Equilibrium Reconstruction in Stellarators: V3FIT

Stellarator Theory Teleconference
26 May 2005

James D. Hanson[1], John Shields[1], Steven P. Hirshman[2], Stephen F. Knowlton[1], Lang Lao[3] and Edward A. Lazarus[2]

[1] *Physics Department, Auburn University, Auburn, AL 36849.*

[2] *Oak Ridge National Laboratory, Oak Ridge, TN 37831.*

[3] *General Atomics, San Diego, CA 92186.*

Equilibrium Reconstruction

- Use measured diagnostic signals to determine current and pressure profiles, and hence the MHD equilibrium configuration.

- Diagnostics
  - Magnetic Diagnostics: magnetic probes, flux loops, saddle coils, Rogowski coils etc.
  - Microwave Interferometry and Polarimetry
  - Thomson Scattering
  - Motional Stark Effect

Equilibrium Reconstruction

- A classic Inverse Problem
  - Forward problem, determine signals, given parameters.
    Known Function **S**(**p**)
  - We know (observe) the signals. What are the parameters?
    Determine Inverse Function **p**(**S**)
  - Use Maximum Likelihood / Least Squares.

- Axisymmetric: EFIT Code.
  - EFIT is *tightly coupled* - the least squares iterations proceed *at the same time* as the equilibrium iterations.

- Non-axisymmetric: V3FIT Code.

V3FIT Code Design Goals

- V3FIT will execute fast enough so that reconstructions can be done between shots.
  - Implies tightly coupled equilibrium and reconstruction iterations.
- V3FIT will be flexible, easy to understand, maintain, and modify.
  - V3FIT will be written in Fortran 95.
  - V3FIT must be written in a modular fashion, with clear and consistent data flow.
- The initial working version of V3FIT will use VMEC as the equilibrium solver, and it will have magnetic diagnostics as the primary signal.
- V3FIT will be designed with future enhancements in mind. Future enhancements include:
  - Adding new diagnostics and signals
  - Changing the equilibrium solver

V3FIT: Variables

- **X**: Equilibrium state. (Quantities that change with each equilibration step.)
  - In VMEC, $\mathbf{X} = (R_{mn}, Z_{mn}, \lambda_{mn})$.
  - *No* implication that **X** corresponds to a converged equilibrium.
- **f**: MHD forces. Very small in equilibrium.
- **p**: parameters. Input for Equilibrium Solver. n_p of them.
  - *The parameters are what the Equilibrium Reconstruction will determine*.
  - Pressure profile (VMEC: parameterized as power series in s.)
  - Either iota or current profile
    - Iota profile (VMEC: Uses iota profile when ncur = 0.)
    - Current profile (VMEC: Uses current profile when ncur = 1.)
  - External currents (free-boundary)
  - Outer surface shape (fixed-boundary)

V3FIT: Variables (cont.)

- **D**: Diagnostics, data that comes from an experiment.
  - Magnetic diagnostics
  - Microwave Interferometry - Polarimetry
  - Motional Stark Effect, Thompson Scattering...
- **S**: Signals, either computed from a *model* [$\mathbf{S}_M(\mathbf{p},\mathbf{X})$] or *observed* (computed from diagnostics) [$\mathbf{S}_O(\mathbf{p},\mathbf{D})$]. n_S of them.
  - To reconstruct an equilibrium we minimize the mismatch between observed and equilibrium signals.
  - Computing $\mathbf{S}_M(\mathbf{p},\mathbf{X})$ is the *forward* problem.
  - Often, the observed signal will be the diagnostic itself, $\mathbf{S}_O(\mathbf{p},\mathbf{D}) = \mathbf{D}$. This is the case for magnetic diagnostics.

