Function definitions :
LagrangeInterp- symbolycially generates the Lagrange interpolation functions for each point, across all the points in data and with the variable x
fun - defines the functional
feval - takes the limit of the functions f as it approaches the points in data pre- vents erant results when the function is evaluated at a singularity
P - symbolically generates the Lagrange polynominal to degree n, the roots of which are the node points for Lagrange interpolation and Gauss Legrende quadrature

```
LagrangeInterp[data_, x_] /; MatrixQ[data] :=
 Flatten[Module[{n = Length[data], xl}, {xl = data}; Table[
     Product[(x - xl[[i]]) / (xl[[j]] - xl[[i]]), {i, Complement[Range[n], {j}]}], {j, n}]]^T]

fun[x_] := (Exp[x] - 1) / x

feval[f_, data_] := Module[{n = Length[data]}, Table[Limit[f, x → data[[j, 1]]], {j, 1, n}]]

P[n_] := Simplify[Sum[1 / (2^n) * ((-1)^i (2 n - 2 i) ! x^(n - 2 i)) / (i ! (n - 2 i) ! (n - i) !),
   {i, 0, IntegerPart[n / 2]}]]
```

The exact answer for integrating the functional from - 1 to 1

```
ExactAns = Integrate[fun[x], {x, -1, 1}]
N[%, 15]
```

```
2 Shi(1)

2.11450175075146
```

Using Lagrange interpolation points nodes

Initializing Arrays

```
fl = ConstantArray[0, 6];
Ifl = ConstantArray[0, 6];
errftn = ConstantArray[0, 6];
errl = ConstantArray[0, 6];
```

Generate the nodal interpolation points from the roots of the Lagrange polynominal in this case for n=2,3,..7

```
ptsroot = Table[N[Root[P[i], j]], {i, 2, 7}, {j, 1, i}];
ncol = {Table["n= " <> ToString[i], {i, 2, 7}]}ᵀ;
Grid[Join[ncol, %%, 2], Frame → All]
```

n= 2	–0.57735	0.57735					
n= 3	–0.774597	0.	0.774597				
n= 4	–0.861136	–0.339981	0.339981	0.861136			
n= 5	–0.90618	–0.538469	0.	0.538469	0.90618		
n= 6	–0.93247	–0.661209	–0.238619	0.238619	0.661209	0.93247	
n= 7	–0.949108	–0.741531	–0.405845	0.	0.405845	0.741531	0.949108

The algrothym implimented is as follows:
1)Generates the Lagrange interpolation functions, LIF, for the appropriate points, ptsroot[[i]].
2)The sybolic interpolating poloynominals, fl, are generated by dot product of the symbolic LIF and the functional evaluated at the same points
3)The Newton Cotes method is used to generate the integral of the functional, Ifl. The weights are generated by integrating the symbolic LIF which are then multiplied and summed with the functional evaluated at the same points in a dot product operation
4)The symbolic error function is generated by subtracting the functional from the sybolic interpolating poloynominals
5)The error for the Newton Cotes integration, errl, is found by the absolute of the difference between Ifl and the previously calculated exact answer
The resulting errors are presented below

```
For[i = 1, i < 7, i++, {
  LIF = LagrangeInterp[{ptsroot[[i]]}ᵀ, x];
  fl[[i]] = Simplify[LIF.feval[fun[x], {ptsroot[[i]]}ᵀ]];
  Ifl[[i]] = Integrate[LIF, {x, -1, 1}].feval[fun[x], {ptsroot[[i]]}ᵀ];
  errftn[[i]] = fl[[i]] - fun[x];
  errl[[i]] = Abs[Ifl[[i]] - ExactAns]}]
```

The sybolic interpolating poloynominals generated above.

```
TableForm[{fl}ᵀ, TableHeadings → {ncol, None}]
```

