Index

SL No.	Experiment Name	Page
1	Solve following 1st order Ordinary Differential Equation (ODE) using Laplace transformation and plot it: *dy* + 4*y* = 2, *y*(0) =1 *dt*	**02-03**
2	Solve following 2nd order Ordinary Differential Equation (ODE) using Laplace transformation and plot it: y″+3y′+2y=0 with initial conditions: y(0)=4, y′(0)=0	**04-05**
3	Solve following 2nd order Ordinary Differential Equation (ODE) using Laplace transformation and plot it: y" + 3y' + 2y = e^{-t}, initial condition y(0)=4 , y'(0)=5	**05-08**
4	Use the Laplace transform to find the unique solution *y*"−*y*'−2*y*= 0, y(0) =1, y'(0) = 2. If y(0)=1, y′(0)=0, find the solution of this equation.	**08-09**
5	Find solution of y″ = 6 – 2x with initial conditions: y(0)=1, y′(0)=4	**09-11**
6	Find solution of y″−6y′+15y=2sin(3t),withinitialconditions:y(0)=−1,y′(0)=−4., and plot it	**11-13**
7	Construct a complex signal of frequency f1=50 Hz, f2=100 Hz and F3=200 Hz and Amplitude A1=A2=A3=1 using Fourier Transformation technique. Plot its time domain and frequency domain	**14-15**
8	Write MATLAB code is used to plot the square wave function along with the Fourier sine series in order to compare the accuracy and error between the approximation and the actual function. [Use the equation □ *F*(*x*) = 4/□□1/*j*sin(*j*□*x*)] *j=odd*	**15-16**
9	Convolute two simple signals of unit step and exponential decay: • x(t) = u(t) → unit step function • h(t) = e^{-t} → exponential decay starting at t=0t = 0t=0 We'll compute: y(t)=x(t)∗ h(t)	**17-19**

Tips:

- Always define your symbolic variables with **syms**.
- Use **laplace()** and **ilaplace()** for forward and inverse transforms.
- **subs()** is used to substitute initial conditions after taking the Laplace transform.
- Use **solve()** to isolate the Laplace domain function.
- Always simplify the final result with **simplify()** or **pretty()** for readability.
- Use **fplot()** or **ezplot()** to plot the result.

Problem List of

1. Solve following 1st order Ordinary Differential Equation (ODE) using Laplace transformation $\boxed{\frac{dy}{dt} + 4y = 2, \quad y(0) = 1}$ and plot it:

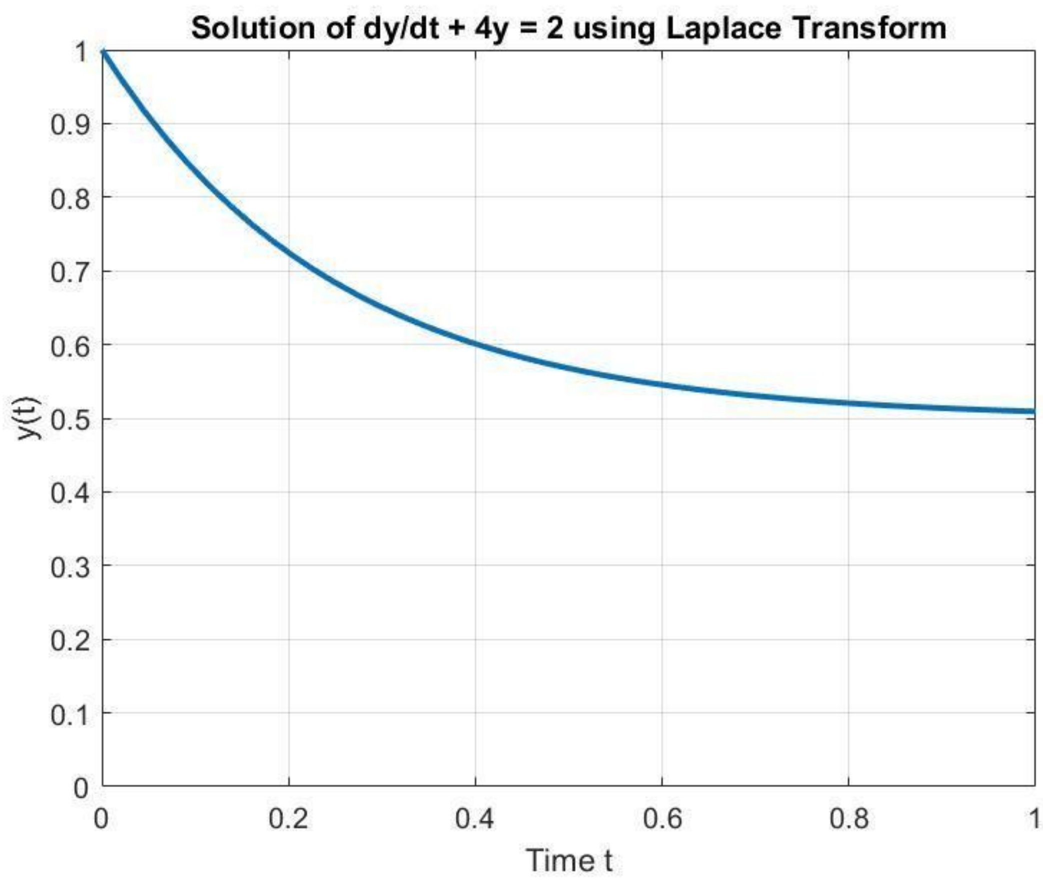
Algorithm:

```
START Algorithm_FirstOrderODE
1.   DEFINE symbolic variables y(t), Y, s
2.   DEFINE ODE: dy/dt + 4y = 2
3.   APPLY Laplace transform to ODE
4.   SUBSTITUTE initial condition y(0) = 1
5.   SOLVE for Y(s) in Laplace domain
6.   APPLY inverse Laplace transform to get y(t)
7.   SIMPLIFY the solution
8.   PLOT y(t) vs t
9.   DISPLAY solution
END Algorithm_FirstOrderODE
```

