Details about the final project

1. You can either find your own project or choose one from the following four given projects. If you choose to find your own project, it would be better to contact the TAs to check whether the project is suitable.
2. The minimum requirements are as following:
   a) The core codes without the plotting part should be more than **50 lines**. This requirement is very strict, and no need to make the codes unnecessary long. The quality is more important than the length.
   b) There should be at least three figures and one animation to show your results;
   c) You need to submit both your codes in .py file and your report.
   d) The report should contain an abstract (no more than 100 words), main text and references. In the main text, you need to describe the background (motivation), the algorithm and the conclusions. You can also show what you have done, what kind of effort you put, and what you have learned through the project.
   e) The length of text (excluding tables, figures, references, etc) should be more than **2 pages** with a standard format (12 pt Times New Roman fonts, single-line spacing, at least 2.5 cm margin on each side)
3. Deadline is **May 25, 2021**. You will get zero points of the final project if the deadline is passed.

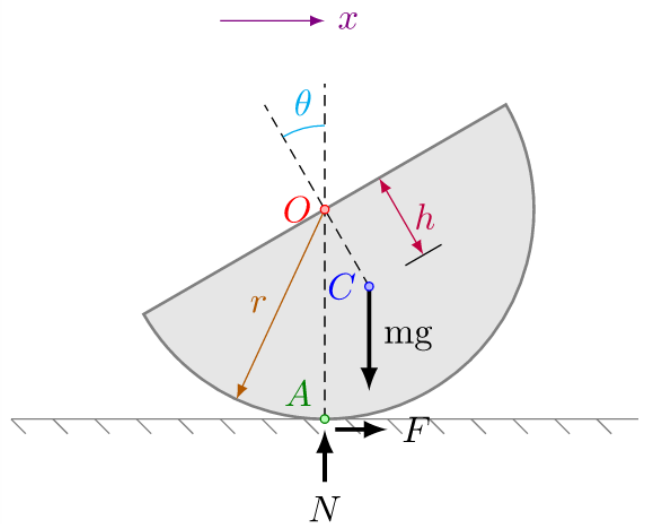
Grading: The purpose of final project is to practice what you have learned through the course. As long as you put effort in it, there is no need to worry about the grade.

1. The total points will be 100. For these extremely good ones, there are some bonus points up to 20. You can take the final project as a chance to improve your GPA if you did not do well in the assignments.
2. The grade is given based on the quality of the codes and report. The main principle is to show that you really put effort in the project and report and you really learn something valuable.
3. Once you can meet the minimum requirements, you will get at least 60 out of 100 points.

Project 1: Roly-poly toy

The roly-poly toy is one of the most common toys. It is usually shaped like an egg, light on top and heavy on the bottom, and can be erected automatically after being flipped. The earliest record of the drunken fairy in the Tang Dynasty is a roly-poly toy. The principle of roly-poly toy not falling is not difficult to understand: objects that are light on the top and heavy on the bottom are relatively stable, and the lower of the center of gravity, the more stable. When the roly-poly toy is in balance, the distance between the center of gravity and the contact point is the smallest, that is, the center of gravity is the lowest. After deviating from the equilibrium position, the center of gravity always rises. Therefore, the balance in this state is a stable balance. So no matter how the roly-poly toy swings, it never fails.

Simpler Model: We first build a simple roly-poly toy model by only considering the lower part.



where, we further simplify the roly-poly toy on the xz plane, where point C is the gravity center, O is the center of the hemisphere, h is the distance between O and C, mg is the gravity of this hemisphere, A is the contacting point, r is the radius of the hemisphere, θ is the variable to describe the movement of this hemisphere, the support force and friction force of the ground to the hemisphere are respectively N and F.

