Lecture 10 Review

Numerical derivatives: forward and central difference.

ODE solver: 4th order Runge-Kutta

Van der Pol Oscillator Equation

We need an ODE to test drive our RK4 algorithm.

$$\boxed{\ddot{x} + \mu\dot{x}(x^2 - 1) + x = 0}$$

"Van der Pol" oscillator equation, $\mu > 0$.

Describes a "non-linear" oscillation.

(Restoring force not proportional to displacement)

$\mu = 0 \Rightarrow$ simple harmonic motion

$|x| > 1$ damped oscillatory motion, $x \to 0$

$|x| < 1$ growing oscillatory motion, $x \to \infty$

Must be happy medium.

Happy medium seen: plot dx/dt v. x. "Phase space" diagram.

We will see this w/ rk4.cc and gnuplot.

Rewriting our ODE

$$\ddot{x} + \mu\dot{x}(x^2 - 1) + x = 0$$

Recall “standard ‘form” to cast our ODE into. This is a change of variables.

$$d\vec{y}/dt = \vec{f}(t, \vec{y})$$

Just a set of <u>first</u> order ODEs. No big deal.

$$\vec{y} = \begin{pmatrix} y^{(0)}(t) \\ y^{(1)}(t) \\ \vdots \\ y^{(N-1)}(t) \end{pmatrix}$$

$$\vec{f} = \begin{pmatrix} f^{(0)}(t, \vec{y}) \\ f^{(1)}(t, \vec{y}) \\ \vdots \\ f^{(N-1)}(t, \vec{y}) \end{pmatrix}$$

$$y^{(0)}(t) = x(t)$$ Displacement of object

$$f^{(0)}(t, \vec{y}) = y^{(1)}(t)$$

$$y^{(1)}(t) = \dot{x}(t)$$ Velocity of object.

These 3 guys are always the same for any ODE.

Still Rewriting our ODE

$$\ddot{x} + \mu\dot{x}(x^2 - 1) + x = 0$$

We need: $f^{(1)}(t, \vec{y})$

Get this by rearranging ODE and make a simple change of variables:

$$\ddot{x} = -x - \mu\dot{x}(x^2 - 1)$$

$$\dot{y}^{(1)} = -y^{(0)} - \mu y^{(1)}((y^{(0)})^2 - 1)$$

$$\dot{y}^{(1)} = f^{(1)}$$

Using: $\begin{cases} y^{(0)}(t) = x(t) \\ y^{(1)}(t) = \dot{x}(t) \end{cases}$

$$\boxed{f^{(1)} = -y^{(0)} - \mu y^{(1)}((y^{(0)})^2 - 1)}$$

$$\boxed{f^{(0)} = y^{(1)}}$$

Initial conditions. $\begin{cases} y^{(0)}(0) = x(0) \\ y^{(1)}(0) = \dot{x}(0) \end{cases}$

rk4.cc

```
/* solves ode via 4th-order runge-kutta method */

#include <iostream>
#include <gsl/gsl_errno.h>
#include <gsl/gsl_matrix.h>
#include <gsl/gsl_odeiv.h>

using namespace std;


int
   func (double t, const double y[], double f[],
         void *params)
   {
    double mu = *(double *)params;
    f[0] = y[1];
    f[1] = -y[0] - mu*y[1]*(y[0]*y[0] - 1); // CHANGE ME
    return GSL_SUCCESS;
```

```
int main ()
  {
   const gsl_odeiv_step_type * T
     = gsl_odeiv_step_rk4;

   gsl_odeiv_step * s
     = gsl_odeiv_step_alloc (T, 2);

   double mu = 0.05;  //CHANGE ME. damping parameter
   gsl_odeiv_system sys = {func, jac, 2, &mu};

   double t = 0.0, t1 = 100.0;   // CHANGE ME. bounds.
   double h = 1e-2;          // CHANGE ME. step size
```

C++ arrays

(1-dim) Array declaration and initialization:

```
double y[2] = { 3.0, 0.0 };
```

Element values.

<u>Size</u> of array. Explicit integer value can be omitted. Size is <u>fixed</u>.

Type of array elements.

Arrays start counting at 0 !!! The value of y[0] is 3.0.

Change value of y[0]:	`y[0] = 45.3;`
Declare/initialize new variable:	`double a = y[0];`
Warning. This is wrong:	`double b = y[2];`

Ref: http://www.cplusplus.com/doc/tutorial/arrays.html

C++ `while` statement

Useful control structure:

```
while (t < t1 ) {
Blah;
Blah;
Yada;
 }
```

Test condition: T or F.

Various statements.
Executed if test condition true

Note: no ending semicolon (;).

→ `while` Used in rk4.cc ←

Ref: http://www.cplusplus.com/doc/tutorial/control.html

Summary

Van der Pol oscillator & ODE solution via RK4 à la GSL

C++ elements: arrays and `while` statement

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