Chapter 6: Basic Plasticity
Ch. 6.6

Myoung-Gyu Lee

TA: Gyu Jang Sim (gyujang95@snu.ac.kr)

ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY
서 울 대 학 교 공 과 대 학

MAMEL
MATERIALS MECHANICS LABORATORY

Linear representation of Newton-Raphson method (N-R)

- **Single variable**

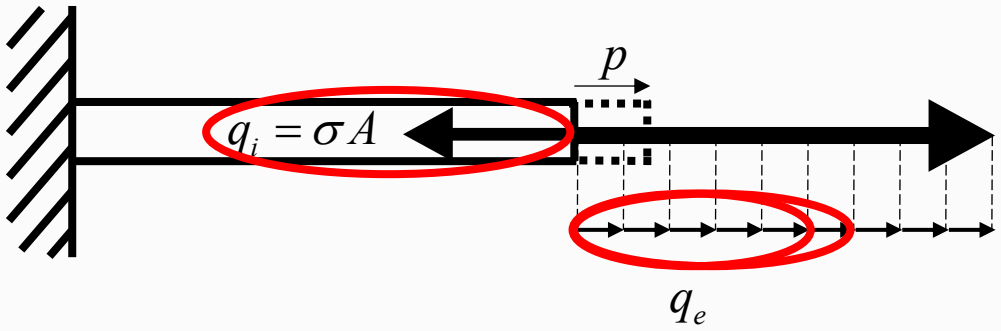
to find root from $f(x)=0$, repeat $f(x_n)=f_n=R_n$ and $x_{n+1}=x_n-\dfrac{R_n}{\left(\dfrac{df}{dx}\right)_n}$

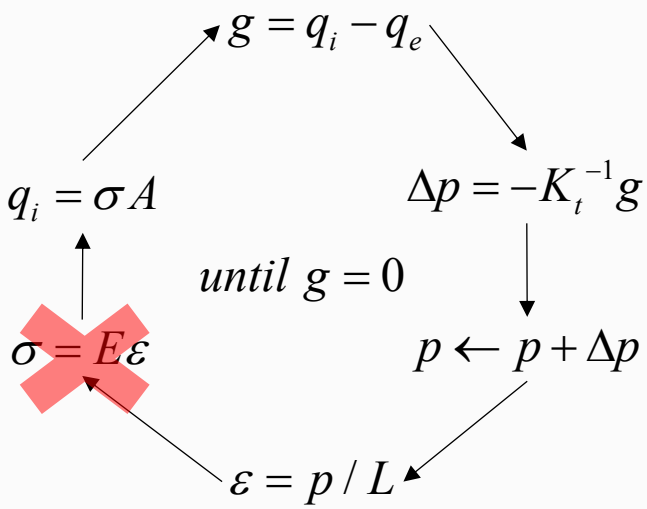
Or, $\Delta x \doteq -\dfrac{R_n}{\left(\dfrac{df}{dx}\right)_n}$ ➡ $f_n+\left(\dfrac{df}{dx}\right)_n \Delta x=0$

- **Multi-variable**

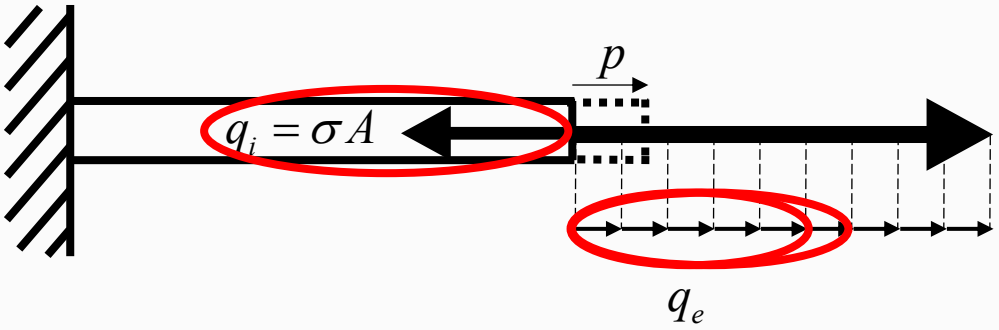
$$R_n+\left(\frac{df}{dx}\right)_n \Delta x+\left(\frac{df}{dy}\right)_n \Delta y+\left(\frac{df}{dz}\right)_n \Delta z=0 \quad \text{or} \quad R_n+\left(\frac{df}{d\mathbf{x}}\right)_n^T \Delta \mathbf{x}=0$$

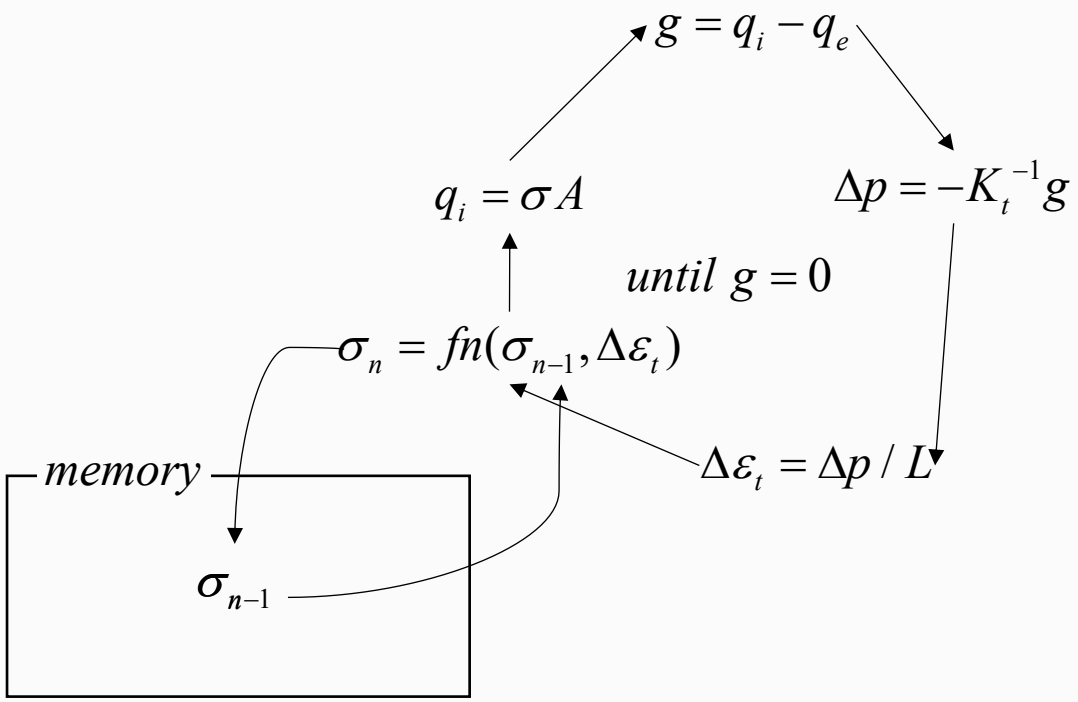
Review: Combined incremental/iterative solution



