# Nonliner solution Methods (Start with Time dependent first)

## Chapter 6: Solution Methods and Stability

from: Nonlinear Finite Elements for Continua and Structures, Ted Belytschko, Wing Kam Liu, Brian Moran and Khalil I. Elkhodary, 2nd Edition, Wiley 2014

Parts of Chapter covered
- Explicit time integration
- Implicit time integration
- Methods as applied to "non-time dependent" cases
- Linearization of non-linear systems
- Stability of discrete systems
- "Material" stability

§6.2 Explicit Methods - Look at basic centered difference methods

Time domain $0 \le t \le t_E$ ← Know state at $t=0$

Move forward in time taking time steps $\Delta t$

There will be $n_{TS}$ time steps $1 \le n \le n_{TS}$

How the methods are applied is a function of the reference frame used:

Lagrangian - mesh follows material motion

Eulerian - Material flows through mesh (which is "fixed")

Arbitrary Lagrangian Eulerian - Mesh moves (but not strictly following material)

We look a Lagrangian reference frame equations here

$$\dot{\underset{\sim}{d}} = \underset{\sim}{v} = \frac{\partial \underset{\sim}{d}}{\partial t}, \quad \ddot{\underset{\sim}{d}} = \underset{\sim}{a} = \frac{\partial^2 \underset{\sim}{d}}{\partial t}, \quad \Delta t^n = t^{n+1/2} - t^{n-1/2}$$

defining using the mid steps we have the central difference velocity as:

$$\dot{\underset{\sim}{d}}^{n+1/2} \equiv \underset{\sim}{v}^{n+1/2} = \frac{\underset{\sim}{d}^{n+1} - \underset{\sim}{d}^{n}}{t^{n+1} - t^{n}} = \frac{1}{\Delta t^{n+1/2}} \left( \underset{\sim}{d}^{n+1} - \underset{\sim}{d}^{n} \right)$$

with this $\underset{\sim}{d}^{n+1} = \underset{\sim}{d}^{n} + \Delta t^{n+1/2} \underset{\sim}{v}^{n+1/2}$

The central difference acceleration is

$$\ddot{\underset{\sim}{d}}^{n} \equiv \underset{\sim}{a}^{n} = \frac{\underset{\sim}{v}^{n+1/2} - \underset{\sim}{v}^{n-1/2}}{t^{n+1/2} - t^{n-1/2}}, \quad \underset{\sim}{v}^{n+1/2} = \underset{\sim}{v}^{n-1/2} + \Delta t^{n} \underset{\sim}{a}^{n}$$

using the terms defined above:

$$\ddot{\underset{\sim}{d}}^{n} \equiv \underset{\sim}{a}^{n} = \frac{\Delta t^{n-1/2} \left( \underset{\sim}{d}^{n+1} - \underset{\sim}{d}^{n} \right) - \Delta t^{n+1/2} \left( \underset{\sim}{d}^{n} - \underset{\sim}{d}^{n-1} \right)}{\Delta t^{n-1/2} \Delta t^{n+1/2} \Delta t^{n}}$$

for equal length time steps this is

$$\ddot{\underset{\sim}{d}}^{n} = \underset{\sim}{a}^{n} = \frac{\underset{\sim}{d}^{n+1} - 2\underset{\sim}{d}^{n} + \underset{\sim}{d}^{n-1}}{(\Delta t^n)^2}$$

Lets apply to equation of motion (focuses on inertial term - bunch of stuff hidden in RHS (see notes from F.E. or Hughes Chapt. 8)

$$\underset{\sim}{m}\, \underset{\sim}{a}^{n} = \underset{\sim}{f}^{n} = \underset{\sim}{f}^{ext}(\underset{\sim}{d}^{n}, t^n) - \underset{\sim}{f}^{int}(\underset{\sim}{d}^{n}, t^n)$$

subject to $g_I(\underset{\sim}{d}^{n}) = 0 \quad I = 1(1) n_c$ $n_c$ = # of essential BC.

Using the difference equations we just constructed -

$$\underset{\sim}{V}^{n+1/2} = \underset{\sim}{V}^{n-1/2} + \Delta t^n \underset{\sim}{a}^n = \underset{\sim}{V}^{n-1/2} + \Delta t^n \underset{\sim}{M}^{-1} \underset{\sim}{f}^n$$

$$\underset{\sim}{d}^{n+1} = \underset{\sim}{d}^n + \Delta t^{n+1/2} \underset{\sim}{V}^{n+1/2} = \underset{\sim}{d}^n + \Delta t^{n+1/2} \left( \underset{\sim}{V}^{n-1/2} + \Delta t^n \underset{\sim}{M}^{-1} \underset{\sim}{f}^n \right)$$

Note - Quite simple to step in time when $\underset{\sim}{M}$ is a diagonal matrix.

Explicit methods use a diag. $\underset{\sim}{M}$
(There are methods defined for constructing diagonal $\underset{\sim}{M}$ matrix

Using two step velocity update supports more useful energy balance checks

## Box 6.1 Flowchart for explicit time integration

1. Initial conditions and initialization: set $\mathbf{v}^0$, $\sigma^0$, and initial values of other material state variables; $\mathbf{d}^0=\mathbf{0}$, $n=0$, $t=0$; compute $\mathbf{M}$
2. *getforce*
3. Compute accelerations $\mathbf{a}^n=\mathbf{M}^{-1}(\mathbf{f}^n-\mathbf{C}^{\text{damp}}\mathbf{v}^{n-\frac{1}{2}})$
4. Time update: $t^{n+1}=t^n+\Delta t^{n+\frac{1}{2}}$, $t^{n+\frac{1}{2}}=\frac{1}{2}(t^n+t^{n+1})$
5. First partial update nodal velocities: $\mathbf{v}^{n+\frac{1}{2}}=\mathbf{v}^n+(t^{n+\frac{1}{2}}-t^n)\mathbf{a}^n$
6. Enforce velocity boundary conditions: if node $I$ on $\Gamma_{v_i}$ : $v_{iI}^{n+\frac{1}{2}}=\overline{v}_i\left(\mathbf{x}_I,t^{n+\frac{1}{2}}\right)$
7. Update nodal displacements: $\mathbf{d}^{n+1}=\mathbf{d}^n+\Delta t^{n+\frac{1}{2}}\mathbf{v}^{n+\frac{1}{2}}$
8. *getforce*
9. compute $\mathbf{a}^{n+1}$
10. Second partial update nodal velocities: $\mathbf{v}^{n+1}=\mathbf{v}^{n+\frac{1}{2}}+(t^{n+1}-t^{n+\frac{1}{2}})\mathbf{a}^{n+1}$
11. Check energy balance at time step $n+1$; see (6.2.14–18)
12. Update counter: $n\leftarrow n+1$
13. Output; if simulation not complete, go to 4.

Subroutine *getforce*

0. Initialization: $\mathbf{f}^n=0$, $\Delta t_{\text{crit}}=\infty$
1. Compute global external nodal forces $\mathbf{f}_{\text{ext}}^n$
2. Loop over elements $e$
   - i. GATHER element nodal displacements and velocities
   - ii. $\mathbf{f}_e^{\text{int},n}=\mathbf{0}$
   - iii. Loop over quadrature points $\xi_Q$
     1. if $n=0$, go to 4
     2. compute measures of deformation: $\mathbf{D}^{n-\frac{1}{2}}(\xi_Q),\mathbf{F}^n(\xi_Q),\mathbf{E}^n(\xi_Q)$
     3. compute stress $\sigma^n(\xi_Q)$ by constitutive equation
     4. $\mathbf{f}_e^{\text{int},n}\leftarrow\mathbf{f}_e^{\text{int},n}+\boldsymbol{B}^T\sigma^n\overline{w}_Q J|_{\xi_Q}$

     END quadrature point loop
   - iv. Compute external nodal forces on element, $\mathbf{f}_e^{\text{ext},n}$
   - v. $\mathbf{f}_e^n=\mathbf{f}_e^{\text{ext},n}-\mathbf{f}_e^{\text{int},n}$
   - vi. Compute $\Delta t_{\text{crit}}^e$, if $\Delta t_{\text{crit}}^e<\Delta t_{\text{crit}}$ then $\Delta t_{\text{crit}}=\Delta t_{\text{crit}}^e$
   - vii. SCATTER $\mathbf{f}_e^n$ to global $\mathbf{f}^n$
5. END loop over elements
6. $\Delta t=\alpha\Delta t_{\text{crit}}$

Explicit methods are simple, but they are conditionally stable - solution will grow in an unbounded manner if time step exceeds $\Delta t_{crit}$
"Courant Condition"
you want $\Delta t = \alpha \Delta t_{crit}$ $0 < \alpha < 1$
a value of 0.8 common in solids

Of course you have to find a good value for, or at least bound, for $\Delta t_{crit}$

to give a feel for how small $\Delta t_{crit}$ is for a linearized system with $l_e$ being a measure of the smallest element size

$$\Delta t_{crit} = \min_e \frac{l_e}{C_e} \qquad (A)$$

$C_e$ is the current (linearized) wave speed in element e - Its a large #

Energy balance check - central to detection of "arrested" instabilities that are common to many nonlinear problems that are not detected by linear stability analysis. That is instabilities that arise even when eq (A) is satisfied Many of these missed instabilities can be identified by an energy balance check since they will violate conservation of energy.

