19

Purely Incremental Methods: Load Control

Purely Incremental Solution Methods

These are predictor-only methods that lack a corrective phase.

They are still used frequently in path dependent problems that involve material nonlinearity, e.g., plasticity, viscoelasticity and fracture.

In this Lecture we will cover such methods under stage parameter control, a.k.a. load control, for simplicity of description.

Source ODE For Purely Incremental Methods

Recall the first-order rate equation

$$\dot{\mathbf{r}} = \mathbf{K}\,\dot{\mathbf{u}} - \mathbf{q}\,\dot{\lambda} = \mathbf{0}$$

Taking pseudotime $t = \lambda$, **this becomes**

$$\boxed{\mathbf{r}' = \mathbf{K}\,\mathbf{u}' - \mathbf{q} = \mathbf{0}}$$

where primes denote differentiation wrt λ. If K is nonsingular we can solve for u':

$$\boxed{\mathbf{u}' = \frac{d\mathbf{u}}{d\lambda} = \mathbf{K}^{-1}\mathbf{q} = \mathbf{v}}$$

Integrate numerically this first-order ODE in λ from the IC:

$$\boxed{\mathbf{u} = \mathbf{u}_0 \quad \text{at} \quad \lambda = 0}$$

Source of Drift Error

The exact integral of

$$\boxed{\mathbf{u}' = \frac{d\mathbf{u}}{d\lambda} = \mathbf{K}^{-1}\mathbf{q} = \mathbf{v}}$$

with the IC

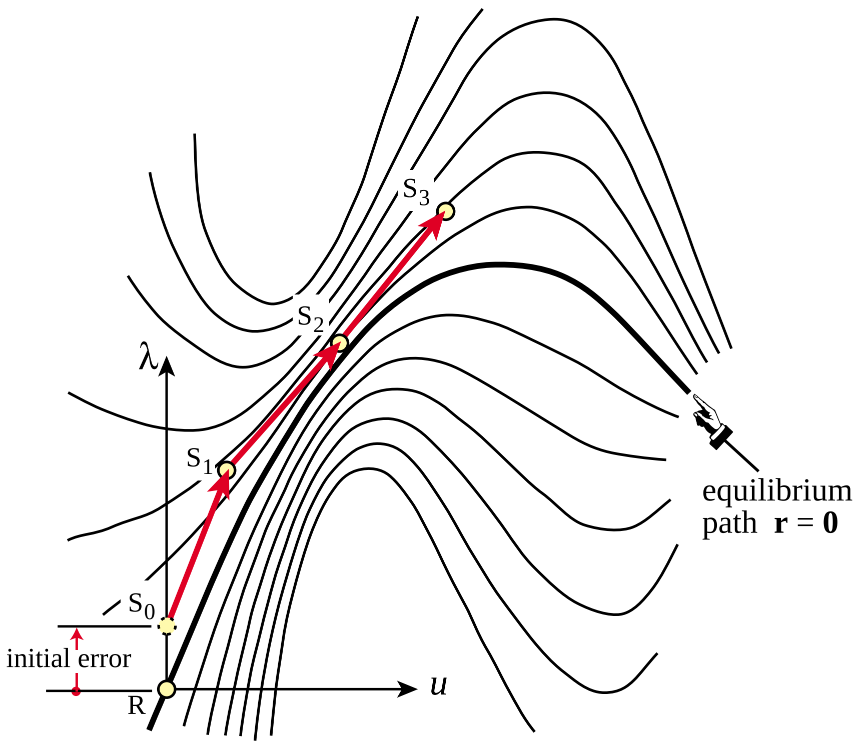
$$\boxed{\mathbf{u} = \mathbf{u}_0 \quad \text{at} \quad \lambda = 0}$$

is

$$\boxed{\mathbf{r}(\mathbf{u}, \lambda) = \mathbf{r}_0}$$

in which $\mathbf{r}_0 = \mathbf{r}(\mathbf{u}_0, 0)$ is an initial residual error. Thus initial errors do not decay, as pictured on the next slide. This is the source of the drift error.

