Analysis of the BIEST vacuum field solver, implementing the Merkel approach

Florian Hindenlang[1]

[1]Numerical Methods division (NMPP), IPP Garching

Problem formulation (sketch)

- We want to solve for a vacuum magetic field in the outer region of the plasma domain

$$\nabla \cdot B = 0\,, \nabla \times B = 0$$

- The outer domain has only one boundary, the plasma torus as a 'hole' (so one harmonic function is needed),
- We are only interested in the field directly on the domain boundary (at the plasma-vacuum interface), which is supposed to be a flux surface, thus $B \cdot n = 0$
- The vacuum field has a contribution from coil currents and from the plasma current:

$$B = B_{\text{coils}} + B_{\text{plasma}}$$

- Merkel method [JCP,1986]: decompose the field into two parts: a periodic solution $\nabla\Phi$ on the flux surface and put 'the rest' into B_0 and solve $B \cdot n = 0$.

$$B \cdot n = B_0 \cdot n + \nabla\Phi \cdot n = 0$$

- Here B_0 contains B_{coils} and one must additionally account for the non-periodic part of Φ, which is related to the loop integral of B enclosing the torus, thus the toroidal current of the plasma

2D testcase in R, Z plane (axi-symmetry)

Some arguments for a axi-symmetric 2D testcase

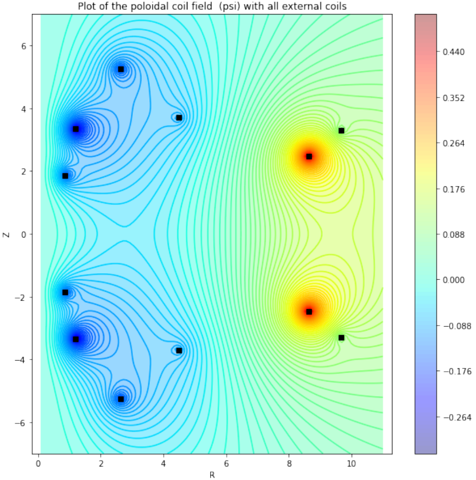
1. closed flux surface exist(!)
2. its probably easier to understand/visualize/analyze. we only need to care about the poloidal magnetic field B_R, B_Z, the toroidal magnetic field $B_\phi = F/R$, where F is a constant everywhere outside the current sources.
3. Known solution of the vacuum field generated by a toroidal current point source of the poloidal flux $\Psi(R, Z)$

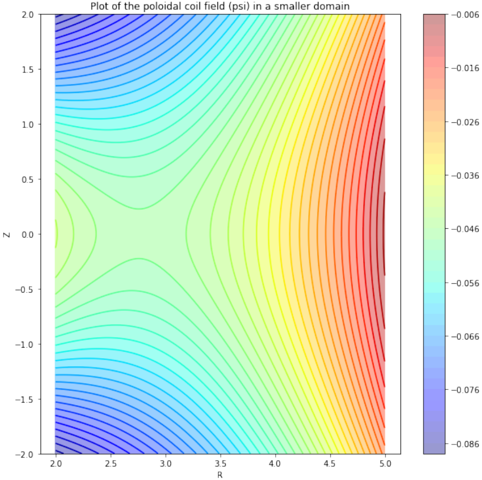
$$\Delta^*\Psi = 0\,, \quad \rightarrow (B_R, B_Z) = \frac{1}{R}(-\frac{\partial\Psi}{\partial Z}, \frac{\partial\Psi}{\partial R})$$

4. We neither have nor know how to use the tools that would be needed to setup a testcase for the 3D case (!) dommaschk potentials could be a possiblity, but did not find any code. probably one would also need a biot-savart solver for evaluating coil fields.
5. Most importantly, the existence of flux surfaces in 3D is not guaranteed(!)/ only approximately satisfied

$\Rightarrow$ 2D Tests/Analysis is implemented in python on a jupyter notebook (on a private github repo), using a python wrapper to BIEST vacuum field code

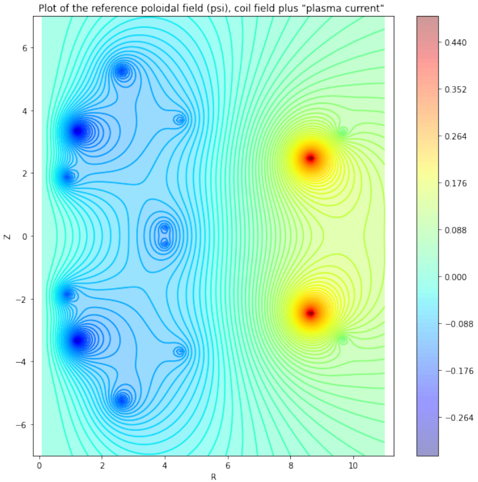
2D testcase: Construct coils+plasma field

