Chapter 2

The Concept Behind MATSTAB

Contents

2.1 The Central Equations

MATSTAB combines well known methods from different fields in a new manner.

- Sets of linearized equations
- Sparse matrix techniques
- Frequency domain calculations (eigenvalues\eigenvectors)

The linearization of the equations reduces the computational workload while giving access to many sophisticated methods of linear systems.

The sparse matrix techniques overcome the huge memory and time demands of an algorithm that solves up to half a million equations simultaneously (≈200MB instead of ≈10GB).

The frequency domain representation allows to calculate eigenvalues and eigenvectors which contain much more information than a time series from a time domain approach. Therefore not only more information is gained, but the results can also be visualized in a very interesting way.

The combination of these methods makes it feasible to model a nuclear reactor in detail. Each of the roughly 650 fuel assemblies is represented with 25 axial nodes. Each node contains up to 20 equations, hence creating a system with roughly $650 * 25 * 20 = 325'000$ equations in the core alone.

2.1.1 Linearization

The dynamic behavior of boiling water reactors can be assumed to be linear for small deviations around steady operating conditions. This makes it possible to study stability of BWRs using locally linearized equations. The reactor is described with an appropriate form of the governing equations as well as the equations needed to close the system. The n_x state variables which are described by differential equations are represented by the vector $\mathbf{x}$ while the n_u variables described by algebraic equations are represented by the vector $\mathbf{u}$. To be consistent with the model description of RAMONA [114], the time is denoted with τ. The continuous-time dynamical system consists, therefore, of a set of n_x first-order differential equations of the form

$$\frac{d}{d\tau}\mathbf{x}(\tau) = \mathbf{f}(\mathbf{x}(\tau), \mathbf{u}(\tau)) \tag{2.1}$$

and n_u algebraic equations of the form

$$\mathbf{g}(\mathbf{x}(\tau), \mathbf{u}(\tau)) = \mathbf{0} \tag{2.2}$$

where

$$\mathbf{x}(\tau) = \begin{pmatrix} x_1(\tau) \\ \vdots \\ x_{n_x}(\tau) \end{pmatrix} \quad \text{and} \quad \mathbf{u}(\tau) = \begin{pmatrix} u_1(\tau) \\ \vdots \\ u_{n_u}(\tau) \end{pmatrix} \tag{2.3}$$

Examples of variables of the type $x_i(\tau)$ are mixture energy, steam mass and fuel temperature. Examples of variables of the type $u_i(\tau)$ are mixture volumetric flux, slip and power generation rate. Table 2.1 shows a complete list of the state variables/equations. A detailed description about the state variables is given in the next chapter and in Appendix A.

Symbol	Variable	Algebraic	Differential	Remarks
Thermal-Hydraulics				
P	Pressure		1	
$u_m\rho_m$	Mixture Energy		1	
m_g	Steam Mass		1	
j_m	Mixture Volumetric Flux	1		
S	Slip	1		
w_g	Phasic Velocity	1		
W_g, W_l	Mass Flow Rate	2		
Γ_v	Vapor Generation Rate	1		
q'_w	Linear Heat Generation Rate	1		
t_l	Liquid Temperature	1		
t_w	Wall Temperature	1		
Neutronics / Power Generation				
φ_1	Fast Flux	1		
φ_2	Thermal Flux	(1)		integrated into φ_1
C_d	Precursors	(6)		integrated into φ_1
q'''	Power Generation Rate	1		
Thermal Conduction				
t_f	Fuel Temperature		4	
t_c	Cladding Temperature		2	
Total		11	9	

Table 2.1: Differential and Algebraic Equations used in MATSTAB

The equations 2.1 and 2.2 build together the set of equations which has to be linearized.

$$\left|\begin{array}{rcl} \frac{d}{d\tau}\mathbf{x}(\tau) & = & \mathbf{f}(\mathbf{x}(\tau),\mathbf{u}(\tau)) \\ \mathbf{0} & = & \mathbf{g}(\mathbf{x}(\tau),\mathbf{u}(\tau)) \end{array}\right| \tag{2.4}$$

