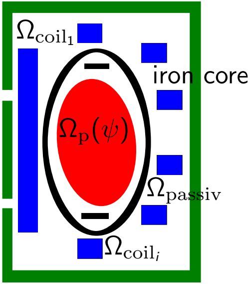
Quasi-static Free-Boundary Equilibrium of Toroidal Plasma: Computational Methods and Applications

J. Blum, C. Boulbe, B. Faugeras, H. Heumann, [1]
J.-M. Ané, S. Brémond, V. Grandgirard, P. Hertout, E. Nardon, [2]

[1] TEAM CASTOR, INRIA and Université de Nice Sophia Antipolis, France
[2] CEA, IRFM, Saint-Paul-lez-Durance, France

Modeling and Numerical Methods for Hot Plasmas II
Bordeaux, 12-14 October 2015

Quasi-static Free-Boundary Equilibrium of Toroidal Plasma



inside plasma and non-conducting parts:

$$\mathbf{grad}\, p = \mathbf{J} \times \mathbf{B}\,, \quad \mathbf{div}\,\mathbf{B} = 0\,, \quad \mathbf{curl}\,\frac{1}{\mu}\mathbf{B} = \mathbf{J}\,,$$

in all other conducting structures:

$$\partial_t \mathbf{B} = \mathbf{curl}\,\frac{1}{\sigma}(\mathbf{J}_{src} - \mathbf{J})\,, \quad \mathbf{div}\,\mathbf{B} = 0\,, \quad \mathbf{curl}\,\frac{1}{\mu}\mathbf{B} = \mathbf{J}\,.$$

with toroidal symmetry: (ψ toroidal comp. of $r\,\mathbf{A}$, $\mathbf{B} = \mathbf{curl}\,\mathbf{A}$)

$$-\nabla\left(\frac{1}{\mu[\psi]r}\nabla\psi\right) = \begin{cases} rp'(\psi) + \frac{1}{\mu_0 r} ff'(\psi) & \text{in } \Omega_{\mathrm{p}}(\psi)\,, \\ \frac{n_i V_i(t)}{R_i S_i} - 2\pi \frac{n_i^2}{R_i S_i^2} \int_{\Omega_{\mathrm{coil}_i}} \frac{\partial \psi}{\partial t}\, drdz =: \frac{I_i(\psi)}{S_i} & \text{in } \Omega_{\mathrm{coil}_i}\,, \\ -\frac{\sigma}{r}\frac{\partial \psi}{\partial t} & \text{in } \Omega_{\mathrm{passive}}\,, \\ 0 & \text{elsewhere}\,, \end{cases}$$

with p' and ff' known. f toroidal component of $r\,\mathbf{B}$.

Infinite domain, plasma domain $\Omega_{\mathrm{p}}(\psi)$ unknown, circuit equations, iron core.

Genealogy

FEM for free-boundary equilibrium in tor. symm.:

Challenges	SCED [BFT81]	Proteus [ABB87]	CEDRES++ [G99]
iron core	+++	+++	+++[Boulbe '11]
infinite domain	not	+++ [ABB86]	+++ [G99]
free boundary	+++	+++	+++
circuit equations	version Blum	version Albanese	Blum/Boulbe/Nardon '14
Newton	+++	+++	+++ [Hetal15]
inverse problem	static [B89]	stat.	stat. & dynam. [Hetal15]

[BFT81] J. Blum, J. Le Foll, B. Thooris, *The self-consistent equilibrium and diffusion code SCED*, CPC, 1981.

[ABB86] R. Albanese, J. Blum, O. Barbieri, *On the solution of the magnetic flux equation in an infinite domain*, 8th Europhys. Conf. Comp. in Plasma Phys., 1986.

[ABB87] R. Albanese, J. Blum, O. Barbieri, 12th Conf. on Num. Simul. of Plasma, 1987.

[B89] J. Blum, *Numerical simulation and optimal control in plasma physics*, 1989.

[G99] V. Grandgirard, *Modelisation de l'equilibre d'un plasma de tokamak*, PhD., 1999.

[Hetal15] H.H. et al., *Quasi-static free-boundary equilibrium of toroidal plasma with CEDRES++ ...*, Journal of Plasma Physics 2015.

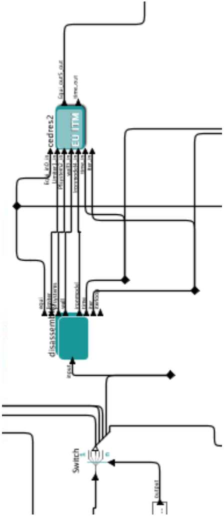
CEDRES++ & FEEQS.M (C. Boulbe, B. Faugeras, H. H.)

Applications, focus on control and scenarios:

- static equlibrium calculations for given currents,
- currents calculation for given static equlibrium,
- evolution of equilibrium calculation for given voltages,
- voltage evolution calculation for given equil. evolution,
- not real-time reconstruction, not for MHD instability

Codes at CASTOR/CEA/UNS

- CEDRES++
  - *Couplage Equilibre Diffusion Resistive pour l'Étude des Scenarios*
  - productive code written in C++;
  - in use at CEA for set up experiment scenarios at upcoming WEST;
  - interfaced for ITM (integrated tokamak modeling) Kepler Platform;
- FEEQS.M



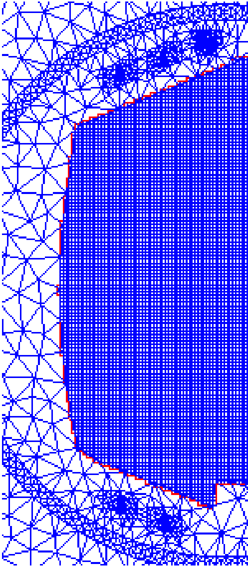
CEDRES++ & FEEQS.M (C. Boulbe, B. Faugeras, H. H.)

Applications, focus on control and scenarios:

- static equlibrium calculations for given currents,
- currents calculation for given static equlibrium,
- evolution of equilibrium calculation for given voltages,
- voltage evolution calculation for given equil. evolution,
- not real-time reconstruction, not for MHD instability

Codes at CASTOR (C. Boulbe, B. Faugeras, H. H.)

- CEDRES++
- FEEQS.M
  - *Finite Element EQuilibrium Solver in Matlab*
  - same core functionality as CEDRES++;
  - test and fast prototyping environment in MATLAB;
  - high performance, thanks to vectorization;
  - new applications;
  - improve and extend numerical methods;
  - simple code distributions;
  - intern and master projects;



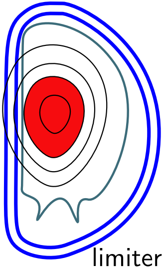
Outline

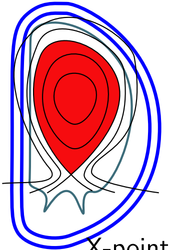
What's next?

