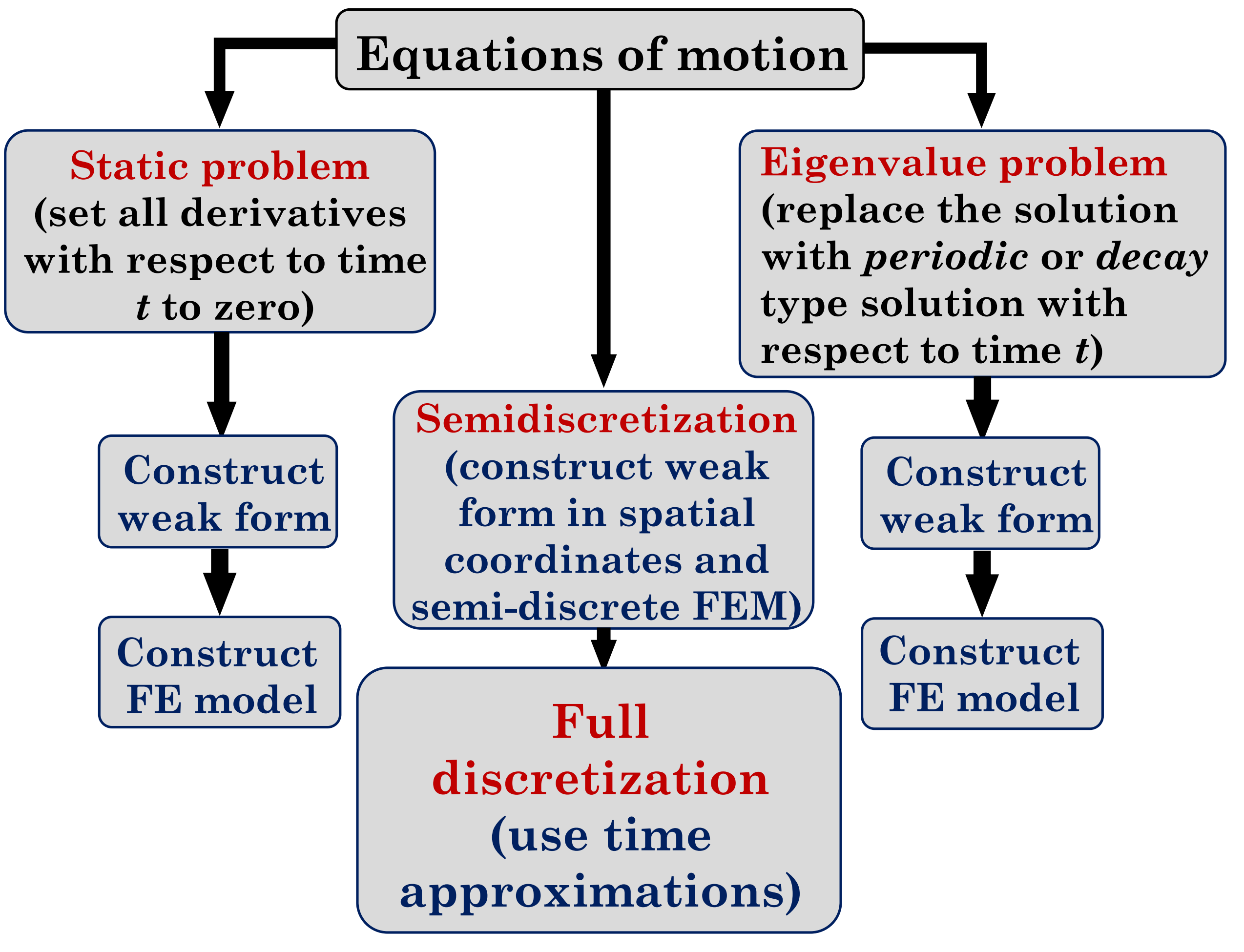
MEEN 673: NONLINEAR FINITE ELEMENT ANALYSIS

Time-Dependent Problems

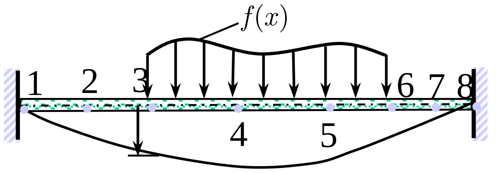
CONTENTS

INTRODUCTION



INTRODUCTION (continued)

Equilibrium (static) problem



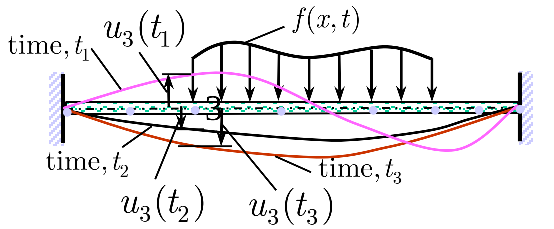
Equilibrium configuration

Equation of equilibrium

$$-\frac{d}{dx}\left(T\frac{du}{dx}\right) = f(x)$$

$$u(0) = 0, \quad u(L) = 0$$

Transient problem



Transient configurations

Equation of motion

$$\frac{\partial}{\partial t}\left(\rho\frac{\partial u}{\partial t}\right) - \frac{\partial}{\partial x}\left(T\frac{\partial u}{\partial x}\right) = f(x,t)$$

$$\text{B.C.}: \; u(0,t) = 0, \qquad u(L,t) = 0$$

$$\text{I.C.}: \; u(x,0) = u_0(x), \;\; \dot{u}(x,0) = v_0(x)$$

TRANSIENT ANALYSIS (steps involved)

