Lecture 1
ROB-GY 7863 / CSCI-GA 3033 7863: Planning, Learning, and Control for Space Robotics

Benjamin Riviere

September 9, 2025

Today's Agenda

- Introductions
- Course Logistics
- Course Overview
- First Lecture: Spacecraft Dynamics and Control

Introductions

- Benjamin Riviere is the course instructor:
  - Assistant Professor in Mechanical Engineering and Computer Science at NYU. Before that, MS/PhD/Postdoc at California Institute of Technology in Aerospace and Computational and Mathematical Sciences and BS at Stanford University in Mechanical Engineering.
  - Contact information: `riviere.b@nyu.edu`
- Rika Valluri is the course assistant (CA):
  - MS student in Robotics at NYU. Prior work at Indian Space Research Organisation (ISRO).
  - Contact information: `srv297@nyu.edu`

Course Logistics

- We use Brightspace for syllabus, lectures and notifications, please make sure you are have access.
- We meet weekly on Mondays from 2:00 PM - 4:30 PM in Jacobs Hall, 6 Metrotech Room 211, Brooklyn Campus
- The grading breakdown is as follows:
  - Attendance (10 %) - There may be small in-person quizzes to check understanding for feedback, but mostly if you show up you get points.
  - Project 1 (40 %) - Implement a physics-based simulation of a (simplified) space robot mission and write a report.
  - Project 2 (50 %) - Design an autonomy algorithm (control, planning, or learning) for that mission and write a report and do a presentation.
- Prerequisites:
  - Math - Linear Algebra, Optimization, Differential Equations, Probability and Statistics
  - **Programming** - Preferably Python.

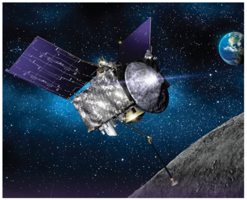
Class Goals

- To introduce mathematical fundamentals of planning, learning, and control for robotics.
- To get experience with modern software tools for robotics
- To get excited about space robotic applications!
- Deliverable: Cool demo video of your project that you can post / talk about / show people as part of your portfolio.

Class Policies

- Generative AI:
  - The risk of generative AI (Chat GPT, Gemini, etc) is learning loss, not cheating.
  - The course policy is that you can use it and I actually encourage it in moderation.
  - Keep in mind that your project grades (majority of grade) will be determined by your ability to communicate and demonstrate understanding. If you over rely on it, it will probably be obvious and you will get a bad grade.
- Typos:
  - If you catch a (conceptual) typo, you get some extra credit. Please email me.

Course Overview


(a) Spacecraft and Dynamics/ Control (4-5)

(b) Rocket and Planning/ Optimization (4-5)

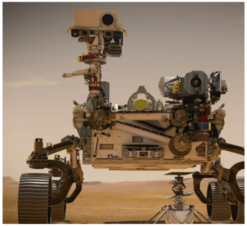

(c) Rovers and Reinforcement Learning (4-5)

Figure: Course Chapters (Estimated Number of Lectures Per Topic.)

- Under the hood, these seemingly different robots have a lot in common, especially wrt the **math for their autonomy**

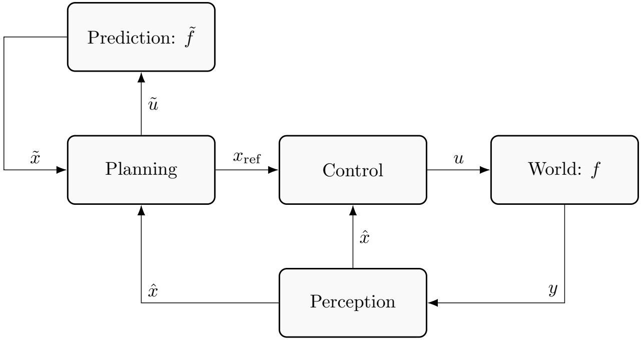
Today's Agenda for Spacecraft and Dynamics/Control

- Example Spacecraft Mission (Application)
- Spacecraft Dynamics Model (Theory)
- General Robotics Dynamics Model (Theory)
- Physics-Based Simulators (Application)
- Comparison Lemmas (Theory)
- Nonlinear Control - Lyapunov (Theory)
- Spacecraft Control (Application)
- Goals:
  - To introduce dynamics models for spacecraft and general robot systems
  - To introduce tools for closed loop analysis