Direct Static Problem (for flux $\psi(r, z)$)

$$-\nabla \cdot \left(\frac{1}{\mu r}\nabla\psi\right) = \begin{cases} r\,p'(\psi) + \frac{1}{\mu_0 r} ff'(\psi) & \text{in } \Omega_{\mathrm{p}}(\psi)\,, \\ \frac{I_i}{S_i} & \text{in } \Omega_{\mathrm{coil}_i}\,, \\ 0 & \text{elsewhere}\,, \end{cases}$$

$$\psi(0, z) = 0\,, \qquad \lim_{\|(r,z)\|\to+\infty} \psi(r, z) = 0\,,$$





Iron (μ_{Fe} experimental data):

$$\mu = \mu(r, |\nabla\psi|) = \begin{cases} \mu_{\mathrm{Fe}}(|\nabla\psi|^2 r^{-2}) & \text{in } \Omega_{\mathrm{iron}}\,, \\ \mu_0 & \text{elsewhere}\,. \end{cases}$$

Model for current density ($\alpha, \beta, \gamma, r_0$ given):

$$p'(\psi) \approx S_{p'}(\psi_{\mathrm{N}}) = \frac{\beta}{r_0}(1 - \psi_{\mathrm{N}}^{\alpha})^{\gamma}\,,$$

$$ff'(\psi) \approx S_{ff'}(\psi_{\mathrm{N}}) = (1-\beta)\mu_0 r_0 (1 - \psi_{\mathrm{N}}^{\alpha})^{\gamma}\,,$$

$$\psi_{\mathrm{N}}(r, z) = \frac{\psi(r, z) - \psi_{\mathrm{ax}}[\psi]}{\psi_{\mathrm{bd}}[\psi] - \psi_{\mathrm{ax}}[\psi]}\,, \qquad \begin{aligned} \psi_{\mathrm{ax}}[\psi] &:= \psi(r_{\mathrm{ax}}[\psi], z_{\mathrm{ax}}[\psi])\,, \\ \psi_{\mathrm{bd}}[\psi] &:= \psi(r_{\mathrm{bd}}[\psi], z_{\mathrm{bd}}[\psi])\,. \end{aligned}$$

Inverse Static Problem (for currents I_i)

Objective and Regularization

$$K(\psi) := \frac{1}{2}\sum_{i=1}^{N_{\mathrm{desi}}}\big(\psi(r_i, z_i) - \psi(r_{\mathrm{desi}}, z_{\mathrm{desi}})\big)^2 \quad , \quad R(I_1, \ldots, I_L) \quad := \sum_{i=1}^{L}\frac{w_i}{2}I_i^2$$

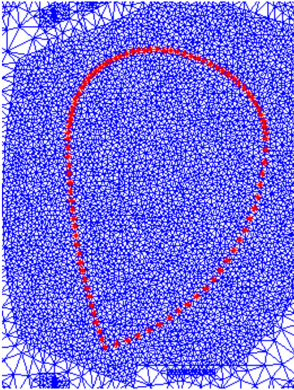
Optimal Control/Inverse Problem:

$$\min_{\psi, I_1, \ldots I_L} K(\psi) + R(I_1, \ldots I_L)$$

subject to

$$-\nabla\cdot\left(\frac{1}{\mu r}\nabla\psi\right) = \begin{cases} r S_p'(\psi_{\mathrm{N}}) + \frac{1}{\mu_0 r}S_{ff'}(\psi_{\mathrm{N}}) & \text{in } \Omega_{\mathrm{p}}(\psi)\,, \\ \frac{I_i}{S_i} & \text{in } \Omega_{\mathrm{coil}_i}\,, \\ 0 & \text{elsewhere}\,, \end{cases}$$

$$\psi(0, z) = 0\,, \qquad \lim_{\|(r,z)\| \to +\infty}\psi(r, z) = 0\,,$$



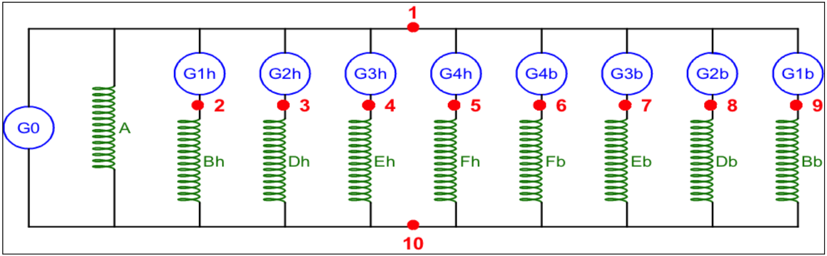
PDE-constrained optimization with non-linear constraints.

Direct Evolution Problem (for flux evolution $\psi(r,z,t)$)

$$-\nabla\cdot\left(\frac{1}{\mu r}\nabla\psi\right)=\begin{cases} rS_p'(\psi_{\mathrm{N}},t)+\frac{1}{\mu_0 r}S_{ff'}(\psi_{\mathrm{N}},t) & \text{in } \Omega_{\mathrm{p}}(\psi)\,,\\ S_i^{-1}\left(\boldsymbol{S}\vec{V}+\boldsymbol{R}\vec{\Psi}(\partial_t\psi)\right)_i & \text{in } \Omega_{\mathrm{coil}_i}\,,\\ -\frac{\sigma_k}{r}\partial_t\psi & \text{in } \Omega_{\mathrm{passive}}\,,\\ 0 & \text{elsewhere}\,,\end{cases}$$

$$\psi(0,z,t)=0\,,\qquad \lim_{\|(r,z)\|\to+\infty}\psi(r,z,t)=0\,,\quad \psi(r,z,0)=\psi_0(r,z)\,,$$

Circuit equations $\vec{I}=\boldsymbol{S}\vec{V}+\boldsymbol{R}\vec{\Psi}(\partial_t\psi)$:



where $\vec{\Psi}(\partial_t\psi)=(\ldots,\int_{\Omega_{\mathrm{coil}_i}}\partial_t\psi drdz,\ldots)$.

Inverse Evolution Problem (for volt. evolution $\vec{V}(t)$)

Objective and Regularization:

$$K(\psi(t)) := \frac{1}{2}\int_0^T \left(\sum_{i=1}^{N_{\mathrm{desi}}} \left(\psi(r_i(t), z_i(t), t) - \psi(r_{\mathrm{desi}}(t), z_{\mathrm{desi}}(t), t)\right)^2 \right) dt\,,$$

$$R(\vec{V}) := \sum_{i=1}^{L} \frac{w_i}{2} \int_0^T \vec{V}_i(t) \cdot \vec{V}_i(t) dt\,.$$

Optimal Control/Inverse Problem:

$$\min_{\psi(t), \vec{V}(t)} K(\psi(t)) + R(\vec{V})$$

subject to

$$-\nabla \cdot \left(\frac{1}{\mu r}\nabla\psi\right) = \begin{cases} r S_p'(\psi_{\mathrm{N}}, t) + \frac{1}{\mu_0 r} S_{ff'}(\psi_{\mathrm{N}}, t) & \text{in } \Omega_{\mathrm{p}}(\psi)\,, \\ S_i^{-1}\left(\boldsymbol{S}\vec{V} + \boldsymbol{R}\vec{\Psi}(\partial_t\psi)\right)_i & \text{in } \Omega_{\mathrm{coil}_i}\,, \\ -\frac{\sigma_k}{r}\partial_t\psi & \text{in } \Omega_{\mathrm{passive}}\,, \\ 0 & \text{elsewhere}\,, \end{cases}$$

$$\psi(0, z, t) = 0\,, \qquad \lim_{\|(r,z)\| \to +\infty} \psi(r, z, t) = 0\,, \quad \psi(r, z, 0) = \psi_0(r, z)\,,$$

What's next?

Weak Formulation

Find $\psi \in V$ such that

$$\mathsf{A}(\psi,\xi) - \mathsf{J}_{\mathrm{p}}(\psi,\xi) + \mathsf{c}(\psi,\xi) = \ell(\vec{I},\xi) \quad \forall \xi \in V\,.$$

with

$$V = \left\{ \psi : \Omega \to \mathbb{R}, \int_\Omega \psi^2 r\, drdz < \infty, \int_\Omega (\nabla\psi)^2 r^{-1}\, drdz < \infty, \psi_{|r=0} = 0 \right\}\,,$$

$$\mathsf{A}(\psi,\xi) := \int_\Omega \frac{1}{\mu r} \nabla\psi \cdot \nabla\xi\, drdz\,, \qquad \ell(\vec{I},\xi) := \sum_{i=1}^{N_{\mathrm{coil}}} S_i^{-1} \vec{I}_i \int_{\Omega_{\mathrm{coil}_i}} \xi\, drdz\,,$$

$$\mathsf{J}_{\mathrm{p}}(\psi,\xi) := \int_{\Omega_{\mathrm{p}}(\psi)} \left(r p'(\psi) + \frac{1}{\mu_0 r} f f'(\psi) \right) \xi\, drdz\,,$$

No integral equations in the spirit of FEM-BEM or "mariage à la mode" (Zienkiewics, Johnson, Nedelec, ...)

