Synthesis Tools for Structural Dynamics and Partitioned Analysis of Coupled Systems

C. A. Felippa and K. C. Park
Department of Aerospace Engineering Sciences
and Center for Aerospace Structures
University of Colorado at Boulder
Boulder, Colorado 80309-0429, USA
carlos.felippa@colorado.edu, kcpark@colorado.edu

Abstract. This tutorial paper is extracted from a set of graduate lectures on the model-based, time-domain simulation of structural dynamics and coupled systems. For the treatment of the latter, emphasis is placed on partitioned analysis procedures. Although the subject emerged in the present form over 20 years ago, the time-consuming study of competing formulations and implementations can be streamlined through a relatively recent tool: computer algebra systems available on inexpensive desktop and laptop computers. The presentation of the lectures emphasizes that symbiosis between human and machine.

Keywords: structural dynamics, multiphysics, coupled systems, computer algebra, Mathematica, differential equations, partitioned analysis, time integration.

INTRODUCTION

What’s hot in computational mechanics? The three “multis”: multiscale, multiphysics and multiprocessing. Collectively these trends lead to the formulation and model-based simulation of coupled systems: systems whose behavior is driven by the interaction of functionally distinct components. The nature of these components broadly defines the “multi” discipline. Material models spanning a range of physical scales are the framework of multiscale simulations. Multiphysics addresses the interaction of different physical behavior, as in structures and fluids, at similar physical scales. Multiprocessing refers to computational methods that use system decomposition to achieve concurrency. Summarizing, the breakdown of a system is dictated by: (S) physical scales in multiscale, (P) physical behavior in multiphysics, and (C) implementation considerations in multiprocessing. Obviously a three-level hierarchy: (S)-(P)-(C), can be discerned, but this level of full generality has not been reached in practice.

These hot areas have a common feature: explosive complexity. The choice among models, algorithms and implementations grows combinatorially in the number of components. Consider for example a fluid-structure interaction problem. Whereas the use of a FEM model for the structure part would be viewed as natural, the choice of fluid model can vary across a wide spectrum, depending on what physical effects (flow, turbulence, gravity waves, acoustic waves, mixing, moving boundaries, bubbles, etc.) are to be captured. Discretization methods vary accordingly. If control is added to the picture, for example to simulate maneuvers of a flexible airplane, further choices emerge. So far this applies to components in isolation. The treatment of interaction requires additional decisions at the interfaces. For example: do the meshes match? can meshes slip past each other? how can reduced or spectral models be linked to physical models? To make things more difficult, often models that work correctly with isolated components break down when coupled. So the modeling level becomes crowded. But that is not all. Proceeding to the solution algorithm and implementation levels brings up further choices, in particular if parallel processing issues are important.

How to cope with this combinatorial explosion of choices? Analytical treatment can go so far in weeding out choices. The traditional way to go beyond that frontier is numerical experimentation. This also has limitations: the most one can do is take “pot shots” at the computational application
domain. It can only show that a model works. A “bridging” tool between human analytics and numerical testing has gained popularity over the past decade: computer algebra systems (CAS) able to carry out symbolic computations. This is due to technical improvements in general-purpose CAS such as Mathematica and Maple, as well as availability on inexpensive personal computers and laptops. (This migration keeps licensing costs reasonable.) Furthermore, Maple is available as a toolbox of the widely used Matlab system. A related factor is wider exposure in higher education: many universities now have site licenses, which facilitate lab access and use of CAS in course assignments and projects.

In computational mechanics, CAS tools can be used for a spectrum of tasks: formulation, prototyping, implementation, performance evaluation, and automatic code generation. Although occasionally advertised as “doing mathematics by computer” the phrase is misleading: as of now only humans can do mathematics. But a CAS can provide timely help. Here is a first-hand example: the first author needed four months to formulate, implement and test the 6-node triangle in the summer–fall of 1965 as part of thesis work [16]. Using a CAS, a similar process can be completed in less than a week, and demonstrated to students in 20 minutes.

The first author has developed finite elements with CAS support since 1984 — using the venerable Macsyma for the Free Formulation elements presented in [6,22]. The development of templates as a unified framework for element families [24,26] would not have been possible without that assistance.

In the present lectures, Mathematica [92] is employed as a CAS “filter tool” for the design and analysis of time integration methods for structural dynamics and coupled systems. The main purpose of the filter is to weed out unsuitable methods by working on test systems. This initial pass streamlines subsequent stages of numerical experimentation.

1. COUPLED SYSTEMS

This chapter introduces the concept of coupled system from a descriptive standpoint.

§1.1. Systems

The American Heritage Dictionary lists eight meanings for system. By itself the term is used here in the sense of a functionally related group of components forming or regarded as a collective entity. This definition uses “component” as a generic term that embodies “element” or “part,” which connote simplicity, as well as “subsystem,” which connotes complexity. We restrict attention to mechanical systems, and in particularly those of importance in Aerospace, Civil and Mechanical Engineering.

§1.2. System Decomposition

Systems are analyzed by decomposition or breakdown. Complex systems are amenable to many kinds of decomposition chosen according to specific analysis or design objectives. These lectures focus on decompositions called partitions that are suitable for computer simulation. Such simulations aim at describing or predicting the state of the system under specified conditions viewed as external to the system. A set of states ordered according to some parameter such as time or load level is called a response.

System designers are not usually interested in detailed response computations per se, but on project goals such as cost, performance, lifetime, fabricability, inspection requirements and satisfaction of mission objectives. The recovery of those overall quantities from simulations is presently an open problem in computer system modeling and one that is not addressed here.

The term partitioning identifies the process of spatial separation of a discrete model into interacting components generically called partitions. The decomposition may be driven by physical, functional, or computational considerations. For example, the structure of a complete airplane can be decomposed into substructures such as wings and fuselage according to function. Substructures can be further decomposed into submeshes or subdomains to accommodate parallel computing requirements. Subdomains are composed of individual elements. Going the other way, if that
flexible airplane is part of a flight simulation, a top-level partition driven by \textit{physics} is into fluid and structure (and perhaps control and propulsion) models. This kind of multilevel partition hierarchy at common physical scales: coupled system, structure, substructure, subdomain and element, is typical of present practice in modeling and computational technology.

§1.3. \textbf{Coupled System Terminology}

Because coupled systems have been studied by many people from many angles, terminology is far from standard. The following summary is one that has evolved for the computational simulation, and does reflect personal choices. Most of the definitions follow usage introduced in a 1983 review article [65]. Casual readers may want to skim the following material and return only for definitions.

A \textit{coupled system} is one in which physically or computationally heterogeneous mechanical components interact dynamically.

The interaction is called \textit{one-way} if there is not feedback between subsystems, as illustrated in Figure 1.1(a) for two subsystems identified as X and Y. The interaction is called \textit{two-way} (or generally multiway) if there is feedback between subsystems, as illustrated in Figure 1.1(b). In this case, which will be the one of primary interest here, the response has to be obtained by solving \textit{simultaneously} the coupled equations which model the system. “\textit{Heterogeneity}” is used in the sense that \textit{component simulation benefits from custom treatment}.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure1.jpg}
\caption{Interaction between two subsystems X and Y: (a) one way, (b) two way.}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure2.jpg}
\caption{Decomposition of an aeroelastic FSI coupled system: \textit{partitioning} in space and \textit{splitting} in time. 3D space is shown as “flat” for visualization convenience. Spatial discretization (omitted for clarity) may predate or follow partitioning. Splitting (here for fluid only) is repeated over each time step and obeys time discretization constraints.}
\end{figure}

As noted above the decomposition of a complex coupled system for simulation is \textit{hierarchical} with two to four levels being common. At the first level one encounters two types of subsystems, embodied in the generic term \textit{field}:

\textbf{Physical Subsystems.} Subsystems are called \textit{physical fields} when their mathematical model is described by field equations. Examples are mechanical and non-mechanical objects treated by continuum theories: solids, fluids, heat and electromagnetics. Occasionally those components may be intrinsically discrete, as in actuation control systems or rigid-body mechanisms. In such a case the term “physical field” is used for expediency, with the understanding that no spatial discretization process is involved.
Artificial Subsystems. Sometimes artificial subsystems are incorporated for computational convenience. Two examples: dynamic fluid meshes to effect volume mapping of Lagrangian to Eulerian descriptions in interaction of structures with fluid flow, and fictitious interface fields, often called "frames", that facilitate information transfer between two subsystems.

§1.3.1. Fields
A coupled system is characterized as two-field, three-field, etc., according to the number of different fields that appear in the first-level decomposition.

For computational treatment of a dynamical coupled system, fields are discretized in space and time. A field partition is a field-by-field decomposition of the space discretization. A splitting is a decomposition of the time discretization of a field within its time step interval. See Figure 1.2. In the case of static or quasi-static analysis, actual time is replaced by pseudo-time or some kind of control parameter.

Partitioning may be algebraic or differential. In algebraic partitioning the complete coupled system is spatially discretized first, and then decomposed. In differential partitioning the decomposition is done first and each field then discretized separately.

Algebraic partitioning was originally developed for matched meshes and substructuring; cf. Figure 1.3, but later work has aimed at simplifying the treatment of nonmatched meshes through frames [68,69]. Differential partitioning often leads to nonmatched meshes, which are typical of fluid-structure interaction as depicted in Figure 1.4, and handles those naturally.

A property is said to be interfield or intrafield if it pertains collectively to all partitions, or to individual partitions, respectively.

A common use of this qualifier concerns parallel computation. Interfield parallelism refers to the implementation of a parallel computation scheme in which all partitions can be concurrently processed for time-stepping. Intrafield parallelism refers to the implementation of parallelism for an individual partition using a second-level decomposition; for example into substructures or subdomains.

§1.3.2. Examples
Experiences discussed here come from systems where a structure is one of the fields. Accordingly, all of the following examples list structures as one of the field components. The number of interacting fields is given in parenthesis.

Fluid-structure interaction (2) Thermal-structure interaction (2)
Control-structure interaction (2) Control-fluid-structure interaction (3)
Electro-thermo-mechanical interaction (3) Fluid-structure-combustion-thermal interaction (4)
When a fluid is one of the interacting fields a wider range of computational modeling possibilities opens up as compared to, say, structures or thermal fields. For the latter finite element discretization methods can be viewed as universal in scope. On the other hand, the range of fluid phenomena is controlled by several major physical effects such as viscosity, compressibility, mass transport, gravity and capillarity. Incorporation or neglect of these effects gives rise to widely different field equations as well as discretization techniques.

For example, the interaction of an acoustic fluid with a structure in aeroacoustics or underwater shock is computationally unlike that of high-speed gas dynamics with an aircraft or rocket, a surface ship with ocean waves, or flow through porous media. Even more variability can be expected if chemistry and combustion effects are considered. Control systems also exhibit modeling variabilities that tend not to be so pronounced, however, as in the case of fluids. The partitioned treatment of some of these examples is discussed further in subsequent sections.

§1.3.3. Interior and Exterior Problems

When interacting fields occupy nonoverlapping regions of space, and one of which is a structure, the following terminology is commonly used.

Interior Problem. The structure surrounds the nonstructural fields. For example, in the tank problem depicted in Figure 1.5.

Exterior Problem. The structure is surrounded by the nonstructural fields, which may be viewed to be unbounded. For example, in the submarine problem depicted in Figure 1.6.

§1.3.4. Splitting and Fractional Step Methods

Splitting methods for the equations of mathematical physics predate partitioned analysis by two decades. In the American literature they can be originally traced to the mid-50s development of alternating direction methods by Peaceman, Douglas and Rachford [11,71]. Similar developments were independently undertaken in the early 1960s by the Russian school led by Bagrinovskii, Godunov and Yanenko, and eventually unified in the method of fractional steps [93]. The main applications of these methods have been the equations of gas dynamics in steady or transient forms, discretized by finite difference methods. They are particularly suitable for problems with layers or stratification, for example atmospheric dynamics or astrophysics, in which different directions are treated by different methods.

The basic idea is elegantly outlined by Richtmyer and Morton [75]. Suppose the governing equations in divergence form are $\frac{\partial W}{\partial t} = AW$, where the operator $A$ is split into $A = A_1 + A_2 + \ldots + A_q$. Pick a time difference scheme and replace $A$ successively in it by $qA_1, qA_2, \ldots$, each for a fraction $h/q$ of the temporal stepsize $h$. Then a multidimensional problem can be replaced by a succession of simpler 1D problems. Splitting may be additive, as in the foregoing example, or multiplicative. Since the present lectures focus on partitioned analysis, the discussion of these variations falls beyond its scope.

Comparison of those methods with partitioned analysis makes clear that little overlap occurs. Splitting is appropriate for the treatment of a field partition such as a fluid, if the physical structure of such partition display strong directionality dependencies. Consequently splitting methods are seen to pertain to a lower level of a top-down hierarchical decomposition.
§1.4. Scenarios
Because of their variety and combinatorial nature, the simulation of coupled problems rarely occurs as a predictable long-term goal. Some more likely scenarios are as follows.

Research Project. Members of a research group develop personal expertise in modeling and simulation of two or more isolated problems. Individuals are then called upon to pool that disciplinary expertise into solving a coupled problem.

Product Development. The design and verification of a product requires the concurrent consideration of interaction effects in service or emergency conditions. The team does not have access, however, to software that accommodates those requirements.

Software House. A company develops commercial application software targeted to single fields as isolated entities: a CFD gas-dynamics code, a structure FEM program and a thermal conduction analyzer. As the customer base expands, requests are received for allowing interaction effects targeted to specific applications. For example the CFD user may want to account for moving rigid boundaries, then interaction with a flexible structure, and finally to include a control system.

The following subsection discusses approaches to these scenarios.

§1.4.1. Approaches
To fix the ideas we assume that the simulation calls for dynamic analysis that involves following the time response of the system. [Static or quasi-static analyses can be included by introducing a pseudo-time history parameter.] Further, the modeling and simulation of isolated components is assumed to be well understood. The three approaches to the simulation of the coupled system are:

Field Elimination. One or more fields are eliminated by techniques such as integral transforms or model reduction, and the remaining field(s) treated by a simultaneous time stepping scheme.

Monolithic or Simultaneous Treatment. The whole problem is treated as a monolithic entity, and all components advanced simultaneously in time.

Partitioned Treatment. The field models are computationally treated as isolated entities that are separately stepped in time. Interaction effects are viewed as forcing effects that are communicated among individual components using prediction, substitution and synchronization techniques.

Elimination is restricted to special linear problems that permit efficient decoupling. It often leads to higher order differential systems in time, or to temporal convolutions, which can be the source of numerical difficulties. On the other hand the monolithic and partitioned treatments are general in nature. No technical argument can be made for the overall superiority of either. Their relative merits are not only problem dependent, but are interwined with human factors as discussed below.

§1.4.2. Monolithic vs. Partitioning
Keywords that favor the partitioned approach are: customization, independent modeling, software reuse, and modularity.

Customization. This means that each field can be treated by discretization techniques and solution algorithms that are known to perform well for the isolated system. The hope is that a partitioned algorithm can maintain that efficiency for the coupled problem if (and that is a big if) the interaction effects can be also efficiently treated.

Independent Modeling. The partitioned approach facilitates the use of non-matching models. For example in a fluid-structure interaction problem the structural and fluid meshes need not coincide at their interface; cf. Figure 1.4. This translates into project breakdown advantages in analysis of complex systems such as aircraft or ships. Separate models can be prepared by different design teams, including subcontractors that may be geographically distributed.

Software Reuse. Along with customized discretization and solution algorithms, customized software (private, public or commercial) may be available. Furthermore, there is often a gamut of customized service tools such as mesh generators and visualization programs. The partitioned approach facilitates taking advantage of existing code. This is particularly suitable to academic
environments, in which software development tends to be cyclical and loosely connected from one project to another.

**Modularity.** New methods and models may be introduced in a modular fashion according to project needs. For example, it may be necessary to include local nonlinear effects in an individual field while keeping everything else the same. Implementation, testing and validation of incremental changes can be conducted in a modular fashion.

These advantages are not cost free. The partitioned approach requires careful formulation and implementation to avoid degradation in stability and accuracy. Parallel implementations are particularly delicate. Gains in computational efficiency over a monolithic approach are not guaranteed, particularly if interactions occur throughout a volume as is the case for thermal and electromagnetic fields. Finally, the software modularity and modeling flexibility advantages, while desirable in academic and research circles, may lead to anarchy in software houses.

In summary, circumstances that favor the partitioned approach for tackling a new coupled problem are: a research environment with few delivery constraints, access to existing and reusable software, localized interaction effects (e.g. surface versus volume), and widespread spatial/temporal component characteristics. The opposite circumstances: commercial environment, rigid deliverable timetable, massive software development resources, global interaction effects, and comparable length/time scales, favor a monolithic approach.

Most of the following lecture material focuses on the partitioned approach.

### 2. PARTITIONED ANALYSIS OVERVIEW

This chapter provides a quick overview of the partitioned analysis procedures applied to a model problem. Focus is on the so-called staggered solution procedure, which is important on account of extensive use in applications as well as simplicity of implementation.

#### §2.1. The Basic Idea

Consider the two-way interaction of two scalar fields, $X$ and $Y$, sketched in Figure 2.1. Each field has only one state variable identified as $x(t)$ and $y(t)$, respectively, which are assumed to be governed by the first-order differential equations

\[ \begin{align*}
3x' + 4x - y &= f(t) \\
y' + 6y - 2x &= g(t)
\end{align*} \quad (2.1)
\]

in which $f(t)$ and $g(t)$ are the applied forces. Treat this by Backward Euler integration in each component:

\[ \begin{align*}
x_{n+1} &= x_n + h x_{n+1} \\
y_{n+1} &= y_n + h y_{n+1}
\end{align*} \quad (2.2)
\]

where $x_n \equiv x(t_n)$, $y_n \equiv y(t_n)$, etc. At each time step $n = 0, 1, 2, \ldots$ we get

\[ \begin{bmatrix} 3 + 4h \\ -2h \end{bmatrix} \begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} h f_{n+1} + 3x_n \\ h g_{n+1} + y_n \end{bmatrix} \quad (2.3)
\]

in which $\{x_0, y_0\}$ are provided by the initial conditions. In the monolithic or simultaneous solution approach, (2.3) is solved at each timestep, and that is the end of the story.

#### §2.1.1. Staggering

A simple partitioned solution procedure is obtained by treating (2.3) with the following staggered partition with prediction on $y$:

\[ \begin{align*}
\begin{bmatrix} 3 + 4h & 0 \\ -2h & 1 + 6h \end{bmatrix} \begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} h f_{n+1} + 3x_n + hy_{n+1} \\ h g_{n+1} + y_n \end{bmatrix}.
\end{align*} \quad (2.4)
\]
1. (P) Predict: \[ y^P_{n+1} = y_n + h \dot{y}_n \] (for example)

2. (Ax) Advance \( x \): \[ x_{n+1} = \frac{1}{3 + 4h} (h f_{n+1} + 3 x_n + h y^P_{n+1}) \]

3. (S) Substitute: \[ x_{n+1} = x_{n+1} \] (trivial here)

4. (Ay) Advance \( y \): \[ y_{n+1} = \frac{1}{1 + 6h} (h g_{n+1} + y_n + 2h x_{n+1}) \]

Figure 2.2. Basic steps of a red/black staggered solution.

Here \( y^P_{n+1} \) is a predicted value or simply the predictor. Two common choices are \( y^P_{n+1} = y_n \) (called the last-solution predictor) and \( y^P_{n+1} = y_n + h \dot{y}_n \). The basic solution steps are displayed in Figure 2.2. The main achievement is that systems \( X \) and \( Y \) can be now solved in tandem.

A state-time diagram of these steps, with time along the horizontal axis, is shown in Figure 2.3.

Suppose that fields \( X \) and \( Y \) are handled by two separate but communicating programs. If intrafield advancing arrows are omitted, we obtain a zigzagged picture of interfie data transfers between the \( X \)-program and the \( Y \)-program, as sketched in Figure 2.4(a). This interpretation motivated the name staggered solution procedure introduced in [59].

§2.1.2. Concerns

In linear problems the first concern with partitioning should be degradation of time-stepping stability caused by prediction.

In the foregoing example this is not significant. The spectral analysis presented in the Appendix of [25], which embodies (2.4) as instance, shows that staggering does not harm stability or even accuracy, if the integrator and predictor are appropriately chosen.

In fact, staggered procedures are very effective for coupled first-order parabolic systems. But for more general problems, particularly those modeled by oscillatory second order ODEs, the reduction of stability can become serious or even catastrophic. Prediction might be done on the \( y \) field, leading to a zigzagged diagram with substitution on \( x \). The stability of both choices can be made to coalesce by adjusting predictors.

Once satisfactory stability is achieved, the next concern is accuracy. This is usually degraded with respect to that attainable by the monolithic scheme. In principle this can be recovered by iterating the state between the fields. Iteration is done by cycling substitutions at the same time station. However, interfield iteration generally costs more than cutting the timestep to attain the same accuracy level. If, as often happens, the monolithic solution is more expensive than the staggered solution for the same timestep, we note the emergence of a tradeoff.

In strongly nonlinear problems, such as gas dynamic flows in the transonic regime, stability and accuracy tend to be interwined (because numerical stability is harder to define) and they are usually considered together in method design. The expectation is for a method that operates well at a reasonable timestep.
Examination of Figure 2.4(a) shows that this simple staggered scheme is unsuitable for interfield parallelization because programs must execute in strictly serial fashion: first X, then Y, etc. This was of little concern when the method was formulated in the mid 1970s [59] as computers were then sequential. The variant sketched in Figure 2.4(b) permits the programs to advance their internal state concurrently, which allows interfield parallelization. More effective schemes, which do not require prediction on both fields, have been developed over the past decade and are discussed at length in [72,73].

§2.1.3. Devices of Partitioned Analysis

As the simple example illustrates, partitioned analysis requires the examination of alternative algorithm and implementation possibilities as well as the study of tradeoffs. Figure 2.5 displays, using interfield time stepping diagrams, the main “tools of the trade.” Some devices such as prediction, substitution and iteration have been discussed along with the foregoing example. Others will emerge in the application problems discussed in later chapters.

![Figure 2.5. Devices of partitioned analysis time stepping.](image)

Notes and Bibliography

The partitioned treatment of coupled systems involving structures emerged independently in the mid 1970s at three locations: Northwestern University, by T. Belytschko and R. Mullen; Cal Tech, by T. J. R. Hughes and W. K. Liu; and Lockheed Palo Alto Research Laboratories (LPARL), by J. A. DeRuntz, C. A. Felippa, T. L. Geers and K. C. Park. These three groups targeted different applications and pursued different problem-decomposition techniques. For example, Belytschko and Mullen [3–5] studied node-by-node partitions and subcycling whereas Hughes, Liu and coworkers developed element-by-element implicit-explicit partitions [44–46]. This topic evolved at Stanford into element-by-element iterative solvers [47]. The work of these two groups focused on structure-structure and fluid-structure interaction treated by all-FEM discretizations.

The remainder of this section (and of the paper) focuses on the authors’ work. Research in Coupled Problems at LPARL originated in the simulation of the elastoacoustic underwater shock problem for the Navy. In this work a finite element computational model of the submerged structure was coupled to Geers’ Doubly Asymptotic Approximation (DAA) boundary-element model of the exterior acoustic fluid [29–32]. In 1975 a staggered solution procedure, used later as case study, was developed for this coupling. This was presented in a 1977 article [59] and later extended to more general applications [20,21]. The staggered solution scheme was eventually subsumed in the more general class of partitioned methods [61,62]. These have been surveyed in several articles [23,25,65,66].

In 1985–86 Geers, Park and Felippa moved from LPARL to the University of Colorado at Boulder. Park and Felippa moved formation of the Center for Aerospace Structures. Research work in coupled problems continued at CAS but along individual interests. Park began work in computational control-structure interaction [2,67], whereas Felippa began studies in superconducting electrothermometrics [78].

---

1 With the exception of an exotic machine known as the ILLIAC IV.
Farhat, who joined CAS in 1987, began research in computational thermoelasticity [13] and aeroelasticity [12]. The latter effort prospered as it acquired a parallel-computing flavor and was combined with advances in the FETI structural solver [14,15].

Research in Coupled Problems at CAS was given a boost in 1992 when the National Science Foundation announced grantees for the first round of Grand Challenge Applications (GCA) Awards. This competition was part of the U.S. High Performance Computing and Communications (HPCC) Initiative established in 1991. An application problem is labeled a GCA if the computational demands for realistic simulations go far beyond the capacity of present sequential or parallel supercomputers. The GCA coupled problems addressed by the award were: aeroelasticity of a complete aircraft, distributed control-structure interaction, and electrothermomechanics with phase changes. A renewal project to address multiphysics problems was awarded in 1997. This grant addressed application problems involving fluid-thermal-structural interaction in high temperature components, turbulence models, piezoelectric and control-surface control of aeroelastic systems, and optimization of coupled systems. These focus applications were interwoven with research in computer sciences, applied mathematics and computational mechanics.

3. INTRODUCTION TO STABILITY ANALYSIS

This chapter provides an introduction to the stability analysis of discretized ODEs. It is a tutorial review of some basic definitions and techniques distributed over many books. The next Chapter discusses tools for symbolic analysis of stability. As noted in Chapter 2, using partitioned analysis gives high flexibility of implementation. The downside of this freedom is the large number of possibilities. The motivation for doing stability analysis first is to filter out unsuitable choices.

§3.1. Stability

The term “stable” informally means resistant to change. For technical use the term has to be defined more precisely in term of the mathematical model, but the same connotation applies.

§3.1.1. Mathematical Models

By mathematical model is meant the governing differential equations used for the simulation of a mechanical system. Generally we will deal with a semidiscrete model: discrete in space and continuous in time. In the time domain the model is given by ordinary differential equations (ODE) in time. For such models the following definition of stability, due to Dirichlet, has survived the test of time:

“The equilibrium of a mechanical system is stable if, in displacing the points of the system from their equilibrium positions by an infinitesimal amount and giving each one a small initial velocity, the displacements of different points of the system remain, throughout the course of the motion, contained within small prescribed limits.”

Although the definition is dynamic in nature, it addresses equilibrium solutions. Definitions beyond that point become more difficult. But most look only at steady state solutions. An exception is chaotic systems, as in turbulence models, but we will not look at that class of problems in this course.

One extension is looking at deterministic periodic solutions: say the Earth orbiting around the Sun (or vice-versa if you believe in Aristotle). The underlying idea is the same. Suppose that a system is undergoing periodic motion: \( u(t + T) = u(t) \). Perturb it by applying an arbitrary but tiny initial displacement or velocity. Study the subsequent motion. If the motion remains within small prescribed limits of the unperturbed motion “orbit” the steady state motion is stable, and unstable otherwise. The static steady-state case addressed by Dirichlet is of course included.

