PHY321: Conservative forces, examples and theory

Morten Hjorth-Jensen[1,2]

[1]Department of Physics and Astronomy and Facility for Rare Ion Beams (FRIB), Michigan State University, USA
[2]Department of Physics, University of Oslo, Norway

Feb 16, 2022

Aims and Overarching Motivation

Monday February 14. Discussion of conditions for conservative forces and summing up our discussion on conservative forces. Discussion of potential surfaces and their interpretations.

Reading suggestion: Taylor sections 4.6, 4.9, 4.10 and 5.1 and 5.2 on harmonic oscillations.

Wednesday February 16. The Earth-Sun problem and energy-conserving algorithms and how to encode in more efficient ways various algorithms for solving the equations of motion (Euler, Euler-Cromer and Velocity Verlet).

- Links to Julie's material on code reusability

Reading suggestions: Taylor section 4.8 and these notes

Friday February 18. Working on the Earth-Sun problem and hw 5. **Reading suggestions:** Taylor chapters 3 and 4 and these notes.

The curl of a force and link between Line Integrals and conservative forces

The concept of line integrals plays an important role in our discussion of energy conservation, our definition of potentials and conservative forces.

Let us remind ourselves of some the basic elements (most of you may have seen this in a calculus course under the general topic of vector fields).

© 1999-2022, "Morten Hjorth-Jensen":"http://mhjgit.github.io/info/doc/web/". Released under CC Attribution-NonCommercial 4.0 license

We define a path integration C, that is we integrate from a point $\boldsymbol{r}_1$ to a point $\boldsymbol{r}_2$. Let us assume that the path C is represented by an arc length s. In three dimension we have the following representation of C

$$\boldsymbol{r}(s) = x(s)\boldsymbol{e}_1 + y(s)\boldsymbol{e}_2 + z(s)\boldsymbol{e}_3,$$

then our integral of a function $f(x, y, z)$ along the path C is defined as

$$\int_C f(x, y, z)ds = \int_a^b f\left(x(s), y(s), z(s)\right) ds,$$

where the initial and final points are a and b, respectively.

Exactness and Independence of Path

With the definition of a line integral, we can in turn set up the theorem of independence of integration path.

Let us define $f(x, y, z)$, $g(x, y, z)$ and $h(x, y, z)$ to be functions which are defined and continuous in a domain D in space. Then a line integral like the above is said to be independent of path in D, if for every pair of endpoints a and b in D the value of the integral is the same for all paths C in D starting from a point a and ending in a point b. The integral depends thus only on the integration limits and not on the path.

Differential Forms

An expression of the form

$$f dx + g dy + h dz,$$

where f, g and h are functions defined in D, is a called a first-order differential form in three variables. The form is said to be exact if it is the differential

$$du = \frac{\partial u}{\partial x}dx + \frac{\partial u}{\partial y}dy + \frac{\partial u}{\partial z}dz,$$

of a differentiable function $u(x, y, z)$ everywhere in D, that is

$$du = f dx + g dy + h dz.$$

It is said to be exact if and only if we can then set

$$f = \frac{\partial u}{\partial x},$$

and

$$g = \frac{\partial u}{\partial y},$$

and

$$h = \frac{\partial u}{\partial z},$$

everywhere in the domain D.

In Vector Language

In vector language the above means that the differential form

$$f dx + g dy + h dz,$$

is exact in D if and only if the vector function (it could be a force, or velocity, acceleration or other vectors we encounter in this course)

$$\boldsymbol{F} = f\boldsymbol{e}_1 + g\boldsymbol{e}_2 + h\boldsymbol{e}_3,$$

is the gradient of a function $u(x, y, z)$

$$\boldsymbol{v} = \boldsymbol{\nabla} u = \frac{\partial u}{\partial x}\boldsymbol{e}_1 + \frac{\partial u}{\partial y}\boldsymbol{e}_2 + \frac{\partial u}{\partial z}\boldsymbol{e}_3.$$

Path Independence Theorem

If this is the case, we can state the path independence theorem which states that with functions $f(x, y, z)$, $g(x, y, z)$ and $h(x, y, z)$ that fulfill the above exactness conditions, the line integral

$$\int_C \left(f dx + g dy + h dz\right),$$

is independent of path in D if and only if the differential form under the integral sign is exact in D.

This is the path independence theorem.

We will not give a proof of the theorem. You can find this in any vector analysis chapter in a mathematics textbook.