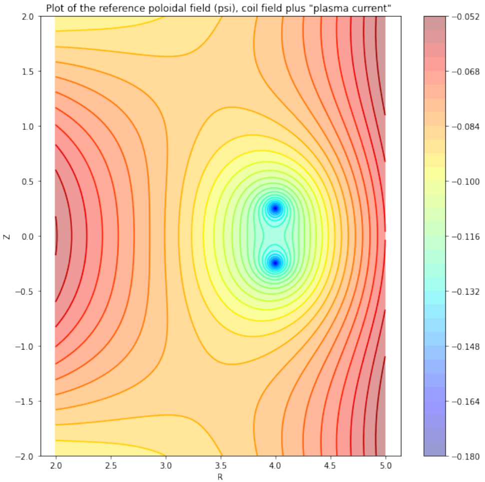




- Symmetric coil field evaluated from a set of point sources

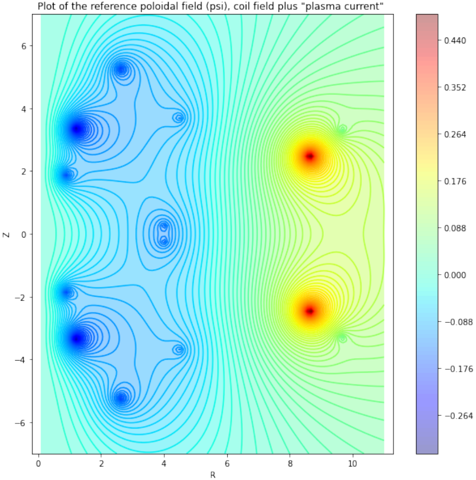
2D testcase: Construct coils+plasma field

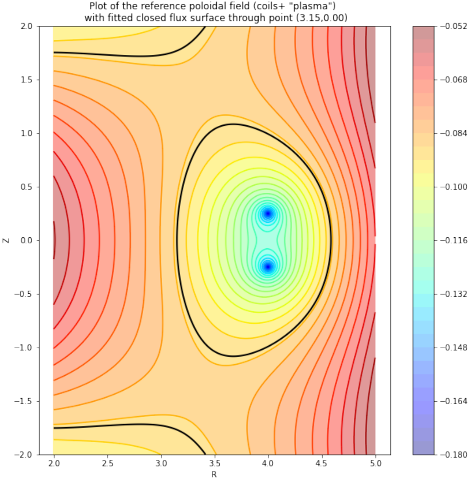




- Add two 'plasma' current sources to generate a 'plasma' region: full field $= B_{\text{ref}}$

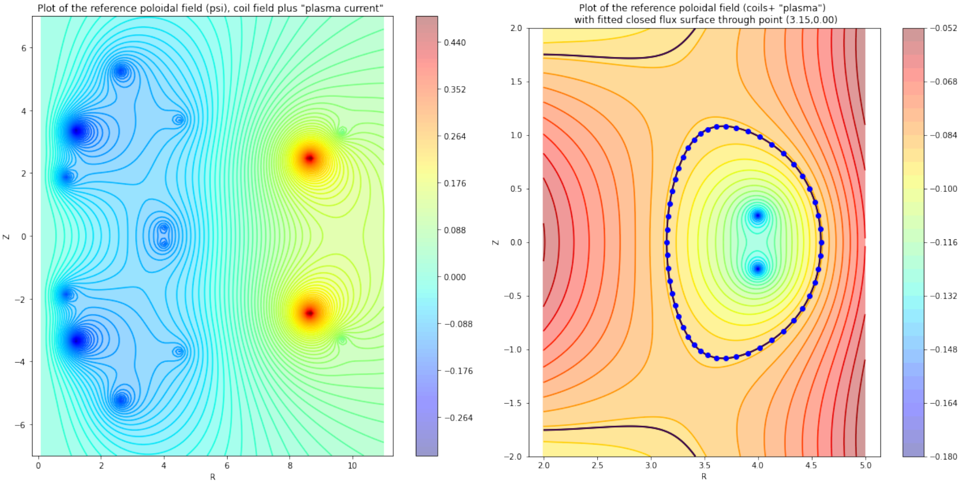
2D testcase: Construct coils+plasma field





- Add two 'plasma' current sources to generate a 'plasma' region: full field $= B_{\text{ref}}$
- Choose a closed flux surface $\Leftrightarrow$ 'plasma'-vacuum boundary

2D testcase: Construct coils+plasma field



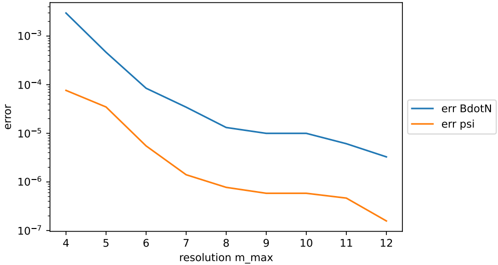
- Add two 'plasma' current sources to generate a 'plasma' region: full field $= B_{\text{ref}}$
- Choose a closed flux surface $\Leftrightarrow$ 'plasma'-vacuum boundary $\Rightarrow$ fit parametrized curve!

