Advanced Numerical Techniques Lab 5

February 11, 2019

1 Lab 5

1.1 Quasilinearization Technique

The non-linear ODE/PDE is linearized iteratively.

Non-linear BVP is

$$F(y'', y', y, x) = 0 y(a) \ = \ y_a \ , \ y(b) \ = \ y_b \qquad a < x < b$$

Treat F as a function of y″,y′ and y at any x. At every iteration F is reduced to a linear form. At (k+1)th iteration expand F about the known form of y″, y′, y evaluated at the kth iteraton i.e. $y''^{(k)}, y'^{(k)}, y^{(k)}$.

Expand $F(y'', y', y, x) = 0$ about $y''^{(k)}, y'^{(k)}, y^{(k)}$ by the taylor series expansion.

1.1.1 Question 5a

$$y'' + (y')^2 - y^2 + y + 1 = 0 y(0) = \frac{1}{2} \ , \ y(\pi) \ = \ -\frac{1}{2}$$

```
In [82]: import numpy as np
         import pandas as pd
```

```
In [83]: x1 = 0
         x2 = np.pi
         y1 = 0.5
         y2 = -0.5
         ep = 0.0001
```

```
In [84]: def thomas_(a,b,c,d):
             c_ = np.zeros(c.size)
             d_ = np.zeros(d.size)

             c_[0] = c[0]/b[0]
             d_[0] = d[0]/b[0]

             for i in range(1, c.shape[0]-1):
                 c_[i] = c[i]/(b[i] - a[i]*c_[i-1])

             for i in range(1, d.shape[0]):
```

```
        d_[i] = (d[i] - a[i]*d_[i-1])/(b[i] - a[i]*c_[i-1])

    return [c_, d_]

def mod(a):
    if a>0:
        return a
    return -1*a

def main_(n=3):
    h = np.pi/n

    y = np.zeros(n+1)
    x_f = np.zeros(n+1)



    for i in range(n+1):
        x_f[i] = (i)*h
        y[i] = 0.5 - np.sin(i*h/2)


    flag = 0
    while flag!=1:
        a = np.zeros(n-1)
        b = np.zeros(n-1)
        c = np.zeros(n-1)
        d = np.zeros(n-1)
        res = np.zeros(n-1)


        for i in range(n-1):
            a[i] = ( 1/(h*h) + (1/(2*h*h))*(y[i+2]-y[i]) )

        for i in range(n-1):
            b[i] = ( -2/(h*h) - 2*y[i+1] + 1 )

        for i in range(n-1):
            c[i] = ( 1/(h*h) - (1/(2*h*h))*(y[i+2]-y[i]) )

        for i in range(n-1):
            d[i] = ( -1*((y[i+2]-y[i])/(2*h)) * ((y[i+2]-y[i])/(2*h)) - y[i+1]*y[i+1] -

        d[0] = d[0] - 0.5 * a[0]
        d[-1] = d[-1] + 0.5*c[-1]

        c_, d_ = thomas_(a,b,c,d)
```

```
            res[-1] = d_[-1]
            for i in range(n-2):
                res[n-3-i] = d_[n-3-i] - res[n-2-i]*c_[n-3-i]

#             print(res)
#             print(y)

            flag=1
            for i in range(n-1):
                if mod(y[i+1]-res[i]) > ep:
                    flag=0

            for i in range(1,n):
                y[i] = res[i-1]


        return [y, x_f]
```

In [85]:
```
a_1, x_1 = main_(3)
a_2, x_2 = main_(7)
a_3, x_3 = main_(13)
a_4, x_4 = main_(30)

print(np.pi/3, np.pi/7, np.pi/13, np.pi/30)
```

```
1.0471975511965976 0.4487989505128276 0.241660973353061 0.10471975511965977
```

In [86]:
```
import pandas as pd
print(pd.DataFrame(np.column_stack((x_1, a_1)), columns=["x", "predicted"]))
print()
print(pd.DataFrame(np.column_stack((x_2, a_2)), columns=["x", "predicted"]))
print()
print(pd.DataFrame(np.column_stack((x_3, a_3)), columns=["x", "predicted"]))
print()
print(pd.DataFrame(np.column_stack((x_4, a_4)), columns=["x", "predicted"]))
```

```
          x  predicted
0  0.000000   0.500000
1  1.047198  -0.238223
2  2.094395  -1.105326
3  3.141593  -0.500000

          x  predicted
0  0.000000   0.500000
1  0.448799   0.100454
2  0.897598  -0.341629
3  1.346397  -0.723208
4  1.795196  -0.959690
```

