Computer Graded Assignment: Week 5

1. For the Runge-Kutta 2nd-order method, for the step size h, the local and global errors are respectively of the order of

☐ h^2 and h

☐ h^3 and h

☐ h^2 and h^3

☑ h^3 and h^2

Solution: Local error is order h^3. Global Error is $n_{\max} \times h^3 \sim h^2$

2. For the Runge-Kutta 4th-order method, for the step size h, the local and global errors are respectively of the order of

☐ h^4 and h^3

☑ h^5 and h^4

☐ h^4 and h^5

☐ h^4 and h^6

Solution: Local error is order h^5. Global Error is $n_{\max} \times h^5 \sim h^4$

3. Global Mean Error function defined in video tutorial

☐ calculates absolute deviation of the computed estimates from the true value of the solution function at t_f.

☐ calculates root mean square value of the deviation of the computed estimates from the true value of the solution function

☑ calculates mean absolute value of the deviation of the computed estimates from the true value of the solution function

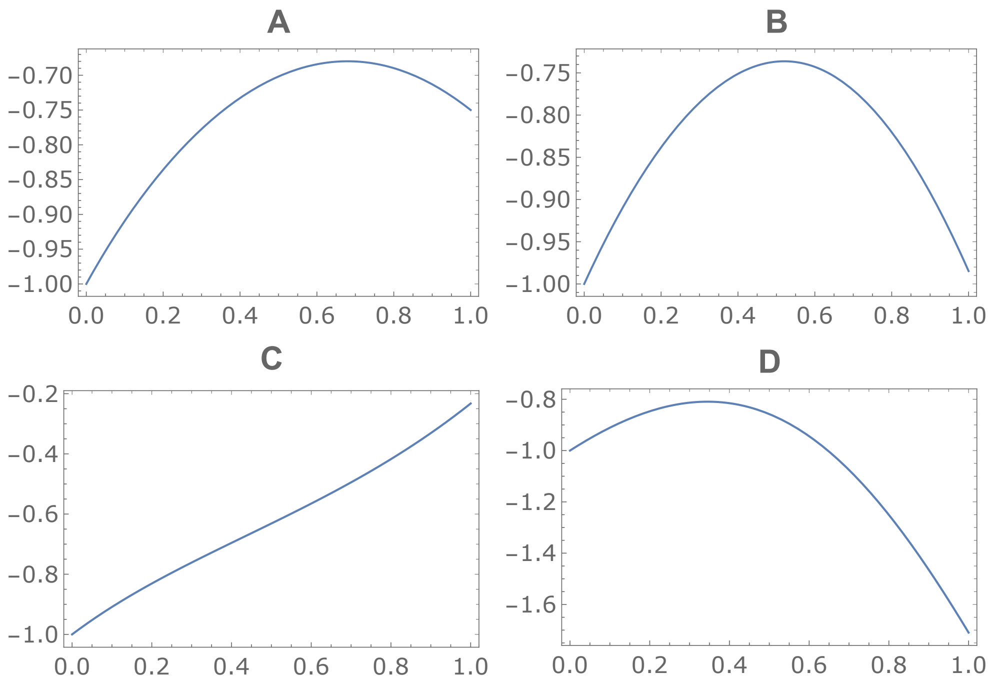
☐ calculates median absolute value of the deviation of the computed estimates from the true value of the solution function

Solution: The Global mean error we defined in the video tutorial is given by the following expression which is the mean absolute deviation of computed value from true value of the function.

$$\mathrm{err} = \frac{1}{N}\sum_{i=1}^{N}|x_i - F(t_i)|$$

4. Which of the plots below correspond to the solution of the initial value problem

$$\dot{x}(t) = x^2 + t^2 \qquad x(0) = -1$$



☐ A

☐ B

☑ C

☐ D

5. Consider Solving the initial value problem

$$\dot{x}(t) = -x\,t \qquad x(0) = 1$$

using the Improved Euler $\left(2^{\text{nd}}\right.$ order Runge–Kutta$\left.\right)$ method with step size $h = 0.4$. At $t = 2$ the value of $x(t)$ is

