Assignment 8 : Improved Euler

By : Abrar Imon

```
In[ ]:= (*This is the same as eulerMethod from last assignment,
        except that the formula for calculating the next
         estimate is changed to the one for improved Euler's method.*)
        improvedEuler[diffEq_, initialT_, initialY_, stepSize_, finalT_] := (
          t = initialT;
          w = initialY;
          totalSteps = (finalT - initialT) / stepSize;
          For[i = 0, i < totalSteps, i++, w = N[w + stepSize * (1 / 2) *
                (diffEq[t, w] + diffEq[t + stepSize, w + stepSize * diffEq[t, w]]), 32];
           t = t + stepSize];
          Return[N[w, 6]]
         )

In[ ]:= (*This cell sets up the euler's method function from last assignment.*)
        eulerMethod[diffEq_, initialT_, initialY_, stepSize_, finalT_] := (
          t = initialT;
          w = initialY;
          totalSteps = (finalT - initialT) / stepSize;
          For[i = 0, i < totalSteps, i++, w = N[w + stepSize * diffEq[t, w], 32];
           t = t + stepSize];
          Return[N[w, 6]]
         )

In[ ]:= (*This cell sets up the differential equation for part 1.*)
        yPrime[t_, y_] := 1 - t + 4 y

In[ ]:= (*This cell sets up the solution to the above differential equation with initial
          condition y(0)=1. We will use this to get the exact solution in our table.*)
        y[t_] := N[t / 4 - 3 / 16 + (19 / 16) E^(4 t), 32]
```

In[ ]:=
```
(*Part 1.a: This cell makes a grid to show the exact values of y[t],
and the eulerMethod's and improvedEuler's estimates for y(t) for
 different values of t and different step sizes using the Euler's method.*)
Grid[Join[{{"t", "Euler; h=.05", "Euler; h=.025", "Euler; h=.01",
    "Euler; h=.001", "Improved Euler; h=.05", "Improved Euler; h=.025",
    "Improved Euler; h=.01", "Improved Euler h = .001", "Exact Value"}},
  Table[{t, eulerMethod[yPrime, 0, 1, 0.05, t], eulerMethod[yPrime, 0, 1, 0.025, t],
    eulerMethod[yPrime, 0, 1, 0.01, t], eulerMethod[yPrime, 0, 1, 0.001, t],
    improvedEuler[yPrime, 0, 1, 0.05, t], improvedEuler[yPrime, 0, 1, 0.025, t],
    improvedEuler[yPrime, 0, 1, 0.01, t], improvedEuler[yPrime, 0, 1, 0.001, t], y[t]},
   {t, {0, 0.1, 0.2, 0.5, 1, 1.5, 2}}]], Frame → All, ItemSize → 9]
```

Out[ ]=

t	Euler; h=.05	Euler; h=.025	Euler; h=.01	Euler; h=.001	Im
0	1.00000	1.00000	1.00000	1.00000	
0.1	1.5475	1.57612	1.59529	1.60763	
0.2	2.3249	2.40801	2.46446	2.50112	
0.5	7.29019	7.92641	8.37669	8.67707	
1	45.5884	53.8079	60.0371	64.3826	
1.5	282.072	361.759	426.408	473.56	
2	1745.67	2432.79	3029.33	3484.16	

Part 1. b : For a given step size value (h), the improved Euler's estimate is always closer to the exact value than the forward Euler's estimate is. According to the table above, the improved Euler with h = 0.05 (at all t-values) gives a better estimate than the forward Euler's method with h = 0.01, does.

Part 1. c : With h = 0.1 and doing 3 steps will mean we are calculating y(0.3). We want to figure out the step size we would need to get a max error that is 0.01 or less for estimation of y(0.3) with the improvedEuler. The step size corresponds to the number of steps. So we will create a new improved Euler and a new Euler's method function that takes "number of steps" as an input, to give us an output. Then, later we can take the "number of steps" and get the step size that corresponds to that "number of steps". The next two cells are the new functions.

In[ ]:=
```
eulerMethodNew[diffEq_, leftBound_, rightBound_, numberOfSteps_, T_] :=
 ( stepSize = ((rightBound - leftBound) / numberOfSteps);
  t = leftBound;
  w = 1;
  totalSteps = (T - leftBound) / stepSize;
  For[i = 0, i < totalSteps, i++, w = N[w + stepSize * diffEq[t, w], 32];
   t = t + stepSize];
  Return[N[w, 6]]
 )
```

In[•]:=
```
improvedEulerNew[diffEq_, leftBound_, rightBound_, numberOfSteps_, T_] :=
 (stepSize = ((rightBound - leftBound) / numberOfSteps);
  t = leftBound;
  w = 1;
  totalSteps = (T - leftBound) / stepSize; For[i = 0, i < totalSteps, i++, w = N[w + stepSize *
       (1 / 2) * (diffEq[t, w] + diffEq[t + stepSize, w + stepSize * diffEq[t, w]]), 32];
   t = t + stepSize];
  Return[N[w, 6]]
 )

(*This cell creates a table that will show the max error for both the forward Euler
 and the imrpoved Euler for estimation of y(3) at different number of steps.*)
Grid[Join[
  {{"Number of steps (n)", "Max Error Forward Euler's", "Max Error Improved Euler's"}},
  Table[{n, Max[Table[Abs[y[t] - eulerMethodNew[yPrime, 0, 2, n, t]], {t, 0, .3, 2 / n}]],
    Max[Table[Abs[y[t] - improvedEulerNew[yPrime, 0, 2, n, t]], {t, 0, .3, 2 / n}]]},
   {n, {10, 80, 320, 640}}]], Frame → All]
```

Out[•]=

Number of steps (n)	Max Error Forward Euler's	Max Error Improved Euler's
10	0.50533	0.12533
80	0.21576	0.00731
320	0.05774	0.00048
640	0.02922	0.00012

From this table it seems like the improved Euler passes a max error of 0.01, between n = 10 and n = 80. So, I will investigate in that range.

In[•]:=
```
Abs[y[0.3] - improvedEulerNew[yPrime, 0, 2, 79, 0.3]]
```

Out[•]= 0.0535942

In[•]:=
```
Abs[y[0.3] - improvedEulerNew[yPrime, 0, 2, 80, 0.3]]
```

Out[•]= 0.00731067

With n = 79, the error is 0.056, which is too high. So n = 80 is when the max error goes below 0.01. When n = 80, the error is 0.00731067. Since the formula for calculating step size in my functions is stepSize = ((rightBound-leftBound)/numberOfSteps), the step size corresponding to n = 80, is 1/40.