

18

Purely Incremental Methods: General Control

In this Chapter we continue the development of the purely incremental methods under general incremental control conditions. A general nonlinear transformation of state and control parameters to pseudo-time is introduced. Computable forms of this constraint are specialized to arclength and hyperelliptic control. The computational implications of these decisions are discussed, and a practical implementation of hyperplane-distance arclength control outlined.

§18.1. Parametric Form

In previous Chapters it was noted that continuation solution methods under λ (load) control have difficulty traversing limit points. This shortcoming can be circumvented through the use of more general increment control schemes. This generalization can be practically effected by adjoining algebraic constraint equations such as those listed in Chapter 16.

These more general forms of increment control can be described on a uniform basis as follows. Because the main effect of enforcing the constraint $c(\Delta \mathbf{u}, \Delta \lambda) = 0$, is to link the increments of \mathbf{u} and λ , we express the response in the pseudo time *parametric form*

$$\mathbf{u} = \mathbf{u}(t), \quad \lambda = \lambda(t). \quad (18.1)$$

If $t \equiv \lambda$ we would of course regress to the λ -control parametrization

$$\mathbf{u} = \mathbf{u}(\lambda), \quad (18.2)$$

of the equilibrium path, and nothing would be gained. But the pseudotime t in (18.1) is now at our disposal and we can try to do better. Differentiating (18.1) with respect to t we get

$$d\mathbf{u} = \dot{\mathbf{u}} dt, \quad d\lambda = \dot{\lambda} dt = (1/f) dt, \quad (18.3)$$

where we have called for convenience

$$f = \frac{1}{\dot{\lambda}} = \frac{dt}{d\lambda}. \quad (18.4)$$

§18.1.1. Requirements

The incremental path equation becomes

$$\dot{\mathbf{r}} = \frac{\partial \mathbf{r}(\mathbf{u}, \lambda)}{\partial t} = \frac{\partial \mathbf{r}}{\partial \mathbf{u}} \dot{\mathbf{u}} - \frac{\partial \mathbf{r}}{\partial \lambda} \dot{\lambda} = \mathbf{K}\dot{\mathbf{u}} - \mathbf{q}/f = \mathbf{0}. \quad (18.5)$$

At limit points $d\lambda = 0$. By “smooth traversal” of a limit point is meant that t varies regularly as a “vehicle odometer” as that limit point is crossed. Consequently $f = dt/d\lambda$ must go to infinity at a limit point. This reasoning shows why $t \equiv \lambda$ does not work, because if so $f = 1$ everywhere. It also follows that the relation between λ and a useful t must be necessarily nonlinear, for if we take $t = c_1\lambda + c_2$ with $c_1 \neq 0$, f is always $1/c_1$.

It is computationally desirable, however, that in “almost linear” portions of the response f approaches a finite value, because in the limit we would like to recover the conventional methods of linear structural analysis.

§18.1.2. The Arclength Choice

Now which quantity becomes infinite at limit points? An obvious choice is the incremental velocity $\mathbf{v} = \mathbf{K}^{-1}\mathbf{q}$ or, more precisely, some norm of it. A particularly attractive choice in light of its geometric significance is

$$f = \sqrt{1 + \mathbf{v}^T \mathbf{v}}, \quad (18.6)$$

This corresponds to taking

$$dt \equiv ds = f d\lambda = \mathbf{v} d\mathbf{u} + d\lambda, \quad (18.7)$$

As shown in Chapter 4, ds is the *differential arclength* of the response curve in state-control space. Consequently s is the arclength traversed along this curve measured from some arbitrary point, such as the last solution. Note the following attractive features of this choice:

- (a) At limit points $f \rightarrow \infty$, as required.
- (b) At turning points where $\mathbf{v} \rightarrow 0$, $f \rightarrow 1$, so (unlike state control) traversal controlled by increasing arclength is not affected.
- (c) If the response is linear f maintains a constant value.

The arclength choice is no panacea, however, because three computational problems remain, listed below in order of increasing difficulty.