## Energy conservation (numerical)

$$|W_{kin}^{n+1} + W_{int}^{n+1} - W_{ext}^{n+1}| \leq \epsilon \max(W_{ext}, W_{int}, W_{kin})$$

want $\epsilon$ to be small but too small not advisable since the calculation of $W_{kin}$, $W_{int}$, $W_{ext}$ are functions of the temporal discretization used.

Note the central difference method's error $\propto \Delta t^2$. This is an appropriate match to use of linear finite elements where error is $\propto h^2$. This may not hold well for all classes of problems. Also strong indicator that with higher order finite elements, higher order difference (or other) time discretization is desired (otherwise the time stepping becomes dominate).

## §6.3 Equilibrium Solutions and Implicit Time Integration.

Discrete momentum equation – residual form

$$S_D \underset{\sim}{M}\underset{\sim}{a}^{n+1} + \underset{\sim}{f}^{int}(\underset{\sim}{d}^{n+1}, t^{n+1}) - \underset{\sim}{f}^{ext}(\underset{\sim}{d}^{n+1}, t^{n+1}) = \underset{\sim}{r}(\underset{\sim}{d}^{n+1}, t^{n+1}) = 0 \quad (9)$$

Note – things like damping and other things are currently hidden in $\underset{\sim}{f}^{int}$, $\underset{\sim}{f}^{ext}$

$S_0 = \begin{cases} 0 & \text{if no inertial terms} \\ 1 & \text{if inertial terms} \end{cases}$

A wide range of implicit methods possible - we consider the two parameter Newmark method

$$\underset{\sim}{d}^{n+1} = \underset{\sim}{\tilde{d}}^{n+1} + \beta \Delta t^2 \underset{\sim}{a}^{n+1} \quad \text{--- (b)}, \qquad \underset{\sim}{\tilde{d}}^{n+1} = \underset{\sim}{d}^{n} + \Delta t \underset{\sim}{v}^{n} + \frac{\Delta t^2}{2}(1-2\beta)\underset{\sim}{a}^{n}$$

$$\underset{\sim}{v}^{n+1} = \underset{\sim}{\tilde{v}}^{n+1} + \gamma \Delta t \, \underset{\sim}{a}^{n+1}, \qquad \underset{\sim}{\tilde{v}}^{n+1} = \underset{\sim}{v}^{n} + (1-\gamma)\Delta t \, a^{n}$$

note -

$\beta = 0, \gamma = 1/2$ - back to explicit central difference

$\beta = 1/4, \gamma = 1/2$ - undamped trapezoidal rule

$\gamma > 1/2$ - numerically dumped, damping $\propto \gamma - 1/2$

Unconditionally stable for $\beta \geq \frac{\gamma}{2} \geq 1/4$

There are conditionally stable conditions - more complex

From (b) above: $\underset{\sim}{a}^{n+1} = \frac{1}{\beta \Delta t^2}\left(\underset{\sim}{d}^{n+1} - \underset{\sim}{\tilde{d}}^{n+1}\right)$ when $\beta > 0$

Substituting into the discrete momentum equation (a)

$$0 = \frac{S_0}{\beta \Delta t^2} \underset{\sim}{M}\left(\underset{\sim}{d}^{n+1} - \underset{\sim}{\tilde{d}}^{n+1}\right) - \underset{\sim}{f}^{ext}\left(\underset{\sim}{d}^{n+1}, t^{n+1}\right) + \underset{\sim}{f}^{int}\left(\underset{\sim}{d}^{n+1}, t^{n+1}\right) = \underset{\sim}{r}\left(\underset{\sim}{d}^{n+1}, t^{n+1}\right)$$

This is (in general) a set of non linear algebraic equations in $\underset{\sim}{d}^{n+1}$. Thus the goal is to find $\underset{\sim}{d}^{n+1}$ such that $\underset{\sim}{r}\left(\underset{\sim}{d}^{n+1}, t^{n+1}\right) = 0$ subject to $\underset{\sim}{g}\left(\underset{\sim}{d}^{n+1}, t^{n+1}\right) = 0$

where $\underset{\sim}{g}\left(\underset{\sim}{d}^{n+1}, t^{n+1}\right)$ captures the essential B.C.

# §6.3.4 Newton's Method

The standard method to address nonlinear problems — Focus on Linearizations of Residual

Consider a single dof case with $\beta > 0$

What we want

$$r(d^{n+1}, t^{n+1}) = \frac{S_D}{\beta \Delta t^2} m(d^{n+1} - \tilde{d}^{n+1}) - f(d^{n+1}, t^{n+1}) = 0$$

Since the problem is nonlinear we will need to iterate until the residual is close enough to zero.

We will use the subscript $v$ for the iteration counter

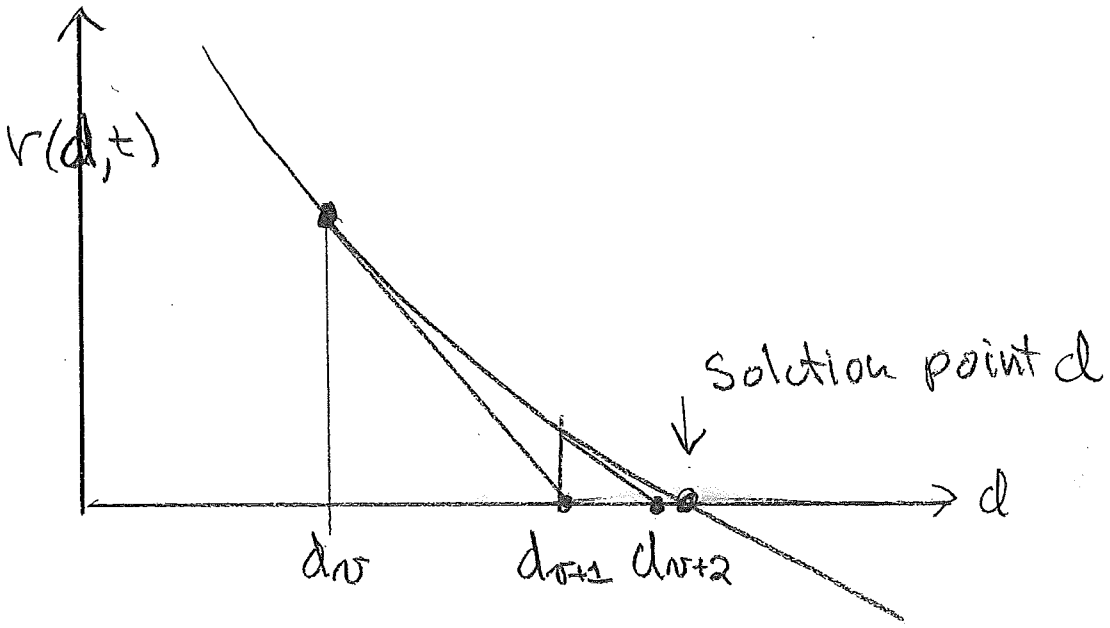
$$d_v^{n+1} = d_v \Leftarrow \text{Concern ourselves with just one time step.}$$

We assume we have a correct starting point to use for the end of the previous time step, $n$. This is $d_0 \equiv d^n$ note when we start $n=0$ and $d_0$ is just the initial condition.

