1. Consider the following limit:

$$\lim_{x\to\infty} \frac{(x+1)(2x+3)}{(x+2)(3x+4)}$$

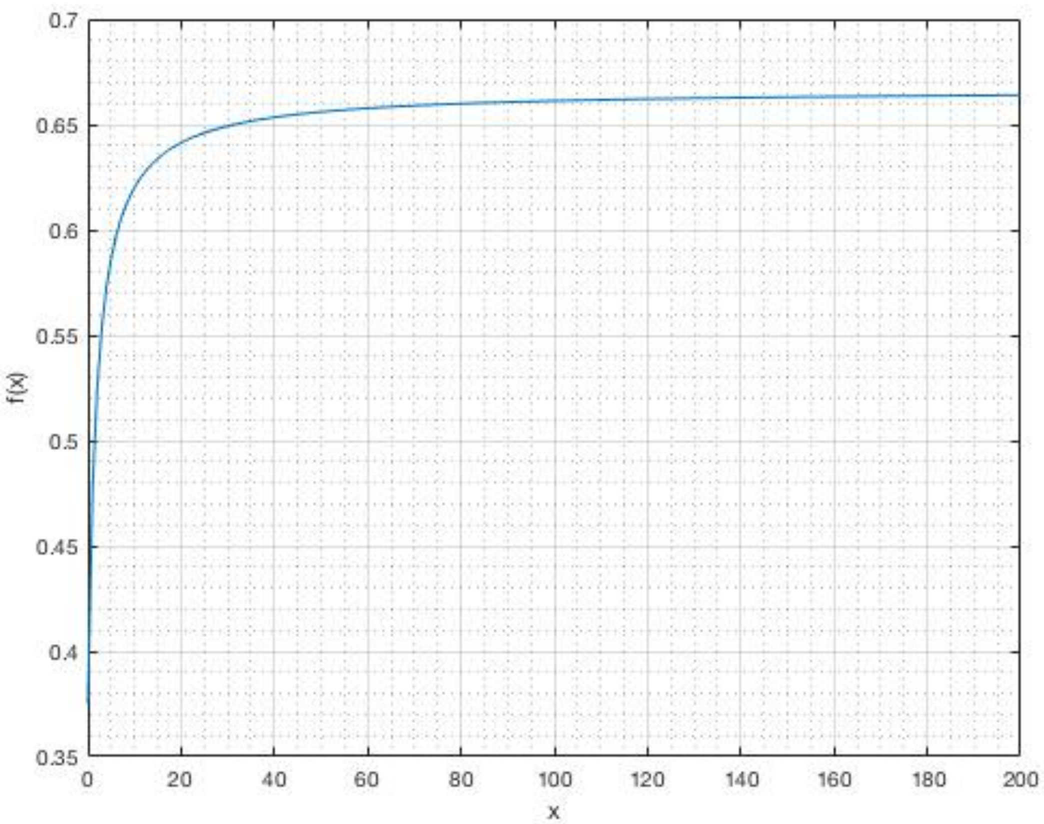
a) Evaluate the limit analytically.

First, I evaluate the limit of numerator and denominator separately. In numerator part, as x keeps increasing to infinity, (x+1).*(2.*x+1) will tend to be 2.*(x.^2). In denominator part, as x keeps increasing to infinity, (x+2).*(3.*x+4) will tend to be 3.*(x.^2). Therefore, the numerator over denominator will equal to (2.*(x.^2))./(3.*(x.^2)) = 2/3 (close to 0.667).

b) Verify (a) by plotting in MATLAB.

From plot1-1, we could see that as the value of x keep increasing (after around 120), the value of f(x) reaches a plateau at a value between 0.66 and 0.67, which verifies the 2/3 analytical evaluation of limit in question (a).

```
x = [0:1:200];
f =@(x) ((x+1).*(2.*x+3))./((x+2).*(3.*x+4));
figure(1)
plot(x,f(x))
grid on
grid minor
xlabel('x')
ylabel('f(x)')
```



plot1-1

2. Consider the following equations describing the trajectory of a point on the *xy*-plane:

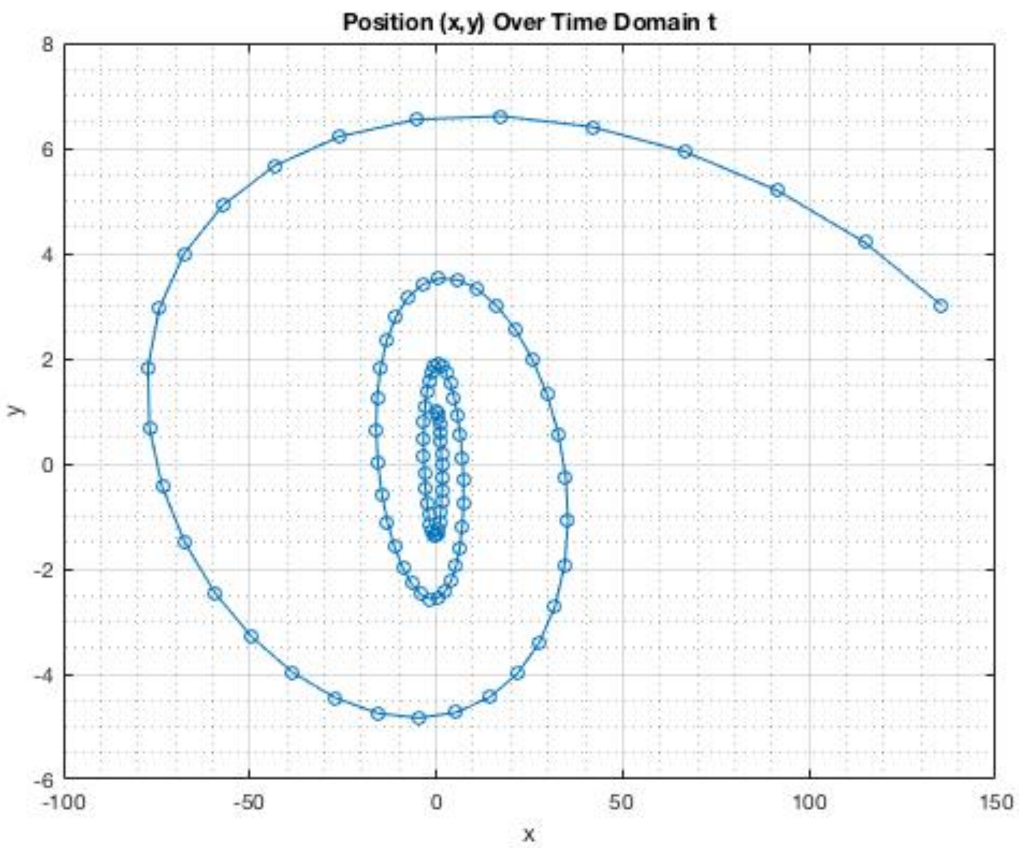
$x = e^{t/2}\sin(2t)$

$y = e^{t/5}\cos(2t)$

a) Plot the position (x, y) over the time domain $0 \le t \le 10$.

See plot2-1.

```
t = [0:0.1:10];
x = exp(t./2).*sin(2.*t);
y = exp(t./5).*cos(2.*t);
figure(1)
plot(x,y,'-o')
xlabel('x')
ylabel('y')
title('Position (x,y) Over Time Domain of t')
grid on
grid minor
```



plot2-1

b) Calculate $u = dx/dt$ and $v = dy/dt$ analytically.

x = exp(t./2).*sin(2*t)

let a = exp(t./2), b = sin(2*t), then x = a.*b

In getting dx/dt, I apply product rule here: u = dx/dt = (a*b)’ = a’.*b + a.*b’= exp(t./2).*sin(2*t)’ + (exp(t./2))’.*sin(2*t) = 2.*cos(2.*t).*exp(t./2) + (1/2).*sin(2.*t).*exp(t./2)

y = exp(t./5).*cos(2.*t)
let c = exp(t./5) , d = cos(2t), then y = c.*d
In getting dy/dt, I apply product rule here: v = dy/dt = (c*d)' = c'.*d + c.*d' = exp(t./5).*(cos(2.*t))' + (exp(t./5))'.*cos(2.*t) = -2.*sin(2.*t).*exp(t./5) + (1/5).*cos(2.*t).*exp(t./5)