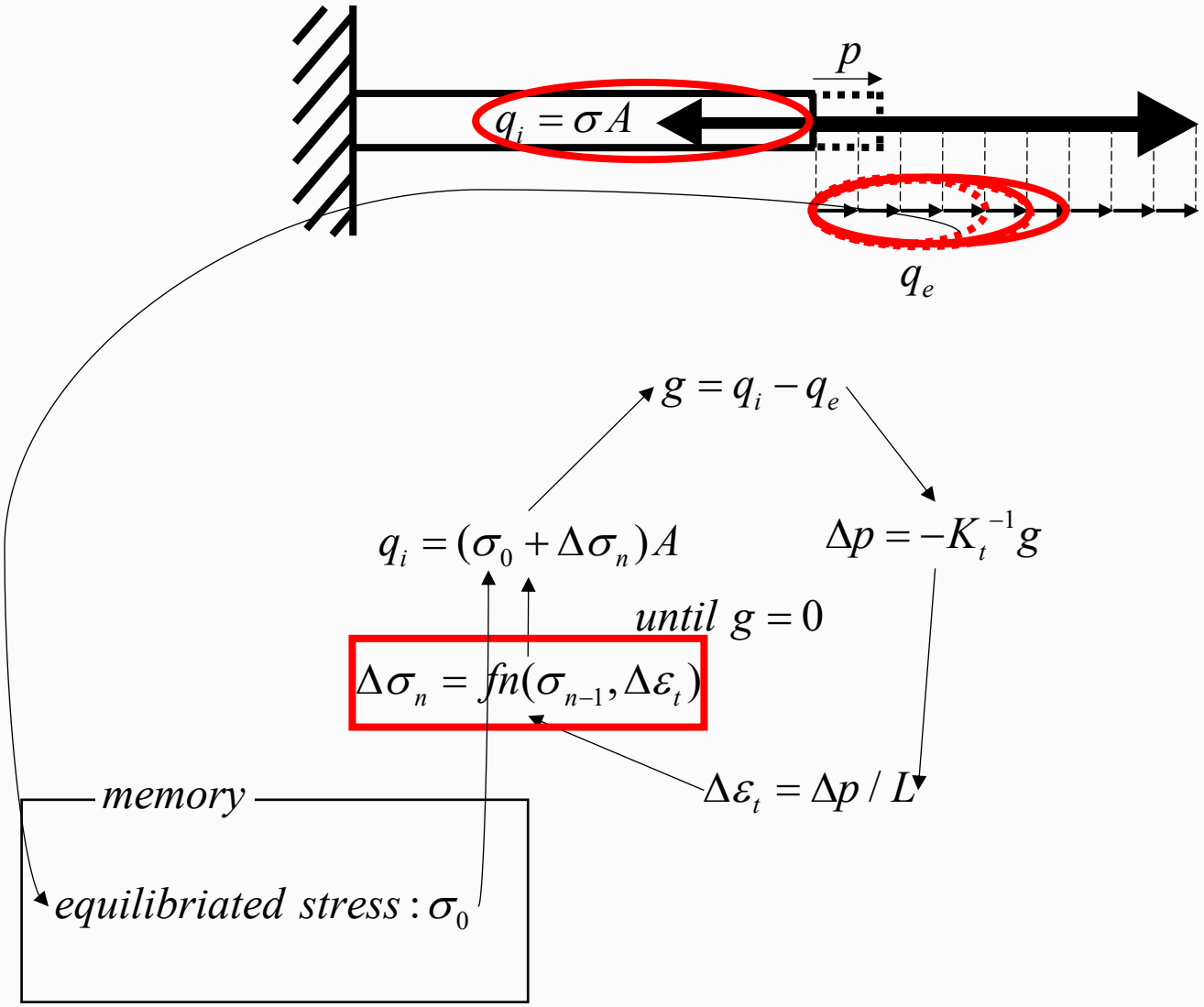


Review: Strategy A - iterative strains





Review: Strategy B - incremental strains



- What we have to formulate: $\Delta\boldsymbol{\sigma}_n = fn(\boldsymbol{\sigma}_{n-1}, \Delta\boldsymbol{\varepsilon}_t)$

- What plasticity theory gives:

$$\dot{\boldsymbol{\sigma}} = \begin{cases} elastic \rightarrow \mathbf{C} : \dot{\boldsymbol{\varepsilon}} \\ plastic \rightarrow \left(\mathbf{C} - \dfrac{3\mu}{\sigma_e^2 \left(1 + \dfrac{A'}{3\mu} \right)} \mathbf{s} \otimes \mathbf{s} \right) : \dot{\boldsymbol{\varepsilon}} \end{cases} \quad \text{[eq. 6.37]}$$

depends on $\boldsymbol{\sigma}$

- Issue: $\Delta\boldsymbol{\varepsilon}$ is finite, but $\dot{\boldsymbol{\varepsilon}}$ is infinitesimal.
- Pure incremental (forward Euler) approach:

$$\Delta\boldsymbol{\sigma}_n = \begin{cases} elastic \rightarrow \mathbf{C} : \Delta\boldsymbol{\varepsilon}_t \\ plastic \rightarrow \left(\mathbf{C} - \dfrac{3\mu}{\sigma_{e(n-1)}^2 \left(1 + \dfrac{A'_{n-1}}{3\mu} \right)} \mathbf{s}_{n-1} \otimes \mathbf{s}_{n-1} \right) : \Delta\boldsymbol{\varepsilon}_t \end{cases}$$

