How GipsyX turns GNSS measurements into position in a frame

Paul Ries

Jet Propulsion Laboratory, California Institute of Technology

Outline

- GipsyX overview
- Finding a position in GipsyX
- Single station time series in GipsyX
- Orbits and clocks: linking PPP to a reference frame

Introduction to GipsyX

GipsyX is a *suite* of software capable of many applications:

- Single station static point positioning with ambiguity resolution
- Network solutions of GNSS stations and GNSS satellites
- Low Earth Orbit precise orbit determination with ambiguity resolution
- Kinematic point positioning with ambiguity resolution of geological to aeronautical speeds

Introduction to GipsyX

- GipsyX Paper

GipsyX/RTGx, a new tool set for space geodetic operations and research

Willy Bertiger [a], Yoaz Bar-Sever [a], Angie Dorsey [a], Bruce Haines [a], Nate Harvey [a], Dan Hemberger [a], Michael Heflin [a], Wenwen Lu [a], Mark Miller [a], Angelyn W. Moore [a], Dave Murphy [a], Paul Ries [a], Larry Romans [a], Aurore Sibois [a], Ant Sibthorpe [a], Bela Szilagyi [a], Michele Vallisneri [a], Pascal Willis [b, c]

Show more

+ Add to Mendeley Share Cite

https://doi.org/10.1016/j.asr.2020.04.015

GipsyX positioning accuracy

GipsyX, gd2e.py, Daily IGb08 Coordinates Reproduction Using JPL Finals June-Nov. 2008, 102 Stations

Bias-Fixing/Trop Model	East(mm)	North(mm)	Vertical(mm)
Unresolved/VMF1	2.9	2.0	6.0
Resolved/VMF1	1.9	2.0	6.0

GipsyX system requirements

- Linux
  - Redhat Enterprise Linux 7, 8, 9
  - Ubuntu 20.04, 22.04
- Windows - Not officially supported
  - Some reported success with VMs and Windows Subsystem For Linux
- Mac – Supported previously
  - Hope to eventually support again

GipsyX compiled programs

- rtgx – main engine for forming residuals, updating parameters, editing data, resolving ambiguities, etc
- gde – GNSS data editor – tool for pre-processing GNSS measurements
- dataRecord* – tools for examining RINEX, GipsyX dataRecord format

GipsyX python scripts

- gd2e.py – main engine for performing PPP
- igs2GipsyX.py – script for using/fetching IGS format products
- sta*.py – series of tools for long term station time series
- net*.py – series of tools for building up reference frames

A First PPP with GipsyX

Command line invocation

- Fetch and PPP a station in two simple commands:

```
$ rinexFetch.py -outDir . -fetch day -rinex only2 -t1 2016-01-13 -
stns amc2 -srv sopac
$ gd2e.py -rnxFile y2016/d013/amc20130.16d.Z
```

What does gd2e.py do?

- Convert rinex headers into staDb (GipsyX meta format)
- Invoke our data editor, gde
  - converts rinex to dataRecord (GipsyX data format)
  - Decimate phase and smooth range input
  - Flags breaks
  - Applies CA-P correction for GPS
- Determine data duration and fetch appropriate orbit and clock inputs
- Creates receiver antenna calibration from ANTEX
- Copy and set up “tree” for rtgx
  - Disable ocean loading for RINEX input

What does gd2e.py do?

- Run rtgx
  - Auto-detects available data, picks “best” for each constellation
  - Sets up appropriate biases, if needed
  - Filter-smoother iteration through data
  - Iterative outlier removal
  - Formulate and apply integer ambiguity constraints
- Summarize output
  - Calculate statistics on residual
  - Output position

First PPP outputs

```
$ ls
AMC2.gde.debug.tree        editData.log                prefitResiduals.out
AMC2.gde.stats             filter.tdp                  rinexStaDb
AmbResSummary              final.pos                   rtgx_ppp_0.tree.err0_0
GNSSinitValues             finalResiduals.out          rtgx_ppp_0.tree.log0_0
Summary                    gde.tree                    runAgain
Trees                      gnssList.txt                smooth0_0.tdp
allStations.xyz            iterRtgx                    smoothFinal.cov
ambres.stats               postfitResiduals.out        smoothFinal.tdp
constraints.txt            preamb_final.pos            stations.txt
dataRecordFile.gz          preamb_finalResiduals.out   treesUsed
debug.tree                 preamb_smooth0_0.tdp        y2016
editData.err               preamb_smoothFinal.cov
```

