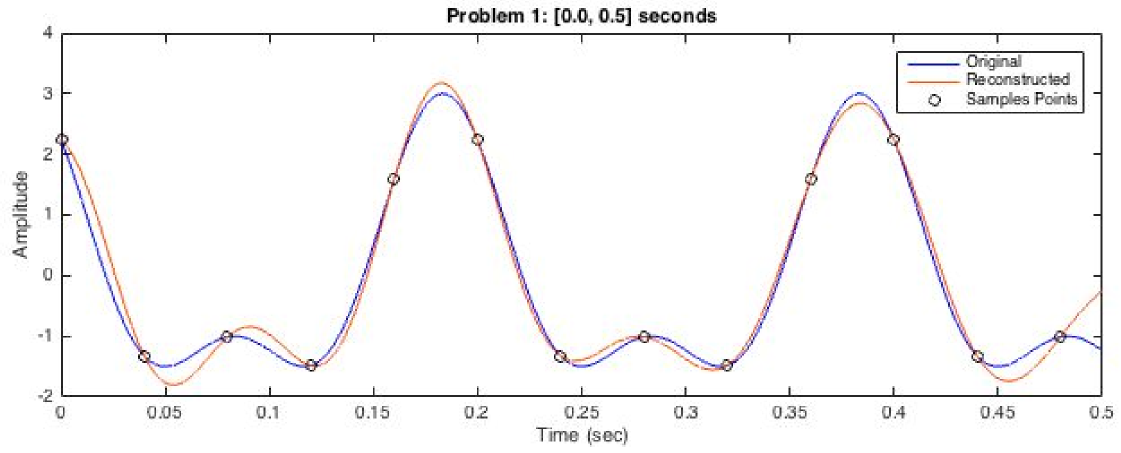
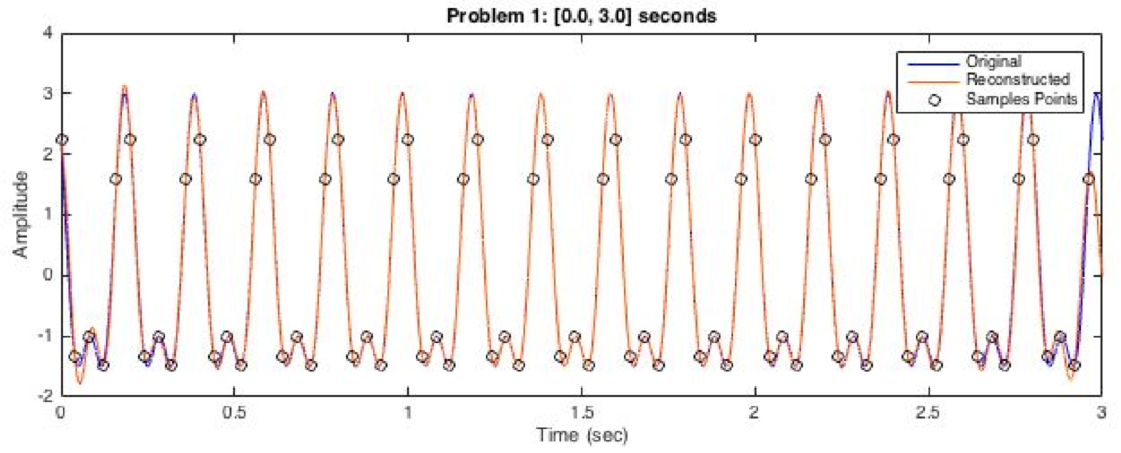
Garrett Sickles - Lab 8 - GPGN 404 - 4 December 2015

---

1. Original signal (blue) and reconstructed signal (orange) constructed with sinc interpolation plotted on the same image. The sampled points belonging to to $x[n]$ sampled at a frequency of $25\ Hz$ are indicated by open black dots in the image. The two

