掩蔽観測のシミュレーション手順

❶新しいフライバイデータで検証するときの下準備

1. 調べたい掩蔽タイミングのExpresデータを<https://maser.lesia.obspm.fr/publications/doi/jovian-auroral-radio-source.html?lang=en>　のExPRESdata→expres\_gll\_jupiter .. .cdf からダウンロード

グラフィカル ユーザー インターフェイス, アプリケーション

自動的に生成された説明

グラフィカル ユーザー インターフェイス, テキスト, アプリケーション, メール

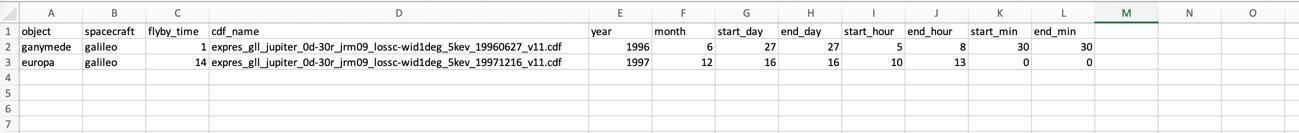
自動的に生成された説明

1. result\_for\_yasudaetal2022/expres\_cdf\_data/の下にうつす

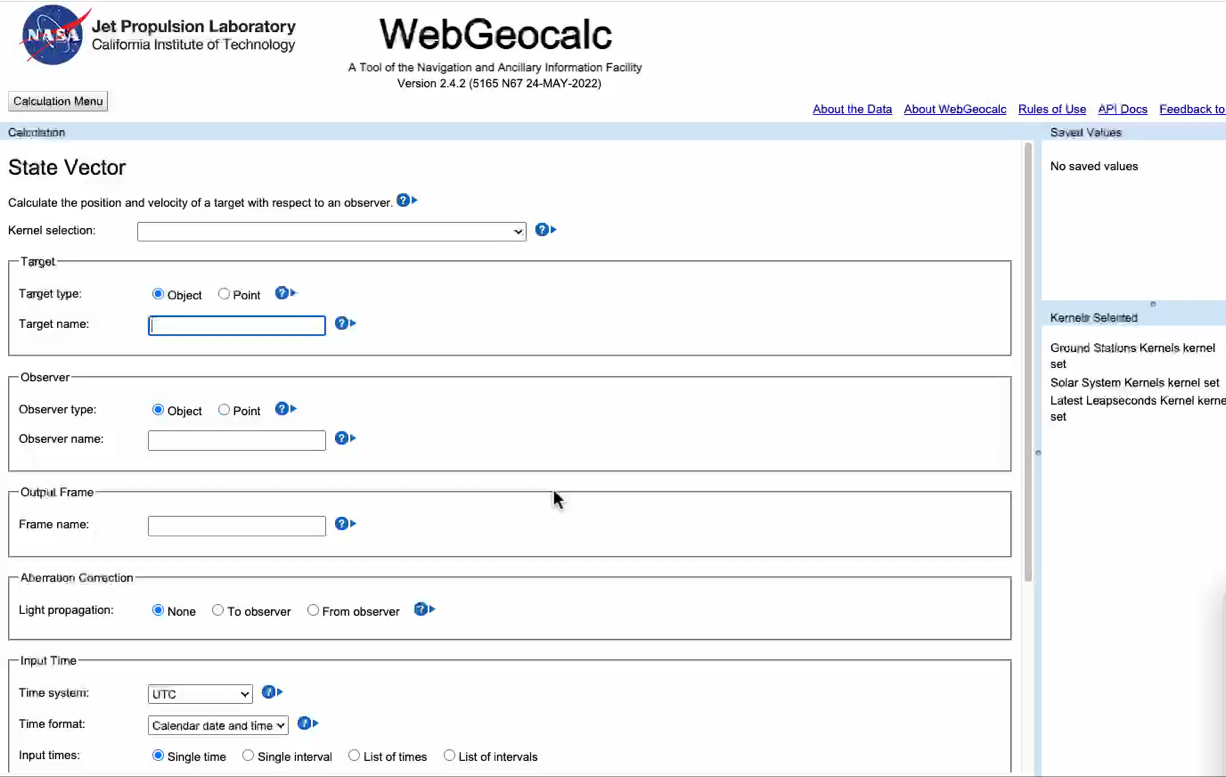
グラフィカル ユーザー インターフェイス, アプリケーション

自動的に生成された説明

1. result\_for\_yasudaetal2022/occultation\_flyby\_listに対象の天体・探査機名・フライバイの回数・cdfのファイル名・使われているデータ開始日時・終了日時を追記