First PPP summary file

```
$ cat Summary
---   Residual Summary:
-----------------------------------------------------------------
---   included residuals  :      4943 (  98.5% )
---   deleted residuals   :        73 (   1.5% )
---      DataType           Status        RMS (m)         Max (m)         Min (m)         number (%)
---   IonoFreeC_1W_2W   included      6.258517e-01    2.487303e+00   -2.410555e+00      2490 (  99.3% )
---   IonoFreeC_1W_2W    deleted      3.233628e+00    5.366257e+00   -4.804870e+00        18 (   0.7% )
---
---   IonoFreeL_1W_2W   included      7.064332e-03    2.429315e-02   -2.431338e-02      2453 (  97.8% )
---   IonoFreeL_1W_2W    deleted      4.684365e-02    5.624284e-02   -7.800784e-02        55 (   2.2% )
-----------------------------------------------------------------

                    PPP Solution: XYZ                                DeltaXYZ(Sol-Nom)              DeltaENV
(meters)
AMC2 -1248596.356423737 -4819428.20094368 3976505.934802891 -1.044E-01  8.306E-02  -9.920E-02  -1.219E-01  -
4.333E-02  -1.044E-01
```

Summary file summary

- First section is summary of residuals
  - Number of data points
  - Percent deleted
  - RMS
- Second section is position
  - Absolute position
  - difference to nominal

Summary file (con'd)

- Typical data processing is IonoFree phase and range combinations
  - IonoFreeC_1W_2W = GPS C1W/C2W combination
- By default, input data is converted from RINEX2 to RINEX3 types at editor, and processed as RINEX3
- This example is GPS-only, but multi-GNSS is supported

AmbRes Summary file

- GipsyX will resolve integer ambiguities when using products from JPL
- Biggest effect on east repeatability

```
$ cat AmbResSummary
Ambiguity Info iter  0 WLCUM, NLCUM < 10 centi-cycles:  64.4 %  93.1 % total fixed:  87.8 %
```

Other outputs

- Residual file – contains each measurement

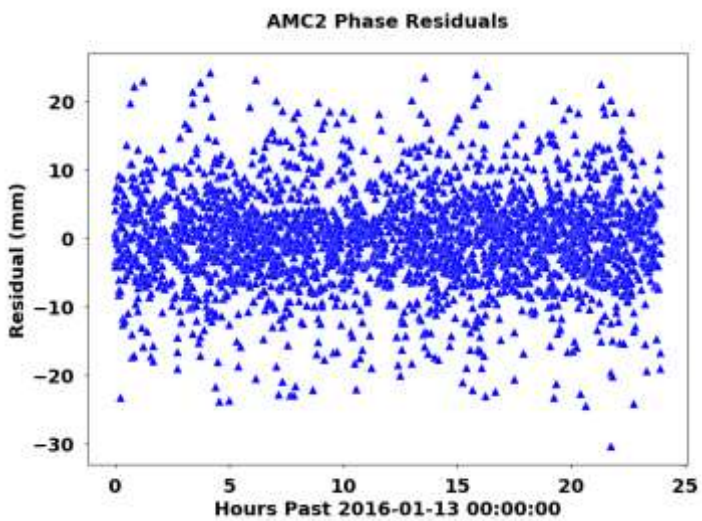
```
$ head -n 2 finalResiduals.out
505915200 {AMC2(1)-GPS71(1)} IonoFreeL_1W_2W 0.00154695692581586 50.7961 62.3629 81.2605 90.3774
505915200 {AMC2(1)-GPS71(1)} IonoFreeC_1W_2W -0.106324715376878 50.7961 62.3629 81.2605 90.3774
```

- Time dependent parameter (tdp) – input/output parameters

```
$ grep Station smoothFinal.tdp | head -n 4
505915200  1.000000000000000e-01  1.841325263352175e-02  2.544943380791311e-03 .Station.AMC2.Trop.WetZ
505915200  0.000000000000000e+00  1.314398726388410e-04  4.263573945510995e-04 .Station.AMC2.Trop.GradNorth
505915200  0.000000000000000e+00  9.990711027752415e-05  3.377871479859210e-04 .Station.AMC2.Trop.GradEast
505915200  0.000000000000000e+00  3.734221901330094e-01  2.693546174188096e-02 .Station.AMC2.Clk.Bias
```