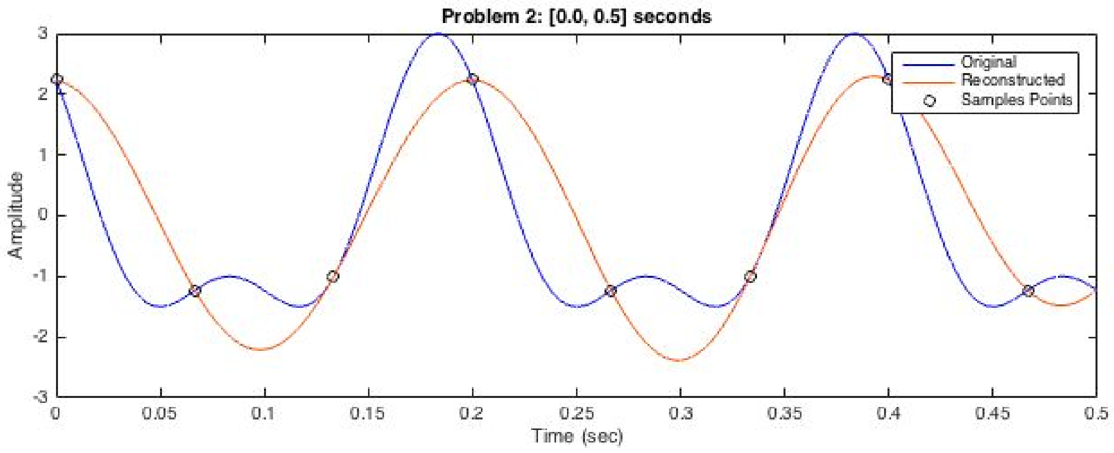


signals do appear to contain similar information. However, $x(t)$ does not appear to be *fully* recovered from $x[n]$. This is a function of the limited sampling size. If we increase the sampling time interval and use sinc interpolation the recovered signal would have the same edge case problem but the interior body of the signal would be practically identical. This can be seen in the following image where the signal length is increased from $0.5\ sec$ to $3.0\ sec$. As you can see on the interior of the signal, $(0.2 \leq t \leq 2.8)\ sec$,

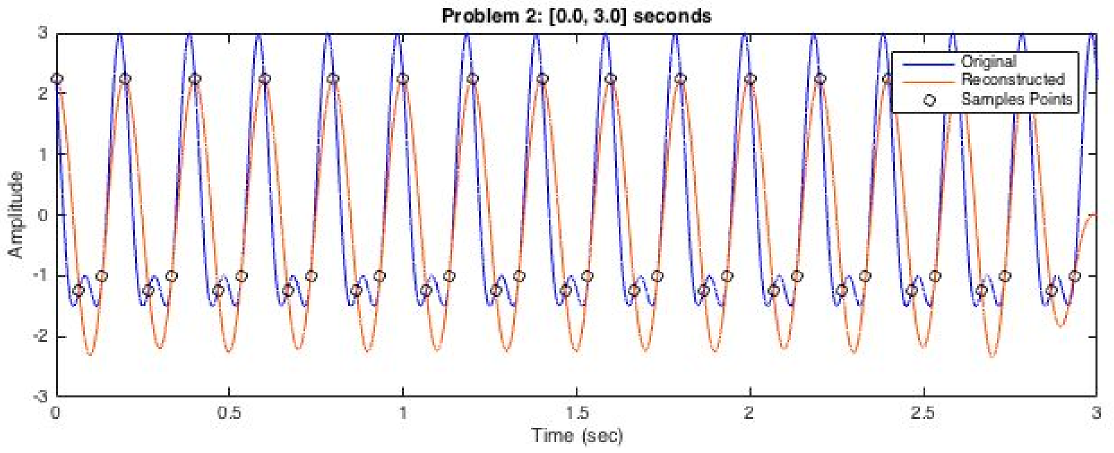


the sinc interpolation (orange) nearly perfectly overlays the original signal (blue). At the edges however the original signal (blue) and the sinc interpolated signal (orange) deviate, especially approaching 3 seconds. The signal is successfully recovered because the sampling frequency is greater than twice the Nyquist frequency, $f_{sampling} \geq 2 * f_{Nyquist}$ or $25\ Hz \geq 2 * 10\ Hz$.