V3FIT: The Inverse Problem

- ***ASSUMPTION*** - Gaussian Errors
  - The probability of observations $\mathbf{S}_0$ is proportional to:

$$f = \exp\left(-\frac{1}{2}(\mathbf{S}_M(\mathbf{p},\mathbf{X}) - \mathbf{S}_O)\cdot\mathbf{C}^{-1}\cdot(\mathbf{S}_M(\mathbf{p},\mathbf{X}) - \mathbf{S}_O)\right)$$

  - For actual observations $\mathbf{S}_O(\mathbf{p},\mathbf{D})$, we reinterpret the above probability distribution to be the probability of the parameters **p**.
- ***Maximum likelihood, -> least squares.***
  - The *most likely* value of the parameter vector **p** will be where the Gaussian distribution function is a *maximum*.
  - *Therefore*, to find the most likely value of **p**, we minimize chi-squared:

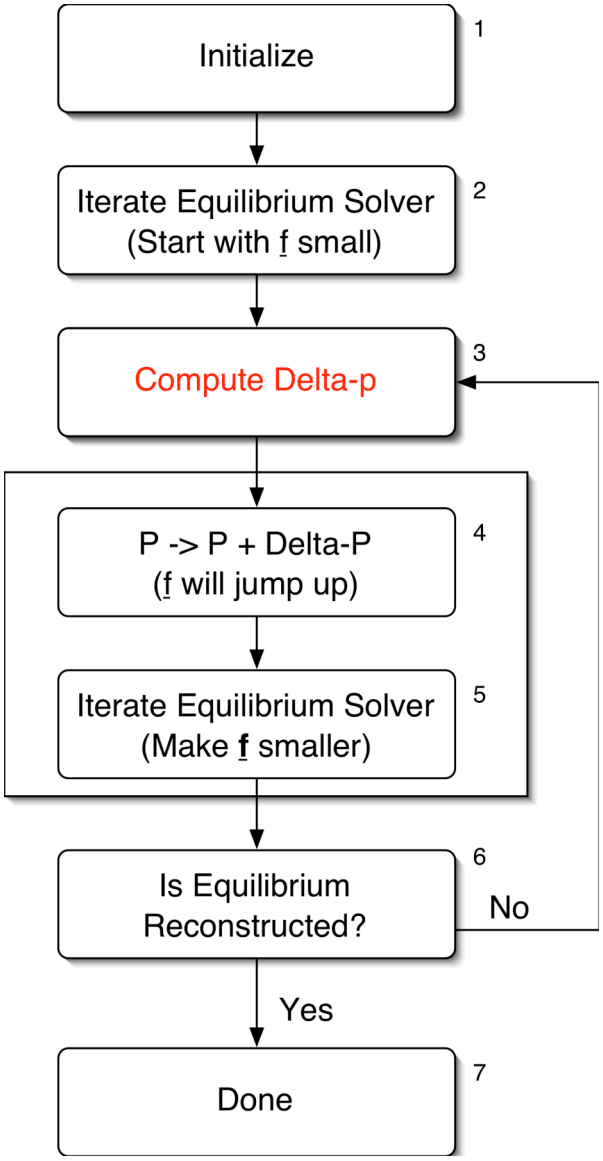
$$\chi^2 = (\mathbf{S}_M(\mathbf{p},\mathbf{X}) - \mathbf{S}_O)\cdot\mathbf{C}^{-1}\cdot(\mathbf{S}_M(\mathbf{p},\mathbf{X}) - \mathbf{S}_O)$$

V3FIT: Two Level Optimizer

- Upper Level : Reconstruction - change **p**
  - Minimize chi-squared by varying **p**
  - **p**: estimate < 100 parameters
  - **S**: estimate about 200 signals
  - Computing the Jacobian is feasible
  - Try standard methods (Levenberg-Marquardt) for optimization.
  - Hope that upper and lower level proceed to convergence at the same time.

