Boundary Value Ordinary Differential Equations

- Project Report

Divye Gupta

B.Sc. Student

Saint Mary's University

→ Defining Boundary Value Ordinary Differential Equations:

$$\underline{y}'(x) = A(x)\underline{y}(x) + \underline{q}(x), \quad B_a\underline{y}(a) + B_b\underline{y}(b) = \underline{\beta},$$

where $a = 0$ and $b = 1$ and

$$A(x) = \begin{pmatrix} 0 & \lambda \\ \lambda & 0 \end{pmatrix}, \quad B_a = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \quad B_b = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix},$$

$$\underline{q}(x) = \begin{pmatrix} 0 \\ \lambda \cos^2(\pi x) + \frac{2}{\lambda}\pi^2 \cos(2\pi x) \end{pmatrix}, \quad \underline{\beta} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

The exact solution to this problem is

$$\underline{y}(x) = \begin{pmatrix} \frac{e^{\lambda(x-1)} + e^{-\lambda x}}{1+e^{-\lambda}} - \cos^2(\pi x) \\ \frac{e^{\lambda(x-1)} - e^{-\lambda x}}{1+e^{-\lambda}} + \frac{\pi}{\lambda}\sin(2\pi x) \end{pmatrix}.$$

Our first step is to verify that the above expression of *y(x)* is the exact solution to the given problem. We can do that by first taking out the derivative of *y(x)* and then comparing it to the $y'(x) = A(x)y(x) + q(x)$.

Attached on the next page, is our working to solve this problem.

We have $\boxed{y'(x) = A(x)\,y(x) + q(x)}$, where

$$A(x) = \begin{pmatrix} 0 & \lambda \\ \lambda & 0 \end{pmatrix}, \quad q(x) = \begin{pmatrix} 0 \\ \lambda\cos^2(\pi x) + \frac{2}{\lambda}\pi^2\cos(2\pi x) \end{pmatrix}$$

and $$y(x) = \begin{pmatrix} \frac{e^{\lambda(x-1)} + e^{-\lambda x}}{1+e^{-\lambda}} - \cos^2(\pi x) \\ \frac{e^{\lambda(x-1)} - e^{-\lambda x}}{1+e^{-\lambda}} + \frac{\pi}{\lambda}\sin(2\pi x) \end{pmatrix}$$

Using Wolfram Alpha to find the derivative of $y(x)$,

$$y'(x) = \begin{pmatrix} \frac{\lambda e^{\lambda(x-1)} - \lambda e^{-\lambda x}}{1+e^{-\lambda}} + \pi\sin(2\pi x) \\ \frac{\lambda e^{\lambda(x-1)} - \lambda e^{-\lambda x}}{1+e^{-\lambda}} + \frac{2\pi^2\cos(2\pi x)}{\lambda} \end{pmatrix} = \underline{\underline{L\cdot H\cdot S}} \quad \text{——— (1)}$$

$\underline{\underline{R\cdot H\cdot S}} = A(x)\,y(x) + q(x)$

$$\Rightarrow \begin{pmatrix} 0 & \lambda \\ \lambda & 0 \end{pmatrix} \begin{pmatrix} \frac{e^{\lambda(x-1)} + e^{-\lambda x}}{1+e^{-\lambda}} - \cos^2(\pi x) \\ \frac{e^{\lambda(x-1)} - e^{-\lambda x}}{1+e^{-\lambda}} + \frac{\pi}{\lambda}\sin(2\pi x) \end{pmatrix} + \begin{pmatrix} 0 \\ \lambda\cos^2(\pi x) + \frac{2}{\lambda}\pi^2\cos(2\pi x) \end{pmatrix}$$

Solving this by Wolfram Alpha, we have:

$$\begin{pmatrix} \frac{\lambda e^{\lambda(x-1)} + \lambda e^{-\lambda x}}{1+e^{-\lambda}} + \pi\sin(2\pi x) \\ \frac{\lambda e^{\lambda(x-1)} + \lambda e^{-\lambda x}}{1+e^{-\lambda}} - \cancel{\lambda\cos^2(\pi x)} + \cancel{\lambda\cos^2(\pi x)} + \frac{2\pi^2\cos(2\pi x)}{\lambda} \end{pmatrix}$$

$$\Rightarrow \begin{pmatrix} \frac{\lambda e^{\lambda(x-1)} - \lambda e^{-\lambda x}}{1+e^{-\lambda}} + \pi\sin(2\pi x) \\ \frac{\lambda e^{\lambda(x-1)} - \lambda e^{-\lambda x}}{1+e^{-\lambda}} + \frac{2\pi^2\cos(2\pi x)}{\lambda} \end{pmatrix}$$, which is equal to (1), and hence equal to $\underline{\underline{L\cdot H\cdot S}}$.

Therefore, $\underline{\underline{L\cdot H\cdot S}} = \underline{\underline{R\cdot H\cdot S}}$

Hence proved.

→ Uniform: Midpoint Scheme and Trapezoidal Scheme:

Table 1: Midpoint scheme with a uniform mesh.

λ	N	y_1 Max Err	y_2 Max Err	y_1 Order	y_2 Order	Time
1	10	0.0147390	0.0565223	2.0436653	1.9510322	0.003961
1	20	0.0035749	0.0146184	2.0109406	2.0032796	0.017598
1	40	0.0008870	0.0036463	2.0027367	2.0008192	0.047793
1	80	0.0002213	0.0009111	0	0	0.158077
20	10	0.1535066	0.1427874	2.0169587	2.0847515	0.003735
20	20	0.0379282	0.0336602	2.1269349	2.1427785	0.018214
20	40	0.0086834	0.0076221	2.0289074	2.0327042	0.046392
20	80	0.0021278	0.0018628	0	0	0.19297
50	10	0.4668580	0.4486886	1.2253177	1.2076202	0.007172
50	20	0.1996767	0.1942742	1.8116351	1.809192	0.017361
50	40	0.0568815	0.0554363	2.197869	2.2033291	0.052247
50	80	0.0123979	0.0120372	0	0	0.157758

By observing the values in the table and the seeing the curves of the graphs *(below)* we found out the following:

- When Lambda = 1
  - The error decreases as the number of subintervals increases.
  - The maximum error when lambda is equal to 1 occurs when the number of subintervals, N = 10.
  - The order of the error remains close to 2. That means it is proportional to $O(h^2)$.
- Lambda = 20
  - The error $e(y_1)$ increases as the value of lambda increases from 1 to 20. However, when we increase the values of N for lambda = 20, the error decreases.
  - However, the error $e(y_2)$ has no effect on the increase of lambda.
- Lambda = 50
  - The maximum error is when N = 10.
  - The order of the error comes close to $O(h^{1.2})$ when we increase the N from 10 to 20, however it gains back its original order i.e. $O(h^2)$ when the value of N is further increased.
- The maximum time taken to compute the solution is when lambda = 20 and N = 80.

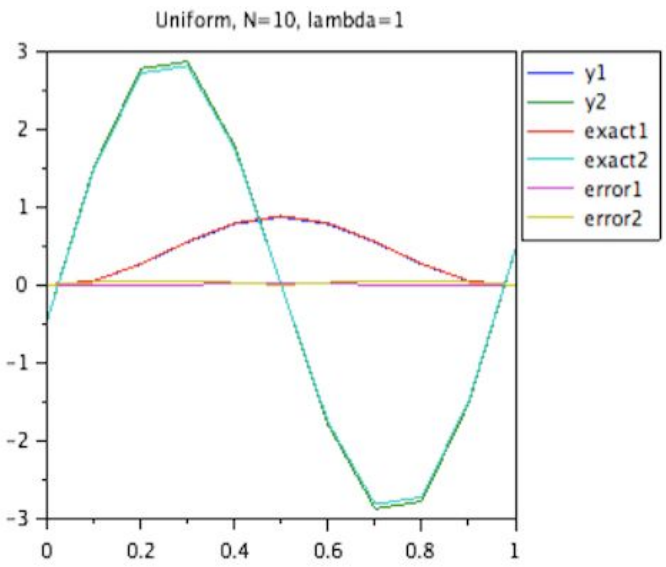
Comparison to Simple One-Step Schemes for Linear BVODEs

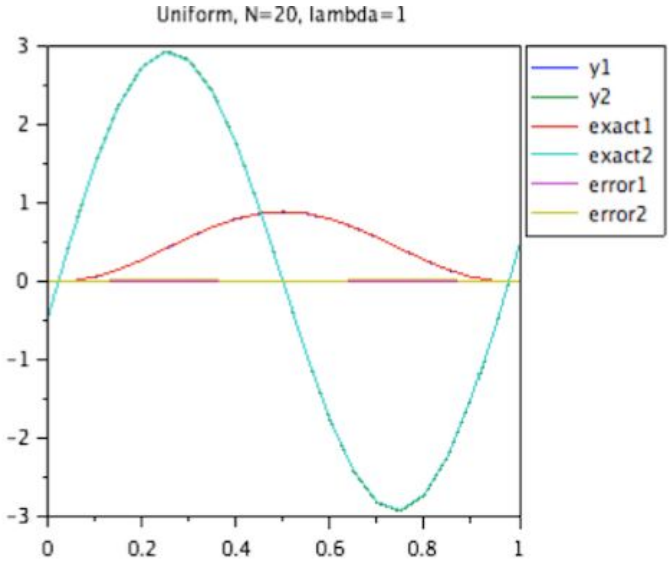
N	Notes: $e(y_1)$	Notes: $e(y_2)$	Computed: $e(y_1)$	Computed: $e(y_2)$
10	.47	.45	.466 = .47	.448 = .45

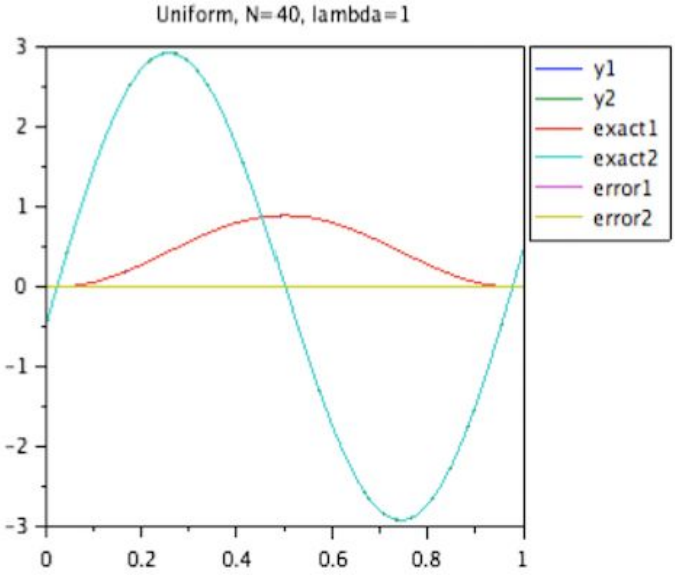
20	.20	.19	.199 = .20	.194 = .19
40	.57-1	.55-1	.0568 = .57-1	.0554 = .55-1
80	.1-1	.12-1	.0123 = .1-1	.0120 = .12-1

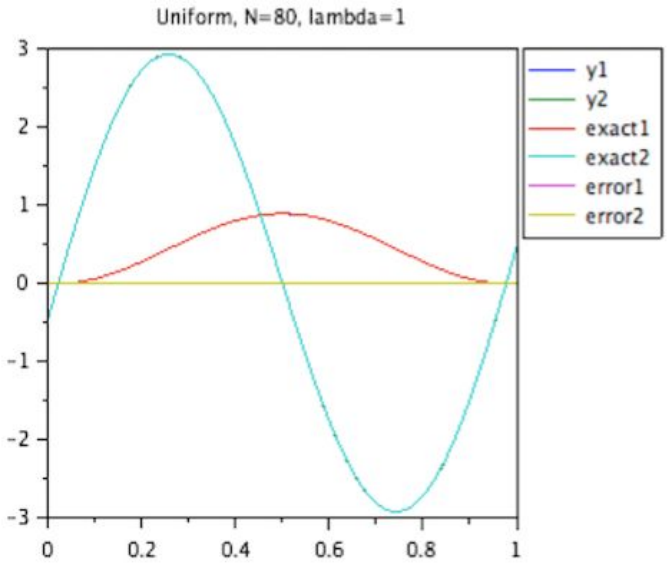
Our results matches with the solutions given in the "Simple One-Step Schemes for Linear BVODEs" very well.

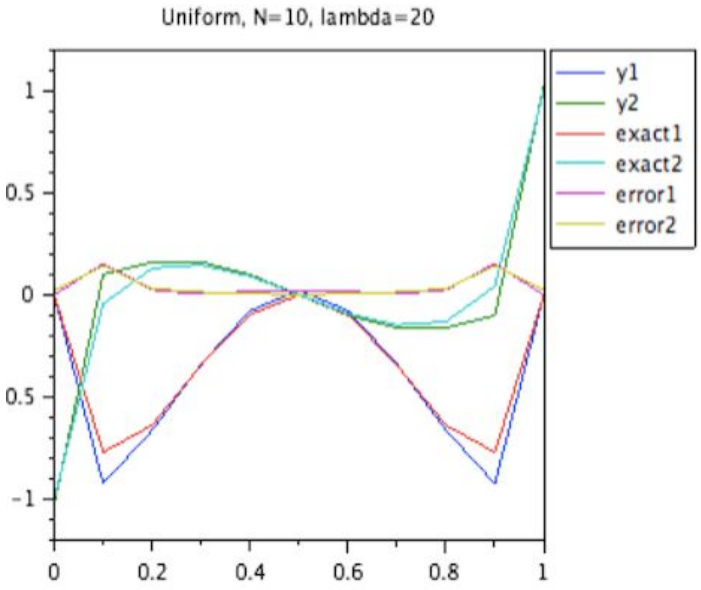
Following are the graphs for each computation we did for the *midpoint scheme.*

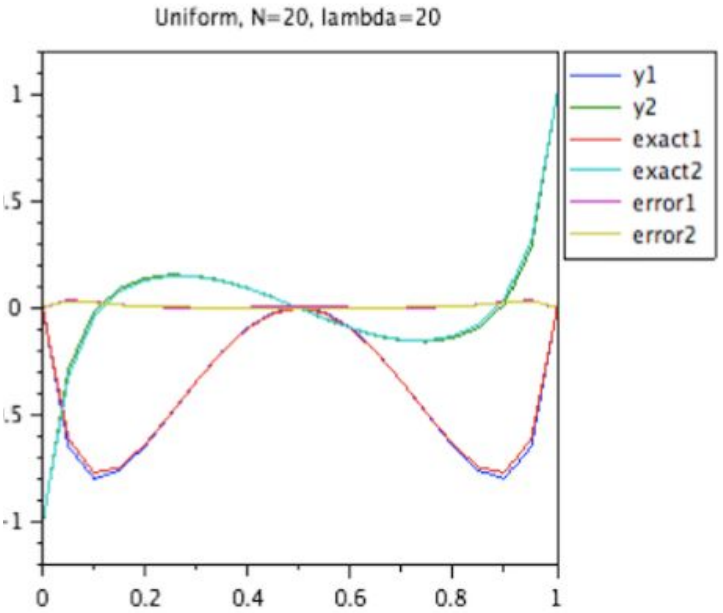


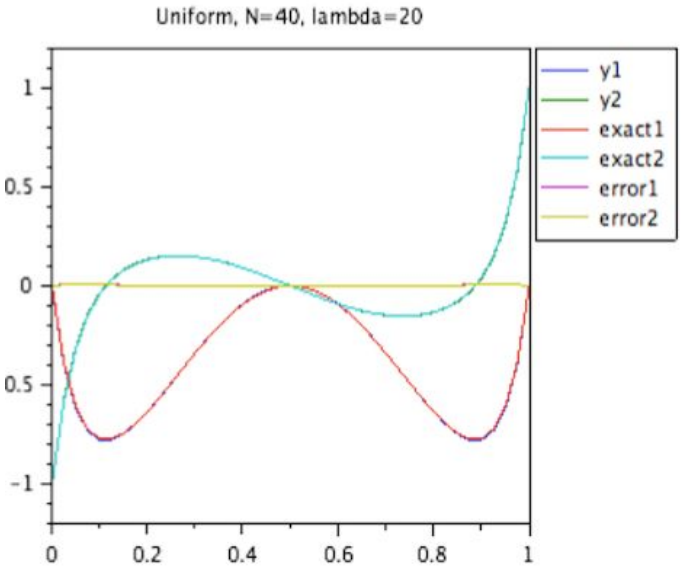


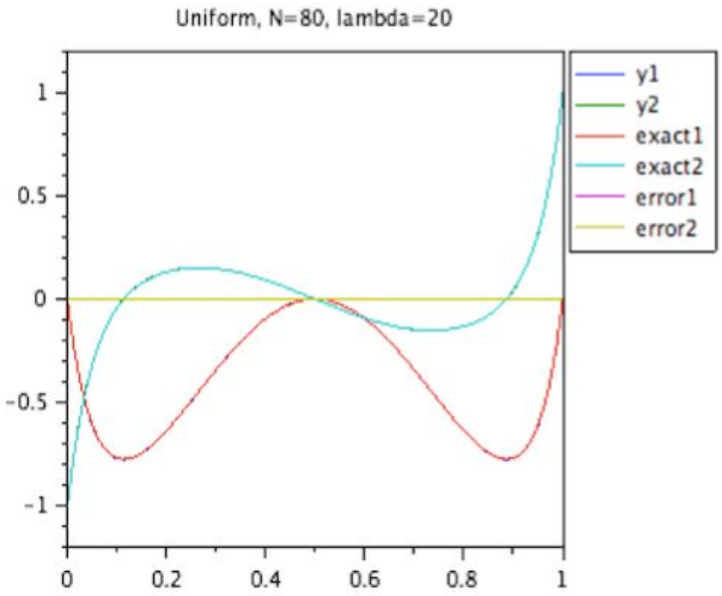


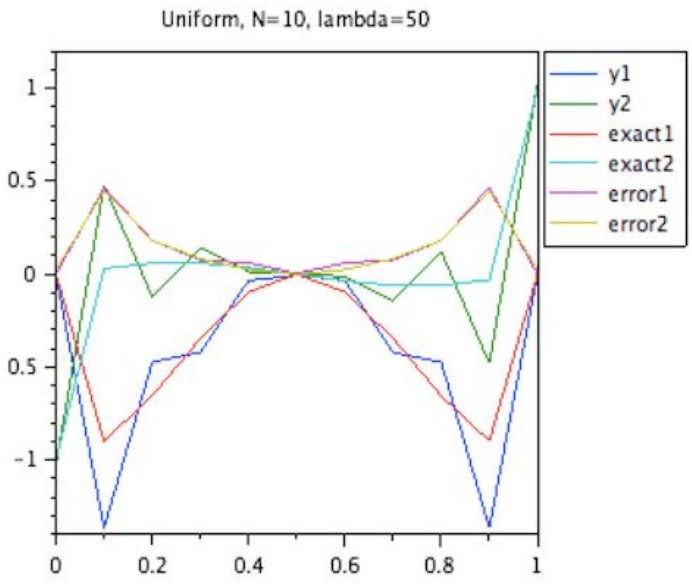


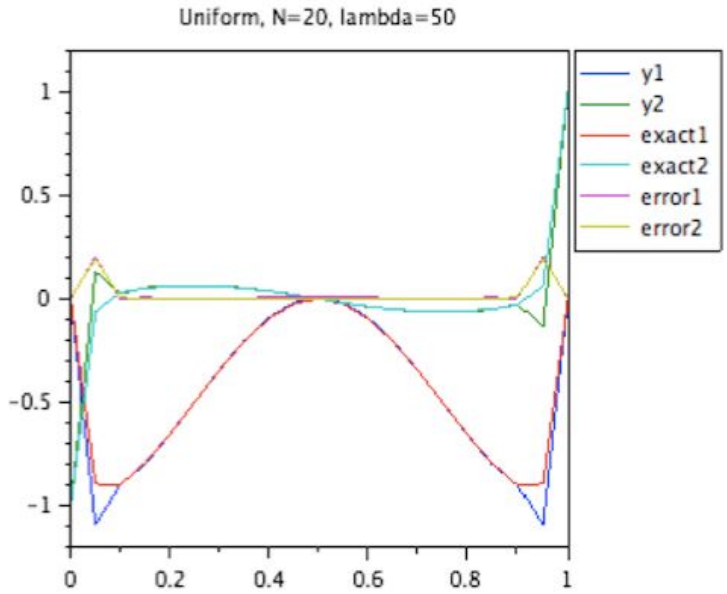

Uniform, N=10, lambda=20
y1
y2
exact1
exact2
error1
error2
1
0.5
0
0.5
-1
0
0.2
0.4
0.6
0.8
1

