MY PRELIM

by

Lewis John Lloyd

A preliminary report submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Nuclear Engineering and Engineering Physics)

at the

UNIVERSITY OF WISCONSIN–MADISON

2012

ACKNOWLEDGMENTS

DOE and MUSC encourage fellows to publish reports and articles in scientific and engineering journals. The fellow must submit any articles, reports, or thesis to MUSC prior to submission for publication. All publications will show the joint affiliation of the fellow with the university and, if appropriate, with the laboratory in which the research was conducted, and should acknowledge fellowship support.

Fellowship support should be acknowledged in the following manner:

This research was performed under appointment to the Rickover Fellowship Program in Nuclear Engineering sponsored by Naval Reactors Division of the U.S. Department of Energy.

DISCARD THIS PAGE

TABLE OF CONTENTS

Page

DISCARD THIS PAGE

LIST OF TABLES

Table Page

Appendix
Table

DISCARD THIS PAGE

LIST OF FIGURES

Figure Page

Appendix
Figure

List of Algorithms

DISCARD THIS PAGE

NOMENCLATURE

ρ_v	Gas Density
ρ_l	Liquid Density
α_{st}	Steam Volume Fraction
α_l	Continuous Liquid Volume Fraction
α_e	Entrained Liquid Volume Fraction
C_1	Constant 1
V	Voltage
$	US Dollars

ABSTRACT

First paragraph

Second paragraph

Third paragraph

Chapter 1

Introduction

1.1 Motivation

Of primary use in the field of nuclear reactor safety analysis is simulation. The ability to predict the behaviour of reactors during off-normal events is the key to the licensing and the operation of nuclear power plants. Within the United States, this simulation capacity is provided by a relatively small number of main stream software suites, among which are the RELAP variants, COBRA variants, and MELCOR. While each of these software products varies in their models and implementations, the underlying numeric techniques and capabilities are similar. Traditionally, these system codes utilize a semi-implicit discritization scheme. In order to solve the resulting system of equations, the most common methodology is to take a single netwon step. The underlying numeric methods are first order methods.

The ability to use higher order methods is important. Since the development of the semi-implicit method, whose form was motivated by the limited computer resources available at the time, there have been great advances in both the methodology used to solve linear algebra problems and computer capabilities.

1.2 Objectives

The objective of this dissertation is the design, implementation, and evaluation of a practical non-linear solution framework for reactor safety systems codes. Specifically, an efficient and reliable solution methodology to the two-phase, three-field, fluid-dynamics and the solid-structure heat transfer system of coupled non-linear partial differential equations is sought. The specific

methodology should be capable of obtaining a consistent solution to the system of PDEs while also possessing.

Chapter 2

Fluid Mechanics

Of primary concern in reactor safety analysis is thermal-hydraulic behaviour of the nuclear core during off-normal conditions. In order to evaluate

2.1 Two Phase Flow

The physical behaviour of fluids within COBRA are represented by a set of differential equations. These governing equations are a collection of balance laws. Balance laws are statements of conservation that include external sources and sinks. The physical quantities being tracked by COBRA are mass, momentum, and energy. Since COBRA models two phase behaviour, the

Chapter 3

Mathematical Formulation

As mentioned earlier, most reactor safety-analysis codes depend upon three discrete sets of physics. These include, but are not necessarily limited to, two-phase hydrodynamics, heat-transfer between the fluid and a solid structure, and a nuclear power source. The different physical phenomena are represented by a system of PDEs and ODEs that constitute a set of balance laws for mass, momentum, and energy. This system encompasses different scales in both space and time.

3.1 Hydrodynamic Conservation Equations

Conserved Variables **Independent Variables**

$$\underline{\mathbf{q}} = \begin{bmatrix} \alpha\rho_g \\ \alpha\left(\rho_v + \rho_g\right) \\ \left(1-\alpha\right)\rho_l \\ \alpha_e\rho_l \\ \alpha\left(\rho_v + \rho_g\right)\underline{\mathbf{U}}_v \\ \left(1-\alpha\right)\rho_l\underline{\mathbf{U}}_l \\ \alpha_e\rho_l\underline{\mathbf{U}}_e \\ \alpha\left(\rho_v H_v + \rho_g H_g\right) \\ \left(1-\alpha\right)\rho_l H_l \end{bmatrix} \quad (3.1) \qquad \underline{\mathbf{x}} = \begin{bmatrix} \alpha \\ \alpha_e \\ P \\ \alpha P_g \\ \alpha\left(\rho_v + \rho_g\right)\underline{\mathbf{U}}_v \\ \left(1-\alpha\right)\rho_l\underline{\mathbf{U}}_l \\ \alpha_e\rho_l\underline{\mathbf{U}}_e \\ \alpha H_v \\ \left(1-\alpha\right)H_l \end{bmatrix} \quad (3.2)$$

Note that $\underline{\mathbf{q}} = f(\underline{\mathbf{x}})$.