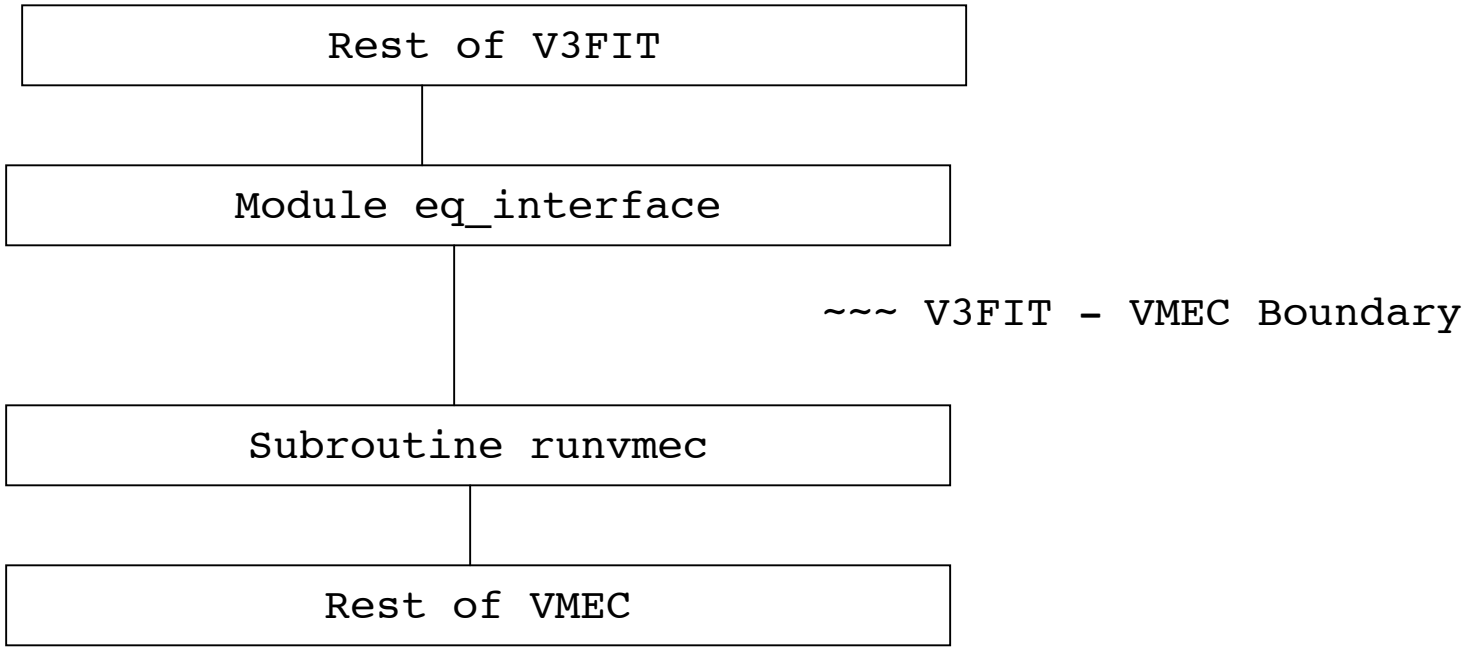
- Lower Level: Equilibration - change **X**
  - Equilibrium Solver (VMEC, or another solver)
  - Minimize **f** by varying **X** (for slowly varying **p**)
  - **X**: (100 modes)(100 Radial)(R,Z,λ) = 30,000
  - **f**: ~ same number as for **X**, = 30,000
  - Hessian d**f**/d**X** very large (~10^9 elements), time consuming to compute.
    => have to be clever (hence VMEC).

V3FIT Flow Chart



- Lots of possibilities for the "Compute Delta-p" step.
  - Jacobian estimation in here
  - Algorithm for Delta-P
- Will need to experiment with how much to iterate equilibrium solver in steps 2 and 5.
- Steps 4 and 5 may be combined: Add Delta-p slowly, over the course of many equilibrium iterations.

V3FIT - VMEC Interface Schematic



- Interaction will occur ONLY through the `eq_interface` module and the `runvmec` subroutine.
- `V3FIT` needs to have access to the object files of the equilibrium solver.