5	2.243995	-0.999674
6	2.692794	-0.834517
7	3.141593	-0.500000

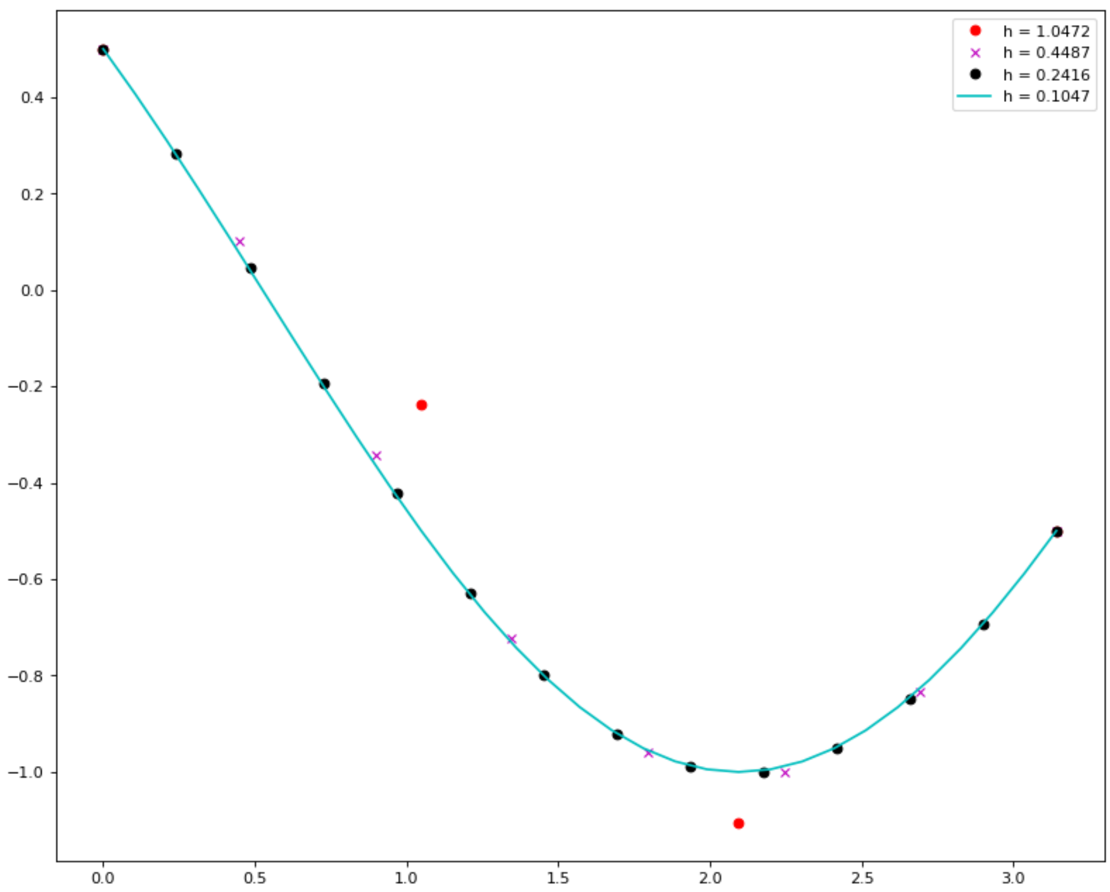
	x	predicted
0	0.000000	0.500000
1	0.241661	0.283144
2	0.483322	0.047274
3	0.724983	-0.192958
4	0.966644	-0.422884
5	1.208305	-0.628620
6	1.449966	-0.797822
7	1.691627	-0.920378
8	1.933288	-0.988979
9	2.174949	-0.999536
10	2.416610	-0.951422
11	2.658271	-0.847503
12	2.899932	-0.693973
13	3.141593	-0.500000

