On Real Time Forecasts (RTF) of Tokamak Discharges[1]

Leonid E. Zakharov[1]

[1] *Princeton Plasma Physics Laboratory, MS-27 P.O. Box 451, Princeton NJ 08543-0451*

JET Seminar,

December 9, 2004, Culham, UK

[1] **This work is supported by US DoE contract No. DE–AC020–76–CHO–3073.**

Abstract

This talk discusses the possibility to extend the existing experience with the real time equilibrium reconstruction by linking it with transport simulations and, thus, to approach the RTF of tokamak discharges. Unlike transport analysis codes (similar to "yesterday" weather analysis) or predictive codes ("next month" weather predictions), RTF targets a forecast of the plasma regime, e.g., in 0.1 τ_e *(like the "next hour" weather predictions).*

Three components, crucial for RTF are discussed: (a) fast equilibrium calculations, (b) fast transport calculations for stiff models, and (c) computer assisted control of numerical codes and their communications, documentation, maintenance and interaction with other codes and drivers.

In this regard, (a) a new set of linearized equilibrium equations have been derived; (b) a shooting technique for stiff transport models has been tested and shown to have two orders of magnitude faster convergence than the conventional implicit scheme; and (c) the CodeBuilder software for controlling the codes and communications was used for ESC-ASTRA-DCON-BALLLON code system as a RTF prototype.

Abstract

A new approach for solving equilibrium problems in both 2- and 3-D configurations is proposed. It uses Jacobian as a prescribed function of flux coordinates. Compared to specification of poloidal (or toroidal) angle or other relationships between $r(a,\theta,\zeta), z(a,\theta,\zeta)$ *coordinates, prescribing Jacobian leads to a significant simplification of equilibrium problems.*

The resulting equilibrium equations have the first (rather than second) order differential . Their linear perturbations with respect to flux coordinates provides a fast Newton scheme for equilibrium calculations.

For 3-D configurations the approach is unique in automatic fulfillment of Hamada constrains on the resonant surfaces.

Contents

1 Three components of the Real Time Forecast (RTF) code

RTF$\equiv$ a transport simulation code running faster than the discharge

RTF challenges near term predictions (in fraction of τ_E) during the tokamak discharge.

- *RTF does not exists yet, but seems to be possible*
- *It would extend EFIT/rtEFIT idea and experience to more “intellegent” prediction and control of a tokamak regime*
- *It would include an adaptation of basic principle transport models with tuning up by the discharge data flow*
- *In its turn, the real time discharge simulation can serve as a “filter” in the data flow and a generator of new signals for feed-backing the regime.*

Speed is the target

RTF is a challenge for the code development

It requires progress, at least, in three areas:

1. *Fast ($<$ 100 msec/eq) equilibrium calculation algorithms and routines*
2. *Fast ($<$ 100 msec/eq) transport calculation algorithms (especially for stiff transport models)*
3. *Special software environment for:*
   - *the code development, its maintenance and documentation*
   - *establishing a rigorous code control and control of its communications between both non- and synchronized processes*
   - *plugging in and out the RFT components with minimal (and non-intrusive) requirements for independently developed components, at the level of executables ("Code talking") rather than level of programming;*

     diametrically different approach to philosophy of modules and libraries (e.g., NTCC).

ESC was developed using CodeBuilder. The code system is controlled by its routines

As a prototype of RTF zcb-ESC-ASTRA-DCON-BALLLON:

1. *uses fastest so far equilibrium algorithms, but*
2. *still standard (and slow for stiff models) transport calculation algorithms,*
3. *uses a special software (`zcb, zhb, zdb`), which is based on first principles for*
   - *the code development, its maintenance (`zdb`) and documentation (`zhb`) (so far only for ESC)*
   - *controlling the codes, their communications (with interactive access to control parameters), and orginizing the data (`zcb`)*
   - *using its components*

   (a) in conventional (subroutine, libraries) manner as well as

   (b) paralles processes,

   (c) "code talking"

RTF idea originated from experience with 4 codes system and $\simeq$ 50 other separate codes

ESC-ASTRA interface is based on principles consistent RTF

Both codes have interactive access to all control parameters.

