16811 A4

November 2022

1.

(a)

$$\frac{dy}{dx} = \frac{1}{3y^2} \tag{1}$$

$$3y^2 dy = dx \tag{2}$$

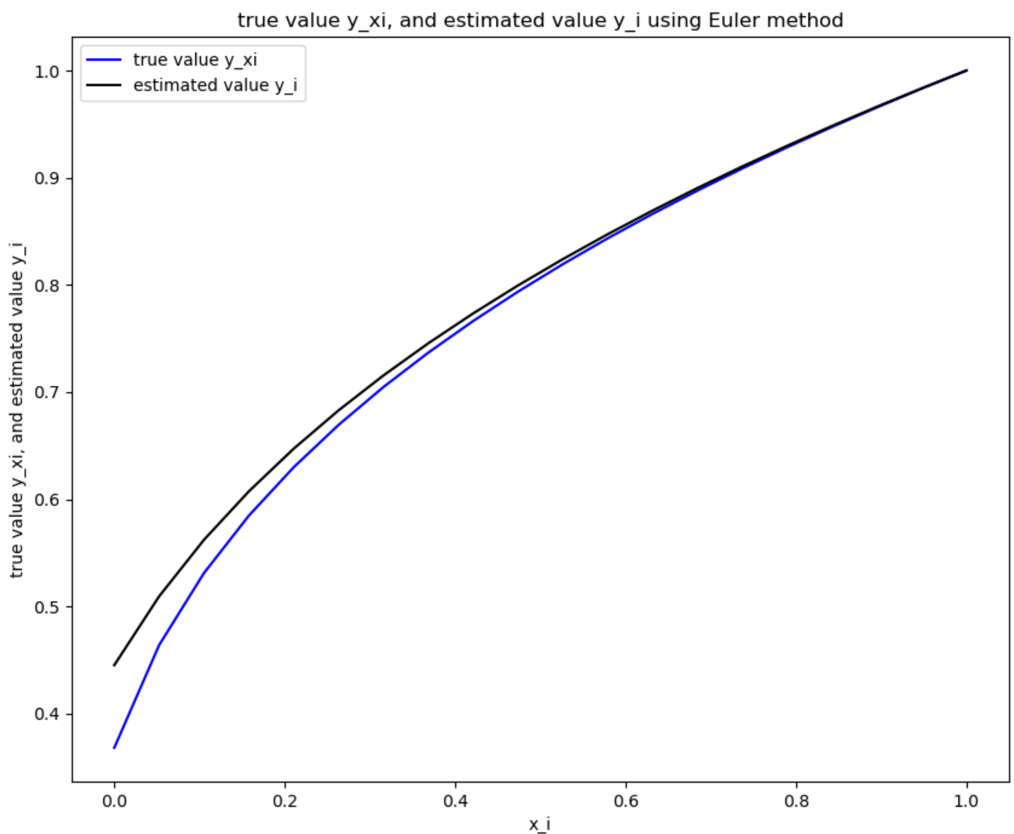
$$y^3 - x = C \tag{3}$$

Since $y(1) = 1$, then we can get $c = 0$.