$\mathsf{c}(\psi,\xi) \approx \int_{\partial\Omega} \xi\, \partial_{\mathbf{n}} \psi\, dS$ taking into account boundary condition at infinity, details ....

What is Ω, what is $\mathsf{c}(\cdot,\cdot)$?

Weak Formulation, from infinite to finite

If Ω semi-circle with radius ρ [Albanese1986, Gatica1995] ($\Gamma = \partial\Omega$):

$$\begin{aligned}
\mathsf{c}(\psi,\xi) &= \int_{\partial\Omega} \xi\,\partial_{\mathbf{n}}\psi\, dS = \int_{\Gamma} \xi(\mathbf{P}_1)\,\partial_{\mathbf{n}_1}\left(\int_{\Gamma} \partial_{\mathbf{n}_2} G(\mathbf{P}_1,\mathbf{P}_2)\psi(\mathbf{P}_2)dS_2\right) dS_1 \\
&= \frac{1}{2\mu_0}\int_{\Gamma}\int_{\Gamma} \psi(\mathbf{P}_1) M(\mathbf{P}_1,\mathbf{P}_2)\xi(\mathbf{P}_2) dS_1 dS_2 \\
&= \frac{1}{\mu_0}\int_{\Gamma} \psi(\mathbf{P}_1) N(\mathbf{P}_1)\xi(\mathbf{P}_1) dS_1 \\
&\qquad + \frac{1}{2\mu_0}\int_{\Gamma}\int_{\Gamma} (\psi(\mathbf{P}_1)-\psi(\mathbf{P}_2)) M(\mathbf{P}_1,\mathbf{P}_2)(\xi(\mathbf{P}_1)-\xi(\mathbf{P}_2)) dS_1 dS_2
\end{aligned}$$

with $G(\mathbf{P}_1,\mathbf{P}_2) \approx \log(\|\mathbf{P}_1-\mathbf{P}_2\|)^{-1}$, fundamental solution of $\nabla \frac{1}{\mu_0 r}\nabla$ and

$$M(\mathbf{P}_1,\mathbf{P}_2) = \frac{k_{1,2}}{2\pi(r_1 r_2)^{\frac{3}{2}}}\left(\frac{2-k_{1,2}^2}{2-2k_{1,2}^2}E(k_{1,2}) - K(k_{1,2})\right)$$

where $\mathbf{P}_i = (r_i, z_i)$, K and E complete elliptic integrals of 1st and 2nd kind, and

$$N(\mathbf{P}_1) = \int_{\Gamma} M(\mathbf{P}_1,\mathbf{P}_2)dS_2\,, \quad k_{1,2} = \sqrt{\frac{4r_1 r_2}{(r_1+r_2)^2+(z_1-z_2)^2}}\,.$$

Weak Formulation, evolution problem

Semi-discretization in time with implicit Euler

Set $\psi^0 = \psi(t_0)$. Find $\psi^1, \dots \psi^N \in V$ approximating $\psi(t_1), \dots \psi(t_N)$ with

$$\begin{aligned}\Delta t_k \mathsf{A}(\psi^k, \xi) - \Delta t_k \mathsf{J}_{\mathrm{p}}^k(\psi^k, \xi) - \mathsf{j}^{\mathrm{ps}}(\psi^k, \xi) - \mathsf{j}^{\mathrm{c}}(\psi^k, \xi) + \Delta t_k \mathsf{c}(\psi^k, \xi) \\ = \Delta t_k \ell(\boldsymbol{S}\vec{V}(t_k), \xi) - \mathsf{j}^{\mathrm{ps}}(\psi^{k-1}, \xi) - \mathsf{j}^{\mathrm{c}}(\psi^{k-1}, \xi) \quad \forall \xi \in V\,.\end{aligned}$$

$$\mathsf{j}^{\mathrm{ps}}(\psi, \xi) := -\sum_{i=1}^{N_{\mathrm{passiv}}} \int\limits_{\Omega_{\mathrm{passive}}} \frac{\sigma_i}{r} \psi \xi \, drdz\,, \quad \mathsf{j}^{\mathrm{c}}(\psi, \xi) := \sum_{i=1}^{N_{\mathrm{coil}}} S_i^{-1} \left(\boldsymbol{R} \vec{\Psi}(\psi) \right)_i \int\limits_{\Omega_{\mathrm{coil}_i}} \xi \, drdz\,.$$

Stationary Problem:

$$\boldsymbol{A}(\mathbf{y}) = \boldsymbol{F}(\mathbf{u})\,,$$

State $\mathbf{y}$ is flux ψ;
Control $\mathbf{u}$ are currents $\vec{I}$;

Evolution Problem:

$$\boldsymbol{A}(\mathbf{y}^{k+1}) + \mathbf{m}(\mathbf{y}^{k+1}) = \boldsymbol{G}(\mathbf{u}^{k+1}) + \mathbf{m}(\widehat{\mathbf{y}}^k)\,,$$

State $\mathbf{y}^1, \dots$ is flux $\psi^1, \dots$;
Control $\mathbf{u}^1, \dots$ are voltages $\vec{V}(t_1), \dots$;

What's next?

Newton's Method, Continuous Approach

Nonlinear variational formulation

$$\mathsf{A}(\psi,\xi) + \mathsf{J}_{\mathrm{p}}(\psi,\xi) + \mathsf{c}(\psi,\xi) = \ell(\xi),$$

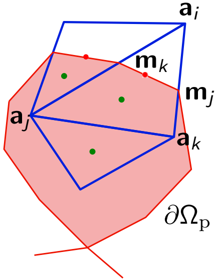
Newton iterations

$$\left(D_\psi \mathsf{A}(\psi^k,\xi) + D_\psi \mathsf{J}_{\mathrm{p}}(\psi^k,\xi)\right)(\psi^{k+1}-\psi^k) = \ell(\xi) - \mathsf{A}(\psi^k,\xi) - \mathsf{J}_{\mathrm{p}}(\psi^k,\xi) - \mathsf{c}(\psi^k,\xi),$$

- $D_\psi \mathsf{A}(\psi^k,\xi)$ simple, $D_\psi \mathsf{J}_{\mathrm{p}}(\psi^k,\xi)$ not so simple;

From shape derivatives, rearrangement, or ....

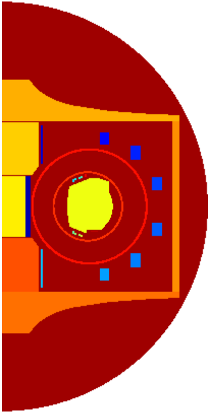
$$D_\psi \mathsf{J}_{\mathrm{p}}(\psi^k,\xi)\,\psi = \int_{\Omega_{\mathrm{p}}(\psi^k)} D_\psi j_{\mathrm{p}}(r,\psi^k(r,z))\,\psi^k \xi_h \,dr dz + \int_{\partial\Omega_{\mathrm{p}}(\psi^k)} j_{\mathrm{p}}(r,\psi^k(r,z))\,\xi \frac{\psi_{\mathrm{bd}}(\psi^k)-\psi(r,z)}{|\nabla\psi^k|}\,dS,$$



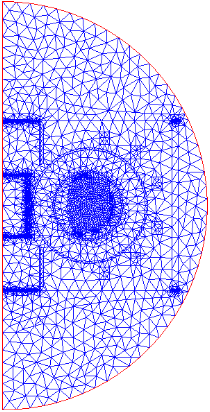
- ψ_h is discretization of ψ with linear finite elements;
- $\partial\Omega_{\mathrm{p}}(\psi_h^k)$ piecewise straight;
- barycenter quadrature rule for surface integrals
- midpoint quadrature rule for line integrals;