“Code talking” communication (through the shared memory) results in:

1. *A simple linking by only 2 additional C-files for ASTRA. They contain*
   (a) *routines setting up the code talking and launching ESC,*
   (b) *with reconstruction routines from equilibrium output data (ESI interface, covering needs of transport, stability, particle orbits, and gyro-kinetic codes).*
2. *Elimination of name conflicts*
3. *Insensitivity to independent maintenance of two codes*
4. *Automatic consistency of executables of different versions of both codes.*

For its implementation, “Code talking” requires CodeBuilder-like software

Linearization is the fastest method of solving equilibrium equations

Grad-Shafranov equation is ready for a Newton scheme

$$\Delta^*\bar{\Psi} \equiv \frac{\partial^2\bar{\Psi}}{\partial r^2} - \frac{1}{r}\frac{\partial\bar{\Psi}}{\partial r} + \frac{\partial^2\bar{\Psi}}{\partial z^2} = -r^2 P - T, \quad P \equiv \frac{d\bar{p}}{d\bar{\Psi}}, \quad T \equiv \bar{F}\frac{d\bar{F}}{d\bar{\Psi}}. \tag{2.1}$$

Its linearization is trivial

$$\bar{\Psi} = \bar{\Psi}_0(a) + \psi(a,\theta), \quad \Delta^*\psi + r^2\frac{dP}{d\bar{\Psi}_0}\psi + \frac{dT}{d\bar{\Psi}_0}\psi = -\Delta^*\bar{\Psi} - r^2 P - T. \tag{2.2}$$

Its differential operator is not significantly different from original Δ^*.

Complimented with coordinate advancing equations for flux coordinates $r(a,\theta), z(a,\theta)$

$$a \rightarrow a + \xi, \quad \xi \equiv -\frac{\psi}{\bar{\Psi}_0'}, \quad r \rightarrow r + r_a'\xi + r_\theta'\sigma, \quad z \rightarrow z + z_a'\xi + z_\theta'\sigma, \tag{2.3}$$

it leads to a fast Newton numerical scheme.

ESC uses this approach for solving nonlinear equilibrium problems since 1996

Advancing r, z for linearized GSh equation contains a hidden problem

Obtaining new coordinates requires differentiation of old ones with respect to a

$$\xi \equiv -\frac{\psi}{\bar{\Psi}_0'}, \quad r \rightarrow r + r_a'\xi + r_\theta'\sigma, \quad z \rightarrow z + z_a'\xi + z_\theta'\sigma \tag{2.4}$$

and deteriorates the differential properties of r, z (σ is an arbitrary function, which affects the definition of the poloidal angle θ).

In practice, cubic splines are used for recovering functional properties of r, z

Equations written for $r(a, \theta), z(a, \theta)$ themselves are desirable

Differential operator in GSh equation for flux coordinates is nonlinear

In flux coordinates, where $\bar{\Psi} = \bar{\Psi}(a)$, GSh equation has a reduced form

$$\bar{G} \equiv \frac{1}{\sqrt{g}} \left(\frac{g_{\theta\theta}}{\sqrt{g}} \bar{\Psi}'_a \right)'_a - \frac{1}{\sqrt{g}} \left(\frac{g_{a\theta}}{\sqrt{g}} \bar{\Psi}'_a \right)'_\theta + P + \frac{T}{r^2} = 0,$$

$$g_{\theta\theta} \equiv r'^2_\theta + z'^2_\theta, \quad g_{a\theta} \equiv r'_a r'_\theta + z'_a z'_\theta, \quad g_{aa} \equiv r'^2_a + z'^2_a, \tag{2.5}$$

$$\sqrt{g} \equiv rD, \quad D \equiv r'_a z'_\theta - z'_a r'_\theta.$$

The diffculties are related mostly to $\sqrt{g}$ in the denominator:

1. It gives major nonlinearity,
2. It make equations difficult to linearized (10 terms comes from $\sqrt{g}$)
3. It makes equation to be second order.