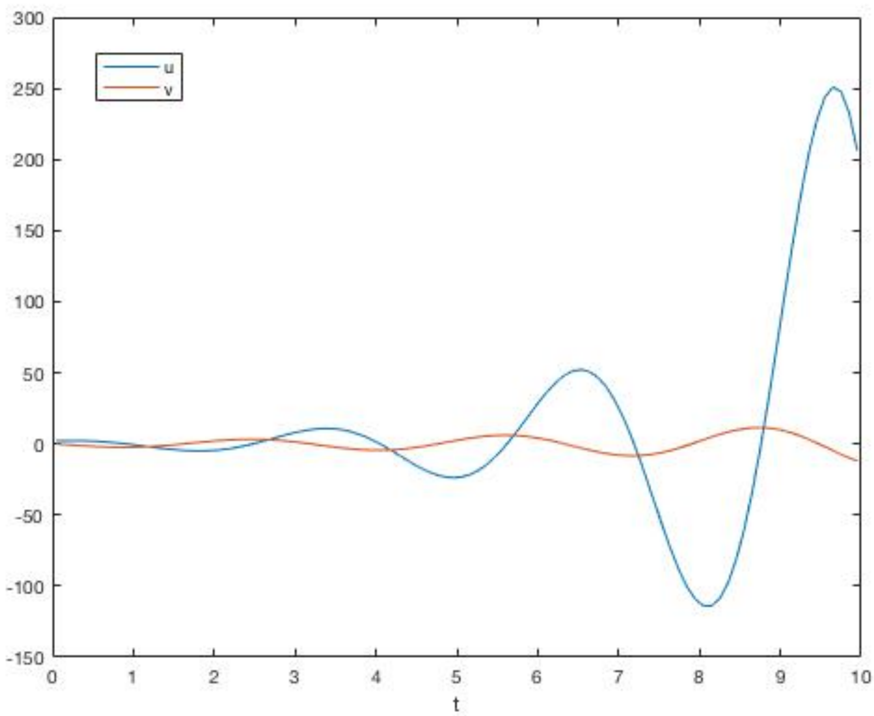
Therefore, the expression of u and v are:

u = 2.*cos(2.*t).*exp(t./2) + (1/2).*sin(2.*t).*exp(t./2)
v = -2.*sin(2.*t).*exp(t./5) + (1/5).*cos(2.*t).*exp(t./5)

c) Calculate u and v numerically in MATLAB over the interval $0 \leq t \leq 10$.

See code as follow and plot2-2.

```
nt = length(t);
dt = t(2:nt)-t(1:nt-1);
dx = x(2:nt)-x(1:nt-1);
u = dx./dt;
dy = y(2:nt)-y(1:nt-1);
v = dy./dt;
tmid = .5*(t(2:nt)+t(1:nt-1));
figure(2)
plot(tmid,u)
hold on
plot(tmid,v)
legend('u','v')
xlabel('t')
```

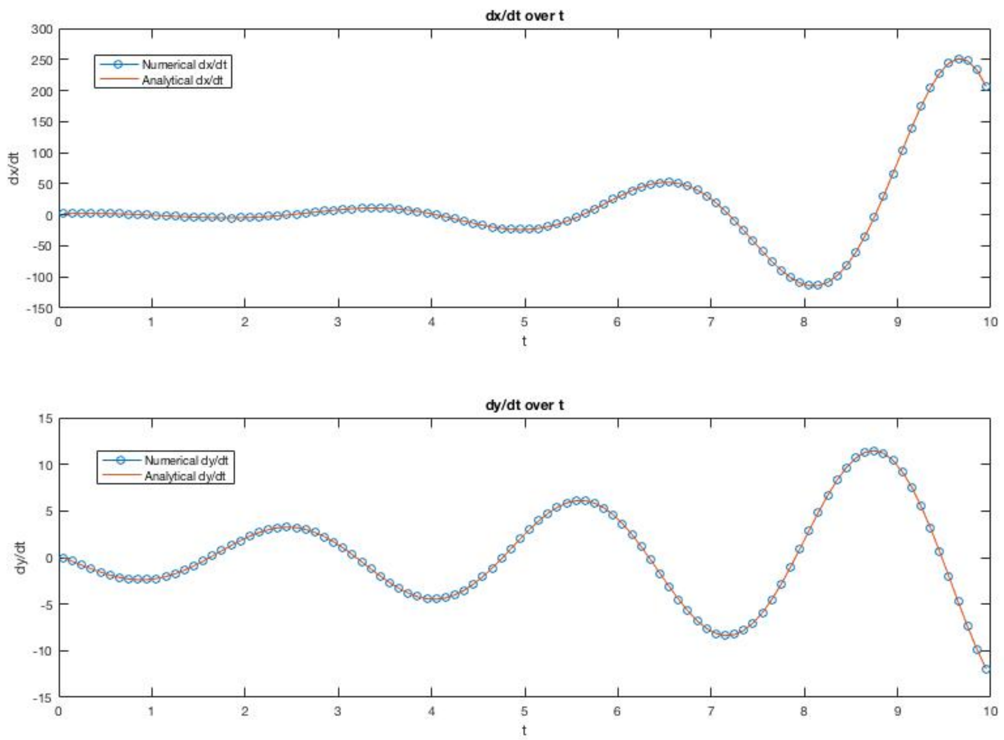


plot2-2

d) Plot your answers from (c) and (b) to compare the numerical and analytical solutions.

From plot2-3, we can know that numerical and analytical solutions to both dx/dt and dy/dt get exactly results.

```
dxadt = 2.*cos(2.*tmid).*exp(tmid./2)+(1./2).*sin(2.*tmid).*exp(tmid./2);
figure(3)
subplot(2,1,1) % dx/dt
plot(tmid,u,'o-')
hold on
plot(tmid,dxadt)
legend('Numerical dx/dt','Analytical dx/dt')
xlabel('t')
ylabel('dx/dt')
title('dx/dt over t')

dyadt = -
2.*sin(2.*tmid).*exp(tmid./5)+(1./5).*cos(2.*tmid).*exp(tmid./5);
subplot(2,1,2)
plot(tmid,v,'o-')
hold on
plot(tmid,dyadt)
legend('Numerical dy/dt','Analytical dy/dt')
xlabel('t')
ylabel('dy/dt')
title('dy/dt over t')
```



plot2-3

3. Consider the following hypothetical data representing the amount of a radioactive element (A, in grams) measured over time (t, in years).

t (yrs)	0	100	200	300	400	500	600	700	800	900	1000
A (g)	535	390	235	166	108	65	47	28	22	12	8

a) Fit a function of the form $A = A_o e^{kt}$ to the above data using the MATLAB function polyfit. What are the values of A_0 and k (express units as well)?

By using polyfit function, I get **k = -0.0042 (years^{-1})**, ln(A0) = 6.3393, so **A0 = 566.3997(g)**.
Therefore:
A = 566.3997.*exp(-0.0042.*t)

```
t_r = [0, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000]; % unit yrs
A_r = [535, 390, 235, 166, 108, 65, 47, 28, 22, 12, 8]; % unit g

P = polyfit(t_r,log(A_r),1);
```

b) What is the half-life of this element? Predict how much material will be remaining after 2000 years.

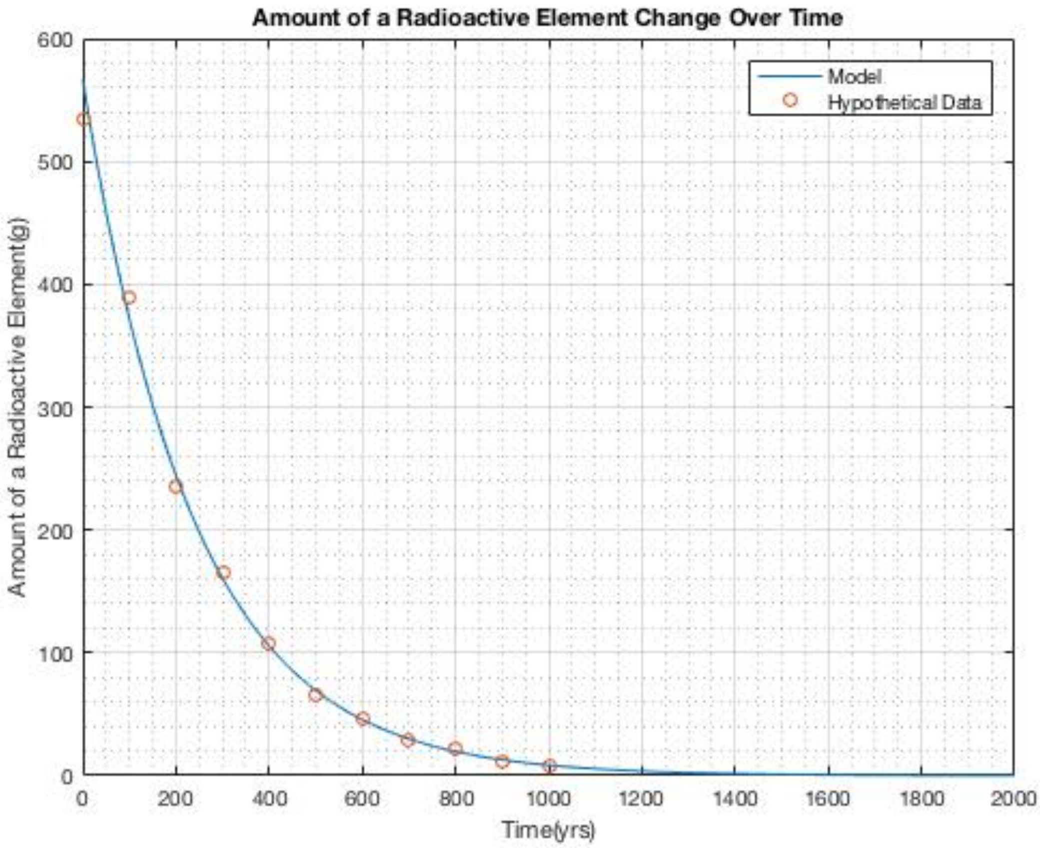
In function A = 566.3997.*exp(-0.0042.*t), as k= -0.0042 < 0, we know this is an exponential decay model. Based on this, I wrote the following code:

```
A(t2)/A(t1) = 1/2;
(566.3997.*exp(-0.0042.*t2))./(566.3997.*exp(-0.0042.*t1))= 1/2
t2_t1 = log(1/2)./(-0.0042) %= 165.0350;
t = [0:1:2000];
A = 566.3997.*exp(-0.0042.*t);
figure(1)
plot(t,A)
hold on
plot(t_r,A_r,'o')
grid on
grid minor
legend('Model','Hypothetical Data')
xlabel('Time(yrs)')
ylabel('Amount of a Radioactive Element(g)')
title('Amount of a Radioactive Element Change Over Time')
```

