ERROR MEASUREMENTS and ANALYSIS [4-33]

BF-9 §7.5, p 470–72

For the system \( A\vec{x} = \vec{b} \), let \( x^* \) be the exact solution, i.e., \( Ax^* = \vec{b} \). Let \( x_n \) be an approximate (computed) solution, \( e_n = x^* - x_n \) be the corresponding error vector, and \( r_n = \vec{b} - Ax_n \) be the corresponding residual vector.

Then (from the residual eq.) \( Ax_n = \vec{b} - r_n \) and \( e_n = x^* - x_n \) implies \( Ae_n = A(x^* - x_n) = Ax^* - Ax_n = \vec{b} - (\vec{b} - r_n) \).

Therefore \( Ae_n = r_n \) and \( e_n = A^{-1}r_n \).

REMEMBER (cf. p. [4-16R], Def of matrix norm), that one def. gave \( \|A\| = \max_{\vec{x}\neq0} \frac{\|A\vec{x}\|}{\|\vec{x}\|} \) which implies \( \|A\| \geq \frac{\|A\vec{x}\|}{\|\vec{x}\|} \) or \( \|A\| \cdot \|\vec{x}\| \geq \|A\vec{x}\| \).

Thus \( Ae_n = r_n \Rightarrow \|r_n\| \leq \|A\| \cdot \|e_n\| \) and \( e_n = A^{-1}r_n \Rightarrow \|e_n\| \leq \|A^{-1}\| \cdot \|r_n\| \).
Take $\|r_n\| \leq \|A\| \cdot \|e_n\|$ and $\|e_n\| \leq \|A^{-1}\| \cdot \|r_n\|$ and divide the first by $\|A\| \cdot \|x^*\|$ and the second by $\|x^*\|$, we get

(1) \[ \frac{\|r_n\|}{\|A\| \cdot \|x^*\|} \leq \frac{\|e_n\|}{\|x^*\|} \quad \text{and} \quad (2) \frac{\|e_n\|}{\|x^*\|} \leq \|A^{-1}\| \frac{\|r_n\|}{\|x^*\|}\]

Since $\vec{b} = Ax^*$ and $x^* = A^{-1}\vec{b}$, we have, similar to the above, (3) $\|\vec{b}\| \leq \|A\| \cdot \|x^*\|$ and (4) $\|x^*\| \leq \|A^{-1}\| \cdot \|\vec{b}\|$. From (4) we get (5) \[ \frac{1}{\|A^{-1}\| \cdot \|\vec{b}\|} \leq \frac{1}{\|x^*\|} \quad \text{and from (3) we get} \quad \frac{1}{\|\vec{b}\|} \geq \frac{1}{\|A\| \cdot \|x^*\|} \quad \text{or (6) } \frac{1}{\|x^*\|} \leq \frac{\|A\|}{\|\vec{b}\|}. \]

Substituting (5) into (1) and (6) into (2), we get

\[ \frac{1}{\|A\| \cdot \|A^{-1}\|} \frac{\|r_n\|}{\|\vec{b}\|} \leq \frac{\|e_n\|}{\|x^*\|} \leq \|A\| \cdot \|A^{-1}\| \frac{\|r_n\|}{\|\vec{b}\|} \]
Since the condition number of the matrix $A$, $\text{cond}(A) = \kappa(A) = \|A\|\|A^{-1}\|$, we have

$$\frac{1}{\kappa(A)} \frac{\|r_n\|}{\|\vec{b}\|} \leq \frac{\|e_n\|}{\|\vec{x}^*\|} \leq \kappa(A) \frac{\|r_n\|}{\|\vec{b}\|}.$$ 

So what does this mean?

$\Rightarrow$ The ratio of the error in the solution to the actual solution (i.e., the relative error) can be as large as the ratio of the residual to the constant vector (i.e., the “relative residual error”) TIMES THE CONDITION NUMBER OF THE SYSTEM MATRIX.
If $\kappa(A) = 1$, we get equality in the above inequality, i.e.,

$$\frac{\|e_n\|}{\|x^*\|} = \frac{\|r_n\|}{\|\vec{b}\|}.$$

However, if $\kappa(A)$ is large, e.g., $10^{14}$, even if $\frac{\|r_n\|}{\|\vec{b}\|}$ is small, the relative error can still be large.