Plotting/column tools

```
$ grep IonoFreeL_1W_2W finalResiduals.out | grep -v DELETED | cm h1 'c4*1e3' | \
matp -nk -t "AMC2 Phase Residuals" -xl "Hours Past 2016-01-13 00:00:00" -yl "Residual (mm)"
```



gd2e.py command line options

- To use custom staDb/editing
  - -drEditedFile – specify edited dr file (rnxEditGde.py)
  - -recList – list of receivers
  - -staDb – specify a custom staDb
- -GNSSproducts – specify non-default orbit and clock product
- -HighRate products – use _hr clock file, if available
- -prodTypeGNSS – use e.g. non-fiducial products

gd2e.py command line options (con'd)

- -gdCov – output gdcov
- -tdpInput – tdp input file (e.g. nominal trops, nominal position for kinematic users)
- -orbClkDir – use pre-fetched orbits and clocks
- -treeSequenceDir – use non-default tree

The Rtgx Tree

```
# Solve for constant position
GRN_STATION_CLK_WHITE ==
    State
      Pos
        ConstantAdj 10.0
    Clk
      Model On
      Bias  0.0
        StochasticAdj 3.0e8 3.0e8 $GLOBAL_DATA_RATE WHITENOISE
    Trop
      Model   On
      Mapping GMF
      WetZ    0.1
        StochasticAdj 0.5 5e-5 $GLOBAL_DATA_RATE RANDOMWALK
      GradEast 0.0
        StochasticAdj 1.0 5e-6 $GLOBAL_DATA_RATE RANDOMWALK
      GradNorth 0.0
        StochasticAdj 1.0 5e-6 $GLOBAL_DATA_RATE RANDOMWALK
```

Customizing Rtgx tree

```
# Solve for constant position
GRN_STATION_CLK_WHITE ==
    State
      Pos
        ConstantAdj 10.0
    Clk
      Model On
      Bias  0.0
        StochasticAdj 3.0e8 3.0e8 $GLOBAL_DATA_RATE WHITENOISE
    Trop
      Model   On
      Mapping GMF
      WetZ    0.1
        StochasticAdj 0.5 5e-5 $GLOBAL_DATA_RATE RANDOMWALK
      GradEast 0.0
      GradNorth 0.0
```

Futher reading on PPP

- "Precise Point Positioning, PPP Using gd2e.py"
- GipsyX has documentation and now video lectures
  - Examples of custom trees (kinematic PPP, second order iono correction)
  - Multi-GNSS PPP
  - More details about the files output from a gd2e.py run
  - Running gd2e.py with edited data

PPP gdCov file

- gdCov files from gd2e.py contain daily position and formal errors, created by -gdCov flag

```
$ cat smoothFinal.gdcov
3 PARAMETERS
1 AMC2.STA.X 505958250 -1.248596356423737e+06 9.517991355951055e-04
2 AMC2.STA.Y 505958250 -4.819428200943680e+06 2.365551608773745e-03
3 AMC2.STA.Z 505958250 3.976505934802891e+06 1.892607495220957e-03
2 1 6.109010953392991e-01
3 1 -5.401214611904720e-01
3 2 -8.331788644412753e-01
```

Creating a series of gdCov file

- Run gd2e.py on daily rinex, save gdcov from each day with appropriate date
  - Simple shell script
  - More advanced python script
  - Existing wrapper tools “NetworkProcessor”

Combine individual station gdCov into gdCat

- gdCat = concatenated gdcov

```
$ ls gdcov/ | head -n 5
2011-01-01.gdcov
2011-01-02.gdcov
2011-01-03.gdcov
2011-01-04.gdcov
2011-01-05.gdcov
$ netSplit.py -i gdcov/*.gdcov
$ head USC1.gdcat
3 PARAMETERS INCLUDE
1 USC1.STA.X 347155200 -2.507565004852250e+06 2.010624088590890e-03
2 USC1.STA.Y 347155200 -4.659953198526880e+06 2.969383268875170e-03
3 USC1.STA.Z 347155200 3.548661082692570e+06 2.258541495109910e-03
2 1 8.681892448460430e-01
3 1 -7.841674268007058e-01
3 2 -8.570286717268020e-01
3 PARAMETERS INCLUDE
1 USC1.STA.X 347241600 -2.507565004607680e+06 2.054574965724030e-03
2 USC1.STA.Y 347241600 -4.659953196740000e+06 3.110942153569990e-03
```

