2.3 Iterative methods for $AX = B$

As the name denotes we will now attempt to solve the system $AX = B$ with an iterative scheme instead of a direct method. The difference is that the solution produced by any of the direct methods presented in the previous section is exact and is determined immediately. In contrast, as is often the case with any iterative scheme, their solutions are obtained after a number of iterations and are not exact but only approximations up to a given tolerance near the true solution.

As we will see in the following section iterative techniques are quite useful when the number of equations to be solved is large (i.e. the size of the matrix is large). Furthermore such methods tend to be stable with regards to matrices $A$ with large condition number. As a result small initial errors do not pile up during the iterative process thus blowing up in the end.

2.4 Jacobi, Richardson and Gauss-Seidel methods

We start by “discovering” the Jacobi and Gauss-Seidel iterative methods with a simple example in two dimensions. The general treatment for either method will be presented after the example.

The most basic iterative scheme is considered to be the Jacobi iteration. It is based on a very simple idea: solve each row of your system for the diagonal entry. Thus if for instance we wish to solve the following system,

\[
\begin{bmatrix}
4 & -3 \\
-2 & 5
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2
\end{bmatrix} =
\begin{bmatrix}
5 \\
6
\end{bmatrix}
\]

we first solve each row for the diagonal element and obtain

\[
\begin{align*}
x_1 &= \frac{3}{4} x_2 + \frac{5}{4} \\
x_2 &= \frac{2}{5} x_1 + \frac{6}{5}
\end{align*}
\]

Thus the Jacobi iterative scheme starts with some guess for $x_1$ and $x_2$ on the right hand side of this equation and hopefully produces after several iterations improved estimates which approach the true solution $\mathbf{x}$. In matrix form the system above can be written as,

\[
\begin{bmatrix}
x_1^m \\
x_2^m
\end{bmatrix} =
\begin{bmatrix}
0 & 3/4 \\
2/5 & 0
\end{bmatrix}
\begin{bmatrix}
x_1^{m-1} \\
x_2^{m-1}
\end{bmatrix} +
\begin{bmatrix}
5/4 \\
6/5
\end{bmatrix}
\]

where you can clearly see how the iteration is progressing. Your previous estimate for the solution $x^{m-1}$ goes in the right hand side and you obtain a new estimate (which is supposed to be better) in the left hand side. Let us examine the output of the Jacobi scheme for a few iterations based on this numerical example. We will assume that the initial guess is taken to be, without loss of generality, $x^0 = [0, 0]^T$

<table>
<thead>
<tr>
<th>$n$</th>
<th>0</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>Exact Sol.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1^n$</td>
<td>0</td>
<td>2.907500</td>
<td>3.063965</td>
<td>3.071030</td>
<td>3.071404</td>
<td>3.0714285714</td>
</tr>
<tr>
<td>$x_2^n$</td>
<td>0</td>
<td>2.318000</td>
<td>2.422670</td>
<td>2.428302</td>
<td>2.428557</td>
<td>2.4285714285</td>
</tr>
</tbody>
</table>

A very simple but effective improvement has been suggested to the Jacobi scheme. It is the Gauss-Seidel method which simply uses the new value of $x_1$ in the second row of (2.2). Thus the Gauss-Seidel
method for the example above is,
\[
\begin{bmatrix}
  x_1^m \\
  x_2^m
\end{bmatrix} = \begin{bmatrix}
  0 & 3/4 \\
  2/5 & 0
\end{bmatrix} \begin{bmatrix}
  x_1^{m-1} \\
  x_2^{m-1}
\end{bmatrix} + \begin{bmatrix}
  5/4 \\
  6/5
\end{bmatrix}
\] (2.4)

Let us similarly compare a small number of iterates using the Gauss-Seidel method in the following table.