$$y(x) = \sqrt[3]{x}$$

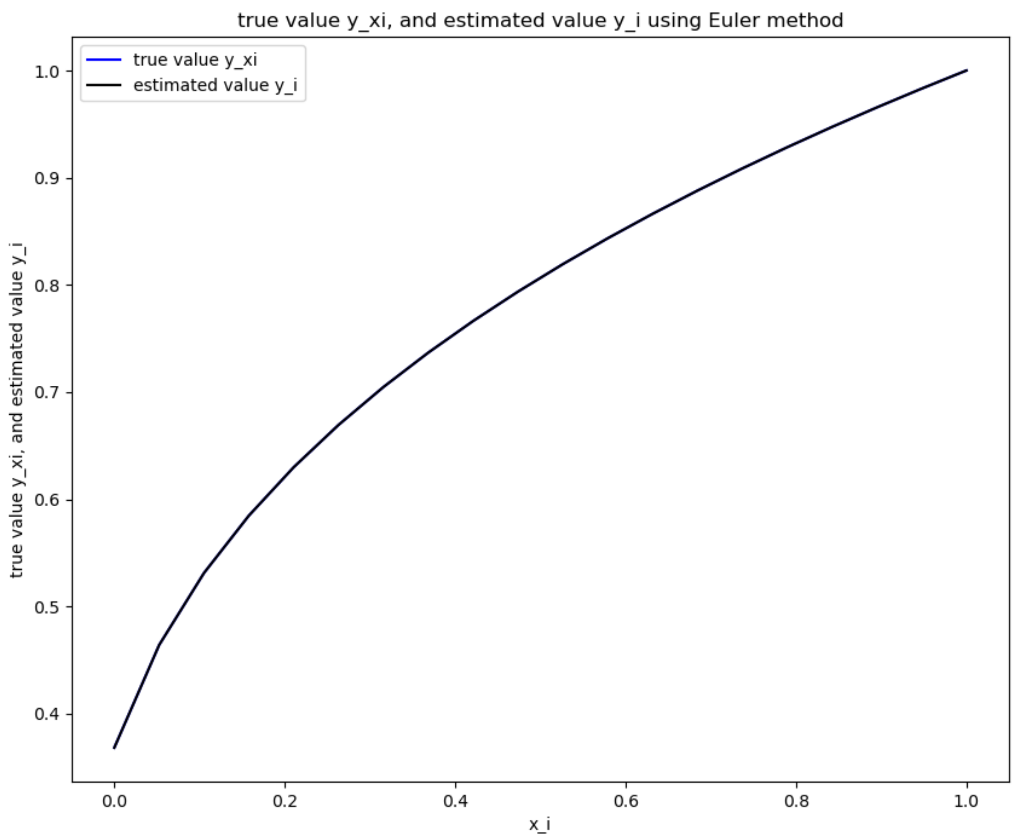
(b) Run *python q1.py* to get the estimated values using Euler's method, and also the plot and printed table. The average error between the true values $y(x_i)$ and estimated values y_i is around 0.01360.

	x_i	true value y(x_i)	estimated value y_i	error abs(y(x_i)-y_i)
0	0.000000	0.368403	0.445424	0.077021
1	0.052632	0.464159	0.509602	0.045443
2	0.105263	0.531329	0.562312	0.030983
3	0.157895	0.584804	0.607476	0.022672
4	0.210526	0.629961	0.647258	0.017298
5	0.263158	0.669433	0.682988	0.013555
6	0.315789	0.704730	0.715540	0.010810
7	0.368421	0.736806	0.745526	0.008720
8	0.421053	0.766309	0.773391	0.007081
9	0.473684	0.793701	0.799467	0.005766
10	0.526316	0.819321	0.824013	0.004692
11	0.578947	0.843433	0.847232	0.003799
12	0.631579	0.866239	0.869288	0.003049
13	0.684211	0.887904	0.890314	0.002410
14	0.736842	0.908560	0.910422	0.001862
15	0.789474	0.928318	0.929704	0.001386
16	0.842105	0.947268	0.948240	0.000972
17	0.894737	0.965489	0.966097	0.000608
18	0.947368	0.983048	0.983333	0.000286
19	1.000000	1.000000	1.000000	0.000000


true value y_xi, and estimated value y_i using Euler method
true value y_xi
estimated value y_i
true value y_xi, and estimated value y_i
x_i
0.0
0.2
0.4
0.6
0.8
1.0
0.4
0.5
0.6
0.7
0.8
0.9
1.0

(c) Run *python q1.py* to get the estimated values using fourth-order Runge-Kutta's method, and also the plot and printed table.
The average error between the true values $y(x_i)$ and estimated values y_i is around $1.824356e-06$.

