OOFEM: Implementace plasticitního materiálového modelu Cam-Clay

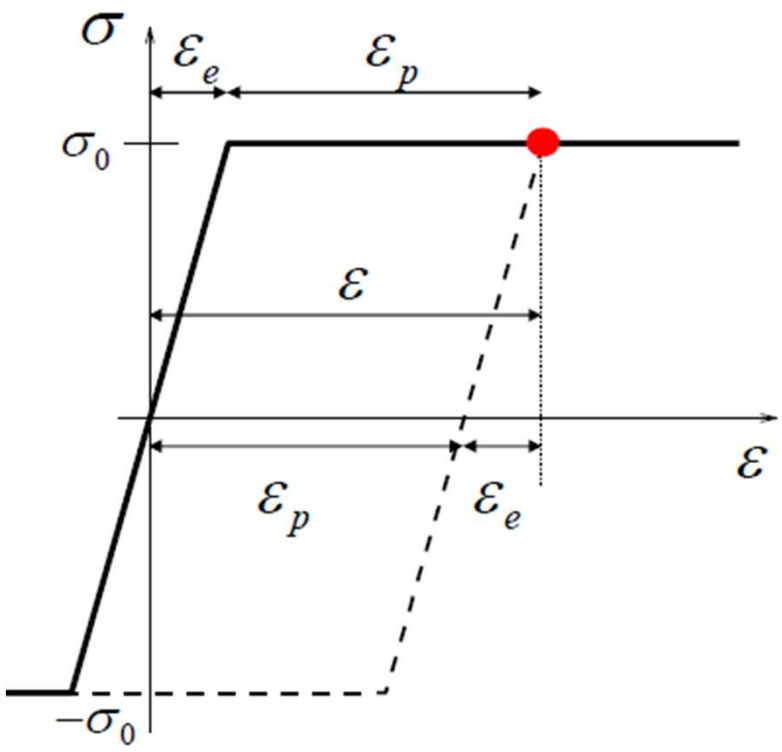
Ondřej Faltus, ZS 2016/17

Vyučující: Ing. Martin Horák, PhD.

Teorie plasticity

Pružnoplastické chování

- Princip: materiál se chová elasticky dokud napětí nedosáhne určité hodnoty, poté dochází k plastizaci
- Ideálně pružnoplastický model: předpoklady
  - Napětí je dáno vztahem $\sigma = E\varepsilon_e$
  - Po překroční meze plasticity σ_0 dochází k plastizaci a ε_{pl} roste nade všechny meze, zatímco se napětí nemění



Pružnoplastické chování

- Toto lze popsat třemi základními vztahy:
  - Podmínka plastické přípustnosti:

$$f(\sigma) = |\sigma| - \sigma_0 \leq 0$$

  - Zákon plastického přetváření:

$$\dot{\varepsilon_p} = \dot{\lambda} \cdot \mathrm{sgn}(\sigma)$$

  - Podmínka komplementarity:

$$\dot{\lambda} \cdot f(\sigma) = 0$$

Zákony plasticity ve 3D (víceosá napjatost)

- Složitější situace – lineární problém přechází na prostorový
- Funkce plasticity zde neudává jen body na ose, ale celý povrch v prostoru napětí
- Veličiny napětí a deformace přecházejí na veličiny tenzorové
- Různé podmínky plasticity (materiálové modely)

- Volumetrická vs deviatorická deformace
- Hydrostatické vs deviatorické napětí

Pružné chování – zobecněný Hookův zákon

- Jednotlivé osy napjatosti se navzájem ovlivňují
- E, G, ν – materiálové konstanty

$$\boldsymbol{\sigma} = \boldsymbol{D}_e \cdot \boldsymbol{\varepsilon}_e$$

- Matice pružné tuhosti:

$$\boldsymbol{D}_e = \frac{E}{(1+\nu)(1-2\nu)} \begin{pmatrix} 1-\nu & \nu & \nu & & & \\ \nu & 1-\nu & \nu & & \emptyset & \\ \nu & \nu & 1-\nu & & & \\ & & & 0{,}5-\nu & 0 & 0 \\ & \emptyset & & 0 & 0{,}5-\nu & 0 \\ & & & 0 & 0 & 0{,}5-\nu \end{pmatrix}$$

