B1 Numerical Algorithms

Computational Class - MT 2023

Wes Armour

30th October 2023

Errors/Typos/Questions to: wes.armour@eng.ox.ac.uk

Question 1 – Difference formulas

a. Derive the five point central difference formula:

$$f_n' \approx \frac{f_{n-2} - 8f_{n-1} + 8f_{n+1} - f_{n+2}}{12h}$$

b. Demonstrate that the error is $O(h^4)$

Question 1 – Difference formulas

Q1/ a&b) To obtain the five point formula taylor series expand: $f(x \pm \delta x)$ & $f(x \pm 2\delta x)$.

$$f(x \pm \delta x) = f(x) \pm \delta x \frac{df(x)}{dx} + \frac{\delta x^2}{2!}\frac{d^2f(x)}{dx^2} \pm \frac{\delta x^3}{3!}\frac{d^3f(x)}{dx^3} + \frac{\delta x^4}{4!}\frac{d^4f(x)}{dx^4}$$

$$f(x \pm 2\delta x) = f(x) \pm 2\delta x \frac{df(x)}{dx} + \frac{4\delta x^2}{2!}\frac{d^2f(x)}{dx^2} \pm \frac{8\delta x^3}{3!}\frac{d^3f(x)}{dx^3} + \frac{16\delta x^4}{4!}\frac{d^4f(x)}{dx^4}$$

Question 1 – Difference formulas

Next note...

$$f(x+\delta x) - f(x-\delta x) = 2\delta x \frac{df(x)}{dx} + \frac{2\delta x^3}{3!}\frac{d^3f(x)}{dx^3} + \frac{2\delta x^5}{5!}\frac{d^5f(x)}{dx^5} \quad (1)$$

1

&

$$f(x+2\delta x) - f(x-2\delta x) = 4\delta x \frac{df(x)}{dx} + \frac{16\delta x^3}{3!}\frac{d^3f(x)}{dx^3} + \frac{32\delta x^5}{5!}\frac{d^5f(x)}{dx^5} \quad (2)$$

2

Question 1 – Difference formulas

To eliminate $O(\delta x^3)$ ....

$$8 \times ① - ② = 8f(x+\delta x) - 8f(x-\delta x) - f(x+2\delta x) - f(x-2\delta x) + O(h^5)$$

$$= 16\delta x \frac{d}{dx}f(x) - 4\delta x \frac{d}{dx}f(x) + O(h^5) = 12\delta x \frac{d}{dx}f(x) + O(h^5)$$

$$\therefore \frac{d}{dx}f(x) = \frac{f(x-2\delta x) - 8f(x-\delta x) + 8f(x+\delta x) - f(x+2\delta x)}{12\delta x} + O(h^4)$$

as required

See Burden and Faires (9th Edition) Chapter 4.1

Question 2 – Numerical integration, the left point rule

a. Using MATLAB, write a code to implement left point rectangular dissection and integrate the following function between $x = -4$ and $x = 4$

$$f(x) = \frac{1}{2\pi\sigma^2} exp(-(x^2)/2\sigma^2)$$

Question 2 – Numerical integration, the left point rule

```
close all;
clear all;
clc;

% Integration range
start_point = -4.0;
end_point = 4.0;

% The exact analytic result
exact = 1;

% The gaussian function
xg = [start_point:.01:end_point];
yg =normpdf(xg,0,1);

% Number of steps to use
steps=10;

% Calculate our step size
h=(end_point-start_point)/steps;

% Create a vector that steps through our range
k=start_point:h:end_point;
```

Question 2 – Numerical integration, the left point rule

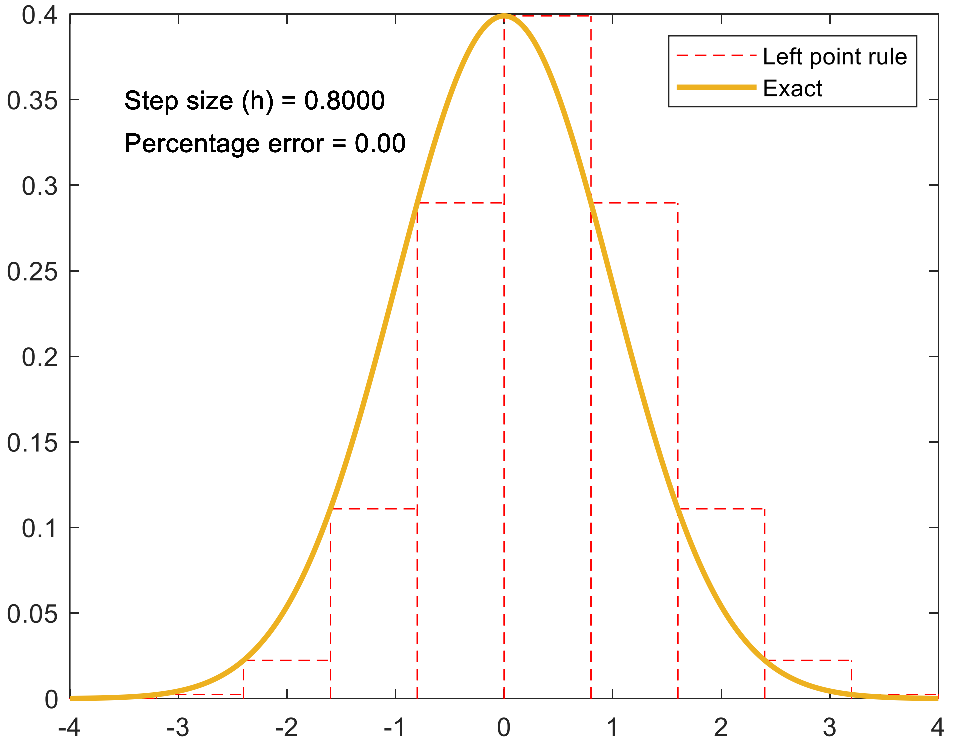
```
% Set the sum that will hold our result to zero
int_lpr=0.0;

% Calculate the value of the function at each left point
y_point = exp(-((k).^2)/2)/sqrt(2*pi);

% Create a vector that holds the area of each strip
lpr=y_point.*h;

% Sum up all of the areas
int_lpr = sum(lpr, 'all')

% Calcluate the error between the exact analytic result and our
left point
% rule approximation
error = (abs(int_lpr - exact)/exact).*100;

% Open a figure to plot to
figure;
```

Question 2 – Numerical integration, the left point rule

```
% Create a fancy plot!
% Plot the rectangles
for i=1:1:steps
    lp=(i-1)*h+start_point;
    rp=(i)*h+start_point;
    pt=plot([lp rp], [y_point(i) y_point(i)], '--r');
    hold on
    plot([rp rp], [0 y_point(i)], '--r');
    hold on
    plot([lp lp], [0 y_point(i)], '--r');
    hold on
    txt = ['Step size (h) = ' num2str(h,'%4.4f') ];
    text(-3.5,0.35,txt);
    txt = ['Percentage error = ' num2str(error,'%4.2f')];
    text(-3.5,0.325,txt);
end

% Plot the actual function
hold on
pe=plot(xg,yg,'-','LineWidth',2);
legend([pt pe],'Left point rule','Exact');

hold off
drawnow
```

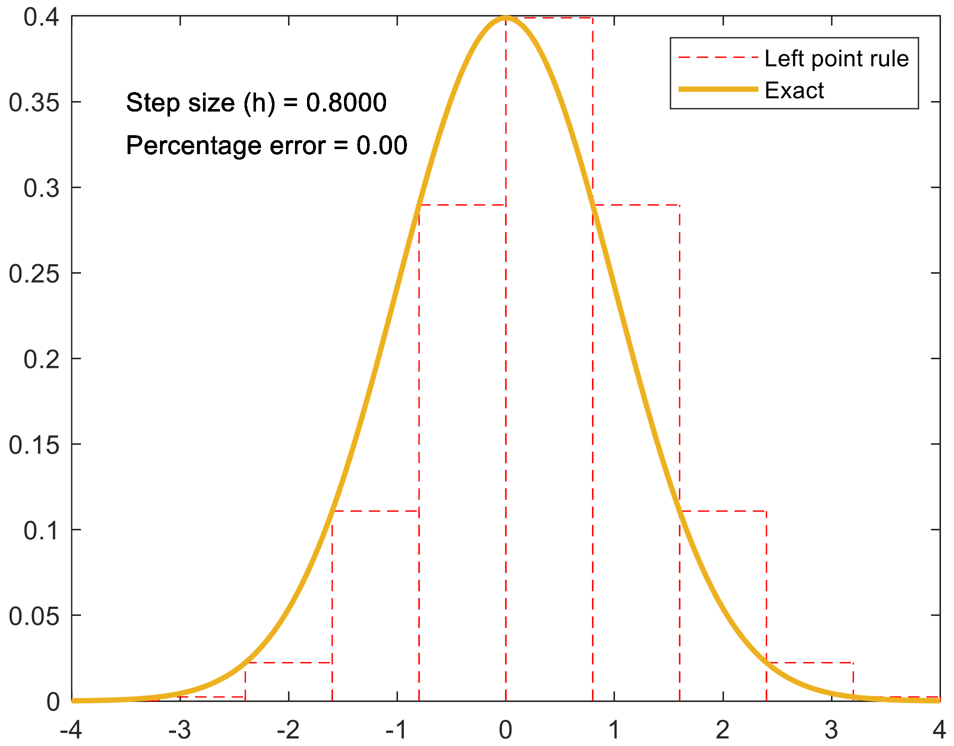
See MATLAB code on canvas – B1_CC_Q2A_left_point_rule.m

Question 2 – Numerical integration, the left point rule



Question 2 – Numerical integration, the left point rule

b. Estimate the error in your result. What do you notice?