{n= 2}	$0.514044\,x + 1.05649$
{n= 3}	$0.171739\,x^2 + 0.525505\,x + 1.$
{n= 4}	$0.0428734\,(x + 2.84086)\left(x^2 + 1.2162\,x + 8.20439\right)$
{n= 5}	$0.00855656\left(x^2 - 0.400678\,x + 11.4573\right)\left(x^2 + 5.45351\,x + 10.2004\right)$
{n= 6}	$0.0014231\,(x + 3.60124)\left(x^2 - 1.79888\,x + 15.5365\right)\left(x^2 + 4.24627\,x + 12.5592\right)$
{n= 7}	$0.000202912\left(x^2 - 3.07086\,x + 20.5504\right)\left(x^2 + 2.82532\,x + 15.2808\right)\left(x^2 + 7.29036\,x + 15.6937\right)$

Using Chebyshev nodes

Initializing Arrays

```
chebf = ConstantArray[0, 6];
Ichebf = ConstantArray[0, 6];
chebweight = ConstantArray[0, 6];
cheberr = ConstantArray[0, 6];
cheberrftn = ConstantArray[0, 6];
```

The nodal locations, Chebyshev nodes, from the roots of the symbolic Chebyshev polynomials, generated by an internal function

```
chebptsroot = Table[N[Root[ChebyshevT[i, x], j]], {i, 2, 7}, {j, 1, i}];
Grid[Join[ncol, %, 2], Frame → All]
```

n= 2	–0.707107	0.707107					
n= 3	–0.866025	0.	0.866025				
n= 4	–0.92388	–0.382683	0.382683	0.92388			
n= 5	–0.951057	–0.587785	0.	0.587785	0.951057		
n= 6	–0.965926	–0.707107	–0.258819	0.258819	0.707107	0.965926	
n= 7	–0.974928	–0.781831	–0.433884	0.	0.433884	0.781831	0.974928

Integration using the Chebyshev nodes is carried out as above using Newton Cotes method and the Lagerangian interpolation functions evaluated at the Chebyshev nodes. The resulting errors are presented below.

```
For[i = 1, i < 7, i++, {
  chebLIF = LagrangeInterp[{chebptsroot[[i]]}ᵀ, x];
  chebf[[i]] = Simplify[chebLIF.feval[fun[x], {chebptsroot[[i]]}ᵀ]];
  Ichebf[[i]] = Integrate[chebLIF, {x, -1, 1}].feval[fun[x], {chebptsroot[[i]]}ᵀ];
  cheberrftn[[i]] = chebf[[i]] - fun[x];
  cheberr[[i]] = Abs[N[ExactAns] - Ichebf[[i]]]}]
```

The symbolic Polynomials generated using Chebyshev nodes and Lagrange Interpolation fountains

```
TableForm[{chebf}ᵀ, TableHeadings → {ncol, None}]
```

{n= 2}	$0.521184\,x + 1.08544$
{n= 3}	$0.173029\,x^2 + 0.532042\,x + 1.$
{n= 4}	$0.0430775\,(x + 2.85017)\left(x^2 + 1.21635\,x + 8.13608\right)$
{n= 5}	$0.00858482\left(x^2 - 0.404784\,x + 11.406\right)\left(x^2 + 5.46419\,x + 10.2126\right)$
{n= 6}	$0.00142656\,(x + 3.60602)\left(x^2 - 1.80404\,x + 15.5028\right)\left(x^2 + 4.25149\,x + 12.5394\right)$
{n= 7}	$0.00020329\left(x^2 - 3.07605\,x + 20.5294\right)\left(x^2 + 2.82747\,x + 15.2505\right)\left(x^2 + 7.29711\,x + 15.7117\right)$

Using GaussianQuadrature nodes

```
Needs["NumericalDifferentialEquationAnalysis` "]
```

The nodal locations and associated weights are generated from an internal function

```
In[203]:= Gausspts = Table[GaussianQuadratureWeights[i, -1, 1], {i, 2, 7}]
```

