Notes from March 26 – Tuesday

- The algorithm of Gaussian Elimination for the solution of simultaneous equation will be discussed. It has been proven that it requires the minimum number of floating point operations to solve a set of linear equation if the matrix is full.
- In a computer, a set of simultaneous equations is first written in a matrix form. For example,
  
  \[
  \begin{align*}
  4x + 3y &= 5 \\
  2x - y &= 5
  \end{align*}
  \]

  could be written in matrix form as

  \[
  \begin{bmatrix}
  4 & 3 \\
  2 & -1
  \end{bmatrix}
  \begin{bmatrix}
  x \\
  y
  \end{bmatrix}
  =
  \begin{bmatrix}
  5 \\
  5
  \end{bmatrix}
  \]

  If you understand the concept of matrix multiplication, the above matrix equation is directly equivalent to the set of simultaneous equations. Inside a computer program, the alphanumeric names of x and y are not keep, but an augmented matrix with numbers only, e.g.,

  \[
  \begin{bmatrix}
  4 & 3 & 5 \\
  2 & -1 & 5
  \end{bmatrix}
  \]

  is used instead. The numbers are then operated on without the variable names; it is assumed that the meaning of the numbers is understood.
- The most important idea about Gaussian Elimination is that it uses row operations. If an equation has a left side and a right side which are equal, i.e., LEFT=RIGHT. It is permitted to multiply both sides by a nonzero constant, i.e., \(a \times \text{LEFT} = a \times \text{RIGHT}\), after the operation, both sides are still equal. It is also permitted to add or subtract a constant from both sides, i.e., LEFT+b=RIGHT+b, or Left-c=RIGHT-c. In general, two equations could be added or subtracted and the results would still be the same, i.e., LEFT1+LEFT2=RIGHT1+RIGHT2.
- Define a constant, alpha as the ratio of the two leading coefficients of

  \[
  \begin{bmatrix}
  4 & 3 & 5 \\
  2 & -1 & 5
  \end{bmatrix}
  \]

  \(\text{alpha} = \frac{a(2,1)}{a(1,1)} = \frac{2}{4} = 0.5\). Now perform the operation

  ROW2 replaced by ROW2 – alpha* ROW1

  \[
  \begin{bmatrix}
  2 & -1 & 5 \\
  \end{bmatrix} - 0.5 \times \begin{bmatrix}
  4 & 3 & 5 \\
  2 & -1 & 5
  \end{bmatrix}
  \]

  is \[
  \begin{bmatrix}
  0 & -2.5 & 2.5
  \end{bmatrix}
  \]. Note that the leading element is 0 by design. After ROW2 is replaced by the new result, the augmented matrix becomes

  \[
  \begin{bmatrix}
  4 & 3 & 5 \\
  0 & -2.5 & 2.5
  \end{bmatrix}
  \]
• The original matrix is now altered, but the LEFT side matrix is now in an upper triangular form. The simultaneous equations are now easily solved because the last row is uncoupled from the other equations. The procedure is now known as back substitution. From the knowledge of how the augmented matrix was created, ROW2 represents the equation: \(-2.5*y=2.5\), it can be solved readily as \(y=-1\).

• After ROW2 was solved, ROW1 has now only one unknown and the equation could be written as \(4*x+3*(-1)=5\), and the solution could be obtained as \(x=(5-3*(-1))/4\). The solution is then \([x \ y]=[2 \ -1]\).

• Below is an example of a larger set of simultaneous equations, or in matrix form, \([A]\{x\}={b}\), in which \([A]\) is an n-by-n matrix and \({b}\) is an n-by-1 column vector.

\[
\begin{bmatrix}
4 & 3 & 0 & -2 & 4 & 1 \\
3 & -1 & -1 & 3 & 4 & 2 \\
-7 & 0 & 0 & 3 & 4 & 3 \\
3 & 2 & 2 & -4 & 9 & 4 \\
8 & -2 & 3 & 5 & 2 & 5
\end{bmatrix}
\]

• The first step is to create a set of alpha factors for all the rows below the first row so that after the loop, all the values below the first row have a value of 0. The code for this case of n=5 can be written as:

```plaintext
i=1;
for j=i+1:n
    alpha=a(j,i)/a(i,i);
    a(j,:)=a(j,:)-alpha*a(i,:);
end
```