Mathematical Steps:

1. L{dy/dt} = sY - y(0)
2. L{4y} = 4Y
3. L{2} = 2/s
4. Solve: (sY - 1) + 4Y = 2/s
5. Y(s) = (s + 2)/(s(s + 4))

6. Partial fractions: Y(s) = 1/(2s) + 1/(2(s+4))
7. y(t) = 1/2 + (1/2)e^(-4t)



MATLAB CODE:

```
syms y(t) Y s

% Define the ODE
Dy = diff(y, t); ode =
Dy + 4*y == 2;

% Take Laplace Transform of both
sides
L_ode = laplace(ode, t, s);

% Substitute initial condition y(0) = 1
L_ode_sub = subs(L_ode,
laplace(y(t), t, s), Y);
L_ode_sub = subs(L_ode_sub, y(0),
1);

% Solve for Y (Laplace of y)


Y_sol = solve(L_ode_sub, Y); %
Take inverse Laplace to find y(t)
y_sol = ilaplace(Y_sol, s, t);

% Simplify and display result
y_simplified = simplify(y_sol);
disp('Solution y(t):');
pretty(y_simplified)

% Optional: Plot the result
fplot(y_simplified, [0, 5], 'LineWidth',
2); xlabel('Time
t');
ylabel('y(t)');
title('Solution of dy/dt + 4y = 2 using
Laplace Transform'); grid on;
```

2. Solve following 2nd order Ordinary Differential Equation (ODE) using Laplace transformation and plot it: $\boxed{y''+3y'+2y=0 \quad \text{with initial conditions:}}$ $\boxed{y(0)=4,\ y'(0)=0}$

Algorithm:

START

1. DEFINE symbolic variables y(t),
    1. Characteristic equation: $r^2 + 3r + 2 = 0$
    2. Roots: $r = -1, -2$
    3. $Y(s) = (4s + 12)/((s+1)(s+2))$
    4. Partial fractions: $Y(s) = 8/(s+1) - 4/(s+2)$
    5. $y(t) = 8e^{(-t)} - 4e^{(-2t)}$

   Y, s

2. DEFINE ODE: $y'' + 3y' + 2y = 0$

3. DEFINE initial conditions: $y(0)=4$, $y'(0)=0$ 4. APPLY

Laplace transform:

- $L\{y''\} = s^2Y - s*y(0) - y'(0)$
- $L\{y'\} = sY - y(0)$
- $L\{y\} = Y$

Mathematical Steps:

5. FORM equation: $(s^2Y - 4s) + 3(sY - 4) + 2Y = 0$

6. SOLVE for Y(s)

7. APPLY inverse Laplace transform 8. SIMPLIFY y(t)

9. PLOT solution

END Algorithm_SecondOrderHomogene o us

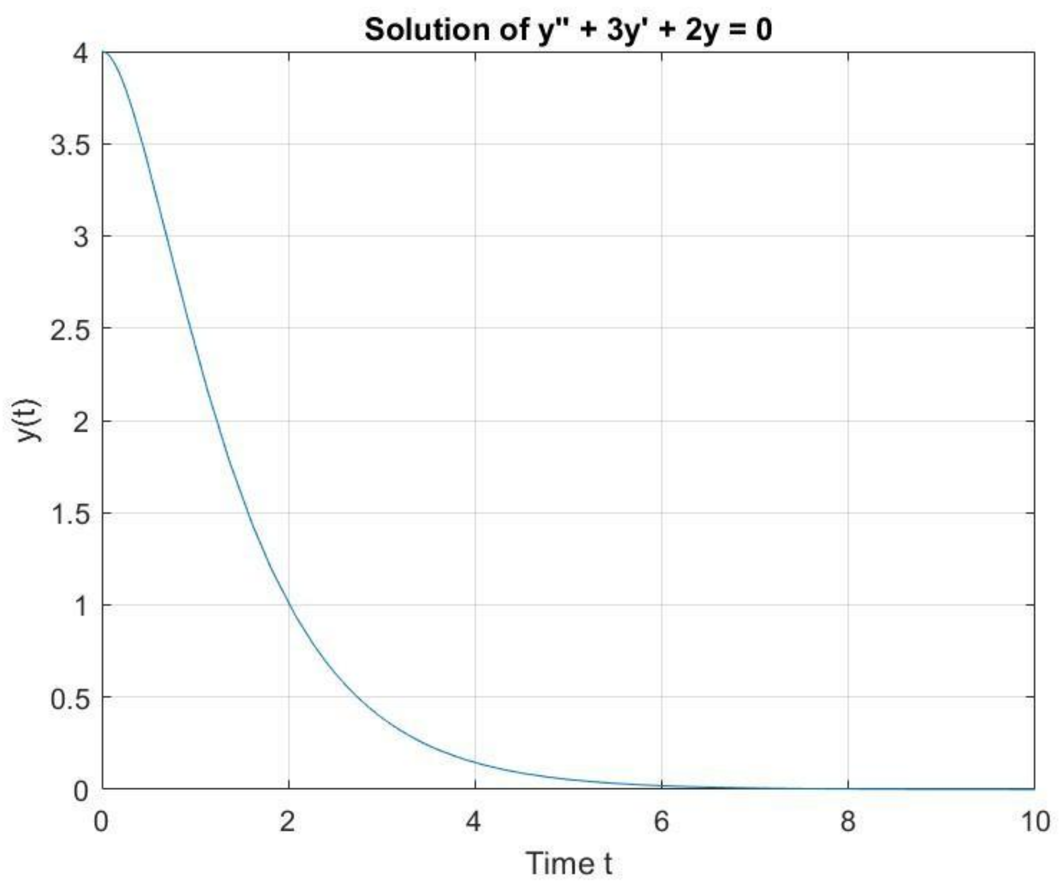
MATLAB CODE:

```
% Define symbolic variables
syms y(t) Y s
% Define the differential equation
Dy = diff(y, t);
D2y = diff(y, t, 2);
% Original ODE ode = D2y + 3*Dy +
2*y == 0; % Initial conditions cond1 =
y(0) == 4; cond2 = Dy(0) == 0; %
Solve using dsolve (symbolic solver)
```

sol = dsolve(ode, [cond1, cond2]); % Display solution disp('Solution y(t):'); pretty(sol)