	x	predicted
0	0.000000	0.500000
1	0.104720	0.407202
2	0.209440	0.309832
3	0.314159	0.208974
4	0.418879	0.105750
5	0.523599	0.001304
6	0.628319	-0.103206
7	0.733038	-0.206626
8	0.837758	-0.307812
9	0.942478	-0.405647
10	1.047198	-0.499052
11	1.151917	-0.586998
12	1.256637	-0.668515
13	1.361357	-0.742704
14	1.466077	-0.808750
15	1.570796	-0.865925
16	1.675516	-0.913598
17	1.780236	-0.951246
18	1.884956	-0.978455
19	1.989675	-0.994924
20	2.094395	-1.000472
21	2.199115	-0.995038
22	2.303835	-0.978682
23	2.408554	-0.951585
24	2.513274	-0.914044
25	2.617994	-0.866472
26	2.722714	-0.809394

```
27  2.827433  -0.743437
28  2.932153  -0.669328
29  3.036873  -0.587883
30  3.141593  -0.500000
```

In [87]:
```python
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
figure(num=None, figsize=(12, 10), dpi=80, facecolor='w', edgecolor='k')

plt.plot(x_1, a_1, 'ro', label = 'h = 1.0472')
plt.plot(x_2, (a_2),  'mx', label = 'h = 0.4487')
plt.plot(x_3, (a_3),  'ko', label = 'h = 0.2416')
plt.plot(x_4, (a_4), 'c-', label = 'h = 0.1047')
plt.legend(loc='best')
plt.show()
```



1.2 Question 5b. Solve by Quasilinearization technique

$$f''' + ff' + (f')^2 = 0$$

$$f(0) = 0 f''(0) = 0 f'(10) = 10$$

In [88]:
```
x1 = 0
x2 = 10
y1_ = 0
y2_ = 0
```

In [114]:
```
def thomas_(a,b,c,d):
    c_ = np.zeros(c.shape)
    d_ = np.zeros(d.shape)

    c_[0] = np.dot(np.linalg.inv(b[0]), c[0] )
    d_[0] = np.dot(np.linalg.inv(b[0]), d[0] )

    for i in range(1, c.shape[0]-1):
        c_[i] = np.dot( np.linalg.inv(b[i] - np.dot(a[i], c_[i-1])), c[i] )

    for i in range(1, d.shape[0]):
        d_[i] = np.dot( np.linalg.inv(b[i] - np.dot(a[i], c_[i-1])), d[i] - np.dot(a[i

    return [c_, d_]

def mod(a):
    if a>0:
        return a
    return -1*a

def main_(n=3):
    e = 0.0001
    h = (x2-x1)/n

    a = np.zeros((n-1,2,2))
    b = np.zeros((n-1,2,2))
    c = np.zeros((n-1,2,2))
    d = np.zeros((n-1,2))
    x_f = np.zeros((n+1))

    f = np.zeros(n+1)
    F = np.zeros(n+1)
    for i in range(n+1):
        x_f[i] = x1 + i*h
        f[i] = 5*x_f[i]*x_f[i] - x_f[i]*x_f[i]*x_f[i]/3
        F[i] = x_f[i]*(10 - x_f[i])
```

```python
#print(x_f)

res_final = np.zeros(n+1)
res_final = f

flag=0
while flag!=1:

    for i in range(0, n-1):
        a[i][0][0] = -1
        a[i][0][1] = -1*h/2
        a[i][1][0] = 0
        a[i][1][1] = 1/(h*h) - f[i+1]/(2*h)

    for i in range(0, n-1):
        b[i][0][0] = 1
        b[i][0][1] = -1*h/2
        b[i][1][0] = (F[i+2]-F[i])/(2*h)
        b[i][1][1] = -2/(h*h) + 2*F[i+1]

    for i in range(0, n-1):
        c[i][0][0] = 0
        c[i][0][1] = 0
        c[i][1][0] = 0
        c[i][1][1] = 1/(h*h) + f[i+1]/(2*h)

    for i in range(0, n-1):
        d[i][0] = 0
        d[i][1] = F[i+1]*F[i+1] + f[i+1]*(F[i+2]-F[i])/(2*h)

    #d[-1][1] = d[-1][1] - c[-1][-1][-1]

    #d[0][1] = d[0][1] + 2*h/3
    b[0][1][1] = b[0][1][1] + (4/3)*(1/(h*h) - f[1]/(2*h))
    c[0][1][1] = c[0][1][1] + (-1/3)*(1/(h*h) - f[1]/(2*h))


    c_,d_ = thomas_(a,b,c,d)

    res = np.zeros((n-1,2))

    res[-1] = d_[-1]
    for i in range(n-2):
        res[n-3-i] = d_[n-3-i] - np.dot(c_[n-3-i], res[n-2-i])
        #print(np.dot(c_[n-3-i],res[n-2-i]))
    #print(res[:,0])

    flag=1
```

```
                for i in range(n-1):
                    if mod(f[i+1]-res[i, 0]) > ep or mod(F[i+1]-res[i,1]) > ep:
                        flag=0

                f[1:-1] = res[:,0]
                F[1:-1] = res[:,1]

            return [res_final, x_f]
```

