

Report No.
UCB/SEMM-2010/07

Structural Engineering
Mechanics and Materials

**Implementation of a user-defined
time integration in FEAP:
Wilson- Θ**

By Karl Steeger

Mentor: Sanjay Govindjee

July 2010

Department of Civil and Environmental Engineering
University of California, Berkeley

Contents

1	The Wilson-Θ Method	4
1.1	Derivation of the Wilson- Θ Method	4
1.2	Stability of the Wilson- Θ Method	5
2	Abstract - How to program	5
2.1	Given displacements (d-form)	5
2.2	Implemetation steps in FEAP	7
2.2.1	udynam.f	8
2.2.2	dparam.f/uparam.f	12
2.2.3	dsetci.f/usetci.f	14
2.2.4	Inputfile	16
3	Numerical Example	16
3.1	Description of the problem	16
3.2	Evaluation	19

List of Figures

1	The Wilson- Θ Method.	4
2	Running schedule	7
3	Main parts of <code>udynam.f</code> for $isw = 1$	10
4	Main parts of <code>udynam.f</code> for $isw = 2$	11
5	Main parts of <code>udynam.f</code> for $isw = 3$	12
6	Main parts of <code>uparam.f</code>	15
7	Main parts of <code>usetci.f</code>	16
8	Setup Cook's membrane.	17
9	Inputfile for Cook's Membrane.	18
10	Procedure to run inputfile.	19
11	Cook's Membrane u_2 -displacement upper right node.	20

List of Tables

1	Different cases for isw in <code>udynam.f.</code>	8
2	isw independent variables in <code>udynam.f.</code>	8
3	Passed variables for $isw = 1$ in <code>udynam.f.</code>	9
4	Passed variables for $isw = 2$ in <code>udynam.f.</code>	11
5	Passed variables in <code>dparam.f.</code>	12
6	Passed variables in <code>uparam.f.</code>	13
7	Setup of variables in <code>uparam.f.</code>	14
8	Parameters $cc1$, $cc2$ and $cc3$	15
9	Residual satisfaction.	19

1 The Wilson- Θ Method

The FEAP internal structure provides different time integration schemes like Newmark, HHT, etc.. Additionally it is possible to add a user defined one.

A possible scheme for time integration is the Wilson- Θ method. The Wilson- Θ Method is a linear multistep method for second order equations, see Wilson et al. [1973] and Hughes [1987].

1.1 Derivation of the Wilson- Θ Method

The time-dependent system of equations which has to be solved is written as

$$\mathbf{M}\ddot{\mathbf{d}} + \mathbf{C}\dot{\mathbf{d}} + \mathbf{K}\mathbf{d} = \mathbf{F}, \quad (1)$$

where \mathbf{M} denotes the mass matrix, \mathbf{C} the viscous damping matrix and \mathbf{K} the stiffness matrix. The Wilson- Θ method assumes a linear change of acceleration within a time interval $[t, t + \Theta\Delta t]$ with time variable τ , see Fig. 1. Using this assumption the velocity

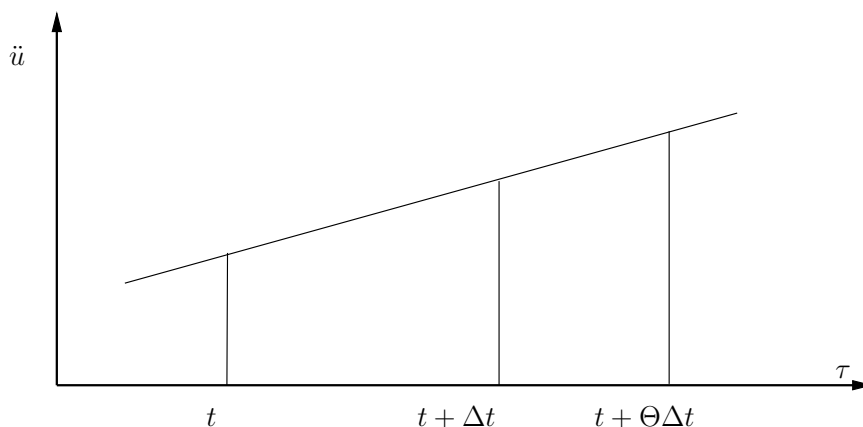


Figure 1: The Wilson- Θ Method.

can be calculated by integration of the acceleration over time and the displacement is obtained by integrating the acceleration twice over time. With the variable Θ the numerical stability and damping can be influenced. A linear change of acceleration over time leads to the following expression for the acceleration at time $t + \tau$

$$\ddot{\mathbf{u}}_{t+\tau} = \ddot{\mathbf{u}}_t + \frac{\tau}{\Theta\Delta t} (\ddot{\mathbf{u}}_{t+\Theta\Delta t} - \ddot{\mathbf{u}}_t). \quad (2)$$

Integration over the control variable τ leads to

$$\dot{\mathbf{u}}_{t+\tau} = \dot{\mathbf{u}}_t + \ddot{\mathbf{u}}_t\tau + \frac{\tau^2}{2\Theta\Delta t} (\ddot{\mathbf{u}}_{t+\Theta\Delta t} - \ddot{\mathbf{u}}_t). \quad (3)$$

The next integration over τ yields the displacements as

$$\mathbf{u}_{t+\tau} = \mathbf{u}_t + \dot{\mathbf{u}}_t\tau + \frac{1}{2}\ddot{\mathbf{u}}_t\tau^2 + \frac{\tau^3}{6\Theta\Delta t} (\ddot{\mathbf{u}}_{t+\Theta\Delta t} - \ddot{\mathbf{u}}_t). \quad (4)$$

Now we replace the control variable τ with the expression $\Theta\Delta t$ in the equations for velocity and displacements and simplify. We obtain

$$\begin{aligned}\dot{\mathbf{u}}_{t+\Theta\Delta t} &= \dot{\mathbf{u}}_t + \frac{\Theta\Delta t}{2} (\ddot{\mathbf{u}}_{t+\Theta\Delta t} + \ddot{\mathbf{u}}_t) \\ \mathbf{u}_{t+\Theta\Delta t} &= \mathbf{u}_t + \Theta\Delta t \dot{\mathbf{u}}_t + \frac{1}{6} (\Theta\Delta t)^2 (\ddot{\mathbf{u}}_{t+\Theta\Delta t} + 2\ddot{\mathbf{u}}_t).\end{aligned}\tag{5}$$

1.2 Stability of the Wilson- Θ Method

The Wilson method is unconditionally stable for $\Theta \geq 1.37$. Further information concerning stability and accuracy can be found in Hughes [1987], Gladwell and Thomas [1980] and Wilson et al. [1973].

2 Abstract - How to program

To program this, we could differentiate between three different cases, depending on which variable is solved by the system of equations and which variables has to be calculated by the time integration scheme. Here we just consider the formulation where the displacement is the solution variable. For our calculation we need the values of the displacement, the velocity and the acceleration at different points in time.

2.1 Given displacements (d-form)

The expressions have to be with respect to the displacements. From (5b) we get, after rearranging, the following equation for the acceleration at time $t + \Theta\Delta t$ depending on the displacements:

$$\ddot{\mathbf{u}}_{t+\Theta\Delta t} = -2\ddot{\mathbf{u}}_t - \frac{6}{\Theta\Delta t} \dot{\mathbf{u}}_t + \frac{6}{(\Theta\Delta t)^2} (\mathbf{u}_{t+\Theta\Delta t} - \mathbf{u}_t).\tag{6}$$

Inserting this in (5a) leads to the velocities in terms of the displacements at time $t + \Theta\Delta t$ as

$$\dot{\mathbf{u}}_{t+\Theta\Delta t} = -2\dot{\mathbf{u}}_t - \frac{\Theta\Delta t}{2} \ddot{\mathbf{u}}_t + \frac{3}{\Theta\Delta t} (\mathbf{u}_{t+\Theta\Delta t} - \mathbf{u}_t).\tag{7}$$

With the abbreviation

$$\Delta\mathbf{u}_{t+\Theta\Delta t} = \mathbf{u}_{t+\Theta\Delta t} - \mathbf{u}_t\tag{8}$$

we get from (7) and (6) the following expressions for velocity and acceleration at time $t + \Theta\Delta t$:

$$\begin{aligned}\dot{\mathbf{u}}_{t+\Theta\Delta t} &= -2\dot{\mathbf{u}}_t - \frac{\Theta\Delta t}{2} \ddot{\mathbf{u}}_t + \frac{3}{\Theta\Delta t} \Delta\mathbf{u}_{t+\Theta\Delta t} \\ \ddot{\mathbf{u}}_{t+\Theta\Delta t} &= -2\ddot{\mathbf{u}}_t - \frac{6}{\Theta\Delta t} \dot{\mathbf{u}}_t + \frac{6}{\Theta^2\Delta t^2} \Delta\mathbf{u}_{t+\Theta\Delta t}.\end{aligned}\tag{9}$$

From these we can extract the predictors as

$$\begin{aligned}\dot{\tilde{\mathbf{u}}}_{t+\Theta\Delta t} &= -2\dot{\mathbf{u}}_t - \frac{\Theta\Delta t}{2}\ddot{\mathbf{u}}_t \\ \ddot{\tilde{\mathbf{u}}}_{t+\Theta\Delta t} &= -2\ddot{\mathbf{u}}_t - \frac{6}{\Theta\Delta t}\dot{\mathbf{u}}_t.\end{aligned}\quad (10)$$

Which leads to the following equations

$$\begin{aligned}\dot{\mathbf{u}}_{t+\Theta\Delta t} &= \dot{\tilde{\mathbf{u}}}_{t+\Theta\Delta t} + \frac{3}{\Theta\Delta t}\Delta\mathbf{u}_{t+\Theta\Delta t} \\ \ddot{\mathbf{u}}_{t+\Theta\Delta t} &= \ddot{\tilde{\mathbf{u}}}_{t+\Theta\Delta t} + \frac{6}{(\Theta\Delta t)^2}\Delta\mathbf{u}_{t+\Theta\Delta t}.\end{aligned}\quad (11)$$

In the Wilson- Θ Method equilibrium is enforced at time $t + \Theta\Delta t$; thus

$$\mathbf{M}\ddot{\mathbf{u}}_{t+\Theta\Delta t} + \mathbf{C}\dot{\mathbf{u}}_{t+\Theta\Delta t} + \mathbf{K}\mathbf{u}_{t+\Theta\Delta t} = \mathbf{F}_{t+\Theta\Delta t}.\quad (12)$$

For the d-form we get the system of equations

$$\underbrace{\left(\frac{6}{(\Theta\Delta t)^2}\mathbf{M} + \frac{3}{\Theta\Delta t}\mathbf{C} + \mathbf{K}\right)}_{\mathbf{K}_{eff}}\Delta\mathbf{u}_{t+\Theta\Delta t} = \mathbf{F}_{t+\Theta\Delta t} - \mathbf{M}\ddot{\tilde{\mathbf{u}}}_{t+\Theta\Delta t} - \mathbf{C}\dot{\tilde{\mathbf{u}}}_{t+\Theta\Delta t} - \mathbf{K}\mathbf{u}_t.\quad (13)$$

As can be seen in (13) and the definitions (10) the displacement, velocity and acceleration at time $t + \Delta t$ of step $k - 1$ (the end of the prior time step) are needed to calculate the variables at time $t + \Theta\Delta t$ in our current timestep k . Taking the expression

$$\ddot{\mathbf{u}}_{t+\Theta\Delta t} = (1 - \Theta)\ddot{\mathbf{u}}_t + \Theta\ddot{\mathbf{u}}_{t+\Delta t},\quad (14)$$

we can rearrange it such that

$$\ddot{\mathbf{u}}_{t+\Delta t} = \frac{1}{\Theta}\ddot{\mathbf{u}}_{t+\Theta\Delta t} - \left(\frac{1}{\Theta} - 1\right)\ddot{\mathbf{u}}_t.\quad (15)$$

Now we insert (9b) and get finally the following equation for the acceleration at time $t + \Delta t$:

$$\ddot{\mathbf{u}}_{t+\Delta t} = \left(1 - \frac{3}{\Theta}\right)\ddot{\mathbf{u}}_t - \frac{6}{\Theta^2\Delta t}\dot{\mathbf{u}}_t + \frac{6}{\Theta^3\Delta t^2}\Delta\mathbf{u}_{t+\Theta\Delta t}.\quad (16)$$

The velocity at time $t + \Delta t$ can be calculated using the knowledge that the acceleration is linear in time, which leads to

$$\dot{\mathbf{u}}_{t+\Delta t} = \dot{\mathbf{u}}_t + \frac{\Delta t}{2}(\ddot{\mathbf{u}}_{t+\Delta t} + \ddot{\mathbf{u}}_t).\quad (17)$$

Inserting (16) yields

$$\dot{\mathbf{u}}_{t+\Delta t} = \left(1 - \frac{3}{\Theta^2}\right)\dot{\mathbf{u}}_t + \left(\Delta t - \frac{3\Delta t}{2\Theta}\right)\ddot{\mathbf{u}}_t + \frac{3}{\Theta^3\Delta t}\Delta\mathbf{u}_{t+\Theta\Delta t}.\quad (18)$$

We also need an expression for the displacement at time $t + \Delta t$ depending just on values of time t and the increment of the solution $\Delta\mathbf{u}_{n+\Theta\Delta t}$. Therefore we take (5b) and set $\Theta = 1$.

$$\mathbf{u}_{t+\Delta t} = \mathbf{u}_t + \Delta t\dot{\mathbf{u}}_t + \frac{\Delta t^2}{6}(\ddot{\mathbf{u}}_{t+\Delta t} + 2\ddot{\mathbf{u}}_t).\quad (19)$$

Now we insert (16) and finally get

$$\mathbf{u}_{t+\Delta t} = \mathbf{u}_t + \left(1 - \frac{1}{\Theta^2}\right)\Delta t \dot{\mathbf{u}}_t + \left(1 - \frac{1}{\Theta}\right)\frac{\Delta t^2}{2}\ddot{\mathbf{u}}_t + \frac{1}{\Theta^3}\Delta \mathbf{u}_{t+\Theta\Delta t}. \quad (20)$$

From that we can extract a predictor

$$\tilde{\mathbf{u}}_{t+\Delta t} = \mathbf{u}_t + \left(1 - \frac{1}{\Theta^2}\right)\Delta t \dot{\mathbf{u}}_t + \left(1 - \frac{1}{\Theta}\right)\frac{\Delta t^2}{2}\ddot{\mathbf{u}}_t \quad (21)$$

and get the final expression

$$\mathbf{u}_{t+\Delta t} = \tilde{\mathbf{u}}_{t+\Delta t} + \frac{1}{\Theta^3}\Delta \mathbf{u}_{t+\Theta\Delta t} \quad (22)$$

respectively

$$\Delta \mathbf{u}_{t+\Delta t} = \mathbf{u}_{t+\Delta t} - \mathbf{u}_t = -\mathbf{u}_t + \tilde{\mathbf{u}}_{t+\Delta t} + \frac{1}{\Theta^3}\Delta \mathbf{u}_{t+\Theta\Delta t}. \quad (23)$$

2.2 Implementation steps in FEAP

In FEAP different subroutines have to be considered in implementing a user-defined time integration method. In Fig. 2 the subroutines are shown relative to their position in the running schedule of the program.

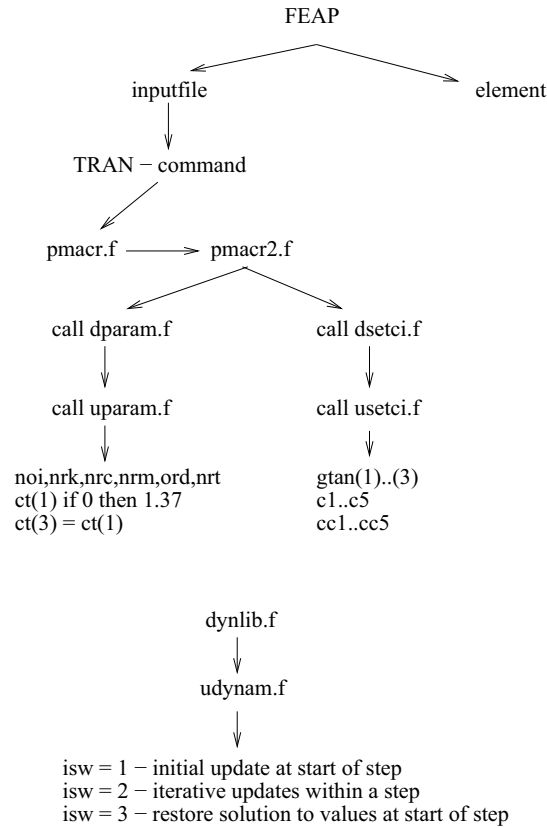


Figure 2: Running schedule

2.2.1 uodynam.f

The subroutine uodynam.f is used for the implementation of the governing equations of the Wilson Θ method. The aim of this subroutine is to calculate updates to the solution vectors. In this subroutine we differentiate between three *isw* cases, see Table 1. Some of the used variables are independent of the *isw* cases. These are shown in Table 2.

Variable	Description
isw	= 1 \Rightarrow initial updates at start of step
	= 2 \Rightarrow iterative updates within a step
	= 3 \Rightarrow restore solution to values at start of step

Table 1: Different cases for *isw* in uodynam.f.

Variable	Description
nneq	Number nodal parameters(ndf*numnp)
ndf	Number degree of freedoms/node
ndfp(*)	Partition number for degree of freedoms
ndfo(*)	Order for number for degree of freedoms
npart	Active partition number

Table 2: *isw* independent variables in uodynam.f.

The increment to the solution is given by the vector $du^{(*)}$ for *isw* = 2; for *isw* = 1 it is the solution at $t + \Delta t$ for the prior time step; i.e. u_t for the current time step. Important to note is that for *isw* = 2, the vector $du^{(*)}$ contains the solution increment at time $t + \Theta\Delta t$ of equation (13). The remaining time-dependent variables which are the solution of the Wilson Θ algorithm are stored in the *ud*-field, which contains the user controlled vectors. It should be mentioned that the *ud*-field is called *urate*-field in all other subroutines (e.g. *dynlib.f*). The Wilson Θ formulas must be implemented for three different *isw* values. For *isw* = 1 the initial updates at the start of a step need to be calculated. This means the predictor values which depend solely on the displacement, velocity or acceleration of the last time step $t_{t+\Delta t}^{k-1} = t_t^k$ are calculated in this section. After that, within *isw* = 2, the time dependent values are updated using the actual increment to the solution $du^{(*)}$. *isw* = 3 gives you the opportunity to restore the calculated values to the values at the start of the time step. The variables passed for *isw* = 1 are described in Table 3 and for *isw* = 2 in Table 4. It should be observed that $t_{t+\Delta t}^{k-1} = t_t^k$, where k denotes the actual and

$k - 1$ the previous calculation step. For a better understanding parts of the subroutine `udynam.f` are shown in Fig.3 for $isw = 1$, Fig.4 for $isw = 2$ and Fig.5 for $isw = 3$

Variable	Description
Inputs	
<code>du(*)</code>	previous solution at time $t_{t+\Delta t}^{k-1} = t_t^k$
<code>ud(nneq,*)</code> <code>u(nneq,*)</code>	User controlled vectors Predictors for solution states at time $t_{t+\Delta t}$ update in <code>updatl.f</code> with <code>cc1</code> , <code>cc2</code> parameters, see Table 8
<code>isw=1</code>	
<code>ud(nneq,1)</code>	predictor for displacement at $t_{t+\Theta\Delta t}^k$
<code>ud(nneq,2)</code>	predictor for velocity at $t_{t+\Theta\Delta t}^k$
<code>ud(nneq,3)</code>	predictor for acceleration at $t_{t+\Theta\Delta t}^k$
<code>ud(nneq,4)</code>	input: velocity at $t_{t+\Delta t}^{k-1}$ output: predictor of velocity at $t_{t+\Delta t}^k$
<code>ud(nneq,5)</code>	input: acceleration at $t_{t+\Delta t}^{k-1}$ output: predictor of acceleration at $t_{t+\Delta t}^k$
<code>ud(nneq,6)</code>	backup of velocity at $t_{t+\Delta t}^{k-1}$
<code>ud(nneq,7)</code>	backup of acceleration at $t_{t+\Delta t}^{k-1}$
<code>u(nneq,1)</code>	input: displacement at $t_{t+\Delta t}^{k-1}$ output: predictor of displacement at $t_{t+\Delta t}^k$
<code>u(nneq,2)</code>	predictor for the increment of displacement at $t_{t+\Delta t}^k$

Table 3: Passed variables for $isw = 1$ in `udynam.f`.

```

c+++++72
c   (1) Initialize step (calculate predictors)
c     if(isw.eq.1) then
c+++++72
c   Write values in a temporary array (for backup)
c     do i = 1,ndf
c       if(ndfp(i).eq.npart .and. ndfo(i).ge.1) then
c         do n = i,nneq,ndf
c           ud(n,6) = ud(n,4)
c           ud(n,7) = ud(n,5)
c         end do
c       endif
c     end do ! i
c+++++72
c   Calculate predictors
c     do i = 1,ndf
c       if(ndfp(i).eq.npart) then
c         do n = i,nneq,ndf
c           Displacement_t+theta*dt
c             ur1 = u(n,1)
c           Velocity_t+theta*dt
c             ur2 = - 2.d0*ud(n,4)
c             &          - (theta(1)*dt)/(2.d0)*ud(n,5)
c           Accleration_t+theta*dt
c             ur3 = - 2.d0*ud(n,5)
c             &          - 6.d0/(theta(1)*dt)*ud(n,4)
c           Velocity_t+dt
c             ur4 = (1.d0 - 3.d0/theta(1)**2)*ud(n,4)
c             &          + (dt - (3.d0*dt)/(2.d0*theta(1)))*ud(n,5)
c           Accleration_t+dt
c             ur5 = (1.d0 - 3.d0/theta(1))*ud(n,5)
c             &          - 6.d0/(theta(1)**2*dt)*ud(n,4)
c           Displacement_t+dt + updat1.f --> + cc1*du
c             u(n,1) = u(n,1)
c             &          + (1.d0 - 1.d0/(theta(1)**2))*dt*ud(n,4)
c             &          + (1.d0 - 1.d0/theta(1))*(dt**2)/2.d0*ud(n,5)
c           Increment of Displacement_t+dt + updat1.f --> + cc2*du
c             u(n,2) = (1.d0 - 1.d0/(theta(1)**2))*dt*ud(n,4)
c             &          + (1.d0 - 1.d0/theta(1))*(dt**2)/2.d0*ud(n,5)
c           Transfer help variables to original arrays
c             ud(n,1) = ur1
c             ud(n,2) = ur2
c             ud(n,3) = ur3
c             ud(n,4) = ur4
c             ud(n,5) = ur5
c         end do
c       endif
c     end do ! i
c+++++72

```

Figure 3: Main parts of `udynam.f` for $isw = 1$.

Variable	Description
Inputs	
du(*)	Increment to solution at time $t_{t+\Theta\Delta t}^k$
ud(nneq,*)	User controlled vectors
isw=2	
ud(nneq,1)	input : predictor of displacement at $t_{t+\Theta\Delta t}^k$ output: displacement at $t_{t+\Theta\Delta t}^k$
ud(nneq,2)	input : predictor of velocity at $t_{t+\Theta\Delta t}^k$ output: velocity at $t_{t+\Theta\Delta t}^k$
ud(nneq,3)	input : predictor of acceleration at $t_{t+\Theta\Delta t}^k$ output: acceleration at $t_{t+\Theta\Delta t}^k$
ud(nneq,4)	input: predictor of velocity at $t_{t+\Delta t}^k$ output: velocity at $t_{t+\Delta t}^k$
ud(nneq,5)	input: predictor of acceleration at $t_{t+\Delta t}^k$ output: acceleration at $t_{t+\Delta t}^k$

Table 4: Passed variables for $isw = 2$ in uodynam.f.

```

c+++++72
c      (2) Update with in time step
c      elseif(isw.eq.2) then
c+++++72
c      d-form
c      do i = 1,ndf
c          if(ndfp(i).eq.npart) then
c              do n = i,nneq,ndf
c          Evaluation at time t+dt, du is at time t+theta*dt
c              ud(n,2) = ud(n,2) + 3.d0/( theta(1)*dt      )*du(n)
c              ud(n,3) = ud(n,3) + 6.d0/((theta(1)*dt)**2 )*du(n)
c              ud(n,4) = ud(n,4) + 3.d0/( theta(1)**3*dt   )*du(n)
c              ud(n,5) = ud(n,5) + 6.d0/( theta(1)**3*dt**2)*du(n)
c          Displacement_t+theta*dt
c              ud(n,1) = ud(n,1) + du(n)
c              end do
c          endif
c      end do ! i
c+++++72

```

Figure 4: Main parts of uodynam.f for $isw = 2$.

```

c+++++72
c   (3) Backup solution vectors to reinitiate a step
c   elseif(isw.eq.3) then
c+++++72
c     do i = 1,ndf
c       if(ndfp(i).eq.npart) then
c         do n = i,nneq,ndf
c           ud(n,4) = ud(n,6)
c           ud(n,5) = ud(n,7)
c         end do ! n
c       endif
c     end do ! i
c+++++72
endif

```

Figure 5: Main parts of uodynam.f for $isw = 3$.

2.2.2 dparam.f/uparam.f

In the subroutine dparam.f the parameters necessary for the time- stepping algorithms are set. The variables passed are just the three algorithmic parameters, stored in the ct-field and the number/name of the chosen time integration method in lct. The output contains again the ct-field which is possibly redefined and a number of common variables which are set in this subroutine. They are described in Table 5.

Variable	Description
Outputs	
ct(3)	(possibly) redefined algorithmic parameters
theta(i)	Set identical to ct(i), $i=1,2,3$.
noi	Number of the specific integrator
nrk	Pointer to displacements at time $t_{t+\Theta\Delta t}$ in urate(nneq,*) Values related to stiffness matrix \mathbf{K}
nrc	Pointer to velocities at time $t_{t+\Theta\Delta t}$ in urate(nneq,*) Values related to damping matrix \mathbf{C}
nrm	Pointer to accelerations at time $t_{t+\Theta\Delta t}$ in urate(nneq,*) Values related to mass matrix \mathbf{M}
nrt	Maximun number of vectors in urate(nnneq,*)

Table 5: Passed variables in dparam.f.

The subroutine `dparam.f` contains different types of time integration methods e.g. Newmark or Euler. For a user-defined integration scheme the subroutine `uparam.f` is called in `dparam.f`. The variables passed here are described in Table 6.

Variable	Description
Inputs	
<code>ct(3)</code>	Command line input parameters
<code>isw</code>	0 \Rightarrow set value of <code>nrt</code> and return 1 \Rightarrow input/set parameters
Outputs	
<code>nrk</code>	Vector storing stiffness vector (pointer) in <code>urate(nneq,*)</code>
<code>nrc</code>	Vector storing damping vector (pointer) in <code>urate(nneq,*)</code>
<code>nrm</code>	Vector storing mass vector (pointer) in <code>urate(nneq,*)</code>
<code>ord</code>	Order of ODE for this integrator
<code>nrt</code>	Total vectors required by algorithm

Table 6: Passed variables in `uparam.f`.

In `uparam.f` we have to set the parameters for Wilson Θ , see Table 7. Here we consider the case, that the system of equations is solved with respect to the displacements, so the acceleration and the velocity are calculated by Wilson Θ . In Addition to these values, we have to ensure that the value for Θ is not 0. Because the Wilson- Θ Method is unconditionally stable only for $\Theta \geq 1.37$, see Wilson et al. [1973], we set the value to 1.37, if it is 0, which leads to unconditional stability of the Wilson Θ method. Moreover we have to set the third input parameter `ct(3) = ct(1)` to ensure equilibrium will be computed at time $t + \Theta\Delta t$. This is because in subroutine `dparam.f` the parameter `ct(3)` is used to set the parameter alpha, which controls the time point where the external forces are to be computed ($t + \Theta\Delta t$ in our case). The source code is partly shown in Fig. 6.

Variable	Value	Description
noi	-1	Number of user defined integrator
nrk	1	Pointer to the displacements in urate (nneq,*) used for equilibrium computation of element level $[t_{t+\Theta\Delta t}]$
nrc	2	Pointer to the velocities in urate (nneq,*) used for equilibrium computation of element level $[t_{t+\Theta\Delta t}]$
nrm	3	Pointer to the accelerations in urate (nneq,*) used for equilibrium computation of element level $[t_{t+\Theta\Delta t}]$
ord	2	Wilson Θ is of 2nd order
nrt	5	5 vectors for the updates and two for the backup are needed

Table 7: Setup of variables in uparam.f.

