# Lecture 11 Review

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

C++ elements: arrays and `while` statement

# Simultaneous 2nd-order ODEs

Suppose we have a pair of simultaneous 2nd-order ODEs.

$$\ddot{x} = f(t, x, y, \dot{x}, \dot{y})$$

$$\ddot{y} = g(t, x, y, \dot{x}, \dot{y})$$

What do we do?

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

("standard "form")

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

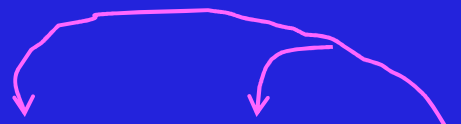
$$\vec{f} = \begin{pmatrix} f^{(0)}(t, \vec{y}) \\ f^{(1)}(t, \vec{y}) \\ \vdots \\ f^{(N-1)}(t, \vec{y}) \end{pmatrix}$$

$$y^{(0)}(t) = x(t)$$

$$y^{(1)}(t) = \dot{x}(t)$$

$$y^{(2)}(t) = y(t)$$

$$y^{(3)}(t) = \dot{y}(t)$$



These y's are different !!

# Example of Simultaneous 2nd-order ODEs

What to use for f(0), f(1), …?

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

$$f^{(1)}(t,\vec{y}) = \dot{y}^{(1)}(t) \;\; (= \ddot{x})$$

$$f^{(2)}(t,\vec{y}) = y^{(3)}(t) \;\; (= \dot{y})$$

$$f^{(3)}(t,\vec{y}) = \dot{y}^{(3)}(t) \;\; (= \ddot{y})$$

If we had more coupled equations, we just add pairs of y and f as here.

Example: Planetary motion. (See *CP*, sec 15.11)

$$f = -\frac{GMm}{r^2}$$ Attractive force exerted on m by M along center-line.

Take to be the sun

$$\vec{f} = m\vec{a} = m\frac{d^2\vec{r}}{dt^2}$$

$$f_x = f\cos\theta = f\frac{x}{r} \qquad f_y = f\sin\theta = f\frac{y}{r} \qquad r = \sqrt{x^2+y^2}$$

$$\frac{d^2x}{dt^2} = -GM\frac{x}{r^3} \qquad \frac{d^2y}{dt^2} = -GM\frac{y}{r^3}$$

# Elements of Orbital Mechanics

Consider the case where a small object (e.g., a comet) orbits the Sun.

$$\vec{F} = -\frac{GMm}{|r|^2}\hat{r}$$

$$E = \frac{1}{2}mv^2 - \frac{GMm}{r}$$

$$\vec{L} = \vec{r} \times (m\vec{v})$$

conserved

$$\frac{mv^2}{r} = \frac{GMm}{r^2}$$

$$v = \sqrt{GM/r}$$

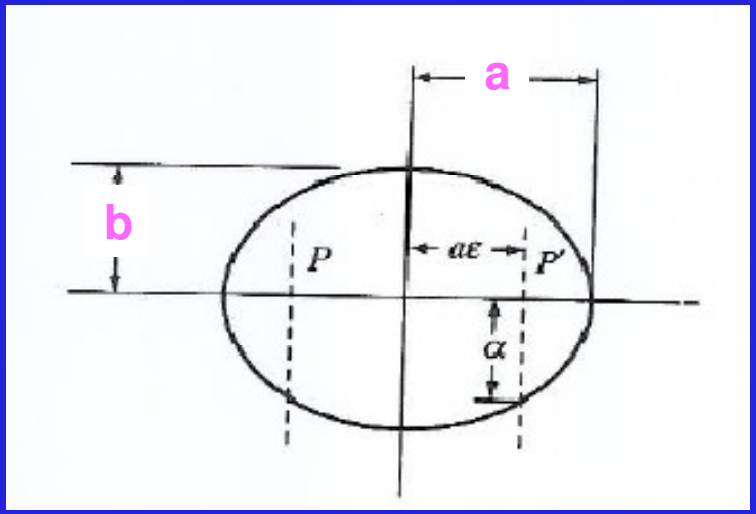
$$E = -\frac{GMm}{2r}$$

Circular orbits only

$$e = \sqrt{1 - b^2/a^2}$$

a = "semi-major axis"

b = "semi-minor axis"




Earth_e = 0.017

# Elliptical Orbits and Kepler's 3rd Law

For elliptical orbits around Sun, we have:

$$E = -\frac{GMm}{2a}$$

$$v = \sqrt{GM\left(\frac{2}{r} - \frac{1}{a}\right)}$$

$$T^2 = \frac{4\pi^2}{GM}a^3$$ Exact for m << M

$G = 6.67 \times 10^{-11}\,\mathrm{m^3/kg{-}sec^2}$

$1\,\mathrm{AU} = 150 \times 10^6\,\mathrm{km} = 1.50 \times 10^{11}\,\mathrm{m}$

$1\,\mathrm{yr} = \pi \times 10^7\,\mathrm{sec}$

$M_\odot = 2.0 \times 10^{30}\,\mathrm{kg}$

[r] = AUs

[t] = years

$GM_\odot = 4\pi^2\,AU^3/yr^2$

Useful for calc's w/ orbit.cc

# orbit.cc

**In class exercise:**

Copy and rename rk4.cc → orbit.cc

Modify orbit.cc to accommodate the 2 above simultaneous 2nd-order ODEs.