Therefore, the half-life of this element is 165 years which can also be seen approximately from plot3-1. The material will be remaining after 2000 years is calculated by :

```
% material remaining after 2000 years
M = 566.3997.*exp(-0.0042.*2000);
% 0.1274
```

So, there will be 0.1274g material remaining after 2000 years.



Plot3-1

c) Derive expressions for dA/dt and $\int A(t)dt$ analytically from the function you found in (a).

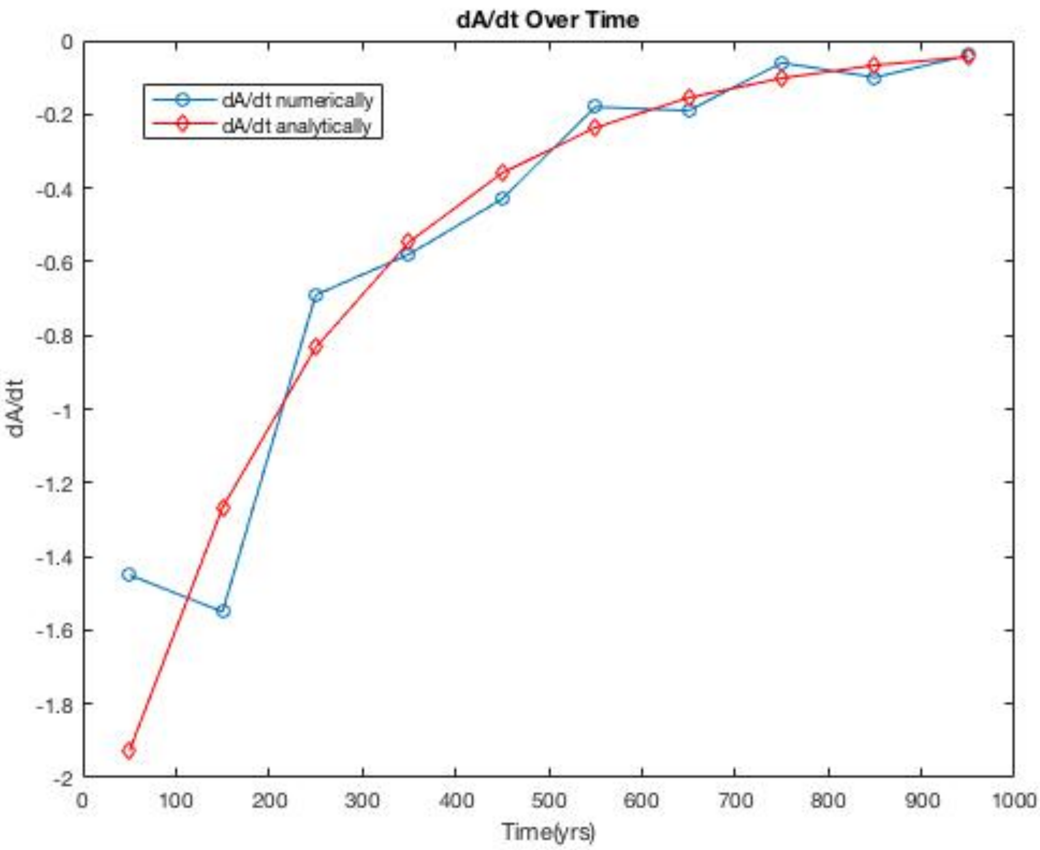
```
A = 566.3997.*exp(-0.0042.*t)
dA/dt = 566.3997.*(-0.0042).*exp(-0.0042.*t)= -2.3789.*exp(-
0.0042.*t)
dA/dt = -2.3789.*exp(-0.0042.*t)
int_A = (566.3997./(-0.0042)).*exp(-0.0042.*t)= -134860.*exp(-
0.0042.*t)+C;
Because when t=0, int_A= A = 535, so C = 135395
Int_A = -134860.*exp(-0.0042.*t)+135395
```

d) Evaluate dA/dt numerically as a function of time from the given data. Plot and compare the numerical value of dA/dt with the analytical expression you derived in (c) over the same time interval.

Plot3-2 compares the numerical value of dA/dt with the analytical expression over the same time interval. I think the analytical expression fit the trend of numerical value of dA/dt pretty well.

```
nt = length(t_r);
dt = t_r(2:nt)-t_r(1:nt-1);
dA = A_r(2:nt)-A_r(1:nt-1);
dAdt = dA./dt;
tmid = .5*(t_r(2:nt)+t_r(1:nt-1));
```

```
dAAdt = -2.3789.*exp(-0.0042.*tmid);% evaluate analytically
figure(2)
plot(tmid,dAdt,'o-')
hold on
plot(tmid,dAAdt,'rd-')
legend('dA/dt numerically','dA/dt analytically')
xlabel('Time(yrs)')
ylabel('dA/dt')
title('dA/dt Over Time')
```



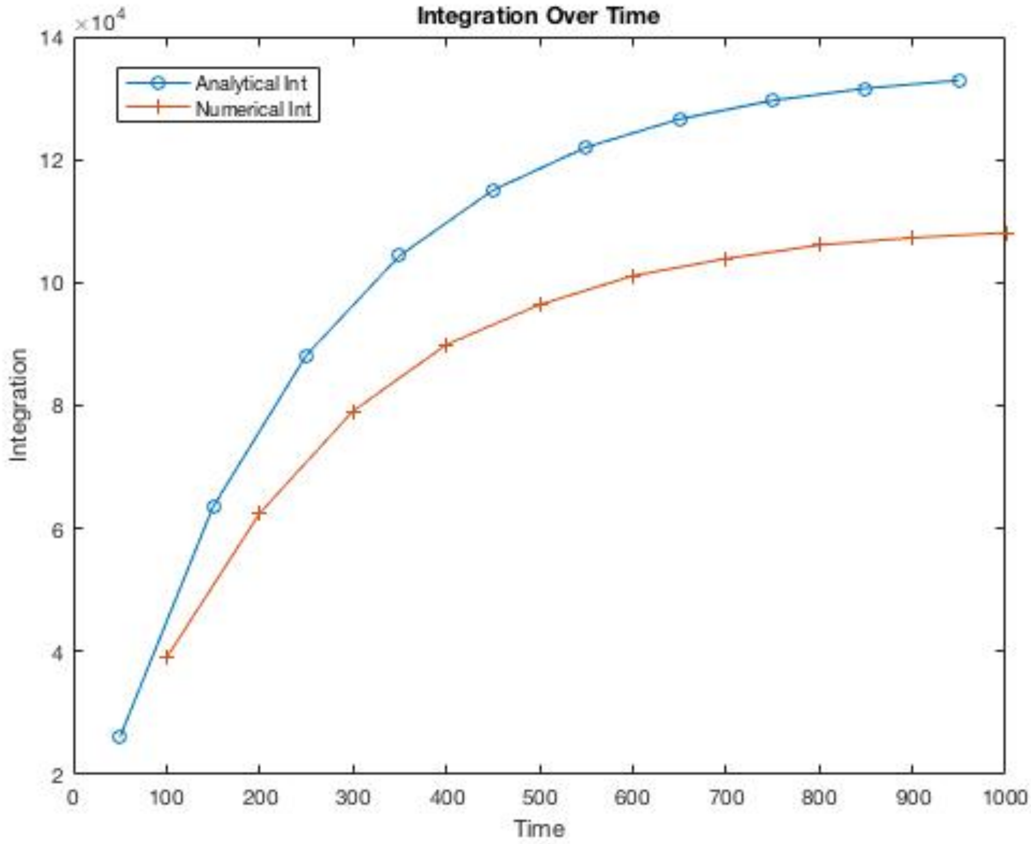
plot3-2

e) Evaluate $\int_0^{1000} A(t)dt$ both numerically from the given data, and analytically from the expression you derived in (c). Compare and discuss.

