Calculating the Critical Time Step of Bonded Damped Flexible Fibers in DEM Simulations

Matt Schramm (Iowa State University)
Mehari Tekeste (Iowa State University)
Carol Plouffe (Deere & Company)
Donald Harby (Deere & Company)

July 8th, 2019

Objectives

- What are critical time steps
- Current methods for determining critical time steps
  - Rayleigh time step
  - Hertzian time step
  - Yu Guo's critical time step for bonded non-damped particles
- Numerical methods for calculating the critical time steps of ODEs
  - Euler update method
  - Velocity verlet update method
- Calculating the critical time step utilizing Python

Critical Time Steps

- Critical time step = highest time step that allows a simulation to run accurately
- Critical time step equations allow users to minimize simulation time
- Current methods do not accurately predict critical time steps for damped bonded particles
- This forces users to guess a stable time step
  - Simulations take longer than they need
  - Simulations start stable but crash later

The Bond Equations

Bond equations used in flexible fiber simulations where (Guo, et al; 2013)

F_n^b, F_t^b are the normal and tangential bond forces, respectively

M_n^b, M_t^b are the normal and tangential bond moments, respectively

$\delta\acute{F}_{n,i}^b, \delta\acute{F}_{t,i}^b$ are the normal and tangential i[th] incremental bond forces caused by the linear spring, respectively

$\delta\acute{M}_{n,i}^b, \delta\acute{M}_{t,i}^b$ are the normal and tangential i[th] incremental bond moments caused by the linear spring, respectively

K_n and K_t are the normal and tangential bond stiffness constants, respectively

A_b is the bond cross sectional area

Δt is the time step

β_{damp} is the local bond damping coefficient

M_e and J_s are the mass and moment of inertial of the individual particles, respectively

v_n and v_t are the normal and tangential relative velocities between the two particles, respectively

w_n and w_t are the normal and tangential relative angular velocities between the two particles, respectively

I and I_p are the second area moment and polar area moments of inertia, respectively

Y is the bond Young's modulus

ν is the Poison's ratio

l_b is the equilibrium bond length

$$\delta\acute{F}_{n,i}^b = K_t A_b v_n \Delta t$$

$$\delta\acute{F}_{t,i}^b = K_t A_b v_t \Delta t$$

$$\delta\acute{M}_{n,i}^b = K_t I_p \omega_n \Delta t$$

$$\delta\acute{M}_{t,i}^b = K_n I \omega_t \Delta t$$

$$F_n^b = \sum_{\forall i} \delta\acute{F}_{n,i}^b + 2\beta_{damp}\sqrt{M_e K_n} v_n$$

$$F_t^b = \sum_{\forall i} \delta\acute{F}_{t,i}^b + 2\beta_{damp}\sqrt{M_e K_t} v_t$$

$$M_n^b = \sum_{\forall i} \delta\acute{M}_{n,i}^b + 2\beta_{damp}\sqrt{J_s K_t I_p} \omega_n$$

$$M_t^b = \sum_{\forall i} \delta\acute{M}_{t,i}^b + 2\beta_{damp}\sqrt{J_s K_n I} \omega_t$$

$$K_n = \frac{Y}{l_b}, \qquad K_t = \frac{K_n}{2(1-\nu)}$$

Current Methods

- Rayleigh time step
  - $dt_{ray} = \frac{\pi r \sqrt{\frac{\rho}{G}}}{0.1631\nu + 0.8766}$
- Hertzian time step
  - $dt_{hert} = 2.87 \left(\frac{M_{eff}^2}{r_{eff} E_{eff}^2 v_{max}} \right)^{0.2}$
- Yu Guo's Method
  - $dt_{Guo} = 0.8165\, l_b \sqrt{\frac{\rho}{E}} = \sqrt{\frac{4\, r \rho\, l_b}{3E}}$, assuming $2r = l_b$
- Where
  - $\rho =$ Density, $E =$ Young's Modulus, $l_b =$ Bond Length, $\nu =$ Poisson's Ratio
    $r =$ Radius, $G =$ Shear Modulus, $v_{max} =$ Max Velocity of Particle
    $M_{eff} = \frac{m_1 m_2}{m_1 + m_2}, r_{eff} = \frac{r_1 r_2}{r_1 + r_2}, \frac{1}{E_{eff}} = \frac{1 - \nu_1^2}{E_1} + \frac{1 - \nu_2^2}{E_2}$

Numerical System Stability (Discontinuous)