We note however that the path integral from a point p to a final point q is given by

$$\int_p^q (f dx + g dy + h dz) = \int_p^q \left(\frac{\partial u}{\partial x}dx + \frac{\partial u}{\partial y}dy + \frac{\partial u}{\partial z}dz\right) = \int_p^q du.$$

Assume now that we have a dependence on a variable s for x, y and z. We have then

$$\int_p^q du = \int_{s_1}^{s_2} \frac{du}{ds} ds = u(x(s), y(s), z(s))|_{s=s_1}^{s=s_2} = u(q) - u(p).$$

This last equation

$$\int_p^q (f dx + g dy + h dz) = u(q) - u(p),$$

is the analogue of the usual formula

$$\int_a^b f(x) dx = F(x)|_a^b = F(b) - F(a),$$

with $F'(x) = f(x)$.

Work-Energy Theorem again

We remember that a the work done by a force $\boldsymbol{F} = f\boldsymbol{e}_1 + g\boldsymbol{e}_2 + h\boldsymbol{e}_3$ on a displacemnt $d\boldsymbol{r}$ is

$$W = \int_C \boldsymbol{F} d\boldsymbol{r} = \int_C (f dx + g dy + h dz).$$

From the path independence theorem, we know that this has to result in the difference between the two endpoints only. This is exact if and only if the force is the force $\boldsymbol{F}$ is the gradient of a scalar function u. We call this scalar function, which depends only the positions x, y, z for the potential energy $V(x, y, z) = V(\boldsymbol{r})$.

We have thus

$$\boldsymbol{F}(\boldsymbol{r}) \propto \boldsymbol{\nabla} V(\boldsymbol{r}),$$

and we define this as

$$\boldsymbol{F}(\boldsymbol{r}) = -\boldsymbol{\nabla} V(\boldsymbol{r}).$$

Such a force is called **a conservative force**. The above expression can be used to demonstrate energy conservation.

Additional Theorem

Finally we can define the criterion for exactness and independence of path. This theorem states that if $f(x, y, z)$, $g(x, y, z)$ and $h(x, y, z)$ are continuous functions with continuous first partial derivatives in the domain D, then the line integral

$$\int_C (f dx + g dy + h dz),$$

is independent of path in D when

$$\frac{\partial h}{\partial y} = \frac{\partial g}{\partial z},$$

and

$$\frac{\partial f}{\partial z} = \frac{\partial h}{\partial x},$$

and

$$\frac{\partial g}{\partial x} = \frac{\partial f}{\partial y}.$$

This leads to the **curl** of $\boldsymbol{F}$ being zero

$$\boldsymbol{\nabla} \times \boldsymbol{F} = \boldsymbol{\nabla} \times (-\boldsymbol{\nabla} V(\boldsymbol{r})) = 0!$$

Summarizing

A conservative force $\boldsymbol{F}$ is a defined as the partial derivative of a scalar potential which depends only on the position,

$$\boldsymbol{F}(\boldsymbol{r}) = -\boldsymbol{\nabla} V(\boldsymbol{r}).$$

This leads to conservation of energy and a path independent line integral as long as the curl of the force is zero, that is

$$\boldsymbol{\nabla} \times \boldsymbol{F} = \boldsymbol{\nabla} \times (-\boldsymbol{\nabla} V(\boldsymbol{r})) = 0.$$

Graphing the potential energy and what we can learn from that

This is taken from homework 4, exercises 5.

A particle is under the influence of a force $F = -kx + kx^3/\alpha^2$, where k and α are constants and k is positive.

Determine $V(x)$ and discuss the motion. It can be convenient here to make a sketch/plot of the potential as function of x.

We assume that the potential is zero at say $x = 0$. Integrating the force from zero to x gives

$$V(x) = -\int_0^x F(x')dx' = \frac{kx^2}{2} - \frac{kx^4}{4\alpha^2}.$$

Making the plot

The following code plots the potential. We have chosen values of $\alpha = k = 1.0$. Feel free to experiment with other values. We plot $V(x)$ for a domain of $x \in [-2, 2]$.

```
import numpy as np
import matplotlib.pyplot as plt
import math

x0= -2.0
xn = 2.1
Deltax = 0.1
alpha = 1.0
k = 1.0
#set up arrays
x = np.arange(x0,xn,Deltax)
n = np.size(x)
V = np.zeros(n)
V = 0.5*k*x*x-0.25*k*(x**4)/(alpha*alpha)
plt.plot(x, V)
plt.xlabel("x")
plt.ylabel("V")
plt.show()
```

Interpreting the results

From the plot here (with the chosen parameters)

1. we see that with a given initial velocity we can overcome the potential energy barrier

and leave the potential well for good.