➡ accumulation of error

➡ **drift from yield surface**

$\boldsymbol{\sigma}_{n-1}$ $\Delta\boldsymbol{\varepsilon}_t$ $\varepsilon_{ps(n-1)}$

$\boldsymbol{\varepsilon}_{n-1}$ $\boldsymbol{\varepsilon}^p_{n-1}$ $\boldsymbol{\varepsilon}^e_{n-1}$ $etc...$

→

Stress update (integration) algorithm

→

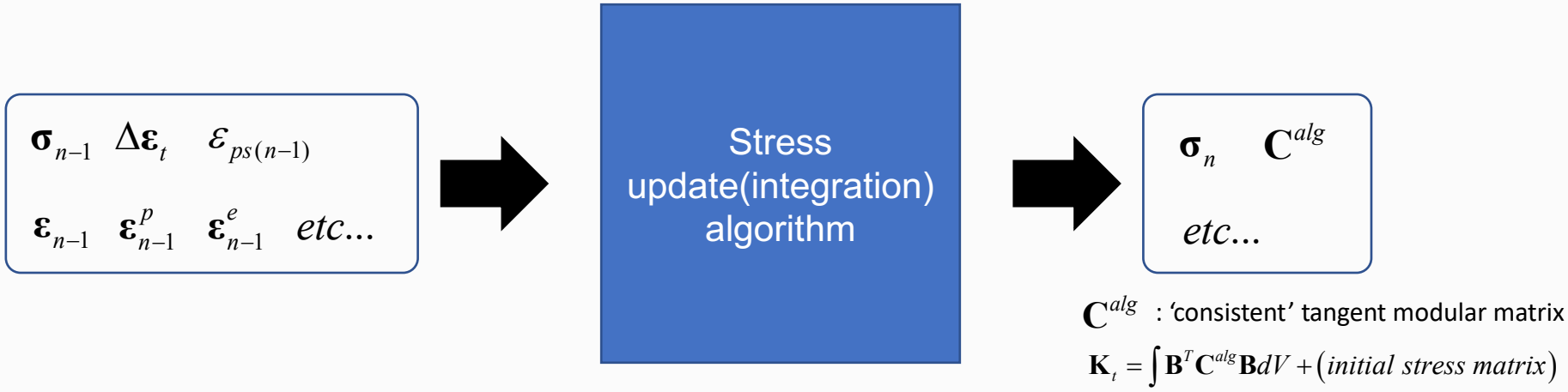
$\boldsymbol{\sigma}_n$ $\mathbf{C}^{alg}$

$etc...$

$\mathbf{C}^{alg}$: 'consistent' tangent modular matrix

$$\mathbf{K}_t = \int \mathbf{B}^T \mathbf{C}^{alg} \mathbf{B} dV + \left(initial\ stress\ matrix\right)$$

```
1  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
2  CCCCCCCCCCCCCCC     Abaqus User Subroutine Interface     CCCCCCCCCCCCCCC
3  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
4
5        SUBROUTINE UMAT(STRESS,STATEV,DDSDDE,SSE,SPD,SCD,
6       1 RPL,DDSDDT,DRPLDE,DRPLDT,
7       2 STRAN,DSTRAN,TIME,DTIME,TEMP,DTEMP,PREDEF,DPRED,CMNAME,
8       3 NDI,NSHR,NTENS,NSTATV,PROPS,NPROPS,COORDS,DROT,PNEWDT,
9       4 CELENT,DFGRD0,DFGRD1,NOEL,NPT,LAYER,KSPT,JSTEP,KINC)
10 C
11       INCLUDE 'ABA_PARAM.INC'
12 C
13       CHARACTER*80 CMNAME
14       DIMENSION STRESS(NTENS),STATEV(NSTATV),
15      1 DDSDDE(NTENS,NTENS),DDSDDT(NTENS),DRPLDE(NTENS),
16      2 STRAN(NTENS),DSTRAN(NTENS),TIME(2),PREDEF(1),DPRED(1),
17      3 PROPS(NPROPS),COORDS(3),DROT(3,3),DFGRD0(3,3),DFGRD1(3,3),
18      4 JSTEP(4)
19
20                                  C^alg   σ_n
21       ! user coding to define DDSDDE, STRESS, STATEV, SSE, SPD, SCD
22       ! and, if necessary, RPL, DDSDDT, DRPLDE, DRPLDT, PNEWDT
23
24
25       RETURN
26       END
```



- 3 categories of stress update(integration) algorithm:

  1. Forward Euler scheme + return to yield surface
  2. Sub-increment scheme
  3. **Backward (or mid-point) Euler scheme**

- **Even though backward Euler scheme is difficult to implement, it can calculate $\mathbf{C}^{alg}$, which ensures quadratic convergence of $\Delta\mathbf{p} = -\mathbf{K}_t^{-1}\mathbf{g}$.**