2.2.3 dsetci.f/usetci.f

The subroutines dsetci.f, usetci.f respectively are used to set the integration constants for the dynamic analysis as well as some update parameters. The integration constants are used to compute a modified tangent matrix \mathbf{K}^* as

$$\mathbf{K}^* = c_1\mathbf{K} + c_2\mathbf{C} + c_3\mathbf{M} \quad (24)$$

with the stiffness matrix \mathbf{K} , the damping matrix \mathbf{C} , the mass matrix \mathbf{M} and the integration constants c_1, c_2 and c_3 . In the subroutine usetci.f the integration constants are denoted by the field gtan(). Additionally there are some integration parameters which could be used for convenience while implementing the equations of the time integration method in uodynam.f. These parameters are denoted by c1, c2, ..., c5. We do not use these parameters for the implementation to keep uodynam.f easier to understand. Other parameters which have to be set are the parameter cc1, cc2 and cc3. They are needed for different update schemes in updatl.f. In case of Wilson- Θ the parameters cc1 and cc2 have to be considered. They are needed to update the displacements and their increment in $u(*,1)$ and $u(*,2)$ at time $t_{t+\Delta t}$, respectively. So they have to be set to the factor $\frac{1}{\Theta^3}$ used in (22) and (23). The parameter cc3 is also set to 1, which it is by default, see subroutine pmacro.f, because it is not needed for the Wilson- Θ Method. The parameters cc1, cc2 and cc3 are summarized in Table 8. The main parts of the source code of this subroutine are shown in Fig. 7.

```

elseif(isw.eq.1) then
c      noi      := Number of the specific integrator
      noi = -1

c      nrk      - Vector storing displacement vector for equilibrium computation
      nrk = 1

c      nrc      - Vector storing velocity vector for equilibrium computation
      nrc = 2

c      nrm      - Vector storing acceleration vector for equilibrium computation
      nrm = 3

c      ord      - Order of ODE for this integrator
      ord = 2

c      nrt      - Total vectors required by algorithm
      nrt = 5

c      Ensure that theta is not 0
      if(ct(1).eq.0.0d0) ct(1) = 1.37
c      Define parameter ct(3) for factor alpha in dparam.f
      ct(3) = ct(1)
      ct(2) = 0.d0
c+++++72
      endif
c+++++72

```

Figure 6: Main parts of uparam.f.

Variable	Description
cc1	Parameter to update the solution at $t_{t+\Delta t}$
cc2	Parameter to update the increment at $t_{t+\Delta t}$
cc3	Parameter to update specified boundary displacement not needed for the Wilson- Θ Method

Table 8: Parameters cc1, cc2 and cc3.

```

c+++++72
c   Set default multiplication factors for residual and tangents
c   do i = 1,3
c     gtan(i) = 1.d0
c   end do ! i
c   Set integration parameters
c     c1      = 0.d0
c     c2      = 0.d0
c     c3      = 0.d0
c     c4      = 0.d0
c     c5      = 0.d0
c     cc1     = 1.0d0/(theta(1)**3)
c     cc2     = 1.0d0/(theta(1)**3)
c     cc3     = 1.0d0
c   Set update parameters for element
c     gtan(1) = 1.d0
c     gtan(2) = 3.d0/ (theta(1)*dt)
c     gtan(3) = 6.d0/((theta(1)*dt)**2)
c+++++72

```

Figure 7: Main parts of usetci.f.

2.2.4 Inputfile

To use a time integration method, the transient analysis has to be called by the `TRANSient` command. To call the user-defined Wilson- Θ method the command

$$TRANSient, USER, \Theta \quad (25)$$

has to be used. For further information concerning the transient command see the user manual of FEAP.

3 Numerical Example

In the following we will show a numerical example for the application of the user-defined Wilson- Θ time integration method.

3.1 Description of the problem

As a numerical example we consider a two-dimensional Cook's membrane with the coordinates of the corner nodes $(0, 0)$, $(48, 44)$, $(48, 60)$, $(0, 60)$. The unit of length is considered as $\{m\}$. The left hand side is clamped. The upper and lower edge are regarded as traction free. On the right edge a σ_{12} stress is applied as a boundary condition with a value of $\sigma \mathbf{n} = (0, 4)$. The unit for force is chosen as $\{10^4 N = \frac{10^4 kg m}{s^2}\}$. The system is loaded over a period of 2 units $\{s\}$. After that the load is held for $3\{s\}$. We consider in total a time interval of $5\{s\}$. We consider the system as plane strain. That means that no strains ε can arise in the third coordinate direction, but stresses σ may. As material parameters

the Young's modulus, the Poisson's ratio and the density are chosen as $E = 1000$ unit $\{\frac{10^4 N}{m^2}\}$, $\nu = 0.499$ and $\rho = 0.12$ unit $\{10^4 \frac{kg}{m^3}\}$. The geometry and the boundary conditions are shown in Fig. 8. The parameter for the Wilson- Θ method is set to $\Theta = 1.4$. The