Model Equation

$$c_1 \frac{\partial u}{\partial t} + c_2 \frac{\partial^2 u}{\partial t^2} - \frac{\partial}{\partial x}\left(a(x,u) \frac{\partial u}{\partial x} \right) + \frac{\partial^2}{\partial x^2}\left(b(x,u) \frac{\partial^2 u}{\partial x^2} \right) = f(x,t)$$

Approximate solution

$$u(x,t) \approx u_h(x,t) = \sum_{j=1}^{n} \Delta_j(t)\,\varphi_j(x)$$

1. Spatial approximation (semidiscretization)

$$\mathbf{C}\dot{\Delta} + \mathbf{M}\ddot{\Delta} + \mathbf{K}(\Delta)\Delta = \mathbf{F}$$

2. Time approximation (full discretization)

$$\hat{\mathbf{K}}(\Delta_s, \Delta_{s+1})\,\Delta_{s+1} = \mathbf{F}_{s,s+1}$$

SPATIAL APPROXIMATION

Model Equation

$$c_1 \frac{\partial u}{\partial t} + c_2 \frac{\partial^2 u}{\partial t^2} - \frac{\partial}{\partial x}\left(a \frac{\partial u}{\partial x}\right) + \frac{\partial^2}{\partial x^2}\left(b \frac{\partial^2 u}{\partial x^2}\right) = f(x,t)$$

Weak Form for semi-discretization

$$0 = \int_{x_a}^{x_b} w_i(x)\left[c_1 \frac{\partial u_h}{\partial t} + c_2 \frac{\partial^2 u_h}{\partial t^2} - \frac{\partial}{\partial x}\left(a \frac{\partial u_h}{\partial x}\right) + \frac{\partial^2}{\partial x^2}\left(b \frac{\partial u_h}{\partial x^2}\right) - f(x,t)\right] dx$$

$$= \int_{x_a}^{x_b} \left[c_1 w_i \frac{\partial u_h}{\partial t} + c_2 w_i \frac{\partial^2 u_h}{\partial t^2} + \frac{dw_i}{dx}\left(a \frac{\partial u_h}{\partial x}\right) + \frac{d^2 w_i}{dx^2}\left(b \frac{\partial^2 u_h}{\partial x^2}\right) - w_i f(x,t)\right] dx$$

$$- w_i(x_a) Q_1 - w_i(x_b) Q_3 - \left(-\frac{dw_i}{dx}\right)_{x_a} Q_2 - \left(-\frac{dw_i}{dx}\right)_{x_b} Q_4$$

SPATIAL DISCRETIZATION
Finite Element Model

Approximation

$$u(x,t) \approx u_h(x,t) = \sum_{j=1}^{n} \Delta_j(t)\,\varphi_j(x)$$

Finite element model

$$\mathbf{C}\dot{\Delta} + \mathbf{M}\ddot{\Delta} + \mathbf{K}\Delta = \mathbf{F}$$

$$C_{ij}^e = \int_{x_a}^{x_b} c_1 \varphi_i \varphi_j \; dx, \quad M_{ij}^e = \int_{x_a}^{x_b} c_2 \varphi_i \varphi_j \; dx$$

$$K_{ij}^e = \int_{x_a}^{x_b} \left(b \frac{d^2\varphi_i}{dx^2} \frac{d^2\varphi_j}{dx^2} + a \frac{d\varphi_i}{dx} \frac{d\varphi_j}{dx} \right) dx$$

$$F_i^e = \int_{x_a}^{x_b} f\,\varphi_i \; dx + \varphi_i(x_a) Q_1 + \varphi_i(x_b) Q_3 + \left(-\frac{d\varphi_i}{dx} \right)_{x_a} Q_2 + \left(-\frac{d\varphi_i}{dx} \right)_{x_b} Q_4$$

TIME APPROXIMATIONS

PARABOLIC EQUATION (heat transfer, fluid mechanics, and like problems

$$\mathbf{C}\dot{\mathbf{u}} + \mathbf{K}(\mathbf{u})\mathbf{u} = \mathbf{F}$$

$$C_{ij}^e = \int_{x_a}^{x_b} c_1 \psi_i \psi_j \, dx, \quad K_{ij}^e = \int_{x_a}^{x_b} \left(a(x, u, u_x) \frac{d\psi_i}{dx} \frac{d\psi_j}{dx} \right) dx$$

$$F_i^e = \int_{x_a}^{x_b} f \, \psi_i \, dx + \psi_i(x_a) Q_1 + \psi_i(x_b) Q_2$$

HYPERBOLIC EQUATION (structural mechanics problems)

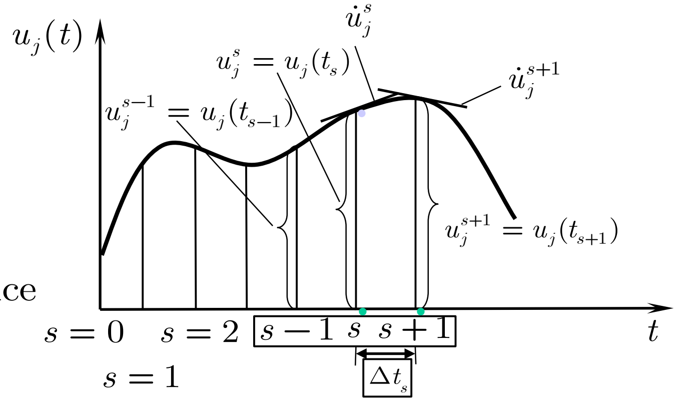
$$\mathbf{C}\dot{\Delta} + \mathbf{M}\ddot{\Delta} + \mathbf{K}(\Delta)\Delta = \mathbf{F}$$