Note — in real cases we will do better using $\underset{\sim}{d}_0 \equiv \underset{\sim}{\tilde{d}}^{n+1} = \underset{\sim}{d}^n + \Delta t \underset{\sim}{v}^n + \frac{\Delta t^2}{2}(1-2\beta)\underset{\sim}{a}^n$

We will set truncated Taylor's series of the residual to zero ($\Delta d = d_{v+1} - d_v$

$$0 = r(d_{v+1}, t^{n+1}) = r(d_v, t^{n+1}) + \frac{\partial r(d_v, t^{n+1})}{\partial d} \Delta d + O(\Delta d^2)$$




Dropping the higher order terms yields a linearized equation for $\Delta d$

$$\Delta d = -\left(\frac{\partial r(d_\nu)}{\partial d}\right)^{-1} r(d_\nu) \;, \; d_{\nu+1} = d_\nu + \Delta d$$

The procedure generalizes to the case of $n_{dof}$ unknowns

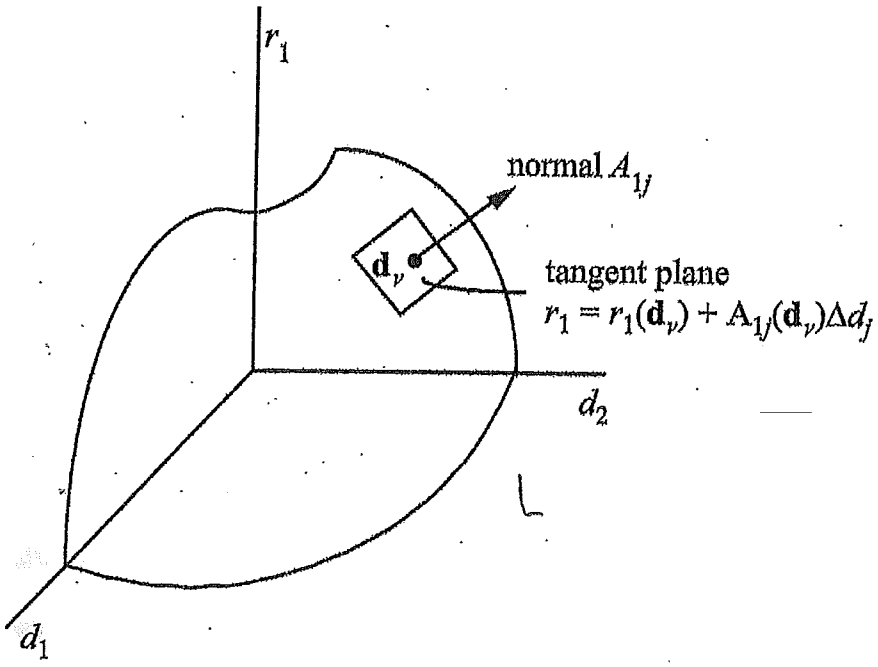
$$0 = \underset{\sim}{r}(\underset{\sim}{d_\nu}) + \frac{\partial \underset{\sim}{r}(d_\nu)}{\partial \underset{\sim}{d}} \Delta \underset{\sim}{d} + O(\Delta \underset{\sim}{d}^2)$$

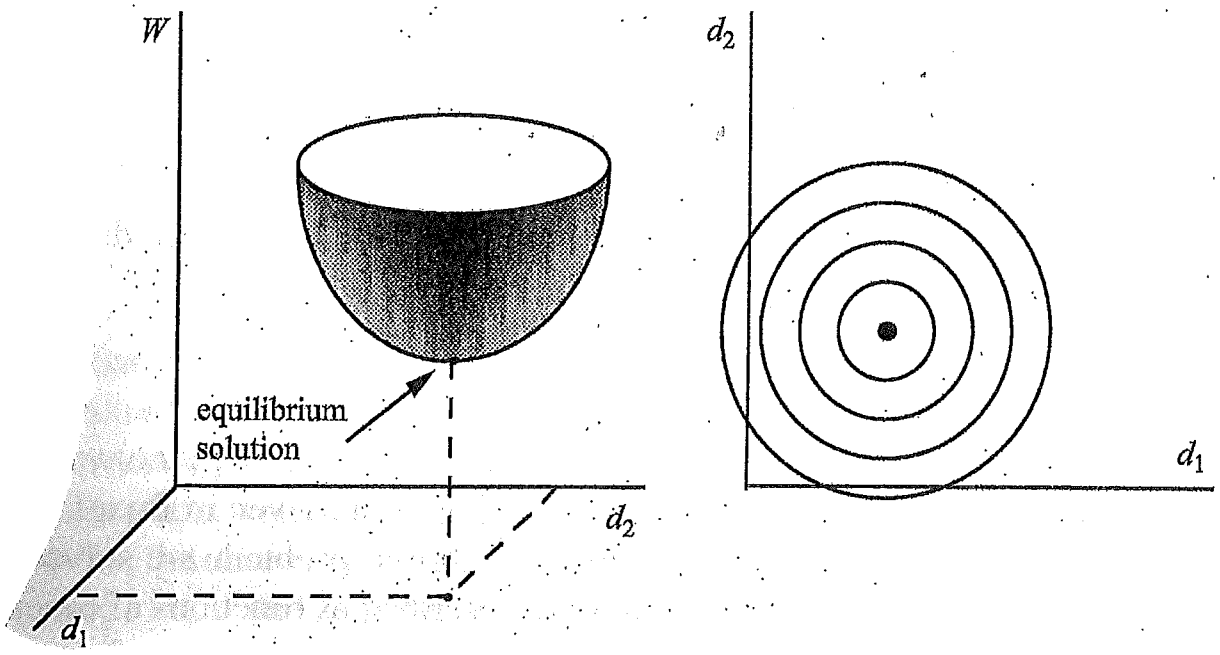
for which each component of $\underset{\sim}{r}$ is

$$0 = r_a(\underset{\sim}{d_\nu}) + \frac{\partial r_a}{\partial d_b} \Delta d_b + O(\Delta \underset{\sim}{d}^2)$$

(sum on $b$)

define the system Jacobian matrix $\underset{\sim}{A}$ where $\underset{\sim}{A} = \frac{\partial \underset{\sim}{r}}{\partial \underset{\sim}{d}}$ where $A_{ab} = \frac{\partial r_a}{\partial d_b}$


$r_1$
normal $A_{1j}$
$\mathbf{d}_\nu$
tangent plane
$r_1 = r_1(\mathbf{d}_\nu) + \mathbf{A}_{1j}(\mathbf{d}_\nu)\Delta d_j$
$d_2$
$d_1$


$W$
equilibrium
solution
$d_2$
$d_1$
$d_2$
$d_1$

Again looking at just the linearized system we have

$$\underset{\sim}{A}\,\Delta\underset{\sim}{d} = -\underset{\sim}{r}\left(\underset{\sim}{d}_v, t^{n+1}\right), \quad \underset{\sim}{d}_{v+1} = \underset{\sim}{d}_v + \Delta\underset{\sim}{d}$$

Keep iterating until $\Delta\underset{\sim}{d}$ is small enough.

There are a substantial number of details that go into any actual implementation

If we assume a Lagrangian reference frame (the mesh will follow the material), the mass matrix $M$ is constant over time, the Newmark integrator Jacobian matrix is

$$\underset{\sim}{A} = \frac{\partial \underset{\sim}{r}}{\partial \underset{\sim}{d}} = \frac{S_D}{\beta \Delta t^2} M + \underbrace{\frac{\partial \underset{\sim}{f}^{int}}{\partial \underset{\sim}{d}}}_{\underset{\sim}{K}^{int}} - \underbrace{\frac{\partial \underset{\sim}{f}^{ext}}{\partial \underset{\sim}{d}}}_{\underset{\sim}{K}^{ext}} \quad (S_D, \beta > 0)$$

$\underset{\sim}{K}^{int}$: tangent stiffness matrix

$\underset{\sim}{K}^{ext}$: Load stiffness matrix

$$\underset{\sim}{A} = \frac{S_D}{\beta \Delta t^2} \underset{\sim}{M} + \underset{\sim}{K}^{int} - \underset{\sim}{K}^{ext}$$

Tangent stiffness

$$\underset{\sim}{K}^{init} = \frac{\partial \underset{\sim}{f}^{init}}{\partial \underset{\sim}{d}}, \quad K^{init}_{ab} = \frac{\partial f^{init}_a}{\partial d_b}, \quad d\underset{\sim}{F}^{init} = \underset{\sim}{K}^{init} d\underset{\sim}{d}$$