% To plot this solution we use following code of MATLAB

fplot(sol, [0, 10]) grid on title('Solution of y'''' + 3y'' + 2y = 0') xlabel('Time t') ylabel('y(t)')



3. Solve following 2nd order Ordinary Differential Equation (ODE) using Laplace transformation and plot it: $\boxed{y'' + 3y' + 2y = e^{-t}, \text{ initial condition } y(0)=4\ ,\ y'(0)=5}$

Algorithm:

START

1. DEFINE symbolic variables y(t), Y, s
2. DEFINE ODE: y'' + 3y' + 2y = 0
3. DEFINE initial conditions: y(0)=4, y'(0)=0
4. APPLY Laplace transform:

- L{y''} = s^2Y - s*y(0) - y'(0)
- L{y'} = sY - y(0)
- L{y} = Y

Mathematical Steps:

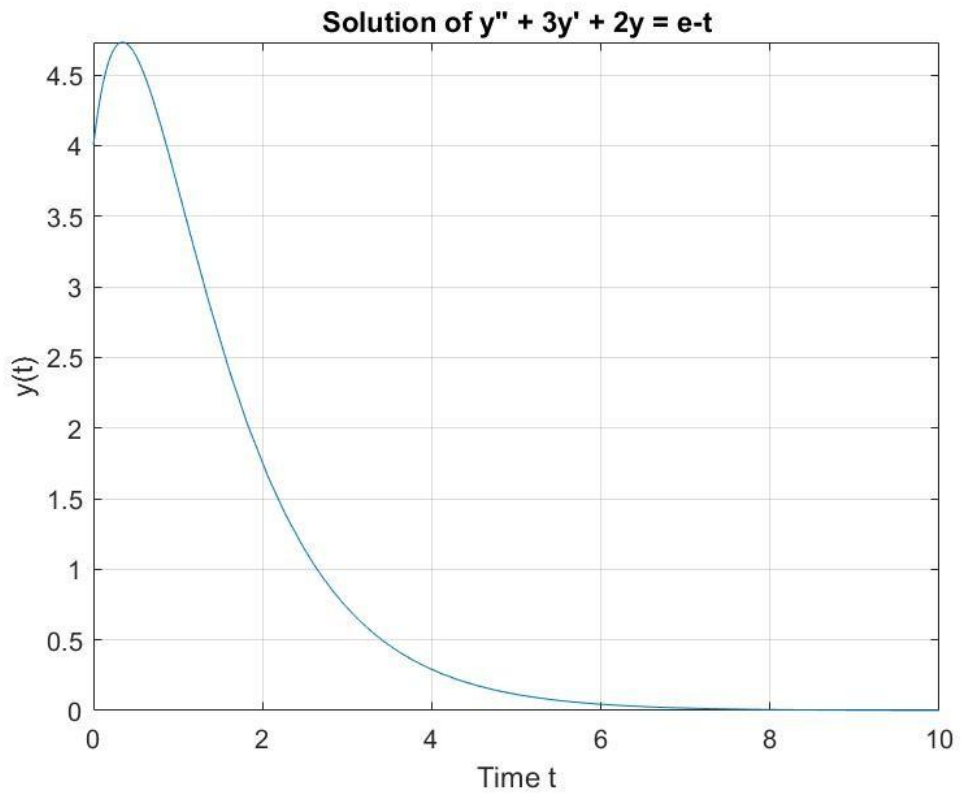
5. FORM equation: (s^2Y - 4s) + 3(sY - 4) + 2Y = 0
6. SOLVE for Y(s)

```
7. APPLY inverse Laplace transform
8. SIMPLIFY y(t)
9. PLOT solution
END
Algorithm_SecondOrderHomogeneo
us
```

1. Characteristic equation: $r^2 + 3r + 2 = 0$
2. Roots: $r = -1, -2$
3. $Y(s) = (4s + 12)/((s+1)(s+2))$
4. Partial fractions: $Y(s) = 8/(s+1) - 4/(s+2)$
5. $y(t) = 8e^{-t} - 4e^{-2t}$

MATLAB CODE:

```
clear all;
% Consider the initial value
problem
% y'' + 3y' + 2y = e-t, y(0)=4 ,
y'(0)=5

% Define the necessary
symbolic variables: syms s t
Y

% Define the right-hand side
function and find its Laplace
transform:
f = exp(-t); F
=
laplace(f,t,s);

% Find the Laplace transform of
y'(t) :
Y1 = s Y - y(0)
Y1 = s*Y - 4;

% Find the Laplace transform of
y''(t) :
Y2 = s Y1 - y'(0)
Y2 = s*Y1 - 5;

% Set the Laplace transform of the left
hand side minus
% the right hand side to zero and solve for
Y:
Sol = solve(Y2 + 3*Y1 + 2*Y - F, Y);

% Find the inverse Laplace transform
of the solution: sol = ilaplace(Sol,s,t);

% Display solution disp('Solution
y(t):');

pretty(sol)  %To plot this
solution we use following code
of MATLAB
fplot(sol, [0, 10]) grid
on
title('Solution of y'''' + 3y'' + 2y = e-t')
xlabel('Time t')
ylabel('y(t)')
```



4. Use the Laplace transform to find the unique solution $y'' - y' - 2y = 0, \quad y(0) = 1$, y = 2. If $\boxed{y(0)=1, y'(0)=0}$, find the solution of this equation.