Now, we derive the equations of motion (there are translational motion and rotary motion):

$$\begin{aligned} \mathrm{F} &= \mathrm{m}a_{Cx} \\ \mathrm{N} - \mathrm{mg} &= \mathrm{m}a_{Cy} \qquad (1) \\ -(\mathrm{h}\sin\theta)\mathrm{N} + (\mathrm{r} - \mathrm{h}\cos\theta)\mathrm{F} &= I_C\ddot{\theta} \end{aligned}$$

where, a_{Cx} and a_{Cy} are the accelerated speed of the gravity center and it satisfies:

$$\begin{aligned} a_{Cx} &= \ddot{x} + h\ddot{\theta}\cos\theta - h\dot{\theta}^2\sin\theta \qquad (2) \\ a_{Cy} &= h\ddot{\theta}\sin\theta - h\dot{\theta}^2\cos\theta \end{aligned}$$

where x is the x-coordinate of O . Considering there is no slip on the contact point, we also have: $x = -\theta r$. Then, we have:

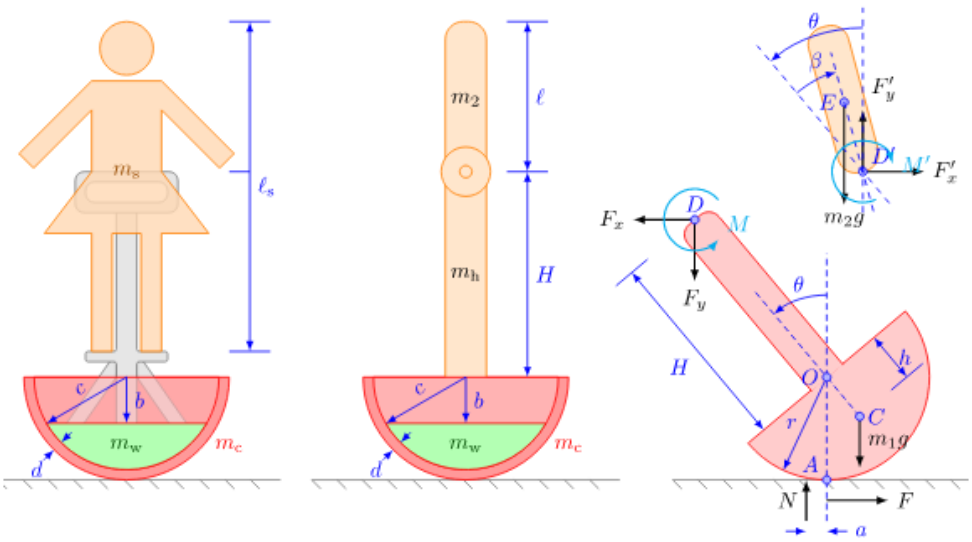
$$\ddot{\theta} = -\frac{hmsin\theta(g + r\dot{\theta}^2)}{I_C + m(r^2 + h^2 - 2hrcos\theta)} \qquad (3)$$

The above final movement equation can also be derived by using Euler-Lagrange equation.

Please first using the final equation (3) to solve $\theta(t)$,$\dot{\theta}(t)$, $h_c(t)$ and plot them. Then, give the animation of this roly-poly toy.

Here, h_c is the y-coordinate of C and we take these parameters: $h = \frac{3}{8}r$, $I_c = \frac{83}{320}mr^2$,$r = 0.5m$, $m = 250kg$, $\theta(t = 0) = 60°$.

More complicated model: Now we consider the model with one person standing on the roly-poly toy (see the below figure):



where it is just a compound pendulum model. We simplify this person as two cylinders with the same radius ($r_s = 0.094m$) and different heights ($H = 0.9m$, $l = 0.73m$). Thus, we take the lower part of this person and this roly-poly toy as a whole whose moment of inertia is $I_1 = 28.34kg \cdot m^2$ and mass is $m_1 = 275kg$ and the higher part of this person is another part whose moment of inertia is $I_2 = 0.94kg \cdot m^2$ and mass is $m_2 = 20kg$. This figure gives the force analysis of these two parts and due to the realistic rolling process, the action point of support force N is not at point A but another point a from point A.

1. In this question, please first derive the kinetic equations of these two parts combined with the condition $x = -\theta \cdot r$and then solve $\ddot{\theta} = f(\beta(t))$. (You can also use the Euler-Lagrange equation to solve this). Then we will see that $\ddot{\theta}$ is controlled by $\beta(t)$, now set $\beta(t)$ as $\beta(t) = A_{mp}sin(\frac{2\pi}{T}t)$, where A_{mp} is the amplitude and T is the period.

