

The matrix \( N \) must be nonsingular and it is usually chosen so that the linear systems \( Nx = f \) are relatively easy to solve for general vectors \( f \). For example, \( N \) could be diagonal or triangular.

The iterative method is defined by \( Nx^{(k+1)} = b + Px^{(k)} \).
The iterative method is defined by \( N^{(m)} x^{(m)} = b + P x^{(m)} \).

For a general matrix \( A \in \mathbb{C}^{n \times n} \) of order \( n \), the Jacobi method is defined with \( N = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \) and \( P = N - A \), where for the Gauss–Seidel method \( N \) is defined with \( N = \begin{bmatrix} a_{11} & a_{12} & \cdots & 0 \\ a_{21} & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2} & \cdots & a_{nn} \end{bmatrix} \) and \( P = N - A \).

Notice that \( N z^0 \) is easy to solve because \( N \) is diagonal or triangular, and we will ensure that \( N^{(m)} \) exists.

For the convergence analysis, we define \( \|2\| = \max_{i \in \mathbb{N}} \|z_{2i}\| \) and \( \|4\| = \max_{i \in \mathbb{N}} \|a_{i1}\| \).

It is easy to check that \( 1 \) \( \|y^0\| \leq \|y^1\| \leq \|y^2\| \) \( \Rightarrow \|A + B\| \leq \|A\| + \|B\| \) \( \Rightarrow \|AB\| \leq \|A\| \|B\| \) \( \Rightarrow \|A^2\| \leq \|A\|^2 \| \).

By \( N x = b + P x \) and \( N^{(m)} x^{(m)} = b + P x^{(m)} \), we have \( N e^{(m)} = P e^{(m)} \) and hence \( e^{(m)} = N^{(m)} P e^{(m)} \).

By induction, \( \|e^{(m)}\| \leq \|N^{(m)} P\| \|e^{(m)}\| \), and thus the error converges to zero if \( \|N^{(m)} P\| < 1 \).

For the Jacobi method, \( N^{(m)} P = \begin{bmatrix} 0 & -a_{12} & \cdots & -a_{1n} \\ -a_{21} & 0 & \cdots & -a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -a_{n1} & -a_{n2} & \cdots & 0 \end{bmatrix} \) and so \( \|N^{(m)} P\| = \max_{i \in \mathbb{N}} \|\frac{a_{i1}}{a_{ii}}\| \).

Thus the condition \( \|N^{(m)} P\| < 1 \) is equivalent to \( \frac{a_{i1}}{a_{ii}} < 1 \).

A matrix satisfying this condition is called diagonally dominant, and the Jacobi method converges for all such linear systems \( A x = b \).

For the Gauss–Seidel method, it can be shown that if the matrix is diagonally dominant, then it converges at least as quickly as the Jacobi method.

In general, this method converges for all right-hand sides \( b \) and for all initial guesses \( x^{(0)} \) if and only if all eigenvalues \( \lambda \) of \( N^{(m)} P \) satisfy \( |\lambda| < 1 \). One direction is to use the fact that if \( |\lambda| < 1 \) for all eigenvalues \( \lambda \), then \( \lim_{m \to \infty} (N^{(m)} P)^k = 0 \) and thus the error tends to zero. Another direction is to observe that if \( |\lambda| > 1 \) for some \( \lambda \), then when the corresponding eigenvector is the initial guess, the algorithm will not converge.

---

**Krylov Subspace Methods** [7]

Given a matrix \( A \) and a vector \( b \), the associated Krylov sequence is the set of vectors \( b, Ab, A^2 b, A^3 b, ... \).

The corresponding Krylov subspaces are the spaces spanned by successively larger groups of these vectors.

The Krylov subspace methods are based on finding vectors in the Krylov subspaces that give the best
approximation to solution $x$ of $Ax=b$.

The standard algorithm of this kind is known as GMRES, which stands for "generalized minimal residuals". Our goal is to solve $Ax=b$. Let $x$ be a solution.