<table>
<thead>
<tr>
<th>n</th>
<th>0</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>Exact Sol.</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x_1^n)</td>
<td>0</td>
<td>3.056675</td>
<td>3.071392</td>
<td>3.071428</td>
<td>3.071428</td>
<td>3.0714285714</td>
</tr>
<tr>
<td>(x_2^n)</td>
<td>0</td>
<td>2.422670</td>
<td>2.428557</td>
<td>2.428571</td>
<td>2.428571</td>
<td>2.4285714285</td>
</tr>
</tbody>
</table>

You may be wondering, rightfully so, whether it really is that simple.... In other words, whether the method, as outlined above, works all the time. The answer is NO! The reason that things worked out so nicely in the example presented above is that matrix A is diagonally dominant. In fact we provide the relevant theorems in terms of when things are expected to work out for either the Jacobi or the Gauss-Seidel method below (see Theorems 2.4.3 & 2.4.4).

**Generalization of iterative methods**

We will now generalize our findings and produce a general theory under which to study iterative schemes. In order to do this we make use of an auxiliary matrix \(Q\) to be specified later. The idea relies on what we learned earlier about fixed point problems in one dimension. Let us start by outlining our general set-up. We start as usual from the main system of equations in matrix form
\[
Ax = B
\] (2.5)

where as we have seen before \(A\) and \(B\) are known while \(x\) denotes the vector of the unknowns. First we bring the \(Ax\) term to the left hand side \(0 = -AX + B\) and then we add an auxiliary vector \(Qx\) on both sides of (2.5) and produce the following system
\[
Qx = (Q - A)x + B
\] (2.6)

This new system will be used in order to define our iteration as follows
\[
Qx_m = (Q - A)x_{m-1} + B
\]

First we observe that in fact the solution of (2.6) is simply found from
\[
x = Q^{-1}(Q - A)x + Q^{-1}B = (I - Q^{-1}A)x + Q^{-1}B.
\]

The iterative scheme corresponding to this set-up is clearly recognizable as a **Fixed Point Problem** in \(n\)-dimensions:
\[
x_m = Gx_{m-1} + C
\] (2.7)

where \(G = I - Q^{-1}A\) and \(C = Q^{-1}B\).

The iterative process can now be initiated with a given initial vector \(x^0\) for the solution. This is usually only a guess but if any information is known about the solution should be used in obtaining a better initial such guess for \(x_0\). Given this set-up the only (and most important) thing left to do is choose a matrix \(Q\) so that the iterative process outlined in (2.7) will
• converge to the true solution $x$.
• produce the solution in a small number of iterations

We have already seen a couple of iterative methods which did fulfill these tasks in one way or another. Let us look at a more general approach in constructing such iteratives schemes. Suppose that we can write $A$ as,

$$A = D - L - U$$

where as usual $D$ is diagonal matrix and $L, U$ are lower and upper triangular matrices respectively (with zeros in their diagonal). Then the matrices for each iterative method are given below.

- Jacobi: if $Q = D$ in (2.7) then $G = -D^{-1}(L + U)$ and $C = D^{-1}B$
- Richardson: if $Q = I$ in (2.7) then $G = I - A$ and $C = B$
- Gauss-Seidel: if $Q = D + L$ in (2.7) then $G = -(D + L)^{-1}U$ and $C = (D + L)^{-1}B$

It is important to know when we can expect to have a solution of (2.7). Is it possible to always have a solution of this iterative scheme? The answer is naturally no! We develop below a result which indicates whether we should expect our iteration to be successful or not.

We first define what we mean by “convergence” of an iterative method.

**Definition 2.4.1.** An $n \times n$ matrix $A$ is said to be convergent if

$$\lim_{m \to \infty} A^m(i, j) = 0 \quad \text{for } i, j = 1, 2, \ldots, n$$

Further the following holds:

**Theorem 2.4.2.** The following are equivalent:

- $A$ is a convergent matrix
- $\lim_{m \to \infty} \|A^m\| = 0$
- $\lim_{m \to \infty} A^m x = 0$ for every $x$.
- $\rho(A) < 1$

where $\rho(A)$ denotes the spectral radius of the matrix $A$ which is essentially the largest, in absolute value, eigenvalue of $A$. Then the following theorem gives a very useful result,

**Theorem 2.4.3.** The iterative scheme $x_m = Gx_{m-1} + C$ converges to the unique solution of $x = Gx + C$ for any initial guess $x_0$ if and only if $\rho(G) < 1$.