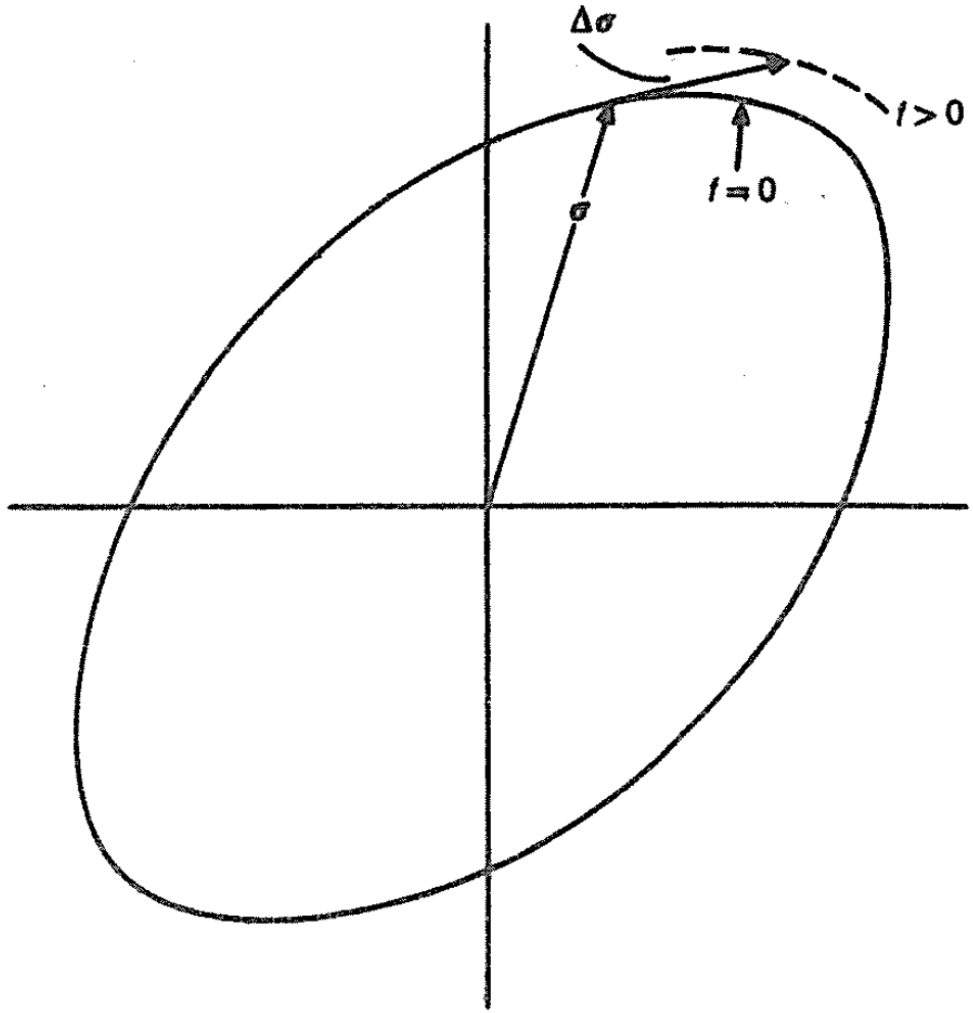
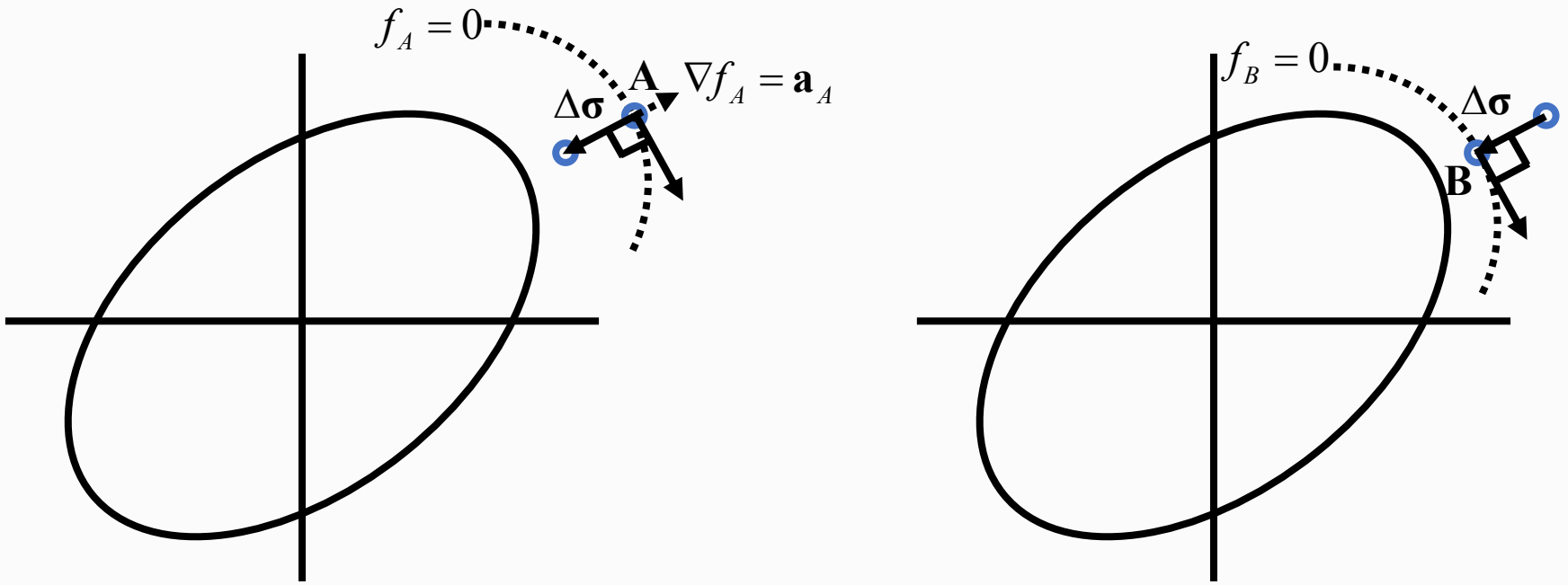