The idea of GMRES is simple. At step $i$, we approximate $x$ by the vector $x_i \in X_i$ that minimizes the norm of the residual $r_i = b-Ax_i$, where $X_i$ denotes the Krylov subspace $<b, Ab, \ldots, A^{i-1}b>$.

So, in each iteration, we compute $x_i$ such that $x_i \in X_i$ and $\|Ax_i - b\|_2$ is minimum.

This is the least squares problem, a basic problem in numerical analysis, which is defined as follows.

In the least squares problem, we are given $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$ where $m \geq n$, and our goal is to find $x \in \mathbb{R}^n$ such that $\|Ax - b\|_2$ is minimized.

(In our setting, we would like to find $x \in \mathbb{R}^n$ such that $\|Ax - b\|_2$ is minimized where $A_{nm} = \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix}$.)

Let $r = b-Ax$.

The main observation in solving the least squares problem is that $x$ minimizes $\|Ax - b\|_2$ if and only if $r \perp \text{range}(A)$, or equivalently, $A^T r = 0$. Pictorially, $r \in \text{range}(A^T)$.

To see this, suppose $z$ is another point in range$(A)$. Since $r = b-Ax$ is orthogonal to $z - y$, the Pythagorean theorem gives $\|b-Ax\|_2^2 = \|b-y\|_2^2 + \|y-z\|_2^2 > \|b-y\|_2^2$, and so $\|Ax - b\|_2$ is minimized iff $A^T r = 0$.

This is equivalent to $A^T(b-Ax) = 0$, iff $A^T A x = 0$ iff $x = (A^T A)^{-1}A^T b$. So, a least squares problem can be solved in $O(n^3)$ time, but this is too slow for our purpose, as we can directly solve $Ax=b$ in $O(n^2)$ time.

For GMRES, the problems at different iterations are very similar, and we can solve these problems incrementally so that each iteration takes $O(n^2)$ time (see [5] for details).

So, the efficiency of the algorithm depends on how many iterations needed to get a reasonably good approximation, and we will study this in next week.

---

Conjugate Gradient Method

The conjugate gradient iteration is the "original" Krylov subspace iteration.

Here is the famous iteration. We assume that the matrix $A$ is symmetric and positive definite.

$$x_0 = 0, \quad r_0 = b, \quad d_0 = r_0$$
\[ x_0 = 0, \ r_0 = b, \ d_0 = r_0 \]
\[ \text{for } i = 1, 2, 3, \ldots \]
\[ \alpha_i = \frac{(r_i^T r_i) / (d_i^T A d_i)}{\text{step length}} \]
\[ x_{i+1} = x_i + \alpha_i d_i \]
\[ r_{i+1} = r_i - \alpha_i A d_i \]
\[ \beta_{i+1} = \frac{(r_{i+1}^T r_{i+1}) / (r_i^T r_i)}{\text{improve this step}} \]
\[ d_{i+1} = r_{i+1} + \beta_{i+1} d_i \]
\[ \text{search direction} \]

It is very simple and very efficient, when \( A \) is a sparse matrix then one iteration takes only \( O(n) \) steps where \( A \) is an \( n \times n \) matrix.

To understand these five lines of code, however, we follow the fifty page exposition \( [5] \) to derive this.

---

**Quadratic Program**

There are three main ingredients in the conjugate gradient algorithm.

The first one is to think of solving \( Ax = b \) as minimizing a quadratic program.

Let \( f(x) = \frac{1}{2} x^T A x - b^T x + c \).

A necessary condition of \( x \) being a minimizer of \( f(x) \) is that \( f'(x) = 0 \), where \( f'(x) = \begin{pmatrix} \frac{\partial f(x)}{\partial x} \\ \frac{\partial f(x)}{\partial x} \end{pmatrix} \) is the gradient of \( f \) at \( x \).

