MAE 8 - Winter 2015
Homework 4

Instructions: Follow the homework solution template. Put all answers in a MATLAB script named **hw4.m**. For this homework, you will need to submit multiple files. Create a zip archive named **hw4.zip**. The zip archive should include the following files: **hw4.m**, **figure1.png**, **divisibility.m**, **piecewise2d.m**, and **polygon.m**. Submit **hw4.zip** through TED before 9 PM on 02/05/2015. Use double precision unless otherwise stated.

Problem 1: Write a function **divisibility.m** that checks the divisibility of a given number by 3, 5, 7, and 11. The function should have the following declaration: **function div = divisibility(a)** where **a** is a number input and **div** is a logical vector output. Vector **div** should have 4 elements corresponding to the divisibility by 3, 5, 7 and 11, respectively. For example, the output of **divisibility(14)** would be [**0 0 1 0**]. The function should also include a description. Use **if** statement when necessary.

(a) Set **p2a=evalc('help divisibility')**.
(b) Check the divisibility of 33 and set the answer to **p1b**.
(c) Check the divisibility of 385 and set the answer to **p1c**.
(d) Check the divisibility of 3855 and set the answer to **p1d**.
(e) Check the divisibility of 232155 and set the answer to **p1e**.

Problem 2: Given the following piecewise function:

$$g(x) = \begin{cases} x^{-2} & x < -1 \\ x^2 + 2 \quad for & -1 \leq x \leq 1 \\ x^{-2} & x > 1 \end{cases}$$

Use **for** loop and **if** statement to create g(x).

(a) Evaluate the function g(x) for x=[-4:0.02:4] and put the answer in vector **p2a**.
(b) Compute the derivative g'(x) and put the answer in vector **p2b**.

Problem 3: Italian mathematician Fibonacci is famous for introducing the 'Fibonacci series' to modern mathematics. Any term in the Fibonacci series is the sum of the previous two terms. For example, the first 5 terms of the series are

$$1, 1, 2, 3, 5$$

Use **for** loop to explore the series.

(a) Compute the first 39 terms of the series and put the answer in vector **p3a**.
(b) What is the sum of all terms in the series in part (a) ? Put the answer in **p3b**.
(c) Find the ratio of two consecutive terms in the series in part (a) and put the answer in vector **p3c**. Hint: set p3c(1) = 0, p3c(2)=p3a(2)/p3a(1) and so on ...

Problem 4: Use nested **for** loops and **if** statement to perform the following for **A=[-2.45:0.05:2.5], A=reshape(A,10,10), B=[0.01:0.01:1], B=reshape(B,10,10) and d=[0.1:0.1:1]'**.

(a) Matrix-vector multiplication **A*d** and put the result in **p4a**.

(b) Matrix-matrix multiplication **A*B** and put the result in **p4b**.

(c) Create a matrix which contains 1 for **B** values that are greater than their corresponding values in **A**, and 0 otherwise. Put the result in **p4c**.

(d) Create a matrix which contains 1 for positive elements of **A**, -1 for negative elements of **A** and 0 for elements of **A** which are zero. Put the result in **p4d**.

Problem 5: Write a function **piecewise2d.m** to compute the following 2-dimensional function f(x,y):

$$f(x,y) = \begin{cases} x+y & for\, x \geq 0 \quad y > 0 \\ -x+y & for\, x < 0 \quad y \geq 0 \\ -x-y & for\, x \leq 0 \quad y < 0 \\ x-y & for\, x > 0 \quad y \leq 0 \end{cases}$$

Function **piecewise2d.m** should have the function declaration: **function f = piecewise2d(x,y)** where **x** and **y** are vector inputs and **f** is a matrix output. Use nested **for** loops and **if** statement when necessary. Remember to include a description for the function.

(a) Set **p5a=evalc('help piecewise2d')**.

(b) Set **p5b=piecewise2d([0:10],[0])**.

(c) Set **p5c=piecewise2d([0],[0:10])**.

(d) Set **p5d=piecewise2d([-10:0],[0])**.

(e) Set **p5e=piecewise2d([0],[-10:0])**.

(f) Set **p5f=piecewise2d([-10:4:10],[-10:4:10])**.

Problem 6: The circumference of a circle is computed as $2\pi r$ where **r** is the radius. The

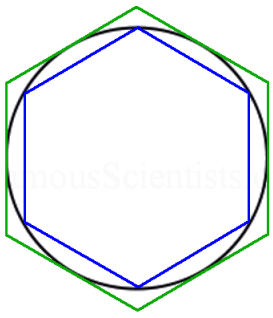

Figure 1: Circumscribing (green) and inscribing (blue) regular polygons for a circle

circumference can also be approximated by the perimeter of an n-sided regular polygon (blue

in figure 1) inscribing the circle by the formula:

$$P_i = 2nr\, sin\,(\pi/n)\,,$$

where P_i is the perimeter of the polygon inside the circle. The circumference can further be approximated by the perimeter of an n-sided regular polygon (green in figure 1) circumscribing the circle by the formula:

$$P_o = 2nr\, tan\,(\pi/n)\,,$$

where P_o is the perimeter of the polygon outside the circle.

In the following exercises, we will compare the accuracy of the two approximation methods. Write function **polygon.m** with the following function declaration: **function [perimeter error] = polygon(r,n,method)** where **r** is the input radius, **n** is the number of sides of the polygon, and **method** is a string input (either 'inscribe' or 'circumscribe'). The function gives two outputs: **perimeter** and **error** associated with the chosen method. Here, **error** is defined as the ratio of the approximated perimeter of the polygon to the exact circumference of the circle ($2\pi r$). If the input method is not one of the two methods above, output **perimeter=8888** and **error=8888**.

Note that the methods used here are only valid when $n \geq 3$. If user inputs $n < 3$, output **perimeter=9999** and **error=9999**. In the following exercises, let the radius **r = 1**. Remember to give the function a description.

(a) Set **p6a=evalc('help polygon')**.

(b,c) Let **n=2**. Set **[p6b p6c] = polygon(r,n,'inscribe')**.

(d,e) Now let **n=10**. Set **[p6d p6e] = polygon(r,n,'circle')**

(f,g) Now let a vector **vecn=[5:5:100]**. Use **for** loop to compute the perimeters and errors for each element of **vecn** with method **'inscribe'**. Put the perimeters in vector **p6f** and the errors in vector **p6g**.

(h,i) Use **for** loop to compute the perimeters and errors for each element of **vecn** with method **'circumscribe'**. Put the perimeters in vector **p6h** and the errors in vector **p6i**.

(j) Make a figure to compare the error in part (g) to the error in part (i) versus **vecn**. Label the axes and give the figure a title and a legend. Save the figure as **figure1.png**. Set **p6j='See figure 1'**.

(k) For the same number of sides, which method gives a better approximation of the circumference ? Give your answer in the format: **p6k='Method ... is better'**.