Advanced Dynamics

David Akre

February 12, 2025

Abstract

This document is a collection of notes from MCEN 5228 at CU-Boulder as well as references from various textbooks (i.e., Classical Mechanics by Goldestein).

1 Newton's Mechanics

1.1 Mechanics of a Particle

Newton's mechanics describes the motion of particles or point masses, where a particle configuration can be completely described by its position vector $\vec{r}$ (i.e., an abstract vector... AKA free vector). Thus the coordinate free form of Newton's second law is: $\vec{F} = \frac{d}{dt}(m\frac{d\vec{r}}{dt}) = \frac{d}{dt}(m\vec{v}) = \frac{d\vec{p}}{dt}$ where m is the mass, $\vec{v}$ is the velocity and $\vec{p}$ is the momentum.

Conservation Theorem for the Linear Momentum of a Particle: If the total force $F = 0$ then $\dot{p} = 0$ and the linear momentum is conserved. For a constant mass m the force vector is described in the following manner:

$$\vec{F} = \frac{d}{dt}(m\vec{v}) = m\frac{d\vec{v}}{dt} = m\vec{a} \tag{1}$$

Where $\vec{a}$ is the acceleration vector. However we can't don anything with the above until we express the above in some coordinate system.

As an example (i.e., 2D Cartesian Coordinates) we can express the point mass as the following:

$$\vec{r} = r_x\hat{e}_x + r_y\hat{e}_y$$
$$\vec{r} = \begin{bmatrix} r_x & r_y \end{bmatrix} \begin{bmatrix} \hat{e}_x \\ \hat{e}_y \end{bmatrix}$$

Where $\begin{bmatrix} r_x & r_y \end{bmatrix}$ are scalar components and $\begin{bmatrix} \hat{e}_x & \hat{e}_y \end{bmatrix}^T$ are unit basis vectors.

$$\vec{v} = \frac{d\vec{r}}{dt} = \dot{r}_x\hat{e}_x + r_x\dot{\hat{e}}_x + \dot{r}_y\hat{e}_y + r_y\dot{\hat{e}}_y$$

Described the velocity of the particle, and if the coordinate system is not moving their respective derivatives equate to 0. We can continute taking the second derivative of the position vector of the point mass to come to a generalized form of Newton's law:

$$\begin{bmatrix} F_x & F_y \end{bmatrix} \begin{bmatrix} \hat{e}_x \\ \hat{e}_y \end{bmatrix} = m \begin{bmatrix} \ddot{r}_x & \ddot{r}_y \end{bmatrix} \begin{bmatrix} \hat{e}_x \\ \hat{e}_y \end{bmatrix}$$

Likewise, the above can be expressed in any coordinate system (i.e., polar in 2D as another example). **Newton's formulation is not coordinate invariant**, thus equations of motion can become very unwieldy to write down as things get more complicated. See 2D inverted pendulum as example (TODO may show here).

The **angular momentum (L)** of a particle about point O in the following fashion:

$$L = r \times p \tag{2}$$

Where r is the radius vector from O to the particle and the **moment of force or torque (N)** about O is:

$$N = r \times F \tag{3}$$

Through derivation we can see $\dot{L} = N$ which leads to the **Conservation Theorem for the Angular Momentum of a Particle**: If the total torque $N = 0$ then $\dot{L} = 0$ and the angular momentum L is conserved.

Next thing to consider is the work done by an external force F upon a particle from point 1 to 2.

$$W_{12} = \int_1^2 F \cdot ds \rightarrow \frac{m}{2} \int \frac{d}{dt}(v^2)dt \rightarrow W_{12} = \frac{m}{2}(v_2^2 - v_1^2) \tag{4}$$

The **kinetic energy** is described as:

$$T = \frac{m}{2}\vec{v} \cdot \vec{v} \tag{5}$$

Thus we can see the relationship from work done is equal to the change in kinetic energy:

$$W_{12} = T_2 - T_1 \tag{6}$$

The **potential energy** of the particle can be described as:

$$F = -\nabla U(r) \tag{7}$$

Where F (force) is the gradient of some scalar function of position and V is the potential energy. For example gravity has the following potential $U_g = mgz$ where z is height and g is the graviational force acting on mass m. Another example is the potential of a spring force $F_{spring} = -kx \rightarrow -\frac{d}{dx}U_{spring}(x)$ where $U_{spring} = \frac{1}{2}kx^2$.