2. Original signal (blue) and reconstructed signal (orange) constructed with sinc interpolation plotted on the same image. The sampled points belonging to to $x[n]$ sampled at a frequency of $15\ Hz$ are indicated by open black dots in the image. The two

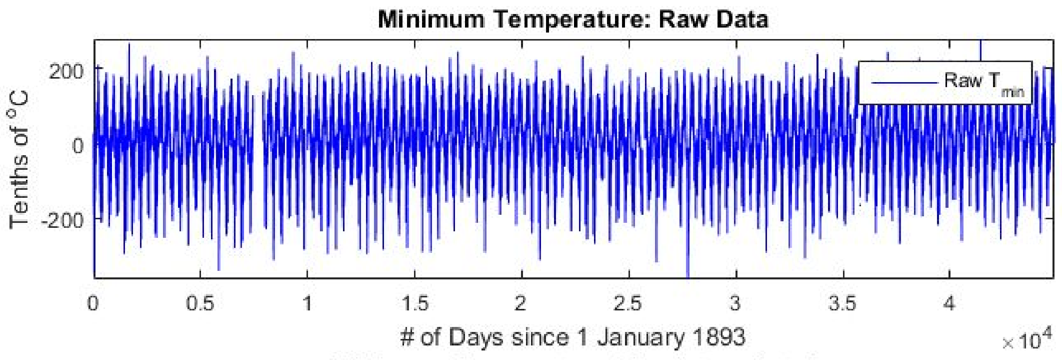


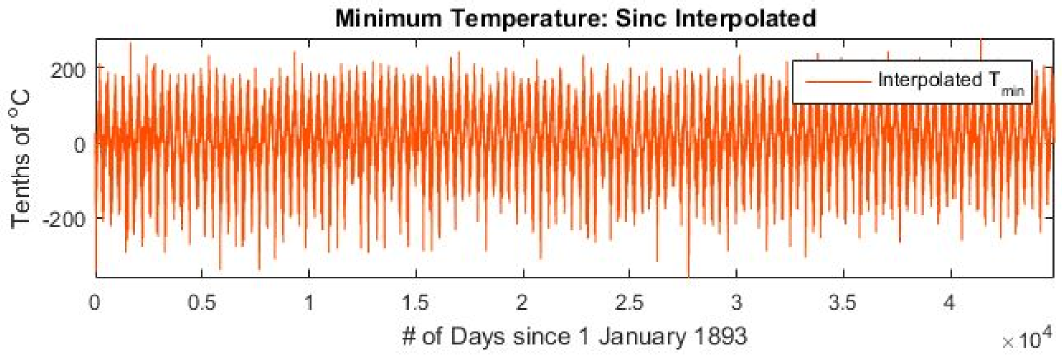
plots do not appear to contain the same information as we would expect dictated by sampling theorem. If we expand the bounds of our time interval from $0.5\ sec$ to $3.0\ sec$ the difference is visibly a mis-interpolation of the signal (below). This discrepancy