2 As it appears in his Appendix to the German translation of Lagrange’s *Mécanique Analytique* [52].
§3.1.2. Difference Equations
An integrator applied for temporal discretization produces a difference system. The foregoing definitions can be briefly recapitulated by looking at the sequence of computed solutions. Precise definitions are provided in texts cited in the Notes and Bibliography Section. Here we are interested in the stability of the time integrator. Informally, we want the time discretization
1. To preserve the stability of stable mathematical models, and
2. To manifest the instability of unstable mathematical models.
The first condition is classical. The second one is not. In the context of coupled systems, over-stability can be as damaging as instability. For example, suppose one tries to determine the flutter boundary of an aircraft by doing simulations at increasing speeds. If the time integrator is strongly dissipative, the onset of flutter may be masked when running at finite step sizes, leading to erroneous conclusions.

§3.1.3. Stability Analysis Methods
There are two general classes of stability analysis methods:
Amplification Methods. Also called von Neumann stability analysis. Based on decomposition of motion into normal modes, often using Fourier analysis, and superposition. The analysis looks at the growth or decay of perturbations from one step to the next, and can be implemented using standard linear algebra procedures. It is local in nature, but so is the concept of stability. A more severe restriction is that it strictly applies only to linear systems. Despite this limitation it is frequently applied to nonlinear systems through linearization.
Energy Methods. Also known, notably in control theory, as Lyapunov methods. These look at the variation of certain function (or functional) measures of the motion amplitude. Often these are related to energy measures of various kinds, hence the name. Energy methods are not restricted to linear systems, but require the construction of suitable measures, and this has to be done case by case.
Because the systems examined in these lectures are linear, the amplification method is used.

§3.1.4. Test Equations
In practice stability analysis is not performed on the discrete systems of actual applications but on test equations, which are highly simplified models of the real thing. The idea is similar to the patch test in space discretizations: time integrators that do not do well on the test equations can be discarded right away. Those that survive can be subjected to further tests. Of course the ultimate test comes on the actual applications, but by then the choices are hopefully narrowed.
The selection of test equations ranges from established procedures (for standard ODE integrators) to more of a black art (in coupled systems). In this Chapter a scalar test equation is taken as given, and the stability analysis done on it. This introduces the reader to commonly used time integrators.

§3.2. A First-Order ODE Test Equation
The most commonly used scalar test equation for first-order ODEs, and also the simplest, is
\[
\dot{y} = \lambda y. \tag{3.1}
\]
Here \( y = y(t) \) and \( \lambda \) is a coefficient independent of time, which is generally complex. To simplify the following study we will restrict its real part to be nonpositive: \( \Re[\lambda] \leq 0 \). If so the solutions \( y(t) = Ce^{\lambda t} \) of (3.1) are stable in the sense of being bounded for \( t > 0 \). We therefore expect the time integrator to preserve this attribute. Gear [28, p. 9] defines absolute stability on the test equation (3.1), treated by an integrator of stepsize \( h \), as follows:

“The region of absolute stability is that set of values of \( h \) (real nonnegative) and \( \lambda \) for which a perturbation in a single value \( y_n \) will produce a change in subsequent values which does not increase from step to step.”
This definition is relevant for $\Re[\lambda] \leq 0$ and will be used in the next two sections. It fails for $\Re[\lambda] > 0$, as discussed in §3.6.

§3.2.1. A One-Step Integrator

The general one-step integrator in the class of Linear Multistep Methods (LMS) is

$$y_{n+1} = y_n + h \left[ \beta \dot{y}_{n+1} + (1 - \beta) \dot{y}_n \right],$$  (3.2)

where $\beta$ is a coefficient in the range $[0, 1]$. Three well known members of this family are:

Forward Euler (FE) integrator (also known as explicit Euler): $\beta = 0$.

Backward Euler (BE) integrator (also known as implicit Euler): $\beta = 1$.

Trapezoidal Rule (TR) integrator: $\beta = \frac{1}{2}$.

To get a difference equation, state that (3.1) is verified at $t = t_n$: $\dot{y}_{n+1} = \lambda y_n$, and at $t = t_{n+1}$: $\dot{y}_{n+1} = \lambda y_{n}$. Replacing those derivatives into (3.2) yields

$$y_{n+1} = y_n + h \left[ \beta \lambda y_{n+1} + (1 - \beta) \lambda y_n \right].$$

Move all terms pertaining to the next time $t_{n+1}$ to the left hand side to get the difference system

$$(1 - \beta \lambda h)y_{n+1} = \left[ 1 + (1 - \beta) \lambda h \right] y_n.$$  (3.3)

Solve for $y_{n+1}$ to obtain the amplification form:

$$y_{n+1} = \frac{1 + (1 - \beta) \mu h}{1 - \beta \lambda h} y_n = \frac{1 + (1 - \beta) \mu}{1 - \beta \mu} y_n = z y_n, \quad \text{where} \quad \mu = \lambda h.$$  (3.4)

Here $z$ denotes the amplification factor. This is the amount by which the solution at one step is multiplied to get to the next step, hence the name. If $\lambda$ is complex, so is $z$. The criterion for stability is that its modulus does not exceed unity:

$$|z| = \left| \frac{1 + (1 - \beta) \mu}{1 - \beta \mu} \right| \leq 1.$$  (3.5)

Note that this is verified for stepsize $h = 0$ because if so $\mu = \lambda h = 0$ for any $\lambda$, and $z = 1$. So the question is: what happens for finite $h > 0$? There are three possibilities:

(C) If (3.5) is verified for $h \in [0, h_{\text{max}}]$, with $h_{\text{max}} > 0$, the integrator is called conditionally stable. In this case $h_{\text{max}}$ is the stability limit and $[0, h_{\text{max}}]$ the stability range.

(U) If (3.5) is violated for any stepsize $h > 0$, the integrator is called unconditionally unstable.

(A) If (3.5) is verified for all $h \geq 0$ (meaning that $h_{\text{max}} \to \infty$) the integrator is called $A$-stable. Obviously (A) is the most desirable objective, followed by (C), whereas (U) is unacceptable.

```
z=(1+(1-β)*μ)/(1-β*μ);
Plot[{z/.β->0,z/.β->1/2,z/.β->1},(μ,-4,0), Frame->True,
PlotStyle->{AbsoluteThickness[2],RGBColor[0,0,0]},
{AbsoluteThickness[2],RGBColor[1,0,0]},
{AbsoluteThickness[2],RGBColor[0,0,1]}];
```

Figure 3.1. Script to generate the plot of Figure 3.2.

§3.2.2. Real Negative $\lambda$

If $\lambda$ is real and negative, the amplification factor behavior for different $\beta$ can be simply obtained by the Mathematica script of Figure 3.1. The generated plot of $z$ versus $\mu = \lambda h \in [0, -4]$ for FE, BE and TR is shown in Figure 3.2. This is called an amplification plot. Clearly BE ($\beta = 1$) and TR ($\beta = \frac{1}{2}$) are in the stable region $|A| \leq 1$, and it is not difficult to show that this happens for all $h > 0$. On the other hand, FE clearly goes unstable for $h > -2/\lambda$. If fact, setting $\beta = 0$ in (3.5) gives $A = 1 + \mu$, which crosses the stability boundary $A = -1$ at $\mu = -2$ and goes to $-\infty$ as $\mu \to \infty$. 


§3.2.3. Complex $\lambda$

The complex $\lambda$ case is of particular interest when the test equation comes from the reduction of a second order equation with oscillatory solutions. An amplification plot such as that in Figure 3.2 can be done via a 3D display that shows $\Re(\mu)$ and $\Im(\mu)$ and $z$ in the $x$, $y$ and $z$ directions, respectively. This has the advantages of showing the magnitude of $z$ but can be difficult to visualize. The alternative is to stick with a 2D plot over the complex $\mu$ plane, with $\Re(\mu)$ and $\Im(\mu)$ along $x$ and $y$, respectively, and displaying only the stable and unstable regions. This is called a stability chart.

![Figure 3.2. Amplification factor $z$ for real $\lambda$, $\mu = \lambda h$, and three integrators.](image)

The Mathematica script shown in Figure 3.3 produces a stability chart for (3.5). The result is shown in Figure 3.4 for the FE ($\beta = 0$) integrator over the left part of the $\mu$ plane. The plot technique deserve some explanation. The function

$$\text{StabilityIndicator}[\mu, \beta] = \text{If}[\text{Abs}[(1+(1-\beta)\mu)/(1-\beta\mu)] <= 1, 1, 0];$$

ContourPlot[StabilityIndicator[\mu R+I*\mu I,0],{\mu R,-4,0},{\mu I,-2,2}, PlotPoints->200,ContourLines->False];

![Figure 3.3. Script to generate the plot of Figure 3.4.](image)

evaluates $z = [1+(1-\beta)\mu]/[1-\beta\mu]$, which is complex if $\mu$ is. It returns 1 if $|z| \leq 1$ and 0 otherwise. So given a $\{\mu, \beta\}$ pair, (3.6) gives 1 if stable, and 0 if unstable. When this discontinuous function is displayed as a contour plot in the complex $\mu$ plane with sufficiently high resolution (requested by saying PlotPoints->200), the only contour lines are very close to stability boundaries.

The results can be clearly observed in Figure 3.4, which shows the stability boundary to be the circle $|1+\mu| \leq 1$. The option ContourLines->False asks that contour lines not be shown to reduce clutter, making contrast between stable vs. unstable region colors sharper.

For real $\lambda$ the stable interval is $\mu = h\lambda = [0, -2]$ as previously found. For purely imaginary $\lambda$ there is no stable interval since the circle does not intersect the imaginary axis and FE becomes unconditionally unstable. Doing this plot for TR and BE gives no stability boundaries over the $\Re(\lambda) \in [-4, 0]$ range: the whole plotted region is stable. It is easily shown that both methods are A-stable for any complex $\lambda$ with $\Re(\lambda) < 0$.

§3.3. A First-Order ODE System

Suppose that instead of one equation we have the (admittedly contrived) homogeneous system of four first-order ODEs with constant coefficients:

$$\begin{bmatrix}
\dot{u}_1 \\
\dot{u}_2 \\
\dot{u}_3 \\
\dot{u}_4
\end{bmatrix} = - \begin{bmatrix}
85 & 51 & -75 & -45 \\
51 & 85 & -45 & -75 \\
-75 & -45 & 85 & 51 \\
-45 & -75 & 51 & 85
\end{bmatrix} \begin{bmatrix}
u_1 \\
u_2 \\
u_3 \\
u_4
\end{bmatrix}.$$  

or $\dot{u} = L u$.  

(3.7)

where the $u_i$ are functions of time.
§3.3.1. Reduction to Spectral Form
Computing the eigenvalues \( \lambda_i \) and eigenvectors \( \mathbf{v}_i \) of \( \mathbf{L} \) furnishes the spectral decomposition \( \mathbf{L} = \mathbf{V} \Lambda \mathbf{V}^T \) and \( \mathbf{V}^T \mathbf{L} \mathbf{V} = \Lambda \), in which

\[
\Lambda = \text{diag}[^{\lambda_i}]= -\begin{bmatrix}
1024 & 0 & 0 & 0 \\
0 & 256 & 0 & 0 \\
0 & 0 & 64 & 0 \\
0 & 0 & 0 & 4
\end{bmatrix}, \quad \mathbf{V} = \frac{1}{2} \begin{bmatrix}
1 & 1 & 1 & 1 \\
1 & 1 & 1 & -1 \\
-1 & -1 & 1 & 1 \\
-1 & -1 & 1 & -1
\end{bmatrix}.
\] (3.8)

Since \( \mathbf{L} \) is real symmetric, the eigenvalues are real and the matrix of normalized eigenvectors \( \mathbf{V} \) is orthogonal: \( \mathbf{V}^T \mathbf{V} = \mathbf{I} \). Make the change of variables:

\[
\begin{bmatrix}
\mathbf{u}_1 \\
\mathbf{u}_2 \\
\mathbf{u}_3 \\
\mathbf{u}_4
\end{bmatrix} = \frac{1}{2} \begin{bmatrix}
1 & 1 & -1 & -1 \\
1 & -1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & -1 & -1 & -1
\end{bmatrix} \begin{bmatrix}
\mathbf{y}_1 \\
\mathbf{y}_2 \\
\mathbf{y}_3 \\
\mathbf{y}_4
\end{bmatrix}, \quad \text{or} \quad \mathbf{u} = \mathbf{V} \mathbf{y}, \quad \mathbf{y} = \mathbf{V}^T \mathbf{u}, \quad \dot{\mathbf{y}} = \mathbf{V}^T \dot{\mathbf{u}}.
\] (3.9)

This change converts (3.7) to \( \mathbf{V} \dot{\mathbf{y}} = \mathbf{V} \Lambda \mathbf{y} \). Since \( \mathbf{V} \) is nonsingular, premultiplying by \( \mathbf{V}^T = \mathbf{V}^{-1} \) yields

\[
\dot{\mathbf{y}} = \Lambda \mathbf{y}, \quad \text{or} \quad \begin{bmatrix}
\dot{\mathbf{y}}_1 \\
\dot{\mathbf{y}}_2 \\
\dot{\mathbf{y}}_3 \\
\dot{\mathbf{y}}_4
\end{bmatrix} = -\begin{bmatrix}
1024 & 0 & 0 & 0 \\
0 & 256 & 0 & 0 \\
0 & 0 & 64 & 0 \\
0 & 0 & 0 & 4
\end{bmatrix} \begin{bmatrix}
\mathbf{y}_1 \\
\mathbf{y}_2 \\
\mathbf{y}_3 \\
\mathbf{y}_4
\end{bmatrix}.
\] (3.10)

This is called the spectral form of (3.7). Note that the four equations in \( \mathbf{y}_i \) uncouple. Thus they can be written as four scalar equations

\[
\dot{y}_1 = -1024 y_1, \quad \dot{y}_2 = -256 y_2, \quad \dot{y}_3 = -64 y_3, \quad \dot{y}_4 = -4 y_4.
\] (3.11)

These have the form of the model system (3.1) with \( \lambda \) set to the four eigenvalues \( \lambda_i \) of \( \mathbf{L} \) in turn.

§3.3.2. FE Stability Analysis
To fix the ideas suppose that (3.7) is treated by the Forward Euler (FE) integrator: \( \mathbf{u}_{n+1} = \mathbf{u}_n + h \dot{\mathbf{u}}_n \). Making the change of variables (3.9) shows that this is equivalent to treating (3.10) by the FE integrator in \( \mathbf{y} \), that is, \( \mathbf{y}_{n+1} = \mathbf{y}_n + h \dot{\mathbf{y}}_n \), which holds for each uncoupled equation. From the results of the last Section, for each of the uncoupled equations (3.11) the stable stepsize is bounded by \( |\lambda_i| \leq 2 \). The stable stepsize for the system is the minimum of the four:

\[
h_{\text{max}} = \min \left( \frac{2}{1024}, \frac{2}{256}, \frac{2}{64}, \frac{2}{4} \right) = \frac{2}{1024} = \frac{2}{\lambda_{\text{max}}}, \quad \text{where} \quad \lambda_{\text{max}} = \max_{i=1}^4 |\lambda_i| = 1024.
\] (3.12)

For the TR and BE integrators it is easy to verify that A-stability is retained.

§3.4. A Second Order ODE
ODEs with second or higher time derivatives can be reduced to a system of first order equations. As an example, consider the second-order ODE that governs the behavior of an undamped, unforced linear oscillator of circular frequency \( \omega \):

\[
\ddot{d} + \omega^2 d = 0.
\] (3.13)

where \( d = d(t) \) is the displacement from equilibrium and \( \omega \) is real nonnegative. This can be reduced to a first order system by introducing the velocity \( v = \dot{d} \) as auxiliary variable:

\[
\begin{bmatrix}
\dot{d} \\
v
\end{bmatrix} = \frac{1}{\omega^2} \begin{bmatrix}
0 & -1 \\
\omega^2 & 0
\end{bmatrix} \begin{bmatrix}
\dot{d} \\
v
\end{bmatrix}, \quad \text{or} \quad \ddot{\mathbf{u}} = -\mathbf{L} \mathbf{u} = \mathbf{0}, \quad \mathbf{u} = \begin{bmatrix}
\dot{d} \\
v
\end{bmatrix}, \quad \mathbf{L} = \begin{bmatrix}
0 & 1 \\
-\omega^2 & 0
\end{bmatrix}.
\] (3.14)
Suppose that this is treated by the FE integrator: \( u_{n+1} = u_n + h\dot{u}_n, \) or equivalently \( d_{n+1} = d_n + h\dot{d}_n \) and \( v_{n+1} = v_n + h\dot{v}_n, \) the resulting difference system is

\[
\begin{bmatrix}
1 & h \\
-h\omega^2 & 1
\end{bmatrix}
\]

(3.15)

where \( P, \) called the *amplification matrix*, is the generalization of the amplification factor found in (3.4). Since \( \det(P) = 1 + \omega^2 h^2, \) the eigenvalues of \( P \) are \( z_1 = 1 + h\omega i \) and \( z_2 = 1 - h\omega i, \) with \( i = \sqrt{-1}. \) The largest eigenvalue modulus is called the *spectral radius* and is denoted by \( \rho. \) Both eigenvalues have the same modulus; thus \( \rho = |z_1| = |z_2| = \sqrt{1 + h^2 \omega^2}. \) A necessary condition for stability is that the spectral radius do not exceed unity:

\[
\rho = |\sqrt{1 + h^2 \omega^2}| \leq 1, \quad \text{or} \quad 1 + h^2 \omega^2 \leq 1.
\]

(3.16)

This condition (3.16) is only satisfied for \( h = 0, \) and otherwise violated. Consequently the FE method is *unconditionally unstable* for the oscillator problem (3.13), a conclusion that generalizes to undamped mechanical systems. The result could also be discerned graphically from Figure 3.4, by moving along the imaginary axis because the eigenvalues of \( L \) are purely imaginary: \( \pm \omega i. \)


### §3.5. General ODE Systems

The rule illustrated by (3.12) can be generalized to a homogeneous first-order linear system with

\[
\dot{u} = Lu, \quad \text{or} \quad \dot{u} - Lu = 0,
\]

(3.17)

where \( L \) is a \( m \times m \) square matrix. For a linear system, the entries of \( L \) cannot be functions of the state \( u, \) but may depend on time.


#### §3.5.1. Constant Coefficients

If \( L \) is independent of \( t, \) the system (3.17) is said to have *constant coefficients*. To simplify matters, it is convenient to assume that \( L \) is diagonalizable with distinct nonzero eigenvalues \( \lambda_i \) through the similarity transformation

\[
V^{-1}LV = \text{diag}(\lambda_i) = \Lambda, \quad \text{or} \quad L = V\Lambda V^{-1}.
\]

(3.18)

Here \( \Lambda \) is the diagonal matrix of eigenvalues \( \lambda_i, \) \( V \) a matrix with the right eigenvectors of \( L \) as columns, and \( V^{-1} \) a matrix with the left eigenvectors as rows. Making the change of variables

\[
u = Vy, \quad y = V^{-1}u, \quad \dot{u} = V\dot{y}, \quad \dot{y} = V^{-1}\dot{u},
\]

(3.19)

reduces (3.17) to \( \dot{y} = V\Lambda y, \) which premultiplied by \( V^{-1} \) yields the uncoupled system \( \dot{y} = \Lambda y. \) This system is treated by an integrator that produces \( k \) uncoupled difference equations, the stability of which can be investigated by the amplification method. Suppose that \( \bar{h}_i \) is the maximum stable stepsize for the \( i^{th} \) equation. The stability limit for the system is the minimum of the stable stepsizes:

\[
h_{max} = \min_{i=1}^{k} \bar{h}_i.
\]

(3.20)

If \( L \) is singular or has a defective eigensystem, additional precautions are necessary. Those are discussed in several of the texts cited in the Notes and Bibliography Section.


#### §3.5.2. Variable Coefficients

If the entries of \( L \) depend on \( t, \) the foregoing technique can still be used by diagonalizing \( L \) at each time step. In this cases the stability analysis is *local*, that is, it represents the local behavior of the system to a first approximation.
§3.5.3. Forced First-Order Linear System
A generalization of (3.17) results on adding a forcing term
\[ \dot{u} - Lu = f(t) \]  
(3.21)

But it turns out that for stability analysis the forcing term \( f(t) \) is irrelevant and can be dropped. Consequently it is sufficient to consider the homogeneous forms analyzed before.

§3.5.4. Linearization of General First-Order System
A fairly general form\(^3\) of a system of first-order nonlinear differential equations is
\[ \dot{u} = g(u, t) \]  
(3.22)

where \( u \) is a vector of \( k \) components \( u_i(t) \) and \( g \) a vector of \( k \) functions \( g_i(u, t) \). To investigate the local behavior of (3.22) at \( u = u_0 \) and \( t = t_0 \) for deviations \( \Delta u = u - u_0 \) and \( \Delta t = t - t_0 \), linearize it by Taylor series
\[
\dot{u} \approx \frac{\partial g}{\partial u} \Delta u + \frac{\partial g}{\partial t} \Delta t = L \Delta u + q \Delta t, \quad L = \frac{\partial g}{\partial u} \bigg|_{u=u_0, t=t_0}, \quad q = \frac{\partial g}{\partial t} \bigg|_{u=u_0, t=t_0}. 
\]  
(3.23)

so matrix \( L \) appears as the Jacobian of \( g \) with respect to \( u \). The amplification analysis may be carried out on (3.23) as first approximation. The results, however, may be suspect for highly nonlinear systems. The energy method cited previously does not suffer from that drawback but has to be reworked for different problems.

§3.6. Overstability
Overstability means getting a bounded numerical solution for a differential equation or system that has unbounded solutions as \( t \) grows. As previously remarked, this can be dangerous in some situations such as flutter simulations, since physical instability is masked. The phenomenon can be easily illustrated for Backward Euler (BE) applied to the test equation \( \dot{y} = \lambda y \) with \( \Re(\lambda) > 0 \). The analytical solutions \( y(t) = Ce^{\lambda t} \) grow up exponentially. Applying BE: \( y_{n+1} = y_n + h\dot{y}_{n+1} \)
gives the amplification factor
\[
z = \frac{1}{1 - \mu}, \quad \mu = \lambda h. 
\]  
(3.24)

The BE stability chart can be generated by the script of 3.3, in which the second argument of StabilityIndicator is 1 whereas the plot range is \( \Re(\mu) \in [-3, 3] \) and \( \Im(\mu) \in [-3, 3] \) so as

---

\(^3\) This form assumes an explicit solution for \( u \). Most general form is implicit: \( F(\dot{u}, u, t) = 0 \). But there are few results and integrators available for the most general case.
Notes and Bibliography

The notion of stability as resistance to change is universal. Specific definitions vary with the field. For
dynamics and control, see [55,56]. For dynamical systems from a broader and more recent perspective, see
[83], which has a comprehensive reference list. For chaotic systems exhibiting “orbit shadowing” see [51].

There are many textbooks and monographs that focus on the numerical solution of ordinary differential
equations. Of the old (pre-1980) crop, by far the best organized and written is Gear’s textbook [28]. As can
be expected from its publication date it is outdated in some topics but still a gem. Lapidus and Seinfeld
[54] contains useful material for chemical engineering. Lambert’s textbook [53] has well chosen examples.
Henrici’s monograph [39] for scalar ODEs was groundbreaking in the 1960s, and the sequel [40] for systems
is still worth keeping as theoretical backup. One of the numerical analysis classics is Dalhquist and Björecker
[9], recently reprinted by Dover. This text contains a tutorial introduction to computational methods for
ODEs, skipping proofs.

Of the more recent crop, one may cite Butcher [7], Hairer and colleagues [36,37], Sewell [80] and
Shampine [81]. The most comprehensive and ambitious is the two-volume set [36,37]. Sample code for
solving ODEs is provided in the multi-language Numerical Recipes series; for example [74] in Fortran (a
dead language, but there is a companion book for C). The implementation quality, as typical of NR, ranges
from mediocre to laughable to dismal; however for one shot student projects it might be acceptable.

There are fewer books dealing with the energy method, as it is highly problem dependent. The book by
Straughan [82] focuses on convective fluid mechanics. Others can be found in the bibliography of [83]. For
the algebraic eigenproblem the “bible” is Wilkinson [91]. Nothing else comes close in comprehensiveness
and understanding, although as can be expected from a 1965 publication date several eigensolution methods
are outdated or missing.

4. STABILITY ANALYSIS TOOLS

This chapter discusses techniques for stability analysis of multilevel systems of difference equations. The key tools are stability polynomials in two flavors: Schur-Cohn and Routh-Hurwitz. The use of a computer algebra system to handle calculations with free parameters is emphasized.

§4.1. A Multistep Difference Scheme

To fix the ideas, suppose that discretization of a linear test equation with \( k \) state variables leads to
a system of difference equations written in the matrix form

\[
A_0 \mathbf{u}_{n+1} + A_1 \mathbf{u}_n + A_2 \mathbf{u}_{n-1} = \mathbf{f}_{n+1}.
\]  

(4.1)

The following notation is consistently used in the sequel:

- \( \mathbf{u}_{n+1}, \mathbf{u}_n, \mathbf{u}_{n-1} \): The state vector with \( k \) components, at time stations \( t = t_{n+1}, t_n \) and \( t_{n-1} \), respectively. Here \( t = t_n \) is the current time station, at which the solution is known, whereas \( t_{n+1} = t_n + h \) is the next time station. Previous state solutions, such as \( \mathbf{u}_{n-1} \) at time station \( t_{n-1} \), are part of the historical data. Historical terms such as \( A_2 \mathbf{u}_{n-1} \) arise if one uses a multistep integration scheme spanning more than one timestep, or a multistep predictor.
- \( A_0, A_1, A_2 \): A set of \( k \times k \) matrices whose entries depend on parameters of the test equation and on the time integration procedure. If state \( \mathbf{u}_{n-1} \) does not enter in the time discretization (4.1), \( A_2 = 0 \). Additional terms such as \( A_3 \mathbf{u}_{n-2} \) may appear if the time integrator uses more previous solutions.
- \( \mathbf{f}_{n+1} \): The discretized forcing function at \( t = t_{n+1} \). In linear systems this term does not depend on the state \( \mathbf{u} \).
§4.2. Polynomial Stability Conditions

§4.2.1. Amplification Polynomial

To convert (4.1) to an amplification polynomial, set $u_{n-1} = u_n/z = u_{n+1}/z^2$ and $u_n = u_{n+1}/z$. Here $z$ is a variable that will play the role of amplification factor.\(^4\) Factoring out $u_{n+1}$ gives

$$ (A_0 + z^{-1}A_1 + z^{-2}A_2) u_{n+1} = f_{n+1}. \quad (4.2) $$

For stability analysis the applied force term $f$ is set to zero. To avoid negative powers it is convenient to multiply both sides by $z^2$ to get

$$ (z^2A_0 + zA_1 + A_2)u_{n+1} = Pu_{n+1} = 0. \quad (4.3) $$

This equation has a nontrivial solution if the determinant of the matrix $P = z^2A_0 + zA_1 + A_2$ vanishes. Expanding the determinant as a polynomial in $z$ yields

$$ P_A(z) = \det P = \det(z^2A_0 + zA_1 + A_2) = a_0 + a_1z + a_2z^2 + \ldots + a_{n_A}z^{n_A}. \quad (4.4) $$

This $P_A$ receives the name amplification polynomial; it has order $n_A$. Denote the $n_A$ (generally complex) roots of $P_A(z) = 0$ by $z_i$, $i = 1, \ldots, n_A$. The polynomial $P_A$ is called stable if all roots lie on or inside the unit circle in the complex $z$ plane:

$$ |z_1| \leq 1, \ |z_2| \leq 1, \ldots \ |z_{n_A}| \leq 1. \quad (4.5) $$

This can be expressed compactly by introducing the spectral radius:

$$ \rho = \max_{i=1}^{n_A} |z_i| \Rightarrow \text{ } P_A \text{ stable if } \rho \leq 1. \quad (4.6) $$

This condition can be tested directly by the so-called Schur-Cohn stability criterion, which is discussed in detail in [50]. We shall not use this condition directly because symbolic computations with free parameters tend to be messy. Instead we transform $P_A(z)$ to a Hurwitz polynomial to apply a more traditional stability test.