After the operation, the new altered matrix would have the form:

\[
\begin{bmatrix}
4 & 3.00 & 0 & -2.0 & 4 \\
0 & -3.25 & -1 & 4.5 & 1 \\
0 & 5.25 & 0 & -0.5 & 11 \\
0 & -0.25 & 2 & -2.5 & 6 \\
0 & -8.00 & 3 & 9.0 & -6
\end{bmatrix}
\]

The first row serves as the pivot row in the above case. The same operation can be performed using the second row as a pivot row so that all rows below the second row would have a 0 value beneath the new matrix element \(a(2,2)\). The alpha factors would be calculated based on \(a(2,2)\). This procedure could then be repeated for the 3rd row with \(a(3,3)\) as the pivot element and so forth.

• The above verbally described algorithm could be written using pivot row \(i\) from 1 to \(n-1\) because there is no row below row \(n\). The reduction of the lower row \(j\) would be from row \(i+1\) to \(n\).
The algorithm can be written in a matlab script as

```matlab
for i=1:n-1
    for j=i+1:n
        alpha=a(j,i)/a(i,i);
        a(j,:)=a(j,:)-alpha*a(i,:);
    end
end
```

The results for matrix \([A]\) after the above code would be:

\[
\begin{bmatrix}
4 & 3.00 & 0 & -2.0000 & 4.0000 \\
0 & -3.25 & -1.0000 & 4.5000 & 1.0000 \\
0 & 0 & -1.6154 & 6.7692 & 12.6154 \\
0 & 0 & 0 & 5.8571 & 22.1429 \\
0 & 0 & 0 & 0 & -44.4797
\end{bmatrix}
\]

The above illustration did row reduction only on the left side, to retain the correct value, the row reduction procedure should include the RIGHT side vector \([b]\). See

```matlab
for i=1:n-1
    for j=i+1:n
        alpha=a(j,i)/a(i,i);
        a(j,i:n)=a(j,i:n)-alpha*a(i,i:n);
        b(j)=b(j)-alpha*b(i);
    end
end
```

Note: the reduction for the matrix \([A]\) in the above code uses a shorter row to avoid operations involving zeroes. The results after the above procedure would be:

\[
\begin{bmatrix}
4 & 3.00 & 0 & -2.0000 & 4.0000 & 1.0000 \\
0 & -3.25 & -1.0000 & 4.5000 & 1.0000 & 1.2500 \\
0 & 0 & -1.6154 & 6.7692 & 12.6154 & (b)= 6.7692 \\
0 & 0 & 0 & 5.8571 & 22.1429 & 11.8571 \\
0 & 0 & 0 & 0 & -44.4797 & -19.3171
\end{bmatrix}
\]

The last row is now uncoupled from the rest of the equations, so the solution for \(x(n)\) is \(b(n)=b(n)/a(n,n)\). Since there was no vector created for the solution vector \([x]\), it is a normal practice to store the result back into \([b]\) since after the entire operation, both \([A]\) and \([b]\) would be changed from the original matrix.

The back substitution algorithm could be written in matlab as

```matlab
b(n)=b(n)/a(n,n);
for i=n-1:-1:1
    b(i)=(b(i)-a(i,i+1:n)*b(i+1:n))/a(i,i);
end
```

The solution of the matrix equation is the column vector: \([b]= [0.0165; 0.0313; 0.8042; 0.3826; 0.4343]\)
Finally, the entire Gaussian Elimination code could be written in a matlab file, gauss0.m, as

```matlab
function b=gauss0(a,b,n)
% Row Reduction----
for i=1:n-1
    for j=i+1:n
        alpha=a(j,i)/a(i,i);
        a(j,i:n)=a(j,i:n)-alpha*a(i,i:n);
        b(j)=b(j)-alpha*b(i);
    end
end
% Back Substitution----
    b(n)=b(n)/a(n,n);
    for i=n-1:-1:1
        b(i)=(b(i)-a(i,i+1:n)*b(i+1:n))/a(i,i);
    end
```

The name gauss0 was used for the above code because the important concept of partial pivoting has not been implemented. It will be a subject in the next lecture.
Application of FFT in Time Response Calculations

To obtain a solution to an equation of motion, it is necessary to set up a basis and preferably, an orthogonal basis. In particle dynamics, the orthogonal vectors, \( \hat{i}, \hat{j}, \) and \( \hat{k} \) are used. In continuum mechanics, orthogonal functions such as Legendre Polynomials or Fourier Series are used to represent the solutions. While a Fourier Series is used in a finite interval, the Fourier Transform is used in an infinite interval.