1. If the initial velocity is smaller (see next exercise) than a certain value, it will remain trapped in the potential well and oscillate back and forth around $x = 0$. This is where the potential has its minimum value.
2. If the kinetic energy at $x = 0$ equals the maximum potential energy, the object will oscillate back and forth between the minimum potential energy at $x = 0$ and the turning points where the kinetic energy turns zero. These are the so-called non-equilibrium points.

Final interpretations

What happens when the energy of the particle is $E = (1/4)k\alpha^2$? Hint: what is the maximum value of the potential energy?

From the figure we see that the potential has a minimum at at $x = 0$ then rises until $x = \alpha$ before falling off again. The maximum potential, $V(x \pm \alpha) = k\alpha^2/4$. If the energy is higher, the particle cannot be contained in the well. The turning points are thus defined by $x = \pm\alpha$. And from the previous plot you can easily see that this is the case ($\alpha = 1$ in the abovementioned Python code).

The Earth-Sun system

We will now venture into a study of a system which is energy conserving. The aim is to see if we (since it is not possible to solve the general equations analytically) we can develop stable numerical algorithms whose results we can trust!

We solve the equations of motion numerically. We will also compute quantities like the energy numerically.

We start with a simpler case first, the Earth-Sun system in two dimensions only. The gravitational force F_G on the earth from the sun is

$$\boldsymbol{F}_G = -\frac{GM_\odot M_E}{r^3}\boldsymbol{r},$$

where G is the gravitational constant,

$$M_E = 6 \times 10^{24}\text{Kg},$$

the mass of Earth,

$$M_\odot = 2 \times 10^{30}\text{Kg},$$

the mass of the Sun and

$$r = 1.5 \times 10^{11}\text{m},$$

is the distance between Earth and the Sun. The latter defines what we call an astronomical unit **AU**.

The Earth-Sun system, Newton's Laws

From Newton's second law we have then for the x direction

$$\frac{d^2x}{dt^2} = -\frac{F_x}{M_E},$$

and

$$\frac{d^2y}{dt^2} = -\frac{F_y}{M_E},$$

for the y direction.

Here we will use that $x = r\cos(\theta)$, $y = r\sin(\theta)$ and

$$r = \sqrt{x^2 + y^2}.$$

We can rewrite

$$F_x = -\frac{GM_\odot M_E}{r^2}\cos(\theta) = -\frac{GM_\odot M_E}{r^3}x,$$

and

$$F_y = -\frac{GM_\odot M_E}{r^2}\sin(\theta) = -\frac{GM_\odot M_E}{r^3}y,$$

for the y direction.

The Earth-Sun system, rewriting the Equations

We can rewrite these two equations

$$F_x = -\frac{GM_\odot M_E}{r^2}\cos(\theta) = -\frac{GM_\odot M_E}{r^3}x,$$

and

$$F_y = -\frac{GM_\odot M_E}{r^2}\sin(\theta) = -\frac{GM_\odot M_E}{r^3}y,$$

as four first-order coupled differential equations

$$\frac{dv_x}{dt} = -\frac{GM_\odot}{r^3}x,$$

$$\frac{dx}{dt} = v_x,$$

$$\frac{dv_y}{dt} = -\frac{GM_\odot}{r^3}y,$$

$$\frac{dy}{dt} = v_y.$$

Building a code for the solar system, final coupled equations

The four coupled differential equations

$$\frac{dv_x}{dt} = -\frac{GM_\odot}{r^3}x,$$

$$\frac{dx}{dt} = v_x,$$

$$\frac{dv_y}{dt} = -\frac{GM_\odot}{r^3}y,$$

$$\frac{dy}{dt} = v_y,$$

can be turned into dimensionless equations or we can introduce astronomical units with 1 AU $= 1.5 \times 10^{11}$.

