

CIEG 675

Homework #4 Due **Wednesday March 18, 2009**

In an m-file do the following and verify it works by copy and pasting into the command window or running your m-file.

- 1) Write a function that will take an integer, n, as input and then output the Fibonacci sequence out to that integer n. It will also plot the sequence as a function of its location in the sequence.**

```
funtion [fibout] = fibonacci(n)
% funtion fibout = fibonacci(n)
%
% function will make fibonacci sequence out to an index of n
%
%input:
% n the index of the sequence
%
%output:
% fibout, the actual sequence

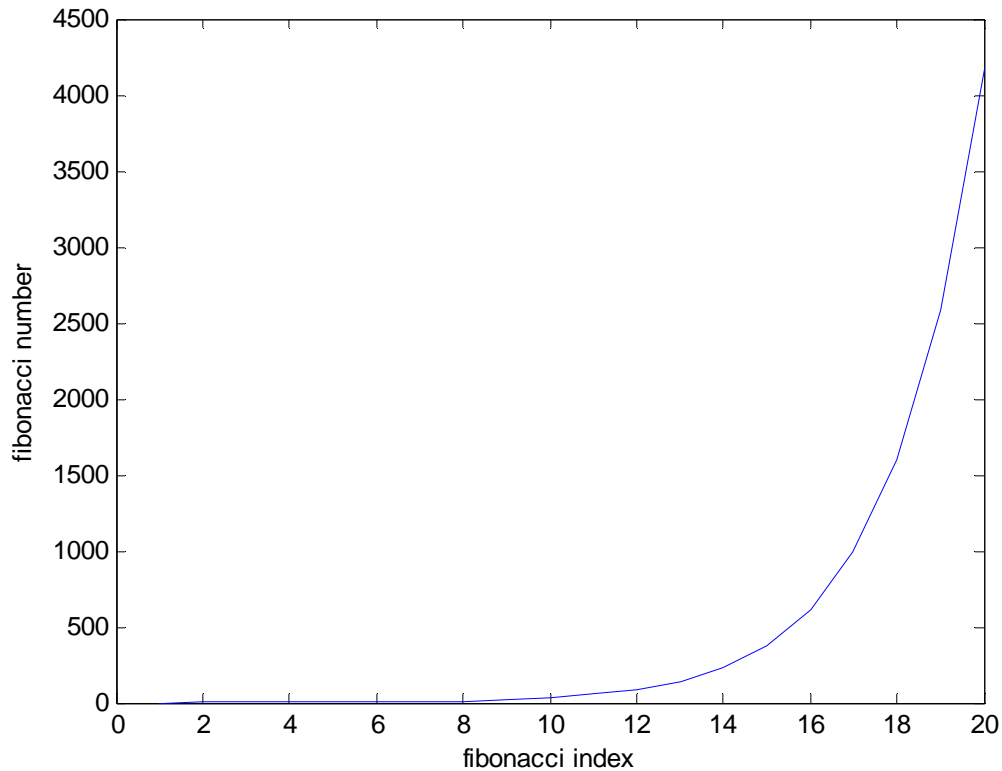
fibout=zeros(n,1); % pre-allocate for correct length

% note the 1st value in the sequence is zero

fibout(2)=1; % second one is always 1

for i=3:n;
    fibout(i)=fibout(i-1)+fibout(i-2);
end

plot(n,fibout)
xlabel('fibonacci index');
ylabel('fibonacci number');
```



- 2) Write a function that will take an arbitrary, but smooth, vector of data and location ALL the local maxima and minima in the vector. It should return the indices in the original vector of where these maxima and minima occur. Test your function on the following data set that you should make quite smooth by using small increments in x, where x should go from 0 to 10;