$\mathbf{x}(\tau)$ in (2.4) can be substituted by $\mathbf{x}(\tau)$= $\mathbf{x}_0$+$\Delta\mathbf{x}(\tau)$, where $\mathbf{x}_0$ is the steady state value and $\Delta\mathbf{x}$ is a small perturbation around $\mathbf{x}_0$. Similarly $\mathbf{u}(\tau)$= $\mathbf{u}_0$+$\Delta\mathbf{u}(\tau)$. The Taylor-series of $\mathbf{f}(\mathbf{x},\mathbf{u})$ and $\mathbf{g}(\mathbf{x},\mathbf{u})$ yield

$$\frac{d}{d\tau}\mathbf{x}_0 + \frac{d}{d\tau}\Delta\mathbf{x} = \mathbf{f}_0 + \frac{\partial\mathbf{f}(\mathbf{x},\mathbf{u})}{\partial\mathbf{x}}\Delta\mathbf{x} + \frac{\partial\mathbf{f}(\mathbf{x},\mathbf{u})}{\partial\mathbf{u}}\Delta\mathbf{u} + O\left(\Delta\mathbf{x}^2,\Delta\mathbf{u}^2,\Delta\mathbf{x}\Delta\mathbf{u}\right) \tag{2.5}$$

$$\mathbf{0} = \mathbf{g}_0 + \frac{\partial\mathbf{g}(\mathbf{x},\mathbf{u})}{\partial\mathbf{x}}\Delta\mathbf{x} + \frac{\partial\mathbf{g}(\mathbf{x},\mathbf{u})}{\partial\mathbf{u}}\Delta\mathbf{u} + O\left(\Delta\mathbf{x}^2,\Delta\mathbf{u}^2,\Delta\mathbf{x}\Delta\mathbf{u}\right) \tag{2.6}$$

Neglecting second and higher order terms and using the fact that $\mathbf{f}_0 = \mathbf{f}(\mathbf{x}_0(\tau),\mathbf{u}_0(\tau)) = 0$ for the steady state under investigation leaves us - for the differential equations - with

$$\begin{aligned} \frac{d}{d\tau}\Delta\mathbf{x} &= \frac{\partial \mathbf{f}(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}\Delta\mathbf{x} + \frac{\partial \mathbf{f}(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}}\Delta\mathbf{u} \\ &\equiv \mathbf{A}\Delta\mathbf{x} + \mathbf{B}\Delta\mathbf{u} \end{aligned} \tag{2.7}$$

where

$$\mathbf{A} = [a_{ij}] = \left[\frac{\partial f_i}{\partial x_j}\right], \quad \mathbf{B} = [b_{ij}] = \left[\frac{\partial f_i}{\partial u_j}\right] \tag{2.8}$$

and for the algebraic equations with

$$\begin{aligned} \mathbf{0} &= \frac{\partial \mathbf{g}(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}\Delta\mathbf{x} + \frac{\partial \mathbf{g}(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}}\Delta\mathbf{u} \\ &\equiv \mathbf{C}\Delta\mathbf{x} + \mathbf{D}\Delta\mathbf{u} \end{aligned} \tag{2.9}$$

where

$$\mathbf{C} = [c_{ij}] = \left[\frac{\partial g_i}{\partial x_j}\right], \quad \mathbf{D} = [d_{ij}] = \left[\frac{\partial g_i}{\partial u_j}\right] \tag{2.10}$$

Solving 2.9 for $\Delta\mathbf{u}$ yields

$$\Delta\mathbf{u} = -\mathbf{D}^{-1}\mathbf{C}\Delta\mathbf{x} \tag{2.11}$$

and inserting the result into 2.7

$$\frac{d}{d\tau}\Delta\mathbf{x} = \left(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)\Delta\mathbf{x} \tag{2.12}$$

is obtained. This reduces the system 2.4 to n_x equations. After introducing the system matrix $\mathbf{A}_s = \left(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)$, equation 2.12 becomes the set of differential equations which is the center of MATSTAB

$$\frac{d}{d\tau}\Delta\mathbf{x} = \mathbf{A}_s\Delta\mathbf{x} \tag{2.13}$$

From an analytical point of view, equation 2.13 is a compact, linearized reformulation of the original equation set 2.4. However, from a practical point of view, a new problem occurs. Even though the matrices **A**, **B**, **C** and **D** are very sparse, the matrix $\mathbf{D}^{-1}$ is not (see Figures 4.4, 4.5 and 4.6). For fast execution, the sparsity of the matrix $\mathbf{A}_s$ is in general much more important than its size. Therefore MATSTAB does not apply equation 2.13 in an explicit but rather in an implicit manner. The specific algorithms used to solve equation 2.12 are outlined in Chapter 4. The main idea for overcoming the loss of sparsity - and hence the numerical problem - is to distinguish between state variables that are only coupled to a few neighboring nodes and state variables that are either coupled with all six neighboring nodes (e.g. neutron flux) or many hydraulic channels (e.g. mass flux).