To my knowledge, linearization was never performed or shown to be practical

Poloidal angle is arbitrary in flux coordinates

Choices used in equilibrium codes

1. Simplest ESC definition
$$z = z_0(a) + b(a)\sin\theta \tag{2.6}$$
2. Polar-like
$$r = R_0 + \rho(a,\theta)\cos\theta, \quad z = Z_0 + \rho(a,\theta)\sin\theta \tag{2.7}$$
3. Equal-arc distribution of θ
$$g_{\theta\theta} \equiv r_\theta'^2 + z_\theta'^2 = g_{\theta\theta}(a) \tag{2.8}$$
4. VMEC optimized poloidal angle based on minimization of some functionals or Fourier spectra.

All lead to same kind of nonlinear equilibrium equations for r, z

Jacobian $\sqrt{g} \equiv J(a,\theta)$ can be prescribed, rather than θ

New set of equations for $r(a,\theta), z(a,\theta)$ becomes very simple

$$J\bar{G} = (g_{\theta\theta}Q)'_a + (g_{a\theta}Q)'_\theta + JP + \frac{J}{r^2}T = 0, \quad Q \equiv \frac{\bar{\Psi}'}{J}, \qquad (2.9)$$

$$r(r'_a z'_\theta - z'_a r'_\theta) = J$$

and represents the first order equations for unknowns.

Prescribing J, rather than θ, removes major nonlinearity

For limited spectrum in parameterization of r, z a variational form is available

Functional for minimization

$$W_J = \int \left[\frac{r_\theta'^2 + z_\theta'^2}{J^2} (2J - \sqrt{g}) \bar{\Psi}'^2 - 2J\bar{p} - \frac{\sqrt{g}}{r^2} \bar{F}^2 + \lambda(\sqrt{g} - J) \right] da d\theta. \quad (2.10)$$

Its variation with respect to two independent test functions, i.e., $r(a, \theta), z(a, \theta)$ and $\lambda(a)$ reproduces both the condition equilibrium equation with a prescribed Jacobian (2.5).

Substitution of $J = \sqrt{g}$ reduces it to Khait/VMEC variational form.

Derivation of LEE is easy from minimization of W_J

In Hamada coordinates $J = J(a)$ LEE have the simplest form

In Hamada coordinates

$$Q = \frac{\bar{\Psi}'}{J} = Q(a), \quad \delta Q = \frac{\delta\bar{\Psi}'}{J} = \delta Q(a) \tag{2.11}$$

and all metric coefficients can be calculated in Fourier space in a fast manner.

Other special choices of J not affecting the first order of equations

1. Hamada-like coordinates

$$J = D(a)r. \tag{2.12}$$

2. PEST coordinates

$$J = f(a)r^2. \tag{2.13}$$

3. Jacobian as a function of flux coordinates

$$J = J(r, z, a), \quad J = J(r, z, r'_\theta, z'_\theta, a). \tag{2.14}$$

Solving equilibrium in Hamada coordinates makes the solution directly pluggable to stability codes

First order 3D equilibrium equations are similar to 2D version

In flux coordinates a, θ, ζ the poloidal $\bar\Psi(a)$ and toroidal $\bar\Phi(a)$ fluxes and corresponding currents $\bar F(a), \bar J(a)$ (bar distinguishes $\bar J$ from Jacobian) are functions of a only.

For 3D use of straight field line coordinates is essential. Then the full set of equations

$$
\begin{aligned}
\bar J' + \nu'_\theta &= \left[-g_{\theta\theta}Q + g_{\theta\zeta}\bar Q\right]'_a + \left[g_{a\theta}Q - g_{a\zeta}\bar Q\right]'_\theta, \\
\bar F' + \nu'_\zeta &= \left[-g_{\zeta\theta}Q + g_{\zeta\zeta}\bar Q\right]'_a + \left[g_{a\theta}Q - g_{a\zeta}\bar Q\right]'_\zeta, \\
\bar p' J &= -\bar\Phi'(\bar F' + \nu_\zeta) + \bar\Psi'(\bar J' + \nu_\theta), \\
\sqrt{g} &= J.
\end{aligned} \tag{2.15}
$$

Equations have only first order derivatives with respect to a and the unknowns are

$$
r(a,\theta,\varphi), \quad z(a,\theta,\varphi), \quad \varphi(a,\theta,\varphi), \quad Q \equiv \frac{\bar\Psi}{J}, \quad \bar Q \equiv \frac{\bar\Psi}{J}. \tag{2.16}
$$

