Assignment 10 : Adams-Moulton and Backward Differentiation

By : Abrar Imon

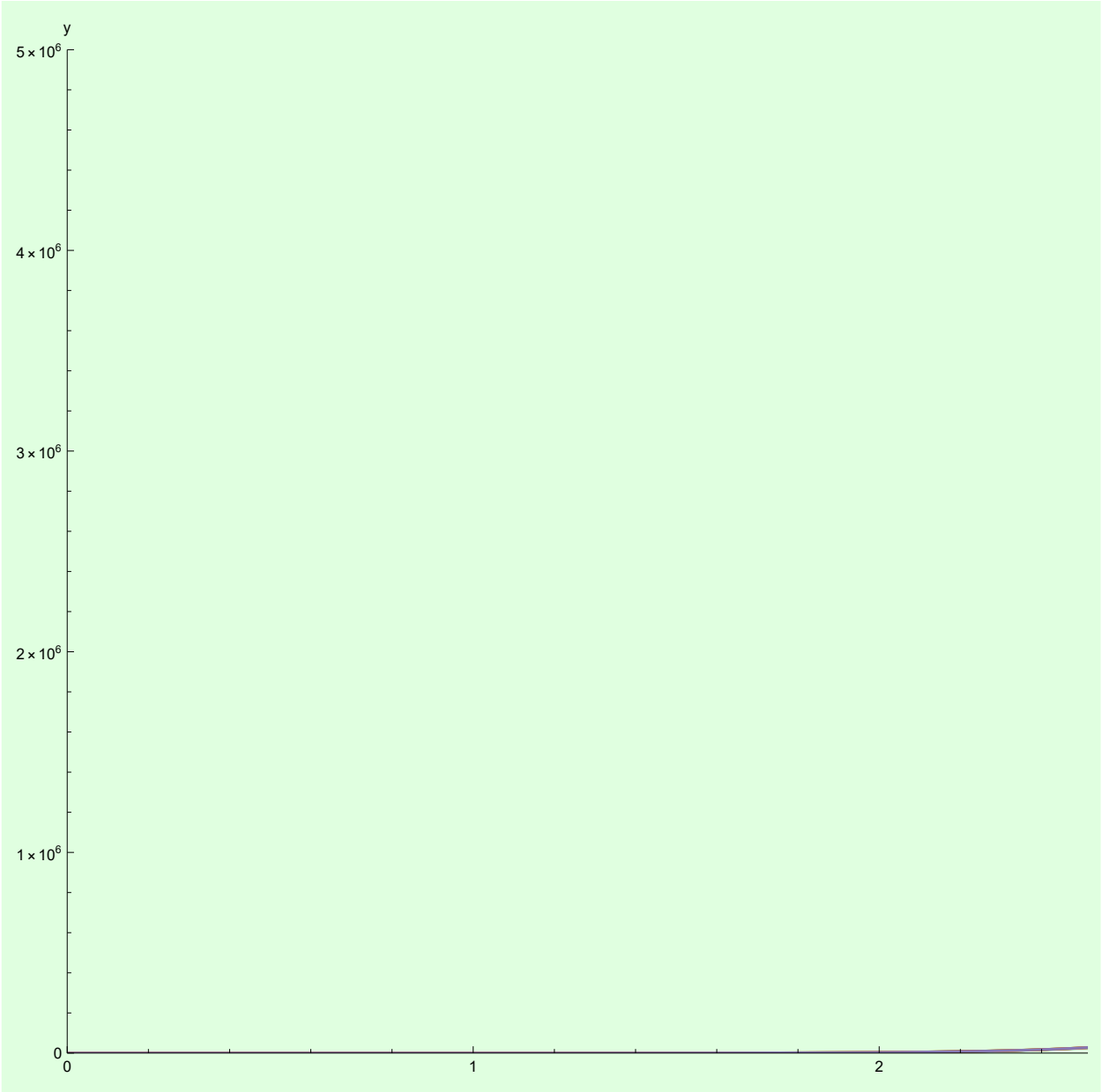
In[1]:=
```
(*This cell sets up the differential equation and the analytical solution to it.*)
yPrime[t_, y_] := 1 - t + 4 y;
y[t_] := N[t / 4 - 3 / 16 + (19 / 16) E^(4 t), 32];

(*This cell creates a function for the second order Adams-
 Moulton method. For each step of the Adams Moulton, to calculate the next "w",
it needs an estimate of that "w". To get that estimate, we used the Euler's method.*)
secondMoulton[diffEq_, initialT_, initialY_, h_, finalT_] := (
  t = initialT;
  w = initialY;
  totalSteps = (finalT - initialT) / h; For[i = 0, i < totalSteps, i++,
   w = N[w + (h / 2) * (diffEq[t + h, w + h * diffEq[t, w]] + diffEq[t, w]), 32];
   t = t + h];
  Return[N[w, 6]]
 )
```