As mentioned before (cf. p [4-16AR]), the common rule of thumb for interpreting condition number is this:

$$\text{if } \kappa(A) = c \cdot 10^n \text{ where } 1 \leq c \leq 10,$$

then the numbers of $A$ and $\vec{b}$ should have greater accuracy than $n$ significant digits.
EXAMPLE OF PROBLEMS

\[ A\vec{x} = \vec{b} \]

<table>
<thead>
<tr>
<th>( A )</th>
<th>( \vec{b} )</th>
<th>( \vec{x} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-0.1) 1</td>
<td>-2</td>
<td>10</td>
</tr>
<tr>
<td>0.12 -1</td>
<td>2.2</td>
<td>-1</td>
</tr>
<tr>
<td>-0.1 1</td>
<td>-2</td>
<td>0</td>
</tr>
<tr>
<td>0.12 -1</td>
<td>2.0</td>
<td>-2</td>
</tr>
<tr>
<td>-0.1 1</td>
<td>-2</td>
<td>20</td>
</tr>
<tr>
<td>0.11 -1</td>
<td>2.2</td>
<td>0</td>
</tr>
<tr>
<td>-0.1 1</td>
<td>-2</td>
<td>10</td>
</tr>
<tr>
<td>0.11 -1</td>
<td>2.1</td>
<td>-1</td>
</tr>
</tbody>
</table>

\[ A\vec{x} = \vec{b} \text{ exact} \]

changing \( \vec{b} \) induces large change in \( \vec{x} \)

changing \( A \) induces large change in \( \vec{x} \)

changing both \( A, \vec{b} \) \( \Rightarrow \) NO change in \( \vec{x} \)

NOTE: \( \kappa_2(A) = 101.2101 \approx 10^2 \) via Matlab.
BRIEF ANALYSIS: LOOK AT EIGENVALUES OF $A$

$$\begin{vmatrix} -0.1 - \lambda & 1 \\ 0.12 & -1\lambda \end{vmatrix} = 0 \quad \Rightarrow \quad (0.1 + \lambda)(1 + \lambda) - 0.12 = 0$$

$$\lambda^2 + 1.1\lambda - 0.02 = 0$$

$$\Rightarrow \lambda = \frac{-1.1 \pm \sqrt{1.1^2 - 4 \cdot 1 \cdot (-0.02)}}{2} = \frac{-1.1 \pm \sqrt{1.29}}{2} = \frac{-1.1 \pm 1.1357816}{2}$$

$$= -2.2357816, \quad 0.0357816$$

$$\lambda = \frac{2 \cdot 1}{2} = \frac{0.0357816}{2}$$

$$\lambda = -1.1178908, \quad 0.0178908$$

Absolute ratio of $\lambda_{\text{max}}/\lambda_{\text{min}} = 62.48411 \approx 10^2$. 
A is not symmetric, so \( \frac{\lambda_{\text{max}}}{\lambda_{\text{min}}} = 62.48 \) is not an actual condition number. However, we can get a “feel” for the condition number from this.

Since the ratio of eigenvalues is close to \( 10^2 \) (and one condition number is actually greater than \( 10^2 \)), we would want \( A \) and \( b \) to be accurate to 2 or 3 significant digits in order to assure that the answer is accurate to at least 1 digit!

There are ways to approximate the largest and smallest eigenvalues to estimate the actual condition number.