3.1.1 Mass

$$\frac{\partial\left(\alpha\rho_v\right)}{\partial t} + \nabla\cdot\left(\alpha\rho_v\underline{\mathbf{U}}_v\right) = \Gamma + \nabla\cdot\underline{\mathbf{G}}_v^T \quad (3.3)$$

$$\frac{\partial\left(\alpha\rho_g\right)}{\partial t} + \nabla\cdot\left(\alpha\rho_g\underline{\mathbf{U}}_v\right) = \Gamma + \nabla\cdot\underline{\mathbf{G}}_g^T \quad (3.4)$$

$$\frac{\partial\left(\alpha_l\rho_l\right)}{\partial t} + \nabla\cdot\left(\alpha_l\rho_l\underline{\mathbf{U}}_l\right) = -\Gamma_l + \nabla\cdot\underline{\mathbf{G}}_l^T - S''' \quad (3.5)$$

$$\frac{\partial\left(\alpha_e\rho_l\right)}{\partial t} + \nabla\cdot\left(\alpha_e\rho_l\underline{\mathbf{U}}_e\right) = -\Gamma_e + S''' \quad (3.6)$$

3.1.2 Momentum

$$\frac{\partial\left(\alpha\rho_g\underline{\mathbf{U}}_g\right)}{\partial t}+\nabla\cdot\left(\alpha\rho_g\underline{\mathbf{U}}_g\underline{\mathbf{U}}_g\right)=$$
$$-\alpha\,\nabla P+\alpha\rho_g g-\tau_{wv}^{'''}-\tau_{I_{lv}}^{'''}-\tau_{I_{ev}}^{'''}+\Gamma_e U^{'}+\nabla\cdot\left(\alpha T_g^T\right) \tag{3.7}$$
$$\frac{\partial\left(\alpha_e\rho_l\underline{\mathbf{U}}_e\right)}{\partial t}+\nabla\cdot\left(\alpha_e\rho_l\underline{\mathbf{U}}_e\underline{\mathbf{U}}_e\right)=$$
$$-\alpha_e\,\nabla P+\alpha_e\rho_l g-\tau_{wl}^{'''}+\tau_{ev}^{'''}+\Gamma_e U^{'}+S^{'''}U^{'} \tag{3.8}$$
$$\frac{\partial\left(\alpha_l\rho_l\underline{\mathbf{U}}_l\right)}{\partial t}+\nabla\cdot\left(\alpha_l\rho_l\underline{\mathbf{U}}_l\underline{\mathbf{U}}_l\right)=$$
$$-\alpha_l\,\nabla P+\alpha_l\rho_l g-\tau_{wl}^{'''}+\tau_{lv}^{'''}-\Gamma_l U^{'}-S^{'''}U^{'}+\nabla\cdot\left(\alpha_l T_l^T\right) \tag{3.9}$$

$$\frac{\underline{\dot{\mathbf{m}}}_{i,j}^{\,n+1}-\underline{\dot{\mathbf{m}}}_{i,j}^{\,n}}{\Delta t}=\min(A_j,A_{j+1})\frac{1}{2}\left(\frac{\underline{\dot{\mathbf{m}}}_{i,j}^{\,n}}{A_j}+\frac{\underline{\dot{\mathbf{m}}}_{i,j+1}^{\,n}}{A_{j+1}}\right)\underline{\hat{\mathbf{U}}}_{i,j-\frac{1}{2}}^{\,n} \tag{3.10}$$

3.1.3 Energy

$$\frac{\partial\left(\alpha\rho_g H_g\right)}{\partial t}+\nabla\cdot\left(\alpha\rho_g H_g\underline{\mathbf{U}}_g\right)$$
$$=\Gamma H_v^{'}+q_{iv}+q_{gl}+Q_g^{'''}-\nabla\cdot\left(\alpha\underline{\mathbf{q}}_g^{\,T}\right) \tag{3.11}$$
$$\frac{\partial\left(\left(1-\alpha\right)\rho_l H_l\right)}{\partial t}+\nabla\cdot\left(\alpha_l\rho_l H_l\underline{\mathbf{U}}_l\right)+\nabla\cdot\left(\alpha_e\rho_l H_l\underline{\mathbf{U}}_e\right)=$$
$$-\Gamma H_l^{'}+q_{il}-q_{gl}+Q_l^{'''}-\nabla\cdot\left(\alpha_l\underline{\mathbf{q}}_l^{\,T}\right) \tag{3.12}$$