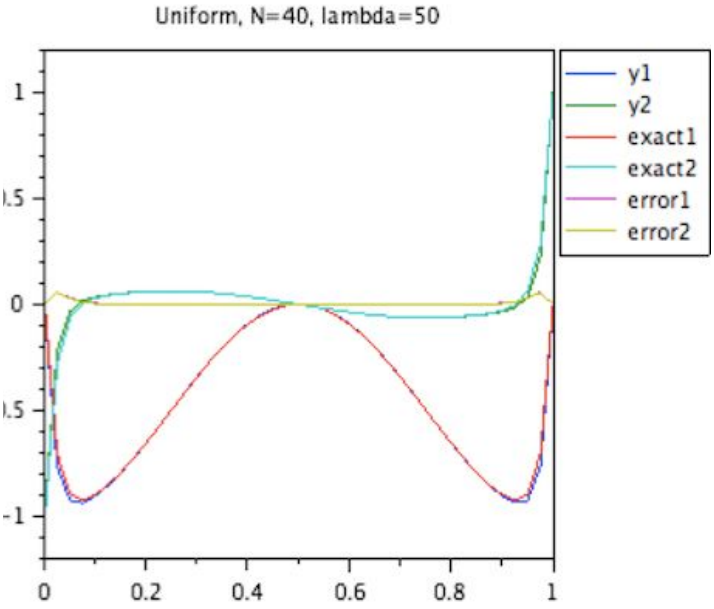

Uniform, N=20, lambda=20
y1
y2
exact1
exact2
error1
error2
1
0
-1
0
0.2
0.4
0.6
0.8
1

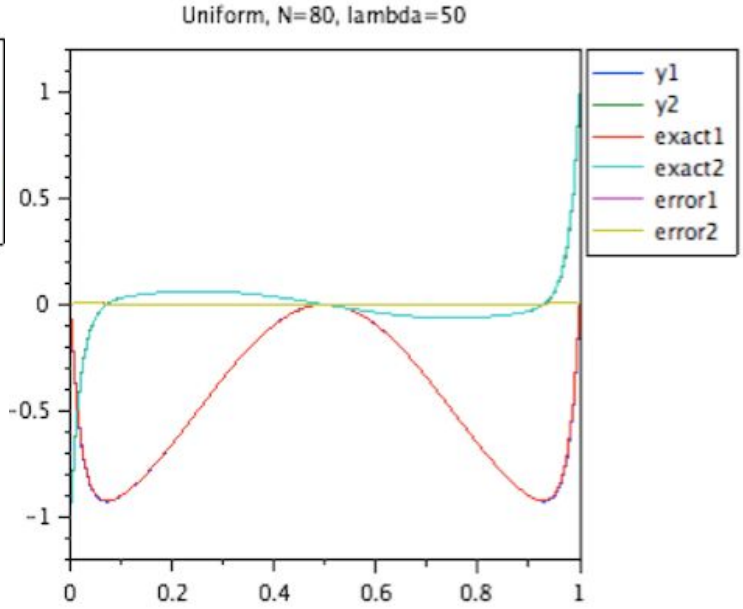

Uniform, N=40, lambda=20
y1
y2
exact1
exact2
error1
error2
1
0.5
0
0.5
-1
0
0.2
0.4
0.6
0.8
1


Uniform, N=80, lambda=20
y1
y2
exact1
exact2
error1
error2
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1



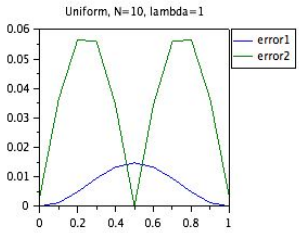


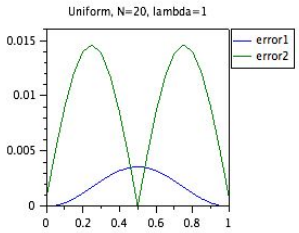


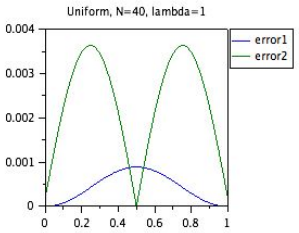


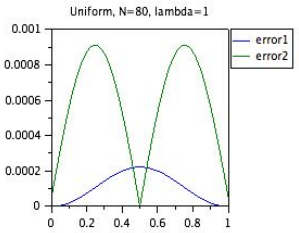
We can see the distortion of the graph when N = 10 and lambda = 50. However, the solution improves when the number of subintervals are further increased to 80.

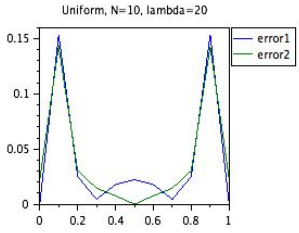
Following are the graphs for all the errors that we computed during uniform-midpoint scheme. Note that the error reaches 0 on halfway to our interval [0,1].

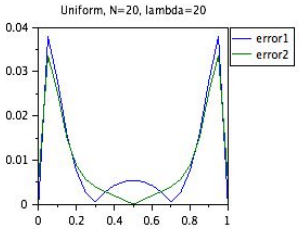


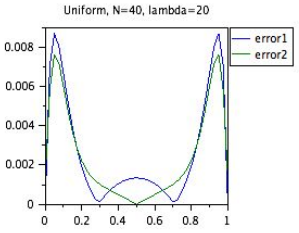


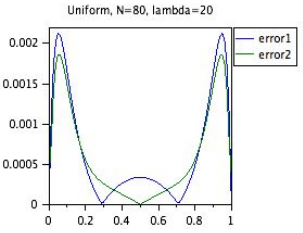


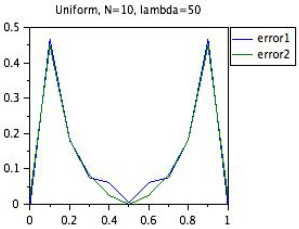


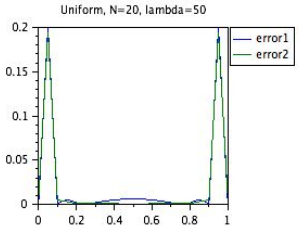


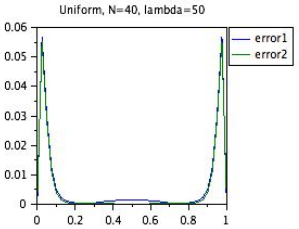


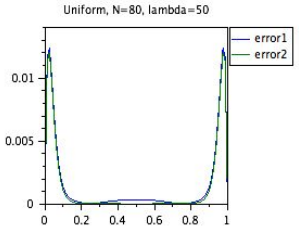












1	10	0.0601956	0.0887246	1.9907122	1.9564479	0.02637
1	20	0.0151461	0.0228610	1.9977346	2.0028055	0.01951
1	40	0.0037925	0.0057041	1.9994371	1.9998236	0.062077
1	80	0.0009485	0.0014262	0	0	0.222688
20	10	0.1328192	0.1379037	1.9622994	1.9730721	0.008785
20	20	0.0340840	0.0351255	2.1331193	2.129374	0.021751
20	40	0.0077699	0.0080282	2.030316	2.0294278	0.062671
20	80	0.0019021	0.0019665	0	0	0.215205
50	10	0.4350377	0.4357242	1.172103	1.1722908	0.006209
50	20	0.1930584	0.1933379	1.7929959	1.7937549	0.01703
50	40	0.0557113	0.0557626	2.2002616	2.2000675	0.064372
50	80	0.0121227	0.0121355	0	0	0.213765

Table 2: Trapezoidal scheme with a uniform mesh.

- Lambda = 1
  - Trapezoidal scheme is quite similar to the midpoint scheme.
  - The order of error remains to be around 2.
- Lambda = 50
  - Maximum error is at N = 10.

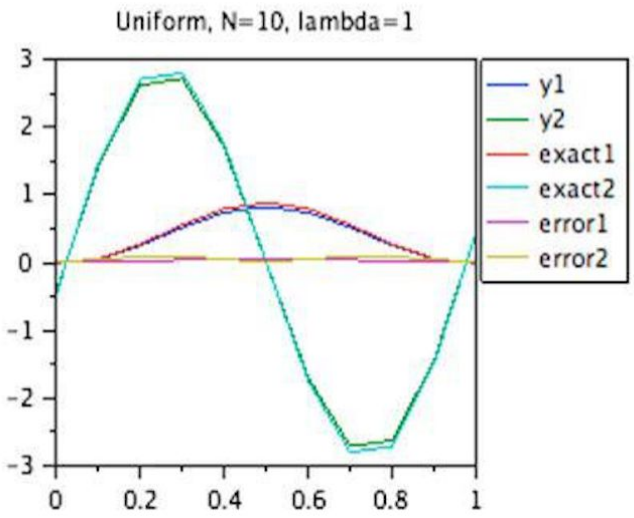
- Eventually, the error is reduced when we have increasing values of N.
    - The maximum time taken to compute the solution is when lambda = 20 and N = 80.

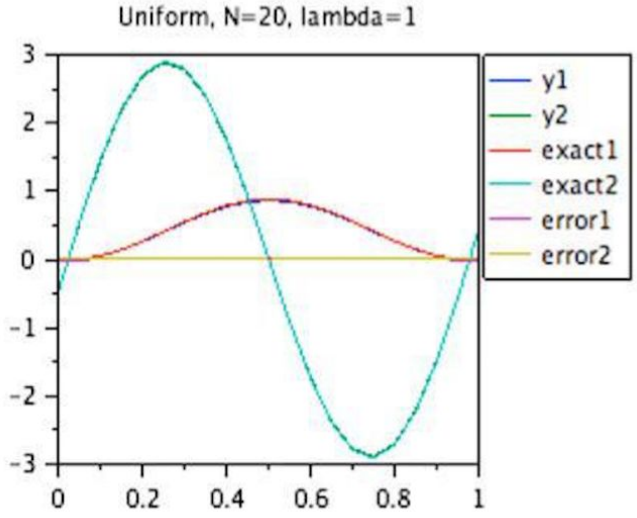
Comparison to Simple One-Step Schemes for Linear BVODEs

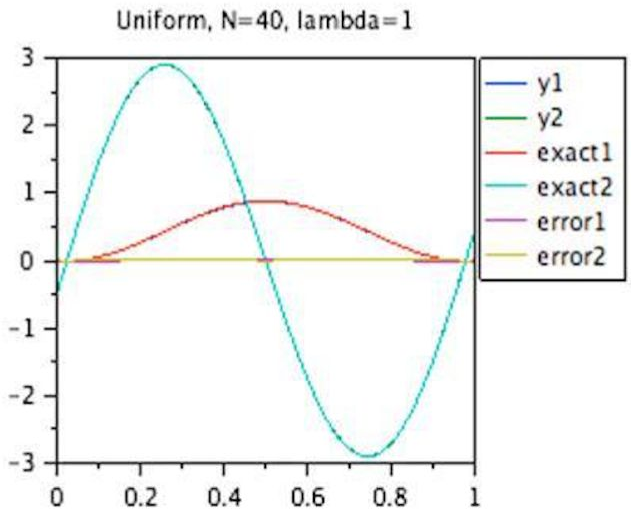
N	Notes: $e(y_1)$	Notes: $e(y_2)$	Computed: $e(y_1)$	Computed: $e(y_2)$
10	.44	.44	.435 = .44	.435 =.44
20	.20	.19	.193 = .19	.193 = .19
40	.56-1	.56-1	0.0557 = .56-1	.00557 = .56-1
80	.12-1	.12-1	.0121 = .12-1	.0121 = .12-1

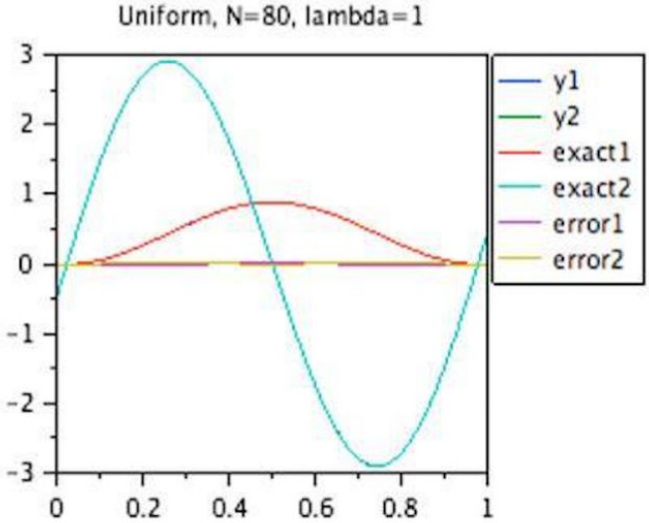
Our results matches with the solutions given in the "Simple One-Step Schemes for Linear BVODEs" very well.

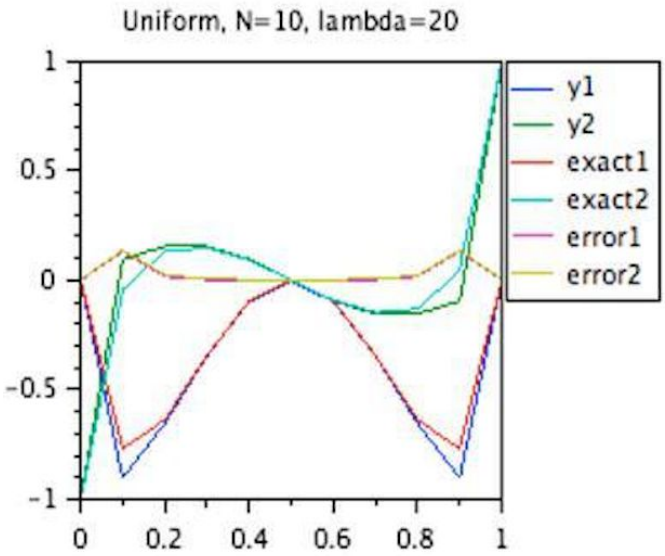
Following are the graphs all the combinations of lambda and N for the *trapezoidal scheme.*

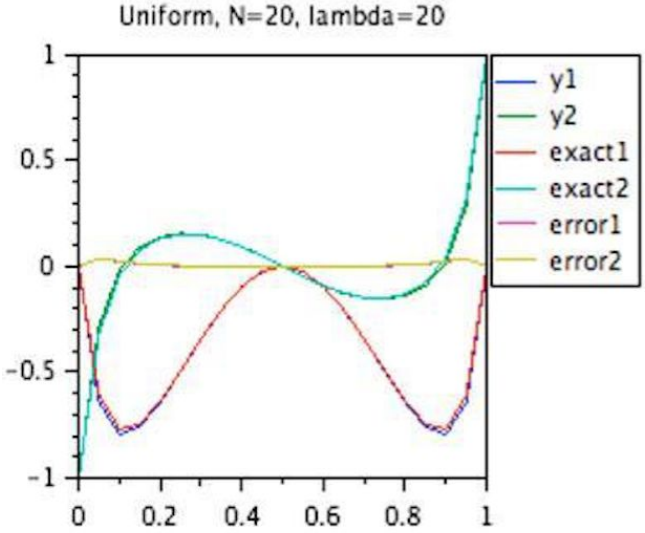


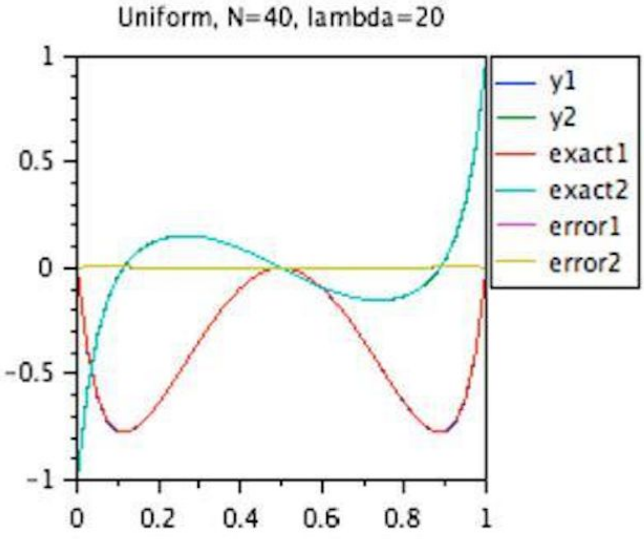


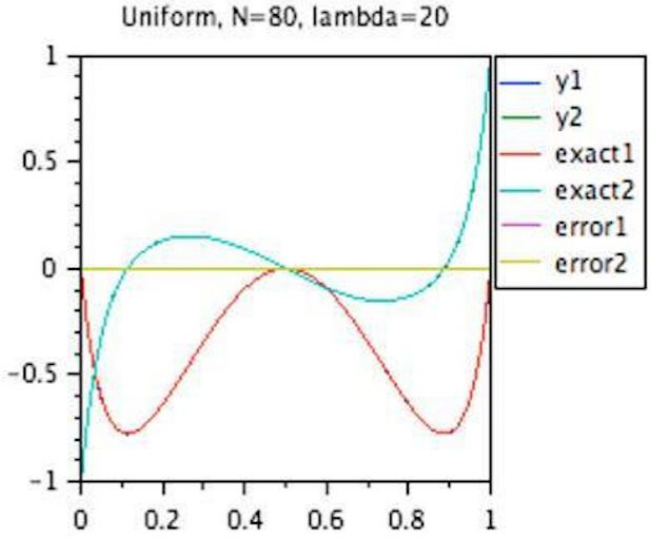

Uniform, N=40, lambda=1
y1
y2
exact1
exact2
error1
error2
3
2
1
0
-1
-2
-3
0
0.2
0.4
0.6
0.8
1

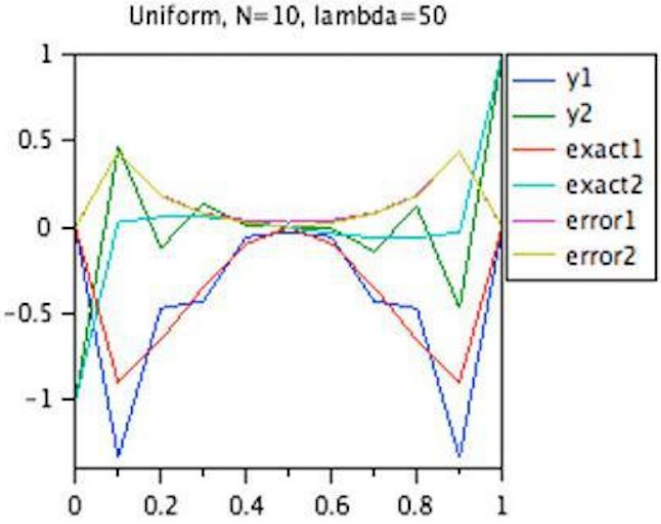

Uniform, N=80, lambda=1
y1
y2
exact1
exact2
error1
error2
3
2
1
0
-1
-2
-3
0
0.2
0.4
0.6
0.8
1

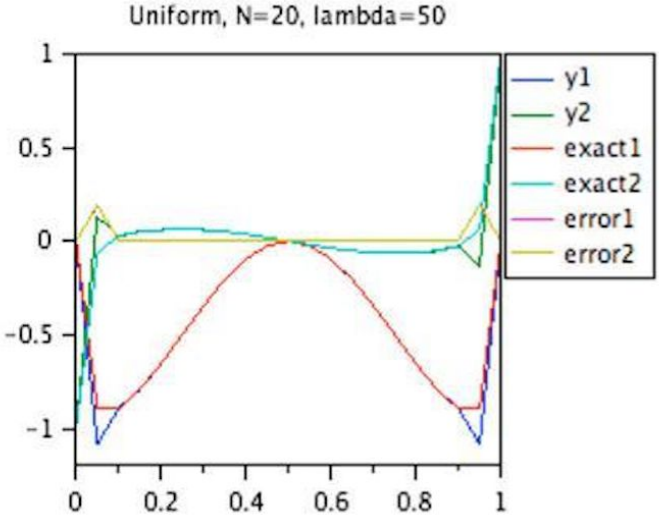

Uniform, N=10, lambda=20
y1
y2
exact1
exact2
error1
error2
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1