3D-function ν does not represent a 3D unknown

Technically LEE in 3D case are similar to 2D version

Symbolically, the perturbations of equilibrium in flux coordinates is

$$\begin{aligned}
&\delta(\bar{J}' + \nu'_\theta) - \delta\left[-g_{\theta\theta}Q + g_{\theta\zeta}\bar{Q}\right]'_a - \delta\left[g_{a\theta}Q - g_{a\zeta}\bar{Q}\right]'_\theta = \Delta_\theta,\\
&\delta(\bar{F}' + \nu'_\zeta) - \delta\left[-g_{\zeta\theta}Q + g_{\zeta\zeta}\bar{Q}\right]'_a - \delta\left[g_{a\theta}Q - g_{a\zeta}\bar{Q}\right]'_\zeta = \Delta_\zeta,\\
&J\delta\bar{p}' \qquad + \delta[\bar{\Phi}'(\bar{F}' + \nu_\zeta)] - \delta[\bar{\Psi}'(\bar{J}' + \nu_\theta)] = \Delta_a,\\
&\delta\sqrt{g} \qquad = J - \sqrt{g},
\end{aligned} \tag{2.17}$$

where $\Delta_a, \Delta_\theta, \Delta_\zeta$ represent the corresponding discrepancy in nonlinear equations.

3D case contains an additional unknown function $\varphi(a,\theta,\zeta)$ and an additional equation

Island-free equilibrium requires absence of resonant harmonics in J

The radial balance equation determines the oscillatory part of the current density ν

$$\bar{p}'J = -\bar{\Phi}'(\bar{F}' + \nu_\zeta) + \bar{\Psi}'(\bar{J}' + \nu_\theta) \qquad (2.18)$$

as a solution of the magnetic differential equation $(\mathrm{B} \cdot \nu) = \bar{p}'J$. It can be solved only in the case when Jacobian does not contain resonance Fourier harmonics at the rational magnetic surfaces.

This Hamada condition creates problems for calculating island-free stellarator equilibria

1. Lab-coordinate code (like PIES) resolve the Hamada problem by island structures.
2. Flux-coordinate codes neglect singularity $\nu \propto 1/x$ eventually created in nonlinear solution.

The approach based on prescribed Jacobian has no difficulty with Hamada condition. E.g.,

Hamada Jacobian resolves the Hamada problem automatically

Inverted transport coefficients promise a way for fast transport calculations

Conventional (κ based) formulation of transport equations, like

$$\frac{\partial T}{\partial t} - (\kappa T'_x)'_x = S(x,t), \quad \kappa = \kappa(x,T,T'_x,t), \tag{3.1}$$

has problems with convergence for “stiff”models, when at some point

$$\left.\frac{\partial \kappa}{\partial T'_x}\right|_{T'_x+\delta T'_x} \gg \left.\frac{\partial \kappa}{\partial T'_x}\right|_{T'_x}. \tag{3.2}$$

Another, Γ based, formulation with reversed

$$\frac{\partial T}{\partial t} + \Gamma'_x = S(x,t), \quad T'_x = T'_x(x,T,\Gamma,t), \tag{3.3}$$

makes solution fast converging.

A straighforward shooting technique was tested with IFS-PPPL model and for any time step and, independent of stiffness, shown only 20-50 times slower convergence than solving the simplest diffusion equation with κ=const.

Accellaration of 10-100 times is expected with further studies of Γ based schemes

`zcb`, `zhb`, `zdb` is a private development and a product of CbResearch

From conventional code sources and additional descriptive files, `zcb` generates a Cb-code with well organized code control and communications.

`zcb:`

- *Was triggered by W.Sadowski at a workshop on code talking (August 1996, DoE).*
- *Was designed and written in 1996-1997.*
- *Initially targeted C- and FORTRAN- codes. FORTRAN was dropped from updating since 1998.*
- *Was three times reported to PPPL (97,99,04),*
- *Was supplemented by the documentation and maintenance system for numerical codes in 2001.*
- *Was supplemented with the OnLine Help system in 2003,*
- *Resulted in a new vision on the CodeBuilder in 2003.*

A license agreement is required for its use (DoE regulations)