Using the equations from circular motion (with $r = 1$AU)

$$\frac{M_E v^2}{r} = F = \frac{GM_\odot M_E}{r^2},$$

we have

$$GM_\odot = v^2 r,$$

and using that the velocity of Earth (assuming circular motion) is $v = 2\pi r/\text{yr} = 2\pi\text{AU/yr}$, we have

$$GM_\odot = v^2 r = 4\pi^2 \frac{(\text{AU})^3}{\text{yr}^2}.$$

Building a code for the solar system, discretized equations

The four coupled differential equations can then be discretized using Euler's method as (with step length h)

$$v_{x,i+1} = v_{x,i} - h\frac{4\pi^2}{r_i^3}x_i,$$

$$x_{i+1} = x_i + hv_{x,i},$$

$$v_{y,i+1} = v_{y,i} - h\frac{4\pi^2}{r_i^3}y_i,$$

$$y_{i+1} = y_i + hv_{y,i},$$

Code Example with Euler's Method

The code here implements Euler's method for the Earth-Sun system using a more compact way of representing the vectors. Alternatively, you could have spelled out all the variables v_x, v_y, x and y as one-dimensional arrays.

```
# Common imports
import numpy as np
import pandas as pd
from math import *
import matplotlib.pyplot as plt
import os

# Where to save the figures and data files
PROJECT_ROOT_DIR = "Results"
FIGURE_ID = "Results/FigureFiles"
DATA_ID = "DataFiles/"

if not os.path.exists(PROJECT_ROOT_DIR):
    os.mkdir(PROJECT_ROOT_DIR)

if not os.path.exists(FIGURE_ID):
    os.makedirs(FIGURE_ID)

if not os.path.exists(DATA_ID):
    os.makedirs(DATA_ID)

def image_path(fig_id):
    return os.path.join(FIGURE_ID, fig_id)

def data_path(dat_id):
    return os.path.join(DATA_ID, dat_id)

def save_fig(fig_id):
    plt.savefig(image_path(fig_id) + ".png", format='png')


DeltaT = 0.001
#set up arrays
tfinal = 10 # in years
n = ceil(tfinal/DeltaT)
# set up arrays for t, a, v, and x
t = np.zeros(n)
v = np.zeros((n,2))
r = np.zeros((n,2))
# Initial conditions as compact 2-dimensional arrays
r0 = np.array([1.0,0.0])
v0 = np.array([0.0,2*pi])
r[0] = r0
v[0] = v0
Fourpi2 = 4*pi*pi
# Start integrating using Euler's method
for i in range(n-1):
    # Set up the acceleration
    # Here you could have defined your own function for this
    rabs = sqrt(sum(r[i]*r[i]))
    a =  -Fourpi2*r[i]/(rabs**3)
    # update velocity, time and position using Euler's forward method
    v[i+1] = v[i] + DeltaT*a
    r[i+1] = r[i] + DeltaT*v[i]
```

```
    t[i+1] = t[i] + DeltaT
# Plot position as function of time
fig, ax = plt.subplots()
#ax.set_xlim(0, tfinal)
ax.set_ylabel('y[AU]')
ax.set_xlabel('x[AU]')
ax.plot(r[:,0], r[:,1])
fig.tight_layout()
save_fig("EarthSunEuler")
plt.show()
```

Problems with Euler's Method

We notice here that Euler's method doesn't give a stable orbit. It means that we cannot trust Euler's method. In a deeper way, as we will see in homework 5, Euler's method does not conserve energy. It is an example of an integrator which is not symplectic.

Here we present thus two methods, which with simple changes allow us to avoid these pitfalls. The simplest possible extension is the so-called Euler-Cromer method. The changes we need to make to our code are indeed marginal here. We need simply to replace

```
r[i+1] = r[i] + DeltaT*v[i]
```

in the above code with the velocity at the new time t_{i+1}

```
r[i+1] = r[i] + DeltaT*v[i+1]
```

By this simple caveat we get stable orbits. Below we derive the Euler-Cromer method as well as one of the most utlized algorithms for sovling the above type of problems, the so-called Velocity-Verlet method.

Deriving the Euler-Cromer Method

Let us repeat Euler's method. We have a differential equation

$$y'(t_i) = f(t_i, y_i) \tag{1}$$

and if we truncate at the first derivative, we have from the Taylor expansion

$$y_{i+1} = y(t_i) + (\Delta t) f(t_i, y_i) + O(\Delta t^2), \tag{2}$$

which when complemented with $t_{i+1} = t_i + \Delta t$ forms the algorithm for the well-known Euler method. Note that at every step we make an approximation error of the order of $O(\Delta t^2)$, however the total error is the sum over all steps $N = (b-a)/(\Delta t)$ for $t \in [a, b]$, yielding thus a global error which goes like $NO(\Delta t^2) \approx O(\Delta t)$.

To make Euler's method more precise we can obviously decrease Δt (increase N), but this can lead to loss of numerical precision. Euler's method is not recommended for precision calculation, although it is handy to use in order to get a first view on how a solution may look like.