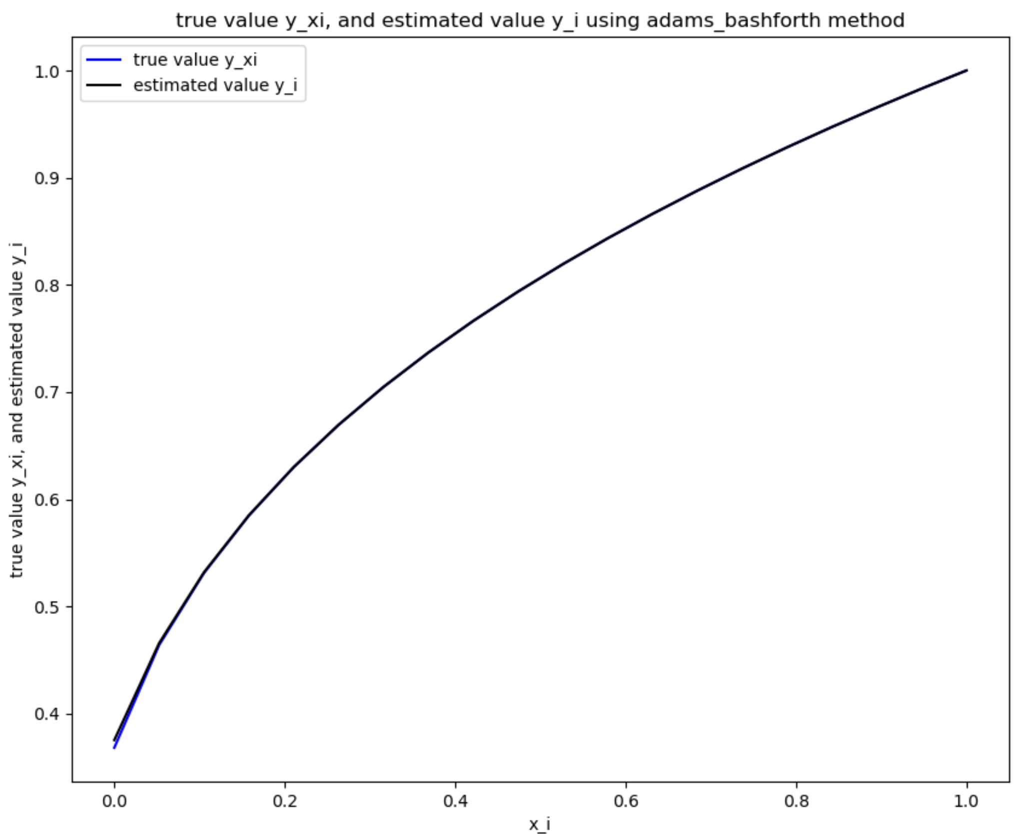
	x_i	true value y(x_i)	estimated value y_i	error abs(y(x_i)-y_i)
0	0.000000	0.368403	0.368374	2.873778e-05
1	0.052632	0.464159	0.464155	3.994514e-06
2	0.105263	0.531329	0.531328	1.079881e-06
3	0.157895	0.584804	0.584803	4.081167e-07
4	0.210526	0.629961	0.629960	1.878323e-07
5	0.263158	0.669433	0.669433	9.825756e-08
6	0.315789	0.704730	0.704730	5.616039e-08
7	0.368421	0.736806	0.736806	3.420962e-08
8	0.421053	0.766309	0.766309	2.183435e-08
9	0.473684	0.793701	0.793701	1.442117e-08
10	0.526316	0.819321	0.819321	9.760490e-09
11	0.578947	0.843433	0.843433	6.712902e-09
12	0.631579	0.866239	0.866239	4.654461e-09
13	0.684211	0.887904	0.887904	3.226010e-09
14	0.736842	0.908560	0.908560	2.211922e-09
15	0.789474	0.928318	0.928318	1.478006e-09
16	0.842105	0.947268	0.947268	9.381146e-10
17	0.894737	0.965489	0.965489	5.354228e-10
18	0.947368	0.983048	0.983048	2.315458e-10
19	1.000000	1.000000	1.000000	0.000000e+00



true value y_xi, and estimated value y_i using Euler method
true value y_xi
estimated value y_i
true value y_xi, and estimated value y_i
x_i
0.0
0.2
0.4
0.6
0.8
1.0
0.4
0.5
0.6
0.7
0.8
0.9
1.0