Initial Error Does Not Decay



Consequence: Solution "Drifts" to Neighboring Non-equilibrium Paths

If the initial error r_0 is nonzero, then even if the stepsize goes to zero, the computed solution will not converge to the equilibrium path, but to another path in the incremental flow.

Integration Scheme: Forward Euler

Basic formula

$$\boxed{\mathbf{u}_{n+1} = \mathbf{u}_n + \Delta\lambda\, \mathbf{u}'_n}$$

in which

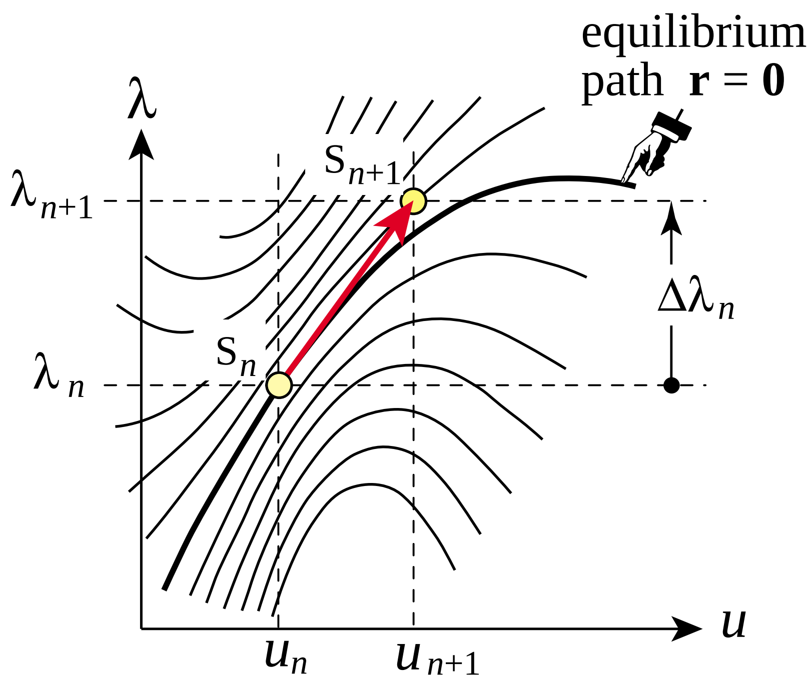
$$\mathbf{u}_n \stackrel{\text{def}}{=} \mathbf{u}(\lambda_n) \qquad \Delta\lambda_n = \lambda_{n+1} - \lambda_n$$

Advancing scheme

$$\boxed{\begin{aligned} \Delta\mathbf{u}_n &= \mathbf{K}_n^{-1}\mathbf{q}_n\, \Delta\lambda_n = \mathbf{v}_n\, \Delta\lambda_n \\ \mathbf{u}_{n+1} &= \mathbf{u}_n + \Delta\mathbf{u}_n \end{aligned}}$$

Picture on next slide

Forward Euler Schematics



Forward Euler Implementation Pseudocode

Set IC: $\mathbf{u}_0$ **= ref state,** $\lambda = 0$

For n **=** 0, 1, ... **do**

Form tangent stiffness $\mathbf{K}_n$ **and incremental load vector** $\mathbf{q}_n$ **at the last solution** $(\mathbf{u}_n, \lambda_n)$

Factor $\mathbf{K}_n$ **and solve** $\mathbf{K}_n \Delta\mathbf{u}_n = \mathbf{q}_n \Delta\lambda_n$

Advance $\mathbf{u}_{n+1} = \mathbf{u}_n + \Delta\mathbf{u}_n$ **and** $\lambda_{n+1} = \lambda_n + \Delta\lambda_n$

Increment n

More Refined Integration Methods

- **Midpoint rule**
- **Runge-Kutta (RK) 3 and 4 step schemes**

Midpoint Rule (a.k.a. Heun' Method)

$$\mathbf{u}_{n+1/2} = \mathbf{u}_n + \tfrac{1}{2}\mathbf{K}_n^{-1}\mathbf{q}_n\,\Delta\lambda_n$$