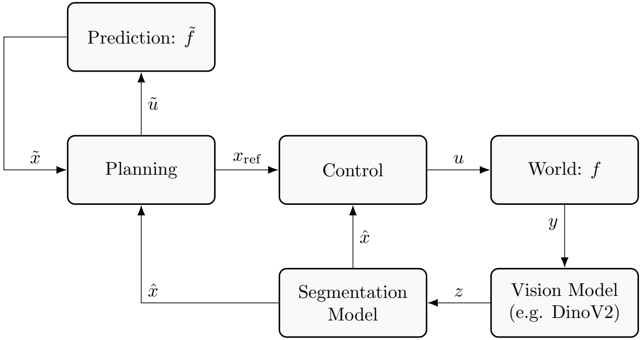
Example Spacecraft Mission: OSIRIS-REx

- OSIRIS-REx Mission: (remember to scroll to bottom)
- Mass: 1,900 kg
- Sensors:
  - Inertial Measurement Unit (IMU)
  - Star trackers
  - Sun sensors
  - GN&C lidar and Camera for TAG mission
  - (Most LEO satellites have GPS)
- Actuators:
  - Thrusters: main engine, trajectory correction, attitude control, low thrust
  - Reaction Wheels: (4x)

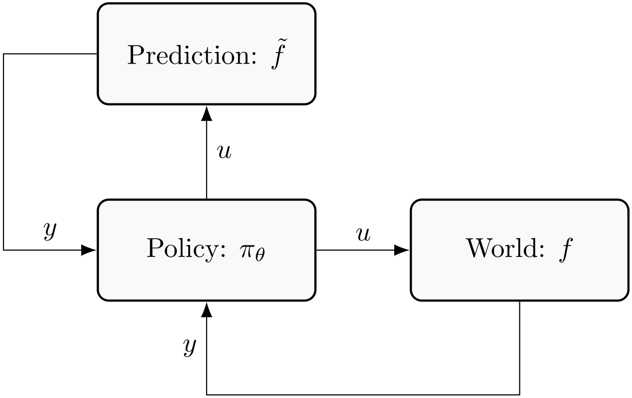
"Classical" Autonomy Stack



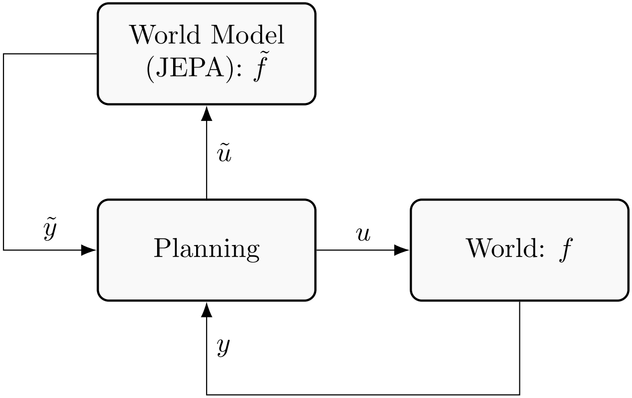
Vision Model Stack



End-to-End RL Stack


$$\begin{tikzcd}
\boxed{\text{Prediction: } \tilde{f}} \arrow[d, bend right, "y"'] & \\
\boxed{\text{Policy: } \pi_\theta} \arrow[u, "u"] \arrow[r, "u"] & \boxed{\text{World: } f} \arrow[l, bend left, "y"]
\end{tikzcd}$$

Planning with World Model Stack



Significance of Evolving Stack

- We will understand all of these components, and also how they interact.
- All of these loops can be analyzed with similar tools:
  - **control** of physical state x
  - **planning** optimization primal variables $\tilde{x}$
  - **reinforcement learning** parameter updates θ

Notation

I will try to consistently use this notation, within reason:

- Let $q \in \mathbb{R}^{d_q}$ be the generalized positions
- Let $\dot{q} \in \mathbb{R}^{d_{\dot{q}}}$ be the generalized velocities
- Let $x \in \mathbb{R}^{d_x}$ be the state vector
- Let $u \in \mathbb{R}^{d_m}$ be the control vector
- Let $y \in \mathbb{R}^{d_y}$ be the sensor data
- Let $\hat{x} \in \mathbb{R}^{d_x}$ be the estimated state
- Let $x_{\text{ref}} \in \mathbb{R}^{d_x}$ be the reference state
- Let f be the true dynamics, depending on context: $\dot{x} = f(x, u)$ or $x_{k+1} = f(x_k, u_{k+1})$ or $f(x_k, u_{k+1}) = p(x_{k+1}|x_k, u_{k+1})$
- Let $\tilde{f}$ be the simulated dynamics, $\tilde{f}(x, u) \approx f(x, u)$
- Let h be the observation function, depending on context: $y = h(x)$ or $h(x) = p(y|x)$
- Let π_θ be the policy model, $\pi_\theta(x) = u$ or $\pi_\theta(x) = p_\theta(u|x)$

Spacecraft Dynamics Model

- We want a dynamics model for spacecraft that looks like this:

$$\dot{x} = f(x, u) \tag{1}$$

- Let $x = [r_x, r_y, r_z, \phi, \theta, \psi, v_x, v_y, v_z, p, \mathrm{q}, r]$ be the state, $u = [\mathbf{f}_{\text{ext}}, \tau_{\text{ext}}]$ be the control.
- We are going to derive this two ways: (i) Newton-Euler and (ii) Euler-Lagrange.
- **TODO:** Picture of the Earth-centered Earth-fixed (ECEF) frame and Body-fixed frame.

Spacecraft Dynamics Model - Translation

- The translation part is the easier of the two:
- Let $\mathbf{r} = [r_x, r_y, r_z] \in \mathbb{R}^3$ be the position of the spacecraft in inertial frame, let $\mathbf{v} = [v_x, v_y, v_z] \in \mathbb{R}^3$ be the velocities in inertial frame.
- We apply Newton's second law ($F = ma$) with gravity and external forces:

$$\dot{\mathbf{r}} = \mathbf{v} \tag{2}$$

$$\dot{\mathbf{v}} = \sum_{i \in \text{bodies}} \frac{G m_i (\mathbf{r}_i - \mathbf{r})}{\|(\mathbf{r}_i - \mathbf{r})\|^3} + \frac{1}{m}\mathbf{f}_{\text{ext}} \tag{3}$$

where G is a fundamental constant, m_i is the mass of the body and $\mathbf{r}_i$ is the position of the body in inertial frame.

Spacecraft Dynamics Model - Attitude I

- The rotation part is harder because dynamics are simple in the body frame, but we also have to keep track of the rotation from body frame to inertial frame.
- Lets start with dynamics: let $\omega = [p, \mathrm{q}, r]$ where p, q, r are the rotation rate about the body frame's x, y, z axis. Then, we apply Newton's second law on a moving frame:

$$I\dot{\omega} + \underbrace{\omega \times (I\omega)}_{\text{gyroscopic coupling}} = \tau_{\text{ext}} \tag{4}$$

- Gyroscopic coupling comes from taking derivative in a moving frame. A sphere experiences no gyroscopic coupling force. Why?

Spacecraft Dynamics Model - Attitude II

- The **cross product** of two vectors is a third, perpendicular vector. The cross product can be written as a matrix via the "hat operator":

$$[\omega\times] = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{5}$$

Spacecraft Dynamics Model - Rotation Matrices I

- Next, we talk about the rotation from body frame to inertial frame. This is done with an element of $SO(3)$, and there are a few choices for attitude representation: Euler angles, quaternion, etc.

$$R \in SO(3) = \{R \in \mathbb{R}^{3\times 3} | R^T R = I, \det(R) = 1\} \tag{6}$$

Spacecraft Dynamics Model - Rotation Matrices II

- What is an **orthonormal matrix**?
  - A matrix is orthonormal if a matrix times its transpose is identity: $R^T R = I$
  - It's called an orthonormal matrix, because the columns (and rows) form an orthonormal basis in $\mathbb{R}^n$.
  - It preserves lengths and angles:

$$\|Rx\| = \sqrt{x^T R^T R x} = \sqrt{x^T x} = \|x\| \tag{7}$$

$$\langle Rx, Ry \rangle = x^T R^T R y = x^T y = \langle x, y \rangle \tag{8}$$

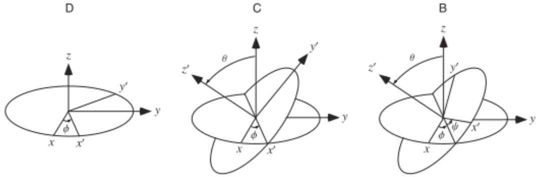
    In other words, it is a rigid body transformation, either a rotation or a reflection.
  - Bonus: All eigenvalues lie on the unit circle and all eigenvalues come in complex-conjugate pairs