§4.2.2. Hurwitz Polynomial

A polynomial $P_H(s)$ is called Hurwitz\(^5\) if the location of its roots in the left-hand plane $\Re(s) \leq 0$ determines stability. More precisely, a Hurwitz polynomial of order $n_H$ is called stable if all roots $s_i$ of $P_H(s) = 0$ lie in the negative complex $s$ plane:

$$ \Re(s_i) \leq 0, \quad i = 1, \ldots, n_H. \quad (4.7) $$

To pass from $P_A(z)$ to $P_H(s)$ or vice-versa one uses the conformal involutory transformations

$$ z = \frac{1+s}{1-s}, \quad s = \frac{z-1}{z+1}. \quad (4.8) $$

To pass from amplification polynomial to Hurwitz, replace $P_A(z)$ by $P_A((1+s)/(1-s))$. On expanding in $z$, this process in general produces the quotient of two polynomials: $P_H(s)/Q_H(s)$. Upon suppressing any common factors, the zeros of $P_H(s)/Q_H(s)$ are those of $P_H(s)$. The Hurwitz polynomial is obtained by taking the numerator of the fraction.

To pass from Hurwitz to amplification polynomial, replace $P_H(s)$ by $P_A((z-1)/(z+1))$. On expanding in $z$, this process in general produces the quotient of two polynomials: $P_A(z)/Q_A(z)$. Upon suppressing any common factors, the zeros of $P_A(z)/Q_A(z)$ are those of $P_A(s)$. The amplification polynomial is obtained by taking the numerator of the fraction.

\(^4\) This substitution will be connected to the so-called “discrete $z$ transform” in Chapter 8.

\(^5\) A nomenclature used by many authors. To be historically correct it should be called a Hermite-Routh-Hurwitz polynomial as discussed in Notes and Bibliography, but that is too long. Hurwitz is credited with closing the problem in 1895 [48].
Example 4.1. Suppose $P_A(z) = -1 - z + 2z^2 = 0$. This has roots $z_1 = 1$ and $z_2 = -1/2$ so $\rho = 1$ and $P_A$ is stable as per (4.6). To get the Hurwitz polynomial, transform using the first of (4.8):

$$P_A = -1 - z + 2z^2 \rightarrow -1 - \frac{1 + s}{1 - s} + 2 \frac{(1 + s)^2}{(1 - s)^2} = 3s + s^2 \rightarrow P_H(s) = 3s + s^2. \quad (4.9)$$

This has roots $s_1 = 0$ and $s_2 = -3$, so it is also stable as per (4.7). To go back to the amplification form, transform using the second of (4.8):

$$P_H = 3s + s^2 \rightarrow 3 \frac{z - 1}{z + 1} + \frac{(z - 1)^2}{(z + 1)^2} = -2 - 2z + 4z^2 \rightarrow P_A(z) = -2 - 2z + 4z^2. \quad (4.10)$$

This reproduces the original $P_A(z)$ except for a factor of 2.

Example 4.2. Take $P_A(z)$ as the most general fourth order polynomial: $P_A(z) = a_0 + a_1z + a_2z^2 + a_3z^3 + a_4z^4$.

Transform to a Hurwitz polynomial with the Mathematica module described below, with the invocation $b$=HurwitzPolynomialList[$a_0,a_1,a_2,a_3,a_4$],1,False]. This gives

$$P_H(s) = a_0 + a_1 + a_2 + a_3 + a_4 + (-4a_0 - 2a_1 + 2a_3 + 4a_4)s + (6a_0 - 2a_2 + 6a_4)s^2 + (-4a_0 + 2a_1 - 2a_3 + 4a_4)s^3 + (a_0 - a_1 - a_2 - a_3 + a_4)s^4. \quad (4.11)$$

Transforming back with AmplificationPolynomialList[b,1,False] gives $P_A(z) = 16(a_0 + a_1z + a_2z^2 + a_3z^3 + a_4z^4)$, which is the original polynomial except for a factor of 16.

§4.2.3. Amplification-to-Hurwitz Transform Module

The module HurwitzPolynomialList show in Figure 4.1 performs the $P_A(z) \rightarrow P_H(s)$ transformation automatically. It is particularly useful for polynomials of high order (over 4) and those with symbolic coefficients. The module is invoked as

$b$=HurwitzPolynomialList[$a$,$r$,$norm$] \quad (4.12)

The arguments are:

- $a$ A list of the coefficients $\{a_0,a_1, \ldots a_n\}$ of the amplification polynomial.
- $r$ Set to 1. (A non-unity value is used for root sensitivity studies, a topic not treated here.)
- $norm$ A normalization flag. Set to True to request that the coefficient of the highest power in $s$ of the Hurwitz polynomial be scaled to one. This may be an useful option in numerical work. It should not be used in symbolic work, as it may complicate subsequent processing.

The module returns:

- $b$ A list of the coefficients $\{b_0,b_1, \ldots b_k\}$ of the Hurwitz polynomial. The list may be shorter than $a$, that is, $k < n$, if one or more trailing coefficients vanish.

For example, to map the amplification polynomial $P_A(z) = (1-z)^4 = 1-4z+6z^2-4z^3+z^4$ to Hurwitz, say $b$=HurwitzPolynomialList[$1,-4,6,-4,1$],1,True]. This returns $\{0,0,0,0,1\}$ in $b$, that is, $P_H(s) = s^4$. This is easy to check: the four roots of $P_A(z) = 0$ are 1, which map into four zero roots of $P_H(s) = 0$. 
§ 4.2.4. Hurwitz-to-Amplification Transform Module

Sometimes it is necessary to pass from a Hurwitz polynomial to the corresponding amplification form. The module AmplificationPolynomialList listed in Figure 4.2 performs the $P_H(s) \rightarrow P_A(z)$ transformation automatically. The module is invoked as

$$a = \text{AmplificationPolynomialList}[b, r, \text{norm}] \quad (4.13)$$

The arguments are:
- \(b\): A list of the coefficients \(\{b_0, b_1, \ldots, b_n\}\) of the amplification polynomial.
- \(r\): Set to 1. (A non-unity value is used for root sensitivity studies, a topic not treated here.)
- \(\text{norm}\): A normalization flag. Set to True to request that the coefficient of the highest power in \(s\) of the amplification polynomial be scaled to one. This may occasionally be a useful option in numerical work. It should not be used in symbolic work, as it may complicate subsequent processing.

The module returns:
- \(a\): A list of the coefficients \(\{a_0, a_1, \ldots, a_k\}\) of the Hurwitz polynomial. The list may be shorter than \(b\), that is, \(k < n\), if one or more trailing coefficients vanish.

For example, the call \(a = \text{AmplificationPolynomialList}[\{0, 1, 3, 3, 1\}, 1, \text{False}]\) returns \(\{0, 0, 0, -1, 1\}\) in \(a\). This is the mapping of \(P_H(s) = s(s + 1)^3\) to \(P_A(z) = z^3(z - 1)\).

§ 4.3. The Routh-Hurwitz Criterion

Consider the Hurwitz polynomial of order \(n\) with real coefficients

$$P_H(s) = b_0 + b_1s + b_2s^2 + \ldots + b_ns^n, \quad \text{with} \quad b_i \text{ real, } b_0 \geq 0. \quad (4.14)$$

Note that \(b_0 \geq 0\) is assumed; if \(b_0 < 0\) the whole polynomial should be scaled by \(-1\), which does not change the zeros of \(P_H\).

§ 4.3.1. The Hurwitz Determinants

To determine if \(P_H(s)\) is stable according to (4.7), introduce the following determinants, known as Hurwitz determinants:

$$\Delta_i = \text{det} \begin{bmatrix} b_1 & b_3 & b_5 & \ldots & b_{2i-1} \\ b_0 & b_2 & b_4 & \ldots & b_{2i-2} \\ 0 & b_1 & b_3 & \ldots & b_{2i-3} \\ 0 & b_0 & b_2 & \ldots & b_{2i-4} \\ \vdots & \vdots & \vdots & \ddots & \ddots \\ \vdots & \vdots & \vdots & \ddots & b_i \end{bmatrix} \quad (4.15)$$

The indices in each row increase by two, whereas the indices in each column decrease by one. The term \(a_j\) is set to zero if \(j < 0\) or \(j > n\). Note that \(\Delta_1 = b_1\). The stability criterion was
enunciated by Hurwitz [48] as: A necessary and sufficient condition that the polynomial (4.14)

\[ \Delta_1(=b_1), \Delta_2, \ldots, \Delta_n, \]  

be all positive. Readable proofs may be found in Henrici [41], Jury [50] or Uspenky [85]. The
proofs of Gantmacher [27] and Hurwitz [48] are difficult to follow. Hurwitz made the following
observations. First, expanding \( \Delta_n \) by the last column, it is easily shown that \( \Delta_n = b_n \Delta_{n-1} \).
The condition that \( \Delta_{n-1} \) and \( \Delta_n \) be positive is equivalent to the requirement that \( \Delta_{n-1} \) and \( b_n \) be
positive. Thus the theorem remains valid if \( \Delta_n \) is replaced by \( b_n \). Second, \( \Delta_{n+1}, \Delta_{n+2}, \ldots \) vanish
identically since all entries of the last column vanish. The theorem can therefore be reformulated
as: all non-identically vanishing terms of the infinite sequence \( \Delta_1, \Delta_2, \ldots \) must be positive.

**Remark 4.1.** A necessary condition for (4.14) to be stable is that all coefficients \( b_0 \) through \( b_0 \) be positive.
The proof is quite simple: if the real part of all roots is negative, every linear factor of \( P_H(s) \) is of the form
\[ s + p < 0, \]  
and every quadratic factor is of the form \( z^2 + p_1 z + p_2 \) with \( p_1 > 0 \) and \( p_2 > 0 \). Since the product
of polynomials with positive coefficients likewise has positive coefficients, it follows that a stable
(4.14) can have only positive coefficients. Consequently finding a negative coefficient is sufficient to mark
the polynomial as unstable. But the condition is not sufficient for \( n > 2 \), as the following examples make clear.

**Example 4.3.** For the quadratic polynomial \( P_H(s) = b_0 + b_1 s + b_2 s^2 \), with \( b_0 > 0 \), the conditions given by
(4.16), with \( \Delta_2 \) replaced by \( b_2 \), are
\[ \Delta_1 = b_1 > 0, \quad b_2 > 0. \]  
These are satisfied if the three coefficients: \( b_0, b_1, b_2 \) are positive.

**Example 4.4.** For the cubic polynomial \( P_H(s) = b_0 + b_1 s + b_2 s^2 + b_3 s^3 \), with \( b_0 > 0 \), and \( \Delta_3 \) replaced by \( b_3 \), the conditions are
\[ b_1 > 0, \quad \Delta_2 = \det \begin{bmatrix} b_1 & b_3 \\ b_0 & b_2 \end{bmatrix} = b_1 b_2 - b_0 b_3 > 0, \quad b_3 > 0. \]  
These are satisfied if the coefficients: \( b_0, b_1, b_3 \) are positive, and \( b_1 b_2 > b_0 b_3 \).

**Example 4.5.** For the quartic polynomial \( P_H(s) = b_0 + b_1 s + b_2 s^2 + b_3 s^3 + b_4 s^4 \), with \( b_0 > 0 \), and \( \Delta_4 \)
 replaced by \( b_4 \), the conditions are
\[ b_1 > 0, \quad \Delta_2 = \det \begin{bmatrix} b_1 & b_3 \\ b_0 & b_2 \end{bmatrix} = b_1 b_2 - b_0 b_3 > 0, \quad \Delta_3 = \det \begin{bmatrix} b_1 & b_3 & 0 \\ b_0 & b_2 & b_4 \\ 0 & b_1 & b_3 \end{bmatrix} = b_1 b_2 b_3 - b_0 b_3^2 - b_1^2 b_4 > 0, \quad b_4 > 0. \]  
These are satisfied if the coefficients: \( b_0, b_1, b_4 \) are positive, \( b_1 b_2 > b_0 b_3 \), and \( b_1 b_2 b_3 > b_0 b_3^2 + b_1^2 b_4 \).

### §4.3.2. Routh-Hurwitz Criterion Modules

Figure 4.3 lists a *Mathematica* module that computes the Hurwitz determinant \( \Delta_k \). The module
is invoked as follows:
\[ \Delta_k = \text{HurwitzDeterminant}[b, k] \]  
The arguments are:

```
HurwitzDeterminant[b_, k_] := Module[{n = Length[b] - 1, i, j, m, A},
   If[k < 1 || k > n || n <= 0, Return[0]]; If[k == n, Return[b[[n + 1]]]];
   If[k == 1, Return[b[[2]]]]; A = Table[0, {k}, {k}];
   For[i = 1, i <= k, i++,
      For[j = 1, j <= k, j++, m = 2*j - i + 1;
        If[m > 0 && m <= n + 1, A[[i, j]] = b[[m]]]]];
   Return[Simplify[Det[A]]];
```
Another example of Henrici [41, p. 558] is: for which values of

Example 4.8

Example 4.7

Take the last coefficient used by Henrici as example [41, p. 557]. A call

c=RouthHurwitzStability[ ]; consequently the polynomial is stable.

HurwitzDeterminant returns the Hurwitz determinant $\Delta_k$. If $k$ is 1 or $n$, the module returns $b_1$ or $b_n$, respectively. If $k$ is outside the range 1 through $n$, it returns zero.

\[
\begin{align*}
\text{RouthHurwitzStability}[b_] := & \text{Module}[[n=\text{Length}[b]-1, i, c], \\
& n=\text{FindLastNonzero}[b]-1; \text{If } [n<3, \text{Return}[b]]; \\
& c=\text{Table}[0, \{n+1\}]; \ c[[1]]=b[[1]]; \\
& \text{For } [i=2, i<=n+1, i++, \ c[[i]]=\text{HurwitzDeterminant}[b, i-1]]; \\
& \text{Return}[\text{Simplify}[c]]];
\end{align*}
\]

Figure 4.4. Module that returns the Routh-Hurwitz stability conditions.

Module RouthHurwitzStability, listed in Figure 4.4, uses HurwitzDeterminant to return the list of Hurwitz determinants (4.16), with $\Delta_n$ replaced by $b_n$. It is invoked as

\[
c=RouthHurwitzStability[b] \tag{4.21}
\]

where the only argument is

- $b$ A list of the coefficients $\{b_0, b_1, \ldots, b_n\}$ of the Hurwitz polynomial $P_H(s) = b_0 + b_1 s + \ldots + b_n s^n$, with real coefficients and $b_0 > 0$. The module returns:

- $c$ A list of Hurwitz determinants arranged as $b_1, \Delta_2, \ldots, \Delta_{n-1}, b_n$.

**Example 4.6.** Investigate the stability of the polynomial $P_H(s) = 1 + 3s + 6s^2 + 12s^3 + 11s^4 + 11s^5 + 6s^6$ used by Henrici as example [41, p. 557]. A call $c=RouthHurwitzStability[[1,3,6,12,11,11,6]]$ returns $c=\{1,3,6,6,20,4,6\}$; consequently the polynomial is stable.

**Example 4.7.** Take the last coefficient of the foregoing polynomial as variable: $P_H(s) = 1 + 3s + 6s^2 + 12s^3 + 11s^4 + 11s^5 + gs^6$. For which values of $g$ is the polynomial stable? Now $c=RouthHurwitzStability[[1,3,6,12,11,11,g]]$ returns $c=\{1,3,6,6,-88+18g,-968+324g-27g^2, g\}$. A study of the sign of the last three entries shows that the polynomial is stable for $5.615099820540244 \leq g \leq 6.384900179459756$, which includes the value $g = 6$ used in the previous example.

**Example 4.8.** Another example of Henrici [41, p. 558] is: for which values of $\tau$ is $P_H(s) = 1 + 4\tau s + 6\tau s^2 + 4\tau s^3 + s^4$ stable? The call $c=RouthHurwitzStability[[1,4\tau,6\tau,4\tau,1]]$ returns $c=\{1,4\tau,4\tau(-1+6\tau),32\tau^2(-1+3\tau), 1\}$. A little study shows that $P_H(s)$ is stable if $\tau > 1/3$. 

<table>
<thead>
<tr>
<th>Order $n$</th>
<th>Positivity conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$b_0, b_1$</td>
</tr>
<tr>
<td>2</td>
<td>$b_0, b_1, b_2$</td>
</tr>
<tr>
<td>3</td>
<td>$b_0, b_1, \Delta_2, b_3$</td>
</tr>
<tr>
<td>4</td>
<td>$b_0, b_1, \Delta_3, b_3, b_4$</td>
</tr>
<tr>
<td>5</td>
<td>$b_0, b_1, \Delta_2, b_3, \Delta_4, b_5$</td>
</tr>
<tr>
<td>6</td>
<td>$b_0, b_1, \Delta_3, b_3, \Delta_5, b_5, b_6$</td>
</tr>
<tr>
<td>7</td>
<td>$b_0, b_1, \Delta_2, b_3, \Delta_4, b_5, \Delta_6, b_7$</td>
</tr>
<tr>
<td>8</td>
<td>$b_0, b_1, \Delta_3, b_3, \Delta_5, b_5, \Delta_7, b_7, b_8$</td>
</tr>
<tr>
<td>9</td>
<td>$b_0, b_1, \Delta_2, b_3, \Delta_4, b_5, \Delta_6, b_7, \Delta_8, b_9$</td>
</tr>
<tr>
<td>10</td>
<td>$b_0, b_1, \Delta_3, b_3, \Delta_5, b_5, \Delta_7, b_7, \Delta_9, b_9, b_{10}$</td>
</tr>
<tr>
<td>11</td>
<td>$b_0, b_1, \Delta_2, b_3, \Delta_4, b_5, \Delta_6, b_7, \Delta_8, b_9, \Delta_{10}, b_{11}$</td>
</tr>
<tr>
<td>12</td>
<td>$b_0, b_1, \Delta_3, b_3, \Delta_5, b_5, \Delta_7, b_7, \Delta_9, b_9, \Delta_{11}, b_{11}, b_{12}$</td>
</tr>
</tbody>
</table>
The only argument is

\[ b = \{b_0, b_1, \ldots, b_n\} \]

of the Hurwitz polynomial \( P_H(s) = b_0 + b_1s + \ldots + b_ns^n \), with real coefficients and \( b_0 > 0 \).

The module returns

\[ c = \text{LiénardChipartStability}[b] \]

\[
\text{(4.22)}
\]

Example 4.9. Given the sixth order Hurwitz polynomial \( P_H(s) = 1 + 3s + 6s^2 + 12s^3 + 11s^4 + 11s^5 + 6s^6 \), \( c = \text{LiénardChipartStability}[\{1,3,6,12,11,11,6\}] \) returns \( c = \{1,3,6,6,11,4,6\} \), confirming stability.

Example 4.10. Given the sixth order Hurwitz polynomial \( P_H(s) = 1 + 3s + 6s^2 + 12s^3 + 11s^4 + 11s^5 + gs^6 \), \( c = \text{LiénardChipartStability}[\{1,3,6,12,11,11,6\}] \) returns \( c = \{1,3,6,6,11,-968+324g-27g^2, g\} \). The study of the sign of the last three entries is simpler than in Example 4.7, and again shows that \( P_H(s) \) is stable for \( 5.615099820540244 \leq g \leq 6.384900179459756 \).

Example 4.11. Given the Hurwitz polynomial \( P_H(s) = 1 + 4\tau s + 6\tau s^2 + 4\tau s^3 + s^4 \) of Example 4.8, \( c = \text{LiénardChipartStability}[\{1,4\tau,6\tau,4\tau,1\}] \) returns \( c = \{1,4\tau,6\tau,32\tau^2(-1+3\tau), 1\} \). This immediately shows that \( P_H(s) \) is stable if \( \tau > 1/3 \).

Notes and Bibliography

The assessment of stability of mechanical control systems created by the nascent Industrial Revolution in England led, by the mid XIX Century, to the mathematical question of whether a given polynomial of order \( n \) and real coefficients has roots with negative real parts. Of course, what makes the problem interesting is to find a solution that can be expressed solely in terms of the coefficients, thus avoiding the explicit computations of the roots.

The question was raised by Maxwell in 1868 in connection with the design of steam engine regulating devices then known as governors. He was not aware that the problem had already been solved in 1856 by Hermite [42]. The applied mathematician Routh [77] gave an algorithm to find the number of roots \( n_U \) (the unstable roots) in the right-half plane, which for \( n_U = 0 \) solves Maxwell’s question. Independently of Routh in 1895 Hurwitz gave another solution in terms of determinants on the basis of Hermite’s paper. Modern proofs may be found in Henrici [41] or Uspenky [85]. The proof of Gantmacher [27] is unreadable.
Hurwitz’ criterion was further streamlined by Lienart and Chipart [57] in 1914. Despite its obvious benefits for high order symbolic polynomials the Liénard-Chipart criterion is described only in a few books. It is discussed in detail by Gantmacher [27, p. 220], but unfortunately the theorem statement is incorrect, and the purported proof is wrong. The criterion is briefly mentioned by Jury [50, p. 62]; however the sui generis notation used in this book makes the linkage to Routh-Hurwitz difficult to understand.

The related problem of requiring the roots of an amplification polynomial to be within the unit circle was solved separately in the early 1900s by Schur [79] and Cohn [8].

The general problem of finding whether the roots of a polynomial lie on a certain region of the complex plane, without actually finding the roots, lead to root clustering criteria. This is important in the design of filters and control devices of various kinds. The clustering problem is exhaustively studied in [50] for several regions. This book is hampered, however, by unfortunate notation and poor organization.

5. ACCURACY ANALYSIS TOOLS

This Chapter introduces the modified equation method as a tool for accuracy analysis of time discretizations for coupled systems. Because the method is relatively unknown as compared to truncation and forward error analysis, it is described in more detail than will normally the case in a lecture.

§5.1. Concept

Unlike stability, accuracy of ODE discretizations is not an easy topic. The effect of instability is highly visual; it jumps out like the robot in a Terminator movie. Assessing accuracy is not so obvious. There are two basic ways to go about the subject: forward or backward.

§5.1.1. Forward Error Analysis

Forward analysis relies on the global error, also called forward error. Given an ODE with unknown state variable $u(t)$ and exact solution $u_{\text{true}}$, the global error is defined as the difference

$$e_G = \|u_{\text{computed}} - u_{\text{true}}\|,$$  \hspace{1cm} (5.1)

where $\|$ is any norm appropriate to the occasion. Aside from the choice of norm the idea is easy to grasp. Unfortunately $u_{\text{true}}$ is unavailable aside from test problems or benchmarks. Consequently it is impossible to directly estimate or control (5.1) in general-purpose implementations.

To make progress, the easily understood but uncomputable $e_G$ is replaced by a more obscure but computable measure: the local error or truncation error $e_T$. This is the difference between the computed $u(t)$ over the interval $[t_n, t_n + h]$ and the locally exact solution of the ODE using the computed solution at $t_n$ as initial condition(s). If $e_T$ is Taylor-series expanded in the stepsize $h$ centered at $t = t_n$, and the first nonvanishing term is $h^{p+1}$, the order of the integration operator is said to be $p$. If $p \geq 1$ the operator is said to be consistent. The local error can be estimated by ODE solvers through well known techniques that usually involve doing some extra computations per step, and used to control $h$.

Though various behavioral assumptions on the differential system, it is possible to derive bounds on the global error $e_G$ from $e_T$. For a scalar equation integrated with average stepsize $h$, the typical form of those bounds is

$$e_G \leq C e^{Lt} h^p.$$  \hspace{1cm} (5.2)

where $L$ is the Lipschitz constant, $C$ a coefficient, and $p$ the order of the method. Over a finite time interval $t = [0, T]$ the error (5.2) goes to zero as $h \to 0$ if $L < \infty$ and $p \geq 1$ (i.e., a consistent integration operator). With some effort, estimates such as (5.2) can be extended to systems. This is the end of the forward analysis theory, which was finalized by the mid-1960s [40].

\hspace{1cm}

---

6 On p. 221 of [27] it is stated that the Liénard-Chipart criterion can be expressed in four equivalent forms, labeled 1 through 4. But form 1 is only valid for even $n$ and form 4 for odd $n$, as can be seen by comparing those statements with Table 4.1. Forms 2 and 3 are always wrong. Like most Russian books, Gantmacher shies away from numerical examples. Any such computation would have immediately displayed the mistake.
§5.1.2. Difficulties with Global Error

There is a wide class of application problems where the estimate (5.2) is practically useless:

Long Term Integration. If $L > 1$, (5.2) grows without bound as $t$ increases. This may result in huge overestimates. In fact it is easy to construct scalar ODE examples where the bound is pessimistic by any order of magnitude desired.

Periodic and Oscillatory Systems. If the integrator has a phase error, as virtually all do, the computed solution of oscillatory and periodic systems may “shadow” the true one. But since (5.1) only accounts for the error at a given $t$, it can be grossly pessimistic if the user is not interested in phase errors.

Conserving Invariants. In physical applications the user often is interested in conservation (or accurate prediction) of certain invariants of the response. For example: energy, linear momentum and angular momentum in conservative mechanical systems. But the global error bounds do not take into account such special requirements.

Forward error analysis in computational linear algebra, which was guilty of unrealistic predictions, disappeared by the late 1960 in favor of backward error, as narrated in Notes and Bibliography. Only recently has a similar process received attention for ODE computations.