In [131]:
```
a_1, x_1 = main_(3)
a_2, x_2 = main_(6)
a_3, x_3 = main_(11)
a_4, x_4 = main_(13)
a_5, x_5 = main_(17)
a_6, x_6 = main_(23)

print(10/3, 10/6, 10/11, 10/13, 10/17, 10/23)
```

```
3.3333333333333335 1.6666666666666667 0.9090909090909091 0.7692307692307693 0.5882352941176471 0
```

In [129]:
```
import pandas as pd
print(pd.DataFrame(np.column_stack((x_1, a_1)), columns=["x", "predicted"]))
print()
print(pd.DataFrame(np.column_stack((x_2, a_2)), columns=["x", "predicted"]))
print()
print(pd.DataFrame(np.column_stack((x_3, a_3)), columns=["x", "predicted"]))
print()
print(pd.DataFrame(np.column_stack((x_4, a_4)), columns=["x", "predicted"]))
print()
print(pd.DataFrame(np.column_stack((x_5, a_5)), columns=["x", "predicted"]))
print()
print(pd.DataFrame(np.column_stack((x_6, a_6)), columns=["x", "predicted"]))
```

```
           x   predicted
0   0.000000    0.000000
1   3.333333    0.077502
2   6.666667    0.184777
3  10.000000  166.666667

           x   predicted
0   0.000000    0.000000
1   1.666667    0.050067
2   3.333333    0.138634
3   5.000000    0.202033
4   6.666667    0.240383
5   8.333333    0.259153
6  10.000000  166.666667
```

	x	predicted
0	0.000000	0.000000
1	0.909091	0.034610
2	1.818182	0.100662
3	2.727273	0.158939
4	3.636364	0.207399
5	4.545455	0.245562
6	5.454545	0.274123
7	6.363636	0.294423
8	7.272727	0.308003
9	8.181818	0.316324
10	9.090909	0.320631
11	10.000000	166.666667

	x	predicted
0	0.000000	0.000000
1	0.769231	0.031227
2	1.538462	0.091483
3	2.307692	0.146241
4	3.076923	0.193786
5	3.846154	0.233366
6	4.615385	0.265065
7	5.384615	0.289541
8	6.153846	0.307759
9	6.923077	0.320771
10	7.692308	0.329576
11	8.461538	0.335054
12	9.230769	0.337936
13	10.000000	166.666667