Rising entropy is unavoidable in large numerical codes

Control parameters user has in mind	Typical FORTRAN namelist
promotion to AL	`igrid`
promotion to group leader	`rleft`
major monetary award	`rright`
promotion within the rank	`zbotto`
...	`ifcoil`
...	`iecoil`
...	...
...	`af2`
...	`fcturn`
minor disciplinary actions	`he`
suspension for a week	`ecid`
layoff	`vsid`
torture	`rvs`
electric chair	`zvs`

Unstructured lists, typical for programming, is a source of entropy

The total number N_0 of possible sporadic matches is

$$N_0 = n!, \quad S_0 = \ln N_0 \simeq n(\ln n - 1) + \frac{1}{2}\ln(2\pi n) \tag{4.1}$$

Now, suppose both sides subdivide each set on n/k matching sections with k elements in each.

Two cases are possible
1. Sporadic permutations in each of sections with N_1

$$\begin{aligned} N_1 &= \underbrace{k!k!\dots k!}_{n/k \text{ times}} = (k!)^{\frac{n}{k}}, \\ S_1 &= \ln N_1 \simeq \frac{n}{k}(k\ln k - k) = n(\ln k - 1) \end{aligned} \tag{4.2}$$

Simple grouping of variables has a little effect on entropy

being deceptively "efficient" at small n

Organization is crucial for reducing the entropy

Matching one section after another in sequence

$$N_2 = \underbrace{k! + k! \ldots + k!}_{n/k \text{ times}} = (k!)^{\frac{n}{k}}, \qquad S_2 = \ln N_2 \simeq (k-1)\ln(k-1) + \ln n \ll S \tag{4.3}$$

Any coherency results in dramatic reduction in entropy applicable for any value of n.

Coherency requires a rigorous management system

Nested structures automatically provide a basic organization

```
type_<name0>{
  list of control parameters;
   content;
  type_<name1>{
    list of control parameters;
     content;
  }
  type_<name2>{
    list of control parameters;
     content;
  }
   content;
}
```

and consistency in processing control parameters

(a) People always think in terms of nested structures

(b) People are limited in maintaining evolution of nested structures

Virtual structures inside the code source are the working ground of `zcb`

Currently they are specified (by the author) with the C-preprocessing instructions, i.e

```
#ifndef blb_<NameB0>          <NameB0>{
#ifndef mzl_<NameM>             <NameM>{
#ifndef blb_<NameB1>              <NameB1>{
........
#endif                             }
#ifndef stg_<NameS0>              <NameS0>{
#endif                             }
#ifndef stg_<NameS1>              <NameS1>{
#endif                             }
#endif                           }
#endif                         }
```

without disturbing the code.

The CodeBuilder introduces the irreducible set of 3 types of structural elements
capable of describing both nested-ness and parallel threads in the code

CodeBuilder = structure recognition + drivers for controlling its elements

In its existing `zcb` implementation it

1. *Is based in 3 types of structural elements reflecting nested structures and virtual parallel processing and separation of code communications from processing the information.*
2. *Provides structural identity of*
   (a) *the code control with*
   (b) *its communications,*
   (c) *I/O data base,*
   (d) *OnLine Help*

The code documentation system maintained by CB implements:

1. *separation of the code source and its documentation*
2. *structural identity of the src- and doc-files*
3. *full capacity of TeX/LaTeX for code documentations*
4. *hyperlinks for navigation.*

CB is intrinsically a non-intrusive technology based on first principles

OnLine Help and Data Base file structure are determined by the code structure

6 command lines create the standard set of subdirectories and templates for the code and communication control files and for OnLine Help, e.g,

```
wrk> mkcbL                     makes link to zcb
wrk> zcb Src/t.c -i t          makes the Code Control file
wrk> mv Out/t.Cb In
wrk> zcb t.Cb                  displays the Control Panel
                               generates templates for
                               Communication Control files
                               and OnLine Help
```

< **inspect and edit names of directories inside** `t.Cb`>

```
wrk> zcb t.Cb -m Src/t.Src     makes final command Mkt
wrk> zhb Src/t.Src             .dvi, .ps versions of Help
```