Plot3-3 compares the integration of A both numerically and analytically over [0,1000] time interval. The integration derived analytically tend to have higher value than the numerical expression, but they both reached plateau at around 1000 years which means that material will finally disappear.

```
% analytical int
dx =100;
b = 1000;
xb = [0:dx:b];
nxb = length(xb);
xbmid =.5*(xb(1:nxb-1)+xb(2:nxb));
```

```
yba_int = -134860.*exp(-0.0042.*xb)+135395;
ybamid_int = -134860.*exp(-0.0042.*xbmid)+135395;

% numerical int
A_r = [535, 390, 235, 166, 108, 65, 47, 28, 22, 12, 8]; % unit g
yn_int = cumsum(A_r(2:end).*dx);
figure(3)
plot(xbmid,ybamid_int,'o-')
hold on
plot(xb(2:end),yn_int,'+-')
legend('Analytical Int','Numerical Int')
xlabel('Time')
ylabel('Integration')
title('Integration Over Time')
```



plot3-3

4. Consider the function:
$f(x, y) = 4x + 2y - e^{x/2}y^2$
a) Evaluate $\partial f/\partial x$ and $\partial f/\partial y$ analytically.

df/dx = 4 - (1/2).*(y.^2).*exp(x./2)
df/dy = 2 – 2.*y.*exp(x./2)

b) Given $y = 2$, expand $f(x)$ in a Taylor series about $x_0 = 1$. Drop terms in the Taylor series that contain higher than 2nd-order derivatives, to get a polynomial of the form $F(x) = a + b(x - x_0) + c(x - x_0)^2$. Determine the values of a, b, and c.

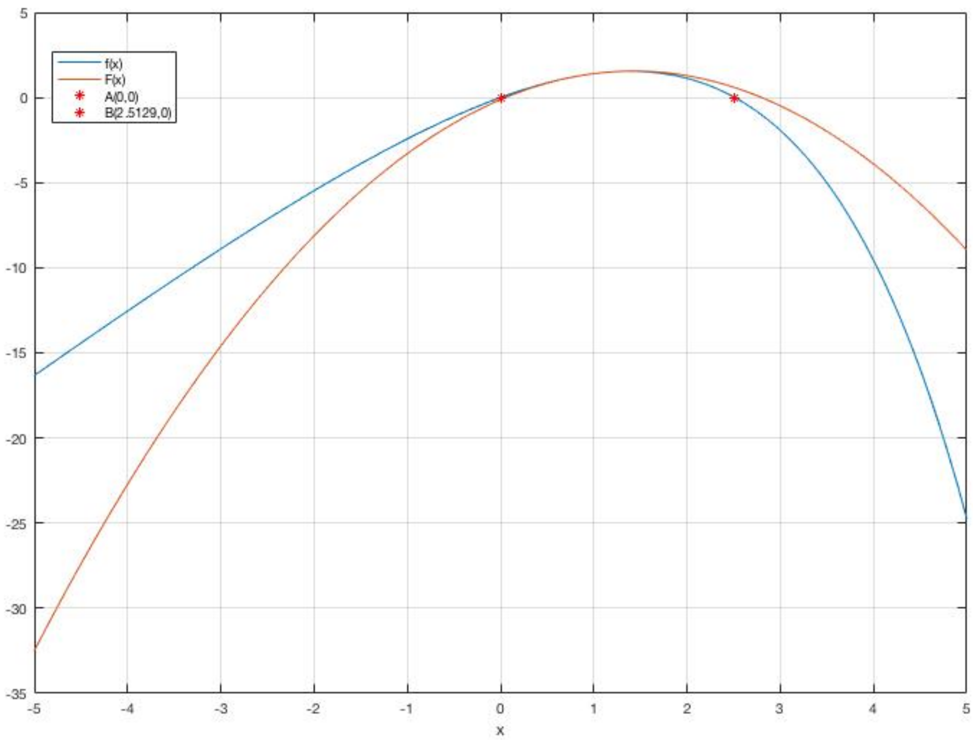
Given y = 2, f(x) = 4.*x + 4 – 4.*exp(x./2)
F(x) = f(x0) + (f'(x0)./1).*(x-x0) + (f''(x0)./(2*1)).*((x-x0).^2)
X0 =1
f(x0)= 8 – 4.*exp(1/2)
f'(x0) = 4 – 2.*exp(1/2)
f''(x0) = -exp(1/2)

F(x) = (8 – 4.*exp(1/2)) + (4 – 2.*exp(1/2)).*(x-x0) + (-(1/2).* exp(1/2)) .*((x-x0).^2)
In the form of F(x) = a + b(x-x0) + c(x-x0).^2,
a = 8 – 4.*exp(1/2)
b = 4 – 2.*exp(1/2)
c = -(1/2).* exp(1/2)

c) Given $y = 2$ as above, plot $f(x)$ and $F(x)$ (the polynomial approximation to $f(x)$ at $x_0 = 1$) over the domain $-5 \le x \le 5$. Compare and discuss.

From plot4-1, we could tell that f(x) and F(x) have quite different shape, but both functions converge at the same spot (critical point) and have increase trend to the left of the critical point and have decrease trend to the right of the critical point.

```
y=2;
x = [-5:0.1:5];
x0 = 1;
a = 8-4.*exp(1/2);
b = 4 - 2.*exp(1/2);
c = -(1/2).*exp(1/2);
f =@(x) 4.*x + 4 - 4.*exp(x./2);
F =@(x) a + b.*(x-x0) + c.*((x-x0).^2);
figure(1)
plot(x,f(x))
hold on
plot(x,F(x))
legend('f(x)','F(x)')
xlabel('x')
grid on
```



plot4-1

d) Given $y = 2$ as above, determine x where $f(x) = 0$ using Newton's method.

From plot4-1, we could tell that f(x) = 0 has two roots: point A and point B. I apply Newton's method twice (initial guess = 1 and 1.5 separately) and get the coordinate of point A (0,0) and B(2.5129, 0), so **x=0 and x =2.5129** are the two roots where f(x) = 0.

```
x = [-5:0.1:5];
f =@(x) 4.*x + 4 - 4.*exp(x./2);
initial_guess = 1;% calculating A
x = initial_guess;
y = f(x); %call function to evaluate initial value
itc = 0; %iteration counter
while(abs(y) > 1e-2) %that means 10^-2; as long as fx hasn't reached 0.01
y = f(x);
dx = 1e-3;
fup = f(x+dx);
fdown = f(x-dx);
dfdx = (fup-fdown)/(2*dx);
dif = -f(x)./dfdx;
x = x+dif;
itc = itc+1;
xi(itc) = x;
yi(itc) = f(x);
end;

% xi = -1.0000   -0.1295   -0.0038   -0.0000
% yi = -2.4261   -0.2671   -0.0076   -0.0000  % A(0,0)
```

```
% mark 0pts
figure(1)
plot(0,f(0),'r*')

initial_guess = 1.5;% calculating B
x = initial_guess;
y = f(x); %call function to evaluate initial value
itc = 0; %iteration counter
while(abs(y) > 1e-2) %that means 10^-2; as long as fx hasn't reached 0.01
y = f(x);
dx = 1e-3;
fup = f(x+dx);
fdown = f(x-dx);
dfdx = (fup-fdown)/(2*dx);
dif = -f(x)./dfdx;
x = x+dif;
itc = itc+1;
xi(itc) = x;
yi(itc) = f(x);
end;

xi = 8.0470    6.3085    4.8038    3.6450    2.9083    2.5816    2.5155    2.5129
yi = -187.3979  -64.5080  -20.9614   -6.1689   -1.4903   -0.2165   -0.0079   -0.0000
% B(2.5129,0)

% mark 0pts
figure(1)
plot(2.5129,f(2.5129),'r*')
legend('f(x)','F(x)','A(0,0)','B(2.5129,0)')
```

