Computational Methods in Physics

# HOW TO SOLVE DIFFERENTIAL EQUATIONS

## *Euler, Euler-Cromer, RK4, and Verlet Methods*

# Outline: Where We Are Going

- Basics: Euler and Euler++ :-) (more stable)
  - This week
  - Harmonic oscillator: pendulum example
  - Homework: rocket equation
- Runge-Kutta (fourth-order examples)
  - Cosmology example in class
  - Homework: the classical wave equation in 1-D
- The Störmer-Verlet “Leapfrog” algorithm
  - N-body Monte Carlo. A longer, tougher HW (20 pt)

Start thinking about your final projects!!

# Web Links for Euler, Pendulum

- http://tutorial.math.lamar.edu/Classes/DE/EulersMethod.aspx
  - Study the many examples on this webpage!
- Full analytical solution with sin(theta) which we will be comparing my example to:
  - http://sbfisica.org.br/rbef/pdf/070707.pdf

We're still assuming mass independence.

- Simple oscillation
  - http://hyperphysics.phy-astr.gsu.edu/hbase/pend.html
- Driven aka forced
  - hyperphysics.phy-astr.gsu.edu/hbase/oscdr.html

# Euler-Cromer “Semi-Implicit”

- Better: conserves energy, dampens runaway errors (greater stability)
- Also known as: Euler-Richardson

- Two versions, Cromer and Richardson:
  - https://stackoverflow.com/questions/77238290/is-the-euler-cromer-method-or-is-it-incorrect

# Preview of Pendulum Code

- theta'' + (g/L)sin(theta) = 0

We will NOT be using sin(theta) is = ~theta (small angle approximation).

  - Define variables for initial and final values (i, i+1)
- Read in 'g' and length from file or set in code
- Provide initial values for theta AND theta'
  - Latter can be 0 but former not 0 (unless reverse)
- Maximum time code simulates
- Step size (in seconds)

Smaller dt means greater precision.

  - Smaller better of course, but code slower
- Damping, etc. terms for some extra pizzazz
- If using angles in degrees, convert to rad!
  - The default in the C++ language

Default of radians: like most calculators.

# ...continued

Output: screen or file, as usual, as you prefer.

- “for” loop from t=0 to maximum time, with a given step size
  - So, uses doubles not integers
  - OR, “while” loop with code ending based on accuracy or convergence (answer ~unchanging)
- theta’’_i = (-g/L)*sin(theta_i)
- theta’_f = theta’_i + dt * theta’’_i
- theta_f = theta_i + dt * theta’_i
  - theta_i + dt * theta’_f for Cromer
- Obviously, output time and angle (back to °)
- theta_i = theta_f and theta’_i = theta’_f

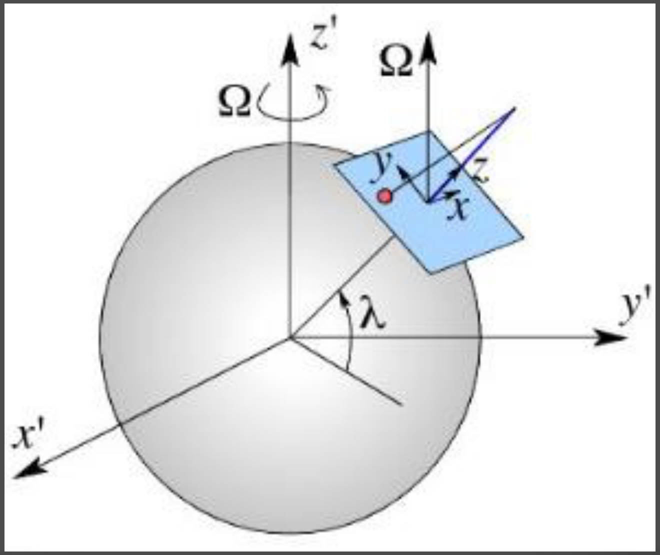
Equations: repeating on the whiteboard.

# Can explore, if time permits

- Midpoint, secant instead of tangent,
- Integration methods from last week.

# Applications: The Foucault Pendulum

Definitive proof the Earth rotates with no astro. We are NOT in a perfectly inertial reference frame




Free pendulum will change plane over course of day, and knock down pins for example. Lat-dependent. Modern versions are table-top even

# Homework #5, due Thurs. 02/29

- Solve the rocket equation instead of pendulum with *2* methods: plain Euler and 1 better. Newton's broader 2nd law: <u>doing on board</u>
  - F=-w (but dm/dt => a>0!) v(t=0) = O(10^3) m/s
  - -m(t)*g(t) = m(t)*a(t)+[dm(t)/dt]*v(t). F isn't ma!!
  - Replace the 'a' & 'v' with h double-dot and dot..

- Plot F, m, a, g, v, h (starts on surf) vs. t all in SI units. In pdf: analyze; draw concl on behavior. (Expected shapes?) <u>Time to LEO? Order mag</u>

LEO = Low Earth Orbit

imp = impulses

  - Get real values for *m*s, imp, thrust but no drag no rotation
- 1st: you have to make sure your *time step* is small enough program is accurate (~0.01 - 1 s)
  - May need to be so small you need millions of points
  - Might need to run program <u>several times</u>, then splice.

- Second-order not first so is tricky, also coupled
  - But, at least you can compare results for accuracy to the classical rocket equation (Tsiolkovsky, “ideal”)

$$\Delta v = v_e \ln \frac{m_0}{m_f} = I_{sp} g_0 \ln \frac{m_0}{m_f}$$

This is idealized (note the fixed g or $g_0$). You can look up specific impulse in tables. Do e.g. Musk’s Falcon Heavy, or Saturn V

where: Use $I_{sp}$, Thrust to find dm/dt (flat).

$\Delta v$ is delta-v – the maximum change of velocity of the vehicle (with no external forces acting).

$m_0$ is the initial total mass, including propellant, also known as wet mass.

$m_f$ is the final total mass without propellant, also known as dry mass.

source: Wikipedia

$v_e = I_{sp} g_0$ is the effective exhaust velocity, where:

Not bad for this

LOOK IT UP!

$I_{sp}$ is the specific impulse in dimension of time.

$g_0$ is standard gravity.

(log in C++)

*ln* is the natural logarithm function.

# Questions?

- Extra credit: Impress me by adding the drag force of the air (with air density going down!!)
  - Go back to your Physics I textbook (hint: F prop to v^2, but also many parameters you must look up)

# your code has to obey the last slide

- Final mass should be the dry mass (initial wet)
- To check $\Delta$v (max) determine final vel (minus 0)
- Modify my pendulum, or write your own fresh code from scratch (this is entirely your choice)
- Don't assume fixed *g* (too easy :) $GM_E/(R_E+h)^2$
- Apply common sense: doesn't take < seconds and doesn't take months-yrs.+ to get into orbit
- Never mix units: convert everything you find online into metric, and keep all there (*m* in kg)
- Feel free to study the many analytic derivations
- Plot ***2*** Euler-*like* methods (e.g., Richardson)