In[6]:=
```
(*This cell makes lists with the secondMoulton function with different step sizes. The
 list will be in a form that we can easily plot, using ListLinePlot[].*)
secondMoultonSizePoint05 =
 Table[{t, secondMoulton[yPrime, 0, 1, .05, t]}, {t, 0, 4, 0.05}];
secondMoultonSizePoint025 =
 Table[{t, secondMoulton[yPrime, 0, 1, .025, t]}, {t, 0, 4, 0.025}];
secondMoultonSizePoint01 = Table[{t, secondMoulton[yPrime, 0, 1, .01, t]}, {t, 0, 4, 0.01}];
secondMoultonSizePoint001 =
 Table[{t, secondMoulton[yPrime, 0, 1, .001, t]}, {t, 0, 4, 0.001}];
exactValues = Table[{t, y[t]}, {t, 0, 4, 0.0001}];
```

```
(*This cell plots the exact solution and also plots
 our secondMoulton estimates with the different step sizes.*)
ListLinePlot[
 {secondMoultonSizePoint05, secondMoultonSizePoint025, secondMoultonSizePoint01,
  secondMoultonSizePoint001, exactValues}, PlotRange → {{0, 4}, {0, 5 * 10^6}},
 PlotLegends → {"h = 0.05", "h = 0.025", "h = 0.01", "h = 0.001", "Exact Solution"},
 ImageSize → 1000, AxesLabel → {"t", "y"}]
```

