**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr No.** | **Aim** | **Date** | **Page No.** |
| 1. | Write a MATLAB program to sample the given continuous time signal at different sampling frequencies and reconstruct the input signal for Perfect Sampling, Under Sampling and Over Sampling. | 30/07/2021 | 2 |
| 2. | Write a MATLAB program to perform time shifting, time scaling and time reversal, summation of the continuous as well as discrete time signals. | 06/07/2021 | 5 |
| 3. | Write a MATLAB program to perform linear and circular convolution of given signals with and without inbuilt function | 13/08/2021 | 12 |
| 4. | Write a MATLAB program to compute DFT and IDFT for the given signal x[n] = [ 1 1 1 1 1 1 0 0 1 ]. Also plot the magnitude and phase response for DFT. | 27/08/2021 | 16 |
| 5. | Write a MATLAB program to compute N point DIT-FFT and verify the results using inbuilt FFT command. | 03/09/2021 | 20 |

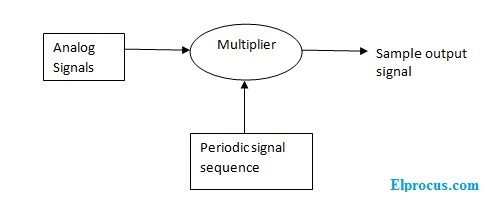
**Practical 1**

**Date: 30/07/2021**

**AIM:** Write a MATLAB program to sample the given continuous time signal at different sampling frequencies and reconstruct the input signal for Perfect Sampling, Under Sampling and Over Sampling.

**THEORY:**

Sampling is needed to convert analog signal into discrete time signal. The sampling theorem essentially says that a signal has to be sampled at least with twice the frequency of the original signal. The minimum sampling rate allowed by the sampling theorem is called the Nyquist rate.



**Over Sampling:** If we use sampling frequency greater than twice the maximum frequency component of signal, then it is called oversampling. If the Nyquist criterion is not met, aliasing will occur.

**Under Sampling:** If we use the sampling frequency less than twice the maximum frequency component in the signal, then it is called undersampling. The aliasing effect due to the undersampling technique can be used for our advantage. When a signal is sampled at a rate less than twice its maximum frequency, the aliased signal appears at Fs – Fin, where Fs is the sampling frequency and Fin in the input signal frequency. As we know in advance that the signal is aliased, we can recover the actual frequency by using the Fs – Fin relationship.

**Perfect Sampling:** If we use the sampling frequency equal to twice the maximum frequency component in the signal, then it is called undersampling.

**ALGORITHM:**

1. Get frequency and amplitude input from user.
2. Construct Input Signal and display discrete and continuous plot.

Signal = Am \* Cos(2\*Π\*F\*t)

1. Construct Signal for F<2\*Fs for under sampling and display the plots.
2. Construct Signal for F=2\*Fs for perfect sampling and display the plots.
3. Construct Signal for F>2\*Fs for oversampling and display the plots.

**CODE:**

close all;

clear all;

t=-10:0.01:10;

fm=input('Enter Input Frequency: '); %Input freq

Am=input('Enter Input Amplitude: '); %input Amplitude

x=Am\*cos(2\*pi\*fm\*t); %Input Signal

subplot(2,2,1)

plot(t,x) %Input signal plot

xlabel('Time (sec)')

ylabel('x(t)')

title('continuous time signal')

%Under Sampling

fs1=1.6\*fm; %Under Sampling freq

n1=-5:1:5; %Discrete Time range

xn1=Am\*cos(2\*pi\*n1\*fm/fs1); %Under Sampling Signal

subplot(2,2,2)

stem(n1,xn1) %Discrete plot

hold on

plot(n1,xn1) %Continuous plot

xlabel('Time (sec)')

ylabel('x(t)')

title('Under Sampling')

hold off;

%Perfect Sampling

fs2=2\*fm; %Perfect Sampling freq

xn2=Am\*cos(2\*pi\*n1\*fm/fs2); %Perfect Sampling Signal

subplot(2,2,3)

stem(n1,xn2) %Discrete plot

hold on

plot(n1,xn2) %Continuous plot

xlabel('Time (sec)')

ylabel('x(t)')

title('Perfect Sampling')

hold off;

%Over Sampling

fs3=7\*fm; %Over Sampling freq

xn3=Am\*cos(2\*pi\*n1\*fm/fs3); %Over Sampling Signal

subplot(2,2,4)

stem(n1,xn3) %Discrete plot

hold on

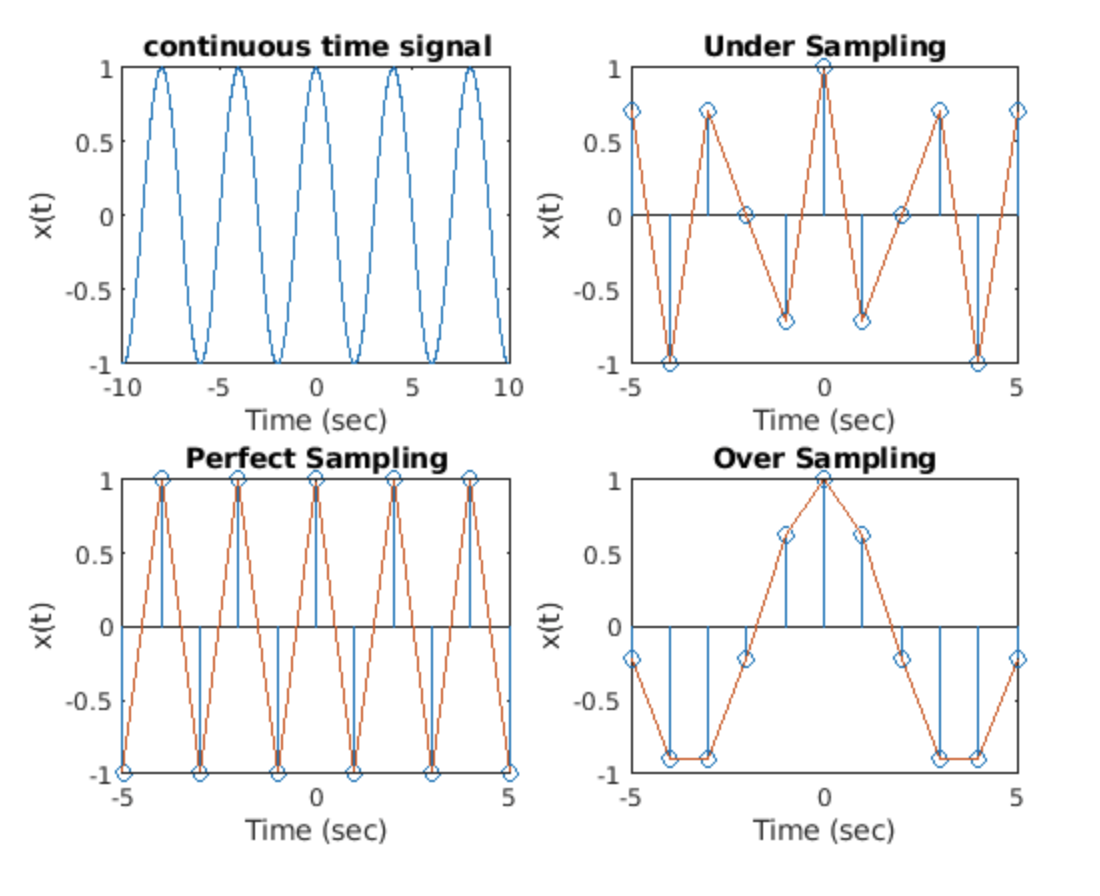
plot(n1,xn3) %Continuous plot

xlabel('Time (sec)')

ylabel('x(t)')

title('Over Sampling')

**RESULT:**

****

**CONCLUSION:** Thus, we can conclude that input signal can be reconstructed completely if the sampling frequency is greater than the input frequency of the signal, else in the case of under sampling, aliasing can be found and hence the signal cannot be reconstructed properly.

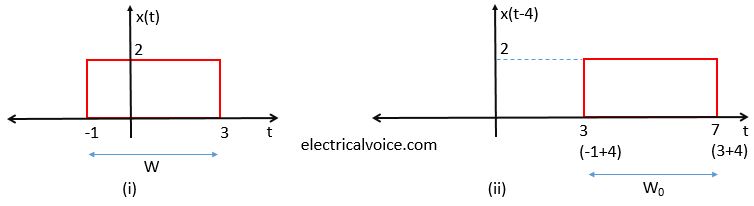
**Practical 2**

**Date: 06/08/2021**

**AIM:** Write a MATLAB program to perform time shifting, time scaling and time reversal, summation of the continuous as well as discrete time signals.

**THEORY:**

**Time Shifting:** A signal x(t) may be shifted in time by replacing the independent variable **t** by either **t−t0** or ***t+t0***. Here t0is called as the shifting factor. Shifting in time may results in time delay or time advancement. Graphically, this kind of signal operation results in a positive or negative “shift” of the signal along its time axis.

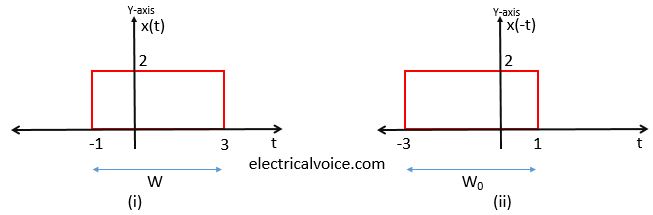


**Time Scaling:** A signal x(t) may be scaled in time by replacing the independent variable **t** by ***at***. Here ‘**a’**is called as the scaling factor. Time scaling may results in signal compression or signal expansion. A positive factor of a either expands (0 < a < 1) or compresses (a > 1) the signal in time.