**Proof:** Subtracting $x = Gx + C$ from $x_m = Gx_{m-1} + C$ we obtain,

$$x_m - x = G(x_{m-1} - x)$$

Simply taking norms on both sides of the above we have,

$$\|x_m - x\| = \|G(x_{m-1} - x)\| \leq \|G\| \|x_{m-1} - x\|$$
Applying this inequality repeatedly for \( m - 1, m - 2, \ldots \) we obtain

\[
\|x_m - x\| \leq \|G\|\|(x_{m-1} - x)\| \\
\leq \|G\|^2\|(x_{m-2} - x)\| \\
\vdots \\
\leq \|G\|^m\|(x_0 - x)\| 
\]

Thus clearly from the above if we assume that \( \rho(G) \leq 1 \) based on Theorem 2.4.2 we have,

\[
\lim_{m \to \infty} \|G\|^m = 0 
\]

and

\[
\|x_m - x\| = 0 
\]

Thus convergence. We leave the opposite direction of this proof to the reader since it follows from this outline.

However this proof is in fact instructive in terms of answering other interesting questions such as how many iterations of the Jacobi iteration are necessary in order for the solution to be found within a given tolerance? Let us look at such an example.

**Example:**
Find the number of iterations so that the Jacobi method starting from the vector \( x_0 = [0, 0]^T \) will reach the solution with a relative error tolerance of \( 10^{-4} \) for the following matrix \( A \),

\[
A = \begin{bmatrix} 4 & -3 \\ -2 & 5 \end{bmatrix} 
\]

**Solution:**
We need to approach the solution \( x \) using the Jacobi iteration,

\[
x_m = Gx_{m-1} + C \tag{2.8}
\]

Suppose then that the solution is \( x \). Then iteration (2.8) has to be satisfied for the solution \( x \) as follows,

\[
x = Gx + C
\]

Subtracting these two equations and taking norms we obtain,

\[
\|x_m - x\| \leq \|G\|\|x_{m-1} - x\|
\]

Repeating this \( m - 1 \) more times we obtain,

\[
\|x_m - x\| \leq \|G\|^m\|x_0 - x\|
\]

Note however that for this problem we have chosen \( x_0 = [0, 0]^T \). Thus the above becomes,

\[
\|x_m - x\| \leq \|G\|^m\|x\|
\]
or
\[ \frac{\|x_m - x\|}{\|x\|} \leq \|G\|^m \]

We know from our matrix algebra review that we may employ the spectral radius in order to calculate the Euclidean norm (you may try other norms if you like instead) of \( G \) as follows: \( \|G\|_2 = \sqrt{\rho(G^T G)} \). Thus using the Euclidean norm everywhere we obtain that the relative error should be
\[ \frac{\|x_m - x\|^2}{\|x\|^2} \leq \rho(G^T G)^{m/2} \]

Note that in fact the left hand side is nothing more than the relative error. Therefore in order to find out how many iterations are necessary in order to approach the solution within a relative error tolerance of \( 10^{-4} \) we must solve for \( m \) the following equation
\[ (\rho(G^T G))^{m/2} = 10^{-4} \]

Since \( \rho(G^T G) \approx .5625 \) then \( \|G\|_2 = .75 \) and most importantly that
\[ m = \frac{\ln 10^{-4}}{\ln \|G\|_2} = \frac{-4\ln 10}{\ln 3/4} = 32.0157 \]

Thus if we choose \( m = 33 \) we should be within \( 10^{-4} \) of the true solution for this system.

Let us now look at some theoretical results for each of the methods presented so far.

**Theorem 2.4.4.** If \( A \) is diagonally dominant then the sequence produced by either the Jacobi or the Gauss-Seidel iterations converges to the solution of \( Ax = B \) for any starting guess \( x_0 \).

We outline the proof here only for the Jacobi iteration since the Gauss-Seidel is similar.

**Proof:** Note that the Jacobi iteration matrix \( G \) can be written as,
\[ G = -D^{-1}(L + U) \]

In that case taking the matrix norm of the above and rearranging we get,
\[ \|G\|_\infty = \frac{\|L + U\|_\infty}{\|D\|_\infty} = \frac{\max_{1 \leq i \leq n} \sum_{j \neq i} |A(i, j)|}{\max_{1 \leq i \leq n} |A(i, i)|} \leq 1 \]

where the last inequality holds simply by the definition of \( A \) being diagonally dominant.