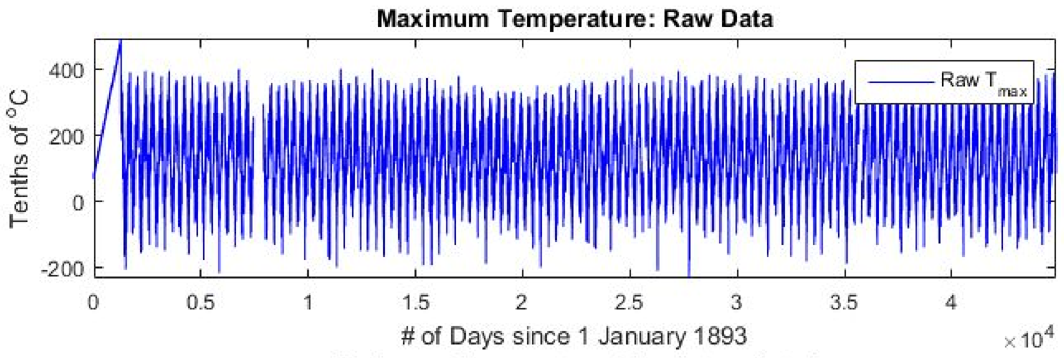


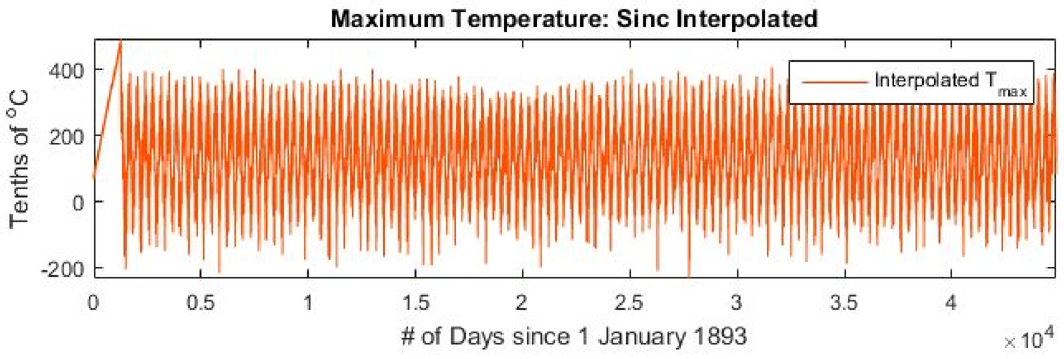
occurs because the sampling frequency is not greater than twice the Nyquist frequency. This means there is not ample information in the discrete time signal to recover the continuous time signal. In terms of the mathematical expression of the sampling theorem, $f_{sampling} < 2 * f_{Nyquist}$ or $15\ Hz < 2 * 10\ Hz$.

3. The following are plots of the raw and sinc interpolated temperature data, min and max, from Lab 6. At this scale it is difficult to to see specific difference besides the filled in gaps present in both the minimum and maximum raw data. Sinc interpolation successfully filled in these gaps of data sith what it calculated to be an accurate interpolation. To do this process I had to use the matrix method of interpolation Jyoti discussed in lab as the method we learned in class was not adequate for the job. I believe this interpolation accurately represents the data and it is a valuable interpolation.