Algorithm

```
START
1. DEFINE symbolic variables
2. DEFINE ODE: y" - y' - 2y = 0
3. DEFINE IC: y(0)=1, y'(0)=0
4. DIRECT Laplace formulation:
  - Y_eq = s²Y - s - (sY - 1) - 2Y = 0
5. SOLVE for Y(s)
6. INVERSE Laplace to get y(t) 7.
   PLOT solution
END Algorithm_SecondOrderZeroRHS
```

MATLAB CODE:

```
syms Y s t

% Define Laplace transform of y(t) as
Y(s)

% Laplace of y" - y' - 2y = 0 (s^2*y-
sy(0)y'(0)-(sy-y(0))-2y=0)...y(0)=1

% & y'(0)=0;

Y_eq = s^2*Y - s*1 - 0 - (s*Y - 1) -
2*Y
== 0;

% Solve for Y(s)
```

Y_s = solve(Y_eq, Y); %

Inverse Laplace to get

y(t) y_t = ilaplace(Y_s, s,

t);

% Display the solution

disp('Solution y(t):'); pretty(y_t)

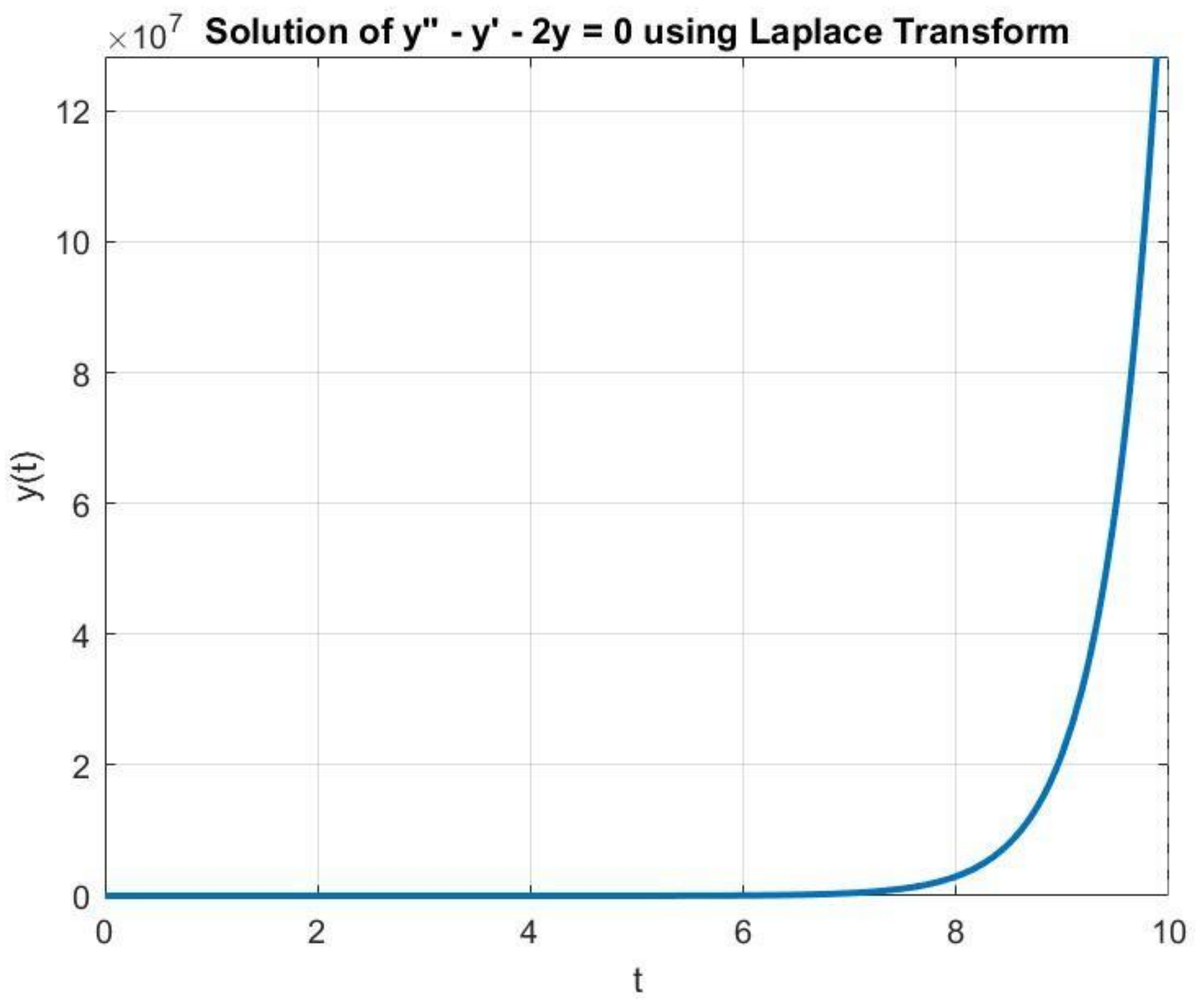
% Plot the solution fplot(y_t, [0,

10], 'LineWidth', 2)

title('Solution of y'''' - y'' - 2y = 0 using

Laplace Transform') xlabel('t')