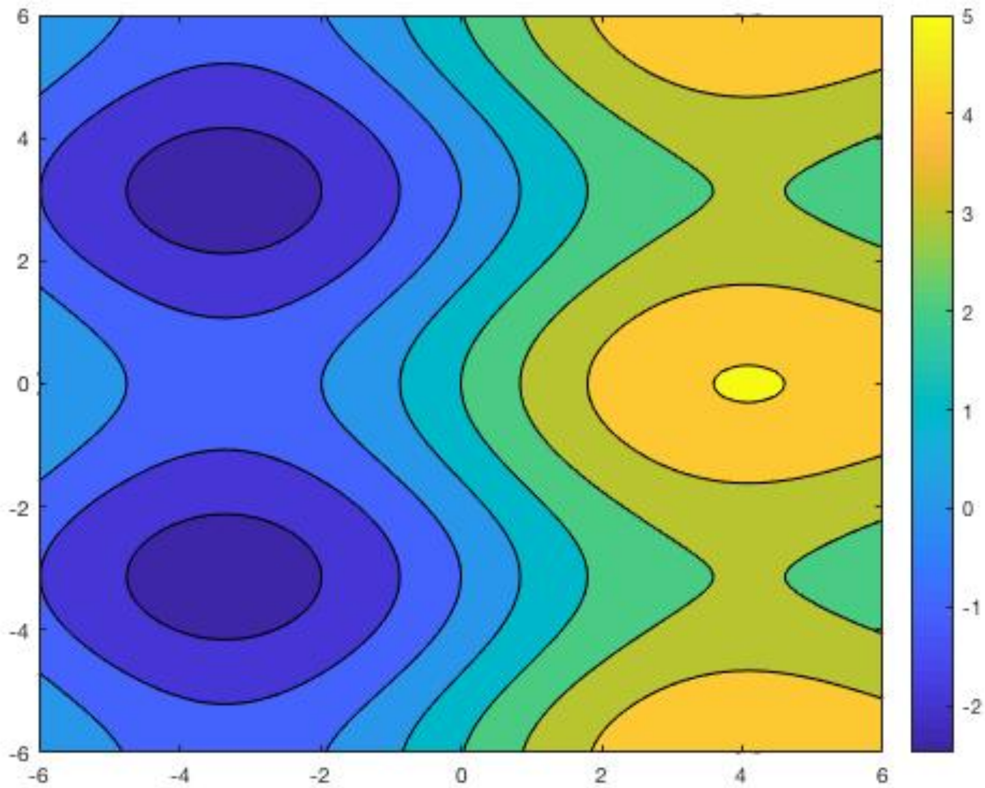
5. Consider the function:

$z = 2\sin(x/2) + \cos(y) + e^{x/5}$

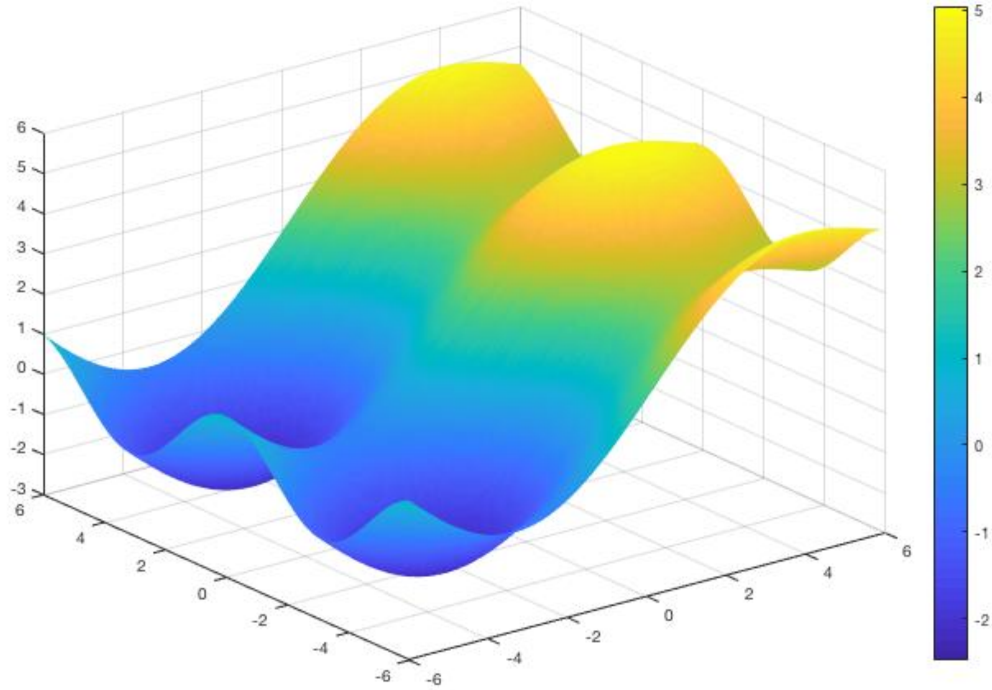
a) Plot z using contourf or surf in MATLAB over the domain $-6 \leq x \leq 6$ and $-6 \leq y \leq 6$.

See plot5-1 and plot5-2.

```
%make a grid
dx = 0.1; %resolution of grid
dy = 0.1;
x = [-6:dx:6]; % domain of interest
y = [-6:dy:6]; % y-domain
nx = length(x);
ny = length(y);
[X,Y] = meshgrid(x,y);

% z
Z = 2.*sin(X./2)+cos(Y)+exp(X./5);

%visualize using contour or surf
figure(1)
clf
contourf(X,Y,Z)

figure(2)
clf
surf(X,Y,Z)
shading('flat')
```



Plot5-1



Plot5-2

b) Find all the critical points of $z(x, y)$. (Critical points are where all the partial derivatives are equal to 0).

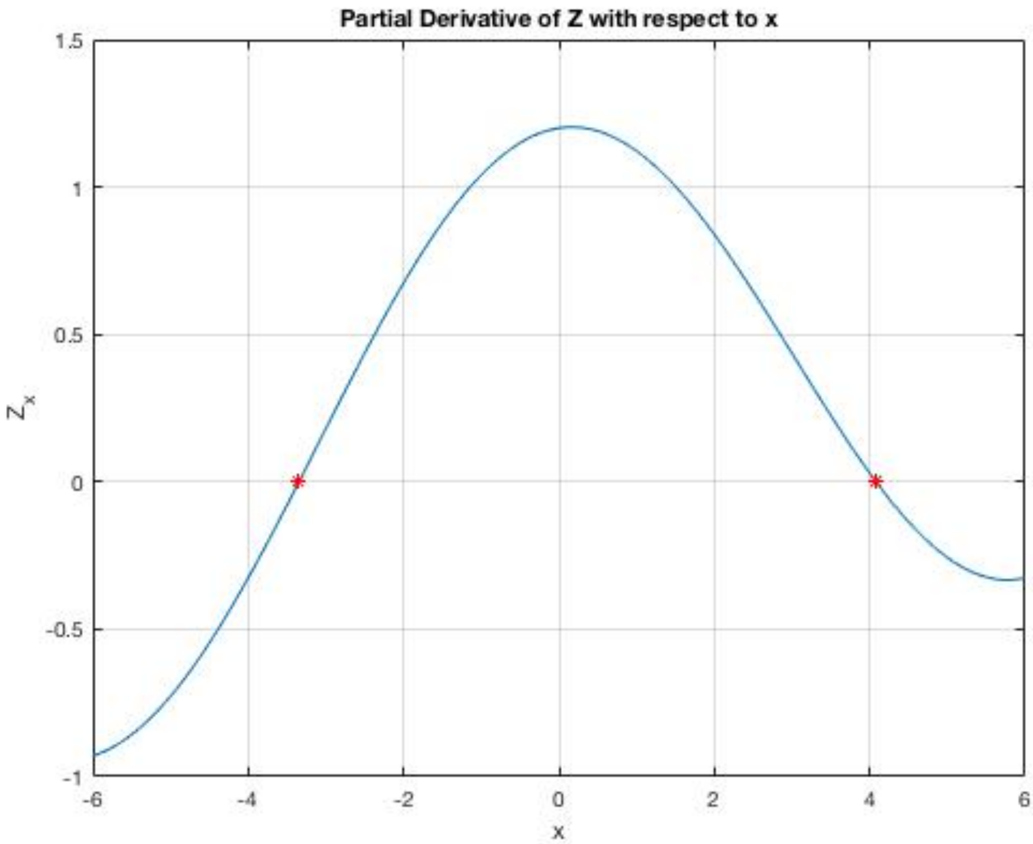
From plot5-1 and plot5-2, we can found 6 critical points. In order to confirm the coordinates of all these 6 points, I calculated the partial derivative of z with respect to x and partial derivative of z with respect to y and plot them separately. (By citing fzero method used in website of https://www.mathworks.com/help/matlab/math/roots-of-polynomials.html#bus0f0s-1 , I get and confirm x and y values of critical points when the partial derivative of z with respect to x equals to 0 (see plot5-3), as well as when partial derivative of z with respect to y equals to 0 (see plot5-4). (note: I confirmed my result by using Newton's method, but I think fzero method is faster).