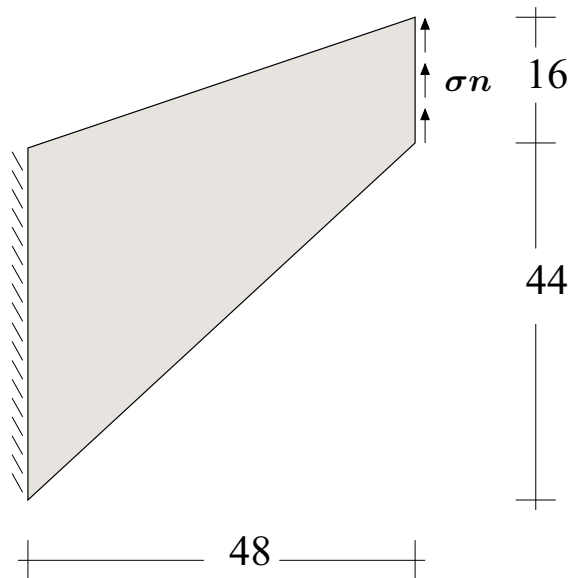


Figure 8: Setup Cook's membrane.

geometry is meshed with 800 six-noded triangular elements which use quadratic shape function for the interpolation. The inputfile is shown in Fig. 9. To run the inputfile the procedure in Fig. 10 has been used.

```

feap * 2d Cook's Membrane Problem
0 0 0 2 2 6
! numnp numel nummat ndm ndf nen
PARAMeter
n = 40
r = n
ee = 1000
nu = 0.499
ro = 0.12
ff = 4e0
be = 1/4
ga = 1/2
! Mesh by edge points of Cook
bloc
CARTesian n,r,,,,,7
1 0 0
2 48 44
3 48 60
4 0 44
! Left hand side clamped
ebou
1 0 1 1
!Load: Surface traction on right hand side
csur
tangential
linear
1 48 44 ff
2 48 60 ff
mate
  solid
  elastic isotropic ee nu
  density mate ro
end
batch
  plot,wipe
  plot,defo,,1
  plot,range,-1,3
end
batch
tran,user,1.4
prop
end
2
0.0 0.0
2.0 1.0
5.0 1.0
BATCh
TPLot
END
disp,1681,2
show
end
inter
stop

```

Figure 9: Inputfile for Cook's Membrane.

```

opti
nopr
dt,,0.0125
loop,,400
  time
  loop,,10
    tang,line,1
  next
plot,wipe
plot,cont,2
next

```

Figure 10: Procedure to run inputfile.

3.2 Evaluation

To check the residual satisfaction of the chosen setup we consider the convergence of our calculation at three different points in time, see Table 9. We can see, that the residual difference is adequately high.

Time	Residual norm 1	Residual norm 2	Proportional load
1.2500×10^{-2}	9.3099708×10^{-2}	$7.1803972 \times 10^{-14}$	6.2500×10^{-3}
2.5000×10^0	$6.1699795 \times 10^{+2}$	3.3720974×10^{-9}	1.0000×10^0
5.0000×10^0	$5.9973547 \times 10^{+2}$	1.7322020×10^{-9}	1.0000×10^0

Table 9: Residual satisfaction.

As a result we consider the u_2 -displacement of the upper right node with the coordinates (48,60). This we compare with the solution of the same boundary value problem using the Newmark method for the dynamic analysis. The graph is shown in Fig. 11. It can be considered, that the solutions are identical.

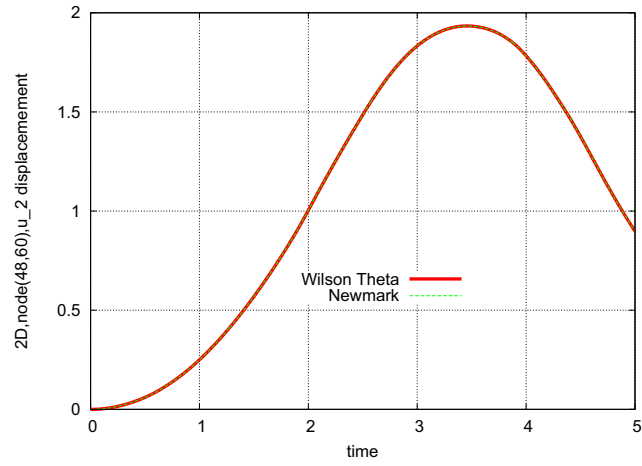


Figure 11: Cook's Membrane u_2 -displacement upper right node.

References

- I. Gladwell and R. Thomas. Stability properties of the newmark, houbolt and wilson θ methods. *Int. J. for num. and anal. meth. in geomech.*, Vol. 4:143–158, 1980.
- T.J.R. Hughes. *The Finite Element Method*. Prentice-Hall Inc, New Jersey, 1987.
- E.L. Wilson, I. Farhoomand, and K.J. Bathe. Nonlinear dynamic analysis of complex structures. *Earthquake Engineering and Structural Dynamics.*, Vol. 1:241–252, 1973.