1. グラフィカル ユーザー インターフェイス, テキスト, アプリケーション, メール

   自動的に生成された説明対象のフライバイ時の探査機の位置データを取得(1の手順参照)
2. result\_for\_yasudaetal2022/spacecraft\_ephemeris/の下にうつす
3. result\_for\_yasudaetal2022/occultation\_flyby\_listにcsvファイル名を追記
4. 対象のフライバイ時の月（衛星）の位置情報を取得　<https://wgc2.jpl.nasa.gov:8443/webgeocalc/#StateVector>設定は“対象のフライバイ時の探査機の位置データ”と調べたい時刻で合わせる（設定・ダウンロード方法は以下の動画の通り）
5. result\_for\_yasudaetal2022/moon\_ephemeris/の下にうつす
6. result\_for\_yasudaetal2022/occultation\_flyby\_listにcsvファイル名を追記
7. python\_Jovian\_radio\_occultation /preparing\_position\_data.py にフライバイの天体名・探査機名・フライバイの回数を入力して実行　⇨

result\_for\_yasudaetal2022/expres\_detectable\_radio\_data\_of\_each\_flyby　に“Expresで観測可能とされている電波の情報が 0 年、1 月、2 日、3 時間、4 分、5 秒、◯、○、◯、9 周波数（MHz)、10 磁力線の経度(0~360)orイオの場合は(-1000)、11 極（北:1 南:-1)、12 x座標,13 y座標,14 z座標 を受かるパターン分だけ格納されたもの” が保存される (All\_galileo\_ganymede\_1\_Radio\_data.txtのように)

result\_for\_yasudaetal2022/expres\_detectable\_radio\_data\_of\_each\_flybyに、時間分解能を1秒に補間した電波源の位置情報が保存される、0 時間, 1 分, 2 秒, 3 周波数(MHz), 4 (0~360)orイオの場合は(-1000), 5 極（北:1 南:-1), 6 x座標, 7 y座標, 8 z座標, 9 電波源経度, 10 探査機経度, 11 電波源タイプ（A,C:1 B,D:-1)]　 (lnterpolated\_all\_galileo\_ganymede\_1\_Radio\_data.txtのように)

result\_for\_yasudaetal2022/calculated\_expres\_detectable\_radio\_data\_of\_each\_flybyには"時間方向に補間した電波源の結果と探査機・月の位置情報を使って掩蔽の判定ができるように探査機の位置情報を座標変換したもの 0 時間, 1 分, 2 秒, 3 周波数(MHz), 4 (0~360)orイオの場合は(-1000), 5 極（北:1 南:-1), 6 座標変換した時のx(tangential point との水平方向の距離), 7座標変換した時のy(tangential pointからの高さ方向の距離), 8 電波源経度, 9 探査機経度, 10 電波源タイプ（A,C:1 B,D:-1)]が保存される。(interpolated\_calculated\_all\_galileo\_ganymede\_1\_Radio\_data.txtのように)

座標変換の方法は以下の図を参照

ダイアグラム

自動的に生成された説明

1. 電波データに関する情報を追記　先行研究から完全に掩蔽されている日時（日、時、分）を目視で確認してresult\_for\_yasudaetal2022/occultation\_flyby\_listに追記するChart

   Description automatically generatedGraphical user interface, application

   Description automatically generated
2. <http://das2.org/browse/uiowa/galileo/pws/survey_electric>　からガリレオの電波データを取得する。電波データの時刻範囲（先行研究の図と同じ）を記入して、「get from Jupiter」でデータ取得。tools/result\_for\_yasudaetal2022/galileo\_radio\_data/　に移す。そのデータ名を、result\_for\_yasudaetal2022/occultation\_flyby\_listのradio\_data\_txtの欄に追記する