I finally get approximate coordinates of 6 critical points:

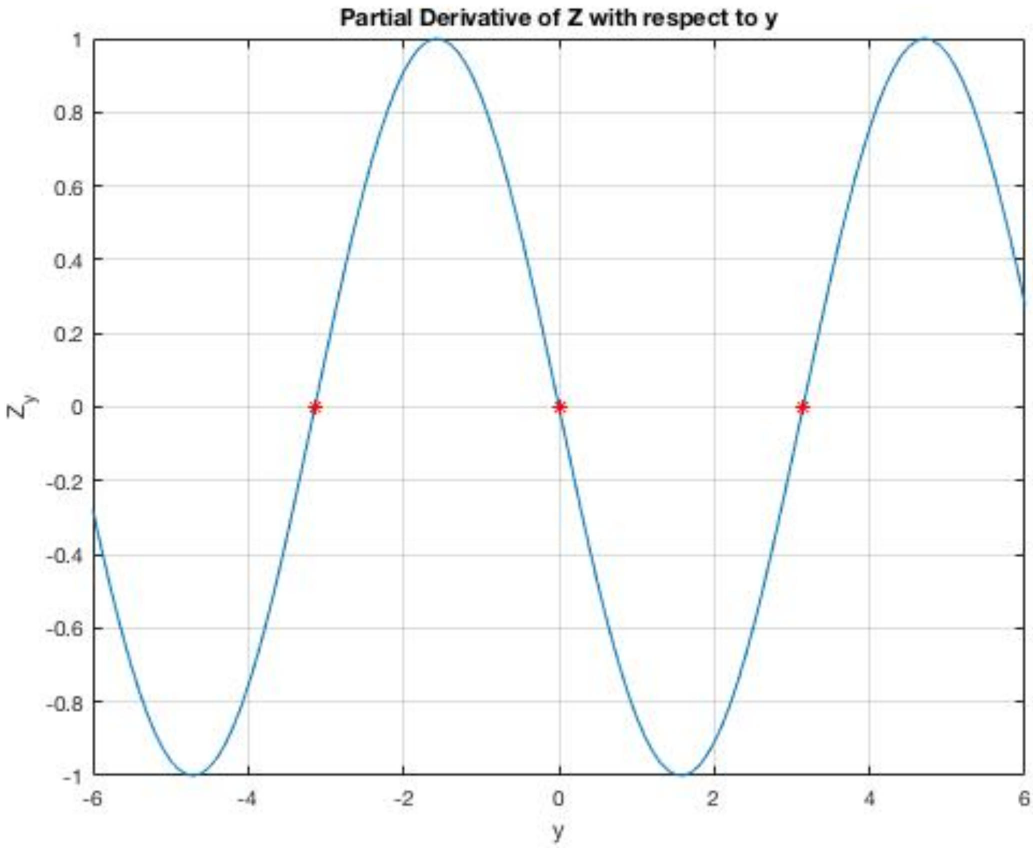
C1(-3.3468,-pi);
C2(-3.3468,0);
C3(-3.3468,pi);
C4(4.0803,-pi);
C5(4.0803,0);
C6(4.0803,pi)

```
% manually calculate partial partial derivative of Z with respect to x
```

```
Z_x = @(x) cos(x./2)+(1/5).*exp(x./5);
figure(3)
plot(x,Z_x(x))
grid on
hold on
f1 = fzero(Z_x,-4) %f1 = -3.3468
f2 = fzero(Z_x,4) % f2 = 4.0803
plot(f1,Z_x(f1),'r*')
hold on
plot(f2,Z_x(f2),'r*')
hold off
xlabel('x')
ylabel('Z_x')
title('Partial Derivative of Z with respect to x')

% manually calculate partial partial derivative of Z with respect to y
Z_y = @(y) -sin(y);
figure(4)
plot(y,Z_y(y))
grid on
hold on
plot(-pi,Z_y(-pi),'r*')
hold on
plot(0,Z_y(0),'r*')
hold on
plot(pi,Z_y(pi),'r*')
hold off
xlabel('y')
ylabel('Z_y')
title('Partial Derivative of Z with respect to y')
```



Plot5-3



plot5-4

c) Determine whether the critical points are maxima, minima, or saddle points. (Use the plot to help you).

From plot5-1 and plot5-2, we can approximately see that c2(-3.3468,0), c4(4.0803,-pi) and c6(4.0803,pi) are saddle points; c5(4.0803,0) is maxima point; c1(-3.3468,-pi) and c3(-3.3468,pi) are minima points. In order to confirm, I wrote up a second partial derivative test function and test values for each critical point.

```
function [H] = second_partial_derivative_test(x,y)
 Z_xx = -(1/2).*sin(x./2) + (1/25).*exp(x./5);
 Z_yy = -cos(y);
 Z_xy =  0;
 H = Z_xx.*Z_yy - Z_xy.^2;
```

Then I did second partial derivative test to all of the 6 critical points:

```
% c1(-3.3468,-pi);c2(-3.3468,0);c3(-3.3468,pi);c4(4.0803,-
pi);c5(4.0803,0);c6(4.0803,pi)
H1 = second_partial_derivative_test(-3.3468,-pi) % H1 = 0.5179  Z_yy =
1>0 minimum
H2 = second_partial_derivative_test(-3.3468,0) % H2 = -0.5179 saddle
```

```
H3 = second_partial_derivative_test(-3.3468,pi) % H3 = 0.5179  Z_yy = 1>0
minimum
H4 = second_partial_derivative_test(4.0803,-pi) % H4 = -0.3555 saddle
H5 = second_partial_derivative_test(4.0803,0) % H5 = 0.3555 Z_yy = -1 <0
maximum
H6 = second_partial_derivative_test(4.0803,pi) % H1 = -0.3555 saddle
```

C1(-3.3468,-pi);
C2(-3.3468,0);
C3(-3.3468,pi);
C4(4.0803,-pi);
C5(4.0803,0);
C6(4.0803,pi)

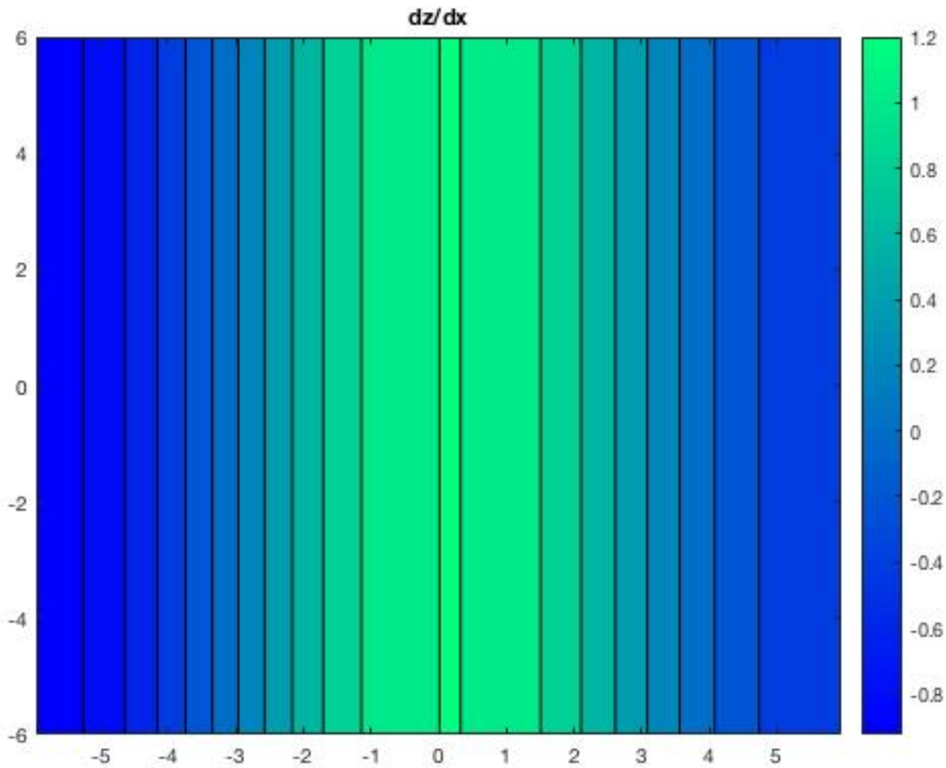
It turns out that the second partial derivative test result of points C2, C4 and C6 are all smaller than 0. These 3 points should be saddle points. Considering the second partial derivative of z with respect to y for points C5 is smaller than 0 and its second partial derivative test result is bigger than 0, C1 is the maximum point. The second partial derivative of z with respect to y for both points C1 and C3 are bigger than 0 and their second partial derivative test result are also both bigger than 0, both points are minimum points. In case one point of C1,C3 is only local minimum, I calculated the z values of C1 and C3 and it turns out they have the same result (=-2.4774), so both C1, C3 points are minimum.

In sum, the result of second partial derivative test is the same as what we see from plot5-1 and plot5-2. Specifically, among all 6 critical points, C2(-3.3468,0), C4(4.0803,-pi) and C6(4.0803,pi) are saddle points; C5(4.0803,0) is maxima point; C1(-3.3468,-pi) and C3(-3.3468,pi) are minima points.