2.1.2 Sparse Matrix Techniques

The idea behind sparse matrix techniques is very simple. Instead of storing a huge n by n matrix element for element and, therefore, using memory space for n^2, elements, one stores only the k nonzero elements and their position. For a full matrix where $k = n^2$ this scheme would use up to 3 times more memory, but for a sparse matrix where n is large and k is much closer to n than to n^2, the latter method is far more efficient.

This way to store a matrix is only useful, if basic linear operations like $\mathbf{C} = \mathbf{A} * \mathbf{B}$ and basic linear problems like $\mathbf{Ax} = \mathbf{b}$ can be calculated directly within the sparse format. The libraries with these 'sparse functions/algorithms' and their use is what is called sparse matrix technique. The programming environment MATLAB which was used to create MATSTAB provides a large and efficient base of functions and algorithms for 'sparse problems'.

2.1.3 Eigenvalues and Eigenvectors

Since the matrix $\mathbf{A}_s$ in 2.13 represents differential and algebraic equations, the standard eigenvalue equation

$$\mathbf{A}_s\mathbf{e}_i = \lambda_i\mathbf{e}_i \tag{2.14}$$

does not apply. Instead the generalized eigenvalue equation must be used [30]:

$$\mathbf{A}_s\mathbf{e}_i = \lambda_i\mathbf{B}\mathbf{e}_i \tag{2.15}$$

where **B** is a diagonal but singular [nxn] matrix of the form

$$\mathbf{B} = \begin{bmatrix} 1 & 0 & & \dots & & 0 \\ 0 & \ddots & & & & \\ & & 1 & & & \\ \vdots & & & 0 & & \vdots \\ & & & & \ddots & \\ 0 & & & \dots & & 0 \end{bmatrix} \begin{matrix} \left.\vphantom{\begin{matrix}1\\0\\\ddots\end{matrix}}\right\} \text{Part referring to differential equations} \\ \left.\vphantom{\begin{matrix}0\\\vdots\\0\end{matrix}}\right\} \text{Part referring to algebraic equations} \end{matrix}$$

Similar to the right eigenvectors $\mathbf{e}_i$ in 2.15, there are left eigenvectors $\mathbf{f}_i$ for $\mathbf{A}_s$

$$\mathbf{f}_i^T\mathbf{A}_s = \lambda_i\mathbf{f}_i^T\mathbf{B} \tag{2.16}$$

The matrices containing all left and right eigenvectors

$$\mathbf{E} = [\mathbf{e_i} \dots \mathbf{e_n}] \ , \quad \mathbf{F} = [\mathbf{f_i} \dots \mathbf{f_n}] \tag{2.17}$$

are related as follows

$$\mathbf{EF} = \mathbf{I} \tag{2.18}$$

The matrices **E** and **F** have one more interesting property.

$$\mathbf{F}\mathbf{A}_s\mathbf{E} = \Lambda = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{bmatrix} \tag{2.19}$$

This allows to rewrite equation 2.13 by multiplying with **F** from the left and inserting the unity matrix **EF**,

$$\mathbf{F}\frac{d}{d\tau}\Delta\mathbf{x}(\tau) = \mathbf{F}\mathbf{A}_s\mathbf{E}\mathbf{F}\Delta\mathbf{x}(\tau) \tag{2.20}$$

to obtain

$$\frac{d}{d\tau}\left(\mathbf{F}\Delta\mathbf{x}(\tau)\right) = \Lambda\left(\mathbf{F}\Delta\mathbf{x}(\tau)\right). \tag{2.21}$$

Solving the differential equation above for $\mathbf{F}\Delta\mathbf{x}(\tau)$ leads to

$$\mathbf{F}\Delta\mathbf{x}(\tau) = e^{\Lambda\tau}\mathbf{F}\Delta\mathbf{x}(0) \tag{2.22}$$

or, after multiplying with **E** from the left

$$\Delta\mathbf{x}(\tau) = \mathbf{E}e^{\Lambda\tau}\mathbf{F}\Delta\mathbf{x}(0) \tag{2.23}$$