In our case, \( f'(x) = \frac{1}{2} A x - b \), and \( \frac{\partial f(x)}{\partial x} = a_{ij} x_j + \frac{1}{2} a_{ij} x_j + \frac{1}{2} a_{jk} x_k - b_i = \frac{1}{2} a_{ij} x_j - b_i \),

and thus \( f(x) = Ax - b \), and so \( x \) is a minimizer to \( f \) only if \( Ax = b \).

If \( A \) is symmetric and positive definite, then this is a global minimum of \( f \).

Let \( x \) be a point that satisfies \( Ax = b \) and \( e \) be an error term.

Then \( f(x + e) = \frac{1}{2} (x + e)^T A (x + e) - b^T (x + e) + c = \frac{1}{2} x^T A x + x^T A e + \frac{1}{2} e^T A e - b^T x - b^T e + e^T c = f(x) + \frac{1}{2} e^T A e \).

Since \( A \) is positive definite, \( \frac{1}{2} e^T A e > 0 \) for all \( e \neq 0 \), and hence \( f(x + e) > f(x) \) for all \( e \neq 0 \).

Therefore, \( f(x) \) is a global minimum when \( A \) is symmetric positive definite.
(All figures are shamelessly taken from [3].) In the figures, we have $A = \begin{bmatrix} 3 & 2 \\ 2 & 3 \end{bmatrix}$, $b = \begin{bmatrix} 2 \\ -8 \end{bmatrix}$, $c = 0$.

Figure (a) is the standard way of visualizing the solution of $Ax = b$. Figure (b) is the quadratic program perspective when $A$ is positive definite, and figure (c) plots the contours of $f$, where each ellipsoidal curve has constant $f(x)$.

Steepest Descent

Now we have transformed the problem into an optimization problem, and we can use techniques from optimization to solve the problem.

To motivate the conjugate gradient algorithm, we first consider the more intuitive steepest descent method. The idea is simple to state. We start from an arbitrary point $x_0$, in each iteration we choose the direction in which $f$ decreases most quickly, and repeat.

Let $\hat{e}_i = x_i - x$ be the error in iteration $i$, and $r_i = b - Ax_i$ be the residual that indicates how far we are from the correct value of $b$.

Notice that $Ar_i = Ax_i - Ax_i - b = -r_i$, and we will use this fact many times.

Observe that $f'(x_i)$ is the direction where $f$ increases most quickly, and so $-f'(x_i)$ is the direction where $f$ decreases most quickly, and so $r_i = b - Ax_i = -f'(x_i)$ is the direction of steepest descent.

Once we have decided a direction, we will set $x_{i+1} = x_i + \alpha r_i$. How should we choose $\alpha$?

A natural choice is to choose $\alpha$ that minimizes $f$ along this line.

By basic calculus, this is the point where $\frac{d}{\alpha} f(x_i + \alpha r_i) = 0$.

By chain rule, $\frac{d}{\alpha} f(x_i + \alpha r_i) = f'(x_i + \alpha r_i) \cdot \frac{d}{\alpha} (x_i + \alpha r_i) = f'(x_i + \alpha r_i) \cdot \alpha r_i = -r_i^T r_i$.

So, we should choose $\alpha$ so that $r_i^T r_i = 0 \Leftrightarrow (Ax_i - b)^T r_i = 0 \Leftrightarrow (Ax_i + \alpha Ar_i - b)^T r_i = 0$. 

L10 Page 6
Figures (from [8]): Some examples of convergence of Steepest Descent. The search direction is orthogonal to the previous search direction, but the zig-zag shape can repeat many times.

One can do a convergence analysis for steepest descent (see [8]). In general it is not too fast.

Conjugate Directions

In steepest descent, the search directions are repeated many times.

It would be nice if every time we took a step in a direction we can get it right.

For example, we may hope to get one coordinate right at a time, i.e. choose \( \alpha_i \) to be the \( i \)-th unit vector. Then we want \( \alpha_i^T \tilde{d}_i = 0 \) \( \Rightarrow \) \((\alpha_i + \alpha_i \tilde{d}_i)^T \tilde{d}_i = 0 \) \( \Rightarrow \) \( \alpha_i = -\frac{\tilde{d}_i^T \tilde{d}_i}{\tilde{d}_i^T \tilde{d}_i} \).