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application

Description automatically generated

以上、新しいフライバイを追加する時の手順

❷新しい電子密度分布で計算を行う時の手順

1. 調べたい電子密度分布のパラメータに/Users/yasudarikuto/research/icymoon\_raytracing/raytrace.tohoku/src/rtc/model/test\_plasma\_model.cpp のパラメータを変更する。

下は地表面の密度400(/cc)・スケールハイト600kmのエウロパ静水圧平衡モデル

double plasma::europa\_nonplume::getDensity(const vector &point) const ///////////////新しいプラズマモデル（z軸方向にexpで減少）

{

const double

r = std::sqrt((pow(point(0), 2.0)) + (pow(point(1), 2.0)) + (pow(point(2) + 1.601e6, 2.0)));

const double

rxy = std::sqrt((pow(point(0), 2.0)) + (pow(point(1), 2.0)));

const double

t = std::fabs(4e8 \* exp(-(r - 1.601e6) / 6e5)); //////////////エウロパ静水圧平行モデル 地表面で4.0\*10^2(/cc) スケールハイト600km

;

return t;

}

1. icymoon\_raytracing/raytrace.tohoku/src/rtc/testing/mkray3.sh　の　“出力ファイル名を指定する。”の部分で最大電子密度(/cc)&スケールハイト(km)を記入する 例）ray-Peuropa\_nonplume\_4e2\_6e2-Mtest\_simple-benchmark-LO-Z-2-FR3.984813988208770752e5
2. icymoon\_raytracing/raytrace.tohoku/src/rtc/testing/mkray3.sh　のHIGのfor文の要素を変更する。0以下では100おき、0以上では500,600,1000,1100 のように+500 +100を繰り返した数字にする。この範囲で計算範囲が十分かを確認するので、やや広めにHIGの幅を持たせておく　例) for HIG in -300 -200 -100 0 100 500 600 1000 1100 1500 1600 2000 2100 2500 2600
3. icymoon\_raytracing/raytrace.tohoku/src/rtc でmakeコマンドでコンパイル実行
4. icymoon\_raytracing/raytrace.tohoku/src/rtc/testing でmakeコマンドでコンパイル実行
5. icymoon\_raytracing/raytrace.tohoku/src/rtc/testing でbash mkray3.sh で計算実行
6. 計算終了後、tools/python\_Jovian\_radio\_occultation/occultaion\_range\_calc.py に、探査対象天体、探査機名、フライバイ回数、最大電子密度、スケールハイト、3.で指定したH I Gの最大値と最小値を記入して実行

object\_name = 'europa' # europa/ganymde/callisto

spacecraft\_name = "galileo" # galileo/JUICE(?)

time\_of\_flybies = 14 # ..th flyby

highest\_plasma = '4e2' # 単位は(/cc) 2e2/4e2/16e2

plasma\_scaleheight = '6e2' # 単位は(km) 1.5e2/3e2/6e2

raytrace\_lowest\_altitude = -300 # レイトレーシングの下端の初期高度(km) 100の倍数で

raytrace\_highest\_altitude = 2600 # レイトレーシング上端の初期高度(km) 500の倍数+100で

1. "Start position need to be far from moon"というメッセージが出力された場合、レイトレーシングの開始座標が、天体に近すぎるので、SXの値を小さくして再実行
2. 角周波数での最高高度・最低高度が出力されるので、その出力結果を参考にしてicymoon\_raytracing/raytrace.tohoku/src/rtc/testing/mkray3.sh　のHIGとFREQのfor文の範囲を指定して実行。

なお、../ python\_for\_yasudaetal2022 /tracing\_range\_'探査機名'\_'天体名'\_'フライバイ数'\_flybys/para\_'最大電子密度'\_'スケールハイト'.csv'に計算が必要な高度範囲が出力されている。