Stating the implicit sums in the matrix notation, equation 2.23 becomes

$$\Delta\mathbf{x}(\tau) = \sum_{i=1}^{n} \mathbf{e_i} e^{\lambda_i\tau}\mathbf{f_i}^T\Delta\mathbf{x}(0) \equiv \sum_{i=1}^{n}\Delta\mathbf{x}_i(\tau). \tag{2.24}$$

The last equation defines the vector $\Delta\mathbf{x}_i(\tau)$. Note the difference between the vector $\Delta\mathbf{x}_i(\tau)$ in equation 2.24 and the scalar x_i in equation 2.3. The vector $\Delta\mathbf{x}_i(\tau)$ is the contribution of the ith mode to all states at time τ, while the scalar $x_i(\tau)$ is the value of the ith component of the state vector $\mathbf{x}(\tau)$ at time τ. Let us study the mode

$$\Delta\mathbf{x}_i(\tau) = \mathbf{e}_i e^{\lambda_i\tau}[\mathbf{f}_i^T\Delta\mathbf{x}_i(0)] \tag{2.25}$$

The following interpretation of the eigenvectors and eigenvalues can be made.

Since both $e^{\lambda_i\tau}$ and $\mathbf{f}_i^T\Delta\mathbf{x}(0)$ are complex numbers, the shape of the mode $\Delta\mathbf{x}_i(\tau)$ is entirely defined by $\mathbf{e}_i$. Therefore, the *right eigenvector* $\mathbf{e}_i$ describes the relative magnitude and phase of the participating states.

On the other hand, the *left eigenvector* $\mathbf{f}_i$ determines how the mode is excited by the initial condition. Note that if $\Delta\mathbf{x}_i(0) = k_i\ \mathbf{e}_i$ for some scalar k_i, then only the i-th mode is excited, since according to equation 2.18, $\mathbf{f}_i^T\mathbf{e}_j$ equals one for $i = j$ and zero for all other cases.

The dominating *eigenvalue* governs the time domain evolution of the mode, in particular if $\lambda_i = \sigma_i + j\omega_i$ with $j = \sqrt{-1}$ then $\Delta\mathbf{x}_i(\tau) = \mathbf{k}_i e^{\sigma_i\tau}(\cos\omega_i\tau + j\sin\omega_i\tau)$.

The *stability* is described by the decay ratio (DR), which is the ratio of two consecutive maxima of the impulse response of the oscillating variable (Figure 2.1).