- Given a system of first order differential equations

$$\mathbf{f} = \begin{bmatrix} f_1(x_1, x_2, \dots, x_m) \\ \vdots \\ f_n(x_1, x_2, \dots, x_m) \end{bmatrix}$$

- And a ODE solver (for example Euler's Equation)

$$\boldsymbol{X}^{n+1} = \boldsymbol{X}^n + \Delta t \mathbf{f}(\boldsymbol{X}^n) = \mathbf{F}$$

- Let **J** be the jacobian of **F** around the stability point ($\boldsymbol{X}^*$)

$$\mathbf{J} = \boldsymbol{I} + \Delta t \frac{\partial \mathbf{f}}{\partial \boldsymbol{X}}\Big|_{X^*}$$

- A system is discontinuously stable if $|\lambda_p| < 1 \; \forall p \in [1, m]$ of **J**
- Example $f = \lambda x \rightarrow X^{n+1} = X^n + \Delta t \lambda X^n \rightarrow |1 + \Delta t \lambda| < 1$
- This equation is discontinuously stable if $\Delta t < \frac{2}{\lambda}$

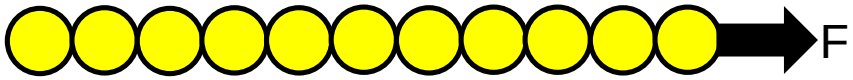
System Definition

- Multiple spheres are bonded together
- One end is pulled via a force
- System was chosen due to simplicity and that only bond forces are used

$$\boldsymbol{f} = [\boldsymbol{K}\ \boldsymbol{D}]\begin{bmatrix}\boldsymbol{X}\\ \boldsymbol{V}\end{bmatrix} + \boldsymbol{F}$$

- Numerical method requires a system of linear equations

$$\begin{bmatrix}\dot{\boldsymbol{X}}\\ \dot{\boldsymbol{V}}\end{bmatrix} = \begin{bmatrix}\boldsymbol{V}\\ \dfrac{\boldsymbol{f}}{\boldsymbol{m}}\end{bmatrix} = \begin{bmatrix}\boldsymbol{0} & \boldsymbol{I}\\ \dfrac{\boldsymbol{K}}{\boldsymbol{m}} & \dfrac{\boldsymbol{D}}{\boldsymbol{m}}\end{bmatrix}\begin{bmatrix}\boldsymbol{X}\\ \boldsymbol{V}\end{bmatrix} + \frac{\boldsymbol{F}}{\boldsymbol{m}} = \begin{bmatrix}\boldsymbol{0} & \boldsymbol{I}\\ \boldsymbol{W} & \boldsymbol{\beta}\end{bmatrix}\begin{bmatrix}\boldsymbol{X}\\ \boldsymbol{V}\end{bmatrix} + \frac{\boldsymbol{F}}{\boldsymbol{m}} = \boldsymbol{J}\begin{bmatrix}\boldsymbol{X}\\ \boldsymbol{V}\end{bmatrix} + \frac{\boldsymbol{F}}{\boldsymbol{m}}$$



Euler Method (System Creation)

- The Euler method yields the following system

$$\begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{V} \end{bmatrix}^{n+1} = \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{V} \end{bmatrix}^{n} + \Delta t \boldsymbol{J} \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{V} \end{bmatrix}^{n} = \boldsymbol{\gamma}$$

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ \boldsymbol{W} & \boldsymbol{\beta} \end{bmatrix}$$

  - Where for a 3 particle system

$J =$

```
[     0,      0,     0,      1,      0,      0]
[     0,      0,     0,      0,      1,      0]
[     0,      0,     0,      0,      0,      1]
[ -w^2,     w^2,     0, -2*b*w,  2*b*w,      0]
[  w^2, -2*w^2,   w^2,  2*b*w, -4*b*w,  2*b*w]
[     0,     w^2,  -w^2,      0,  2*b*w, -2*b*w]
```

  - Taking the derivative of gamma yields

$$\boldsymbol{I} + \Delta t \boldsymbol{J}$$

  - This is what we need to find the eigen values of

Euler Method (System Solution)

- Due to the structure of the Euler method, we can factor out the time step

$$|\boldsymbol{I} + \Delta t \boldsymbol{J}| < 1 == -2 < \Delta t \max(\boldsymbol{\lambda}) == \frac{2}{\max(\boldsymbol{\lambda})} > \Delta t$$

- Given the system, one can now run a DOE to determine the effects of DEM parameters have on the time step