Newton's Method: Example I

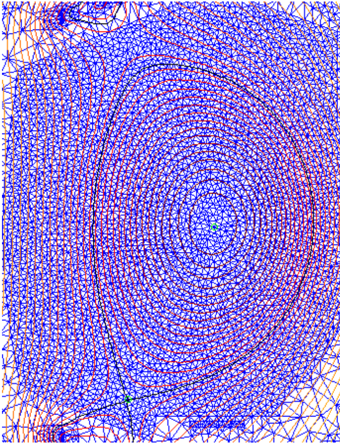
Subdomains



Mesh:



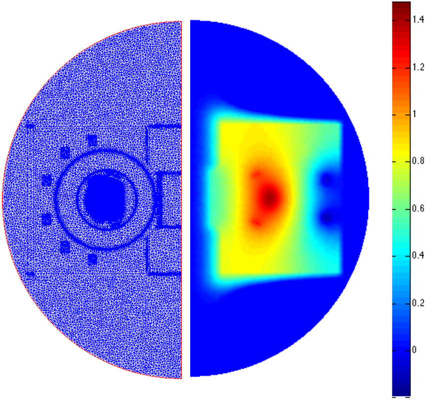
Solution:



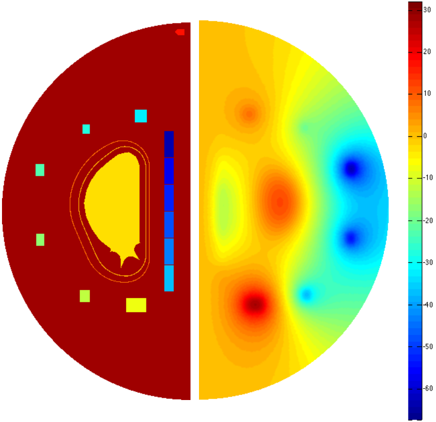
- mesh generation with TRIANGLE linear FEM;
- direct linear solver UMFPACK;

Newton's Method: Example II

West (with Iron)

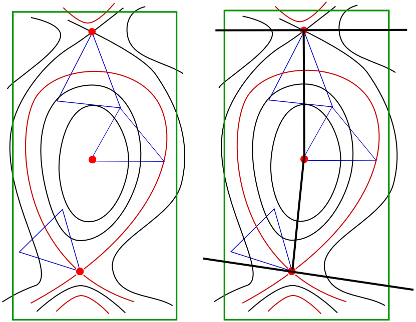


ITER (without Iron):

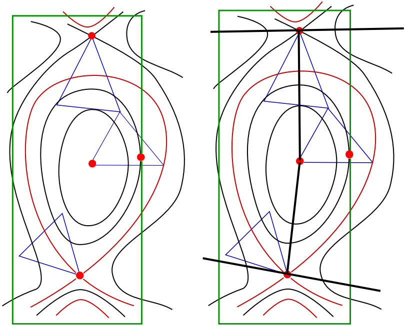


Newton's Method, Plasma Domain $\Omega_{\mathrm{p}}(\psi_h^k)$

How to determine plasma domain $\Omega_{\mathrm{p}}(\psi_h^k)$: *Interior of last closed isoline*



X-ploint plasma



limiter plasma

Algorithm:

1. find the maximum location $\mathbf{P}_{\mathrm{ax}}$ of ψ_h
2. find all discrete saddle points $\mathbf{P}_X$ of ψ_h;
3. construct excluded area, via perpendicular cut lines;
4. ψ_{bd} is the maximal value of all $\psi_h(\mathbf{P}_{\mathrm{X}})$ and ψ_h on limiter;

Newton's Method, Linearization I

Newton iterations

$$\left(D_\psi \mathsf{A}(\psi^k,\xi) + D_\psi \mathsf{J}_{\mathrm{p}}(\psi^k,\xi)\right)(\psi^{k+1}-\psi^k) = \ell(\xi) - \mathsf{A}(\psi^k,\xi) - \mathsf{J}_{\mathrm{p}}(\psi^k,\xi) - \mathsf{c}(\psi^k,\xi),$$

Problem 1: *We did observe fast but not quadratic convergence!*

Problem 2: *Gradient test failed!*

Recall Gradient test:

$$\widehat{C}(\mathbf{u}) := C(\mathbf{y}(\mathbf{u}),\mathbf{u}) \quad \text{with } \mathbf{A}(\mathbf{y}(\mathbf{u})) = \mathbf{F}(\mathbf{u}) \quad \Rightarrow \quad \nabla_{\mathbf{u}}\widehat{C}(\mathbf{u}) = \nabla_{\mathbf{u}} C(\mathbf{y}(\mathbf{u}),\mathbf{u}) + \nabla_{\mathbf{u}}\mathbf{F}(\mathbf{u})^T\mathbf{p} \quad \text{with } \nabla_{\mathbf{y}}\mathbf{A}(\mathbf{y}(\mathbf{u}))^T\mathbf{p} = -\nabla_{\mathbf{y}} C(\mathbf{y}(\mathbf{u}),\mathbf{u})$$

$$\left\|\frac{\widehat{C}(\mathbf{u}+\varepsilon\delta\mathbf{u}) - \widehat{C}(\mathbf{u})}{\varepsilon} - \nabla_{\mathbf{u}}\widehat{C}(\mathbf{u})\delta\mathbf{u}\right\| = O(\varepsilon)$$

Reason: Quadrature of analytic derivative $D_\psi \mathsf{J}_{\mathrm{p}}(\psi_h^k,\xi_h))$ is not a derivative!

A discrete non-linear current:

$$\mathsf{J}_h(\psi_h,\xi_h) = \sum_T |T\cap\Omega_{\mathrm{p}}(\psi_h)|\, j_{\mathrm{p}}(\mathbf{b}_T,\psi_h(\mathbf{b}_T))\,\xi_h(\mathbf{b}_T)$$

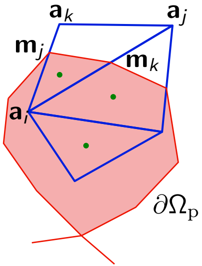
and $\mathbf{b}_T = \mathbf{b}_T(\psi_h)$

Newton's Method, Linearization II

A discrete non-linear current:

$\mathsf{J}_h(\psi_h, \xi_h) = \sum_T |T \cap \Omega_\mathrm{p}(\psi_h)|\, j_\mathrm{p}(\mathbf{b}_T, \psi_h(\mathbf{b}_T))\, \xi_h(\mathbf{b}_T)$ and $\mathbf{b}_T = \mathbf{b}_T(\psi_h)$

The derivative (the true discrete derivative!):



$$
\begin{aligned}
D_\psi \mathsf{J}_h(\psi_h, \xi_h)(\lambda_i) &= \frac{d}{d\psi_i} \mathsf{J}_h(\psi_h, \xi_h) \\
&= \sum_T \frac{d}{d\psi_i} |T \cap \Omega_\mathrm{p}(\psi_h)|\, j_\mathrm{p}(\mathbf{b}_T, \psi_\mathrm{N}(\mathbf{b}_T))\, \xi_h(\mathbf{b}_T) \\
&\quad + \sum_T |T \cap \Omega_\mathrm{p}(\psi_h)| \frac{d}{d\psi_i} j_\mathrm{p} \mathbf{b}_T, \psi_\mathrm{N}(\mathbf{b}_T))\, \xi_h(\mathbf{b}_T) \\
&\quad + \sum_T |T \cap \Omega_\mathrm{p}(\psi_h)|\, j_\mathrm{p}(\mathbf{b}_T, \psi_\mathrm{N}(\mathbf{b}_T)) \frac{d}{d\psi_i} \xi_h(\mathbf{b}_T)
\end{aligned}
$$

General implementation philosophy (everything local)

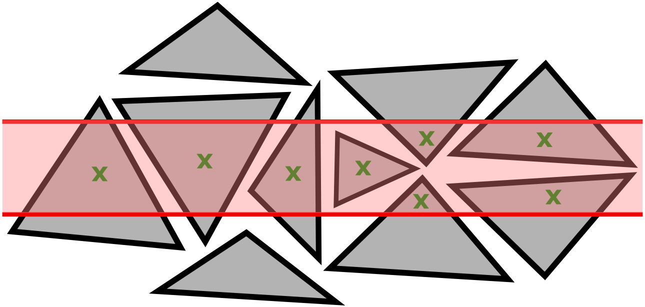
- Compute barycenter & intersection and their derivatives at same time!
- *Assemble vector $\mathsf{J}_h(\psi_h, \lambda_j)$ and matrix $D_\psi \mathsf{J}_h(\psi_h, \lambda_j)(\lambda_i)$ at same time.*

Newton's Method; Levelset and Mesh

Core function

Find intersection and quadrature points (and derivatives) of all elements that have non-zero intersection with levelline between ψ_l and ψ_u.