Modules for Derived Types

- Derived types are implemented in modules (generally with names ending in T)
- The derived types reflect the structure of the problem.
- All have subroutines to construct, destroy, and assign derived types.
- Module bsc - Biot-Savart Coil
  - Derived type bsc_coil - a magnetic field coil
  - Derived type bsc_coilcoll - a collection of bsc_coil
  - Subroutines to compute **A**, **B**, gradients of **B**, magnetic flux through a coil, rotate and shift a coil
- Module diagnostic_T
  - Derived type diagnostic_desc - describe a diagnostic
  - Derived type diagnostic_data - carry the diagnostic values
- Module signal_T
  - Derived type signal_desc - describe a signal
  - Derived type signal_data - carry the signal values
  - Derived type signal_mrf - Magnetic Response Function

Modules for Derived Types

- Module eq - Equilibrium
  - Derived type eq_param_fix - Fixed parameters of the equilibrium solver
    - Logical switches
    - Array dimensions
    - Numbers of modes
    - Convergence, iteration parameters
  - Derived type eq_param_var - Variable parameters of the equilibrium solver
  - Derived type eq_state - Equilibrium state, the X variables
  - Derived type eq_aux_*n* - Variables computable from the state

- Module model - The model
  - Right now, just contains a pointer to an eq_state
  - Later may need to add on electron density, information about vacuum vessel (for eddy currents), etc.

Other Modules

- Module eq_interface - Interface with the equilibrium code (VMEC)
  - Subroutine eq_init_file
    - Read a namelist input file, and initialize the equilibrium code.
    - Construct an eq_param_fix, eq_param_var, and eq_state.
  - Subroutine eq_init_structure
    - Input an eq_param_fix, eq_param_var, and eq_state
    - Initialize the equilibrium code
    - (Not yet implemented)
  - Subroutine eq_step
    - Iterate an eq_state for some number of equilibrium iterations
- Module signal_model - Computations that need both a signal and a model
  - Subroutine signal_model_compute
    - Computes the solution to the forword problem, $\mathbf{S}_M(\mathbf{p},\mathbf{X})$

Forward Problem - Magnetic Diagnostics

- Compute the magnetic flux through a diagnostic coil
  - Have to integrate over all the source currents.
  - Currents in field generating coils - effectively need mutual inductances
  - Currents in plasma - have to integrate over plasma volume
- Implemented in earlier codes V3RFUN and V3POST
- V3RFUN and V3POST being used to design magnetic diagnostic for NCSX
- Sections of code from V3POST reused in V3FIT

Forward Problem - Microwave Interferometry and Polarimetry

- Signal for Interferometry is phase change along path of microwave - proportional to the electron density

$$\Delta\varphi \propto \int n_e \, d\ell$$

- Signal for Polarimetry is the change in the angle of polarization of the microwave - proportional to the electron density times the parallel magnetic field.

$$\Delta\alpha \propto \int n_e \, \vec{\mathbf{B}} \bullet d\vec{\ell}$$

- Coding for solution to forward problem is in progress.
- Will be implemented in existing structure of V3FIT

Status

- V3FIT structural coding is complete
- Modules for Derived Types coded, tested
- Interface with VMEC works.
- Implementation of microwave interferometry-polarimetry in progress
- Next TO DO item - coding of reconstruction algorithm

Extra slides follow

V3FIT: Variables (cont.)

- Why is there a distinction between Diagnostics (**D**) and observed Signals ($\mathbf{S}_O$)?
  - Because for some diagnostics, we can't compute what we expect the diagnostic to be, using the MHD equilibrium information.
  - Example of when an observed signal is NOT the diagnostic:
    - Consider Te data along a chord through the plasma. The values of Te along the chord are the diagnostic.
    - We can't compute the Te measurements we expect from **p** and **X**, so can't consider the Te values as a signal.
    - However, we can use the Te data to constrain the flux surface geometry so that Te is a flux surface quantity.
    - If λ is a parameter along a chord through the plasma, then an appropriate signal would be $\lambda_{outer}(\lambda_{inner})$, suitably discretized.
    - We can calculate both an observed signal $S_O(\mathbf{D})$ from the Te diagnostic, and an equilibrium signal $S_E(\mathbf{p},\mathbf{X})$ from the equilibrium state.

