Finding the Contact Force

April 8, 2024

[1]:
```python
import sympy as sp

# Defining symbols
p_s, v_s, a_s = sp.symbols(r'\vec{p}_s \vec{v}_s \vec{a}_s')
k, n = sp.symbols(r'k n')
del_t = sp.Symbol(r'\Delta t')
xi, eta = sp.symbols(r'\xi \eta')
p_k, v_k, a_k = sp.symbols(r'\vec{p}_k \vec{v}_k \vec{a}_k')
phi_k = sp.Function(r'\phi_k')(xi, eta)
F_s, fc, R_s = sp.symbols(r'\vec{F}_s f_c \vec{R}_s')
F_k, N, R_k = sp.symbols(r'\vec{F}_k \vec{N} \vec{R}_k')
m_s, m_k = sp.symbols('m_s m_k')
```

1 Introduction

Consider a contact pair (a patch and a single node) in which a force resolution needs to be acquired between these entities. The goal is to ensure that applied force moves the node and patch in such a way that node lies on the surface of the patch at the next time step. Such a condition can be achieved by solving the following equation:

$$\vec{p}_s + \vec{v}_s\Delta t + \frac{1}{2}\vec{a}_s\Delta t^2 = \sum_{k=0}^{n-1}\phi_k(\xi,\eta)\left[\vec{p}_k + \vec{v}_k\Delta t + \frac{1}{2}\vec{a}_k\Delta t^2\right]$$

where p, v, and a are the position, velocity, and acceleration of the node and patch at the current time step, and subscript s denotes the slave node while k denotes the nodes that bound the master patch. Alternatively, the above equation can be represented as a matrix multiplication instead of a summation:

$$\vec{p}_s + \vec{v}_s\Delta t + \frac{1}{2}\vec{a}_s\Delta t^2 = \underbrace{\begin{bmatrix} p_{x0} + v_{x0}\Delta t + \frac{1}{2}a_{x0}\Delta t^2 & p_{x1} + v_{x1}\Delta t + \frac{1}{2}a_{x1}\Delta t^2 & \cdots \\ p_{y0} + v_{y0}\Delta t + \frac{1}{2}a_{y0}\Delta t^2 & p_{y1} + v_{y1}\Delta t + \frac{1}{2}a_{y1}\Delta t^2 & \cdots \\ p_{z0} + v_{z0}\Delta t + \frac{1}{2}a_{z0}\Delta t^2 & p_{z1} + v_{z1}\Delta t + \frac{1}{2}a_{z1}\Delta t^2 & \cdots \end{bmatrix}}_{A}\begin{bmatrix}\phi_0(\xi,\eta)\\ \phi_1(\xi,\eta)\\ \vdots \\ \phi_{n-1}(\xi,\eta)\end{bmatrix}$$

The acceleration for the slave node and the master patch node can be written as

$$\vec{a}_s = \frac{\vec{F}_s + f_c\vec{N} + \vec{R}_s}{m_s}$$
$$\vec{a}_k = \frac{\vec{F}_k - f_c\vec{N}\cdot\phi_k(\xi,\eta) + \vec{R}_k}{m_k}$$

where $\vec{F}$ is the internal force known prior to the analysis, f_c is the incremental contact force between the current node and the patch, and $\vec{R}$ is the force due to other contact pairs. $\vec{N}$ is the unit normal at the contact point (ξ,η) and must be in the outward direction of the patch, facing the non-penetrated slave node. This normal is found by taking the cross-product between the partial derivatives with respect to ξ and η.

$$N = \frac{A\frac{\partial}{\partial\xi}\phi_k \times A\frac{\partial}{\partial\eta}\phi_k}{\left|A\frac{\partial}{\partial\xi}\phi_k \times A\frac{\partial}{\partial\eta}\phi_k\right|}$$

```
[2]: eq1 = sp.Eq(p_s + v_s*del_t + sp.Rational(1, 2)*a_s*del_t**2, sp.Sum(phi_k*(p_k␣
     ↪+ v_k*del_t + sp.Rational(1, 2)*a_k*del_t**2), (k, 0, n-1)))
     eq1
```