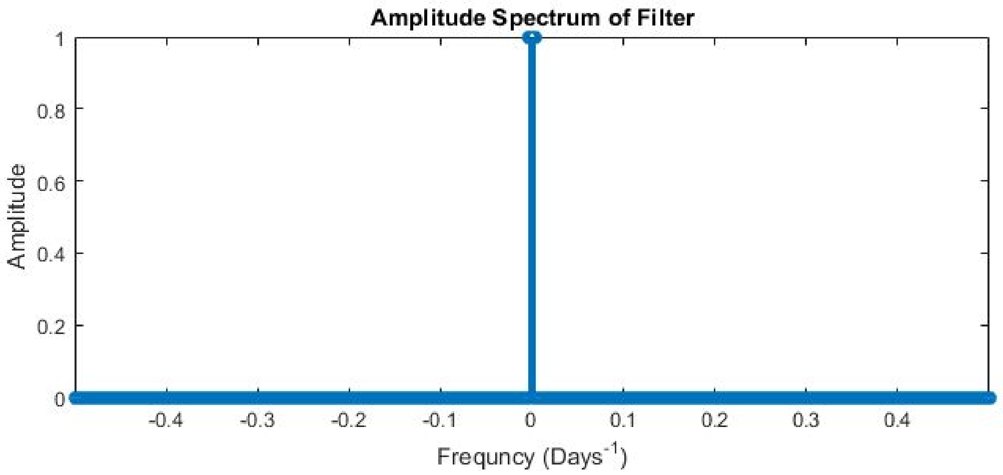




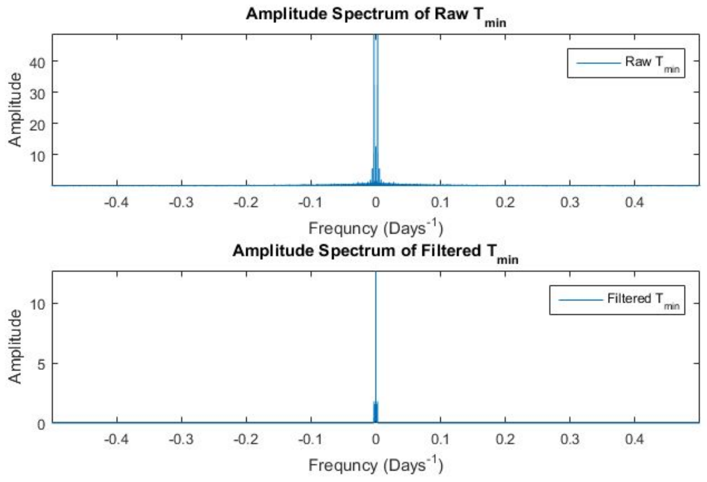


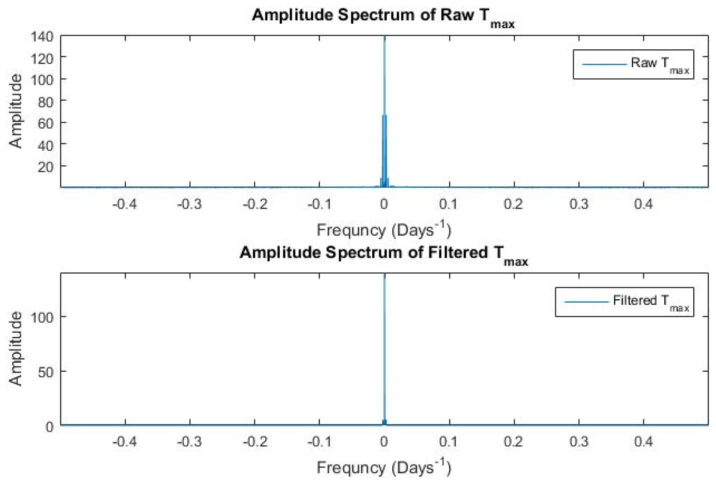


4. The filter I designed zeros frequencies corresponding to yearly variation in temperature, the seasons, by zeroing data with period in between 2 and 380 days.

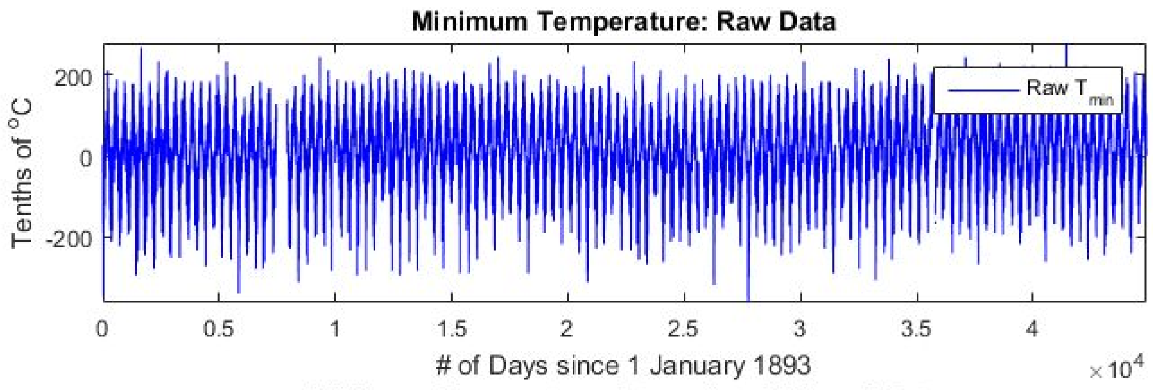


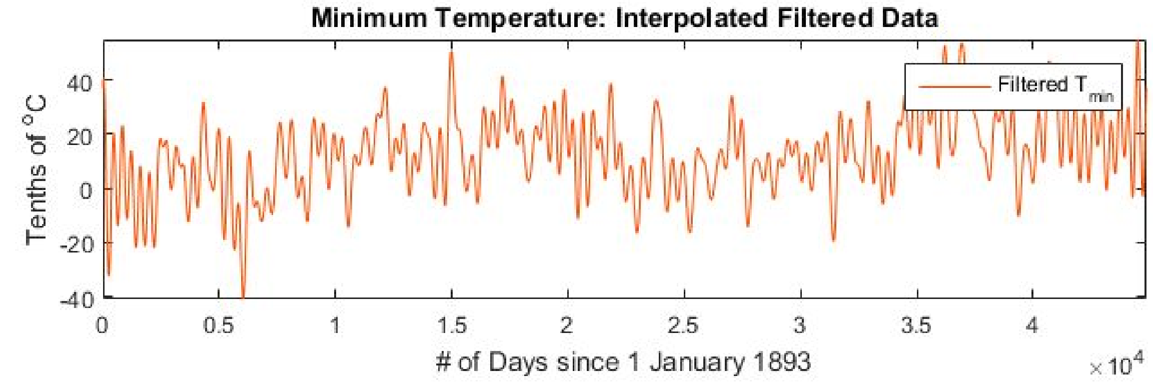
Then next plots are frequency amplitude plots of the fourier transform of the min and max temperature data with and without the filter applied.

