CIEM1110-1: FEM, lecture 4.3

Viscoelasticity in pyJive

Iuri Rocha, Cor Kasbergen

TU Delft

Recap — NonlinModule with history dependency

Require: Nonlinear relation $\mathbf{f}_{\text{int}}(\mathbf{a})$ with $\mathbf{K}(\mathbf{a}) = \frac{\partial \mathbf{f}_{\text{int}}}{\partial \mathbf{a}}$ ⇐ `SolidModel._get_matrix(params,globdat)`

1: Initialize new solution at old one: $\mathbf{a}^{n+1} = \mathbf{a}^n$

2: Update material: $\boldsymbol{\sigma}^{n+1}, \mathbf{D}^{n+1}, \boldsymbol{\alpha}_{\text{new}} = \mathcal{M}\left(\boldsymbol{\varepsilon}^{n+1}, \boldsymbol{\alpha}_{\text{old}}\right)$ ⇐ `MaxwellMaterial.update(strain,ipoint)`

3: Compute internal force and stiffness: $\mathbf{f}_{\text{int}}^{n+1} = \int_\Omega \mathbf{B}^{\mathrm{T}} \boldsymbol{\sigma}^{n+1} \mathrm{d}\Omega$, $\mathbf{K}^{n+1} = \int_\Omega \mathbf{B}^{\mathrm{T}} \mathbf{D}^{n+1} \mathbf{B} \mathrm{d}\Omega$

4: Constrain $\mathbf{K}^{n+1}$ so that $\Delta \mathbf{a}_c = \overline{\mathbf{a}}^{n+1} - \overline{\mathbf{a}}^n$ ⇐ `DirichletModel._advance(params,globdat)`

5: Evaluate first residual: $\mathbf{r} = -\mathbf{f}_{\text{int},f}^{n+1}$

6: **repeat**

7: Solve linear system of equations: $\mathbf{K}^{n+1} \Delta \mathbf{a} = \mathbf{r}$

8: Update solution: $\mathbf{a}^{n+1} = \mathbf{a}^{n+1} + \Delta \mathbf{a}$

9: Update material: $\boldsymbol{\sigma}^{n+1}, \mathbf{D}^{n+1}, \boldsymbol{\alpha}_{\text{new}} = \mathcal{M}\left(\boldsymbol{\varepsilon}^{n+1}, \boldsymbol{\alpha}_{\text{old}}\right)$ ⇐ `MaxwellMaterial.update(strain,ipoint)`

10: Compute internal force and stiffness: $\mathbf{f}_{\text{int}}^{n+1} = \int_\Omega \mathbf{B}^{\mathrm{T}} \boldsymbol{\sigma}^{n+1} \mathrm{d}\Omega$, $\mathbf{K}^{n+1} = \int_\Omega \mathbf{B}^{\mathrm{T}} \mathbf{D}^{n+1} \mathbf{B} \mathrm{d}\Omega$

11: Evaluate residual: $\mathbf{r} = -\mathbf{f}_{\text{int},f}^{n+1}$

12: Constrain $\mathbf{K}^{n+1}$ so that $\Delta \mathbf{a}_c = 0$

13: **until** $|\mathbf{r}| <$ tolerance

14: Commit history: $\boldsymbol{\alpha}_{\text{old}} = \boldsymbol{\alpha}_{\text{new}}$ ⇐ `MaxwellMaterial.commit()`

MaxwellMaterial

```
def update(self, strain, ipoint=None):
    # Compute delta time
    dt = self._globdat[gn.TIME] - self._oldtime

    # Compute delta epsilon
    self._neweps[:,ipoint] = strain
    deps = self._neweps[:,ipoint] - self._oldeps[:,ipoint]

    # Use ElasticMaterial to compute long-term stresses
    stiff_inf, sig_inf = super().update(strain,ipoint)

    # Update Prony elements and get current viscoelastic stress
    stiff_visco, sig_visco = self._prony[ipoint].update(deps,dt)

    # Combine contributions
    stiff  = stiff_inf + stiff_visco
    stress = sig_inf + sig_visco

    return stiff, stress
```

```
def commit(self, ipoint=None):
    # Store time and current strain
    self._oldtime = self._globdat[gn.TIME]
    self._oldeps = np.copy(self._neweps)

    # Commit Prony history
    for prony in self._prony:
        prony.commit()
```

MaxwellMaterial — stress and consistent linearization

From the content of yesterday (1D):

$$\sigma = E_\infty \varepsilon + \sum_k E_k H_k^t \quad \text{with} \quad H_k^t = e^{-\frac{\Delta t}{\rho_k}} H_k^{\mathrm{o}} + e^{-\frac{\Delta t}{2\rho_k}} \Delta\varepsilon$$

But we also need the consistent tangent:

$$D = \frac{\partial \sigma}{\partial \varepsilon} = E_\infty + \sum_k E_k \frac{\partial H_k^t}{\partial \Delta\varepsilon} \frac{\partial \Delta\varepsilon}{\partial \varepsilon} \quad \Rightarrow \quad D = E_\infty + \underbrace{\sum_k e^{-\frac{\Delta t}{2\rho_k}} E_k}_{D_{\mathrm{ve}}}$$

To reduce this back and forth, we can just store historical stresses instead of H:

$$\sigma = E_\infty \varepsilon + \sum_k \sigma_{\mathrm{ve},k}^{\mathrm{new}} (\Delta\varepsilon, \Delta t) \quad \text{where} \quad \sigma_{\mathrm{ve},k}^{\mathrm{new}} = e^{-\frac{\Delta t}{\rho_k}} \sigma_{\mathrm{ve},k}^{\mathrm{old}} + D_{\mathrm{ve},k}^{\mathrm{new}} \Delta\varepsilon$$

For 2D/3D stress states (constant Poisson): $D_{\mathrm{ve},k} \Rightarrow \mathbf{D}_{\mathrm{ve},k} \left(e^{-\frac{\Delta t}{2\rho_k}} E_k, \nu \right)$