The problem is that we don't know \( \alpha_i \) (otherwise we have solved the problem already).

The second main ingredient of conjugate gradient is to make the search directions A-orthogonal instead of orthogonal.

Two vectors \( \tilde{d}_i \) and \( \tilde{d}_j \) are A-orthogonal or conjugate if \( \tilde{d}_i^T A \tilde{d}_j = 0 \).

The new requirement is that \( \tilde{d}_i \) and \( \tilde{d}_j \) are A-orthogonal.

In fact, this is equivalent to finding the minimum point along the search direction \( \tilde{d}_i \).

To see this, set the directional derivative to zero:

\[ \frac{df(\lambda \tilde{d}_i)}{d \lambda} = 0 \quad \Rightarrow \quad \phi^T(\lambda \tilde{d}_i) \tilde{d}_i = 0 \quad \Rightarrow \quad -\tilde{r}_i \tilde{d}_i = 0 \]

\[ \Rightarrow \quad \tilde{d}_i^T A \tilde{d}_{i+1} = 0. \]

So, \( \alpha_i \) should be chosen such that \( \tilde{d}_i^T A \tilde{e}_{i+1} = \tilde{d}_i^T A (\tilde{e}_i + \alpha \tilde{d}_i) = 0 \) \( \Rightarrow \) \( \alpha_i = -\frac{\tilde{d}_i^T A \tilde{e}_i}{\tilde{d}_i^T A \tilde{d}_i} = \frac{\tilde{d}_i^T \tilde{r}_i}{\tilde{d}_i^T A \tilde{d}_i} \).

The point is that \( \alpha_i \) can be computed from the information that we know.

Now, suppose we have n A-orthogonal search directions \( \tilde{d}_0, \tilde{d}_1, \ldots, \tilde{d}_n \). We show that we
only need to search each direction at most once, and get the correct answer in $n$ steps.

Write the initial error $e_0$ as a linear combination of $d_i$, that is $e_0 = \sum_{i=0}^{n} b_i d_i$.

The coefficients $b_i$ can be computed as follows:

$$d_i^T A e_0 = \frac{1}{a_{ii}} \sum_{j=0}^{n} b_j d_j d_i^T A d_j = b_i d_i^T A d_i \quad \text{(by orthogonality)}.$$

The important point here is that $a_{ii} = b_i$.

So, in each step, we cancel one term in the summation of the initial error term.

Therefore, in $n$ steps, the algorithm will give the correct answer.

More precisely, using the search directions $d_0, d_1, \ldots, d_n$, $x_i = x_0 + \frac{1}{a_{ii}} a_{ij} d_i$, and thus

$$x_i = x_0 - \frac{a_{ii}}{a_{ij}} d_i = \sum_{j=0}^{n} b_j d_j d_i^T A d_i = \sum_{j=0}^{n} b_j d_i^T A d_j \quad \text{(since $a_{ii} = b_i$)}.$$

**Gram-Schmidt Conjugation**

The next step is to find $n$ search directions $d_0, d_1, \ldots, d_n$ that are $A$-orthogonal.

One can obtain such search directions from $n$ linearly independent vectors $u_0, u_1, \ldots, u_n$, by a process similar to the Gram-Schmidt orthogonalization, subtracting components that are not $A$-orthogonal to the previous vectors.

Set $d_0 = u_0$ and $d_i = u_i + \frac{1}{a_{ii}} \sum_{j=0}^{i-1} b_{ij} d_j$.

Using the same idea as above, we multiply $A d_j$ on both sides to get $d_i^T A d_j = u_i^T A d_j + \frac{1}{a_{ii}} \sum_{j=0}^{i-1} b_{ij} d_j^T A d_j$.

Since we want $d_i^T A d_j = 0$, we should set $b_{ij} = -\frac{a_{ij}}{a_{ii}}$.