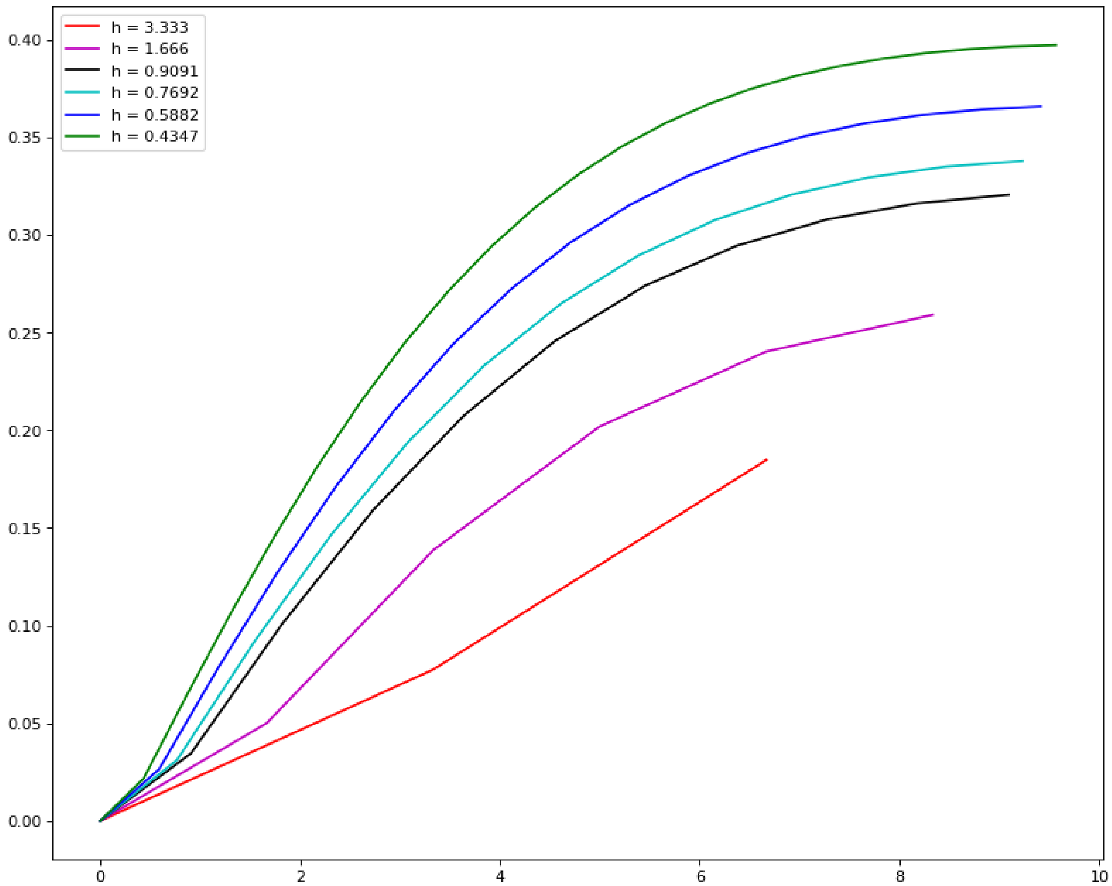
	x	predicted
0	0.000000	0.000000
1	0.588235	0.026417
2	1.176471	0.078039
3	1.764706	0.126560
4	2.352941	0.170821
5	2.941176	0.210079
6	3.529412	0.244002
7	4.117647	0.272621
8	4.705882	0.296231
9	5.294118	0.315298
10	5.882353	0.330373
11	6.470588	0.342022
12	7.058824	0.350788
13	7.647059	0.357160
14	8.235294	0.361569
15	8.823529	0.364377
16	9.411765	0.365890

```
17  10.000000  166.666667

            x   predicted
0    0.000000    0.000000
1    0.434783    0.021793
2    0.869565    0.064766
3    1.304348    0.106144
4    1.739130    0.145257
5    2.173913    0.181579
6    2.608696    0.214749
7    3.043478    0.244568
8    3.478261    0.270984
9    3.913043    0.294072
10   4.347826    0.313996
11   4.782609    0.330984
12   5.217391    0.345304
13   5.652174    0.357236
14   6.086957    0.367062
15   6.521739    0.375048
16   6.956522    0.381443
17   7.391304    0.386473
18   7.826087    0.390337
19   8.260870    0.393210
20   8.695652    0.395245
21   9.130435    0.396573
22   9.565217    0.397304
23  10.000000  166.666667
```

In [132]:
```python
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
figure(num=None, figsize=(12, 10), dpi=80, facecolor='w', edgecolor='k')

plt.plot(x_1[:-1], a_1[:-1], 'r-', label = 'h = 3.333')
plt.plot(x_2[:-1], (a_2[:-1]),  'm-', label = 'h = 1.666')
plt.plot(x_3[:-1], (a_3[:-1]),  'k-', label = 'h = 0.9091')
plt.plot(x_4[:-1], (a_4[:-1]), 'c-', label = 'h = 0.7692')
plt.plot(x_5[:-1], (a_5[:-1]), 'b-', label = 'h = 0.5882')
plt.plot(x_6[:-1], (a_6[:-1]), 'g-', label = 'h = 0.4347')
plt.legend(loc='best')
plt.savefig('books_read.png')
plt.show()
```



h = 3.333
h = 1.666
h = 0.9091
h = 0.7692
h = 0.5882
h = 0.4347
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
2
4
6
8
10


```
In [ ]:
```

```
In [ ]:
```