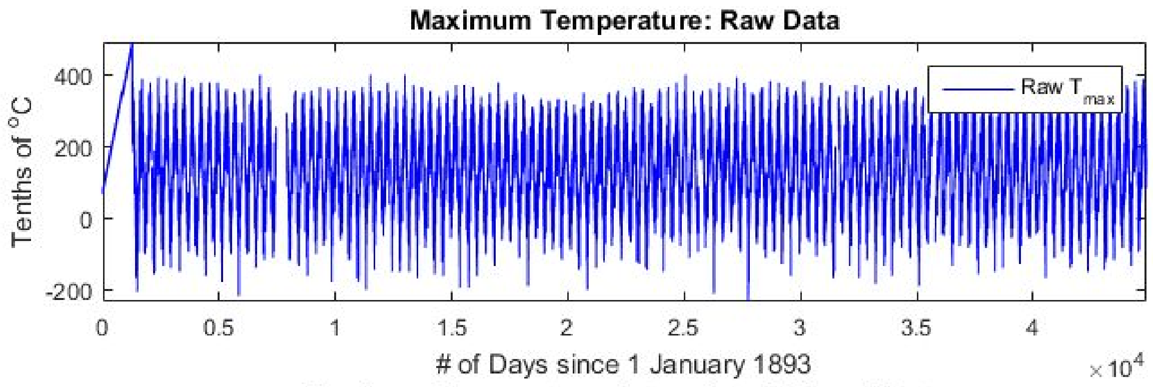


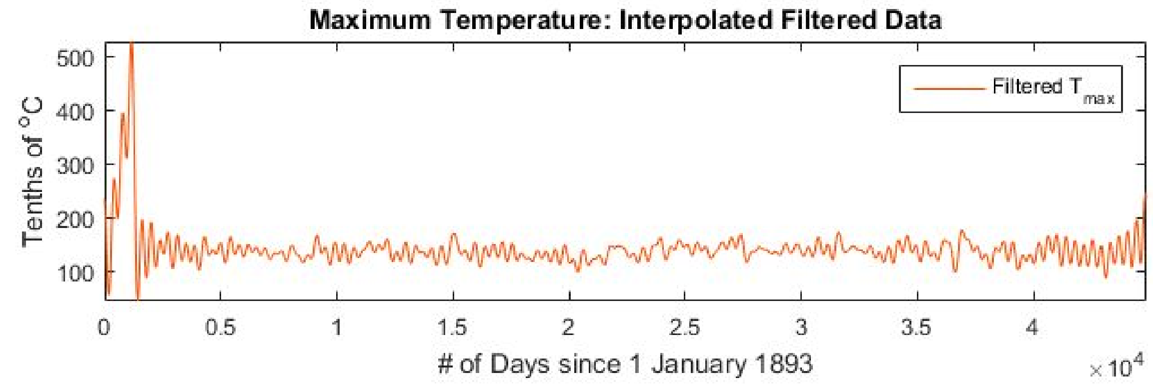


Although small, the differences are clear upon close inspection. The frequency content in the range of a time period of one year has been zeroed out. Next we can apply the inverse fourier transform and get the filtered time data back out.









The filtering clearly has suppressed higher frequency variation and maintained lower frequency seasonal variations.