[2]:

$$\frac{\Delta t^2\vec{a}_s}{2} + \Delta t\vec{v}_s + \vec{p}_s = \sum_{k=0}^{n-1}\left(\frac{\Delta t^2\vec{a}_k}{2} + \Delta t\vec{v}_k + \vec{p}_k\right)\phi_k(\xi,\eta)$$

[3]:
```
eq2 = eq1.subs([
    (a_s, (F_s + fc*N + R_s)/m_s),
    (a_k, (F_k - fc*N*phi_k + R_k)/m_k)
])
eq2
```

[3]:
$$\frac{\Delta t^{2} \left(\vec{F}_s + \vec{N} f_c + \vec{R}_s\right)}{2 m_s} + \Delta t \vec{v}_s + \vec{p}_s = \sum_{k=0}^{n-1} \left(\frac{\Delta t^{2} \left(\vec{F}_k - \vec{N} f_c \phi_k(\xi, \eta) + \vec{R}_k\right)}{2 m_k} + \Delta t \vec{v}_k + \vec{p}_k\right) \phi_k(\xi, \eta)$$

In the matrix form, this results in

[4]:
```
A = sp.Matrix([sp.Function('A')(xi, eta, fc)])
eq3 = sp.Eq(eq2.lhs, sp.MatMul(A, sp.Matrix([phi_k])), evaluate=False)
eq3
```

[4]:
$$\frac{\Delta t^{2} \left(\vec{F}_s + \vec{N} f_c + \vec{R}_s\right)}{2 m_s} + \Delta t \vec{v}_s + \vec{p}_s = \left[A(\xi, \eta, f_c)\right] \left[\phi_k(\xi, \eta)\right]$$

2 Solving

To solve this with the Newton-Raphson method, we have the following

$$\begin{bmatrix} \xi_{i+1} \\ \eta_{i+1} \\ f_{ci+1} \end{bmatrix} = \begin{bmatrix} \xi_{i} \\ \eta_{i} \\ f_{ci} \end{bmatrix} - \mathbf{J}^{-1}\mathbf{F}$$

[5]:
```
F = sp.Matrix([eq3.lhs]) - eq3.rhs
F
```

[5]:
$$\left[\frac{\Delta t^{2} \left(\vec{F}_s + \vec{N} f_c + \vec{R}_s\right)}{2 m_s} + \Delta t \vec{v}_s + \vec{p}_s - A(\xi, \eta, f_c) \phi_k(\xi, \eta)\right]$$

[6]:
```
J = F.jacobian([xi, eta, fc])
J
```

[6]:
$$\Big[-A(\xi, \eta, f_c) \frac{\partial}{\partial \xi} \phi_k(\xi, \eta) - \phi_k(\xi, \eta) \frac{\partial}{\partial \xi} A(\xi, \eta, f_c) \quad -A(\xi, \eta, f_c) \frac{\partial}{\partial \eta} \phi_k(\xi, \eta) - \phi_k(\xi, \eta) \frac{\partial}{\partial \eta} A(\xi, \eta, f_c) \quad \frac{\Delta t^{2} \vec{N}}{2 m_s} - \phi_k(\xi, \eta) \frac{\partial}{\partial f_c}$$

Here is a better look of the result:

$$\left[-A(\xi,\eta,f_c) \frac{\partial}{\partial \xi} \phi_k(\xi,\eta) - \frac{\partial}{\partial \xi} A(\xi,\eta,f_c) \phi_k(\xi,\eta) \quad -A(\xi,\eta,f_c) \frac{\partial}{\partial \eta} \phi_k(\xi,\eta) - \frac{\partial}{\partial \eta} A(\xi,\eta,f_c) \phi_k(\xi,\eta) \quad \frac{\Delta t^2 \vec{N}}{2 m_s} - \frac{\partial}{\partial f_c} A(\xi,\eta,f_c) \phi_k(\xi,\eta) \right]$$