- $\psi_u = \psi_{ax}$ and $\psi_l = \psi_{bnd}$ for plasma domain;



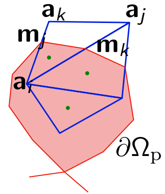
Centroid formula to generate quadrature formulas

$$Area_{tot}\mathbf{Bary_{tot}} = \sum_i Area_i\mathbf{Bary_i}$$

Newton's Method, Linearization III

Case A: $T \cap \Omega_{\mathrm{p}}(\psi_h) = T$:
barycenter $(r_T, z_T) = \frac{1}{3}(\mathbf{a}_i + \mathbf{a}_j + \mathbf{a}_k)$.

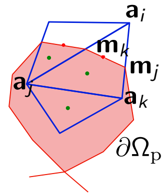
Case B: $T \cap \Omega_{\mathrm{p}}(\psi_h) =$ triangle



$$(r_T(\psi_h), z_T(\psi_h)) = \frac{1}{3}(\mathbf{a}_i + \mathbf{m}_k(\psi_h) + \mathbf{m}_j(\psi_h))$$

$$\mathbf{a}_i + \frac{1}{3}\lambda_j(\mathbf{m}_k(\psi_h))(\mathbf{a}_j - \mathbf{a}_i) + \frac{1}{3}\lambda_k(\mathbf{m}_j(\psi_h))(\mathbf{a}_k - \mathbf{a}_i)$$

Case C: $T \cap \Omega_{\mathrm{p}}(\psi_h) =$ quadrilateral



$$(r_T(\psi_h), z_T(\psi_h)) = \mathbf{a}_i + \frac{1}{3}\frac{1 - \lambda_j^2(\mathbf{m}_k(\psi_h))\lambda_k(\mathbf{m}_j(\psi_h))}{1 - \lambda_j(\mathbf{m}_k(\psi_h))\lambda_k(\mathbf{m}_j(\psi_h))}(\mathbf{a}_j - \mathbf{a}_i)$$

$$+ \frac{1}{3}\frac{1 - \lambda_j(\mathbf{m}_k(\psi_h))\lambda_k^2(\mathbf{m}_j(\psi_h))}{1 - \lambda_j(\mathbf{m}_k(\psi_h))\lambda_k(\mathbf{m}_j(\psi_h))}(\mathbf{a}_k - \mathbf{a}_i)$$

$$\lambda_j(\mathbf{m}_k(\psi_h)) = \frac{\psi_{\mathrm{bd}}(\psi_h) - \psi_i}{\psi_j - \psi_i}, \qquad \lambda_k(\mathbf{m}_j(\psi_h)) = \frac{\psi_{\mathrm{bd}}(\psi_h) - \psi_i}{\psi_k - \psi_i},$$

Newton's Method, "Semi-Automatic" Differentiation

```
function [J_NL,DJ] = assemblePlasma_NL(Mesh,psi,jpHandle)
% J_NL: vector, non-linear operator at psi,
% DJ:   matrix, derivative of J_NL at psi
...
% levelset is strucure levelset.ratio, level.bary
% containing ratio of intersection domain, barycenter
% and derivatives for each element
levelset = find_Plasma(Mesh,psi);
....
....
% non-linear operator
J_NL = [0.5*det_BK.*ratio(:,1).*jplasma_bary(:,1).*N1(:,1);...
        0.5*det_BK.*ratio(:,1).*jplasma_bary(:,1).*N2(:,1);...
        0.5*det_BK.*ratio(:,1).*jplasma_bary(:,1).*N3(:,1)]
....
....
% derivative of non-linear operator
DJ.E = [0.5*det_BK.*ratio(:,2).*jplasma_bary(:,1).*N1(:,1)+...
        0.5*det_BK.*ratio(:,1).*jplasma_bary(:,2).*N1(:,1)+...
        0.5*det_BK.*ratio(:,1).*jplasma_bary(:,1).*N1(:,2);
        0.5*det_BK.*ratio(:,2).*jplasma_bary(:,1).*N2(:,1)+...
        0.5*det_BK.*ratio(:,1).*jplasma_bary(:,2).*N2(:,1)+
```

What's next?

Inverse Static Problem (for currents I_i)

Objective and Regularization

$$K(\psi) := \frac{1}{2}\sum_{i=1}^{N_{\mathrm{desi}}}\big(\psi(r_i,z_i)-\psi(r_{\mathrm{desi}},z_{\mathrm{desi}})\big)^2 \quad , \quad R(I_1,\ldots,I_L) \quad := \sum_{i=1}^{L}\frac{w_i}{2}I_i^2$$

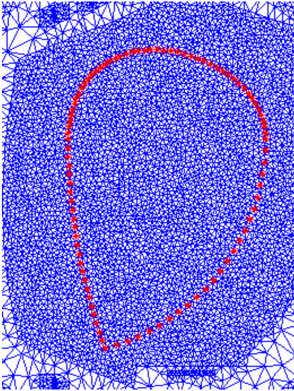
Optimal Control/Inverse Problem:

$$\min_{\psi, I_1,\ldots I_L} K(\psi) + R(I_1,\ldots I_L)$$

subject to

$$-\nabla\cdot\left(\frac{1}{\mu r}\nabla\psi\right) = \begin{cases} r S_p'(\psi_{\mathrm{N}}) + \frac{1}{\mu_0 r} S_{ff'}(\psi_{\mathrm{N}}) & \text{in } \Omega_{\mathrm{p}}(\psi)\,, \\ \frac{I_{i,j}}{S_{i,j}} & \text{in } \Omega_{\mathrm{coil}_i}\,, \\ 0 & \text{elsewhere}\,, \end{cases}$$

$$\psi(0,z) = 0\,, \qquad \lim_{\|(r,z)\|\to+\infty}\psi(r,z) = 0\,,$$



PDE-constrained optimization with non-linear constraints.

Optimal Control: Numerical Methods

1.) unconstrained optimization: $\min_{\mathbf{u}} C(\mathbf{u}) \iff \nabla_{\mathbf{u}} C(\mathbf{u}) = 0$

$$\underbrace{\mathbf{u}^{k+1} = \mathbf{u}^k - s_k \nabla C(\mathbf{u}^k)}_{\text{global, slow convergence}} \quad \underbrace{\mathbf{u}^{k+1} = \mathbf{u}^k - \mathbf{M}_k \nabla C(\mathbf{u}^k)}_{\text{global, not too slow convergence}} \quad \underbrace{\mathbf{u}^{k+1} = \mathbf{u}^k - \nabla^2 C(\mathbf{u}^k) \nabla C(\mathbf{u}^k)}_{\text{local, fast convergence}}$$

2.) constrained optimization: $\min_{\mathbf{y},\mathbf{u}} C(\mathbf{y},\mathbf{u})$ s.t. $\begin{array}{c} \mathbf{A}_1(\mathbf{y}) = \mathbf{F}_1(\mathbf{u}) \\ \dots \\ \mathbf{A}_n(\mathbf{y}) = \mathbf{F}_n(\mathbf{u}) \end{array} \iff$ (A) $\Leftrightarrow$ (B)

(A) reduced approach/steepest descent:

$\min_{\mathbf{u}} \widehat{C}(\mathbf{u}) := C(\mathbf{y}(\mathbf{u}),\mathbf{u}) \Leftrightarrow$

i) $\nabla_{\mathbf{u}} \widehat{C}(\mathbf{u}) = \nabla_{\mathbf{u}} C(\mathbf{y}(\mathbf{u}),\mathbf{u}) + \nabla_{\mathbf{y}} C(\mathbf{y}(\mathbf{u}),\mathbf{u}) \nabla_{\mathbf{u}} \mathbf{y}(\mathbf{u}) = 0$

with $\nabla_{\mathbf{y}} \mathbf{A}(\mathbf{y}(\mathbf{u})) \nabla_{\mathbf{u}} \mathbf{y}(\mathbf{u}) = \nabla_{\mathbf{u}} \mathbf{F}(\mathbf{u})$

ii) $\nabla_{\mathbf{u}} \widehat{C}(\mathbf{u}) = \nabla_{\mathbf{u}} C(\mathbf{y}(\mathbf{u}),\mathbf{u}) + \nabla_{\mathbf{u}} \mathbf{F}(\mathbf{u})^T \mathbf{p} = 0$

with $\nabla_{\mathbf{y}} \mathbf{A}(\mathbf{y}(\mathbf{u}))^T \mathbf{p} = -\nabla_{\mathbf{y}} C(\mathbf{y}(\mathbf{u}),\mathbf{u})$

To use methods from 1) we need to compute $\mathbf{y}(\mathbf{u})$ and $\nabla_{\mathbf{u}} \mathbf{y}(\mathbf{u})$! expensive !

(B) Lagrange multipliers p_i/ SQP:

stationary point of Lagrangian $L(\mathbf{y},\mathbf{u},\mathbf{p}) = C(\mathbf{y},\mathbf{u}) + \mathbf{p}^T(\mathbf{A}(\mathbf{y}) - \mathbf{F}(\mathbf{u}))$

Fastest algorithm! Newton for (B) $=$ Sequential quadratic programming (SQP)

Steepest Descent vs. SQP I: fminunc

Command Window

Iteration	f(x)	step	optimality	CG-itera
0	0.0748239		3.33e-06	
1	0.0747357	10	3.33e-06	2
2	0.0745595	20	3.33e-06	2
3	0.0742077	40	3.32e-06	2
4	0.073506	80	3.32e-06	2
5	0.0721101	160	3.3e-06	2
6	0.0693473	320	3.27e-06	2
7	0.0644181	640	2.55e-06	2
8	0.0562001	1280	2.44e-06	1
16	0.00655371	40960	6.72e-07	4
17	0.00603318	669.542	3.49e-08	1
18	0.00603318	81920	3.49e-08	7
19	0.00603318	20480	3.49e-08	0
20	0.00603318	5120	3.49e-08	0
21	0.00600083	1280	6.15e-08	0
22	0.00592413	2560	3.2e-08	5
23	0.00581343	5120	5.59e-08	7
24	0.00581343	10240	5.59e-08	5
25	0.00581343	2560	5.59e-08	0
26	0.00581343	640	5.59e-08	0
27	0.00581343	160	5.59e-08	0
28	0.00581343	40	5.59e-08	0
29	0.00581224	10	5.16e-08	0
30	0.0058102	20	4.32e-08	5

Solver stopped prematurely.

fminunc stopped because it exceeded the function evaluation limit,
options.MaxFunEvals = 30 (the selected value).

$\boxed{\text{Elapsed time is 206.081131 seconds.}}$

Steepest Descent vs. SQP II: home made sqp

Command Window

```
Newtoniteration 0; relativ residuum 3.84e-05
                   cost 7.41e-02; regularization 7.25e-04; objective 7.48e-02
                   resid_stat = 2.00e+00, resid_ctrl = 2.31e+03, resid_adj = 1.13e
Newtoniteration 1; relativ residuum 5.86e-01
                   cost 1.70e-03; regularization 1.50e-03; objective 3.20e-03
                   resid_stat = 5.41e-01, resid_ctrl = 5.47e-13, resid_adj = 5.86e
Newtoniteration 2; relativ residuum 2.84e-01
                   cost 1.90e-03; regularization 1.34e-03; objective 3.24e-03
                   resid_stat = 5.90e-01, resid_ctrl = 2.21e-16, resid_adj = 2.84e
Newtoniteration 3; relativ residuum 4.34e-02
                   cost 1.86e-03; regularization 1.38e-03; objective 3.24e-03
                   resid_stat = 1.48e-03, resid_ctrl = 2.20e-16, resid_adj = 4.34e
Newtoniteration 4; relativ residuum 4.50e-03
                   cost 1.86e-03; regularization 1.38e-03; objective 3.24e-03
                   resid_stat = 2.92e-04, resid_ctrl = 5.05e-16, resid_adj = 4.50e
Newtoniteration 5; relativ residuum 1.70e-04
                   cost 1.86e-03; regularization 1.38e-03; objective 3.24e-03
                   resid_stat = 1.23e-04, resid_ctrl = 5.30e-16, resid_adj = 1.70e
Newtoniteration 6; relativ residuum 1.54e-05
                   cost 1.86e-03; regularization 1.38e-03; objective 3.24e-03
                   resid_stat = 5.67e-05, resid_ctrl = 2.65e-16, resid_adj = 1.54e
Newtoniteration 7; relativ residuum 4.06e-10
                   cost 1.86e-03; regularization 1.38e-03; objective 3.24e-03
                   resid_stat = 7.50e-07, resid_ctrl = 4.21e-16, resid_adj = 4.06e
Newtoniteration 8; relativ residuum 1.12e-12
                   cost 1.86e-03; regularization 1.38e-03; objective 3.24e-03
                   resid_stat = 4.01e-09, resid_ctrl = 3.16e-16, resid_adj = 1.12e
Elapsed time is 11.649052 seconds.
```

Inverse Evolution Problem (for volt. evolution $\vec{V}(t)$)

Objective and Regularization:

$$K(\psi(t)) := \frac{1}{2}\int_0^T \left(\sum_{i=1}^{N_{\mathrm{desi}}} \left(\psi(r_i(t), z_i(t), t) - \psi(r_{\mathrm{desi}}(t), z_{\mathrm{desi}}(t), t)\right)^2\right) dt\,,$$

$$R(\vec{V}) := \sum_{i=1}^{L} \frac{w_i}{2}\int_0^T \vec{V}_i(t)\cdot\vec{V}_i(t)dt\,.$$

Optimal Control/Inverse Problem:

$$\min_{\psi(t),\vec{V}(t)} K(\psi(t)) + R(\vec{V})$$

subject to

$$-\nabla\cdot\left(\frac{1}{\mu r}\nabla\psi\right) = \begin{cases} rS_p'(\psi_{\mathrm{N}}, t) + \frac{1}{\mu_0 r} S_{ff'}(\psi_{\mathrm{N}}, t) & \text{in } \Omega_{\mathrm{p}}(\psi)\,, \\ S_i^{-1}\left(\boldsymbol{S}\vec{V} + \boldsymbol{R}\vec{\Psi}(\partial_t\psi)\right)_i & \text{in } \Omega_{\mathrm{coil}_i}\,, \\ -\frac{\sigma_k}{r}\partial_t\psi & \text{in } \Omega_{\mathrm{passive}}\,, \\ 0 & \text{elsewhere}\,, \end{cases}$$

$$\psi(0, z, t) = 0\,, \qquad \lim_{\|(r,z)\|\to+\infty} \psi(r, z, t) = 0\,, \quad \psi(r, z, 0) = \psi_0(r, z)\,,$$

PDE-constrained optimization with non-linear constraints.