The relationship between work done and potential energy is the following:

$$W_{12} = U_1 - U_2 \tag{8}$$

Intuitively we can think of kinetic energy as the particle in motion and potential energy as "how much" could the particle move based on its current position. So we can see if we have maximal potential energy the kinetic energy will be 0 and vice versa, which leads to the formulation of the **Energy Conservation Theorem for a Particle**: If the forces acting on a particle are conservative, then the total energy of the particle, $T + V$ is conserved.

$$T_1 + U_1 = T_2 + U_2 \tag{9}$$

The total energy in the system is then described by:

$$E = T + U \tag{10}$$

Forces that can be derived from a potential are always **conserved** since they don't change the total energy. Thus, friction, drag, damping, etc are all non-conservative and they dependent on U and the change in E.

1.2 Rigid Body Kinematics

Rigid body can be defined as a system of particles in which the distances r_{ij} are fixed and cannot vary with time, the internal forces do no work, and the internal potential must remain constant.

$$||r_i - r_j||_2 = c_{ij} \text{ where } i, j \in [1, ..., N] \tag{11}$$

Where r_i and r_j are positions in $\mathbb{R}^3$ and c_{ij} is a distance in $\mathbb{R}$. This is a useful approximation but is never really true in reality, the appropriate way to think about this is frequency of characteristic of the modes of the material. As an example, a lightweight rocket may appear rigid but as force is applied the rocket exterior will bend and oscillate at a frequency distinct in each mode. Thus, you wouldn't want your controller exciting a particular bending mode and shake the rocket apart, but if the flexible dynamics are much faster than the control inputs then can ignore and use rigid body assumptions (useful in robotics). In practice, so model the moment of bending you can setup accelerometers on a beam (like the ski), vibrate it on a sinusoidal pattern then extract the modes (a form of system id).

This leads into the topic of **constraints** which limit the motion of a system (e.g., rigid body has a constraint on its internal particles having a fixed distance between each other).

Two types of constraints:

- **Holonomic** constraints $f(r_1, r_2, ..., t) = 0$ describe conditions of the constraint connecting the coordinates of particles (and time)... example is a rigid body $((r_i - r_j)^2 - c_{ij}^2 = 0)$
- **Non-Holonomic** constraints that cannot be expressed in the above formulation (i.e., particle constrained to a surface of a sphere $r^2 - a^2 \geq 0$)

A system of N particles free from constraints has $3N$ independent coordinates or **Degrees of Freedom (DoF)**. If there exists k holonomic constraints expressed as equations in the above form we then eliminate k of the DoF (i.e., $3N - k =$ DoF). The elimination of dependent coordinates can be expressed as independent variables $q_1, q_2, ..., q_{3N-k}$ in terms of $r_1, r_2, ..., r_k$ expressed by the equations of the following form:

$$r_i = r_i(q_1, r_2, ..., q_{3N-k}, t) \text{ for } i \in \{1, ..., N\} \tag{12}$$

These are transformation equations from the set r to q. If the constraint(s) are not holonomic **the equations expressing the constraint cannot be used to eliminate the dependent coordinates**. Another method is used to eliminate non-holonomic constraints (i.e., method of lagrangian multipliers).

A rigid body has 6 DoF in total: 3 DoF for translation and 3 DoF for rotation (position and orientation about center of mass). When we look at rotation specifically (i.e., 3x3 matrix) we see that we have 9 equations but 6 constrained equations which leads to 3 DoF.

1.2.1 Coordinate Frames

We have two fundamental sets of coordinate frames (or reference frames):

1. **Fixed coordinate frame** (i.e., intertial frame or newtonian) denoted as F which is typically Earth fixed. $F = \{\hat{f}_x, \hat{f}_y, \hat{f}_z\}$ defined with mutually orthogonal unit vectors (i.e., some basis vectors)
2. **Body frame** denoted as B which refers to translating and rotating rigid body attached to center of mass or gravity (i.e., CoG / CoM). $B = \{\hat{e}_x, \hat{e}_y, \hat{e}_z\}$

Furthermore, we define some position of center of mass relative to the origin or point O in the fixed frame as the following: $[\vec{r}^G]_F = x\hat{f}_x + y\hat{f}_y + z\hat{f}_z$, likewise the component x in F maybe described as $x =< \vec{r}^G, \hat{f}_x >$.