Základní vztahy pro plasticitu při víceosé napjatosti

- Toto lze popsat třemi základními vztahy:
  - Podmínka plastické přípustnosti zůstává (bude se lišit tvar funkce plasticity):

$$f(\boldsymbol{\sigma}) \leq 0$$

  - Zákon plastického přetváření - obecně:

$$\dot{\boldsymbol{\varepsilon}}_p = \dot{\lambda} \cdot \boldsymbol{g}(\boldsymbol{\sigma})$$

  - Podmínka komplementarity též zůstává:

$$\dot{\lambda} \cdot f(\boldsymbol{\sigma}) = 0$$

Zákon plastického přetváření

- Rychlost deformace jako taková je stále dána společným násobitelem $\dot{\lambda}$
- Plastický gradient $\boldsymbol{g}(\boldsymbol{\sigma})$ je tenzor závislý na stavu napjatosti, udávající poměr rychlostí deformace v jednotlivých směrech
- Sdružený zákon plastického přetváření = princip, používaný u některých materiálových modelů, že $\boldsymbol{g}(\boldsymbol{\sigma}) = \boldsymbol{f}(\boldsymbol{\sigma}) = \frac{\partial f(\boldsymbol{\sigma})}{\partial \boldsymbol{\sigma}}$ (gradient funkce plasticity)

Materiálové modely

- Nemožnost jednotně popsat chování různých materiálů
- Různé materiálové modely – teoretické představy o plasticitním chování
- Liší se tvarem funkce plasticity a sdružeností zákona plastického přetváření (u nesdružených třeba jinak definovat funkci $\boldsymbol{g}(\boldsymbol{\sigma})$)
- Tvary funkcí $f(\boldsymbol{\sigma})$, popř. $\boldsymbol{g}(\boldsymbol{\sigma})$ lze upravovat ke konkrétnímu používání empiricky pomocí škálování určitých koeficientů na základě zkoušek materiálu
- Např. modely Von Misesův, Drucker-Pragerův apod.

Modifikovaný Cam-Clay model

Základní údaje

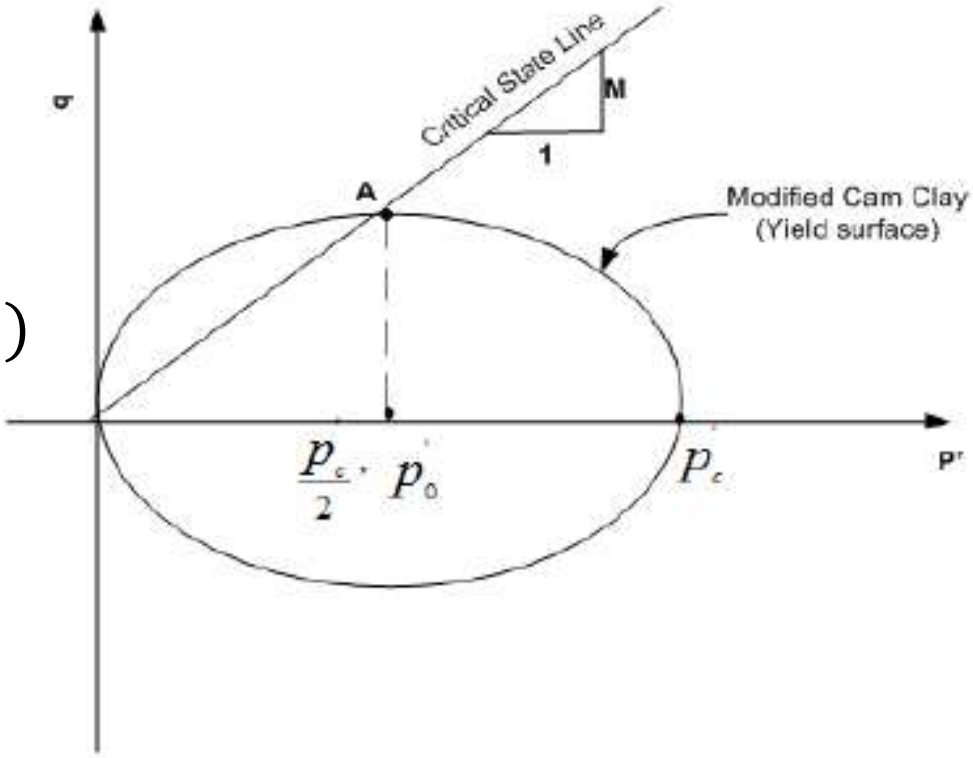
- Používán pro popis chování zemin
- Vyvinut v 50. až 60. letech 20. století na univerzitě v Cambridge prací prof. K. H. Roscoea a J. Burlanda
- Upravuje původní model Cam-Clay tak, aby jeho funkce plasticity měla eliptický tvar -> jednoduchost numerické implementace