§5.1.3. Backward Error Analysis

Backward error analysis takes the reverse approach to accuracy. Given the computed solution, it asks: which problem has the method actually solved? In other words, we seek an ODE which, if exactly solved, would reproduce the computed solution. This ODE is called the modified differential equation, often abbreviated to MoDE. The solution of the MoDE that passes through the computed points is called the reproducing solution. The difference between the modified equation and the original one provides an estimate of the error.

As noted, this approach is now routine in computational linear algebra. It agrees with common sense. Application problems involve physical parameters such as mass, damping, stiffness, conductivity, inductance, etc., which are known approximately. If the modified equation models a “nearby problem” with parameters still within the range of uncertainty, it is certainly as good as the original one. This “defect correction” can be used as the basis for controlling accuracy.

In view of the disadvantages noted for the forward error, why is backward analysis not the standard approach to ODE accuracy? Two reasons can be cited.

Intellectual Inertia. Many ODE solvers have been developed by numerical analysts without the foggiest idea of the practical use of simulations. Lacking application exposure they tend to stick to the ideas hammered in by their predecessors, and forward analysis has been the establishment way of assessing accuracy even since Euler invented piecewise linearization.

Technical Difficulties. In linear algebra the modified equation is another algebraic system, with the same configuration as the original one. But a MoDE constructed by applying Taylor series to a delay-difference (DD) equation has infinite order. (Nothing mysterious about this, since DD equations have an infinite number of characteristic roots.) As such, it has an infinite manifold of solutions through each initial condition point. The reproducing solution is included but so are infinitely many others. The maze is filtered by reduction to finite order. The reduction, however, is not an easy process to master and in fact constitutes the bulk of the effort.

§5.2. Linear Multistep Methods

One way to learn backward error analysis of ODEs is to go through examples for simple scalar equations. As the difference equations become more complex, hand processing rapidly becomes prohibitive. Help from a computer algebra system (CAS) is required. The following examples for Linear Multistep (LMS) methods were actually processed using the Mathematica modules described in §5.4.
§5.2.1. Forward Euler

Consider the solution of the scalar test equation \( u' = \lambda u, \ u = u(t), u(0) = u_0 \), by the Forward Euler method: \( u_{n+1} = u_n + h u'_n = u_n + h \lambda u_n \), in which a prime denotes derivative with respect to time.\(^7\) First, relabel \( t_n \) and \( t_{n+1} \) as generic times: \( t \) and \( t + h \), respectively:

\[
 u(t + h) - u(t) = h \lambda \ u(t). \tag{5.3}
\]

This is the delay-difference form of the modified equation\(^8\) and usually abbreviated to DDMoDE. It defines a new function \( u(t) \), assumed infinitely differentiable, which agrees with the computed FE solutions at time stations \( t_n, t_{n+1}, \ldots \). Next, expand the left-hand side in a Taylor series centered at \( t \) and solve for \( u'(t) \). This gives

\[
 u'(t) = \lambda u(t) - \frac{1}{2} h u''(t) - \frac{1}{6} h^2 u'''(t) - \frac{1}{24} h^3 u''''(t) - \ldots \tag{5.4}
\]

This is a modified ODE of infinite order, called IOMoDE for short. Because of the presence of an infinite number of derivatives, it does not provide insight. To make progress (5.4) is truncated to a derivative order \( m_d \) on the right, and then differentiated repeatedly \( m_d \) times with respect to \( t \) while discarding derivatives higher than \( m_d \). For \( m_d = 4 \) we get \( u'(t) = \lambda u(t) - \frac{1}{2} h u''(t) - \frac{1}{6} h^2 u'''(t) - \frac{1}{24} h^3 u''''(t), u''(t) = h \lambda u'(t) - \frac{1}{2} h u''(t) - \frac{1}{6} h^2 u'''(t), u'''(t) = h \lambda u''(t) - \frac{1}{2} h u'''(t), \) and \( u''''(t) = h \lambda u''''(t) \). Collect these as a matrix\(^9\) system, moving all derivatives of \( u \) to the left hand side:

\[
\begin{bmatrix}
1 & \frac{1}{2}h & \frac{1}{6}h^2 & \frac{1}{24}h^3 \\
-\lambda & 1 & \frac{1}{2}h & \frac{1}{6}h^2 \\
0 & -\lambda & 1 & \frac{1}{2}h \\
0 & 0 & -\lambda & 1
\end{bmatrix}
\begin{bmatrix}
u' \\ u'' \\ u'''
\end{bmatrix} =
\begin{bmatrix}
\lambda \\ 0 \\ 0 \\ 0
\end{bmatrix} u. \tag{5.5}
\]

Solve for the derivatives and expand in powers of \( h \lambda \):

\[
u' = \frac{4\lambda(6 + 6h\lambda + h^2\lambda^2)}{(2 + h\lambda)(12 + 12h\lambda + h^2\lambda^2)} u = (1 - \frac{1}{2}h\lambda + \frac{1}{3}h^2\lambda^2 - \frac{1}{4}h^3\lambda^3 + \ldots) \lambda u, \tag{5.6}
\]

Only the series for \( u' \) is of interest for what follows; while those for \( u'', u''' \) etc., might be of interest for other investigations. The coefficients of \( h \lambda \) in the \( u' \) series follow a recognizable pattern, which may be confirmed by retaining higher derivatives:

\[
u' = (1 - \frac{1}{2}h\lambda + \frac{1}{2}h^2\lambda^2 - \frac{1}{3}h^3\lambda^3 + \frac{1}{3}h^4\lambda^4 + \ldots) \lambda u = \frac{\log(1 + h \lambda)}{h} u. \tag{5.7}
\]

This is a modified equation of finite order, often abbreviated to FOMoDE. The general solution of this first-order equation is

\[
u(t) = C(1 + h \lambda)^{1/h}, \tag{5.8}
\]

which for the initial condition \( u(0) = u_0 \) yields \( C = u_0 \) and \( u(t) = u_0(1 + h \lambda)^{1/h} \). At time station \( t_n = nh, u_n(t) = u_0(1 + h \lambda)^n \), which coincides with the value computed by Forward Euler. As \( t/h \to \infty \) by making \( h \to 0 \) while keeping \( t \) fixed, (5.8) obviously tends to the solution \( u(t) = u_0 e^{\lambda t} \) of the original equation \( u' = \lambda u \). So for this case the backward analysis can be completed fully.

\(^7\) This change with respect to the usual superposed-dots (Newton’s notation) is to allow simpler notation for higher derivatives.

\(^8\) Note that it is not a difference equation, because \( u(t) \) in \( u(t + h) - u(t) = h \lambda u(t) \) is continuous (in fact infinitely differentiable) and hence defined between time stations.

\(^9\) The coefficient matrix of (5.5) is a Toeplitz matrix since all diagonals contain the same entry.
§5.2.2. Backward Euler

Next, \( u' = \lambda u, u = u(t), u(0) = u_0 \), is treated by the Backward Euler method: \( u_{n+1} = u_n + h\lambda u_{n+1} = u_n + h\lambda u_{n+1} \). The DDMoDE is \( u(t + h) - u(t) = h\lambda u(t + h) \). Expanding in Taylor series centered at \( t_n \) yields

\[
u'(t) + \frac{1}{2}h\nu''(t) + \frac{1}{6}h^2\nu'''(t) + \ldots = \lambda [u(t) + h\nu'(t) + \frac{1}{2}h^2\nu''(t) + \frac{1}{6}h^3\nu'''(t) + \ldots] \quad (5.9)
\]

Solve for \( u' \) to produce IOMoDE:

\[
(1 - h\lambda)u' = \lambda u(t) - \frac{1}{2}h\nu''(t) + \frac{1}{6}h^2(3\lambda\nu''(t) - u''(t)) + \frac{1}{24}h^3(4\lambda\nu''(t) - u'''(t)) + \ldots \quad (5.10)
\]

Differentiate this relation repeatedly with respect to \( t \) while keeping \( u \)-derivatives up to \( m_d \) order to set up a linear system. For \( m_d = 4 \) one gets

\[
\begin{bmatrix}
1 & \frac{1}{2}h & \frac{1}{6}h^2 & \frac{1}{24}h^3 \\
-(\lambda + h\lambda^2 + h^2\lambda^2) & 1 & \frac{1}{2}h & \frac{1}{6}h^2 \\
0 & -(\lambda + h\lambda^2) & 1 & \frac{1}{2}h \\
0 & 0 & -\lambda & 1
\end{bmatrix}
\begin{bmatrix}
u' \\
u'' \\
u''' \\
u''''
\end{bmatrix}
=
\begin{bmatrix}
\lambda + h\lambda^2 + h^2\lambda^3 + h^3\lambda^4 \\
0 \\
0 \\
0
\end{bmatrix}
\cdot u.
\]

Solving for \( u' \) and expanding in \( h\lambda \) gives

\[
u' = \frac{4\lambda(6 + 12h\lambda + 16h^2\lambda^2 + 17h^3\lambda^3 + 11h^4\lambda^4 + 5h^5\lambda^5 + h^6\lambda^6)}{24 + 36h\lambda + 38h^2\lambda^2 + 31h^3\lambda^3 + 16h^4\lambda^4 + 6h^5\lambda^5 + h^6\lambda^6} u
\]

\[
= (1 + \frac{1}{2}h\lambda + \frac{1}{6}h^2\lambda^2 + \frac{1}{4}h^3\lambda^3 + \ldots) \lambda u = -\frac{\log(1 - h\lambda)}{h} u = \frac{u}{h \log(1 - h\lambda)}.
\]

as FOMoDE. Its solution with \( u(0) = u_0 \) is \( u(t) = u_0/(1 - h\lambda)^{1/h} \). If \( t \) is kept fixed while \( h \to 0 \) this tends to \( u_0 e^{\lambda t} \), as expected.

§5.2.3. General One-Step Method

The test equation \( u' = \lambda u, u = u(t), u(0) = u_0 \), is treated by the general one-step LMS method: \( u_{n+1} - u_n = h\beta u_{n+1} + h(1 - \beta)u_n = h(\beta u_{n+1} + (1 - \beta)u_n) \), where \( \beta \) is a free parameter. The DDMoDE is \( u(t + h) - u(t) = h\lambda(\beta u(t + h) + (1 - \beta)u(t)) \). Expanding in Taylor series and solving for \( u' \) gives the infinite-order MoDE:

\[
(1 - h\lambda)u' = \lambda u(t) - \frac{1}{2}h\nu''(t) + \frac{1}{6}h^2(3\beta\lambda\nu''(t) - u''(t)) + \frac{1}{24}h^3(4\lambda\nu''(t) - u'''(t)) + \ldots \quad (5.13)
\]

Differentiating and retaining derivatives up to order 4:

\[
\begin{bmatrix}
\frac{1}{2}h & \frac{1}{6}h^2 & \frac{1}{24}h^3 \\
-(\lambda + \beta h\lambda + \beta^2 h^2\lambda^2) & 1 & \frac{1}{2}h & \frac{1}{6}h^2 \\
0 & -(\lambda + \beta h\lambda) & 1 & \frac{1}{2}h \\
0 & 0 & -\lambda & 1
\end{bmatrix}
\begin{bmatrix}
u' \\
u'' \\
u''' \\
u''''
\end{bmatrix}
= \begin{bmatrix}
1 + \beta h\lambda + \beta^2 h^2\lambda^2 + \beta^3 h^3\lambda^3 \\
0 \\
0 \\
0
\end{bmatrix} \lambda u.
\]

Solving for \( u' \) and expanding in series gives:

\[
u' = (1 + \frac{1}{2}h\lambda(2\beta - 1) + \frac{1}{3}h^2\lambda^2(1 - 3\beta + 3\beta^2) + \ldots) \lambda u = \frac{\log(1 + (1 - \beta)h\lambda)}{h \log(1 - \beta h\lambda)} u. \quad (5.15)
\]

This is the FOMoDE equation. It includes FE (\( \beta = 0 \)) and BE (\( \beta = 1 \)). The solution with \( u(0) = u_0 \) is \( u(t) = u_0[(1 + \frac{1}{2}h\lambda)/(1 - \frac{1}{2}h\lambda)]^{1/h} \to e^{\lambda t} \) for any \( \beta \) as \( h \to 0 \) for fixed \( t \). The principal error term \( \frac{1}{2}h\lambda(2\beta - 1) \) flags a first order method unless \( \beta = \frac{1}{2} \). With \( \beta = \frac{1}{2} \), which gives the Trapezoidal Rule (TR), the principal error is \( \frac{1}{12}h^2\lambda^2 \), and the method is of second order accuracy.
§5.2.4. Can Forward Euler be Made Exact?

Yes, but only for a scalar test equation such as \( \dot{y} = \lambda y \). The idea is to replace \( \lambda \) by \( \hat{\lambda} \), determined by solving

\[
\log(1 + \hat{\lambda}h) = \hat{\lambda}.
\]

(5.16)

and this \( \hat{\lambda} \) is used in the integration. This makes the modified equation (5.7) identical with the original one. For example, if \( \lambda = 20 \) and \( h = 1/10, \hat{\lambda} = 63.896 \); whereas if \( \lambda = -20, h = 1/10, \hat{\lambda} = -8.44665 \). Running Forward Euler with these \( \hat{\lambda} \)'s matches the exact solutions at the time stations. This is shown in Figure 5.1.

The trick has been discovered many times before. It is often called exponential fitting. One well known form originally due to Liniger and Willoughby is covered in Lambert’s textbook [53]. Liniger’s form keeps \( \lambda \) fixed and adjusts the method parameters, whereas the version (5.16) changes the exponent. This variant works for any \( \lambda \) and \( h \), even when \( h\lambda \) is in the unstable range, but is not applicable to ODE systems, which exhibit multiple characteristic values.

§5.3. Runge-Kutta Methods

§5.3.1. Explicit Runge Kutta

The classical four-step Runge-Kutta (CRK) for \( \dot{u} = \lambda u \) is \( k_1 = \lambda u_n; \ k_2 = \lambda (u_n + \frac{1}{2}hk_1); \ k_3 = \lambda (u_n + \frac{1}{2}hk_2); k_4 = \lambda (u_n + hk_3); u_{n+1} - u_n = (k_1 + 2k_2 + 2k_3 + k_4)/6 \). The DDMoDE is \( u(t + \Delta t) - u(t) = (k_1 + 2k_2 + 2k_3 + k_4)/6 \), \( k_1 = \lambda u(t), \ k_2 = \lambda (u(t) + \frac{1}{2}hk_1), \ k_3 = \lambda (u(t) + \frac{1}{2}hk_2), \ k_4 = \lambda (u(t) + hk_3) \). Expanding in Taylor series centered at \( u(t) \) and solving for \( \dot{u}(t) \) gives the IOMoDE

\[
\dot{u} = \lambda u + \frac{1}{2}h(\lambda^2 u - u''') + \frac{1}{6}h^2(\lambda^3 u - u''') + \frac{1}{24}h^3(\lambda^4 u - u''') + \ldots.
\]

(5.17)

Truncating at \( m_d = 4 \) and using repeated differentiation one constructs the linear system

\[
\begin{bmatrix}
1 & \frac{1}{2}h & \frac{1}{6}h^2 & \frac{1}{24}h^3 \\
-\lambda(1 + \frac{1}{2}h\lambda + \frac{1}{6}h^2\lambda^2) & 1 & \frac{1}{2}h & \frac{1}{24}h^3 \\
0 & -\lambda(1 + \frac{1}{2}h\lambda) & 1 & \frac{1}{2}h \\
0 & 0 & -\lambda & 1
\end{bmatrix}
\begin{bmatrix}
\dot{u}' \\
\dot{u}'' \\
\dot{u}''' \\
\dot{u}''''
\end{bmatrix}
= \begin{bmatrix}
1 + \frac{1}{2}h\lambda & \frac{1}{3}h^2\lambda^2 & \frac{1}{24}h^3\lambda^3 \\
0 & 0 & 0
\end{bmatrix} \lambda u,
\]

(5.18)

Solving for the derivatives \( \dot{u}' \) one simply gets \( \dot{u}' = \lambda u, \dot{u}'' = \lambda^2 u, \) etc. On raising \( m_d \) an identifiable series appears:

\[
\dot{u}' = (1 - \frac{1}{120}h^4\lambda^4 + \frac{1}{144}h^5\lambda^5 - \frac{1}{336}h^6\lambda^6 + \ldots)\lambda u = h^{-1} \log(1 + h\lambda + \frac{1}{2}h^2\lambda^2 + \frac{1}{6}h^3\lambda^3 + \frac{1}{24}h^4\lambda^4) u.
\]

(5.19)

which shows the method to be of fourth order.

For the simplest explicit RK, also known as Heun method: \( u_{n+1} - u_n = h\lambda(u_n + \frac{1}{2}h\lambda u_n) \), the DDMoDE is \( u(t + \Delta t) - u(t) = h\lambda(u(t) + \frac{1}{2}h\lambda u(t)) \). Proceeding as above one finds the FOMoDE

\[
\dot{u}' = (1 + \frac{1}{6}h^2\lambda^2 + \frac{1}{8}h^3\lambda^3 - \frac{1}{20}h^4\lambda^4 + \ldots)\lambda u = h^{-1} \log(1 + h\lambda + \frac{1}{2}h^2\lambda) u.
\]

(5.20)
§5.4. Mathematica Modules

The IMR is the simplest implicit Runge-Kutta (RK) method. Applied to \( u' = \lambda u \) it reads \( u_{n+1} - u_n = h u'_{n+1/2} = h \lambda u_{n+1/2} \), where \( u'_{n+1/2} \) and \( u_{n+1/2} \) denote the slope and solution, respectively, at \( t_n + \frac{1}{2}h \). DDMoDE form: \( u(t + h) - u(t) = \frac{1}{2} h \lambda u(t + \frac{1}{2}h) \). Expanding both sides in Taylor series and solving for \( u' \) gives IOMoDE as

\[
(1 - \frac{1}{2}\lambda h) u'(t) = \lambda u - \frac{1}{2}h u'' + \frac{1}{24} h^2 (3u'' - 4u''') + \ldots \tag{5.21}
\]

Differentiating, expanding in \( h \lambda \) and retaining derivatives up to order 4:

\[
\begin{bmatrix}
-\lambda - \frac{1}{2} h \lambda^2 - \frac{1}{4} h^2 \lambda^3 \\
\frac{1}{2} h + \frac{1}{8} h^2 \lambda + \frac{1}{16} h^3 \lambda^2 \\
0 \\
0
\end{bmatrix}
\begin{bmatrix}
\frac{1}{2} h + \frac{1}{8} h^2 \lambda + \frac{1}{16} h^3 \lambda^2 \\
\frac{1}{2} h + \frac{1}{8} h^2 \lambda \\
0 \\
0
\end{bmatrix}
\begin{bmatrix}
u' \\
u'' \\
u'''
\end{bmatrix}
= \begin{bmatrix}
\varphi \\
0 \\
0
\end{bmatrix}
\lambda u,
\]

in which \( \varphi = 1 + \frac{1}{2} h \lambda + \frac{1}{4} h^2 \lambda^2 + \frac{1}{8} h^3 \lambda^3 \). Solving for \( u' \):

\[
u' = \frac{96 + 48h \lambda + 52h^2 \lambda^2 + 12h^3 \lambda^3 + 7h^4 \lambda^4}{96 + 48h \lambda + 56h^2 \lambda^2 + 14h^3 \lambda^3 + 8h^4 \lambda^4} \lambda u = (1 - \frac{1}{24} h^2 \lambda^2 + \frac{3}{640} h^4 \lambda^4 - \ldots) \lambda u. \tag{5.23}\]

The series in parentheses is more difficult to identify. Retaining more terms reveals a recurrence relation that yields the FOMoDE equation

\[
u' = (1 - \frac{1}{24} h^2 \lambda^2 + \frac{3}{640} h^4 \lambda^4 - \frac{5}{1152} h^6 \lambda^6 + \ldots) \lambda u = h^{-1} \log \left[ 1 + h \lambda \sqrt{1 + \frac{1}{4} h^2 \lambda^2 + \frac{1}{2} h^2 \lambda^2} \right] u. \tag{5.24}\]

which is a new result. Note that IMR is a second order method, as is TR, but it is twice more accurate. For the 2-point Gauss-Legendre implicit Runge-Kutta \( u(t + h) - u(t) = \frac{1}{2} h \lambda [u(t + \gamma_1 h) + u(t + \gamma_2 h)] \), in which \( \gamma_1 = \frac{1}{2} - \sqrt{3}/6 \) and \( \gamma_2 = \frac{1}{2} + \sqrt{3}/6 \) are Gauss points. The FOMoDE is \( u' = (1 - \frac{1}{4320} h^4 \lambda^4 + \ldots) \lambda u \), showing fourth order accuracy. The series in parenthesis has not yet been identified.

§5.4. Mathematica Modules

The foregoing examples were actually carried out by two Mathematica modules. These are displayed in Figures 5.2 and 5.3. Module ModifiedScalarODEbySolve carries out the process through a direct solve method, whereas ModifiedScalarODEbyRecursion does it by recursive descent. Both forms receive the CD Form of the MoDE assumed to be in the canonical form

\[
u(t + h) - u(t) = hf(u, t) \tag{5.25}\]

§5.4.1. FOMoDE by Direct Solve

Module ModifiedScalarODEbySolve receives the delay form of the MoDE for a first-order scalar ODE. It returns the finite-order MoDE computed up to a given order. It should be used only for linear ODEs. For nonlinear ODEs it will attempt to compute solutions via Mathematica Solve function, but it is unlikely to succeed unless the order is very low and the equation of simple polynomial form. The module is invoked as follows:

\[
\text{MoDE=ModifiedScalarODEbySolve[lhs-rhs,u,t,h,h-tab,order,m,isol]};
\]

The arguments are:

- \( \text{lhs} \) The left hand side of the canonical form (5.25). Typically this is \( u[t+h]-u[t] \).
- \( \text{rhs} \) The right hand side of the canonical form (5.25).
- \( u \) The name of the unknown function.
ReplaceIncrement[u_, t_, h_, c_, order_] :=
    Sum[((c*h)^i/i!)*D[u[t], {t, i}], {i, 0, order}];

ModifiedScalarODEbySolve[res_, u_, t_, h_, htab_, order_, m_, isol_] :=
    Module[{r = res, d, v, usol, up, ups, vsol, umod, i, j, k, n, ns, nh, c},
        nh = Length[htab];
        If [nh == 0, r = r /. ReplaceIncrement[u, t, h, 1, order]];  
        If [nh > 0, For[i = 1, i <= nh, i++, c = htab[[i]];
            r = r /. ReplaceIncrement[u, t, h, c, order]];  
        r = Simplify[r];
        usol = Together[Simplify[Solve[r == 0, u'[t]]]];
        up = Simplify[u'[t] /. usol[[1]]];
        ups = Normal[Series[up, {h, 0, order - 1}]];
        d[[]] = ups;  
        For [i = 2, i <= order, i++,
            d[[i]] = Simplify[Normal[Series[D[d[[i - 1]], t], {h, 0, order - i}]]];
            v[[i]] = D[v[[i - 1]], t];
            eqs = {};
            For[i = 1, i <= order, i++, AppendTo[eqs, v[[i]] == d[[i]]]];  
            vsol = Solve[eqs, v];
            vsol = Simplify[vsol];
            ns = Length[vsol];
            umod = Table[0, {ns}];
            If [ns > 1, Print["***ModifiedScalarODEbySolve: ",
                ns, ", solutions found for nonlinear ODE"];]
            For [k = 1, k <= ns, k++, umod[[k]] = Take[v, m] /. vsol[[k]]];
            For[i = 1, i <= m, i++, umod[[k, i]] = Simplify[Normal[Series[umod[[k, i]], {h, 0, order - i}]]];
        ];
        If [isol > 0, Return[umod[[isol]]];
        Return[umod];
    ];

Figure 5.2. Module to calculate finite order MoDE of first-order scalar equation by direct solve.

ModifiedScalarODEbyRecursion[res_, u_, t_, h_, htab_, order_, m_, isol_] :=
    Module[{r = res, d, v, usol, up, i, j, k, klim, n, nh, c, ijtab = {}, dold, dnew},
        nh = Length[htab];
        klim = 0;
        If [nh == 0, r = r /. ReplaceIncrement[u, t, h, 1, order]];  
        If [nh > 0, For[i = 1, i <= nh, i++, c = htab[[i]];
            r = r /. ReplaceIncrement[u, t, h, c, order]];  
        r = Simplify[r];
        usol = Simplify[Solve[r == 0, u'[t]]];
        up = Simplify[u'[t] /. usol[[1]]];
        up = Normal[Series[up, {h, 0, order - 1}]];
        d = v = Table[0, {order}];
        d[[1]] = Simplify[up];
        v[[1]] = u'[t];
        For [i = 2, i <= order, i++,
            d[[i]] = Simplify[Normal[Series[D[d[[i - 1]], t], {h, 0, order - i}]]];
            v[[i]] = D[v[[i - 1]], t];
            For[i = order - 1, i >= 1, i--, For[j = order, j >= i, j--, AppendTo[ijtab, {i, j, 0}]]];
            For[i = 2, i <= m, i++, For[j = 1, j <= i, j++, AppendTo[ijtab, {i, j, 0}]]];
            For[j = 1, j <= Length[ijtab], j++, {i, j, kk} = ijtab[[j]]];
            rep = {v[[j]] -> d[[j]]};
            dold = Simplify[Normal[Series[d[[i]], {h, 0, order - i}]]];
            kmax = order + 1;
            For[k = 1, k <= kmax, k++,
                dold = Simplify[Normal[Series[D[d[[i]], t], {h, 0, order - i}]]];
                klim = Max[k, klim];  
                If [dnew == dold, Break[]];
                If [k > kmax, Print["Recursion OVW, j", j, " rep ", rep];  
                Return[Null];
                dold = Simplify[Normal[Series[dnew, {h, 0, order - i}]]];
            ];
        ];
        Return[Take[d, m]];
    ];

Figure 5.3. Module to calculate finite order MoDE of first-order scalar equation by recursive descent.
The name of the stepsize variable.

htab  A list of all multiples of $h$ different from one that appear in the canonical form (5.25). For example, in the Forward Euler: $u(t + h) - u(t) = h\lambda u(t)$, this list would be empty because only $u(t + h)$ appears. But in the implicit MR: $u(t + h) - u(t) = h\lambda u(t + \frac{1}{2}h)$ the multiple $\frac{1}{2}$ has to be listed and htab should be $\{1/2\}$. In the 2-point Gauss implicit Runge Kutta: $u(t + h) - u(t) = h\lambda \left[ u(t + \theta_1 h) + u(t + \theta_2 h) \right]$, where $\theta_1 = \frac{1}{2} - \frac{1}{6}\sqrt{3}$ and $\theta_2 = \frac{1}{2} + \frac{1}{6}\sqrt{3}$, htab should be $\{(3-\sqrt[3]{3})/6, (3+\sqrt[3]{3})/6\}$.

order  The order of the largest derivative of $u$ retained in the computations.

m  The number of FOMoDEs returned in the function: 1 through order. If $m=1$ only the MoDE for $u'$ is returned. If $m=2$ return FOMoDEs for $u'$ and $u''$, etc. Normally only that for $u'$ is of practical interest.

isol  If the input equation is nonlinear, ModifiedScalarODEbySolve will return $k>1$ FOMoDEs if it can compute any at all. If $isol>0$, return only the $isol^{th}$ solution; if $isol=0$ return all. If the input equation is linear, this is a dummy argument.