1.2.2 Rotation Matrices and SO(3)

Given some fixed frame $F = \{\hat{x}_1, \hat{x}_2\}$ and rotated frame $B = \{\hat{y}_1, \hat{y}_2\}$ we want to find the map between $[x_1 x_2] \to [y_1 y_2]$. In this 2D case we can arrive at the mapping between coordinate frames via geomtry or trig but arrive at the following (note positive CCW about Right Handed Coordinate System):

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \cos(\psi) & \sin(\psi) \\ -\sin(\psi) & \cos(\psi) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$
$$[\vec{r}^G]_B = R_{BF}(\psi)[\vec{r}^G]_F \tag{13}$$

Where R_{BF} describes the rotation matrix from the fixed frame to the body. Likewise, we can invert this matrix to obtain the opposite transformation

$$R_{BF}^{-1} = R_{BF}^T = R_{FB} \quad \text{(i.e., the rotation from the body to the fixed frame).}$$

Since R_{BF} is an orthonormal matrix, its inverse is the transpose, and its determinant is $+1$. Note that -1 would describe a reflection, which corresponds to a left-handed coordinate system. This is allowable in $O(3)$, but not in $SO(3)$.

Furthermore, we can see that

$$R_{BF}^{-1} R_{BF} = R_{BF}^T R_{BF} = I.$$

Note that the **determinant** calculates the volume scaling (i.e., stretching) of the matrix, and a positive value gives the signed volume spanned by the columns.

Attitude is defined as the rotation between the body-fixed frame and the inertial frame, leading to the topic of **Lie Group** $SO(3)$ (i.e., the special orthogonal group), which defines the group of allowable rotation-based transformations:

$$SO(3) = \left\{ R \in \mathbb{R}^{3\times 3} \mid RR^T = I, \det R = 1 \right\}.$$

Here, "S" stands for Special (meaning $\det R = 1$), "O" stands for orthogonal, and "3" indicates $\mathbb{R}^{3\times 3}$. $SO(3)$ has 9 elements and is 3-DoF with no singularities. It has linear/bilinear kinematics but 3 redundant equations.

These are both **passive** rotations, where the coordinate frame rotates. The alternative is called an **active** rotation, where the vector actually rotates. For this course, we focus solely on passive rotations.

Expanding the 2D case above to the 3D case we arrive at these three **Elementary Rotation Matrices** (positive CCW rotation with RH coordinate system):

- 3-axis Rotation (about $\hat{x}_3$ or yaw axis):

$$R_3(\psi) = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{14}$$

- 1-axis Rotation (about $\hat{x}_1$ or roll axis):

$$R_1(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \tag{15}$$

- 2-axis Rotation (about $\hat{x}_2$ or pitch axis):

$$R_2(\theta) = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{16}$$

Thus from the above we can see that R_{BF} is a nice linear transformation between the body and intertial frame. One nifty way to describe the columns and rows of R_{BF} is the following:

$$R_{BF} = \begin{bmatrix} \hat{e}_1 \\ \hat{e}_2 \\ \hat{e}_3 \end{bmatrix} \begin{bmatrix} \hat{f}_1 & \hat{f}_2 & \hat{f}_3 \end{bmatrix} = \begin{bmatrix} \hat{e}_1\hat{f}_1 & \hat{e}_1\hat{f}_2 & \hat{e}_1\hat{f}_3 \\ \hat{e}_2\hat{f}_1 & \hat{e}_2\hat{f}_2 & \hat{e}_2\hat{f}_2 \\ \hat{e}_3\hat{f}_1 & \hat{e}_3\hat{f}_2 & \hat{e}_3\hat{f}_2 \end{bmatrix}$$

We can see that the rows of R_{BF} describe the body frame expressed in inertial components (i.e., B basis in F components), and likewise the columns of R_{BF} describe the opposite relationship.

$$\begin{bmatrix} (\hat{e}_1^F)^T & (\hat{e}_2^F)^T & (\hat{e}_3^F)^T \end{bmatrix} = \begin{bmatrix} \hat{f}_1^B & \hat{f}_2^B & \hat{f}_3^B \end{bmatrix}$$

1.2.3 Euler Angles and Rate Kinematics

Described by 3 numbers, 3 DoF with no constraints, but has singularities (i.e., $\det = 0$ which introduced gimbal lock scenarios) and nonlinear kinematics and composition.

1.2.4 Quaternions and Quaternion Rate Kinematics

Described by 4 numbers and 3 DoF with 1 constraint with no singularities, linear/bilinear kinematics, but have "double cover" meaning there are two quaternions which represent the same rotation sequence.