The Fourier Transform "pair" can be expressed as

\[
F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t} \, dt, \\
f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{i\omega t} \, d\omega.
\]

The factor, \( 1/2\pi \), is sometimes distributed evenly to both part of the pair as \( \sqrt{1/2\pi} \).

The Fourier Transformation, when it is applied to differential equations, can transform a differential equation into an algebraic equation. Consider the simple equation of a single degree of freedom oscillator excited by ground acceleration,

\[
m(\ddot{x} + \ddot{x}_g) + c\dot{x} + kx = 0,
\]

in which \( x \) is the displacement of the oscillator with respect to the ground and the ground acceleration is represented by, \( \ddot{x}_g \). Rearranging the equation by moving the ground excitation to the right side, we have

\[
m\ddot{x} + c\dot{x} + kx = -m\ddot{x}_g.
\]

Now define the solution \( x(t) \) as a Fourier Transform of \( X(\omega) \) as

\[
x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\omega) e^{i\omega t} \, d\omega
\]

and perform the time derivatives as follows:

\[
\dot{x}(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} i\omega X(\omega) e^{i\omega t} \, d\omega
\]

and

\[
\ddot{x}(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} (i\omega)^2 X(\omega) e^{i\omega t} \, d\omega
\]

The ground excitation, \( \ddot{x}_g \), can be expressed as

\[
\ddot{x}_g = \frac{1}{2\pi} \int_{-\infty}^{\infty} A(\omega) e^{i\omega t} \, d\omega
\]

if the excitation record is given as an accelerogram, i.e.,

\[
A(\omega) = \int_{-\infty}^{\infty} \ddot{x}_g(t) e^{-i\omega t} \, dt
\]
or it can be expressed as
\[ \ddot{x}_g = \frac{1}{2\pi} \int_{-\infty}^{\infty} (i\omega)^2 X_g(\omega) e^{i\omega t} \, d\omega \]
if the excitation record is given as a displacement record, i.e.,
\[ X_g(\omega) = \int_{-\infty}^{\infty} x_g(t) e^{-i\omega t} \, dt \]

Substitute the terms into the equation of motion using the transformed expressions:
\[ \frac{1}{2\pi} \int_{-\infty}^{\infty} \left[ -\omega^2 m X(\omega) + i\omega c X(\omega) + k X(\omega) \right] e^{i\omega t} \, d\omega = \frac{1}{2\pi} \int_{-\infty}^{\infty} \omega^2 m X_g(\omega) e^{i\omega t} \, d\omega \]
and for each value of \( \omega \), the solution, \( X(\omega) \), can be obtained as
\[ X(\omega) = \frac{\omega^2 m}{(k - \omega^2 m) + i\omega c} X_g(\omega) = \frac{\omega^2}{(k/m - \omega^2) + i\omega c/m} X_g(\omega) \]

Define now, \( \omega_n^2 = k/m \) and \( 2\omega_n\zeta_n = c/m \), then
\[ X(\omega) = \frac{\omega^2}{(\omega_n^2 - \omega^2) + i2\omega_n\zeta_n} X_g(\omega) = \frac{\omega^2/\omega_n^2}{(1 - \omega^2/\omega_n^2) + i\zeta_n \omega/\omega_n} X_g(\omega) \]

The factor,
\[ T(\omega) = \frac{\omega^2/\omega_n^2}{(1 - \omega^2/\omega_n^2) + i\zeta_n \omega/\omega_n} \]
is known as the transfer function between the input function and output function (the solution). In general, the response at a certain location can be calculated as the product of the transfer function and the input function and then transformed back into the time domain as
\[ x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} T(\omega) X_g(\omega) e^{i\omega t} \, d\omega \]

The same approach can be used in much more complicated problems.
Application of DFT and FFT algorithms

DFT, Discrete Fourier Transform, is the numerical version of analytical Fourier Transforms. Since a finite time increment, $\Delta t$, is specified, there is a limited resolution for frequency, known as the Nyquist Frequency. Also, a finite interval over time is necessary, restricted by $T = N\Delta t$, in which $N$ is the number of sample points used. The time signal is assumed to be repeat in consecutive intervals of length $T$.