![Chart, box and whisker chart

Description automatically generated](data:image/jpeg;base64,/9j/4AAQSkZJRgABAQEAYABgAAD/4RDgRXhpZgAATU0AKgAAAAgABAE7AAIAAAAHAAAISodpAAQAAAABAAAIUpydAAEAAAAOAAAQyuocAAcAAAgMAAAAPgAAAAAc6gAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEhhcm9vbgAAAAWQAwACAAAAFAAAEKCQBAACAAAAFAAAELSSkQACAAAAAzI2AACSkgACAAAAAzI2AADqHAAHAAAIDAAACJQAAAAAHOoAAAAIAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAyMDE5OjA5OjEzIDIzOjI4OjI0ADIwMTk6MDk6MTMgMjM6Mjg6MjQAAABIAGEAcgBvAG8AbgAAAP/hCxlodHRwOi8vbnMuYWRvYmUuY29tL3hhcC8xLjAvADw/eHBhY2tldCBiZWdpbj0n77u/JyBpZD0nVzVNME1wQ2VoaUh6cmVTek5UY3prYzlkJz8+DQo8eDp4bXBtZXRhIHhtbG5zOng9ImFkb2JlOm5zOm1ldGEvIj48cmRmOlJERiB4bWxuczpyZGY9Imh0dHA6Ly93d3cudzMub3JnLzE5OTkvMDIvMjItcmRmLXN5bnRheC1ucyMiPjxyZGY6RGVzY3JpcHRpb24gcmRmOmFib3V0PSJ1dWlkOmZhZjViZGQ1LWJhM2QtMTFkYS1hZDMxLWQzM2Q3NTE4MmYxYiIgeG1sbnM6ZGM9Imh0dHA6Ly9wdXJsLm9yZy9kYy9lbGVtZW50cy8xLjEvIi8+PHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9InV1aWQ6ZmFmNWJkZDUtYmEzZC0xMWRhLWFkMzEtZDMzZDc1MTgyZjFiIiB4bWxuczp4bXA9Imh0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC8iPjx4bXA6Q3JlYXRlRGF0ZT4yMDE5LTA5LTEzVDIzOjI4OjI0LjI2MzwveG1wOkNyZWF0ZURhdGU+PC9yZGY6RGVzY3JpcHRpb24+PHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9InV1aWQ6ZmFmNWJkZDUtYmEzZC0xMWRhLWFkMzEtZDMzZDc1MTgyZjFiIiB4bWxuczpkYz0iaHR0cDovL3B1cmwub3JnL2RjL2VsZW1lbnRzLzEuMS8iPjxkYzpjcmVhdG9yPjxyZGY6U2VxIHhtbG5zOnJkZj0iaHR0cDovL3d3dy53My5vcmcvMTk5OS8wMi8yMi1yZGYtc3ludGF4LW5zIyI+PHJkZjpsaT5IYXJvb248L3JkZjpsaT48L3JkZjpTZXE+DQoJCQk8L2RjOmNyZWF0b3I+PC9yZGY6RGVzY3JpcHRpb24+PC9yZGY6UkRGPjwveDp4bXBtZXRhPg0KICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICA8P3hwYWNrZXQgZW5kPSd3Jz8+/9sAQwAHBQUGBQQHBgUGCAcHCAoRCwoJCQoVDxAMERgVGhkYFRgXGx4nIRsdJR0XGCIuIiUoKSssKxogLzMvKjInKisq/9sAQwEHCAgKCQoUCwsUKhwYHCoqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioq/8AAEQgA7ANqAwEiAAIRAQMRAf/EAB8AAAEFAQEBAQEBAAAAAAAAAAABAgMEBQYHCAkKC//EALUQAAIBAwMCBAMFBQQEAAABfQECAwAEEQUSITFBBhNRYQcicRQygZGhCCNCscEVUtHwJDNicoIJChYXGBkaJSYnKCkqNDU2Nzg5OkNERUZHSElKU1RVVldYWVpjZGVmZ2hpanN0dXZ3eHl6g4SFhoeIiYqSk5SVlpeYmZqio6Slpqeoqaqys7S1tre4ubrCw8TFxsfIycrS09TV1tfY2drh4uPk5ebn6Onq8fLz9PX29/j5+v/EAB8BAAMBAQEBAQEBAQEAAAAAAAABAgMEBQYHCAkKC//EALURAAIBAgQEAwQHBQQEAAECdwABAgMRBAUhMQYSQVEHYXETIjKBCBRCkaGxwQkjM1LwFWJy0QoWJDThJfEXGBkaJicoKSo1Njc4OTpDREVGR0hJSlNUVVZXWFlaY2RlZmdoaWpzdHV2d3h5eoKDhIWGh4iJipKTlJWWl5iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uLj5OXm5+jp6vLz9PX29/j5+v/aAAwDAQACEQMRAD8A+kaKKKACiiigAooooAKKKKACiiigAooooAKKKKACo554rW2kuLiRYoYkLyOxwFUDJJ9sVJRQB5r8KPivD8RL3XLV1WGazuWktUxgvak4Qkf3gev+8K9Krzr4S/8AH948/wCxqu/5JXLeJDbJ471GeNSPAn9qWw8SFWAiN8FYbsYx5QYW4m6ZOM5w9AHpzeLrWPWtdtZo/LstBtEuL283FtjMrOUCAEnEYDH/AHwAKdqvimDTYNEvFi8/T9Vuorf7Tv2eT5qkxOVI5DNtXsRvBrhv7Kj1bQfixpl5aR3l21/PJHDJFvPNlEYCFOeQRkEDORxVTUT4ek/Z+tF8IDTR9ouNNXGnqgBvDNb53hP+WnAzn5sDmgD2KiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAry/wCMnj7W/Bsvh6LQNOu7k3F+huHjgZkkXOBbhgMb35wBz8vTmvUK86+Mf/Hj4Q/7Gqw/m9AHoFtOLq0huFSSNZUVwkqFHXIzhlPIPqD0qtrer22gaDfavftttrG3eeUjrtVSSB78YA9azPHp0X/hBNVXxMrPpzQ7XSP/AFjPkbBH/wBNN+3bjndtrx+8GpJ4B8WReOSG8Yw2lg90zEMH05ZEOY+ny5E3mY48wk9CtAHsEnia/g03T5W8Mand3lzai4ntrAxMtscDKmSV41Y5OMDk4JwBT9O8a6Hqel6Lf290wi1xzHYh4mDO4VmKEY+UgI2c8cdelUfGWr6REItP1Hxq/hWSaJpElWWCHz1PynbJMjDI6/IQwyD3FeZLYaPN4L+Hep+KND0uz0611OW2lmurUrCbUpOInfzssqOdj/OeWYHuKAPYbTxAs/i/UfD89v5E9pbw3UL78/aIX3KWAxwVdGUjn+E98DYrhQIT8bdKGm+ULeLwzNu8oDZ5bTxeUBjjHytj2BruqACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAK8v+Mnj7W/Bsvh6LQNOu7k3F+huHjgZkkXOBbhgMb35wBz8vTmvUK86+Mf/AB4+EP8AsarD+b0AegW04urSG4VJI1lRXCSoUdcjOGU8g+oPSpaz9e1dNA0G71WW0uruO0TzJIrRA0hQH5iASM4GSRnOAcAnAPJ+JviPap4V8QXPhnfeS2NhE8N8gU2/nXAAhQHOWb50cgDG1l5ycUAd5XOaLrup3vjTXNH1G2tYIbC2tJ4PJdnZhK0wO8kAf8shwBxk8mmR+KfDHhe2t9F1nxbpsd9ZQRxTDUNTjE7HYPmfe27LDDZPXOa57TdY8E6d4uvfEH/C0bS5mvUWOa2m1Ox8kom8xqAqBgF8xsfNk55JoA9HorgPE3xPtbWLTG8GtZ+JDcanb2l19gu4JhCsjYCn96u13w2wn5cqdxFXW1a9tPiZoyTrdQWuvaZKrWVxIrfZp4Crg4VmUEpK4O0kHYvJxQB2VFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAedfCYEX3jzIx/wAVVd/ySvRa86+Ff/Ic8ff9jJP/AOgrXotAGdBodrbeI7zWoGlS4vYI4Z4ww8t/LLbXxjO7DEZz0x6UarodtrF1pst48pXTroXccSsAjyBWVS/GSF3bgMjkA84rRooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACvO/jEP+Jf4SbsvimwLH0GWr0SvOvjN/wAgPw5/2Mlj/wChNQB6LVe/sbfU9NubC+jEttdQvDNGejowKsPxBNWKKAKumWI0zSbSwWea4W1hSETTkGSQKANzEAAk45wBVqiigDNtNDtbTxBqGsq0sl3fpFE5kIIjjjB2onHAyzMc55Y+wGlRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABXnXxj40/wkT0XxVYEn0GWr0WvOvjN/yA/Dn/YyWP8A6E1AHaa/Hq02hXUPh2S3h1GRNkM1yTsiJ4L4AOSoyQOhIAJA5rzXUPhpfeG/h34k8O+HTHdaM1tHe6fBK7G4S7iZXdMYwyuY1YHIwzMMYxj1uigDm7jx/wCHbPSbTUbq9cQXVol6Ght5Z/LgYZEj+WrbE/2mwODzxWnqmu6fo+lLqN7LIbZ2REa3gedpGcgIFSNWZskjoDWJ40k1e5CaVaeH7/UtLuYm+2yWU9ujOvTyf3sqFQwzuYZwOBycqzX49Q1LSjZXPhK4uLWJ7WRI7HVEt5QRlm8tgybTGyqB8y7s5BGKAF8SWc/jPw5YXPh9THc2WpW99DFqlvPaCQwyAlWDpvQEZw2w9uDVW5S51j4seHlnhijl0PTJ7y9WGQyJFLcbYo0DlVJyEmOSBkL0Fa3gmy1aw0GSPW2uNzXUr20N3c/aJoICfkjkkyd7Dnnc3UDccZroqACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKAPOvhYCNc8fZGP+Kkm/9BWvRa84+GytJ4i+IcaSvEW11wrpglCY15GQRn6ginWlvNo/xd0vR/D+p6peWq6bNNrUV9fy3ax8qIGzIzbHZt/C4yATjAoA9FooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACvO/jKpOheHSBwviSxJ/77NeiV518av+RV0f/sPWX/odAHotFYPi2w8