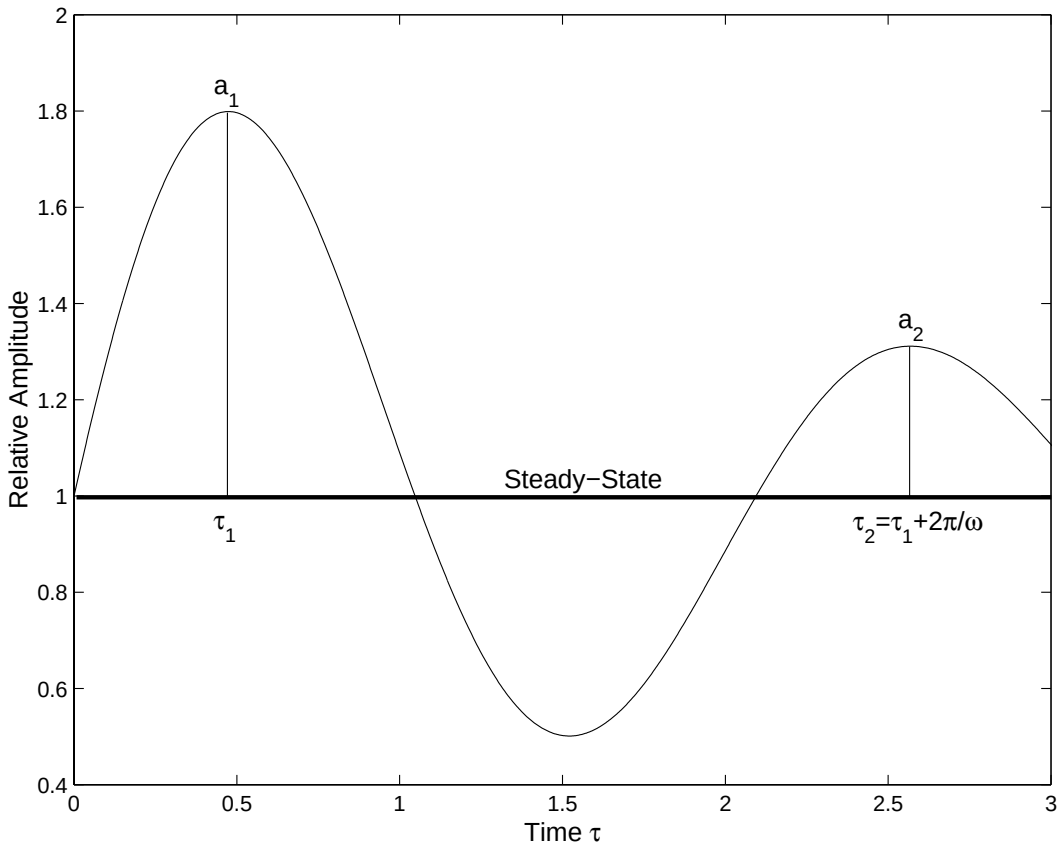

Figure 2.1: Definition of the Decay Ratio (DR)

$$DR = \frac{a_2}{a_1} = \frac{ke^{\sigma\tau_2}(\cos(\omega\tau_2)+j\sin(\omega\tau_2))}{ke^{\sigma\tau_1}(\cos(\omega\tau_1)+j\sin(\omega\tau_1))} \tag{2.26}$$

The times τ_1 and τ_2 are defined by the local maxima. Therefore the cosine of $\omega\tau_1$ and $\omega\tau_2$ equals one. The sine of $\omega\tau_1$ and $\omega\tau_2$ equals zero respectively.

$$DR = \frac{ke^{\sigma\tau_2}(1+j0)}{ke^{\sigma\tau_1}(1+j0)} = \frac{ke^{\sigma\tau_2}}{ke^{\sigma\tau_1}} \tag{2.27}$$

$$= e^{\sigma(\tau_2-\tau_1)} \tag{2.28}$$

$$= e^{2\pi\frac{\sigma}{\omega}} \tag{2.29}$$

The dominating eigenvalue and its two associated eigenvectors (left and right) are the main computational results of MATSTAB. They are the basis of all further investigations. Chapter 5 shows ways to visualize and interpret the results in a much more complete manner than just displaying decay ratios.

2.2 The Structure of MATSTAB

MATSTAB is a family of modules and functions written in the high-performance computing environment MATLAB [59]. The five parts of MATSTAB are (Figure 2.2):

- INPUT:
  The collection and processing of the steady state plant data
- PROBLEM:
  The calculation of the system matrix $\mathbf{A_s}$
- SOLUTION:
  The calculation of the dominating eigenvalue and eigenvectors
- SOLUTION+:
  The calculation of additional (regional) eigenvalues and eigenvectors
- VISUALIZATION
  Visualization and analysis of the calculated properties

These parts themselves contain numerous modules which have a well defined interface for handing over the data. The high degree of modularization makes it possible to replace or extend existing models without restructuring larger parts of the code.

2.2.1 Input Data

To get reliable calculation results, it is essential to have input data which describes the actual plant state sufficiently accurate. Most of the codes used today use extensive input desks which are normally generated by a time consuming and tedious manual procedure.

Some of this workload was reduced with the introduction of a conversion tool developed by Vattenfall [43]. This program uses data files from the online steady state core simulator which are used to calculate and oversee thermal margins and are therefore running in almost every nuclear power plant. These so called distribution and master files [51] contain all relevant dynamic data.