Spacecraft Dynamics Model - Rotation Matrices III

- What is the determinant? For square matrices, **determinant** is the product of eigenvalues. Intuitively, it is the measure of signed volume scaling of the transformation. So det $= 1$ means no scaling and preserves orientation (no reflection).

Spacecraft Dynamics Model - Euler Angles I

- To parameterize R, we use Euler angles: $\Theta = [\phi, \theta, \psi]$ are the roll, pitch, yaw. We use the 3–2–1 intrinsic convention: 3-2-1 denotes the order of rotation z then y then x and intrinsic means that you are always rotating about the "new" coordinate frame:



Spacecraft Dynamics Model - Euler Angles II

- From Euler angles and the selected convention, we can compute a rotation matrix:

$$R = R_x(\phi)R_y(\theta)R_z(\psi) \tag{9}$$

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \tag{10}$$

$$R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{11}$$

$$R_z(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{12}$$

Spacecraft Dynamics Model - Euler Kinematic Transformation Matrix I

- Almost there. Last step before writing equations is to build the kinematic relationship between the body rates and the derivatives of the Euler angles: we want something like $\omega = B(\Theta)\dot{\Theta}$.
- Lets take the time derivative of orthogonal condition: $R^T R = I$.

$$\dot{R}^T R + R^T \dot{R} = 0 \tag{13}$$

Spacecraft Dynamics Model - Euler Kinematic Transformation Matrix II

- What is a **skew symmetric matrix**?
  - A skew symmetric matrix is equal to the negative of its transpose, $A = -A^T$
  - For every skew symmetric matrix, there exists a vector for which the matrix transformation is equivalent to a cross product: $Ax = v \times x$.
  - Intuitively, skew symmetric matrices could be thought of as "time derivatives" of rotation matrices, (but really they are elements of the tangent space of $SO(3)$).
  - The spectrum of skew symmetric matrices: all eigenvalues are purely imaginary, all eigenvalues come in complex conjugate pairs.
  - What is $x^T Ax$ if $A = -A^T$?

Spacecraft Dynamics Model - Euler Kinematic Transformation Matrix III

- We will show that ω is the corresponding vector for $\dot{R}^T R$. Let x^B be any fixed-body frame vector, then $x^I = Rx^B$. Compute the derivative of x^I two ways, set them equal to each other, and left multiply by R^T:

$$\begin{aligned}
\dot{x}^I &= \dot{R}x^B && (14)\\
\dot{x}^I &= w^I \times x^I && (15)\\
R^T \dot{R} x^B &= R^T (w^I \times x^I) && (16)\\
&= R^T R w^B \times R^T R x^B = w^B \times x^B && (17)
\end{aligned}$$

- Recall previous fact of cross product and hat operator. If we manipulate the expression $[\omega\times] = R^T \dot{R}$, we can write $w = B(\Theta)\dot{\Theta}$, where $B^{-1}(\Theta)$ is the matrix we are interested in.

Spacecraft Dynamics Model - Euler Kinematic Transformation Matrix IV

- For completeness, here is $B^{-1}(\Theta)$:

$$B^{-1}(\Theta) = \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos\phi\tan\theta \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix} \qquad (18)$$

- Gimbal lock is when B is not invertible. This is a consequence of choice of Euler angle representation of $SO(3)$.