Load stiffness

$$\underset{\sim}{K}^{ext} = \frac{\partial \underset{\sim}{F}^{ext}}{\partial \underset{\sim}{d}}, \quad K^{ext}_{ab} = \frac{\partial f^{ext}_a}{\partial d_b}, \quad d\underset{\sim}{F} = \underset{\sim}{K}^{ext} d\underset{\sim}{d}$$

Forming $\underset{\sim}{K}^{init}$ and $\underset{\sim}{K}^{ext}$ is referred to as the linearization process – Its execution requires careful consideration – there are lots of options on approximations made – and when thing are non-linear approximations must be made

Comments on the conservative equilibrium case – that is $S_0=0$ ⇐ no inertial terms

In this case when the loads and constitutive equations are conservative we can look at basic energy principals where we can equate the changes in Potential (energy) for the internal and external forces

$$0=\underset{\sim}{r}=\frac{\partial W}{\partial \underset{\sim}{d}}=\frac{\partial W^{int}}{\partial \underset{\sim}{d}}-\frac{\partial W^{ext}}{\partial \underset{\sim}{d}}=\underset{\sim}{f}^{init}-\underset{\sim}{f}^{ext}$$

This case has a stable local minimum. That is we want the $\underset{\sim}{d}$ for of

$$\min W(\underset{\sim}{d}) \text{ subject to } \underbrace{g_I(\underset{\sim}{d})=0,\ I=1(1)n_c}_{\text{essential BC}}$$

Typically work our way to the minimum through an iterative process where we are at $\underset{\sim}{d}_n$ and want to determine a $\underset{\sim}{\Delta d}$ from

a Taylor's series expansion (Linearized eq)

$$0 = \underset{\sim}{r}(\underset{\sim}{d}_v) + \frac{\partial \underset{\sim}{r}(\underset{\sim}{d}_v)}{\partial \underset{\sim}{d}} \underset{\sim}{\Delta d} = \underset{\sim}{r}(\underset{\sim}{d}_v) + \frac{\partial^2 W(\underset{\sim}{d}_v)}{\partial \underset{\sim}{d} \partial \underset{\sim}{d}} =$$

$$\underset{\sim}{r}(\underset{\sim}{d}_v) + \underset{\sim}{A}\,\underset{\sim}{\Delta d} \quad ; \quad A_{ab} = \frac{\partial^2 W}{\partial d_a \partial d_b}$$

$\underset{\sim}{A} \Leftarrow$ Hessian matrix.

This is identical to the Jacobian matrix (with $S_D = 0$) we had before

$$\underset{\sim}{A} = \underset{\sim}{K}^{int} - \underset{\sim}{K}^{ext} \quad ; \quad K^{int}_{ab} = \frac{\partial^2 W^{int}}{\partial d_a \partial d_b} \; ; \; K^{ext}_{ab} = \frac{\partial^2 W^{ext}}{\partial d_a \partial d_b}$$

with this the linearized form is

$$(\underset{\sim}{K}^{int} - \underset{\sim}{K}^{ext})\,\underset{\sim}{\Delta d} = \underset{\sim}{r}(d_v)$$

## §6.3.7 Implementation of Newton's method

For Dynamic problems $S_D \neq 0$ we need to use time stepping (increment in time)

For $S_D = 0$ can not usually do the whole thing in one step - Convergence of Newton requires a starting point not too far away. Will typically increment loads

### Box 6.3 Flowchart for implicit time integration

1. Initial conditions and initialization of parameters:

   set $\mathbf{v}^0$, $\boldsymbol{\sigma}^0$; $\mathbf{d}^0=\mathbf{0}$, $n=0$, $t=0$; compute $\mathbf{M}$

2. Get $\mathbf{f}^0=\mathbf{f}(\mathbf{d}^0, 0)$
3. Compute initial accelerations $\mathbf{a}^n=\mathbf{M}^{-1}\mathbf{f}^n$
4. Estimate next solution: $\mathbf{d}_{\text{new}}=\mathbf{d}^n$ or $\mathbf{d}_{\text{new}}=\tilde{\mathbf{d}}^{n+1}$
5. Newton iterations for time step $n+1$:
   a. *getforce* computes $\mathbf{f}(\mathbf{d}_{\text{new}}, t^{n+1})$, see Box 6.1
   b. $\mathbf{a}^{n+1}=1/\beta\Delta t^2\left(\mathbf{d}_{\text{new}}-\tilde{\mathbf{d}}^{n+1}\right), \mathbf{v}^{n+1}=\tilde{\mathbf{v}}^{n+1}+\gamma\Delta t\mathbf{a}^{n+1}$ : see (6.3.4–5)
   c. $\mathbf{r}=\mathbf{M}\mathbf{a}^{n+1}-\mathbf{f}$
   d. compute Jacobian $\mathbf{A}(\mathbf{d})$
   e. modify $\mathbf{A}(\mathbf{d})$ for essential boundary conditions
   f. solve linear equations $\Delta\mathbf{d}=-\mathbf{A}^{-1}\mathbf{r}$
   g. $\mathbf{d}_{\text{new}}\leftarrow\mathbf{d}_{\text{old}}+\Delta\mathbf{d}$
   h. check convergence criterion; if not met, go to step 5a.
6. Update displacements, counter and time: $\mathbf{d}^{n+1}=\mathbf{d}_{\text{new}}$, $n\leftarrow n+1$, $t\leftarrow t+\Delta t$
7. Check energy balance
8. Output; if simulation not complete, go to 4.

### Box 6.4 Flowchart for equilibrium solution

1. Initial conditions and initialization: set $\mathbf{d}^0=\mathbf{0}$; $\boldsymbol{\sigma}^0$; $n=0$; $t=0$; $\mathbf{d}_{\text{new}}=\mathbf{d}^0$
2. Newton iterations for load increment $n+1$:
   a. *getforce* computes $\mathbf{f}(\mathbf{d}_{\text{new}},t^{n+1})$; $\mathbf{r}=\mathbf{f}(\mathbf{d},t^{n+1})$
   b. compute $\mathbf{A}(\mathbf{d}_{\text{new}})$
   c. modify $\mathbf{A}(\mathbf{d}_{\text{new}})$ for essential boundary conditions
   d. solve linear equations $\Delta\mathbf{d}=-\mathbf{A}^{-1}\mathbf{r}$
   e. $\mathbf{d}_{\text{new}}\leftarrow\mathbf{d}_{\text{new}}+\Delta\mathbf{d}$
   f. check error criterion; if not met, go to 2a.
3. Update displacements, step count and time: $\mathbf{d}^{n+1}=\mathbf{d}_{\text{new}}$, $n\leftarrow n+1$, $t\leftarrow t+\Delta t$
4. Output; if simulation not complete, go to 2.

The flowchart shows a procedure often called a full Newton algorithm, where the J matrix is evaluated and inverted in every iteration of the procedure. Many program modified Newton algorithm, in which the Jacobian is assembled and triangulated onl

The flow charts were for a full Newton algorithm where the Jacobian is formed and solved on each iteration

There a alternative strategies for dealing with the set-up (and solution) of the Jacobian (or alternative form).

where $\mathbf{g}_i$ is the out-of-balance force vector at the start of iteration $i$ . In this case a linear set of equations is solved at every iteration.
Next sections describe the methods that are available in DIANA. First three pure iterative procedures are presented: the *Newton-Raphson* method, the *Quasi-Newton* method and the *Constant Stiffness* method. Next, two variations that can be used in combination with these procedures are considered: the *Continuation* method and the *Line Search* method. Finally, several criteria to stop the iteration loop will be discussed.

Another variation of the iteration algorithm is the Arc-length method [§30.1.5.2]. This method adapts the increment size.

### 30.1.1.1 Newton-Raphson

Within the class of Newton-Raphson methods, generally two subclasses are distinguished: the *Regular* and the *Modified* Newton-Raphson method. Both methods use (30.7) to determine the iterative increment of the displacement vector. In a Newton-Raphson method, the stiffness matrix $\mathbf{K}_i$ represents the tangential stiffness of the structure:

$$\mathbf{K}_i = \frac{\partial \mathbf{g}}{\partial \Delta \mathbf{u}} \tag{30.8}$$

The difference between the Regular and the Modified Newton-Raphson method is the point at which the stiffness matrix is evaluated.