Correlations between Observations

- The covariance operator **C**:
  - Diagonal elements are the variances of the signals: $\mathbf{C}_{ii} = \sigma_i^2$
  - Off-diagonal elements contain information about the correlations between the signals. $\mathbf{C}_{ij} = r_{ij}\sigma_i\sigma_j \quad i \neq j \quad -1 \le r_{ij} \le 1$
  - No correlations between the different signals - set the r_{ij} to 0.
  - Note how the covariance operator takes care of the differing dimensions of the signals.
  - Given a probability density f(**x**), the mean is: $\langle \mathbf{x} \rangle = \int d\mathbf{x}\, \mathbf{x}\, f(\mathbf{x})$
  - The Covariance is (in component notation):
    $$\mathbf{C}_{ij} = \int d\mathbf{x}\,(x_i - \langle x_i \rangle)(x_j - \langle x_j \rangle) f(\mathbf{x})$$
  - The normalized Gaussian probability density:
    $$f(\mathbf{x}) = \left((2\pi)^N \det(\mathbf{C})\right)^{-(1/2)} \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{x}_0)\cdot\mathbf{C}^{-1}\cdot(\mathbf{x}-\mathbf{x}_0)\right)$$
    has mean $\mathbf{x}_0$ and covariance **C**.
  - Even for a linear problem, the correlations can change the most likely parameter value.

Normalized Variables

- Normalize the parameters and the signals:

$$a_k = p_k / \tilde{p}_k \qquad y_{M\,i}(a_k, \mathbf{X}) = S_{M\,i}(a_k \tilde{p}_k, \mathbf{X}) / \sigma_i$$

$$y_{O\,i} = S_{O\,i} / \sigma_i$$

- Error: $e_i = y_{O\,i} - y_{M\,i}(\mathbf{a}, \mathbf{X}) \qquad i = 1, n_s$

- chi-squared: $\chi^2 = \sum_{i,j} (y_{M\,i}(\mathbf{a}, \mathbf{X}) - y_{O\,i}) w_{ij} (y_{M\,j}(\mathbf{a}, \mathbf{X}) - y_{O\,j}) = \mathbf{e} \cdot \mathbf{w} \cdot \mathbf{e}$

- weights $w_{ij} = \left(\mathbf{C}^{-1}\right)_{ij} \sigma_i \sigma_j$

- Jacobian: $J_{ik} = \dfrac{\partial y_{M\,i}(\mathbf{a}, \mathbf{X})}{\partial a_k} \qquad i = 1, n_s;\ k = 1, n_p$

- Gradient:

$$\frac{\partial \chi^2}{\partial a_k} = \sum_{i,j} 2 (y_{M\,i}(\mathbf{a}, \mathbf{X}) - y_{O\,i}) w_{ij} \left(\frac{\partial y_{M\,j}(\mathbf{a}, \mathbf{X})}{\partial a_k} \right)$$

$$= -2\,\mathbf{e} \cdot \mathbf{w} \cdot \mathbf{J} = -2\,\mathbf{J}^{\mathbf{T}} \cdot \mathbf{w} \cdot \mathbf{e}$$

Minimization

- Following *Numerical Recipes*, 2nd edition, section 15.5.

$$f(\mathbf{a}+\delta\mathbf{a}) \approx f(\mathbf{a}) - 2\boldsymbol{\beta}\cdot\delta\mathbf{a} + \delta\mathbf{a}\cdot\boldsymbol{\alpha}\cdot\delta\mathbf{a}$$

$$\boldsymbol{\beta} = -\frac{1}{2}\nabla f = \mathbf{e}\cdot\mathbf{w}\cdot\mathbf{J} = \mathbf{J}^{\mathbf{T}}\cdot\mathbf{w}\cdot\mathbf{e}$$