2. Set $A_{mp} = 0$,$a = 0$, $\theta(t = 0) = 30°$, plot $\theta(t)$, $\beta(t)$, $F_x(t)$,$F_y(t)$ and also provide the animation of this roly-poly toy with person standing on it.

3. Set $A_{mp} = 30°$, $T = 2.18s$, $\theta(t=0) = 0°$, plot $\theta(t)$, $\beta(t)$, $F_x(t)$,$F_y(t)$ and also provide the animation of this roly-poly toy with person standing on it.

Hint: After the force analysis, you may get:

$$\begin{aligned}
&\mathrm{F} - F_x = m_1 a_{Cx} \\
&\mathrm{N} - \mathrm{F_y} - m_1 g = m_1 a_{Cy} \\
&-(\mathrm{h}\sin\theta + \mathrm{a})\mathrm{N} + (\mathrm{r} - \mathrm{h}\sin\theta)\mathrm{F} + \mathrm{M} = \mathrm{I}_1\ddot{\theta} \\
&\mathrm{F_x} = \mathrm{m}_2\mathrm{a}_{\mathrm{Ex}} \\
&\mathrm{N} - \mathrm{m}_2 g = m_2 a_{Ey} \\
&0.5l\cos(\theta + \beta)\,\mathrm{F_x} + 0.5l\sin(\theta + \beta)\,F_y - M = I_2(\ddot{\theta} + \ddot{\beta})
\end{aligned}$$

Where M is the torque the upper part provides to the lower part, and further reduce these equations you may find the F_x and F_y satisfy:

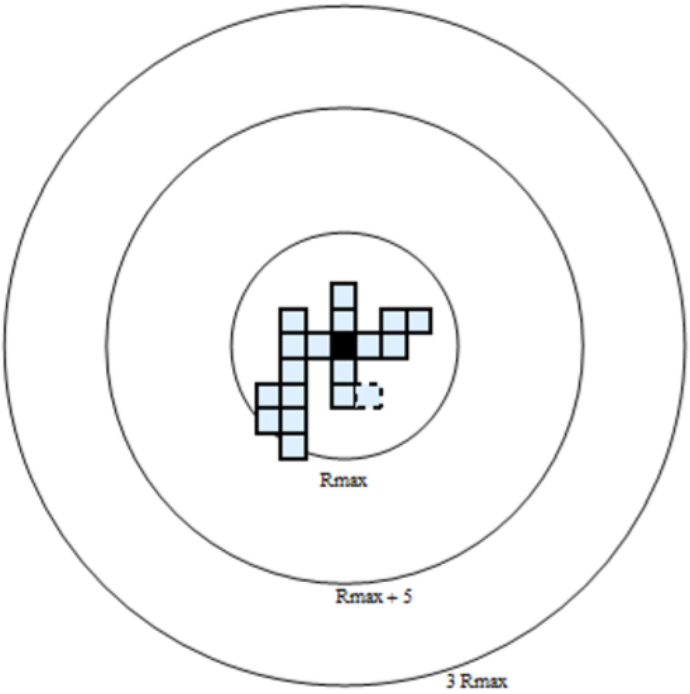
$$\begin{aligned}
\mathrm{F_x} = m_2(&-r\,\ddot{\theta} - 0.5l\cos(\theta + \beta)\left(\ddot{\theta} + \ddot{\beta}\right) + H\sin(\theta)\dot{\theta}^2 - H\cos(\theta)\ddot{\theta} \\
&+ 0.5l\sin(\theta + \beta)\left(\dot{\theta} + \dot{\beta}\right)^2) \\
\mathrm{F_y} = m_2(&g - 0.5l\sin(\theta + \beta)\left(\ddot{\theta} + \ddot{\beta}\right) - H\cos(\theta)\dot{\theta}^2 - H\sin(\theta)\ddot{\theta} \\
&- 0.5l\cos(\theta + \beta)\left(\dot{\theta} + \dot{\beta}\right)^2)
\end{aligned}$$

Project 2: Diffusion-limited aggregation（DLA）model

DLA is the process whereby particles undergoing a random walk due to Brownian motion cluster together to form aggregates of such particles. This theory, proposed by T.A. Witten Jr. and L.M. Sander in 1981, is applicable to aggregation in any system where diffusion is the primary means of transport in the system. DLA can be observed in many systems such as electrodeposition, Hele-Shaw flow, mineral deposits, and dielectric breakdown.