MTaWdS8aW9nPp+nK0p+3r5kMecDcYzlWbsDgsMkD7xzlfCnTr/Tfh9aR6lDNaiWWWa1s7hiXtLZ3LRRNnoVQgbe3TtQB2dFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABXnXxm/5Afhz/sZLH/0Jq9Frzr41f8AIq6P/wBh6y/9DoA9ForB8W2HhibSzqXjS3s59P05WlP29fMhjzgbjGcqzdgcFhkgfeOcr4U6df6b8PrSPUoZrUSyyzWtncMS9pbO5aKJs9CqEDb26dqAOzooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiuM+Ln/ACSzV/8Atj/6PSj/AIVH4I/6An/k3P8A/F1jKc+flivxtvfyfY9KlhsP9XVevOSvJpJRUvhUX1lH+Y7OiuM/4VH4I/6An/k3P/8AF0f8Kj8Ef9AT/wAm5/8A4ujmrfyr73/kHs8u/wCfs/8AwXH/AOWHZ0Vxn/Co/BH/AEBP/Juf/wCLo/4VH4I/6An/AJNz/wDxdHNW/lX3v/IPZ5d/z9n/AOC4/wDywyPh5HcPr3xGjtJVt7htakEUrx7xGxiG1iuRuAODjIz61r+D/CGveFiUn1zS7+OeZp76c6TJHdXch/iaU3DAHoPuYAGAAK4fwJ8O/C2peLPG1te6X5kNjqixW6/aJRsUxg4yG559c13H/Co/BH/QE/8AJuf/AOLpt1eiX3/8AmNPANe9Unf/AAJ/+5EdnRXGf8Kj8Ef9AT/ybn/+Lo/4VH4I/wCgJ/5Nz/8AxdLmrfyr73/kV7PLv+fs/wDwXH/5YdnRXGf8Kj8Ef9AT/wAm5/8A4uj/AIVH4I/6An/k3P8A/F0c1b+Vfe/8g9nl3/P2f/guP/yw7OiuM/4VH4I/6An/AJNz/wDxdH/Co/BH/QE/8m5//i6Oat/Kvvf+Qezy7/n7P/wXH/5YdnRXGf8ACo/BH/QE/wDJuf8A+Lo/4VH4I/6An/k3P/8AF0c1b+Vfe/8AIPZ5d/z9n/4Lj/8ALDs6K4z/AIVH4I/6An/k3P8A/F1maD4d0rwz8Zns9DtfstvJoBlZPMZ8sbgAnLEnoopOpUi1zJa+f/ANI4XCVYTdGpJuKb1gktPNTf5Ho1FFFbnkhRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAVDd3lrp9q9zf3MNrbx43yzSBEXJwMk8DkgVNXGfFz/AJJZq/8A2x/9HpUVJckHLsjqwdBYnE06DdlKSX3uxtf8Jl4Y/wChj0n/AMDov/iq4D4v+JNCvvDGlJZa1p1w8et2cjrFdxsVUPy2Aeg9a7//AIQ3wx/0Lmk/+AMX/wATXAfGPw1oVh4Ns5bHRdOtpDq1oheG0RCVMgyMgdD6VK9rfWxtL6hyvl57/I3PFX/CJ+K7jTZZ/HMNj/Zs32iKO0vrRo3k/hd0lVwxXqvHBOeuCNXSvEeg6datFeeObXVXZywmvLu0V1GANo8pUXHGeRnk89MXv+EN8Mf9C5pP/gDF/wDE0f8ACG+GP+hc0n/wBi/+JpfvvL8Sv+E7+/8A+Sh/wmXhj/oY9J/8Dov/AIqj/hMvDH/Qx6T/AOB0X/xVH/CG+GP+hc0n/wAAYv8A4mj/AIQ3wx/0Lmk/+AMX/wATR++8vxD/AITv7/8A5KaFhqdhqsDT6Xe297ErbDJbyrIobAOMqTzgjj3qzXE/DOGK2g8TwW8aRRR+IrtEjRQqooCAAAdAB2rtqqnJzgpMwxtCOHxEqUHdL/K4UUUVocgUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABTJporaCSe4kSKKNS7yOwVUUDJJJ6ADvT6xvGX/ACIuvf8AYNuP/RTVMnaLZrRpqpVjB9Wl94f8Jl4Y/wChj0n/AMDov/iqP+Ey8Mf9DHpP/gdF/wDFVkeFPCnh258GaLPcaBpcssmnwO8j2cbM7GNSSSRySe9a/wDwhvhj/oXNJ/8AAGL/AOJrKLqtJ6fiejVhl9OpKD59Hb7PQP8AhMvDH/Qx6T/4HRf/ABVH/CZeGP8AoY9J/wDA6L/4qj/hDfDH/QuaT/4Axf8AxNH/AAhvhj/oXNJ/8AYv/iaf77y/Ez/4Tv7/AP5KH/CZeGP+hj0n/wADov8A4qj/AITLwx/0Mek/+B0X/wAVR/whvhj/AKFzSf8AwBi/+Jo/4Q3wx/0Lmk/+AMX/AMTR++8vxD/hO/v/APkof8Jl4Y/6GPSf/A6L/wCKo/4TLwx/0Mek/wDgdF/8VR/whvhj/oXNJ/8AAGL/AOJo/wCEN8Mf9C5pP/gDF/8AE0fvvL8Q/wCE7+//AOSh/wAJl4Y/6GPSf/A6L/4quA+MHiXQr7wzpMdlrWnXDprdnIyxXaOVUPyxAPQetd//AMIb4Y/6FzSf/AGL/wCJrgPjH4a0Kx8G2UtlounW0h1a0UvDaIhKmTBGQOh9Ka9rfWxM/qHK+Xnv8jc8Vf8ACJ+K7jTZZ/HMNj/Zs32iKO0vrRo3k/hd0lVwxXqvHBOeuCNXSvEeg6datFeeObXVXZywmvLu0V1GANo8pUXHGeRnk89MXv8AhDfDH/QuaT/4Axf/ABNH/CG+GP8AoXNJ/wDAGL/4ml++8vxK/wCE7+//AOSh/wAJl4Y/6GPSf/A6L/4qj/hMvDH/AEMek/8AgdF/8VR/whvhj/oXNJ/8AYv/AImj/hDfDH/QuaT/AOAMX/xNH77y/EP+E7+//wCSh/wmXhj/AKGPSf8AwOi/+Ko/4TLwx/0Mek/+B0X/AMVR/wAIb4Y/6FzSf/AGL/4mj/hDfDH/AELmk/8AgDF/8TR++8vxD/hO/v8A/kof8Jl4Y/6GPSf/AAOi/wDiqP8AhMvDH/Qx6T/4HRf/ABVH/CG+GP8AoXNJ/wDAGL/4mj/hDfDH/QuaT/4Axf8AxNH77y/EP+E7+/8A+Sh/wmXhj/oY9J/8Dov/AIqj/hMvDH/Qx6T/AOB0X/xVH/CG+GP+hc0n/wAAYv8A4mj/AIQ3wx/0Lmk/+AMX/wATR++8vxD/AITv7/8A5KH/AAmXhj/oY9J/8Dov/iqP+Ey8Mf8AQx6T/wCB0X/xVH/CG+GP+hc0n/wBi/8AiaP+EN8Mf9C5pP8A4Axf/E0fvvL8Q/4Tv7//AJKH/CZeGP8AoY9J/wDA6L/4qj/hMvDH/Qx6T/4HRf8AxVH/AAhvhj/oXNJ/8AYv/iaP+EN8Mf8AQuaT/wCAMX/xNH77y/EP+E7+/wD+Sh/wmXhj/oY9J/8AA6L/AOKq5p+s6Zq3mf2VqNpfeVjzPs06ybM5xnaTjOD+VU/+EN8Mf9C5pP8A4Axf/E1z/g6ztdP+JHjW2sLaG1t4/sOyKGMIi5hYnAHA5JNLmqRlFStr/k3+hp7DCVaNWdFyvBJ62t8UY9P8R3NFFFbnkhRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAHGfFz/klmr/APbH/wBHpXZ1xnxc/wCSWav/ANsf/R6V2dYx/jS9F+bPSqf8i6l/jqf+k0wooorY80KKKKAPOvhuCPHHxCyMf8ThP/RQr0WvPPh45PxC+IadhqcJ/OEf4V6HQAUUUUAFFFFABRRRQAUUUUAFcZ/zXX/uW/8A25rs64z/AJrr/wBy3/7c1jV+z6o9LL/+Xv8Agl+h2dFFFbHmhRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAVxnxc/5JZq/wD2x/8AR6V2dcZ8XP8Aklmr/wDbH/0elY4j+DP0f5HpZT/yMaH+OP8A6Ujs687+Nik+BrMgcLrFmT/39FeiV538b2KfDyNl6rqdmR/3+WtjzT0SiiigAooooA4z4c/8zX/2Ml5/7JXZ1xnw5/5mv/sZLz/2SuzrGh/DR6Waf75P5fkgooorY80KKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigArG8Zf8iLr3/YNuP/RTVs1jeMv+RF17/sG3H/opqip8DOnCf7xT/wAS/MPBv/Ii6D/2Dbf/ANFLWzWN4N/5EXQf+wbb/wDopa2aKfwL0DF/7xU9X+YUUUVZzBRRRQAUUUUAFedfG3/kR7L/ALDFn/6MFei1518cOPh7E4+8mqWbL9fNFAHotFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABXGeGf+Sp+OP+3D/wBEGuzrjPDP/JU/HH/bh/6INY1Pih6/oz0sH/AxP+Bf+nKZ2dFFFbHmhRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAHGfFz/klmr/9sf8A0eldnXGfFz/klmr/APbH/wBHpXZ1jH+NL0X5s9Kp/wAi6l/jqf8ApNMKKKK2PNCiiigDzr4eAj4jfETIx/xMoP8A0TXoteeeAnz8TviKmOl9anP1g/8ArV6HQAUUUUAFFFFABRRRQAUUUUAFcZ/zXX/uW/8A25rs64z/AJrr/wBy3/7c1jV+z6o9LL/+Xv8Agl+h2dFFFbHmhRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAVxnxc/5JZq/wD2x/8AR6V2dcZ8XP8Aklmr/wDbH/0elY4j+DP0f5HpZT/yMaH+OP8A6Ujs687+OCk/DgEDhdStCf8Av8teiV558cX8v4XzuOSt9aH/AMjpWx5p6HRRRQAUUUUAcZ8Of+Zr/wCxkvP/AGSuzrjPhz/zNf8A2Ml5/wCyV2dY0P4aPSzT/fJ/L8kFFFFbHmhRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFY3jL/kRde/7Btx/wCimrZrG8Zf8iLr3/YNuP8A0U1RU+BnThP94p/4l+YeDf8AkRdB/wCwbb/+ilrZrG8G/wDIi6D/ANg23/8ARS1s0U/gXoGL/wB4qer/ADCiiirOYKKKKACiiigArzr44/8AJOV/7CVp/wCjlr0WvOvjn8vwxkk7x39owHr++X/GgD0WiiigAooooAKKKKACiiigAooooAKKKKACiiigArjPDP8AyVPxx/24f+iDXZ1xnhn/AJKn44/7cP8A0QaxqfFD1/RnpYP+Bif8C/8