We can also look at the matrix itself. If the proportioning of the various elements is bad (very large vs. very small), ill-conditioning is very possible.
E.g., Hilbert Matrix

\[ H_n = \begin{pmatrix} 1 & 1/2 & 1/3 & \ldots & 1/n \\ 1/2 & 1/3 & 1/4 & \ldots & 1/(n + 1) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1/n & 1/(n + 1) & 1/(n + 2) & \ldots & 1/(2n + 1) \end{pmatrix} \]

Inverses are known as well as the condition numbers:

\[ \kappa(H_3) = 5.24 \times 10^2 \]
\[ \kappa(H_5) = 4.77 \times 10^5 \]
\[ \kappa(H_7) = 4.75 \times 10^8 \]
\[ \kappa(H_{10}) = 1.6 \times 10^{13} \]
ITERATIVE SOLUTIONS OF $A\vec{x} = \vec{b}$ [4-37]

Jacobi’s Method: Take a single equation, e.g., the $j^{th}$ equation of a system and solve for $x_j$.

$$\sum_{i=1}^{n} a_{ji} x_i = b_j \Rightarrow a_{jj} x_j = b_j - \sum_{i \neq j} a_{ji} x_i$$

$$\Rightarrow x_j = \frac{1}{a_{jj}} \left[ b_j - \sum_{i \neq j} a_{ji} x_i \right]$$

$$\Rightarrow x_j = \frac{1}{a_{jj}} \left[ b_j - \sum_{i \neq j} a_{ji} x_i \right] + \left( x_j - \frac{a_{jj} x_j}{a_{jj}} \right)$$

$$\Rightarrow x_j = x_j + \frac{1}{a_{jj}} \left[ b_j - \sum_{i=1}^{n} a_{ji} x_i \right]$$

BF-9 §7.3, p 450
Let the left side be the “new” $x_j$ and we get Jacobi’s Method.

$$\Rightarrow x_j^{new} = x_j^{old} + \frac{1}{a_{jj}} \left[ b_j - \sum_{i=1}^{n} a_{ji} x_i^{old} \right]$$

Gauss-Seidel

If we use the most recent value of the $x$-variables on the right side, we get the Gauss-Seidel method.

$$\Rightarrow x_j^{new} = x_j^{old} + \Delta x = x_j^{old} + \frac{1}{a_{jj}} \left[ b_j - \sum_{i=1}^{n} a_{ji} x_i^{most \ recent} \right]$$

Note, however, that the order of equations can be important!
IMPROVING CONVERGENCE

1. Aitken’s formula

2. Relaxation

Rewrite Gauss-Seidel with a parameter

\[
\Rightarrow x_j^{new} = x_j^{old} + \omega \Delta x = x_j^{old} + \omega \frac{1}{a_{jj}} \left[ b_j - \sum_{i=1}^{n} a_{ji} x_i^{most \ recent} \right]
\]

\(\omega\) is called the RELAXATION or ACCELERATION parameter.

If \(0 < \omega < 1\), it is called UNDER-RELAXATION.

If \(1 < \omega < 2\), it is called OVER-RELAXATION.

If \(\omega = 1\), it is GAUSS-SEIDEL.
Frequently, we speak of “Successive Over-Relaxation” or SOR methods.

Finding $\omega$ is not easy. Often it is done by trial and error for a few iterations. But the “correct” $\omega$ can drastically accelerate convergence.
CONJUGATE GRADIENT METHOD (1952) [4-39] 

This method works for a symmetric positive definite matrix.

Choose initial $x_0$ arbitrary.

Calculate $p_0 = r_0 = b - Ax_0$

Iterative loop

\[ a_i = \frac{(p_i, r_i)}{(p_i, Ap_i)} \] (distance for shifting $x_i$)  \hspace{1cm} (BF-9 p 485, $t_k = a_i$)

\[ x_{i+1} = x_i + a_ip_i \]

\[ r_{i+1} = r_i - a_iAp_i \]

\[ b_i = -\frac{(r_{i+1}, Ap_i)}{(p_i, Ap_i)} \] \hspace{1cm} (BF-9 p 485, $s_k = b_i$)

\[ p_{i+1} = r_{i+1} + b_ip_i \] (direction for shifting $x_i$) \hspace{1cm} (BF-9 p 485, $v_i = p_i$)

NOTE: In exact arithmetic, convergence is GUARANTEED in $n$ steps or less (where $n$ is the size of $A$).
A-ORTHOGONALITY [4-39A]

One variation of orthogonality (of 2 vectors) is called “A-orthogonality.”