Inverse Evolution Problem

Non-linear, finite-dimensional, constrained optimization problem:

$$\min_{\bar{\mathbf{u}},\bar{\mathbf{y}}} \sum_{i=1}^{N} \frac{\tau}{2}\langle \mathbf{u}_i, \mathbf{R}_i \mathbf{u}_i\rangle + \frac{\tau}{2}\langle \mathbf{y}_i, \mathbf{K}_i \mathbf{y}_i\rangle$$

$$\text{s.t.} \quad \bar{\mathbf{A}}(\bar{\mathbf{y}}) - \bar{\mathbf{A}}_0(\mathbf{y}_0) := \begin{pmatrix} \tau\mathbf{A}(\mathbf{y}_1) + \mathbf{m}(\mathbf{y}_1) \\ \tau\mathbf{A}(\mathbf{y}_2) + \mathbf{m}(\mathbf{y}_2 - \mathbf{y}_1) \\ \vdots \\ \tau\mathbf{A}(\mathbf{y}_N) + \mathbf{m}(\mathbf{y}_N - \mathbf{y}_{N-1}) \end{pmatrix} - \begin{pmatrix} \mathbf{m}(\mathbf{y}_0) \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} \tau\mathbf{F}(\mathbf{u}_1) \\ \tau\mathbf{F}(\mathbf{u}_2) \\ \vdots \\ \tau\mathbf{F}(\mathbf{u}_N) \end{pmatrix} =: \bar{\mathbf{F}}(\bar{\mathbf{u}})$$

$\bar{\mathbf{u}} = (\mathbf{u}_1, \ldots \mathbf{u}_N)$, voltages in coils at $t_1, \ldots t_N$,
$\bar{\mathbf{y}} = (\mathbf{y}_1, \ldots \mathbf{y}_N)$, flux ψ at $t_1, \ldots t_N$
Quasi-Newton for $\bar{\mathbf{y}}^{k+1} = \bar{\mathbf{y}}^k + \Delta\bar{\mathbf{y}}, \bar{\mathbf{u}}^{k+1} = \bar{\mathbf{u}}^k + \Delta\bar{\mathbf{u}}, \bar{\mathbf{p}}^{k+1} = \bar{\mathbf{p}}^k + \Delta\bar{\mathbf{p}}$:

$$\begin{pmatrix} \bar{\mathbf{K}} & 0 & D_{\bar{\mathbf{y}}}\bar{\mathbf{A}}^T(\bar{\mathbf{y}}^k) \\ 0 & \bar{\mathbf{R}} & -D_{\bar{\mathbf{u}}}\bar{\mathbf{F}}^T(\bar{\mathbf{u}}^k) \\ D_{\bar{\mathbf{y}}}\bar{\mathbf{A}}(\bar{\mathbf{y}}^k) & -D_{\bar{\mathbf{u}}}\bar{\mathbf{F}}(\bar{\mathbf{u}}^k) & 0 \end{pmatrix} \begin{pmatrix} \Delta\bar{\mathbf{y}} \\ \Delta\bar{\mathbf{u}} \\ \Delta\bar{\mathbf{p}} \end{pmatrix} = -\begin{pmatrix} \bar{\mathbf{K}}\bar{\mathbf{y}}^k + D_{\bar{\mathbf{y}}}\bar{\mathbf{A}}^T(\bar{\mathbf{y}}^k)\bar{\mathbf{p}}^k \\ \bar{\mathbf{R}}\bar{\mathbf{u}}^k - D_{\bar{\mathbf{u}}}\bar{\mathbf{F}}^T(\bar{\mathbf{u}}^k)\bar{\mathbf{p}}^k \\ \bar{\mathbf{A}}(\bar{\mathbf{y}}^k) - \bar{\mathbf{A}}_0(\mathbf{y}_0) - \bar{\mathbf{F}}(\bar{\mathbf{u}}^k) \end{pmatrix},$$

The "all-at-once" approach: Solve large ($N_{\mathrm{timesteps}}(2N_{\mathrm{FEM}} + N_{\mathrm{coils}})$) linear system
a) with a direct solver b) with an iterative solver ,

Inverse Evolution Problem, Details

$$D_{\bar{\mathbf{y}}}\bar{\mathbf{A}}(\bar{\mathbf{y}}) = \begin{pmatrix} \tau D_{\mathbf{y}}\mathbf{A}(\mathbf{y}_1) + D_{\mathbf{y}}\mathbf{m}(\mathbf{y}_1) & 0 & 0 \\ -D_{\mathbf{y}}\mathbf{m}(\mathbf{y}_1) & \tau D_{\mathbf{y}}\mathbf{A}(\mathbf{y}_2) + D_{\mathbf{y}}\mathbf{m}(\mathbf{y}_2) & 0 \\ \ddots & \ddots & \ddots \\ 0 & -D_{\mathbf{y}}\mathbf{m}(\mathbf{y}_{N-1}) & \tau D_{\mathbf{y}}\mathbf{A}(\mathbf{y}_N) + D_{\mathbf{y}}\mathbf{m}(\mathbf{y}_N) \end{pmatrix},$$

$$D_{\bar{\mathbf{u}}}\bar{\mathbf{F}}(\bar{\mathbf{u}}) = \begin{pmatrix} \tau D_{\mathbf{u}}\widetilde{\mathbf{F}}(\mathbf{u}_1) & 0 & 0 \\ 0 & \tau D_{\mathbf{u}}\widetilde{\mathbf{F}}(\mathbf{u}_2) & 0 \\ \ddots & \ddots & \ddots \\ 0 & 0 & \tau D_{\mathbf{u}}\widetilde{\mathbf{F}}(\mathbf{u}_N) \end{pmatrix}, \bar{\mathbf{K}} = \begin{pmatrix} \tau\mathbf{K}_1 & 0 & 0 \\ 0 & \tau\mathbf{K}_2 & 0 \\ \ddots & \ddots & \ddots \\ 0 & 0 & \tau\mathbf{K}_N \end{pmatrix},$$

$$\bar{\mathbf{R}} = \begin{pmatrix} \tau\mathbf{R}_1 & 0 & 0 & 0 \\ 0 & \tau\mathbf{R}_2 & 0 & 0 \\ \ddots & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & \tau\mathbf{R}_N \end{pmatrix},$$

Inverse Evolution Problem

Sequential Quadratic Programming "Solve a sequence of quadratic problems"

1.) Solve full Newton system:

Quasi-Newton for $\bar{\mathbf{y}}^{k+1} = \bar{\mathbf{y}}^k + \Delta\bar{\mathbf{y}}, \bar{\mathbf{u}}^{k+1} = \bar{\mathbf{u}}^k + \Delta\bar{\mathbf{u}}, \bar{\mathbf{p}}^{k+1} = \bar{\mathbf{p}}^k + \Delta\bar{\mathbf{p}}$:

$$\begin{pmatrix} \bar{\mathbf{K}} & 0 & D_{\bar{\mathbf{y}}}\bar{\mathbf{A}}^T(\bar{\mathbf{y}}^k) \\ 0 & \bar{\mathbf{R}} & -D_{\bar{\mathbf{u}}}\bar{\mathbf{F}}^T(\bar{\mathbf{u}}^k) \\ D_{\bar{\mathbf{y}}}\bar{\mathbf{A}}(\bar{\mathbf{y}}^k) & -D_{\bar{\mathbf{u}}}\bar{\mathbf{F}}(\bar{\mathbf{u}}^k) & 0 \end{pmatrix} \begin{pmatrix} \Delta\bar{\mathbf{y}} \\ \Delta\bar{\mathbf{u}} \\ \Delta\bar{\mathbf{p}} \end{pmatrix} = - \begin{pmatrix} \bar{\mathbf{K}}\bar{\mathbf{y}}^k + D_{\bar{\mathbf{y}}}\bar{\mathbf{A}}^T(\bar{\mathbf{y}}^k)\bar{\mathbf{p}}^k \\ \bar{\mathbf{R}}\bar{\mathbf{u}}^k - D_{\bar{\mathbf{u}}}\bar{\mathbf{F}}^T(\bar{\mathbf{u}}^k)\bar{\mathbf{p}}^k \\ \bar{\mathbf{A}}(\bar{\mathbf{y}}^k) - \bar{\mathbf{A}}_0(\mathbf{y}_0) - \bar{\mathbf{F}}(\bar{\mathbf{u}}^k) \end{pmatrix},$$

System is roughly twice as large as for direct problem.