The function returns a list of FOMoDEs. If the input equation is linear, this list is one-dimensional and contains $m$ entries. For example if $m=3$, it returns the list \{ FOMode for $u'$, FOMode for $u''$, FOMode for $u'''$ \}.

If the input equation is nonlinear with $isol=0$ and if ModifiedScalarODEbySolve is able to compute $k$ solutions, the result will be a two-dimensional list dimensioned with $k$ rows and $m$ columns. Only one of the rows will be relevant in the sense of being a FOMoDE while the others are spurious. It is then possible to use $isol>0$ in subsequent invocations to weed out the output.

But since the module will rarely be able to solve nonlinear equations this case is largely irrelevant.

§5.4.2. FOMoDE by Recursion

There is a second way to obtain the FOMoDE, which involves using recursive substitution of derivatives rather than a direct solution. This method is implemented in the module ModifiedScalarODEbyRecursion, which is listed in Figure 5.3.

The arguments of this module are the same as those of ModifiedScalarODEbySolve, except that isol is a dummy argument. The module returns only one FOMoDE so isol is ignored. The logic of this module is quite hairy and only understandable to Mathematica experts so it wont be explained here.

Which module is preferable? For simple linear ODEs with low to moderate order, try both modules and compare answers. If everything goes well, the answers will be identical, which provides a valuable check. The recursion module, however, should be preferred in the following cases:

1. The ODE is nonlinear. In this case ModifiedScalarODEbySolve will typically fail unless the ODE is of polynomial type and the internal Solve is asked to solve a quadratic or cubic polynomial. This can only happen if order is very low, typically 1 or 2.

2. The ODE is linear but order is high enough that ModifiedScalarODEbySolve begins taking inordinate amounts of time. This is because the computational effort increases roughly as the cube of order when direct solving symbolic systems and simplifying. For the test systems used here and one-step LMS methods, this happens when order reaches 8–10. On the other hand, the computational effort growth of ModifiedScalarODEbyRecursion is milder, typically as the square of order. This is important when one is trying to identify series to express in closed form by taking more and more terms.

Notes and Bibliography

The study of errors in automatic matrix computations started during WWII. It was a byproduct of the appearance of programmable computers: punched card equipment, soon to be followed by digital computers. The paper that established the method as a science was co-authored by von Neumann and Goldstine [86]. This was soon followed by Turing [84]. All of these authors were pioneers of early digital computation.
The original (early 1940s) predictions of forward error analysis for the canonical problem of linear algebra: solving $Ax = b$ by elimination, were highly pessimistic. They showed worst case exponential error growth typified by $4^n$ or $2^n$, where $n$ is the number of equations. This lead early workers to predict that direct solvers will lose all accuracy by $n \approx 50$–100. But these dire predictions were not verified in actual computations. The discrepancy was explained by backward error analysis in a landmark paper by Wilkinson [89], who continued developing the approach throughout his career [90,91].

The concept of existence of a “nearby problem” as perturbation of the given data was actually first mentioned in [86]. Hence von Neumann deserves the credit of being the originator of backward error analysis, among other brilliant contributions “worth at least three and a half Nobel Prizes” [10]. His priority has been unfortunately belittled, however, by a “revisionist” view of early matrix computations [70].

The concept of forward-truncation error for ODE systems is much older since stepsize control was used since the late XIX Century for hand and slide-rule computations. The theory was brought to an elegant conclusion by Henrici [40].

Modified differential equations originally appeared in conjunction with finite difference discretizations for computational fluid dynamics (CFD). The prescription for constructing them can be found in Richtmyer and Morton’s textbook [75, p. 331]. They used it to interpret numerical dissipation and dispersion in the Lax-Wendroff CFD treatment of shocks, and to derive corrective operators. Similar ideas were used by Hirt [43] and Roache [76].

Warming and Hyett [88] were the first to describe the correct procedure for eliminating high order time derivatives. They used modified PDEs for studying accuracy and stability of several methods and attributed the “modified equation” name to Lomax [58]. In this work the original equation typically models flow effects of conduction and convection, $h$ includes grid dimensions in space and time, and feedback is used to adjust parameters in terms of improving stability as well as reducing spurious oscillations (e.g. by artificial viscosity or upwinding) and dispersion.

The first use of modified equations to study spatial finite-element discretizations can be found in [87]. However the derivative elimination and force lumping procedures were faulty, which led to incorrect conclusions. This was corrected by Park and Flaggs [63,64], who used modified equations for a systematic study of C⁰ beam, plate and shell FEM discretizations.

The method has recently attracted attention from the numerical mathematics community since it provides an effective tool to understand long-time structural behavior of computational dynamic systems, both deterministic and chaotic. Recommended references are [34,35,38,83]. Web accessible Maple scripts for the MoDE→FOMoDE reduction process are presented in [1].

6. STRUCTURE DYNAMICS TIME INTEGRATION

Partitioned analysis procedures that couple a structure to other fields (acoustics, control, fluids, thermal, etc.) can be derived from a second order differential system in time. Thus it is convenient at this point to introduce a general treatment of the governing equations of structural dynamics. Through a reduction to first order system, the treatment acquires enough power to be later extended to the staggered analysis of coupled systems that contain non-structural components.

§6.1. Governing Equations

The semi-discrete governing equations of linear structural dynamics are

$$\ddot{M}\ddot{u} + C\dot{u} + Ku = f,$$

where $M$, $C$ and $K$ are the mass, damping and stiffness matrices, respectively, $f = f(t)$ is the applied force and $u = u(t)$ the state-vector response. Unless the contrary is stated, the mass matrix will be assumed to be positive definite; consequently $M^{-1}$ exists.\(^\text{10}\)

§6.1.1. Reduction to First Order

This second order system can be reduced to first order through the auxiliary vector introduced by Jensen [49]:

$$v = AM\ddot{u} + Bu.$$

\(^\text{10}\) It will be seen that some formulations do not involve $M^{-1}$ and consequently can be used for a singular (or even zero) mass matrix. This is crucial for the treatment of coupled systems which do not possess an equivalent matrix in one or more fields.
Here $A$ and $B$ are arbitrary square matrices, with the proviso that $A$ be nonsingular. $A$ and $B$ are called the \textit{weighting matrices}. These matrices may be suitably chosen to reduce operation counts or to achieve other implementation goals. They are independent of time and state although in nonlinear problems they may be functions of the integration stepsize. Differentiating (6.2) yields

$$
\dot{v} = AM\dot{u} + B\dot{u} = A(f - Ku) + (B - AC)\dot{u}.
$$

(6.3)

Collecting (6.2) and (6.3) gives the first order system equivalent to (6.1) as

$$
\begin{bmatrix}
AM & 0 \\
AC - B & I
\end{bmatrix}
\begin{bmatrix}
\dot{u} \\
\dot{v}
\end{bmatrix}
+ 
\begin{bmatrix}
B & -I \\
AK & 0
\end{bmatrix}
\begin{bmatrix}
u \\
v
\end{bmatrix}
= 
\begin{bmatrix}
0 \\
Af
\end{bmatrix}.
$$

(6.4)

\section*{§6.1.2. Time Discretization by Linear Multistep Operators}

We consider the numerical integration of the first order system (6.4) through the use of $m$-step, one-derivative, linear multistep methods (LMS) difference operators to discretize the temporal variation of $u$ and $v$. A detailed algorithmic treatment of these methods can be found in the numerical analysis literature cited in \textit{Notes and Bibliography}. Suppose that the solution is known at time stations $t_{n}, \ldots, t_{1}$, with $n \geq m - 1$. For a constant stepsize $h$, LMS operators at time station $t_{n+1}$ can be written

$$
\begin{align*}
\mathbf{u}_{n+1} + \sum_{i=1}^{m} \alpha_{i}^{u} \mathbf{u}_{n-i+1} &= h \sum_{i=0}^{m} \beta_{i}^{u} \mathbf{u}_{n-i+1}, \\
\mathbf{v}_{n+1} + \sum_{i=1}^{m} \alpha_{i}^{v} \mathbf{v}_{n-i+1} &= h \sum_{i=0}^{m} \beta_{i}^{v} \mathbf{v}_{n-i+1}
\end{align*}
$$

(6.5)

Here $\{\alpha_{i}^{u}, \beta_{i}^{u}, \alpha_{i}^{v}, \beta_{i}^{v}\}$ are coefficients associated with specific finite difference operators, whereas $\mathbf{u}_{k}, \mathbf{v}_{k}, \ldots$ denote the vectors $u(t_{k}), v(t_{k}), \ldots$ computed through the discretization procedure. If $\beta_{0}^{v}$ is nonzero, the discretization operator for $u$ is said to be \textit{implicit} and \textit{explicit} otherwise. Similarly for the $v$ operator. If the operators have different number of steps: $m_{u}$ and $m_{v}$, we take $m = \max(m_{u}, m_{v})$ and pad the shorter operator with $m - m_{v}$ zero coefficients.\footnote{The notation (6.5) for the LMS operators agrees broadly with that used in the literature. Subscripting details, however, vary significantly across authors. Few publications consider the use of different LMS operators for $u$ and $v$.}

With a view to subsequent manipulations, it is convenient to introduce a more compact notation:

$$
\mathbf{u}_{n+1} = \delta_{u} \mathbf{u}_{n} + \mathbf{h}_{n}^{u}, \quad \mathbf{v}_{n+1} = \delta_{v} \mathbf{v}_{n} + \mathbf{h}_{n}^{v}.
$$

(6.6)

where $\delta_{u} = h \beta_{0}^{u}$ and $\delta_{v} = h \beta_{0}^{v}$ are called the \textit{generalized stepsizes}, and

$$
\mathbf{h}_{n}^{u} = h \sum_{i=1}^{m} \beta_{i}^{u} \mathbf{u}_{n-i+1}, \quad \mathbf{h}_{n}^{v} = h \sum_{i=1}^{m} \beta_{i}^{v} \mathbf{v}_{n-i+1}.
$$

(6.7)

will be called, following Jensen [49], the \textit{historical vectors} pertaining to $u$ and $v$, respectively.\footnote{The superscript notation adopted here is different from that used in the 1970–80s publications cited in the \textit{Notes and Bibliography} Section.}

The notation (6.6) separates the \textit{next-state variables} $\mathbf{u}_{n+1}, \mathbf{v}_{n+1}, \mathbf{u}_{n+1}, \mathbf{v}_{n+1}$ from historical terms that involve past solutions. Using this notation, the next solution can be expressed compactly as the difference scheme

$$
\begin{bmatrix}
\mathbf{u}_{n+1} \\
\mathbf{v}_{n+1}
\end{bmatrix}
= 
\begin{bmatrix}
\delta_{u} \mathbf{u}_{n} \\
\delta_{v} \mathbf{v}_{n}
\end{bmatrix}
+ 
\begin{bmatrix}
\mathbf{h}_{n}^{u} \\
\mathbf{h}_{n}^{v}
\end{bmatrix}.
$$

(6.8)
§6.1.3. Implicit Integration

We shall be primarily concerned with using a pair of implicit operators, for which both $\beta^u_0$ and $\beta^v_0$ are nonzero. If so $\delta_u = h\beta^u_0$ and $\delta_v = h\beta^v_0$ are also nonzero. Expressing (6.4) to be valid at $t_{n+1}$ and eliminating the time derivative vectors $\dot{u}_{n+1}$ and $\dot{v}_{n+1}$ from (6.8) produces the linear algebraic system

$$
\begin{bmatrix}
AM + \delta_u B & -\delta_u I \\
\delta_u (AC - B) + \delta_v \delta_u AK & \delta_v I
\end{bmatrix}
\begin{bmatrix}
\dot{u}_{n+1} \\
\dot{v}_{n+1}
\end{bmatrix}
= 
\begin{bmatrix}
\delta_v AM h^u_n \\
\delta_v (AC - B) h^u_n + \delta_v h^v_n + \delta_u \delta_v A f_{n+1}
\end{bmatrix}
$$

(6.9)

Elimination of $\dot{v}_{n+1}$ from (6.9) yields

$$Hu_{n+1} = g_{n+1}
$$

(6.10)

in which

$$H = M + \delta_v C + \delta_u \delta_v K + (\delta_u - \delta_v)A^{-1}B,
$$

$$g_{n+1} = [M + \delta_v (C - A^{-1}B)] h^u_n + \delta_v A^{-1} h^v_n + \delta_u \delta_v f_{n+1}.
$$

(6.11)

Note that the order (number of equations) of (6.10) is identical to that of the original ODE differential system (6.1). In practice it is common to take the same integration operator for $u$ and $v$. In that case we denote

$$\alpha_i = \alpha^u_i = \alpha^v_i, \quad \beta_i = \beta^u_i = \beta^v_i, \quad \delta = \delta_u = \delta_v = h \beta_0,
$$

(6.12)

and the components of the linear system (6.10) become

$$H = M + \delta_v C + \delta^2 K,
$$

$$g_{n+1} = [M + \delta_v (C - A^{-1}B)] h^u_n + \delta A^{-1} h^v_n + \delta^2 f_{n+1}.
$$

(6.13)

The coefficient matrix $H$ becomes independent of the choice of $A$ and $B$, as noted by Jensen [49].

§6.2. Implementation

Implementations of the integration algorithm outlined in the previous Section may differ in two respects:

- **Computational paths**: How the auxiliary vector $v$ and its derivative $\dot{v}$ are computed at each step.
- **Weighting matrices**: The selection of $A$ and $B$ in the definition (6.2) of $v$.

§6.2.1. Computational Paths

Three basic computational paths, labeled (0), (1) and (2), may be followed in advancing the discrete solution over a typical time step. The associated calculation sequences are flowcharted in Figure 6.1.

The path identification index (0,1,2) gives the number of backward difference operations performed in the determination of $v$ and $\dot{v}$ in each step. The articles [17,60] show that this index plays an important role in the computational error propagation of the associated computer implementations.

Path (0) is consistent with the first-order difference system (6.8); that is, the computed vectors satisfy this system exactly if computational errors due to limited precision are neglected. The original governing ODE system (6.1) is also satisfied. On the other hand, the computed $u_{n+1}$, $\dot{u}_{n+1}$ and $\dot{v}_{n+1}$ do not in general verify the auxiliary vector definition (6.2), which is satisfied only in the limit $h \to 0$.

A slight variant of path (0), labeled (0'), is also shown in Figure 6.1. In this variant the velocity vector $\dot{u}_n$ at the last time step is recomputed so that (6.2) is also verified at past stations. This path occurs naturally in the conventional choice of auxiliary vector $v = \dot{u}$ discussed in the next subsection. Path (1) satisfies both differential expressions (6.1) and (6.2) at the expense of (6.8).
Finally, path (2) enforces the auxiliary vector definition (6.2) and the difference system (6.8) at the expense of (6.1).

Actual implementations may “cyclically roll over” the order of the steps displayed in Figure 6.1. The main decision is whether \( v_n \) and \( \dot{v}_n \) should be computed at the start of the time stepping sequence, or \( v_{n+1} \) and \( \dot{v}_{n+1} \) computed at the end. If the latter approach is chosen, steps (a,b) would follow (h) with subscripts incremented by one.

### §6.2.2. Selection of Auxiliary Vector

The computational effort per step and, to a lesser extent the storage requirements, can be significantly reduced for certain choices of \( A \) and \( B \) because some of the computational steps displayed in Figure 6.1 can be either simplified or entirely bypassed. In [17,60] the following selections were studied for linear problems:

\[
(C): \quad v = \ddot{u}, \quad \text{by taking} \quad A = M^{-1}, \quad B = 0, \quad \quad (6.14)
\]

\[
(J): \quad v = M\ddot{u} + Cu, \quad \text{by taking} \quad A = I, \quad B = C.
\]

Selection (C) is called the *conventional choice* for \( v \), because that is the one invariably followed in publications that deal with reduction of second order systems to first order.

Selection (J) is called the *Jensen choice* on account of having been proposed in [49]. If \( C = 0 \), \( v = M\ddot{u} \) is the momentum vector, usually denoted as \( p \) in Mechanics; then \( \dot{p} = v = M\ddot{u} \) is the inertial force. If \( C \neq 0 \), \( v \) is a modified momentum vector whereas \( \dot{v} = f - Ku \) can be interpreted as the dynamic force resisted by inertia and damping effects.

When these two choices are combined with the four computational paths of Figure 6.1, a total of eight formulations of the advancing cycle results. The computational sequences associated with six of these: (C0’), (C1), (C2), (J0), (J0’) and (J2) are shown in Tables 6.1 and 6.2. Combinations (C0) and (J1) are omitted from these Tables as they have little value.

Other formulations of interest in the nonlinear case are obtained if \( A \) and \( B \) are allowed to depend on the generalized stepsizes \( \delta_u \) and \( \delta_v \). Two possibilities are discussed in [19]. One
particular selection, called the (K) choice, results by bringing K into the weighting matrix B as follows:

\[
\text{(K): } \mathbf{v} = \mathbf{M} \mathbf{\ddot{u}} + (\mathbf{C} + \eta h \mathbf{K}) \mathbf{u}.
\]  

(6.15)

Here \(\eta\) is an adjustable dimensionless coefficient which if zeroed reduces (K) to the choice (J) of auxiliary vector. Note that scaling \(\mathbf{K}\) by the stepsize \(h\) restores the correct physical dimension vis a vis the damping matrix \(\mathbf{C}\).

---

### Table 6.1. Computational Sequences Associated with the (C) Choice of \(v\)

<table>
<thead>
<tr>
<th>Form</th>
<th>Step</th>
<th>Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>(\ddot{u}_n = \mathbf{M}^{-1} (f_n - \mathbf{C} \dot{u}_n - K_n u_n))</td>
<td></td>
</tr>
<tr>
<td>(b,b')</td>
<td>(\mathbf{u}<em>n = \mathbf{h}^u</em>{n-1} + \delta_i \mathbf{\ddot{u}}_n)</td>
<td></td>
</tr>
<tr>
<td>(c)</td>
<td>(\mathbf{h}^u_n = h \sum_{i=1}^{i=m} \beta^u_i \mathbf{\ddot{u}}<em>{n-1} - \sum</em>{i=1}^{i=m} \alpha^u_i \mathbf{u}_{n-1} + 1)</td>
<td></td>
</tr>
<tr>
<td>(d)</td>
<td>(\mathbf{h}^u_n = h \sum_{i=1}^{i=m} \beta^u_i \mathbf{u}<em>{n-1} - \sum</em>{i=1}^{i=m} \alpha^u_i \mathbf{u}_{n-1})</td>
<td></td>
</tr>
<tr>
<td>(e)</td>
<td>(g_{n+1} = (\mathbf{M} + \delta_i \mathbf{C}) \mathbf{h}^u_n + \delta_i \mathbf{M} \mathbf{h}^\nu_n + \delta_i \delta_i \mathbf{f}_{n+1})</td>
<td></td>
</tr>
<tr>
<td>(f)</td>
<td>(\mathbf{H} = \mathbf{M} + \delta_i \mathbf{C} + \delta_i \delta_i \mathbf{K}) (if (h) changes)</td>
<td></td>
</tr>
<tr>
<td>(g)</td>
<td>(u_{n+1} = H^{-1} g_{n+1})</td>
<td></td>
</tr>
<tr>
<td>(h)</td>
<td>(\mathbf{u}<em>{n+1} = (\mathbf{u}</em>{n+1} - \mathbf{h}^u_n) / \delta_i)</td>
<td></td>
</tr>
</tbody>
</table>

---

### Table 6.2. Computational Sequences Associated with the (J) Choice of \(v\)

<table>
<thead>
<tr>
<th>Form</th>
<th>Step</th>
<th>Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>(v_n = f_n - \mathbf{K} u_n)</td>
<td></td>
</tr>
<tr>
<td>(b)</td>
<td>(v_n = \mathbf{h}^v_{n-1} + \delta_i \mathbf{v}_n)</td>
<td></td>
</tr>
<tr>
<td>(c)</td>
<td>(\mathbf{h}^v_n = h \sum_{i=1}^{i=m} \beta^v_i \mathbf{v}<em>{n-1} - \sum</em>{i=1}^{i=m} \alpha^v_i \mathbf{v}_{n-1})</td>
<td></td>
</tr>
<tr>
<td>(d)</td>
<td>(\mathbf{h}^v_n = h \sum_{i=1}^{i=m} \beta^v_i \mathbf{u}<em>{n-1} - \sum</em>{i=1}^{i=m} \alpha^v_i \mathbf{u}_{n-1})</td>
<td></td>
</tr>
<tr>
<td>(e)</td>
<td>(g_{n+1} = \mathbf{M} \mathbf{h}^v_n + \delta_i \mathbf{h}^v_n + \delta_i \delta_i \mathbf{f}_{n+1})</td>
<td></td>
</tr>
<tr>
<td>(f)</td>
<td>(\mathbf{H} = \mathbf{M} + \delta_i \mathbf{C} + \delta_i \delta_i \mathbf{K}) (if (h) changes)</td>
<td></td>
</tr>
<tr>
<td>(g)</td>
<td>(u_{n+1} = H^{-1} g_{n+1})</td>
<td></td>
</tr>
<tr>
<td>(h)</td>
<td>(u_{n+1} = (u_{n+1} - \mathbf{h}^v_n) / \delta_i)</td>
<td></td>
</tr>
</tbody>
</table>

---

Known as a vis the damping matrix \(\mathbf{C}\).
§6.2.3. Example: Trapezoidal Rule Integration

The formulation outlined in the preceding Section may be illustrated for the reader by considering a specific yet important integration formula. The Trapezoidal Rule (TR) operator will be used for both \( u \) and \( v \):

\[
\begin{align*}
\mathbf{u}_{n+1} &= \mathbf{u}_n + \frac{1}{2} h \left( \mathbf{u}_{n+1} + \mathbf{u}_n \right), \\
\mathbf{v}_{n+1} &= \mathbf{v}_n + \frac{1}{2} h \left( \mathbf{v}_{n+1} + \mathbf{v}_n \right).
\end{align*}
\tag{6.16}
\]

Comparing with the LMS operators (6.5) it is seen that for this choice \( m = 1 \), \( \alpha_1 = \alpha_1^u = -1 \), \( \beta_0 = \beta_0^u = \beta_1 = \beta_1^u = \beta_\omega = \frac{\delta}{\delta_0} = \delta_0 = \frac{1}{\tau} \). The historical vectors are

\[
\begin{align*}
\mathbf{h}_a^u &= \mathbf{u}_n + \frac{1}{2} h \mathbf{u}_n, \\
\mathbf{h}_b^u &= \mathbf{v}_n + \frac{1}{2} h \mathbf{v}_n.
\end{align*}
\tag{6.17}
\]

The components of the algebraic system are

\[
\mathbf{H} = \mathbf{M} + \frac{1}{2} h \mathbf{C} + \frac{1}{4} h^2 \mathbf{K}, \\
\mathbf{g}_{n+1} = \left[ \mathbf{M} + \frac{1}{2} h \left( \mathbf{C} - \mathbf{A}^{-1} \mathbf{B} \right) \right] \mathbf{h}_a^u + \frac{1}{4} h \mathbf{A}^{-1} \mathbf{h}_b^u + \frac{1}{4} h^2 \mathbf{f}_{n+1}.
\tag{6.18}
\]

The computational sequence for the (J0) formulation, in which \( \mathbf{A} = \mathbf{I} \) and \( \mathbf{B} = \mathbf{C} \), becomes

\[
\begin{align*}
(a) \quad & \mathbf{v}_n = \mathbf{f}_n - \mathbf{K} \mathbf{u}_{n-1}, \\
(b) \quad & \mathbf{v}_n = \mathbf{h}_a^u + \frac{1}{2} h \mathbf{v}_n, \\
(c) \quad & \mathbf{h}_b^u = \mathbf{v}_n + \frac{1}{2} h \mathbf{v}_n, \\
(d) \quad & \mathbf{h}_a^u = \mathbf{u}_n + \frac{1}{2} h \mathbf{u}_n, \\
(e) \quad & \mathbf{g}_{n+1} = \mathbf{M} \mathbf{h}_a^u + \frac{1}{2} h \mathbf{h}_b^u + \frac{1}{4} h^2 \mathbf{f}_{n+1}, \\
(f) \quad & \mathbf{H} = \mathbf{M} + \frac{1}{2} h \mathbf{C} + \frac{1}{4} h^2 \mathbf{K}, \\
(g) \quad & \mathbf{u}_{n+1} = \mathbf{H}^{-1} \mathbf{g}_{n+1}, \\
(h) \quad & \mathbf{u}_{n+1} = (\mathbf{u}_n - \mathbf{h}_a^u)/(\frac{1}{2} h), \\
\end{align*}
\tag{6.19}
\]

set \( n \leftarrow n + 1 \) and return to (a).

Note that the inverse of \( \mathbf{M} \) never appears in this formulation. This means that this matrix can be singular, or even vanish. As shown in 17 this favorable property is shared by (J0’), (J2) and (C1). On the other hand, (C0’), (C2) and (J1) do require the inverse \( \mathbf{M}^{-1} \).

The sequence (6.19) has to be complemented by a starter from the initial conditions \( \{ \mathbf{u}_0, \mathbf{\dot{u}}_0 \} \) at \( t = t_0 \).

Since the TR is a one-step method the starting procedure is straightforward:

\[
\begin{align*}
\mathbf{\dot{v}}_0 &= \mathbf{f}_0 - \mathbf{K} \mathbf{u}_0, \\
\mathbf{v}_0 &= \mathbf{M} \mathbf{\dot{u}}_0 + \mathbf{C} \mathbf{u}_0, \\
\mathbf{h}_b^u &= \mathbf{v}_0 + \frac{1}{2} h \mathbf{v}_0, \\
\mathbf{h}_a^u &= \mathbf{u}_0 + \frac{1}{2} h \mathbf{\dot{u}}_0.
\end{align*}
\tag{6.20}
\]

in which \( \mathbf{f}_0 \) is the force at \( t = 0^+ \). The same starting procedure is used for all (J) computational paths. For the (C) computational paths, the second equation in (6.20) is \( \mathbf{v}_0 = \mathbf{\dot{u}}_0 \), and the first one becomes the computation of either the initial acceleration \( \mathbf{\ddot{u}}_0 \) or the initial inertial force \( \mathbf{M} \mathbf{\ddot{u}}_0 \) in accordance with step (a) in Table 6.1.