FFT. Fast Fourier Transform, is an efficient algorithm in which $N$ is required to be a power of 2, e.g., 256, 1024, 4096, etc. DFT requires $N^2$ operations but FFT requires only $N \log_2 N$ operations. For example, if $N = 4096$, $N^2 = 16777216$, $N \log_2 N = 49152$, a factor of 340 times faster. There are many version of FFT available, but the version of particular interest for earthquake response problems is that the time function, $f(t)$, is real and for that reason, the complex Fourier Transform has the property,

$$F(-\omega) = \overline{F(\omega)},$$

therefore, only the half with positive $\omega$ has to be stored. The half with negative $\omega$ can be obtained as complex conjugates of the other half.

Some of the important parameters involved in FFT are:

- Input $N$ as a power of 2, $\Delta t$ as the time increment of $f(t)$. The time history, $f(t)$, is a real array of length $N$, but $N + 2$ must be reserved to receive the complex Fourier Transform.
- The frequency increment of the resulting complex Fourier Transform, $F(\omega)$, is

$$d\omega = \frac{2\pi}{N\Delta t}$$

and the number of complex values is

$$N_f = \frac{N}{2} + 1.$$

The frequencies range from 0 to the Nyquist Frequency, defined as

$$\omega_{\text{max}} = \frac{\pi}{\Delta t} \quad \text{or} \quad f_{\text{max}} = \frac{1}{2\Delta t}$$

- Since $F(\omega)$ is complex, it requires a memory location for the real and the imaginary parts, thus $2N_f$ or $N + 2$ memory locations is required to return the complex Fourier Transform. In the inverse transformation, a real time history of $N$ values is reconstructed from $N_f$ complex values.
- A normalizing factor, analogous to the $1/2\pi$ factor of the inverse Fourier Transform, is

$$c = \frac{N}{2}$$

for this particular version of the FFT algorithm.

Note: The value of $N$ should be large enough to have the oscillation die down. In some of the examples in the following two pages, the lowly damped oscillators do not have zero amplitudes with 1024 points, so it spilled over to the "silence" period at the beginning of the time histories. For example, in the second input function, the movement does not start before 1 second, therefore, the response function should also begin after 1 second.
Input ground displacement

ω_n = 5, ζ_n = 0.02

ω_n = 5, ζ_n = 0.05

ω_n = 5, ζ_n = 0.10

ω_n = 2, ζ_n = 0.02

ω_n = 2, ζ_n = 0.05

ω_n = 2, ζ_n = 0.10

ω_n = 10, ζ_n = 0.02

ω_n = 10, ζ_n = 0.05

ω_n = 10, ζ_n = 0.10
Input ground displacement

\( \omega_n = 5, \zeta_n = 0.02 \)

\( \omega_n = 5, \zeta_n = 0.05 \)

\( \omega_n = 5, \zeta_n = 0.10 \)

\( \omega_n = 2, \zeta_n = 0.02 \)

\( \omega_n = 2, \zeta_n = 0.05 \)

\( \omega_n = 2, \zeta_n = 0.10 \)

\( \omega_n = 10, \zeta_n = 0.02 \)

\( \omega_n = 10, \zeta_n = 0.05 \)

\( \omega_n = 10, \zeta_n = 0.10 \)
Input ground displacement

$\omega_n = 2, \zeta_n = 0.02, \ N=1024$

$\omega_n = 2, \zeta_n = 0.02, \ N=2048$

$\omega_n = 2, \zeta_n = 0.02, \ N=4096$

$\omega_n = 2, \zeta_n = 0.02, \ N=8192$

$\omega_n = 1, \zeta_n = 0.02, \ N=1024$

$\omega_n = 1, \zeta_n = 0.02, \ N=2048$

$\omega_n = 1, \zeta_n = 0.02, \ N=4096$

$\omega_n = 1, \zeta_n = 0.02, \ N=8192$

$\omega_n = 1, \zeta_n = 0.01, \ N=8192$