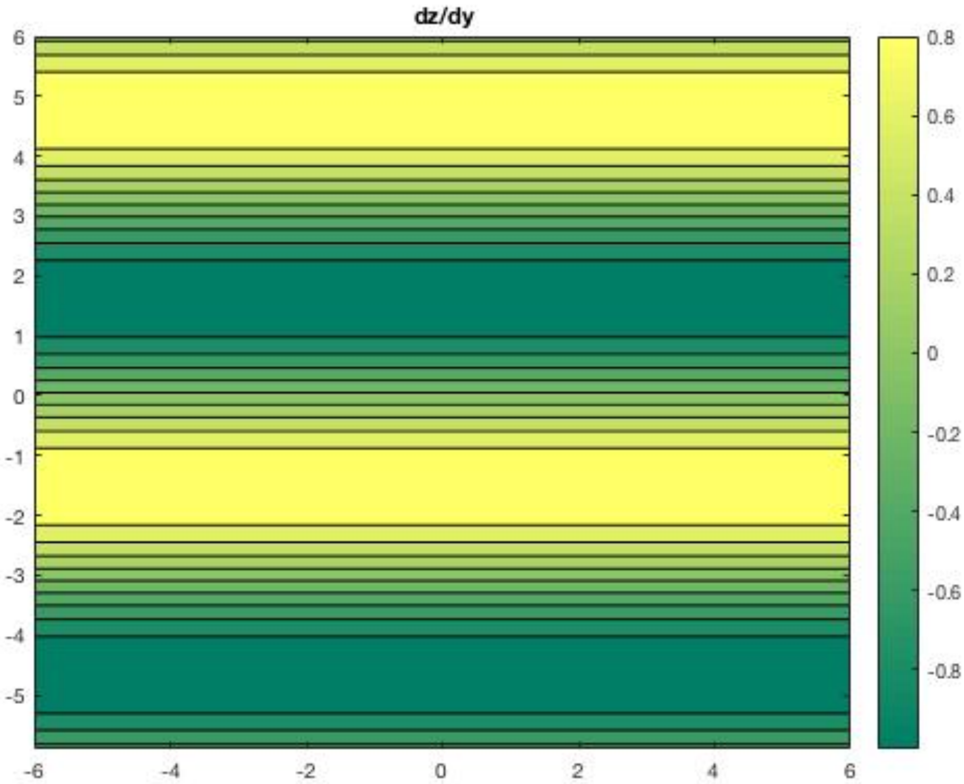
d) Evaluate $\partial z/\partial x$ and $\partial z/\partial y$ numerically and plot each using contourf or surf.

```
% partial derivative of z with respect to x
dX = X(:,2:nx)-X(:,1:nx-1); % dx= x(2:nx)-x(1:nx-1)
dZx = Z(:,2:nx)-Z(:,1:nx-1); % dZ in x-direction
Z_x = dZx./dX; % Z_x is shorthand for dz/dx

%plot x derivative at the mid points
Xmx = 0.5*(X(:,1:nx-1)+X(:,2:nx));
Ymx = Y(:,2:nx); % just taking the last 2:nx elements of Y in x-direction
figure(5)
contourf(Xmx,Ymx,Z_x)
colormap(winter)
colorbar
title('dz/dx')

% plot z derivative with respect to y
dY = diff(Y,1,1); % for diff(variable, nth order of derivative,
dimension)
dZy = diff(Z);
Z_y = dZy./dy;
%plot y derivative at the mid points
Xmy = 0.5*(X(1:ny-1,:)+X(2:ny,:));
Ymy = Y(2:ny,:); % just taking the last 2:nx elements of Y in x-direction
figure(6)
contourf(Z_y)
```

```
colormap(summer)
colorbar
title('dz/dy')
```



Plot5-5



Plot5-6

6. Consider the differential equation:

$$\frac{dy}{dt} = k(y_m - y)$$

a) Find the analytical solution $y = f(t)$.

dy/dt = k*(ym-y)
dy = k*(ym – y)*dt
(1/(ym-y))*dy = k*dt (note: integrate both sides)
- ln(ym-y) = kt + C   (note: C represents constant)
exp(ln(ym-y) = exp(-k*t-C)
ym – y = C*exp(-k*t)
y = ym – C*exp(-k*t)

b) Solve the above equation numerically given $k = 1/2$, $y_m = 10$, and $y(t = 0) = 1$.

i) Use the Euler forward method with a time-step of $dt = 0.1$.
ii) Use the Euler backward method with a time-step of $dt = 0.1$.
iii) Use the midpoint method with a time-step of $dt = 0.1$.

Please see the code below:

```
%(b)
k = 1/2;
ym = 10;
% time domain
dt = 0.1;
t = [0:dt:10];

% differential equation
f = @(t,y)k.*(ym-y);

% initial condition
y0 = 1;

% solve with Euler forward
yf = 0*t; % placeholder
yf(1) = y0;

for i = 2:length(t)
    yf(i) = yf(i-1) + dt*f(t(i-1),yf(i-1));
end

% plot
plot(t,yf,'-og')
hold on
xlabel('t')
ylabel('y')

% (ii) solve with euler backward
yb = 0*t; %placeholder
yb(1)= y0;
for i = 2:length(t)
    yb(i)=(yb(i-1)+dt*k*ym)./(1+dt*k);
end;

%plot
plot(t,yb,'-<m')

% (iii) mid-point Euler backward
```

```
yg = 0*t; %placeholder
yg(1)= y0;
for i = 2:length(t)
    % yg(i) = yg(i-1) +
    % dt*f(t(i-1)+0.5*dt,yg(i-1)+0.5*dt*f(t(i-1),yg(i-1)))
    yg(i)= (1-dt*k+0.5*(dt.^2)*(k.^2))*yg(i-1) + dt*k*ym*(1+0.5*dt*k);
end;

%plot
plot(t,yg,'bd')
```

c) Plot and compare the various numerical solutions and the analytical solution. Discuss.

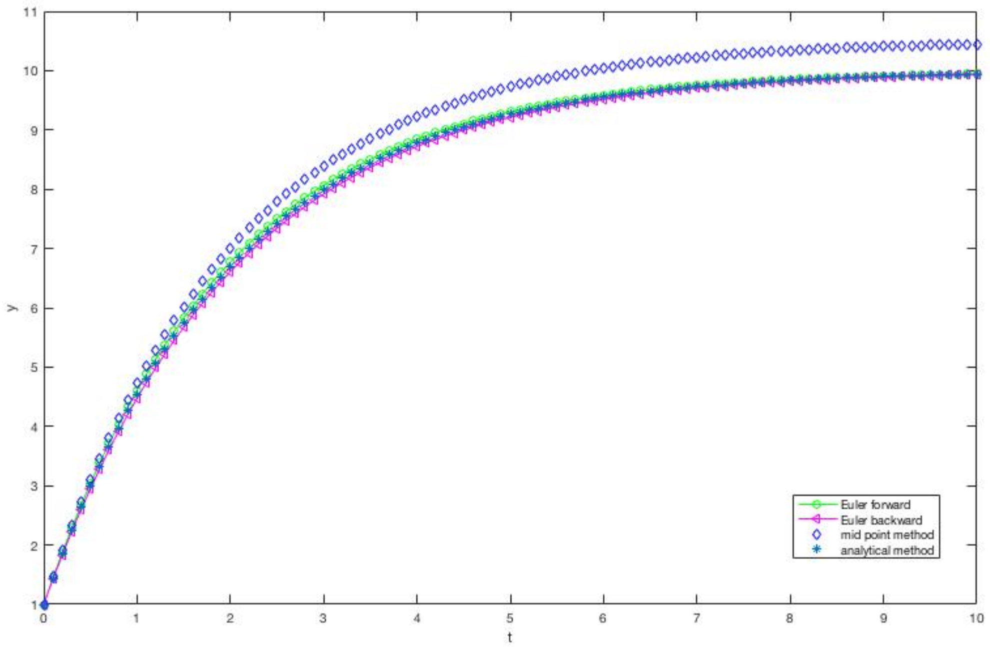
For the analytical solution y = ym – C*exp(-k*t), add values of k =1/2, ym = 10
When t=0, y =1. We could get that the constant C = 9 and we get the analytical solution:

y = 10 – 9*exp(-0.5*t)

plot analytical solution together with the above three numerical solutions and we get plot6

```
yh = 10 - 9*exp(-0.5*t);
plot(t,yh,'*');
legend('Euler forward','Euler backward','mid point method','analytical
method');
```

From plot6, we could tell that numerical solutions of both Euler forward and Euler backward methods are much closer to analytical solution than the numerical solution of midpoint method. The value of analytical solution falls in between solutions of Euler forward and Euler backward.



Plot6

7. Consider the following system of differential equations that represents a model of two populations, x and y, competing for the same resources:

$$\frac{dx}{dt} = x(A - Px - Qy) + C$$

$$\frac{dy}{dt} = y(B - Rx - Sy) + D$$

a) Write the system of equations in matrix-vector form $dv/dt = \mathbf{A}v + q(v) + r$. Determine the values of v, $\mathbf{A}$, $q(v)$ and r.