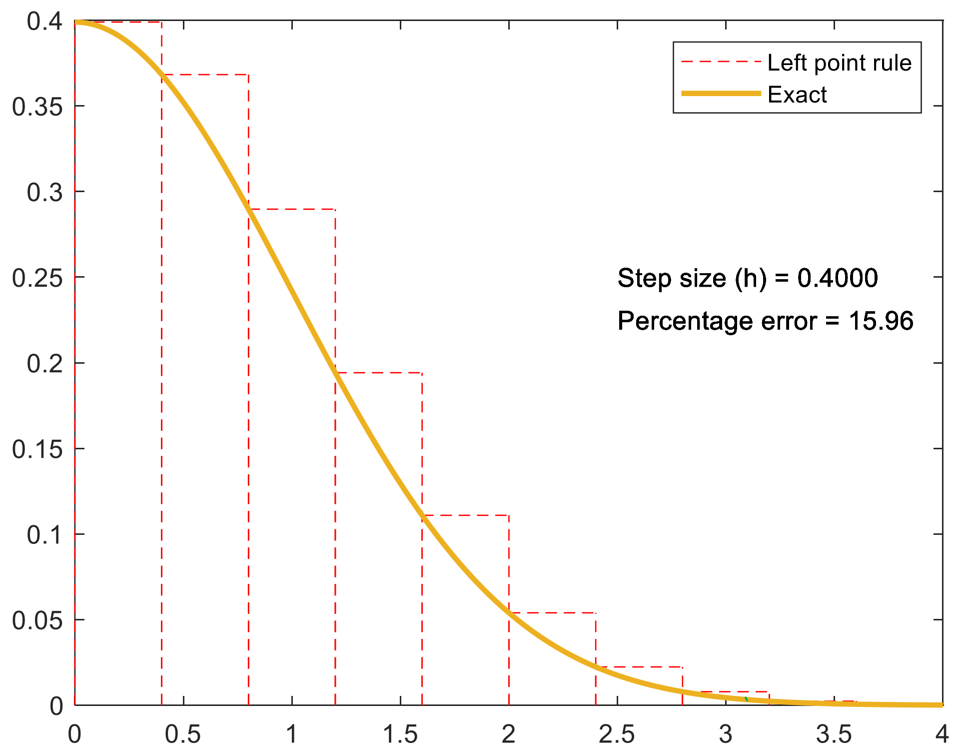


Question 2 – Numerical integration, the left point rule

c. Perform a) and b) above for the range $x = 0$ and $x = 4$.

Question 2 – Numerical integration, the left point rule

```
% The exact analytic result
exact = 0.5;



% Integration range
start_point = 0.0;
end_point = 4.0;



 txt = ['Step size (h) = ' num2str(h,'%4.4f') ];
 text(2.5,0.25,txt);
 txt = ['Percentage error = ' num2str(error,'%4.2f')];
 text(2.5,0.225,txt);
```



Question 3 – Numerical integration, the midpoint rule

a. Using MATLAB, implement the midpoint rule to integrate the following function between $x = -4$ and $x = 4$

$$f(x) = \frac{1}{2\pi\sigma^2} exp(-(x^2)/2\sigma^2)$$

Question 3 – Numerical integration, the midpoint rule

```
close all;
clear all;
clc;

% Integration range
start_point = -4.0;
end_point = 4.0;

% The exact analytic result
exact = 1.0;

% The gaussian function
xg = [start_point:.01:end_point];
yg =normpdf(xg,0,1);

% Number of steps to use
steps=10;

% Calculate our step size
h=(end_point-start_point)/steps;

% Create a vector that steps through our range
k=start_point:h:end_point;
```

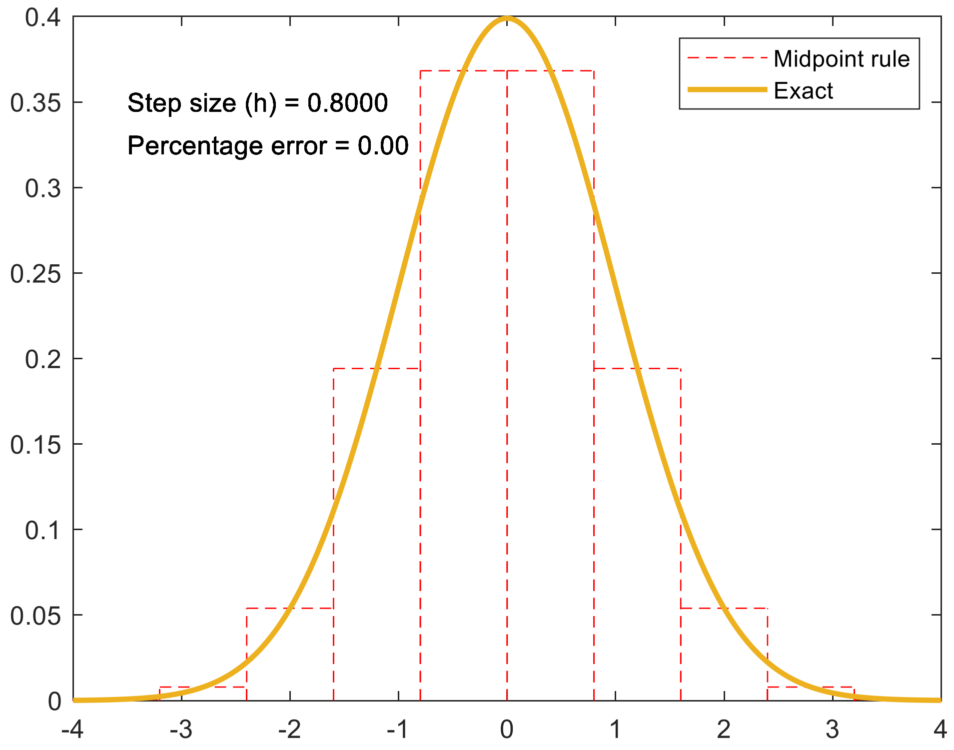
Question 3 – Numerical integration, the midpoint rule

```
% Set the sum that will hold our result to zero
int_mpr=0.0;

% Calculate the value of the function at each midpoint
y_point = exp(-((k+0.5*h).^2)/2)/sqrt(2*pi);

% Create a vector that holds the area of each strip
mpr=y_point.*h;

% Sum up all of the areas
int_mpr = sum(mpr, 'all')

% Calcluate the error between the exact analytic result and our
midpoint
% rule approximation
error = (abs(int_mpr - exact)/exact).*100;

% Open a figure to plot to
figure;
```

Question 3 – Numerical integration, the midpoint rule

```
% Create a fancy plot!
% Plot the rectangles
for i=1:1:steps
    lp=(i-1)*h+start_point;
    rp=(i)*h+start_point;
    pt=plot([lp rp], [y_point(i) y_point(i)], '--r');
    hold on
    plot([rp rp], [0 y_point(i)], '--r');
    hold on
    plot([lp lp], [0 y_point(i)], '--r');
    hold on
    txt = ['Step size (h) = ' num2str(h,'%4.4f') ];
    text(-3.5,0.35,txt);
    txt = ['Percentage error = ' num2str(error,'%4.2f')];
    text(-3.5,0.325,txt);
end

% Plot the actual function
hold on
pe=plot(xg,yg,'-','LineWidth',2);
legend([pt pe],'Midpoint rule','Exact');

hold off
drawnow
```

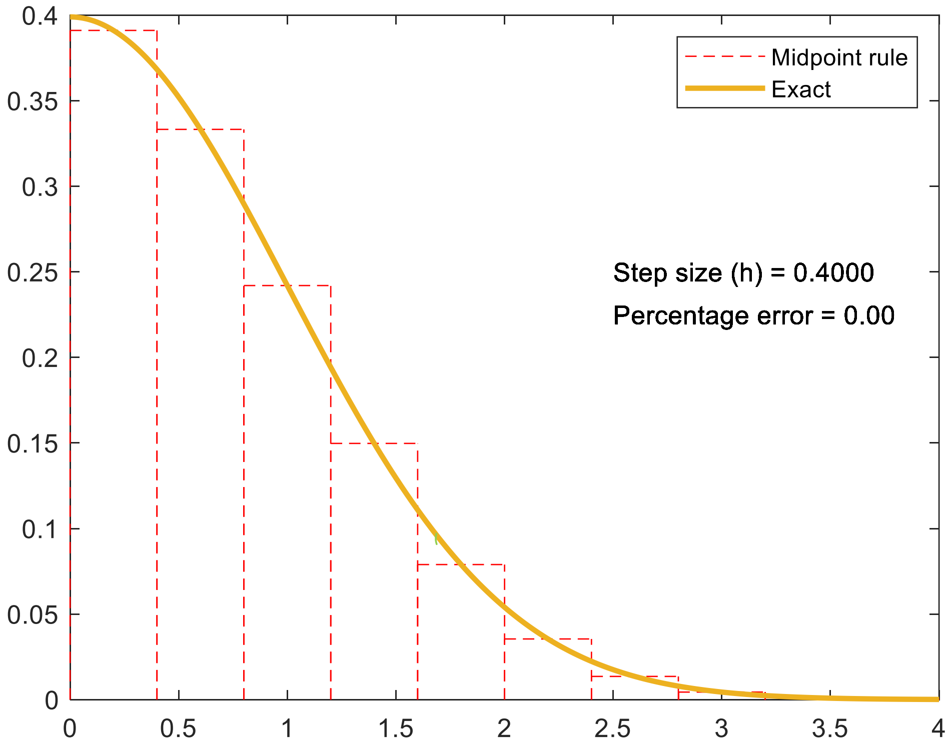
Question 3 – Numerical integration, the midpoint rule



Question 3 – Numerical integration, the midpoint rule

Bonus Question! – how about the range:

$$x = 0 \text{ and } x = 4$$



Question 3 – Numerical integration, the midpoint rule

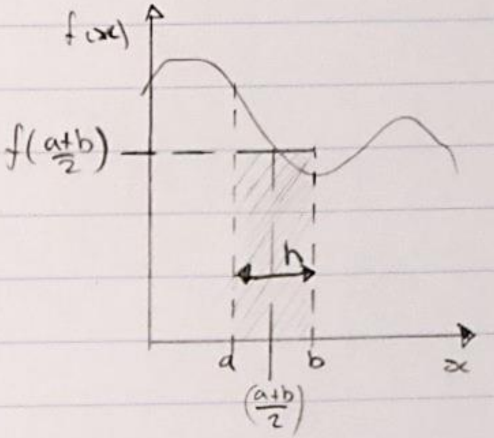
b. Derive the error of the midpoint rule.