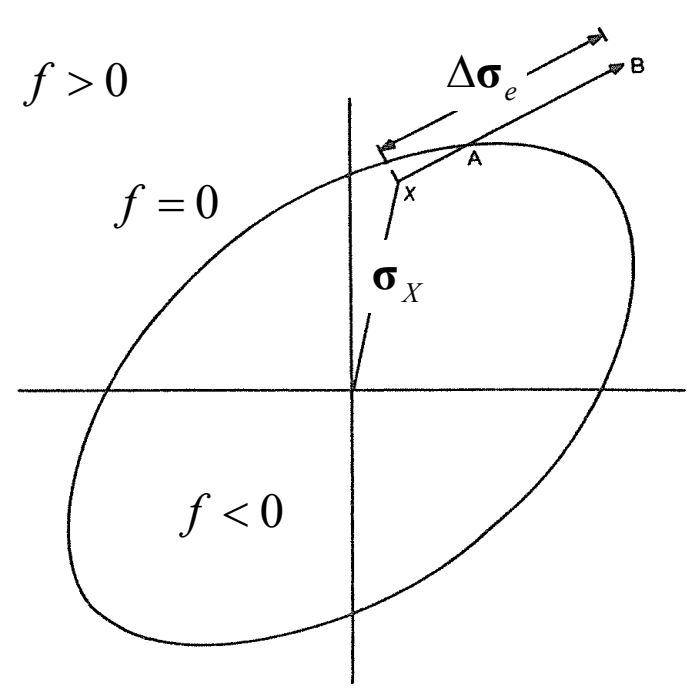
Figure 6.8 Potential accumulation of error with the forward-Euler procedure.

Representation of normality rule (flow rule)



- Since $\nabla f = \mathbf{a}$ is strain quantity, the direction cannot be directly compared with $\Delta\boldsymbol{\sigma}$.
- In these kinds of figures, normality $\left(f_{A\,or\,B} = 0\right) \perp \Delta\boldsymbol{\sigma}$ represents $\Delta\boldsymbol{\sigma} \,||\, \left(\mathbf{Ca}_{A\,or\,B}\right)$.

6.6.1 Crossing the yield surface



[Fig 6.9 The forward-Euler procedure]
(a) Locating the intersection print, A

- Elastic increment

$$\Delta\boldsymbol{\sigma}_e = \mathbf{C}\Delta\boldsymbol{\varepsilon}_t$$

- If the elastic guess results in $f(\boldsymbol{\sigma}_X + \Delta\boldsymbol{\sigma}_e) > 0$, in some cases, the point **A** is required.

$$f(\boldsymbol{\sigma}_X + \alpha\Delta\boldsymbol{\sigma}_e) = 0 \qquad \text{[eq. 6.48]}$$

- For **Mises criterion,** α can be obtained by

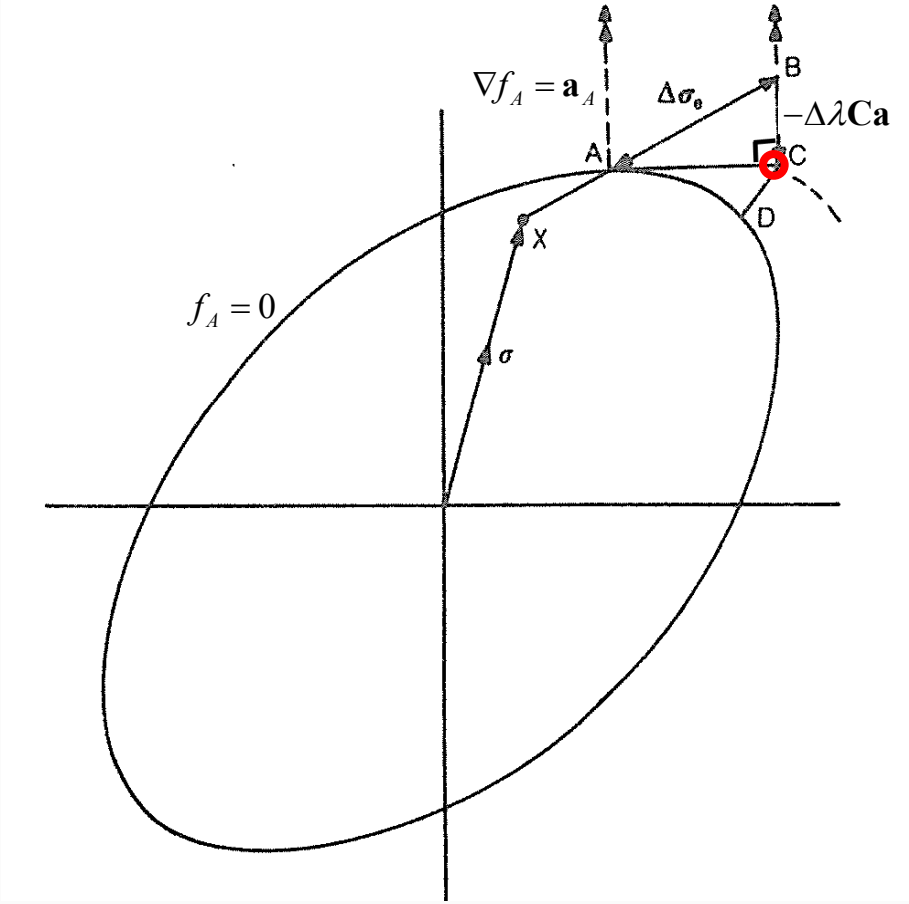
$$f = \alpha^2\sigma_e(\Delta\boldsymbol{\sigma}_e)^2 + \alpha\Delta\boldsymbol{\sigma}_e^T\mathbf{A}\boldsymbol{\sigma}_X + \sigma_e(\boldsymbol{\sigma}_X)^2 - \sigma_o^2 = 0 \qquad \text{[eq. 6.52]}$$

- For general yield criterion, α can be obtained by N-R.