To simulate diffusion limited cluster aggregation process we can follow the model proposed by Witten and Sander and later developed by Meakin. The model and algorithm are based on the Monte Carlo method. Suppose we use the two dimensional simple cubic lattice. Particles are allowed to occupy only the lattice sites. Initially a seed particle is placed in the center of the system. Then the first particle is added to the lattice at sufficiently far distance from the seed and undergoes a random walk over the lattice via successive random jumps to one of the four nearest neighbor sites. Probability of a jump to any of the nearest neighbor sites is the same and normalized in order to have the overall jump probability equal to unity. Once the particle reaches one of the adjacent to the seed sites it could be attached to the seed according to the sticking probability at the nearest neighbor sites (P_{nn}) and the sticking probability at the second nearest neighbor sites (P_{snn}) . To simplify this question, we set $P_{nn} = 1$, and $P_{snn} = 0$ meaning that we only consider the nearest neighbor interaction.

During the simulation each of the first nearest neighbor sites is checked after each jump. If the moving particle sticks to the cluster at one of the nearest neighbor sites then the checking is stopped. Such approach is equivalent to the computation of the conditional probabilities for the moving particle to stick at each of the nearest sites, but it rather works faster if implemented in the simulation. So, if the particle doesn't sticks to the nearest neighbor sites for the first time, then it continues the random walk till it will be attached to the cluster or till it moves too far from the cluster. In later case particle is deleted from the lattice (”killed”) and starts off again from the closer distance to the cluster. After the first particle is finally attached to the cluster the second particle is introduced to the lattice and undergoes the same procedure. In such way the cluster is formed. We have started off the particles at the distance $R_{max} + 5$ from the cluster, where R_{max} is the maximum distance from the seed to the outermost particle in the cluster and all the distances are measured in the units of the lattice spacing. If the particle moves further then $3R_{max}$ from the cluster, it is killed and started off again at random position on the circle of radius $R_{max} + 5$ centered at the same point with the seed particle. Schematic depiction of this system can be found in the following figure. We have used different values for the radius of the start off circle and for the distance at which particle is killed. And the corresponding results were the same within a statistical error. To decrease the computational time the checking of the nearest neighbor sites is started if the particle reaches the distance $R_{max} + 2$ from the cluster.



1. Use the typical two dimensional cluster model and the algorithm above to grow $N = 10000$ particles. Plot the figure where the seed particle fixed at the center of the figure. The nearest sticking probability (P_{nn}) is 1 a here.

2. Using the same lattice model and algorithm to grow $N = 10000$ particles, now the nearest sticking probability P_{nn} is 0.3 here. Plot this figure.

3. Now we define the density-density correlation function: $C(r)$ is the number of particles at distance r from the reference particle for the all particles in the cluster and then normalizes it using the total number of particles and the corresponding volume between two spheres of radius $(r - \delta r)$ and $(r + \delta r)$ respectively. Please calculate the relationship of $ln(C(r))$ and $ln(r)$ for both sticking probability of ($P_{nn} = 1$ and $P_{nn} = 0.3$). Then use the averaging values over 20 clusters to plot the final relationship figures.

4. It could be shown that the density-density correlation function $C(r)$ follows the next power law dependency on r: $C(r) \sim r^{-\alpha}$, the fractal dimension D and the density function exponent α follows the relationship $D = d - \alpha$, where d is the Euclidian dimension. Calculate the fractal dimension for both $P_{nn} = 1$, and $P_{nn} = 0.3$.