Fit parametrized curve to a contour

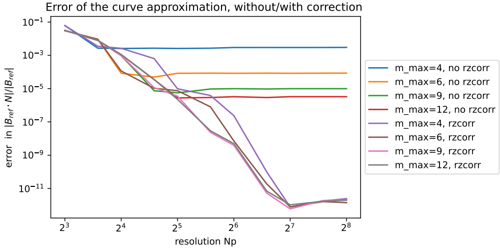
- Find periodic curve $R_c(t), Z_c(t)$ with a 'meaningful' parametrization (for example 'equal arc length')
- Fourier representation of R_c, Z_c (up-down symmetry)
- Least squares minimization of the 'squared distance' in the magnetic flux, sampled on a set of point along the curve:
$$\min\left(\frac{1}{2}\sum_i \frac{|\Psi(R_c(t_i),Z_c(t_i))-\Psi_{\text{contour}}|^2}{|\Psi_{\text{contour}}|^2}+\frac{|B\cdot n_i|^2}{|B|^2}\right)$$
- Iteratively fit + increase mode number $m_{\max}=4\dots12$

$\Rightarrow$ Accuracy seems limited to single precision (due to square...?)

- But gives a good parametrization.
Since BEAST needs only points, add a 2nd step:
find the root to the contour in normal direction:
$$\Psi\left((R_i+\alpha N_R),(Z_i+\alpha N_Z)\right)-\Psi_{\text{contour}}=0$$

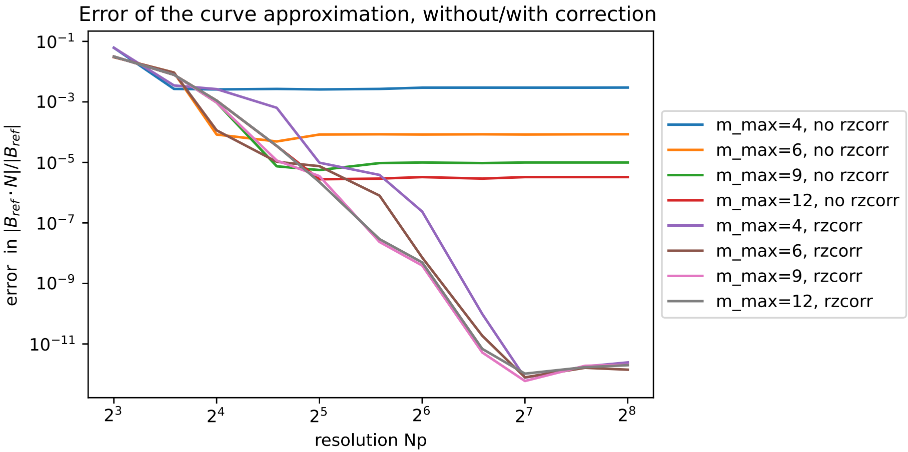


Error of the fitted curves with LS



Approximation error of the normal $|B_{ref}\cdot N|/|B_{ref}|$ w/o and with correction

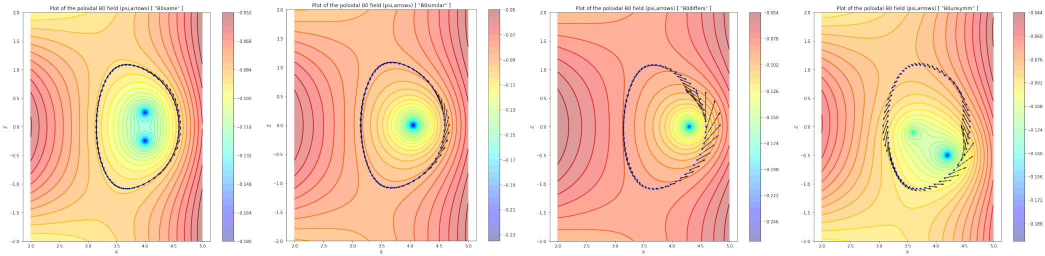
Fit parametrized curve to a contour



N_p is the number of points on the curve, N, B_{ref} is the reference first at these points and $B_{\mathrm{ref}} \cdot N$ is computed within BIEST. Machine precision reached at $N_p = 128$

$\Rightarrow$ Keep in mind: when comparing the field to B_{ref}, we already have an approximation error of the normal direction N!

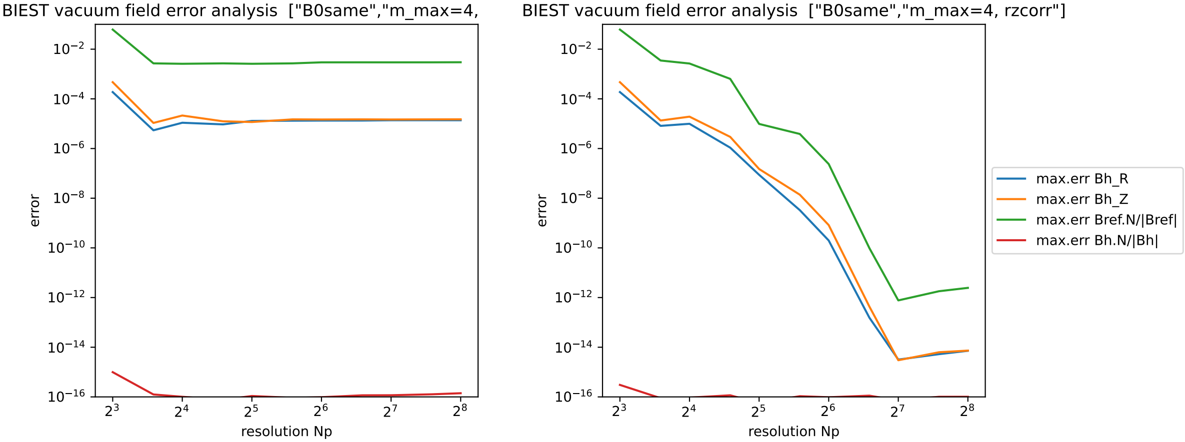
Testing the influence of the choice of B_0