$$\beta_k = -\frac{1}{2}\frac{\partial f}{\partial a_k} = \sum_{i,j}(y_{O\,i} - y_{M\,i}(\mathbf{a},\mathbf{X}))w_{ij}\left(\frac{\partial y_{M\,j}(\mathbf{a},\mathbf{X})}{\partial a_k}\right)$$

$$\boldsymbol{\alpha} = +\frac{1}{2}\nabla\nabla f = \mathbf{J}^{\mathbf{T}}\cdot\mathbf{w}\cdot\mathbf{J} - \mathbf{e}\cdot\mathbf{w}\cdot\nabla\mathbf{J}$$

$$a_{kl} = +\frac{1}{2}\frac{\partial^2 f}{\partial a_k \partial a_l} = \sum_{i,j}\left(\frac{\partial y_{M\,i}(\mathbf{a},\mathbf{X})}{\partial a_k}\right)w_{ij}\left(\frac{\partial y_{M\,j}(\mathbf{a},\mathbf{X})}{\partial a_k}\right) - (y_{O\,i} - y_{M\,i}(\mathbf{a},\mathbf{X}))w_{ij}\left(\frac{\partial^2 y_{M\,j}(\mathbf{a},\mathbf{X})}{\partial a_k \partial a_l}\right)$$

- Following *NR* and everybody else, drop the second derivative term in alpha. $\boldsymbol{\alpha} \approx \mathbf{J}^{\mathbf{T}}\cdot\mathbf{w}\cdot\mathbf{J}$

Minimization (continued)

$$f(\mathbf{a}+\delta\mathbf{a}) \approx f(\mathbf{a}) - 2\boldsymbol{\beta}\cdot\delta\mathbf{a} + \delta\mathbf{a}\cdot\boldsymbol{\alpha}\cdot\delta\mathbf{a}$$

- Two Algorithms
  1) Steepest Descent - go downhill fast. Implies $\delta\mathbf{a} = \mu\boldsymbol{\beta} \qquad \mu > 0$
  2) Hit minimum when gradient is zero, $\nabla f(\mathbf{a}+\delta\mathbf{a}) = -2\boldsymbol{\beta} + 2\boldsymbol{\alpha}\cdot\delta\mathbf{a}$

     Set gradient to zero, solve: $\boldsymbol{\alpha}\cdot\delta\mathbf{a} = \boldsymbol{\beta}$
- Levenberg -Marquardt: Clever way to interpolate between these two algorithms.

$$\alpha'_{kl} = \alpha_{kl}(1+\lambda\delta_{kl}) \qquad \delta\mathbf{a} = \boldsymbol{\alpha}'^{-1}\boldsymbol{\beta}$$

  - λ small, solving for gradient = 0.
  - λ large, matrix diagonally dominant, steepest descent-like step.
  - More signals than parameters, use SVD to get approximate inverse.
- Levenberg - Marquardt step (in $\lambda \to 0$ limit) with SVD minimizes

$$\left|\boldsymbol{\alpha}\cdot\delta\mathbf{a} - \boldsymbol{\beta}\right|^2 = \left|\left(\mathbf{J}^{\mathbf{T}}\cdot\mathbf{w}\right)\cdot\mathbf{J}\cdot\delta\mathbf{a} - \left(\mathbf{J}^{\mathbf{T}}\cdot\mathbf{w}\right)\cdot\mathbf{e}\right|^2$$

  - n_p equations, n_p deviations $\delta\mathbf{a}$. Expect zero minimum.

Jacobian Calculation and Update

- Consider three types of Jacobian calculations.
  - Separately computable (I don't like this phrase. Got a better one?)
  - Finite Difference
  - Broyden Update
- Illustrative Example. Split the parameter vector into pieces.

$$\mathbf{a} = (\mathbf{a}_{ec}, \mathbf{a}_{cp}, \mathbf{a}_{pp}, \mathbf{a}_{other})$$

  - $\mathbf{a}_{ec}$ - all external currents (helical coils, VF coils, correction coils, etc.)
  - $\mathbf{a}_{cp}$ - current profile parameters
  - $\mathbf{a}_{pp}$ - pressure profile parameters
  - $\mathbf{a}_{other}$ - other parameters
  - $\hat{\mathbf{a}}_{ec\,k}$ - unit vector in kth slot of external current portion of the **a** vector.
  - Current, pressure profiles are function of the VMEC radial variable s.

$$I(s) = \sum_{k=1} p_{l(k)} g_k(s) \qquad P(s) = \sum_{k=1} p_{m(k)} h_k(s)$$

  - g, h are basis functions for profiles. $l(k)$ and $m(k)$ convert index values.
  - Assume Signals are magnetic diagnostics.