$$y = x^{1.01} + 4 \cos(3\pi x / 4) - 2 \sin(2\pi x / 3) - 0.25 .$$

Plot your function and then plot the maxima and minima on top as different symbols.

```
function [peaks troughs]=extrema(vec);
%function [peaks troughs]=extrema(vec);
%
% function will find all local maxima and minima and return indices
%
% input:
% the vector of data to be operated on
%
```

```

% output:
% peaks, the indices of the maxima
% troughs, the indices of the minima

% easy way without loop
l=length(vec); % vector length

i=2:l-1; % central indices
ibef=1:l-2; % indices before
iaft=3:l; % indices after

% this approach shifts indices and looks for those times where vec(i)
% greater than vec(ibef) and vec(iaft). Similar for mins

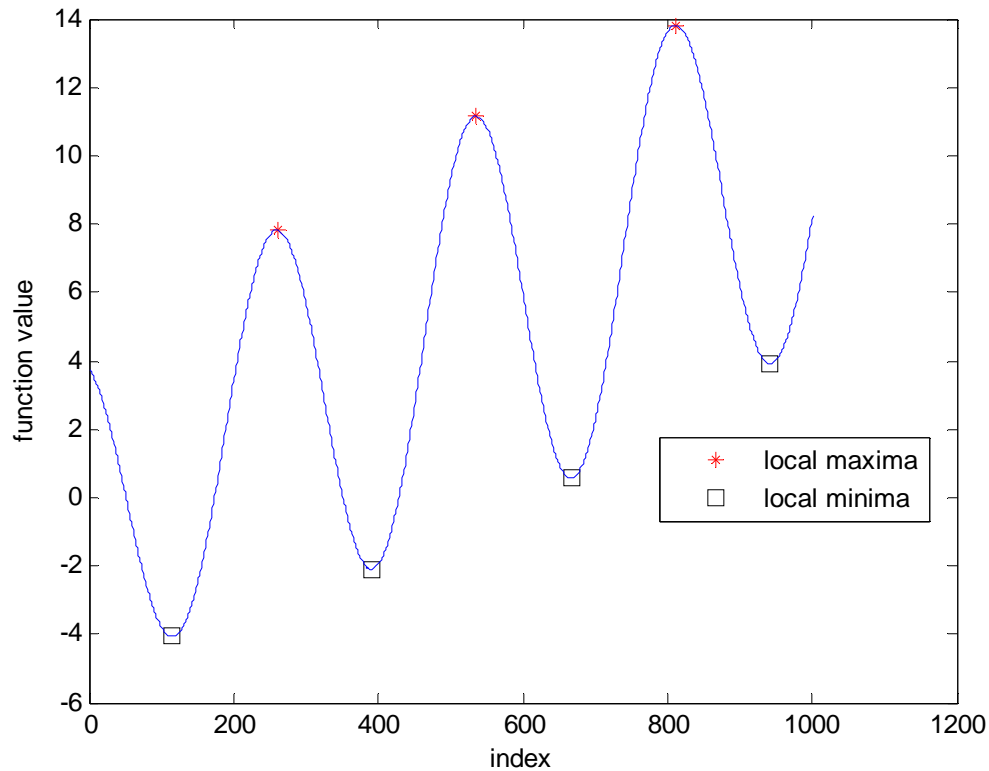
% find peaks
peaks=find(vec(i)>vec(ibef) & vec(i)>vec(iaft)); % with respect to index i
peaks=i(peaks); % so must redefine with respect to i

% find troughs
troughs=find(vec(i)<vec(ibef) & vec(i)<vec(iaft)); % with respect to index i
troughs=i(troughs); % so must redefine with respect to i

clf
plot(peaks,vec(peaks),'r*');
hold on
plot(troughs,vec(troughs),'ksquare');
legend('local maxima','local minima');
plot(vec);

xlabel('index');
ylabel('function value');

```



3) In coastal engineering an equation exists for linear wave theory to predict the wavelength (distance between 2 wave crests) as a function of wave period and water depth. Unfortunately the wave length exists on both sides of the equation. Write a function to determine the wave length for a specific wave period and water depth as input. It should output the wave length. The equation is

$$L = \frac{gT^2}{2\pi} \tanh\left(\frac{2\pi h}{L}\right),$$

where L is wavelength in meters, g is gravitational acceleration,

h is water depth in meters and \tanh is the hyperbolic tangent and is a built in function in matlab. As a first guess, use what is called the deep water value where the \tanh term goes to 1. Check your code for a 10s wave in 20m depth. $L= 121.2$ m.

```
function L=wavelength(T,h);
%function L=wavelength(T,h);
%
% find the wave length for given wave period and water depth
% input:
% T, wave period in seconds
% h, water depth in meters
%
% output:
```

```

% L, wavelength in meters

g=9.81; % m/s^2
errtol=0.1; % error tolerance. We want wave length to nearest 10 cm.
err=10; % set initial error high so we enter loop
count=0; % a counter
L=g*T^2/(2*pi); % first guess is deep water.

while err>errtol & count<100
    Lguess=g*T^2/(2*pi) * tanh(2*pi*h/L); % the guessed wavelength
    err=abs(L-Lguess); % check error
    L=Lguess; % reset L
end % end while loop

if count==100
    disp('something went awry'); % we will learn about text and strings later.
end % if loop

```

- 4) Test your function from problem 3 by making a plot of wave length as a function of water depth from 100m depth to 10 m depth for a 10 s wave. Test your function from problem 3 by making a plot of wave length as a function of wave period from 1 to 30 s for waves in 40 m deep water. Use 2 subplots**

```

h=100:-1:10;
T=10;
for k=1:length(h);
    L(k)=wavelength(T,h(k));
end

```

```

subplot(211)
plot(h,L);
xlabel('water depth (m)');
ylabel('wavelength (m)');

