A Monte Carlo Method
Due Tuesday Sept. 17 by 9:30 AM

Consider the problem of finding the measure of the portion of the interval [0, 10] where $\sin(x^{1.1}) \geq .2$. Would you first try to invert the sine to get some intervals where $x^{1.1} \geq \arcsin(.2)$? What if the situation were not as easily invertible such as finding the measure of the portion of the interval [0, 10] where $x \cdot \sin(x^{1.1}) \geq .2$?

A totally different way of approaching this is to estimate the fraction of points on the interval [0, 10] that satisfy $\sin(x^{1.1}) \geq .2$ and then multiplying that fraction by the length of the interval (i.e., 10). In the Monte Carlo Method, we estimate the fraction by testing random values on the interval and computing the ratio of those passing the test $\sin(x^{1.1}) \geq .2$ to the total. Thus if you tested 1000 points uniformly randomly distributed on [0, 10] and found that 466 satisfied $\sin(x^{1.1}) \geq .2$, then you would estimate the fraction as 466/1000 and then the measure as ten times this or 4.66.

Generalize this now to using any function f (of a single variable), any threshold t, and any interval $[a,b]$. We seek to find the measure of the interval $[a,b]$ of those values of x such that $f(x) \geq t$ using *n* uniformly distributed random deviates on $[a,b]$.

Write a Matlab function *MCmeasure* to these specifications:

Input:

	f	the name of a real-valued function defined on $[a,b]$,
	t	a real threshold value,
	a	a real value to be the lower limit of an interval,
	b	a real value to be the upper limit of an interval, and
	n	the number of random values to use.

Output:

	measure	the measure of the interval $[a,b]$ of those values of x such that $f(x) \geq t$.

Notes:
The use of the name of a function as a parameter is quite common in scientific computation although rarer in other areas of computing. We need to have a procedure that works on any function – we cannot rewrite our procedure with the name of the particular function. This applies not just for integration but also in the solution of differential equations, the finding of roots of equations, and the location of maxima and minima.

Matlab allows three different ways for function names to be passed as parameters. In fact, two of these don't even have the function names appearing explicitly. On pages 134-136 of Recktenwald, you can read about one of these ways and use of the *feval* function to do the function evaluation.

The three ways are exemplified by:

a. A string expression involving a single variable:

```
syms x
m = MCmeasure('1/(x^3-2*x-5)', -.5, 0, 2, 10000);
```

b. An inline object

```
F = inline('1/(x^3-2*x-5)');
m = MCmeasure (F, -.5, 0, 2, 10000);
```

c. A function handle

```
m = MCmeasure (@myfun, -.5, 0, 2, 10000);
```

where *myfun.m* is an M-file.

```
function y = myfun(x)
y = 1/(x^3-2*x-5);
```

You will want to have something like

```
fun = fcnchk (f);
```

near the top of your function *MCmeasure* to produce an inline object *fun* suitable for usage by *feval*.. Thus, later inside *MCmeasure* when you want to obtain *f(z)* you just use

```
… feval(fun, z) …
```

(Actually *fcnchk* can do more than this, but I don't want to add that complication yet.)

Examples:

All of the above calls should result in a value close to 1.89. Your result depends upon random numbers and thus will not be identical even with identical invocations.

Extra Stuff:

1. Do some experimental work to support the hypothesis that the values returned by *MCmeasure* have an error related to the inverse of the square root of *n*.

2. Attempt to vectorize the usage of *feval* (**fcnchk** can do this for you). Then use *tic* and *toc* to see how much more efficient this is over the non-vectorized version.

Submission: Attach your M-file named MCmeasure.m to a mail message to aniket@cs.utexas.edu.