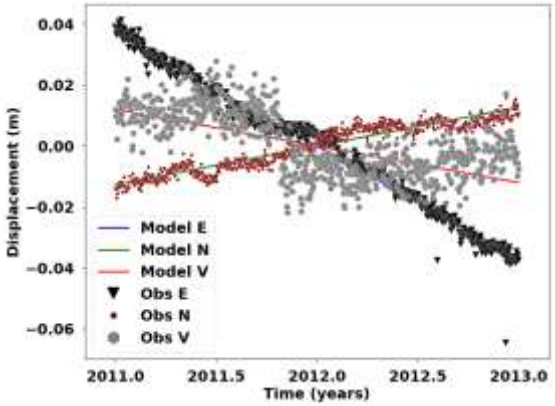
Fit gdcat to solution gdCov

```
$ staFit.py -i USC1.gdcat -o USC1.gdcov -v 2012-01-01
 CHI^2      =  1.0307e+04
 DOF        =        2178
 CHI^2/DOF  =        4.73
$ head USC1.gdcov
6 PARAMETERS
1 USC1.STA.X 378691200 -2.507565031012119e+06 7.691017578896084e-05
2 USC1.STA.Y 378691200 -4.659953164444597e+06 1.155649508259956e-04
3 USC1.STA.Z 378691200 3.548661087286912e+06 8.472188999735669e-05
4 USC1.VEL.X 378691200 -2.431014410018155e+01 1.358553908766281e-01
5 USC1.VEL.Y 378691200 3.245007751444996e+01 2.046768417723794e-01
6 USC1.VEL.Z 378691200 3.837523665832726e+00 1.488291961884231e-01
... (covariance stuff) ...
```

Generate time series with staSeries.py

- staSeries – compute E N V model, obs, and resid

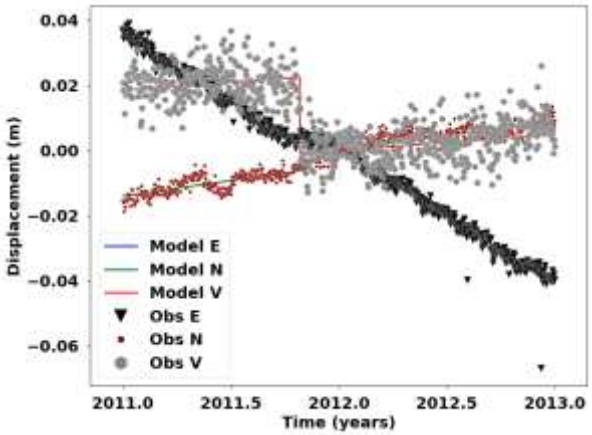
```
$ staSeries.py -r USC1.gdcov -i USC1.gdcat
$ ls
gdcov                    USC1.gdcat       USC1.gdcov  USC1.resid   USC1.sum
singleStationGdcov.tgz   USC1.gdcat.orig  USC1.model  USC1.series
$ matp -xy USC1.model:l:0:1,2,3:"Model E":"Model N":"Model V" \
           USC1.series:p:0:1,2,3:"Obs E":"Obs N":"Obs V" \
           -xl "Time (years)" -yl "Displacement (m)" -lx 2010.85 2013.15
```



Break detection and outlier removal

staBreak.py – estimate breaks from a series

staEdit.py – remove outliers from a gdcat



More information on time series

- Step-by-step guide “Single Station PPP Time Series” portion of documentation
- Additional information about seasonal terms and other options

Linear PPP solutions to reference frame

- netBuild.py -i sta1.gdcov sta2.gcov … -o frame.gdcov

How to build a reference frame

- PPP each station individually across a long period of time, outputting one gdCov / station / day
- Fit a linear time series for each station independently
- Combine desired linear solutions into a reference frame

What global reference frame is PPP in?

