AEM 2012: Dynamics
Numerical Integration

Prof. Ryan Caverly

Department of Aerospace Engineering and Mechanics
University of Minnesota

UNIVERSITY OF MINNESOTA

Fall 2018

- Kinematics - the geometry of motion.
- Dynamics - the science of motion; how forces interact with bodies causing deflections and/or motion.

What kinematics and dynamics is really about is *deriving the differential equations that describe the motion of a body*.

Equation of Motion of a Damped Pendulum

For example, consider a pendulum with a rod of negligible mass with a particle of mass m at its end. In class we derived

$$\underset{\rightarrow}{v}^{yw/a} = \underset{\rightarrow}{\mathcal{F}}_b^{\mathsf{T}} \begin{bmatrix} -\ell\dot{\theta} \\ 0 \\ 0 \end{bmatrix}, \qquad \underset{\rightarrow}{a}^{yw/a/a} = \underset{\rightarrow}{\mathcal{F}}_b^{\mathsf{T}} \begin{bmatrix} -\ell\ddot{\theta} \\ -\ell\dot{\theta}^2 \\ 0 \end{bmatrix}.$$

From our free-body diagram we determined

$$\underset{\rightarrow}{f}^{yg} = \underset{\rightarrow}{\mathcal{F}}_a^{\mathsf{T}} \begin{bmatrix} 0 \\ -mg \\ 0 \end{bmatrix} = \underset{\rightarrow}{\mathcal{F}}_b^{\mathsf{T}} \begin{bmatrix} mg\sin(\theta) \\ mg\cos(\theta) \\ 0 \end{bmatrix}, \quad \underset{\rightarrow}{f}^{yp} = \underset{\rightarrow}{\mathcal{F}}_b^{\mathsf{T}} \begin{bmatrix} 0 \\ f_{b2}^{yp} \\ 0 \end{bmatrix}.$$

We will now also consider a drag force acting on the particle modeled as a linear viscous damping force:

$$\underset{\rightarrow}{f}^{yd} = -c\,\underset{\rightarrow}{v}^{yw/a} = \underset{\rightarrow}{\mathcal{F}}_b^{\mathsf{T}} \begin{bmatrix} c\ell\dot{\theta} \\ 0 \\ 0 \end{bmatrix}.$$

Equation of Motion of a Damped Pendulum

Using Newton's Second Law, we get

$$\underset{\rightarrow}{f}^{yg} + \underset{\rightarrow}{f}^{yp} + \underset{\rightarrow}{f}^{yd} = m\,\underset{\rightarrow}{a}^{yw/a/a}$$

$$\underset{\rightarrow}{\mathcal{F}}_b^{\mathsf{T}} \begin{bmatrix} mg\sin(\theta) \\ mg\cos(\theta) \\ 0 \end{bmatrix} + \underset{\rightarrow}{\mathcal{F}}_b^{\mathsf{T}} \begin{bmatrix} 0 \\ f_{b2}^{yp} \\ 0 \end{bmatrix} + \underset{\rightarrow}{\mathcal{F}}_b^{\mathsf{T}} \begin{bmatrix} c\ell\dot{\theta} \\ 0 \\ 0 \end{bmatrix} = \underset{\rightarrow}{\mathcal{F}}_b^{\mathsf{T}} \begin{bmatrix} -m\ell\ddot{\theta} \\ -m\ell\dot{\theta}^2 \\ 0 \end{bmatrix}$$

Removing the vectrices and rearranging the equations gives

$$m\ell\ddot{\theta} + c\ell\dot{\theta} + mg\sin(\theta) = 0, \tag{1}$$

and

$$f_{b2}^{yp} = -m\left(\ell\dot{\theta}^2 + g\cos(\theta)\right). \tag{2}$$

Equation (1) is our differential equation of motion, while equation (2) is our expression for the reaction force.

Equation of Motion of a Damped Pendulum

Notice that

$$m\ell\ddot{\theta} + c\ell\dot{\theta} + mg\sin(\theta) = 0 \tag{3}$$

is a *nonlinear* ordinary differential equation (ODE)!

How do we find a solution to this nonlinear ODE?

Often nonlinear ODEs cannot be solved analytically; as such, we resort to numerical integration methods to numerically find solutions given a set of initial conditions (ICs).

First-Order Form

The first thing we need to do is put our ODE into *first-order* form:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t)$$

where $\mathbf{x} = [x_1 \;\; x_2 \;\; \ldots \;\; x_{n-1} \;\; x_n]^\mathsf{T}$ are the n states of the system.