We will look at 4 cases, all have the same 'plasma' current:

1. 'same': Set $B_0 = B_{\text{ref}}$ (so that $\nabla\Phi = 0$)
2. 'similar': Change 'plasma' current soure position slightly $B_0 \neq B_{\text{ref}}$
3. 'differs': Change positions more
4. 'unsymm.': Change positions more and make B_0 unsymmetric

Results: Case B_0 same as B_{ref}

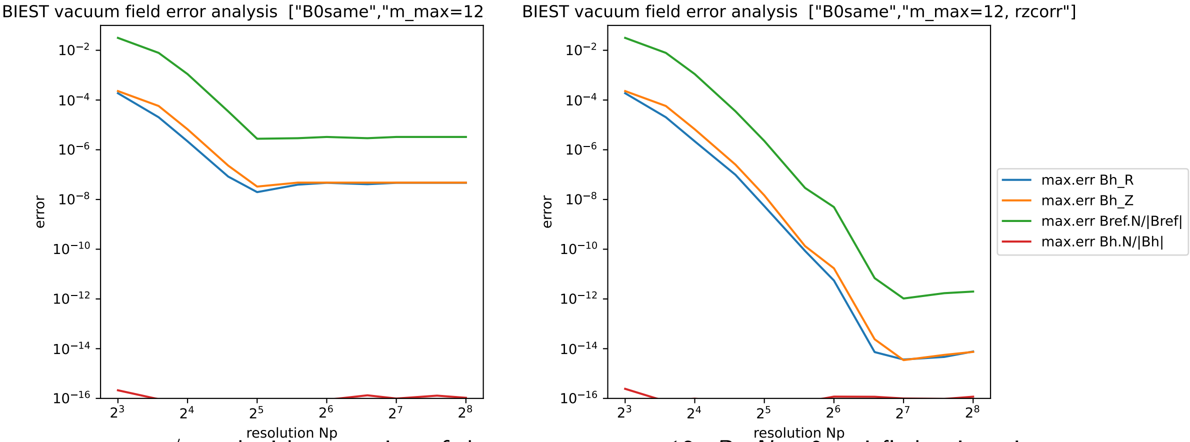


w/o and with correction of the curve, $m_{\max} = 4$

$\Rightarrow$ error of magnetic field lower than the curve approximation error

$\Rightarrow$ error without correction dominated by curve appoximation

Results: Case B_0 same as B_{ref}

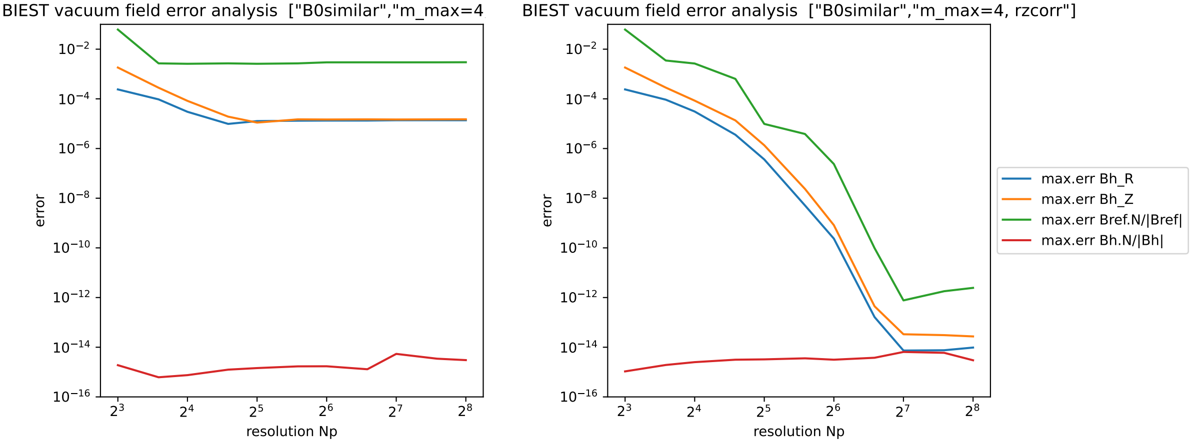


w/o and with correction of the curve, $m_{\max} = 12$. $B \cdot N = 0$ satisfied point-wise.