Out[9]=



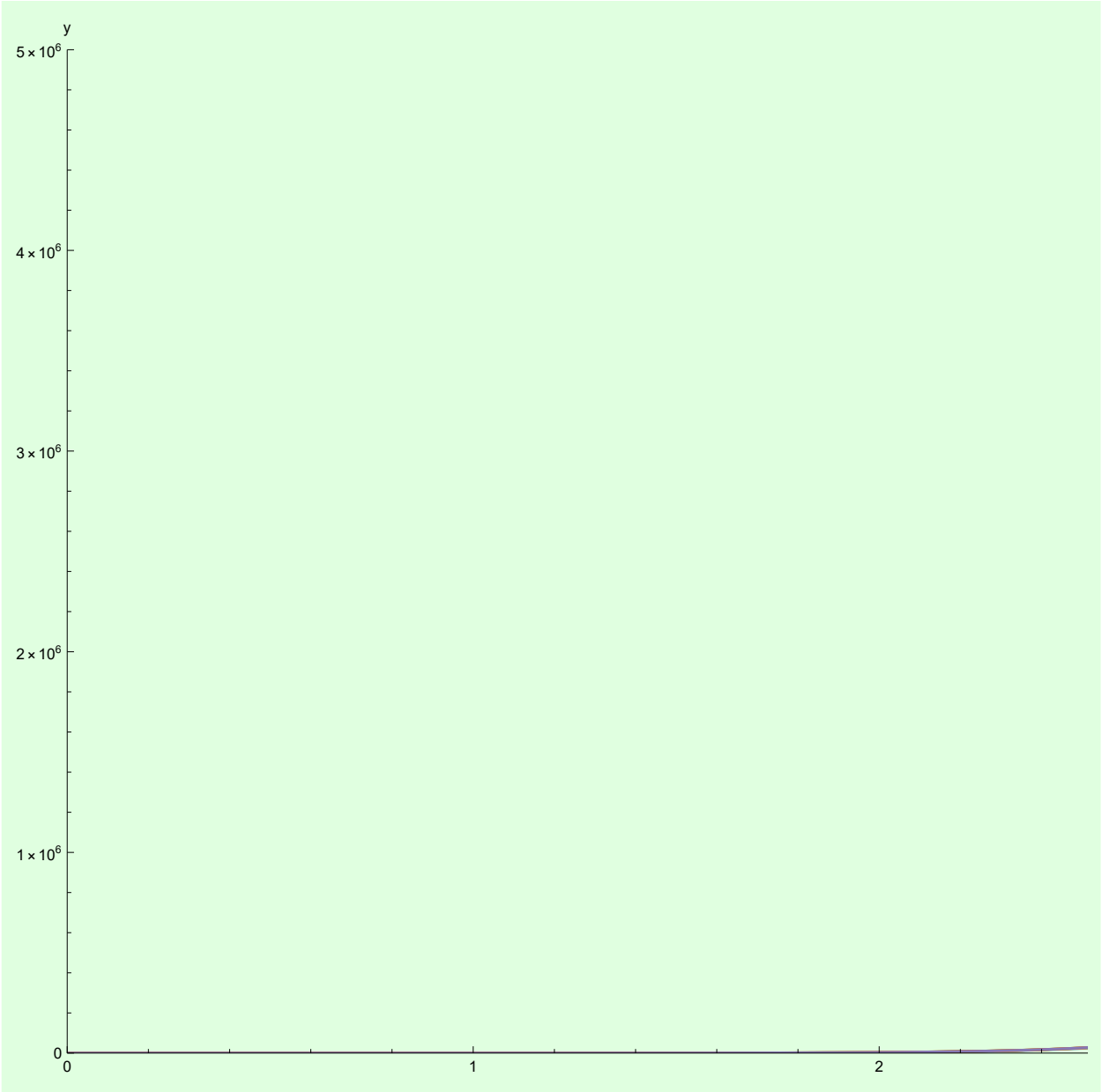
As we go farther away from t=0, the error increases for all h values. The error increases more for the higher h values.

In[31]:=
```
(*This cell creates a function for the fourth order Adams Moulton method. It uses
 a list to refer to estimates from previous steps. For the first two steps,
it uses the Euler's method. This function also uses the Euler'
 s method in each step to get an estimate for the next "w",
which we feed into the fourth order Adams-Moulton formula to get
  what will probably be an even better estimate for that same "w".*)
fourthMoulton[diffEq_, initialT_, initialY_, h_, finalT_] := (
  t = initialT;
  w = initialY;
  lis = {w};
  totalSteps = (finalT - initialT) / h; For[i = 0, i < totalSteps,
   i++, If[i > 1, w = N[w + h / 24 * (9 * diffEq[t + h, w + h * diffEq[t, w]] +
          19 * diffEq[t, w] - 5 * diffEq[t - h, lis[[2]]] + diffEq[t - 2 * h, lis[[3]]]), 32];
    lis = Join[{w}, lis], w = N[w + h * diffEq[t, w], 32];
    lis = Join[{w}, lis]];
   t = t + h];
  Return[N[w, 6]]
 )
```

In[34]:=
```
(*This cell makes lists with the fourthMoulton function with different step sizes. The
 list will be in a form that we can easily plot, using ListLinePlot[].*)
fourthMoultonSizePoint05 =
 Table[{t, fourthMoulton[yPrime, 0, 1, .05, t]}, {t, 0, 4, 0.05}];
fourthMoultonSizePoint025 =
 Table[{t, fourthMoulton[yPrime, 0, 1, .025, t]}, {t, 0, 4, 0.025}];
fourthMoultonSizePoint01 = Table[{t, fourthMoulton[yPrime, 0, 1, .01, t]}, {t, 0, 4, 0.01}];
fourthMoultonSizePoint001 =
 Table[{t, fourthMoulton[yPrime, 0, 1, .001, t]}, {t, 0, 4, 0.001}];
```

In[35]:=
```
(*This cell plots the exact solution and also plots
 our fourthMoulton estimates with the different step sizes.*)
ListLinePlot[
 {fourthMoultonSizePoint05, fourthMoultonSizePoint025, fourthMoultonSizePoint01,
  fourthMoultonSizePoint001, exactValues}, PlotRange → {{0, 4}, {0, 5 * 10^6}},
 PlotLegends → {"h = 0.05", "h = 0.025", "h = 0.01", "h = 0.001", "Exact Solution"},
 ImageSize → 1000, AxesLabel → {"t", "y"}]
```

Out[35]=



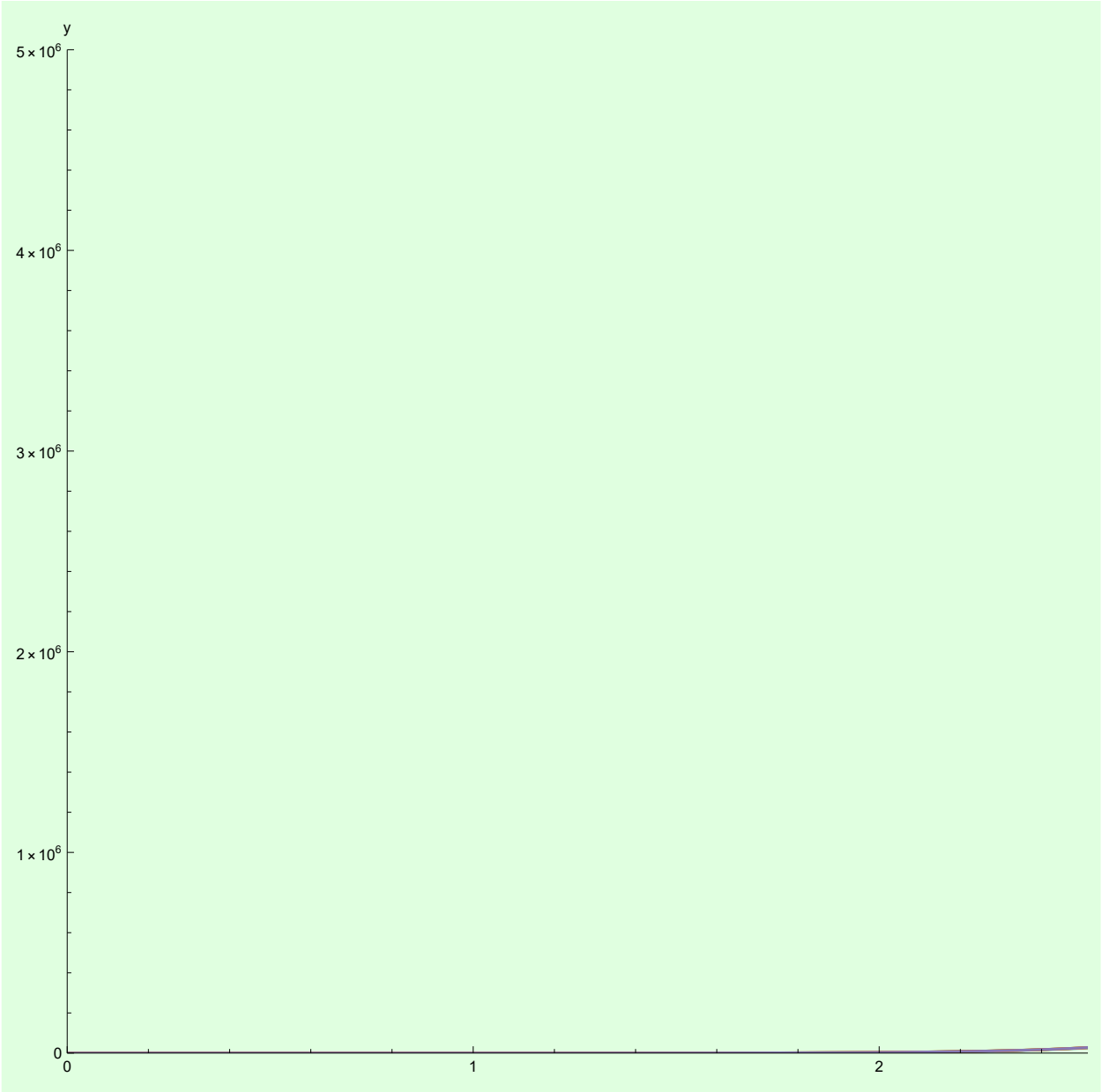
Similar to the error in the second order Adams Moulton, the error in the fourth ordered version increases as we go away from t = 0 for all h values and the error increases more for the higher h values.

In[37]:=
```
(*This cell creates a function for the second order Backward differentiation
 method. This function includes an "If" statement so that it uses the
 Euler's method for the first step. This function also uses a list called
 "lis" to refer to an estimates from previous steps. For each step,
the second order Backward Differentiation function,
requires an estimate for the next "w" before it can calculate
 the next "w". To get that estimate, we use Euler's method.*)
secondBackward[diffEq_, initialT_, initialY_, h_, finalT_] := (
  t = initialT;
  w = initialY;
  lis = {w};
  totalSteps = (finalT - initialT) / h; For[i = 0, i < totalSteps, i++,
   If[i > 0, w = N[(1 / 3) * (4 * w - lis[[2]] + 2 * h * diffEq[t + h, w + h * diffEq[t, w]]), 32];
    lis = Join[{w}, lis], w = N[w + h * diffEq[t, w], 32];
    lis = Join[{w}, lis]];
   t = t + h];
  Return[N[w, 6]]
 )
```

In[42]:=
```
(*This cell makes lists with the secondBackward function with different step
 sizes. The list will be in a form that we can easily plot, using ListLinePlot[].*)
secondBackwardSizePoint05 =
 Table[{t, secondBackward[yPrime, 0, 1, .05, t]}, {t, 0, 4, 0.05}];
secondBackwardSizePoint025 =
 Table[{t, secondBackward[yPrime, 0, 1, .025, t]}, {t, 0, 4, 0.025}];
secondBackwardSizePoint01 =
 Table[{t, secondBackward[yPrime, 0, 1, .01, t]}, {t, 0, 4, 0.01}];
secondBackwardSizePoint001 =
 Table[{t, secondBackward[yPrime, 0, 1, .001, t]}, {t, 0, 4, 0.001}];
```

In[43]:=
```
(*This cell plots the exact solution and also plots
 our secondBackward estimate with the different step sizes.*)
ListLinePlot[
 {secondBackwardSizePoint05, secondBackwardSizePoint025, secondBackwardSizePoint01,
  secondBackwardSizePoint001, exactValues}, PlotRange → {{0, 4}, {0, 5 * 10^6}},
 PlotLegends → {"h = 0.05", "h = 0.025", "h = 0.01", "h = 0.001", "Exact Solution"},
 ImageSize → 1000, AxesLabel → {"t", "y"}]
```

Out[43]=



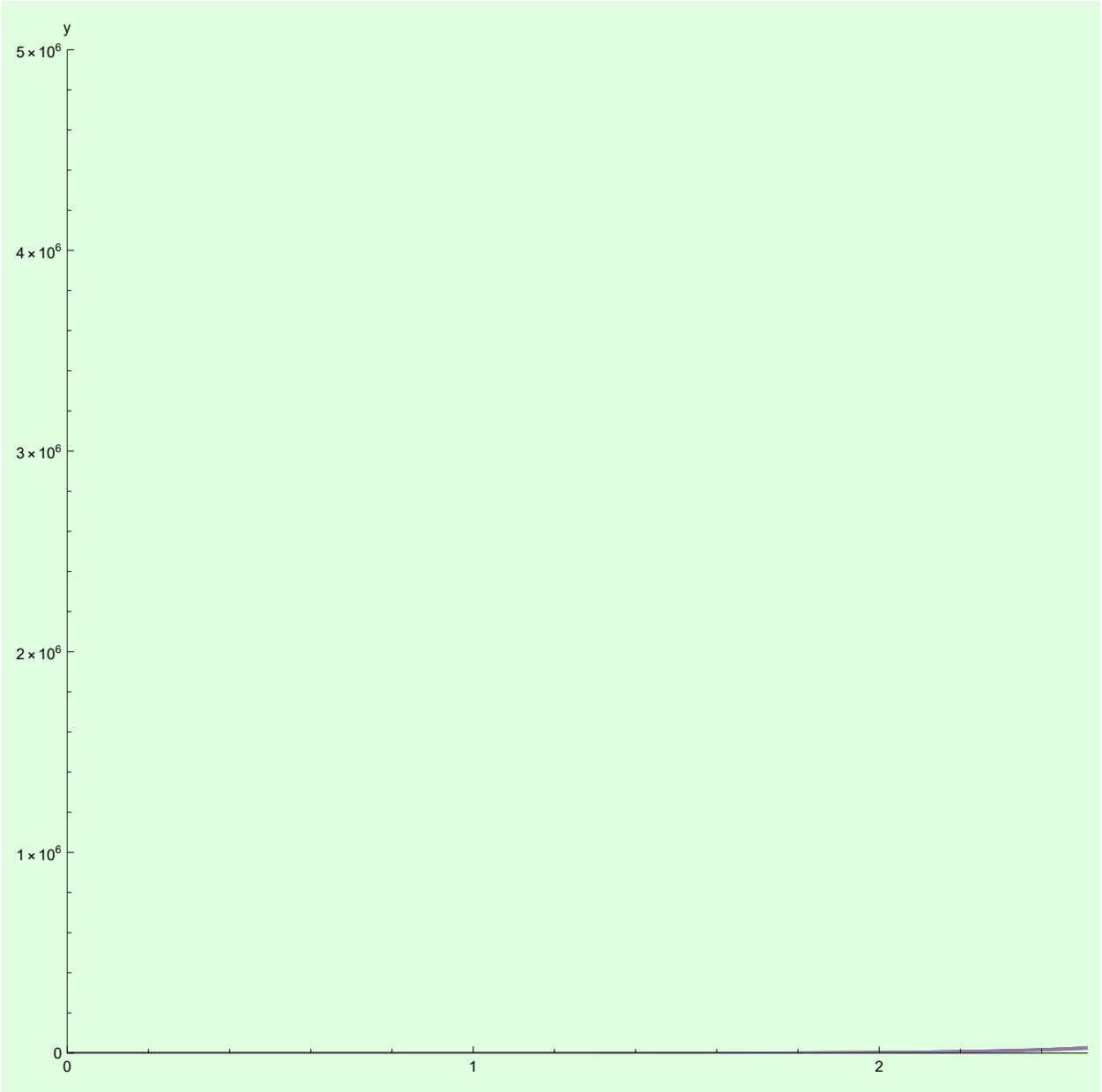
Similar to the error in the second order Adams Moulton and the fourth order Adam Moulton, the error in the second order Backward differentiation increases as we go away from t = 0 for all h values and the error increases more for the higher h values. We will see the differences between these different methods later in this assignment, when we create a table to compare their errors.