Question 3 – Numerical integration, the midpoint rule

1

Error of Midpoint rule

Consider one integration interval ...



Area of midpoint rectangle

$$= (b-a) \cdot f\left(\frac{a+b}{2}\right)$$

Now consider taylor expanding f(x) about "α"

$$f(x) = f(\alpha) + (x-\alpha) f'(\alpha) + \frac{(x-\alpha)^2}{2!} f''(\alpha) + \frac{(x-\alpha)^3}{3!} f'''(\alpha) + \dots$$

Question 3 – Numerical integration, the midpoint rule

Next make some substitutions...

let $x_{1/2} = \frac{a+b}{2} \Rightarrow f\left(\frac{a+b}{2}\right) = f(x_{1/2})$

Taylor expand $f(x)$ about $x_{1/2} \Rightarrow$

$$f(x) = f(x_{1/2}) + (x - x_{1/2}) f'(x_{1/2}) + \frac{(x-x_{1/2})^2}{2!} f''(x_{1/2}) + \ldots$$

Now integrate between a & b $\Rightarrow$

$$\underbrace{\int_a^b f(x)\,dx}_{\text{EXACT RESULT}} = \underbrace{\int_a^b f(x_{1/2})\,dx}_{= (b-a) f(x_{1/2}) \;\leftarrow\; \text{WHAT WE CALCULATE!}} + \int_a^b (x - x_{1/2}) f'(x_{1/2})\,dx + \int_a^b \frac{(x-x_{1/2})^2}{2!} f''(x_{1/2})\,dx + \ldots$$

Question 3 – Numerical integration, the midpoint rule

So Error $= \displaystyle\int_a^b f(x)\,dx - \int_a^b f(x_{\frac{1}{2}})\,dx$

$$= \underbrace{\int_a^b (x - x_{\frac{1}{2}})\, f'(x_{\frac{1}{2}})\,dx}_{1} + \underbrace{\int_a^b \frac{(x - x_{\frac{1}{2}})^2}{2!}\, f''(x_{\frac{1}{2}})\,dx}_{2} + \ldots$$

$$1 \Rightarrow \int_a^b (x - x_{\frac{1}{2}})\, f'(x_{\frac{1}{2}})\,dx = f'(x_{\frac{1}{2}})\left[\int_a^b x\,dx - x_{\frac{1}{2}}\int_a^b dx\right]$$

$$= 0 \quad \text{I.M.V.T.}$$

Question 3 – Numerical integration, the midpoint rule

check $\Rightarrow$ $$= f'(x_{\frac{1}{2}})\left[\left(\frac{x^2}{2}\right)_a^b - \left(\frac{b+a}{2}\right)(b-a)\right]$$

$$= f'(x_{\frac{1}{2}})\left[\cancel{\frac{b^2}{2}} - \cancel{\frac{a^2}{2}} - \left(\cancel{\frac{b^2 + ab - ab - a^2}{2}}\right)\right]$$

$$= 0$$ as required.

② $$\int_a^b \frac{(x-x_{\frac{1}{2}})^2}{2!} f''(x_{\frac{1}{2}})\,dx = \frac{f''(x_{\frac{1}{2}})}{2}\int_a^b (x-x_{\frac{1}{2}})^2\,dx$$

Can integrate by hand (term by term) or use identity

$$\int (ax+b)^n\,dx = \frac{(ax+b)^{n+1}}{a(n+1)} + c \qquad (n \neq -1\,!)$$

Question 3 – Numerical integration, the midpoint rule

$$\Rightarrow \quad ② = \frac{f''(x_{\frac{1}{2}})}{2} \int_a^b (x - x_{\frac{1}{2}})^2 \, dx$$

$$= \frac{f''(x_{\frac{1}{2}})}{2} \left[\frac{(x - x_{\frac{1}{2}})^3}{3} \right]_a^b$$

$$= \frac{f''(x_{\frac{1}{2}})}{6} \left[\left(b - \left(\frac{a+b}{2}\right)\right)^3 - \left(a - \left(\frac{a+b}{2}\right)\right)^3 \right]$$

$$= \frac{f''(x_{\frac{1}{2}})}{6} \left[\left(\frac{\cancel{2}b - a - \cancel{b}}{2}\right)^3 - \left(\frac{\cancel{2}a - \cancel{a} - b}{2}\right)^3 \right]$$

$$b - a = h \quad \& \quad a - b = -h$$

$$\therefore \quad ② = \frac{f''(x_{\frac{1}{2}})}{6} \left[\frac{h^3}{8} + \frac{h^3}{8} \right]$$

Question 3 – Numerical integration, the midpoint rule

$$\therefore \quad ② = \frac{f''(x_{\frac{1}{2}})}{6}\left[\frac{h^3}{8} + \frac{h^3}{8}\right]$$

$$= \frac{h^3}{24} f''(x_{\frac{1}{2}})$$

Hence <u>Local</u> Error $= \dfrac{h^3}{24} f''(x_{\frac{1}{2}}) + \ldots$ * FOR ONE INTEGRATION INTERVAL! *

For Global error we add the error @ each step.

For $n = \dfrac{L}{h}$ steps error is proportional to $\dfrac{1}{h}$

$$\boxed{\text{Global Midpoint Error} = \frac{h^2 L}{24} f''(\beta)} \quad \beta \in [L]$$

Question 4 – Richardson extrapolation

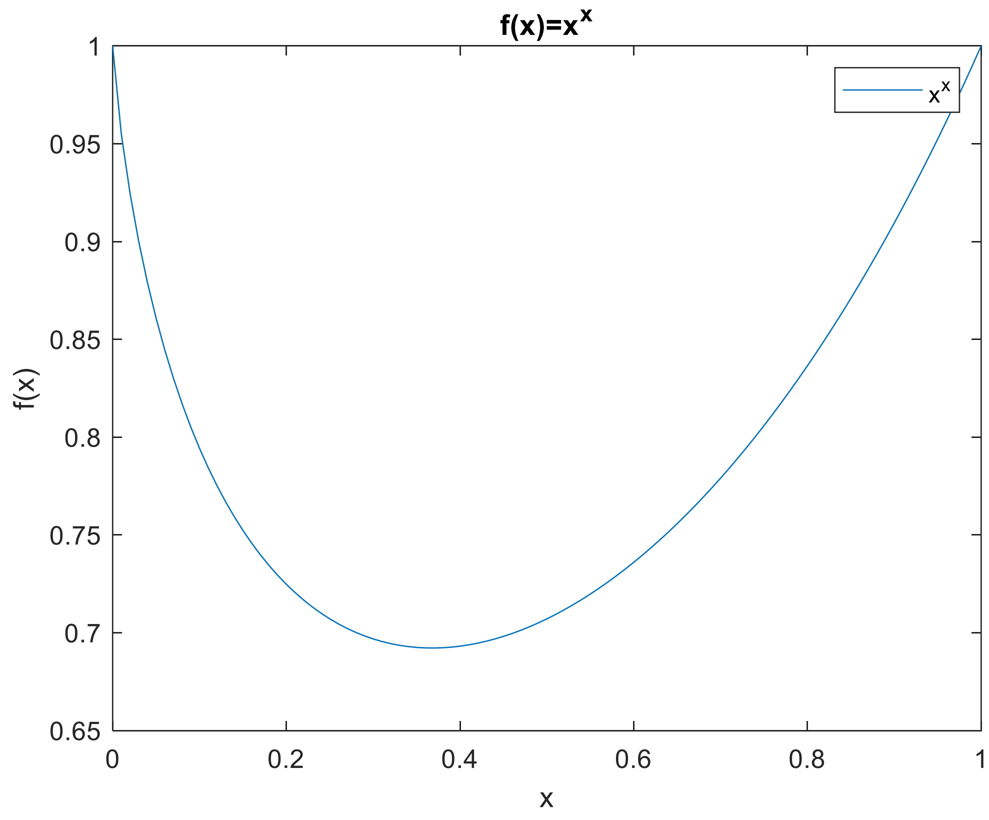
a. Use Richardson's method and the forward difference to gain an accurate estimate for the derivative of