For example, to write the equation of motion of the pendulum in first-order form let $x_1 = \theta$ and $x_2 = \dot{x}_1 = \dot{\theta}$ so that

$$m\ell\dot{x}_2 + c\ell x_2 + mg\sin(x_1) = 0,$$

or alternatively,

$$\dot{x}_2 = \frac{1}{m\ell}\left(-c\ell x_2 - mg\sin(x_1)\right).$$

We can now write out the entire ODE in first-order form as

$$\underbrace{\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix}}_{\dot{\mathbf{x}}(t)} = \underbrace{\begin{bmatrix} x_2(t) \\ \frac{1}{m\ell}\left(-c\ell x_2(t) - mg\sin(x_1(t))\right) \end{bmatrix}}_{\mathbf{f}(\mathbf{x}(t),t)} \tag{4}$$

Notice that all the “dots" are on the left-hand side, while each of the states, x_1 and x_2, are on the right-hand side.

First-Order Form

- Note that we have not “changed" the system; we’ve simply taken a second-order scalar ODE (i.e., Eq. (3)) and written it as a first-order matrix ODE (i.e., Eq. (4)).
- The form of $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t)$ is not unique. For instance

$$\underbrace{\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix}}_{\dot{\mathbf{x}}(t)} = \underbrace{\begin{bmatrix} \frac{1}{m\ell}\left(-c\ell x_1(t) - mg\sin(x_2(t))\right) \\ x_1(t) \end{bmatrix}}_{\mathbf{f}(\mathbf{x}(t),t)}$$

  is also correct; here $x_2 = \theta$ and $\dot{x}_2 = x_1 = \dot{\theta}$.

Numerical Integration - Euler Integration

- Denote

$$\mathbf{x}_k = \mathbf{x}(t_k)$$

  and $h = t_{k+1} - t_k$ as the step-size.
- Approximate $\dot{\mathbf{x}}$ using a simple forward difference:

$$\dot{\mathbf{x}}(t) \doteq \frac{\mathbf{x}(t_{k+1}) - \mathbf{x}(t_k)}{t_{k+1} - t_k} = \frac{\mathbf{x}_{k+1} - \mathbf{x}_k}{h}$$

- Use the above approximation in $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t)$ yeilding

$$\begin{aligned} \frac{\mathbf{x}_{k+1} - \mathbf{x}_k}{h} &= \mathbf{f}(\mathbf{x}_k, t_k), \\ \mathbf{x}_{k+1} &= \mathbf{x}_k + h\mathbf{f}(\mathbf{x}_k, t_k). \end{aligned}$$

- Given $\mathbf{x}_1 = \mathbf{x}(0)$ at $t_1 = 0$, “march forward in time".
- Issues: poor accuracy (global error on the order of h); numerical stability.

Numerical Integration - Runge-Kutta 4

- RK4 integration is based on a higher-order Taylor series approximation of $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t)$.
- $$\mathbf{x}_{k+1} = \mathbf{x}_k + \frac{1}{6}\left(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4\right)$$

  where

$$\begin{aligned}
\mathbf{k}_1 &= h\mathbf{f}(\mathbf{x}_k, t_k) \\
\mathbf{k}_2 &= h\mathbf{f}(\mathbf{x}_k + \frac{1}{2}\mathbf{k}_1, t_k + \frac{1}{2}h) \\
\mathbf{k}_3 &= h\mathbf{f}(\mathbf{x}_k + \frac{1}{2}\mathbf{k}_2, t_k + \frac{1}{2}h) \\
\mathbf{k}_4 &= h\mathbf{f}(\mathbf{x}_k + \mathbf{k}_3, t_k + h)
\end{aligned}$$

- Given $\mathbf{x}_1 = \mathbf{x}(0)$ at $t_1 = 0$, “march forward in time".
- Global error on the order of h^4; local error on the order of h^5.

Numerical Integration in `matlab`

- `ode45` in `matlab` uses the fourth order Runge-Kutta-Fehlberg method, which also gives an error estimate at each step.
- It is variable step size integrator; the solver is able to choose a step size which meets the error tolerance specified.
- Absolute tolerance (`AbsTol`) and Relative tolerance (`RelTol`).*
  - At each time-step, the solver estimates the local error, $e_{k,i}$, in the i^{th} state, $x_{k,i}$.
  - At that time step the simulation converges if

$$|e_{k,i}| \leq \max(\texttt{RelTol} \cdot |x_{k,i}|, \texttt{AbsTol}).$$

  - When state values are large, `RelTol` dictates convergence. As the states approach zero, `AbsTol` dictates convergence.
  - In general, values for `AbsTol` and `RelTol` are problem dependent.

* `http://www.mathworks.com/help/simbio/ref/absolutetolerance.html`.

References

These slides are based on slides by Prof. James Richard Forbes (McGill).