

Lecture 6 Review

- Errors for trapezoidal & Simpson's NI techniques.
- Monte Carlo ("dart throwing") technique for integration.
- Code for random number generation.
- C++ features: `if`, declaring functions, `random`, ...
- Just a peek at plotting: `gnuplot`.

<http://www.gnuplot.info/help.html>
<http://sparky.rice.edu/~hartigan/gnuplot.html>

Consolidation

Reading for NI: *CP*, secs. 5.1 – 5.6 (skip code)

- Generate and plot 500 random numbers in (0,1).
- Using NI, calculate π to a 4 sig-fig precision.

Beg, borrow, steal code ...

gnuplot Miscellaneous

Load “junk.gnu”

This is a comment. Stuff to right ignored.

f1(x) = a1*tanh(x/b1) # define the function to be fit

a1 = 30; b1 = 0.05; # initial guess for a1 and b1

plot [-3:3] f1(x) lw 2 # lw = linewidth

plot “physics.dat” using 1:2 title ‘Results’ # quotes around data file, cols 1 & 2

to store a plot in a file:

set terminal postscript eps enhanced

set output “file.eps”

replot

set output # set output back to default

set terminal X11 # ditto for terminal type

<http://www.duke.edu/~hpgavin/gnuplot.html>

<http://sparky.rice.edu/~hartigan/gnuplot.html>

Using GSL Libraries (just a peek)

We can recycle code using “libraries” (collections of source code).
The GNU scientific library (gsl) is an example.

Eg.: `gsl_rng_uniform` generates random numbers (0,1).

➤ How do we use pre-existing code?

Typically, such libraries installed in special directories.

1) Need to compile our code (containing pre-existing stuff) with specific options

```
g++ -Wall -I/usr/include -c ranstuff.cc
```

Creates `ranstuff.o`

compile option

See course web page

2) Need to link your compiled code with pre-existing compiled code

```
g++ -Wall -L/usr/lib -lgsl -lgslcblas ranstuff.o -o ranstuff
```

Creates executable `ranstuff`

// gsl routine to calculate random numbers (0,1)

ranstuff.cc

```
#include <iostream>
#include <gsl/gsl_rng.h>

int main ()
{
    const gsl_rng_type * T;
    gsl_rng * r;

    int i, n = 10; // generate 10 random numbers

    gsl_rng_env_setup();

    T = gsl_rng_default;
    r = gsl_rng_alloc (T);

    for (i = 0; i < n; ++i)
    {
        double u = gsl_rng_uniform (r);
        std::cout << u << std::endl;
    }

    gsl_rng_free (r);

    return 0;
}
```

!! If a line contains `gsl`, stay away !!

Using GSL Libraries (another example)

We need a fancy NI technique. Trapezoidal and Simpson too crude.

The GNU scientific library (gsl) comes to our rescue.

`gsl_integration_qags` is the code we want. Let's try it!

➤ Scoop up sample code on links [www](#) page: `integrate.cc`

- ❑ Examine `integrate.cc` to see how to use `gsl_integration_qags`
- ❑ Examine also chap 16.4 in `gsl` manual for meaning of input params.

Need to compile & link our code. Same as w/ `ranstuff.cc` :

```
g++ -Wall -I/usr/include -c integrate.cc
```

```
g++ -Wall -L/usr/lib -lgsl -lgslcblas integrate.o -o integrate
```

integrate.cc

```
#include <iostream>
#include <iomanip>
#include <cmath>
#include <gsl/gsl_integration.h>

using namespace::std;

// don't touch next line:
double f (double x, void * params) {
    double f = pow(x,x);
    return f;
}

int main ()
{
    // don't touch:
    gsl_integration_workspace * w
        = gsl_integration_workspace_alloc (1000);

    double result, error;

    gsl_function F;
    F.function = &f;

    // don't touch:
    gsl_integration_qags (&F, 0, 1, 0, 1e-7, 1000,
                          w, &result, &error);

    cout << setprecision(10) << "result = \t" << result << endl;
    cout << setprecision(10) << "estimated error = \t" << error << endl;

    // the "w->size" is magic for now:

    cout << setprecision(6) << "intervals = \t" << w->size << endl;

    // don't touch:

    gsl_integration_workspace_free (w);

    return 0;
}
```

Help with gnuplot and Linking/Compiling

My head is exploding.



I need something to read quietly, at my own pace.

<http://www.gnuplot.info>

http://www.gnu.org/software/gsl/manual/html_node/Compiling-and-Linking.html

Link also available from PHYS 3340 links page

Summary

- Basic gnuplot commands
- GSL routine `gsl_rng_uniform` for uniform random number generation in interval (0,1).
- GSL routine `gsl_integration_qags` for NI.

Don't suffer in silence. Scream for help!!!