ylabel('y(t)') grid on



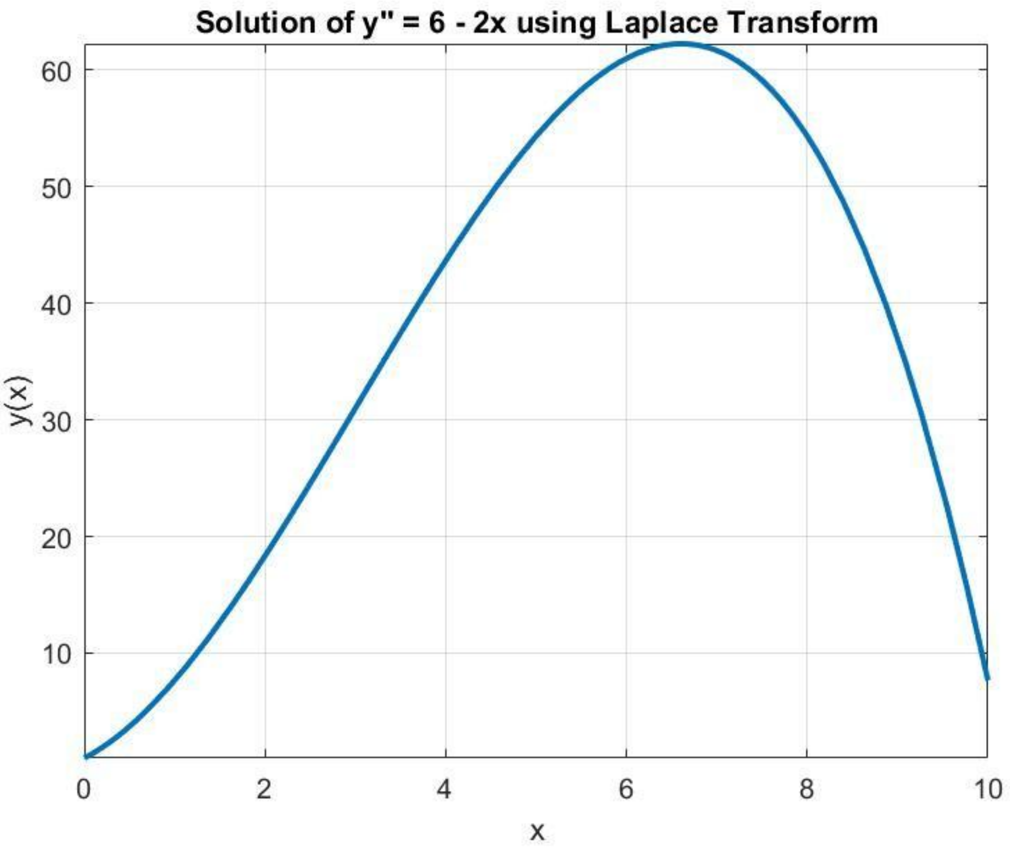
5. Find solution of $\boxed{y'' = 6 - 2x \text{ with initial conditions: } y(0)=1,\ y'(0)=4}$

Algorithm

```
START

1. DEFINE symbolic variables

2. DEFINE ODE: y" = 6 - 2x

3. DEFINE IC: y(0)=1, y'(0)=4

4. FIND Laplace of RHS: L{6-2x}

5. APPLY Laplace to LHS: L{y"} = s²Y - s - 4

6. SOLVE: s²Y - s - 4 = L{6-2x}

7. INVERSE Laplace for y(x)

8. PLOT y(x) vs x

END Algorithm_PolynomialODE
```

MATLAB CODE:

```
% Define symbolic variables
syms Y s x
% Define Laplace transform of RHS: L{6 - 2x}
F = laplace(6 - 2*x, x,
s); % Initial conditions
y0 = 1; dy0 = 4;
% Laplace transform of y" = 6 - 2x
% L{y"} = s^2*Y(s) - s*y(0) - y'(0)
LHS = s^2*Y - s*y0 - dy0;
% Solve for Y(s)
Ysol = solve(LHS == F, Y);
% Inverse Laplace to get y(x)
ySol(x) = ilaplace(Ysol, s, x);
ySol = simplify(ySol); % Display
```



```
the solution disp('Solution y(x)
using Laplace transform:')
disp(ySol)


% Plot the solution over [0, 10]
fplot(ySol, [0, 10], 'LineWidth', 2)
grid on xlabel('x') ylabel('y(x)')
title('Solution of y''' = 6 - 2x using Laplace
Transform')
```

3. DEFINE IC: y(0)=-1, y'(0)=4

4. FIND Laplace of RHS: L{2sin(3t)} = 6/(s²+9)

5. APPLY Laplace to LHS:

   - L{y''} = $s^2Y + s - 4$
   - L{y'} = sY + 1
   - L{y} = Y

6. FORM equation: $(s^2Y + s - 4) - 6(sY + 1) + 15Y = 6/(s^2+9)$

6. Find solution of $\boxed{y''-6y'+15y=2\sin(3t),\ \text{with initial conditions: } y(0)=-1, y'(0)=-4}$., and plot it **<u>Algorithm:</u>**