Parametry modelu

- Nelineární elasticita $\boldsymbol{D}_e = \boldsymbol{D}_e(\boldsymbol{\sigma})$
- Zpevnění
- Funkce plasticity:

$$f(p, q, p_c) = q^2 - M^2 p(p_c - p)$$

,kde $p = -\sigma_m$ a $q = \sqrt{\frac{3}{2}}\|\boldsymbol{s}\|$



Parametry modelu

$I_1 = 3\sigma_m, p = -\frac{I_1}{3}$

$q = \sqrt{3J_2},\ \frac{\partial J_2}{\partial \boldsymbol{\sigma}} = \boldsymbol{s}$

- Sdružený zákon plastického přetváření, tj.:

$$\boldsymbol{g}(\boldsymbol{\sigma}) = \frac{\partial f(\boldsymbol{\sigma})}{\partial \boldsymbol{\sigma}} = \frac{\partial f}{\partial p}\frac{\partial p}{\partial I_1}\frac{\partial I_1}{\partial \boldsymbol{\sigma}} + \frac{\partial f}{\partial q}\frac{\partial q}{\partial J_2}\frac{\partial J_2}{\partial \boldsymbol{\sigma}}$$

$$= M^2(2p - p_c)\left(-\frac{1}{3}\right)\boldsymbol{\delta} + 2q\frac{\sqrt{3}}{2\sqrt{J_2}}\boldsymbol{s}$$

$$= \frac{M^2}{3}(2\sigma_m + p_c)\boldsymbol{\delta} + 3\boldsymbol{s}$$

Parametry modelu

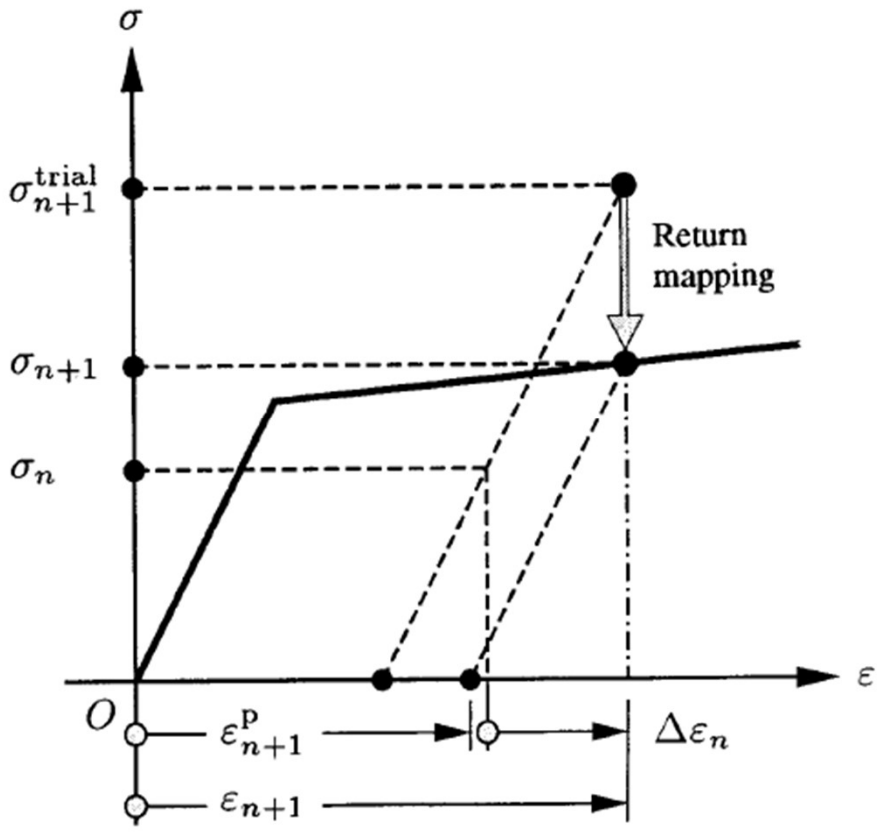
- Zpevňování/změkčování, tj. princip, že hodnota funkce plasticity se v průběhu zatěžování vyvíjí
- Zde vyjádřeno proměnnou hodnotou parametru p_c