§6.3. The Test System

The ODE test system for investigations of stability and accuracy is the unforced undamped linear oscillator

\[
\ddot{u} + \omega^2 u = 0.
\tag{6.21}
\]

The reduction to first order takes the form

\[
\begin{align*}
\dot{v} &= \dot{u} + \eta \omega^2 u, \\
\dot{u} &= \ddot{u} + \eta \omega^2 \dot{u} = -\omega^2 u + \eta \omega^2 \dot{u}.
\end{align*}
\tag{6.22}
\]

where the (K) form (6.15) is used. Setting \( \eta = 0 \) reduces the auxiliary vector to the (C) or (J) forms, which coalesce to \( v = \ddot{u} \) for the test system. Associated matrix relations are

\[
\begin{align*}
\begin{bmatrix} v \\ \dot{v} \end{bmatrix} &= \begin{bmatrix} \eta \omega^2 & 1 \\ -\omega^2 & \eta \omega^2 \end{bmatrix} \begin{bmatrix} u \\ \dot{u} \end{bmatrix}, \\
\begin{bmatrix} u \\ \dot{u} \end{bmatrix} &= \begin{bmatrix} 1 \\ \omega^2(\eta^2 \omega^2 - 1) \end{bmatrix} \begin{bmatrix} \eta \omega^2 \\ -\omega^2 \end{bmatrix} \begin{bmatrix} v \\ \dot{v} \end{bmatrix}, \\
\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} &= \begin{bmatrix} -\eta \omega^2 \\ -\omega^2(\eta^2 \omega^2 - 1) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}, \\
\begin{bmatrix} u \\ v \end{bmatrix} &= \begin{bmatrix} \eta \omega^2 \\ \eta \omega^2 \end{bmatrix} \begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix}.
\end{align*}
\tag{6.23}
\]

With \( y = [u \ v]^T \) the last two equations will be abbreviated to \( \dot{y} = \mathbf{Q} y \) and \( y = \mathbf{Q}^{-1} \dot{y} \), respectively. \( \mathbf{Q} \) is nonsingular if \( \omega \neq 0 \).
§6.3.1. Difference Equations

To generate difference equations, the $\dot{y} = Qy$ relation in (6.23) is evaluated at $t_{n+1}$: $\dot{y}_{n+1} = Qy_{n+1}$. The two rows of $Q$ are scaled by $\delta_u$ and $\delta_v$ to transform $\dot{u}$ and $\dot{v}$ to $\delta_u \dot{u}$ and $\delta_v \dot{v}$, respectively. Application of (6.6) to eliminate the derivatives then gives

$$
\begin{bmatrix}
1 + \delta_u \eta h \omega^2 \\
\delta_v h^2 (1 + \eta^2 \omega^2 h^2)
\end{bmatrix}
\begin{bmatrix}
\dot{u}_{n+1} \\
\dot{v}_{n+1}
\end{bmatrix}
= 
\begin{bmatrix}
h_n^{u} \\
h_n^{v}
\end{bmatrix}, 
\text{ or } A_0 y_{n+1} = h_n. \quad (6.24)
$$

The historical terms can be placed in matrix form as

$$
\begin{aligned}
&\begin{bmatrix}
h_n^{u} \\
h_n^{v}
\end{bmatrix} =
\sum_{i=1}^{m} \begin{bmatrix}
\beta_i^n & 0 \\
0 & \beta_i^v
\end{bmatrix}
\begin{bmatrix}
-\eta h \omega^2 \\
-\omega^2 (1 + \eta^2 \omega^2 h^2)
\end{bmatrix}
\begin{bmatrix}
\alpha_i^n & 0 \\
0 & \alpha_i^v
\end{bmatrix}
\begin{bmatrix}
\dot{u}_{n-i+1} \\
\dot{v}_{n-i+1}
\end{bmatrix}
-
\sum_{i=1}^{m} A_i y_{n-i+1}.
\end{aligned} \quad (6.25)
$$

On inserting (6.25) into (6.24), the discretized test equation can be compactly expressed as

$$
A_0 y_{n+1} + A_1 y_n + \ldots + A_m y_{n-m+1} = 0. \quad (6.26)
$$

where the $A_i$ matrices are defined in the foregoing two displays.

§6.3.2. Stability Analysis

The amplification polynomial $P(z)$ is obtained by assuming a solution of the form $y_{k+1} = z y_k$, where $z$ is an amplification factor. Introducing into (6.26) and cancelling superfluous $z$s one gets

$$
P_A(z) = \det(P), \quad \text{where } P = z^m A_0 + z^{m-1} A_1 + \ldots + A_m. \quad (6.27)
$$

Because each $A_i$ is a $2 \times 2$ real matrix, the amplification polynomial has order $2m$. The roots are either real or form complex conjugate pairs. The integrator is stable if the modulus of the $2m$ roots does not exceed one. Checking the roots directly becomes cumbersome as $m$ exceeds 1. If so the techniques explained in Chapter 4: reducing $P_A(z)$ to a Routh-Hurwitz polynomial $P_H(s)$ and testing the coefficients, are preferable.

Example 6.1. Analyze the A-stability of the general one-step method

$$
u_{n+1} = v_n + h \left[\beta_0^n \dot{u}_{n+1} + (1 - \beta_0^v) \dot{v}_n\right], \quad \text{for } \eta = 0. \quad (6.28)
$$

for $\eta = 0$. Here $\alpha_i^n = \alpha_i^v = -1$, $\beta_i^n = 1 - \beta_0^n$ and $\beta_i^v = 1 - \beta_0^v$. These conditions follow from ODE consistency. The two free parameters are $\beta_0^n$ and $\beta_0^v$. A Mathematica analysis gives

$$
P_A(z) = 1 + \omega^2 h^2 (\beta_0^n - 1) (\beta_0^v - 1) - [2 - \omega^2 h^2 (\beta_0^n + \beta_0^v - 2 \beta_0^n \beta_0^v)] z + [1 + \omega^2 h^2 (\beta_0^n - 1) (\beta_0^v - 1)] z^2. \quad (6.29)
$$

For $A$-stability the three coefficients of $P_H(s)$ must be nonnegative for any $\omega^2 h^2$. The first coefficient $b_0$ is always nonnegative. The condition $b_1 \geq 0$ requires $\beta_0^n + \beta_0^v \geq 1$. The condition $b_2 \geq 0$ becomes critical as $\omega h \rightarrow \infty$, which forces $(2 \beta_0^n - 1) (2 \beta_0^v - 1) \geq 0$. This can be satisfied in conjunction with the previous one, only if $\beta_0^n \geq \frac{1}{2}$ and $\beta_0^v \geq \frac{1}{2}$. If $\beta_0^n = \beta_0^v = \beta_0$ this reduces to the well known condition $\beta_0 \geq \frac{1}{2}$. 


Notes and Bibliography

The treatment of time integration methods for structural dynamics in the engineering literature usually focuses on the direct treatment of the second order equations by methods such as Newmark’s and its variants. As noted in the Appendix of [17], those methods can be expressed as particular instances of LMS methods for the first order form, as long as different integrators for \( u \) and \( v = \dot{u} \) are allowed. Thus they provide nothing new over the framework presented here. A drawback of sticking with this historical tradition is reduced freedom in method selection.

Reduction to first order in conjunction with stiffly stable methods was first advocated by Jensen [49]. This paper also studied choices for the auxiliary vector. This work was further extended by Felippa and Park in [17,60]. These articles introduced computational paths and studied their effect in implementation and error propagation. Extension to nonlinear structural analysis was discussed in [19]. The additional flexibility furnished by these choices has been fundamental in the analysis of staggered solution procedures derived through the algebraic partitioning method discussed in Chapter 7 [20,62,65].

7. STAGGERED SOLUTION PROCEDURES

§7.1. Partition Strategies

Two partition strategies may be used to construct staggered solution procedures.

Algebraic Partitioning. The full system is treated with an implicit integration operator as described previously. The resulting algebraic system is partitioned in accordance with a field separation. Finally, a predictor is applied to the state vector that appears on the right hand side.

Differential Partitioning. The matrices appearing in the differential system are partitioned. The right hand side is treated with a predictor. Finally, an integrator operator is applied to the predicted differential system.

The distinction between these two strategies can be more easily remembered through three letter acronyms paired with the formulation steps. Algebraic partitioning: integrate-partition-predict or IPP. Differential partitioning: partition-predict-integrate, or PPI.

Anticipating results, it is emphasized that the two strategies do not lead to the same results. However, for certain predictor choices they can be made to coalesce. Historically, differential partition was the original approach used in the derivation of staggered solution procedures for fluid-structure interaction, and remains a natural way to handle physically diverse systems. Algebraic partitioning was initially formulated for structure-structure and control-structure interaction to provide additional flexibility of implementation.

§7.2. Staggered Algebraic Partition

To keep the notation comparatively simple we assume that the same integration operator has been used on both \( u \) and \( v \). We therefore start from the algebraic system resulting from the discretization of \( M \ddot{u} + C \dot{u} + Ku = f(t) \), and split the coefficient matrix as

\[
H = H_1 + H_2,
\]

where

\[
H_1 = M + \delta C_1 = \delta^2 K_1, \quad H_2 = \delta C_2 + \delta^2 K_2.
\]

The term involving \( H_2 \) is then transferred to the right-hand side and a predictor is applied:

\[
H_1 u_{n+1} = g_{n+1} - H_2 u^p_{n+1}.
\]

To complete the specification of the time integration procedure, the configuration of the matrices \( C_1 \) through \( K_2 \) must be defined. For the case of two partitions, with field state vectors \( x \) and \( y \), a staggered solution procedure is obtained if we select

\[
C = C_1 + C_2 = \begin{bmatrix}
C_{xx} & 0 \\
C_{yx} & C_{yy}
\end{bmatrix} + \begin{bmatrix}
0 & C_{xy} \\
0 & 0
\end{bmatrix},
\]

\[
K = K_1 + K_2 = \begin{bmatrix}
K_{xx} & 0 \\
K_{yx} & K_{yy}
\end{bmatrix} + \begin{bmatrix}
0 & K_{xy} \\
0 & 0
\end{bmatrix}.
\]
Note that the mass matrix $M$ is not partitioned.

It must be stressed, however, that the staggered partition is only one a myriad of possible partitions. This number increases rapidly. For three fields the number of possible matrix partitions rises to 120, and to 2520 for four fields [20].

§7.2.1. Implementation

Having decided upon a staggered solution procedure, it remains to work out the implementation details. By this it is meant the selection of auxiliary vectors, computational paths, and predictor formula. The latter two can be be crucial to the success of the solution method, whereas the former affects computational efficiency and ease of programming of tasks such as starting and stepsize changing procedures.

As regards the choice of auxiliary vector and computational paths, the discussion of Chapter 6 applies with only formal changes. These changes pertain to the mechanics of going through calculation sequences such as those illustrated in that Chapter, and breaking down the computations in accordance with the partitions (7.4).

To fix the ideas, Figure 7.1 details the calculation sequence associated with the (J0) formulation. Recall that this identification corresponds to pairing the computational path (0) with the (J) choice of auxiliary vector $v$. Since $u$ is partitioned into $x$ and $y$, $v$ is partitioned into two subvectors $p$ and $q$ as per the definition

$$v = \begin{bmatrix} p \\ q \end{bmatrix} = \dot{u} + Cu.$$  \hfill (7.5)

In partitioned form this becomes

$$p = M_{xx} \dot{x} + C_{xx} x + C_{xy} y,$$

$$q = M_{yy} \dot{y} + C_{yx} x + C_{yy} y.$$  \hfill (7.6)

In Figure 7.1, the computations pertaining to the $x$ and $y$ fields are placed on the right and left columns, respectively. Steps associated with single field (decoupled field) analysis calculations are enclosed in solid line boxes. Steps required on account of field coupling are displayed in dashed boxes. Thus Figure 7.1 helps to answer the question; given a specific partition and computational path, what extension to the single-field analysis programs are necessary? In this respect, Figure 7.1 is more detailed than the largely conceptual advancing diagrams presented in Chapter 2.

For formulations other than (J0) only a few computational steps in the upper half of Figure 7.1 have to be modified. The reader is referred to Tables 6.1 and 6.2 for details.

Note that the question: which predictor? has not been answered. Decisions in this respect must wait until the next Chapter. Therein we shall see that the selection of computational path and predictor are strongly linked.

§7.2.2. Corrective Iteration

A corrective iteration cycle may be implemented in Figure 7.1 (and also in Figure 7.2) by re-entering the “predict box” on the right column, and replacing $y_{n+1}$ with the last computed $y_{n+1}$. This is called a corrective iteration, which can be obviously extended to more cycles. This may be useful, for instance, if the $y$ field contains “hard” nonlinearities, for example in free-boundary, moving interface or contact problems.

For linear and midly nonlinear problems, it has been our experience than cutting the time step is more effective than iterating. There is a notable exception, however. If the response of the coupled mechanical system is dominated by rigid body motions, a staggered partition treatment may induce significant accuracy degradation on $x$-field components near the partition boundary [62]. In such case, a one-cycle iteration may result in significant accuracy improvement at modest cost. Another way out is to use three-field partitions, in which the boundary between $x$ and $y$ is treated as a separate field. Variants of the three-field partitions, such as the element-by-element variant, preserve exactness of rigid body motions [62,65] but demand a more complex implementation than staggered procedures.
7.3 Staggered Differential Partition

In this approach we start from the governing differential equations. Introduce the splittings $\mathbf{C} = \mathbf{C}_1 + \mathbf{C}_2$, $\mathbf{K} = \mathbf{K}_1 + \mathbf{K}_2$ of the damping and stiffness matrices, and transfer the terms involving $\mathbf{C}_2$ and $\mathbf{K}_2$ to the right hand side:

$$\ddot{\mathbf{u}} + \mathbf{C}_1 \dot{\mathbf{u}} + \mathbf{K}_1 \mathbf{u} = \mathbf{f} - \mathbf{C}_2 \dot{\mathbf{u}} - \mathbf{K}_2 \mathbf{u}. \quad (7.7)$$

The next step is to apply a predictor at $t = t_{n+1}$:

$$\ddot{\mathbf{u}}_{n+1} + \mathbf{C}_1 \dot{\mathbf{u}}_{n+1} + \mathbf{K}_1 \mathbf{u}_{n+1} = \mathbf{f}_{n+1} - \mathbf{C}_2 \dot{\mathbf{u}}_{n+1}^p - \mathbf{K}_2 \mathbf{u}_{n+1}^p. \quad (7.8)$$

The right hand side of (7.8) can be viewed as a pseudo- or effective-force vector $\hat{\mathbf{f}}_{n+1}$:

$$\ddot{\mathbf{u}}_{n+1} + \mathbf{C}_1 \dot{\mathbf{u}}_{n+1} + \mathbf{K}_1 \mathbf{u}_{n+1} = \hat{\mathbf{f}}_{n+1}. \quad (7.9)$$

Finally, (7.10) is treated with a LMS integrator. This step produces the difference system

$$\mathbf{H}_1 \mathbf{u}_{n+1} = \mathbf{g}_{n+1}, \quad (7.10)$$

where, for the case of a common integration formula and $v = \mathbf{AM}\ddot{\mathbf{u}} + \mathbf{Bu}$,

$$\mathbf{H}_1 = \mathbf{M} + \delta \mathbf{C}_1 + \delta^2 \mathbf{K}_1, \quad (7.11)$$

$$\mathbf{g}_{n+1} = \mathbf{M} - \delta (\mathbf{C}_1 - \mathbf{A}^{-1} \mathbf{B}) h_n^\alpha + \mathbf{A}^{-1} h_n^\beta + \delta^2 \hat{\mathbf{f}}_{n+1}. \quad (7.11)$$

7.3.1 Implementation

As in the case of the staggered algebraic partition, the implementation of the time advancing step is completed through the selection of auxiliary vector, computational path and predictors. Figure 7.2 illustrates the calculation sequence involved in the programming of the (J0) formulation. This results from combining path (0) with the auxiliary vector selection

$$\mathbf{v} = \begin{bmatrix} p \\ q \end{bmatrix} = \mathbf{M} \ddot{\mathbf{u}} + \mathbf{C}_1 \mathbf{u}. \quad (7.12)$$
Further analysis of both partition types may be found in [20, 61, 62, 65, 66].

partitions emerged later when other applications such as structure-structure interaction were considered.

outlined in the Notes and Bibliography which in partitioned form is

\[
\mathbf{p} = \mathbf{M}_{xx} \dot{x} + \mathbf{C}_{xx} x \\
\mathbf{q} = \mathbf{M}_{yy} \dot{y} + \mathbf{C}_{yy} x + \mathbf{C}_{yy} y.
\]  

(7.13)

Note that (7.13) is not generally equivalent to the auxiliary vector definition for the algebraic partition unless the coupled damping terms \(C_{xy}\) and \(C_{yx}\) vanish.

The calculation sequence in Figure 7.2 displays more “anisotropy” in the treatment of the fields than that of Figure 7.1. This results primarily from the pseudo-force treatment; that is, embodying predicted terms in \(\hat{\mathbf{F}}\), which is later reused in the calculation of \(\dot{\mathbf{p}}\). Another distinctive feature is the need for two predictors, one for \(y_{n+1}\) and one for \(y_{P+1}\), whereas in the algebraic staggered procedure only one predictor is necessary. However for many application problems only one of the terms occur.

Notes and Bibliography

The appearance of staggered solution procedures in what is now called a differential partition form, is outlined in the Notes and Bibliography of Chapter 2. The distinction between algebraic and differential partitions emerged later when other applications such as structure-structure interaction were considered. Further analysis of both partition types may be found in [20, 61, 62, 65, 66].

8. OPERATIONAL ANALYSIS OF TIME INTEGRATORS

This Chapter derives operational forms of time-discretized second order governing equations, in preparation for the derivation of characteristic stability equations in Chapter 9. Those equations will be able to encompass both algebraic and differential partition schemes.

Emphasis is placed on Linear Multistep (LMS) integrators since they include a large number of integration formulas used in practice. Both the functions and their derivatives are evaluated at the same times. Formulas where function and/or derivative evaluation occurs at fractional steps, for example \(t_{n+1/2}\), belong to a wider class called multivalue methods, which are not treated here.
§8.1. Linear Multistep Methods

Linear Multistep (LMS) methods were introduced in Chapter 6 as covering a large number of integration schemes used in practice. A \(m\)-step LMS formula uses a linear combination of value of the dependent variable and its first derivatives at \(m + 1\) different time stations. We recall their general expression:

\[
 w_{n+1} + \sum_{i=1}^{m} \alpha_i w_{n-i+1} = h \sum_{i=0}^{m} \beta_i w_{n-i+1}, \quad m \geq 1. \tag{8.1}
\]

Here \(w\) is a generic vector that generally stands for a component of a state vector such as \(u\) or \(v\). As previously noted the use of different integrators for different components is allowed and in fact fits well the philosophy of partitioned analysis. We note that (8.1) can be viewed as a recursive digital filter that maps an input sequence \([\dot{w}_0, w_1, \ldots]\) onto an output sequence \([w_0, w_1, \ldots]\). This viewpoint is occasionally useful when interpreting properties of numerical algorithms in terms of control theory concepts \([59,65]\). That interpretation was extremely useful in developing the concept of augmentation, which is not treated here.

§8.1.1. Shifted Operational Form

All sample function terms that enter in (8.1) can be related to \(w_{n+1}\) and \(\dot{w}_{n+1}\) through the shift operator: \(S_i w_{n-i+1} \text{ def } w_{n+1}\) and \(S_i \dot{w}_{n-i+1} \text{ def } \dot{w}_{n+1}\). Substituting, taking the Laplace transform (with transformed variable \(s\)) and using the shift theorem yields

\[
 w_{n+1} \left(1 + \sum_{i=1}^{m} \alpha_i e^{-ish}\right) = \dot{w}_{n+1} \sum_{i=0}^{m} \beta_i e^{-ish}. \tag{8.2}
\]

Assuming \(\beta_0 \neq 0\) this can be placed in the compact operational form

\[
 \rho(s)w_{n+1} = \delta \sigma(s) \dot{w}_{n+1}, \tag{8.3}
\]

where \(\delta = \beta_0 h\) and

\[
 \rho(s) = 1 + \sum_{i=1}^{m} \alpha_i e^{-ish}, \quad \sigma(s) = 1 + \sum_{i=0}^{m} (\beta_i/\beta_0) e^{-ish}. \tag{8.4}
\]

Two other operational forms of (8.3) are

\[
 \dot{w}_{n+1} = \frac{\rho(s)}{\delta \sigma(s)} w_{n+1} = \phi(s) w_{n+1}, \quad w_{n+1} = \frac{\delta \sigma(s)}{\rho(s)} \dot{w}_{n+1} = \tau(s) \dot{w}_{n+1}. \tag{8.5}
\]

The functions \(\phi(s)\) and \(\tau(s) = (\phi(s))^{-1}\) will be called the amplification and transfer functions, respectively, of the LMS method (8.1). These names are consistent with control theory terminology when (8.1) is viewed as a recursive digital filter.

```
LMSOperationalShiftForm[α_, β_, h_, s_] := Module[
{m = Length[α] - 1, ρs, cs, φs, β0 = β[[1]], δ},
δ = h*β0;
ρs = 1 + Sum[α[[i + 1]]*Exp[-i*s*h], {i, 1, m}];

 cs = 1 + Sum[(β[[i + 1]]/β0)*Exp[-i*s*h], {i, 1, m}];

 φs = cs/ρs; τs = 1/φs;

 Return[Simplify[{
ρs, cs, φs, τs}]]];

ClearAll[α, β, h, s]; α = {1, -1}; β = {1/2, 1/2}; (*TR*)
{ρs, cs, φs, τs} = LMSOperationalShiftForm[α, β, h, s];
Print[{ρs, cs, φs, τs}];

Print[Simplify[Normal[Series[{ρs, cs, φs, τs}, {s, 0, 6}]]]]; α = {1, -3/2, 3/5, 1/10}; β = {3/5, 0, 0, 0}; (*Park 3 step*)
{ρs, cs, φs, τs} = LMSOperationalShiftForm[α, β, h, s];
Print[{ρs, cs, φs, τs}];

Print[Simplify[Normal[Series[{ρs, cs, φs, τs}, {s, 0, 6}]]]];
```

Figure 8.1. Module to compute functions \(\rho(s), \sigma(s), \phi(s)\) and \(\tau(s)\) for an LMS operator.

Invocation for TR and Park 3-step illustrated; results given in Example 8.1.
For consistent LMS integration formulas, the function $\phi(s)$ must approximate $s$ for small stepsizes. More precisely, $\phi$ must have the formal series expansion $\phi(s) \approx s[1 + O(h)]$. Under certain conditions the discrete Laplace transform approaches, as $h \to 0$, the conventional Laplace transform. Now for the latter the transform of a derivative of a function $f(t)$ is $sF(s) - f(0^+)$. Thus $\phi(s) \approx s[1 + O(h)]$, simply says that $\phi(s)$ is a consistent approximation to the time derivative $d/dt$. Moreover, the difference $\phi(s) - s$ provides a representation of the local truncation error of the LMS formula (8.1).

**Example 8.1.** For the Trapezoidal Rule (TR), (8.1) is $w_{n+1} - w_n = \frac{1}{2} h(\dot{\bar{u}}_{n+1} + \dot{\bar{u}}_n)$. Thus $m = 1$, $\alpha_0 = -\alpha_1 = 1$, $\beta_0 = \beta_1 = \frac{1}{2}$ and $\delta = \frac{1}{2} h$. The Mathematica module LMSOperationalShiftForm, listed in Figure 8.1, returns

$$
\begin{align*}
\rho(s) &= 1 - e^{-sh} = hs - \frac{1}{2} s^2 h^2 + \frac{1}{6} s^3 h^3 - \frac{1}{24} s^4 h^4 + \ldots \\
\sigma(s) &= 1 + e^{-sh} = 2h - s^2 h^2 + \frac{1}{2} s^3 h^3 + \frac{1}{24} s^4 h^4 + \ldots, \\
\phi(s) &= -2 + 2e^{hs} = \frac{2h^{-1}(1 + \tanh(\frac{1}{2}h s)) = s(1 - \frac{1}{12} s^2 h^2 + \frac{1}{120} s^4 h^4 + \ldots),} \\
\tau(s) &= 1/\phi(s) = \frac{1}{2}h \coth(\frac{1}{2}hs) = s^{-1}(1 + \frac{1}{12} s^2 h^2 - \frac{1}{720} s^4 h^4 + \ldots).
\end{align*}
$$

(8.6)

The local truncation error is recovered from backtransforming $[\phi(s) - s] w = (\frac{1}{12} s^2 h^2 + \frac{1}{120} s^4 h^4 + \ldots) w$ to the time domain: $-\frac{1}{12} h^2 w'' + \frac{1}{120} h^4 u'' + \ldots$.

For the Park 3-step method, (8.1) is $w_{n+1} = \frac{3}{2} w_n + \frac{1}{3} w_{n-1} + \frac{1}{18} w_{n-2} = \frac{3}{2} h \bar{u}_{n+1}$. Thus $m = 3$, $\alpha_0 = 1$, $\alpha_1 = -3/2$, $\alpha_2 = 3/5$, $\alpha_3 = 1/10$, $\beta_0 = 3/5$, $\beta_1 = \beta_2 = \beta_3 = 0$, and $\delta = 3h/5$. The module gives

$$
\begin{align*}
\rho(s) &= 1 - \frac{1}{2} e^{-sh} + \frac{3}{5} e^{-2sh} - \frac{1}{18} e^{-3sh} = \frac{1}{2}h - \frac{1}{6} s^2 h^3 + \frac{11}{120} s^4 h^5 - \ldots, \\
\sigma(s) &= 1 \\
\phi(s) &= \frac{\rho(s)}{3h/5} = s(1 - \frac{1}{6} s^2 h^2 + \frac{11}{120} s^4 h^4 - \frac{1}{12} s^5 h^5 + \ldots), \\
\tau(s) &= 1/\phi(s) = s^{-1}(1 + \frac{1}{6} s^2 h^2 - \frac{23}{360} s^4 h^4 + \frac{1}{12} s^5 h^5 + \ldots).
\end{align*}
$$

(8.7)

The local truncation error is the preimage of $\phi(s) - s$: $-\frac{1}{6} h^2 w'' + \frac{11}{360} h^4 w'' + \ldots$.