START

1. DEFINE symbolic variables

2. DEFINE ODE: y'' - 6y' + 15y = 2sin(3t)

3. DEFINE IC: y(0)=-1, y'(0)=4

4. FIND Laplace of RHS: L{2sin(3t)} = 6/(s²+9)

7. SOLVE for Y(s)

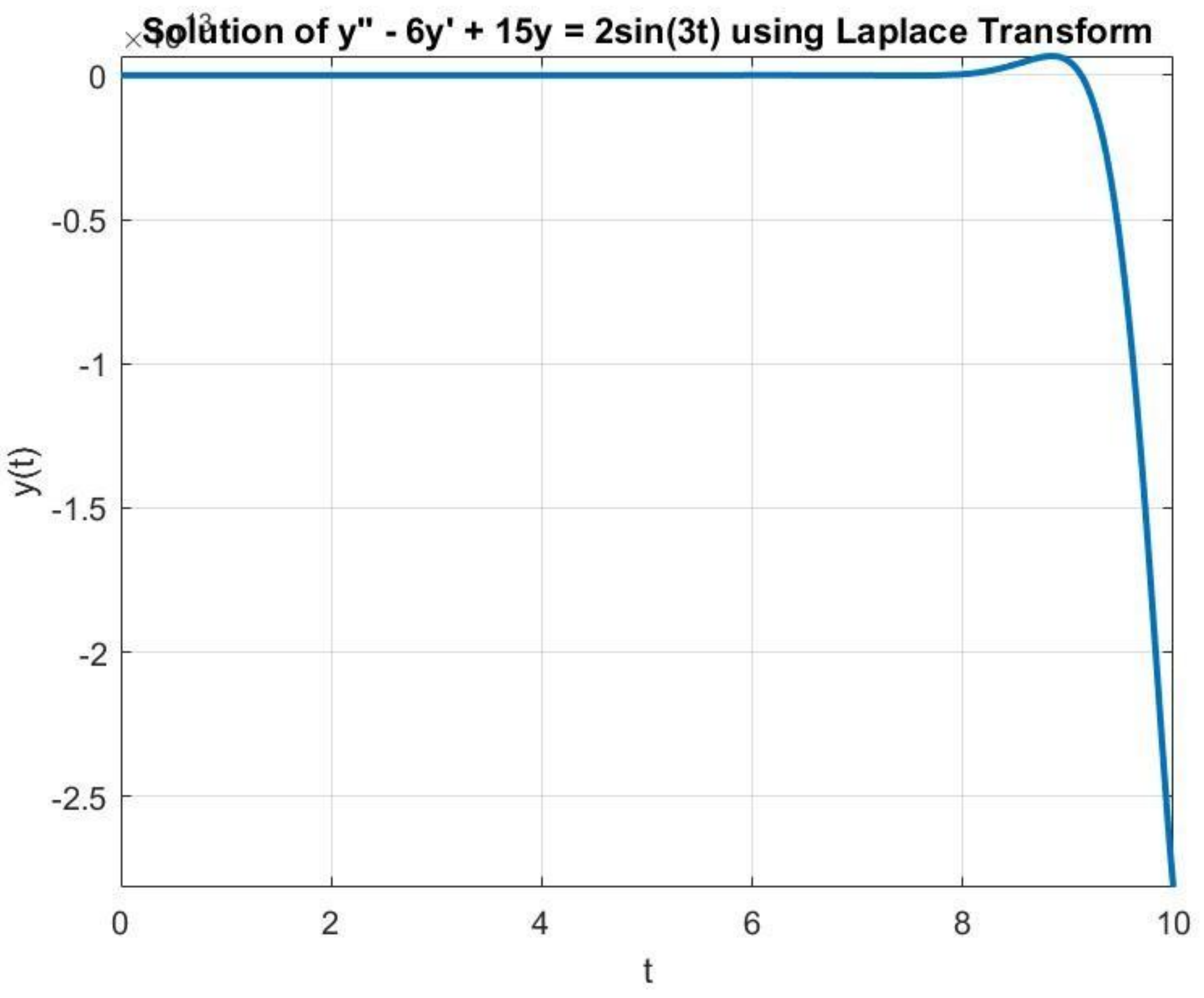
8. INVERSE Laplace for y(t) 9. PLOT

   solution

END Algorithm_SinusoidalForcing

MATLAB CODE:

```
% Define symbolic variables
syms Y s t
% Define Laplace transform of RHS
F = laplace(2*sin(3*t), t,
s); % Define initial
conditions y0 = -1; dy0 =
4;
% Laplace transform of the differential
equation
% L{y''} = s^2*Y - s*y(0) - y'(0)
% L{y'} = s*Y - y(0)
% L{y} = Y
LHS = s^2*Y - s*y0 - dy0 - 6*(s*Y - y0) +
15*Y;
% Solve for Y(s)

Ysol = solve(LHS == F, Y); %
Inverse Laplace to get y(t) ySol =
ilaplace(Ysol, s, t); ySol =
simplify(ySol); % Display the
solution disp('Solution y(t) using
Laplace transform:') disp(ySol)
% Plot the solution over [0, 10] fplot(ySol,
[0, 10], 'LineWidth', 2) grid on xlabel('t')
ylabel('y(t)') title('Solution of y'''' - 6y'' +
15y = 2sin(3t) using Laplace Transform')
```



Fourier transform using MATLAB

- The Fourier Transform is a mathematical technique that transforms a function of time (or space) into a function of frequency.
- It as a way to break down complex signals into their basic building blocks — sine and cosine waves of various frequencies.

- The **Inverse Fourier Transform** is the process of **converting a signal from the frequency domain back to the time domain**. It's like unmixing the frequencies to reconstruct the original signal.
- If the Fourier Transform breaks a signal down into its sine and cosine components (i.e., analyzes *what frequencies are present*), then the Inverse Fourier Transform tells you *how to put them back together* to recreate the original signal.
- `fft(x)` → Transforms from **time to frequency**
- `ifft(X)` → Transforms from **frequency to time** ○ Here bold **f** indicates **Fast** Fourier transform, a fast calculation technique. You will learn about it in Digital Signal Processing.

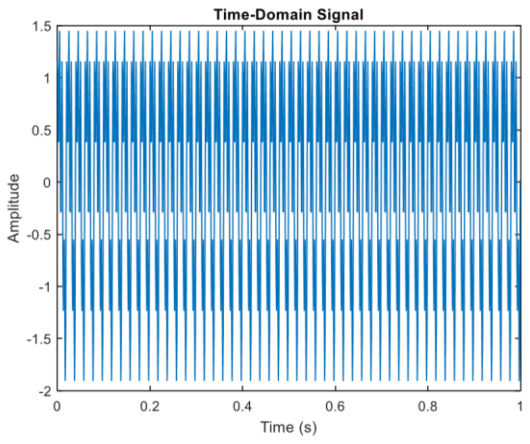
7. Construct a complex signal of frequency f1=50 Hz, f2=100 Hz and F3=200 Hz and Amplitude A1=A2=A3=1 using Fourier Transformation technique. Plot its time domain and frequency domain