For simplicity, set GM = 1.

$f[0] = ?$ $f[2] = ?$

$f[1] = ?$ $f[3] = ?$

Initial conditions:

$x(0) = 0.5$ $y(0) = 0.0$ $v_x(0) = 0.0$ $v_y(0) = 1.63$

Q: what are the units

Adjust number and size of time steps to see orbit close.

Visualize w/ `gnuplot`.

# A peak inside orbit.cc

Define the vectors y and f.

```
int func (double t, const double y[], double f[], void *params)
{
double GM = *(double *)params;
f[0] = y[1];
 f[1] = -GM*y[0]/(pow(y[0]*y[0] + y[2]*y[2],1.5));
 f[2] = y[3];
 f[3] = -GM*y[2]/(pow(y[0]*y[0] + y[2]*y[2],1.5));
 return GSL_SUCCESS;
}
```

# A peak inside orbit.cc (2)

Define df/dy and df/dt :

```
int jac (double t, const double y[], double *dfdy, double dfdt[], void *params)
{
double GM = *(double *)params;
gsl_matrix_view dfdy_mat = gsl_matrix_view_array (dfdy, 4, 4);

gsl_matrix * m = &dfdy_mat.matrix;
gsl_matrix_set (m, 0, 0, 0.0);
gsl_matrix_set (m, 0, 1, 1.0);
gsl_matrix_set (m, 0, 2, 0.0);
gsl_matrix_set (m, 0, 3, 0.0);

gsl_matrix_set (m, 1, 0, -GM*(-2.0*pow(y[0],2) + pow(y[2],2))/
(pow(y[0]*y[0] + y[2]+y[2],2.5)));

gsl_matrix_set (m, 1, 1, 0.0); gsl_matrix_set (m, 1, 2, GM*(3.0*y[0]*y[2])/
(pow(y[0]*y[0] + y[2]+y[2],2.5)));

gsl_matrix_set (m, 1, 3, 0.0);
```

matrix

4 rows

4 cols

m = df/dy

Entry in $0^{th}$ row, $1^{st}$ col in matrix m = 1.0.

useful mnemonic for matrix A: $A_{rc}$

$df^{(1)}/dy^{(0)}$

$df^{(1)}/dy^{(1)}$

# A peak inside orbit.cc (3)

Define df/dt.

```
...
dfdt[0] = 0.0;
dfdt[1] = 0.0;
dfdt[2] = 0.0;
dfdt[3] = 0.0;
return GSL_SUCCESS;
 }

int main () {
const gsl_odeiv_step_type * T = gsl_odeiv_step_rk4;
 gsl_odeiv_step * s = gsl_odeiv_step_alloc (T, 4);

double GM = 1.00; //CHANGE ME. value of G*M_sun
gsl_odeiv_system sys = {func, jac, 4, &GM}; // GM is ODE dependent

double t = 0.0, t1 = 100.0; // CHANGE ME. bounds.
double h = 0.05; // CHANGE ME. step size

// initial conditions: y[0], y[2] = position, y[1],y[3] = velocity
double y[4] = { 0.5, 0.0, 0.0, 1.25 }, y_err[4];
double dydt_in[4], dydt_out[4];
```

df/dt = 0

Runge-Kutta 4th order algorithm.

4 is the dimension of f and y.

initial and final times. Careful of units.

step size.

4 is the dimension of f and y.

# C++ file IO (rd_data_file.cc)

```
/* simple code to read from an input file
   and then write to an output file. */

#include <iostream>
#include <fstream>

using namespace::std;

int main()
{
  ifstream in_file;
  ofstream out_file;

  int r,s,t;
// open inoput and output files:

  in_file.open("myjunk.dat");
  out_file.open("example_out.dat");

// checking file is open for business:
  if ( in_file.is_open() )
    {
      while(! in_file.eof() )  // still stuff to read
        {
          in_file >> r >> s >> t;
          out_file << "first col = " << r
                   << "\t 2nd col " << s
                   << "\t 3rd col = " << t << endl;
        }
    }

  in_file.close();    // good practice
  out_file.close();   // good practice
  return 0;
}
```

# Summary

Simultaneous 2nd-order ODEs & solution via RK4 à la GSL

Elliptical orbits as an example of simultaneous 2nd-order ODEs

C++ file IO.

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