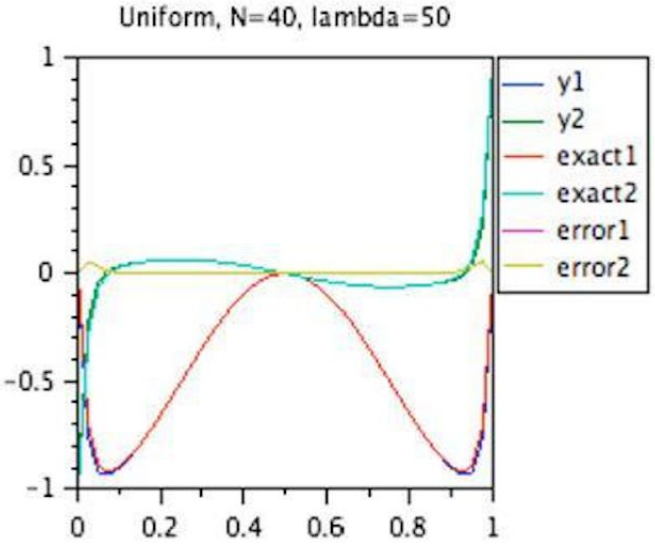

Uniform, N=20, lambda=20
y1
y2
exact1
exact2
error1
error2
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1

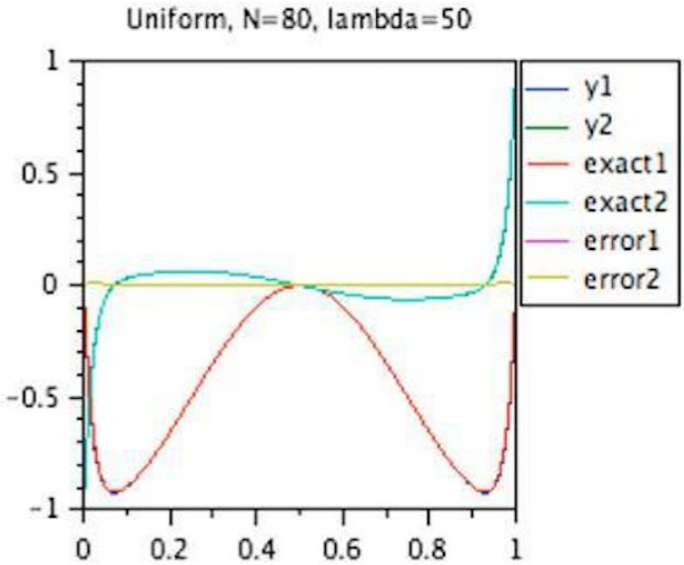

Uniform, N=40, lambda=20
y1
y2
exact1
exact2
error1
error2
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1

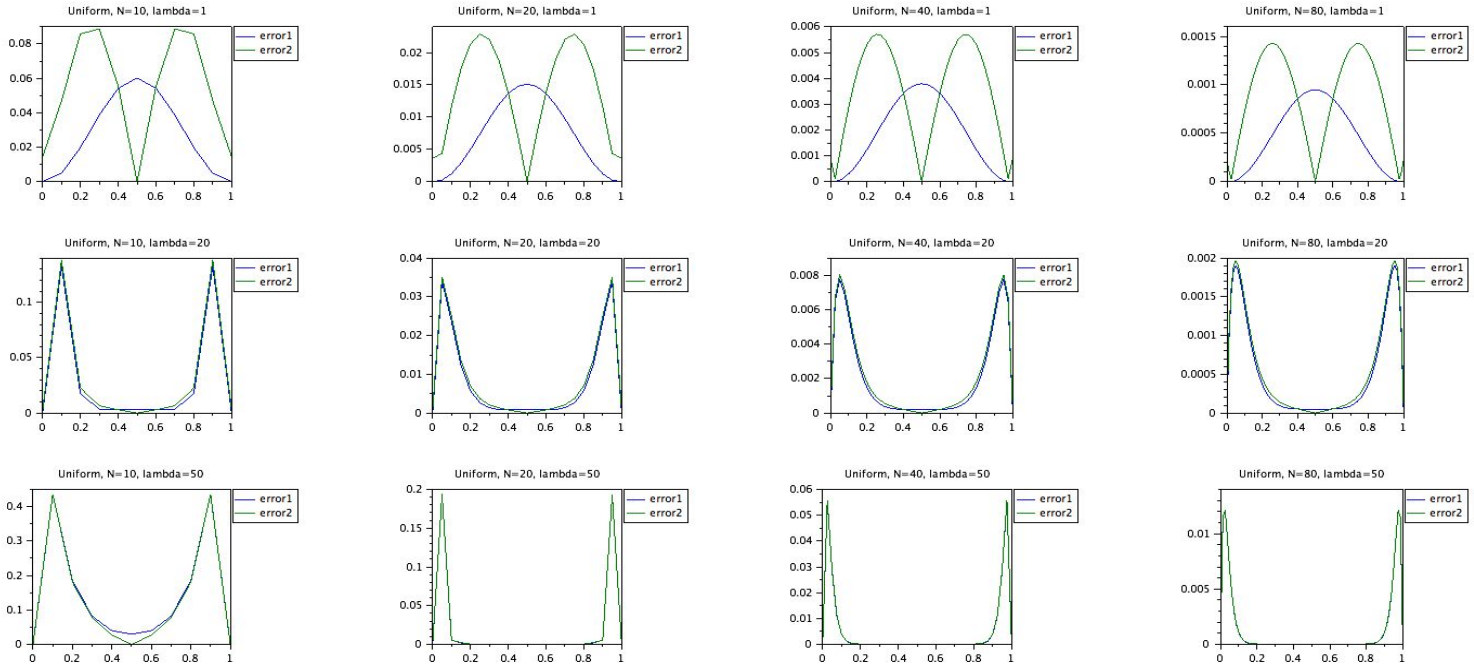

Uniform, N=80, lambda=20
y1
y2
exact1
exact2
error1
error2
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1


Uniform, N=10, lambda=50
y1
y2
exact1
exact2
error1
error2
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1


Uniform, N=20, lambda=50
y1
y2
exact1
exact2
error1
error2
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1







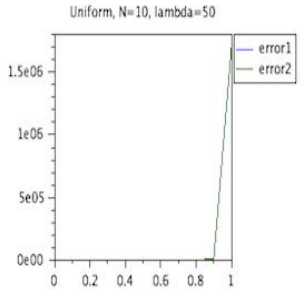
(Above) Figure showing plots of all the errors computed for uniform-trapezoidal scheme. The plots look quite similar to the midpoint scheme.

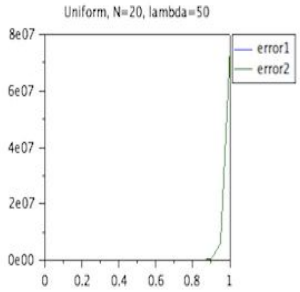
→ Uniform: Simple Shooting Scheme:

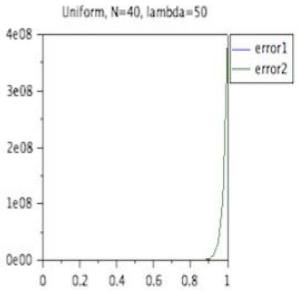
Table 3: Simple shooting scheme with a uniform mesh.

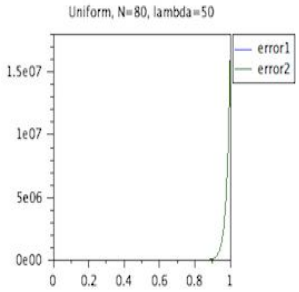
λ	N	y_1 Max Err	y_2 Max Err	y_1 Order	y_2 Order	Time
1	10	2.846D-08	0.0000001	-0.0035596	-0.0014660	0.011018
1	20	2.853D-08	0.0000001	-0.1063131	-0.0035121	0.010225
1	40	3.071D-08	0.0000001	-0.0471796	-0.0041893	0.01225
1	80	3.173D-08	0.0000001	0	0	0.015987
20	10	0.0000007	0.0000007	-0.0395339	-0.0395067	0.023914
20	20	0.0000007	0.0000007	-0.0125464	-0.0125465	0.019669
20	40	0.0000007	0.0000007	-1.6224155	-1.6224331	0.019371
20	80	0.0000022	0.0000022	0	0	0.020271
50	10	1712709.1	1712709.1	-5.4010035	-5.4010035	0.039773
50	20	72368184	72368184	-2.3763005	-2.3763005	0.03487
50	40	3.757D+08	3.757D+08	4.5124108	4.5124108	0.03228
50	80	16463206	16463206	0	0	0.033777

- Simple shooting does not use different values of N, and hence the solution is computed at N =1.
- The errors are still calculated at multiple points of the solution and hence we have varying answers for the error.
- For lambda = 1, 20
  - The error in simple shooting is quite low as compared to that of the above two schemes.
  - The order remains negative because the error remains same/increases slightly when the value of N increases.
- For lambda = 50
  - Because of the exponential growth of the function, the error at lamda = 50 reaches up to 375,700,000 and is too big to be controlled by newton iteration.



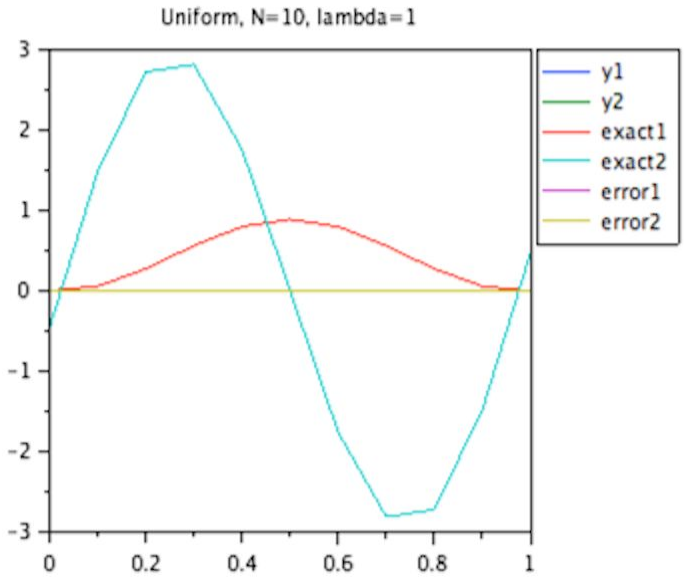


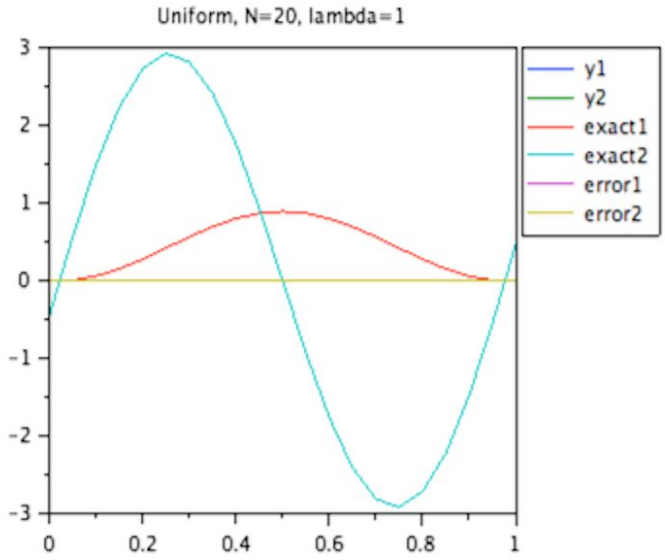


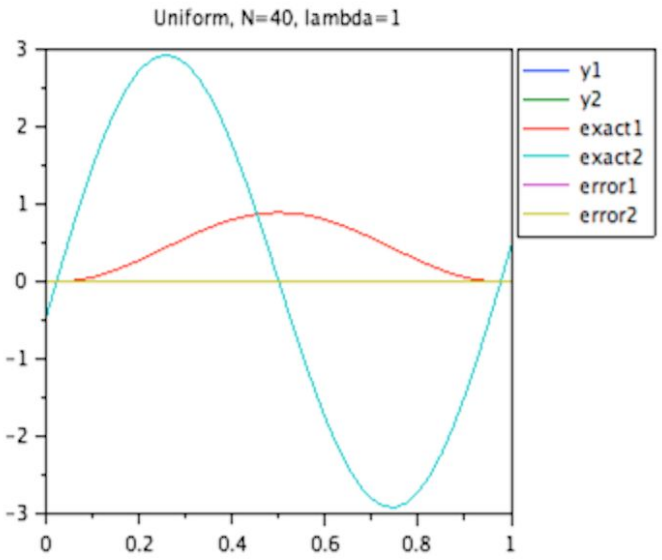


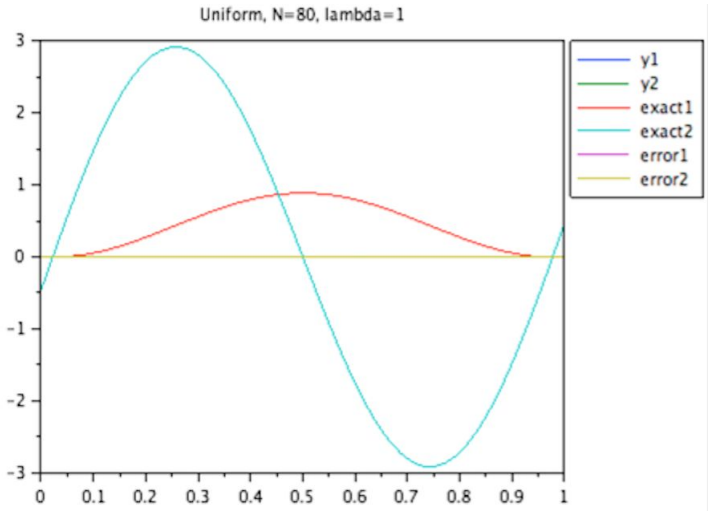
Errors increasing substantially when x goes past 0.85. More detailed images below.

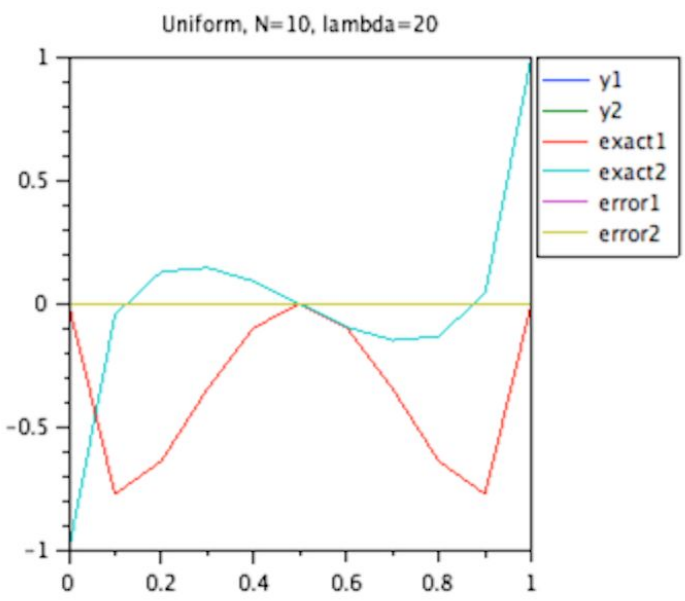
Following are the graphs that we formed for Simple Shooting Scheme:

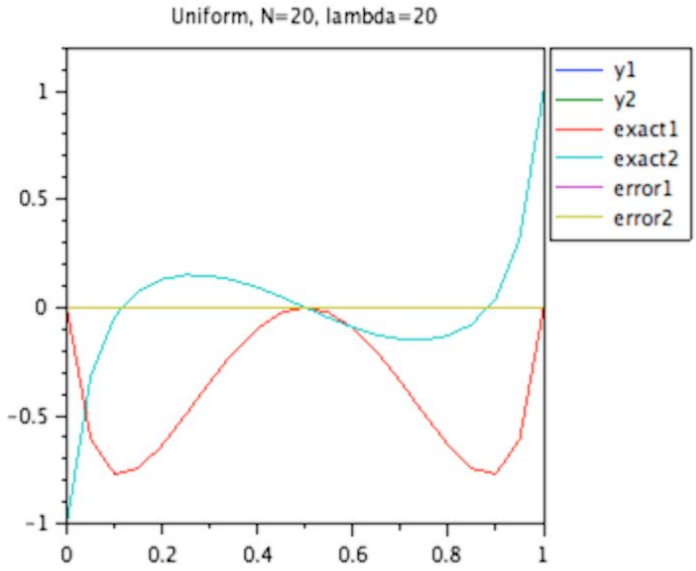


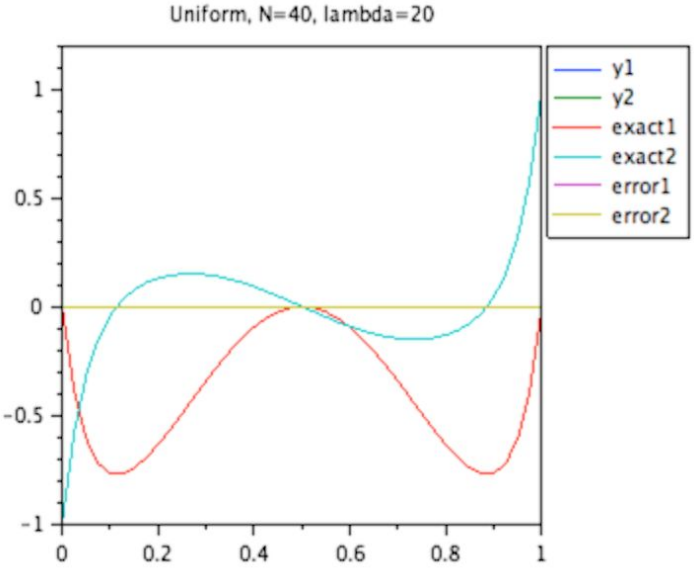


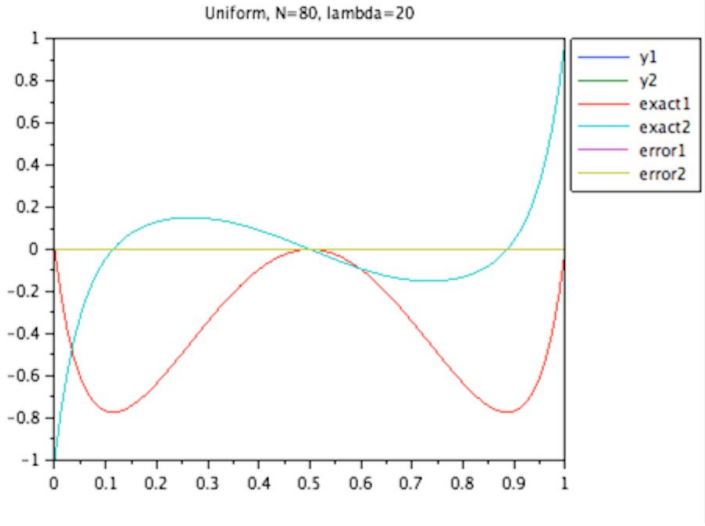


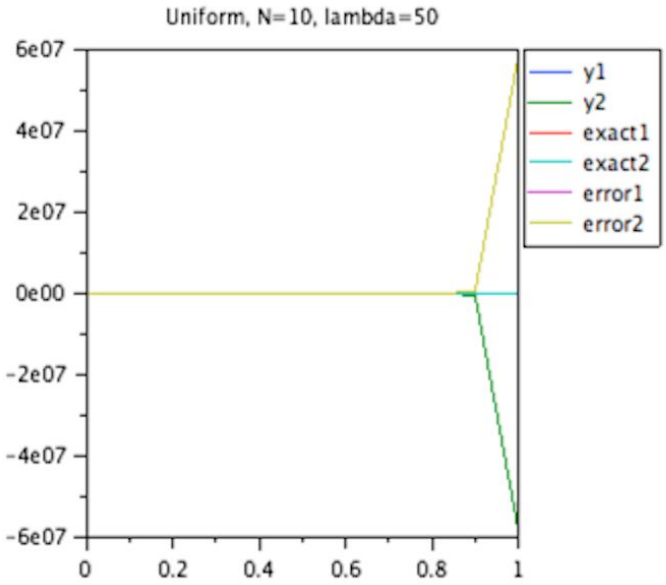


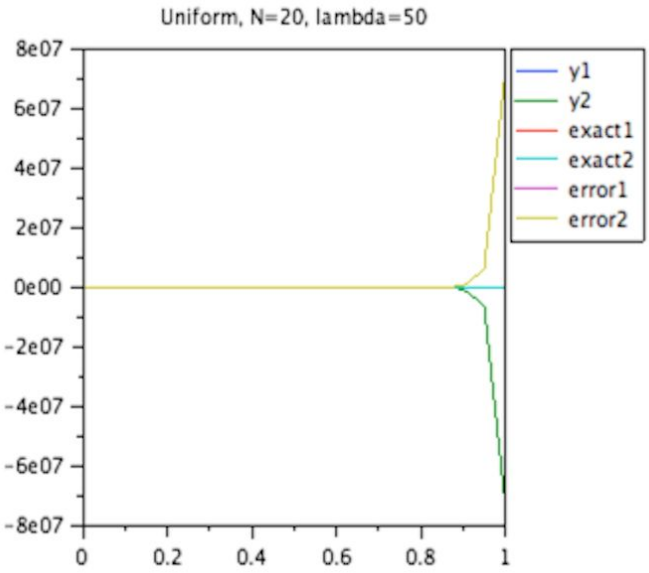

Uniform, N=10, lambda=20
y1
y2
exact1
exact2
error1
error2

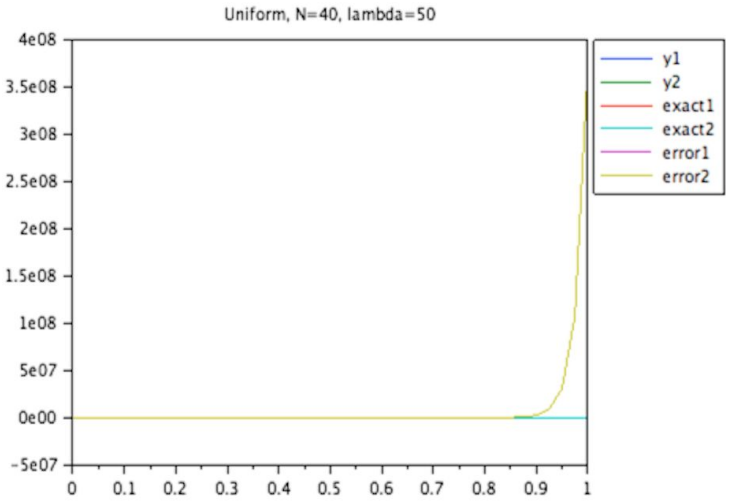

Uniform, N=20, lambda=20
y1
y2
exact1
exact2
error1
error2

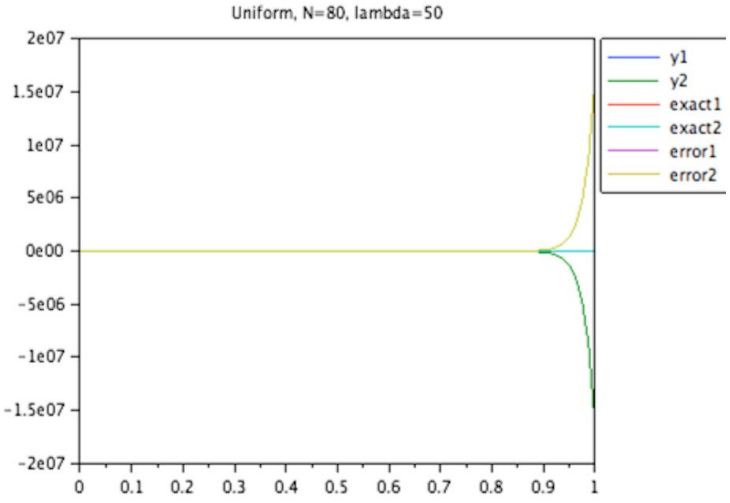

Uniform, N=40, lambda=20
y1
y2
exact1
exact2
error1
error2

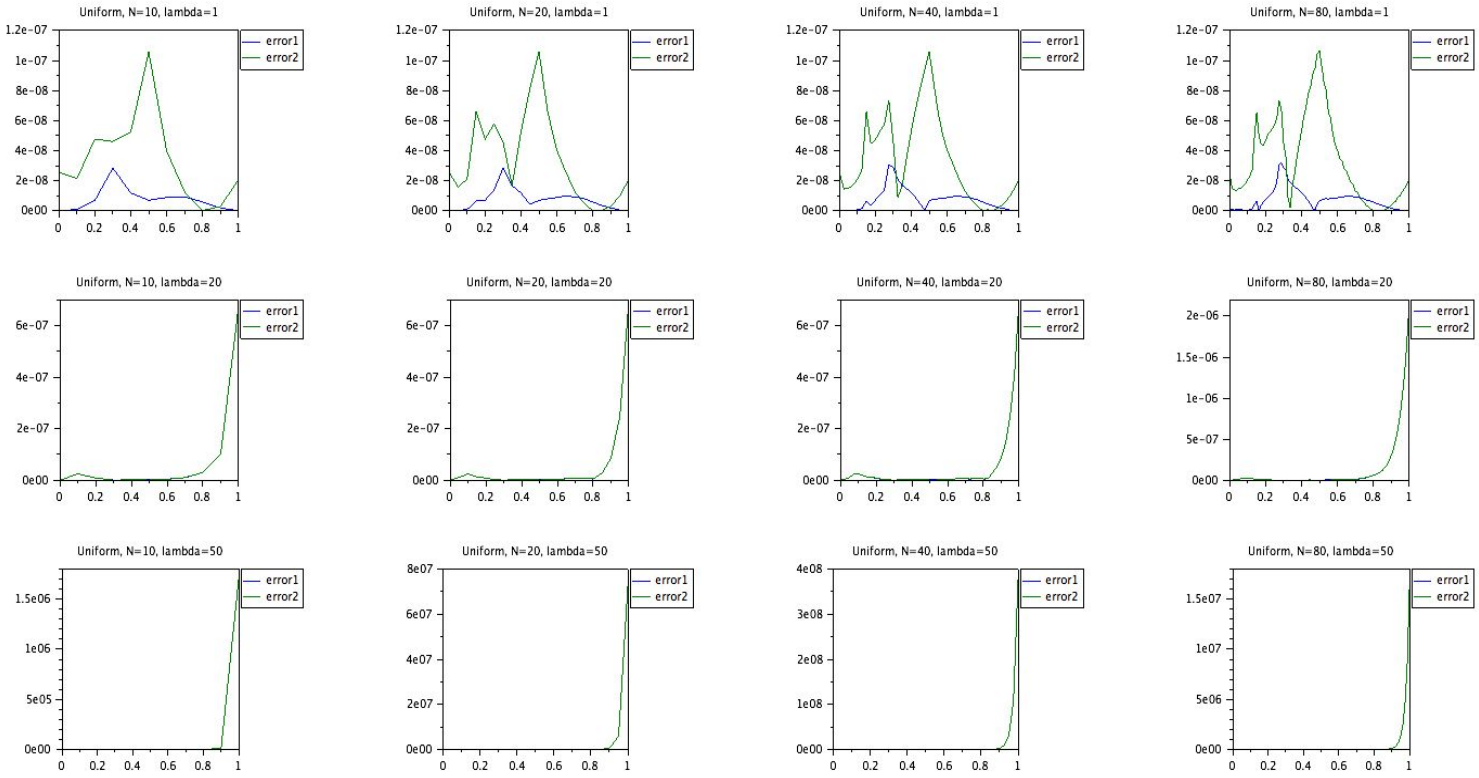

Uniform, N=80, lambda=20
y1
y2
exact1
exact2
error1
error2


Uniform, N=10, lambda=50
6e07
4e07
2e07
0e00
-2e07
-4e07
-6e07
0
0.2
0.4
0.6
0.8
1
y1
y2
exact1
exact2
error1
error2


Uniform, N=20, lambda=50
8e07
6e07
4e07
2e07
0e00
-2e07
-4e07
-6e07
-8e07
0
0.2
0.4
0.6
0.8
1
y1
y2
exact1
exact2
error1
error2


Uniform, N=40, lambda=50
4e08
3.5e08
3e08
2.5e08
2e08
1.5e08
1e08
5e07
0e00
-5e07
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
y1
y2
exact1
exact2
error1
error2


Uniform, N=80, lambda=50
2e07
1.5e07
1e07
5e06
0e00
-5e06
-1e07
-1.5e07
-2e07
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
y1
y2
exact1
exact2
error1
error2



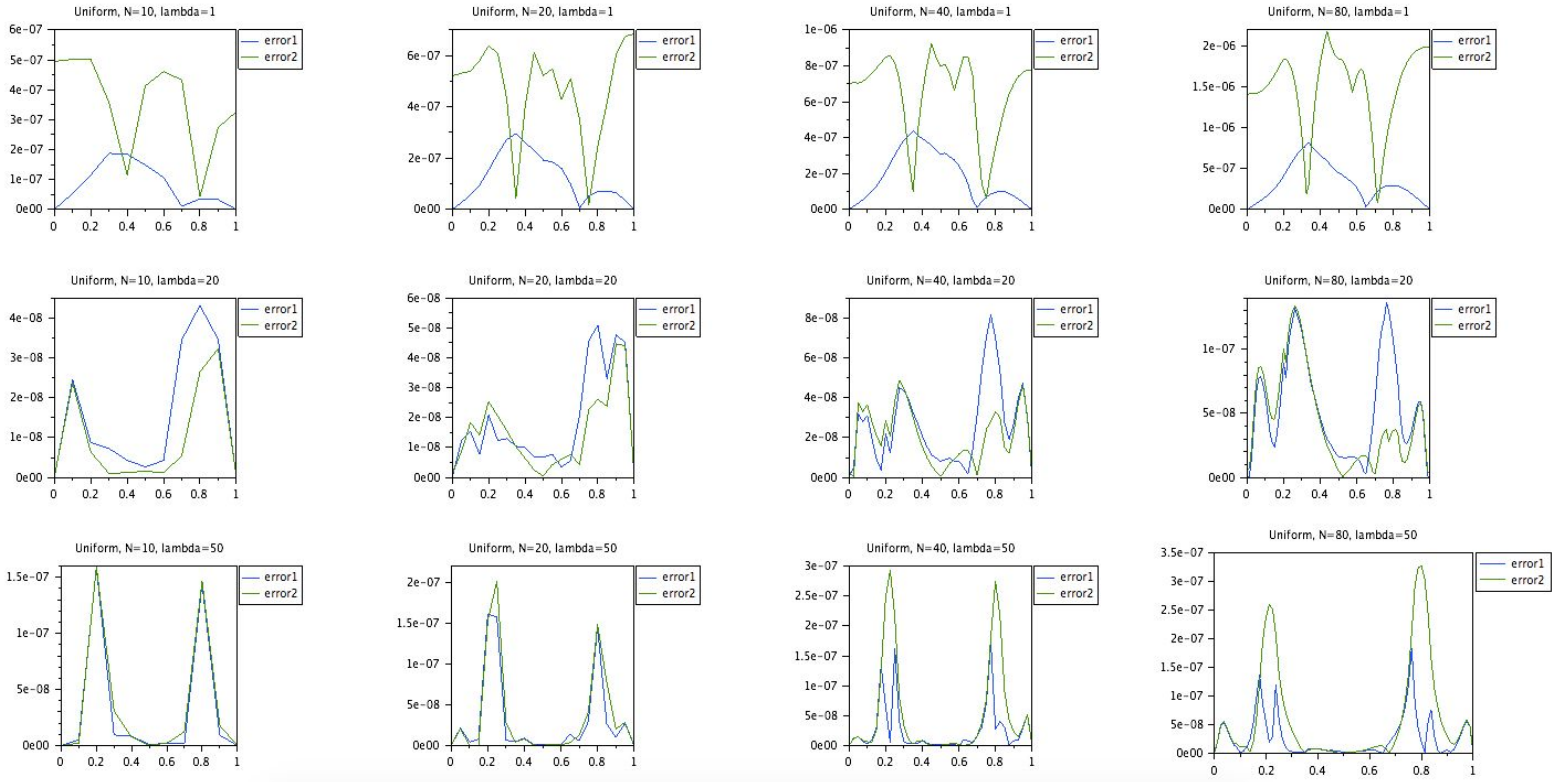
→ Uniform: Multiple Shooting Scheme:

Table 4: Multiple shooting scheme with a uniform mesh.

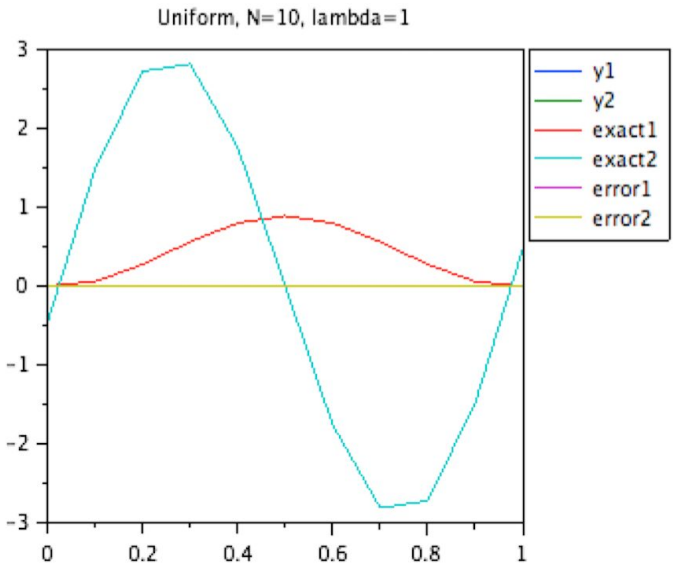
λ	N	y_1 Max Err	y_2 Max Err	y_1 Order	y_2 Order	Time
1	10	0.0000002	0.0000005	-0.6574772	-0.4500963	0.005474
1	20	0.0000003	0.0000007	-0.5625555	-0.4356636	0.008416
1	40	0.0000004	0.0000009	-0.9016626	-1.2364923	0.017031
1	80	0.0000008	0.0000022	0	0	0.035309
20	10	4.314D-08	3.228D-08	-0.2384894	-0.4679142	0.006214
20	20	5.090D-08	4.465D-08	-0.6771532	-0.1261426	0.013645
20	40	8.139D-08	4.873D-08	-0.7423471	-1.4557758	0.018553
20	80	0.0000001	0.0000001	0	0	0.027942
50	10	0.0000002	0.0000002	-0.0210660	-0.3383182	0.008818
50	20	0.0000002	0.0000002	-0.0940482	-0.5390553	0.011337
50	40	0.0000002	0.0000003	-0.0901573	-0.1587148	0.019907
50	80	0.0000002	0.0000003	0	0	0.028784

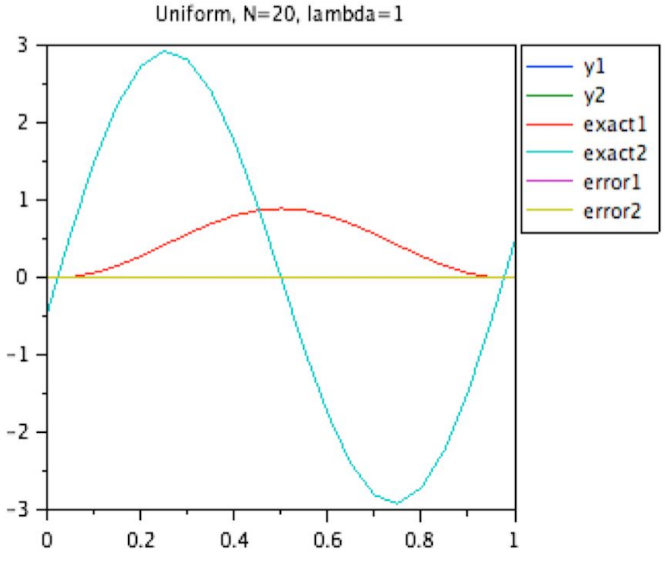
- Multiple shooting computes the solution of the system at multiple points, unlike simple shooting.
- Multiple points are used to control the error below the tolerance.

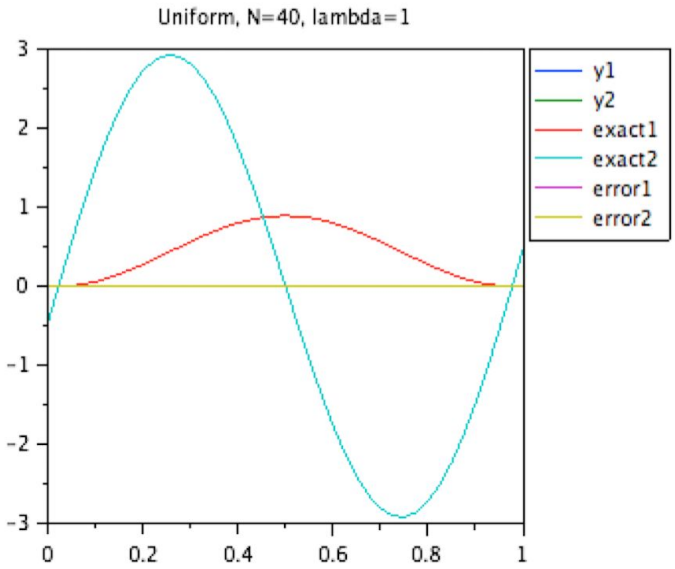
- Surprisingly, the error is reduced when lambda is increased from 1 to 20 but later it is increased when lambda reaches 50.
- The error in multiple shooting behaves quite differently. So we took out a different plot for all the errors that we computed.

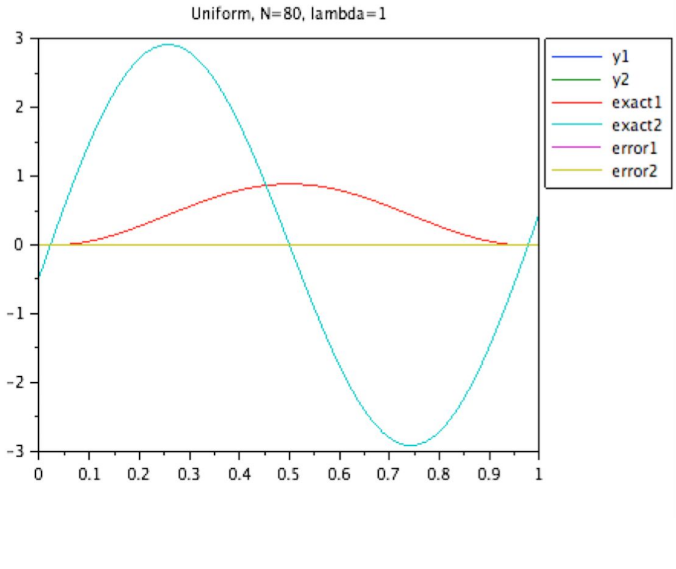


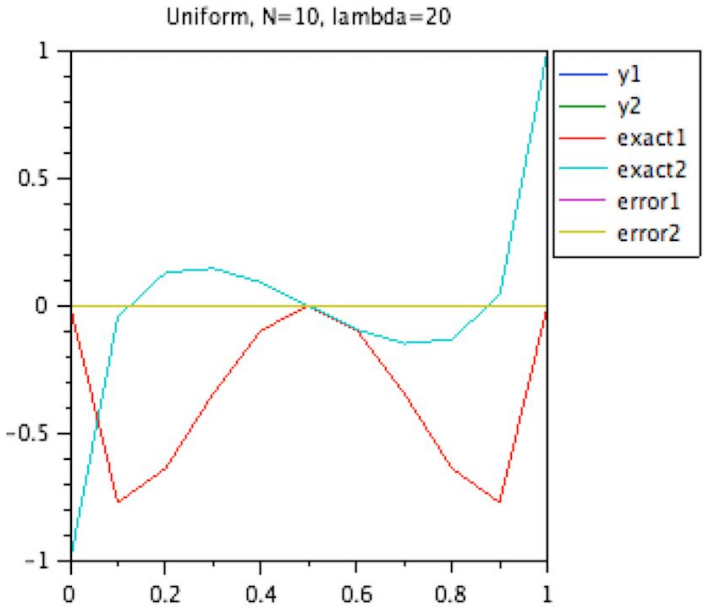
- For *lambda = 50,* we can see the error is close to 0 but rises when *x = 0.1*. The error reaches its maximum point and then drops back to 0 near *x = 0.5* and repeats the same cycle again from *x=0.5 to x = 1.0.*
- The order comes out to be negative because the error increases on the increase of number of mesh points.
- The error remains quite low throughout the *multiple-shooting scheme* and hence our plots for the computed and exact solutions look pretty clean:

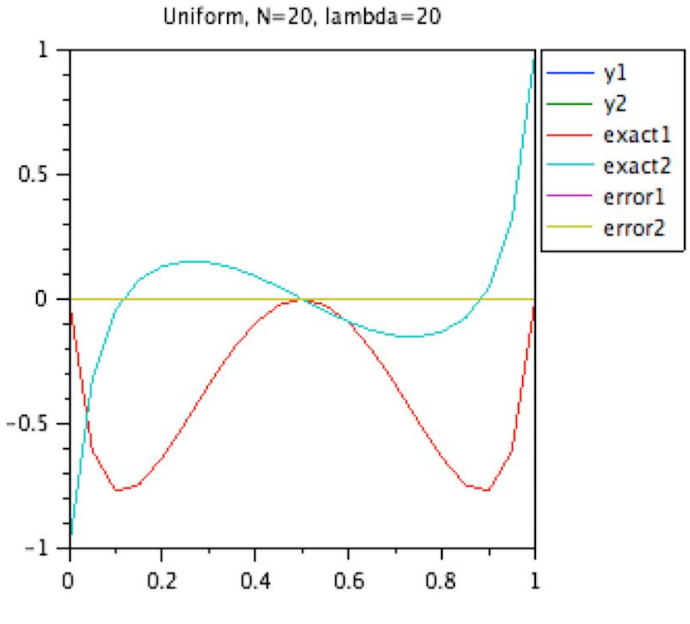

Uniform, N=10, lambda=1
y1
y2
exact1
exact2
error1
error2
3
2
1
0
-1
-2
-3
0
0.2
0.4
0.6
0.8
1

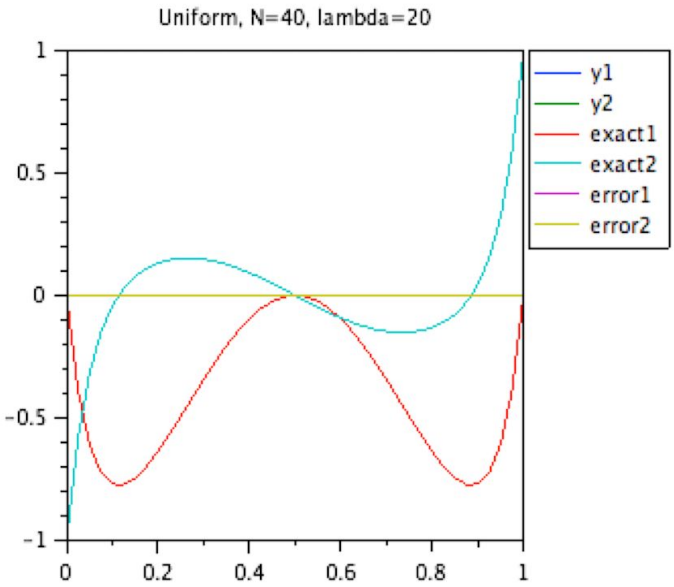

Uniform, N=20, lambda=1
y1
y2
exact1
exact2
error1
error2
3
2
1
0
-1
-2
-3
0
0.2
0.4
0.6
0.8
1

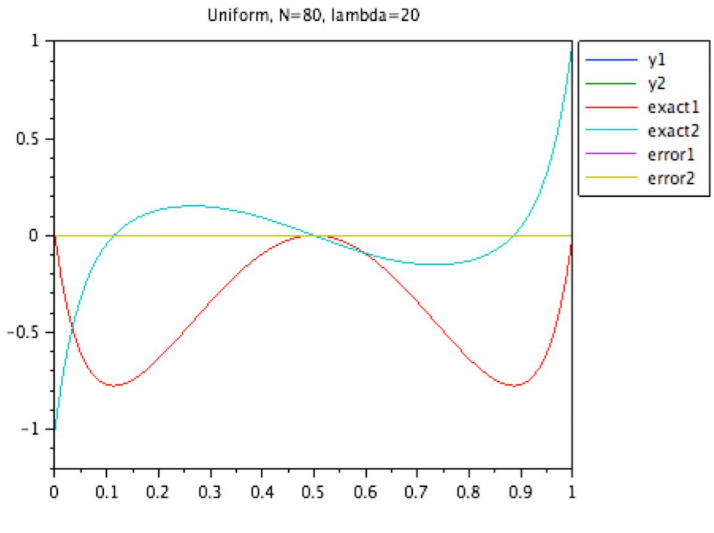

Uniform, N=40, lambda=1
y1
y2
exact1
exact2
error1
error2
3
2
1
0
-1
-2
-3
0
0.2
0.4
0.6
0.8
1

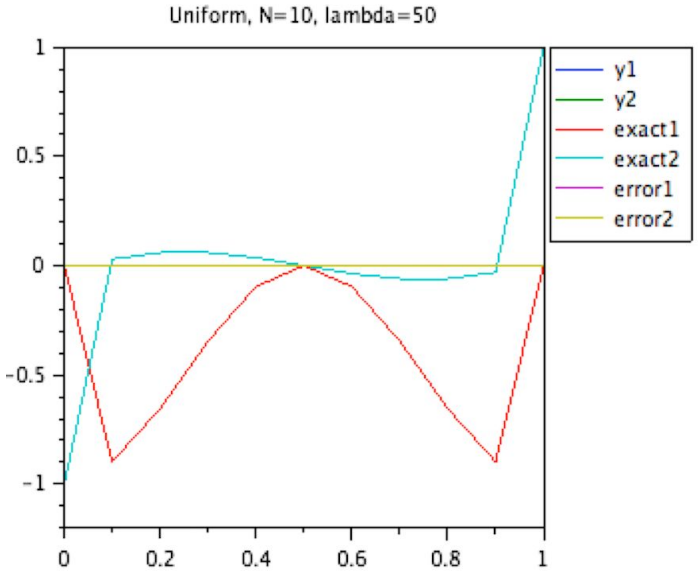

Uniform, N=80, lambda=1
y1
y2
exact1
exact2
error1
error2
3
2
1
0
-1
-2
-3
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

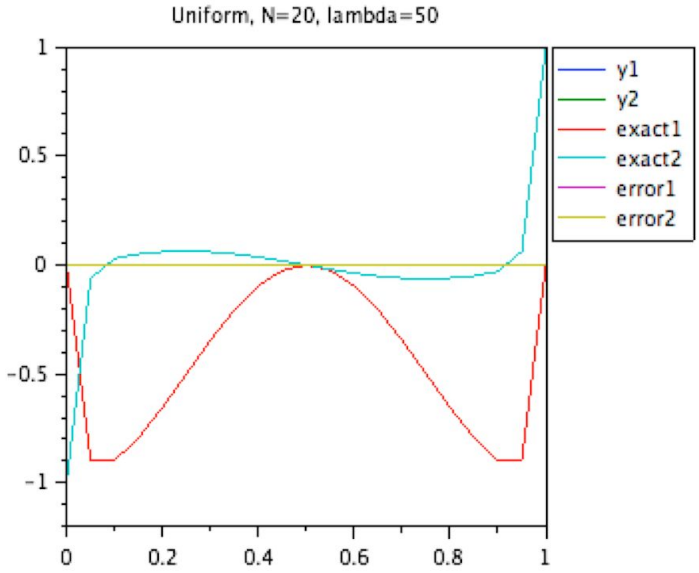

Uniform, N=10, lambda=20
y1
y2
exact1
exact2
error1
error2
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1

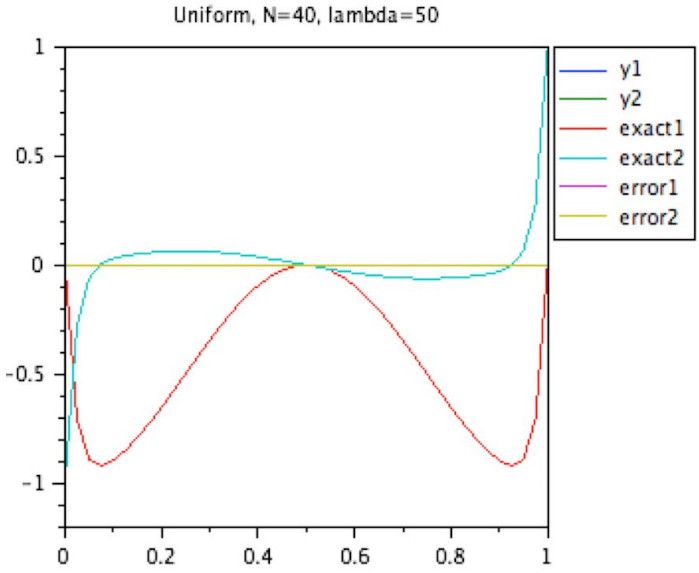

Uniform, N=20, lambda=20
y1
y2
exact1
exact2
error1
error2
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1

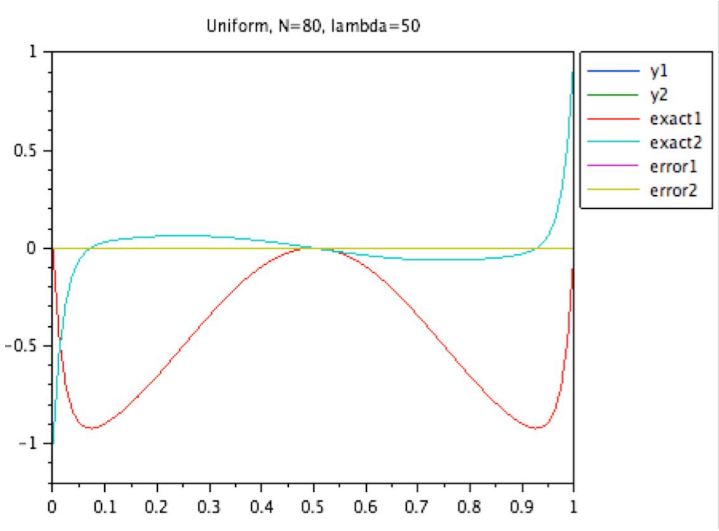

Uniform, N=40, lambda=20
y1
y2
exact1
exact2
error1
error2
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1


Uniform, N=80, lambda=20
y1
y2
exact1
exact2
error1
error2
1
0.5
0
-0.5
-1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Uniform, N=10, lambda=50
y1
y2
exact1
exact2
error1
error2
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1


Uniform, N=20, lambda=50
y1
y2
exact1
exact2
error1
error2
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1


Uniform, N=40, lambda=50
y1
y2
exact1
exact2
error1
error2
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1


Uniform, N=80, lambda=50
y1
y2
exact1
exact2
error1
error2
1
0.5
0
-0.5
-1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

→ Comparison Analysis

ERRORS OF DIFFERENT COMPUTATIONAL METHODS IN A UNIFORM MESH

		Midpoint Scheme		Trapezoidal Scheme		Simple Shooting		Multiple Shooting	
λ	N	y_1 Max Err	y_2 Max Err	y_1 Max Err	y_2 Max Err	y_1 Max Err	y_2 Max Err	y_1 Max Err	y_2 Max Err
1	10	0.0147390	0.0565223	0.0601956	0.0887246	2.846D-08	0.0000001	0.0000002	0.0000005
1	20	0.0035749	0.0146184	0.0151461	0.0228610	2.853D-08	0.0000001	0.0000003	0.0000007
1	40	0.0008870	0.0036463	0.0037925	0.0057041	3.071D-08	0.0000001	0.0000004	0.0000009
1	80	0.0002213	0.0009111	0.0009485	0.0014262	3.173D-08	0.0000001	0.0000008	0.0000022
20	10	0.1535066	0.1427874	0.1328192	0.1379037	0.0000007	0.0000007	4.314D-08	3.228D-08
20	20	0.0379282	0.0336602	0.0340840	0.0351255	0.0000007	0.0000007	5.090D-08	4.465D-08
20	40	0.0086834	0.0076221	0.0077699	0.0080282	0.0000007	0.0000007	8.139D-08	4.873D-08
20	80	0.0021278	0.0018628	0.0019021	0.0019665	0.0000022	0.0000022	0.0000001	0.0000001
50	10	0.4668580	0.4486886	0.4350377	0.4357242	1712709.1	1712709.1	0.0000002	0.0000002
50	20	0.1996767	0.1942742	0.1930584	0.1933379	72368184	72368184	0.0000002	0.0000002
50	40	0.0568815	0.0554363	0.0557113	0.0557626	3.757D+08	3.757D+08	0.0000002	0.0000003
50	80	0.0123979	0.0120372	0.0121227	0.0121355	16463206	16463206	0.0000002	0.0000003

- Multiple Shooting seems the best method for computing the solution of a BVODE since the error remains low for any lambda value.
- Simple Shooting seems the worst method for lambda > 1. However, it looks like the best method for lambda = 1.
- Midpoint Scheme and Trapezoidal scheme are quite similar in computing the errors. The error increases with increase of lambda in both of the cases whereas the error decreases as we increase N.

Non-Uniform Mesh

Now we will compare each of the computation methods based on a non-uniform mesh. We used the following mesh as our non-uniform mesh for N = 10. However, the mesh was subdivided for N = 20, 40 and 80.

[0, 1/lambda, 3/lambda, 5/lambda, 8/lambda, 0.5, 1-8/lambda, 1-5/lambda, 1-3/lambda, 1-1/lambda, 1]

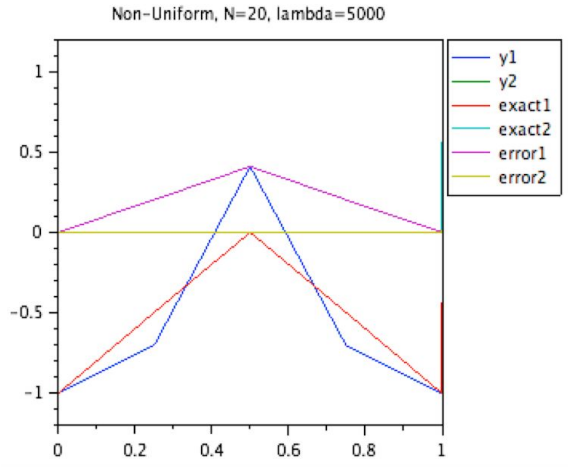
Note: We removed the values 0.25 because 8/lambda came out to be more than 0.25 for lambda = 20.

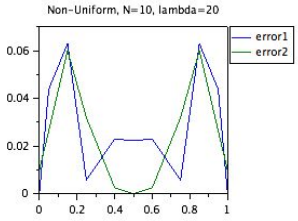
→ Non-Uniform: Midpoint Scheme and Trapezoidal Scheme:

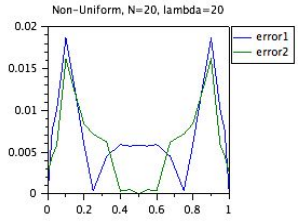
Table 5: Midpoint scheme with a non-uniform mesh.

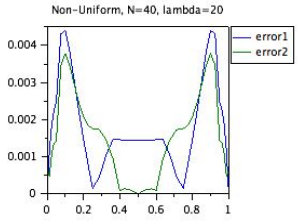
λ	N	y_1 Max Err	y_2 Max Err	y_1 Order	y_2 Order	Time
20	10	0.0629894	0.0600439	1.7525625	1.8909437	0.01955
20	20	0.0186936	0.0161897	2.0857431	2.0964631	0.02753
20	40	0.0044037	0.0037856	1.9920167	2.0218413	0.0922
20	80	0.0011070	0.0009322	0	0	0.326515
50	10	0.1969546	0.0551958	0.9740435	1.075522	0.007468
50	20	0.1002651	0.0261904	2.4624066	2.079965	0.052887
50	40	0.0181925	0.0061945	2.0490947	2.6507227	0.093516
50	80	0.0043959	0.0009864	0	0	0.327057
5000	10	0.0497875	0.0497871	-3.040121	1.6990057	0.010783
5000	20	0.4095318	0.0153343	2.3326019	2.094068	0.031974
5000	40	0.0813026	0.0035916	2.0908873	1.9831757	0.093406
5000	80	0.0190847	0.0009084	0	0	0.329206

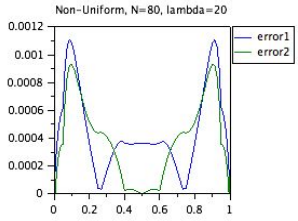
- Midpoint scheme for non-uniform mesh handles error well for lambda = 20.
- The error increases as we increase the lambda value but further decreases on the increase of number of mesh points. The order seems to be close to 2 for all of the cases except for Lambda = 5000, N = 20.
- Maximum time is taken to compute the solution at N = 80 for any case. Hence, the time is solely dependent on the number of subintervals.
- However, when Lambda = 5000 and N = 20, the error is surprisingly big for y1 and hence the graph looks distorted.

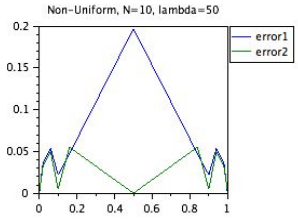


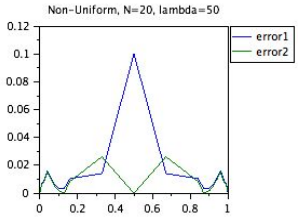


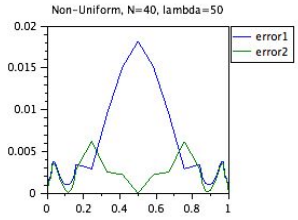


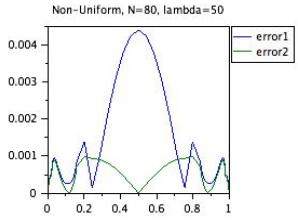


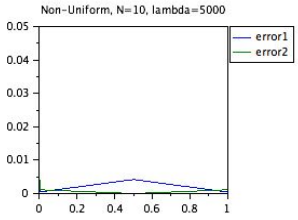


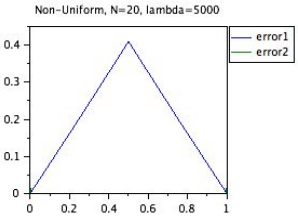


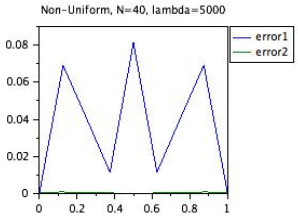


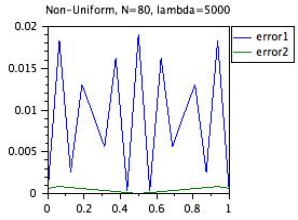












Comparison to <u>Simple One-Step Schemes for Linear BVODEs</u>

<u>For lambda = 50</u>

N	Notes: $e(y_1)$	Notes: $e(y_2)$	Computed: $e(y_1)$	Computed: $e(y_2)$
10	.16	.51-1	.20	.55-1
20	.42-1	.14-1	.10	.26-1
40	.96-2	.34-2	.18-1	.61-2

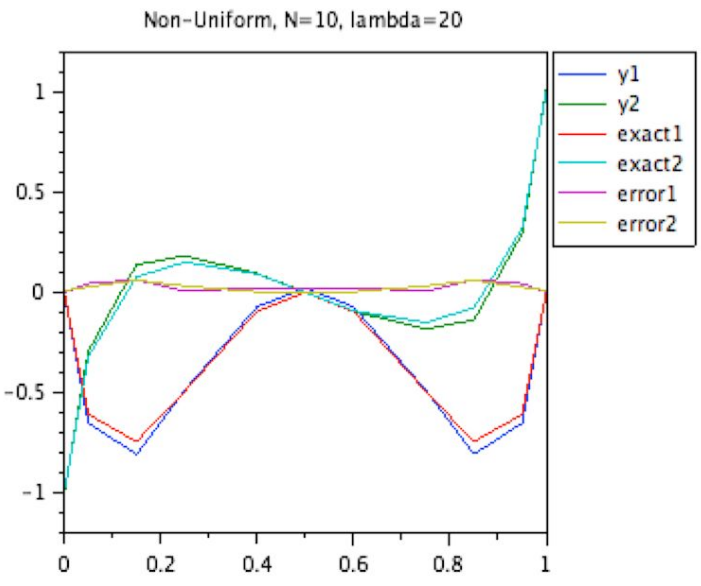
80	.24-2	.85-2	.42-2	.99-3

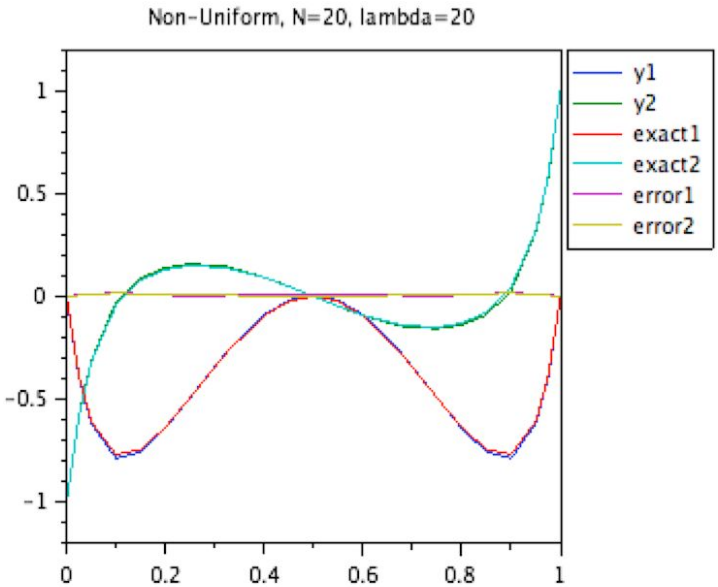
For lambda = 5000

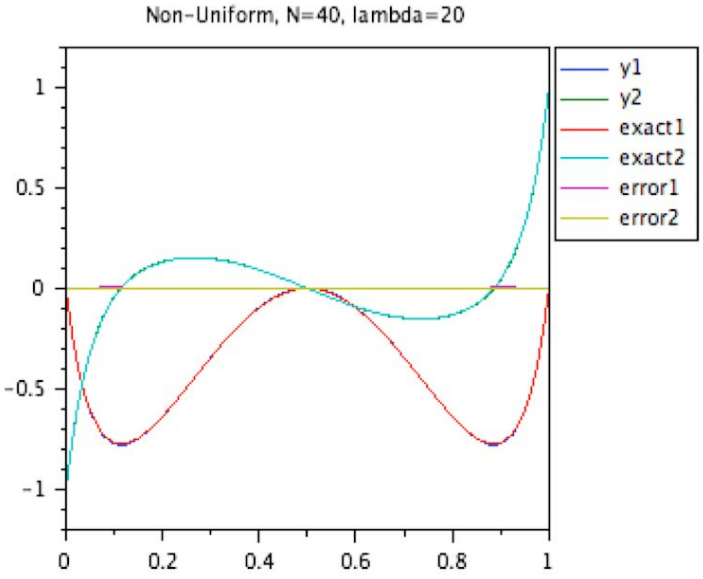
N	Notes: $e(y_1)$	Notes: $e(y_2)$	Computed: $e(y_1)$	Computed: $e(y_2)$
10	.95	.95	.49-1	.49-1
20	.82-1	.15-1	.40	.015-1
40	.19-1	.36-2	.08	.36-2
80	.44-2	.91-3	.019	.90-3

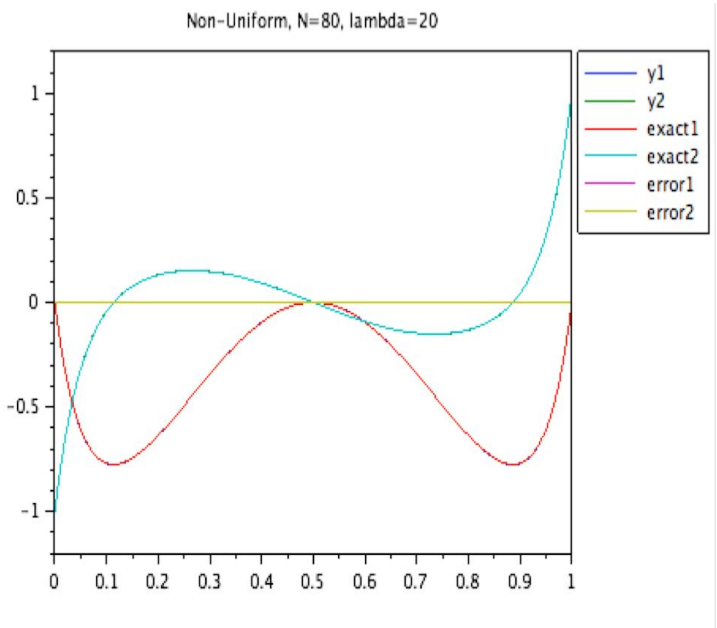
- Our results does not exactly match the results given in "Simple One-Step Schemes for Linear BVODEs". That is may be because we used a slightly modified mesh than what is used in "Simple One-Step Schemes for Linear BVODEs".

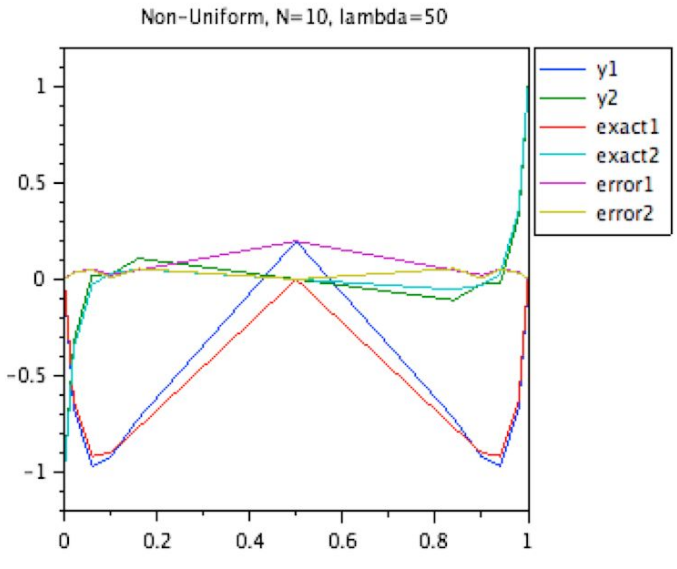
  On the next page we will see the plots for the solution of *Non-uniform midpoint scheme*:

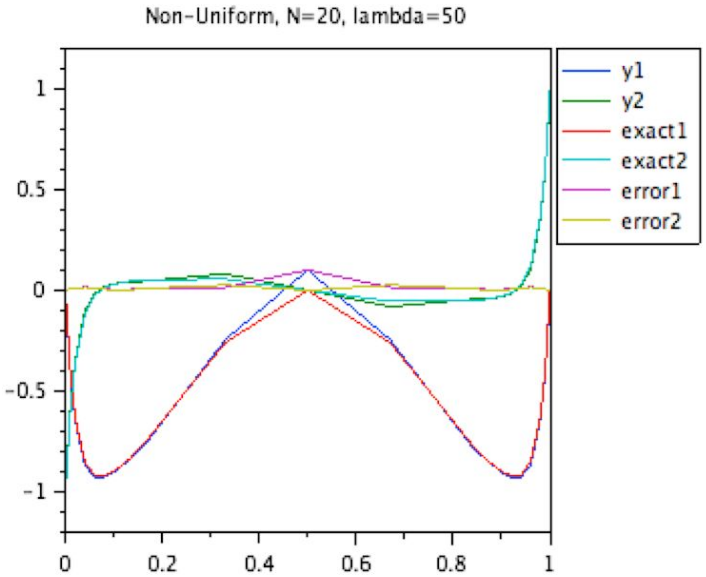


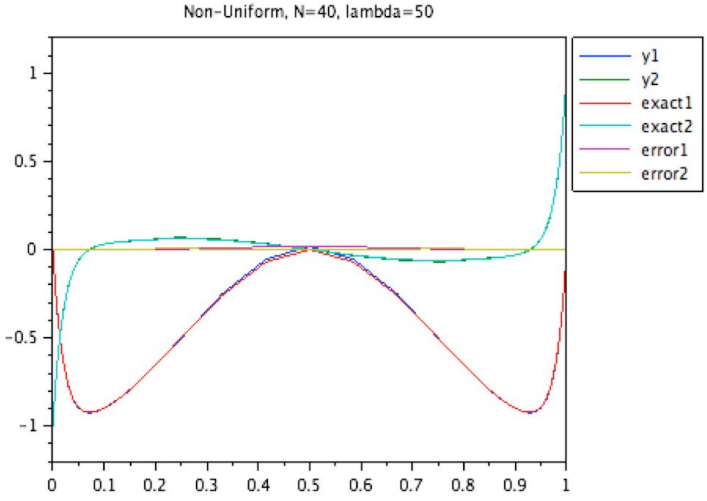


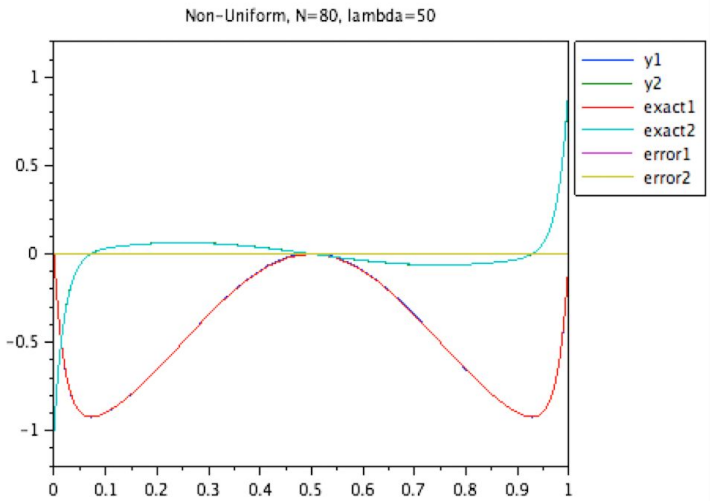

Non-Uniform, N=40, lambda=20
y1
y2
exact1
exact2
error1
error2
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1

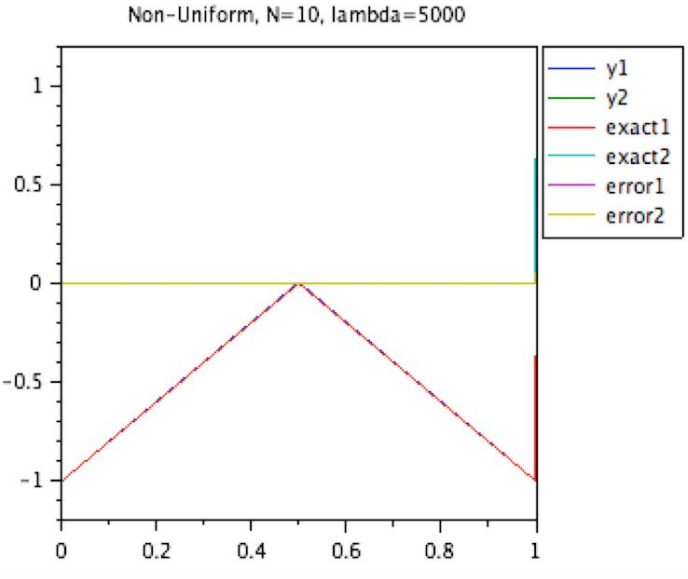

Non-Uniform, N=80, lambda=20
y1
y2
exact1
exact2
error1
error2
1
0.5
0
-0.5
-1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

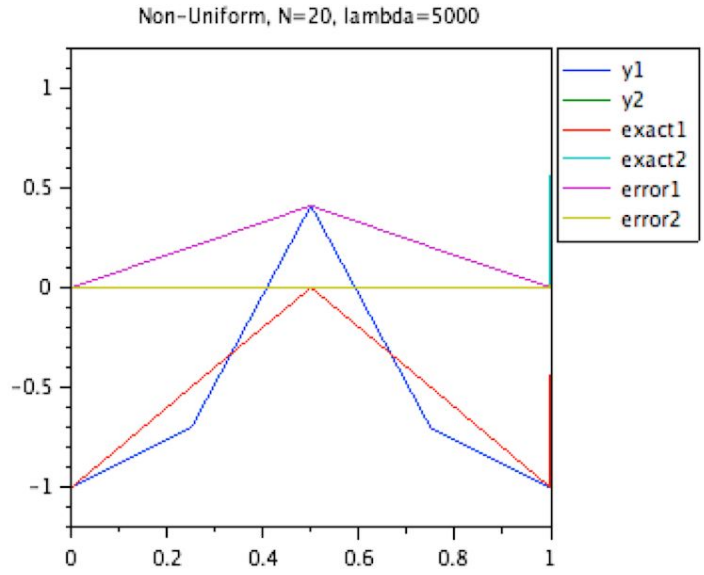

Non-Uniform, N=10, lambda=50
y1
y2
exact1
exact2
error1
error2
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1

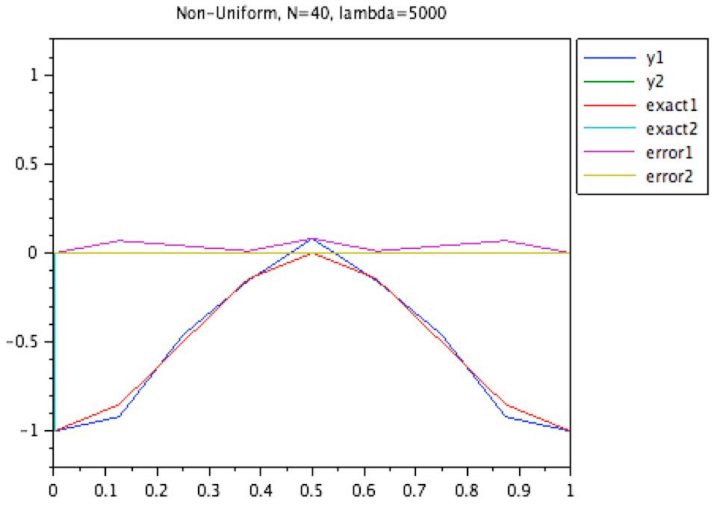

Non-Uniform, N=20, lambda=50
y1
y2
exact1
exact2
error1
error2
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1

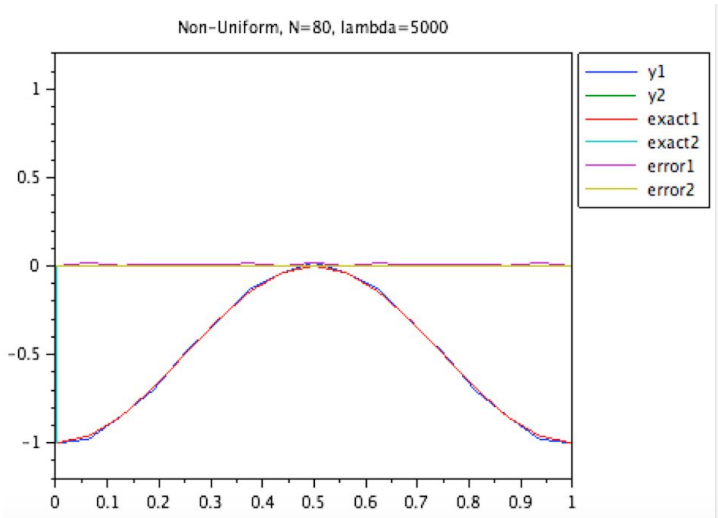

Non-Uniform, N=40, lambda=50
y1
y2
exact1
exact2
error1
error2


Non-Uniform, N=80, lambda=50
y1
y2
exact1
exact2
error1
error2


Non-Uniform, N=10, lambda=5000
y1
y2
exact1
exact2
error1
error2


Non-Uniform, N=20, lambda=5000
y1
y2
exact1
exact2
error1
error2

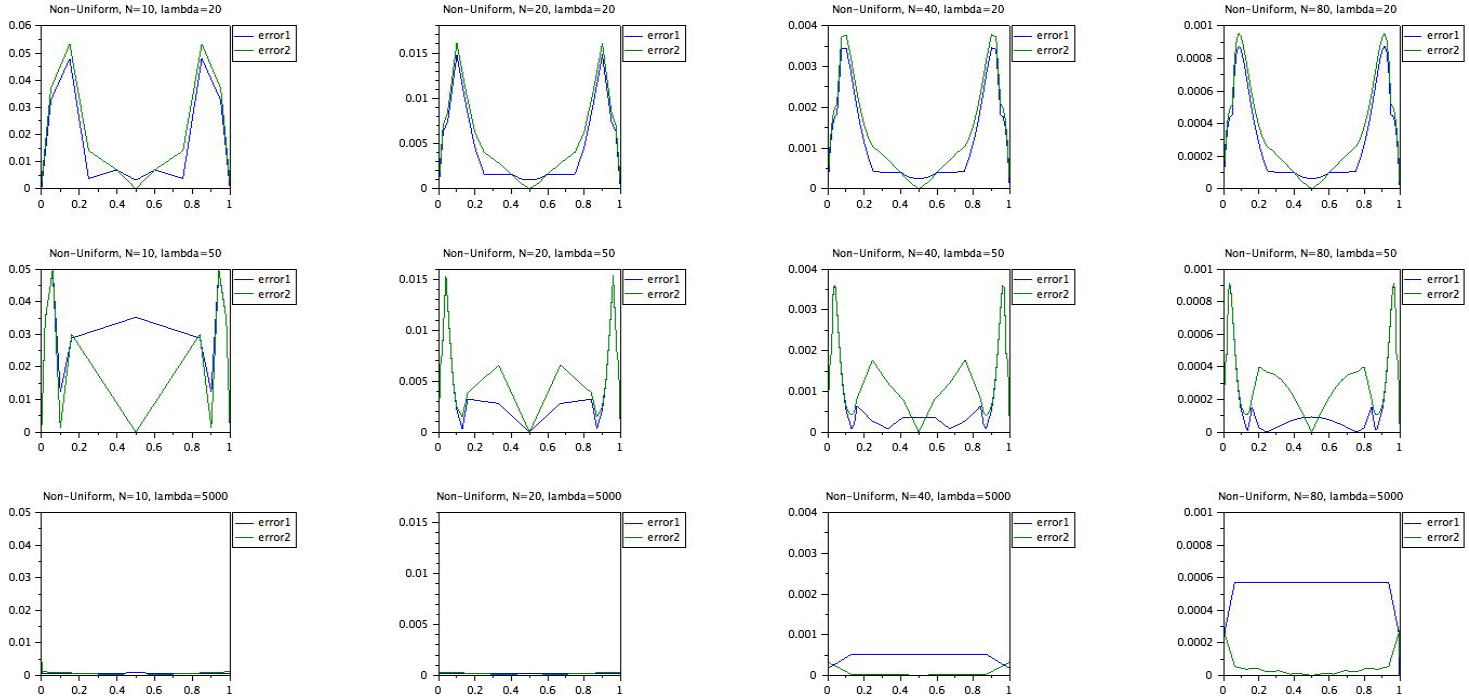




20	10	0.0479591	0.0533222	1.6969568	1.7298768	0.020867
20	20	0.0147923	0.0160755	2.0963607	2.0881811	0.044018
20	40	0.0034591	0.0037806	1.9807647	1.9871913	0.124603
20	80	0.0008764	0.0009536	0	0	0.471559
50	10	0.0497103	0.0499058	1.6987149	1.69986	0.010205
50	20	0.0153138	0.0153618	2.0939718	2.0929907	0.035764
50	40	0.0035870	0.0036007	1.9816779	1.9821572	0.118037
50	80	0.0009082	0.0009114	0	0	0.457644
5000	10	0.0497871	0.0497871	1.6989226	1.6989097	0.013613
5000	20	0.0153352	0.0153354	2.0937152	2.093329	0.042366
5000	40	0.0035927	0.0035937	1.9824028	1.981305	0.063248
5000	80	0.0009092	0.0009101	0	0	0.490215

Table 6: Trapezoidal scheme with a non-uniform mesh.

- In the non-uniform trapezoidal scheme the error hardly changes when lambda is increased. Hence, it doesn't depend on the value of lambda.
- As noticed earlier, the error decreases with the number of intervals and the order of the error remains to be close to 2.
- The maximum time taken to compute the solution is at the case of N = 80 for any lambda value.