For example, we can set $u_0, u_1, \ldots, u_n$, to be the standard basis and apply the conjugate gradient method, but it would take $O(n^3)$ steps to compute $d_0, d_1, \ldots, d_n$, making this approach useless.

Later on we will find a better basis to generate $d_0, d_1, \ldots, d_n$.

**Optimality of the Error Term**

Conjugate gradient method has the nice property that it finds the best solution within the search space.

Let $D_i = \text{span} \{d_0, d_1, \ldots, d_i\}$.

The conjugate gradient method minimizes the $A$-norm of the error term, denoted by $\|e_i\|_A = (e_i^T A e_i)^{1/2}$.
over the search space $x_0 + D_i$.

Recall that $f(x_{k+1}) = f(x_k) + \frac{1}{2} \Delta^2 \mathbf{A} \mathbf{e}$, and so minimizing the $\mathbf{A}$-norm of the error term is equivalent to minimizing $f$ over the search space $x_0 + D_i$.

To see this, again write $\mathbf{e}_i = \sum_{j=1}^{n} \mathbf{b}_j \mathbf{d}_j$, then

$$
\mathbf{e}_i^T \mathbf{A} \mathbf{e}_i = \left( \sum_{j=1}^{n} \mathbf{b}_j \mathbf{d}_j \right)^T \mathbf{A} \left( \sum_{j=1}^{n} \mathbf{b}_j \mathbf{d}_j \right)
= \sum_{j=1}^{n} \mathbf{b}_j^T \mathbf{d}_j \mathbf{A} \mathbf{d}_j
= \sum_{j=1}^{n} \mathbf{b}_j^T \mathbf{d}_j \mathbf{A} \mathbf{d}_j
$$

by orthogonality.

Notice that this expression only depends on the future directions, and they appear in every error term from the space $x_0 + D_i$, and thus $\mathbf{e}_i$ minimizes the $\mathbf{A}$-norm.

---

**Residual**

An important property of the residual is that it is orthogonal to the previous search directions.

To see this, for $i < j$, $d_i^T r_j = -d_i^T \mathbf{A} \mathbf{e}_j = -d_i^T \mathbf{A} \left( \sum_{k=1}^{n} \mathbf{b}_k \mathbf{d}_k \right) = -\sum_{k=1}^{n} \mathbf{b}_k^T \mathbf{d}_j \mathbf{A} \mathbf{d}_k = 0$ by orthogonality.

Since the search directions are constructed from the $\mathbf{e}_k$, the residual $r_i$ is also orthogonal to the subspace spanned by $\mathbf{u}_0, ..., \mathbf{u}_{i-1}$. To see this, recall that $d_i = \mathbf{u}_i + \sum_{k=0}^{i-1} \beta_k \mathbf{e}_k$, and thus $0 = d_i^T r_j = \mathbf{u}_i^T r_j + \sum_{k=0}^{i-1} \beta_k r_k^T r_j$, since $i < j$ and $d_k^T r_j = 0$ for $k < i < j$.

When $i = j$, we have $d_i^T r_i = \mathbf{u}_i^T r_i$.

Finally, we observe that $r_{i+1} = A \mathbf{e}_{i+1} = -A \mathbf{e}_i - \alpha d_i = r_i - \alpha d_i$.

---

**Conjugate Gradient Algorithm**

We are now ready to derive the conjugate gradient algorithm.

The final main ingredient is to set $\mathbf{u}_i = r_i$. This is possible since $r_i$ will be orthogonal to the previous $\mathbf{d}_i$ and $\mathbf{u}_j$, and so the new $\mathbf{u}_i = r_i$ is linearly independent from $\mathbf{u}_0, ..., \mathbf{u}_{i-1}$.

Since we know $\mathbf{u}_i^T \mathbf{r}_j = 0$ for $i < j$, by setting $\mathbf{u}_i = r_i$, we have $r_i^T r_j = 0$ for $i < j$.