The reference density-density correlation formula[1]:

$$\mathrm{C(r)} = N^{-1} \sum_{\{r'\}} \rho(r')\rho(r' + r)$$

[1] Witten Jr T A, Sander L M. Diffusion-limited aggregation, a kinetic critical phenomenon[J]. Physical review letters, 1981, 47(19): 1400.

Project 3: Planetary orbits

In your assignment 10, you have plot the orbits of the Earth and Pluto, but we set the Sun is the fixed point because of its great quality. Now we consider something more general. The gravity between two bodies is always

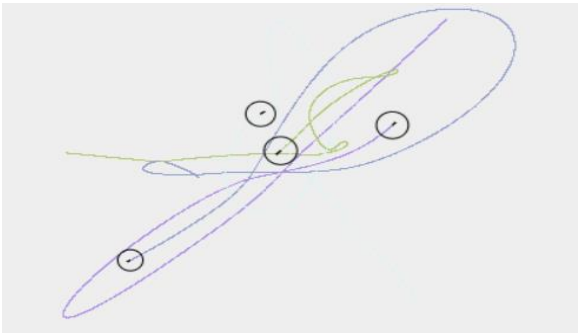
$$F = m\frac{d^2\vec{r}}{dt^2} = -(\frac{GMm}{r^2})\frac{\vec{r}}{r}$$

, where $G = 6.6738 \times 10^{-11}m^3kg^{-1}s^{-2}$ is Newton's gravitational constant, $M = 1.9891 \times 10^{30}kg$ is the mass of the Sun, and $r = \sqrt{x^2 + y^2}$ in 2D and $\sqrt{x^2 + y^2 + z^2}$ in 3D. In the follow question, you can try to consider solve them by different method to compare them.

1. Use the same initial condition (i.e. mass and initial condition) in Assignment 10, the Earth is at its closest approach to the Sun, its perihelion, it is moving precisely tangentially (i.e., perpendicular to the line between itself and the Sun) and it has distance $1.4710 \times 10^{11}m$ from the Sun and linear velocity: $3.0287 \times 10^4 m/s$. The distance between the Sun and Pluto at perihelion is 4.4368×10^{12} m and the linear velocity is $6.1218 \times 10^3 m/s$.

   Now we assume the mass of the Sun is the same with the Earth, and its initial velocity is zero. That means the sun **cannot be considered as a fix point**. Plot the orbits of the Sun with Earth, and the Sun with Pluto.

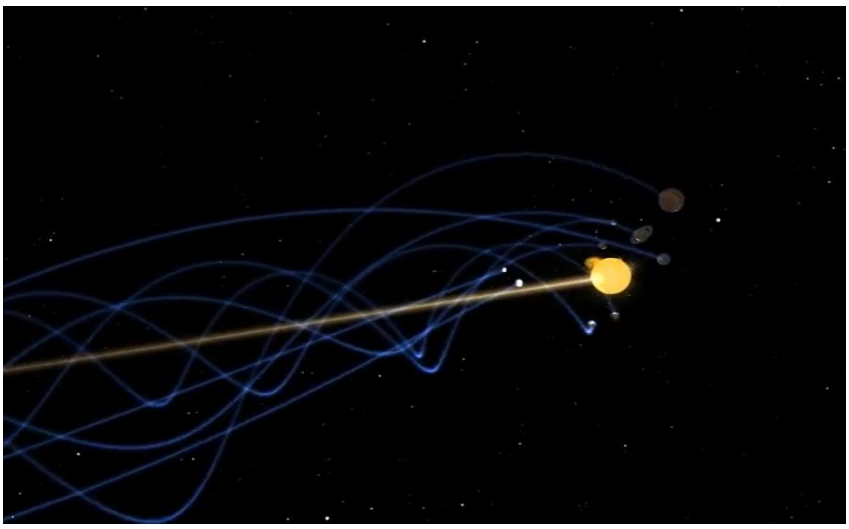
2. Then, consider the three-body problem: There are three Suns and one Planet (Earth) in the galaxy. The mass of Suns should **much bigger than** the Planet (you can just use the mass of our Sun), but similar with each other. Choose any three different sets of initial conditions and plot their orbit. Will they eventually approach the same? For the same initial condition, if the time is long enough, is the trajectory the same?