$$f(x) = x^x$$

At $x = 0.3679$

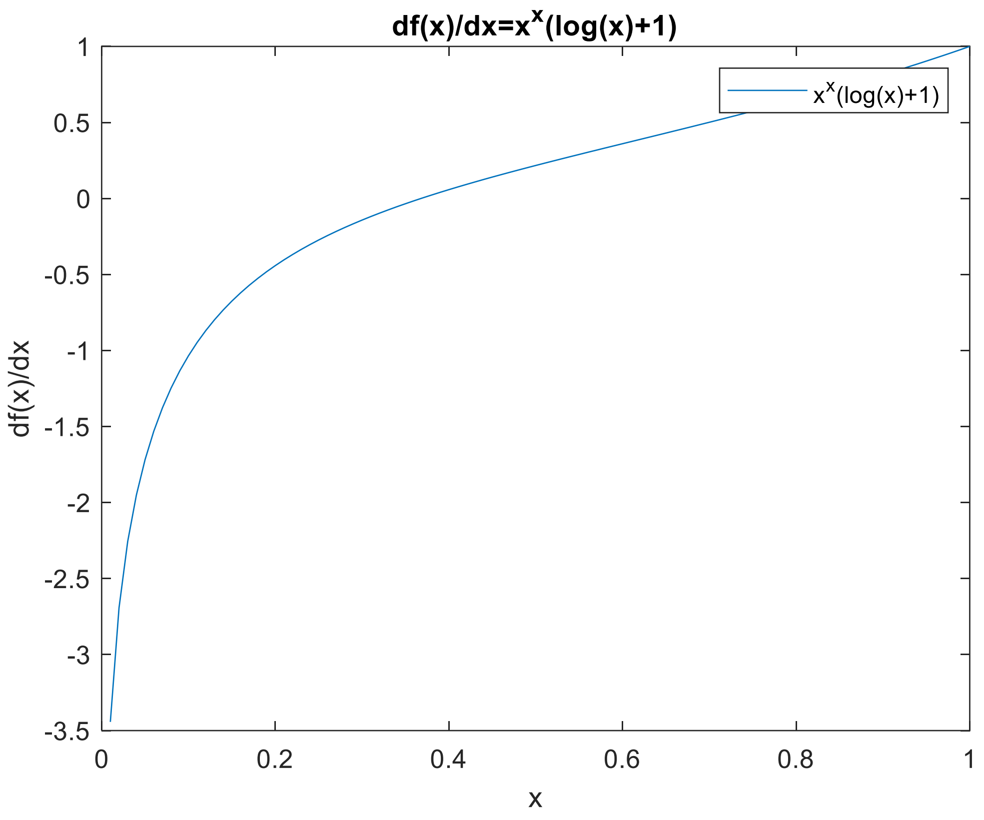
Question 4 – Richardson extrapolation

```
clear all;
close all;
clc;

% We want to estimate dy/dx at x=0.3679
start_point=0.3679;
end_point=1.0;

% To use Richardson extrapolation we need
% several values of step size
step=0.1;
max_step=0.5;
min_step=0.2;

% The actual function and derivative
x = 0:0.01:end_point;
y = (x.^x);
dydx = (x.^x).*(log(x)+1);
dydx_exact=0;

% Plot of f(x)
figure;
plot(x,y,'-');
legend('x^x');
xlabel('x'),ylabel('f(x)');
title('f(x)=x^x');
```



Question 4 – Richardson extrapolation

```
% Plot of f'(x)
figure;
plot(x,dydx,'-');
legend('x^x(log(x)+1)');
xlabel('x'),ylabel('df(x)/dx');
title('df(x)/dx=x^x(log(x)+1)');

% Arrays to hold different step size values
% and forward difference values at those points.
forward_difference_size = [];
h_size = [];

% Open a figure to plot to
figure;
```



Question 4 – Richardson extrapolation

```
% Next step through the step sizes
for h=max_step:-step:min_step

    % Values of x and x+h
    x=start_point;
    x_plus_h=x+h;

    % Forward difference approximation
    forward_difference = ((x_plus_h^x_plus_h)-(x^x))/h;
    % Store this step size and value of forward difference
    forward_difference_size = [ forward_difference_size…
          forward_difference ];
    h_size = [ h_size h];

    % Clear the figure
    clf;

    % Plot the actual function and the forward difference
    subplot(3,1,1);
    fplot(@(x) (x.^x), [0,1], 'b');
    xlabel('x'),ylabel('y');
    hold on;
    y=(x^x);
    y_h=(x_plus_h^x_plus_h);
    plot([x,x_plus_h],[y,y_h],'-o');
    legend('f(x)=x^x');
    axis([0 1 0.6 1]);
```

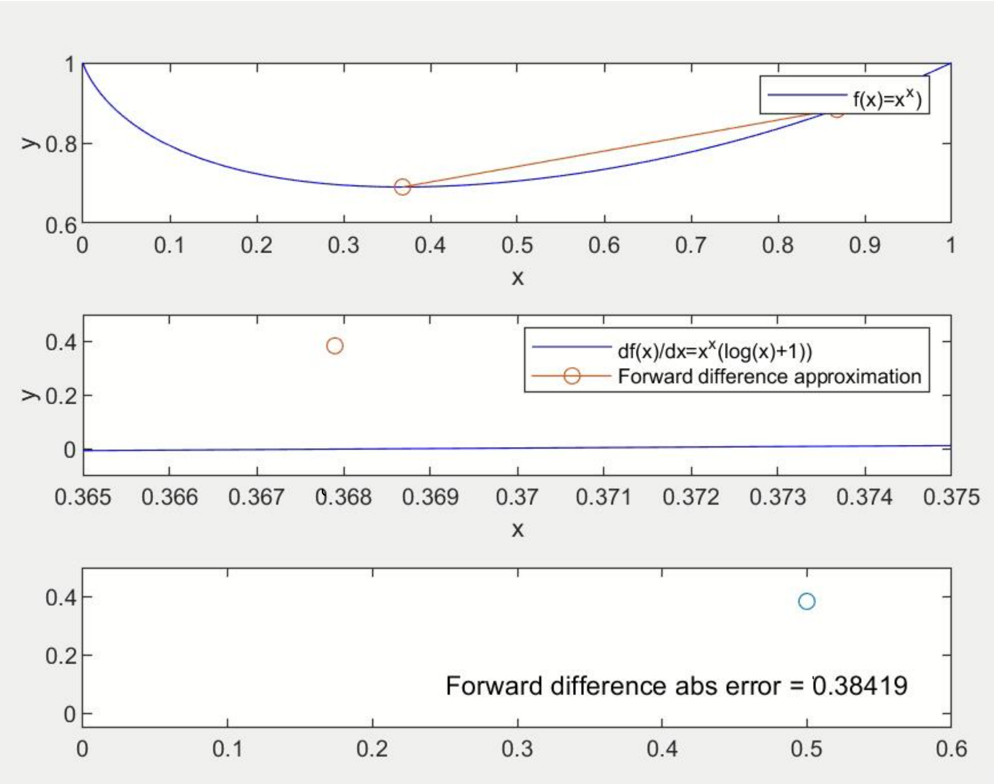
Question 4 – Richardson extrapolation

```
    % Plot the actual df(x)/dx against the forward difference
    % approximation at the x point of interest x=1/e
    subplot(3,1,2);
    fplot(@(x) (x.^x).*(log(x)+1), [0,1], 'b');
    xlabel('x'),ylabel('y');
    hold on;
    plot(x,forward_difference,'-o');
    legend('df(x)/dx=x^x(log(x)+1))','Forward difference…
          approximation');
    axis([0.365 0.375 -0.1 0.5]);

    % Plot the forward difference against step size
    subplot(3,1,3);
    plot(h_size,forward_difference_size,'-o');
    axis([0.0 max_step+0.1 -0.05 0.5]);
    txt = ['Forward difference abs error = '...
        num2str(abs(dydx_exact - forward_difference))'.'];
    text(0.25,0.10,txt);

    pause(0.75);
    drawnow;
end
```

Question 4 – Richardson extrapolation

```
% Fit a polynomial (straight line for O(h) error in our
algorithm
P = polyfit(h_size,forward_difference_size,1);

% Plot the extrapolation based on the above fit
x=[0:0.001:h_size(1)];
yfit = P(1).*x+P(2);
P(2)
hold on;
plot(x,yfit,'r-.');

% Calculate the Richardson extrapolation error and print
r_error=abs(dydx_exact - P(2));
txt = ['Richardson extrapolation abs error = '
num2str(r_error)'.'];
text(0.025,0.415,txt);
drawnow;
```

Question 4 – Richardson extrapolation



Question 4 – Richardson extrapolation

b. Derive the analytic derivative of the function, use this to find the minima of the function.

Question 4 – Richardson extrapolation

Q4// b) $\frac{d}{dx} x^x$?

$$x^x = e^{\ln x^x}$$

but $\ln(a^b) = b \ln(a)$

$$\therefore \frac{d}{dx} x^x = \frac{d}{dx} e^{x \ln x}$$

From chain rule .. $\frac{d}{dx} e^{f(x)} = f'(x)\, e^{f(x)}$

$$= \frac{d}{dx} e^{x \ln x} = e^{x \ln x} \frac{d}{dx} x \ln x$$

Question 4 – Richardson extrapolation

$$= \frac{d}{dx} e^{x \ln x} = e^{x \ln x} \frac{d}{dx} x \ln x$$

but $\frac{d}{dx} x \ln x = x \cdot \frac{1}{x} + 1 \cdot \ln x$

$$\boxed{\Rightarrow \frac{d}{dx} x^x = x^x (\ln x + 1)}$$

Minima $\frac{d}{dx} x^x = 0 \Rightarrow x^x \ln x + x^x = 0$

$$\Rightarrow x^x \ln x = -x^x$$

or $\ln x = -1$

$$\therefore x = 0.367879\ldots$$

Question 5 – Solving ODEs and the shooting method

a. Write a predictor-corrector code to solve the following ODE

$$y'' = \frac{d^2y}{dx^2} = -y$$

Question 5 – Solving ODEs and the shooting method

```
clear all;
close all;
clc;

% Start and end points.
x_start=0.0;
x_end=11.0;

% The analytic solution to compare numerical results
xx=[x_start:0.001:x_end]';
fn=cos(xx);

% Open a figure to plot to
figure;
```

Question 5 – Solving ODEs and the shooting method

```
% Loop over a different number of steps to see how
% step size influences solution
for numSteps=2:1:100;

    % clear the figure
    clf;

    % Calculate step size
    h=(x_end-x_start)/numSteps;

    % The x range for our step size
    x=[x_start:h:x_end]';

    % Size vectors for Euler and PC methods.
    y=x;
    dydx=x;

    y_pc=x;
    dydx_pc=x;

    % Initial values (as in lecture).
    y(1)=1;
    dydx=0;

    y_pc(1)=1;
    dydx_pc=0;
```

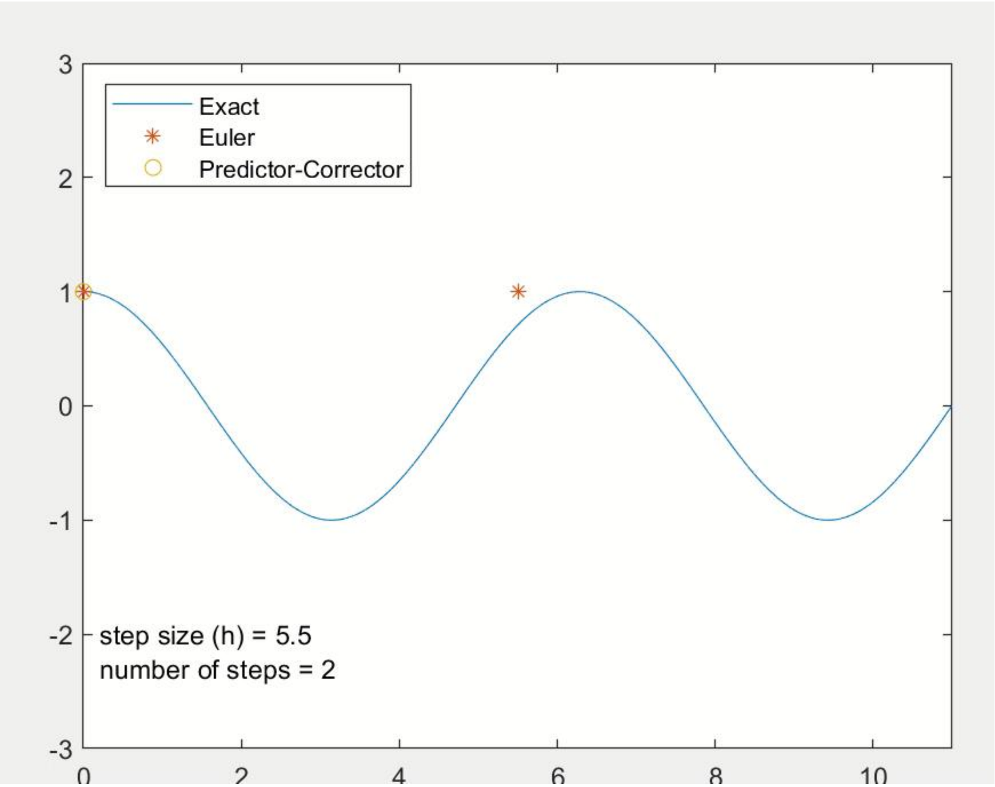
Question 5 – Solving ODEs and the shooting method

```
% Perform Euler
for i=1:(length(x)-1)
    y(i+1)=y(i)+h*dydx(i);
    dydx(i+1)=dydx(i)-h*y(i);
end

% Perform PC
for i=1:(length(x)-1)
    ypred=y_pc(i)+h*dydx_pc(i);
    dydxpred=dydx_pc(i)-h*y_pc(i);

    y_pc(i+1)=y_pc(i)+(h/2.0)*(dydx_pc(i)+dydxpred);
    dydx_pc(i+1)=dydx_pc(i)-(h/2.0)*(y_pc(i)+ypred);
end

% Plot Euler and PC results for this step
plot(xx,fn);
hold on
plot(x,y,'*');
hold on
plot(x,y_pc,'o');
```

Question 5 – Solving ODEs and the shooting method

```
    % Axis, legend, step size adn number of steps
    legend('Exact','Euler','Predictor-Corrector','Location','northwest');
    axis([x_start x_end -3 3]);
    txt = ['step size (h) = ' num2str(h)'.'];
    text(0.2,-2.0,txt);
    txt = ['number of steps = ' num2str(numSteps)'.'];
    text(0.2,-2.3,txt);

    drawnow;

    if mod(numSteps,20)==0
    %     pause(5);
    end
end
```

Question 5 – Solving ODEs and the shooting method



Question 5 – Solving ODEs and the shooting method

b. Given the boundary values $y(0) = 1$ and $y(11) = 0$, find the value of $y'(0)$ using the shooting method. Verify your answer.

Question 5 – Solving ODEs and the shooting method

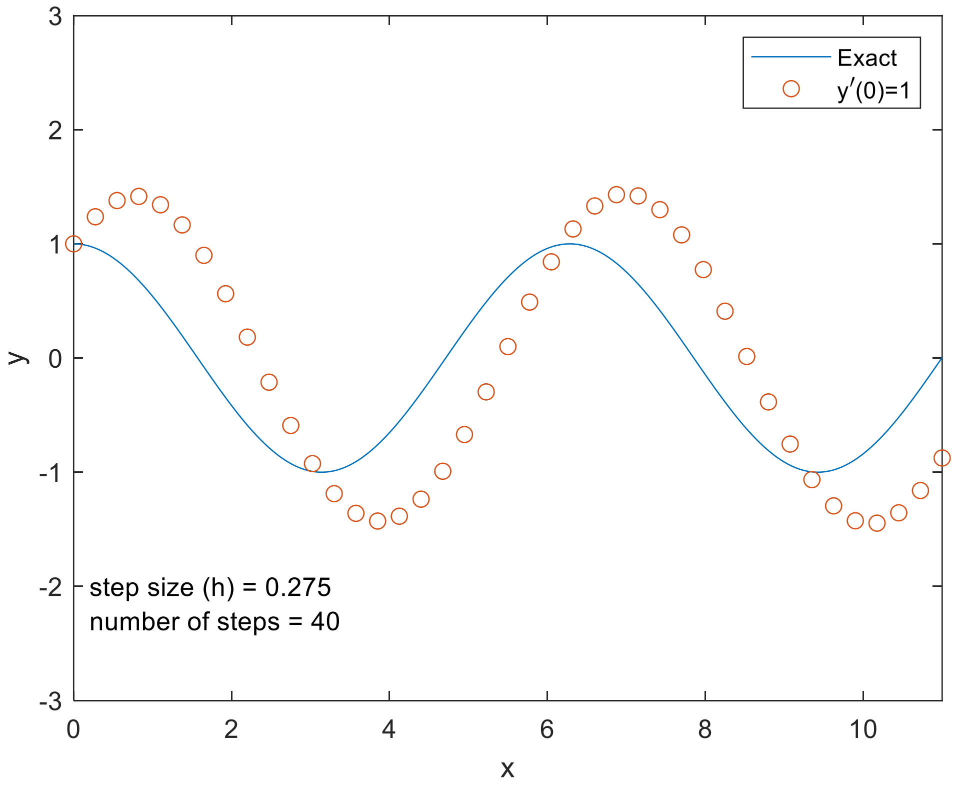
```
clear all;
close all;
clc;

% Start and end points
x_start=0.0;
x_end=11.0;

% The analytic solution to compare numerical results
xx=[x_start:0.001:x_end]';
fn=cos(xx);

% Let's work with a fixed number of steps, 40 is good.
numSteps=40;

% Calculate our step size
h=(x_end-x_start)/numSteps;

% The x points for us to evaluate at
x=[x_start:h:x_end]';
```

Question 5 – Solving ODEs and the shooting method

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Now we make two guesses dy(0)/dx = 1 and -1
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% First Guess...

% Size our vectors
y1_pc=x;
dydx1_pc=x;

% Initial Value y(0)=1
y1_pc(1)=1;

% Initial guess dy(0)/dx = 1
dydx1_pc=1;

% Now perform predictor corrector
for i=1:(length(x)-1)
    ypred=y1_pc(i)+h*dydx1_pc(i);
    dydxpred=dydx1_pc(i)-h*y1_pc(i);
    y1_pc(i+1)=y1_pc(i)+(h/2.0)*(dydx1_pc(i)+dydxpred);
    dydx1_pc(i+1)=dydx1_pc(i)-(h/2.0)*(y1_pc(i)+ypred);
end
```

Question 5 – Solving ODEs and the shooting method

```
% Plot our results for the first guess
figure;
plot(xx,fn);
hold on
plot(x,y1_pc,'o');
hold on

legend('Exact','y^{\prime}(0)=1');
xlabel('x');
ylabel('y');
axis([x_start x_end -3 3]);
txt = ['step size (h) = ' num2str(h)'.'];
text(0.2,-2.0,txt);
txt = ['number of steps = ' num2str(numSteps)'.'];
text(0.2,-2.3,txt);
```

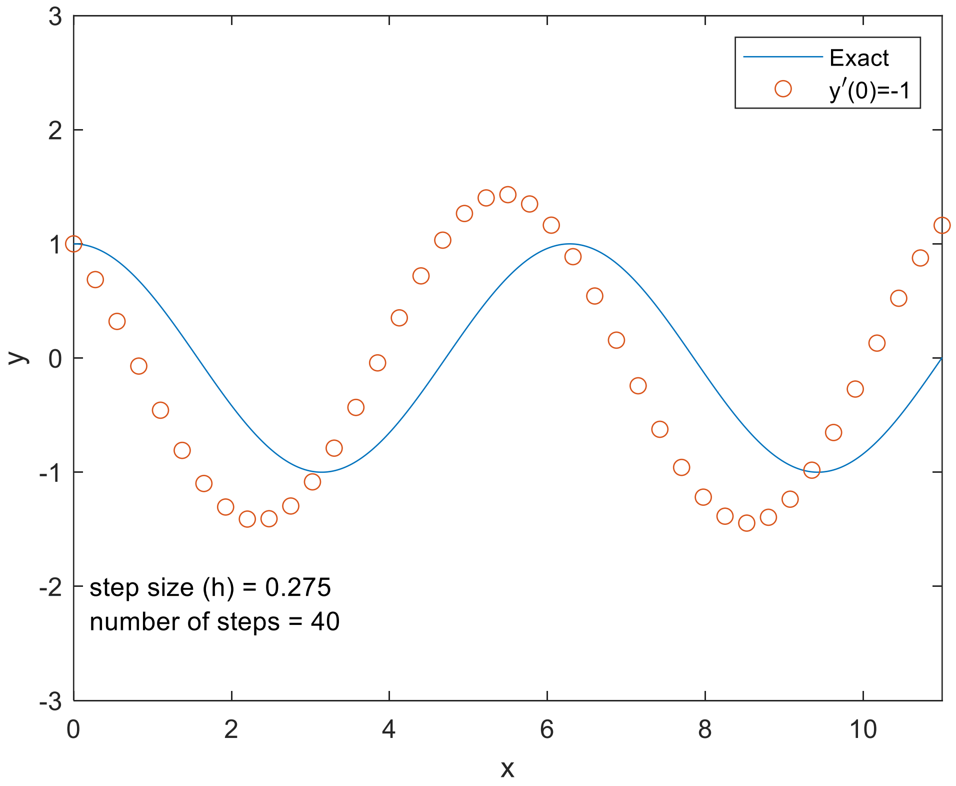


Question 5 – Solving ODEs and the shooting method

```
% Second Guess...