The master file contains the information about the reactor/fuel geometry and the reactor model. The file stays the same for at least one cycle. The distribution file contains the data calculated by the simulator and is therefore different for every operating point. In contrast to manual and semiautomatic procedures, MATSTAB is able to access distribution and master files directly and automatically. This import function is very much hardware independent, i.e. it is possible to read any (binary) distribution file on any common (DEC, SUN, HP, SGI, LINUX, WINDOWS NT, WINDOWS 9x) operating system and platform. The only input which needs to be generated by the user, are some geometric data (downcomer, steam dome, pumps etc.) of the ex-core system, which is plant, but not time dependent. These data are

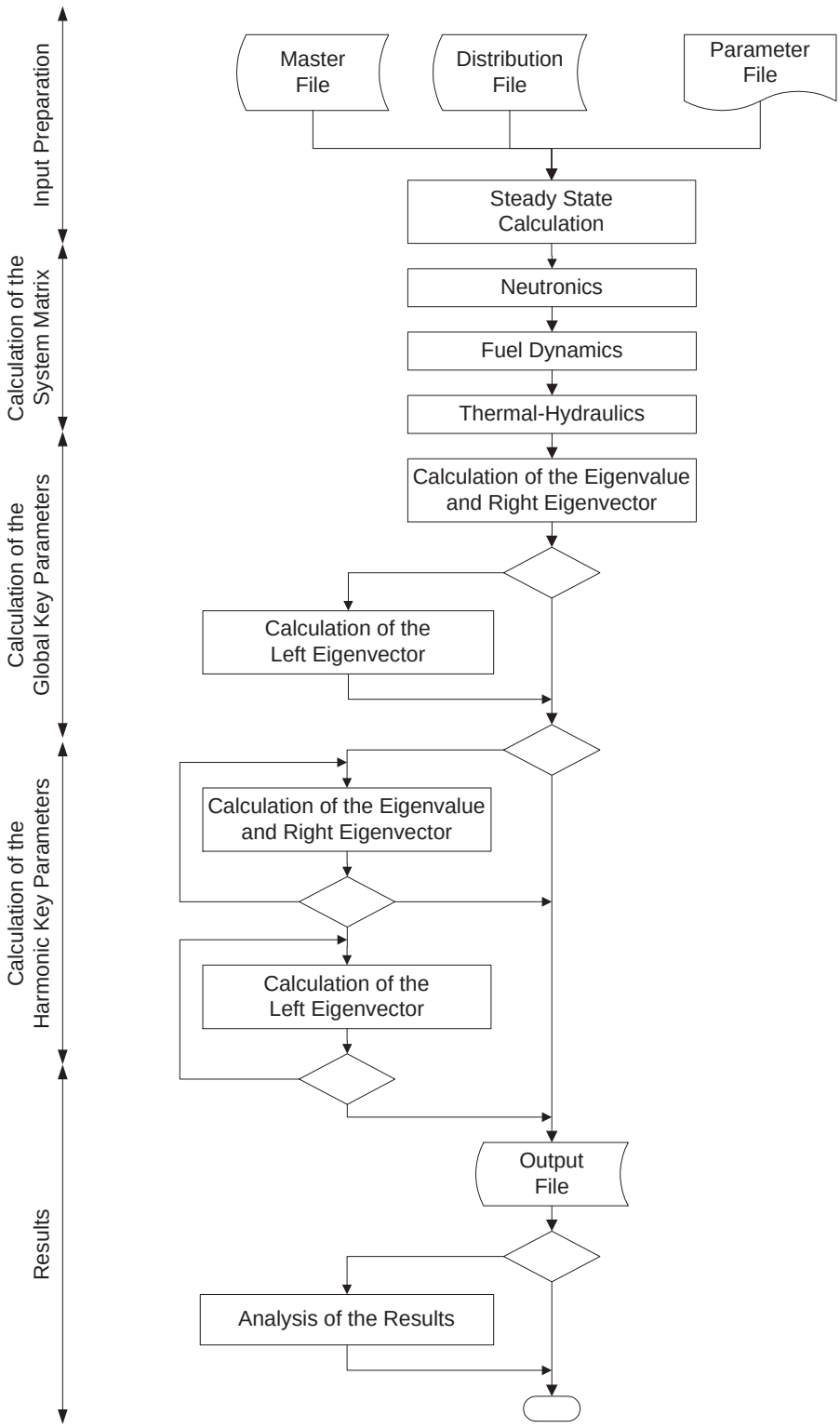

Figure 2.2: Structure of MATSTAB

supplied as a standard input file. In fact, this input file is a small subset of the RAMONA input file. To maintain some "backward" compatibility with RAMONA, it is possible, though not recommended, to use a normal RAMONA input desk combined with a RAMONA steady state calculation instead of the distribution and master files [92].