(Above) Figure showing plots of all of the graphs computed for the errors of *Non-Uniform Trapezoidal Scheme.*

Comparison to Simple One-Step Schemes for Linear BVODEs

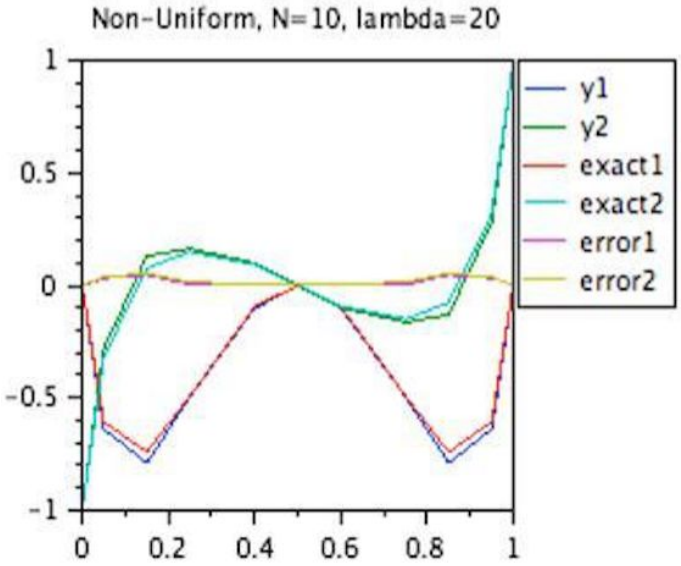
For lambda = 50

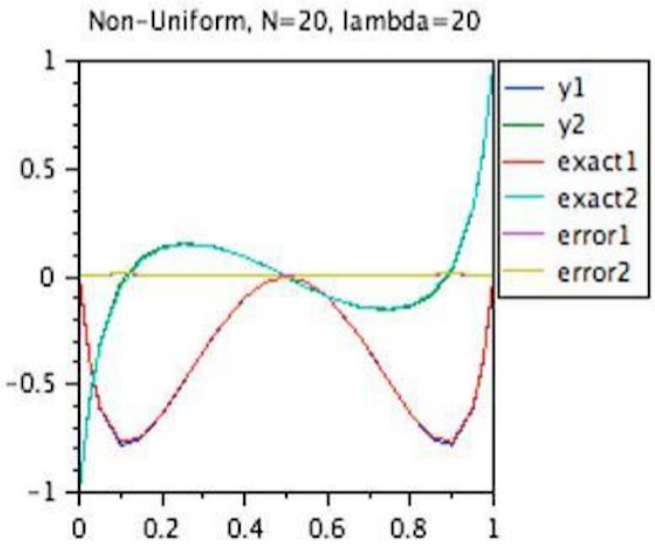
N	Notes: $e(y_1)$	Notes: $e(y_2)$	Computed: $e(y_1)$	Computed: $e(y_2)$
10	.55	.56	.49-1	.50-1
20	.15-1	.15-1	.15-1	.15-1
40	.36-2	.36-2	.36-2	.36-2
80	.91-3	.91-3	.90-3	.91-3

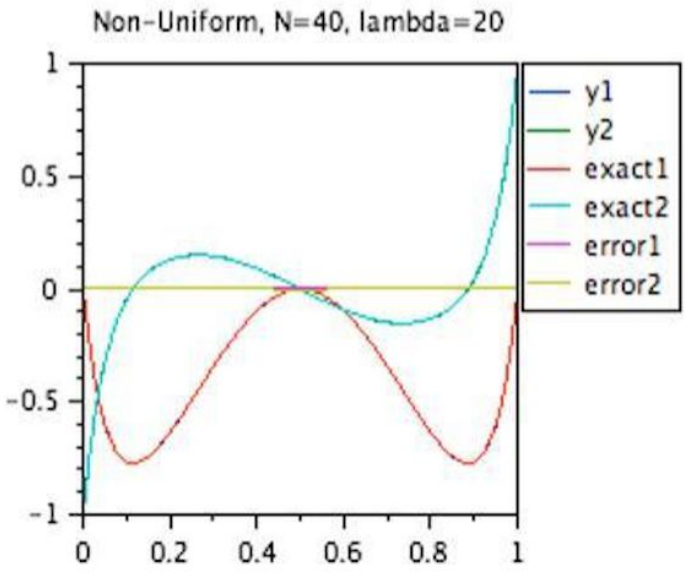
For lambda = 5000

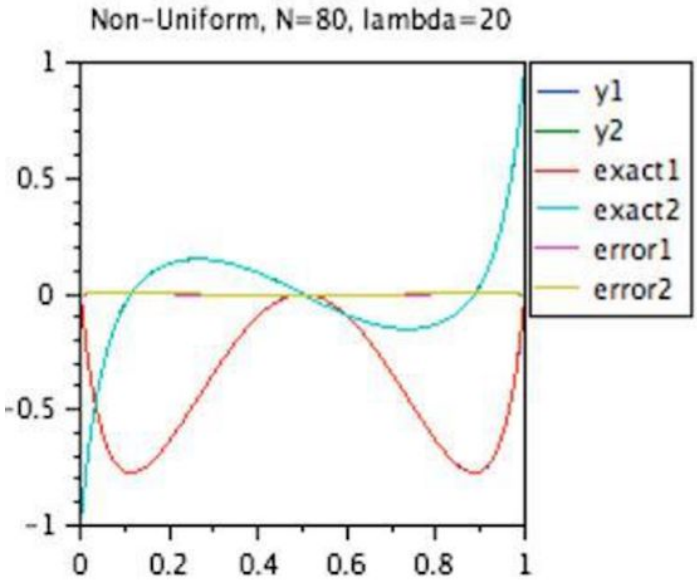
N	Notes: $e(y_1)$	Notes: $e(y_2)$	Computed: $e(y_1)$	Computed: $e(y_2)$
10	.45	.45	.50-1	.50-1
20	.15-1	.15-1	.15-1	.15-1
40	.36-2	.36-2	.36-2	.36-2
80	.91-3	.91-3	.90-3	.91-3

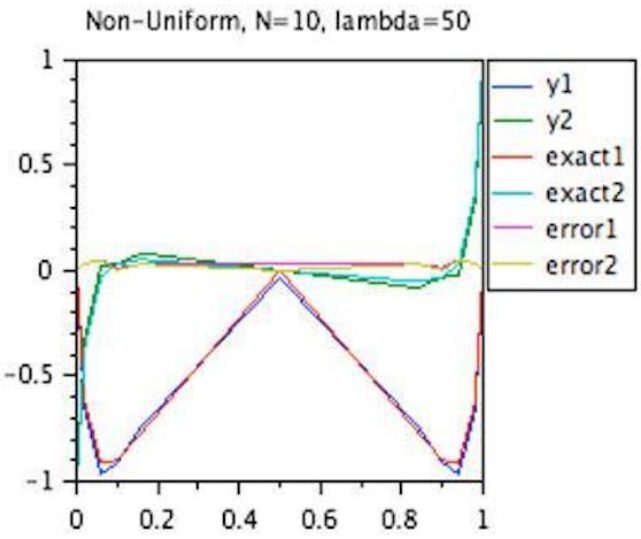
- Our solution is quite similar to the results given in *Simple One-Step Schemes for Linear BVODEs.*
- Observations are quite similar to what we discussed above; the error seems to be unaffected with the increase of lambda values but decreases with the increase of N.

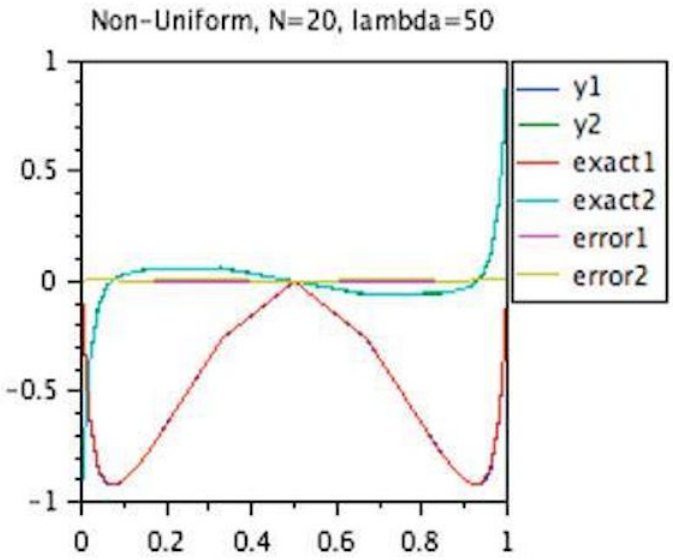


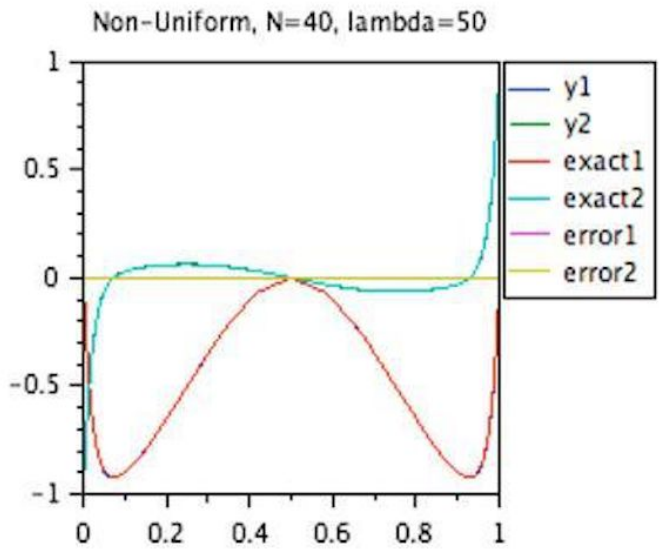


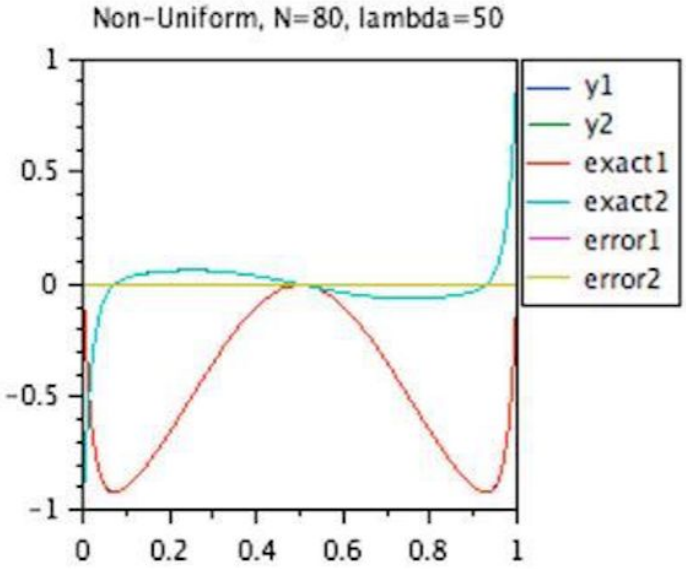


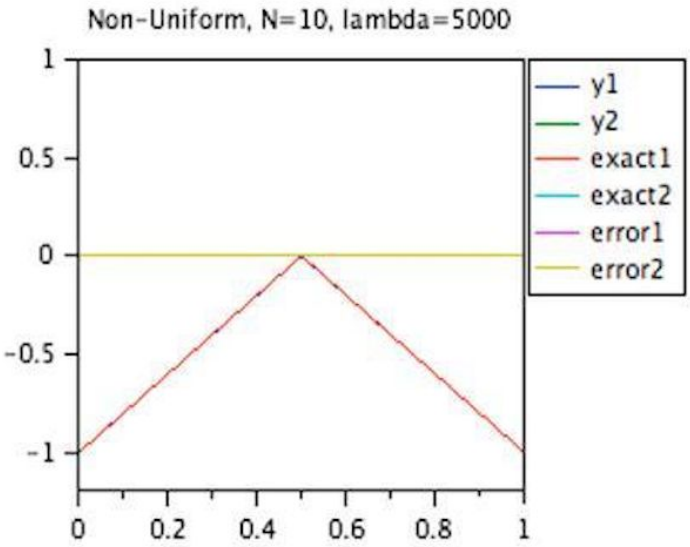


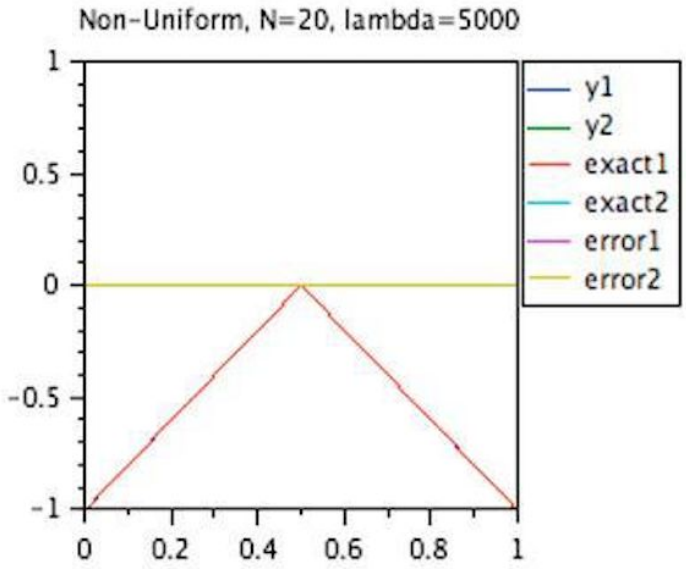

Non-Uniform, N=10, lambda=50
y1
y2
exact1
exact2
error1
error2
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1

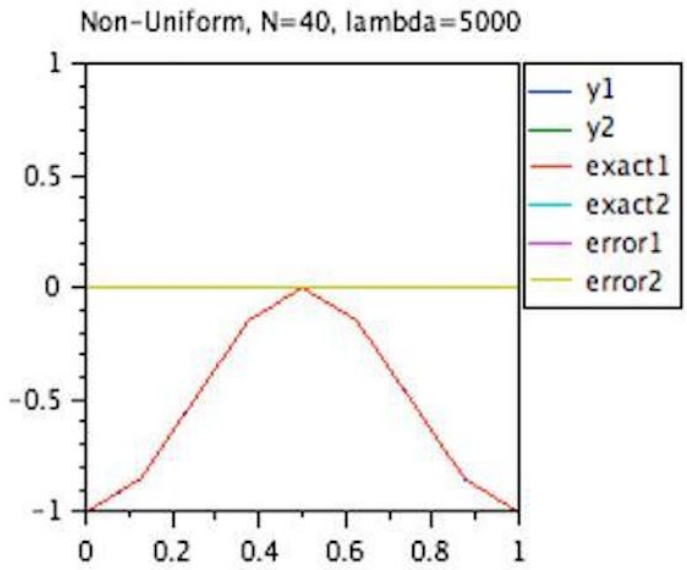

Non-Uniform, N=20, lambda=50
y1
y2
exact1
exact2
error1
error2
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1

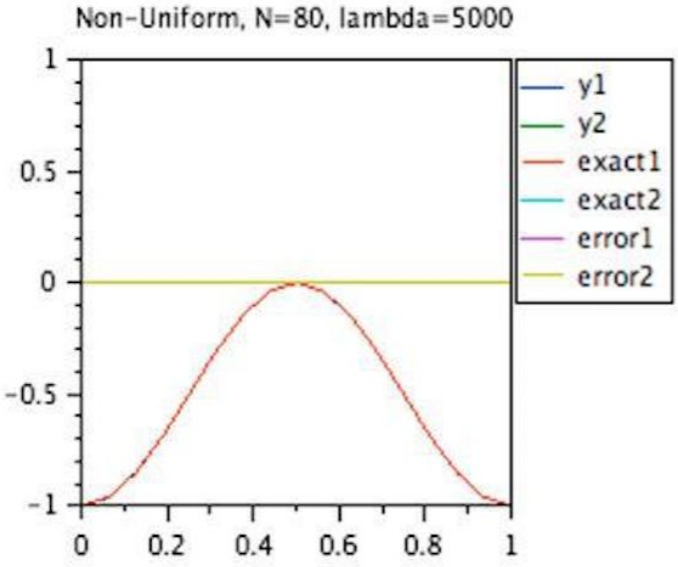

Non-Uniform, N=40, lambda=50
y1
y2
exact1
exact2
error1
error2
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1


Non-Uniform, N=80, lambda=50
y1
y2
exact1
exact2
error1
error2
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1


Non-Uniform, N=10, lambda=5000
y1
y2
exact1
exact2
error1
error2
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1


Non-Uniform, N=20, lambda=5000
y1
y2
exact1
exact2
error1
error2
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1


Non-Uniform, N=40, lambda=5000
y1
y2
exact1
exact2
error1
error2
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1

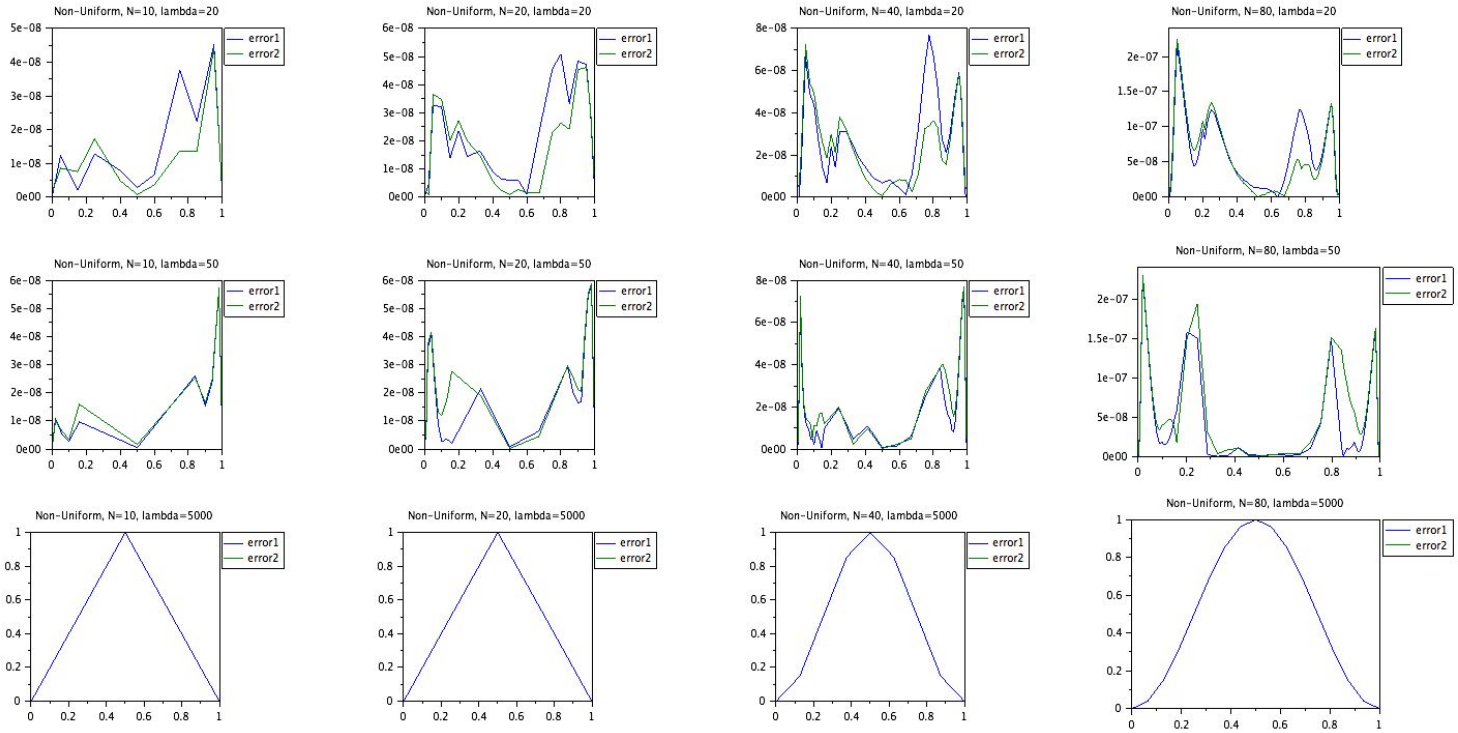

Non-Uniform, N=80, lambda=5000
y1
y2
exact1
exact2
error1
error2
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1

→ Non-Uniform: Multiple Shooting Scheme:

Table 7: Multiple shooting scheme with a non-uniform mesh.