$\Rightarrow$ error of magnetic field lower than the curve approximation error

$\Rightarrow$ error without correction dominated by curve appoximation, but same up to $N_p = 32$

Results: Case B_0 similar

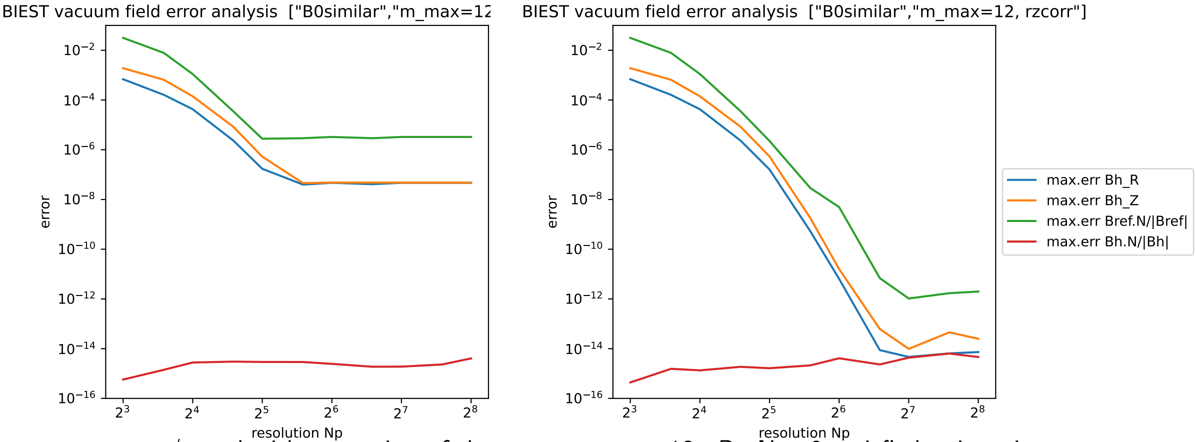


w/o and with correction of the curve, $m_{\max} = 4$

$\Rightarrow$ error of magnetic field smaller than the curve approximation error

$\Rightarrow$ error without correction dominated by curve appoximation

Results: Case B_0 similar

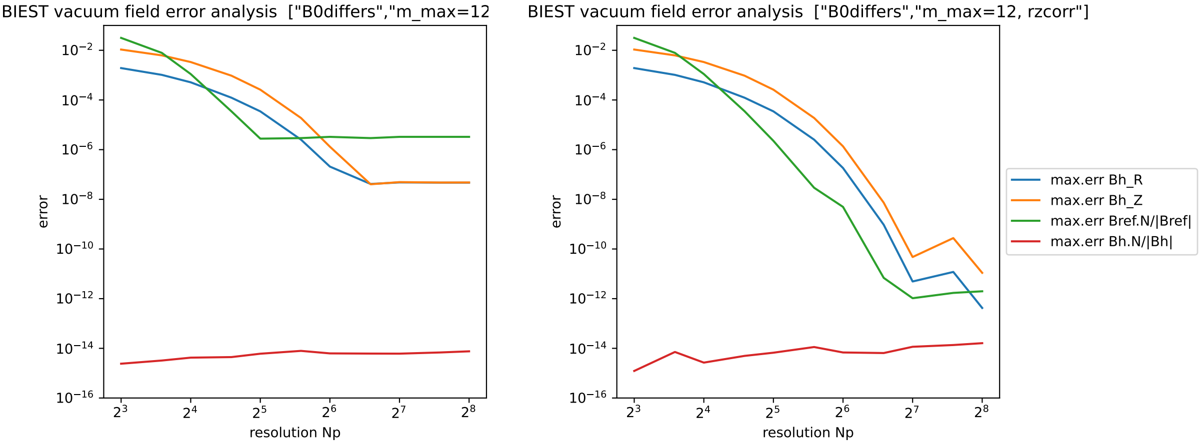


w/o and with correction of the curve, $m_{\max} = 12$. $B \cdot N = 0$ satisfied point-wise.

$\Rightarrow$ error of magnetic field smaller than the curve approximation error

$\Rightarrow$ error without correction dominated by curve appoximation, but same up to $N_p = 32$

Results: Case B_0 differs

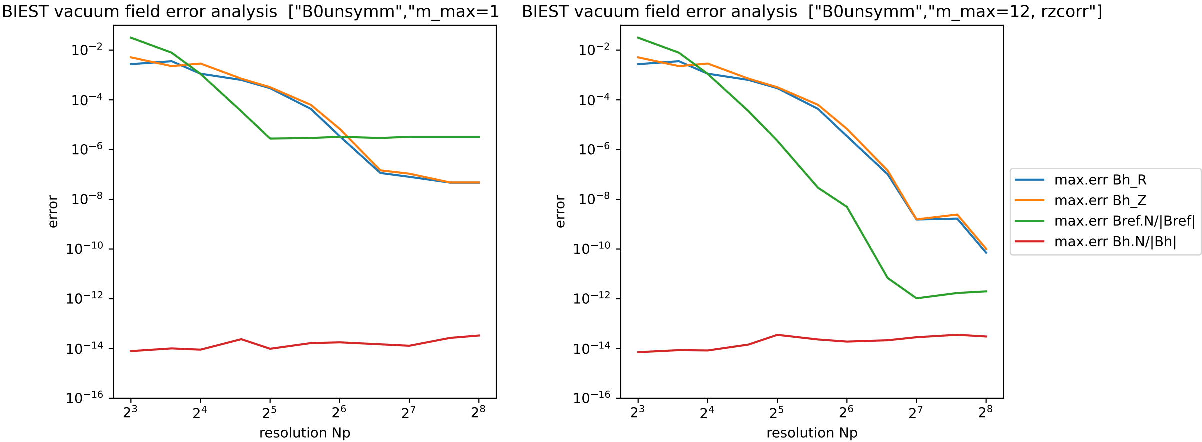


w/o and with correction of the curve, $m_{\max} = 12$. $B \cdot N = 0$ satisfied point-wise.