```

```

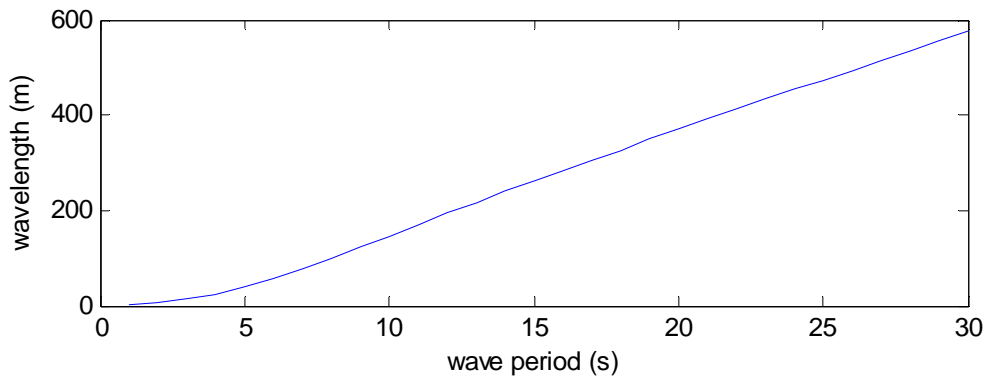
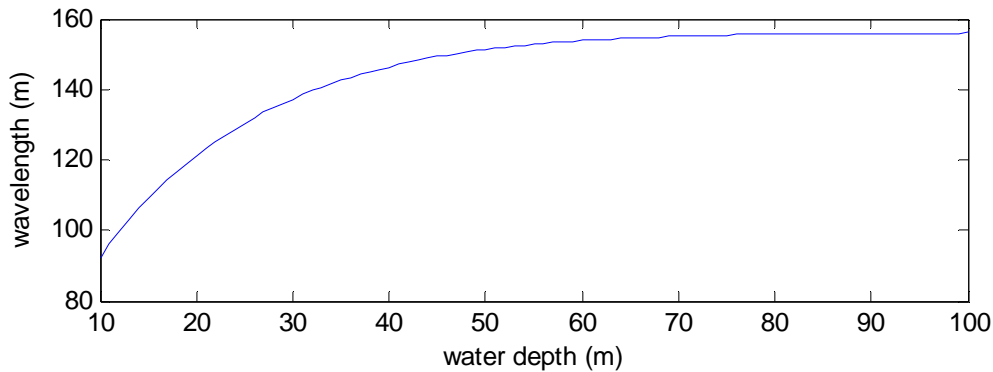
clear L
h=40;
T=1:30;
for k=1:length(T);
    L(k)=wavelength(T(k),h);
end

```

```

subplot(212)
plot(T,L);
xlabel('wave period (s)');
ylabel('wavelength (m)');

```



5) You are asked to write a simple function to parse out grades for your teacher's class. The code should divide the grades based on a standard scale as $\text{score} \geq 90$ is an A, $80 \leq \text{score} < 90$ is a B, $70 \leq \text{score} < 80$ is a C, $60 \leq \text{score} < 70$ is a D and anything below is failing. For now, just return the indices of the A's, B's, C's and D's. Later we can use something called a structure or come up with a way to use strings to actually attach a letter grade to that persons file. For your input scores use a vector made by `scores = 100*rand(100,1)` assuming 100 students. *Write your function two ways: 1 using if statements and the other using the find command.* **WARNING** this slow way with if statements will be a tricky since you do not know in advance how long each vector of indices will be a priori

```
scores=100*rand(100,1); % false grade vector
```

```
function [A,B,C,D,F]=grades(scores)
%function [A,B,C,D,F]=grades(scores)
% determien index of A's B's etc based on standard grade scale
%
% input:
% vec, vector of grades
%
% output:
```

```
% index of A,B, C, D, F
```

```
%% slow way
```

```
ctA=0;  
ctB=0;  
ctC=0;  
ctD=0;  
ctF=0; %% counter for indexing A,B,C,D,F indices
```

```
for i=1:length(scores);  
    if scores(i)>=90;  
        ctA=ctA+1;  
        A(ctA)=i;  
  
    elseif scores(i)<90 & scores(i)>=80  
        ctB=ctB+1;  
        B(ctB)=i;  
  
    elseif scores(i)<80 & scores(i)>=70  
        ctC=ctC+1;  
        C(ctC)=i;  
  
    elseif scores(i)<70 & scores(i)>=60  
        ctD=ctD+1;  
        D(ctD)=i;  
  
    else  
        ctF=ctF+1;  
        F(ctF)=i;  
    end % if loop
```

```
end % for loop
```

```
%% fast way
```

```
IA=find(scores>=90);  
IB=find(scores>=80 & scores<90);  
IC=find(scores>=70 & scores<80);  
ID=find(scores>=60 & scores<70);  
IF=find(scores<60);
```

```
%% a quick check have it spit to screen  
sum(IA-A') % bette be zero since they should be the same
```

sum(1B-B')
sum(1C-C')
sum(1D-D')
sum(1F-F')