% Size our vectors
y2_pc=x;
dydx2_pc=x;

% Initial Value y(0)=1
y2_pc(1)=1;

% Next guess dy(0)/dx = -1
dydx2_pc=-1;

% Now perform predictor corrector
for i=1:(length(x)-1)
    ypred=y2_pc(i)+h*dydx2_pc(i);
    dydxpred=dydx2_pc(i)-h*y2_pc(i);
    y2_pc(i+1)=y2_pc(i)+(h/2.0)*(dydx2_pc(i)+dydxpred);
    dydx2_pc(i+1)=dydx2_pc(i)-(h/2.0)*(y2_pc(i)+ypred);
end
```

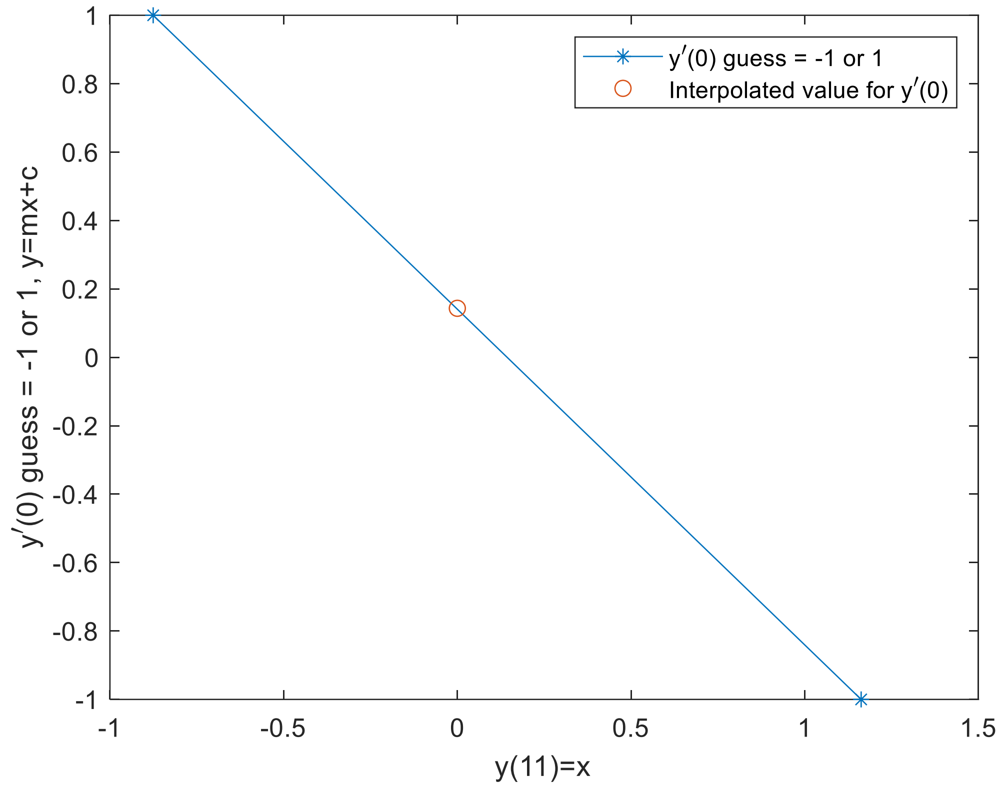
Question 5 – Solving ODEs and the shooting method

```
% Plot our results for the second guess
figure;
plot(xx,fn);
hold on
plot(x,y2_pc,'o');
hold on

legend('Exact','y^{\prime}(0)=-1');
xlabel('x');
ylabel('y');
axis([x_start x_end -3 3]);
txt = ['step size (h) = ' num2str(h)'.'];
text(0.2,-2.0,txt);
txt = ['number of steps = ' num2str(numSteps)'.'];
text(0.2,-2.3,txt);
```



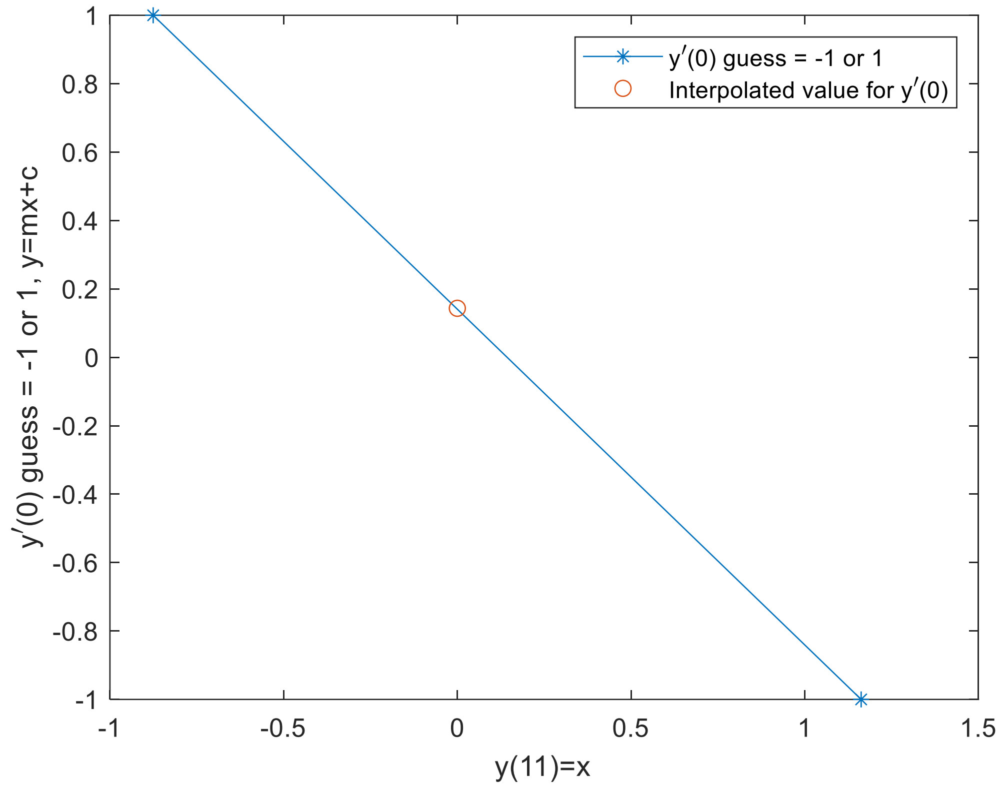
Question 5 – Solving ODEs and the shooting method

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Use our two guesses to interpolate a value for dy(0)/dx
%
% ??? HOW ???
%
% 1. I know that y(11)=0 for the correct value of dy(0)/dx
%       -> It's one of my boundary conditions.
%
% 2. I know when dy(0)/dx =  1, y(11) = a
% 3. I know when dy(0)/dx = -1, y(11) = b
%
% 4. Now, "plot" y(11) values vs dy(0)/dx values,
%       draw a straight line between them.
%
% 5. We know y(11)=0 for the correct value of dy(0)/dx
%       -> Read off the "correct" value of dy(0)/dx, where y(11) = 0
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Create the plot
figure;
plot([ y1_pc(i+1) y2_pc(i+1)],[dydx1_pc(1) dydx2_pc(1)],'-*');
xlabel('y(11)=y=mx+c');
ylabel('y^{\prime}(0) guess = {-1 or 1}');
hold on
```



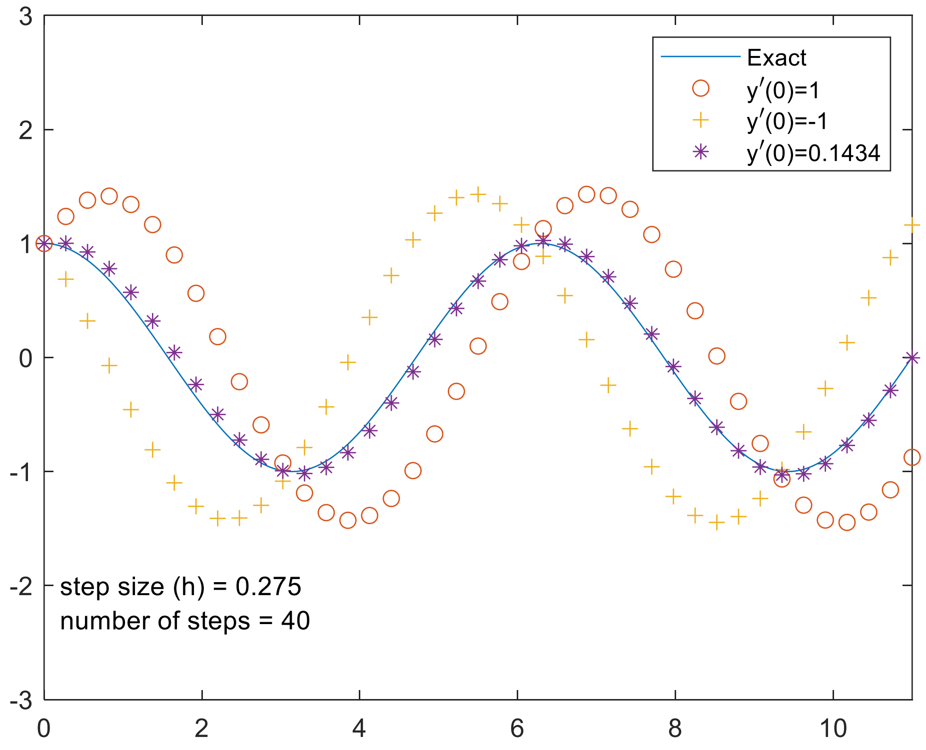
Question 5 – Solving ODEs and the shooting method

```
% Find "correct" value of dy(0)/dx, where y(11) = 0
grad=(y1_pc(i+1) - y2_pc(i+1))/(dydx1_pc(1)-dydx2_pc(1));
const=y1_pc(i+1)-dydx1_pc(1)*grad;

% Plot the result
plot([0],[const],'o');
legend('y^{\prime}(0) guess = {-1 or 1}','Interpolated value for
y^{\prime}(0)');

% Use the "correct/estimate" value of dy(0)/dx and perform PC
again...

ye_pc=x;
dydxe_pc=x;

ye_pc(1)=1;
dydxe_pc=const;

for i=1:(length(x)-1)
    ypred=ye_pc(i)+h*dydxe_pc(i);
    ydpred=dydxe_pc(i)-h*ye_pc(i);
    ye_pc(i+1)=ye_pc(i)+(h/2.0)*(dydxe_pc(i)+ydpred);
    dydxe_pc(i+1)=dydxe_pc(i)-(h/2.0)*(ye_pc(i)+ypred);
end
```



