Test 1

Problem 1. This is easily done by hand, but here I will do it using Maple. The decimal points tell Maple that these are floating point numbers. Alternately, you could write them in scientific notation as done in class.

```maple
> Digits := 3;

> A1r := (1000.0+15.0)-994.0;
   A1 := (1000+15)-994;
   abs(A1r-A1)/abs(A1);
   A1 := 21
   0.238

> A2r := (1000.0-994.0)+15.0;
   A2 := (1000-994)+15;
   abs(A2r-A2)/abs(A2);
   A2r := 21.0
   A2 := 21
   0.238

> A3r := (1000.0+5.0)-984.0;
   A3 := (1000+5)-984;
   abs(A3r-A3)/abs(A3);
   A3r := 16.
   A3 := 21
   0.238
```

By mistake the table had a different problem for part 3. Given this, you could have done either one.

```maple
> A3ar := (1000.0+25.0)-984.0;
   A3a := (1000+25)-984;
   abs(A3ar-A3a)/abs(A3a);
   A3ar := 36.
   A3a := 41
   0.122
```

Problem 2. Maple thinks f[0] is part of an array. This will cause serious errors if one sets f[0] := f(0), so I call the function ff and use f[0] etc. for the divided differences.

```maple
> ff := x -> x^3;

> f[0] := ff(0);
   f[1] := ff(1);

   f[0] := 0
   f[1] := 1

> f[0,1] := (ff(1)-ff(0))/(1-0);
   f[1,1] := subs(x=1,diff(ff(x),x));
   f[0,1] := 1
```

\[ f_{1, 1} := 3 \]

> \( f[0, 1, 1] := (f[1, 1] - f[0, 1]) / (1 - 0); \)

\[ f_{0, 1, 1} := 2 \]

Part 2 of Problem 2

> \( p2 := x \rightarrow f[0] + f[0, 1] \times x + f[0, 1, 1] \times x \times (x - 1); \)

\[ p2 := x \rightarrow f_{0} + f_{0, 1} \times x + f_{0, 1, 1} \times x \times (x - 1) \]

> \( p2(x); \)

\[ x + 2 \times (x - 1) \]

Problem 3

Part 1

> restart;

> \( f := x \rightarrow \exp(-x^2); \)

\[ f := x \rightarrow e^{-x^2} \]

> with(CurveFitting);

\[ \text{[ArrayInterpolation, BSpline, BSplineCurve, Interactive, LeastSquares, PolynomialInterpolation, RationalInterpolation, Spline, ThieleInterpolation]} \]

> \( N := 2; \)

> \( x := \text{Vector}(N + 1); \)

> \( y := \text{Vector}(N + 1); \)

> for \( j \) from 1 to \( N + 1 \)

\[ x[j] := (j - 1) / N; \]

\[ y[j] := f(x[j]); \]

od;

\( p := \text{unapply(PolynomialInterpolation}(x, y, t), t) : \)

\( \text{Approx1 := evalf(int(p(t), t=0..1))}; \)

\[ N := 2 \]

\[ \text{Approx1 := 0.7471804290} \]

Part 2

> \( N := 6; \)

> \( x := \text{Vector}(N + 1); \)

> \( y := \text{Vector}(N + 1); \)

> for \( j \) from 1 to \( N + 1 \)

\[ x[j] := (j - 1) / N; \]

\[ y[j] := f(x[j]); \]

od;

\( p := \text{unapply(PolynomialInterpolation}(x, y, t), t) : \)

\( \text{Approx2 := evalf(int(p(t), t=0..1))}; \)

\[ N := 6 \]

\[ \text{Approx2 := 0.7468237564} \]

Part 3 Let's now do the whole procedure for finding the points and weights of
for Gaussian integration on $[a,b] = [0,1]$.

> a := 0; b := 1;

$a := 0$
$b := 1$

(16)

Here is the inner product

> IP := proc(f,g) evalf(Int(f*g, t=a..b)) end;

\[
IP := \text{proc}(f,g) \ \text{evalf}(\text{Int}(f*g, t = a..b)) \ \text{end proc}
\]

(17)

> f := x -> exp(-x^2);

\[f := x \rightarrow e^{-x^2}\]

(18)

> n:=3;
> p := array(0..n);
> p[0] := 1:
> j:='j'; k:='k';
> for j from 0 to n-1 do;
> vv := t*p[j]:
> for k from 0 to j do;
> vv := vv-IP(t*p[j],p[k])/IP(p[k],p[k])*p[k]:
> od:
p[j+1] := expand(vv):
> od:

(19)

> p[3];

\[
\begin{align*}
\int_{0}^{1} &- 1.500000000 \cdot t^2 + 0.600000000 \cdot t - 0.05000000002 \\
\end{align*}
\]

(20)

> sol := fsolve(p[3], t);

\[sol := 0.1127016654, 0.4999999999, 0.8872983347\]

(21)

> N := 2;
> x := Vector(N+1):
> y := Vector(N+1):
> for j from 1 to N+1 do
> x[j] := sol[j];
> y[j] := f(x[j]);
> od:
> p := unapply(PolynomialInterpolation(x, y, t), t):
> Approx3 := evalf(Int(p(t), t = 0..1));

(22)

> $N := 2$

\[Approx3 := 0.7468145838\]
Not needed, but let's compute the relative errors

\[
> \text{trueInt := IP(f(t), 1);}
\]

\[
\text{trueInt := 0.7468241328 (23)}
\]

\[
> \text{for j from 1 to 3 do}
\]
\[
\text{abs(trueInt-Approx||j)/abs(trueInt);}
\]
\[
\text{od;}
\]

\[
0.0004770817979
\]
\[
5.040008530 \times 10^{-7}
\]
\[
0.00001278614279 (24)
\]

\[
> \text{restart;}
\]

Problem 4. In the Euler-Maclaurin Formula, the periodicity condition causes terms
\[
B \frac{2}{2B}(f(2\pi i)-f(0))+...+B \frac{2}{2B}(f''(2\pi i)-(2\pi i)-f''(2\pi i))(0)
\]
to cancel, so that we get trap rule with h as the subinterval length equal to the integral we want to compute + O(h^{2N+2})
for any N such that f(x) is 2N+2 times differentiable. Of course the constant in front of h^{2N+2} may be large, so h will need to be sufficiently small for this estimate to hold.

\[
> \]

Problem 5. The 1-norm is the max over columns of the sums of absolute values of entries in a given column, i.e., 12. The infinity norm is the max over rows of the sums of absolute values of entries in a given row, i.e., 13.

\[
> \]

Problem 6.
Part 1. The Intermediate Value Theorem says that if f is a continuous real-valued function on an interval [a,b], then all the values in the interval [f(a),f(b)] are taken on by f(x) on [a,b].

\[
> f := x \rightarrow x^5-4x^3+1.2;
\]

\[
f:=x \rightarrow x^5-4x^3+1.2 (25)
\]

\[
> f(0); f(1); f(0.5);
\]

\[
1.2
\]
\[
-1.8
\]
\[
0.73125 (26)
\]

Part 2. Since f(1) and f(0.5) have different signs, it follows that the 2nd approximation is 0.75.

\[
> \]

Part 3. The error at stage n is bounded by (b-a)/2^n. So we need to have n at least 10 if we want to be sure the error is bounded by 0.001

\[
> \]

Part 4.

\[
> \text{restart;}
\]

\[
f := x \rightarrow x^5-4x^3+1.2;
\]

\[
f:=x \rightarrow x^5-4x^3+1.2 (27)
\]
> x[-1] := 0; x[0]:=1; x[1] := 0.5;
  x_1:=0
  x_0:=1
  x_1:=0.5

> for j from 1 to 9 do
  if f(x[j])*f(x[j-1]) < 0 then
    x[j+1]:= (x[j-1]+x[j])/2;
  else x[j+1]:= (x[j]+x[j-2])/2;
  fi;
  od;
> x[10];

0.6982421875

Problem 7

> Newt := x -> x - (x^5-4*x^3+1.2)/(5*x^4-12*x^2);

Newt := x -> x - \frac{x^5 - 4x^3 + 1.2}{5x^4 - 12x^2}

> x:=1.0;
  for j from 1 to 3 do x:= Newt(x); od;
  x := 1.0
  x := 0.7428571429
  x := 0.7009845978
  x := 0.6991629462

Let's compute the relative errors for bisection and Newton.

> x:='x';
  fsolve(x^5-4*x^3+1.2,x);

-2.035272840, 0.6991594161, 1.959735462

> BisectionError := abs(0.6972656250-0.6991594161)/0.6991594161;

BisectionError := 0.002708668518

> NewtonError := abs(0.6991629462-0.6991594161)/0.6991594161;

NewtonError := 0.000005049063086