- (d) The exact arclength s_n from the last computed solution at $P_n(\mathbf{u}_n, \lambda_n)$ to another point $P(\mathbf{u}, \lambda)$ on the equilibrium path is given by the path integral

$$s_n = \int_{P_n \rightarrow P} ds = \int_{P_n \rightarrow P} \sqrt{1 + \mathbf{v}^T \mathbf{v}} ds. \quad (18.8)$$

But as written this expression is not directly computable because it requires knowledge of $\mathbf{v} = \mathbf{K}^{-1}\mathbf{q}$ along the equilibrium path that emanates from the last solution, which is precisely what we want to compute. Consequently an increment control constraint such as

$$s_n = \ell_n, \quad (18.9)$$

makes no computational sense. This minor difficulty is eliminated by using an *approximation* Δs_n to the arclength. The approximation is directly or indirectly effected by introducing an appropriate constraint equation, as discussed below.

- (e) The choice (18.6) intermixes state parameters, which have generally physical dimension (of displacement), with the dimensionless scalar 1. This mixture can introduce difficulties in that the computed solutions are not invariant with respect to the choice of state parameter dimensions. It can be corrected by employing appropriate state-scaling techniques as discussed in Chapter 4.
- (f) Traversal of bifurcation points remain difficult or impossible without additional “tricks”. The key roadblock is that the continuous parametrization (18.1) breaks down at bifurcation points,

because it cannot simultaneously represent the two or more branches that intersect there.¹ Circumventing this difficulty within the context of purely incremental methods is not easy, and consideration is deferred to later Chapters.

We now study two computable approximations to the arclength parametrization.

§18.2. Hyperplane Distance Control

The simplest computable approximation to the exact arclength constraint $s_n = \ell_n$ is $\Delta s_n = \ell_n$, where Δs_n is the hyperplane distance (4.23) to the last solution \mathbf{u}_n, λ_n . Thus the increment control constraint is

$$\Delta s_n = (\mathbf{v}_n^T \Delta \mathbf{u}_n + \Delta \lambda_n) / f_n = \ell_n, \quad \text{where} \quad f_n = \sqrt{1 + \mathbf{v}_n^T \mathbf{v}_n}. \quad (18.10)$$

in which ℓ_n , which controls the magnitude of the n^{th} incremental step, is either specified or automatically adjusted through some accuracy control rule, as discussed below.

Substituting $\Delta \mathbf{u}_n = \mathbf{v}_n \Delta \lambda_n$ into the above we get the Forward Euler incremental scheme

$$\boxed{\begin{aligned} \mathbf{v}_n &= \mathbf{K}_n^{-1} \mathbf{q}_n, & f_n &= \sqrt{1 + \mathbf{v}_n^T \mathbf{v}_n}, & \Delta \lambda_n &= \pm \frac{\ell_n}{f_n}, \\ \mathbf{u}_{n+1} &= \mathbf{u}_n + \mathbf{v}_n \Delta \lambda_n, & \lambda_{n+1} &= \lambda_n + \Delta \lambda_n. \end{aligned}} \quad (18.11)$$

It remains to select the sign of $\Delta \lambda_n$ and the magnitude of ℓ_n . The proper sign for $\Delta \lambda_n$ can be chosen according to the criteria discussed in §16.7. If the positive external work criterion (16.16) is used, $\Delta \lambda$ is chosen to have the sign of $\mathbf{q}^T \mathbf{v}$. As noted there, this simple rule fails at bifurcation and turning points, wherein the angle criterion discussed there should be used.

As for the magnitude of ℓ_n , two possibilities exist. Either ℓ_n is kept constant and equal to the given ℓ_0 , or it may be automatically adjusted in an adaptive scheme. If the latter strategy is adopted, parameter ϵ is used to bound the local error as explained in §16.5. Without going through the derivation, which is given in §16.4 below, the result is

$$\ell_n = \frac{2\epsilon \ell_{n-1}}{a_{n-1}} \quad (18.12)$$

where a_{n-1} is estimated by (16.23) in which \mathbf{v} is replaced by \mathbf{v}/f . This must be complemented by a *minimum arclength travel* distance condition:

$$\ell_n \geq \ell_{min}, \quad (18.13)$$

which avoids “getting stuck” at limit points under certain conditions, as well as a *maximum arclength travel* distance condition

$$\ell_n \leq \ell_{max}, \quad (18.14)$$

to avoid surprises in hitting rapidly changing portions of the response.

¹ A second order rate form is in fact required in the vicinity of a simple bifurcation point.

