Slide 1 – 2 introduction

Slide 3 : We put an xy axis on the map and we start at point (0,0). We see the CN tower looks like it is about NE from us or 45o. We will walk East down the street and guess the direction of the CN tower from other positions to make a system of equations and solve for the location of the CN tower relative to our start position.

Slide4 : Tank battle game I made in MATLAB. You can play it by putting tankbattle.m file in your Mydocuments/MATLAB folder, then typing tankbattle in the MATLAB command line.

Slide5 : MONTE CARLO Jellyfish graphic I made in MATLAB. You can play it by putting jellyguy.m and fastthreedrandomize.m file in your Mydocuments/MATLAB folder, then typing jellyguy in the MATLAB command line. Try different number of tentacles from 1-100 and different lengths from 10 – 50 cm. very high numbers of either may crash the program.

The other picture shows a much more complicated Monte Carlo simulation of neutron scattering and fission in a reactor core that I made. I will show that later.

Slide6: Matrices

Slide7: We are going to construct a system of equations from our CN tower example. Since we have three points of measurement we have 3 linear equations. We cannot use an invertible matrix because the system is over constrained. Also our angles are just best guesses and not perfect measurements so we will try to find the “Best solution” to the system of equations.

Slide 8: On the map we guess the angle at our start position. We walk due east 50m and guess the angle again (relative tour our East-West x axis). We walk another 50m and guess the angle again.

Slide 9: The angles of the CN tower we guessed were 45o , 49o and 51o from our three positions. We will now put that into our code (Healthphysics1.m) as angle1 angle2 and angle3.

Slide 10: The next part of the MATLAB code finds the slope m1 m2 m3 of the linear equations from each position. We also find the y-intercepts for the equations b1 b2 b3. Then We make a matrix equation from our linear equations of the form A\****x*** = ***b*** where A is the matrix containing the slopes, ***x*** is the vector for the unknown CN tower coordinates x,y and b is the vector for the y-intercepts.

This is done in the code. First we solve the invertible matrix using only equations 1 and 2 with matrix A. This is called posguess. Then we use MATLAB’s \ matrix “division” operation and all three of our equations. In this case MATLAB finds the “best” solution to the over constrained system of equations (by using the least squares method).

Slide 10: Shows our MATLAB solution on the map

Slide 11: Look at the tank battle game again. We will make a simple version Healthphysics2.m

The code begins with plotting a sin curve for hills. Then we plot the tank points.

Next our tank will fire. We take an input of our firing velocity. We will fire at 45o for simplicity.

Next the code uses equations for projectile motion to find the parabola path of our tank’s bullet. We could plot that parabola directly with the plot() command. However the final loop plots the path in steps so we can see the bullet fly (just for fun).

Try to see if you can hit the tank directly. HINT: it is 360m away at the same elevation. Use projectile equations to solve for the ideal velocity.

Slide 12-13: Now I will discuss random numbers. Type rand into the command prompt to get a random number from 0 – 1. You can make a random number from 0 -6 by using 6\*rand.

Now try to simulate a 6 sided dice. You can round numbers with the ceil() function. This will give numbers from 1 - 6.

Now open Healthphysics3.m

This code uses random numbers to guess the value of π.

First we choose a random x and y point between -1 and 1. This is in a square of area 4. Then we draw a circle inside the square with radius 1. The circle will have an area of pi and the square has an area of 4. So each time we make a random x,y point we check to see if it is in the circle. If it is we count it. In the end the ratio of those in the circle to the total should be pi/4. So if we multiply our #counts/totalpoints by 4 we will get an estimate for the value of pi. Note you can increase the number of points “iterations” to get a better guess of pi but it takes longer to compute.

Slides 14-16 will be used in the next Workshop. This is the end. Thanks for coming