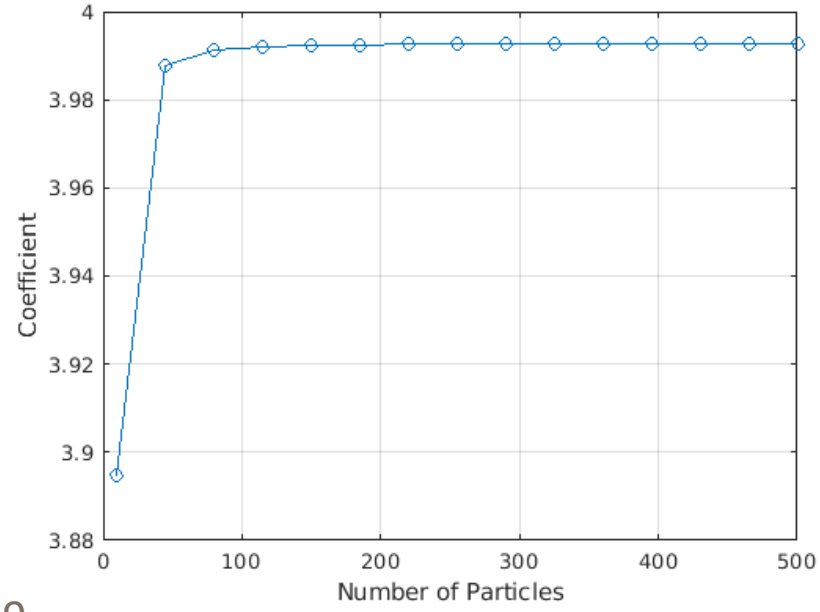
- DOE chosen Full Factorial
  - Young's Modulus [1 GPa, 10 GPa] at 15 levels
  - Local Bond Damping [0, 200] at 15 levels

- Model to fit as it gives the same results at Guo's model when beta == 0

$$\Delta t_{Euler} = \frac{\Delta t_{Guo}}{1 + \alpha_1 \beta}$$

Euler Method (DOE Results)

- Results were dependent on number of particles



- For a 500-particle system the model was

$$\Delta t_{Euler} = \frac{\Delta t_{Guo}}{(1 + 3.993\beta)}$$

with an adjusted r-squared value of 0.999

- This yielded much smaller time steps then what has been observed utilizing the LIGGGHTS DEM software with the main difference being that LIGGGHTS uses the Velocity Verlet equations instead of the Euler equations to do their updates

Velocity Verlet Method (System Creation)

- Figuring out the eigenvalues of the velocity verlet equations is not a simple task

$$\boldsymbol{V}^{n+\frac{1}{2}} = \boldsymbol{V}^{n} + \frac{\Delta t}{2}\boldsymbol{F}\left(\boldsymbol{X}^{n}, \boldsymbol{V}^{n-\frac{1}{2}}\right)$$
$$\boldsymbol{X}^{n+1} = \boldsymbol{X}^{n} + \Delta t \boldsymbol{V}^{n+\frac{1}{2}}$$
$$\boldsymbol{V}^{n+1} = \boldsymbol{V}^{n+\frac{1}{2}} + \frac{\Delta t}{2}\boldsymbol{F}\left(\boldsymbol{X}^{n+1}, \boldsymbol{V}^{n+\frac{1}{2}}\right)$$

- The above velocity verlet equations result in the following linear system utilizing the sympy toolbox for Python

$$\begin{bmatrix}\boldsymbol{X}\\ \boldsymbol{V}\end{bmatrix}^{n+1} = \begin{bmatrix}\boldsymbol{A}_{11} & \boldsymbol{A}_{12}\\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22}\end{bmatrix}\begin{bmatrix}\boldsymbol{X}\\ \boldsymbol{V}\end{bmatrix}^{n} + \begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{F}\end{bmatrix} = \boldsymbol{A}\begin{bmatrix}\boldsymbol{X}\\ \boldsymbol{V}\end{bmatrix}^{n} + \begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{F}\end{bmatrix} = \boldsymbol{\gamma}$$