(d) Run *python q1.py* to get the estimated values using Adams-Bashforth's method, and also the plot and printed table.
The average error between the true values $y(x_i)$ and estimated values y_i is around 0.000544460.

	x_i	true value y(x_i)	estimated value y_i	error abs(y(x_i)-y_i)
0	0.000000	0.368403	0.375509	7.106033e-03
1	0.052632	0.464159	0.465967	1.808299e-03
2	0.105263	0.531329	0.532009	6.792772e-04
3	0.157895	0.584804	0.585118	3.140695e-04
4	0.210526	0.629961	0.630126	1.654727e-04
5	0.263158	0.669433	0.669528	9.531883e-05
6	0.315789	0.704730	0.704788	5.853453e-05
7	0.368421	0.736806	0.736844	3.767943e-05
8	0.421053	0.766309	0.766335	2.511883e-05
9	0.473684	0.793701	0.793718	1.718142e-05
10	0.526316	0.819321	0.819333	1.196591e-05
11	0.578947	0.843433	0.843441	8.426712e-06
12	0.631579	0.866239	0.866245	5.959506e-06
13	0.684211	0.887904	0.887908	4.200090e-06
14	0.736842	0.908560	0.908563	2.920944e-06
15	0.789474	0.928318	0.928320	1.975377e-06
16	0.842105	0.947268	0.947270	1.267191e-06
17	0.894737	0.965489	0.965490	7.319590e-07
18	0.947368	0.983048	0.983048	3.132697e-07
19	1.000000	1.000000	1.000000	0.000000e+00


true value y_xi, and estimated value y_i using adams_bashforth method
true value y_xi
estimated value y_i
true value y_xi, and estimated value y_i
x_i
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.0
0.2
0.4
0.6
0.8
1.0

2.

(a)

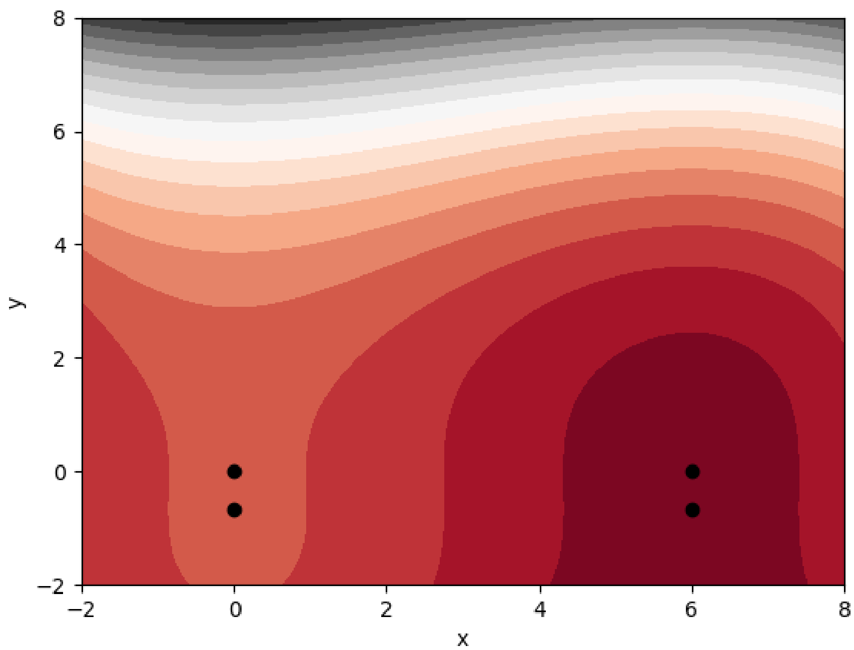
$$\frac{\partial f}{\partial x} = 3x^2 - 18x = 0 \tag{4}$$