Out[203]=

$$\left\{\begin{pmatrix} -0.57735 & 1. \\ 0.57735 & 1. \end{pmatrix}, \begin{pmatrix} -0.774597 & 0.555556 \\ 0. & 0.888889 \\ 0.774597 & 0.555556 \end{pmatrix}, \begin{pmatrix} -0.861136 & 0.347855 \\ -0.339981 & 0.652145 \\ 0.339981 & 0.652145 \\ 0.861136 & 0.347855 \end{pmatrix}, \begin{pmatrix} -0.90618 & 0.236927 \\ -0.538469 & 0.478629 \\ 0. & 0.568889 \\ 0.538469 & 0.478629 \\ 0.90618 & 0.236927 \end{pmatrix}, \begin{pmatrix} -0.93247 & 0.171324 \\ -0.661209 & 0.360762 \\ -0.238619 & 0.467914 \\ 0.238619 & 0.467914 \\ 0.661209 & 0.360762 \\ 0.93247 & 0.171324 \end{pmatrix}, \begin{pmatrix} -0.949108 & 0.129485 \\ -0.741531 & 0.279705 \\ -0.405845 & 0.38183 \\ 0. & 0.417959 \\ 0.405845 & 0.38183 \\ 0.741531 & 0.279705 \\ 0.949108 & 0.129485 \end{pmatrix}\right\}$$

The integration is carried out by evaluating the functional at the Gaussian points dotted with an array of the weights
The error is found by taking the difference between the exact answer and the integral
The resulting errors are presented below

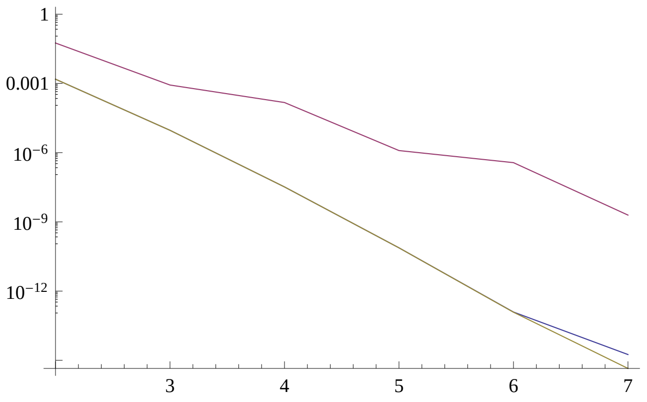
```
Gaussints = Table[feval[fun[x], Gausspts[[j, ;;]]].Gausspts[[j, ;;]]ᵀ[[2]], {j, 1, 6}];
Gausserr = Abs[ExactAns - Gaussints];
```

Results and Conclusion

The errors from the above three methods are presented below in table and graph form.
The Lagrange and Gauss methods are theoretically identical but in this application a slight difference appears in the error terms at n=6. The Chebyshev nodes do not produce the same accuracy as the Lagrange nodes as they are intended to be used with the Clenshaw Curtis rule, not the Newton Cotes rule. Implementation of the Clenshaw Curtis rule was attempted but unfortunately it was not successful.

```
TableForm[{errl, cheberr, Gausserr}ᵀ,
 TableHeadings → {ncol, { "Lagrange", "Chebvyshev", "Gauss"}}]
ListLogPlot[{{{2, 3, 4, 5, 6, 7}, errl}ᵀ, {{2, 3, 4, 5, 6, 7}, cheberr}ᵀ,
   {{2, 3, 4, 5, 6, 7}, Gausserr}ᵀ}, Joined → True]
```

	Lagrange	Chebvyshev	Gauss
{n= 2}	0.00152402	0.0563815	0.00152402
{n= 3}	9.28826×10^{-6}	0.000851212	9.28826×10^{-6}
{n= 4}	3.26461×10^{-8}	0.000148501	3.26461×10^{-8}
{n= 5}	7.47429×10^{-11}	1.22954×10^{-6}	7.47429×10^{-11}
{n= 6}	1.23457×10^{-13}	3.67947×10^{-7}	1.21236×10^{-13}
{n= 7}	1.77636×10^{-15}	1.96537×10^{-9}	4.44089×10^{-16}


1
0.001
10^{-6}
10^{-9}
10^{-12}
3
4
5
6
7