**Regular Newton-Raphson.**

In the Regular Newton-Raphson iteration the stiffness relation (30.8) is evaluated every iteration [Fig.30.2]. This means that the prediction of (30.7) is based on the last known or predicted situation, even if this is not an equilibrium state.

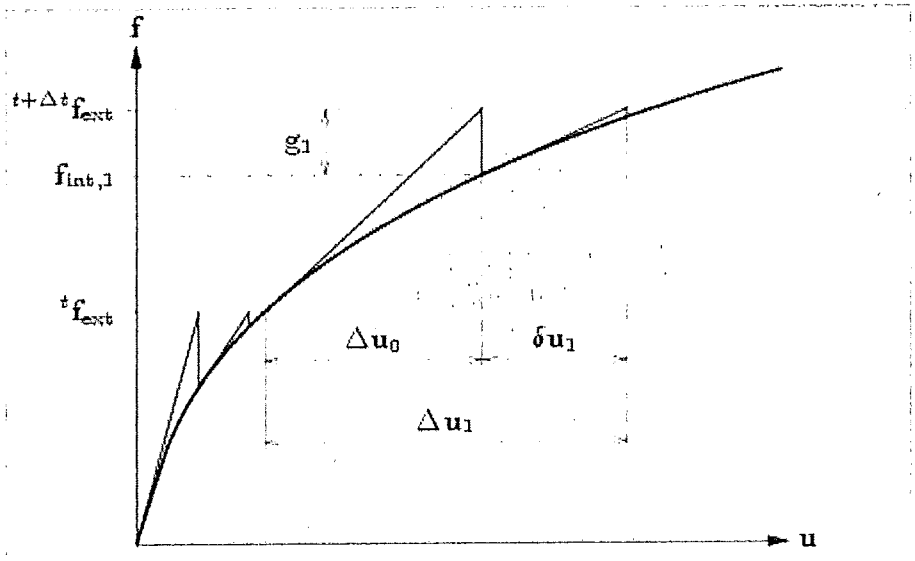



**Figure 30.2:** Regular Newton-Raphson iteration

The Regular Newton-Raphson method yields a quadratic convergence characteristic, which means that the method converges to the final solution within only a few iterations.

A disadvantage of the method is that the stiffness matrix has to be set up at every iteration and, if a direct solver is used to solve the linear set of equations, the time consuming decomposition of the matrix has to be performed every iteration as well. Moreover, the quadratic convergence is only guaranteed if a correct stiffness matrix is used and if the prediction is already in the neighborhood of the final solution. If the initial prediction is far from the final solution, the method easily fails because of divergence. In short:

*The Regular Newton-Raphson method usually needs only a few iterations, but every iteration is relatively time consuming.*

**Modified Newton-Raphson.**

The Modified Newton-Raphson method only evaluates the stiffness relation (30.8) at the start of the increment [Fig.30.3]. This means that the prediction is always based on a converged equilibrium state.

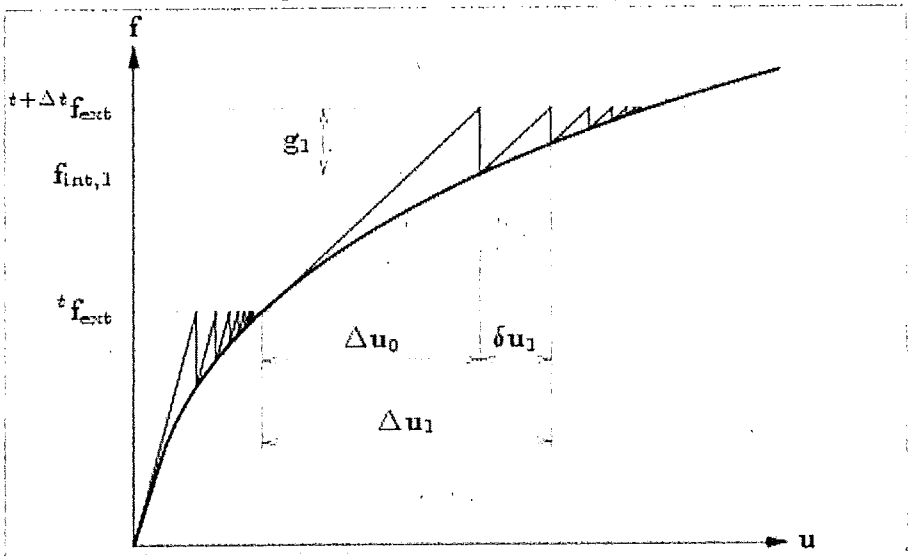



**Figure 30.3:** Modified Newton-Raphson iteration

Usually, Modified Newton-Raphson converges slower to equilibrium than Regular Newton-Raphson. However, for every iteration only the prediction of the iterative incremental displacements and the internal force vector has to be calculated, it is not necessary to set up a new stiffness matrix. If a direct solver for the linear set of equations is used, it is not necessary to perform the decomposition again, only the relatively fast substitution part will do. In short:

*The Modified Newton-Raphson method usually needs more iterations, but every iteration is faster than in Regular Newton-Raphson.*

In situations where Regular Newton-Raphson does not converge anymore, the Modified Newton-Raphson process can sometimes still converge. Small variations of both processes are possible by using the linear or previous stiffness for the first prediction and by setting up the current stiffness matrix after the first prediction. If unloading occurs, it can be advantageous to return to the linear stiffness, e.g. in a plasticity analysis.

## 30.1.1.2 Quasi-Newton

The Quasi-Newton method (also called 'Secant method') essentially uses the information of previous solution vectors and out-of-balance force vectors during the increment to achieve a better approximation [Fig.30.4]. Unlike Regular Newton-Raphson, the Quasi-Newton method does not set up a completely new stiffness

matrix every iteration.

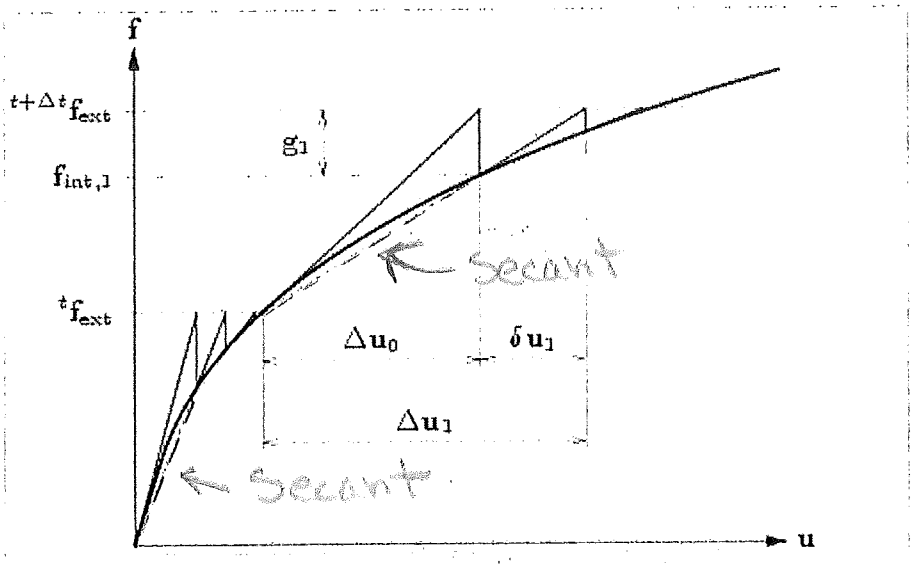



**Figure 30.4:** Quasi-Newton iteration

In this case the stiffness of the structure is determined from the known positions at the equilibrium path. If the iterative displacement increment is called $\delta \mathbf{u}_i$ and the change in out-of-balance force vector related to this increment $\delta \mathbf{g}_i = \mathbf{g}_{i+1} - \mathbf{g}_i$, the Quasi-Newton relation is

$$\mathbf{K}_{i+1}\ \delta \mathbf{u}_i = \delta \mathbf{g}_i \tag{30.9}$$

With a matrix $\mathbf{K}_i$ that fulfills (30.9), the next iterative increment is calculated from (30.7). For a system with more than one degree of freedom, the secant stiffness matrix **K** is not unique. The methods implemented in DIANA are known as the *Broyden*, the *Broyden-Fletcher-Goldfarb-Shanno* (BFGS) and the *Crisfield* methods. By substitution it can be seen that the following two matrices fulfill the Quasi-Newton relation (30.9).