One simple way to define these minmax values is to set

$$\ell_{min} = \frac{\ell_0}{\ell_{fac}}, \quad \ell_{max} = \ell_0 \ell_{fac}, \quad (18.15)$$

where the factor ℓ_{fac} , which is typically 1 to 20, is part of the input data. If $\ell_{fac} = 1$ this rule effectively forces fixed a fixed $\ell_n = \ell_0$.

Remark 18.1. The essential difference between the arclength and load control schemes is the arclength form does not have trouble crossing limit points. As the limit point is approached, $\Delta\lambda$ is driven to zero and automatically changes sign upon crossing it. The only numerical danger is that of hitting the singularity exactly so that the factorization of \mathbf{K} fails; however, handling of this emergency in the computer implementation is not difficult.

Remark 18.2. In a poorly scaled problem the previous scheme should be modified by replacing f by its scaled equivalent:

$$f_n = \sqrt{1 + \mathbf{v}_n^T \mathbf{S}^2 \mathbf{v}_n}, \quad (18.16)$$

where \mathbf{S}^2 is a diagonal scaling matrix. Some appropriate choices for \mathbf{S}^2 are discussed in the next Chapter.

§18.3. Global Hyperelliptic Constraint

As a second example, if we adopt the unscaled global hyperelliptical constraint (16.27) reproduced here for convenience

$$a_n^2 \Delta \mathbf{u}_n^T \Delta \mathbf{u}_n + b_n^2 = \ell_n^2, \quad (18.17)$$

where a_n , b_n and ℓ_n are scalars given at each step, we get the formula

$$\Delta \lambda_n = \frac{\ell_n}{\pm \sqrt{a_n^2 \mathbf{v}_n^T \mathbf{v}_n + b_n^2}}, \quad (18.18)$$

which is followed by solving for $\Delta \mathbf{u}_n$.

For poorly scaled problems, or problems in which \mathbf{u} collects quantities of different physical magnitudes, we should use the scaled form

$$a_n^2 \Delta \mathbf{u}_n^T \mathbf{S}^2 \Delta \mathbf{u}_n + b_n^2 = \ell_n^2,$$

where the diagonal matrix \mathbf{S}^2 is chosen to take care of scaling. Then

$$\Delta \lambda_n = \frac{\ell_n}{\pm \sqrt{a_n^2 \mathbf{v}_n^T \mathbf{S}^2 \mathbf{v}_n + b_n^2}} \quad (18.19)$$

One interesting possibility is to choose $\mathbf{S}^2 = \mathbf{K}$. This may be viewed as an *energy constraint* because $\mathbf{v}^T \mathbf{K} \mathbf{v} = \mathbf{q}^T \mathbf{v}$ is external incremental work. With that choice, and adopting the positive-external-work criterion to choose the sign of $\Delta\lambda$, we have

$$\Delta \lambda_n = \frac{\ell}{+\sqrt{a_n^2 |\mathbf{q}_n^T \mathbf{v}_n| + b_n^2}} \text{sign}(\mathbf{q}_n^T \mathbf{v}_n). \quad (18.20)$$

Remark 18.3. Equation (18.19) represents a “closed” constraint surface (a hyperellipse in state-control space) and thus it may be thought of as “safer” than the hyperplane distance constraint, which is “open”. Although this consideration has some merit if a corrective phase that moves on that surface follow, it has no weight in a purely incremental method. One drawback of the hyperelliptic form is the need to choose a_n and b_n in addition to ℓ_n at each step. Although this adds flexibility, it can complicate the implementation and require more run-time decisions.

Remark 18.4. The scalar $\Delta \mathbf{u}^T \mathbf{K} \Delta \mathbf{u}$ is not necessarily positive if \mathbf{K} is indefinite; hence the need for taking the absolute value of $\mathbf{q}^T \mathbf{v}$ in (18.20).

Remark 18.5. Diagonal scaling on displacement increments provides an intermediate choice between unscaled and \mathbf{K} -scaled forms. That is, choose $\mathbf{S}^2 = \text{diag}(\mathbf{K})$.