3.2 Discrete Hydrodynamic Equations

3.2.1 Axial Momentum

$$
\begin{aligned}
F_g(\underline{\mathbf{x}}^{n+1}) = \quad & \overbrace{E_g(\underline{\mathbf{x}}^{n})}^{\text{Purely explicit terms}} \quad \overbrace{- A_{mom,j}\Delta z_j \left[\alpha_g^n \frac{P_{J+1}^{n+1} - P_J^{n+1}}{\Delta z_j}\right]}^{\text{Semi-Implicit Pressure Term}} \\
& \overbrace{- A_{mom,j}\Delta z_j \left[\left.\frac{dP}{dz}\right|_{w,g}^{*} + \left.\frac{dP}{dz}\right|_{i,lg}^{*} + \left.\frac{dP}{dz}\right|_{i,eg}^{*}\right]}^{\text{Semi-Implicit Drag Terms}} \\
& + \overbrace{\mathcal{S}p_{g,j}^{n+1}}^{\text{Implicit Source Term}} \quad \overbrace{- \frac{\left(M_{g,j}^{n+1} - M_{g,j}^{n}\right)}{\Delta t}\Delta z}^{\text{Time rate of change}} \\
= \ & 0
\end{aligned}
\tag{3.13}
$$

The nonlinear functional associated with this equation is as follows:

$$
\overbrace{\underline{\mathbf{F}}(\underline{\mathbf{M}}^{n+1},\ P^{n+1})}^{\text{Nonlinear Functional}} = 0
$$

$$
= \underbrace{\underline{\mathbf{E}}(\underline{\mathbf{M}}^{n})}_{\text{Explict Terms}} + \underbrace{\underline{\mathbf{I}}(\underline{\mathbf{M}}^{n+1},\ P^{n+1})}_{\text{Implicit Terms}}
$$

$$
\underline{\mathbf{x}}^{n+1} = \begin{bmatrix} M_l^{n+1} \\ M_g^{n+1} \\ M_e^{n+1} \\ P^{n+1} \end{bmatrix}
$$

$$
\underline{\mathbf{F}}(\underline{\mathbf{x}}^{n+1}) = \underline{\mathbf{E}}(\underline{\mathbf{x}}^{n}) + \underline{\mathbf{I}}(\underline{\mathbf{x}}^{n+1})
$$

Now, this nonlinear functional is solved using a Newton Step.

$$\underline{\mathbf{F}}\left(\underline{\mathbf{x}}_{k+1}^{n+1}\right) = \underline{\mathbf{F}}\left(\underline{\mathbf{x}}_{k}^{n+1}\right) + \underline{\underline{\mathbf{J}}} \cdot \underline{\delta \mathbf{x}}_{k} = 0$$

$$\underline{\delta \mathbf{x}}_{k} = \underline{\mathbf{x}}_{k+1}^{n+1} - \underline{\mathbf{x}}_{k}^{n+1}$$

$$\underline{\mathbf{F}}\left(\underline{\mathbf{x}}_{k}^{n+1}\right) + \underline{\underline{\mathbf{J}}} \cdot \underline{\delta \mathbf{x}}_{k} = 0$$

$$\underline{\underline{\mathbf{J}}} \cdot \underline{\delta \mathbf{x}}_{k} = -\underline{\mathbf{F}}\left(\underline{\mathbf{x}}_{k}^{n+1}\right)$$

$$\underline{\underline{\mathbf{J}}}_{[:,1:3]} \cdot \underline{\delta \mathbf{M}}_{k} + \underline{\mathbf{J}}_{[:,4]} \cdot \underline{\delta \mathbf{P}}_{k} = -\underline{\mathbf{F}}\left(\underline{\mathbf{x}}_{k}^{n+1}\right)$$

$$\underline{\underline{\mathbf{J}}}_{[:,1:3]} \cdot \left[\underline{\mathbf{M}}_{k+1}^{n+1} - \underline{\mathbf{M}}_{k}^{n+1}\right] = -\underline{\mathbf{F}}\left(\underline{\mathbf{x}}_{k}^{n+1}\right) - \underline{\mathbf{J}}_{[:,4]} \cdot \underline{\delta \mathbf{P}}_{k}$$

$$\underline{\underline{\mathbf{J}}}_{[:,1:3]} \cdot \underline{\mathbf{M}}_{k+1}^{n+1} = \underline{\underline{\mathbf{J}}}_{[:,1:3]} \cdot \underline{\mathbf{M}}_{k}^{n+1} - \underline{\mathbf{F}}\left(\underline{\mathbf{x}}_{k}^{n+1}\right) - \underline{\mathbf{J}}_{[:,4]} \cdot \underline{\delta \mathbf{P}}_{k}$$

$$\underline{\mathbf{M}}_{k+1}^{n+1} = \underline{\mathbf{M}}_{k}^{n+1} - \left[\underline{\underline{\mathbf{J}}}_{[:,1:3]}\right]^{-1} \cdot \left[\underline{\mathbf{F}}\left(\underline{\mathbf{x}}_{k}^{n+1}\right) + \underline{\mathbf{J}}_{[:,4]} \cdot \underline{\delta \mathbf{P}}_{k}\right] \tag{3.14}$$

$$\underline{\underline{\mathbf{J}}}_{[1:3,1:3]} \equiv -\frac{2A_{mom}\Delta t}{\Delta z} \cdot \begin{bmatrix} K_{w,l}^{n} + \frac{K_{i,lg}^{n}}{\overline{\alpha\rho}_{l}^{n}} + \frac{1}{2} & -\frac{K_{i,lg}^{n}}{\overline{\alpha\rho}_{g}^{n}} & 0 \\ -\frac{K_{i,lg}^{n}}{\overline{\alpha\rho}_{l}^{n}} & K_{w,g}^{n} + \frac{K_{i,lg}^{n}}{\overline{\alpha\rho}_{g}^{n}} + \frac{K_{i,eg}^{n}}{\overline{\alpha\rho}_{g}^{n}} + \frac{1}{2} & -\frac{K_{i,eg}^{n}}{\overline{\alpha\rho}_{e}^{n}} \\ 0 & -\frac{K_{i,eg}^{n}}{\overline{\alpha\rho}_{g}^{n}} & K_{w,e}^{k} + \frac{K_{i,eg}^{n}}{\overline{\alpha\rho}_{e}^{n}} + \frac{1}{2} \end{bmatrix} \tag{3.15}$$

$$\underline{\underline{\mathbf{J}}}(\underline{\mathbf{x}}^{k})_{[1:3,4]} \equiv \frac{A_{mom}\Delta t}{\Delta z} \begin{bmatrix} <\alpha_{l}^{n}> \\ <\alpha_{g}^{n}> \\ <\alpha_{e}^{n}> \end{bmatrix} \tag{3.16}$$

3.2.2 Linearization

The semi-implicit method, with a single Newton Step linearizes about the old time value and solves for a single delta. This has several ramifications.

3.2.2.1 Wall Drag

The beginning equation to formulate the wall drag is shown in Eqn.

$$\left.\frac{dP}{dz}\right|_{\text{Fric},k} = \frac{f}{D_h}\frac{\overline{\rho}_k U_k^2}{2} \tag{3.17}$$

$$\left.\frac{dP}{dz}\right|_{Fric,k} = \frac{f}{D_h}\frac{M_k^2}{A_{mom}^2}\frac{1}{2\overline{\rho}_k}\frac{1}{\overline{\alpha}^2} \tag{3.18}$$

$$\left.\frac{dP}{dz}\right|_{k} = \frac{f}{D_h}\frac{M_k^2}{A_{mom}^2}\frac{1}{2\overline{\rho}_k}\frac{\alpha_b^2}{\overline{\alpha}^2} \tag{3.19}$$

$$K_{wf,k} = \frac{A_{mom}}{M_k}\left.\frac{dP}{dz}\right|_{k} \tag{3.20}$$

$$K_{wf,k} = \frac{A_{mom}}{M_k}\frac{f}{D_h}\frac{M_k^2}{A_{mom}^2}\frac{1}{2\overline{\rho}_k}\frac{\alpha_b^2}{\overline{\alpha}^2} \tag{3.21}$$

$$K_{wf,k} = \frac{f}{D_h}\frac{M_k}{A_{mom}}\frac{1}{2\overline{\rho}_k}\frac{\alpha_b^2}{\overline{\alpha}^2} \tag{3.22}$$

$$\left.\frac{dP}{dz}\right|_{\text{Form},k} = \frac{f}{D_h}\frac{\overline{\rho}_k U_k^2}{2} \tag{3.23}$$

$$\left.\frac{dP}{dz}\right|_{w,k} = \frac{f}{D_h}\frac{M_k^2}{A_{mom}^2}\frac{1}{2\overline{\alpha\rho}_k} \tag{3.24}$$

$$\left.\frac{dP}{dz}\right|_{k} = \frac{f}{D_h}\frac{M_k^2}{A_{mom}^2}\frac{1}{2\overline{\rho}_k}\frac{\alpha_b^2}{\overline{\alpha}^2} \tag{3.25}$$

$$K_{wf,k} = \frac{A_{mom}}{M_k}\left.\frac{dP}{dz}\right|_{k} \tag{3.26}$$

$$K_{wf,k} = \frac{A_{mom}}{M_k}\frac{f}{D_h}\frac{M_k^2}{A_{mom}^2}\frac{1}{2\overline{\rho}_k}\frac{\alpha_b^2}{\overline{\alpha}^2} \tag{3.27}$$

$$K_{wf,k} = \frac{f}{D_h}\frac{M_k}{A_{mom}}\frac{1}{2\overline{\rho}_k}\frac{\alpha_b^2}{\overline{\alpha}^2} \tag{3.28}$$

The Momentum Equations, outlined in Section **??** contain an implicit wall drag. This means that we need to evaluate the wall drag at the future time value. To do this in the semi-implicit methodology requires that we linearize the future value using Newton's Method. The current implementation uses the following derivation.

$$\left.\frac{dP}{dz}\right|_{w,k}^{*} = \left.\frac{dP}{dz}\right|_{w,k}^{n} + \left.\frac{d(\frac{dP}{dz}|_{w,k})}{dM_k}\right|^{n} \delta M_k \tag{3.29}$$

$$\frac{d(\frac{dP}{dz}|_{w,k})}{dM_k} = \frac{d}{dM_k}\left[\left(\frac{f}{D_h}+\frac{K_{form}}{\Delta z_j}\right)\left(\frac{M_k^2}{A_{mom}^2}\right)\left(\frac{1}{2\rho_k}\right)\right] \tag{3.30}$$

$$= 2\left(\frac{f}{D_h}+\frac{K_{form}}{\Delta z_j}\right)\left(\frac{M_k}{A_{mom}}\right)\left(\frac{1}{2\rho_k}\right) \tag{3.31}$$

$$= \left(\frac{2}{M_k}\right)\left(\left.\frac{dP}{dz}\right|_{w,k}\right) \tag{3.32}$$

$$\left.\frac{dP}{dz}\right|_{w,k}^{n+1} = \left.\frac{dP}{dz}\right|_{w,k}^{n}\left(1+\frac{2\delta M_k}{M_k}\right) \tag{3.33}$$

$$\left.\frac{dP}{dz}\right|_{w,k}^{n+1} = \frac{1}{M_k}\left.\frac{dP}{dz}\right|_{w,k}^{n}(M_k+2\delta M_k) \tag{3.34}$$

$$\left.\frac{dP}{dz}\right|_{w,k}^{n+1} = K_{wz,k}^{n}(M_k+2\delta M_k) \tag{3.35}$$

$$\tag{3.36}$$

Modifying this derivation to take into account the multiple Newton Steps results in the equations shown below. For brevity, the phase index, $_k$, is dropped from this derivation. Once the linearization starts, the index, $_k$, will refer to Newton iteration.

$$\left.\frac{dP}{dz}\right|_{w} = \left(\frac{f}{D_h}+\frac{K_{form}}{\Delta z_j}\right)\left(\frac{M^2}{A_{mom}^2}\right)\left(\frac{1}{2\rho}\right) \tag{3.37}$$

$$\left.\frac{dP}{dz}\right|_{w} = f(M) \tag{3.38}$$

$$f|_{M_k^{n+1}} = f|_{M_{k-1}^{n+1}} + \left.\frac{d\,f}{dM}\right|_{M_{k-1}^{n+1}} \delta M_{k-1}^{n+1} \tag{3.39}$$

$$\left.\frac{d\,f}{dM}\right|_{M_{k-1}^{n+1}} = \frac{2}{M_{k-1}^{n+1}}\,f|_{M_{k-1}^{n+1}} \tag{3.40}$$

$$K_{wz,k-1}^{n+1} = \frac{f|_{M_{k-1}^{n+1}}}{M_{k-1}^{n+1}} \tag{3.41}$$

$$f|_{M_k^{n+1}} = [K_{wz,k-1}^{n+1}]M_{k-1}^{n+1} + 2\left[K_{wz,k-1}^{n+1}\right]\delta M_{k-1}^{n+1} \tag{3.42}$$

$$M^* = M_k \frac{\alpha_b}{\overline{\alpha}} \tag{3.43}$$

$$G_k = \frac{M^*}{A_{mom}} = \frac{M_k \alpha_b}{A_{mom}\overline{\alpha}} \tag{3.44}$$

$$\left.\frac{dP}{dz}\right|_k = f \frac{1}{D_h} \frac{G_k^2}{2\overline{\rho}_k} \tag{3.45}$$

$$\left.\frac{dP}{dz}\right|_k = f \frac{1}{D_h} \frac{1}{2\overline{\rho}_k} \frac{M_k^2 \alpha_b^2}{A_{mom}^2 \overline{\alpha}^2} \tag{3.46}$$

$$K_i = \frac{1}{2} fi_{ia} \cdot coefd \cdot rvp \cdot urvl \cdot aintfh \cdot dxi \cdot \frac{1}{(\alpha^*)^3} \tag{3.47}$$

$$alrl = A_{mom}\overline{\alpha\rho} \tag{3.48}$$

$$C[2] = \frac{\Delta t}{\Delta x} \frac{K_i \Delta x}{A_{mom}\overline{\alpha\rho}} \tag{3.49}$$

3.3 Structural Thermal Energy Equations

3.4 Newton's Method

COBRA currently uses a single linearized Newton step to solve the hydrodynamic equations. This method has been shown to be adequate **??** given a small enough time step. One potential improvement that can be made to the current method is multiple Newton steps being taken with a frozen Jacobian. The Jacobian will be evaluated at the old time value and not updated during the Newton process.

3.5 Krylov Solvers

To solve a give newton iteraete, a Krylov subspace based method is used. This methodology eliminates the requirement of analytically forming the Jacobian.

$$\begin{bmatrix} h \\ hu \end{bmatrix}_t + \begin{bmatrix} uh \\ hu^2 + \frac{1}{2}gh^2 \end{bmatrix}_x = 0 \tag{3.50}$$

Chapter 4

Jacobian-Free Newton Krylov Overview

In the Jacobian-Free Newton Krylov framework, the following procedure is taken:

Algorithm 4.1 Transient Loop

Require: $\underline{\mathbf{x}}^0$ and t^0
1: **Set:** $n = 0$
2: **loop** Take a Time Step
3: **Set:** $\underline{\mathbf{x}}^n$
4: **Calculate:** Δt
5: $t^{n+1} := t^n + \Delta t$
6: **Black Box:** Solve for $\underline{\mathbf{x}}^{n+1}$
7: **Test:** CCFL ▷ Time-step Failure Mechanism (ccfl_fail)
8: **Black Box:** Interfacial Area Transport Equation
9: **Calculate:** Courant Numbers
10: **end loop** $n = n + 1$

4.1 Non-linear Function

The vector function, $\underline{\mathbf{F}}(\underline{\mathbf{x}})$, is the nonlinear residual of the discrete version of the governing PDEs. For the physics of interest in this work, $\underline{\mathbf{F}}(\underline{\mathbf{x}})$ is a non-linear function. The degree of non-linearity of $\underline{\mathbf{F}}(\underline{\mathbf{x}})$ depends upon the choice of numerical method used to distretize the governing PDEs. For implementational reasons, the non-linear function is broken into two parts: the explicit and the implicit portions. As shown in equation (4.1), the explicit component, $\underline{\mathbf{E}}$, is independent of $\underline{\mathbf{x}}$, while the implicit component, $\underline{\mathbf{I}}(\underline{\mathbf{x}})$, is a function of $\underline{\mathbf{x}}$.

$$\underline{\mathbf{F}}(\underline{\mathbf{x}}) = \underline{\mathbf{E}} + \underline{\mathbf{I}}(\underline{\mathbf{x}}) \tag{4.1}$$

Algorithm 4.2 Modified Newton's Method - Frozen Jacobian

Define: $\underline{\mathbf{x}}_0^{n+1}$
Calculate: Junction Values
Calculate: $\underline{\mathbf{u}}_0^{n+1}$
Set: $\underline{\mathbf{u}}_0^{*} = \underline{\mathbf{u}}_0^{n+1}$
Apply BCs
Set: $k = 0$
loop Take a Newton Step
  if $k = 0$ **then**
    Calculate: $\mathbf{J}_0 \equiv \underline{\underline{\mathbf{J}}}(\underline{\mathbf{x}}_0^{n+1})$
    Calculate: $\mathbf{E} \equiv \underline{\mathbf{E}}(\underline{\mathbf{x}}^n)$
  else
    Apply BCs
  end if
  Calculate: $\mathbf{I}_k \equiv \underline{\mathbf{I}}\,(\underline{\mathbf{x}}_k^{n+1})$
  Calculate: $\mathbf{F}_k = \mathbf{E} + \mathbf{I}_k$
  Black Box: Scale $\mathbf{F}_k$ and $\mathbf{J}_0$.
  Black Box: $\underline{\delta \mathbf{x}}_k = -\mathbf{J}_0^{-1} \cdot \mathbf{F}_k$ ▷ Solver Choice
  Black Box: $\underline{\mathbf{x}}_{k+1}^{n+1}$ ▷ Line-search or Trust Region
  Update: Material Properties ▷ UpdateVariables
  Test: Material Properties
  Update: Junction Values ▷ UpdateJunctions
  Update: $\underline{\mathbf{u}}_{k+1}^{n+1}$ ▷ CalcAxialVelocity and CalcTransVelocity
  Update: $\underline{\mathbf{u}}_{k+1}^{*}$ ▷ CalcModAxialVelocity and CalcModTransVelocity
  Calculate: Convergence Norms
  if Converged **then**
    exit loop
  end if
end loop $k = k + 1$

Solving the non-linear function depends upon finding a $\underline{\mathbf{x}}$ such that equation (4.2) is satisfied.

$$\underline{\mathbf{F}}\,(\underline{\mathbf{x}}\,) = 0 \tag{4.2}$$

4.2 Newton's Method

Newton's Method for solving the nonlinear system, Equation (4.2), is an iterative process involving a multidimensional Taylor Series expansion. A first order Taylor series expansion of $\underline{\mathbf{F}}\,(\underline{\mathbf{x}}\,)$ near $\underline{\mathbf{x}}_0$ is shown in equation (4.3).

$$\underline{\mathbf{F}}\,(\underline{\mathbf{x}}\,) = \underline{\mathbf{F}}\,(\underline{\mathbf{x}}_0 + \underline{\delta\mathbf{x}}\,) = \underline{\mathbf{F}}\,(\underline{\mathbf{x}}_0) + \underline{\underline{\mathbf{J}}}(\underline{\mathbf{x}}_0) \cdot \underline{\delta\mathbf{x}} + \mathcal{O}(\underline{\delta\mathbf{x}}^2) \tag{4.3}$$

Let $\underline{\mathbf{x}}_n$ represent a vector of known values of independent parameters, with $\underline{\mathbf{x}}_0$ being the initial conditions of the problem. Let $\underline{\delta\mathbf{x}}_n$ be the changes in $\underline{\mathbf{x}}_n$ that will bring you from $\underline{\mathbf{x}}_n$ to $\underline{\mathbf{x}}_{n+1}$, where $\underline{\mathbf{x}}_{n+1}$ is the future time value of independent parameters.

$$\underline{\mathbf{x}}_{n+1} = \underline{\mathbf{x}}_n + \underline{\delta\mathbf{x}}_n \tag{4.4}$$

Note that $\underline{\underline{\mathbf{J}}}(\underline{\mathbf{x}}_0)$ is the Jacobian of $\underline{\mathbf{F}}\,(\underline{\mathbf{x}}\,)$ evaluated at $\underline{\mathbf{x}}_0$. Using (4.3), equation (4.2) can be recast as a linear system where the unknown is $\underline{\delta\mathbf{x}}$.

$$\underline{\underline{\mathbf{J}}}(\underline{\mathbf{x}}\,) \cdot \underline{\delta\mathbf{x}} = -\underline{\mathbf{F}}\,(\underline{\mathbf{x}}\,) \tag{4.5}$$

Solving for $\underline{\delta\mathbf{x}}$ in (4.5) constitutes one Newton step.

4.2.1 Krylov Solver

Once (4.5) is solved for $\underline{\delta\mathbf{x}}$, a new $\underline{\mathbf{x}}$ is obtain by equation (4.4). in order to update $\underline{\mathbf{x}}_k$ to $\underline{\mathbf{x}}_{k+1}$. To solve this linear system, a Krylov subspace method is used, in particular GMRES. We will first drop the $_k$ and $_{k+1}$ subscripts because we will be using $_j$ and $_{j+1}$ to denote the Krylov (inner) iteration count.

Algorithm 4.3 Newton's Method

Define: x_0
Define: tolerance
$\gamma := 1$
$k := 0$
while $\gamma \leq$ tolerance **do** ▷ Test convergence of non-linear step.
    $F^k := F(x^k)$ ▷ Evaluate non-linear function at x^k.
    $J^k := J(x^k)$ ▷ Evaluate Jacobian matrix at x^k.
    $\delta x^k := J^{-k} \cdot F^k$ ▷ Solve for δx by applying the inverse of J^k to F^k.
    $x^{k+1} := x^k + \delta x^k$ ▷ Calculate x^{k+1}.
    $\gamma := \min\left(\frac{\|J\delta x\|_2}{\|F\|_2}, \frac{\|\delta x\|_2}{\|x\|_2}\right)$ ▷ Calculate convergence criteria.
    $k := k + 1$ ▷ Increment index.
end while

First, an initial guess for $\underline{\delta \mathbf{x}}$ is chosen, $\underline{\delta \mathbf{x}}_0$. In the literature, this initial guess is often zero for a transient simulation; the assertion is that $\underline{\delta \mathbf{x}}$ should be small within a time step. Allowing for a nonzero initial guess, define the residual:

$$\underline{\mathbf{r}}_0 = -\underline{\mathbf{F}}(\underline{\mathbf{x}}_k) - \underline{\underline{\mathbf{J}}}(\underline{\mathbf{x}}_k) \cdot \underline{\delta \mathbf{x}}_0 \tag{4.6}$$

We now normalize the residual, $\underline{\mathbf{v}}_1 = \frac{\underline{\mathbf{r}}_0}{\|\underline{\mathbf{r}}_0\|_2}$. This vector will serve as our first basis vector (we need to start somewhere). We then enter into an iterative process starting with j = 1. Now compute the Jacobian-vector product, $\underline{\underline{\mathbf{J}}}(\underline{\mathbf{x}}_k) \cdot \underline{\mathbf{v}}_j$. We do this using a variant of Equation (4.3).

$$\underline{\mathbf{F}}(\underline{\mathbf{x}}_k + \epsilon \underline{\mathbf{v}}_j) = \underline{\mathbf{F}}(\underline{\mathbf{x}}_k + \epsilon\, \underline{\underline{\mathbf{J}}}(\underline{\mathbf{x}}_k) \cdot \underline{\mathbf{v}}_j + \mathcal{O}((\epsilon \underline{\mathbf{v}}_j)^2) \tag{4.7}$$

$$\underline{\underline{\mathbf{J}}}(\underline{\mathbf{x}}_k) \cdot \underline{\mathbf{v}}_j = \frac{\underline{\mathbf{F}}(\underline{\mathbf{x}}_k + \epsilon \underline{\mathbf{v}}_j) - \underline{\mathbf{F}}(\underline{\mathbf{x}}_k)}{\epsilon} \tag{4.8}$$

It is apparent that each Jacobian-vector product requires the evaluation of $\underline{\mathbf{F}}(\underline{\mathbf{x}})$ at a slightly perturbed value from the base state. The epsilon is a small value that is one of the tweaks that can affect convergence performance. However, traditionally this is near machine round-off.

The vector resulting from the Jacobian-Vector product, $\underline{\mathbf{w}}_j$, is then put through a Gram-Schmidt orthogonalization with all previous Krylov vectors $\underline{\mathbf{v}}_1,\ \underline{\mathbf{v}}_2,\ \ldots,\ \underline{\mathbf{v}}_{j-1},\ \underline{\mathbf{v}}_j$. There are alternative options for this point in the algorithm (such as the Householder variation of the Arnoldi process) - these are additional options that you can change in PETSc. Part of this process (the inner-product with previous Kyrlov vectors) generates entries for the j^{th} column of a matrix, **H**. The projections of each previous Krylov vector onto $\underline{\mathbf{w}}_j$ are subtracted from $\underline{\mathbf{w}}_j$ (this is Gram-Schmidt). This new Krylov vector is determined by normalizing $\underline{\mathbf{w}}_j$, $\underline{\mathbf{v}}_{j+1} = \frac{\underline{\mathbf{w}}_j}{\|\underline{\mathbf{w}}_j\|_2}$. **Side Note:** the norm of $\underline{\mathbf{w}}_j$ is the $h_{j+1,j}$ entry in the matrix **H**.

The stopping criteria for this process is related to the norm of the latest residual, Equation (4.9), and the norm of $-\underline{\mathbf{F}}(\underline{\mathbf{x}}_k)$. The exact relation is determined by three tunable knobs in PETSc.

$$\|\underline{\mathbf{r}}_j\|_2 = \| - \underline{\mathbf{F}} - \underline{\mathbf{J}} \cdot \underline{\delta \mathbf{x}}_j\|_2 \tag{4.9}$$

NOTE: The residual in Equation (4.9) is generated independently of constructing $\underline{\delta \mathbf{x}}_j$ (the final answer), which is only done at the end of the GMRES process.

After the process has stopped, a least square process is applied using $\underline{\mathbf{H}}$ to determine the coefficients for a linear combination of $\underline{\mathbf{v}}$ that will result in a $\underline{\delta \mathbf{x}}$ that minimizes $\| - \underline{\mathbf{F}}(\underline{\mathbf{x}}_k) - \underline{\underline{\mathbf{J}}}(\underline{\mathbf{x_k}}) \cdot \underline{\delta \mathbf{x}} \|_2$. That concludes the Krlov (inner) iteration.

This $\underline{\delta \mathbf{x}}_k$ then used to computer $\underline{\mathbf{x}}_{k+1}$, which is then used in a line search (or trust region) globalization step. If good enough (PETSc options), then accept $\underline{\delta \mathbf{x}}_k$ and $\underline{\mathbf{x}}_{k+1}$ as $\underline{\delta \mathbf{x}}_n$ and $\underline{\mathbf{x}}_{n+1}$. This ends the Newton (outer) iteration.

Take next time step.

4.2.2 Globalization Strategies

4.2.2.1 Line Search

4.2.2.2 Trust Region

DISCARD THIS PAGE

Appendix A: Model stuff

This is an example of a Matlab m-