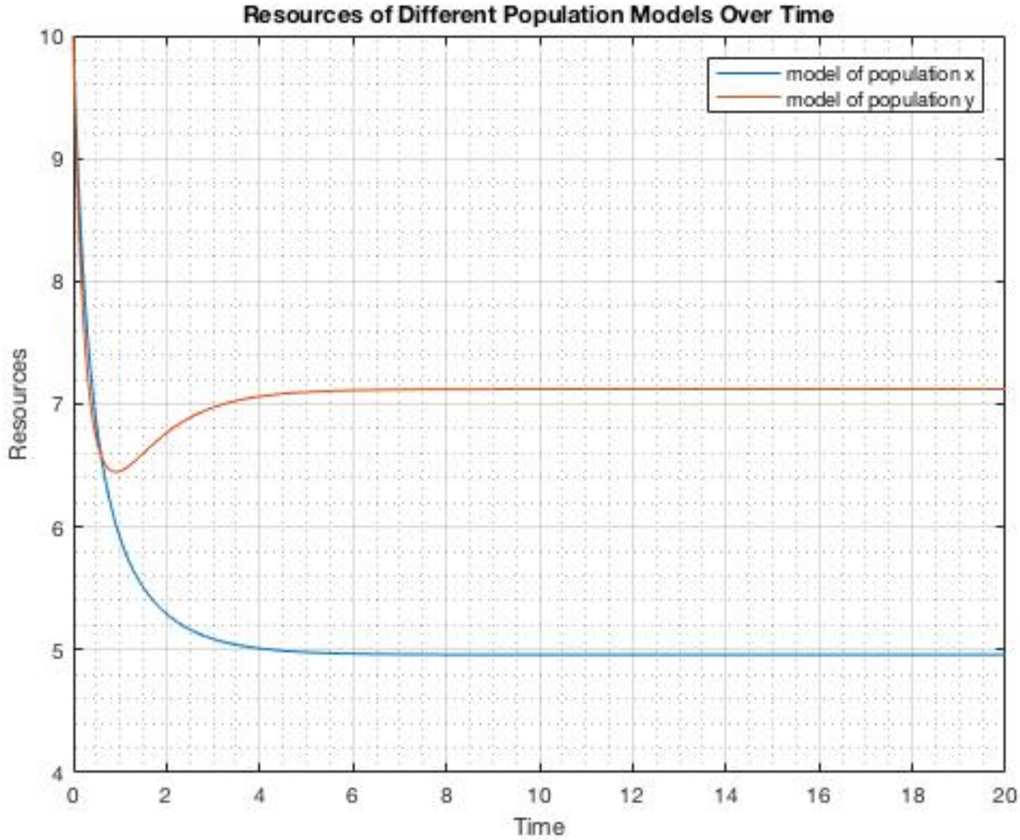
v = [x;y]
A = [A,0;0,B]
q(v) = [-(P.*(x.^2) + Q.*x.*y);-(R*x.*y + S.*(y.^2))]
r = [C;D]

b) Find the equilibrium solution to the system of equations using a suitable time-stepping method. Use A = 0.2, P = 0.1, Q = 0.1, B = 0.3, R = 0.2, S = 0.1, C = 5, D = 10. Use the initial condition $x(t = 0) = y(t = 0) = 10$. Plot your solution. When do the two populations reach steady-state? What is the equilibrium value of x and y?

From plot7-1, we can see that the two populations reach steady-state at around the 8th time unit. The equilibrium value of x is around 4.95 and the equilibrium value of y is around 7.12.

```
% (b)
A1 = 0.2;
P = 0.1;
Q = 0.1;
B = 0.3;
R = 0.2;
S = 0.1;
C = 5;
D = 10;

A = [A1,0;0,B];
q = @(x,y)[-(P.*(x.^2) + Q.*x.*y);-(R*x.*y + S.*(y.^2))];
r = [C;D];

% time domain
dt = .01;
t = [0:dt:20];
nt = length(t);

% initial conditions
v0 = [10 10]';

v(:,1) = v0;

% pass through
for i = 2:nt
    v(:,i) = v(:,i-1) + dt*(A*v(:,i-1)+q(v(1,i-1),v(2,i-1))+r);
end

figure(1)
plot(t,v')
legend('model of population x','model of population y')
xlabel('Time')
```

```
ylabel('Resources')
title('Resources of Different Population Models Over Time')
grid on
grid minor
```



plot7-1

c) Solve the system of equations in (a) for the equilibrium (steady-state) solution where $dv/dt = 0$ using Newton's method. Compare your answer to the equilibrium solution you determined in (b).

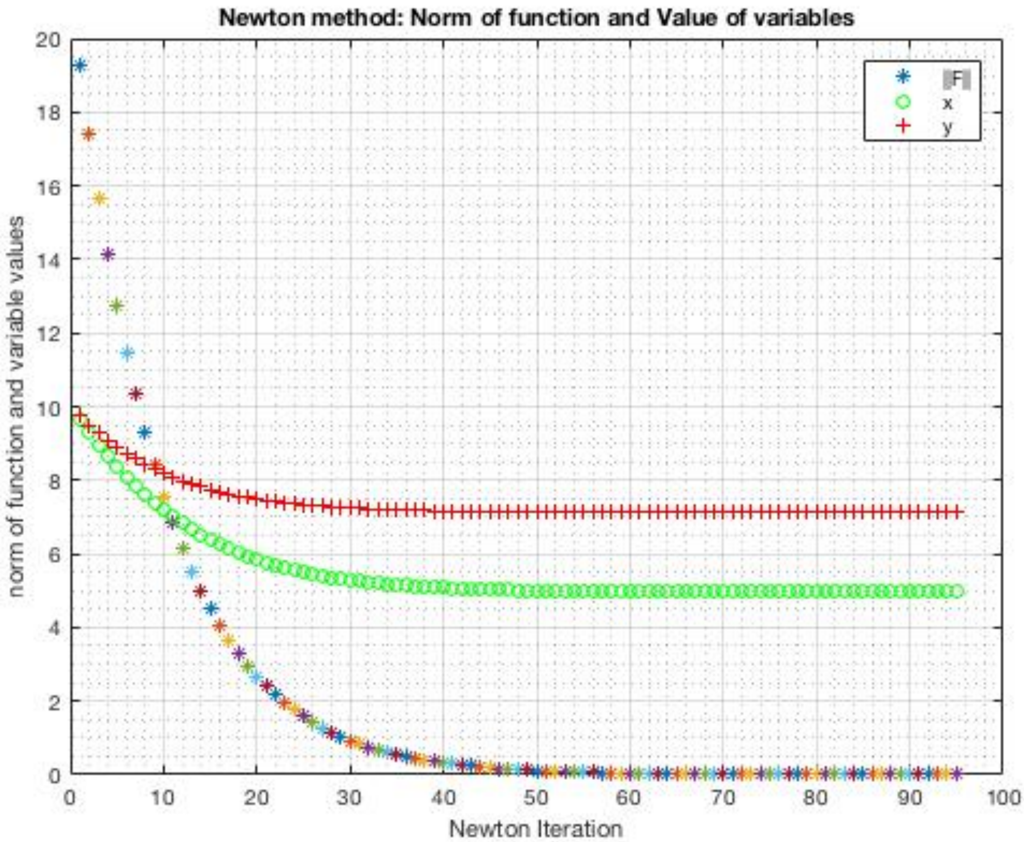
Compare plot7-1 and plot7-2, my answer to the equilibrium solution in (b) is the same as the answer I get in (c) using Newton's method.

%% Jacobian

```
function [F,J] = myfunc_q7(v, A, r)
A1 = 0.2;
P = 0.1;
Q = 0.1;
B = 0.3;
R = 0.2;
S = 0.1;
C = 5;
D = 10;

x = v(1);
y = v(2);
```

```
q = [-(P.*(x.^2) + Q.*x.*y);-(R*x.*y + S.*(y.^2))];
F = A*v + q +r;

dqdv = [(-2.*P.*x-Q.*y),(-Q.*x); (-R.*y) , (-R.*x - 2.*S.*y)];
J = A + dqdv;
```

```
% c
% linear part of system (A)
A = [A1,0;0,B];

% constants
r = [C;D];

% initial guess
v = [10 10]'; % v = [x y]'

% iterate to solution with Newton's method
F = myfunc_q7(v,A,r); % initial function evaluation
itc = 0; % iteration counter
while norm(F)> 1e-3 % 1e-3 tolerance

    % evaluate function and Jacobian
    [F,J] = myfunc_q7(v,A,r);

    % update v with Newtion step
    v = v - .1*(J\F);

    % update iteration counter
    itc = itc + 1;

    % collect (x,y) values
    xi(itc)=v(1);
    yi(itc)=v(2);

    % evaluate function and Jacobian
    [F,J] = myfunc_q7(v,A,r);
        % plot the norm of the function
    figure(2)
    plot(itc,norm(F),'*','DisplayName',['Iteration', num2str(itc)])
    hold on
    plot(itc,v(1),'go','DisplayName',['Iteration', num2str(itc)])
    hold on
    plot(itc,v(2),'r+','DisplayName',['Iteration', num2str(itc)])

    legend('||F||','x','y')
    drawnow % tell matlab to make plot
    set(gca,'YGrid','on')
    set(gca,'XGrid','on')
    xlabel('Newton Iteration')
    ylabel('norm of function and variable values')
    title('Newton method: Norm of function and Value of variables')
end
grid minor
```



Newton method: Norm of function and Value of variables
norm of function and variable values
Newton Iteration
‖F‖
x
y
0
2
4
6
8
10
12
14
16
18
20
0
10
20
30
40
50
60
70
80
90
100

Plot7-2