Appendix I: The Code

```
function Lab08()
    Problem1();
    Problem2();
    Problem3();
end

function Problem1()
    Fs = 25.0;
    Ts = 1/Fs;

    Tmin = 0.0;
    Tmax = 3.0;

    N = ceil((Tmax-Tmin)*Fs);
    n = 1:N;
    ts = (n-1).*Ts;
    tc = Tmin:0.001:Tmax;

    x = (2.0)*cos(3.1415926*(10*ts+1/6))+cos(3.1415926*(20*ts+1/3));
    y = (2.0)*cos(3.1415926*(10*tc+1/6))+cos(3.1415926*(20*tc+1/3));
    z = zeros(length(tc),1);

    for i=1:length(tc)
        z(i) = sum(x .* sinc(tc(i)./Ts-(0:N-1)));
    end

    figure('position', [0, 0, 750, 250]);
    plot(tc, y, 'color', 'b');
    hold on;
    plot(tc, z, 'color', [1,0.3,0]);
    hold on;
    plot(ts, x, 'o', 'color', 'k');
    xlabel('Time (sec)');
    ylabel('Amplitude');
    legend('Original','Reconstructed','Samples Points');
    title('Problem 1: [0.0, 3.0] seconds');
end

function Problem2()
    Fs = 15.0;
    Ts = 1/Fs;

    Tmin = 0.0;
    Tmax = 3.0;

    N = ceil((Tmax-Tmin)*Fs);
    n = 1:N;
    ts = (n-1).*Ts;
    tc = Tmin:0.001:Tmax;

    x = (2.0)*cos(3.1415926*(10*ts+1/6))+cos(3.1415926*(20*ts+1/3));
    y = (2.0)*cos(3.1415926*(10*tc+1/6))+cos(3.1415926*(20*tc+1/3));
    z = zeros(length(tc),1);
    for i=1:length(tc)
        z(i) = sum(x .* sinc(tc(i)./Ts-(0:N-1)));
    end

    figure('position', [0, 0, 750, 250]);
    plot(tc, y, 'color', 'b');
    hold on;
    plot(tc, z, 'color', [1,0.3,0]);
    hold on;
    plot(ts, x, 'o', 'color', 'k');
    xlabel('Time (sec)');
    ylabel('Amplitude');
    legend('Original','Reconstructed','Samples Points');
    title('Problem 2: [0.0, 3.0] seconds');
end

function Problem3()
    % Load the data
    filename = 'Lab6_t_T.csv';
    rawData = importdata(filename);
    rawData = rawData.data;
    rawDates = rawData(:, 1);
    rawTmax = rawData(:, 2);
    rawTmin = rawData(:, 3);
    invalid = -9999;

    % Fill in the missing days
    serialDates = datenum(num2str(rawDates), 'yyyymmdd');
    serialDates = serialDates - serialDates(1);
    dates = (serialDates(1):1:serialDates(length(serialDates))) - serialDates(1);

    Tmin = fixData(serialDates, rawTmin, invalid);
    Tmax = fixData(serialDates, rawTmax, invalid);

    % Interpolate the data
    Ts_interp = 365;
    dates_interp = dates(1):Ts_interp:dates(length(dates));
    disp('Dates Outputted');
    interpTmin = sincInterp(dates, Tmin, dates_interp, Ts_interp, 1e-1);
    disp('Tmin Outputted');
    interpTmax = sincInterp(dates, Tmax, dates_interp, Ts_interp, 1e-1);
    disp('Tmax Outputted');

    figure('position', [0, 0, 750, 450]);
    subplot(2,1,1);
    plot(dates, Tmin, 'color', 'b');
    axis tight;
    title('Minimum Temperature: Raw Data');
    xlabel('# of Days since 1 January 1893');
    ylabel('Tenths of ^{o}C');
    legend('Raw T_{min}');
    subplot(2,1,2);
    plot(dates_interp, interpTmin, 'color', [1,0.3,0]);
    axis tight;
    title('Minimum Temperature: Sinc Interpolated');
```

```
xlabel('# of Days since 1 January 1893');
ylabel('Tenths of ^{o}C');
legend('Interpolated T_{min}');

figure('position', [0, 0, 750, 450]);
subplot(2,1,1);
plot(dates, Tmax, 'color', 'b');
axis tight;
title('Maximum Temperature: Raw Data');
xlabel('# of Days since 1 January 1893');
ylabel('Tenths of ^{o}C');
legend('Raw T_{max}');
subplot(2,1,2);
plot(dates_interp, interpTmax, 'color', [1,0.3,0]);
axis tight;
title('Maximum Temperature: Sinc Interpolated');
xlabel('# of Days since 1 January 1893');
ylabel('Tenths of ^{o}C');
legend('Interpolated T_{max}');

Tmin_Xk = fftshift(fft(interpTmin))/length(Tmin);