TIME APPROXIMATIONS OF PARABOLIC EQUATIONS

Approximation of the first derivative

$$\dot{u}_j^s \approx \frac{u_j^{s+1} - u_j^s}{\Delta t_{s+1}}, \text{ forward difference}$$

$$\dot{u}_j^{s+1} \approx \frac{u_j^{s+1} - u_j^s}{\Delta t_{s+1}}, \text{ backward difference}$$



Alfa (α)-family of approximation

$$\alpha \dot{u}_j^{s+1} + (1-\alpha)\dot{u}_j^s \approx \frac{u_j^{s+1} - u_j^s}{\Delta t_{s+1}}, \qquad 0 \le \alpha \le 1$$

$$u_j^{s+1} = u_j^s + \Delta t_{s+1}\left[\alpha \dot{u}_j^{s+1} + (1-\alpha)\dot{u}_j^s\right]$$

TIME APPROXIMATIONS (Parabolic)

Alfa-family of approximation (*Parabolic equation*)

$$\mathbf{C}\dot{\mathbf{u}} + \mathbf{K}(\mathbf{u})\mathbf{u} = \mathbf{F},\ \ 0 < t < T$$

$$\Rightarrow \mathbf{C}\dot{\mathbf{u}}^{s} + \mathbf{K}_{s}\mathbf{u}^{s} = \mathbf{F}_{s},\ \mathbf{C}\dot{\mathbf{u}}^{s+1} + \mathbf{K}_{s+1}\mathbf{u}^{s+1} = \mathbf{F}_{s+1}$$

$$\mathbf{u}^{s+1} = \mathbf{u}^{s} + \Delta t_{s+1}\left[\alpha\dot{\mathbf{u}}^{s+1} + (1-\alpha)\dot{\mathbf{u}}^{s}\right]$$

$$\mathbf{C}\mathbf{u}^{s+1} = \mathbf{C}\mathbf{u}^{s} + \Delta t_{s+1}\left[\alpha\mathbf{C}\dot{\mathbf{u}}^{s+1} + (1-\alpha)\mathbf{C}\dot{\mathbf{u}}^{s}\right]$$

$$\mathbf{C}\dot{\mathbf{u}}^{s+1} = \mathbf{F}^{s+1} - \mathbf{K}_{s+1}\mathbf{u}^{s+1} \qquad \mathbf{C}\dot{\mathbf{u}}^{s} = \mathbf{F}^{s} - \mathbf{K}_{s}\mathbf{u}^{s}$$

$$\left(\mathbf{C} + \alpha\Delta t_{s+1}\mathbf{K}_{s+1}\right)\mathbf{u}^{s+1} = \left(\mathbf{C} - (1-\alpha)\Delta t_{s}\mathbf{K}_{s}\right)\mathbf{u}^{s} + \Delta t_{s+1}\left[\alpha\mathbf{F}^{s+1} + (1-\alpha)\mathbf{F}^{s}\right]$$

$$\hat{\mathbf{K}}_{s+1}\mathbf{u}^{s+1} = \hat{\mathbf{F}}^{s+1}$$

where

$$\hat{\mathbf{K}}_{s+1} = \alpha\Delta t_{s+1}\mathbf{K}_{s+1} + \mathbf{C},$$

$$\hat{\mathbf{F}}^{s+1} = \left[(1-\alpha)\Delta t_{s+1}\mathbf{K}_{s+1} + \mathbf{C}\right]\mathbf{u}^{s} + \Delta t_{s+1}\left[\alpha\mathbf{F}^{s+1} + (1-\alpha)\mathbf{F}^{s}\right]$$

TIME APPROXIMATIONS (Hyperbolic)

Semidiscrete FE model

$$\mathbf{C}^e\dot{\mathbf{u}}^e + \mathbf{M}^e\ddot{\mathbf{u}}^e + \mathbf{K}^e(\mathbf{u}^e)\mathbf{u}^e = \mathbf{F}^e$$

Newmark scheme (*hyperbolic equations*)

$$\mathbf{u}^{s+1} = \mathbf{u}^s + \Delta t\,\dot{\mathbf{u}}^s + \tfrac{1}{2}(\Delta t)^2\,\ddot{\mathbf{u}}^{s,\gamma}$$

$$\dot{\mathbf{u}}^{s+1} = \dot{\mathbf{u}}^s + \Delta t\,\ddot{\mathbf{u}}^{s,\alpha},\quad \ddot{\mathbf{u}}^{s,\theta} \equiv (1-\theta)\ddot{\mathbf{u}}^s + \theta\,\ddot{\mathbf{u}}^{s+1}$$

Fully discretized model

$$\hat{\mathbf{K}}_{s+1}\mathbf{u}^{s+1} = \hat{\mathbf{F}}^{s+1},\qquad \hat{\mathbf{K}}_{s+1} = \mathbf{K}_{s+1} + a_3\mathbf{M}_{s+1} + a_5\mathbf{C}_{s+1}$$

$$\hat{\mathbf{F}}^{s+1} = \mathbf{F}^{s+1} + \mathbf{M}_{s+1}\left(a_3\mathbf{u}^s + a_4\dot{\mathbf{u}}^s + a_5\ddot{\mathbf{u}}^s\right) + \mathbf{C}_{s+1}\left(a_5\mathbf{u}^s + a_6\dot{\mathbf{u}}^s + a_7\ddot{\mathbf{u}}^s\right)$$