$$\frac{\dot{p}_c}{p_c} = -\frac{3\dot{\varepsilon}_{mp}}{\lambda^* - \kappa^*}$$

Algoritmus pro numerický výpočet

Používané algoritmy

- Metoda konečných prvků
- Plasticita se řeší na úrovni jednoho prvku
- „návrat na plochu plasticity“



Návrat na plochu plasticity

- Předepisovaná deformace v pseudočasových krocích
- Idea: dva kroky
- 1) elastická predikce
  - $\Delta\boldsymbol{\varepsilon}_{p,n+1} = 0 \;\rightarrow \boldsymbol{\sigma}_{n+1,tr} = \boldsymbol{D}_e\left(\boldsymbol{\varepsilon}_{n+1} - \boldsymbol{\varepsilon}_{p,n}\right)$
  - Pokud $f\left(\boldsymbol{\sigma}_{n+1,tr}\right) \leq 0 \rightarrow$ hotovo
  - Pokud $f\left(\boldsymbol{\sigma}_{n+1,tr}\right) > 0 \rightarrow$ plastická korekce

Návrat na plochu plasticity

- 2) plastická korekce = vrácení stavu napjatosti do přípustných hodnot
  - $\Delta\boldsymbol{\varepsilon}_{p,n+1} \neq 0 \rightarrow \Delta\lambda_{n+1} \neq 0 \rightarrow f(\boldsymbol{\sigma}_{n+1}) = 0 \qquad (1.)$
  - $\boldsymbol{\sigma}_{n+1} = \boldsymbol{D}_e \left(\boldsymbol{\varepsilon}_{n+1} - \boldsymbol{\varepsilon}_{p,n} - \Delta\lambda_{n+1}\boldsymbol{g}(\boldsymbol{\sigma}_{n+1})\right) \qquad (2.)$
  - Dvě rovnice, neznámé $\boldsymbol{\sigma}_{n+1}$ a $\Delta\lambda_{n+1}$
  - Řešitelnost?

Metoda projekce na nejbližší bod

- Iterační metoda řešení problému z předchozí strany
- Algoritmus popsán v *SIMO, J. C. a Thomas J. R. HUGHES. Computational inelasticity. New York: Springer, c1998. ISBN 0-387-97520-9* (obrázek také převzat)
- V současné době varianta pro dokonalou plasticitu