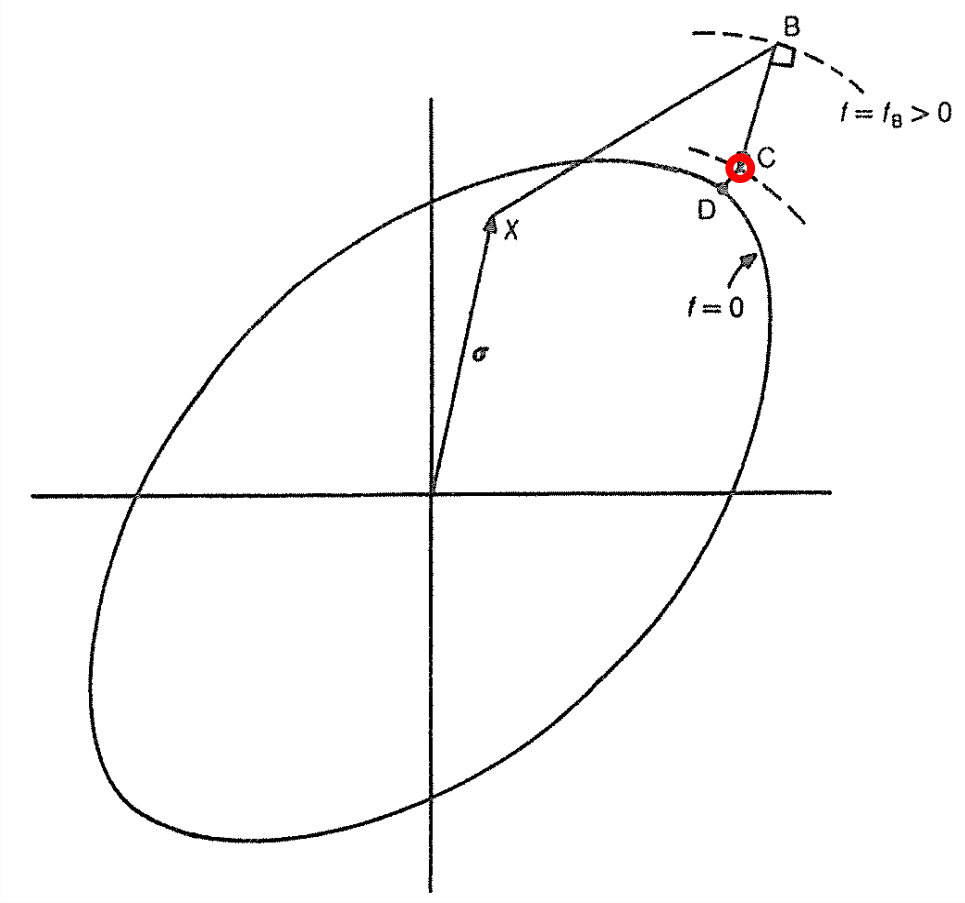
$$0 = f + \Delta f = f + \frac{\partial f}{\partial\boldsymbol{\sigma}}\frac{\partial\boldsymbol{\sigma}}{\partial\alpha}\Delta\alpha = f + \left(\mathbf{a}^T\Delta\boldsymbol{\sigma}_e\right)\Delta\alpha \qquad \text{[eq. 6.54]}$$

$$\Rightarrow \quad \alpha_n = \alpha_{n-1} - \frac{f_{n-1}}{\left(\mathbf{a}^T\Delta\boldsymbol{\sigma}_e\right)_{n-1}} \quad \text{with initial guess} \quad \alpha_0 = -\frac{f_X}{f_B - f_X} \qquad \text{[eq. 6.53]}$$

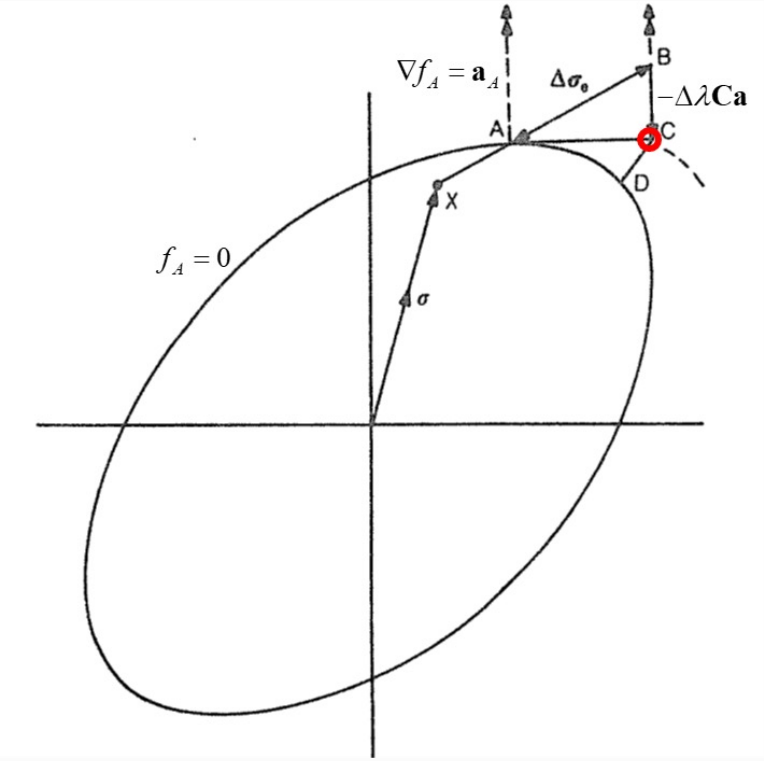
6.6.2 Two alternative 'predictors'



[Fig 6.9(b) predictor by Owen]



[Fig 6.10 alternative predictor]