$\Rightarrow$ error of magnetic field now larger than the curve approximation error

$\Rightarrow$ error without correction dominated by curve appoximation, but same up to $N_p = 64$

Results: Case B_0 unsymmetric



w/o and with correction of the curve, $m_{\max} = 12$

$\Rightarrow$ error of magnetic field now larger than the curve approximation error

$\Rightarrow$ error without correction dominated by curve appoximation, but same up to $N_p = 48$

Conclusion on the results

- The choice of the 'plasma' current position for B_0 has a strong influence on the accuracy of the result
- Error in the fields behaves similar to the error of the geometry approximation
- convergence behavior is spectral but also affected for the worst case

BIEST compilation, python interface

- Using Makefile means that changes in compilers/libraries must be edited locally, without being able to save the configuration in the repo...
- Managed to compile on my machine (linux, ubuntu) with gcc, blas, lapack, fftw3 after installing all these libraries on my machine, and uncommenting the correct lines in the Makefile, it worked.
- Managed to compile on cluster (linux, module env. for libraries) with gcc, mkl, fftw3 added path to the loaded libraries to the Makefile (-L and -I flags) and executable needs module path in LD_LIBRARY_PATH, seems cluster specific)
  compilation with INTEL 21.4 failed first, fixed with loading new version of gcc as well!
- Forked Bharats virtual_casing[1] repo, as a blueprint, to my github as BIEST_to_python[2]
- Setup only for the vacuum_field application, by modifying the cpp file for the pybind11 tool. With cmake and pybind11 a shared object is built that can be called from python (!)

⇒ Can call setup, computeBdotN, computeGradPhi from python (3D-arrays ordered correctly & flattened), simply by placing the generated .so file into the folder of the python script.

[1] github.com/hiddenSymmetries/virtual-casing
[2] github.com/fhindenlang/biest_to_python

Code specific questions

- The main C++ code of `BIEST` is written with only templates, and thus they are only included in the compilation against a `.cpp` file, being an executable (or an interface?). What is the reason this is done like this?

  Answer: easier to compile against other codes, since no need to compile Biest as a library first.

- How do you choose the coordinate system? (right now its (R, ϕ, Z), which is then also used for the vector components)

  Answer: (R, ϕ, Z) *is actually cartesian* (x, y, z)*, but can be used likewise in the axi-symmetric case...*

- How do you then control axi-symmetry, only by choosing one 'toroidal' integration point?

  Answer: Yes, if only one plane is given, its supposed to be axi-sym. If more planes are given, coordinates must then be cartesian (x, y, z)*!!*

- Is there a reason for setting `NFP=4` in the axi-symmetric case?

  Answer: Yes, its again a trick for the axi-symmetric case to get enough integration points!

- Can one input the geometry with a certain set of points and get the result on a different/higher sampled set of points? (useful for GVEC and also for error analysis)

  Answer: Yes, its the last Nt,Np parameters in the setup

- Can one evaluate the field(s) at other points in the outer domain, for post-processing?

  Answer: An additional routine will be added b Dhairya (thanks!)

More physical questions

- For the free-boundary computations (especially in 3D), we need to compute the coil field on the plasma boundary.
  If we can define a smooth 'outer domain boundary' (excluding the coils, including the plasma domain):
  is it possible to use BIEST to evaluate the coil field once at that boundary and use it to evaluate the field on the plasma boundary?

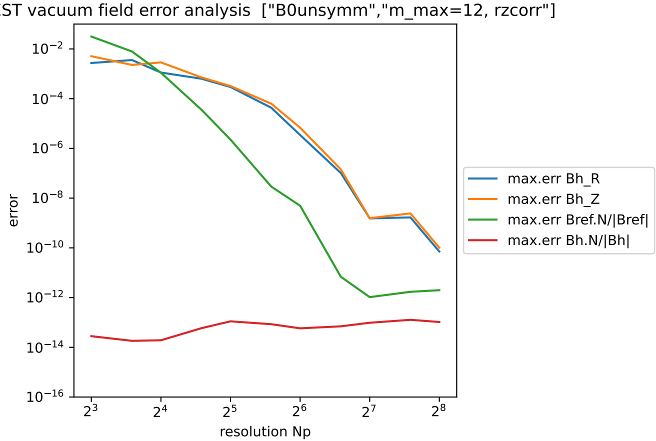
  Answer: Possible via virtual casing, but not recommendable from an accuracy vs. computational effort view.

  Should be computed via Biot-Savard directly from the coils (without Mgrid file!)
- Instead of computing an additional contribution to B_0 representing the toroidal current from a loop around the plasma, could we just pass that current as a constraint?