Jacobian - continued

- The magnetic diagnostic signals are linear in the currents.
- For external current parameters:

$$\frac{\partial y_{M\,i}}{\partial a_{ec\,k}} \approx S_{M\,i}((\hat{\mathbf{a}}_{ec\,k}, 0, \mathbf{a}_{pp}, \mathbf{a}_{other}), \mathbf{X})/\sigma_i$$

  - Doesn't depend on approach to equilibrium. Compute once, and don't update.
- For the current profile parameters:

$$\frac{\partial y_{M\,i}}{\partial a_{cp\,k}} \approx S_{M\,i}((0, \hat{\mathbf{a}}_{cp\,k}, \mathbf{a}_{pp}, \mathbf{a}_{other}), \mathbf{X})/\sigma_i$$

  - Depends on flux surface shape. Update as the equilibrium iteration proceeds.
- These two are examples of what I call "Separately Computable" Jacobian elements. We use physical knowledge about how the signals depend on the parameters.

Jacobian - Finite Difference

- Define the iteration operator T for the equilibrium solver:
  - Takes a state $\mathbf{X}_0$ and iterates it with the equilibrium solver m times, using the normalized parameters **a**.
  - $T^m(\mathbf{a},\mathbf{X}_0)=\mathbf{X}_m$
- Jacobian: $$J \approx \frac{\partial \mathbf{y}_M}{\partial \mathbf{a}} \equiv \frac{\mathbf{y}_M(\mathbf{a}+\delta\mathbf{a},T^m(\mathbf{a}+\delta\mathbf{a},\mathbf{X})) - \mathbf{y}_M(\mathbf{a},T^m(\mathbf{a},\mathbf{X}))}{\delta\mathbf{a}}$$
  - m = 0: fixed geometry. (Probably won't work for pressure profile parameters.)
  - m ≥ 1: allows geometry to change.
  - m -> Infinity: Using fully converged equilibria to compute Jacobian. Implemented with STELLOPT.
- How important is accuracy of signal computation?

Jacobian - Broyden Update

- Assume we started at parameter values $\mathbf{a}^n$, with computed signals $\mathbf{y}_M(\mathbf{a}^n, \mathbf{X}(\mathbf{a}^n))$. A new set of parameters $\mathbf{a}^{n+1}$ are computed, and equilibrium $\mathbf{X}$'s iterated some more toward convergence, and new signal $\mathbf{y}_M(\mathbf{a}^{n+1}, \mathbf{X}(\mathbf{a}^{n+1}))$ computed. We would like to update the Jacobian $\mathbf{J}^n$ we computed earlier, so that it will satisfy:

$$\mathbf{J}^{n+1} \cdot (\mathbf{a}^{n+1} - \mathbf{a}^n) = \mathbf{y}_M(\mathbf{a}^{n+1}, \mathbf{X}(\mathbf{a}^{n+1})) - \mathbf{y}_M(\mathbf{a}^n, \mathbf{X}(\mathbf{a}^n))$$

- The new Jacobian gets the most recent change correct.
- Best way to do this is the Broyden update:

$$\mathbf{J}_B^{n+1} = \mathbf{J}^n + \frac{(\delta\mathbf{y}_M - \mathbf{J}^n \cdot \delta\mathbf{a}) \otimes \delta\mathbf{a}}{\delta\mathbf{a} \cdot \delta\mathbf{a}}$$

- Root find with Broyden update - similar to secant method in one dimension.

VMEC Flow Diagram