Recall that for A, we have

$$\begin{bmatrix} p_{x0} + v_{x0}\Delta t + \frac{1}{2}\frac{F_{x0} - N_x f_c \cdot \phi_0(\xi,\eta) + R_{x0}}{m_0}\Delta t^2 & p_{x1} + v_{x1}\Delta t + \frac{1}{2}\frac{F_{x1} - N_x f_c \cdot \phi_1(\xi,\eta) + R_{x1}}{m_1}\Delta t^2 & \cdots \\ p_{y0} + v_{y0}\Delta t + \frac{1}{2}\frac{F_{y0} - N_y f_c \cdot \phi_0(\xi,\eta) + R_{y0}}{m_0}\Delta t^2 & p_{y1} + v_{y1}\Delta t + \frac{1}{2}\frac{F_{y1} - N_y f_c \cdot \phi_1(\xi,\eta) + R_{y1}}{m_1}\Delta t^2 & \cdots \\ p_{z0} + v_{z0}\Delta t + \frac{1}{2}\frac{F_{z0} - N_z f_c \cdot \phi_0(\xi,\eta) + R_{z0}}{m_0}\Delta t^2 & p_{z1} + v_{z1}\Delta t + \frac{1}{2}\frac{F_{z1} - N_z f_c \cdot \phi_1(\xi,\eta) + R_{z1}}{m_1}\Delta t^2 & \cdots \end{bmatrix}$$

The following matrices are constructed using the outer product:

$$\frac{\partial}{\partial \xi}A = -\vec{N} \otimes \frac{\partial}{\partial \xi}\phi_k \frac{f_c}{2m_k}\Delta t^2$$
$$\frac{\partial}{\partial \eta}A = -\vec{N} \otimes \frac{\partial}{\partial \eta}\phi_k \frac{f_c}{2m_k}\Delta t^2$$
$$\frac{\partial}{\partial f_c}A = -\vec{N} \otimes \phi_k \frac{1}{2m_k}\Delta t^2$$

3 Glued Condition

The above method is determining the force and reference coordinates subjected to some constant normal direction. However, for the glued case, the reference coordinates remain constant, and the direction of the force as well as its magnitude need to be determined. If we denote this force as $\vec{G}$, the matrix equation above changes to

[7]:
```
Gx, Gy, Gz = sp.symbols(r'G_x G_y G_z')
G = sp.Transpose(sp.transpose(sp.Matrix([Gx, Gy, Gz])))
A = sp.Matrix([sp.Function('A')(Gx, Gy, Gz)])
eq4 = sp.Eq(eq3.lhs.subs(N*fc, G), sp.MatMul(A, sp.Matrix([phi_k])),␣
↪evaluate=False)
eq4
```

[7]:
$$\frac{\Delta t^2\left(\vec{F}_s + \vec{R}_s + \begin{bmatrix} G_x & G_y & G_z \end{bmatrix}^T\right)}{2m_s} + \Delta t\vec{v}_s + \vec{p}_s = \begin{bmatrix} A(G_x, G_y, G_z) \end{bmatrix} \begin{bmatrix} \phi_k(\xi, \eta) \end{bmatrix}$$

The matrix A is the same as above, but the $N_x f_c$ component is replaced with the glued force component G_x for each x, y, and z.

[8]:
```
F = sp.Matrix([eq4.lhs]) - eq4.rhs
F
```

[8]:
$$\begin{bmatrix} \frac{\Delta t^2\left(\vec{F}_s + \vec{R}_s + \begin{bmatrix} G_x \\ G_y \\ G_z \end{bmatrix}\right)}{2m_s} + \Delta t\vec{v}_s + \vec{p}_s - A(G_x, G_y, G_z)\phi_k(\xi, \eta) \end{bmatrix}$$

[9]:
```
J = F.jacobian([Gx, Gy, Gz])
J
```

[9]:
$$\left[-\phi_k(\xi,\eta)\frac{\partial}{\partial G_x}A(G_x, G_y, G_z) + \begin{bmatrix} \frac{\Delta t^2}{2m_s} \\ 0 \\ 0 \end{bmatrix} \quad -\phi_k(\xi,\eta)\frac{\partial}{\partial G_y}A(G_x, G_y, G_z) + \begin{bmatrix} 0 \\ \frac{\Delta t^2}{2m_s} \\ 0 \end{bmatrix} \quad -\phi_k(\xi,\eta)\frac{\partial}{\partial G_z}A(G_x, G_y, G_z) + \right.$$