  Answer: Dhairya will look into this, it should be possible to pass B_{coil} and only the current from the loop integral

Appendix

Results: Case B_0 unsymmetric: Increase N_p of output

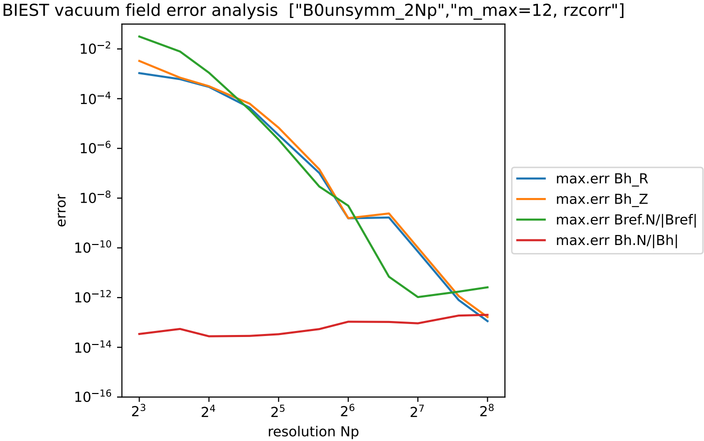


$$N_p^{\text{out}} = N_p$$

$\Rightarrow$ Increasing the number of output points increases the cost, and improves the error ($B_0 \cdot N$ is higher resolved)

$\Rightarrow$ Same errors for same N_p^{out}. Note that increasing N_p is not exactly the same!

Results: Case B_0 unsymmetric: Increase N_p of output

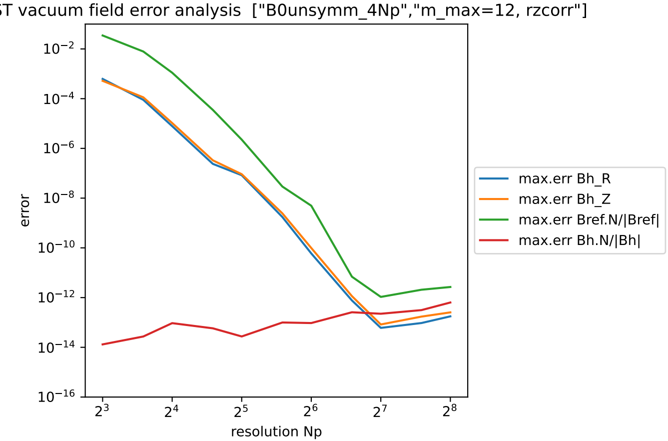


$$N_p^{\text{out}} = 2N_p$$

$\Rightarrow$ Increasing the number of output points increases the cost, and improves the error ($B_0 \cdot N$ is higher resolved)

$\Rightarrow$ Same errors for same N_p^{out}. Note that increasing N_p is not exactly the same!

Results: Case B_0 unsymmetric: Increase N_p of output

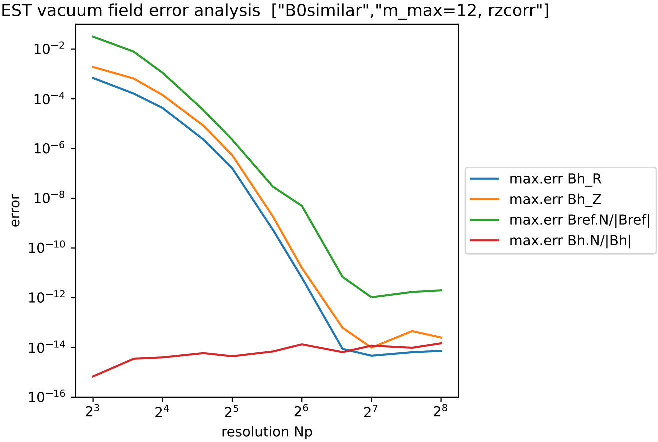


$$N_p^{\text{out}} = 4N_p$$

$\Rightarrow$ Increasing the number of output points increases the cost, and improves the error ($B_0 \cdot N$ is higher resolved)

$\Rightarrow$ Same errors for same N_p^{out}. Note that increasing N_p is not exactly the same!

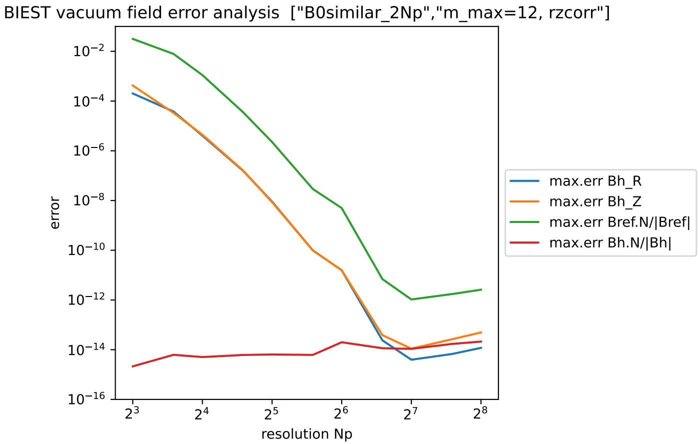
Results: Case B_0 similar: Increase N_p of output



$$N_p^{\text{out}} = N_p$$

$\Rightarrow$ Increasing the number of output points increases the cost but error remains similar, thus here $B_0 \cdot N$ is already well resolved