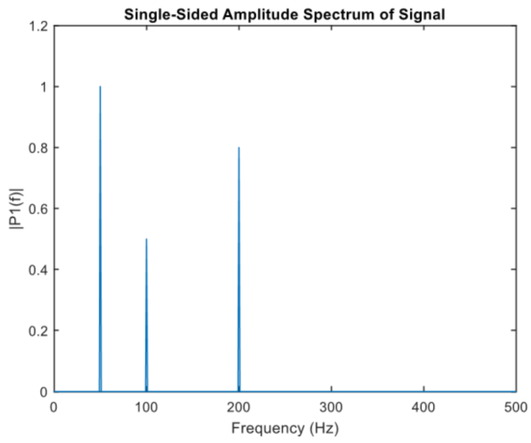
Algorithm:

```
START
1. SET parameters: Fs=1000Hz, T=1/Fs,
   L=1000
2. CREATE time vector t
3. DEFINE frequencies: f1=50Hz,
   f2=100Hz, f3=200Hz
4. CREATE composite signal: sum of
   3 sine waves
5. PLOT time domain signal
6. APPLY FFT to signal
7. COMPUTE frequency axis
8. COMPUTE single-sided amplitude
   spectrum
9. PLOT frequency spectrum
END
```

```
% Solution
% Parameters
Fs = 1000;          % Sampling
frequency (Hz)
T = 1/Fs;          % Sampling period
(s)
L = 1000;           % Length of signal
t = (0:L-1)*T;       % Time vector

% Construct a complex signal (sum
of sine waves) f1 = 50;            %
Frequency 1 (Hz) f2 = 100;
% Frequency 2 (Hz) f3 = 200;
% Frequency 3 (Hz)

% Amplitudes and phases
A1 = 1;
A2 = 0.5;
A3 = 0.8;
phi1 = 0;
phi2 =
pi/4; phi3
= pi/2; %
Time-
domain
signal
signal =
A1*sin(2*
pi*f1*t +
phi1) +
...
    A2*sin(2*pi*f2*t + phi2) + ...
    A3*sin(2*pi*f3*t + phi3);

% Plot the time-domain signal
figure; plot(t, signal);
title('Time-Domain Signal');
xlabel('Time (s)');
ylabel('Amplitude');

% Compute the FFT (Fast Fourier
Transform)
```

```
Y = fft(signal);

% Compute frequency axis f
= Fs*(0:(L/2))/L;
P1 = P2(1:L/2+1);    % Single-sided
spectrum
P1(2:end-1) = 2*P1(2:end-1);

% Plot the frequency domain
representation

% Compute magnitude
P2 = abs(Y/L);      % Two-sided
spectrum
 figure; plot(f,
 P1);
 title('Single-Sided Amplitude
 Spectrum of Signal');
 xlabel('Frequency (Hz)');
 ylabel('|P1(f)|');
```





8. Write MATLAB code is used to plot the square wave function along with the Fourier sine series in order to compare the accuracy and error between the approximation and the actual function. [Use the equation $F(x) = 4/\pi \sum^{\infty}{}_{j=odd} 1/ jsin(j\pi x)$]

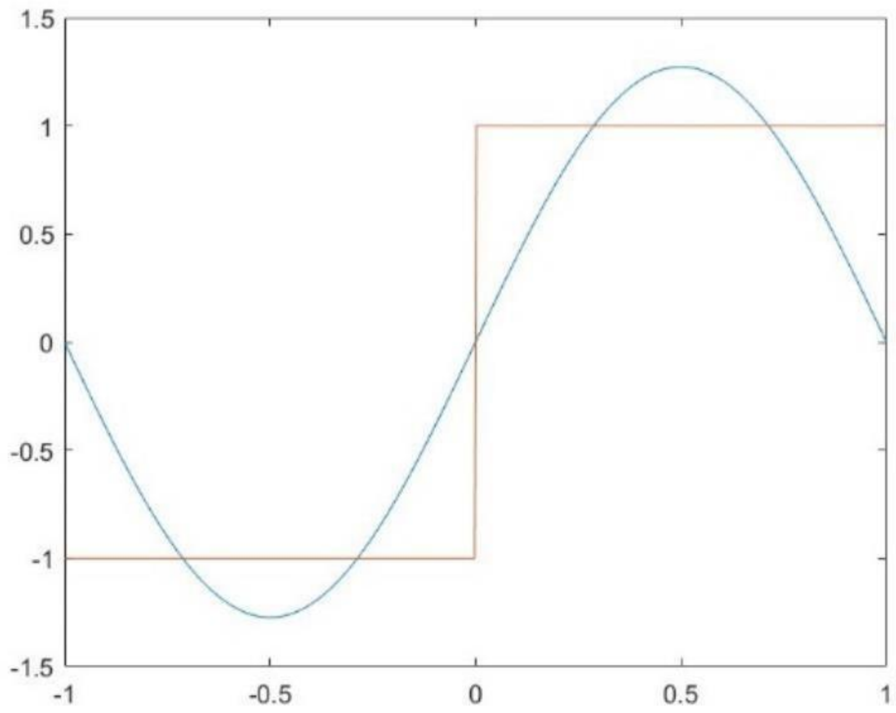
Algorithm:

```
START
1. SET N = 500 points
2. CREATE x from -1 to 1
3. DEFINE square wave: f = sign(x)
4. INITIALIZE sum = zeros
   - ADD term: (4/π)*sin(jπx)/j
6. END FOR
7. PLOT Fourier approximation vs actual square wave
8. VARY M to see convergence
END
```

5. FOR j = 1,3,5,...,M

MATLAB CODE:

```
clc;

clear all;

N = 500; %Number of oints

plotted x = linspace(-1,1,N); f =

sign(x); sum = 0.* x;

M = 1;

% number of coefficients.
Varing M you can draw
different waves

for j = 1:2:M sum = sum +

4/pi*sin(j*pi*x)/j; end plot(x,

sum) hold on plot(x,f) hold on
```



9. Convolute two simple signals of unit step and exponential decay:

- x(t) = u(t) → unit step function
- h(t) = e^{-t} → exponential decay starting at t=0t = 0t=0

We’ll compute: y(t)=x(t)∗h(t)

Algorithm:

```
START
1. DEFINE time axis: t = 0 to 10
2. DEFINE signals:
        - x(t) = unit step function
        - h(t) = e^(-t)
3. PERFORM convolution: y = conv(x,h)*dt
4. CREATE output time vector
5. PLOT all three signals:
        - Input x(t)
        - Impulse response h(t)
        - Output y(t)
```

MATLAB CODE:

% Time axis t = 0:0.01:10; % simulate continuous time

% Define signals x = ones(size(t)); % x(t) = u(t) h = exp(-t); % h(t) = e^(-t) * u(t)

