CIEM5110-2: FEM, workshop 6.1

Nonlinear FEM with plasticity

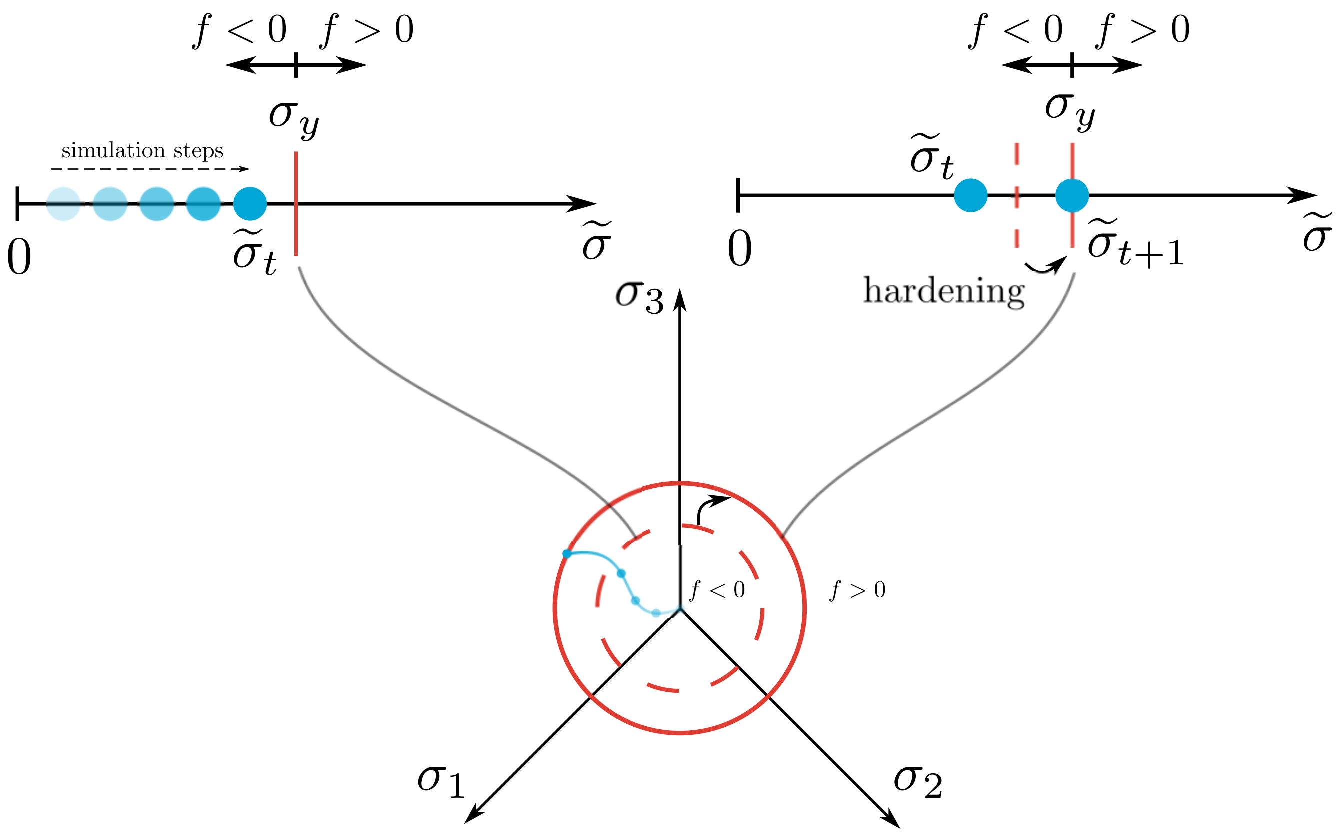
Iuri Rocha, Frans van der Meer

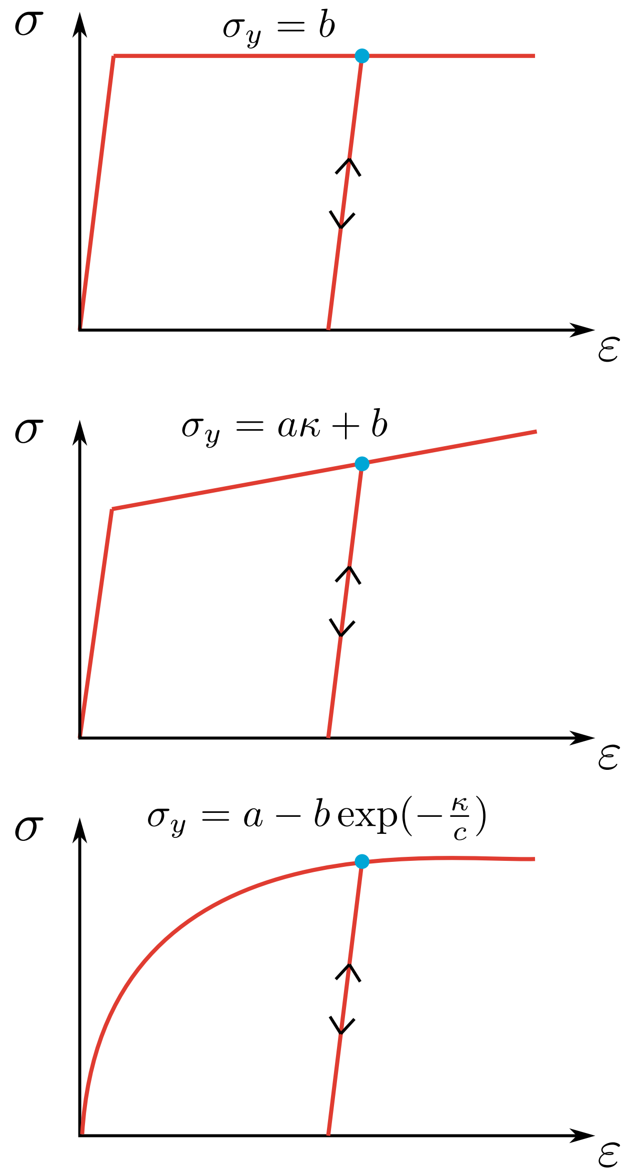
TU Delft

Von Mises (J_2) plasticity

Model from the previous lecture:

- In the code we call the accumulated plastic strain `kappa`





Load control with a history-dependent material

Require: Nonlinear relation $\mathbf{f}_{\text{int}}(\mathbf{a})$ with $\mathbf{K}(\mathbf{a}) = \frac{\partial \mathbf{f}_{\text{int}}}{\partial \mathbf{a}}$

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

Update material model: $\left\{\boldsymbol{\sigma}^{n+1}, \mathbf{D}^{n+1}, \boldsymbol{\alpha}_{\text{new}}\right\} = \mathcal{M}\left(\boldsymbol{\varepsilon}^{n+1}, \boldsymbol{\alpha}_{\text{old}}\right)$ $\Leftarrow$ `j2material.update(strain, ipoint)`

Compute internal force and stiffness: $\mathbf{f}_{\text{int}}^{n+1}, \mathbf{K}^{n+1}$ $\Leftarrow$ `SolidModel._get_matrix(params, globdat)`

Get new external force vector: $\mathbf{f}_{\text{ext}}^{n+1}$ $\Leftarrow$ `NeumannModel._advance_step(params, globdat)`

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

repeat

    Solve linear system of equations: $\mathbf{K}^{n+1}\Delta\mathbf{a} = \mathbf{r}$ $\Leftarrow$ `NonlinModule.run()`

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

    Update material model: $\left\{\boldsymbol{\sigma}^{n+1}, \mathbf{D}^{n+1}, \boldsymbol{\alpha}_{\text{new}}\right\} = \mathcal{M}\left(\boldsymbol{\varepsilon}^{n+1}, \boldsymbol{\alpha}_{\text{old}}\right)$ $\Leftarrow$ `j2material.update(strain, ipoint)`

    Compute internal force and stiffness: $\mathbf{f}_{\text{int}}^{n+1}, \mathbf{K}^{n+1}$ $\Leftarrow$ `SolidModel._get_matrix(params, globdat)`

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

until $|\mathbf{r}| <$ tolerance

Commit material history: $\boldsymbol{\alpha}_{\text{old}} \leftarrow \boldsymbol{\alpha}_{\text{new}}$ $\Leftarrow$ `j2material.commit()`

Displacement control with a history-dependent material

Require: Nonlinear relation $\mathbf{f}_{\text{int}}(\mathbf{a})$ with $\mathbf{K}(\mathbf{a}) = \frac{\partial \mathbf{f}_{\text{int}}}{\partial \mathbf{a}}$

Initialize new solution at old one: $\mathbf{a}^{n+1} \leftarrow \mathbf{a}^{n}$
Update material model: $\left\{\boldsymbol{\sigma}^{n+1}, \mathbf{D}^{n+1}, \boldsymbol{\alpha}_{\text{new}}\right\} = \mathcal{M}\left(\boldsymbol{\varepsilon}^{n+1}, \boldsymbol{\alpha}_{\text{old}}\right)$ $\Leftarrow$ `j2material.update(strain, ipoint)`
Compute internal force and stiffness: $\mathbf{f}_{\text{int}}^{n+1}, \mathbf{K}^{n+1}$ $\Leftarrow$ `SolidModel._get_matrix(params, globdat)`
Constrain $\mathbf{K}^{n+1}$ so that $\Delta \mathbf{a}_c = \bar{\mathbf{a}}^{n+1} - \bar{\mathbf{a}}^{n}$ $\Leftarrow$ `DirichletModel._advance_step(params, globdat)`
Evaluate first residual: $\mathbf{r} = -\mathbf{f}_{\text{int},f}^{n+1}$
repeat
    Solve linear system of equations: $\mathbf{K}^{n+1} \Delta \mathbf{a} = \mathbf{r}$ $\Leftarrow$ `NonlinModule.run()`
    Update solution: $\mathbf{a}^{n+1} \leftarrow \mathbf{a}^{n+1} + \Delta \mathbf{a}$
    Update material model: $\left\{\boldsymbol{\sigma}^{n+1}, \mathbf{D}^{n+1}, \boldsymbol{\alpha}_{\text{new}}\right\} = \mathcal{M}\left(\boldsymbol{\varepsilon}^{n+1}, \boldsymbol{\alpha}_{\text{old}}\right)$ $\Leftarrow$ `j2material.update(strain, ipoint)`
    Compute internal force and stiffness: $\mathbf{f}_{\text{int}}^{n+1}, \mathbf{K}^{n+1}$ $\Leftarrow$ `SolidModel._get_matrix(params, globdat)`
    Evaluate residual: $\mathbf{r} = -\mathbf{f}_{\text{int},f}^{n+1}$
    Constrain $\mathbf{K}^{n+1}$ so that $\Delta \mathbf{a}_c = 0$
until $|\mathbf{r}| <$ tolerance
Commit material history: $\boldsymbol{\alpha}_{\text{old}} \leftarrow \boldsymbol{\alpha}_{\text{new}}$ $\Leftarrow$ `j2material.commit()`

Modified Newton-Raphson

The importance of consistent linearization:

- Implement a Modified Newton-Raphson scheme in `nonlinmodule.py`
- Investigate the effect on convergence under displacement control
- In load control, set `['neumann']['values']` to $[3.0]$ and `['nonlin']['itermax']` to 50000