$$x_1 = 6, x_2 = 0 \tag{5}$$

$$\frac{\partial f}{\partial y} = 3y^2 + 2y = 0 \tag{6}$$

$$y_1 = 0, y_2 = \frac{-2}{3} \tag{7}$$

Hessian $= \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix}$, D $= \frac{\partial^2 f}{\partial x^2}\frac{\partial^2 f}{\partial y^2} - (\frac{\partial^2 f}{\partial x \partial y})^2 = (6x - 18)(6y + 2)$. The four critical points:



$$(6, 0) : D > 0, f_{xx} > 0 \Rightarrow \text{local minimum}$$
$$(6, \frac{-2}{3}) : D < 0, f_{xx} > 0 \Rightarrow \text{saddle point}$$
$$(0, 0) : D < 0, f_{xx} < 0 \Rightarrow \text{saddle point}$$
$$(0, \frac{-2}{3}) : D > 0, f_{xx} < 0 \Rightarrow \text{local maximum}$$

(b)

$$x - t\nabla f(1, 1) = x - t \begin{pmatrix} -15 \\ 5 \end{pmatrix} = \begin{pmatrix} 1 + 15t \\ 1 - 5t \end{pmatrix}$$

$$f(x - t\nabla f(1, 1)) = (1 + 15t)^3 + (1 - 5t)^3 - 9(1 + 15t)^2 + (1 - 5t)^2 + 7$$
$$\nabla f(x - t\nabla f(1, 1)) = 45(1 + 15t)^2 - 15(1 - 5t)^2 - 18 * 15(1 + 15t) - 10(1 - 5t) = 0$$
$$\Longrightarrow t_1 = \frac{1}{3}, t_2 = -\frac{1}{13}.$$

Take $t > 0$, then we get $t = \frac{1}{3}$.
Then $\nabla f(x - t\nabla f(1,1)) = \nabla f(6, -\frac{2}{3}) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$.
Then we only need one step of steepest descent to converge to a saddle point of f. (cannot converge to a overall local minimum of f)

3.

Given two distinct eigenvalues λ_i, λ_j and corresponding eigenvectors v_i, v_j, from the definition of eigenvector, we know that $Qv_i = \lambda_i v_i \Rightarrow v_j^T Q v_i = \lambda_i v_j^T v_i$
$Qv_j = \lambda_j v_j, \Rightarrow v_i^T Q v_j = \lambda_j v_i^T v_j$.
$(v_j^T Q v_i)^T = v_i^T Q^T v_j$
Since $Q = Q^T$, then we can get $(v_j^T Q v_i)^T = v_i^T Q^T v_j \Rightarrow \lambda_i v_j^T v_i = \lambda_j v_i^T v_j \Rightarrow (\lambda_i - \lambda_j) v_j^T v_i = 0$
Since $\lambda_i \neq \lambda_j$, then we get $v_j^T v_i = 0$
Then the two eigenvectors are orthogonal. And $v_j^T Q v_i \lambda_i v_j^T v_i = 0$, thus we can conclude that any two eigenvectors of symmetric positive definite nxn matrix Q corresponding to distinct eigenvalues of Q are Q-orthogonal.

4.