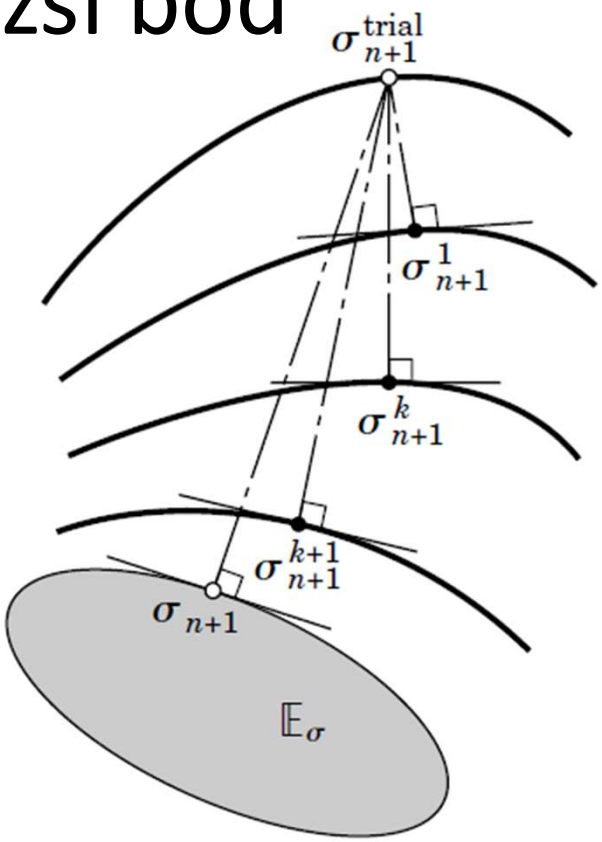


FIGURE 3-10. A geometric interpretation of the closest point projection algorithm in stress space. At each iterate $(\bullet)^{(k)}$, the constraint is linearized to find the intersection (cut) with $f = 0$. The next iterate $(\bullet)^{(k+1)}$, located on level set $f_{n+1}^{(k+1)} > 0$, is the closest point of that level set to the previous iterate $(\bullet)^{(k)}$ in the metric defined by the elasticities $\mathbf{C}$.

Metoda projekce na nejbližší bod

Define the plastic flow residual $\boldsymbol{R}_{n+1}$ and yield condition:

$$\left.\begin{aligned} \boldsymbol{R}_{n+1} &:= -\boldsymbol{\varepsilon}^{p}_{n+1} + \boldsymbol{\varepsilon}^{p}_{n} + \Delta\gamma\,\partial_{\boldsymbol{\sigma}} f_{n+1} \\ f_{n+1} &:= f(\boldsymbol{\sigma}_{n+1}) \end{aligned}\right\} \tag{3.6.1}$$

where $\boldsymbol{\sigma}_{n+1} = \mathbf{C} : [\boldsymbol{\varepsilon}_{n+1} - \boldsymbol{\varepsilon}^{p}_{n+1}]$.

2. Linearize the above equations. Since $\boldsymbol{\varepsilon}_{n+1}$ is *fixed during the return-mapping stage*, it follows that $\Delta\boldsymbol{\varepsilon}^{p^{(k)}}_{n+1} = -\mathbf{C}^{-1} : \Delta\boldsymbol{\sigma}^{(k)}_{n+1}$, and one is led to the linearized problem

$$\left.\begin{aligned} \boldsymbol{R}^{(k)}_{n+1} + [\boldsymbol{\Xi}^{(k)}_{n+1}]^{-1} : \Delta\boldsymbol{\sigma}^{(k)}_{n+1} + \Delta^{2}\gamma^{(k)}_{n+1}\partial_{\boldsymbol{\sigma}} f^{(k)}_{n+1} &= \mathbf{0} \\ f^{(k)}_{n+1} + \partial_{\boldsymbol{\sigma}} f^{(k)}_{n+1} : \Delta\boldsymbol{\sigma}^{(k)}_{n+1} &= 0 \end{aligned}\right\} \tag{3.6.2}$$

where $\boldsymbol{\Xi} := \left[\mathbf{C}^{-1} + \Delta\gamma\,\partial^{2}_{\boldsymbol{\sigma}\boldsymbol{\sigma}} f\right]^{-1}$ is the exact Hessian matrix.

Metoda projekce na nejbližší bod

3a. Solve the linearized problem to obtain $\Delta^2\gamma_{n+1}^{(k)}$ and $\Delta\boldsymbol{\varepsilon}_{n+1}^{p(k)}$:

$$\left.\begin{aligned} \Delta^2\gamma_{n+1}^{(k)} &:= \frac{f_{n+1}^{(k)} - \boldsymbol{R}_{n+1}^{(k)} : \boldsymbol{\Xi}_{n+1}^{(k)} : \partial_{\boldsymbol{\sigma}} f_{n+1}^{(k)}}{\partial_{\boldsymbol{\sigma}} f_{n+1}^{(k)} : \boldsymbol{\Xi}_{n+1}^{(k)} : \partial_{\boldsymbol{\sigma}} f_{n+1}^{(k)}} \\ \Delta\boldsymbol{\sigma}_{n+1}^{(k)} &:= \boldsymbol{\Xi}_{n+1}^{(k)} : \left[-\boldsymbol{R}_{n+1}^{(k)} - \Delta^2\gamma_{n+1}^{(k)} \partial_{\boldsymbol{\sigma}} f_{n+1}^{(k)} \right] \end{aligned}\right\} \qquad (3.6.3)$$

and

$$\Delta\boldsymbol{\varepsilon}_{n+1}^{p(k)} := -\mathbf{C}^{-1} : \Delta\boldsymbol{\sigma}_{n+1}^{(k)}$$