3. Now improve your code to simulate a Solar system model. You can choose multiple planets of the solar system. The initial condition you can set like this: Planets along the same line and all are at their perihelion (nearest points to the sun). Their initial velocities relative to the Sun are perpendicular to the line between itself and the Sun. You can find their linear velocity relative to the sun in the below table. Give the Sun a nonzero initial velocity. Plot in **3D** figure.

Planet	perihelion (m)	Linear velocity(m/s)
Mercury	$4.60017 * 10^{10}$	53711.756

Venus	$1.0747158 * 10^{11}$	35140.627
Earth	$1.4708 * 10^{11}$	30038.588
Mars	$2.066764 * 10^{11}$	25340.2493
Jupiter	$7.408839 * 10^{11}$	13312.576
Saturn	$1.35569 * 10^{12}$	9894.096
Uranus	$2.74745 * 10^{12}$	6950.11
Neptune	$4.49574 * 10^{12}$	5433.20



Project 4: Defend system

Follow our Cannon ball problem in the Lec18, let us design a defend system.

The cannon ball should consider air drag. Now set its mass as $1000kg$, the acceleration due to gravity is $g = 9.8m/s^2$, the Radius of the spherical cannonball is $R = 0.08m$, the air density is $\rho = 1.22$, and the coefficient of drag is $C = 0.47$.And the air resistance on a moving sphere is a force in the opposite direction to the motion with magnitude

$$\overrightarrow{f_d} = -\frac{1}{2}\pi R^2 \rho C v \vec{v}$$

1. Firstly, just generate it into three dimensions. Now the z axis represents the height. Then find and plot a trajectory which can hit the plane with the initial condition like this:

   You are located at zero point, the plane is found at $(500, -500, 700)m$, its velocity is $(-10, 50, 0)m/s$ with a acceleration $(-5, 4, 4)m/s^2$.

   Now use your code to find a trajectory can hit the plane and plot it in 3D figure.

2. Now assume the cannon ball has a maximum initial speed $200m/s$. Then improve the code again to find the trajectory that can hit the plane **first**. Plot the trajectory from the above initial condition of plane. The trajectory should be in one 3D figure.

3. Now consider the cannon ball is a rocket, its shape, mass and air drag formula will not change. However, now it will receive constant thrust. The thrust force is $25000N$, its direction is always the same with the velocity. Now the maximum initial speed is set as $100m/s$. Improve your code to find and plot the first trajectory can hit the plane with this initial state:

   You are located at zero point, the plane is found at $(5000, -5000, 7000)m$, its velocity is $(-50, 50, 0)m/s$ with a acceleration $(4, 4, 4)m/s^2$.

4. A defend system should be able to detect and measure the Motion parameters of the plane by itself. Now assume your detection system can only detect the position of the aircraft every 0.1 second (i.e. you can only input the position with corresponding time of the plane in your code, the velocity and acceleration are unknown). And the system can immediately calculate the aircraft's motion parameters and launch rockets at the same time.

   Please design a detection system and simulate one trajectory to hit a plane. You should explain your idea about the detection system. And then you need plot the trajectory of the rocket and the track of the plane in 3D figure, and show the information of the plane detected by the system and the motion parameters calculated by it.

   Note: The simulation should satisfy this: you type a function to output positions of the plane with the time, then your defend system receive this information to launch attack. You can use the above parameters of plane, or you can choose any

sets of initial position, velocity and acceleration to simulation, but you can only input the positions with the corresponding time into the detection system.

Hint: In all questions, the plane will not change its acceleration after it is detected for the first time.

Through my test, a good initial condition is especially important when the initial velocity of the solution is small. My suggestion is that you can choose the initial velocity obtained in the previous step (previous "hit time (tf)") as the initial condition for this time. But the Newton method may still fail to converge when the initial velocity of the solution is too small (or the "hit time" is too long). Change your parameters of the plane to avoid it.