§18.4. Accuracy Control

Assuming that $t = s$ so that arclength increment control is used, the Taylor expansion about a solution point reads

$$\begin{bmatrix} \Delta \mathbf{u} \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{v}_f \\ (1/f) \end{bmatrix} \Delta s + \frac{1}{2} \begin{bmatrix} \frac{d\mathbf{v}_f}{ds} \\ \frac{d(1/f)}{ds} \end{bmatrix} (\Delta s)^2 + O(\Delta s^3), \quad (18.21)$$

where $\mathbf{v}_f = \mathbf{v}/f = \mathbf{v}/\sqrt{1 + \mathbf{v}^T \mathbf{v}}$. The truncation error is the quadratic term in Δs . Proceeding as in the load control case we obtain

$$\begin{bmatrix} e_v \\ e_\lambda \end{bmatrix} = \frac{1}{2} \begin{bmatrix} |d\mathbf{v}_f/ds| \\ |d(1/f)/ds| \end{bmatrix} (\Delta s)^2. \quad (18.22)$$

where e_v and e_λ are the norms of the local truncation errors associated with \mathbf{u} and λ , respectively. We shall focus on controlling accuracy by monitoring e_v , since sufficient accuracy on λ generally follows. Again defining as in §14.5 the desired local accuracy level by the scalar ϵ , we arrive at the rule: adjust ℓ_n by

$$|\Delta s_n| \leq \frac{2\epsilon |\Delta s_{n-1}|}{a_{n-1}} \quad (18.23)$$

in which a_{n-1} is estimated by

$$a_{n-1} = \frac{\mathbf{v}_n/f_n - \mathbf{v}_{n-1}/f_{n-1}}{\mathbf{v}_n/f_n}, \quad (18.24)$$

Observe that this is the same as the adjustment rule (14.23)–(14.24) but with \mathbf{v} replaced by $\mathbf{v}_f = \mathbf{v}/f$.

§18.5. Numerical Stability

Following the same procedure as §14.4 it can be shown that the linearized problem that governs numerical stability becomes

$$\begin{bmatrix} \frac{d\Delta \mathbf{u}}{ds} \\ \frac{d\Delta \lambda}{ds} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathbf{v}_f}{\partial \mathbf{u}} & \frac{\partial \mathbf{v}_f}{\partial \lambda} \\ \frac{\partial (1/f)}{\partial \mathbf{u}} & \frac{\partial (1/f)}{\partial \lambda} \end{bmatrix} \begin{bmatrix} \Delta \mathbf{u} \\ \Delta \lambda \end{bmatrix}, \quad (18.25)$$

in which $\mathbf{v}_f = \mathbf{v}/f = \mathbf{v}/\sqrt{1 + \mathbf{v}^T \mathbf{v}}$. The amplification matrix \mathbf{A} is that shown in brackets above. So far this problem appears to be analytically intractable and no general conclusions may be drawn so far. However, numerical evidence show that arclength-control incremental methods are as stable as those using load control. Thus the empirical rule of Chapter 14 may be used; namely if ℓ is adjusted for accuracy, setting $\epsilon < 1$ takes care of stability.

§18.6. What Happens at a Limit Point

Suppose that the computed solution process approaches an isolated limit point at which \mathbf{K} has the null eigenvector \mathbf{z} normalized to unit length, that is

$$\mathbf{K}\mathbf{z} = \mathbf{0}, \quad \mathbf{z}^T\mathbf{z} = 1. \quad (18.26)$$

As the critical point is approached, it can be shown (using the theory of inverse iteration) that

$$d\lambda \rightarrow 0, \quad \mathbf{v} = \mathbf{K}^{-1}\mathbf{q} \rightarrow \alpha\mathbf{z}, \quad \alpha \rightarrow \pm\infty, \quad (18.27)$$

That is, \mathbf{v} tends to become parallel to \mathbf{z} although its norm goes to (plus or minus) infinity. But the f -normalized \mathbf{v} does approach the normalized eigenvector:

$$\mathbf{v}_f = 1/f\mathbf{v} = \frac{\mathbf{v}}{\sqrt{1 + \mathbf{v}^T\mathbf{v}}} \rightarrow \pm\mathbf{z}. \quad (18.28)$$

and the incremental equations approach

$$\begin{bmatrix} \Delta\mathbf{u} \\ \Delta\lambda \end{bmatrix} = \begin{bmatrix} \pm\mathbf{z} \\ 0 \end{bmatrix} \ell \quad (18.29)$$

Therefore the f -normalization of \mathbf{v} automatically take care of aligning the incremental direction normally to the λ axis.