Tmax_Xk = fftshift(fft(interpTmax))/length(Tmax);
Fk = getFrequencies(dates);
Ff = 1/(380.00);
Fl = 1/(2);
fTmin_Xk = Tmin_Xk;
fTmax_Xk = Tmax_Xk;
filter = length(Fk);
for i=1:length(Fk)
    if abs(Fk(i)) > Ff && abs(Fk(i)) < Fl
        fTmin_Xk(i) = 0.0;
        fTmax_Xk(i) = 0.0;
        filter(i) = 0.0;
    else
        filter(i) = 1.0;
    end
end
invTmin = ifft(ifftshift(fTmin_Xk)).*N;
invTmax = ifft(ifftshift(fTmax_Xk)).*N;

figure('position', [0, 0, 750, 300]);
stem(Fk, filter);
axis tight;
xlabel('Frequncy (Days^{-1})');
ylabel('Amplitude');
title('Amplitude Spectrum of Filter');

figure('position', [0, 0, 750, 450]);
subplot(2,1,1);
plot(Fk, abs(Tmin_Xk));
axis tight;
xlabel('Frequncy (Days^{-1})');
ylabel('Amplitude');
title('Amplitude Spectrum of Raw T_{min}');
legend('Raw T_{min}');
subplot(2,1,2);
```

```
    plot(Fk, abs(fTmin_Xk));
    axis tight;
    xlabel('Frequncy (Days^{-1})');
    ylabel('Amplitude');
    title('Amplitude Spectrum of Filtered T_{min}');
    legend('Filtered T_{min}');

    figure('position', [0, 0, 750, 450]);
    subplot(2,1,1);
    plot(Fk, abs(Tmax_Xk));
    axis tight;
    xlabel('Frequncy (Days^{-1})');
    ylabel('Amplitude');
    title('Amplitude Spectrum of Raw T_{max}');
    legend('Raw T_{max}');
    subplot(2,1,2);
    plot(Fk, abs(fTmax_Xk));
    axis tight;
    xlabel('Frequncy (Days^{-1})');
    ylabel('Amplitude');
    title('Amplitude Spectrum of Filtered T_{max}');
    legend('Filtered T_{max}');

    figure('position', [0, 0, 750, 450]);
    subplot(2,1,1);
    plot(dates, Tmin, 'color', 'b');
    axis tight;
    title('Minimum Temperature: Raw Data');
    xlabel('# of Days since 1 January 1893');
    ylabel('Tenths of ^{o}C');
    legend('Raw T_{min}');
    subplot(2,1,2);
    plot(dates_interp, invTmin, 'color', [1,0.3,0]);
    axis tight;
    title('Minimum Temperature: Interpolated Filtered Data');
    xlabel('# of Days since 1 January 1893');
    ylabel('Tenths of ^{o}C');
    legend('Filtered T_{min}');

    figure('position', [0, 0, 750, 450]);
    subplot(2,1,1);
    plot(dates, Tmax, 'color', 'b');
    axis tight;
    title('Maximum Temperature: Raw Data');
    xlabel('# of Days since 1 January 1893');
    ylabel('Tenths of ^{o}C');
    legend('Raw T_{max}');
    subplot(2,1,2);
    plot(dates_interp, invTmax, 'color', [1,0.3,0]);
    axis tight;
    title('Maximum Temperature: Interpolated Filtered Data');
    xlabel('# of Days since 1 January 1893');
    ylabel('Tenths of ^{o}C');
    legend('Filtered T_{max}');
end
```

```
% Fill in missing data
function [result] = fixData(input, values, invalid)
    result(1) = values(1);
    for i=2:length(input)
        d = input(i) - input(i-1);
        if d > 1
           result =  vertcat(result, nan(d-1, 1));
        end
        result = vertcat(result, values(i));
    end
    result(isnan(result)) = [];
    result(result == invalid) = [];
end

% Sinc Interpolate the data, by Jyoti Behura
function [result] = sincInterp(input, values, output, Ts, tolerance)
    ni = length(input);
    no = length(output);
    A = zeros(ni,no);
    for i=0:ni-1
        for j=0:no-1
            A(i+1,j+1) = sinc((input(i+1)-output(j+1))/Ts);
        end
    end
    result = pinv(A, tolerance)*values;
end

% Produce a list of frequencies given a list of evenly spaced times
function [result] = getFrequencies(times)
    N = length(times);
    Ts = (times(N) - times(1)) / (N - 1);
    dF = 1 / (N * Ts);
    result = ((0:N-1) - ceil(N/2))*dF;
end
```