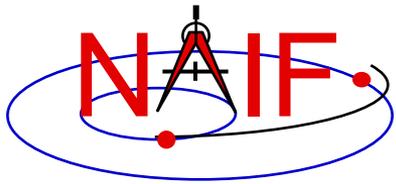




Navigation and Ancillary Information Facility

Preparing for Programming Using the SPICE Toolkits

June 2019



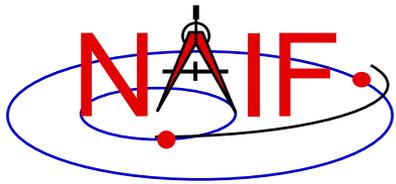
Setting Path to Toolkit Executables (1)

Navigation and Ancillary Information Facility

Recommended for all Toolkits

- **Unix (OS X, Linux, BSD, execute the command `echo $SHELL` to determine your shell name)**
 - **csh, tcsh: Use the `set` command to add the location of toolkit executables to your path.**
 - » `set path = ($path /my_directory/toolkit/exe)`
 - » `set path = ($path /my_directory/cspice/exe)`
 - » `set path = ($path /my_directory/icy/exe)`
 - » `set path = ($path /my_directory/mice/exe)`
 - **sh, bash, zsh, dash, ksh: Assign the `$PATH` environment variable.**
 - » `PATH=$PATH:/my_directory/toolkit/exe`
 - » `PATH=$PATH:/my_directory/cspice/exe`
 - » `PATH=$PATH:/my_directory/icy/exe`
 - » `PATH=$PATH:/my_directory/mice/exe`

Replace `my_directory` with the path in which you installed the toolkit on your computer.



Setting Path to Toolkit Executables (2)

Navigation and Ancillary Information Facility

Recommended for all Toolkits

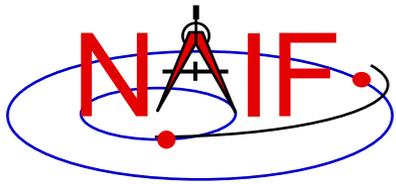
- **Windows**

- **DOS shell: Use the set command to add the location of toolkit executables to your path. Use setx for a persistent setting.**

- » `set PATH=drive:\my_directory\toolkit\exe;%PATH%`
- » `set PATH=drive:\my_directory\cspice\exe;%PATH%`
- » `set PATH=drive:\my_directory\icy\exe;%PATH%`
- » `set PATH=drive:\my_directory\mice\exe;%PATH%`

- **Or edit the environment variable PATH from the Advanced pane on the System Control Panel (Control Panel->System->Advanced).**

Replace `drive:\my_directory` with the path in which you installed the toolkit on your computer.



Unix: Build a SPICE Executable

Navigation and Ancillary Information Facility

- Assume your Toolkit distribution is installed at:

`/naif/cspice/` for CSPICE (C toolkits)

`/naif/toolkit/` for SPICE (Fortran toolkits)

- Compile and link an application—let's pretend it's named *program*—against the CSPICE or SPICELIB library.

– For C:

```
$ gcc program.c -I/naif/cspice/include /naif/cspice/lib/cspice.a -lm
```

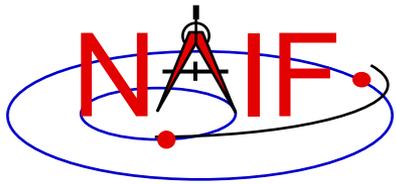
– For FORTRAN:

```
$ gfortran program.f /naif/toolkit/spicelib.a
```

- The default SPICE library names do not conform to the UNIX convention `libname.a`. So you cannot use the conventional library path/name options `-L` and `-l`, e.g.

```
$ gcc ... -L/path_to_libs/ -lname
```

unless you rename the SPICE library.