Remark 18.6. Some minor safeguards remain: if the tangent stiffness is evaluated at or very near the limit point, $\mathbf{K}(\mathbf{u})$ may be numerically singular, in which case a simple remedy is to change \mathbf{u} by a tiny amount and try again (a more theoretically sound approach based on penalty spring stabilization is discussed later). This simple technique is used in the computer program for Exercises 18.1–18.5. Also if ℓ is automatically adjusted by accuracy requirements it must not be allowed to fall under a minimum value.

§18.7. An Automated Incremental Algorithm

We are now in a position to give the outline of a arclength-controlled incremental algorithm essentially based on the scheme described previously. This is done in Table 18.1. The steps listed therein are for a single stage. (In programs that accept multistage analysis, “stop” is replaced by a save, exit, check error conditions and recover if possible, reset and restart process with the next stage. Although programming that sequence can be elaborate, it does not involve conceptual difficulties.)

§18.8. Assessment of Purely Incremental Methods

The simplest load-controlled incremental scheme described in Chapter 17 has been extensively used as a stand-alone method (that is, not combined with equilibrium corrections) in early implementations (1955-1965) of finite-element-based nonlinear structural analysis. It survives in this primitive form in a surprisingly large proportion of finite element computer programs; particularly those aimed at treating highly nonlinear material behavior. From current perspective it suffers from two serious disadvantages.

Drift Error. The residual force vector \mathbf{r} never enters the calculations. Consequently, the deviation from the equilibrium path due to the propagation and accumulation of local integration errors cannot

**Table 18.1 - Arclength-Controlled Incremental Solution
Forward Euler Procedure for a Single Stage**

At $\lambda = 0$ we know $\mathbf{u} = \mathbf{u}_0$. We want to advance λ in an external-work-increasing process until either the norm of the state vector \mathbf{u} exceeds u_{max} , the magnitude of λ exceeds λ_{max} , the number of incremental steps exceeds n_{max} , or an impassable bifurcation point is reached. Parameters $\epsilon < 1$, and ℓ_0 are specified. Set $n = 0$ and perform the following steps.

- Step 1.* Form and factor stiffness matrix \mathbf{K}_n . If the factorization fails on account of singularity, perturb \mathbf{u} by a tiny amount and repeat. If this failure repeats after a certain number of tries, stop with appropriate error message.
- Step 2.* Form right-hand side \mathbf{q}_n and solve $\mathbf{K}_n \mathbf{v}_n = \mathbf{q}_n$ for the incremental velocity \mathbf{v}_n . Form $f_n = \sqrt{1 + \mathbf{v}_n^T \mathbf{v}_n}$.
- Step 3.* If an adaptive stepsize scheme is used, adjust ℓ_n as per §18.2, else keep $\ell_n = \ell_0$. Set $\Delta\lambda_n = \ell_n / f_n$, and give it the sign of $\mathbf{q}_n^T \mathbf{v}_n$.
- Step 4.* Compute $\Delta\mathbf{u}_n = \mathbf{v}_n \Delta\lambda_n$. Advance $\mathbf{u}_{n+1} = \mathbf{u}_n + \Delta\mathbf{u}_n$ and $\lambda_{n+1} = \lambda_n + \Delta\lambda_n$. If adaptive ℓ control is used, save \mathbf{v}_f to be used in the estimation of a_{n-1} .
- Step 5.* If $|\mathbf{u}_{n+1}|$ exceeds u_{max} , or $|\lambda_{n+1}|$ exceeds λ_{max} , or n exceeds n_{max} , stop. Else set $n \leftarrow n + 1$ and return to Step 1.

If scaling of \mathbf{v} is introduced because of the physical dimensionality discrepancy between λ and \mathbf{u} , the simplest implementation is to define a reference length L_{ref} as part of the inputs. Then $f_n = \sqrt{1 + \tilde{\mathbf{v}}_n^T \tilde{\mathbf{v}}_n}$, in which $\tilde{\mathbf{v}}_n = \mathbf{v}_n / L_{ref}$.

be eliminated by corrective iteration. In practical terms, this means that realistic estimates of the response tracing accuracy can be obtained only by rerunning the problem with several increment sizes.

Computational Expense. To keep the drifting error down, many small steps may be required, particularly in “difficult” regions of the response. But at each step the stiffness matrix must be formed and factored. This can be an expensive proposition in two- and three-dimensional problems. To reduce the stiffness recalculation cost, the “pseudo-force” methods discussed in Chapter 19 have been used extensively in plasticity and viscoelasticity calculations. These methods are affected, however, by serious numerical stability difficulties.