2.) Solve reduced Newton system with CG or directly:

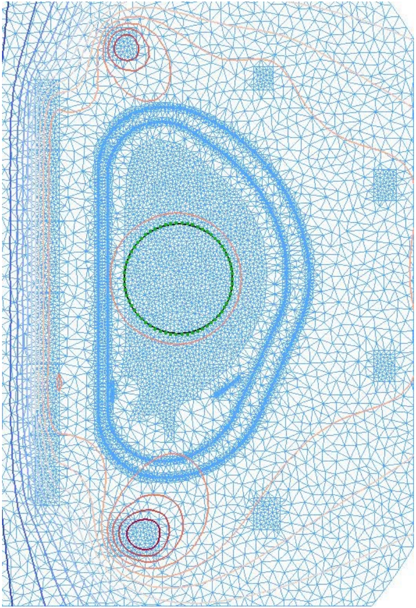
Eliminate $\Delta\bar{\mathbf{y}}, \Delta\bar{\mathbf{p}}$: $\bar{\mathbf{M}}(\bar{\mathbf{u}}^k, \bar{\mathbf{y}}^k)\Delta\bar{\mathbf{u}} = \bar{\mathbf{h}}(\bar{\mathbf{u}}^k, \bar{\mathbf{y}}^k)$,

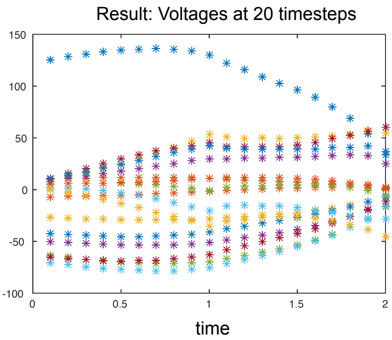
where $\bar{\mathbf{M}}(\bar{\mathbf{u}}, \bar{\mathbf{y}}) = \bar{\mathbf{R}} + D_{\bar{\mathbf{u}}}\bar{\mathbf{F}}^T(\bar{\mathbf{u}}) D_{\bar{\mathbf{y}}}\bar{\mathbf{A}}(\bar{\mathbf{y}})^{-T}\bar{\mathbf{K}} D_{\bar{\mathbf{y}}}\bar{\mathbf{A}}(\bar{\mathbf{y}})^{-1} D_{\bar{\mathbf{u}}}\bar{\mathbf{F}}(\bar{\mathbf{u}}^k)$

and $\bar{\mathbf{h}}(\bar{\mathbf{u}}, \bar{\mathbf{y}}) = -\bar{\mathbf{R}}\bar{\mathbf{u}} - D_{\bar{\mathbf{u}}}\bar{\mathbf{F}}^T(\bar{\mathbf{u}}) D_{\bar{\mathbf{y}}}\bar{\mathbf{A}}(\bar{\mathbf{y}})^{-T}\bar{\mathbf{K}} D_{\bar{\mathbf{y}}}\bar{\mathbf{A}}(\bar{\mathbf{y}})^{-1}(\bar{\mathbf{F}}(\bar{\mathbf{u}}) - \bar{\mathbf{A}}(\bar{\mathbf{y}}) + D_{\bar{\mathbf{y}}}\bar{\mathbf{A}}(\bar{\mathbf{y}})\bar{\mathbf{y}})$.

- CG-solver: Very few CG-iterations, but one inversion of $D_{\bar{\mathbf{y}}}\bar{\mathbf{A}}(\bar{\mathbf{y}})$ and $D_{\bar{\mathbf{y}}}\bar{\mathbf{A}}(\bar{\mathbf{y}})^T$ in each iteration.
- direct sover: $\bar{\mathbf{M}}(\bar{\mathbf{u}}^k, \bar{\mathbf{y}}^k)$ is relatively small but not sparse.

Control of Transient Plasma Equilibrium



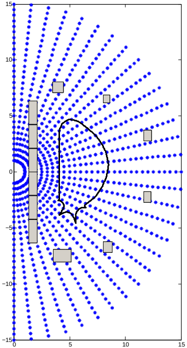


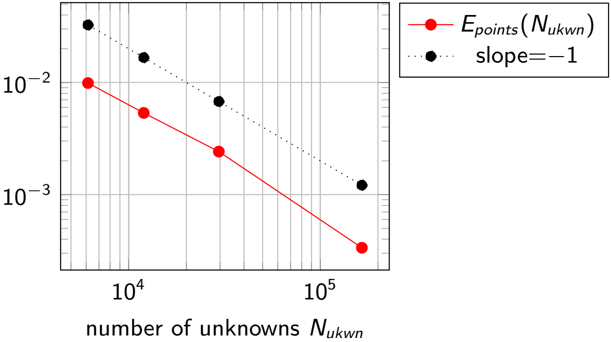
Major critics: electrodynamic effects of the plasma are not included, yet!
Remaining equations:

- conservation of density and energy;
- $-\partial_t \mathbf{B}_T = \mathbf{curl}\, \mathbf{E}_P$ in plasma;
- $\mathbf{E} + \mathbf{v} \times \mathbf{B} = \eta \mathbf{J}$ in plasma;

What's next?

Validation & Performance



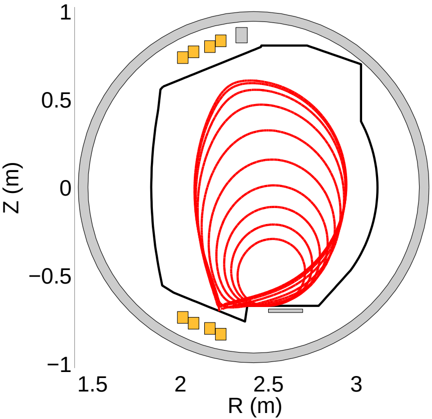


number of triangles	number of unknowns	computing time (in s)
12099	6134	2
23723	11985	5
58744	29556	11
328693	164887	88
1153174	577415	368

iteration	relative residual
1	$2.667473 \times 10^{+00}$
2	9.157459×10^{-02}
3	1.781645×10^{-03}
4	0.525234×10^{-06}
5	3.935226×10^{-12}

What's next?

Application: Vertical Displacement



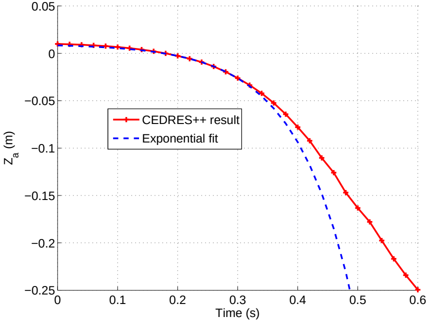


Figure: Left: Plasma boundary at intervals of $100ms$ in a vertical instability simulation for WEST. Right: Time evolution of the vertical position of the magnetic axis z_{ax} in a vertical instability simulation for WEST.

What's next?

Conclusions & Outlook

Conclusions:

- mature and sound equilibrium calculation;
- ready to use for applications and automation;
- Coupling of CEDRES and ETS (European Transport Solver) in ITM; (C. Boulbe & B. Faugeras with J.F. Artaud, P. Huyn, V. Basiuk, E. Nardon, J. Urban, D. Kalupin at CEA, Munich, Prag)
- FEEQS.M with Edge Plasma Code for divertor load optimization; (H.H. with M. Bloomart, T. Baelmans, N. Gauger, D. Reiter at Jülich, Leuven, Kaiserslautern),

Outlook:

1. towards monolitic solver for equilibrium and transport;
2. optimal control for scenario optimization for tokamaks;
3. control engineers are interested in realtime solutions of the coupled problem;
4. can not be achieved by only increasing computational power;