- Orbits and clocks generally
- Frame approaches
  - Fixed-site
  - Free
  - No-net-rotation
  - No-net-rotation, translation, and scale
- Using xfiles with GipsyX

What is PPP?

- In PPP, GNSS satellite orbits and GNSS satellite clocks are fixed
- Other global parameters, such as Earth Rotation Parameters are also fixed
- Only free parameters are at target site
  - Position
  - Clock
  - Trop

What frame is PPP output?

- PPP frame is realized through the input orbit and clock products
- Earth Rotation Parameters can have an effect in low latency cases (i.e. starting from predicted vs observed)

Frame fixing in orbits and clocks

- Orbit and clock products from JPL and other IGS ACs are typically tied to an IGS realization of the ITRF (e.g. IGb14, a realization of ITRF2014)
- ITRF consists of linear station position models and post-seismic correction (as of ITRF2014)

Network solutions

- Orbits and clocks are products of network solutions
  - Global network of ground stations
  - Free ground network clocks (except for reference)
  - Free satellite orbits and clocks
  - Free ERP (except for UT1-UTC for GNSS)
- Many options for ground network *positions*

Individual fixed stations

- Fix some or all ground stations to reference frame solution
  - Nx3 constraints, where N is number of fixed sites
  - JPL Rapids = 40 fixed sites, 40 free sites, 120 constraints
- Use cases
  - JPL Rapids (also other AC contributions to IGS Rapids)
  - JPL Ultras

Individual fixed stations

- Advantages
  - Computationally inexpensive
  - Straightforward implementation
  - PPP will accurately reproduce frame
- Disadvantages
  - Any frame errors will be absorbed by orbit and by subsequent PPP

Free network solution

- No constraints at all
- Use cases
  - JPL Finals initial solution

Free network solution

- Advantages
  - Frame errors are irrelevant
  - Can detect relative errors in reference frame (used in JPL Finals to detect stations to remove from constraints)
- Disadvantages
  - Not in any reference frame (frame of date)
  - Large day-to-day rotations (meters)

No-net-rotation (NNR) solution

- 3 constraints (RX,RY,RZ = 0)
- Applies average constraint of all reference stations
- Use cases
  - JPL Final submission to IGS
  - Other AC Final submissions to IGS
  - AC IGS reprocessing campaign solutions

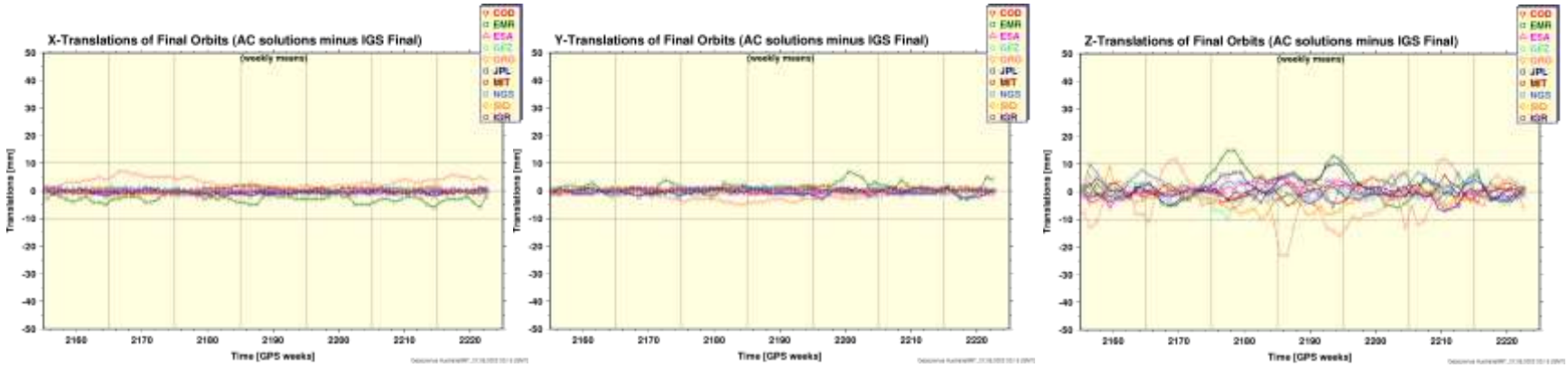
No-net-rotation (NNR) solution

- Advantages
  - Reference frame rotational errors typically small
  - Unconstrained translations have real signals (e.g. geocenter)
- Disadvantages
  - Day-to-day translations contain “noise” (~cm level)