ATlM7OiiitjzQooooAKKKbJKkMZeV1RB1ZjgCgB1FFFABRRRQAUUUUAFFFFAHGfFz/klmr/8AbH/0eldnXGfFz/klmr/9sf8A0eldnWMf40vRfmz0qn/Iupf46n/pNMKKKK2PNCiiigDzrwCCPip8R8jH+mWf/oivRa898DsD8W/iOvcXFgT+Nv8A/Wr0KgAooooAKKKKACiiigAooooAK4z/AJrr/wBy3/7c12dcZ/zXX/uW/wD25rGr9n1R6WX/APL3/BL9Ds6KKK2PNCiiigAooqnqmr6dolib3WL2CxtVZUM1xIEUEnAGT70AXKKbHIk0SyQuskbgMrqchgehBp1ABRRRQAUUVQ0/XNK1W8vLXTNRtruexYJcxwShzCxzgNjoeD+RoAv0UUUAFFFFABRRRQAUUUUAFFFFABXGfFz/AJJZq/8A2x/9HpXZ1xnxc/5JZq//AGx/9HpWOI/gz9H+R6WU/wDIxof44/8ApSOzrzv46KT8K7ogcLeWhP8A3/SvRK88+Oj+X8I9RcjO24tTj/t4jrY809DooooAKKKKAOM+HP8AzNf/AGMl5/7JXZ1xnw5/5mv/ALGS8/8AZK7OsaH8NHpZp/vk/l+SCiiitjzQooooAKKKKACiiigAooooAKKKwIPHHh6fxZc+GjqCw6vbkZtp1MZkyAQULAB+vbNAG/RRRQAUUVQ0/XNK1W8vLXTNRtruexYJcxwShzCxzgNjoeD+RoAv0UUUAFY3jL/kRde/7Btx/wCimrZrG8Zf8iLr3/YNuP8A0U1RU+BnThP94p/4l+YeDf8AkRdB/wCwbb/+ilrZrG8G/wDIi6D/ANg23/8ARS1s0U/gXoGL/wB4qer/ADCiiirOYKKKKACiiigArzr46/8AJK7n/r8tf/R6V6LXnXx2+X4S30naO6tWI9f36f40Aei0UUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFcZ4Z/5Kn44/7cP/AEQa7OuM8M/8lT8cf9uH/og1jU+KHr+jPSwf8DE/4F/6cpnZ0UVgQeOPD0/iy58NHUFh1e3IzbTqYzJkAgoWAD9e2a2PNN+iiigBsvmeS/kbTJtOzf8Adz2z7V8weOm+IJ8e23/CwxpK6X5v+hi98/8Asndn5c+T82f+unvnivqGmyxRzxmOaNZEPVXXIP4UAeef8Xp/6kP/AMnKPL+M78tP4HjPoiXZH616LRQB515Pxl/5+/BX/fu6o8n4y/8AP34K/wC/d1XotFAHnXk/GX/n78Ff9+7qj7P8ZW4N74LUeqxXJI/OvRaKAPOvsXxi/wCgt4S/8Bp6PsXxi/6C3hL/AMBp69FooA8o8QeFviv4k0O40nUdX8LC2uNu8xQTq3ysGGDz3UUn/CO/G7/oddD/APAIf/Gq9YrkvHnjiXwXp63FvoV5qrYDyNGfKggTzFTLykEZy/CgFjg8AZNS4pu5vGvUjDk6XvZpPV2vv6L7jll8N/Gkj5vHOjKfQWCn/wBp0v8AwjXxo/6HvR//AAXr/wDG69Voo5UL20uy+5f5HlX/AAjXxo/6HvR//Bev/wAbo/4Rr40f9D3o/wD4L1/+N16rRRyoPbS7L7l/kfPnhfQvibL8RvGUFh4t0+11GGS0/tG5axVluCYSYyqlOMLwema7H/hF/jF/0UDTP/BXH/8AEVoeDWH/AAuj4kLnkNphI/7djXoVPlRPtJeX3L/I8t/4Rf4xf9FA0z/wVx//ABFH/CL/ABi/6KBpn/grj/8AiK9Soo5UP2svL7l/keW/8Iv8Yv8AooGmf+CuP/4injwr8XSPm+ItgD6DSIj/AOy16fRRZC9pLy+5f5HmH/CKfFz/AKKNY/8Agni/+Jo/4RT4uf8ARRrH/wAE8X/xNen0UWF7R+X3I8w/4RT4uf8ARRrH/wAE8X/xNOXwn8Wifn+JFko9V0aI/wBK9NoosHO/L7keaf8ACI/Fb/oplr/4I4aqR/D34jpr/wDbTfEGzkv/ALL9k81tHj/1W7ftwCB15zjNerVS1a51C2sd2kWCX90zqqxS3AgjUE8sz4YgAeisc447g5U9xxqzjfldr6HD/wDCM/FL/ooNh/4JU/xo/wCEZ+KX/RQbD/wSp/jXR+GPE8+t6hrGmalp6WGpaPOkVzHDcefEweMSIyOVUnKsMgqMGuipmZ51/wAIn8TzyfiXbj2Ggw4H60f8Il8Tv+im2/8A4IIf8a9FooA86/4RL4nf9FNt/wDwQQ/41g+NNA8eaf4SvJtb8cLq9mQqNYw+F4Z2nLMAqhM+uOT0r2OuU8R+PbTw34r0jRbmwv7j+0Uldpraznm8sIuRtEcbeYSeoByowTwRQBwXwQ8CeOvDKm517VWsdKkyyaK6iQ89G64i9cKST3xXtFAOVBHf1GKKACiiigDgPiz4V8W+KvDv2bwfr39nkKfOtMeX9qHp5o5X0x0Oea4H4XeEfHEei3Ol6d4qufCs9g6rdWc/hi3IZiDhlm3fvhgfe69OxGffaKAPOv8AhC/iSeG+K5x3x4dtgfz3Uf8ACDfEH/oqtx/4JLf/ABr0WigDzr/hBviD/wBFVuP/AASW/wDjR/wg3xB/6Krcf+CS3/xr0WigDzr/AIQb4g/9FVuP/BJb/wCNH/CBeO25k+Kd6W/2NJgUflmvRaKAPOv+EB8cf9FSv/8AwVwUf8ID44/6Klf/APgrgr0WigDzr/hAfHH/AEVK/wD/AAVwUf8ACv8Axq3D/FLUSvfbpsKn869FooA86/4V34u/6Khq/wD4BxVV1P4UeItY06Ww1P4karcWsuN8TWcWGwQR09wDXp9cp8T7Y3Hwv8RsLi4g8jTLmYeRKYyzLCxUEjnGQDgEZxg5BIKaTVmVGUoSUouzRyv/AAqHxJ/0VXxL/wB/D/8AF1yHxS+G+uaH8Or/AFG98f65q8EMkAe0unJjfdMign5j0JBHuK9807/kF2v/AFxT/wBBFcL8dmCfBbXGborWpP8A4ExUuWPYt1qrVnJ/eZ4+EOv4+f4peKCe5E5H/s1H/Codd/6Kj4p/8CD/APFV6lRRyrsHtqn8z+88t/4VDrv/AEVHxT/4EH/4qj/hUOu/9FR8U/8AgQf/AIqvUqp6vdXllpNxcaXp7aleIv7m0WZYvNbOAC7cKO5PPAOATgF8q7B7ap/M/vPO9P8AhBq2mRzR2nxF1+JJ5mnl2BA0kjYyxY5JJwM1b/4Vhrn/AEUvxN/38T/Ct34f+INT8SeHJ7rXYrSG9gv7m0kWz3eX+6lZOC3J+714z6DpXUUJJKyIlKU3zSd2edf8Kw1z/opfib/v4n+FH/CsNc/6KX4m/wC/if4V6LRTJPOv+FYa5/0UvxN/38T/AAo/4VZqh5b4k+LM98XSgfltr0WigDzr/hVep/8ARSPFv/gWv/xNH/Cq9T/6KR4t/wDAtf8A4mvRaKAPOv8AhVep/wDRSPFv/gWv/wATR/wqm/biX4j+MCP9i+VT/wCg16LRQB51/wAKluf+ijeNf/BmP/iKP+FS3P8A0Ubxr/4Mx/8AEV6LRQB51/wqW5/6KN41/wDBmP8A4ivPfEPwX8S+K/GS2J1fWjpOnsP+Jrrd6tw0hIBPkRqARj1JA47YxX0M7iONnbJCgk7VJP4AcmuW8GeO7Xxi1+kNhfWj2d3PAPPtJ0R1jk2Bt7xqoc9THncvII4NAG1oOk/2FodrppvrvUPs6bftN7J5ksnuzd60KKKAOA+LPhXxb4q8O/ZvB+vf2eQp860x5f2oenmjlfTHQ55rgfhd4R8cR6Lc6Xp3iq58Kz2Dqt1Zz+GLchmIOGWbd++GB97r07EZ99ooA86/4Qv4knhviucd8eHbYH891H/CDfEH/oqtx/4JLf8Axr0WigDzr/hBviD/ANFVuP8AwSW/+NQ3nw78d39jPZ3fxSnkguI2ilT+xIBuVhgjIOehr0uqmqf2idPcaKbVbxmUI92GMaAsNzFVwWIGSFyuSANy5yDcabi7rc8sh+Dfi62gjgt/ixrEUUahEjS3KqigYAAEvAA7VKvwh8XY+f4ta8T/ALKsP/aldX4W8R6vd+MPEHhvXvsVxPpSW00d7YQvCjpMpOx42dyrAo3O8ggjgY5zvFnja60fx5Dof9veH9AtH037Z9q1iEv5knmlNi/v4h057ng1HJDsbvFYh7zf3sxv+FQ+LP8AorPiD8m/+OUf8Kh8Wf8ARWfEH5N/8cr1CwaV9Pt2ubiC5laNS09vGUjkOPvKpZsA9huP1NWKfJHsL6xW/nf3s8n/AOFQ+LP+is+IPyb/AOOUo+EPirPz/FjxCR3A3D/2pXq9FHJHsH1it/O/vZ5V/wAKh8Sf9FV8S/8Afw//ABdH/CofEn/RVfEv/fw//F16rRRyx7C9vV/mf3s8q/4VD4k/6Kr4l/7+H/4uuP8Ain8Odb0L4d32oX3xA1zV4Y5IA1pdOTG+6VFBPzHoTke4r6Frzr49cfBnWHP3Y5LZm+n2iOjlj2E61Vqzk/vKQ+EOv4+f4peKCe5E5H/s1H/Codd/6Kj4p/8AAg//ABVepUUcq7B7ap/M/vPLf+FQ67/0VHxT/wCBB/8AiqP+FQ67/wBFR8U/+BB/+Kr1KinyrsHtqn8z+88uX4Q63n5/ih4rI/2boj+tO/4VDq3/AEU7xd/4GmvT6KOVdhe1qfzM8w/4VDq3/RTvF3/gaaP+FQ6t/wBFO8Xf+Bpr0+iiyF7SfdnmH/CodW/6Kd4u/wDA008fCDUMfN8S/GRPfGokf0r0yiiyD2k31PNP+FQX/wD0Urxn/wCDI/4Uf8Kgv/8AopXjP/wZH/CvS6KLIXPLueaf8Kgv/wDopXjP/wAGR/wpV+EF5n958SfGpH+zqeP6V6URlSD39DivM9b0OKDx54c0HQNR1yK5eU6jfu+u3sypaQkfKUeUqfMkKpyOm760WQc0u5J/wqCf/oo/jj/wbf8A2NRQfBYW11PcwePfGMdxc7fPlXUV3y7Rhdx2ZOBwM9K6T4h2Ws3/AIYjh8PJeyXAvrZ5o7G7+zTPAJQZVWTemCUyPvD61F4J1DTHvNU0q2ttbsNRs2ja5s9avpLqQK6ko6O0sqlDhh8jdVOQDRZBzSSavuZH/Cpbn/oo3jX/AMGY/wDiK898Q/BfxL4r8ZLYnV9aOk6ew/4mut3q3DSEgE+RGoBGPUkDjtjFfQzuI42dskKCTtUk/gBya5bwZ47tfGLX6Q2F9aPZ3