3b. Update the plastic strain $\boldsymbol{\varepsilon}_{n+1}^{(k)}$ and consistency parameter $\Delta\gamma_{n+1}^{(k)}$:

$$\boldsymbol{\varepsilon}_{n+1}^{p(k+1)} = \boldsymbol{\varepsilon}_{n+1}^{p(k)} + \Delta\boldsymbol{\varepsilon}_{n+1}^{p(k)}$$

and

$$\Delta\gamma_{n+1}^{(k+1)} = \Delta\gamma_{n+1}^{(k)} + \Delta^2\gamma_{n+1}^{(k)}. \qquad (3.6.4)$$

Konkrétně pro model Cam-Clay

$$\frac{\partial^2 f(\boldsymbol{\sigma})}{\partial \boldsymbol{\sigma}^2} = 3\boldsymbol{S} + \frac{2}{3}M^2\Delta$$

$$\boldsymbol{S} = \begin{pmatrix} \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} & & & \\ -\frac{1}{3} & \frac{2}{3} & -\frac{1}{3} & & \emptyset & \\ -\frac{1}{3} & -\frac{1}{3} & \frac{2}{3} & & & \\ & & & 2 & 0 & 0 \\ & \emptyset & & 0 & 2 & 0 \\ & & & 0 & 0 & 2 \end{pmatrix}, \Delta = \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & & & \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & & \emptyset & \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & & & \\ & & & & & \\ & \emptyset & & & \emptyset & \\ & & & & & \end{pmatrix}$$

Implementace do programu OOFEM

OOFEM

- SW pro konečné prvky
- C++
- Můj úkol = materiálový model Cam-Cay
- Objektová struktura programu -> vše soustředěno v jedné třídě *CamClayMat* v jednom souboru *camclaymat.c*
- Jednoduchá integrace se zbytkem
- Metoda *performPlasticityReturn()* a další

performPlasticityReturn()

- Funkce, která provádí celý algoritmus (prediktor+korektor)
- Argumenty: GaussPoint, nový $\boldsymbol{\varepsilon}_{n+1}$, timeStep, ostatní si program pro daný bod pamatuje
- Využívá funkce již obsažené v programu OOFEM (především pro maticové operace)
- Na konci zpracuje výsledky

performPlasticityReturn()

```
void
CamClayMat :: performPlasticityReturn(GaussPoint *gp, const FloatArray &totalStrain, TimeStep *tStep)
{

    CamClayMatStatus *status = static_cast< CamClayMatStatus * >( this->giveStatus(gp) );
    double pc = status->givePreconsolidationPressure();
    FloatArray plasticStrain;
    FloatArray fullStress;
    // get the initial plastic strain and initial kappa from the status
    plasticStrain = status->givePlasticStrain();
    // elastic predictor
    FloatArray elStrain = totalStrain;
    elStrain.subtract(plasticStrain);
    FloatArray elStrainDev;
    double elStrainVol;
    elStrainVol = computeDeviatoricVolumetricSplit(elStrainDev, elStrain);
    FloatArray trialStressDev;
    applyDeviatoricElasticStiffness(trialStressDev, elStrainDev, G);
    double trialStressVol = 3 * K * elStrainVol;
    // check the yield condition at the trial state
    double trialQ = sqrt(3. / 2.) * computeStressNorm(trialStressDev); //because q = sqrt(3/2)*||s||
    double trialP = -trialStressVol; //because p=-simga_m
    double yieldValue = trialQ*trialQ - M*M*trialP*(pc-trialP);
    double lambda = status->givePlasticMultiplier();
```