$$\mathbf{K}_{i+1} = \mathbf{K}_i + \frac{(\delta \mathbf{g}_i - \mathbf{K}_i\, \delta \mathbf{u}_i)\, \mathbf{c}^{\mathrm{T}}}{\mathbf{c}^{\mathrm{T}} \delta \mathbf{u}_i} \tag{30.10}$$

$$\mathbf{K}_{i+1} = \mathbf{K}_i + \frac{(\delta \mathbf{g}_i - \mathbf{K}_i\, \delta \mathbf{u}_i)\mathbf{c}^{\mathrm{T}} + \mathbf{c}(\delta \mathbf{g}_i - \mathbf{K}_i\, \delta \mathbf{u}_i)^{\mathrm{T}}}{\mathbf{c}^{\mathrm{T}}\, \delta \mathbf{u}_i} - \frac{(\delta \mathbf{g}_i - \mathbf{K}_i\, \delta \mathbf{u}_i)^{\mathrm{T}}\, \delta \mathbf{u}_i\, \mathbf{c}\, \mathbf{c}^{\mathrm{T}}}{(\mathbf{c}^{\mathrm{T}}\, \delta \mathbf{u}_i)^2} \tag{30.11}$$

In (30.10) and (30.11) the vector **c** can be chosen freely. The Quasi-Newton methods can be used efficiently because the inverse of the new stiffness matrix can be derived directly from the previous secant stiffness and the update vectors by using the Sherman-Morrison formula.

**Broyden.**

If in (30.10) **c** is substituted by $\delta \mathbf{u}$ and $\mathbf{K}_{i+1}$ is inverted, the Broyden method results:

$$\mathbf{K}_{i+1}^{-1} = \mathbf{K}_i^{-1} + \frac{(\delta \mathbf{u}_i - \mathbf{K}_i^{-1}\,\delta \mathbf{g}_i)\,\delta \mathbf{u}_i^{\mathrm{T}}\,\mathbf{K}_i^{-1}}{\delta \mathbf{u}_i^{\mathrm{T}}\,\mathbf{K}_i^{-1}\,\delta \mathbf{g}_i} \tag{30.12}$$

**BFGS.**

More elaborative, (30.11) can yield the relation attributed to Broyden, Fletcher, Goldfarb and Shanno, and therefore known as the BFGS method:

$$\mathbf{K}_{i+1}^{-1} = \left( \mathbf{I} + \frac{\delta \mathbf{u}_i\,\delta \mathbf{g}_i^{\mathrm{T}}}{\delta \mathbf{u}_i^{\mathrm{T}}\,\delta \mathbf{g}_i} \right) \mathbf{K}_i^{-1} \left( \mathbf{I} - \frac{\delta \mathbf{g}_i\,\delta \mathbf{u}_i^{\mathrm{T}}}{\delta \mathbf{u}_i^{\mathrm{T}}\,\delta \mathbf{g}_i} \right) + \frac{\delta \mathbf{u}_i\,\delta \mathbf{u}_i^{\mathrm{T}}}{\delta \mathbf{u}_i^{\mathrm{T}}\,\delta \mathbf{g}_i} \tag{30.13}$$

The inverse secant stiffness matrices are not calculated explicitly, but the iterative displacements $\delta \mathbf{u}$ are calculated directly by substitution of (30.12) or (30.13) in (30.7). By successive application of (30.12) or (30.13), the correct secant stiffness can be calculated from the stiffness $\mathbf{K}_0$ that was used at the start of the increment and an update vector for every iteration. For every intermediate iteration one additional update vector is to be stored with size `number of degrees of freedom'. The higher the iteration number, the more additional storage is needed and the more additional vector calculations are to be performed.

**Crisfield.**

To avoid the increasing storage and computation time requirements for the Broyden and BFGS methods, Crisfield [15, §9.8] suggested to use only the most recent correction vector. For a one-dimensional situation this method still behaves as in Figure 30.4, but the Quasi-Newton relation (30.9) is not matched.
All three Quasi-Newton methods can be used irrespectively of the stiffness matrix $\mathbf{K}_0$ used for the first prediction. This could be a tangential stiffness matrix, as used in Figure 30.4, as well as a linear elastic stiffness matrix. These methods usually have a convergence rate between that of the Regular Newton-Raphson and the Modified Newton-Raphson schemes. This holds also for the time consumption. For large systems, especially when using a direct solver, the time used per iteration will be more closely to the Modified Newton-Raphson than to the Regular Newton-Raphson scheme. For the Broyden and the BFGS schemes the memory and time consumption will increase with the number of iterations.

## 30.1.1.3 Linear and Constant Stiffness

The Linear and Constant Stiffness iteration methods can be used if the other methods become unstable, or if it is desirable to keep certain characteristics.

**Linear Stiffness.**

The Linear Stiffness iteration method uses the linear stiffness matrix all the time [Fig.30.5].

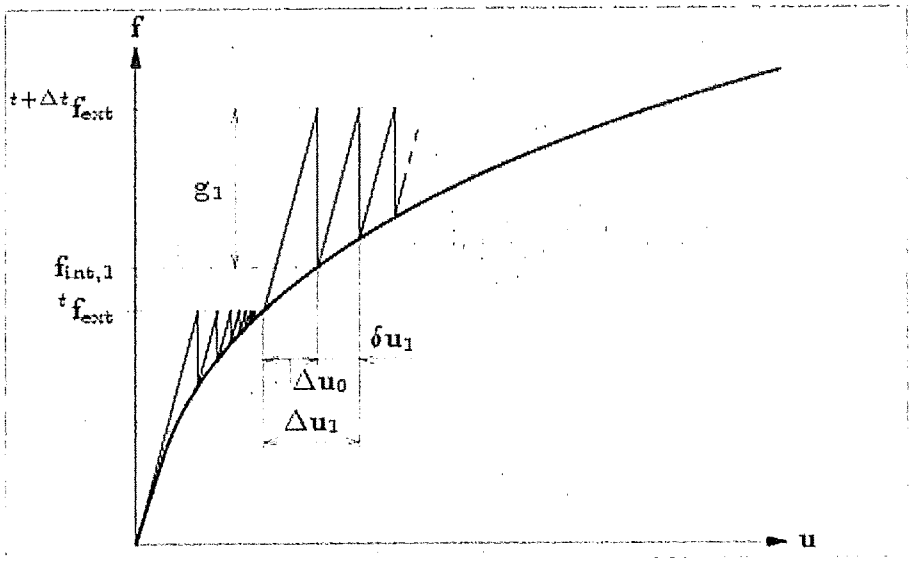



**Figure 30.5:** Linear Stiffness iteration

This method potentially has the slowest convergence, but it costs the least time per iteration since the stiffness matrix needs to be set up only once. Moreover, in case of a direct linear solver, the costly decomposition has to be performed only once. The Linear Stiffness method can also be advantageous if it is desirable that the stiffness matrix remains symmetric where the tangential stiffness matrix would become non-symmetric.

*The Linear Stiffness method is usually very robust, but it is very well possible that it follows unstable equilibrium paths after bifurcations.*

**Constant Stiffness.**

The Constant Stiffness method uses the stiffness matrix left behind by the previous increment. This means that if Newton-Raphson iterations are used during the first phase of an analysis and Constant Stiffness iterations in a second phase, the stiffness in the latter will be equal to the last calculated stiffness in the first. If the Constant Stiffness iteration is used since the first increment, this method equals the Linear Stiffness method.

*The Constant Stiffness method can be used if Newton-Raphson or Quasi-Newton methods fail after a number of successful increments.*

---

Next | Up | Previous | Contents | Index

**Next:** 30.1.2 Continuation **Up:** 30.1 Incremental-Iterative Solution **Previous:** 30.1 Incremental-Iterative Solution **Contents** **Index**

*DIANA-9.4.3 User's Manual - Analysis Procedures*
*First ed.*

# Implementation of essential B.C.

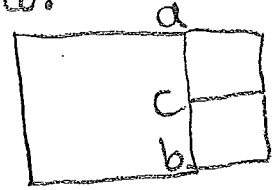
If all are homogeneous, that is $g_I(d) = 0, I = 1(1)n_c$ are of the specific form that each $g_I(d) = d_j = 0$ for some $j \in \{1, 2, \ldots n_{dof}\}$, $I = 1(1)n_c$

then we simply eliminate rows and columns, or put the dummy main diag. term. of 1 and 0 in RHS vector.