- We now need to find a Δt such that the eigenvalues of $|\boldsymbol{A}| < 1$

A Matrix Values

- Where

$$A_{11} = I + \frac{\Delta t^2}{2}\left(W - \frac{\Delta t}{2}\beta W\right)$$

$$A_{12} = \Delta t\left(I + \frac{\Delta t}{2}\beta\left(I - \frac{\Delta t}{2}\beta\right)\right)$$

$$A_{21} = \frac{\Delta t}{2}\left(W\left(I + \frac{\Delta t^2}{2}\left(W - \frac{\Delta t}{2}\beta W\right)\right) + \frac{\Delta t}{2}\beta\left(W - \frac{\Delta t}{2}\beta W\right)\right) + \frac{\Delta t}{2}\left(W - \frac{\Delta t}{2}\beta W\right)$$

$$A_{22} = I + \frac{\Delta t}{2}\left(\beta\left(I + \frac{\Delta t}{2}\beta\left(I - \frac{\Delta t}{2}\beta\right)\right) + \Delta t W\left(I + \frac{\Delta t}{2}\beta\left(I - \frac{\Delta t}{2}\beta\right)\right)\right) + \frac{\Delta t}{2}\beta\left(I - \frac{\Delta t}{2}\beta\right)$$

Velocity Verlet Method (System Solution)

- Due to the structure of the system, we cannot separate Δt from the rest of the system like we were able to with the Euler update
- A non-linear zero finder is used to find the stable time steps for the system
- Given the system, one can now run a DOE to determine the effects of DEM parameters have on the time step
- DOE chosen Full Factorial
  - Young's Modulus [1 GPa, 10 GPa] at 15 levels
  - Local Bond Damping [0, 200] at 15 levels
- Model to fit as it gives the same results at Guo's model when beta == 0

$$\Delta t_{vv} = \frac{\Delta t_{Guo}}{1 + \alpha_1 \beta}$$

Velocity Verlet Method (DOE Results)

- Results were dependent on number of particles
- Results show used 100 particles in the system

$$\Delta t_{vv} = \frac{\Delta t_{Guo}}{1 + 2.82\beta}$$

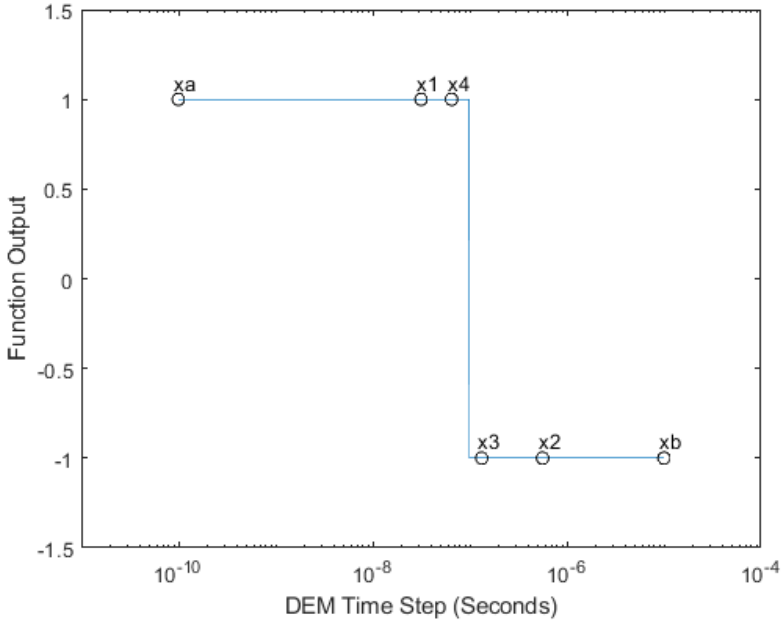
with an adjusted r-squared value = 0.999

- This method sometimes gave too large of a time step but needs little improvement
- This could be caused by the simple system that the eigenvalues were solved for and a more complex system may yield better results

Utilizing Python

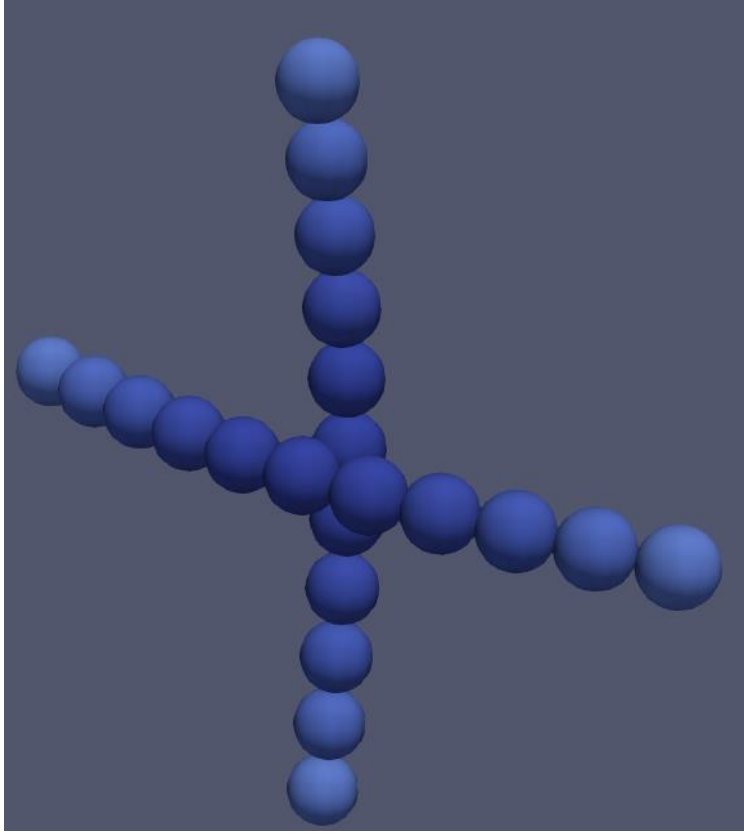
- Python (2.7) was used as a wrapper for LIGGGHTS in choosing a stable time step utilizing the bisection method

