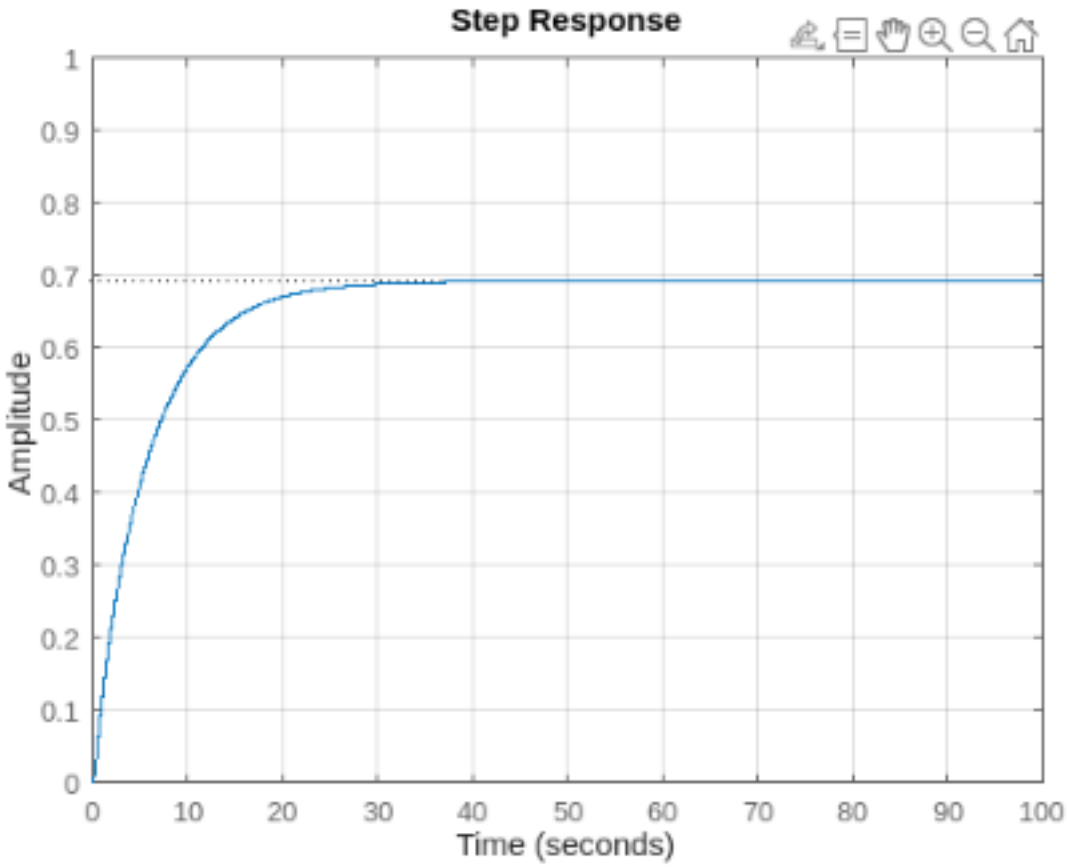
Transfer function using Block diagram reduction technique

Code:

```
n1=1;d1=[1 7]; %Initialising num and den of block G1
n2=1;d2=[1 3 7]; %Initialising num and den of block G2
n3=1;d3=[1 8]; %Initialising num and den of block G3
n4=1;d4=[1 0]; %Initialising num and den of block G4
n5=7;d5=[1 3]; %Initialising num and den of block G5
n6=1;d6=[1 7 5]; %Initialising num and den of block G6
n7=5;d7=[1 5]; %Initialising num and den of block G7
n8=1;d8=[1 9]; %Initialising num and den of block G8
nblocks=8; %Assigning number of blocks to nblocks
blkbuild %Creating state-space matrices
q=[1 -2 -5  0  0;
   2  1  8  0  0;
   3  1  8  0  0;
   4  1  8  0  0;
   5  3  4 -6  0;
   6  7  0  0  0;
   7 -6  3  4  0;
   8  7  0  0  0];
input=1; %Input block
output=7; %Output block
[aa,bb,cc,dd]=connect(a,b,c,d,q,input,output) %Connecting the blocks
[num,den]=ss2tf(aa,bb,cc,dd)
printsys(num,den) %Printing overall transfer function
k= tf(num,den);
figure;
step(k);
title('Step Response'); %Plotting step response
xlim([0 100]);
ylim([0 1]);
```