If the essential B.C are not all homogeneous but limited to the form of $g_I(d) = d_j = C$ for some $j \in \{1, 2, \ldots n_{dof}\}$, $I = 1(1)n_c$ then we can use the procedures we have done in class.

However, many find that procedure difficult to apply (I do not think it is) or have more complex "constraints", Like Linear constraints coupling multiple dof, then more alternative methods are required.

examples:




$d_c = \frac{1}{2}(d_a + d_b)$

The alternative methods include:

- Lagrange multipliers
- Penalty
- Augmental Lagrangian
- Perturbed Lagrangian

## Lagrange Multipliers

Augument the functional to include the constraints

$$W_L = W + \lambda_I g_I(d) = W + \underset{\sim}{\lambda}^T \underset{\sim}{g}(d)$$

$\lambda = \{\lambda_i \; \lambda_I\}$ is the vector of Lagrange multipliers

At the constrained equilibrium point

$$dW_L = 0 = dW + d(\underset{\sim}{\lambda}^T \underset{\sim}{g}(d)) = dW + d\lambda_I g_I(d)$$

satisfaction of this condition requires that derivatives WRT $d_a$; $a = 1(1) n_{dof}$, and $\lambda_I$, $I = 1(1) n_c$ are 0

WRT $d_a$

$$0 = \frac{\partial W_L}{\partial d_a} = \frac{\partial W}{\partial d_a} + \lambda_I \frac{\partial g_I}{\partial d_a} = r_a + \lambda_I \frac{\partial g_I}{\partial d_a} \quad a = 1(1) n_{dof}$$

also write as $\frac{\partial W}{\partial \underset{\sim}{d}} + \underset{\sim}{\lambda}^T \frac{\partial \underset{\sim}{g}}{\partial \underset{\sim}{d}}$

With respect to $\lambda_I$

$$0 = \frac{\partial W}{\partial \lambda_I} + \frac{\partial}{\partial \lambda_I}(\lambda_J g_J) = 0 + \frac{\partial \lambda_J}{\partial \lambda_I} g_J + \lambda_J \frac{\partial g_J}{\partial \lambda_I}$$

($g_J \to g_I$; $\frac{\partial \lambda_J}{\partial \lambda_I} \to 0$; $I = 1(1) n_c$)

get the desired constraint $\Rightarrow 0 = g_I$, $I = 1(1) n_c$

We have $n_{dof} + n_c$ equations with $n_{dof}$ unknowns in $d$ and $n_c$ unknowns in $\lambda$

Since we are concerned primarily in nonlinear systems - lets look at a linearized version - that is again take the first terms in a Taylor's expansion of

$$0 = r_a + \lambda_I \frac{\partial g_I}{\partial d_a} \overset{\approx}{\Rightarrow} \text{Linearized}$$

$$r_a + \lambda_I \frac{\partial g_I}{\partial d_a} + \frac{\partial r_a}{\partial d_b}\Delta d_b + \frac{\partial g_I}{\partial d_a}\Delta\lambda_I + \lambda_I \frac{\partial^2 g_I}{\partial d_a \partial d_b}\Delta d_b = 0$$

$r_a = \frac{\partial W}{\partial d_a}$

$a = 1(1)n_{dof}$
$b = 1(1)n_{dof}$

$$0 = g_I \overset{\approx}{\Rightarrow} g_I \frac{\partial g_I}{\partial d_a}\Delta d_a = 0 \qquad I = 1(1)n_c$$

define matrices:

$$\underset{\sim}{G} = [G_{Ia}] = \left[\frac{\partial g_I}{\partial d_a}\right], \quad \underset{\sim}{H}_I = [H_{ab}]_I = \left[\frac{\partial^2 g_I}{\partial d_a \partial d_b}\right]$$

Then we can the system for solving for the $\Delta$ terms

$$\begin{bmatrix} \underset{\sim}{A} + \lambda_I \underset{\sim}{H}_I & \underset{\sim}{G}^T \\ \underset{\sim}{G} & \underset{\sim}{0} \end{bmatrix} \begin{Bmatrix} \Delta d \\ \Delta\lambda \end{Bmatrix} = \begin{Bmatrix} -\underset{\sim}{r} - \lambda^T \underset{\sim}{G} \\ -\underset{\sim}{g} \end{Bmatrix}$$

$$\underset{\sim}{A} = \left[\frac{\partial^2 W}{\partial d_a \partial d_b}\right]$$

---

For a linear static system with linear constraints, $\underset{\sim}{G}\underset{\sim}{d} = \underset{\sim}{a}$, this simplifies to

$$\begin{bmatrix} \underset{\sim}{K} & \underset{\sim}{G}^T \\ \underset{\sim}{G} & \underset{\sim}{0} \end{bmatrix} \begin{Bmatrix} \underset{\sim}{d} \\ \underset{\sim}{\lambda} \end{Bmatrix} = \begin{Bmatrix} \underset{\sim}{f}^{ext} \\ \underset{\sim}{a} \end{Bmatrix}$$

$\underset{\sim}{K} \Rightarrow$ linear stiffness matrix

Note the $\underset{\sim}{Q}$ on the matrix diag makes the system really hard to solve.

Note: the $\underset{\sim}{\Lambda}$ are force type quantities (force needed to enforce the constraint) Thus we have a mixed system - It is not well conditioned (even without considering the $\underset{\sim}{Q}$ sub matrix.

## Penalty Methods

The augumented system is now:

$$W_p(\underset{\sim}{d}) = W(\underset{\sim}{d}) + \frac{1}{2}\beta g_I(\underset{\sim}{d}) g_I(\underset{\sim}{d}) = W + \frac{1}{2}\beta \underset{\sim}{g}^T \underset{\sim}{g}$$

Where $\beta$ is as large a number as you can work with

The simple idea is with $\beta$ very large the min of $W_p(d)$ will be attained only when the constraints are satisfied (other wise the $\frac{1}{2}\beta g_I(\underset{\sim}{d}) g_I(\underset{\sim}{d})$ term will contribute too much)

Stationary point (a minimum) satisfies

$$\frac{\partial W_p}{\partial d_a} = \frac{\partial W}{\partial d_a} + \beta g_I \frac{\partial g_I}{\partial d_a} = 0 \quad \text{or} \quad \underset{\sim}{r} + \beta \underset{\sim}{G}^T \underset{\sim}{G} = 0$$

The linearized form is

$$\left( \frac{\partial r_a}{\partial d_b} + \beta \frac{\partial g_I}{\partial d_b} \frac{\partial g_I}{\partial d_a} + \beta g_I \frac{\partial^2 g_I}{\partial d_a \partial d_b} \right) \Delta d_b = \left( -r_a - \beta g_I \frac{\partial g_I}{\partial d_a} \right)$$

$$\underset{\sim}{A}_p \Delta d = (\underset{\sim}{A} + \beta \underset{\sim}{G}^T \underset{\sim}{G} + \beta g_I \underset{\sim}{H}_I) \Delta d_b = \underset{\sim}{r} - \beta \underset{\sim}{g}^T \underset{\sim}{G}$$

# Augmented Lagrangian Method

A combination of the Lagrangian and Penalty method - Tends to improve conditioning of penalty method and helps the fact that the Lagrangian method is not necessarily positive-definite

$$W_{AL}(\underset{\sim}{d},\underset{\sim}{\lambda}) = W(\underset{\sim}{d}) + \underset{\sim}{\lambda}^T \underset{\sim}{g}(\underset{\sim}{d}) + \tfrac{1}{2}\beta \underset{\sim}{g}^T(\underset{\sim}{d})\,\underset{\sim}{g}(\underset{\sim}{d})$$

$$\frac{\partial W_{AL}}{\partial d_a} = \frac{\partial W}{\partial d_a} + \lambda_I \frac{\partial g_I}{\partial d_a} + \beta g_I \frac{\partial g_I}{\partial d_a} = 0$$

$$\underset{\sim}{r} + \underset{\sim}{\lambda}^T \underset{\sim}{G} + \beta \underset{\sim}{g}^T \underset{\sim}{G} = 0$$

$$\frac{\partial W_{AL}}{\partial \lambda_I} = g_I = 0 \;,\; \underset{\sim}{g} = 0$$