Windows: C compiler settings

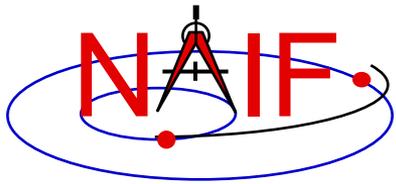
Navigation and Ancillary Information Facility

- **The standard installation of Microsoft Visual Studio may not update environment variables needed to use the C compiler (cl) from the standard DOS shell. This depends on your version of the Microsoft development environment.**
 - If programming in an XP 32-bit environment, you can set the environment variables by executing from a DOS shell one of the “vars32” batch scripts supplied with Microsoft compilers:
 - » `vars32.bat`, `vcvars32.bat`, `vsvars32.bat`
 - Recent versions of Visual Studio include scripts to spawn a DOS shell with the needed environment. The scripts exist under *Visual Studio “version”* found in the *Programs* menu:

Programs -> Visual Studio “version”

The scripts’ names for a 64bit (x64) environment or a 32bit (x86) are:

- » ***VSversion x64 Native Tools Command Prompt***
 - Example, *VS2015 x64 Native Tools Command Prompt*
- » ***VSversion x86 Native Tools Command Prompt***
 - Example, *VS2015 x86 Native Tools Command Prompt*



Windows: ifort compiler settings

Navigation and Ancillary Information Facility

- **The standard installation of Intel ifort may not update environment variables needed to use the Fortran compiler (ifort) from the standard DOS shell.**
 - Intel provides batch scripts to spawn DOS shells properly configured for 32-bit or 64-bit Fortran development. Find the scripts by navigating to the menu

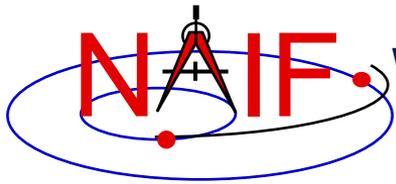
*Programs -> Intel Software Development Tools -> Intel Visual Fortran
Compiler (version)*

The script for a 32-bit ifort environment is:

`Fortran Build Environment for applications running on IA-32`

The script for a 64-bit ifort environment is:

`Fortran Build Environment for applications running on Intel 64`



Windows: Build a SPICE Executable

Navigation and Ancillary Information Facility

- Assume the SPICE distribution is installed at:

`C:\naif\cspice\` for C toolkits

`C:\naif\toolkit\` for Fortran toolkits

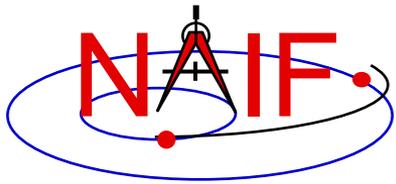
- Compile and link an application, say *program*, against the CSPICE or SPICELIB library.

- For C toolkits:

```
> cl program.c -IC:\naif\cspice\include C:\naif\cspice\lib\cspice.lib
```

- For FORTRAN toolkits:

```
> ifort program.f C:\naif\toolkit\lib\SPICELIB.LIB
```



Icy: Register the Icy DLM to IDL (1)

Navigation and Ancillary Information Facility

Required for “Icy”

- **Unix and Windows**

- Use the IDL register command:

```
IDL> dlm_register, '_path_to_directory_containing_icy.dlm_'
```

e.g.

```
IDL > dlm_register, '/naif/icy/lib/icy.dlm'
```

- Or, copy `icy.dlm` and `icy.so` (or `icy.dll`) to IDL's binary directory located at *{The IDL install directory}/bin/bin.user_architecture*, e.g.

- » For Unix, X86 architecture

```
cp icy.dlm icy.so /Applications/exelis/idl/bin/bin.darwin.x86_64/
```

- » For Windows, X86 architecture

```
cp icy.dlm icy.dll C:\Program Files\Exelis\idl83\bin\bin.x86_64\
```

continued on next page



Icy: Register the Icy DLM to IDL (2)