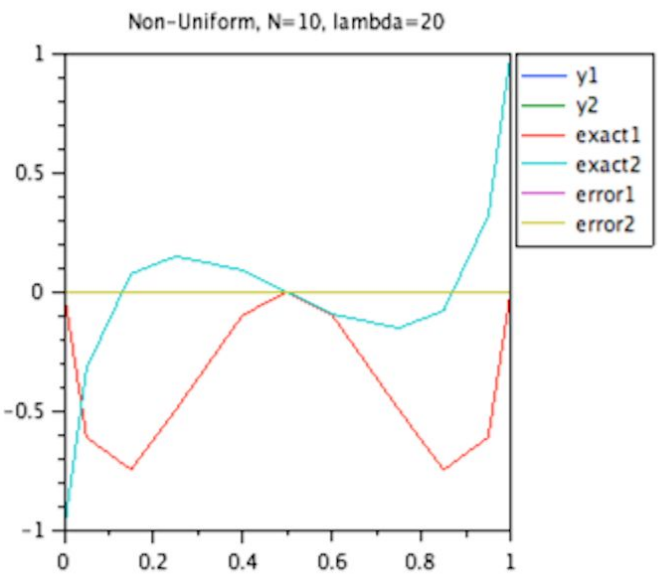
λ	N	y_1 Max Err	y_2 Max Err	y_1 Order	y_2 Order	Time
20	10	4.513D-08	4.407D-08	-0.1738322	-0.0650013	0.006403
20	20	5.090D-08	4.610D-08	-0.5915992	-0.6455397	0.009997
20	40	7.671D-08	7.211D-08	-1.491159	-1.6390234	0.015104
20	80	0.0000002	0.0000002	0	0	0.026876
50	10	5.701D-08	5.717D-08	-0.0313320	-0.0387727	0.008094
50	20	5.826D-08	5.872D-08	-0.3808679	-0.3914130	0.011827
50	40	7.587D-08	7.703D-08	-1.5859188	-1.5807249	0.017068
50	80	0.0000002	0.0000002	0	0	0.029676
5000	10	0.9993101	0.0003292	1.098D-09	-0.9327334	0.029114
5000	20	0.9993101	0.0006283	-3.627D-09	-0.0000004	0.060942
5000	40	0.9993101	0.0006283	5.440D-09	-0.0000215	0.137791
5000	80	0.9993101	0.0006283	0	0	0.365727

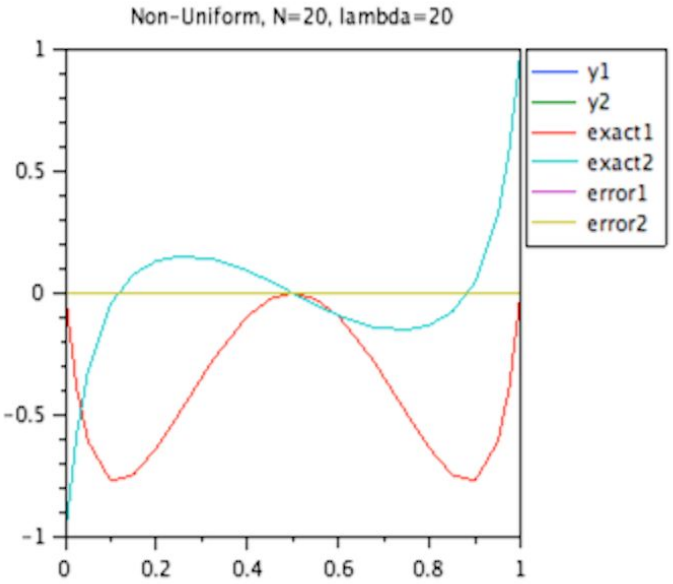
- Error remains unchanged when we increase the lambda from 20-50. However, the error slightly increases on increase of number of intervals i.e. N.
- Error *(especially* $e(y_1)$*)* increases drastically at lambda = 5000. It remains constant at any N values.
- Order is negative in most of the cases because the error increases as N increases.
- The order in case of lambda = 5000 is close to 0 because the error remains constant.

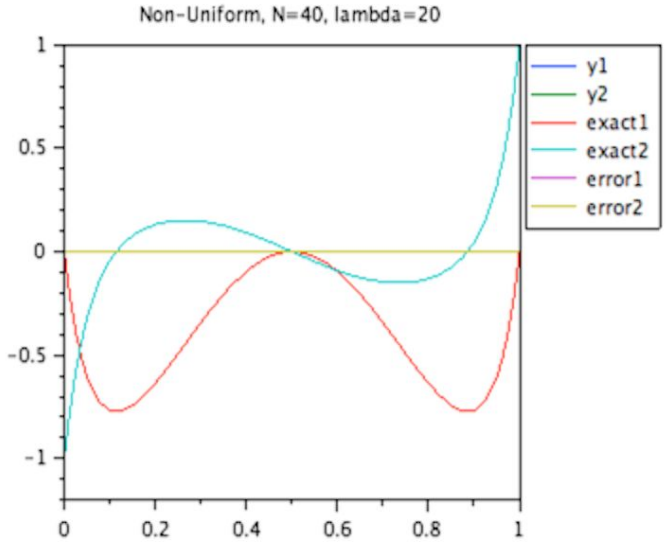


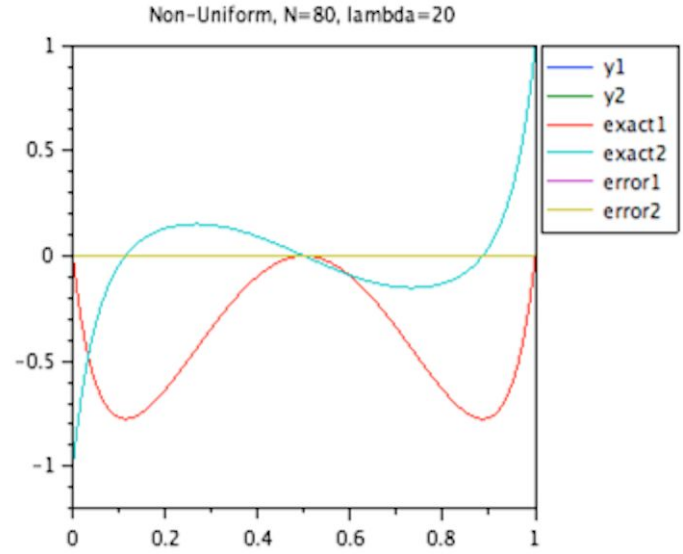
- The error $e(y_1)$ reaches close to 1 at x = 0.5 and then starts decreasing for lambda = 5000. All the graphs for lambda = 5000 look the same because the error at this point is constant across all the N values.
- However, the surprising part is that the error reaches almost 0 at x = 0.5 in the other two cases of lambda i.e. 20 and 50.

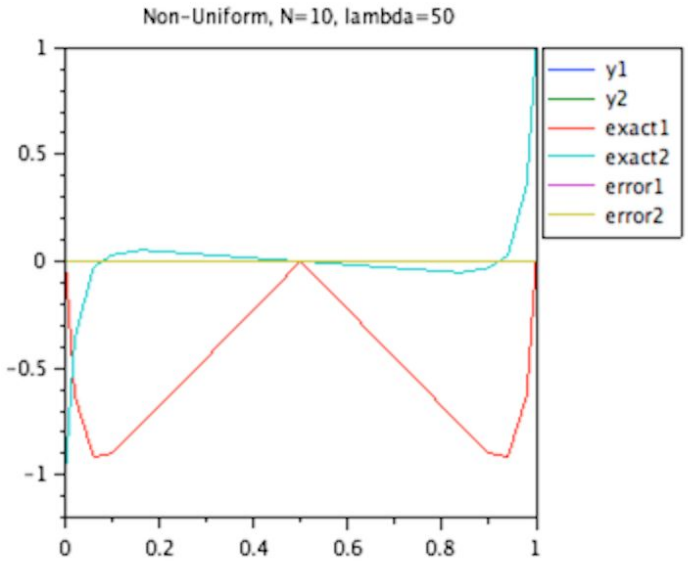
  Following are the plots for all the lambda and N values for Non-Uniform Multiple Shooting Scheme.

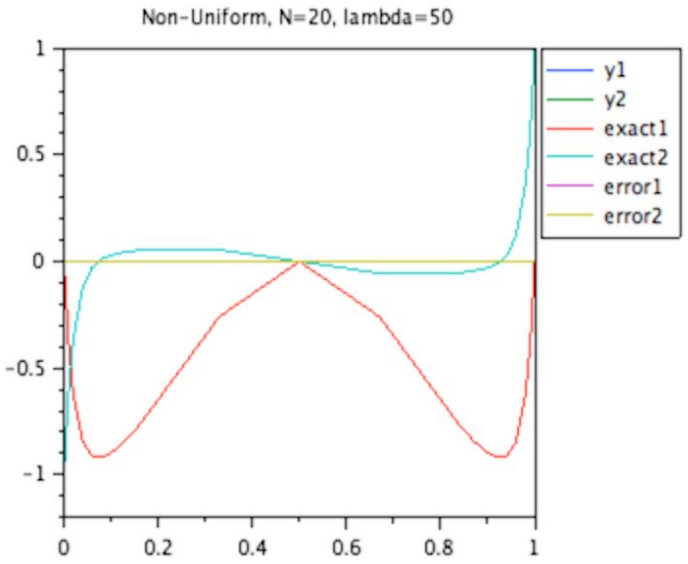


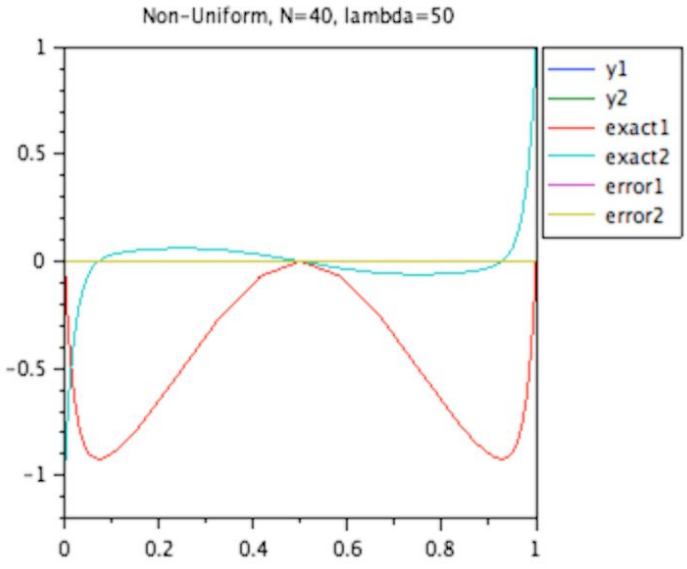


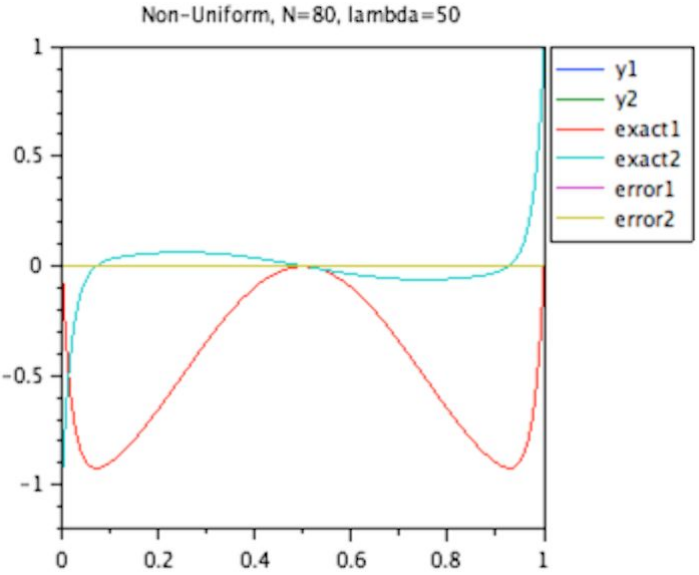


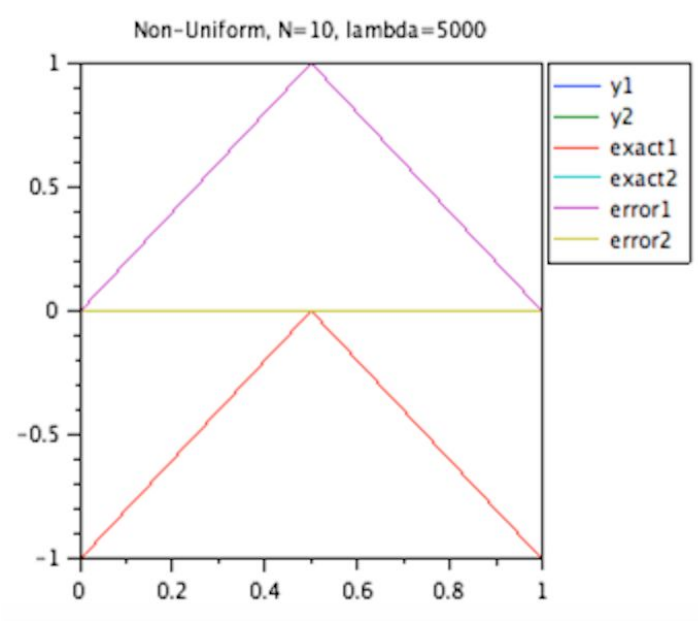


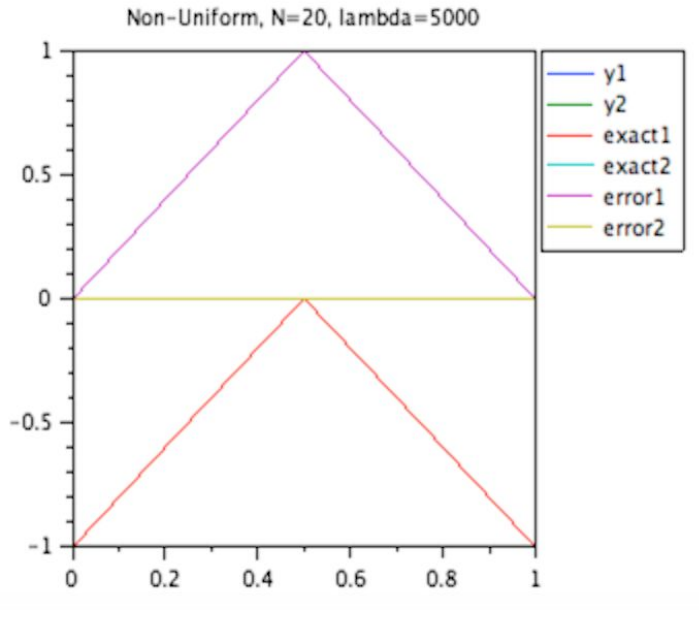

Non-Uniform, N=10, lambda=50
y1
y2
exact1
exact2
error1
error2
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1

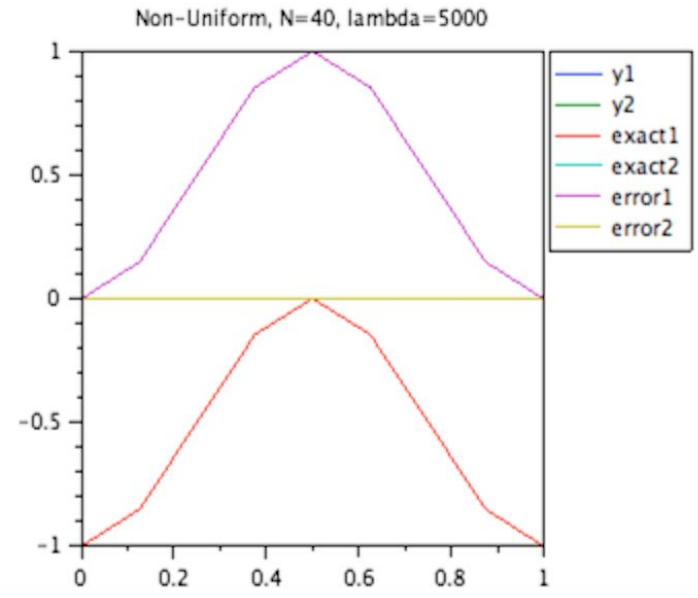

Non-Uniform, N=20, lambda=50
y1
y2
exact1
exact2
error1
error2
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1

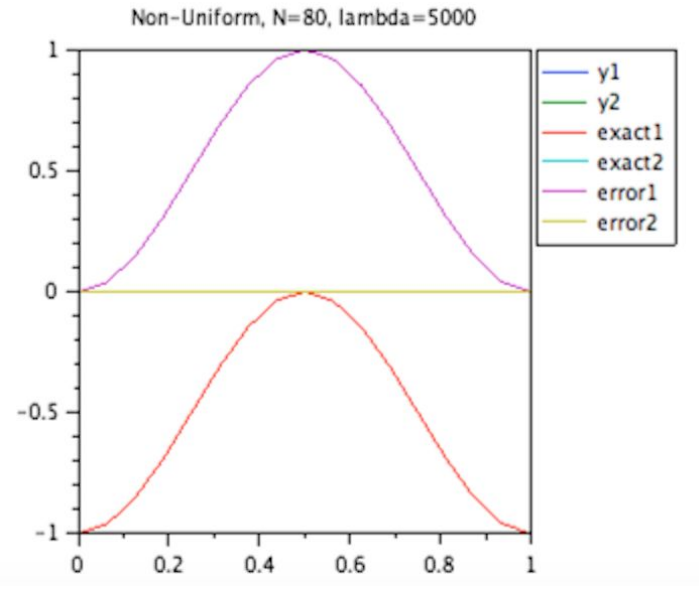

Non-Uniform, N=40, lambda=50
y1
y2
exact1
exact2
error1
error2
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1


Non-Uniform, N=80, lambda=50
y1
y2
exact1
exact2
error1
error2
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1


Non-Uniform, N=10, lambda=5000
y1
y2
exact1
exact2
error1
error2
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1


Non-Uniform, N=20, lambda=5000
y1
y2
exact1
exact2
error1
error2
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1


Non-Uniform, N=40, lambda=5000
y1
y2
exact1
exact2
error1
error2
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1


Non-Uniform, N=80, lambda=5000
y1
y2
exact1
exact2
error1
error2
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1

→ Error Analysis:

ERRORS OF DIFFERENT COMPUTATIONAL METHODS IN A NON UNIFORM MESH

		Midpoint Scheme		Trapezoidal Scheme		Multiple Shooting	
λ	N	y_1 Max Err	y_2 Max Err	y_1 Max Err	y_2 Max Err	y_1 Max Err	y_2 Max Err
20	10	0.0629894	0.0600439	0.0479591	0.0533222	4.513D-08	4.407D-08
20	20	0.0186936	0.0161897	0.0147923	0.0160755	5.090D-08	4.610D-08
20	40	0.0044037	0.0037856	0.0034591	0.0037806	7.671D-08	7.211D-08
20	80	0.0011070	0.0009322	0.0008764	0.0009536	0.0000002	0.0000002
50	10	0.1969546	0.0551958	0.0497103	0.0499058	5.701D-08	5.717D-08
50	20	0.1002651	0.0261904	0.0153138	0.0153618	5.826D-08	5.872D-08
50	40	0.0181925	0.0061945	0.0035870	0.0036007	7.587D-08	7.703D-08
50	80	0.0043959	0.0009864	0.0009082	0.0009114	0.0000002	0.0000002
5000	10	0.0497875	0.0497871	0.0497871	0.0497871	0.9993101	0.0003292
5000	20	0.4095318	0.0153343	0.0153352	0.0153354	0.9993101	0.0006283
5000	40	0.0813026	0.0035916	0.0035927	0.0035937	0.9993101	0.0006283
5000	80	0.0190847	0.0009084	0.0009092	0.0009101	0.9993101	0.0006283

- Trapezoidal Scheme seems like the best method for non-uniform meshes. The error doesn't seem to change on the increasing values of lambda and decreases as we increase N.
- If we only had to take lambda 20 and 50, multiple shooting would be the best method since the error is close to 0. However in multiple shooting, error increases (slightly) on increase of N.
- The midpoint scheme has the worst errors for lambda = 20 and 50. The error for e(y1) reaches quite high at lambda = 5000 and N = 20.