1. tools/ python\_Jovian\_radio\_occultation/detectable\_radio\_check.py 探査対象天体、探査機名、フライバイ回数、最大電子密度、スケールハイトを記入して実行。その結果が　result\_for\_yasudaetal2022/raytracing\_'天体名'\_results/'天体名'\_'最大電子密度'\_'スケールハイト'/Interpolated\_'天体名'\_'探査機名'\_'フライバイ数'\_'最大電子密度 '\_'スケールハイト'\_dectable\_radio\_data.txt',に格納される

object\_name = 'ganymede' # ganydeme/europa/calisto``

spacecraft\_name = "galileo" # galileo/JUICE(?)

time\_of\_flybies = 1 # ..th flyby

highest\_plasma = '4e2' # 単位は(/cc) 2e2/4e2/16e22

plasma\_scaleheight = '6e2' # 単位は(km) 1.5e2/3e2/6e2

1. tools/ python\_Jovian\_radio\_occultation/plot\_f-t\_2.py　に探査対象天体、探査機名、フライバイ回数、最大電子密度、スケールハイト、背景ノイズの何倍を閾値にするか、（csvファイルに記載した開始時刻からプロット時の開始・目盛・終了に使う時刻までの秒数、それに対応する時刻）を記入して実行。「（）の内容は自動で代入されるように改良済み」

A computer screen shot of a code

Description automatically generated

* 1. '../result\_for\_yasudaetal2022/raytracing\_'+天体名+'\_results/' +天体名+'\_'+最大電子密度+'\_'+スケールハイト+'/と'../result\_for\_yasudaetal2022/f-t\_plot\_'+探査機名+'\_'+天体名+'\_'+フライバイ数+'\_flyby/radio\_boundary\_intensity\_="閾値にするノイズの倍率"dB, にf-tダイアグラム（full, ingress. Egress）の３つがそれぞれ保存される。( "interpolated\_"探査機名"\_"天体名"\_"フライバイ数"\_"最大電子密度"\_"スケールハイト"\_boundary\_int="閾値にするノイズの倍率"dB\_”full or ingress or egress"のような名前で）
  2. '../result\_for\_yasudaetal2022/radio\_data\_occultation\_timing\_'+探査機名+'\_'+天体名+'\_'+フライバイ数 '\_flyby/'+天体名+ '\_'+"閾値にするノイズの倍率"dB +'\_ingress\_time\_data.txt', のようにingressとegressのそれぞれの時に、電波データから閾値を超えるタイミングを周波数ごとに調べた配列が保存される。（結果は[[周波数一覧][掩蔽開始時刻一覧]]の二次元データ 掩蔽開始時刻…レイトレーシング開始時刻からの秒数で出力）
  3. レイトレーシングにおける掩蔽開始時刻・終了時刻を周波数ごとに計算したものが保存される　'../result\_for\_yasudaetal2022/raytracing\_occultation\_timing\_'+探査機名+'\_'+天体名+'\_'+フライバイ数+'\_flyby/interpolated\_'+探査機名+'\_'+天体名+'\_'+フライバイ数+'\_'+最大電子密度+'\_'+スケールハイト+'\_ingress\_time\_list.txt'
  4. '../result\_for\_yasudaetal2022/radio\_raytracing\_occultation\_timing\_def\_'+探査機名+'\_'+天体名+'\_'+フライバイ数+'\_flyby\_radioint\_"閾値にするノイズの倍率"dB +'/interpolated\_’+天体名+’\_’+最大電子密度+'\_'+スケールハイト+'\_ingress\_defference\_time\_'+電波の種類（A~D）+'\_'閾値にするノイズの倍率"dB.txt'にレイトレーシングと観測結果の掩蔽開始時刻・終了時刻の差分を周波数ごとに計算したものが保存される。

1. tools/ python\_Jovian\_radio\_occultation /evaluate\_ft\_2.py　の天体名、探査機名、フライバイ数、調べる周波数幅、閾値の電波強度、掩蔽開始or終了時(ingress or egress)、電波の種類（A~D）を記入して実行。今までのレイトレーシング結果から、どの電子密度分布が観測とよくあっているのか、その結果が図とテキストで出力される。出力先は'../result\_for\_yasudaetal2022/evaluate\_f-t\_diagram\_plot\_'+探査機名+'\_'+天体名+'\_'+フライバイ数+'\_flyby\_radioint\_'+"閾値にするノイズの倍率"dB

####################################################

object\_name = 'ganymede' # ganydeme/europa/calisto`

spacecraft\_name = "galileo" # galileo/JUICE(?)

time\_of\_flybies = 1 # ..th flyby

"""lowest frequency and highest\_freqency(MHz)"""

using\_frequency\_range = [5.5e-1, 6] # egress

boundary\_intensity\_str = '7e-16' # '7e-16' '1e-15'

occultaion\_type = 'egress' # 'ingress' or 'egress'

radio\_type = 'A' # 'A' or 'B' or 'C' or 'D'

❸掩蔽観測の観測地点をプロットする手法

1. 観測で利用している時間帯における　“衛星”系座標・木星中心から見た探査機と月の位置データを取得　（<https://wgc2.jpl.nasa.gov:8443/webgeocalc/#StateVector>から）

Application, table

Description automatically generated

時間範囲や時間ステップはシミュレーションに合わせる（/Users/yasudarikuto/research/icymoon\_raytracing/tools/result\_for\_yasudaetal2022/occultation\_flyby\_list.csv　が参考になる）

1. /Users/yasudarikuto/research/icymoon\_raytracing/tools/result\_for\_yasudaetal2022/ephemeris\_for\_coordinate\_transformation/moon　or spacecraft に保存
2. ファイル名を/Users/yasudarikuto/research/icymoon\_raytracing/tools/result\_for\_yasudaetal2022/occultation\_flyby\_list.csv　に記入
3. Table

   Description automatically generated
4. /Users/yasudarikuto/research/icymoon\_raytracing/tools/python\_for\_yasudaetal2022/tangential\_position\_revised.py に天体名・探査機名・フライバイ回数・該当フライバイ時の掩蔽開始 & 終了時刻・電波源を記入して実行（表のヘッダーとフッターの行数が異なるとエラーが出るのでread\_csv(

spacecraft\_csv\_path, header=15, skipfooter=4)　の値を変更して再度実行する）

1. うまくいけば/Users/yasudarikuto/research/icymoon\_raytracing/tools/result\_for\_yasudaetal2022/calculated\_expres\_detectable\_radio\_data\_of\_each\_flyby/calculated\_all\_galileo\_callisto\_9\_tangential\_point\_revised.txtに掩蔽観測の観測地点を含む情報が出力される。以下注意点
   1. ここでいう観測地点というのは、探査機と電波源を直線で結んだ時の接点である
   2. 衛星座標とはIAU\_“衛星名”。SPICE grocalcにもあるはず
   3. 出力結果には観測地点の緯度経度を含む以下の情報が含まれている

[hour, min, frequency(MHz), 電波源データの磁力線(根本)の経度 orイオの場合は(-1000), 電波源の南北, tangential pointのIAUmoon経度, tangential pointnoのIAUmoon緯度, tangential pointからの探査機方向（tangential から90度回転）のIAUmoon経度, tangential pointからの探査機方向（tangential から90度回転）のIAUmoon緯度,座標変換した時のy(tangential pointからの高さ方向の距離)] west longitudで経度は出力されている

* 1. この結果には電波の屈折によって受信できなくなると予想される電波も含まれていることを留意すること

1. コマンド上での出力として該当事項での最大・最小・平均緯度・経度が出力される
2. さafTable

   Description automatically generatedfff
3. レーダー用（反射効果あり）or掩蔽用（光消失）

・レーダー用のとき

src/rtc/src/tracer.inl のif checkReflection の部分を使い、

src/rtc/src/ray.cpp の Check state の if checkReflectionをコメントアウト

・掩蔽用のとき

その逆

1. エウロパでの地面認定

src/rtc/src/cosmos.cppのgetHightの関数をチェック

エウロパ半径1560800mになっていればよし

天体を変更するときにはその天体の半径をmで入力

1. エウロパでのプラズマ密度

・src/rtc/testing/mkray.shのplasmadensityをeuropa plume or europa nonplume

・src/rtc/testing/mkray.shのplanetをbenchmark

・src/rtc/model/test\_plasma\_model.cppをplasma density のeuropa plume or europa nonplumeを書き換えることで別天体のプラズマ密度

1. データ出力の際は注意

プロットするpythonのコードの書き換え・

src/rtc /tools/map\_model で新しいプラズマプロットするとき..

make -f makefile.gcc

ghp\_1Qz42SVkpyIF17vUDS0oyiN5i1hogg3AutZq

-2000〜500 km の高度

2700〜10000km

衛星・探査機の位置情報の取得方法

ガリレオ探査機のデータを取得するときはマニュアルで取得する必要があるので注意

＃カリスト30フライバイに関しては計算範囲を狭めるため12:15分くらいまで有効な電波の伝搬経路長を採用！　occultation\_flyby\_listを作成！！

＃カリスト9フライバイに関しては計算範囲を狭めるため/計算簡略化のため、12:50分くらいからの有効な電波の電波の伝搬経路長を採用！　occultation\_flyby\_listを作成！！

❹Latmos電離圏モデル（Lathys）の結果を用いる方法

1. Latyhsの密度データ（イオン種ごと）を取得する。  
   A screenshot of a computer

   Description automatically generated
2. icymoon\_raytracing/tools/python\_for\_3D\_occultation/3d\_ionospheric\_model/check\_model\_code/remake\_cdf\_file.pyを使って、LathysのデータをRay tracing コードに代入可能なデータに変形する。

変更する事項

* + 1. 各種イオン密度の足し合わせ
    2. イオン密度の単位変更（/cc → /m3）
    3. 位置データの単位変更（m → km）

使用方法

1. src \_filesに足し合わせたい密度データのパスを全て記載する

2. 実行する

3. 出力データがsummed\_density.ncという名前のデータで出力される  
A black background with text

Description automatically generated

1. Ray tracingコードに密度データをインプットし、正しく補間されているかを確認する。

方法

1. summed\_density.ncをmodel/3d\_modelの中に移す（Ex: icymoon\_raytracing/src\_venv/rtc\_cost\_reduction/model/3d\_model）

2. 使用する密度データのパスを/model/Lathys\_europa\_3d.cppのconst std::string& filenameに記入A black background with text

Description automatically generated

3. modelの上位ディレクトリ（例: rtc\_cost\_reduction）でmakeをする。これにより密度モデルがコンパイルされる

4. /tools/map\_modelで make -f makefile.gccでコンパイル。これにより密度出力用のコードがコンパイルされる

5. /tools/map\_modelで ./mapで密度出力用のコードを実行する。

6. icymoon\_raytracing/tools/python\_for\_3D\_occultation/3d\_ionospheric\_model/check\_raytracing\_model/europa\_ionosphere\_plot2.pyを用いて密度データをプロットする。出力データの密度の単位（/m3）・距離の単位（km）があっているか。密度データの座標系（x,y,zの向き）が反転していないか、を確認する。

1. testing/mkray3.shのようなbashファイルを書き換える  
   1. –plot rayの下の行に – parallelと記入する（並列計算を行えるようになる）  
   2. --round-div ${ROUND\_DIV}と記入する。ROUND\_DIVに値を代入する。（これで並列計算するレイの数を指定する。もし– parallelを記入しないとfor文で処理される）  
    A screen shot of a computer program

   Description automatically generatedA screenshot of a computer screen

   Description automatically generated  
   3. testing/main.cpp  
   353 raytrace rtrc(env, 0.0, 100e3\*round)  
   371 raytrace \*r = new raytrace (env, 0.0, 100e3\*round)  
   のroundの係数を並列したいレイのz座標間隔(m)で指定する。  
   ※ SZ = 100 (km), roundの係数500e3 (m), ROUND\_DIV =3の場合、計算されるレイの初期座標は、z = 100, 600, 1100 (km)となる  
     
   A screen shot of a computer program

   Description automatically generated