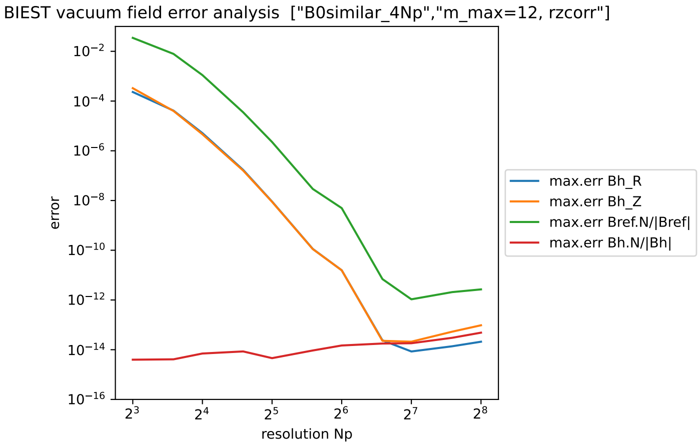
Results: Case B_0 similar: Increase N_p of output



$$N_p^{\text{out}} = 2N_p$$

$\Rightarrow$ Increasing the number of output points increases the cost but error remains similar, thus here $B_0 \cdot N$ is already well resolved

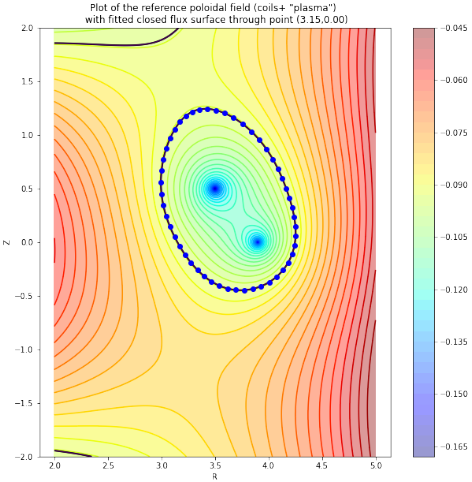
Results: Case B_0 similar: Increase N_p of output

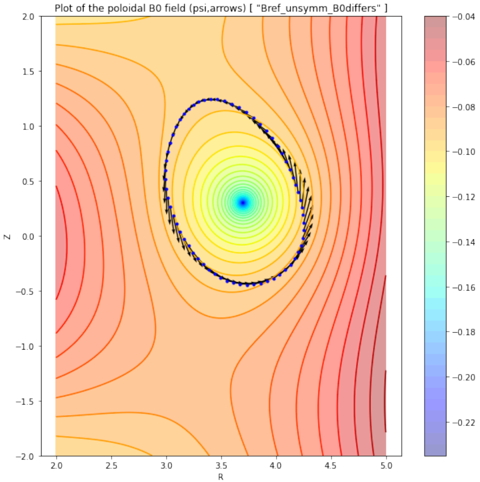


$$N_p^{\text{out}} = 4N_p$$

$\Rightarrow$ Increasing the number of output points increases the cost but error remains similar, thus here $B_0 \cdot N$ is already well resolved

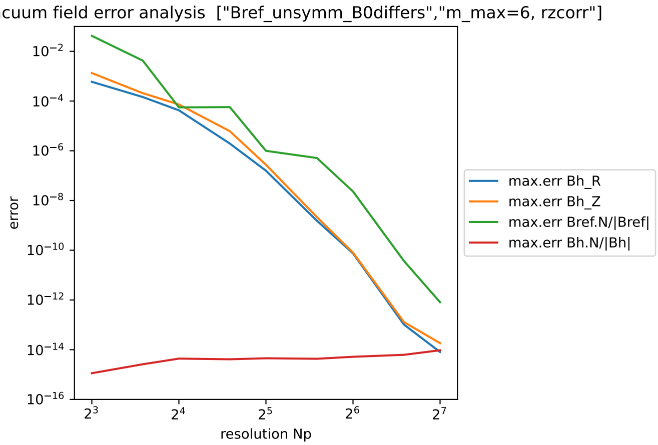
2D Testcase with B_{ref} unsymmetric





- Fully unsymmetric flux surface (left)
- B_0 including a single 'plasma' current differs (right)

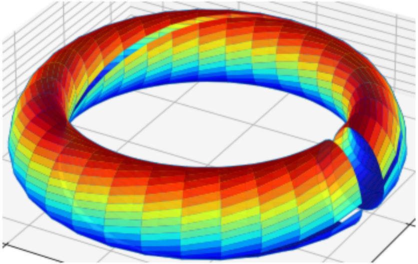
Results: B_{ref} unsymmetric



$$N_p^{\text{out}} = 4N_p$$

$\Rightarrow$ Axisymmetric case ($N_t = 1, NFP = 4$ behaves as expected. Same results for '3D' case with $N_t = 4, NFP = 1$.

Results: B_{ref} unsymmetric, 3D test!



$$N_p^{\text{out}} = 4N_p \ , \ N_t = N_p$$

Full 3D testcase from the axi-symmetric testcase by adding a 'Moebius torque'

$$R(\vartheta, \phi) = r(\theta + \phi)$$

$\Rightarrow$ Converges, but slightly slower, (max. resolution: 96x96 input 4x96x4x96 output!)