Recall that $\mathbf{r}_{i+1} = r_i - \alpha \mathbf{A} \mathbf{d}_i$. Since we now set $\mathbf{u}_{i+1} = r_{i+1}$, so $D_{i+1}$ is formed from the union of the previous subspace $D_i$ and the subspace $\mathbf{A} \mathbf{d}_i$, and thus $D_i = \text{span} \{ \mathbf{d}_0, \mathbf{A} \mathbf{d}_0, ..., \mathbf{A}^{i-1} \mathbf{d}_0 \} = \text{span} \{ \mathbf{r}_0, \mathbf{A} \mathbf{r}_0, ..., \mathbf{A}^{i-1} \mathbf{r}_0 \}$. Initially, $\mathbf{r}_0 = \mathbf{b}$ and this is just the krylov subspace span\{$\mathbf{b}, \mathbf{A} \mathbf{b}, ..., \mathbf{A}^{i} \mathbf{b}$\}.

The big advantage of setting $\mathbf{u}_i = r_i$ is the computational efficiency.

As $r_{i+1}$ is orthogonal to $\mathbf{d}_0, ..., \mathbf{d}_i$ and $\mathbf{u}_0, ..., \mathbf{u}_i$, the Gram-Schmidt process can be greatly simplified.

Recall that $d_i = \mathbf{u}_i + \sum_{k=0}^{i-1} \beta_k \mathbf{d}_k$ and $\mathbf{p}_i = -\mathbf{u}_i^T \mathbf{A} \mathbf{d}_i = -r_i^T \mathbf{A} \mathbf{d}_i$ as we set $\mathbf{u}_i = r_i$. 

---
Recall that \( r_{ij} = r_{ij} - \alpha_j \tilde{A}_{ij} \), and thus 
\[
\begin{align*}
\frac{r_{ij}^T}{r_{j1}^T} = r_{ij}^T - \alpha_j r_{ij} \tilde{A}_{ij} \quad \text{and} \quad \frac{r_{ij}^T}{r_{j1}^T} \tilde{A}_{ij} = r_{ij}^T - \alpha_j r_{ij} \tilde{A}_{ij},
\end{align*}
\]
As \( r_{ij}^T r_{j1} = 0 \) for \( i < j \), we have 
\[
\frac{r_{ij}^T}{r_{j1}^T} \tilde{A}_{ij} = \begin{cases} 
\frac{r_{ij}^T}{r_{j1}^T} & \text{when } i = j \\
0 & \text{when } i > j \end{cases}.
\]

Therefore, \( \beta_{ij} = \begin{cases} 
\frac{r_{ij}^T}{r_{j1}^T} & \text{when } i = j + 1 \\
0 & \text{when } i > j + 1 
\end{cases} \)

Let \( \beta_i = \beta_{i,i-1} = \frac{r_i^T}{d_i^T} r_{i-1} \),
\[
\begin{align*}
\alpha_i &= \frac{d_i^T r_{i}}{d_i^T A d_i} \quad \text{step length} \\
x_i &= x_{i-1} + \alpha_i d_i \quad \text{approximate solution} \\
r_i &= r_{i-1} - \alpha_i A d_i \quad \text{residual} \\
\beta_i &= \frac{r_i^T r_{i-1}}{r_{i-1}^T r_{i-1}} \quad \text{improvement} \\
d_i &= r_{i-1} + \beta_i d_i \quad \text{search direction}
\end{align*}
\]

Putting together, we set \( x_0 = 0 \) and \( d_0 = r_0 = b - Ax_0 = b \).

for \( i = 0, 1, 2, 3, \ldots (\leq n-1) \)

We have rederived the magical five lines, and proved that it will converge in \( n \) steps.

Note that the storage required for this algorithm is \( O(n) \) (besides the matrix \( A \)), and actually we do not need to know \( A \) as long as we have a function (an oracle) to perform matrix vector multiplication with \( A \). This is sometimes called “black box” linear algebra.

Next week we will see the convergence analysis of the conjugate gradient method, using the elegant connection of its spectrum and polynomial approximation.

References