What are the advantages? First, given the early applications of finite element methods (see Remark below), incremental methods are quite easy to program *as extensions of linear analysis codes*. The absence of the residual vector is in fact helpful, since linear analysis does not require it whereas the stiffness matrix is always available. And the underlying concept (follow the physics) is readily understood by practicing engineers. The virtues of simplicity and physical transparency should

never be underestimated!

Second, in problems that exhibit strong path-dependency, as typified by flow plasticity models, the many-small-steps requirement is not necessarily a hindrance, since the increments have to be kept small anyway to avoid unacceptable element-level errors (for example, yield surface drift) in material-law calculations.

Finally, the high frequency of stiffness matrix assembly and refactorization has a silver lining: it is difficult to miss critical points. The fact that purely incremental methods have trouble tracing post-buckling branches beyond bifurcation points is irrelevant to situations in which the determination of such points, rather than traversing them, is the main objective of the analysis.

Remark 18.7. The preference of incremental over corrective methods in the early implementation of nonlinear finite element analysis has historical roots. Finite element methods were invented in the aircraft/aerospace industry where *linear* analysis dominates. (This also helps to explain the initial popularity of the force method, which is ill-suited for nonlinear analysis.) Furthermore, early excursions in the nonlinear world involved fairly mild nonlinearities. Thus, developers of displacement-based finite element codes passed naturally from the linear stiffness equations

$$\mathbf{K}\mathbf{u} = \mathbf{q} \quad (18.30)$$

to the incremental form

$$\begin{aligned} \mathbf{K} \Delta \mathbf{u} &= \Delta \mathbf{q}, \\ \mathbf{u}_{new} &= \mathbf{u}_{old} + \Delta \mathbf{u} \end{aligned} \quad (18.31)$$

and gave scant thought to the nonlinear equilibrium equations.

Homework Exercise for Chapters 16-18

Incremental Solution Methods

Exercises 18.1 and 18.2 pertain to the response analysis by purely incremental methods, of the nonlinear response of the two-bar arch example structure of Chapter 8, which was treated by the TL description.

The structure properties are: span $S = 2$, height $H = 1$ (hence the rise angle is $\alpha = 45^\circ$), elastic modulus $E = 1$ and cross section area $A_0 = 1$. The applied loads are $f_X = 0$ and $f_Y = \lambda F$ where $F = -1$ is constant. This gives a downward vertical applied load if $\lambda > 0$. For this rise angle, the first limit point of the Total Lagrangian model analytically occurs at load level $\lambda_L \approx 0.136$ and a vertical deflection of $u_{YL} \approx -0.425$. Traversal of limit points is done through a positive-work advancing criterion: $\mathbf{q}^T \mathbf{v} > 0$.

Exercises 18.1 and 18.2 should be done with the *Mathematica* Notebook called `IncSolTwoBarArch.nb`, which has been posted on the course Web site. This is a very rough conversion from the ancient Fortran code used in previous offerings. [The conversion took much longer than anticipated despite the simplicity of the code.]

Despite the roughness of this implementation, it can be used to illustrate the behavior of two integrators: Forward Euler (FE) and Midpoint Rule (MR), combined with three increment control strategies: Load Control (LC), Displacement Control (DC) and Arclength Control (AC). [The Classical Runge Kutta (RK4) integrator will be added later as an Exercise.] One significant advantage of *Mathematica* over Fortran is the availability of built-in graphics. Thus response plots, for example, can be immediately generated as part of the output, helping quick visualization of method performance.

EXERCISE 18.1 [D:15] Prepare a hierarchical diagram of Cells 1-12 of the `IncSolTwoBarArch.nb` Notebook, beginning with the main program given in Cell 12. Note which module calls which and write down the purpose of each module in one or two lines along the module name. Return this diagram as answer to the homework.*

EXERCISE 18.2 [C:20] Run the four scripts (main programs) in Cells 12 through 15 of `IncSolTwoBarArch.nb`. Briefly explain what they do, and how the four method combinations driven by them stack up in terms of (i) robustness in traversing limit points, (ii) accuracy in locating the first limit point (analytical values are given above), and (iii) accuracy in crossing $\lambda = 0$ at $u_Y = -1$. Please attach the four response plots of u_Y versus λ to your returned homework (do not bother with printouts). (I forgot to include method labels in the `ListPlot` commands to identify which method is which; these may be written by hand).