Def. Two vectors \( \vec{u} \) and \( \vec{v} \) are called \( A \)-orthogonal iff \( \langle \vec{u}, A\vec{v} \rangle = 0 \) (whenever \( \vec{u} \neq \vec{v} \)).

The Conjugate Gradient (CG) method builds a set of \( A \)-orthogonal vectors \( p_i \) in the direction of steepest descent toward the solution point.
EXAMPLE

Given

\[\begin{align*}
4x_1 + 3x_2 &= 24 \\
3x_1 + 4x_2 - x_3 &= 30 \\
-x_2 + 4x_3 &= -24
\end{align*}\]

we rewrite it in the form of \( A\vec{x} = \vec{b} \) to get

\[
\begin{pmatrix}
4 & 3 & 0 \\
3 & 4 & -1 \\
0 & -1 & 4
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
x_3
\end{pmatrix}
=
\begin{pmatrix}
24 \\
30 \\
-24
\end{pmatrix}
\]

We assert that \( p_1^T = (1, 0, 0)^T \), \( p_2^T = (-3/4, 1, 0)^T \), 
\( p_3^T = (-3/7, 4/7, 1)^T \) form an \( A \)-orthogonal set (i.e., every pair is \( A \)-orthogonal). (overhead slide [4-39-B])
CG uses the residual (cf. notes [4-32]) to create a direction vector $p_i$ and determines a distance $a_i$. (BF-9 p 480) Using these quantities, CG adjusts the current approximate solution $x_i$ and creates a new approximation via the formula $x_{i+1} = x_i + a_ip_{i+1}$. The basic aim is to achieve a residual with all components 0.
Using the system given earlier and vectors \( \{p_1, p_2, p_3\} \) we get: (BF-9 p 483)

\[
\begin{align*}
x_0 &= (0, 0, 0)^T \Rightarrow r_0 = b - Ax_0 = (24, 30, -24)^T \\
\Rightarrow a_0 &= \frac{(p_1, r_0)}{(p_1, Ap_1)} = \frac{24}{4} = 6 \\
x_1 &= x_0 + 6 \cdot (1, 0, 0)^T = (6, 0, 0)^T \\
\Rightarrow r_1 &= (0, 12, -24)^T \quad \text{[NOTE: 0 is 1st comp.]} \\
a_1 &= \frac{48}{7} \\
x_2 &= x_1 + a_1 p_2 = (6, 0, 0)^T + \frac{48}{7}(-3/4, 1, 0)^T = (6/7, 48/7, 0)^T \\
\Rightarrow r_2 &= (0, 0, -120/7)^T \quad \text{[NOTE: 0 in 1st 2 comp.]} \\
a_2 &= -5
\end{align*}
\]
\[ x_3 = (6/7, 48/7, 0)^T + (-5)(-3/7, 4/7, 1)^T = (3, 4, -5)^T \quad [\text{Exact sol.}] \]

Note the decrease in the norm of the residuals, even though the \( x_i \)-values do not seem to be converging:

\[ |r_0| = 45.3, \quad |r_1| = 26.8, \quad |r_2| = 17.14 \]
Suppose $x^*$ is the desired solution of $Ax = b$.

Suppose we can determine a matrix $M$ approximately equal to $A$ (in some sense) whose inverse is (relatively) easy to compute.

Thus, we hope, $M^{-1} \approx A^{-1}$ and therefore $M^{-1}A \approx I$. Thus,

$$M^{-1}Ax = M^{-1}b \approx x^*$$

$M^{-1}$ is called the preconditioning matrix (or preconditioner).

Even if $M^{-1}A$ is not close to being $I$, often the new system $M^{-1}Ax = M^{-1}b$ is far easier to solve than the original.

One main challenge in Numerical Analysis today is to find easy-to-compute preconditioners.
MISCELLANEOUS

There are variations of the CG method that are popular, e.g., Bi-Conjugate Gradient for non-symmetric systems, etc.

⇒ Assignment: Find numbers of pages on Google (or other search engine) for some iterative solvers.