Question 5 – Solving ODEs and the shooting method

```
% Finally plot all three solutions to compare
figure;
plot(xx,fn);
hold on
plot(x,y1_pc,'o');
hold on
plot(x,y2_pc,'+');
hold on
plot(x,ye_pc,'*');

legend('Exact','y^{\prime}(0)=1','y^{\prime}(0)=-
1','y^{\prime}(0)=0.1434');
axis([x_start x_end -3 3]);
txt = ['step size (h) = ' num2str(h)'.'];
text(0.2,-2.0,txt);
txt = ['number of steps = ' num2str(numSteps)'.'];
text(0.2,-2.3,txt);

drawnow;
```



Question 6 – Solving PDEs

a. Solve the following PDE over the range $x = 0$ to $x = 20$, $y = 0$ to $y = 100$, with T=67.5 applied to one of the boundaries (for example T(1:xdim,1) = 67.5) and all other boundary points held at zero.

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$$

Question 6 – Solving PDEs

```
% Clearing variables in memory and Matlab command screen
clear all;
close all;
clc;

% Dimensions of the simulation grid in x (xdim) and y (ydim)
directions
xdim=20; ydim=100;

% Error cutoff
cutoff=0.001;

% Initializing previous (T_prev) and present (T_now) temp
matrices
T_now  = zeros(xdim,ydim);
T_prev = zeros(xdim,ydim);

% Constant temperature of 67.5C applied to one boundary
T_now(1:xdim,1)  = 67.5;

% Open a figure to plot to
figure;

% Set the iteration counter to one
t=1;
```

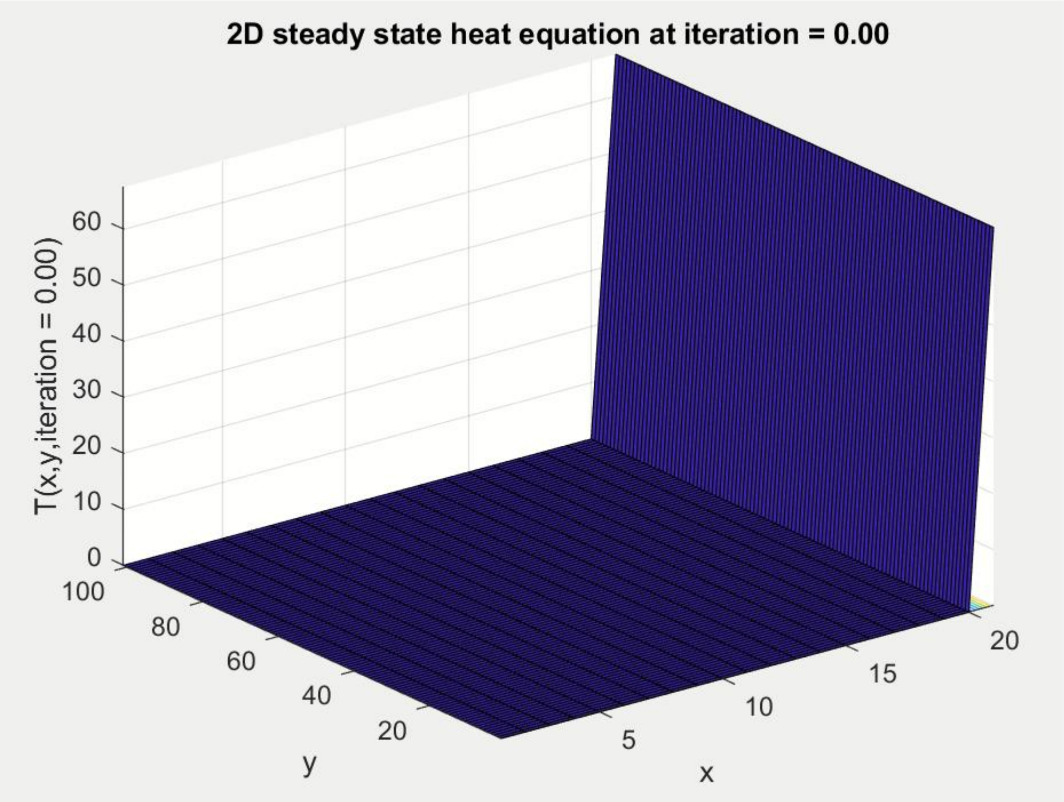
Question 6 – Solving PDEs

```
% Set a starting value for the error > cutoff;
error=2*cutoff;

% Keep iterating until the error between the current solution
% and last solution is below a predefined cutoff value
while error > cutoff

    % Solve for this time step, using the last time steps values
    % over our solution space (mesh).
    for i=2:1:xdim-1
        for j=2:1:ydim-1
            T_now(i,j)=(T_prev(i+1,j)+T_prev(i-1,j)+T_prev(i,j+1)+T_prev(i,j-1))/4.0;
        end
    end

    % Calculate the difference between this time step and the last
    error=max(max(abs(T_now-T_prev)))

    % Now make this time step the last time step.
    T_prev=T_now;
```

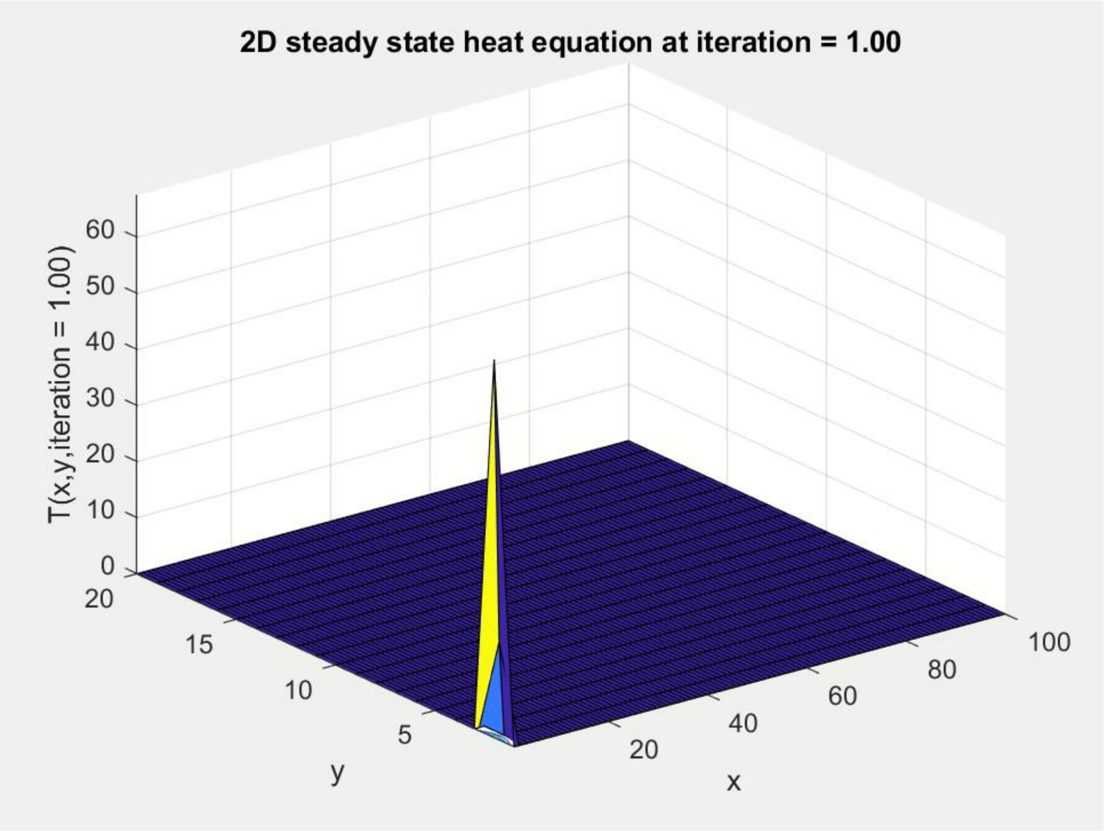
Question 6 – Solving PDEs

```
    % Plot the current time step results
    surfc(T_now);
    title(sprintf('2D steady state heat equation at iteration = %1.2f',t),'Fontsize',11);
    xlabel('x','Fontsize',11); ylabel('y','Fontsize',11);
    zlabel(sprintf('T(x,y,iteration = %1.2f)',t),'Fontsize',11);
    drawnow;

    % Update the iteration counter and go again
    t=t+1;
end
```