Linearized model is

$$r_a + \lambda_I \frac{\partial g_I}{\partial d_a} + \beta g_I \frac{\partial g_I}{\partial d_a} + \frac{\partial r_a}{\partial d_b}\Delta d_b + \Delta\lambda_I \frac{\partial g_I}{\partial d_a} + \lambda_I \frac{\partial^2 g_I}{\partial d_a \partial d_b}\Delta d_b + \beta \frac{\partial g_I}{\partial d_a}\frac{\partial g_I}{\partial d_b}\Delta d_b + \beta g_I \frac{\partial^2 g_I}{\partial d_a \partial d_b}\Delta d_b = 0$$

and $$g_I + \frac{\partial g_I}{\partial d_b}\Delta d_b = 0$$

In matrix form

$$\begin{bmatrix} \underset{\sim}{A} + \lambda_I \underset{\sim}{H}_I + \beta\left(\underset{\sim}{G}^T \underset{\sim}{G} + g_I \underset{\sim}{H}_I\right) & \underset{\sim}{G}^T \\ \underset{\sim}{G} & 0 \end{bmatrix} \begin{Bmatrix} \Delta\underset{\sim}{d} \\ \Delta\underset{\sim}{\lambda} \end{Bmatrix} = \begin{Bmatrix} -\left(\underset{\sim}{r} + \underset{\sim}{\lambda}^T \underset{\sim}{G} + \beta \underset{\sim}{g}^T \underset{\sim}{G}\right) \\ -\underset{\sim}{g} \end{Bmatrix}$$

Peturbed Lagrangian

$$W_{PL}(\underset{\sim}{d},\underset{\sim}{\lambda}) = W(\underset{\sim}{d}) + \underset{\sim}{\lambda}^T \underset{\sim}{g}(\underset{\sim}{d}) - \tfrac{1}{2}\epsilon\, \underset{\sim}{\lambda}^T \underset{\sim}{\lambda}$$

(not heavily used)

now $\epsilon$ is small

# Convergence of the Newton Iteration -

Common convergence criteria -
i) Magnitude of residual $\underset{\sim}{r}$
ii) magnitude of solution increment $\Delta \underset{\sim}{d}$
iii) energy error criteria

Basic forms

residual error criterion

$$\|\underset{\sim}{r}\|_{l_2} = \left(\sum_{a=1}^{ndof} r_a^2\right)^{1/2} \leq \epsilon \max\left(\|\underset{\sim}{f}^{ext}\|_{l_2}, \|\underset{\sim}{f}^{int}\|_{l_2}, \|M\underset{\sim}{a}\|_{l_2}\right)$$

displace increment criteria

$$\|\Delta \underset{\sim}{d}\|_{l_2} = \left(\sum_{a=1}^{ndof} \Delta d_a^2\right)^{1/2} \leq \epsilon \quad \|\underset{\sim}{d}\|_{l_2}$$

energy convergence (← kinetic energy)

$$|\Delta \underset{\sim}{d}^T \underset{\sim}{r}| = |\Delta d_a r_a| \leq \epsilon_{max}(W^{ext}, W^{int}, W^{kin})$$

---

Line search to improve convergence when there are convergence issues

A form of line search -

Basic of a line search -
i) Fine a good direction to go
ii) Look along that direction to decide how far.

We assume that the direction of $\Delta \underset{\sim}{d}$ is good - however, using the specific vector may not be optimal.

Note that it is much less expensive to evaluate the residual than it is to solve for a new direction.

Thus we consider updating

$\underset{\sim}{d} = \underset{\sim}{d}_{last} + \xi \Delta d$ where $\xi$ is determined by "searching" along the direction $\Delta d$

Can use methods like golden sections to find $\xi$

The $\alpha$-Method for

-High frequency noise can be a problem when using Newmark $\beta$ method - this is why the $\gamma$ term, the artificial vecosity is included - we would like to minimize the $\gamma$ term since this degrades accuracy - "artifical" is not good.

The goal of the $\alpha$ method is to have get more control with less detriment to accuracy

-This will be done by introducing an intermedite evaluation time for use with the force terms -

$$d^{n+\alpha} = (1+\alpha) d^{n+1} - \alpha d^{n}$$ to be use as follows

$$0 = \underset{\sim}{r}(\underset{\sim}{d}^{n+1}, t^{n+1}) = S_D \underset{\sim}{M} \underset{\sim}{a}^{n+1} - f^{ext}(\underset{\sim}{d}^{n+\alpha}, t^{n+1}) + f^{int}(\underset{\sim}{d}^{n+\alpha}, t^{n+1})$$

this will yield a linearized Jacobian matrix

$$A = \frac{\partial \underset{\sim}{r}(d_v^{n+\alpha})}{\partial \underset{\sim}{d}} = \frac{S_D}{\beta \Delta t^2} \underset{\sim}{M} + (1+\alpha) \frac{\partial \underset{\sim}{f}^{int}(d_v^{n+\alpha})}{\partial \underset{\sim}{d}} - (1+\alpha) \frac{\partial \underset{\sim}{f}^{ext}(d_v^{n+\alpha})}{\partial \underset{\sim}{d}}$$

Adv. of Implicit Methods - They are unconditionally stable for linear transient problems.

For nonlinear problems there are specific cases where stability is proven - but there is no proof for the range of situations found in practice

Experience does indicate that in all cases time steps for implicit can always be longer (by a lot) than for explicit

The primary driver in time step selection in implicit is accuracy - remember it is time discretization.
There is also the robustness of the newton iteration - put simply steps must be small - enough that linearization is acceptable

The Newmark method is second order accurate - truncation error $O(\Delta t^3)$ -
⇒ One can look into more accurate time stepping methods -

# Convergence of Newton - Plates conditions on $\tilde{A}$

Simply Stated:

i) $\tilde{A}$ should be a smooth function of $\underline{d}$

ii) $\tilde{A}$ must be invertable and stay well conditioned throughout the space of solutions to $\underline{d}$

And there can be points where $\tilde{A}^{-1}$ does not exist.

In many problems of interest $\underline{d}$ is not smooth throughout the space of solution within the entire domain. However, Newton still heavily used since it often works for the entire process - although at possibly at a reduce rate of convergence

It is of value to monitor convergence to see how you are doing on your problem for representative test cases

The quadratic convergence of Newton (for linear) means that

$$\| \underline{d}_{\nu+1} - \underline{d} \|_{l_2} \leq c \| \underline{d}_{\nu} - \underline{d} \|_{l_2}$$

loss of smoothness

By doing this - one can see where you are having problems

Nonlinear static problems often have slower convergence than dynamic problems since the Mass matrix make Inertia more positive.

→ Discontinuities — like elastic/plastic, contact cause greatest problems.

When do you use which Method -

Depends on -

1. Form of PDE - Elliptic, Parabolic, Hyperbolic
2. Smoothness of Data -
3. Response of interest

For Parabolic problems - implicit methods win when data is smooth. Often still best when there is "roughness" in data (eg. elastic/plastic)

Issue with Explicit methods - since stable time step is function of element size (quadratic reduction) they become too expensive.

There are cases where the degree of roughness is so large that the loss of convergence of Newton with any reasonable time step makes it unacceptable due to high cost per time step. - Then want to go to explicit methods (also easier to get parallel performance). Standard examples are crash analysis, high speed dynamics with contact/friction etc.

In hyperbolic problems the questions are harder - often a function of the response of interest and behavior

For example - structural dynamics often best with implicit

For wave problem with high frequency domains of interest → explicit may be better

Other issues-

Linearization - We hid a whole lot in our terms $\underset{\sim}{f}^{int}$ and $\underset{\sim}{f}^{ext}$

You have to carefully deal with those and doing that is a strong function of the problem classes - Belytschko focus on solids/structures

Stability -

Have to account for shape of response surfaces - find points with solutions can change from one surface to another -

standard case - structural stability - buckling

→ Have to be careful to follow branches of interest

Lots of technical details that must account for properties of systems being considered.

---

Note there are additional Newton like methods. Often referred to as Quasi Newton Methods -

Modified Newton - use same Jacobian for a while

Secant methods

Broyden update

Broyden - Fletcher - Gold farb - Shanno BFGS