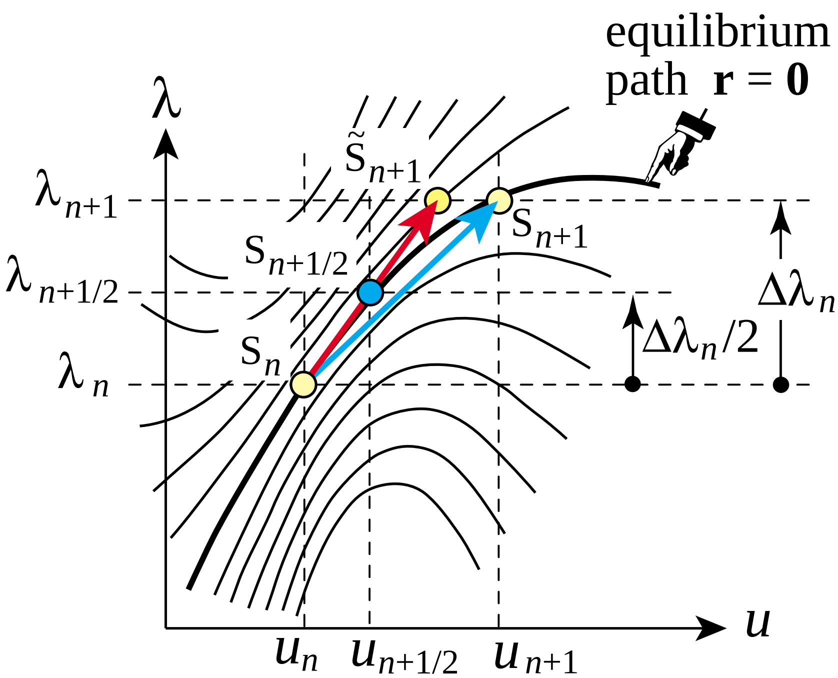
$$\mathbf{K}_{n+1/2} \stackrel{\text{def}}{=} \mathbf{K}(\mathbf{u}_{n+1/2}), \qquad \mathbf{q}_{n+1/2} \stackrel{\text{def}}{=} \mathbf{q}(\mathbf{u}_{n+1/2})$$

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \mathbf{K}_{n+1/2}^{-1}\,\mathbf{q}_{n+1/2}\,\Delta\lambda_n.$$

Schematics on the next slide

Note: do not confuse Midpoint Rule with Trapezoidal Rule. For linear problems these rules coalesce, but are different in the case of nonlinear problems

Midpoint Rule Schematics



Midpoint Rule Implementation Pseudocode

Set IC: $\mathbf{u}_0$ **= ref state,** $\lambda = 0$

For $n = 0, 1, \ldots$ **do**

Perform a FE halfstep advancing to $\lambda_{n+1/2} = \lambda_n + \Delta\lambda_n/2$

Form tangent stiffness matrix $\mathbf{K}_{n+1/2}$ **and incremental load vector** $\mathbf{q}_{n+1/2}$ **at the halfstep solution (**$\mathbf{u}_{n+1/2}, \lambda_{n+1/2}$**)**

Factor $\mathbf{K}_{n+1/2}$ **and solve** $\mathbf{K}_{n+1/2}\,\Delta\mathbf{u} = \mathbf{q}_{n+1/2}\,\Delta\lambda_n$

Advance $\mathbf{u}_{n+1} = \mathbf{u}_n + \Delta\mathbf{u}_n$ **and** $\lambda_{n+1} = \lambda_n + \Delta\lambda_n$

Increment n **and cycle**

Good Feature of Midpoint Rule

Recommended method for plasticity when there is possibility of local unloading.

Why: unloading can be checked at the midpoint station and tangent stiffness adjusted accordingly.
Trapezoidal Rule goes completely wrong in that case.

Two More Topics Addressed in Chapter 19

- **Numerical Stability**
- **Accuracy by stepsize adaptation**

Neither is very important today per se, since this kind of stepsize control strategy has been superseded by more advanced one such as arclength. These will be covered in the next Lecture