Lecture started with an example numerical integration using Gaussian Quadrature. The example was to integrate \((x+2)^5\) from -0.1 to 4.1. The formula for integration was derived previously, it is in the form of \(I = \alpha \sum wi f(zi)\), in which \(zi = \alpha + xi + \beta\). \([a,b]\) is the integration interval, \(\alpha = (b-a)/2\) and \(\beta = (b+a)/2\). \(xi\) and \(wi\) are the sample locations and sample weights for Gaussian integration.

In particular, the case of \(N=3\) was used. The sample locations or abscissa are \(x1 = -sqrt(3/5)\), \(x2 = 0\), \(x3 = sqrt(3/5)\); the corresponding weights are \(5/9\), \(8/9\) and \(5/9\). In the example, \(\alpha = 2.15\), \(\beta = 1.95\), the numerical integration result is 8581.06. It will be compared by a matlab implementation.

The Gaussian Quadrature implementation will be similar to those for Trapezoidal Rule and Simpson’s Rule. It is just that the values of \(wi\) and \(wi\) have to be given through the argument list.

The class decided the program name should be gaussian.m and the code generated was

```matlab
function answer=gaussian(f,a,b,N,xi,wi)
alpha=(b-a)/2;
beta=(b+a)/2;
answer=0;
for i=1:N
    answer=answer+wi(i)*f(alpha*xi(i)+beta)
end
answer=alpha*answer;
```

After the file was saved, the program was executed to demonstrate its effectiveness:

```matlab
N=3;
xi(1)=-sqrt(3/5);
xi(2)=0;
xi(3)=-xi(1);
wi(1)=5/9;
wi(2)=8/9;
wi(3)=5/9;
doc=@(x) (x+2)^5;
result=gaussian(doc,-0.2,4.1,N,xi,wi);
format long
result
8.58106035616666e+03
```

The program was also applied to do one of the home work problems for the case of \(N=3\).

```matlab
hwkN3=@(x) exp(-x^2/4);
```
A recall from a previous lecture that to implement polynomials in matlab, a row vector (a matrix) can be used to represent it by storing its coefficients only. For example, \(x^2+2x-3\) could be represented as \([1 2 -3]\). When it is multiplied by \(x\) to become \(x^3+2x^2-3x+0\), it could be written as \([1 2 -3 0]\). Whenever a polynomial is multiplied by \(x\), the coefficients can simply be shifted to the left. A demonstration was made:

\[
\begin{align*}
a &= \begin{bmatrix} 1 & 2 & -3 \end{bmatrix}; \\
a &= \begin{bmatrix} a & 0 \end{bmatrix} \\
1 & 2 & -3 & 0 \\
\end{align*}
\]

\[
\begin{align*}
b &= \begin{bmatrix} 1 & 1 \end{bmatrix}; \\
b &= \begin{bmatrix} 0 & 0 & b \end{bmatrix} \\
0 & 0 & 1 & 1 \\
\end{align*}
\]

\[
a + b \\
1 & 2 & -2 & 1
\]

The above array manipulation was need to perform the recursion relationship for the Legendre Polynomials given in the Project 8 handout. The recursion formula takes \(P_n\) and multiply it by \(x\) and a coefficient and subtract a coefficient times \(P_{n-1}\) to form the new polynomial \(P_{n+1}\). The fact \(P_n\) is multiplied by \(x\), its coefficient vector is one larger than before. Meanwhile, \(P_{n-1}\) started one smaller than \(P_n\) must be increased in size by two to be able to be added or subtracted. The mechanism is now in place to program a file to get the roots of the Legendre Polynomial.