No-net-rotation, translation, or scale (NNRTS) solution

- 7 constraints (RX, RY, RZ, TX, TY, TZ, S=0)
- Use cases
  - JPL Finals (GipsyX default)
  - IGS finals (via combining NNR inputs)

https://igs.org/acc/gps-only/#final

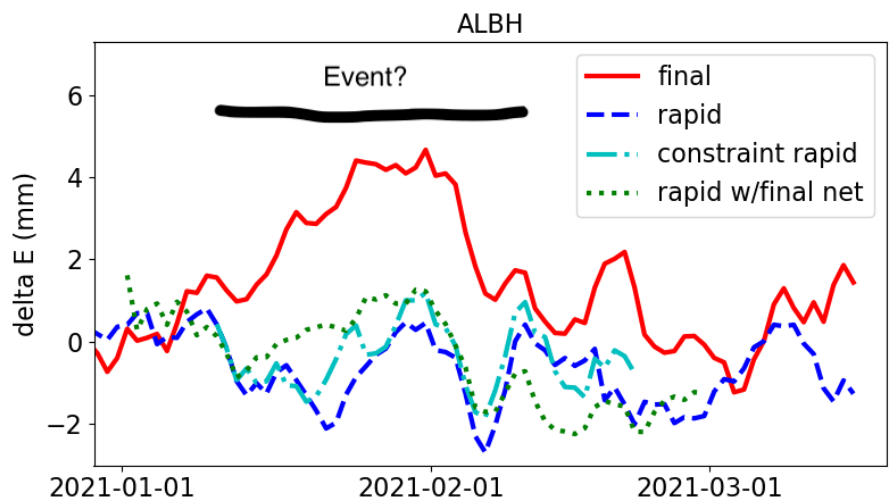


No-net-rotation, translation, or scale (NNRTS) solution

- Advantages
  - Relatively insensitive to single/few station errors
  - Millimeter-level frame reproducibility
- Disadvantages
  - Remove real translational signals (e.g. geocenter)

Case study comparison

- Ries et al 2021 "Recovering Seismic Signals with Different Reference Frame Realization Methods" – AGU 2021
- Subtle event only detected by fortuitous stations + constraints



GipsyX xfiles

- Best fit Helmert (e.g. T, R,S) parameters between reference gdcov and solution gdcov

```
$ zcat 2021-01-01.x.gz | head -n 15
 CHI^2       =  1.4894e+02
 DOF         =         113
 CHI^2/DOF   =        1.32
 RX =   4.663e-10  +-   4.972e-11  rad
 RY =   1.095e-09  +-   4.792e-11  rad
 RZ =  -4.490e-07  +-   5.390e-11  rad
 TX =  -1.966e-02  +-   5.060e-04  m
 TY =  -5.147e-03  +-   5.611e-04  m
 TZ =  -1.521e-02  +-   4.359e-04  m
 S  =   1.635e-10  +-   1.481e-10  parts

 RES NAME          N          E          V         SN         SE         SV
 POS AREQ     -1.204     -4.705      6.799      2.089      2.408      7.647 mm
 POS ARTU     -2.545      1.773    -10.151      2.680      1.973      7.362 mm
 POS AUCK      1.636     -2.298      2.266      2.763      2.608      8.086 mm
```

GipsyX xfile use cases

- Use NNR or Free orbits to do PPP, then transform daily station gdcov to frame using JPL Product xfile (netApply.py)
  - Potentially avoid re-PPP for each new reference frame
- Generate your own xfile between reference frame of your choice and daily combined PPP solution (netXfile.py)
  - Can then put any station into arbitrary reference frame

More information about GipsyX

- GipsyX is developed and supported by a large team at JPL
- Support available through user forum
- Software information
  - https://gipsyx.jpl.nasa.gov/index.php?page=software

Jet Propulsion Laboratory
California Institute of Technology

jpl.nasa.gov