performPlasticityReturn()

```
if ( yieldValue > 0. ) {
  // closest-point projection
  //retrieving elasticity matrix
  FloatMatrix dE;
  give3dMaterialStiffnessMatrix(dE, MatResponseMode::ElasticStiffness, gp, tStep);//placeholder
  //initialization of parametres for the projection algorithm
  int iterationNumber = 0; //step counter
  double deltaLambda = 0.0;
  FloatArray nextPlasticStrain; //eps_p,n+1
  nextPlasticStrain.copySubVector(plasticStrain, 1);//initially there is no increment in plastic strain (equals trial state)
  FloatArray nextStress; //sigma_n+1
  nextStress.beProductOf(dE, elStrain);//initially equals trial stress
  bool convergence;
  double plasticStrainError;

  do {
    //compute Hessian matrix
    FloatMatrix inverseHessianMatrix;
    inverseHessianMatrix.beInverseOf(dE);
    FloatMatrix yieldFunctionDoubleDerivative;
    giveYieldFunctionDoubleDerivative(yieldFunctionDoubleDerivative);
    yieldFunctionDoubleDerivative.times(deltaLambda);
    inverseHessianMatrix.add(yieldFunctionDoubleDerivative);
    FloatMatrix hessianMatrix;
    hessianMatrix.beInverseOf(inverseHessianMatrix);

    //compute gradient
    FloatArray gradient;
    givePlasticGradientAtStress(gradient, nextStress, pc);
```

performPlasticityReturn()

```
        //update stress, strain, deltaLambda
        nextStress.add(deltaNextStress);
        nextPlasticStrain.add(deltaNextPlasticStrain);
        deltaLambda += deltaDeltaLambda;

        //update yield value
        yieldValue = giveYieldValueAtStress(nextStress, pc);
        plasticStrainError = sqrt( residual.dotProduct(residual) );

        iterationNumber++;
        convergence = ( fabs(yieldValue) < yieldTolerance && plasticStrainError < plasticStrainTolerance );

      } while (iterationNumber <= maxIter && !convergence);

      //convergence reached
      //updating all values to values reached by algorithm
      fullStress = nextStress;
      plasticStrain = nextPlasticStrain;
      lambda += deltaLambda;
      if(!convergence) {
        OOFEM_WARNING("Local equlibrium of CPP algorithm not reached in %d iterations, Element number %d, gp %d, continuing";
      }

    } else {
      // assemble the stress from the elastically computed volumetric part
      // and scaled deviatoric part
      computeDeviatoricVolumetricSum(fullStress, trialStressDev, trialStressVol);
    }


    // store the stress in status
    status->letTempStressVectorBe(fullStress);
    // store the plastic strain, preconsolidation stress, and plastic multiplier
    status->letTempPlasticStrainBe(plasticStrain);
    status->setTempPreconsolidationPressure(pc);
    status->setTempPlasticMultiplier(lambda);
}
```

Další funkce

- *giveYieldValueAtStress()* – jednoduchá funkce pro vyhodnocení funkce plasticity v daném napjatostním stavu
- *givePlasticityGradientAtStress()* – podobné, ale vrací hodnotu $\boldsymbol{g}(\boldsymbol{\sigma})$ (sdružený zákon -> je to derivace)
- *giveYieldFunctionDoubleDerivative()* – tato hodnota kupodivu nezávisí na napětí – viz výše