Output:

```
num/den =

 10 s^7 + 260 s^6 + 2680 s^5 + 14500 s^4 + 46140 s^3 + 89240 s^2 + 95370 s + 37800
-------------------------------------------------------------------------------
  s^10 + 42 s^9 + 751 s^8 + 7502 s^7 + 46368 s^6 + 186272 s^5 + 496688 s^4

 + 866280 s^3 + 908756 s^2 + 459660 s + 58800
```

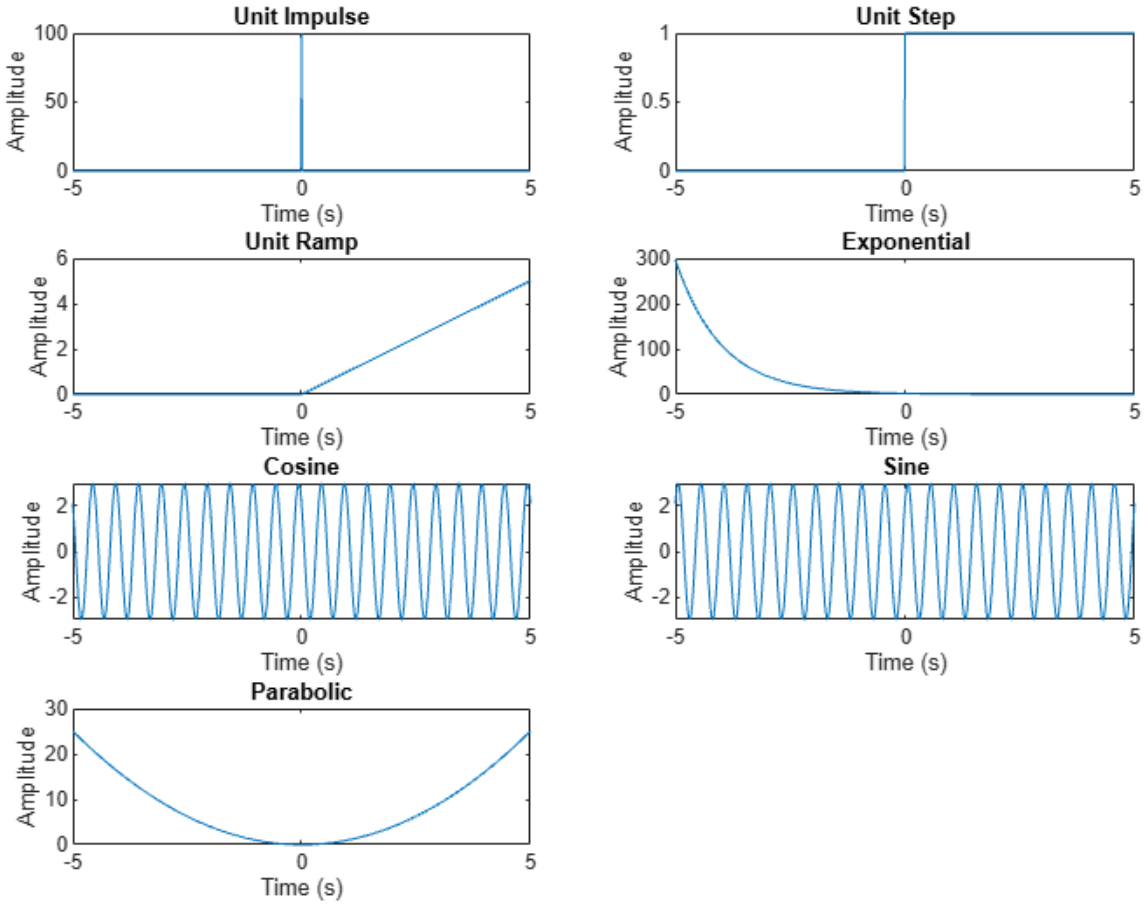

Step Response
Amplitude
Time (seconds)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
10
20
30
40
50
60
70
80
90
100

Generation of standard signals

Code:

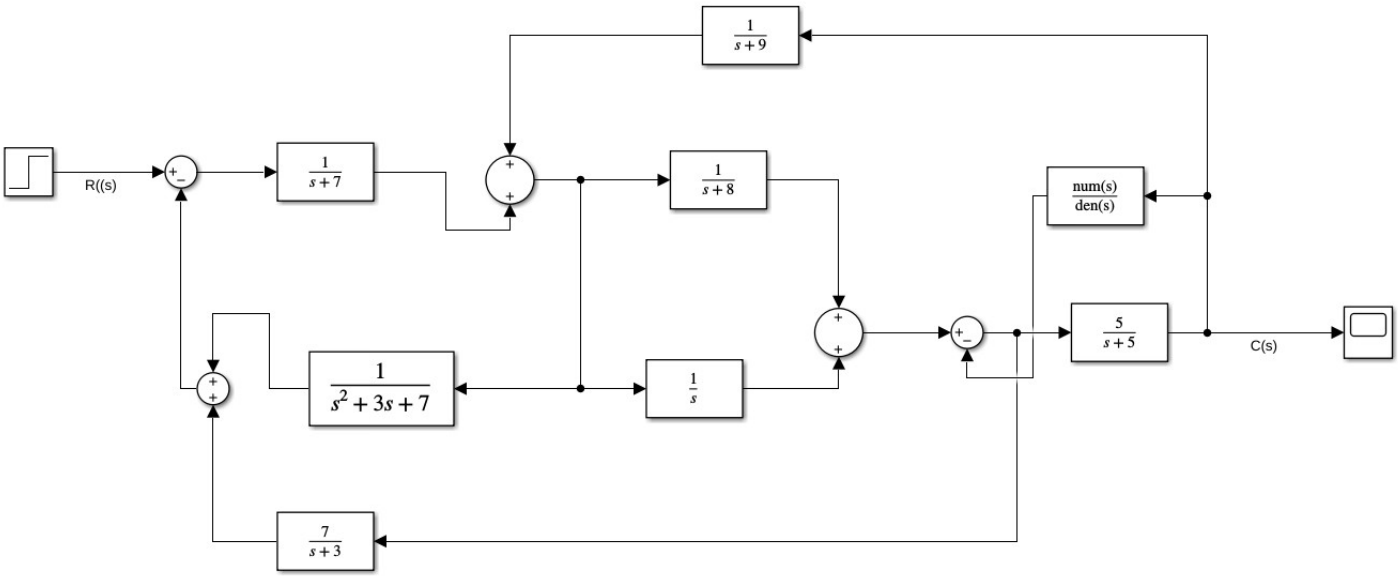
```
% Define time vector

t = -5:0.01:5; % Create a time vector ranging from -5 to 5 with a step
size of 0.01

% Generate signals
impulse = zeros(size(t)); % Initialize the impulse signal with zeros
impulse(find(t==0)) = 1/0.01; % Set the value at t=0 to approximate an
impulse
step = t >= 0; % Create a unit step signal (1 for t >= 0, 0 for t < 0)
ramp = t .* step; % Create a unit ramp signal (t for t >= 0, 0 for t < 0)
A = 2; % Set the amplitude of the exponential function
s = -1; % Set the exponent value (decay rate)
exponential = A * exp(s*t); % Create an exponential signal A*e^(s*t)
amplitude = 3; % Set the amplitude of the cosine wave
frequency = 2*pi*2; % Set the angular frequency
phase = pi/4; % Set the phase shift of the cosine wave
cosine = amplitude * cos(frequency*t + phase); % Create a cosine wave
sine = amplitude * sin(frequency*t + phase); % Create sine wave as above
a = 1; % Coefficient for the quadratic term in Parabolic signal
b = 0; % Coefficient for the linear term
c = 0; % Constant term (no constant term)
parabolic = a*t.^2 + b*t + c; % Create a parabolic signal

% Plotting
figure;
xlabel('Time (s)');
ylabel('Amplitude');

subplot(4,2,1);plot(t, impulse);title('Unit Impulse');
subplot(4,2,2);plot(t, step);title('Unit Step');
subplot(4,2,3);plot(t, ramp);title('Unit Ramp');
subplot(4,2,4);plot(t, exponential);title('Exponential');
subplot(4,2,5);plot(t, cosine);title('Cosine');
subplot(4,2,6);plot(t, sine);title('Sine');
subplot(4,2,7);plot(t, parabolic);title('Parabolic');
```

Output:


Unit Impulse
Amplitude
100
50
0
-5
0
5
Time (s)
Unit Step
Amplitude
1
0.5
0
-5
0
5
Time (s)
Unit Ramp
Amplitude
6
4
2
0
-5
0
5
Time (s)
Exponential
Amplitude
300
200
100
0
-5
0
5
Time (s)
Cosine
Amplitude
2
0
-2
-5
0
5
Time (s)
Sine
Amplitude
2
0
-2
-5
0
5
Time (s)
Parabolic
Amplitude
30
20
10
0
-5
0
5
Time (s)

Transfer function using block reduction technique using Simulink

Simulink System:



Output: