CON2020 Current Sheet Model Code

IDL: https://github.com/marissav06/con2020_idl
MATLAB: https://github.com/marissav06/con2020_matlab
Python: https://github.com/gabbyprovan/con2020

Developed by: Fran Bagenal, Marty Brennan, Matt James, Gabby Provan, Marissa Vogt, and Rob Wilson, with thanks to Jack Connerney and Masafumi Imai

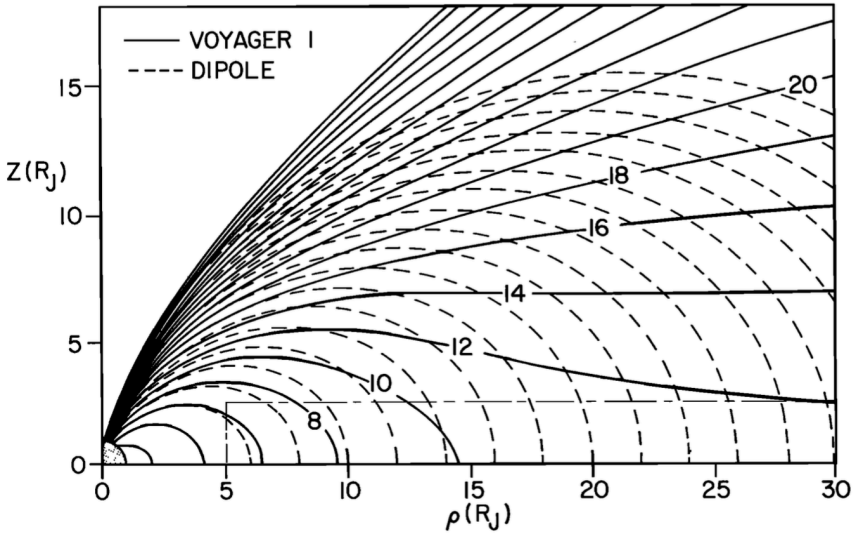
This document last updated Sept. 13, 2021

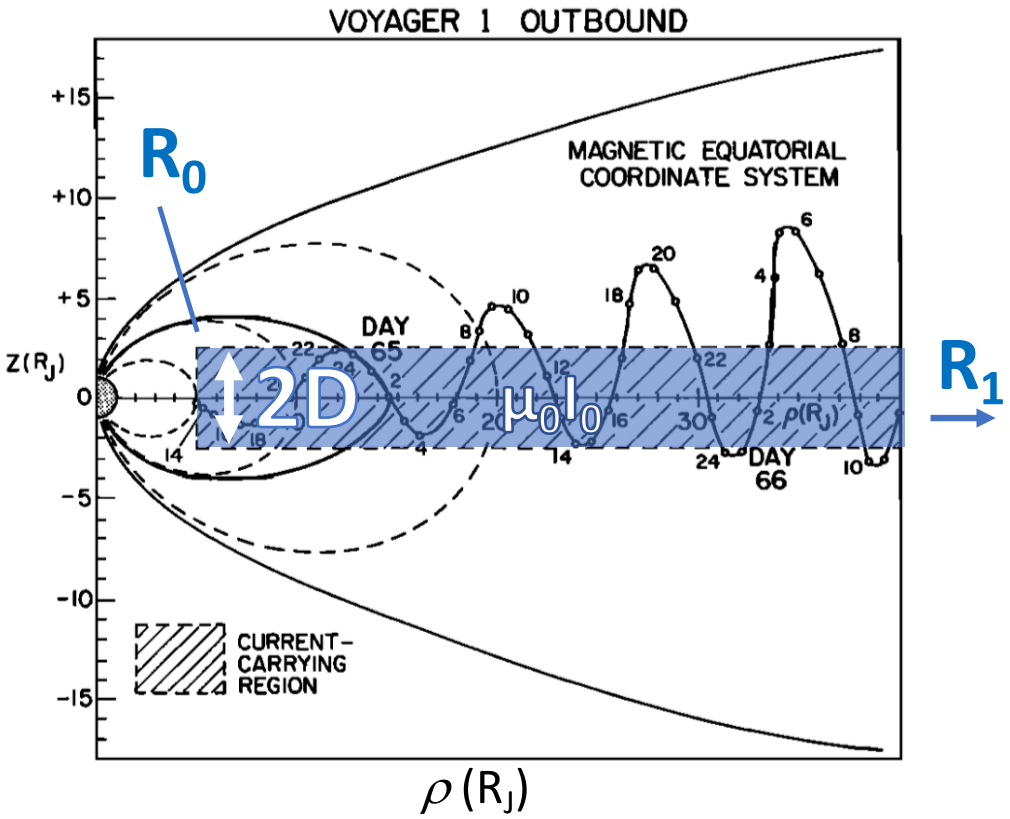
In this document

- Intro to the Connerney et al. (1981, 2020) current sheet model
- The CON2020 code:
  - Where to download the code
  - Inputs, outputs, and optional parameters
  - Approach and equations used
  - Numerical integration assumptions
- Code testing details
  - Accuracy checks
  - Time test results for IDL, MATLAB, and Python
- Future projects
- Team contact information for questions
- References

1. The Connerney et al. (1981, 2020) current sheet model

Connerney et al. 1981: Voyager-era model





- Connerney et al. (1981) used Voyager observations to model the field produced by a thin annular disc of current
- The current disc has parameters that are fit to data: inner edge R_0, outer edge R_1, half-thickness D, and current density
- The model provides the magnetic field produced by Jupiter's current sheet, or the perturbation field.

Connerney et al. 1981

CON2020: Juno-era update to 1981 model

Table 1
Magnetodisc Parameters

Parameter	Value	Description	Units
R_0	7.8	Disc inner radius	Jovian radii
R_1	51.4	Disc outer radius	Jovian radii
D	3.6	Half thickness	Jovian radii
$\mu_0 I/2$	139.6	Current constant	nT
θ_D	9.3	Disc normal from rotation axis	degrees
φ_D	204.2	Azimuth angle of disc normal	degrees

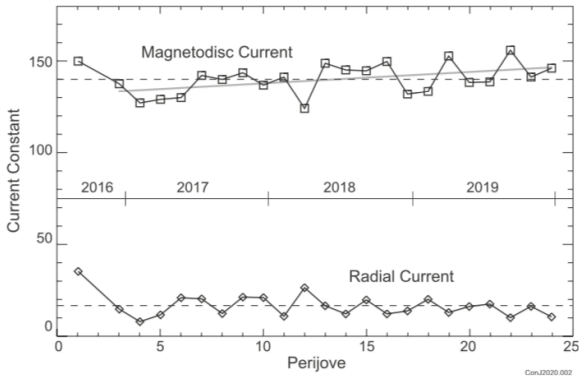


Figure 6. Time variability of the magnetodisc and radial current intensities obtained in fitting data from each Juno periJove independently. Dashed lines indicate average values (139.6 and 16.7 respectively) and the half-tone line represents a linear fit to magnetodisc currents PJ3–PJ24.

Connerney et al. 2020

Connerney et al. (2020) fit Juno magnetic field data to an updated version of the 1981 Voyager-era model and obtained new fits for the current disk parameters (see table at left).

The two major changes to 1981 model are:

1) Inclusion of a radial current, which yields an azimuthal field (see next slide)
2) Fit the current constant parameters ($\mu_0 I_0$, the azimuthal current density, and $\mu_0 I_\rho$, the radial current constant) to each orbit individually, showing temporal variability – see figure at left

Azimuthal field derivation

- The azimuthal field due to the radial current (I_ρ) can be derived from Ampere's Law:

$$\int B \cdot dl = \mu_0 I_{encl} \qquad \textit{MKS units}$$

$$B_\varphi 2\pi r = 4\pi \times 10^{-7} I_{encl} \qquad \textit{B in Webers/m}^2\textit{, I in Amperes, r in m}$$

$$B = \frac{200\, I_{encl}}{r} \qquad \textit{B in nT, I in } 10^6 \textit{ Amperes, r in m}$$

$$B = 2.7975\, \frac{I_{encl}}{r} \qquad \textit{B in nT, I in } 10^6 \textit{ Amperes, r in } R_J$$

- Connerney et al. (2020) expresses the field in terms of a "radial current constant" $\frac{\mu_0 I_\rho}{2\pi\rho}$, which has an average value of 16.7 nT. The final equations for the azimuthal field are:

  $B_\varphi = 2.7975\, \frac{\mu_0 I_\rho}{2\pi\rho}$ for |z| > D, otherwise $B_\varphi = 2.7975\, \frac{\mu_0 I_\rho}{2\pi\rho} \frac{|z|}{D}$

Connerney et al. (1981) model equations

- Connerney et al. (1981) give equations for B_ρ and B_z, the radial and z-component of the magnetic field in current sheet cylindrical coordinates.
  - Different equations apply to different regions of space (e.g. |z| < *D*)
- The magnetic field is given by integrals of Bessel functions (eqs. 14, 15, 17, 18) but the paper also gives analytic versions (eqs. A1, A2, A7, A8, A9).
- Edwards et al. (2001) derived divergence-free versions of the analytic equations – see next slide

Connerney et al. 1981 Integral Equations

For |z| > D:

$$B_\rho = (\text{sgn } z)\,\mu_0 I_0 \int_0^\infty J_1(\lambda\rho)J_0(\lambda a)\sinh(\lambda D)e^{-\lambda|z|}\frac{d\lambda}{\lambda}$$

$$B_z = \mu_0 I_0 \int_0^\infty J_0(\lambda\rho)J_0(\lambda a)\sinh(\lambda D)e^{-\lambda|z|}\frac{d\lambda}{\lambda}$$

For |z| < D:

$$B_\rho^{\,i} = \mu_0 I_0 \int_0^\infty J_1(\lambda\rho)J_0(\lambda a)\sinh(\lambda z)e^{-\lambda D}\frac{d\lambda}{\lambda}$$

$$B_z^{\,i} = \mu_0 I_0 \int_0^\infty J_0(\lambda\rho)J_0(\lambda a)\{1 - e^{-\lambda D}\cosh(\lambda z)\}\frac{d\lambda}{\lambda}$$

Edwards et al. (2001) equations

$$B_\rho(\rho,z) \approx \frac{\mu_0 I_0}{2}\left\{\frac{\rho}{2}\left[\frac{1}{\sqrt{(z-D)^2+a^2}} - \frac{1}{\sqrt{(z+D)^2+a^2}}\right]\right.$$
$$\left. + \frac{\rho^3}{16}\left[\frac{(a^2-2(z-D)^2)}{(a^2+(z-D)^2)^{5/2}} - \frac{(a^2-2(z+D)^2)}{(a^2+(z+D)^2)^{5/2}}\right]\right\}, \quad (9a)$$

$$B_z(\rho,z) \approx \frac{\mu_0 I_0}{2}\left\{\log\left[\frac{(z+D)+\sqrt{(z+D)^2+a^2}}{(z-D)+\sqrt{(z-D)^2+a^2}}\right]\right.$$
$$\left. + \frac{\rho^2}{4}\left[\frac{(z+D)}{((z+D)^2+a^2)^{3/2}} - \frac{(z-D)}{((z-D)^2+a^2)^{3/2}}\right]\right\}. \quad (9b)$$

Applicable to the region where $\rho < R_0$

Corresponding analytic equations in Connerney et al. (1981) are eqs. A1 and A2.

$$B_\rho(\rho,z) \approx$$
$$\frac{\mu_0 I_0}{2}\left\{\begin{array}{l} \frac{1}{\rho}[\sqrt{(z-D)^2+\rho^2} - \sqrt{(z+D)^2+\rho^2}] \\ + \frac{\rho a^2}{4}\left[\frac{1}{((z+D)^2+\rho^2)^{3/2}}\right. \\ \left. - \frac{1}{((z-D)^2+\rho^2)^{3/2}}\right] \\ + \frac{2}{\rho}\left\{\begin{array}{ll} D & \text{for } z \geqslant D \\ z & \text{for } |z| \leqslant D \\ -D & \text{for } z \leqslant -D \end{array}\right. \end{array}\right\} \quad (13a)$$

and

$$B_z(\rho,z) \approx \frac{\mu_0 I_0}{2}\left\{\log\left[\frac{(z+D)+\sqrt{(z+D)^2+\rho^2}}{(z-D)+\sqrt{(z-D)^2+\rho^2}}\right]\right.$$
$$\left. + \frac{a^2}{4}\left[\frac{(z+D)}{((z+D)^2+\rho^2)^{3/2}} - \frac{(z-D)}{((z-D)^2+\rho^2)^{3/2}}\right]\right\}. \quad (13b)$$

Applicable to the region where $\rho > R_0$

Corresponding analytic equations in Connerney et al. (1981) are eqs. A7, A8, and A9.

Analytic vs. integral equation error estimates

from Connerney et al. (1981), left, and Edwards et al. (2001), right

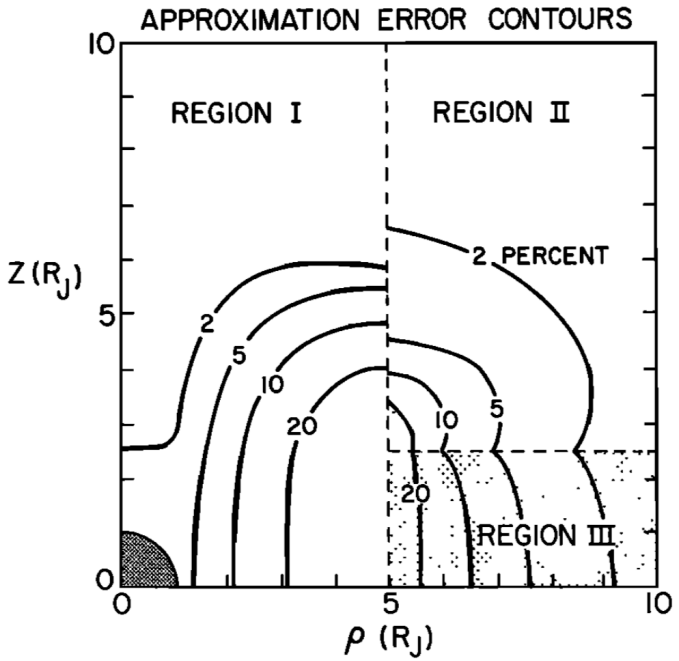
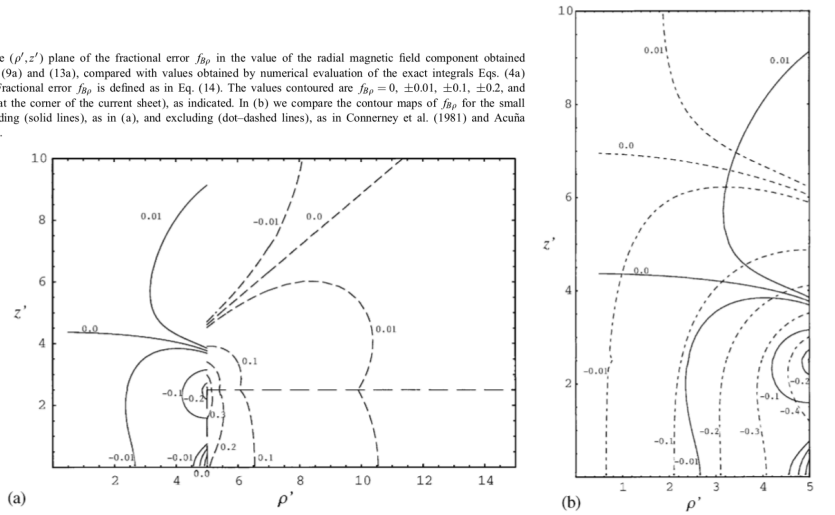


Fig. A1. Contour map showing maximum errors in current sheet model magnetic field components, B_ρ or B_z, in three regions of ρ, z space for which simple analytic forms exist (see the appendix).

Fig. 5. In (a) we show a contour plot on the (ρ', z') plane of the fractional error $f_{B\rho}$ in the value of the radial magnetic field component obtained from the approximate analytic formulae Eqs. (9a) and (13a), compared with values obtained by numerical evaluation of the exact integrals Eqs. (4a) and (5a). The format is the same as Fig. 2. Fractional error $f_{B\rho}$ is defined as in Eq. (14). The values contoured are $f_{B\rho} = 0$, ± 0.01, ± 0.1, ± 0.2, and ± 0.3 ($+0.4$ just being visible beyond $\rho' = 5$ at the corner of the current sheet), as indicated. In (b) we compare the contour maps of $f_{B\rho}$ for the small ρ approximation for B_ρ, Eq. (9a), both including (solid lines), as in (a), and excluding (dot–dashed lines), as in Connerney et al. (1981) and Acuña et al. (1983), the second term in the equation.



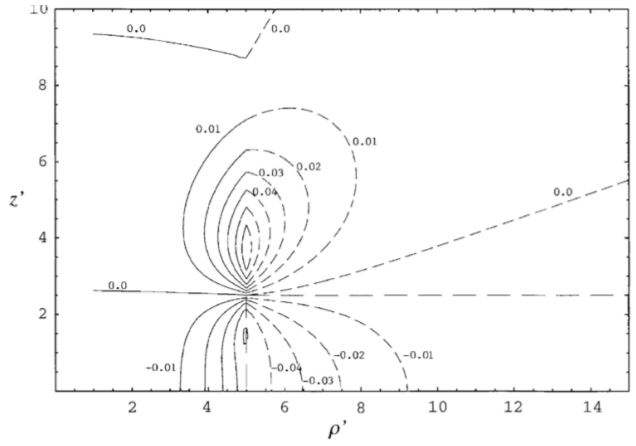


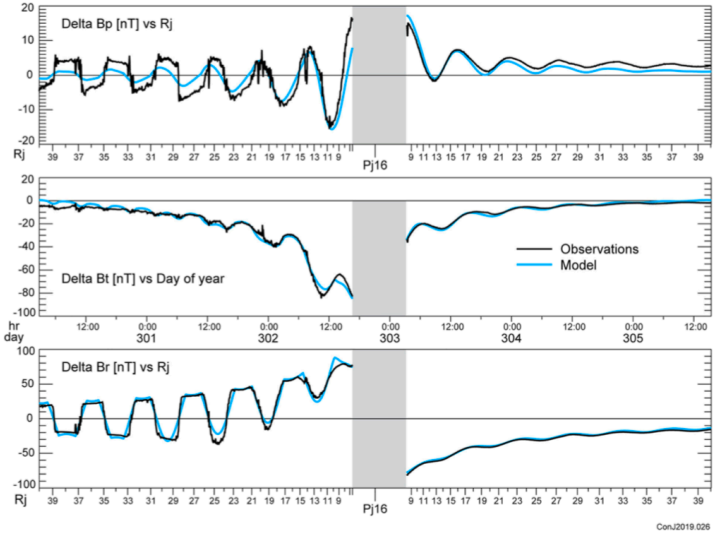
Fig. 7. Contour plot on the (ρ', z') plane of the fractional error f_{Bz} in the value of the axial magnetic field component obtained from the approximate analytic formulae Eqs. (9b) and (13b), compared with values obtained by numerical evaluation of the exact integrals Eqs. (4b) and (5b). The format is the same as in Fig. 2. The fractional error f_{Bz} is defined as in Eq. (14). The values contoured are $f_{B\rho} = 0$, ± 0.01, ± 0.02, ± 0.03, ± 0.04, ± 0.05 and $+0.06$. The contours are continuous across $\rho' = 5$ in this case.

2. The CON2020 current sheet code

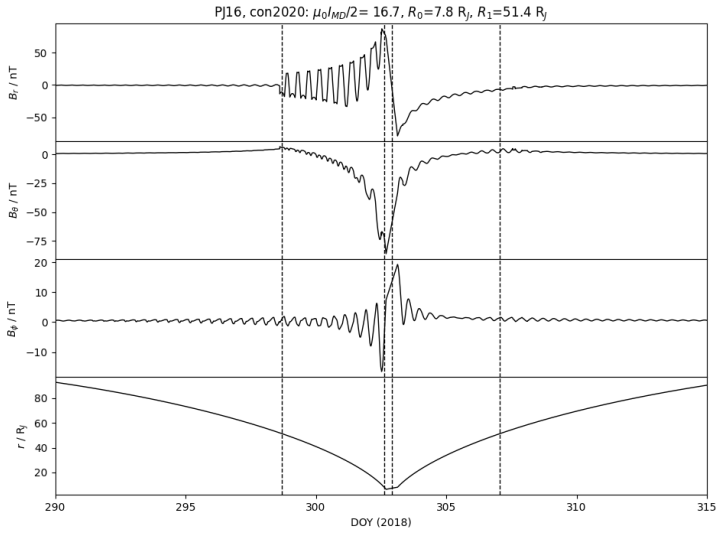
CON2020 code

- We have developed code that calculates the CON2020 current sheet magnetic field
  - IDL: https://github.com/marissav06/con2020_idl
  - MATLAB: https://github.com/marissav06/con2020_matlab
  - Python: https://github.com/gabbyprovan/con2020 (This is Python 3 code, not Python 2.7)
- This code does **not** incorporate or assume any internal field model (i.e. VIP4, JRM09)

Connerney et al. (2020) Juno data and model fit



CON2020 Python code output



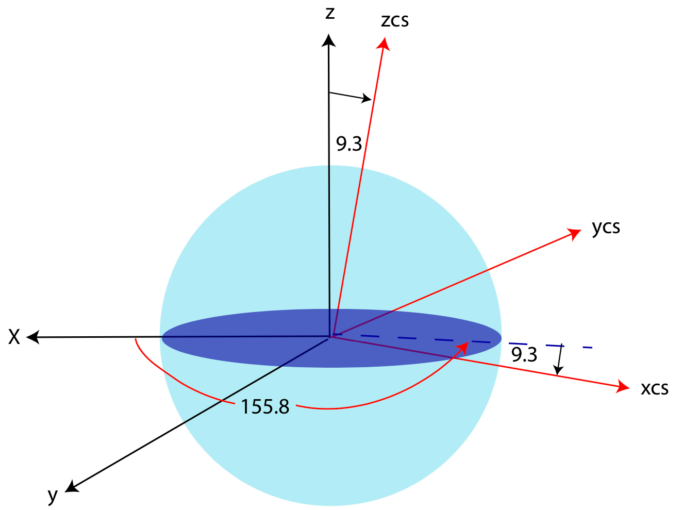
CON2020 code

The implementation varies slightly, but all versions of the CON2020 code (IDL, MATLAB, Python) have the same inputs, outputs, and options.

- Inputs: position in SIII spherical coordinates (R in R_J [1 R_J = 71492 km], colatitude θ in radians, **right-handed** longitude φ in radians) or cartesian coordinates (X, Y, Z in R_J, also right-handed). The code is vectorized (it will take in an array of positions).
- Outputs: B_R, B_θ, and B_φ in SIII right-handed spherical coordinates or B_X, B_Y, B_Z in SIII cartesian coordinates, in nT
- Optional inputs/options:
  - Variable parameters (R_0, R_1, D, $\mu_0 I_0$, $\mu_0 I_\rho$) that the user can specify to match specific time intervals/orbits
    - To turn off the radial current the user can specify $\mu_0 I_\rho$ = 0
  - Current sheet tilt θ_D and azimuthal angle of the tilt φ_D
  - Default values given by Connerney et al. (2020), table 1
  - Whether to use the Connerney et al. (1981) integral equations or the Edwards et al. (2001) analytic equations, or a hybrid version (using the integral equations, which are slower, only where there is a large error compared to using the analytic equations, see slide 9). Hybrid is the default.

Approach and equations used

- The code takes the input position (SIII coordinates) and rotates to current sheet coordinates using θ_D and φ_D
- The code then calculates the magnetic field due to the current sheet using integral equations (Connerney et al. 1981, eqs. 14, 15, 17, and 18) or analytic equations (Edwards et al. 2001, eqs. 9a, 9b, 13a, 13b).
- The code then takes the model output (B_ρ, B_φ, B_Z) in current sheet coordinates and rotates into the output coordinates – (B_R, B_θ, and B_φ) or (B_X, B_Y, B_Z)



Example of rotation to current sheet coordinate for the default θ_D and φ_D values from Connerney et al. (2020)

Numerical integral assumptions for integral equations

- The code performs the integration using the pre-defined integration function for each programming language (e.g. `int_tabulated` for IDL)
- To identify the appropriate step size $d\lambda$ and integration limit λ_{max} we calculated the integrals at a range of ρ and *z* values and examined how they changed with $d\lambda$ and λ_{max}.
- For example, the plots on the next page show how the B_ρ integration function (the part inside the red ovals at right) falls off with λ and illustrates our choices of λ_{max}, which varies with ρ and *z*.

B_ρ and B_ρ from Connerney et al. (1981), Eqs. 14, 15, 17, 19

$$B_\rho = (\text{sgn } z)\,\mu_0 I_0 \int_0^\infty J_1(\lambda\rho) J_0(\lambda a) \sinh(\lambda D) e^{-\lambda|z|} \frac{d\lambda}{\lambda}$$

$$B_\rho{}^i = \mu_0 I_0 \int_0^\infty J_1(\lambda\rho) J_0(\lambda a) \sinh(\lambda z) e^{-\lambda D} \frac{d\lambda}{\lambda}$$

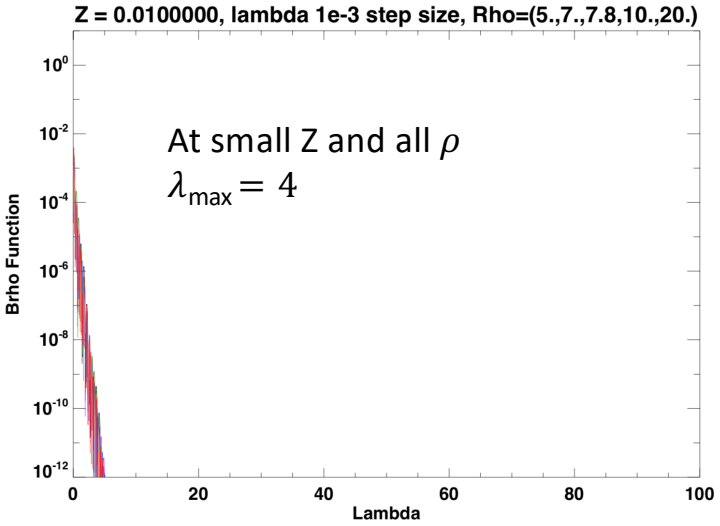
$$B_z = \mu_0 I_0 \int_0^\infty J_0(\lambda\rho) J_0(\lambda a) \sinh(\lambda D) e^{-\lambda|z|} \frac{d\lambda}{\lambda}$$

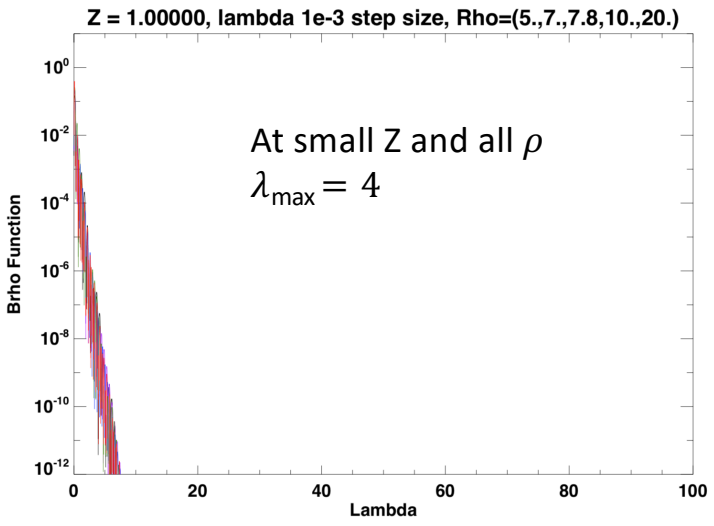
$$B_z{}^i = \mu_0 I_0 \int_0^\infty J_0(\lambda\rho) J_0(\lambda a) \{1 - e^{-\lambda D} \cosh(\lambda z)\} \frac{d\lambda}{\lambda}$$

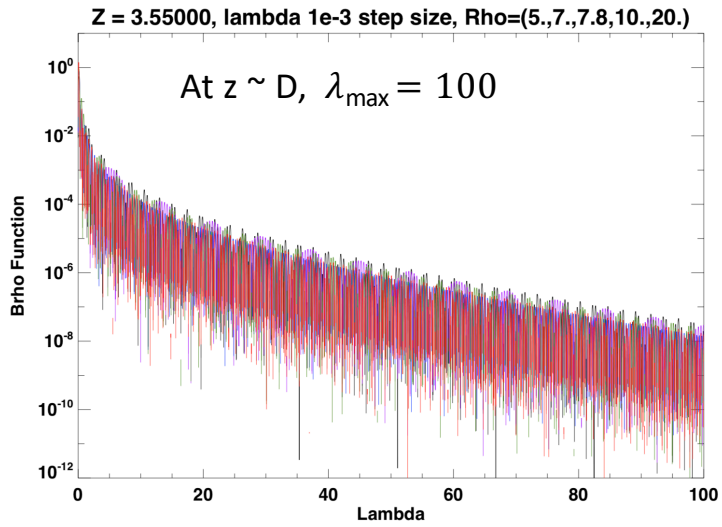
Testing how the quanitity inside the B_ρ **integral** varies as a function of λ for Rho = **5, 7, 7.8, 10, 20** at different Z values (for all, R_0 = 7.8, D = 3.6)

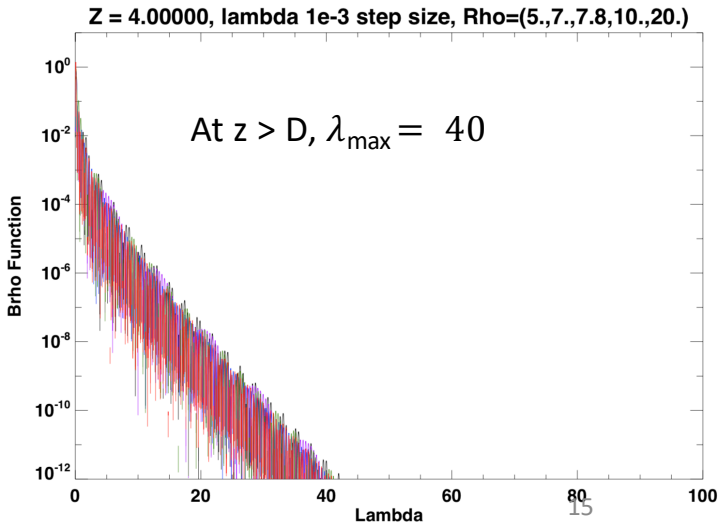
$$B_\rho = (\text{sgn } z)\,\mu_0 I_0 \int_0^\infty J_1(\lambda\rho) J_0(\lambda a) \sinh(\lambda D) e^{-\lambda|z|} \frac{d\lambda}{\lambda}$$

$$B_\rho{}^i = \mu_0 I_0 \int_0^\infty J_1(\lambda\rho) J_0(\lambda a) \sinh(\lambda z) e^{-\lambda D} \frac{d\lambda}{\lambda}$$







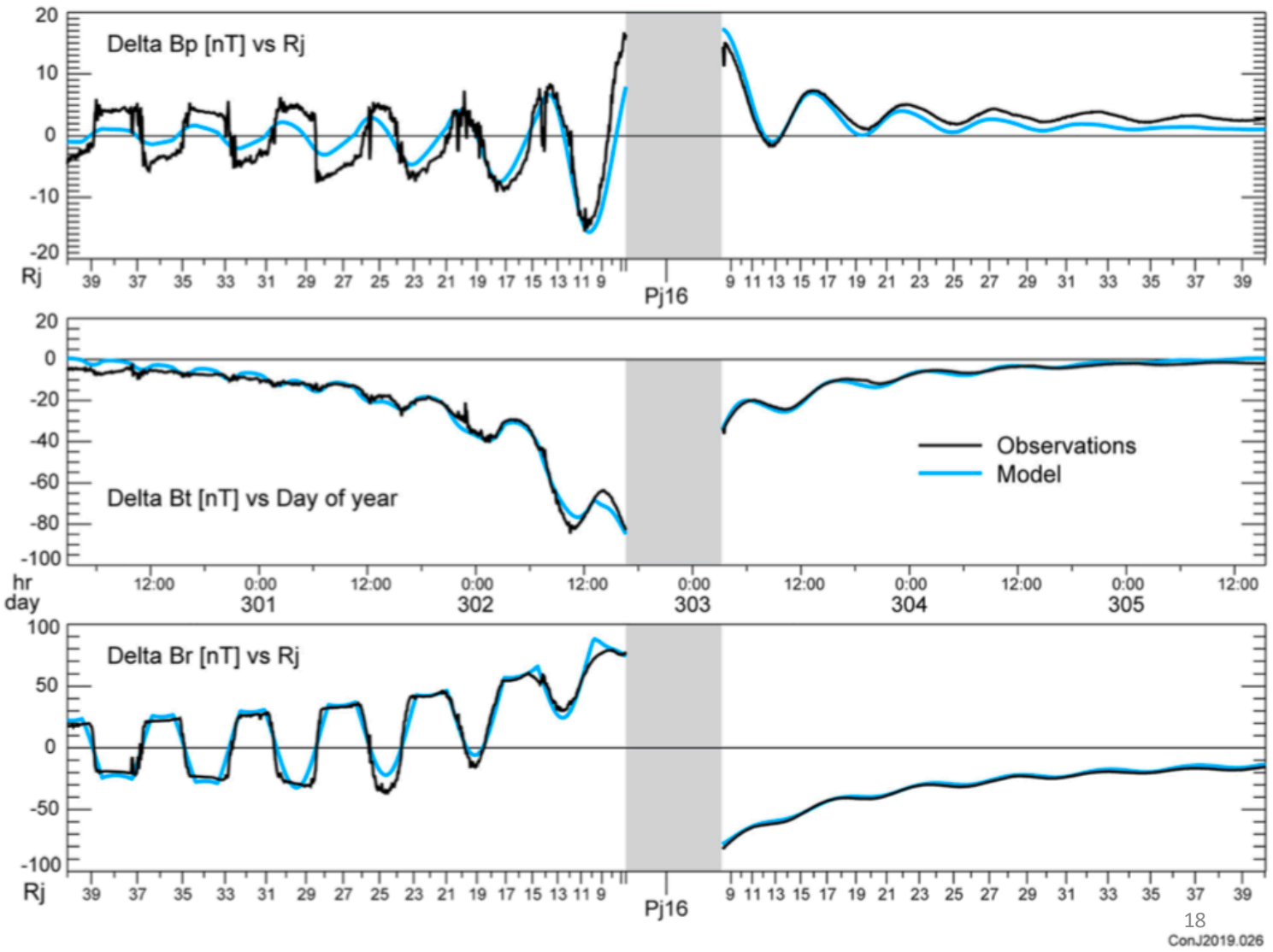


3. Code testing details

Accuracy testing and error checks

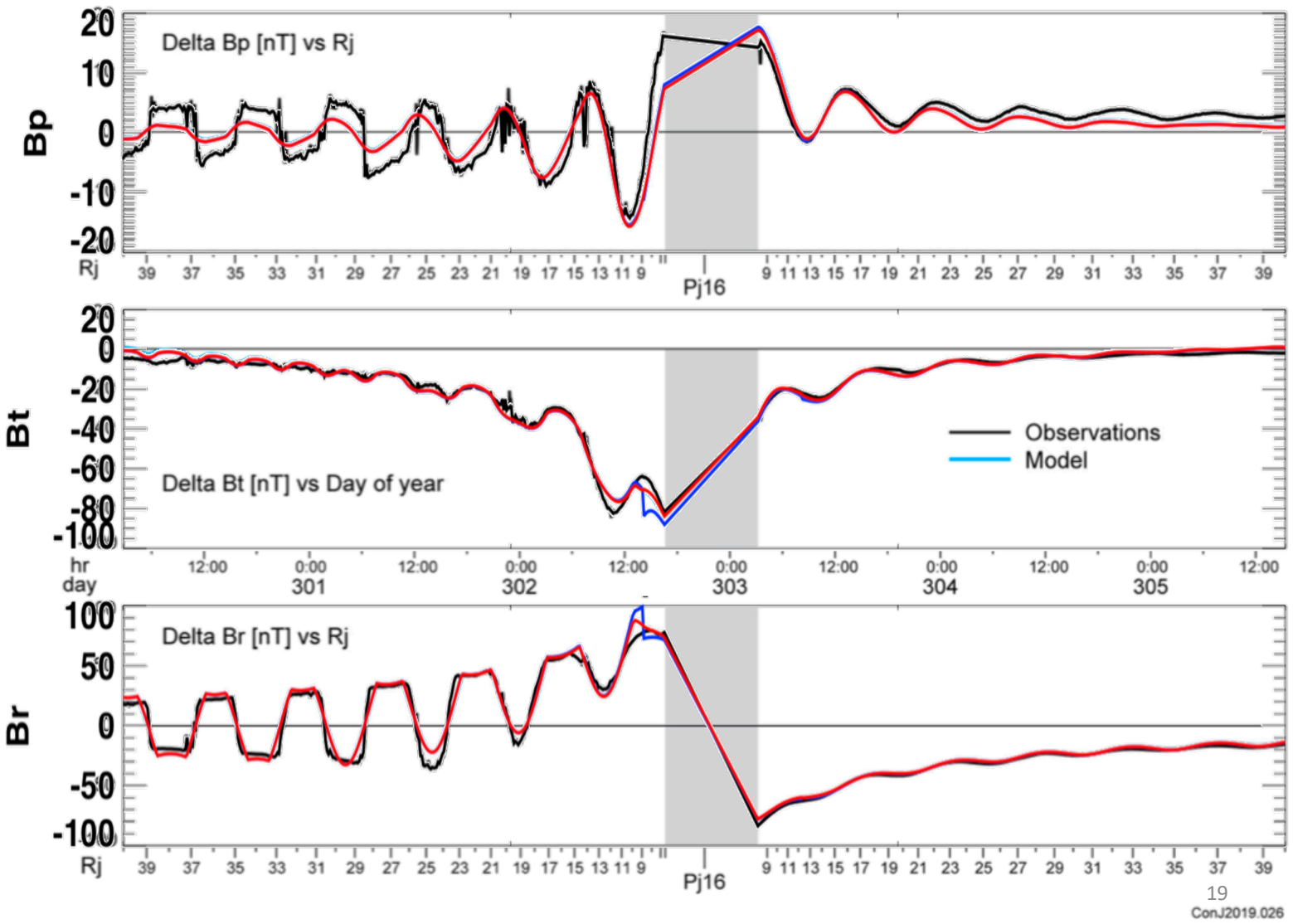
- Our testing and validation process included comparing output to published values and figures.
  - See figures on next two pages
- We compared outputs of the IDL, MATLAB, and Python codes and confirmed that they agree to within rounding errors
- The code has an optional error check to test whether the user provides an obviously wrong input (i.e. provides an input radial distance > “200”, suggesting input in km rather than R_J, or input longitude is “200”, suggesting degrees not radians).

Connerney et al. (2020) figure 2 for comparison to IDL code output (next slide)



Connerney et al. (2020) figure 2 for comparison to IDL code output

IDL code output:
Integral equations in red, Analytic equations in bright blue



ConJ2019.026

Speed testing

- We calculated the times to run the CON2020 model code along one Juno orbit (75k records, run through a `for` loop for the scalar runs):

		Matlab	IDL	Python	
Analytic	Scalar	0.774575	2.07722	7.23478	(s)
Analytic	Vectorized	0.0637563	0.0322421	0.368392	(s)
Hybrid	Scalar	22.3733	34.1308	20.1937	(s)
Hybrid	Vectorized	10.74	17.8947	12.7495	(s)
Integral	Scalar	4208.75	5548.01	2143.63	(s)
Integral	Vectorized	1826.78	3056.19	2084.64	(s)

- MATLAB is typically slowest and Python is typically fastest.
- The code runs faster with cartesian inputs/outputs than with spherical inputs/outputs
- The full integral version significantly slower than using the analytic equations. In the hybrid run Juno is only briefly in the region where the code uses the integral equations.

4. Summary

Team contact info, Future Work, & References

Questions?

- Please reach out to the group by email with general questions:
  - Fran Bagenal (bagenal@lasp.colorado.edu), Marty Brennan (martin.brennan@jpl.nasa.gov), Matt James (mkj13@leicester.ac.uk), Gabby Provan (gp31@leicester.ac.uk), Marissa Vogt (mvogt@bu.edu), and Rob Wilson (rob.wilson@lasp.colorado.edu)
- For questions about the code specific to your preferred programming language, contact:
  - IDL: Marissa Vogt and Rob Wilson
  - Python: Gabby Provan and Matt James
  - MATLAB: Marty Brennan
- We hope that community members will improve/alter the code and continue to share it through GitHub. Questions about future iterations of the code should be directed to the most recent developer.
- If you have questions about this document please contact Marissa Vogt

What other codes would the community find useful?

- We hope to provide a GitHub repository with IDL, Python, and MATLAB versions of the internal field models JRM09 and VIP4 soon.
- We also hope to provide some sample fieldline tracing code that can be used with the full field models (i.e. JRM09+CON2020).
- Please let us know if you have other suggestions, ideas, or code to contribute.

References

- Connerney, J. E. P., Acuña, M. H., & Ness, N. F. (1981). Modeling the Jovian current sheet and inner magnetosphere. *Journal of Geophysical Research*, **86**, 8370– 8384. https://doi.org/10.1029/JA086iA10p08370
- Connerney, J. E. P., Timmins, S., Herceg, M., & Joergensen, J. L. (2020). A Jovian magnetodisc model for the Juno era. *Journal of Geophysical Research: Space Physics*, 125, e2020JA028138. https://doi.org/10.1029/2020JA028138
- Edwards, T. M., Bunce, E. J., & Cowley, S. W. H. (2001). A note on the vector potential of Connerney et al.'s model of the equatorial current sheet in Jupiter's magnetosphere. *Planetary and Space Science*, **49**, 1115–1123. https://doi.org/10.1016/S0032-0633(00)00164-1