☐ 0.106929

☐ 0.135335

☐ 0.135782

☑ 0.155635

Solution: You can use the RK2 method we developed in the video tutorial to do this as below. Choose t_f and $n_{\max}$ so that you get $h = 0.4$.

```
In[•]:=
rk2[F_, X0_, tf_, nMax_] := Module[{h, datalist, prev, rate1, rate2, next},
  h = (tf - X0[[1]]) / nMax // N;
  For[datalist = {X0},
   Length[datalist] ≤ nMax,
   AppendTo[datalist, next],
   prev = Last[datalist];
   rate1 = F@prev;
   rate2 = F@(prev + h rate1);
   next = prev + h/2 (rate1 + rate2);
  ];
  Return[datalist];
 ]
```

```
In[283]:=
solx[t_] := e^(-t^2/2);
rateFunc[{t_, x_}] = {1, -x t};
initial = {0, 1};
tf = 2;
nMax = 5;
data2 = rk2[rateFunc, initial, tf, nMax]
```

```
Out[288]=
{{0, 1}, {0.4, 0.92}, {0.8, 0.722752}, {1.2, 0.489159}, {1.6, 0.290365}, {2., 0.155635}}
```

6. Consider Solving the initial value problem

$$\ddot{x}(t) = -x \qquad x(0) = 1 \qquad \dot{x}(0) = 0$$

using the 4th order Runge–Kutta method with step size $h = 1.0$. At $t = 10$ the value of $x(t)$ is

☑ –0.816618

☐ –0.839072

☐ 0.231445

☐ 0.283662

Solution: You can use the RK4 method we developed in the video tutorial to do this as below. Choose t_f and n_{max} so that you get $h = 1$.

In[497]:=
```
rk4[F_, X0_, tf_, nMax_] := Module[{h, datalist, prev, rate1, rate2, rate3, rate4, next},
  h = (tf - X0[[1]]) / nMax // N;
  For[datalist = {X0},
   Length[datalist] ≤ nMax,
   AppendTo[datalist, next],
   prev = Last[datalist];
   rate1 = F@prev;
   rate2 = F@(prev + h/2 rate1);
   rate3 = F@(prev + h/2 rate2);
   rate4 = F@(prev + h rate3);
   next = prev + h/6 (rate1 + 2 rate2 + 2 rate3 + rate4);
  ];
  Return[datalist];
 ]
```

In[505]:=
```
solx[t_] := Cos[t];
rateFunc[{t_, x_, v_}] = {1, v, -x};
initial = {0, 1, 0};
tf = 10;
nMax = 10;
data2 = rk4[rateFunc, initial, tf, nMax]
```

Out[510]=
```
{{0, 1, 0}, {1., 0.541667, -0.833333}, {2., -0.401042, -0.902778},
 {3., -0.969546, -0.154803}, {4., -0.654173, 0.724103},
 {5., 0.249075, 0.937367}, {6., 0.916055, 0.300178}, {7., 0.746344, -0.600783},
 {8., -0.0963825, -0.947378}, {9., -0.841689, -0.432844}, {10., -0.816618, 0.46695}}
```

7. Consider Solving the initial value problem

$$\dot{x}(t) = \text{sech}^2(t) \qquad x(2) = \frac{1}{2} \qquad t \in [-2, 2]$$

using the 4th order Runge–Kutta method with step size $h = -0.4$. The Global Mean error, as defined by err $= \frac{1}{N} \sum_{i=1}^{N} |x_i - F(t_i)|$, where N is the number of time steps in the computation, for this computation is of the order of

- ☐ 10^{-3}
- ☐ 10^{-4}
- ☑ 10^{-5}
- ☐ 10^{-6}

Solution: You can use the RK4 method and err function as we developed in the video tutorial to do this as below. Choose $n_{\max}$ so that you get $h = 0.4$.

In[655]:=

```
rk4[F_, X0_, tf_, nMax_] := Module[{h, datalist, prev, rate1, rate2, rate3, rate4, next},
  h = (tf - X0[[1]]) / nMax // N;
  For[datalist = {X0},
   Length[datalist] ≤ nMax,
   AppendTo[datalist, next],
   prev = Last[datalist];
   rate1 = F@prev;
   rate2 = F@(prev + h/2 rate1);
   rate3 = F@(prev + h/2 rate2);
   rate4 = F@(prev + h rate3);
   next = prev + h/6 (rate1 + 2 rate2 + 2 rate3 + rate4);
  ];
  Return[datalist];
 ]
```

In[763]:=

```
solx[t_] := 1/2 (1 - 2 Tanh[2] + 2 Tanh[t]);
rateFunc[{t_, x_}] = {1, Sech[t]^2};
initial = {2, 1/2};
tf = -2;
nMax = 10;
data = rk4[rateFunc, initial, tf, nMax];
Show[ListPlot[data, PlotRange → Full], Plot[solx[t], {t, -2, 2}],
 PlotLabel → "global mean error" == ScientificForm[err[data, solx]]]
```

Out[769]=