In[44]:=
```
(*This cell creates a function for the fourth order Backward differentiation
 method. It is similar to the second order Backward differentiation function,
but it performs the Euler's method for the first three steps. Also,
the formula for calculating the next "w" is changed to the
 formula for the fourth order Backward differentiation formula.*)
fourthBackward[diffEq_, initialT_, initialY_, h_, finalT_] := (
  t = initialT;
  w = initialY;
  lis = {w};
  totalSteps = (finalT - initialT) / h;
  For[i = 0, i < totalSteps, i++, If[i > 2, w = N[(1 / 25) * (48 * w - 36 * lis[[2]] +
         16 * lis[[3]] - 3 * lis[[4]] + 12 * h * diffEq[t + h, w + h * diffEq[t, w]]), 32];
    lis = Join[{w}, lis], w = N[w + h * diffEq[t, w], 32];
    lis = Join[{w}, lis]];
   t = t + h];
  Return[N[w, 6]]
 )
```

In[47]:=
```
(*This cell makes lists with the fourthBackward function with different step
 sizes. The list will be in a form that we can easily plot, using ListLinePlot[].*)
fourthBackwardSizePoint05 =
 Table[{t, fourthBackward[yPrime, 0, 1, .05, t]}, {t, 0, 4, 0.05}];
fourthBackwardSizePoint025 =
 Table[{t, fourthBackward[yPrime, 0, 1, .025, t]}, {t, 0, 4, 0.025}];
fourthBackwardSizePoint01 =
 Table[{t, fourthBackward[yPrime, 0, 1, .01, t]}, {t, 0, 4, 0.01}];
fourthBackwardSizePoint001 =
 Table[{t, fourthBackward[yPrime, 0, 1, .001, t]}, {t, 0, 4, 0.001}];
```

In[50]:=
```
(*This cell plots the exact solution and also plots
 our fourthBackward estimate with the different step sizes.*)
ListLinePlot[
 {fourthBackwardSizePoint05, fourthBackwardSizePoint025, fourthBackwardSizePoint01,
  fourthBackwardSizePoint001, exactValues}, PlotRange → {{0, 4}, {0, 5 * 10^6}},
 PlotLegends → {"h = 0.05", "h = 0.025", "h = 0.01", "h = 0.001", "Exact Solution"},
 ImageSize → 1000, AxesLabel → {"t", "y"}]
```

Out[50]=



The estimates with the fourth order Backward differentiation are least accurate for h = 0.05 and most accurate more h = 0.001. The estimates get less accurate as t increases.

In[52]:=
```
(*This cell will create a grid that displays the max
 error for each of the methods at the different h values.*)
Grid[Join[{{"Step size (h)", "Max Error Second Moulton", "Max Error Fourth Moulton",
    "Max Error Second Backward", "Max Error Fourth Backward"}},
  Table[{h, Max[Table[N[(Abs[y[t] - secondMoulton[yPrime, 0, 1, h, t]]), 6], {t, 0, 4, h}]],
    Max[Table[N[(Abs[y[t] - fourthMoulton[yPrime, 0, 1, h, t]]), 6], {t, 0, 4, h}]],
    Max[Table[N[(Abs[y[t] - secondBackward[yPrime, 0, 1, h, t]]), 6], {t, 0, 4, h}]],
    Max[Table[N[(Abs[y[t] - fourthBackward[yPrime, 0, 1, h, t]]), 6], {t, 0, 4, h}]]},
   {h, {0.05, 0.025, 0.01, 0.001}}]], Frame → All]
```

Out[52]=

Step size (h)	Max Error Second Moulton	Max Error Fourth Moulton	Max Error Second Backward	Max Error Fourth Backward
0.05	926 828.	1.37506×10^6	1.04416×10^6	2.59151×10^6
0.025	257 926.	387 445.	306 752.	836 334.
0.01	43 603.5	65 565.	54 081.	152 794.
0.001	448.872	673.551	573.153	1634.93

Every single one of these max errors occurred at t = 4; I removed the t values from the table because it seemed redundant to have 16 boxes that just says t =4. Every single one of the methods had their lowest max error at h = 0.001. For the h values tested, as the h value increased, the max error increased; so the highest max error for all of the methods occurred for h = 0.05. Out of the four methods tested, the second order Adams Moulton performed the best at all h values. At all h values, the fourth order Backward differentiation performed the worst, with the highest max errors. Keep in mind, these are a very specific version of the general methods as they all used the Euler's method in some way. It seems that the less a method had to use the Euler's method, the more accurate it was for this specific differential equation at this specific interval.