An example of the simplest case was done first, the case with \(n=1\), it will be coded in getxw.m. The variable names \(pnm1\) refers to \(P_{n-1}\), \(pn\) refers to \(P_n\) and \(pnpl\) refers to \(P_{n+1}\). Since \(n=1\), \(n-1=0\) and \(n+1=2\). Using \(P0=1\) and \(P1=x\) (well known properties of Legendre Polynomial), \(P2\) can be obtained when \(n=1\).

```matlab
function xi=getxw(N)
    n=1;
    pnm1=1;
    pn=[1 0];
    c1=(2*n+1)/(n+1);
    c2=n/(n+1);
    pnpl=c1*[pn 0]-c2*[0 0 pnm1];
    xi=roots(pnpl);
```
In the above program, \( n=1 \), it produced \( P_2=3x^2-1 \) and that is why the roots came out to be \( 1/\sqrt{3} \). The code could now be modified to handle the case of \( n=2 \), using \( P_1 \) and \( P_2 \), \( P_3 \) can be obtained the same way. These statement can be added to the above file and soon it is realized a loop could be used to get much higher order polynomials.

```
function xi=getxw(N)
    n=1;
    pnml=1;
    pn=[1 0];
    c1=(2*n+1)/(n+1);
    c2=n/(n+1);
    pnp1=c1*[pn 0]-c2*[0 0 pnml];
    pnml=pn;
    pn=pnp1;
    n=2;
    c1=(2*n+1)/(n+1);
    c2=n/(n+1);
    pnp1=c1*[pn 0]-c2*[0 0 pnml];
    xi=roots(pnp1);
end
```

```
getxw(2)
-0.577350269189626
 0.577350269189626
```

To get the roots in order, change the last line to
```
xi=sort(roots(pnp1));
```

```
getxw(3)
-0.7745966692
 0.7745966692
 0
```

It is clear from the previous code that a loop could be used because a number of lines were repeated. The argument \( N \) was also never used. To get \( P_N \), recycle the loop \( N-1 \) times as \( pnp1 \) would be the \( P_N \) we needed.

```
function xi=getxw(N)
    pnml=1;
    pn=[1 0];
    for n=1,N-1
        c1=(2*n+1)/(n+1);
        c2=n/(n+1);
        pnp1=c1*[pn 0]-c2*[0 0 pnml];
    end
    xi=roots(pnp1);
end
```
To get 6 \( \xi \) for \( N=6 \), just do
\[
\xi = \text{getxw}(6)
\]
- To go the next level to get \( \xi \) and \( \omega \), the formula calls for \( P_{N+1} \) in the formula for \( \xi \). Therefore, we need to use \( P_N \) to get the roots but \( P_{N+1} \) to get the values of \( \omega \). For that reason, the \( n \) loop must go one extra loop, using “for \( n=1,N \)” but at the end of the loop, we do not want the last two line to prepare for another loop, i.e., erasing \( p_n \) with \( p_{n+1} \). So a few minor revisions must be made carefully:

\[
\text{function } [\xi,\omega] = \text{getxw}(N)
\]

\[
p_{n+1} = \text{getxw}(N)
\]

\[
p_n = \begin{bmatrix} 1 & 0 \end{bmatrix};
\]

\[
\text{for } n=1,N
\]

\[
c_1 = (2*n+1)/(n+1);
\]

\[
c_2 = n/(n+1);
\]

\[
p_{n+1} = c_1*\begin{bmatrix} p_n & 0 \end{bmatrix} - c_2*\begin{bmatrix} 0 & 0 & p_{n-1} \end{bmatrix};
\]

\[
\text{if } (n<N)
\]

\[
p_n = \text{getxw}(N);
\]

\[
p_{n+1} = \text{getxw}(N);
\]

\[
\text{end}
\]

\[
\xi = \text{sort}(\text{roots}(p_n));
\]

\[
\% \text{determination of weights}
\]

\[
\text{for } i=1:N
\]

\[
x = \xi(i);
\]

\[
c_3 = 2*(1-x^2)/(N+1)^2;
\]

\[
ex = N+1:-1:0
\]

\[
P = \text{sum}(p_{n+1}.*(x.*ex));
\]

\[
\omega(i) = c_3/P^2;
\]

\[
\text{end}
\]

- It is important to note that the roots were obtained using \( P_N \) but the weights must be calculated by using the values of \( P_{N+1} \) at the sample locations of \( \xi \). The students should now test getxw for various values of \( N \) to compare with the table values given online.

- The project and the homework can now be done using the two m files developed during the class, for example, to do the numerical integration using \( N=5 \) can now be done by entering in a main program the code:

\[
\text{hwkN5} = @(x) \exp(-x^2/4);
\]
N=5;
[xi, wi] = getxw(N);
result = gaussian(hwkN5, -0.25, 2.5, N, xi, wi);
result
2.125572780240008