1.2.5 Axis Angle

Described by 3 numbers and 3 DoF, but have singularities and nonlinear knematics and composition like Euler Angles.

1.2.6 Gibbs / Rodrigues Vector

Described by 3 numbers and 3DoF, have quadratic kinematics and composition but have singularities.

Modified Rodrigues is described 3 numbers and a singularity only at 360 deg where its kinematics are quartic.

A Appendix: Vectors

A.1 Three Fundamental Types of Vectors

- **Free Vectors**: specifies magnitude and direction (i.e., translational velocity)
- **Sliding Vectors**: specifies the above in addition to line of action relative to a point of rotation (i.e., angular momentum)
- **Bound Vectors**: specifies point of application (i.e., force acting on an elastic body)

A.2 How a Vector can be Expressed

$A = M\hat{e}_A$
$A =$ is a vector, $M =$ is a scalar, $, \hat{e}_A =$ is a unit vector
$A = A_1\hat{e}_1 + A_2\hat{e}_2 + A_3\hat{e}_3$ where $\{\hat{e}_1, \hat{e}_2, \hat{e}_3\}$ are linearly independent unit vectors

A.3 Vector Operations

- **Vector Addition**: both commutative ($A+B = B+A$) and associative ($(A+B)+C = A+(B+C)$)
- **Dot Product**: scalar product of 2 vectors is both commutative ($A \cdot B = B \cdot A$) and distributive ($(A+B)\cdot C = A\cdot C + B\cdot C$)... likewise this describes the angle between two vectors ($|A||B|\cos(\theta)$) which can be thought of how these two vectors align (i.e., orthogonal vectors are $\perp$)
- **Cross Product**: vector product of 2 vectors is not commutative ($A \times B = -(B \times A)$) but is distributive ($A \times (B + C) = A \times B + A \times C$)... likewise it can be describe as the following $|A||B|\sin(\theta)\hat{k}$ where $\hat{k}$ is an unit vector on some basis of coordinates. Additionally the cross product can be defined in terms of the determinant w.r.t. mutually orthogonal unit vectors (i.e., a basis): $A \times B = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ A_x & A_y & A_z \\ B_x & B_y & B_z \end{vmatrix}$.
- **Scalar Triple Product**: $A \cdot (B \times C) = A \cdot \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ B_x & B_y & B_z \\ C_x & C_y & C_z \end{vmatrix} = \begin{vmatrix} A_x & A_y & A_z \\ B_x & B_y & B_z \\ C_x & C_y & C_z \end{vmatrix}$
- **Vector Triple Product**: $A \times (B \times C) = (A \cdot C)B - (A \cdot B)C$ and is not associative
- **Vector Derivatives**: $\frac{dA}{dv} = \lim_{\Delta v \to 0} = \frac{\Delta A}{\Delta v}$. Differentiation is distributive: $\frac{d(A+B)}{dv} = \frac{dA}{dv} + \frac{dB}{dv}$. Product rule applies: $\frac{d(gA)}{dv} = \frac{dg}{dv}A + g\frac{dA}{dv}$. Differentiation of a vector w.r.t. some coordinate system: $\frac{dA}{dt} = \frac{dA_x}{dt}\hat{i} + \frac{dA_y}{dt}\hat{j} + \frac{dA_z}{dt}\hat{k}$. Lastly, if the coordinate system is moving we have to apply the product rule: $\frac{dA}{dt} = \frac{dA_1}{dt}\hat{e}_1 + \frac{dA_2}{dt}\hat{e}_2 + \frac{dA_3}{dt}\hat{e}_3 + A_1\frac{\hat{e}_1}{dt} + A_2\frac{\hat{e}_2}{dt} + A_3\frac{\hat{e}_3}{dt}$

B Appendix: Integration

B.1 Explicit Euler

Explicit Euler Integration is equivalent to taking the 1st order taylor expansion of $x(t_{n+1})$ about $x(t_n)$

$$x_{n+1} = x(t_n + h) \approx x_n + \frac{dx}{dt}\Big|_{x_n} h \tag{17}$$

Where h is the time step and the derivative evaluated at x_n is $hf(x_n)$. To get a better integration result we can take more terms in this expansion via taylor expansion:

$$x_{n+1} = x(t_n + h) \approx x_n + \frac{dx}{dt}\Big|_{x_n} h + \frac{d^2x}{dt^2}\Big|_{x_n} \frac{h^2}{2} + ... \tag{18}$$

Where $\frac{dx}{dt} = f(x)$ and $\frac{d^2x}{dt^2} = \frac{d}{dt}f(x) = \frac{df}{dx}\frac{dx}{dt} = \frac{df}{dx}f(x)...$ can continue chain rule for order n (i.e., 3rd order is a tensor and so on). The drawback to this method is that it becomes very computationally expensive.