Spacecraft Dynamics

- Then, we have complete system:

$$\dot{\mathbf{r}} = \mathbf{v} \tag{19}$$

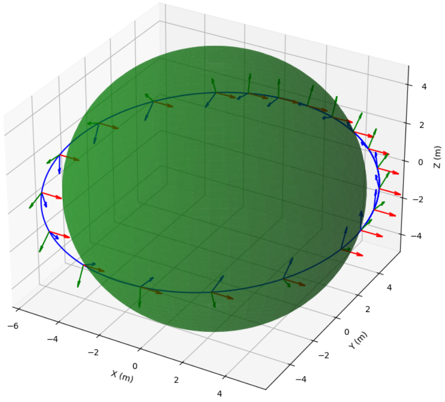
$$\dot{\mathbf{v}} = \sum_{i \in \text{bodies}} \frac{G m_i (\mathbf{r}_i - \mathbf{r})}{\|(\mathbf{r}_i - \mathbf{r})\|^3} + \frac{1}{m}\mathbf{f}_{\text{ext}} \tag{20}$$

$$\dot{\Theta} = B(\Theta)^{-1}\omega \tag{21}$$

$$\dot{\omega} = I^{-1}(-\omega \times (I\omega) + \tau_{\text{ext}}) \tag{22}$$

Spacecraft Simulation

- Implemented dynamics in Python on class github. Find bugs? or do extensions?



Lagrangian Mechanics I

- We just did the **Newton-Euler** derivation of spacecraft dynamics in a particular frame.
- Instead, we can do the **Lagrange** derivation, which is more general (coordinate free) and is at the core of most physics-based simulators.
- The core object of Euler-Lagrangian derivation is the Lagrangian:

$$L(q, \dot{q}) = T(q, \dot{q}) - V(q) \tag{23}$$

- Why? Option 1: Reverse engineering to whatever function applying variational calculus to collapses to Newton's laws. Option 2: Veritasium has a nice video

Lagrangian Mechanics II

- From Brown, 2013: "The principle of least action – really the principle of stationary action – is the most compact form of the classical laws of physics. This simple rule (it can be written in a single line) summarizes everything! Not only the principles of classical mechanics, but electromagnetism, general relativity, quantum mechanics, everything known about chemistry – right down to the ultimate known constituents of matter, elementary particles."

Principle of Least Action and Calculus of Variations I

- The principle of least action says that physical trajectories are the extremum of the action:

$$S[q] = \int_{t_0}^{t_1} L(q, \dot{q})\, dt \tag{24}$$

  where the action S is a scalar-valued function of a function (a functional), because q is a continuous function of t.

- For a function on vector space, an extremum is a local minima or maxima and it is where the derivative to zero. For example, for scalar function $f : \mathbb{R} \to \mathbb{R}$:

$$\nabla_x f = \frac{d}{dx} f = \lim_{\epsilon \to 0} \frac{f(x+\epsilon) - f(x)}{\epsilon} = 0 \tag{25}$$

Principle of Least Action and Calculus of Variations II

- For functionals, we need a new derivative concept called a **variation** or a Gateaux derivative. Let $q : [t_0, t_1] \to \mathbb{R}^n$ and $\delta q : [t_0, t_1] \to \mathbb{R}^n$ where $\delta q(t_0) = \delta q(t_1) = 0$.

$$q_\epsilon(t) = q(t) + \epsilon \delta q(t) \tag{26}$$

$$\delta S[q] := \frac{d}{d\epsilon} S[q_\epsilon] = \lim_{\epsilon \to 0} \frac{S[q_\epsilon] - S[q]}{\epsilon} \tag{27}$$

- **TODO:** picture

Principle of Least Action and Calculus of Variations III

- With this transformation, we are in good shape because we can now apply "normal" derivative rules like chain rule and integration by parts:

$$\delta S[q] = \frac{d}{d\epsilon}\int_{t_0}^{t_1} L(q_\epsilon, \dot{q}_\epsilon)\,dt = \int_{t_0}^{t_1} \frac{\partial L}{\partial q}\frac{dq_\epsilon}{d\epsilon} + \frac{\partial L}{\partial \dot{q}}\frac{d\dot{q}_\epsilon}{d\epsilon}\,dt \tag{28}$$

$$= \int_{t_0}^{t_1} \frac{\partial L}{\partial q}\delta q\,dt + \left.\frac{\partial L}{\partial \dot{q}}\delta q\right|_{t_0}^{t_1} - \int_{t_0}^{t_1} \frac{d}{dt}\frac{\partial L}{\partial \dot{q}}\delta q\,dt \tag{29}$$

$$= \int_{t_0}^{t_1} \left(\frac{\partial L}{\partial q} - \frac{d}{dt}\frac{\partial L}{\partial \dot{q}}\right)\delta q\,dt = 0 \tag{30}$$

- Because it must hold true for all δq, the integrand must always be zero, and that is the Euler-Lagrange Equation.