Note: before you can run those scripts, Cells 1-11 should be initialized. A quick way to accomplish that in version 3.0 is to click Kernel \rightarrow Evaluate \rightarrow Evaluate Initialization. Do this twice to get rid of error message boxes.

* If you are not sure of what a hierarchical diagram is, go to the IFEM Web page: <http://caswww.colorado.edu/courses.d/IFEM.d/Home.html> and look up Exercise 15.7 in Chapter 15.

EXERCISE 18.3 [C:25] Consider the one-DOF nonlinear problem (not a structure) governed by the force residual equation

$$(\lambda - 1)^2 + (u + 1)^2 - 2 = 0 \quad (\text{E18.1})$$

This represents a circle of radius $\sqrt{2}$ and center $(1, -1)$ in the (λ, u) plane. Suppose that one starts at the reference state $u = \lambda = 0$ and tries to trace the response, with a constant stepsize, by going around the circle counterclockwise and eventually returning to the origin. In doing so you need to cross two turning points and two limit points. While traversing limit points is easy using Arclength Control and a positive work criterion, turning points are trouble.

Program from scratch an incremental solution to this problem using the Midpoint Rule, Arclength Control, constant ℓ , and the angle criterion² to keep a positive traversal direction, and try it.³ If your implementation works all the way you are ahead of all commercial nonlinear FEM codes in this regard.

EXERCISE 18.4 [C:25] The fixed-step classical 4th order Runge Kutta integrator (RK4) is implemented in Cell 10 of the `IncSolTwoBarArch.nb` Notebook.⁴ It is identified as `integ="RK4"`. Check visually if the implementation is correct by comparing to the description, for example, in Numerical Recipes.

Run the same arch problem as in Exercise 8.2 using RK4 combined with Load Control and Arclength Control; experiment with `e11` and `nmax` and report on whether you can obtain high accuracy (see 8.2 for how to assess it) with a fairly large stepsize, say `e11=0.1`.

EXERCISE 18.5 [C:25] It was shown in Chapter 8 that if the rise angle α of the two-bar arch example structure exceeds 60° , i.e. $\tan \alpha > \sqrt{3}$, it will fail by bifurcation first. For example $S = 2$ and $H = 2$ would do it because $\tan \alpha = 2$. However if you set those inputs and run the programs in `IncSolTwoBarArch.nb` the bifurcation point will be completely masked; you will see only the symmetric solution passing two limit points.

One simple technique to make “dumb incremental solvers,” like those provided in the Notebook, pay attention to bifurcation points is to inject *artificial imperfections*. This can be done, for example, by putting a fictitious but tiny load system that disturbs the symmetric response. For example, define the reference crown load in force as $\{-0.001, -1\}$ instead of $\{0, -1\}$. Set $S = H = 2$ and play with the tiny X -force, the increment length `e11` and number of steps `nmax` until you see a decent tracing of bifurcation post-buckling: at a certain load level u_X will increase rapidly, signaling that the arch is buckling horizontally. To see that better, do the `ListPlot` of u_X versus λ collected in `list uXvslambda` found near the bottom of the driver cell. Comment on what combination of method and solution parameters let you succeed.

Note (1). Bifurcation experiments can be found in the Notebook `IncBifSolTwoBarArch.nb` also posted on the web site.

Note (2). If you take the response long enough you may be able to have the structure return to the primary symmetric path upon passing through the second bifurcation point, but that may take lots and lots of steps since the implementation uses a constant stepsize ℓ .

² The computation of the angle between \mathbf{v}_n and \mathbf{v}_{n-1} is illustrated in the posted Notebook. It is saved as part of the solution table, although it is not used in the solution procedure therein. The variable is called `a` or `an` and is actually the cosine of that angle, which is simply the dot product of those velocity vectors normalized to unit length.

³ The positive-work criterion $\mathbf{q}^T \mathbf{v} > 0$ fails because one needs to release work along some parts of the response trajectory. In fact the net work on doing a complete circle is zero.

⁴ For a description of the RK4 algorithm see any book on numerical methods for ODEs. For example the widely used *Numerical Recipes in Fortran*; it is presented in Section 16.1 of the second edition.