B.2 Runge-Kutta Methods

We want to achieve taylor expansion result and accuracy without computing the higher derivatives of $f(x)$ explicitly. So the key idea is to use multiple evaluations of $f(x)$ over the time step to fit a polynomial to $x(t)$. The number of $f(x)$ evaluations per time step is called the number of stages. So the best we can do in one stage is the following:

$$x_{n+1} \approx x_n + hf((1-\alpha)x_n + \alpha x_{n+1}) \text{ where } 0 \leq \alpha \leq 1$$
$$x_{n+1} \approx x_n + hf(x_n) + \alpha h^2 \frac{df}{dx} f(x) + h.o.t$$
$$\alpha = 0.5 \implies \text{second order method}$$

This results in the **Implicit Midpoint**:

$$x_{n+1} \approx x_n + hf(\frac{1}{2}x_n + \frac{1}{2}x_{n+1}) \tag{19}$$

The above implicit midpoint does require a newton step to solve (see below) because x_{n+1} is both on LHS and RHS of function being evaluated. However, this has really great energy conservation because the other explicit methods introduce numerical damping leading to drift over time (so implicit handle **stiff** aka hard to solve ODEs / higher frequencies such as a rigid body systems with constraints generally better). Furthermore, it adheres to the continuous time stability region ($Re(A) < 0$) just like the real system (best to use in simulaiton).

Explicit Midpoint approach avoids root finding and evaluates the midpoint using Euler step:

$$x_{n+1} = x_n + hf(x_n + \frac{h}{2}f(x_n)) \text{ an approximate midpoint}$$
$$x_{n+1} \approx x_n + hf(x) + \frac{h^2}{2}\frac{df}{dx}f(x) + h.o.t \tag{20}$$

This method is second order and has two stages.

Runge-Kutta can be continued to 3rd, 4th, and etc. order (nominal simulation us RK4 like ode45 in matlab). Essentially you expand the above with h.o.t. weighted by new α, β, γ terms. The following is how you can build a higher order RK method:

- Choose the number of stages
- Choose implicit vs/ explicit
- Write down integrator step with undetermined weights
- Write down taylor expansion of integrator step
- Choose weights to match taylor expansion to desired order

For example RK4 is the 4th order Runge-Kutta integration method. Weights can be chosen in an optimized fashion such as L1 to enforce sparsity and improved computational costs, or other things. The 4th order 4 stage RK method appears as the following:

$$f_1 = f(x_n)$$
$$f_2 = f(x_n + ha_{21}f_1)$$
$$f_3 = f(x_n + ha_{31}f_1 + ha_{32}f_2)$$
$$f_4 = f(x_n + ha_{41}f_1 + ha_{42}f_2 + ha_{43}f_3)$$
$$x_{n+1} = x_n + h[b_1f_1 + b_2f_2 + b_3f_3 + b_4f_4] \tag{21}$$

Each stage uses a past x and at the end you compute x_{n+1}. We can organize these coefficients into a matrix A and vector b called "Butcher Tableau" (note A is lower triangular for explicit and full for implicit).

B.3 Newton's Method

Another way is a root finding method to determine x_{n+1} called **Newton's Method** where we do the following algorithmically:

$$
\begin{aligned}
&r(x_{n+1}) = x_{n+1} - x_n - hf(\frac{1}{2}x_n + \frac{1}{2}x_{n+1}) = 0 \\
&r(x_{n+1} + \Delta x_{n+1}) \approx r(x_{n+1}) + \frac{dr}{dx}\Big|_{x_{n+1}} \Delta x_{n+1} = 0 \\
&\implies \Delta x_{n+1} = -(\frac{dr}{dx}\Big|_{x_{n+1}})^{-1} r(x_{n+1}) \\
&x_{n+1} \leftarrow x_{n+1} + \Delta x_{n+1} \text{ and repeat until convergence (3-4 iterations)}
\end{aligned} \tag{22}
$$

Downside to this is that root-finding is expensive to compute jacobian repeatedly, therefore there is quasi-newton methods to help speed up computation and convergence time.

Reference CMU lectures 4-5 https://github.com/dynamics-simulation-16-715/lecture-notebooks