The input data from the online core simulator are self consistent, but from a simpler set of equations than the MATSTAB model. Therefore, it is not possible to use the distributions directly. To overcome this problem, the online steady state data is used as an input for a MATSTAB steady state module, generating the distributions which satisfy the set of equations used by MATSTAB.

2.2.2 Constructing the System Matrix $\mathbf{A}_s$

The system matrix $\mathbf{A}_s$ introduced in 2.13 is composed of the sub-matrices $\mathbf{A},\mathbf{B},\mathbf{C}$ and $\mathbf{D}$ defined in 2.7 and 2.9. The linearized form of the $n_x + n_u$ equations is evaluated with the steady state values $\mathbf{x_0}$ and $\mathbf{u_0}$ (defined in 2.5, 2.6) to obtain the matrix coefficients a_{ij}, b_{ij}, c_{ij} and d_{ij} in 2.10,2.8. For example, the algebraic equation for the void fraction is a function of the steam mass in a cell and the pressure.

$$\alpha = \alpha(m_g, P) = \frac{m_g}{\rho_g(P)V} \tag{2.30}$$

With the help of 2.10, this can be written as

$$\begin{aligned} \Delta\alpha &= c_{\alpha m_g}\Delta m_g + c_{\alpha P}\Delta P \\ &= \frac{\partial\alpha}{\partial m_g}\Delta m_g + \frac{\partial\alpha}{\partial P}\Delta P \\ &= \frac{1}{\rho_g(P)V}\Delta m_g - \frac{m_g}{\rho_g^2(P)V}\frac{\partial\rho_g}{\partial P}\Delta P \end{aligned} \tag{2.31}$$

The equation 2.31 which is valid for each node in the core, just with a different set of m_g, ρ_g and V, is actually programmed as a vector equation. $\Delta\alpha, \Delta m_g, \Delta P$ and ΔV are vectors of size 25 times the number of fuel assemblies.

After calculating $\mathbf{A},\mathbf{B},\mathbf{C}$ and $\mathbf{D}$, the rows and columns in the matrix $\mathbf{A}_s$ are scaled to eliminate numerical problems due to the large absolute difference between the coefficients in the neutronics section and the values in the thermal hydraulic section.

It remains to be said, that the matrix $\mathbf{A}_s$ does not only contain all the equations describing the physics in the core, but also all the equations describing the ex-core system. On the one hand, the thermal hydraulic equations that are valid in the core extend naturally to the outer part of the reactor, on the other hand, some equations are needed in addition, for example; pump equations, system pressure and the flow distribution model.

A detailed description of the model and all equations used follows in the next Chapter. A detailed description of the structure of the matrix $\mathbf{A}_s$ is given in Appendix A.

2.2.3 Eigenvalue Calculation

The eigenvalues and their eigenvectors are the main output of MATSTAB. Therefore, the implemented calculation methods and their reliability are crucial for valid results. The generalized eigenvalue problem

$$\mathbf{A}_s\mathbf{e} = \lambda\mathbf{B}\mathbf{e} \tag{2.15}$$

has a well known direct solution method [30]. However, the direct method contains steps which are not feasible for very large systems (e.g. 200'000x200'000) from a practical point of view (days of calculation time instead of minutes).

Fortunately, it is not necessary to calculate all eigenvalues and eigenvectors. MATSTAB uses iterative methods, which directly calculate the few dominant eigenvalues and their corresponding eigenvectors. This solution can be extended to the regional modes with the construction of a good starting guess, as will be described in Section 4.7.

As eigenvalue problems are very common in many engineering applications, there are a wide variety of codes available that work with iterative methods. Even though these codes are very sophisticated and advanced, as for example the functions available in MATLAB, they are not able to solve the specific system created by MATSTAB within reasonable time (minutes).

The method used in MATSTAB is not new, it is basically Newton's method, but it is extended and combined with subspace methods to take full advantage of the known and fixed structure of $\mathbf{A}_s$.

The detailed description of the methods follows in Chapter 4.

2.2.4 Visualization

The final numerical result, the eigenvalue/decay ratio, is interesting but does not give the complete picture. The real advantage of the frequency domain approach is not the faster execution time, but the additional information present in the eigenvectors.

Because the eigenvectors can be scaled in any way, the interesting aspect is their shape.

The right eigenvector describes the relative phase and magnitude of the participating states, the left eigenvector determines how the mode is exited by the initial conditions 2.25.