c8A8+0nRHWOTYG3vGqhz1Medy8gjg0yTa0HSf7C0O100313qH2dNv2m9k8yWT3Zu9aFFFABRRRQAUUUUAFFFFABRRRQAUUUUAU9UsbjULPybTVLvS5Nwbz7RYmfH93EqOuPwzx1riPig00Pw3udCRdY1nUbhIzHJFpsk7S7ZkYl2giEanAPGFzjpXodFAFXTtQh1SxS7tkuEjfOFubaS3kGDjlJFVh07jnrVqiigAooooA4Pwlp17bfGL4hXtxZ3EVrd/2b9mnkiZY5tluwbYxGGweDjoa7yiigAooooAKKKKACiiigAooooAKyPE1zb22jkX1tqk9tM6xytpQmM0QPO/8AckS4yADsyeeRjJGvRQBwPw4sZ7LWvEZsbS9t/Ds80Utk2pQPHcyzlT57sZQJnGdgDS5PBAO0Cu+oooAKKKKACuR8X2GpjxT4Y13TNMm1SPTJblbi3tpYklCywlQy+ayKcMoBG4H5sgGuuooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigDm/wDhY/gj/ocvD/8A4NIP/iq0dZ0bTfFWiNY6j5k9hcr8y291JEJUIIwWjYFlIPTODWnXIfESx0yfSIbzWYPDIgtWbde+IbUXEdsGxwiErkswUffXoOvAoA6fT7GHTLCKztTMYYRtTzp3mfHu7ksfxNcn8XdA1PxR8K9X0fQrb7Vf3Pk+VD5ipu2zxseWIA4Unk9qsfDmws4fCdtqMHh3StDu79N1wmm2qwpKFZhG+AM4KncA2Su8jrmusoAKKKKACq9/cy2djJPb2M9/ImNttbNGJH5xwZGVfflh0/CrFFAHEfDW31jTbHUbLWdAvdNM2o3d7HNNNbujLLOzqv7uVjuAbnIxweTXb1ha34z0Lw7cmDV7uSF0iWeUpayyrDGzFQ8jIpEakqw3MQOD6VughlBU5B5BHegAooooAKKKKACiiigAooooAKKKKACuQ8D6fqWiXmvafqGmTRQT6tdX9vfCWJopkmk3hQA28MNxBBUDjqa6+igAooooAKKKKACiiigArlNT0y98KeHr+78F2Goa3q0iBILW/wBZmmQtn7xNxKQoGSTtIJxjPcdXRQBwfhODU9A0DUzH4Y1e51qVHvLi61K4s421S6xwuYpnEY6KowFRQB9djUta1uCDy/8AhDLvUxNaKxS3vLbashB3wyea6cDgblDAgnjjB6So7i4htLd57qWOGGMbnkkYKqj1JPAoAwvAfh658K+BtN0a+liluLVGDmEkxqWdm2pnnau7aMgcAcCuhqlo+s2Gv6Wmo6RcC5tJHdElVSASjlG6gHhlI/DirtABRRRQAUUUUAFcX8XdA1PxR8K9X0fQrb7Vf3Pk+VD5ipu2zxseWIA4Unk9q7SigAooooAKKKKACiiigAooooAKKKKACiiigArB0fwydN8Ua5rt1efa7rVGiRP3ewW0Ea4SIcnPzF2J4yW6cVvUUAYD6V4kW3u/I8SxG4e/a4tmn01XjhtyuBbsqupcA5beGVs4zwCC3w/4Zn0zWNR1rVtQXUdW1FY4pZIoPIhjij3bEjjLMQPmYklmJJ7DAroa4m58T6zafECx0Q3WiXK3U7h9PgD/AGq2tgjMs7uXxyVUbNn8XDHBoA7auQ8D6fqWiXmvafqGmTRQT6tdX9vfCWJopkmk3hQA28MNxBBUDjqa6+igAooooAKKKKACiiigAooooAKKKKACiiigAoormdRsvEz+IbaW3vYfsAkzsVCAgwfvrkFv++vwFXCKk7N2InJxWiudNRWf5Os/8/8AY/8AgE//AMdo8nWf+f8Asf8AwCf/AOO0cq7/AJ/5BzPt+X+ZoUVn+TrP/P8A2P8A4BP/APHaPJ1n/n/sf/AJ/wD47Ryrv+f+Qcz7fl/maFFZ/k6z/wA/9j/4BP8A/HaPJ1n/AJ/7H/wCf/47Ryrv+f8AkHM+35f5mhRWf5Os/wDP/Y/+AT//AB2jydZ/5/7H/wAAn/8AjtHKu/5/5BzPt+X+ZoUVn+TrP/P/AGP/AIBP/wDHaPJ1n/n/ALH/AMAn/wDjtHKu/wCf+Qcz7fl/maFFZ/k6z/z/ANj/AOAT/wDx2jydZ/5/7H/wCf8A+O0cq7/n/kHM+35f5mhRWf5Os/8AP/Y/+AT/APx2jydZ/wCf+x/8An/+O0cq7/n/AJBzPt+X+ZoUVzOsWXiiW9sTp19bjaX8x0iMaAcfeBZt3txxXRQCVbdBcujygfOyLtBPsMnFOUFFJp3uEZttq1iSiiisywooooAKKyzr9qvi1PDpjm+1vYtfB9o8vyxIEIznO7LDtjHetSgAopGZUUs5CqoySTgAVHaXdvf2cV1Y3EVzbTKHimhcOkinoQw4I9xQBLRVE6pjxCmk/Yb07rVrn7YIf9HGHC+WXz987shcdAT2q9QAUVT1fUodF0S+1S6WR4LG3kuJFjALFUUsQASBnA9RUmn3seo6bbX0AZYrmFJkDjDAMARnHfmgCxRRVG51T7NrVhp32G9m+2JK32qKHdDBsCnEj5+Utu+X1wfSgC9RRRQAUUUUAFFFFABRRRQAVma0dbWGJtAg0+5bcRNBfSvEHUjs6q+0g+qnPTjrWnRQBi+ENCfw34XttMlkjd42lkbygQimSRpCiA9FXftHsBW1RRQAUUUUAFZviHW7bw34bv8AWb3JgsYGmZV6vgcKPcnAHua0q4n4u5Hw4uXP+pjvLKSf/rkt1EW/DAyfYGgCp47tfFPiDRY7Gy0i5AuLNHeKC6t2t2nOd0NyJArtCMLzHywLAjpnp4df2+MG8O3dusUpsFvYJlfKzAOUkULjIKHYe+Q46YpNQ0HUb2/kuLbxZrGnxPjbbW0VmY0wAODJAzc4zyx5PYcVgatE8fxb8FW63ElzPb6bqBuJpAoeRMQLubaAoJfB4AGc4AoA7qiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACsfQ/EC65qGsRW9vtttNvPsYuN+fOkVVMmBjgKW29eSG6YrYrzfwBpd5qHhO4ig17UNIubbWtSW6+xR27ea5unPzedFJjAxjGPvc54wAd1qthc6jarFZ6teaU6uGM1mkLOwwRtPmxuuOc8DPA565p2mnX+jwXNxcavq3iFhHmO1nS0Riw5+QpHEMn/abH0qr/wi+r/9D34g/wC/Gn//ACLR/wAIvq//AEPfiD/vxp//AMi0AVfhvBqtj4Zkstb0a50udL26nAmmhkEizXEso2mN26BwDnHPTPWtnxDd6pp2mtfaRbRXptwZJrNsiSdB1EbZwHxnAIIJwMr1qh/wi+r/APQ9+IP+/Gn/APyLWtpdlPplm8d9q93qjby/n3qwqyDA+X91Gi4GCeRnk89MAEum6ja6vpVrqOnyia1u4VmhkH8SMMg/kas1xvwjVx8KNDLggPE7x57o0jFCPbaRj2rsqACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAK5nwPrN54i02/1e5l3WtxqM6WEWwDy4Im8odOSWaN35J+9gcAV01eY+C9G8NyfDuKy8Zabpd03h++u7J21KCOUQt57EYLg4LqYzx1yOvFAHp1c2/gu3m16DUbrVdSuIra7a9t7GWSMwxTMCCynZ5mPmPy79oz0qnpvg74davo9vqun+FfD0tlcxCaKb+yoVDIRkNgoCOPWsSaP4TWum219feFdHtori2S7wdBWQwwOSEll2RsI0bacM+Bwe4NAHpdYXiy11BtJk1HQ7uWDUrCN5oI958m4wMmOROhDYxuxuXOQeucjVvCfw50TTRfah4R0MQF0jXyNESdnZ2CqFSONmYkkdAasC68OeHvh9qeo6Lpa6RpyRSytbpprWLPJt2/6pkVtzYVQdvPGM8UAb2iatb69oFhq9ln7PfW0dxHnqFdQwB9+avVgeBNJuNC+Hug6XfDFzaafDFMo/hcINw/A5Fb9ABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAHl0vi7xFD8PNWtEvEk8VW+tNotvO8SDMkky+TIV27cCKRW6Y+U571ctPGep61ofgNbGX7NqWtXKm/BjU7Y4EZrlcEYGXUJ0BG7jBq3ceBb6T4tReIorm2GjN5d1cWzFvNa8jikhRxxt2+XIM5OcotN8K+A77QvH+r6tdz20umM0x0qGMsZIPtEiy3G4EBR86jbgnjPSgDvKKKKACiiigAooooAKKKKACiiigAryzWj4g8S/ETX9GtT+5062g+yRr4gudMZBImWmAgjYy/P8ALlztGzAHJz6nWZq/hrQtfaJte0XTtTaEERG9tEmKA9cbgcZwOnpQB5hdaprenasuqvdWmqa1b+AJ7hbqzw8NzKsiMJEwAGViA3AAOeAKu+HLfxRFcaFrNvfWjW11bSPOsviO4vzqeYS6mOF4lRHDKG/dFQF3DGMAenCxtBfLei1hF0sXkrP5Y3iPOdm7rtyAcdMiqWneGdB0i+mvdJ0TTrG7nBEtxbWkcckgJyQzKATkgHnvQBw3hOwttf8AAGma1qPiHVJtS1mydp0Gryok0rKWeNIw4WPYQR+7CsAhBJG7PP6VayWHwV8C/wBjarqVlJql9psNxNFfyuVVztdUDsyoOvyqAue3Fet2nhzQ7DVp9UsdG0+21C43eddw2qJNLuOW3OBk5IBOTyabb+GNBtAwtdE06ANcLdMI7SNczKciU4H3wejdaAOH168v/C/i2e10K4vZ0s/CGoXlvbXN3Lc+ZcJNGUZjIzFzkkAkkgHA44oN3pvhr4er4kfxHrF3PdaT57gap5puS3l5ljSUlI9rOBlAqqH5HC49FNlam/W+NtCbtYjCtwYx5gjJBKBuu0kA46ZAqha+FfD1i14bLQdMtzfIUuzDZxp9oU5yHwPmBycg560AeV3dzqunXXjPQb5vJhPhGe7a0/tyfVPLkG9d2+dQ6Eq33R8pCg1prr9/4Ot45/OmubfVfDUMum28jllW+hVIxEgJwvmebEcdyjHrnPoNn4U8O6dBLDp+gaXaxTRNDJHBZxorxt95CAOVOOR0NWp9H0y5S0S5060mWxdZLVZIFYW7qMKyZHykDoRjFAHnuj6XPeeK7vwz4j1/VXj0LR7Mw+Tqc1s908m/zrl3RleT5kCjJKr6ZOaz/DviTXL2bwRLLfXF2ZodYGd+Fv0hYLBI6rgMSoBBxzuJHWvTNX8OaH4g8n+3tG0/U/I3eV9ttUm8vdjO3cDjOBnHoKtHT7Nri2na0gM1qrLbyGIboVYAMEOMqCAAcdcCgDzTQruWP4WweM4dcvrzxBdaTcXQglv3eG5uPKdzEtuSUXYy4AjVWATBP3sp4QtvEj33hrWf7StVtL9CbtpvElxe/wBohoi/7uB4VjjcMu/92VAAYYxxXoVp4c0Ow1afVLHRtPttQuN3nXcNqiTS7jltzgZOSATk8mm6f4Z0HSdRm1DS9E06yvZwwluba0jjkkDEMQzKATkgE57igDmPE+sRa1r3hTTrHV2GiatPcie6068MZneFMpCs0ZBXLB8hSCfLI9RVOeJLnx5pPhP+3tSbR1sbu6UxanIs1xcJMqeS1wjCRhErt8u7d0LFttdrJ4b0ObRhpEujafJpgYuLJrVDCGLFidmNudxJ6dTmmXHhbw/d6RBpN3oWmz6dbsGhs5LONoYiMjKoRtB+Y9B3PrQB5fJqeqz2kGl2+uaj9ji8bDTLe+iuSZZbXyCzRmQ5L7XLLuOT8oydy5q1LpVw2pePNPHiDX1s9FtY59PjXVZt0EkluWLebu8x8FchXZkGT8tenJo2mR2lpax6daLb2Lq9rCIFCW7KCFKDGFIBIBGMZNSf2bY+ZdSfYrffeKFuW8pczgDaA5x8wAJHPagCl4Uv7jVfBmi6jesHubvT4J5WAwC7xqxOO3JNa1R29vDaW0VvaxJDBCgSOKNQqooGAoA4AA4xUlABRRRQAUUZooAKKKKACqmq6Za61o95pmoR+Za3kLwTJ6qwIP6GrdGaAPMviMp0nRJPsWr6215p+nRl54NRWNbCNSw+2TRKyNMWw2V2sreXgBeSe5ttGtR4gk1/zZZrqezjtVLkbI41Zn+UYGCxbJznO1emKk1Dw/o2r3UNzqukWN9Pb/6mW5tkkaPv8pYEj8K0KACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACseHw7DYzatNp95c2bapOtxNsKERuFVWZAykAsqDOc+owa2Kr31hZ6pYyWep2kF5ayjEkFxGJEcZzypyDyKAON8MeILiz+C9lrFxPJqF59lxG00m57iZnKRqWPdnKr+NM1rw8LLwxZaPb6rqh1U2rx20dletbebcHl7mRkwdoZtzFsr82NpYgHdXwZpNq9hHo9lZaVZW159smtbO0SJbiQIVQttx90kN0zlF7CrmreGNA1+WOTXdD03U5IlKxveWkcxQHqAWBwKAMPxC8moeC49M07X7WTU99iHmGpG0M4MqMw8yLLx+aiSAFRk7uKpaFav4o8L3+iia70uK1vpbHUl/tCW/eb5VZliuZTuCkOATtBHzAbSM11c3hzQ7m1ktbjRtPlt5USOSF7VGR1QYRSCMEKOAO3arVjYWel2UdnptpBZ2sQxHBbxiNEHsowBQBJb28VpaxW9tGsUMKCOONBgIoGAAPQCpKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKz71V0u1vL7TtHa9u5WWSSG0ESS3DgKgJZ2VSQoHLN91cDPArQooA800e31Sz+EuheEtT0640zUrlYtIkSSWNyYwmZ5UaJ2wPLWQAnB3YyBkZ2vGJ1SZU0az8MXup6NJbn7T9jmtkEw6C3PmyoVQgfMQDwcDqSOte3hkuI53hjaaIMI5CoLIGxkA9RnAz9BUlAHJ+I0v9T02azufC1zeW0VzbtGtlqi28zgAOXjYMm1kcBcF1z1Bpnhbw1cPpkL+KorieW2vJZ7C3vrs3L20ZI8sSNuKySLjIYliucBjyT19FABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAHMJ4Z1JPEv29tauJIijKDhd8eewBUrj6AVr/2bdf9Bq+/74g/+N1oUVrKrKW/5IzjSjHb82Z/9m3X/Qavv++IP/jdH9m3X/Qavv8AviD/AON15zN4v8Tr8LZ7W2vEk8YJrDaGlw0SYM3ncSFNu3/UYfG3HfFa1v4yvtdt/AS6XN9muNb/ANLv1KBisEURMycjj96yLng9cYqed/0kPkX9NnYf2bdf9Bq+/wC+IP8A43WXrvhzUNShtkt9ZuN0cwcvKEGwYIyuxFOefXFdJRVRqyi7r8kKVOMlZ/myCyt5LWzjhnuZLp1GDLIAC35VPRRWbd3ctKysFFFFIYUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQBmXHiLS7bUY7GS7Q3EjbdinO3/ePQfjVs6jZAZN5bge8q/41UuPD2l3OoR30lpGLiNt29Rjd/vDofxq4LG0ByLWEH/rmK1fs7K1zNe0u72Gf2nYf8/1t/3+X/Gg6pp4GTfWwH/XZf8AGpPsdt/z7xf98CgWlsDkW8QP+4Kn3B+8Q/2vpv8A0ELX/v8Ar/jR/bGmj/mI2n/f9f8AGrH2eH/nlH/3yKPIh/55J/3yKPcD3ir/AG3pX/QTs/8AwIT/ABo/tvSv+gnZ/wDgQn+NXPLT+4v5UeWn9xfyo9zsHvdyj/b+j/8AQWsf/AlP8aP7f0f/AKC1j/4Ep/jWhRReHb8f+AFpd/w/4Jn/ANv6R/0E7T/v8v8AjR/b2kf9BK1/7/L/AI1oUUXh2/H/AIAWl3/D/gmf/b2ldtQtz9HBo/t7Sv8An/g/77rQoovDt+P/AAAtLv8Ah/wTP/t7S+17Gfpk0f29pn/P4n5GtCii8O34/wDAC0u/4f8ABM/+3dN7XQP0Vj/Sj+3dO/5+P/HG/wAK0KKLw7fj/wAALS7/ANfeZ/8Abun9p2P0ic/0o/tzT/8Anq//AH5f/CtCii8O34/8ALS7/h/wTP8A7csO0kp+lvIf/ZaP7bsf70//AICy/wDxNaFFF4dvx/4AWl3/AA/4Jn/23ZdvtB+lpKf/AGWj+27P+7df+AU3/wATWhRReHb8f+AFpd/w/wCCc/rHi600m2im8i5lVpQjAwPHgYJyCygE8dM1sWF7FqNlHdQCRY5BkCRCh/I1JNbQ3Oz7REkvlvvTeoO1sYyPfk1JTk4cqSWooqfM23oFFFFZmgUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAcKPAt8nxbPiFZ7X+xWAvGtst5ovvJMG8DG3b5R65znt3qhovgTWPDWpeJ9RjuIJYvs1ymgw25Yy24mkeeRWBAGTIUAxnhRmvSaKAPOfhx/YZ1Yt4Q+zmxOj232422Nputz/6zH/LfGd+fn+5u7V6NRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFVdU06DV9HvNNuxm3vIHt5R6q6lT+hq1RQB4Jqj6t4r+Hfhvw1ZXDx6vo0V3cXZU8+dpwMMYb2aYxt74rsbLxDYeJfG1n4llmWLSNC8PC/aZzhY5Lsbjn/dihP03/n1OkeCtM0TxdrHiGzkuTdatt82KSQGKLA+YxrjK7iAWyTkjtWGvw2tfDvgHXdI8LebcTX0huY472UEblC7IQQBtjAQKBzgGgDrNH12w12GaTTnmzBJ5U0VxbSW8sbbQwDRyKrDIYEZHIORWjXPeHIL+XVtX1fUNPm0wXrQpFazyRvIFjTBZvLZlBJY4AY8AZx0HQ0AFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUjuscbPIwRFBLMxwAPU0tcv8AEySeL4VeKHtciRdKucEHBA8tsn8Bk0AWtL8WwaxHos1lpupm21mCS4huTb/u4kUAqZTn5N4YFB3Gc4xVzQNctfEWjx6hZLLGjO8TxTACSKRHKOjAEgEMpHU1yslzJaeMfh/YWFxLHYTafdhoI3IjkCQxbNyjg4zxnpmpvh3ldU8axx/8e6eI5jH/ALzQws4/76JP1JoA7aiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAK5+bxhap/an2PT9S1A6XdxWcws4BIWkfYTtGckIJFLHsM9cGugrxqK9u7H4J6te2tzLa6g/iZ/PlgkKuGbVVRhuGDynyn1BxQB6Lqfi+20iDWbi/07Uo7XR2iE1wYAI5UcKTJGSfnVA3znqNrYBI56AEMoKnIPII7151PcyXuo/FCy1G4llsYbWJEhkcskSNZZcKDwAcknHWum8ByTzfDnw5Jef8fD6VbNJk5O4xLmgDfooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKy7jX7a38QxaKIp5rt7OS8YRKCscaMq/Mc9WLYUDOdrdMVqVwukkt8XPGry8Sxabp6QHPIjInJx/wPP5CgDdsvFH2z+zv+JHrMH2+0e6/f2m37Ptx+7l5+WQ7uF5zg+lJZeLba6uNFt7iw1DT5tZt5JreO9hEbIyBS0TjJKybWLY6YRueK8/8AB2tapdf8K9+06leTfbPDl5Nc+ZOzefIvkbXfJ+Zhk4J5GTTbW8ur34cfCrULq4ludQk1WzJnmkLSSbopRJljkn5Nx564oA9gooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKiubaG8tJrW6jWWCdGjkRujKRgg/UGpaKAMXw/4ch0bRdIs7t49RudJt/s1teywASqmAvB5IJVVDEH5sZ9qd4X8Pr4b0Y2huPtVxNcS3V1c7NnnTSuXZtuTgZOAMnAAFbFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFYGn+ErOzm1lJxDeafql6t/9jngVlilwpY85By6Bxxw2T9N+igDD8ReF7fXNC1XT7cx2EmrosV7dRQAyTR8KykjBJMe5ASTtznBxitmKJIIUihQJHGoVFUYCgcAU+igAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKxpPDyHxmniCGcIzWLWV1AY8idd4eNs54KkyDochz0xWzRQBSg0XS7X7N9m02zh+xxNDbeXAq+RG2NyJgfKpwMgcHArOm8J2kmtaHcwCG2sNESU22nwQBEErqEVxjgBUMgCgdXznit6igAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigD/2Q==)

**Time Reversal:** If the independent variable **t** is replaced by ‘−***t’***, this operation is known as **time reversal** of the signal about the y-axis or amplitude axis. This can be achieved by taking mirror image of the signal x(t) about y-axis or by rotating x(t) by 180° about y-axis. Hence, time reversal is known as folding or reflection.



**ALGORITHM:**

1. Get input amplitude and frequency from user.
2. Construct input signal and display plot.

Signal = A\*Cos(2\*Π\*f\*t)

1. For time shifting: t = **t−t0**  OR t = **t+t0.**
2. For time scaling: t = ***at*  (**0 < a < 1) and (a>1).
3. For time reversal: t = -**t**
4. For sum: Add zeros to signal to make their range same and then add the signals.
5. Construct required signal and plot continuous and discrete signal.

**CODE:**

1. Time Shifting:

close all;

clear all;

t=-10:0.01:10; %Time range

f=Input('Enter frequency: '); %Input freq

A=Input('Enter Amplitude: '); %Input amplitude

x=A\*cos(2\*pi\*f\*t); %Continuous input signal

subplot(3,2,1)

plot(t,x) %Plot

xlabel('Time (sec)')

ylabel('x(t)')

title('Continuous time signal')

%Right Shift

t0=2; %Shifting factor

x=A\*cos(2\*pi\*f\*(t-t0)); %Right shift signal

subplot(3,2,3)

plot(t,x) %Plot

xlabel('Time (sec)')

ylabel('x(t)')

title('Right Shift (Positive Shift)')

%Left Shift

x=A\*cos(2\*pi\*f\*(t+t0)); %Left shift signal

subplot(3,2,5)

plot(t,x) %Plot

xlabel('Time (sec)')

ylabel('x(t)')

title('Left Shift (Negative Shift)')

n=0:4; %Discrete signal range

x=[0 1 2 3 3]; %Discrete signa;

subplot(3,2,2);

stem(n,x); %Discrete plot

xlabel('Time (sec)')

ylabel('x(t)')

title('Discrete time signal');

axis([-4 8 0 4]) %Adjust axis

%Right shift

m=n+2; %Shifting Factor

subplot(3,2,4);

stem(m,x); %Discrete Plot

xlabel('Time (sec)')

ylabel('x(t)')

title('Right Shift (Positive Shift)');

axis([-4 8 0 4])

%Left shift

m=n-2; %Shifting factor

subplot(3,2,6);

stem(m,x); %Discrete Plot

xlabel('Time (sec)')

ylabel('x(t)')

title('Left Shift (Negative Shift)');

axis([-4 8 0 4])

**Shape

Description automatically generated**

1. Time Scaling:

close all;

clear all;

t=-10:0.01:10; %time range

f=Input('Enter frequency: '); %Input freq

A=Input('Enter Amplitude: '); %Input amplitude

x=A\*cos(2\*pi\*f\*t); %Continuous input signal

subplot(3,2,1)

plot(t,x) %Continuous Signal plot

xlabel('Time (sec)')

ylabel('x(t)')

title('Continuous time signal')

%Compression

t0=2; %Scaling factor

x=A\*cos(2\*pi\*f\*(t\*t0)); %Compression freq

subplot(3,2,3)

plot(t,x) %Continuous Signal plot

xlabel('Time (sec)')

ylabel('x(t)')

title('Compressed')

%Expansion

x=A\*cos(2\*pi\*f\*(t/t0)); %Expansion Signal

subplot(3,2,5)

plot(t,x) %Continuous Signal plot

xlabel('Time (sec)')

ylabel('x(t)')

title('Expand')

n=0:4; %Discrete signal range

x=[0 1 2 3 3]; %Discrete signal

subplot(3,2,2);

stem(n,x); %Discrete plot

xlabel('Time (sec)')

ylabel('x(t)')

title('Discrete time signal');

axis([-4 10 0 4]) %adjust axis

%Compression

m=n/2; %Scaling factor

subplot(3,2,4);

stem(m,x); %Discrete plot

xlabel('Time (sec)')

ylabel('x(t)')

title('Compressed');

axis([-4 10 0 4])

%Expansion

m=n\*2; %Scaling factor

subplot(3,2,6);

stem(m,x); %Discrete plot

xlabel('Time (sec)')

ylabel('x(t)')

title('Expand');

axis([-4 10 0 4])

Diagram

Description automatically generated

1. Time Reversal:

close all;

clear all;

t=-10:0.01:10; %time range

T=10;

f=1/T;

A=1;

%f=Input('Enter frequency: '); %Input freq

%A=Input('Enter Amplitude: '); %Input amplitude

x=A\*cos(2\*pi\*f\*t); %Continuous input signal

subplot(3,2,1)

plot(t,x) %Continuous Signal plot

xlabel('Time (sec)')

ylabel('x(t)')

title('Continuous time signal')

%reversal

x=A\*cos(2\*pi\*f\*(t.\*-1)); %reversal signal

subplot(3,2,3)

plot(t,x)

xlabel('Time (sec)')

ylabel('x(t)')

title('Reverse')

n=-4:4; %Discrete signal range

x=[-3 -3 -2 -1 0 1 2 3 3]; %Discrete signal

subplot(3,2,2);

stem(n,x); %Discrete plot

xlabel('Time (sec)')

ylabel('x(t)')

title('Discrete time signal');

axis([-8 8 -4 4]) %adjust axis

%Reversal using inbuilt function

m=fliplr(n); %inbuit function

subplot(3,2,4);

stem(m,x); %discrete plot

xlabel('Time (sec)')

ylabel('x(t)')

title('Reverse');

axis([-8 8 -4 4])

Chart, line chart

Description automatically generated

1. Summation of Signals:

close all;

clear all;

n1=-2:1:6; %signal 1 range

x1=[1 2 3 4 5 -1 2 -3 1]; %signal 1

n2=-5:1:3; %signal 2 range

x2=[2 3 -1 -3 2 6 3 1 -2]; %signal 2

subplot(3,1,1);

stem(n1,x1); %plot signal 1

xlabel('Time (sec)')

ylabel('x(t)')

title('1st Signal');

axis([-7 7 -4 7]) %adjust axis

subplot(3,1,2);

stem(n2,x2); %plot signal 2

xlabel('Time (sec)')

ylabel('x(t)')

title('2nd Signal');

axis([-7 7 -4 7])

%Adding zeros to left and right of signal to match range

n=min(min(n1),min(n2)):max(max(n1),max(n2));

y1=zeros(1,length(n));

y2=zeros(1,length(n));

y1((n>=min(n1))&(n<=max(n1)))=x1(); %signal 1 with zeros added

y1((n>=min(n2))&(n<=max(n2)))=x2(); %signal 2 with zeros added

x=y1+y2; %Add signals

subplot(3,1,3)

stem(n,x); %plot sum signal

xlabel('Time (sec)')

ylabel('x(t)')

title('Addition of Signal');

axis([-7 7 -4 7])

Timeline

Description automatically generated with medium confidence

**CONCLUSION:** Thus, different properties of discrete and continuous time signal such as time shifting, time scaling and time reversal can be implemented . Also summing of two discrete and continuous time signal can be implemented using MATLAB.

**Practical 3**

**Date: 13/08/2021**

**AIM:** Write a MATLAB program to perform linear convolution of given signals with and without inbuilt function



Write a MATLAB program to perform circular convolution of given signals with and without inbuilt function

x[n] = cos(2πn/N) and h[n] = sin(2πn/N) where 0 ≤ n ≤ N-1 and N = 8

**THEORY:**

**Linear Convolution:** Linear convolution is a mathematical operation done to calculate the output of any Linear-Time Invariant (LTI) system given its input and impulse response. It is applicable for both continuous and discrete-time signals. We can represent Linear Convolution as

y(n)=x(n)\*h(n)

where, y(n) 🡪 output, x(n) 🡪 signal, and h(n)🡪 impulse response of the LTI system.

It is possible to find the response of a filter using linear convolution. Linear convolution may or may not result in a periodic output signal.

**Circular Convolution:** Circular convolution is essentially the same process as linear convolution. Just like linear convolution, it involves the operation of folding a sequence, shifting it, multiplying it with another sequence, and summing the resulting products. However, in circular convolution, the signals are all periodic. Thus the shifting can be thought of as actually being a rotation. Since the values keep repeating because of the periodicity. Hence, it is known as circular convolution. We can represent Circular Convolution as

y(n)=x(n)⊕h(n)

where, y(n) 🡪 output, x(n) 🡪 signal, and h(n)🡪 impulse response of the LTI system.

The output of a circular convolution is always periodic, and its period is specified by the periods of one of its inputs.