**§8.1.2. The $z$ Transform**

The $z$ transform is obtained through the substitution

$$
z = e^{hs}
$$

(8.8)

The transformation (8.8) maps the left complex $s$ plane onto the exterior of the unit disk of the $z$ plane. Exponential polynomials such as $\rho(s)$ and $\sigma(s)$ become rational functions in $z$:

$$
\rho(z) = 1 + \sum_{i=1}^{m} \alpha_i z^{-i}, \quad \sigma(z) = 1 + \sum_{i=1}^{m} (\beta_i / \beta_0) z^{-i}.
$$

(8.9)

The amplification and transfer functions $\phi(z) = \rho(z) / (\delta \sigma(z))$ and $\tau(z) = \delta \sigma(z) / \rho(z)$ become rational polynomials in $z$.

**§8.2. Predictors**

The general expression of a linear multistep predictor or LMS predictor for constant stepsize $h$ is

$$
w_{n+1}^p = \sum_{i=1}^{m} \alpha_i^p w_{n-i+1} + \beta_0 h \sum_{i=1}^{m} \beta_i^p w_{n-i+1} + (\beta_0 h)^2 \sum_{i=1}^{m} \gamma_i^p w_{n-i+1} + \ldots
$$

(8.10)

where $\alpha_i^p$, $\beta_i^p$ and $\gamma_i^p$ are numeric coefficients. The segregation of $\beta_0$ factors to compose the generalized stepsize $\delta = \beta_0 h$ simplifies subsequent developments. Since for second-order governing equations (as considered here), historical information involves at most computed second derivatives, we shall not consider LMS predictors continuing beyond the $u_{n-i+1}$ terms. Note also that (8.10) is assumed to involve $m$ past terms, where $m$ is the number of steps in the LMS integrator; if it involve less, the coefficient arrays are completed with zeros. The operational version of (8.10) is

$$
w_{n+1}^p = e(s) w_{n+1} + e'(s) \delta \bar{u}_{n+1} + e''(s) \delta^2 \bar{u}_{n+1},
\quad \text{with}
$$

$$
e(s) = \sum_{i=1}^{m} \alpha_i^p e^{-ish}, \quad e'(s) = \sum_{i=1}^{m} \beta_i^p e^{-ish}, \quad e''(s) = \sum_{i=1}^{m} \gamma_i^p e^{-ish}.
$$

(8.11)
If one assumes that $w_{n+1}$ is evaluated by the backward difference formulas in (8.5), that is, $w_{n+1} = \phi(s) w_n$ and likewise $\dot{w}_{n+1} = [\phi(s)]^\prime w_n$. To simplify the notation call $\psi(s) = \delta \phi(s) / \sigma(s)$, and we arrive at the compact operational form

$$w_{n+1} = e(s) w_{n+1}, \quad \text{in which } e(s) = e + \psi \epsilon' + \psi^2 \epsilon''.$$ (8.12)

If the historical derivatives in (8.10) are not evaluated from the backward difference expressions (8.5), then (8.12) does not hold. As discussed in Chapter 7, this is a function of the computational path. It will be shown in §9.4, however, that desirable predictors for coupled systems do have the form (8.12).

§8.2.1. Historical Predictor

The operational equivalent of $w_{n+1} = \delta w_{n+1} + h^w_w$ is $w_{n+1} = \delta w_{n+1} + h^w_w (s)$, where the historical vector in $s$ space can be expressed as

$$h^w_w (s) = [1 - \rho(s)] w_{n+1} + \delta [\sigma(s) - 1] \dot{w}_{n+1}. \quad (8.13)$$

The right hand side can be viewed as a special case of the predictor formula (8.11) if

$$\epsilon = 1 - \rho(s), \quad \epsilon' = \sigma(s) - 1, \quad \epsilon'' = 0.$$ (8.14)

The foregoing choice for $w_{n+1}^p$, that is $w_{n+1}^p = h^w_w$, will be called the historical predictor.

§8.3. Dynamic Force

The derivation of characteristic stability equations for the partitioned integration procedures introduced in Chapter 7 requires the operational expression of the right hand side term $g_{n+1}$, commonly called the dynamic force, in terms of previously computed solutions. In doing so, the computational path details discussed in Chapter 6 come into the picture.

Below is the derivation for path (2), which is the easiest to follow. The same integration formula for all components of the state vector is assumed. Operators $\rho(s), \sigma(s), \phi(s) = \rho(s) / (\delta \sigma(s))$ and $\psi(s) = \delta \phi(s) / \sigma(s)$ are abbreviated to $\rho, \sigma, \phi$ and $\psi$, respectively.

$$h^w_n = (1 - \rho) u_{n+1} + \delta (\sigma - 1) \dot{u}_{n+1} = [(1 - \rho) + \delta (\sigma - 1)] \phi | u_{n+1} = (1 - \psi) u_{n+1},$$

$$h^\psi_n = (1 - \psi) v_{n+1} = (1 - \psi) (AM \phi + B) u_{n+1}.$$ (8.15)

$$g_{n+1} = [M + \delta (C - A^{-1} B)] h^w_n + \delta A^{-1} h^\psi_n + \delta^2 f_n$$

$$= [(1 - \psi^2) M + \delta (1 - \psi) C] u_{n+1} + \delta^2 f_{n+1}.$$ For other computational paths the calculations are also straightforward but more laborious. The general result can be expressed as follows:

$$g_{n+1} = [a_M M + a_C C + a_K K] u_{n+1} + a_f f_{n+1}. \quad (8.16)$$

where the coefficients $a_M, a_C, a_K$ and $a_f$ are operators that can be expressed as functions of $\rho$ and $\sigma$. These functions are listed in Table 8.1.

For all paths except (0'), the expressions are independent of the weighting matrices $A$ and $B$. For path (0'), the expressions listed in Table 8.1 are only valid for the (J) choice of auxiliary vector $A = I$ and $B = C$. Otherwise the coefficients become extremely complex, involving matrices $M, C$ and $K$. Consequently the stability of the time integrator is independent of the choice of $v$ in paths (0), (1) and (2), as further discussed in the next Chapter.

If two different integrators are used for $u$ and $v$ of a second order system, the calculations for the operational form of $g_{n+1}$ become impractical to do by hand. They have been actually implemented in the Mathematica module listed in Figure 8.2. Here the LMS integrator for $u$ is defined by $\{\rho^u, \sigma^u, \delta_u\}$ whereas for $v$ is defined by $\{\rho^v, \sigma^v, \delta_v\}$. Results from running the script are transcribed to Table 8.2. The script also prints the specialization $\rho^u = \rho^v = \rho, \sigma^u = \sigma^v = \sigma$ and $\delta_u = \delta_v = \delta$, which reproduces the results of Table 8.1.
ClearAll[A,B,Cm,M,K,ρ,σ,δ,ρu,ρv,σu,σv,δu,δv,un,fn];

\[ \phi_v = \frac{\rho_v}{(\delta_v * \sigma_v)}; \quad \phi_u = \frac{\rho_u}{(\delta_u * \sigma_u)}; \quad \psi_v = \frac{\rho_v}{\sigma_v}; \quad \psi_u = \frac{\rho_u}{\sigma_u}; \]

same = {ρu->ρ, ρv->ρ, σu->σ, σv->σ, δu->δ, δv->δ};

gn=Table[0,{4}]; pathlist={"0'","0","1","2"};
For [i=1, i<=4, i++, p=pathlist[[i]]; ClearAll[A,B];
If [p=="0'", A=1; B=Cm;
 vdotn=A*(fn-K*un-Cm*φ_u*un)+B*φ_u*un; vn=vdotn/φ_v;
hvn=(1-ρ_v)*vn+δ_v*(σ_v-1)*vdotn;
 udotn=(vn-B*un)/(A*M);
 unh=(1-ψ_u)*un; hvn=(1-ρ_v)*vn+δ_v*σ_v*vn;
 gn[[1]]=(M+δ_v*(Cm-B/A))*hun+δ_u/A*hvn+δ_u*δ_v*fn];
If [p=="0", vdotn=A*(fn-K*un-Cm*φ_u*un)+B*φ_u*un; vn=vdotn/φ_v;
hun=(1-ψ_u)*un; hvn=(1-ρ_v)*vn+δ_v*σ_v*vn;
gn[[2]]=(M+δ_v*(Cm-B/A))*hun+δ_u/A*hvn+δ_u*δ_v*fn];
If [p=="1", vn=(A*M*φ_u+B)*un; vdotn=A*(fn-K*un-Cm*φ_u*un)+B*φ_u*un; unh=(1-ψ_u)*un; hvn=(1-ρ_v)*vn+δ_v*σ_v*vn;
gn[[3]]=(M+δ_v*(Cm-B/A))*hun+δ_u/A*hvn+δ_u*δ_v*fn];
If [p=="2", vn=(A*M*φ_u+B)*un; hun=(1-ψ_u)*un; hvn=(1-ψ_v)*vn;
gn[[4]]=(M+δ_v*(Cm-B/A))*hun+δ_u/A*hvn+δ_u*δ_v*fn];
]

aM=Simplify[Coefficient[gn/.un->1,M]];
aC=Simplify[Coefficient[gn/.un->1,Cm]];
aK=Simplify[Coefficient[gn/.un->1,K]];
af=Simplify[Coefficient[gn/.un->1,fn]];
]

a=Simplify[Together[Transpose[{aM,aC,aK,af}]]];
Print["a (diff integr)="];Collect[a,(δ_u,δ_v)]//MatrixForm;
Print["a (same integr)="];Collect[Simplify[a/.same],δ]//MatrixForm;

Figure 8.2. Script to compute coefficients \(a_M\) through \(a_f\) for the dynamic force term \(g_{n+1}\) as functions of the computational path. Matrices A, B, etc, are represented as scalar symbols for simplicity. For path (0') A and B are set to the (J) choice of auxiliary vector, else they are left arbitrary.

Table 8.1. Operational Dynamic Form Term with Same Integrator for u and v

<table>
<thead>
<tr>
<th>Path</th>
<th>(a_M)</th>
<th>(a_C)</th>
<th>(a_K)</th>
<th>(a_f)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0')</td>
<td>(1 - \rho)</td>
<td>(\delta(1 - \sigma))</td>
<td>(\delta^2(1 - \frac{\sigma^2}{\rho}))</td>
<td>(\delta^2 \frac{\sigma^2}{\rho})</td>
</tr>
<tr>
<td>(0)</td>
<td>(1 - \frac{\rho}{\sigma})</td>
<td>0</td>
<td>(\delta^2(1 - \frac{\sigma}{\rho}))</td>
<td>(\delta^2 \frac{\sigma}{\rho})</td>
</tr>
<tr>
<td>(1)</td>
<td>(1 - \frac{\rho^2}{\sigma})</td>
<td>(\delta(1 - \rho))</td>
<td>(\delta^2(1 - \frac{\sigma}{\rho}))</td>
<td>(\delta^3 \sigma)</td>
</tr>
<tr>
<td>(2)</td>
<td>(1 - \frac{\rho^2 \rho^2}{\sigma^2})</td>
<td>(\delta(1 - \rho))</td>
<td>0</td>
<td>(\delta^2)</td>
</tr>
</tbody>
</table>

Results for path (0') only valid for (J) choice of auxiliary vector.

Notes and Bibliography

Most of the material is extracted from portions of an unpublished report [18], as well as from [20]. An exception is Table 8.2, which is new (and would have been impossible to fill out without the help of a computer algebra system). In [18] it was stated that the dynamic force coefficients \(a_M\) through \(a_f\) shown in Table 8.1 for path (0') were independent of the auxiliary vector. Recomputation with Mathematica showed that this is only correct for choice (J). For other choices the coefficients were far more complex and would in fact have ruined stability. This mistake did not affect the Lockheed structural dynamic and underwater shock analysis codes used by the Navy and NASA, which were based on either the (J0) or (C2) formulations.

The extraordinary simplicity of the operational method as regards answering questions in truncation error...
Table 8.2. Operational Dynamic Form Term with Different Integrators for \( u \) and \( v \)

<table>
<thead>
<tr>
<th>Path</th>
<th>( a_M )</th>
<th>( a_C )</th>
<th>( a_K )</th>
<th>( a_f )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0')</td>
<td>( 1 - \rho^u )</td>
<td>( \delta_u(1 - \sigma^u) )</td>
<td>( \delta_u \delta_v(1 - \frac{\sigma^u \sigma^v}{\rho^u \rho^v}) )</td>
<td>( \delta_v \delta_u \frac{\sigma^u \sigma^v}{\rho^u \rho^v} )</td>
</tr>
<tr>
<td>(0)</td>
<td>( 1 - \frac{\rho^u}{\sigma^u} )</td>
<td>( \delta_u(1 - \frac{\rho^u \sigma^v}{\rho^u \sigma^v}) )</td>
<td>( \delta_u \delta_v(1 - \frac{\sigma^u \sigma^v}{\rho^u \rho^v}) )</td>
<td>( \delta_v \delta_u \frac{\sigma^u \sigma^v}{\rho^u \rho^v} )</td>
</tr>
<tr>
<td>(1)</td>
<td>( 1 - \frac{\rho^u \sigma^v}{\sigma^v} )</td>
<td>( \delta_u(1 - \frac{\rho^u \sigma^v}{\sigma^v}) )</td>
<td>( \delta_u \delta_v(1 - \sigma^v) )</td>
<td>( \delta_v \delta_u \sigma^v )</td>
</tr>
<tr>
<td>(2)</td>
<td>( 1 - \frac{\rho^u \sigma^v}{\sigma^v} )</td>
<td>( \delta_u(1 - \frac{\rho^u \sigma^v}{\sigma^v}) )</td>
<td>0</td>
<td>( \delta_u \delta_v )</td>
</tr>
</tbody>
</table>

Results for path (0') only valid for (J) choice of auxiliary vector.

and stability has not been given proper credit in the literature. Compare, for example, the one-line derivation of the local error of the Trapezoidal Rule in Example 8.1 with the several pages expended on such matters in the ODE textbooks cited in Chapter 3.

9. CHARACTERISTIC STABILITY EQUATIONS

This Chapter derives characteristic polynomials for time-discrete second-order equations expressed in operational form. These polynomials form the basis for stability analysis of both monolithic and partitioned analysis procedures for coupled systems.

§9.1. Stability Analysis Steps

A typical stability analysis involves the following steps.

1. The governing equations are time-discretized using a LMS integrator and an algebraic or differential partition approach. The applied force term is dropped. The system is transformed to image space (s space) using the ordinary Laplace transform as explained in Chapter 8. This step produces a characteristic function that is a matrix exponential-polynomial in \( e^{sh} \).

2. Through the substitution \( z = e^{sh} \) of the Laplace transform and appropriate scaling by a power of \( z \) produces a characteristic matrix polynomial in the \( z \) argument. This is called the amplification characteristic polynomial. The amplification polynomial is stable if all of its zeros lie in the unit disk of the complex \( z \) plane. If the polynomial is numeric, this can be quickly investigated through the Jury stability test [50], which is a convenient implementation of the Schur-Cohn criterion.

3. If the stability polynomial involves free parameters, the use of the Routh-Hurwitz stability test is usually simpler than Jury’s. The amplification polynomial can be converted into a Routh polynomial through the mapping \( s = \frac{(z - 1)/(z + 1)}{a} \).

The following sections are devoted to the formulation of the characteristic functions and polynomials. A key result is that the effect of the computational path on stability can be compensated through adjustments in the predictor.

§9.2. Characteristic Function for Algebraic Partitioning

The governing time-discrete equation is \( H_1 u_{n+1} = g_{n+1} - H_2 u_{n+1}^P \). Using the results of Chapter 8 we can write its Laplace transform as

\[
H_1 u_{n+1} = (a_M M + a_C C + a_K K)u_{n+1} + a_f f_{n+1} - H_2 u_{n+1}^P. \tag{9.1}
\]

To derive the characteristic equation, delete the applied force term \( f_{n+1} \), express the predicted term in the operational form derived before, and finally collect all terms premultiplying \( u_{n+1} \). This produces the homogeneous characteristic system

\[
P(s)u_{n+1} = 0. \tag{9.2}
\]
of the computational path. Note also that (9.5) holds for any partition of the form $C$. Computations are carried out by the Mathematica

Remark 9.1

Remark 9.2

where $P(s) = H_1 + e(s)H_2 - a_M(s)M - a_C(s)C - a_K(s)K$. (9.3)

where $e, a_M, a_C$ and $a_K$ are defined in §8.3. Replacing $H_1 = M + \delta C_1 + \delta^2 K_1$ and $H_2 = \delta C_2 + \delta^2 C_2$

we obtain

$P(s) = P_1(s) + P_2(s).$ (9.4)

If the same $\{\rho, \sigma, \delta\}$ integrator is used for $u$ and $v$

$P_1(s) = \rho^2 M + \delta \rho \sigma C + \delta^2 \sigma^2 K,$

$P_2(s) = c_D C_2 + c_K K_2.$ (9.5)

The coefficients $c_D$ and $c_K$ are scalar monomial functions of $\rho$ and $\sigma$ listed in the rightmost columns of Table 9.1 for the four computational paths $(0'), (0), (1)$ and $(2)$. Also given are values of $c_C$ and $c_K$ for the differential partition case, which is studied below and turns out to be independent of the computational path. Note also that (9.5) holds for any partition of the form $C = C_1 + C_2$ and $K = K_1 + K_2$. That is, its validity goes beyond the specialized case of a staggered partition.

Remark 9.1. Setting $e(s) = 1$ reduces $P(s)$ to $P_1$. But this is equivalent to saying that $u_{i+1} = u^p_{i+1}$; that is, the system is treated by a fully implicit integration scheme. This is the same as taking $C_1 = K_2 = 0$. Consequently, $P_1(s)$ is merely the characteristic function of a monolithic solution. It is well known that the stability of such solution procedure does not depend on the computational path followed. At the other extreme, if the partition $C_2 = C$ and $K = K_2$ is selected, a fully explicit scheme is obtained. Thus the foregoing expressions are applicable to a full spectrum of integration methods. In a partitioned procedure situated between the two extreme cases (fully explicit and fully implicit) mentioned above, $P(s)$ does depend on the computational path. It is however, independent of the choice of auxiliary vector $v$ for paths $(0)-(2)$, whereas for path $(0')$ the independence only holds for the $(I)$ choice $v = M u + Cu$.

Remark 9.2. If the integrators for $u$ and $v$ are different, the characteristic equation still splits as (9.4). Computations are carried out by the Mathematica script shown in Figure 9.1. Now $P_1(s) = \rho^2 \rho_M + \delta \rho_0 \sigma C + \delta \delta_0 \sigma_0 K$ for paths $(0), (1)$ and $(2)$, whereas for path $(0')$ the coefficient of $C$ is $\delta_0 (\sigma_0 - 1) \rho_C + \delta_0 \rho_e$. The other component is still $P_2(s) = c_D C_2 + c_K K_2$, where $c_C$ and $c_D$ are given in the middle columns of Table 9.1.

Figure 9.1. Script to compute coefficients for the characteristic equation. Different integrators for $u$ and $v$ allowed.

$$aMCKf = \{(1 - p u, \delta u \ast (1 - \sigma u), \delta u \ast \delta v \ast (\rho v - \sigma u \ast \sigma v) / \rho v, \delta u \ast \delta v \ast \sigma u \ast \sigma v / \rho v),$$

$$\{1 - p u / \sigma u, \delta v \ast (1 - \rho v \ast \sigma v) / \rho v, \delta u \ast \delta v \ast \sigma v / \rho v, \delta u \ast \delta v \ast \sigma u / \rho u, \delta u \ast \delta v \ast (-1 + \sigma v), \delta u \ast \delta v \ast \sigma v / \rho v \},$$

$$\{1 - p u / \rho v / \sigma u \ast \sigma v, \delta v \ast (1 - \rho u / \rho u) 0, \delta u \ast \delta v \}, \quad P_1 = P_2 = \{};$$

For $i = 1, i = 4, i++,$ $\{aM, aC, aK, af\} = aMCKf[[i]]; H1 = M + \delta \ast v \ast (Cm - Cm2) + \delta u \ast \delta v \ast (K - K2);$ $P = \text{Simplify}[H1 + (ee + 1) H2 - \text{aM} * \text{aC} * \text{aK}];$ $P = \text{Numerator}[\text{Together}[P]]; P1 = \text{Simplify}[P1, \text{Coefficient}[P1, \{M, Cm, K\}]]; \text{AppendTo}[P1c, \text{Coefficient}[P1, \{M, Cm, K\}]]; P2 = \text{Simplify}[Coefficient[P2, \{M, Cm, K\}]]; \text{AppendTo}[P2c, \text{Coefficient}[P2, \{Cm, K2\}]]; \text{Print}[^\text{P1c} = ^\text{P1c} / \text{MatrixForm}, P2c = ^\text{P2c} / \text{MatrixForm};$ $\text{same} = \{\rho u, \rho v \rightarrow \rho, \sigma u \rightarrow \sigma, \sigma v \rightarrow \sigma, \delta u \rightarrow \delta, \delta v \rightarrow \delta\};$ Print[^\text{P1c} = Simplify[P1c / same] / MatrixForm, $^\text{P2c} = \text{Simplify}[P2c / same] / \text{MatrixForm};$
Table 9.1. Coefficients in Characteristic Matrix Equations

<table>
<thead>
<tr>
<th>Partition</th>
<th>Path</th>
<th>Different Integrators</th>
<th>Same Integrator</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$c_C$</td>
<td>$c_D$</td>
</tr>
<tr>
<td>Algebraic</td>
<td>(0')</td>
<td>$\delta_x \rho_u$</td>
<td>$\delta_x \delta \rho_u$</td>
</tr>
<tr>
<td>Algebraic</td>
<td>(0)</td>
<td>$\delta_x \rho_u \sigma_v$</td>
<td>$\delta^2 \rho_x \sigma_u$</td>
</tr>
<tr>
<td>Algebraic</td>
<td>(1)</td>
<td>$\delta_x \sigma_v$</td>
<td>$\delta \delta \rho_u \sigma_v$</td>
</tr>
<tr>
<td>Algebraic</td>
<td>(2)</td>
<td>$\delta_x \sigma_v \sigma_v$</td>
<td>$\delta \delta \delta \rho_u \sigma_v$</td>
</tr>
<tr>
<td>Differential</td>
<td>all</td>
<td>$\delta_x \rho_u \sigma_v$</td>
<td>$\delta \delta \delta \sigma_v$</td>
</tr>
</tbody>
</table>

Results for path (0') only valid for (J) choice of auxiliary vector.

§ 9.3. Characteristic Function for Differential Partitioning

The governing time-discrete equation is $\mathbf{M}\ddot{\mathbf{u}} + \mathbf{C}_1 \dot{\mathbf{u}} + \mathbf{K}_1 \mathbf{u} = \mathbf{f}_{n+1}$. Its Laplace transform is

$$H_1(s) = \mathbf{A}_M + \mathbf{a}_C \mathbf{C} + \mathbf{a}_K \mathbf{K}_1 \mathbf{u}_{n+1} + a \hat{\mathbf{f}}_{n+1}$$

(9.6)

Proceeding as in the previous Section we find

$$P(s) = P_1 - e_C c_D C_2 - e_K c_K K_2.$$  

(9.7)

Here $P_1$ is the same function as in the algebraic partition case, $e_C$ and $e_K$ are the predictor operators applied to get $\dot{\mathbf{u}}^P_{n+1}$ and $\mathbf{u}^P_{n+1}$, respectively, and $c_C$ and $c_K$ are given in Table 9.1.

The characteristic function turns out to be independent of the computational path. The source of this property can be traced to the appearance of the pseudo-force term

$$\mathbf{f}_{n+1} = \mathbf{f}_{n+1} - C_2 \mathbf{u}^P_{n+1} - K_2 \mathbf{u}^P_{n+1},$$  

(9.8)

in (9.6). Survival of the predicted terms after setting $\mathbf{f} = 0$ dilutes all computational path effects.

If the differential system is undamped, that is $\mathbf{C} = 0$ and $e_K$ is identified with $e$, it is obvious that differential partitioning becomes identical to algebraic partitioning under path (0). On the other hand, if the differential system has damping but no stiffness, that is, $\mathbf{K} = 0$, differential partitioning becomes identical to algebraic partitioning under path (0) if $e_C$ is identified with $e$. This has implications as regards predictor selection, as discussed in the following Section, and shows that differential partitioning may be regarded as a sort of “hybrid” algebraic partitioning.

§ 9.4. Predictor Selection

§ 9.4.1. Stable Predictors

In what follows the same integrator for $\mathbf{u}$ and $\mathbf{v}$ is assumed. The predictor

$$e^* = 1 - \rho,$$

(9.9)

used in conjunction with algebraic partitioning and path (0') yields the characteristic function

$$P^*(s) = \rho^2 (\mathbf{M} - \delta C_2 - \delta^2 K_2) + \delta \rho \sigma \mathbf{C} + \delta^2 \sigma^2 \mathbf{K}.$$  

(9.10)

The characteristic equation $\det(P^*) = 0$ was found to be stability optimal in the case of an undamped system treated by the trapezoidal rule. The term optimality is used here in the following sense: (i) the A-stability of the fully implicit scheme $K_2 = 0$ was preserved for implicit-implicit partitions, such as the staggered partition; and (ii) the stability limit of the fully explicit scheme ($K_1 = 0$) was retained for implicit-explicit partitions. In other words, system partitioning did not affect intrinsic stability characteristics.
Numerical experiments have shown that the foregoing properties also hold for other \( \{\rho, \sigma\} \) LMS integration operators if the predictor (9.9) is used. A rigorous proof of this property, however, has not been found yet.

What if the system is damped? The foregoing conclusions still hold if \( C \) is a linear combination of \( K \) and \( M \) (the so-called Rayleigh damping in mechanical systems). For system with more general damping models, the search for stability optimal predictors requires case-specific computations, since no general theory is available. Nevertheless, the above predictor (as well as the equivalent predictors discussed below) provide a suitable starting point for the search.