Steps are prompted by calling `zcb` with no arguments

Communications of any section can be described in `.CmC` files

Communication control files contain a list of different communications as wells as the list of objects inside each of them.
Example of sections, e.g.

```
Tbl_Link2dB(){}
Hlp{}
Plt{}
BRc_Link2dB(){}
BSv_Link2dB(){}
BnrILink2dB(){}
BnrOLink2dB(){}
AscILink2dB(){}
AscOLink2dB(){}
"d"= (int) "dd";
"c"= (str*8)"cc";
"a"= (dbl) "aa";
"f"= (Chr*8)"cc";
```

```
nn=(dbl){
   "nn1"={
     "n11"="nn11";
     "n12"="nn12";
     "n16"="nn16";
   }
   "nn2"={
     "n22"="nn22";
     "n26"="nn26";
   }
 }
```

All sections, as well as examples of object descriptions are listed in templates

`zcb` allows for distributed I/O without loosing the consistency

Each section in Cb-code may have its own set of communications.

CntrPanel has two sides: `<Output>` and `<Input>`:

- *The* `<Input>`*-dB manager provides consistency in the distributed input data.*
- *The* `<Output>`*-dB manager provides consistency in the distributed outputs*
- *All sections of Cb-code are controlled in the same manner by switches on the Control Panel*

Structuring the code communications and an independent software like `zcb` are crucial for consistency

Code maintenance documentation is different from the OnLine Help

Existing methods of documenting the codes:

1. Comments inside the code: limited in all capacities, contaminate the source code.
2. Web by D.Knuth: unlimited in typesetting capacities, but contaminate the source code, and non functional (no navigation, essentially a nice printing),
3. `html` based: good in navigation, incapable to reproduce a single formula.
4. Separate manuals: always outdated.
5. . . . .

No one approach provides consistency of the state of the code and its documentation.

The CodeBuilder separates source code and documentations while reinforcing their identical virtual structures

In Cb-code:

1. All routines and global variables are positioned in a Cb-structure based on the same three types of sections as in the communication structure.
2. The structure of the document files is automatically generated and maintained based the current structure of the source code.
3. Only a skeleton of the routine is provided. A special place is allocated for LaTeX documentation.
4. Use of external and global variables by each routine is analyzed and listed, if routine and a variable belong to different "virtual" processors.

The approach results in:

1. Clean source codes with short names of variables.
2. Use of full capacity of TeX/LaTeX for documentations.
3. Easy navigations (at present, in the `.dvi` form).
4. Crucial functionality in organizing use of external variables.

The CodeBuilder maintained documentation is always consistent with the state of the code

The structure of Cb-code is the structure of its communications

Structuring and separation of communications from other programming provides:

1. *Transparent and uniform control of code and all its sections.*
2. *Transparency, uniformity, unambiguity of all kinds of communications.*
3. *Resolution of so-called "legacy" problem. In all codes communications and code control fail first. There is no such issue in the Cb-codes.*
4. *Resolution of the "code talking" problem. Communications programmed by a software are automatically consistent.*
5. *Resolution of the code distribution problem. No one control parameter in the Cb-code is hidden from the user and every one is explained.*
6. *Collective work on common projects becomes trivial as soon as the code control is distributed over its logical section.*
7. *. . .*

`zcb` makes programming essentially instantaneous.

Three types of `zcb` structuring and separation of communications and processing cannot

be avoided in any meaningful attempt to solve the problem

The CodeBuilder separates communications from other processing

Accordingly, the full source of Cb-code consists of

1. Conventional source files.
2. `<CodeName>.Src` file with a list of the conventional source files (the make file was abandoned).
3. `<CodeName>.Cb` Code Control file.
4. `<SectionName>.CmC` Communication Control files.

OnLine Help files are generated by `zhb` from the full source.

In the CodeBuilder the communications are described in the `.CmC` files, rather than programmed.

The CodeBuilder is consistent with philosophy of C having no I/O operators

Since May, 2004, the first steps in assembling components of RTF were performed

1. ESC-ASTRA launched in the (equilibrium reconstruction)/(transport simulation) mode.
2. LEE equations are programmed in a separate code, which is under debugging
3. “Code talking” regime was tested and is ready to implementation into CodeBuilder
4. A scheme based on Γ flux was first tested as an approach for fast transport calculations.

Colalboration with JET plays the exceptional role in this development