**ALGORITHM:** Linear Convolution

1. Initialize 1st and 2nd sequence.
2. Add zeros to both sequences to make range equal.
3. Perform convolution with and without inbuilt function.
4. For convolution without inbuilt function:

for n=1:N

for k=1:n

y(n)=y(n)+x(k)\*h(n-k+1);

end

end

1. For inbuilt function use conv(x,h).
2. Plot discrete plot of 1st, 2nd and convoluted signal.

**CODE:**

1. **Linear Convolution (Without Inbuilt Function):**

clear all;

close all;

a=0:6;

x=(a./3); %first sequence

b=-2:2;

h=ones(1,length(b)); %second sequence

disp('First sequence is: ');

disp(x);

disp('Second sequence is: ');

disp(h);

n1=length(x); %length of 1st sequence

n2=length(h); %length of 2nd sequence

N=n1+n2-1;

x=[x,zeros(1,N-n1)]; %Added zeros to 1st sequence to make equal range

h=[h,zeros(1,N-n2)]; %Added zeros to 2nd sequence to make equal range

y=zeros(1,N);

%convolution

for n=1:N

for k=1:n

y(n)=y(n)+x(k)\*h(n-k+1); %convolution equation

end

end

disp('Convolution without using inbuilt function');

disp(y);

ny=0:N-1; %range

subplot(3,1,1);

stem(ny,x); %discrete plot

xlabel('n---->');

ylabel('x(n)---->');

title('x');

subplot(3,1,2);

stem(ny,h); %discrete plot

xlabel('n---->');

ylabel('h(n)---->');

title('h');

subplot(3,1,3);

stem(ny,y); %discrete plot

xlabel('n---->');

ylabel('y(n)---->');

title('Convolution of x and h');

**Chart, box and whisker chart

Description automatically generated**

1. **Linear Convolution (With Inbuilt Function):**

close all;

clear all;

a=0:6;

x=(a./3); %first sequence

b=-2:2;

h=ones(1,length(b)); %second sequence

disp('First sequence is: ');

disp(x);

disp('Second sequence is: ');

disp(h);

n1=length(x); %length of 1st sequence

n2=length(h); %length of 2nd sequence

N=n1+n2-1;

y=zeros(1,N);

y=conv(x,h); %Convolution using inbuilt function

disp('Convolution with inbuilt function')

disp(y);

Chart, text

Description automatically generated

**ALGORITHM:** Circular Convolution

1. Initialize 1st and 2nd sequence.
2. Perform convolution with and without inbuilt function.
3. For convolution without inbuilt function:

for n=1:N

for m=1:N

z=mod(n-m,N);

y(n)=y(n)+x(m)\*h(z+1);

end

end

1. For inbuilt function use cconv(x,h,N).
2. Display results.

**CODE:**

close all;

clear all;

n=0:7;

N=8;

x=cos(2\*pi\*n./N); %first sequence

h=sin(2\*pi\*n./N); %second sequence

disp('1st Sequence is: ');

disp(x);

disp('2nd Sequence is: ');

disp(h);

y=zeros(1,N);

%Convolution

for n=1:N

for m=1:N

z=mod(n-m,N);

y(n)=y(n)+x(m)\*h(z+1); %Convolution equation.

end

end

disp('Circular Convolution without inbuilt function');

disp(y);

z=cconv(x,h,N); %Convolution using inbuilt function

disp('Circular Convolution with inbuilt function');

disp(z);

**RESULT:**

**Text, letter

Description automatically generated**

**CONVOLUTION:** Thus, we implemented Linear and Circular Convolution for the given input signal and impulse response with and without using inbuilt function and also plotted both the signals and hence verified the two output plots obtained from with and without using inbuilt function.

**Practical 4**

**Date: 27/08/2021**

**AIM:** Write a MATLAB program to compute DFT and IDFT for the given signal x[n] = [ 1 1 1 1 1 1 0 0 1 ]. Also plot the magnitude and phase response for DFT.

**THEORY:**

The discrete Fourier transform (DFT) is the primary transform used for numerical computation in digital signal processing. It is very widely used for spectrum analysis , fast convolution , and many other applications. The DFT is widely used in part because it can be computed very efficiently using fast Fourier transform (FFT)algorithms.

The inverse DFT (IDFT) transforms N discrete-frequency samples to the same number of discrete-time samples. The IDFT has a form very similar to the DFT,

**CODE:**

1. Without inbuilt:

clear all;

close all;

x=input('enter sequence: '); %Input signal

N=input('enter N= '); %Input N

L=length(x); %Length of input signal

n=0:N-1;

x=[x zeros(1,N-L)];

subplot(3,2,1)

stem(n,x) %Discrete plot of input signa;

axis([0 10 -2 2]) %Adjust axis

xlabel('n')

ylabel('amplitude')

title('Input Sequence')

%DFT

y=zeros(1,N);

for k=0:N-1

for n=0:N-1

y(k+1)=y(k+1)+x(n+1)\*exp((-j\*2\*pi\*k\*n)/N); %DFT equation

end

end

%disp(y);

k=0:N-1;

subplot(3,2,2);

stem(k,y) %plot dtf

axis([0 10 -2 2])

xlabel('k')

ylabel('amplitude')

title('DFT')

magnitude=abs(y); %magnitudfe response

subplot(3,2,3)

stem(k,magnitude) %plot magnitude

axis([0 10 -2 9])

xlabel('k')

ylabel('amplitude')

title('Magnitude')

phase=angle(y); %phase response

subplot(3,2,4);

stem(k,phase) %plot phase

axis([0 10 -3 3])

xlabel('k')

ylabel('phase')

title('Phase')

N=length(y);

%IDFT

m=zeros(1,N);

for n=0:N-1

for k=0:N-1

m(n+1)=m(n+1)+((1/N)\*(y(k+1)\*exp((j\*2\*pi\*k\*n)/N))); %IDFT equation

end

end

% disp(m)

n=0:N-1;

subplot(3,2,5)

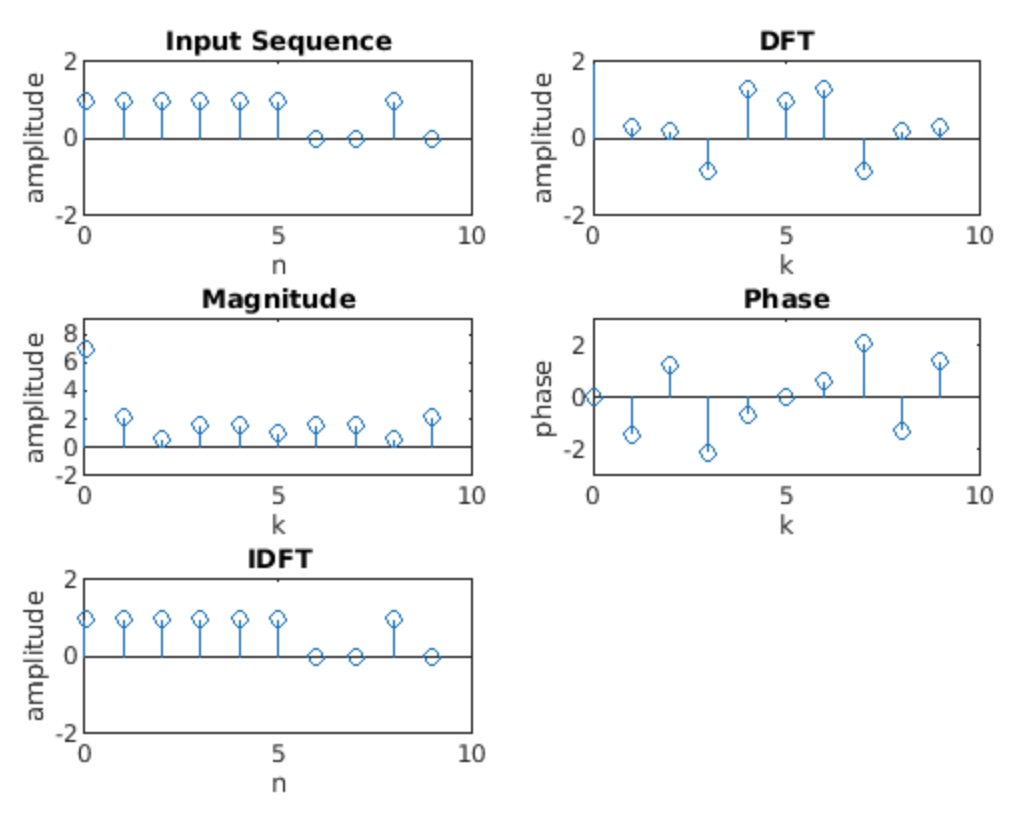
stem(n,m) %plot idft

axis([0 10 -2 2])

xlabel('n')

ylabel('amplitude')

title('IDFT')

****

1. With Inbuilt function:

clear all;

close all;

% x=input('enter sequence: '); %Input signal

% N=input('enter N= '); %Input N

xn=[1 1 1 1 1 1 0 0 1];

N=10;

k=0:N-1;

L=length(xn); %Length of input signal

xn=[xn zeros(1,N-L)];

subplot(3,2,1)

stem(k,xn)

axis([0 10 -2 2])

xlabel('k')

ylabel('xn')

title('Input Sequence')

%DFT

Xk=fft(xn,N); %DFT inbiult function

subplot(3,2,2)

stem(k,Xk) %plot dft

axis([0 10 -2 2]) %adjust axis

xlabel('k')

ylabel('xk')

title('DFT')

%IDFT

Id=ifft(Xk,N); %IDFT inbuilt function

subplot(3,2,3)

stem(k,Id); %plot idft

axis([0 10 -2 2])

xlabel('k')

ylabel('xk')

title('IDFT')

Chart, scatter chart

Description automatically generated

**CONCLUSION:** Thus, we implemented the Discrete Fourier Transform for the given input signal and obtained its output signal with and without using inbuilt function and also plotted both the output signals and hence verified the two output plots. We also plotted and observed the magnitude and phase plots of DFT. Also the input signal was obtained from the output signal by using Inverse Discrete Fourier Transform with and without using the inbuilt function.

**Practical 5**

**Date: 03/09/2021**

**AIM:** Write a MATLAB program to compute N point DIT-FFT and verify the results using inbuilt FFT command.

**THEORY:**

Text

Description automatically generated with medium confidence

Diagram

Description automatically generated with medium confidence

**CODE:**

clear all;

close all;

y=[0 1 2 3 4 5 6 7]; %Input signal

n=length(y);

p=nextpow2(n); % Increasing the performance of fft when the length

%of the signal is not a power of two

z=zeros(1,2^p-n);

x=[y z];

y=bitrevorder(x); % Bit Reversal to obtain the correct order

n=length(y); % Length of the Input Signal

s=log2(n); % Number of stages

w=exp(-2\*1j\*pi/n).^(0:(n/2-1)); % Twiddle Factor

%DIT

for m=1:s

for k=1:2^m:n

for l=0:2^(m-1)-1

a=y(k+l);

b=y(k+l+2^(m-1))\*w(l\*n/(2^m)+1);

y(k+l)=a+b;

y(k+l+2^(m-1))=a-b;

end

end

end

k=1:n;

subplot(3,1,1);

stem(k,x); % Plot for Input Signal

xlabel('n');

ylabel('Amplitude');

title('Input Signal');

y=(round(y\*100))/100;

subplot(3,1,2);

stem(x,y);

xlabel('y');

ylabel('Amplitude');

title('DIT');

disp(y);

%DFT using inbuilt function

y1=fft(x);

y1=(round(y\*100))/100;

subplot(3,1,3);

stem(x,y1);

ylabel('Amplitude');

xlabel('y');

title('DFT')

disp(y1);

**RESULT:**

**Chart

Description automatically generated**

**CONCLUSION:** Thus, we implemented the Decimation in Time Fast Fourier Transform Algorithm or Radix 2 Algorithm and verified the output obtained from inbuilt function.

page23image7249024