[Fig 6.9(b) predictor by Owen]

$$\dot{\boldsymbol{\sigma}} = \begin{pmatrix} \dot{\sigma}_x \\ \dot{\sigma}_y \\ \dot{\sigma}_{xy} \end{pmatrix} = [\mathbf{C}]\left(\begin{pmatrix} \dot{\varepsilon}_x \\ \dot{\varepsilon}_y \\ \dot{\varepsilon}_{xy} \end{pmatrix} - \begin{pmatrix} \dot{\varepsilon}_{px} \\ \dot{\varepsilon}_{py} \\ \dot{\varepsilon}_{pxy} \end{pmatrix} \right) = \mathbf{C}\left(\dot{\boldsymbol{\varepsilon}}_t - \dot{\boldsymbol{\varepsilon}}_p\right) = \mathbf{C}\left(\dot{\boldsymbol{\varepsilon}}_t - \dot{\lambda}\mathbf{a}\right) \quad \text{[eq. 6.5]}$$

$$\dot{\lambda} = \frac{\mathbf{a}^T\mathbf{C}\dot{\boldsymbol{\varepsilon}}}{\mathbf{a}^T\mathbf{Ca} + A'} = \frac{\mathbf{a}:\mathbf{C}:\dot{\boldsymbol{\varepsilon}}}{\mathbf{a}:\mathbf{C}:\mathbf{a} + A'} \quad \text{[eq. 6.17]}$$

- Move from **X** to **A** by previously obtained α .

$$\boldsymbol{\sigma} \leftarrow \boldsymbol{\sigma} + \mathbf{C}\left(\alpha\Delta\boldsymbol{\varepsilon}_t\right)$$
$$\Delta\boldsymbol{\varepsilon}_t \leftarrow (1-\alpha)\Delta\boldsymbol{\varepsilon}_t$$

- Use forward Euler method from eq.6.5.

$$\dot{\boldsymbol{\sigma}} = \mathbf{C}\left(\dot{\boldsymbol{\varepsilon}}_t - \dot{\boldsymbol{\varepsilon}}_p\right) = \mathbf{C}\left(\dot{\boldsymbol{\varepsilon}}_t - \dot{\lambda}\mathbf{a}\right) \quad \text{[eq. 6.5]}$$

$$\Rightarrow \Delta\boldsymbol{\sigma} = \mathbf{C}\Delta\boldsymbol{\varepsilon}_t - \Delta\lambda\mathbf{Ca}_A \quad \text{[eq. 6.56]}$$

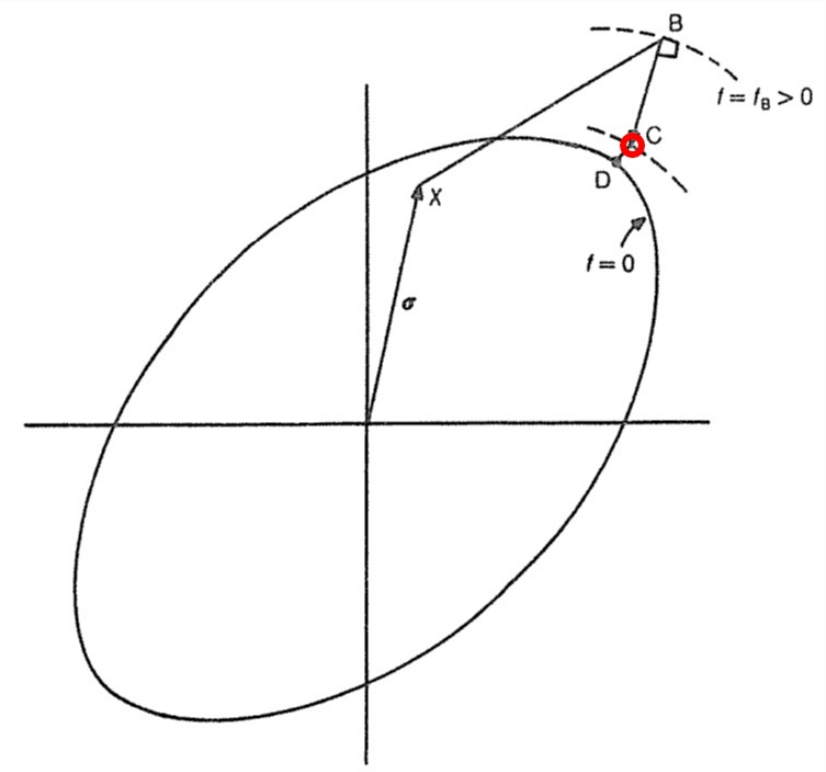
$$\dot{\lambda} = \frac{\mathbf{a}^T\mathbf{C}\dot{\boldsymbol{\varepsilon}}}{\mathbf{a}^T\mathbf{Ca} + A'} = \frac{\mathbf{a}:\mathbf{C}:\dot{\boldsymbol{\varepsilon}}}{\mathbf{a}:\mathbf{C}:\mathbf{a} + A'} \Rightarrow \Delta\lambda = \frac{\mathbf{a}^T\mathbf{C}\Delta\boldsymbol{\varepsilon}}{\mathbf{a}^T\mathbf{Ca} + A'} = \frac{\mathbf{a}:\mathbf{C}:\Delta\boldsymbol{\varepsilon}}{\mathbf{a}:\mathbf{C}:\mathbf{a} + A'}$$

- **A → B** is called **elastic increment**, and **B → C** is called **plastic return**.

$$\boldsymbol{\sigma}_C = \boldsymbol{\sigma}_A \underbrace{+\Delta\boldsymbol{\sigma}_e}_{A\to B} \underbrace{-\Delta\lambda\mathbf{Ca}_A}_{B\to C} \quad \text{[eq. 6.57]}$$

- **B → C** is normal to $f_A = 0$

- **C** is Owen's predictor.