Here is a better look:

$$\begin{bmatrix} -\frac{\partial}{\partial G_x}A(G_x,G_y,G_z)\phi_k(\xi,\eta) + \begin{bmatrix} \frac{\Delta t^2}{2m_s} \\ 0 \\ 0 \end{bmatrix} & -\frac{\partial}{\partial G_y}A(G_x,G_y,G_z)\phi_k(\xi,\eta) + \begin{bmatrix} 0 \\ \frac{\Delta t^2}{2m_s} \\ 0 \end{bmatrix} & -\frac{\partial}{\partial G_z}A(G_x,G_y,G_z)\phi_k(\xi,\eta) + \begin{bmatrix} 0 \\ 0 \\ \frac{\Delta t^2}{2m_s} \end{bmatrix} \end{bmatrix}$$

The partial derivatives can be constructed by

$$\frac{\partial}{\partial G_x} A = -\begin{bmatrix}1 & 0 & 0\end{bmatrix}^T \otimes \phi_k \frac{1}{2m_k}\Delta t^2$$
$$\frac{\partial}{\partial G_y} A = -\begin{bmatrix}0 & 1 & 0\end{bmatrix}^T \otimes \phi_k \frac{1}{2m_k}\Delta t^2$$
$$\frac{\partial}{\partial G_z} A = -\begin{bmatrix}0 & 0 & 1\end{bmatrix}^T \otimes \phi_k \frac{1}{2m_k}\Delta t^2$$

4 Complete Algorithm

The analysis presented here computes the force for a single contact pair (that is a single patch and single node); however, the force from other contact pairs does affect the calculated force at the current contact pair. The way to combat this is by iteratively solving for the contact force for each pair and storing the solution into $\vec{R}$ for each node. Then, repeat the process until eventually, the value of $\vec{R}$ converges. The algorithm looks something like this

```
for i in 0...max_iter
    for each contact pair
        compute fc (or G)
        # update R for each node
        Rs = Rs + N*fc (or Rs = Rs + G) # for slave node
        Rk = Rk - N*fc*phi_k (or Rk = Rk - G*phi_k) # for patch node

    # If R is a vector that houses all the forces for each node
    if i > 0 and norm(R[i] - R[i-1]) < tol
        break
```

5 Force Guess

For guessing the increment f_c, create a virtual node at the contact point that has a mass equal to the average of all the nodes. Find the acceleration and velocity at this virtual point in the normal direction (this is done by computing the projection of acceleration and velocity on the normal direction). The increment in force is then calculated by

[10]:
```python
psn, vsn, Fsn, Rsn = sp.symbols(r'p_{sn} v_{sn} F_{sn} R_{sn}')
m_avg, pkn, vkn, Fkn, Rkn = sp.symbols(r'm_{avg} p_{kn} v_{kn} F_{kn} R_{kn}')
eq5 = sp.Eq(psn + vsn*del_t + (Fsn + Rsn + fc)/(2*m_s)*del_t**2, pkn +
 ↪vkn*del_t + (Fkn + Rkn - fc)/(2*m_avg)*del_t**2)
eq5
```

[10]:
$$\frac{\Delta t^2\left(F_{sn} + R_{sn} + f_c\right)}{2m_s} + \Delta t v_{sn} + p_{sn} = \frac{\Delta t^2\left(F_{kn} + R_{kn} - f_c\right)}{2m_{avg}} + \Delta t v_{kn} + p_{kn}$$

[11]:
```python
fc_guess = sp.solve(eq5, fc)[0]
fc_guess
```

[11]:

$$\frac{F_{kn}\Delta t^2 m_s - F_{sn}\Delta t^2 m_{avg} + R_{kn}\Delta t^2 m_s - R_{sn}\Delta t^2 m_{avg} + 2\Delta t m_s m_{avg} v_{kn} - 2\Delta t m_s m_{avg} v_{sn} + 2 m_s m_{avg} p_{kn} - 2 m_s n}{\Delta t^2 \left(m_s + m_{avg}\right)}$$

The quantities F_{kn}, R_{kn}, v_{kn}, and p_{kn} are found by using the summation operation with the iso-parametric quantities.