§9.4.2. Equivalent Predictors
The *same characteristic equation* can be obtained for different computational paths if predictors are appropriately adjusted. This leads to the notion of *equivalent predictors*. Let us identify equivalent predictors by the operator symbols

\[
\begin{align*}
  e_0' : & \text{ predictor operator for path (0')} \\
  e_0, e_1, e_2 : & \text{ equivalent predictors for paths (0), (1) and (2), respectively.} \\
  e_C, e_K : & \text{ equivalent predictor operators for } u \text{ and } u, \text{ respectively}
\end{align*}
\]

We consider \( e_0' \) as basic predictor as it is usually the simplest one. Then a straightforward calculation shows that the equivalent predictors are related to \( e_0' \) as follows

\[
\begin{align*}
  e_0 &= 1 + (e_0' - 1)/\sigma, & e_1 &= 1 + (e_0' - 1)\rho/\sigma, \\
  e_2 &= 1 + (e_0' - 1)\rho/\sigma^2, & e_C &= e_0, & e_K &= e_2.
\end{align*}
\]

If \( e_0' = e_* \), the equivalent predictors are

\[
\begin{align*}
  e_0^* &= 1 - \rho/\sigma, & e_1^* &= 1 - \rho^2/\sigma, & e_2^* &= 1 - \rho/\sigma^2, & e_C^* &= e_0^*, & e_K^* &= e_2^*.
\end{align*}
\]

These predictors are generally rational polynomials in \( z \). How can these be implemented in term of past information? The easiest way is to use computed historical derivatives, as explained previously. For example:

\[
\begin{align*}
  e_0^* &= 1 - \rho/\sigma = (1 - \rho) + (\sigma - 1)\rho/\sigma.
\end{align*}
\]

Comparing this to (8.11) yields \( \epsilon = 1 - \rho, \epsilon' = \sigma - 1 \) and \( \epsilon'' = 0 \). Note that this is precisely the historical predictor (8.13), so \( u_{n+1} = h^n u \) in path (0). Operational expressions for various paths are collected in Table 9.2. Reverting these expressions to the time domain provides predictor formulas such as those listed in Table 9.3 for two specific time integrators: Trapezoidal Rule and Park 3-step.

**Remark 9.3.** If the predictor for path (0') involves computed derivative terms up to order \( k \), the equivalent predictors for paths (0), (1) and (2) generally involve computed derivative terms up to order \( k + 1, k + 1 \) and \( k + 2 \), respectively. It is seen that for second-order governing equations, \( k = 0 \) (no historical derivatives in the (0') predictor is the only choice.

**Remark 9.4.** Coalescence of the characteristic equation for equivalent predictors implies identical stability characteristics. It does not imply the same accuracy, however, because the characteristic equation does not account for external forcing actions and treatment of initial conditions.

**Notes and Bibliography**
Most of the material is extracted from portions of an unpublished report [18], as well as from [20], with some misprints in the predictor formula for path (1) corrected. The formulas using different integrators for \( u \) and \( v \) are new.
Table 9.2. Stability-Optimal Predictors for Undamped Systems

<table>
<thead>
<tr>
<th>Partition</th>
<th>Path</th>
<th>$\epsilon$</th>
<th>$\epsilon'$</th>
<th>$\epsilon''$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algebraic</td>
<td>(0')</td>
<td>$1 - \rho$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Algebraic</td>
<td>(0)</td>
<td>$1 - \rho$</td>
<td>$\sigma - 1$</td>
<td>0</td>
</tr>
<tr>
<td>Algebraic</td>
<td>(1)</td>
<td>$1 - \rho$</td>
<td>$\sigma - \rho$</td>
<td>0</td>
</tr>
<tr>
<td>Algebraic</td>
<td>(2)</td>
<td>$1 - \rho$</td>
<td>$\sigma - \rho$</td>
<td>$\sigma - 1$</td>
</tr>
<tr>
<td>Differential</td>
<td>all</td>
<td>Same as for paths (2) and (0) above for $\mathbf{u}<em>{n+1}^p$ and $\mathbf{\dot{u}}</em>{n+1}^p$, respectively.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Results for path (0') only valid for (J) choice of auxiliary vector.
Same integrator assumed for $\mathbf{u}$ and $\mathbf{v}$

Table 9.3. Examples of Stability-Optimal Predictors for Undamped Systems

<table>
<thead>
<tr>
<th>Partition</th>
<th>Path</th>
<th>Trapezoidal rule</th>
<th>Park 3-step</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algebraic</td>
<td>(0')</td>
<td>$\mathbf{u}_{n+1}^p = \mathbf{u}_n$</td>
<td>$\mathbf{u}<em>{n+1}^p = 1.5\mathbf{u}<em>n - 0.6\mathbf{u}</em>{n-1} + 0.1\mathbf{u}</em>{n-2}$</td>
</tr>
<tr>
<td>Algebraic</td>
<td>(0)</td>
<td>$\mathbf{u}_{n+1}^p = h\mathbf{u}_n + \frac{1}{2}h\mathbf{\dot{u}}_n$</td>
<td>Same as for (0')</td>
</tr>
<tr>
<td>Algebraic</td>
<td>(1)</td>
<td>$\mathbf{u}_{n+1}^p = \mathbf{u}_n + \frac{1}{2}h\mathbf{\dot{u}}_n$</td>
<td>$\mathbf{u}<em>{n+1}^p = 1.5(\mathbf{u}<em>n + 0.6h\mathbf{\dot{u}}<em>n) - 0.6(\mathbf{u}</em>{n-1} + 0.6h\mathbf{\dot{u}}</em>{n-1}) + 0.1(\mathbf{u}</em>{n-2} + 0.6h\mathbf{\dot{u}}_{n-2})$</td>
</tr>
<tr>
<td>Algebraic</td>
<td>(2)</td>
<td>$\mathbf{u}_{n+1}^p = \mathbf{u}_n + h\mathbf{\dot{u}}_n + \frac{1}{2}h^2\mathbf{\ddot{u}}_n$</td>
<td>Same as for (1)</td>
</tr>
<tr>
<td>Differential</td>
<td>all</td>
<td>Same as for path (2)</td>
<td>Same as for path (2)</td>
</tr>
</tbody>
</table>

$\mathbf{u}_{n+1}^p = 1.5\mathbf{u}_n - 0.6\mathbf{u}_{n-1} + 0.1\mathbf{u}_{n-2}$ |

Results for path (0') only valid for (J) choice of auxiliary vector.
Same integrator assumed for $\mathbf{u}$ and $\mathbf{v}$

10. STRUCTURE-STRUCTURE INTERACTION

This Chapter derives characteristic polynomials for time-discrete second-order equations expressed in operational form. These polynomials form the basis for stability analysis of both monolithic and partitioned analysis procedures for coupled systems.

§10.1. The Need for Model Systems

The characteristic matrix equations derived in Chapter 9 may look frightening to an unprepared reader. Consider, for example, a second order system with one million equations, and apply a 3-step integration method. Then $\mathbf{P}(z)$ has $2 \times 1,000,000 \times 3 = 6$ million or more eigenvalues (depending on the computational path and the predictor chosen). To verify numerical stability, the location of each eigenvalue would have to be determined. This is further complicated by the fact that the constituent matrices are generally unsymmetric and that in method design it is necessary to work with free parameters.

The presence of this root morass would pose no problem if a simultaneous solution scheme, whether implicit or explicit, is used. The normal modes of the semidiscrete (time-continuous) system survive in the time-discrete (difference) system. One can therefore look at a one-degree-of-freedom model problem, and infer from its spectral analysis the numerical stability of the entire difference system.
Normal modes do not generally survive, however, in the difference system if a partitioned integration scheme is used. This has two implications: (i) the full coupled system has to be considered in the stability evaluation, and (ii) more sophisticated mathematical techniques, such as the theory of positive-real polynomials, must be resorted to in order to arrive at general conclusions.

In the following stability analysis examples a compromise is struck. A model system is still used, but it is no longer one-dimensional: it contains as many equations as partitions. This makes the stability analysis feasible on a CAS while keeping free parameters.

§10.2. Structure Partitions

Consider a structural system governed by the semidiscrete equations of motion

\[ M \ddot{u} + C \dot{u} + K u = f(t). \]  

(10.1)

For the ensuing development of a staggered solution procedure, this system is assumed to be partitioned into two fields: \( X \) and \( Y \), with state vectors \( u_x \) and \( u_y \), respectively, that interact as sketched in Figure 10.1. The partitioned equations of motion are

\[
\begin{bmatrix}
M_{xx} & 0 \\
0 & M_{yy}
\end{bmatrix}
\begin{bmatrix}
\ddot{u}_x \\
\ddot{u}_y
\end{bmatrix}
+ \begin{bmatrix}
C_{xx} & C_{xy} \\
C_{yx} & C_{yy}
\end{bmatrix}
\begin{bmatrix}
\dot{u}_x \\
\dot{u}_y
\end{bmatrix}
+ \begin{bmatrix}
K_{xx} & K_{xy} \\
K_{yx} & K_{yy}
\end{bmatrix}
\begin{bmatrix}
u_x \\
u_y
\end{bmatrix}
= \begin{bmatrix}
f_x \\
f_y
\end{bmatrix}. 
\]

(10.2)

Note that a block diagonal mass matrix is assumed. In other words, coupling mass terms \( M_{xy} \) and \( M_{yx} \) must vanish. This can always be accomplished by linking the coupled structures through forces, or via massless connection elements. In addition, both \( M_{xx} \) and \( M_{yy} \) are assumed to be symmetric and positive definite. The stiffness matrix \( K \) is assumed to be symmetric and nonnegative.

§10.2.1. Staggered Partition

A staggered solution scheme with prediction on \( u_x \) is obtained through the following rearrangement of (10.2):

\[
\begin{bmatrix}
M_{xx} & 0 \\
0 & M_{yy}
\end{bmatrix}
\begin{bmatrix}
\ddot{u}_x \\
\ddot{u}_y
\end{bmatrix}
+ \begin{bmatrix}
C_{xx} & 0 \\
0 & C_{yy}
\end{bmatrix}
\begin{bmatrix}
\dot{u}_x \\
\dot{u}_y
\end{bmatrix}
+ \begin{bmatrix}
K_{xx} & K_{xy} \\
K_{yx} & K_{yy}
\end{bmatrix}
\begin{bmatrix}
u_x \\
u_y
\end{bmatrix}
= \begin{bmatrix}
f_x - C_{xy} \dot{u}_P^x \\
f_y - K_{xy} u_P^x
\end{bmatrix}. 
\]

(10.3)

Because structural systems are usually lightly damped, further investigation of the stability of the staggered procedure will focus on the undamped system

\[
\begin{bmatrix}
M_{xx} & 0 \\
0 & M_{yy}
\end{bmatrix}
\begin{bmatrix}
\ddot{u}_x \\
\ddot{u}_y
\end{bmatrix}
+ \begin{bmatrix}
K_{xx} & 0 \\
K_{yx} & K_{yy}
\end{bmatrix}
\begin{bmatrix}
u_x \\
u_y
\end{bmatrix}
= \begin{bmatrix}
f_x - K_{xy} u_P^x \\
f_y
\end{bmatrix}. 
\]

(10.4)

§10.2.2. The Test System

To construct a test system for stability analysis of (10.4), the applied forcing terms in are dropped, and the following vibration eigenproblems are considered:

\[
M_{xi} z_{xi} = \omega_{xi}^2 K_{xi} z_{xi}, \quad M_{yi} z_{yj} = \omega_{yj}^2 K_{yi} z_{yj}, 
\]

(10.5)

where indices \( i \) and \( j \) run over the number of freedoms in partitions \( X \) and \( Y \), respectively.

The eigenproblems (10.5) are called the uncoupled eigensystems since they arise if the coupling terms \( K_{xy} \) and \( K_{yx} \) vanish. The uncoupled squared frequencies \( \omega_{xi}^2 \) and \( \omega_{yj}^2 \) are zero or positive.
If $\omega^2_{ij} = 0$, $z_{xi}$ is a rigid-body mode for the uncoupled partition $X$, and likewise for $Y$. The eigenvectors are normalized to unit generalized mass: $v^T_{xi} M_{xx} v_{xi} = 1$ and $v^T_{yj} M_{yy} v_{yj} = 1$.

We pass to modal coordinates through

$$u_i = V_i x, \quad u_y = V_y y.$$  \hspace{1cm} (10.6)

where $x$ and $y$ collect modal amplitudes. These are substituted in (10.4) and the equations premultiplied by $V_i^T$ and $V_y^T$ to produce a modal system of the form

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \dot{x} + \begin{bmatrix} \Omega_{xx} & 0 \\ \Omega_{yx} & \Omega_{yy} \end{bmatrix} \begin{bmatrix} u_x \\ u_y \end{bmatrix} = \begin{bmatrix} -\Omega_{xy} y^p \\ 0 \end{bmatrix}. \hspace{1cm} (10.7)$$

Here $\Omega_{xx}$ and $\Omega_{yy}$ are diagonal matrices formed by the $\omega^2_{ii}$ and $\omega^2_{jj}$ squared frequencies, respectively. On the other hand $\Omega_{yx}$ and $\Omega_{xy} = \Omega_{yx}^T$, are generally full matrices, since the modes of the structure partitions are not necessarily modes of the coupled system. For convenience the $i,j$th entry of $\Omega_{xy}$ will be written $\Omega_{xij} = k_{ij} \omega_i \omega_j$, in which $|k_{ij}| \leq 1$ is the coupling coefficient between partition modes $i$ and $j$.\(^{13}\) Now if (10.8) is stable for any mode pair $\{x_i, y_j\}$ so is the response of (10.4) since the response in physical coordinates $x$ and $y$ is a discrete linear combination as per (10.6). We therefore can restrict consideration to a typical mode pair. Supressing the indices $i$ and $j$ for brevity we arrive at the test system

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \dot{x} + \begin{bmatrix} \omega^2_x & 0 \\ k \omega_x \omega_y & \omega^2_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -k \omega_x \omega_y y^p \\ 0 \end{bmatrix}. \hspace{1cm} (10.8)$$

in which $\omega_x \geq 0$, $\omega_y \geq 0$ and $|k| \leq 1$. The last condition comes from the assumption that $K$ is nonnegative.

\(^{13}\) If $\omega_i = 0$ or $\omega_j = 0$ then $\Omega_{xij} = 0$; else $K$ would be indefinite. In that case one can take $k_{ij} = 0$ for convenience.
§10.3. Stability Analysis

The stability analysis of the test system (10.8) confirms the general results of Chapter 9. For specificity the Trapezoidal Rule defined by the operators \( \rho = 1 - 1/z, \sigma = 1 + 1/z \) and \( \delta = \frac{1}{2}h \) was used on both \( u \) and \( v \). For the test system treated with TR the optimal stability polynomial is

\[
\det(P^*(z)) = \det(\rho^2M + \delta^2\sigma^2K + (e^* - 1)\delta^2\xi K) = \frac{1}{16\xi^4}\left[16(-1+z)^4 + h^4(1+z)^2\right]
\]

\[(1+z)^2 - 4\kappa^2)\omega^2_1\omega^2_2 + h^2(4(z^2 - 1)^2\omega^2_1 + 4(z - 1)^2(1+z)^2\omega^2_2)\]

(10.9)

Here \( e^* \) is the optimal predictor as listed in Table 9.2, and \( \xi \) is \( \rho, \rho\sigma, \sigma \) and \( \sigma^2 \) for paths (0'), (0), (1) and (2), respectively. To convert to an amplification polynomial, simply rationalize via multiplication by \( z^4 \). The Hurwitz polynomial, obtained on replacing \( s = (z - 1)/(z + 1) \) and taking its numerator, is

\[
P(s) = h^4\omega^2_1\omega^2_2 - h^4\kappa^2\omega^2_1\omega^2_2 + (4h^2\omega^2_1 + 4h^2\omega^2_2 + h^4\kappa^2\omega^2_1\omega^2_2)s^2 + 16s^4.
\]

(10.10)

The only nontrivial stability condition is

\[
h^4\omega^2_1\omega^2_2 - h^4\kappa^2\omega^2_1\omega^2_2 \geq 0.
\]

(10.11)

This is satisfied for any \( h \) if \( |\kappa| \leq 1 \), which is guaranteed from the construction of the test system and the definiteness conditions on the original governing equations.

The symbolic verification that indeed (10.9) is obtained according to theory is done with the script listed in Figure 11.2, which loops over four computational paths and adjusts the predictor operator accordingly. The script calls auxiliary modules described in Chapter 4. The characteristic polynomial is compared against (10.9) and all paths verify coalescence.

In summary, the treatment of structure-structure interaction of undamped or Rayleigh damped systems by staggered methods can be considered as being in good shape. Unconditional stability is attained by carefully chosing the predictor in accordance to the computational path. (As remarked before, equivalent stability does not mean equivalent accuracy; that property has to be analyzed later by the method of modified equations outlined in Chapter 5.)

For generally damped structural systems no general theory is available, and the search for stable partitioned procedures still has to proceed on a case by case basis. However the results of Chapter 9 can be used as a starting point. This suggestion is in fact used in the next Chapter, which deals with control-structure interaction.

Notes and Bibliography

Most of the material in this Chapter has been extracted from portions of an unpublished report [18], as well as from [20]. At that time stability tests were verified numerically since no computer algebra systems were available.

11. CONTROL-STRUCTURE INTERACTION

The second example case concerns the synthesis of a staggered solution procedure for the interaction of a structure with an external force field proportional to velocities and displacements. This scenario naturally fits the structure-control interaction problem, but it has additional applications such as interaction of a structure with nonconservative forces such as dynamic fluid pressure.
§11.1. Governing Equations

The governing CSI equations to be studied are

\[ M \ddot{u} + C \dot{u} + Ku = f_A(t) + f_C(t) \]
\[ f_C(t) = -A \dot{u} - Bu. \]  

(11.1)

Here \( u \) is the structural state displacement vector, \( M, C \) and \( K \) are the structural mass, damping and stiffness matrices, respectively; \( f_A(t) \) and \( f_C(t) \) are the applied and control-interaction forces, respectively, and \( A \) and \( B \) are CSI matrices that depend on the implementation of the control system. The minus signs in the second of (11.1) are for convenience. We note that \( M, C \) and \( K \) are nonnegative-definite symmetric matrices, with \( M \) being usually positive definite. On the other hand \( A \) and \( B \) are square but generally unsymmetric.

Note that the governing equations can be immediately reduced to a single field upon elimination of the control force \( f_C(t) \):

\[ M \ddot{u} + (C + A) \dot{u} + (K + B)u = f_A(t). \]  

(11.2)

This form allows a fully implicit treatment. It is computationally inconvenient, however, because the insertion of \( A \) and \( B \) into the damping and stiffness terms of the original structural equations renders them unsymmetric. Furthermore the sparseness of those matrices may be affected. This may require substantial changes to the structural analyzer.

Figure 11.1 diagrams control-structure interaction as a two-field partitioned analysis process.

§11.1.1. Formulation as a Two Field Problem

In what follows it will be assumed that

(i) \( M \) is positive definite. Consequently \( M \) and \( K \) can be simultaneously diagonalized.

(ii) Structural damping is light and proportional: \( C = c_1M + c_2K \), where \( c_1 \) and \( c_2 \) are nonnegative and small in the sense that the effective damping coefficient is \( \ll 1 \).

It follows from assumptions (i) and (ii) that \( M, C \) and \( K \) can be simultaneously diagonalized, which greatly simplifies the formulation of test equations below. Equations (11.1) can be restated as a two field coupled problem by renaming the state vector appearing in the control equation as \( s \), and appending it as an auxiliary equation:

\[ M \ddot{u} + C \dot{u} + Ku = f_S(t) - A \dot{s} - Bs \]
\[ Ms = Mu \]

(11.3)

Note that the second equation has been scaled by the mass matrix, which is permissible since \( M \) is assumed to be positive definite. This scaling will simplify the test equation derived below.

This formulation is computationally attractive in that the control appears to the structure analyzer as a state dependent force term. In practice the computation of \( A \dot{u} + B u \) can be done by a separate software module that receives displacements and velocities and returns a control force. Handling of retarded (delays) effect in software becomes straightforward.

We now study the solution of (11.3) by a partitioned procedure. The first task is to derive a test system.
§11.1.2. Transformation to Modal Coordinates

The construction of a test system is carried out by passing to modal coordinates $x$ and $y$ through the transformations

$$ u = Vx, \quad y = Vs, \quad \text{(11.4)} $$

The modal matrix $V$ is formed by lining up the mass-normalized in vacuo structural modes $v_i$ as columns. We note that $V^T MV = I$ and $V^T K V = \Omega^2$, where $\Omega^2$ is the diagonal matrix formed by the in vacuo squared frequencies $\omega_i^2 = v_i^T K v_i$. Recalling that proportional structural damping is assumed, the transformed system in modal coordinates is

$$ \ddot{x} + 2P\Omega \dot{x} + \Omega^2 x = g_i(t) - G \dot{y} - Hy, \quad y = x. \quad \text{(11.5)} $$

Here $P = \text{diag}(\zeta_i)$ is a diagonal matrix of modal damping coefficients $\zeta_i \geq 0$, $g_i(t) = V^T f_A(t)$ is a vector of generalized applied forces, $G = V^T A V$ and $H = V^T B V$ are generalized CSI matrices. The last equation follows from $V^T MV = I$.

Since $G$ and $H$ are generally fully populated matrices, (11.5) shows that every mode pair: $\{v_i, v_j\}$ is generally coupled. So the test system for each $\{i, j\}$ mode pair takes the form:

$$ \ddot{x}_i + 2\zeta_i \omega_i \dot{x}_i + \omega_i^2 x_i = g_i(t) - G_{ij} \dot{y}_j - H_{ij} y_j, \quad y_j = x_i. \quad \text{(11.6)} $$

For further analysis if is assumed that $\omega_i > 0$, which excludes rigid body modes. If so the coefficients on the right are expressible as $G_{ij} = 2\mu_{ij} \omega_i$ and $H_{ij} = \mu_{ij} \omega_i^2$, where $\mu_{ij}$ and $v_{ij}$ are dimensionless coefficients. The above equation becomes

$$ \ddot{x}_i + 2\zeta_i \omega_i \dot{x}_i + \omega_i^2 x_i = g_i(t) - 2\mu_{ij} \omega_i \dot{y}_j - v_{ij} \omega_i^2 y_j, \quad y_j = x_i. \quad \text{(11.7)} $$

This equation must be physically stable for any $\{i, j\}$ pair. This happens if

$$ \mu_{ij} \geq -2\zeta_i, \quad v_{ij} \geq -1. \quad \text{(11.8)} $$

which must be satisfied as part of the design of the control system.

Next, drop subscripts $\{i, j\}$ as well as the applied forcing term, which does not affect the stability behavior. The test system becomes

$$ \ddot{x} + 2\zeta \omega \dot{x} + \omega^2 x = -2\mu \omega \dot{y} - v \omega^2 y, \quad y = x. \quad \text{(11.9)} $$

with $\mu$ and $v$ satisfying (11.8). If $\mu = v = 0$, modes $\{v_i, v_j\}$ are said to be uncoupled, and coupled otherwise.

§11.2. Stability Analysis

Applying a predictor on $\dot{y}$ and $y$, the test system FSICSITestSystem can be placed in matrix form as

$$ \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \dot{y} \end{bmatrix} + \begin{bmatrix} 2\zeta \omega & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} + \begin{bmatrix} \omega^2 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 & 2\mu \omega \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y}^P \end{bmatrix} + \begin{bmatrix} 0 & v \omega^2 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y^P \end{bmatrix}. \quad \text{(11.10)} $$

This system varies significantly from that studied in Chapter 9: the mass matrix is not positive definite, and the damping and stiffness matrices are unsymmetric. Consequently the theory of optimal characteristic equations and optimal predictors is not valid. The theory of equivalent
produces stability polynomials for BE using algebraic partitioning, zero structural the last solution (BE), which is generated by stability for certain ranges of stable LMS integrators, which contain no historical derivatives, showed promise in attaining A-Rule, were unable to achieve A-stability for any physically relevant choice of setting predictors holds. For simplicity the structural damping was dropped in the investigations by setting $\zeta = 0$ since this assumption is conservative.

Numerical experiments with integrators with historical derivatives, such as the Trapezoidal Rule, were unable to achieve A-stability for any physically relevant choice of $\mu$ and $\nu$. Only stiffly stable LMS integrators, which contain no historical derivatives, showed promise in attaining A-stability for certain ranges of $\mu$ and $\nu$. Of these the simplest and most stable is Backward Euler (BE), which is generated by $\rho = 1 - 1/z$, $\sigma = 1$ and $\delta = h$. The Mathematica script of Figure 11.2 produces stability polynomials for BE using algebraic partitioning, zero structural the last solution as predictor: $u^P = u_n$ or $e = 1 - \rho$. It provided the following Routh-Hurwitz polynomials for paths $(0')$ and $(0)$:

$$ P(s) = h^3 \omega(2\mu + h(1 + \nu)\omega) + 2h^3 \omega(2\mu + h(1 + \nu)\omega)s + h^3(4 - 6h\mu\omega + h^2(1 - 3\nu)\omega^2)s^2, $$

(11.11)

whereas for paths $(1)$ and $(2)$ it gives

$$ P(s) = h^3 \omega(2\mu + h(1 + \nu)\omega) + 2h^4 \omega^2s + h^2(4 - 2h\mu\omega + h^2(\nu - 1)\omega^2)s^2. $$

(11.12)

Because this polynomial is of order 2 it is sufficient to require that the 3 coefficients be positive. For nonnegative $\mu$ and $\nu$ only the last coefficient can become negative. A straightforward analysis implemented in the script of Figure 11.2 provides the following A-stable stability limits:

$$ \text{Paths (0),(0): } v = \frac{1}{3} - \frac{3}{4}\mu^2, $$

$$ \text{Paths (1),(2): } v = 1 - \frac{1}{4}\mu^2. $$

(11.13)

These curves are plotted in Figure 11.3. The A-stable regions lie under the two curves.
The influence of the computational paths on stability should be noted: the stability region for paths (1) and (2) is about 3 times larger than the region for (0') and (0). The difference cannot be compensated in this case by equivalent predictors.

The stability range is sufficient to take care of \( \{\mu, \nu\} \) pairs that arise in practical CSI problems, as those typically are on the order of 5% to 20%.

Of course BE is very inaccurate, injecting significant amount of artificial damping. Two other one-leg A-stable LMS methods: Gear 2-step and Park 3-step, provide second order accuracy although some degradation of the stability regions can be expected. The stability regions of those A-stable integrators has not been yet investigated on the test system.

Notes and Bibliography

The construction of the CSI test system is similar to one developed in the report [18] for a simple model of structure-aeroelastic interaction. The symbolic stability analysis of Backward Euler for control-structure interaction is new.

CONCLUSIONS

A major goal of these lectures is to call attention to the widespread availability of computer algebra systems. These provide a welcome helping hand in facilitating the synthesis of partitioned analysis procedures to handle more complex interacting systems. Analytical calculations that were prohibitively difficult by hand when partitioned methods were created in the mid-1970s, can be now done in minutes. Another key point emphasized throughout the need for developing balanced test systems. These should be neither too simple (thus leaving out important physics) nor too complex (thus obscuring primary behavior in a forest of details). The two examples presented in Chapters 10 and 11 illustrate how this important stage in method synthesis can be addressed.

Acknowledgments

The preparation of this paper was partially supported by the National Science Foundation under grant High-Fidelity Simulations for Heterogeneous Civil and Mechanical Systems, CMS-0219422.

References


Grcar, J., Birthday of Modern Numerical Analysis, sepp@california.sandia.gov, 2003

References:


