Jacobian Evaluation Project

George Mesina

International RELAP5-3D User Group Meeting

Date: August 13, 2015

Overview

- Background
- Select minimal fluid state input
- Determine state from input
- Calculate analytical Jacobian with RELAP5 coding
- Calculate numerical approximation of Jacobian
- Implement

Background

- Project Aim
  - **Simplifications**, such as linearizations, are made in going from the PDE form to the FDE form of the governing equations
  - **Examine** the FDEs to determine if <u>improvements</u> can be made
  - **Compare** <u>analytical</u> (from RELAP5-3D) with <u>numerical</u> (obtained by perturbation) forms
- Restrictions
  - The momentum equations are NOT included in the comparisons
  - The only terms in the mass and energy equations to include are:
    - Temporal derivative
    - Interfacial mass and energy transfer
    - Energy sink and source term

Background

- *Mass (gas, liquid, noncondensable) & Energy Equations (gas, liquid)*
  - Conserved quantities in **red**

Vector of ***conserved*** quantities

$$\bar{F} = \begin{bmatrix} \alpha_g \rho_g X_n \\ \alpha_g \rho_g U_g \\ \alpha_f \rho_f U_f \\ \alpha_g \rho_g \\ \alpha_f \rho_f \end{bmatrix}$$

- $\frac{\partial(\boldsymbol{\alpha_g \rho_g})}{\partial t} + \frac{1}{A}\frac{\partial}{\partial x}(\alpha_g \rho_g v_g A) = \Gamma_g$
- $\frac{\partial(\boldsymbol{\alpha_f \rho_f})}{\partial t} + \frac{1}{A}\frac{\partial}{\partial x}(\alpha_f \rho_f v_f A) = \Gamma_f = -\Gamma_g$
- $\frac{\partial(\boldsymbol{\alpha_g \rho_g X_n})}{\partial t} + \frac{1}{A}\frac{\partial}{\partial x}(\alpha_g \rho_g X_n v_g A) = 0$
- $\frac{\partial(\boldsymbol{\alpha_g \rho_g U_g})}{\partial t} + \frac{1}{A}\frac{\partial \alpha_g \rho_g U_g v_g A}{\partial x} = -P\frac{\partial \alpha_g}{\partial t} - \frac{P}{A}\frac{\partial(\alpha_g v_g A)}{\partial x} + Q_{wg} + Q_{ig} - Q_{gf} + \Gamma_{ig} h_g^* + \Gamma_w h_g' + DISS_g$
- $\frac{\partial(\boldsymbol{\alpha_f \rho_f U_f})}{\partial t} + \frac{1}{A}\frac{\partial \alpha_f \rho_f U_f v_f A}{\partial x} = -P\frac{\partial \alpha_f}{\partial t} - \frac{P}{A}\frac{\partial(\alpha_f v_f A)}{\partial x} + Q_{wf} + Q_{if} + Q_{gf} - \Gamma_{ig} h_f^* - \Gamma_w h_f' + DISS_f$

Background – Jacobian

- Independent variables are: $\bar{x} = (X_n, U_g, U_f, \alpha_g, P)^T$
- Jacobian Matrix: $J_{i,j} = \partial \bar{F}_i / \partial x_j$

- $$J = \begin{bmatrix} \frac{\partial(\alpha_g \rho_g X_n)}{\partial X_n} & \frac{\partial(\alpha_g \rho_g X_n)}{\partial U_g} & 0 & \frac{\partial(\alpha_g \rho_g X_n)}{\partial \alpha_g} & \frac{\partial(\alpha_g \rho_g X_n)}{\partial P} \\ \frac{\partial(\alpha_g \rho_g U_g)}{\partial X_n} & \frac{\partial(\alpha_g \rho_g U_g)}{\partial U_g} & \frac{\partial(\alpha_g \rho_g U_g)}{\partial U_f} & \frac{\partial(\alpha_g \rho_g U_g)}{\partial \alpha_g} & \frac{\partial(\alpha_g \rho_g U_g)}{\partial P} \\ \frac{\partial(\alpha_f \rho_f U_f)}{\partial X_n} & \frac{\partial(\alpha_f \rho_f U_f)}{\partial U_g} & \frac{\partial(\alpha_f \rho_f U_f)}{\partial U_f} & \frac{\partial(\alpha_f \rho_f U_f)}{\partial \alpha_g} & \frac{\partial(\alpha_f \rho_f U_f)}{\partial P} \\ \frac{\partial(\alpha_g \rho_g)}{\partial X_n} & \frac{\partial(\alpha_g \rho_g)}{\partial U_g} & \frac{\partial(\alpha_g \rho_g)}{\partial U_f} & \frac{\partial(\alpha_g \rho_g)}{\partial \alpha_g} & \frac{\partial(\alpha_g \rho_g)}{\partial P} \\ \frac{\partial(\alpha_f \rho_f)}{\partial X_n} & \frac{\partial(\alpha_f \rho_f)}{\partial U_g} & \frac{\partial(\alpha_f \rho_f)}{\partial U_f} & \frac{\partial(\alpha_f \rho_f)}{\partial \alpha_g} & \frac{\partial(\alpha_f \rho_f)}{\partial P} \end{bmatrix}$$

Background

- Use actual **RELAP5-3D subroutines** to build analytical Jacobian
  - Either ensures the actual coding works or finds errors
  - Requires modification to allow call from alternate program
- **New program** supplies all required data to RELAP5-3D routines
  - Link & load these routine into new program executable
  - **Goal**: Minimize input data
- **Analyze** Jacobian for many fluid state inputs
- **Process**: at each input fluid state
  - Determine the state of the fluid to calculate required quantities
  - Calculate terms of Jacobian matrix as RELAP5-3D does
  - Calculate terms w/ numerical derivatives
  - Calculate differences and other measures
- Ultimately, determine if Jacobian calculation can be improved

Analysis: What Should Be in Input State Point?

- What is needed to build the analytical Jacobian matrix?
- Examine Jacobian coefficients built by subroutine PRESEQ
- EXAMPLE – Row 5, from the sum density equation

Element	RELAP5-3D calculation Supply data to calculate these	Symbols in Vol. 1
a51_	vlm(mi)%voidg*vlm(mi)%drgdxa	$\alpha_g \frac{\partial \rho_g}{\partial X_n}$
a52_	vlm(mi)%voidg*vlm(mi)%drgdug	$\alpha_g \frac{\partial \mathrm{\rho_g}}{\partial \mathrm{U_g}}$
a53_	vlm(mi)%voidf*vlm(mi)%drfduf	$\alpha_f \frac{\partial \mathrm{\rho_f}}{\partial \mathrm{U_f}}$
a54_	vlm%rhog - vlm%rhof	$\mathrm{\rho_g} - \mathrm{\rho_f}$
a55_	agrgp + afrfp	$\alpha_g \frac{\partial \boldsymbol{\rho_g}}{\partial \boldsymbol{P}} + \alpha_f \frac{\partial \boldsymbol{\rho_f}}{\partial \boldsymbol{P}}$

Analysis for State Point – Example of a22

- Color matches code to symbols
- The implicit coupling terms of TH and Heat Conduction are shown in **blue**

a22_ = agug_*vlm%drgdug + vlm%voidg*vlm%rhog + **a2**_

- (htcgg_ + htgcgg_*vlm%sathf + htgwfg_*vlm%sathg)*vlm%dtgdug

- (htcgp_ + htgcgp_*vlm%sathf + htgwfp_*vlm%sathg)*vlm%dtdug

- In terms of variables from the manual

$$A_{22} = \alpha_g u_g \frac{\partial \rho_g}{\partial u_g} + \alpha_g \rho_g + h_f^* \left(\frac{\Delta t}{h_g^* - h_f^*}\right)\left(\frac{P_S}{P} H_{ig}\right)\left[\frac{\partial T^s}{\partial U_g} - \frac{\partial T_g}{\partial U_g}\right]$$

$$+ h_g^* \left(\frac{\Delta t}{h_g^* - h_f^*}\right)\left(H_{if}\right)\frac{\partial T^s}{\partial U_g} + \Delta t\left(1 - \frac{P_{ps}}{P}\right) H_{gf} \frac{\partial T_g}{\partial U_g}$$

$$-\Delta t\left(Q_{wgg} + \Gamma_{wgg} h_f^{sat} + \Gamma_{wfg} h_g^{sat}\right)\frac{\partial T_g}{\partial U_g}$$

$$-\Delta t\left(Q_{wgp} + \Gamma_{wgp} h_f^{sat} + \Gamma_{wfg} h_g^{sat}\right)\frac{\partial T^s}{\partial U_g}$$

Analysis for State Point

- To build 5x5 Jacobian matrix, the following quantities are needed:

- For explicit coupling between TH and heat conduction only
  - Non-derivative quantities
    - α_g, h_f, h_g, h'_f, h'_g, h^*_f, h^*_g, P, P_s, H_{gf}, H_{if}, H_{ig}, ρ_f, ρ_g, T_f, T_g, T^s, U_f, U_g, X_n, Δt.
  - Derivative quantities
    - $\frac{\partial \rho_f}{\partial U_f}, \frac{\partial \rho_f}{\partial P}, \frac{\partial \rho_g}{\partial P}, \frac{\partial \rho_g}{\partial U_g}, \frac{\partial \rho_g}{\partial X_n}, \frac{\partial T_f}{\partial P}, \frac{\partial T_f}{\partial U_f}, \frac{\partial T^s}{\partial P}, \frac{\partial T^s}{\partial U_g}, \frac{\partial T^s}{\partial X_n}, \frac{\partial T_g}{\partial P}, \frac{\partial T_g}{\partial U_g}, \frac{\partial T_g}{\partial X_n}$.

- For implicit coupling need 16 more (in **blue** on previous slide)
    - Γ_{wgf}, Γ_{wgg}, Γ_{wgp}, Γ_{wgt}, Γ_{wgf}, Γ_{wgg}, Γ_{wgp}, Γ_{wgt},
    - Q_{wgf}, Q_{wgg}, Q_{wgp}, Q_{wgt}, Q_{wgf}, Q_{wgg}, Q_{wgp}, Q_{wgt}

State Point Specification

- Many quantities calculated by STATEP and GETSTATE routines
- Subroutine HTADV calculates the Q and Γ quantities
- The rest calculated by VEXPLT or PRESEQ

- To select a minimal set of input:
  1. **Examine** the Jacobian matrix coefficients
  2. **Choose** familiar (easily measurable) physical quantities
  3. **Include** heat transfer coefficients (they are necessary)

State Specification: the Input State-Point

- Minimum input to specify fluid state **EXPLICIT** COUPLING

FDE	Variable	Description
P_L^n	vlm(L)%p	Total Pressure
$\alpha_{g,L}^n$	vlm(L)%voidg	Void (volume) fraction
$X_{n,L}^n$	vlm(L)%quala	Noncondensable quality
$T_{g,L}^n$	vlm(L)%tempg	Gas temperature
$T_{f,L}^n$	vlm(L)%tempf	Liquid temperature
$T_L^{s,n}$	vlm(L)%tsatt	Saturation Temperature (used for saturation pressure)
$H_{gf,L}^n$	vlm(L)%hgf	Direct heating heat transfer coefficient per unit volume
$H_{ig,L}^n$	vlm(L)%hig	Gas interfacial heat transfer coefficient per unit volume
$H_{if,L}^n$	vlm(L)%hif	Liquid interfacial heat transfer coefficient per unit volume
R	Relative Flag	Flag to indicate whether temperature values are absolute or relative.

  - *L = control volume = 1, n = time-level = 1*

State Specification: the Input State-Point

- **IMPLICIT** COUPLING, additional required input for Mass Transfer

FDE	Derivative	Var.	Description
$\Gamma^{n}_{wff,L}$	$\frac{\partial \Gamma_{wf}}{\partial T_f} = \frac{\partial \Gamma_w}{\partial T_f}$	htgwff	Mass transfer rate to liquid in the thermal boundary layer at the wall w/ T_f as the reference temperature
$\Gamma^{n}_{wfg,L}$	$\frac{\partial \Gamma_{wf}}{\partial T_g} = \frac{\partial \Gamma_w}{\partial T_g}$	htgwfg	Mass transfer rate to liquid in the thermal boundary layer at the wall w/ T_g as the reference temperature
$\Gamma^{n}_{wfp,L}$	$\frac{\partial \Gamma_{wf}}{\partial T^s} = \frac{\partial \Gamma_w}{\partial T^s(P_s)}$	htgwfp	Mass transfer rate to liquid in the thermal boundary layer at the wall w/ $T^s(P_s)$ as the reference temperature
$\Gamma^{n}_{wft,L}$	$\frac{\partial \Gamma_{wf}}{\partial T_t} = \frac{\partial \Gamma_w}{\partial T_t(P)}$	htgwft	Mass transfer rate to liquid in the thermal boundary layer at the wall w/ $T^s(P_{Total})$ as the reference temperature
$\Gamma^{n}_{wgf,L}$	$\frac{\partial \Gamma_{wg}}{\partial T_f} = \frac{\partial \Gamma_c}{\partial T_f}$	htgcgf	Mass transfer rate to vapor/gas in the thermal boundary layer at the wall w/ T_f as the reference temperature
$\Gamma^{n}_{wgg,L}$	$\frac{\partial \Gamma_{wg}}{\partial T_g} = \frac{\partial \Gamma_c}{\partial T_g}$	htgcgg	Mass transfer rate to vapor/gas in the thermal boundary layer at the wall w/ T_g as the reference temp.
$\Gamma^{n}_{wgp,L}$	$\frac{\partial \Gamma_{wg}}{\partial T^s} = \frac{\partial \Gamma_c}{\partial T^s(P_s)}$	htgcgp	Mass transfer rate to vapor/gas in the thermal boundary layer at the wall w/ $T^s(P_s)$ as the reference temp.
$\Gamma^{n}_{wgt,L}$	$\frac{\partial \Gamma_{wg}}{\partial T_t} = \frac{\partial \Gamma_c}{\partial T_t(P)}$	htgwff	Mass transfer rate to vapor/gas in the thermal boundary layer at the wall w/ $T^s(P_{Total})$ as the reference temperature

State Specification: the Input State-Point

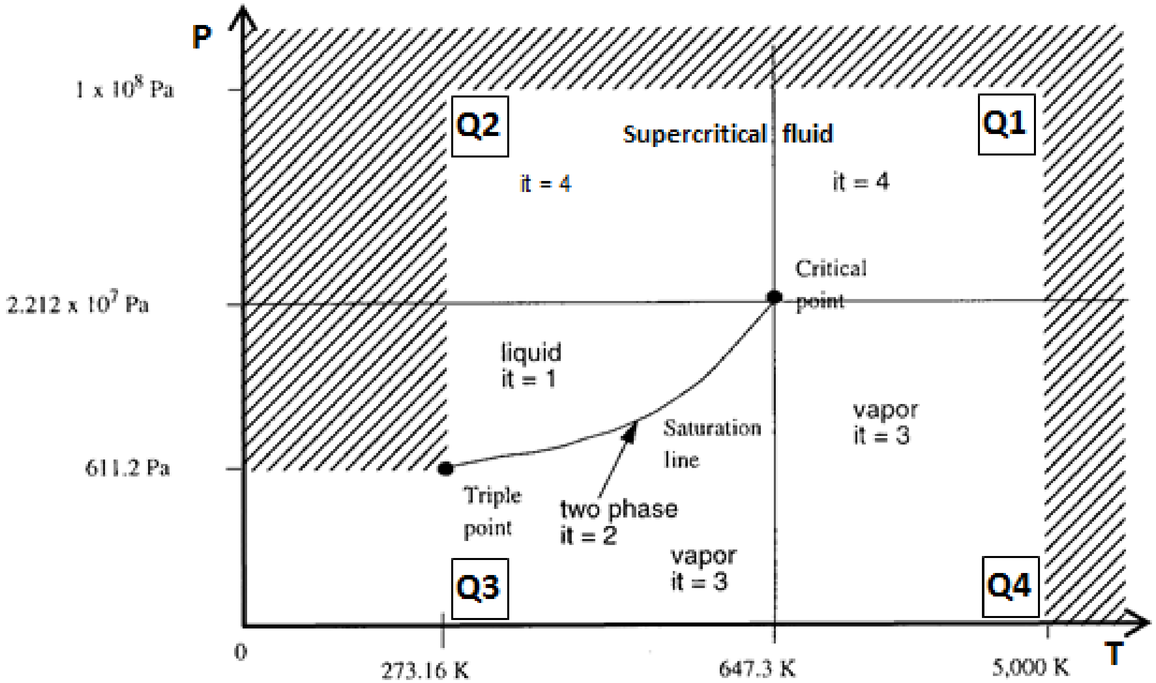
- **IMPLICIT** COUPLING, additional required input for Heat Transfer

FDE	Deriv	Var.	Description
$Q^n_{\mathrm{wff,L}}$	$\frac{\partial Q_{wf}}{\partial T_f}$	htcff	**Wall heat transfer rate to the liquid per unit volume w/ T_f as the reference temperature**
$Q^n_{\mathrm{wfg,L}}$	$\frac{\partial Q_{wf}}{\partial T_g}$	htcfg	**Wall heat transfer rate to the liquid per unit volume w/ T_g as the reference temperature**
$Q^n_{\mathrm{wfp,L}}$	$\frac{\partial Q_{wf}}{\partial T^s}$	htcfp	**Wall heat transfer rate to the liquid per unit volume w/ $T^s(P_s)$ as the reference temperature**
$Q^n_{\mathrm{wft,L}}$	$\frac{\partial Q_{wf}}{\partial T_t}$	htcft	**Wall heat transfer rate to the liquid per unit volume w/ $T^s(P_{Total})$ as the reference temperature**
$Q^n_{\mathrm{wgf,L}}$	$\frac{\partial Q_{wg}}{\partial T_f}$	htcgf	**Wall heat transfer rate to the vapor/gas per unit volume w/ T_f as the reference temperature**
$Q^n_{\mathrm{wgg,L}}$	$\frac{\partial Q_{wg}}{\partial T_g}$	htcgg	**Wall heat transfer rate to the vapor/gas per unit volume w/ T_g as the reference temperature**
$Q^n_{\mathrm{wgp,L}}$	$\frac{\partial Q_{wg}}{\partial T^s}$	htcgp	**Wall heat transfer rate to the vapor/gas per unit volume w/ $T^s(P_s)$ as the reference temperature**
$Q^n_{\mathrm{wgt,L}}$	$\frac{\partial Q_{wg}}{\partial T_t}$	htcgt	**Wall heat transfer rate to the vapor/gas per unit volume w/ $T^s(P_{Total})$ as the reference temperature**

Calculate Other Quantities Needed for Jacobian

- For **EXPLICIT** coupling of TH and Heat Conduction, the following were excluded from input
  - Must be calculated or defaulted
- Δt = timestep
- V = volume
- Non-derivative properties
  - $h_f, h_g, h'_f, h'_g, h^*_f, h^*_g, P_s, \rho_f, \rho_g, U_f, U_g,$
- Derivative quantities (calculated in STATEP)
  - $\frac{\partial \rho_f}{\partial U_f}, \frac{\partial \rho_f}{\partial P}, \frac{\partial \rho_g}{\partial P}, \frac{\partial \rho_g}{\partial U_g}, \frac{\partial \rho_g}{\partial X_n}, \frac{\partial T_f}{\partial P}, \frac{\partial T_f}{\partial U_f}, \frac{\partial T^s}{\partial P}, \frac{\partial T^s}{\partial U_g}, \frac{\partial T^s}{\partial X_n}, \frac{\partial T_g}{\partial P}, \frac{\partial T_g}{\partial U_g}, \frac{\partial T_g}{\partial X_n}.$
- Must know state of the fluid to calculate these
  - Determine fluid state from input quantities

Determine Fluid State from Input



- *2 tests in Q3 handle:*
  - *1- and 2-phase*
  - *Stable & metastable combinations*

- **Q1, Q2, Q4 fluid state**
  - If $P > P_{crit}$, IT = 4
  - Else if $T_g > T_{crit}$, IT = 3
- **Q3** requires 2 tests

Test 1: Vapor Phase

if α_g **> 0**, then
- IF $T_g > T^{sat}$, Stable Vapor
- ELSE: Metastable Vapor

Test 2: Liquid Phase

If α_g **< 1**, then
- IF $T_f < T^{sat}$, Stable Liquid
- ELSE: Metastable Liquid

Algorithm for Fluid State Determination

- For Metastable and saturated phases

Input: (1) T_{in} = Either T_f or T_g

(2) IT = 1 for liquid, 3 for gas

Obtain saturation properties from GETSTATE calls with input quantities:

– $v_g, U_g, h_g, \rho_g, \beta_g, \kappa_g, C_{p,g}, S_g, v_f, U_f, h_f, \rho_f, \beta_f, \kappa_f, C_{p,f}, S_f$

***metastable** liquid* ***metastable** gas*

If (**IT == 1 and $\boldsymbol{T^s < T_{in}}$**) OR (**IT == 3 and $\boldsymbol{T^s > T_{in}}$**) then

Set $\boldsymbol{T_{Meta} = T_{in}}$

Calculate $\boldsymbol{U = U_{Meta} = U^s + (T_{Meta} - T^s)(C_P^s - P v^s \beta^s)}$ Vol. 1 (3.2-6)

Else if $\boldsymbol{T^s = T_{in}}$ ***saturated** liquid or gas*

Calculate $\boldsymbol{U = \alpha_g U_g + \alpha_f U_f}$

Set IT = 2

Call POLATES with *IT*, *U*, *P and ISTATE=6*

Other Required Non-derivative Fluid Quantities

- After fluid state determined from obtain remaining quantities
  - Already have: h_g, h_f, P^s, U_f, U_g from GETSTATE calls
  - Need: ρ_f, ρ_g, h_f^*, and h_g^*
- Densities, $\rho_f = 1/v_f$ and $\rho_g = 1/v_g$
- h_g^*, h_f^* are calculated as sathgx_ and stahfx_ in VEXPLT
  – For example:
  – $h_f^* = \begin{Bmatrix} h_f & if\, \Gamma_{ig} \geq 0,\ vaporization \\ h_f^s = h_f(P_s) & if\ \Gamma_W < 0,\ condensation \end{Bmatrix}$

Obtaining Necessary Derivatives

- To obtain $\frac{\partial\rho_f}{\partial U_f}, \frac{\partial\rho_f}{\partial P}, \frac{\partial\rho_g}{\partial P}, \frac{\partial\rho_g}{\partial U_g}, \frac{\partial\rho_g}{\partial X_n}, \frac{\partial T_f}{\partial P}, \frac{\partial T_f}{\partial U_f}, \frac{\partial T^s}{\partial P}, \frac{\partial T^s}{\partial U_g}, \frac{\partial T^s}{\partial X_n}, \frac{\partial T_g}{\partial P}, \frac{\partial T_g}{\partial U_g}, \frac{\partial T_g}{\partial X_n}$
- Use Vol. 1 Eqns. (3.2-5, 6, 7, 8) for partials of temperature and phasic densities w.r.t. phasic specific internal energies and pressure
  - Implemented in STATEP and POLATES
- Eqn. (3.2-5), $\left(\frac{\partial\rho_f}{\partial U_f}\right)_P = \frac{v_f\beta_f}{(C_{pf}-v_f\beta_f P)v_f^2}, \frac{\partial\rho_g}{\partial U_g} = \frac{v_g\beta_g}{(C_{pg}-v_g\beta_g P)v_g^2}$
- Eqn. (3.2-6), $\left(\frac{\partial T_f}{\partial U_f}\right)_P = \frac{1}{C_{pf}-v_f\beta_f P}, \left(\frac{\partial T_g}{\partial U_g}\right)_P = \frac{1}{C_{pg}-v_g\beta_g P}$
- Eqn. (3.2-7), $\left(\frac{\partial\rho_f}{\partial P}\right)_{U_f} = \frac{C_{pf}v_f\kappa_f-T_f(v_f\beta_f)^2}{(C_{pf}-v_f\beta_f P)v_f^2}, \frac{\partial\rho_g}{\partial U_g} = \frac{C_{pg}v_g\beta_g-T_g(v_g\beta_g)^2}{(C_{pg}-v_g\beta_g P)v_g^2}$
- Eqn. (3.2-8), $\left(\frac{\partial T_f}{\partial P}\right)_{U_f} = \frac{Pv_f\kappa_f-T_f v_f\beta_f}{C_{pf}-v_f\beta_f P}, \left(\frac{\partial T_g}{\partial U_g}\right)_{U_f} = \frac{Pv_g\kappa_g-T_g v_g\beta_g}{C_{pg}-v_g\beta_g P}$
- Leaves only derivatives w.r.t. X_n and two more derivatives of T^s

Obtaining Necessary Derivatives

CASE 1: No NONCONDENSABLE

- If no noncondensable present, derivatives w.r.t. $X_n = 0$
  - So $\frac{\partial \rho_g}{\partial X_n} = \frac{\partial T^s}{\partial X_n} = \frac{\partial T_g}{\partial X_n} = 0$ and $\frac{\partial T^s}{\partial U_g} = 0$
- If no noncondensable present, saturation temp. is a function of P only
  - So $\frac{\partial T^s}{\partial U_g} = 0$
- The Clausius-Clapeyron equation relates fluid properties along the saturation line, *s*.
  - $\frac{\partial T^s}{\partial P} = \frac{T^s v_{fg}}{h_{fg}}$, where $h_{fg} = h_g - h_{fg}$ and $v_{fg} = v_g - v_f$

- Have all 13 derivatives for case of no noncondensable

Obtaining Necessary Derivatives

CASE 2: NONCONDENSABLE present

- Solve Eqn. (3.2-42) for $\left(\frac{\partial P_s}{\partial P}\right)_{U_g,X_n}$ and $\left(\frac{\partial U_s}{\partial P}\right)_{U_g,X_n}$
- Analogs to (3.2-42) give $\left(\frac{\partial P_s}{\partial U_g}\right)_{P,X_n}$, $\left(\frac{\partial U_s}{\partial U_g}\right)_{P,X_n}$, $\left(\frac{\partial P_s}{\partial X_n}\right)_{P,U_g}$, $\left(\frac{\partial U_s}{\partial X_n}\right)_{P,U_g}$
- Obtain $\left[\frac{\partial T_g}{\partial P_s}\right]_{U_s}$ and $\left[\frac{\partial T_g}{\partial U_s}\right]_{P_s}$ from (3.2-6, 8). Then
  - $\frac{\partial T_g}{\partial P} = \left[\frac{\partial T_g}{\partial P_s}\right]_{U_s}\left(\frac{\partial P_s}{\partial P}\right)_{U_g,X_n} + \left[\frac{\partial T_g}{\partial U_s}\right]_{P_s}\left(\frac{\partial U_s}{\partial P}\right)_{U_g,X_n}$ (3.2-46)
  - $\frac{\partial T_g}{\partial U_g} = \left[\frac{\partial T_g}{\partial P_s}\right]_{U_s}\left(\frac{\partial P_s}{\partial U_g}\right)_{P,X_n} + \left[\frac{\partial T_g}{\partial U_s}\right]_{P_s}\left(\frac{\partial U_s}{\partial U_g}\right)_{P,X_n}$ (3.2-47)
  - $\frac{\partial T_g}{\partial X_n} = \left[\frac{\partial T_g}{\partial P_s}\right]_{U_s}\left(\frac{\partial P_s}{\partial X_n}\right)_{P,U_g} + \left[\frac{\partial T_g}{\partial U_s}\right]_{P_s}\left(\frac{\partial U_s}{\partial X_n}\right)_{P,U_g}$ (3.2-48)
- Similarly for T_f and T^s

Numerical Derivative

- Recall

$$\bar{F} = \begin{bmatrix} \alpha_g \rho_g X_n \\ \alpha_g \rho_g U_g \\ \alpha_f \rho_f U_f \\ \alpha_g \rho_g \\ \alpha_f \rho_f \end{bmatrix}, \bar{x} = \begin{bmatrix} X_n \\ U_g \\ U_f \\ \alpha_g \\ P \end{bmatrix}$$

- Use = $\Delta\overline{x_j} = \delta \bar{x}_j \bar{e}_j$, $\delta = 10^{-6}$, $\bar{e}_j$ = unit vector in direction j
- Simplest approximation of a numerical derivative is

$$J_{i,j} = \partial \bar{F}_i / \bar{x}_j \approx \frac{\bar{F}_i(\bar{x} + \Delta\overline{x_j}) - \bar{F}_i(\bar{x})}{\Delta x}$$

Coding

- Main program – jacobian
  - Calls subroutines to **read states**, analyze, output
- Module – jacobmod
  - **Memory**, subroutines act on jacobmod memory, data dictionary
- Subroutine – jacobstate
  - **Determines fluid state** based on input state-point
- Subroutine – jnumderiv
  - Calculates **numerical derivative**
- Subroutine – preseq
  - Modified to be called from Jacobian main program
  - Calculates **analytical derivative**
- Many auxiliary subroutines from RELAP5, POLATES, LAPACK, etc.

Coding

- Jacobmod.F90
  - Declares memory for Jacobian matrices, analysis arrays, scalars
  - Data dictionary and other documentation
  - Subprograms
    - Open Jacobian I/O files, read input header data
    - Allocate and eliminate
    - Check that a state point is valid
    - Copy subroutine from RELAP5 memory to Jacobian matrices
    - Condition number calculation
    - Output of state point analysis data
    - Output of summary data

Coding

- Jacobian.F90
  - Opens Jacobian input and output and fluid property files
  - Allocates memory and writes header info on Jacobian output file
  - Allocates and initializes certain RELAP5-3D data
  - Fluid State Loop
    - Read and determine state
    - Calculate Analytical Jacobian
    - Calculate Numerical Jacobian
    - Analyze: differences, condition number, etc.
    - Write results on Jacobian output file
  - Write summary information on Jacobian output file and close files
- *Note: Jacobian runs separately from RELAP5-3D. None of this coding is active when RELAP5-3D runs*

Progress

- Main program, module and auxiliary programs listed written
  - Unit tested check of state validity and Jacobian condition number
- Completed determination of state of fluid based on input
- State loop tested for 100s of input state points
- Coding of analytical derivative for explicit coupling complete
  - Rewrite of PRESEQ finished and tested
  - Jacobian program and RELAP5-3D can run w/ same PRESEQ
- Development of numerical derivative underway