Euler's method is asymmetric in time, since it uses information about the derivative at the beginning of the time interval. This means that we evaluate the position at y_1 using the velocity at v_0. A simple variation is to determine x_{n+1} using the velocity at v_{n+1}, that is (in a slightly more generalized form)

$$y_{n+1} = y_n + v_{n+1} + O(\Delta t^2) \tag{3}$$

and

$$v_{n+1} = v_n + (\Delta t)a_n + O(\Delta t^2). \tag{4}$$

The acceleration a_n is a function of $a_n(y_n, v_n, t_n)$ and needs to be evaluated as well. This is the Euler-Cromer method.

Exercise: go back to the above code with Euler's method and add the Euler-Cromer method.

Deriving the Velocity-Verlet Method

Let us stay with x (position) and v (velocity) as the quantities we are interested in.

We have the Taylor expansion for the position given by

$$x_{i+1} = x_i + (\Delta t)v_i + \frac{(\Delta t)^2}{2}a_i + O((\Delta t)^3).$$

The corresponding expansion for the velocity is

$$v_{i+1} = v_i + (\Delta t)a_i + \frac{(\Delta t)^2}{2}v_i^{(2)} + O((\Delta t)^3).$$

Via Newton's second law we have normally an analytical expression for the derivative of the velocity, namely

$$a_i = \frac{d^2x}{dt^2}|_i = \frac{dv}{dt}|_i = \frac{F(x_i, v_i, t_i)}{m}.$$

If we add to this the corresponding expansion for the derivative of the velocity

$$v_{i+1}^{(1)} = a_{i+1} = a_i + (\Delta t)v_i^{(2)} + O((\Delta t)^2) = a_i + (\Delta t)v_i^{(2)} + O((\Delta t)^2),$$

and retain only terms up to the second derivative of the velocity since our error goes as $O(h^3)$, we have

$$(\Delta t)v_i^{(2)} \approx a_{i+1} - a_i.$$

We can then rewrite the Taylor expansion for the velocity as

$$v_{i+1} = v_i + \frac{(\Delta t)}{2}\left(a_{i+1} + a_i\right) + O((\Delta t)^3).$$

The velocity Verlet method

Our final equations for the position and the velocity become then

$$x_{i+1} = x_i + (\Delta t)v_i + \frac{(\Delta t)^2}{2}a_i + O((\Delta t)^3),$$

and

$$v_{i+1} = v_i + \frac{(\Delta t)}{2}\left(a_{i+1} + a_i\right) + O((\Delta t)^3).$$

Note well that the term a_{i+1} depends on the position at x_{i+1}. This means that you need to calculate the position at the updated time t_{i+1} before the computing the next velocity. Note also that the derivative of the velocity at the time t_i used in the updating of the position can be reused in the calculation of the velocity update as well.

Adding the Velocity-Verlet Method

We can now easily add the Verlet method to our original code as

```
DeltaT = 0.01
#set up arrays
tfinal = 10 # in years
n = ceil(tfinal/DeltaT)
# set up arrays for t, a, v, and x
t = np.zeros(n)
v = np.zeros((n,2))
r = np.zeros((n,2))
# Initial conditions as compact 2-dimensional arrays
r0 = np.array([1.0,0.0])
v0 = np.array([0.0,2*pi])
r[0] = r0
v[0] = v0
Fourpi2 = 4*pi*pi
# Start integrating using the Velocity-Verlet  method
for i in range(n-1):
    # Set up forces, air resistance FD, note now that we need the norm of the vecto
    # Here you could have defined your own function for this
    rabs = sqrt(sum(r[i]*r[i]))
    a =  -Fourpi2*r[i]/(rabs**3)
    # update velocity, time and position using the Velocity-Verlet method
    r[i+1] = r[i] + DeltaT*v[i]+0.5*(DeltaT**2)*a
    rabs = sqrt(sum(r[i+1]*r[i+1]))
    anew = -4*(pi**2)*r[i+1]/(rabs**3)
    v[i+1] = v[i] + 0.5*DeltaT*(a+anew)
    t[i+1] = t[i] + DeltaT
# Plot position as function of time
fig, ax = plt.subplots()
ax.set_ylabel('y[AU]')
ax.set_xlabel('x[AU]')
ax.plot(r[:,0], r[:,1])
fig.tight_layout()
save_fig("EarthSunVV")
plt.show()
```

You can easily generalize the calculation of the forces by defining a function which takes in as input the various variables. We leave this as a challenge to you.