CS 322 Homework 3

out:	Thursday 22 February 2007
due:	**Wednesday 28 February 2007**

This homework includes both a written and MATLAB component, and involves floating point representations and error visualization. Your MATLAB code will be turned in via CMS, while your writeup should be turned in at the beginning of class on the due date.

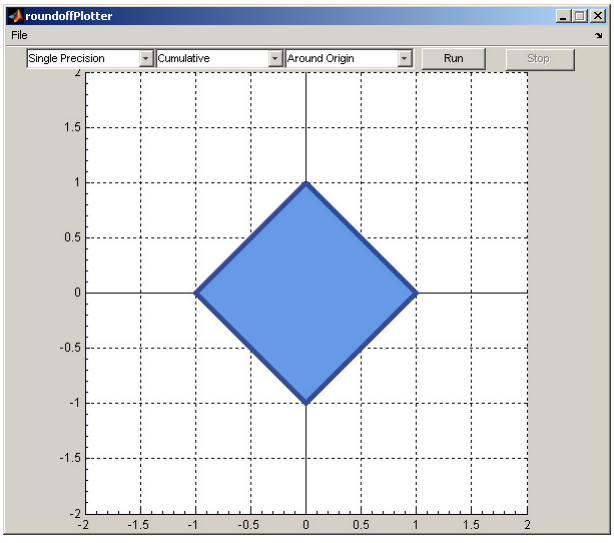
Part A - Half-Precision Arithmetic

Recall that half-precision floats are like IEEE 754 standard floats except that they fit in 16 bits, with 1 sign bit, 5 exponent bits, and 10 mantissa bits.

Problem 1 What is the half-precision representation of the number 3? Of -27.5 in half-precision?

Problem 2 What is the value of epsilon for $\frac{1}{10}$ in half-precision? For 1000.1 in half-precision?

Part B - Error Visualization



The framework for this part of the assignment is a MATLAB function that displays a square along with three drop-down boxes and explores the error caused by rotating the square in a variety of different ways. The GUI can be started by calling the function `roundoffPlotter()`, which can

take a single optional argument (which will be discussed below). When the "Run" button is clicked, the square will be slowly rotated around the origin based on the options that are specified in the drop-down box, where each step is the result of rotating the square by one degree more than the previous step. When a maximum number of steps has been reached, or the "Stop" button is clicked, the square will stop spinning and the function `plotError` will be called. `plotError` takes as an argument a matrix of $8 \times n$, where n is the number of steps taken while rotating. This matrix looks like

$$\begin{bmatrix} x_{1,0} & x_{1,1} & \cdots & x_{1,n} \\ y_{1,0} & y_{1,1} & \cdots & y_{1,n} \\ x_{2,0} & x_{2,1} & \cdots & x_{2,n} \\ y_{2,0} & y_{2,1} & \cdots & y_{2,n} \\ x_{3,0} & x_{3,1} & \cdots & x_{3,n} \\ y_{3,0} & y_{3,1} & \cdots & y_{3,n} \\ x_{4,0} & x_{4,1} & \cdots & x_{4,n} \\ y_{4,0} & y_{4,1} & \cdots & y_{4,n} \end{bmatrix}$$

where $(x_{i,j}, y_{i,j})$ is the position of the i-th point at step j. The other two functions are `rotatePoints` and `roundoff_painter`. You do not need to modify them, but you may want to look at them to see what they are doing. `rotatePoints` takes in a matrix of points, an amount to rotate by in radians, a precision argument, and the location that the square should be rotated around. `roundoff_painter` is used by the GUI to update the screen after each rotation, store the history of all points, and eventually call `plotError` when enough steps have been completed.

Each drop down box has two values:

1. Precision, which can be set to either "Single" or "Double". Controls whether single or double-precision floating point numbers are used in computation.
2. Method, which can be set to "Cumulative" or "Exact". In "Cumulative" mode, the result of the previous step is rotated by a small amount (a single degree) to get the result of the next step. In "Exact" mode, the original positions of the square are rotated by the total amount needed for that step. For instance, on the fourth step "Cumulative" will take the result of the previous step and rotate by 1 degree, while "Exact" will take the original square and rotate by 4 degrees.
3. Location, which can either be "Origin" or "Centered at (loc, loc)", where loc is the optional argument to `roundoffPlotter(loc)`. For instance, to center around $(1e6, 1e6)$, `roundoffPlotter(1e6)` should be called. When "Origin" is selected, the square is rotated around the origin. When "Centered at (loc, loc)" is selected, the square is moved to location (loc, loc) and a transformation matrix is constructed which performs the specified rotation *around* (loc, loc). If no argument is passed in, $loc = 1e4$.

Problem 1 Implement `plotError` so that it generates some interesting visualizations of the error in the computation. *hint:* One thing to recall is that rotations preserve both lengths and angles.

Problem 2 Turn in with your writeup several printouts of visualizations produced by your function for a variety of parameters that display something interesting or noteworthy. Discuss potential

sources of any errors you are seeing.

Problem 3 For certain large values of the parameter `loc`, the single precision computation will cease working entirely. Why? What happens to the floating point computation?

Problem 4 Predict how much error you would expect to see for ”(Cumulative, centered at origin)” if there were an option for half-precision.