### 2.5 Comparisons

In terms of speed you should always keep in mind that iterative, direct or other methods always depend on the problem at hand. For instance each iteration using either the Gauss-Seidel or Jacobi method requires about \( n^2 \) operations. However if you are solving a small size system of equations then Gaussian elimination is much faster. Take for example a small \( 3 \times 3 \) system. If you perform a Jacobi iteration on it you will require about 9 operations per iteration and you may need to perform more than 100 iterations to obtain a very good estimate. Thus a total of at least 900 operations. On the other hand Gaussian elimination only needs to perform \( 3^3 = 27 \) operations to solve the
whole system and produce the exact solution! In fact it can be shown that \textit{iteration if preferable to Gaussian elimination} if

\[
\frac{\ln \epsilon}{\ln \rho} < \frac{n}{3}
\]  

(2.9)

Here \( n \) corresponds to the size of the matrix \( A \), \( \rho \) refers to the spectral radius of the iterative scheme \( \rho \equiv \rho(G) \) and \( \epsilon \) is the given relative error tolerance we wish to obtain from the iteration.

Let us look at a simple example of this result:

\textit{Example:}

As usual we wish to solve the matrix system \( Ax = B \). Suppose that \( A \) is a 30 \( \times \) 30 matrix and that the spectral radius for the Gauss-Seidel iterative scheme is found to be \( \rho(G) = .4 \). Suppose also that we wish to find the solution accurate to within \( \epsilon = 10^{-5} \). Is it best to perform the Gauss-Seidel iteration or just simple Gaussian elimination?

\textit{Solution:}

Note that for this example

\[
\frac{\ln \epsilon}{\ln \rho(G)} = \frac{-11.51}{-.91} = 12.56
\]

Therefore inequality (2.9) gives

\[
12.56 < \frac{30}{3} = 10
\]

Not true! Thus in this case Gaussian elimination is actually going to be faster!

One thing to keep in mind is that in fact there are matrix systems for which one method might converge while the other might not (the reason being that the spectral radius of the iteration is not less than 1). So speed is not everything when it comes to iterative schemes. Let us outline some important points about these methods and compare them with other techniques:

- Gauss-Seidel is faster than Jacobi
- Gauss-Seidel and Jacobi methods have a cost which is about \( n^2 \) operations.
- One iterative scheme may converge to the solution while another may not. This may depend on the choice of initial guess \( x_0 \) but more importantly on the spectral radius of the iterative scheme \( (\rho(G) < 1) \).
- Gaussian elimination although it costs about \( n^3 \) operations may be faster when it comes to moderate size systems.

Let us see the pseudo-code for some methods:

\textbf{Jacobi:}

Suppose that we are provided with a matrix \( A \) a vector \( B \) and a starting guess vector \( x_0 \).

1. For \( i = 1 \) to \( n \) do \( Y(i) = x_0(i) \)
2. While a given tolerance \( \epsilon \) is satisfied do the following:
   - For \( i = 1 \) to \( n \) do
     \[
     Z(i) = \frac{B(i) - \sum_{j=1}^{i-1} A(i,j)Y(j) - \sum_{j=i+1}^{n} A(i,j)Y(j)}{A(i,i)}
     \]
• For \( i = 1 \) to \( n \) do \( Y(i) = Z(i) \).

3. Print out the vector \( Z \).

Gauss-Seidel:
Suppose that we are provided with a matrix \( A \) a vector \( B \) and a starting guess vector \( x_0 \).
1. For \( i = 1 \) to \( n \) do \( Y(i) = x_0(i) \)
2. While a given tolerance \( \epsilon \) is satisfied do the following:
   • For \( i = 1 \) to \( n \) do the following two steps:
     \[
     Z(i) = \frac{B(i) - \sum_{j=1}^{i-1} A(i, j) Y(j) - \sum_{j=i+1}^{n} A(i, j) Y(j)}{A(i, i)}
     \]
     \[
     Y(i) = Z(i)
     \]
3. Print out the vector \( Z \).

However we can in fact come up with methods which can converge to the solution under appropriate conditions even faster (see SOR and SSOR methods for instance).