(a) In the purely quadratic form of the conjugate gradient method, we know that d_k, d_{k-1} are Q-orthogonal. This means $d_k^T Q d_{k-1} = 0$. Then we can derive

$$
\begin{aligned}
d_k^T Q d_k &= d_k^T Q(-g_k + \beta_{k-1} d_{k-1}) \\
&= -d_k^T Q g_k + \beta_{k-1}(d_k^T Q d_{k-1}) \\
&= -d_k^T Q g_k
\end{aligned}
$$

To obtain x_{k+1} from x_k,

$$
\begin{aligned}
x_{k+1} &= x_k + \alpha_k d_k \\
&= x_k - \frac{d_k^T g_k}{d_k^T Q d_k} d_k \\
&= x_k + \frac{d_k^T g_k}{d_k^T Q g_k} d_k
\end{aligned}
$$

Therefore, we proved that to obtain x_{k+1} from x_k does not need to use Q explicitly, assuming one already has available the vectors g_k, Q_{gk}, and d_k.

(b) Take a unit step from x_k in the direction of the negative gradient descent, i.e. $y_k = x_k - g_k$. By the definition of $\nabla f(x)$, we know that $p_k = \nabla f(y_k) = Qy_k + b$, then

$$
\begin{aligned}
g_k &= x_k - y_k \\
Qg_k &= Qx_k - Qy_k \\
Qg_k &= (Qx_k + b) - (Qy_k + b) \\
\Rightarrow Qg_k &= g_k - p_k
\end{aligned}
$$

(c) $\alpha_k = -\frac{d_k^T g_k}{d_k^T Q d_k} = \frac{d_k^T g_k}{d_k^T Q g_k} = \frac{d_k^T g_k}{d_k^T (g_k - p_k)}$

$\beta_k = \frac{g_{k+1}^T Q d_k}{d_k^T Q d_k} = -\frac{(Qg_{k+1})^T Q d_k}{d_k^T Q g_k} = -\frac{(g_{k+1} - p_{k+1})^T d_k}{d_k^T (g_k - p_k)}$

The conjugate gradient algorithm combing the results from (a) and (b), assuming results from (a) and (b) carrying over to general functions:

0: $d_0 \leftarrow -\nabla f(x_0)$
0: $g_0 \leftarrow -d_0$
0: **for** $k = 0, 1, .., n-1$ **do**
0: $\quad \alpha_k \leftarrow \frac{d_k^T g_k}{d_k^T (g_k - p_k)}$
0: $\quad x_{k+1} \leftarrow x_k + \alpha_k d_k$
0: $\quad \beta_k \leftarrow -\frac{(g_{k+1} - p_{k+1})^T d_k}{d_k^T (g_k - p_k)}$
0: $\quad d_{k+1} \leftarrow \beta_k d_k - g_{k+1}$
return x_n

This algorithm does not require knowledge of the Hessian of f or a line search.

5.

Define the length and width of the rectangle as x, y. Then we want to maximize $f(x) = xy$ subject to $g(x) = 2x + 2y = c$, where c is the given perimeter.
This is equal to minimize $f(x) = -xy$ subject to $g(x) = 2x + 2y = c$.
Use Lagrange multiplier, let $F(x) = -xy + \lambda(2x + 2y - c)$.
Then $\nabla F(x) = \begin{bmatrix} -y + 2\lambda \\ -x + 2\lambda \\ 2x + 2y - c \end{bmatrix}$
Let $\nabla F(x) = 0$, we get the Lagrange multiplier first-order necessary conditions: $x = y = \frac{c}{4}, \lambda = \frac{c}{8}$.
$\nabla^2 f(x) + \lambda \nabla^2 g(x) = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$
Let $\nabla F(x) + d^T \nabla h(x) = 0$, $\Rightarrow d^T \begin{pmatrix} 2 \\ 2 \end{pmatrix} = 0$, then we get $d = t_1 \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \begin{pmatrix} t_1 \\ -t_1 \end{pmatrix}$
Since $d^T(\nabla^2 f(x) + \lambda \nabla^2 g(x))d = \begin{pmatrix} t_1 & -t_1 \end{pmatrix} \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} t_1 \\ -t_1 \end{pmatrix} = 2t_1^2 \geq 0$, which means it is positive definite, then we have verified the second-order sufficiency conditions.

6.