Question 6 – Solving PDEs



Question 6 – Solving PDEs



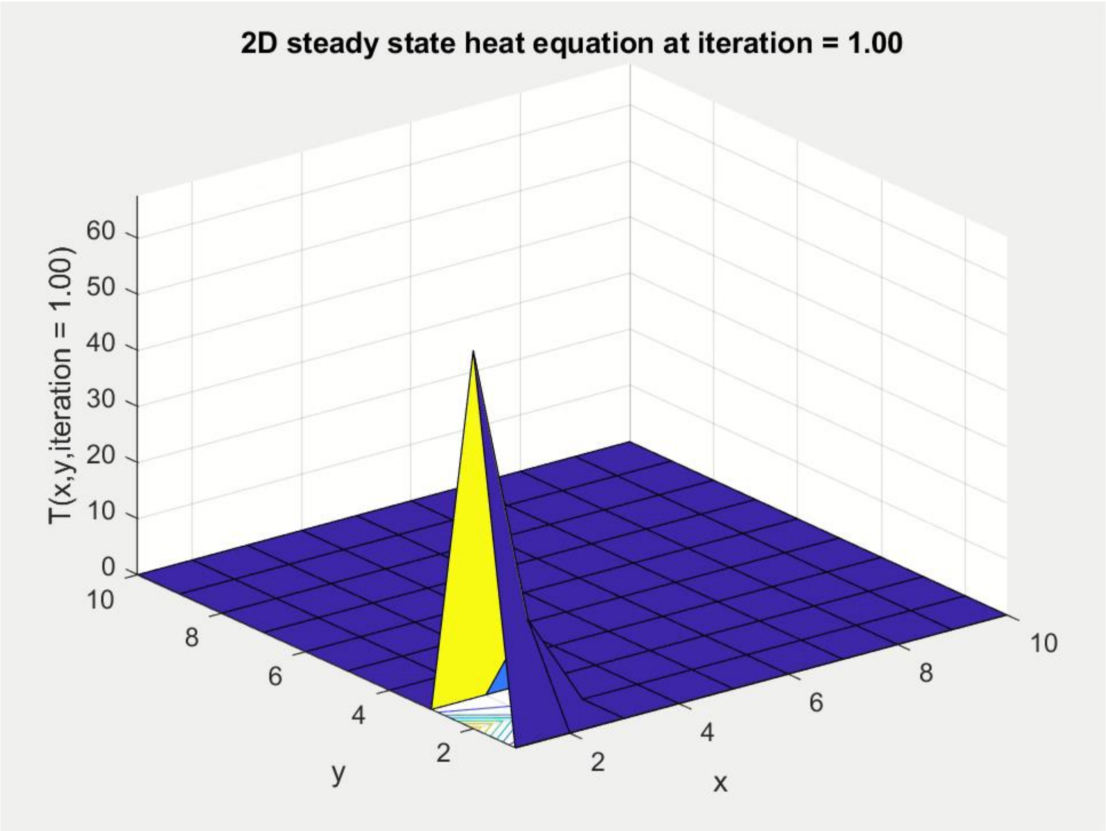
Question 6 – Solving PDEs

b. Suggest how you might reach a quicker numerical solution.

c. Implement your idea in MATLAB.

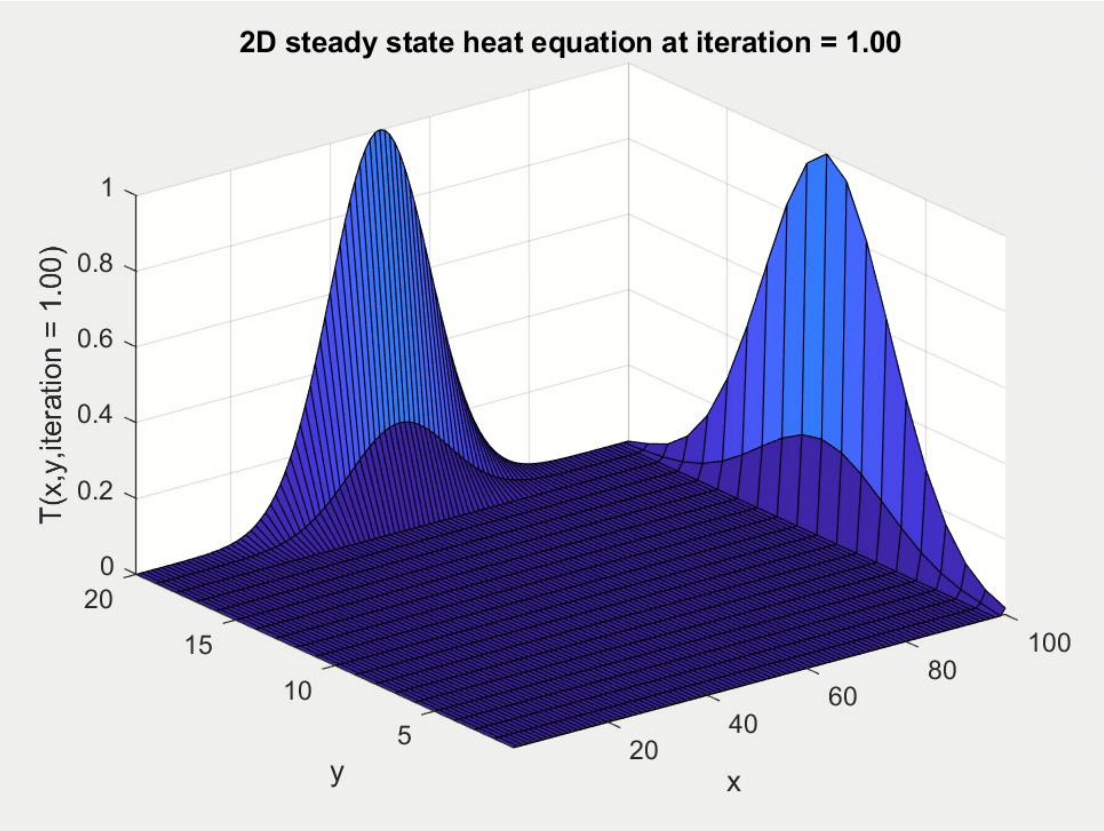
Question 6 – Solving PDEs

```
% Dimensions of the simulation grid in x (xdim) and y (ydim) directions
xdim=10; ydim=10;
```



Question 6 – Solving PDEs

```
.
.
.
% Constant temperature of 67.5C applied to one boundary
%T_prev(2,1) = 67.5;
%T_now(2,1)  = 67.5;

% A simulation less ordinary

i=1:1:xdim; %x-co-ordinates for boundary
T_now(i,ydim)=exp((-1.*((xdim/2-i).*(xdim/2-i)))./(1.0.*xdim));

j=1:1:ydim; %x-co-ordinates for boundary
T_now(xdim,j)=exp((-1.*((ydim/2-j).*(ydim/2-j)))./(2.0.*ydim));

T_prev=T_now;

% Open a figure to plot to
figure;

.
.
.
```

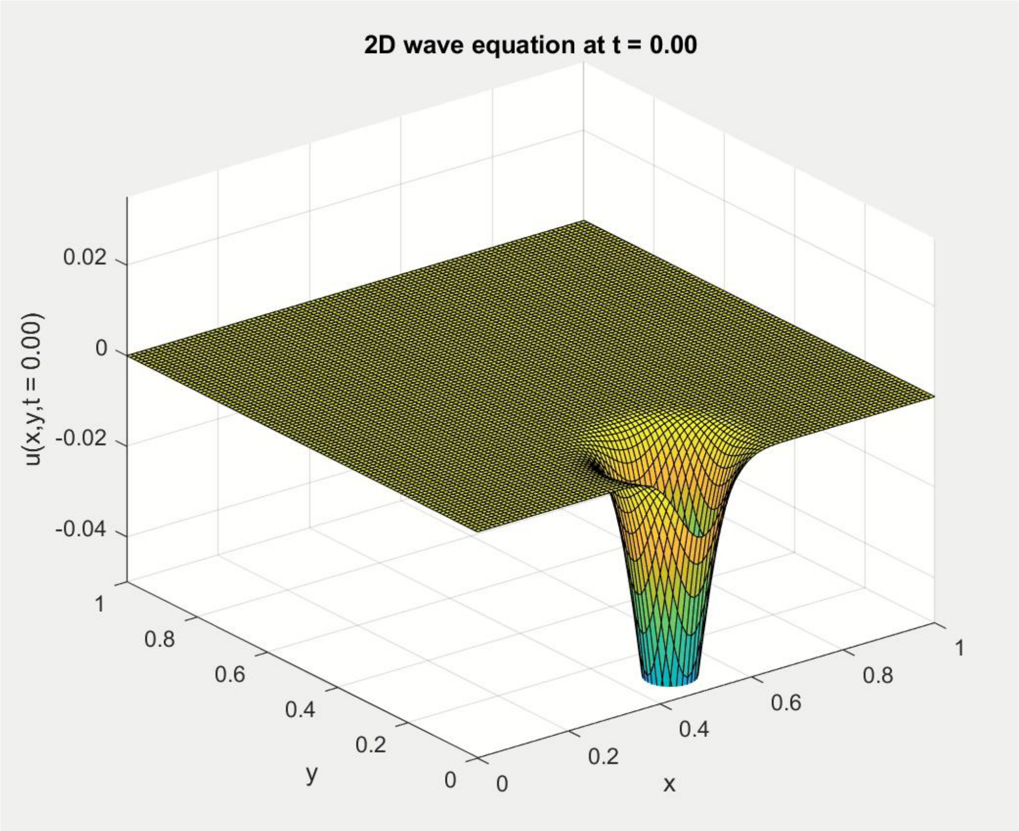


Closing remarks

Hopefully you have enjoyed this course on numerical algorithms and you now have a good grounding in some of the techniques used in numerical methods.

<u>Please, please</u> do leave feedback, positive, or negative, constructive criticism and typo's.

Your feedback helps me improve the course for next year, just as last years students helped me improve the course for you.




That's all Folks!