- Algorithm for choosing a stable time step
  - LIGGGHTS was given a guessed time step (1.0e-6)
  - LIGGGHTS would try to run a simulation
  - If the kinetic energy was found to be greater than the theoretical maximum energy, the simulation was a failure where as if the simulation finished, it was assigned a success



Python Simulation

- Two mega-particles consisting of 11 spheres each are created
- Mega-particles are given initial velocities and are set to collide with each other
- The maximum energy is set as the total kinetic energy of the system at simulation creation



Python DOE

- The same DOE that was used for both the Euler update and the velocity verlet update was used with the Python method with the addition of impact velocity

- DOE chosen Full Factorial
  - Young's Modulus [1 GPa, 10 GPa] at 15 levels
  - Local Bond Damping [0, 200] at 15 levels
  - Initial velocity [1 m/s, 100 m/s]

- Model to fit as it gives the same results at Guo's model when beta == 0

$$\Delta t_{Python} = \frac{\Delta t_{Guo}}{1 + \alpha_1 \beta}$$

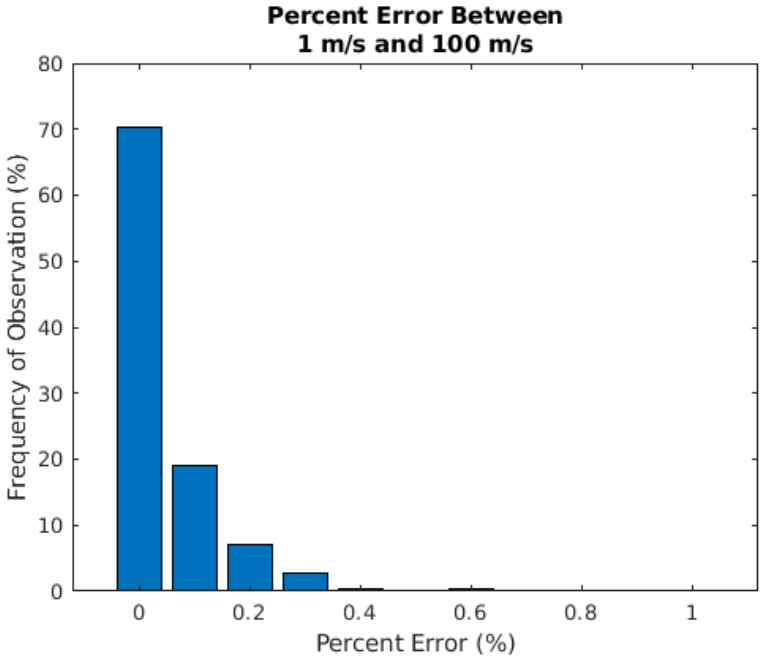
Python Results

- Impact Velocity played an insignificant role to the critical time step
- Model was determined to be

$$\Delta t_{Python} = \frac{\Delta t_{Guo}}{1 + 2.93\beta}$$

With an adjusted r-squared value of 0.999

- Results have been shown to be promising when utilizing LIGGGHTS with other simulations for damped flexible fibers
- Python method took the longest time to complete due to the high number of DEM simulations that must be ran



Conclusions

- Calculating the eigenvalues of the update method to find the critical time step can be very difficult depending on the update method
- Euler update gave critical time steps too small to be practical
- Velocity verlet gave critical time steps sometimes to large to keep a system stable
- Python version can be inheritably slow due to the large number of DEM simulations that must be run
- Python gave critical time step estimates that have been shown to work well