EXPLICIT AND IMPLICIT FORMULATIONS

General form of the time-marching scheme

$$\hat{\mathbf{K}}\mathbf{u}^{s+1} = \mathbf{B}\mathbf{u}^{s} + \mathbf{F}^{s,s+1}$$

The scheme is called *explicit* if the coefficient matrix $\hat{\mathbf{K}}$ is diagonal (and hence, no inversion of equations is necessary); otherwise, the scheme is said to be implicit.

$$\hat{\mathbf{K}}^{\mathbf{s+1}}\mathbf{u}^{s+1} = \mathbf{F}^{s+1}$$

$$\hat{\mathbf{K}}^{s+1} = \alpha \Delta t_{s+1} \mathbf{K}_{s+1} + \mathbf{C}, \; \hat{\mathbf{F}}^{s+1} = \left[(1-\alpha)\Delta t_{s+1}\mathbf{K}_{s+1} + \mathbf{C}\right]\mathbf{u}^{s} + \Delta t_{s+1}\left[\alpha \mathbf{F}^{s+1} + (1-\alpha)\mathbf{F}^{s}\right]$$

The alfa-family scheme is *explicit* if and only if

$$(1)\;\; \alpha = 0 \quad \text{and} \quad (2)\;\; \mathbf{C} \;\; \text{is diagonal}.$$

ROW-SUM MASS LUMPING

For the Lagrange linear and quadratic elements we have

$$[M^e]_C = \frac{\rho A_e h_e}{6}\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}, \quad [M^e]_C = \frac{\rho A_e h_e}{30}\begin{bmatrix} 4 & 2 & -1 \\ 2 & 16 & 2 \\ -1 & 2 & 4 \end{bmatrix}$$

$$[M^e]_L = \frac{\rho A_e h_e}{2}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad [M^e]_L = \frac{\rho A_e h_e}{6}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

STABILITY OF APPROXIMATIONS

$$\hat{\mathbf{K}}^{s+1}\mathbf{u}^{s+1} = \bar{\mathbf{K}}\mathbf{u}^{s} + \mathbf{F}^{s,s+1}$$

$$\mathbf{u}^{s+1} = \mathbf{A}\mathbf{u}^{s} + \mathbf{B}^{s,s+1}$$

The scheme is called *stable* if the repeated solution of the above equation does not result in unbounded solution $\mathbf{u}_{s+1}$. The necessary and sufficient condition for the above scheme to be stable is that the maximum eigenvalue of the coefficient matrix **A** is less than unity:

$$\lambda_{\max}^{A} \leq 1$$

STABILITY OF APPROXIMATIONS
(continued)

Alfa-family of approximation scheme

$\alpha \geq \frac{1}{2}$, the scheme is *stable*

$\alpha < \frac{1}{2}$, the scheme is *conditionally stable*

$$\left(-\lambda \mathbf{C} + \mathbf{K}\right)\mathbf{u} = \mathbf{0}$$

$$\text{Stability condition: } \Delta t \leq (\Delta t)_{\text{crit}} = \frac{2}{(1-2\alpha)\lambda_{\max}}$$

$\alpha = 0.0$, Forward difference (Euler) scheme (conditionally stable)

$\alpha = 0.5$, Crank-Nicolson's scheme (stable)

$\alpha = \dfrac{2}{3}$, Galerkin's scheme (stable)

$\alpha = 1.0$, Backward difference scheme (stable)

STABILITY OF APPROXIMATIONS
(continued)

Newmark's scheme for Structural Dynamics

$$\left(-\lambda \mathbf{M} + \mathbf{K}\right)\mathbf{u} = \mathbf{0}$$

$$\text{Stability condition:} \quad \Delta t \leq (\Delta t)_{\text{crit}} = \frac{2}{(\alpha - \gamma)\lambda_{\max}}$$

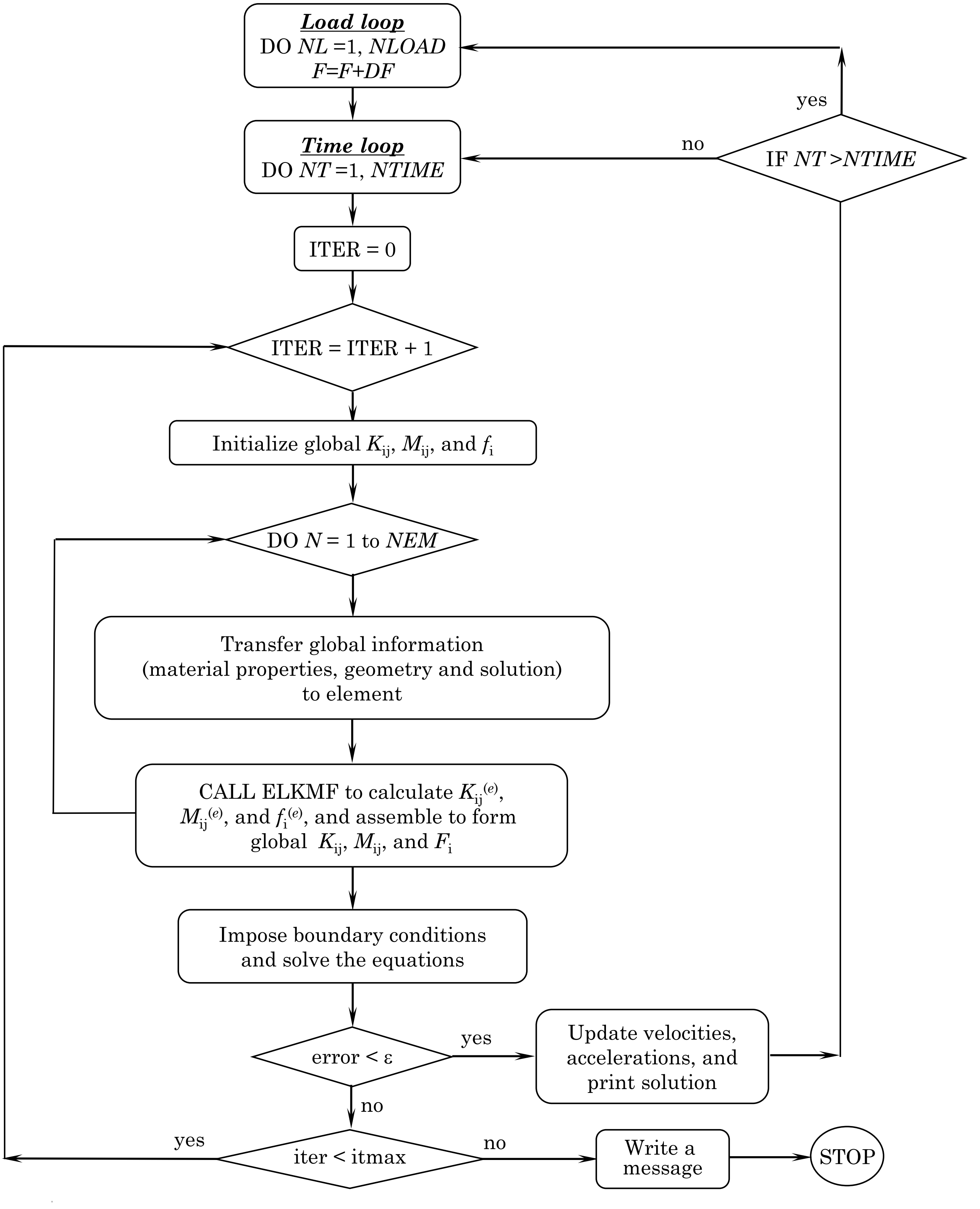
$\alpha = 0.5,\ \gamma = 2\beta = 0.5,$ Constant-average acceleration scheme (stable)

$\alpha = 0.5,\ \gamma = 2\beta = \frac{1}{3},$ Linear acceleration scheme (conditionally stable)

$\alpha = 1.5,\ \gamma = 2\beta = 1.6,$ Galerkin's scheme (stable)

$\alpha = 1.5,\ \gamma = 2\beta = 2.0,$ Backward difference scheme (stable)

$(\Delta t)_{\text{crit}}$ gets smaller as the mesh is refined.


Load loop
DO NL =1, NLOAD
F=F+DF
Time loop
DO NT =1, NTIME
IF NT >NTIME
yes
no
ITER = 0
ITER = ITER + 1
Initialize global K_{ij}, M_{ij}, and f_i
DO N = 1 to NEM
Transfer global information (material properties, geometry and solution) to element
CALL ELKMF to calculate $K_{ij}^{(e)}$, $M_{ij}^{(e)}$, and $f_i^{(e)}$, and assemble to form global K_{ij}, M_{ij}, and F_i
Impose boundary conditions and solve the equations
error < ε
yes
Update velocities, accelerations, and print solution
no
iter < itmax
yes
no
Write a message
STOP

```
C
C     Read the necessary data for time-dependent problems
C
      IF(ITEM.NE.0) THEN
          READ(IN,*) C0,CX,CY
          WRITE(ITT,820)
          WRITE(ITT,540) C0,CX,CY
          READ(IN,*) NTIME
          READ(IN,*) DT,ALFA,GAMA,EPSLN
          A1=ALFA*DT
          A2=(1.0-ALFA)*DT
          DO 40 I=1,NEQ
   40     GLU(I)=0.0
          IF(ITEM.EQ.1) THEN
              IF(NSSV.NE.0) THEN
                  DO 50 I=1,NSSV
   50             VSSV(I)=VSSV(I)*DT
              ENDIF
          ELSE
              DT2=DT*DT
              A3=2.0/GAMA/DT2
              A4=A3*DT
              A5=1.0/GAMA-1.0
C
C ***It is assumed that the initial conditions are all zero ***
C
              DO 70 I=1,NEQ
              GLV(I)=0.0
70            GLA(I)=0.0
          ENDIF
      ENDIF
C
C ***Initialize the arrays***
C
      . . . . . . . . . . .
      DO 250 N=1,NEM
      DO 200 I=1,NPE
      NI=NOD(N,I)
      IF(NONLIN.GT.0 .OR. ITEM.GT.0)THEN
         ELU(I)=GLU(NI)         !Transfer of the current solution
         IF(ITEM.GT.0)THEN
             ELU0(I)=GLP(NI)  !Transfer of the previous time step solution
         ENDIF
      ENDIF
      IF(ITEM.EQ.2) THEN
          ELU1(I)=GLV(NI)      !Transfer of the previous first time derivative
          ELU2(I)=GLA(NI)      !Transfer of the previous second time derivative
      ENDIF
      ELXY(I,1)=GLXY(NI,1)
      ELXY(I,2)=GLXY(NI,2)
200 CONTINUE
C
C     Call Subroutine ELMATRCS2D to compute ELK, ELK-HAT, etc.
C     and assemble them into global matrices GLK
C
       . . . . . . . . . .

250 CONTINUE
```

```
      SUBROUTINE ELMATRCS2D(MODEL,NPE,NN,NONLIN)
C     ________________________________________________________________
C
C     Element  calculations based on  linear and quadratic rectangular
C     elements with isoparametric  formulation.
C     ________________________________________________________________
C
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/STF/ELF(9),ELK(9,9),ELXY(9,2),ELU(9)
      COMMON/PST/A10,A1X,A1Y,A20,A2X,A2Y,A00,F0,FX,FY,
     *           A1U,A1UX,A1UY,A2U,A2UX,A2UY
      COMMON/SHP/SFL(9),GDSFL(2,9)
      COMMON/PNT/IPDF,IPDR,NIPF,NIPR
      DIMENSION  GAUSPT(5,5),GAUSWT(5,5),TANG(9,9)
      COMMON/IO/IN,IT
C
      DATA GAUSPT/5*0.0D0, –0.57735027D0, 0.57735027D0, 3*0.0D0,
     2  –0.77459667D0, 0.0D0, 0.77459667D0, 2*0.0D0, –0.86113631D0,
     3  –0.33998104D0, 0.33998104D0, 0.86113631D0, 0.0D0, –0.90617984D0,
     4  –0.53846931D0,0.0D0,0.53846931D0,0.90617984D0/
C
      DATA GAUSWT/2.0D0, 4*0.0D0, 2*1.0D0, 3*0.0D0, 0.55555555D0,
     2   0.88888888D0, 0.55555555D0, 2*0.0D0, 0.34785485D0,
     3 2*0.65214515D0, 0.34785485D0, 0.0D0, 0.23692688D0,
     4   0.47862867D0, 0.56888888D0, 0.47862867D0, 0.23692688D0/
C
      NDF=NN/NPE
C
C   Initialize the arrays
C
      DO 100 I = 1,NPE
        ELF(I)  = 0.0
      DO 100 J = 1,NPE
        IF(NONLIN.GT.1)THEN
            TANG(I,J)=0.0
        ENDIF
100   ELK(I,J)= 0.0
C
C   Do-loops on numerical (Gauss) integration begin here.
C
      DO 200 NI = 1,IPDF
      DO 200 NJ = 1,IPDF
        XI  = GAUSPT(NI,IPDF)
        ETA = GAUSPT(NJ,IPDF)
        CALL INTERPLN2D(NPE,XI,ETA,DET,ELXY)
        CNST = DET*GAUSWT(NI,IPDF)*GAUSWT(NJ,IPDF)
        X=0.0
        Y=0.0
        DO 120 I=1,NPE
          X=X+ELXY(I,1)*SFL(I)
120       Y=Y+ELXY(I,2)*SFL(I)
```

```
      IF(MODEL.EQ.1)THEN
C
C   Define the coefficients of the differential equation
C
        AXX=A10+A1X*X+A1Y*Y
        AYY=A20+A2X*X+A2Y*Y
        FXY=F0+FX*X+FY*Y
C
        IF(NONLIN.GT.0)THEN
            U=0.0
            DO 140 I =1,NPE
            U=U+ELU(I)*SFL(I)
140         CONTINUE
            AXX=AXX+A1U*U+A1UX*UX+A1UY*UY
            AYY=AYY+A2U*U+A2UX*UX+A2UY*UY
        ENDIF
C
        DO 180 I=1,NPE
        ELF(I)=ELF(I)+FXY*SFL(I)*CNST
        DO 160 J=1,NPE
             S00=SFL(I)*SFL(J)*CNST
             S11=GDSFL(1,I)*GDSFL(1,J)*CNST
             S22=GDSFL(2,I)*GDSFL(2,J)*CNST
             S12=GDSFL(1,I)*GDSFL(2,J)*CNST
             S21=GDSFL(2,I)*GDSFL(1,J)*CNST
             ELK(I,J) = ELK(I,J) + AXX*S11 + AYY*S22 + A00*S00
C
C   Write statements here for the part needed to be added to K to obtain T
C                          * * * * * * * * * * * * * * *
160  CONTINUE
180  CONTINUE
       ENDIF
200  CONTINUE
C
C   Write statements here to compute the residual vector and tangent matrix
C                          * * * * * * * * * * * * * * *
      RETURN
      END
```

```
    DO 200 NI = 1,NGPF            ! full Gauss integration loop
    DO 200 NJ = 1,NGPF
       . . . . . . . . .
C
C    Define the linear and nonlinear coefficients of the differential equation
C
       . . . . . . . . .
      IF(ITEM.GT.0)THEN
        CXY=C0+CX*X+CY*Y    ! Define the coefficient of the time derivative
      ENDIF
C
       . . . . . . . . .
      IF(ITEM.GT.0)THEN       ! Define the solution vector u_s and its derivatives
          UP =0.0
          UPX=0.0
          UPY=0.0
          DO 140 I=1,NPE
          UP =UP +ELU0(I)*SF(I)
          UPX=UPX+ELU0(I)*GDSF(1,I)
140       UPY=UPY+ELU0(I)*GDSF(2,I)
          APXX=A11+A1U*UP+A1UX*UPX+A1UY*UPY
          APYY=A22+A2U*UP+A2UX*UPX+A2UY*UPY
      ENDIF
C
C     Define the element coefficient matrices ELK, ELF, and ELM
C
          . . . . . . . . .
      IF(ITEM.GT.0)THEN
          ELM(I,J)=ELM(I,J)+CXY*S00
          IF(NONLIN.GT.0)THEN        ! Define K_s   (a_xy = a_yx = 0)
              ELK0(I,J)=ELK0(I,J)+APXX*SXX+APYY*SYY+A00*S00
          ENDIF
       ENDIF
C     . . . . . . . . .
C
200 CONTINUE
C
C    Compute K̂_{s+1} and F̂
C
      IF(ITEM.EQ.1) THEN  ! for parabolic equations
         DO 220 I=1,NN
         SUM=0.0
         DO 210 J=1,NN
         IF(NONLIN.GT.0)THEN
         SUM = SUM+(ELM(I,J)-A2*ELK0(I,J))*ELU0(J)
         ELSE
         SUM = SUM+(ELM(I,J)-A2*ELK(I,J))*ELU0(J)
         ENDIF
210      ELK(I,J) = ELM(I,J)+A1*ELK(I,J)
220      ELF(I) = (A1+A2)*ELF(I)+SUM
      ENDIF
      IF(ITEM.GT.1) THEN  ! for hyperbolic equations
         DO 270 I = 1,NN
         SUM  = 0.0
         DO 260 J = 1,NN
         SUM  = SUM+ELM(I,J)*(A3*ELU0(J)+A4*ELU1(J)+A5*ELU2(J))
260      ELK(I,J)= ELK(I,J)+A3*ELM(I,J)
270      ELF(I)  = ELF(I)+SUM
      ENDIF
```