[Fig 6.10 alternative predictor]

$$\dot{\varepsilon}_{ps} = B(\boldsymbol{\sigma})\dot{\lambda} \quad \text{[eq. 6.15]}$$
$$B(\boldsymbol{\sigma}) = 1$$
$$\Rightarrow \Delta\varepsilon_{ps} = \Delta\lambda$$

- Do not use α , just calculate **B** by elastic trial.

$$\boldsymbol{\sigma}_B = \boldsymbol{\sigma}_X + \mathbf{C}\Delta\boldsymbol{\varepsilon}_t$$

- Actual stress state is

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}_X + \mathbf{C}\left(\Delta\boldsymbol{\varepsilon}_t - \Delta\boldsymbol{\varepsilon}_p\right) = \boldsymbol{\sigma}_B - \mathbf{C}\Delta\boldsymbol{\varepsilon}_p$$
$$= \boldsymbol{\sigma}_B - \mathbf{C}\left(\Delta\lambda\mathbf{a}\right) \quad \Leftarrow \quad \Delta\boldsymbol{\varepsilon}_p = \Delta\lambda\frac{\partial f}{\partial\boldsymbol{\sigma}} = \Delta\lambda\mathbf{a}$$

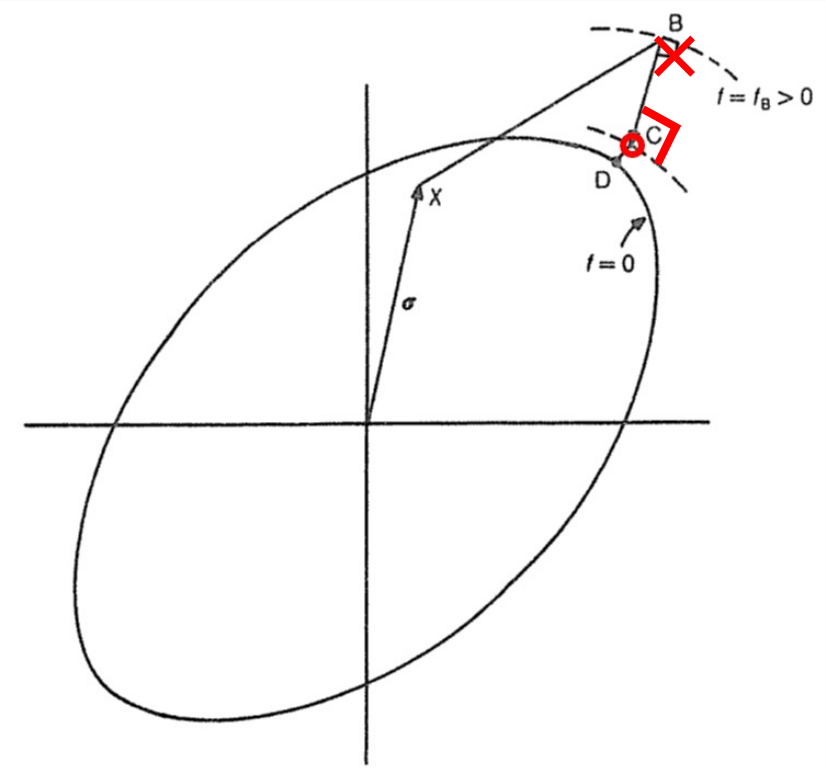
- For now, let's just think f as function of $\Delta\lambda$ and start N-R.

$$0 = f_B + \Delta f = f_B + \left(\frac{\partial f}{\partial\boldsymbol{\sigma}}\right)^T \Delta\boldsymbol{\sigma} + \left(\frac{\partial f}{\partial\varepsilon_{ps}}\right)^T \Delta\varepsilon_{ps}$$

$$\Leftarrow \quad f = \sigma_e - \sigma_o(\varepsilon_{ps}) \ , \ \frac{\partial\sigma_o}{\partial\varepsilon_{ps}} = A' \ , \ \Delta\varepsilon_{ps} = \Delta\lambda$$

$$= f_B - \mathbf{a}_B^T\left(\mathbf{C}\left(\Delta\lambda\mathbf{a}_B\right)\right) - A'_B\Delta\lambda = f_B - \left(\mathbf{a}_B^T\mathbf{C}\mathbf{a}_B + A'_B\right)\Delta\lambda \quad \text{[eq. 6.58]}$$

$$\Rightarrow \quad \Delta\lambda = \frac{f_B}{\mathbf{a}_B^T\mathbf{C}\mathbf{a}_B + A'_B} \quad \text{[eq. 6.59]} \quad \Rightarrow \quad \boldsymbol{\sigma}_C = \boldsymbol{\sigma}_B - \Delta\lambda\mathbf{C}\mathbf{a}_B \quad \text{[eq. 6.60]}$$



[Fig 6.10 alternative predictor]

- For **C** to represent actual stress state, $\Delta\lambda\mathbf{Ca}$ should be perpendicular to $f_C = 0$, not $f_B = 0$.

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}_X + \mathbf{C}\left(\Delta\boldsymbol{\varepsilon}_t - \left(\Delta\boldsymbol{\varepsilon}_p\right)_C\right) = \boldsymbol{\sigma}_B - \mathbf{C}\left(\Delta\boldsymbol{\varepsilon}_p\right)_C$$

$$= \boldsymbol{\sigma}_B - \mathbf{C}\left(\Delta\lambda\mathbf{a}_C\right) \quad \Leftarrow \quad \left(\Delta\boldsymbol{\varepsilon}_p\right)_C = \Delta\lambda\left(\frac{\partial f}{\partial\boldsymbol{\sigma}}\right)_C = \Delta\lambda\mathbf{a}_C$$

$$\Rightarrow \quad \Delta\lambda = \frac{f_B}{\mathbf{a}_B^T\mathbf{C}\mathbf{a}_C + A'_B} \quad \Rightarrow \quad \boldsymbol{\sigma}_C = \boldsymbol{\sigma}_B - \Delta\lambda\mathbf{C}\mathbf{a}_C$$

- However, this is nonsense:
  To obtain $\boldsymbol{\sigma}_C$, we have to know $\boldsymbol{\sigma}_C$.

- This can be solved by **full backward Euler method** (section 6.6.6).

- $\mathbf{a}_B \parallel \mathbf{a}_C$ in 3D von Mises criterion:
  The full backward Euler method can be simplified to **radial return algorithm** (section 6.6.7).

Thank you!