Principle of Virtual Work

- Framework can be extended to external forces with principle of virtual work ($\delta W = (Bu)^T \delta q$):

$$\delta S[q] + \int_{t_0}^{t_1} \delta W dt = 0 \tag{31}$$

- Get forced equations:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = Bu \tag{32}$$

Robot Manipulator Equation - Derivation I

- Define Potential and Kinetic Energy:

$$T(q, \dot{q}) = \frac{1}{2}\dot{q}^T M(q)\dot{q} \tag{33}$$
$$\nabla_q V(q) = G(q) \tag{34}$$

- Take derivatives in Euler Lagrange equation:

$$\frac{\partial T}{\partial q} = \frac{1}{2}\dot{q}^T \frac{\partial M}{\partial q}\dot{q}, \quad \frac{\partial V}{\partial q} = G(q), \quad \frac{\partial T}{\partial \dot{q}} = M(q)\dot{q} \tag{35}$$
$$\frac{\partial V}{\partial \dot{q}} = 0, \quad \frac{d}{dt}\frac{\partial T}{\partial \dot{q}} = \dot{M}(q)\dot{q} + M(q)\ddot{q} \tag{36}$$

Robot Manipulator Equation - Derivation II

- Plug in terms into Euler Lagrange equation (Christoffel form of C):

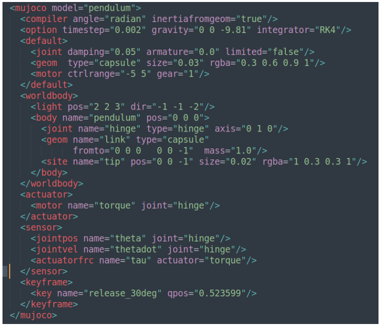
$$M(q)\ddot{q} + \dot{M}(q)\dot{q} - \frac{1}{2}\dot{q}^T \frac{\partial M}{\partial q}\dot{q} + G(q) = Bu \tag{37}$$

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = Bu \tag{38}$$

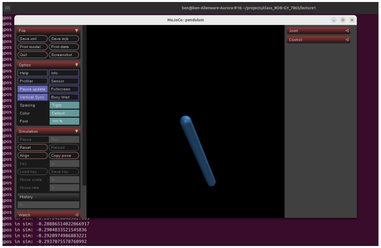
- $M(q)$ is a positive definite symmetric matrix, $M(q) \in \mathbb{S}^n_{>0} = \{M \in \mathbb{R}^{n\times n} \mid M = M^T \succ 0\}$:
  - What does it mean to be symmetric? All real eigenvalues and real part of eigenvalues scales length.
  - What does it mean to be positive definite? $x^T Mx > 0 \ \forall x$. All eigenvalues are positive. Always invertible.
  - Very useful: $\lambda_{\min}(M)\|x\|^2 \leq x^T Mx \leq \lambda_{\max}(M)\|x\|^2$
- "Conservation of Energy": $\dot{M}(q) - 2C(q,\dot{q})$ is skew symmetric.
- Other important external forces are contact and damping: $J(q)^T\lambda + D\dot{q}$