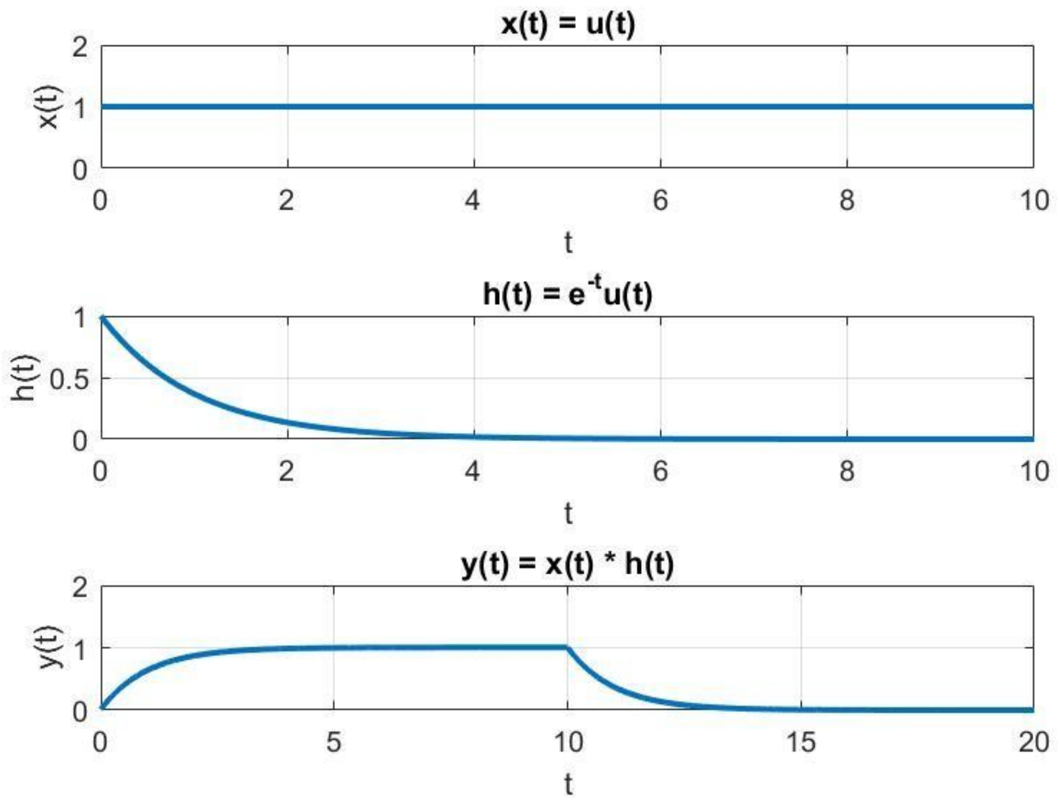
% Perform convolution

% multiply by dt to approximate continuous convolution y = conv(x, h) * 0.01;

% Create new time vector for output t_conv = 0:0.01:(2*max(t)); % Plot the signals and their convolution subplot(3,1,1);

plot(t, x, 'LineWidth', 2); title('x(t) = u(t)'); xlabel('t'); ylabel('x(t)'); grid on; subplot(3,1,2); plot(t, h, 'LineWidth', 2); title('h(t) = e^{-t}u(t)'); xlabel('t'); ylabel('h(t)');

```
grid on; subplot(3,1,3); plot(t_conv, y,
'LineWidth', 2); title('y(t) = x(t) * h(t)');
xlabel('t'); ylabel('y(t)'); grid on;
```



Laplace transform

- **Laplace transform** is named in honour of the great French mathematician, Pierre Simon De Laplace (1749-1827). Like all transforms, the Laplace transform changes one signal into another according to some fixed set of rules or equations. The best way to convert differential equations into algebraic equations is the use of Laplace transformation.
- Laplace transformation plays a major role in control system engineering. To analyze the control system, Laplace transforms of different functions have to be carried out. Both the properties of the Laplace transform and the inverse Laplace transformation are used in analyzing the dynamic control system.
- The **Laplace transform** is a powerful mathematical tool used in engineering, physics, and control theory. It transforms a given function of time *f(t)* into a function of a complex variable *s*. It is particularly useful for solving linear differential equations.

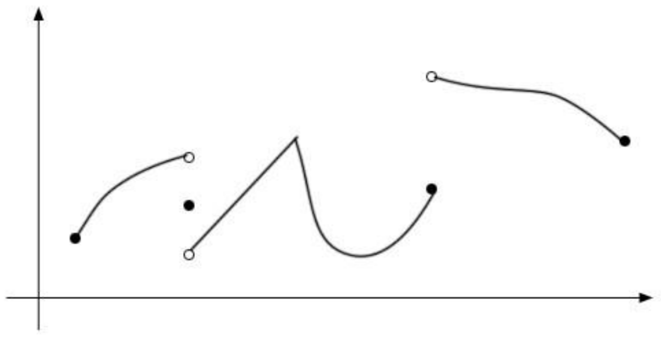
- The **Laplace transform** *L{f(t)}* of a function *f(t)* is defined as: $\boxed{L\{f(t)\} = F(s) = \int_0^\infty f(t)e^{-st}dt}$

  where: *f(t)* is the time-domain function (i.e., a function of *t*),
  s is a complex number, s=σ+jω, where σ and ω are real numbers, *F(s)*
  is the transformed function in the *s*-domain (the Laplace domain).

- The **inverse Laplace transform** is used to recover the original time-domain function from its Laplace transform.

  The formula of the inverse Laplace transform is typically: $\boxed{L}^{-1}\boxed{\{f(s)\} = F(t)}$
- There are various methods to compute the inverse Laplace transform, such as:
- Partial Fraction Decomposition
- Residue Theorem
- Inverse Transform Tables

- **Applications:** Laplace transforms are commonly used in:
  - Solving **differential equations**, especially for systems of differential equations.
  - **Control theory**, where the Laplace transform is used to analyze and design systems. o **Signal processing**, particularly in analyzing systems in the frequency domain.
- Before we start with the definition of the Laplace transform we need to get another definition.
- A function is called **piecewise continuous** on an interval if the interval can be broken into a finite number of subintervals on which the function is continuous on each open subinterval (*i.e.* the subinterval without its endpoints) and has a finite limit at the endpoints of each subinterval. Below is a sketch of a piecewise continuous function.



https://www.codewithc.com/c-program-for-solution-of-laplace-equation/