NUMERICAL EXAMPLES

Heat transfer in a rod

$$\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} = 0, \qquad 0 < x < 1$$

$$u(0,t) = 0, \quad \frac{\partial u}{\partial x}(1,t) = 0$$

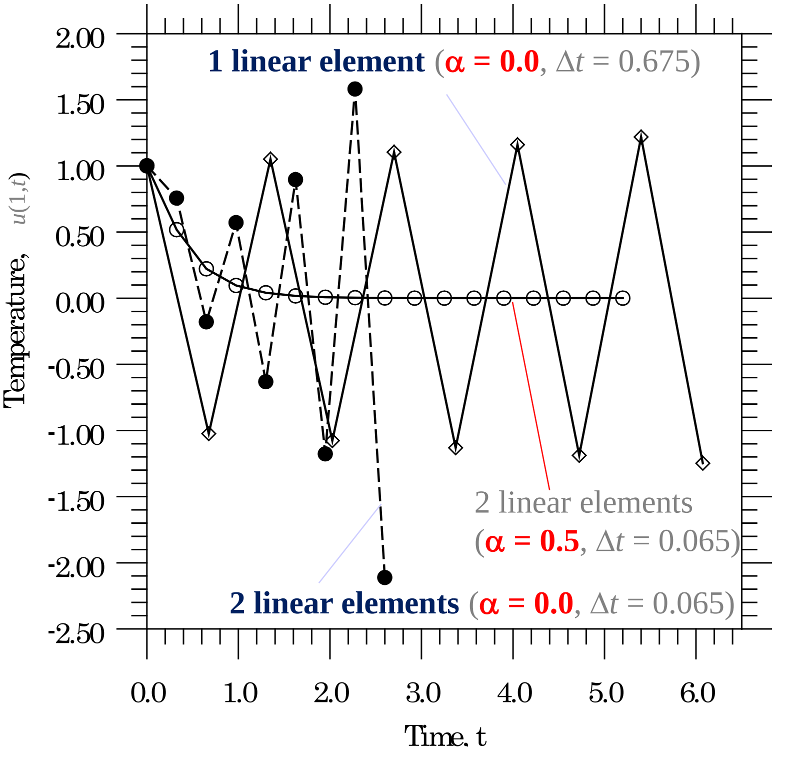
$$u(x,0) = 1$$

one-element model:

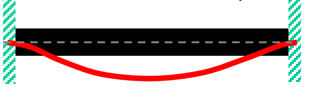
$$\Delta t_{crit} = 2/3 = 0.66667$$

two-element model:

$$\Delta t_{crit} = 2/31.689 = 0.063$$



NUMERICAL EXAMPLES (continued)



Bending of a clamped beam

$$\frac{\partial^2 w}{\partial t^2}+\frac{\partial^4 w}{\partial x^4}=0, \qquad 0<x<1$$

$$w(0,t)=0, \quad \frac{\partial w}{\partial x}(0,t)=0,$$

$$w(1,t)=0, \quad \frac{\partial w}{\partial x}(1,t)=0$$

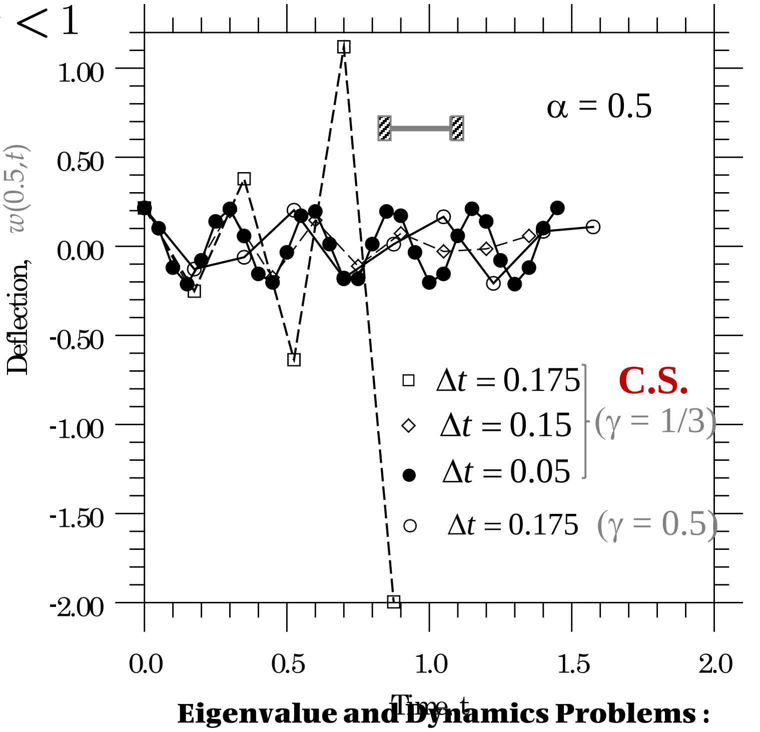
$$w(x,0)=\sin \pi x-\pi x(1-x)$$

one-element model:

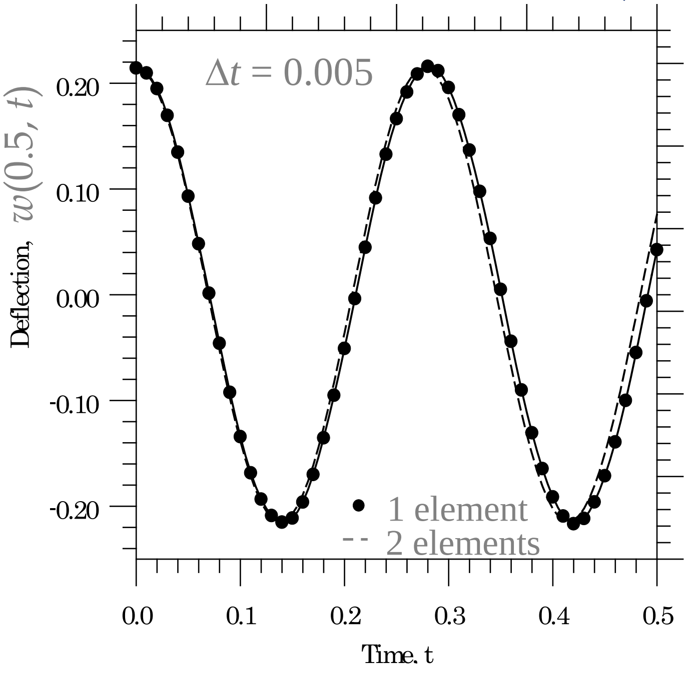
$$\Delta t_{crit}=12/516.93$$
$$=0.023214$$

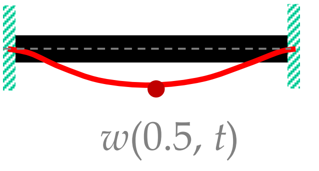
two-element model:

$$\Delta t_{crit}=0.00897$$



NUMERICAL EXAMPLES
(continued)





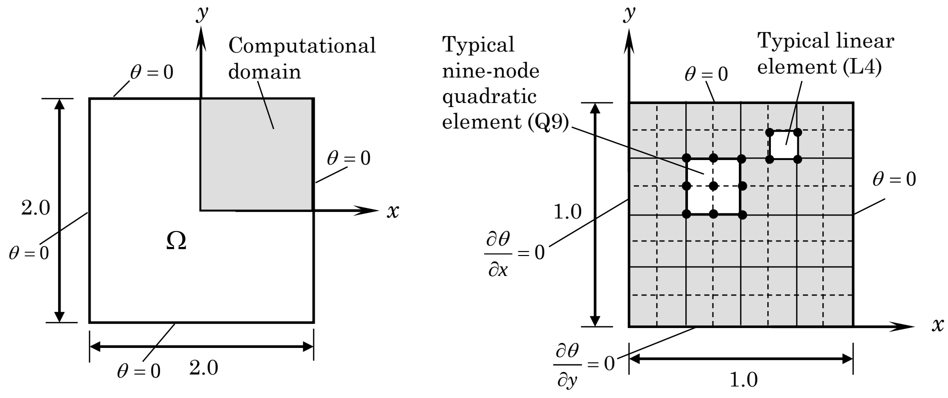


Figure 5.7.3: Actual and computational domains of the transient heat transfer problem.

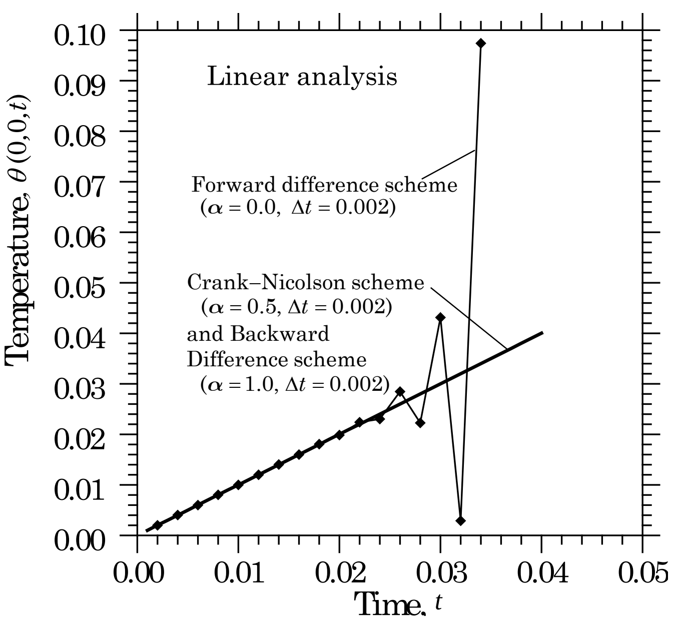


Figure 5.7.4: Transient response predicted by various schemes.