Physics-Based Simulators

- These are the equations that govern physics-based simulators like Mujoco and IsaacSim.
- Usually, the interface we provide is an xml file.
- Mujoco example on class github



```xml
<mujoco model="pendulum">
  <compiler angle="radian" inertiafromgeom="true"/>
  <option timestep="0.002" gravity="0 0 -9.81" integrator="RK4"/>
  <default>
    <joint damping="0.05" armature="0.0" limited="false"/>
    <geom  type="capsule" size="0.03" rgba="0.3 0.6 0.9 1"/>
    <motor ctrlrange="-5 5" gear="1"/>
  </default>
  <worldbody>
    <light pos="2 2 3" dir="-1 -1 -2"/>
    <body name="pendulum" pos="0 0 0">
      <joint name="hinge" type="hinge" axis="0 1 0"/>
      <geom name="link" type="capsule"
            fromto="0 0 0   0 0 -1"  mass="1.0"/>
      <site name="tip" pos="0 0 -1" size="0.02" rgba="1 0.3 0.3 1"/>
    </body>
  </worldbody>
  <actuator>
    <motor name="torque" joint="hinge"/>
  </actuator>
  <sensor>
    <jointpos name="theta" joint="hinge"/>
    <jointvel name="thetadot" joint="hinge"/>
    <actuatorfrc name="tau" actuator="torque"/>
  </sensor>
  <keyframe>
    <key name="release_30deg" qpos="0.523599"/>
  </keyframe>
</mujoco>
```



```
qpos in sim: -0.28886514022066917
qpos in sim: -0.2904833521545836
qpos in sim: -0.2920974986883221
qpos in sim: -0.2937075578760992
```

Notes on Project 1

- Project 1 is to implement a physics-based simulation for a mission.
- Good Options:
  - Manually implementation
  - Isaac Sim (computational specifications are a limitation, limited capacity for those who don't have access to sufficient compute)
  - Mujoco (easier to install)
  - Basilisk (specialized astrodynamics software, and there is a Basilisk RL)
  - Please contact me if you have other ideas
- Pick a mission that will be compatible with Project 2, which is to design an autonomous solution for that mission.

Break

- Recap so far: we just derived general forms of f and talked about how we implement them in practice.
- Next, we will study the control loop and develop tools to understand whether these loops are "behaving correctly".
- Break

Tracking Control Problem I

- We start with a control dynamical system:

$$\dot{x} = f(x, u) \tag{39}$$

- In the tracking control problem, we assume we are given a desired trajectory for the system: $x^{\text{ref}}(t)$, $u^{\text{ref}}(t)$ such that: $\dot{x}^{\text{ref}} = f(x^{\text{ref}}, u^{\text{ref}})$.
- Goal: construct a controller u to get exponential stability:

$$\|x(t) - x^{\text{ref}}(t)\| \leq Me^{-\lambda t}\|x(0) - x^{\text{ref}}(0)\| \tag{40}$$

- **TODO:** draw picture

Tracking Control Problem II

- Special cases are linear systems and regulation:

$$f(x,u) = Ax + Bu \qquad \text{Linear Systems}$$

$$x^{\text{ref}}(t) = x^{\text{ref}},\ u^{\text{ref}}(t) = 0,\ f(x^{\text{ref}}, 0) = 0 \qquad \text{Regulation}$$

Recap of Today

- To introduce the course logistics and overview
- To introduce dynamics models for spacecraft and general robot systems:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = Bu \tag{41}$$

Significance: the general robot dynamics are the core of every modern physics simulator. Important for our overview picture of autonomy to understand what f is.

Recap of Facts

- We learned about mathematical structures: orthonormal matrices, determinants, dot products, norms, skew symmetric matrices, cross products, symmetric matrices, etc.
- We learned about how they exist in kinematics, dynamics, and stability.
- Significance: We don't live in a random unstructured world! Also understanding and exploiting structure will be useful for autonomy algorithm design.
- We also learned about stable and unstable iterations and ODEs, and how we can compare our systems against those.

Upcoming Topics

- Contraction Theory
- Robustness to Learned Model Error
- Adaption to unknown parameters (Adaptive Control)
- Applications:
  - Hubble Spacecraft Pointing Control
  - Spacecraft Swarms: Distributed Apertures
- Optimal Control:
  - Orbital Maneuvers
  - In-Orbit Self Assembly

For next time

- Check that you have access to Brightspace.
- Check out the pdf notes from today online.
- Check out class github, setup environment, and run sample scripts.
- Read/listen to something about space robots and be ready to talk about it, e.g. Space Policy Podcast.
- Start thinking about what you want your project to be.
- If you want to use an external simulator for Project 1, try doing the software installation / compatibility / unit tests as early as possible.

References

Brown, Robert G. (June 2013). “The Theoretical Minimum: What You Need to Know to Start Doing Physics”. In: *Physics Today* 66.6, pp. 54–55. ISSN: 0031-9228. DOI: 10.1063/PT.3.2015. URL: https://doi.org/10.1063/PT.3.2015 (cit. on p. 32).