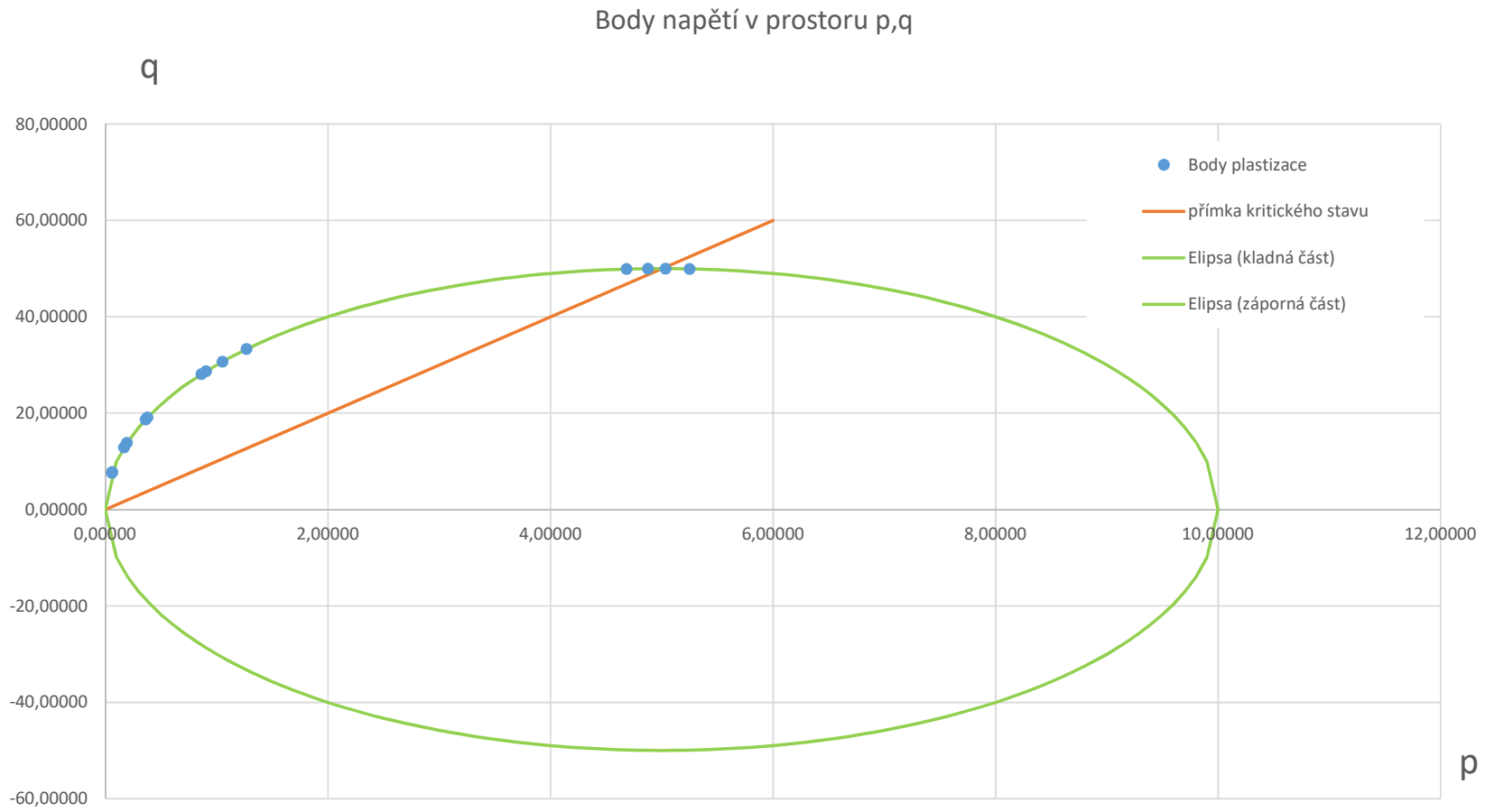
Funkce zůstávající k implementaci

- *give3dMaterialStiffnessMatrix()* – tato funkce nyní vrací elastickou matici – pro splnění předpokladu nelineární elasticity by měla vracet tečnou matici, případně elastoplastickou v případě že materiál již plastizuje
- Úprava algoritmu v *performPlasticityReturn()* tak aby zohlednil zpevnění

Demonstrace správnosti plochy plasticity

- Zkušební vstupní soubor s jedním MKP prvkem – předepisovány deformace ve dvou směrech
- $p_c = 10\ \mathrm{MPa}, M = 10$
- Předepsání různých poměrů deformací – různé hodnoty *p*, *q* vzniklých napětí
- Zobrazení vypočtených napětí v plastických stavech v prostoru p,q a porovnání s teoretickou elipsou plasticity dle parametrů modelu

Výsledek


Body napětí v prostoru p,q
q
80,00000
60,00000
40,00000
20,00000
0,00000
-20,00000
-40,00000
-60,00000
0,00000
2,00000
4,00000
6,00000
8,00000
10,00000
12,00000
p
Body plastizace
přímka kritického stavu
Elipsa (kladná část)
Elipsa (záporná část)

Děkuji za pozornost