Navigation and Ancillary Information Facility

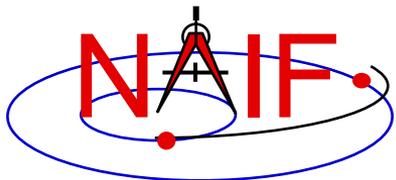
- **Unix specific:**

- Start the IDL application from a shell in the directory containing both `icy.dlm` and `icy.so`.
- Append the path to your `icy.dlm` to the `IDL_DLM_PATH` environment variable to include the directory containing `icy.dlm` and `icy.so`, e.g.:

```
setenv IDL_DLM_PATH "<IDL_DEFAULT>:_path_to_directory_containing_icy.dlm_"
```

Warning: do not invoke IDL from the Icy source directory, *icy/src/icy*, nor register that directory, and do not append that directory to `IDL_DLM_PATH`. This directory contains an “`icy.dlm`” but not “`icy.so`.”

continued on next page



Icy: Register the Icy DLM to IDL (3)

Navigation and Ancillary Information Facility

- **Windows specific:**
 - Set environment variable `IDL_DLM_PATH` from the *Advanced* pane of the *System Control Panel*.
- **Once registered as specified on earlier pages, confirm IDL recognizes and can access Icy.**
 - Using the help command:

```
IDL> help, 'icy', /DL
**ICY - IDL/CSPIICE interface from JPL/NAIF (not loaded)
```

» Appearance of the words “not loaded” might suggest something is wrong, but this is expected state until you execute an Icy command.

- **Execute a trivial Icy command:**

```
IDL> print, cspice_icy('version')
% Loaded DLM: ICY.
Icy 1.4.20 25-DEC-2008 (EDW)
```

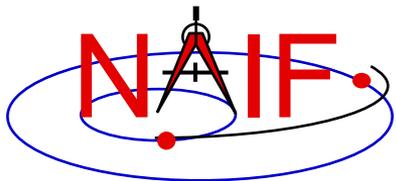


Icy: Using the IDL IDE

Navigation and Ancillary Information Facility

Recommended for “Icy”

- Use the IDL IDE’s preferences panel to set the current working directory to the location where you will be developing your code.
- Optional: Place your `d1m_register` command in a start up script. Specify the script using the IDL IDE’s preferences panel.



Mice

Navigation and Ancillary Information Facility

Required for “Mice”

- **Assume the Mice distribution is installed at `C:\naif\mice\` on Windows, or `/naif/mice/` on Unix/Linux. Use of Mice from Matlab requires the Mice source and library directories exist in the Matlab search path. The easiest way to update the Matlab path is with the “addpath” command.**

- **On Windows:**

```
>> addpath('C:\naif\mice\lib')  
>> addpath('C:\naif\mice\src\mice')
```

- **On Unix/Linux:**

```
>> addpath('/naif/mice/lib')  
>> addpath('/naif/mice/src/mice')
```



Backup

- **Icy programming example**
- **Mice programming example**
- **References**
- **Matlab 2016a MEX Change**



Simple Icy Example

Navigation and Ancillary Information Facility

- **As an example of Icy use with vectorization, calculate and plot the trajectory in the J2000 inertial frame of the Cassini spacecraft from June 20, 2004 to December 1, 2005.**

```
;; Construct a meta kernel, "standard.tm", which will be used to load the needed
;; generic kernels: "naif0009.tls," "de421.bsp," and "pck0009.tpc."

;; Load the generic kernels using the meta kernel, and a Cassini spk.

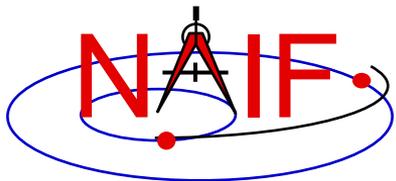
cspice_furnsh, 'standard.tm'
cspice_furnsh, '/kernels/cassini/spk/030201AP_SK_SM546_T45.bsp'

;; Define the number of divisions of the time interval and the time interval.
STEP = 10000
utc = [ 'Jun 20, 2004', 'Dec 1, 2005' ]
cspice_str2et, utc, et
times = dindgen(STEP)*(et[1]-et[0])/STEP + et[0]

cspice_spkpos, 'Cassini', times, 'J2000', 'NONE', 'SATURN BARYCENTER', pos, ltime

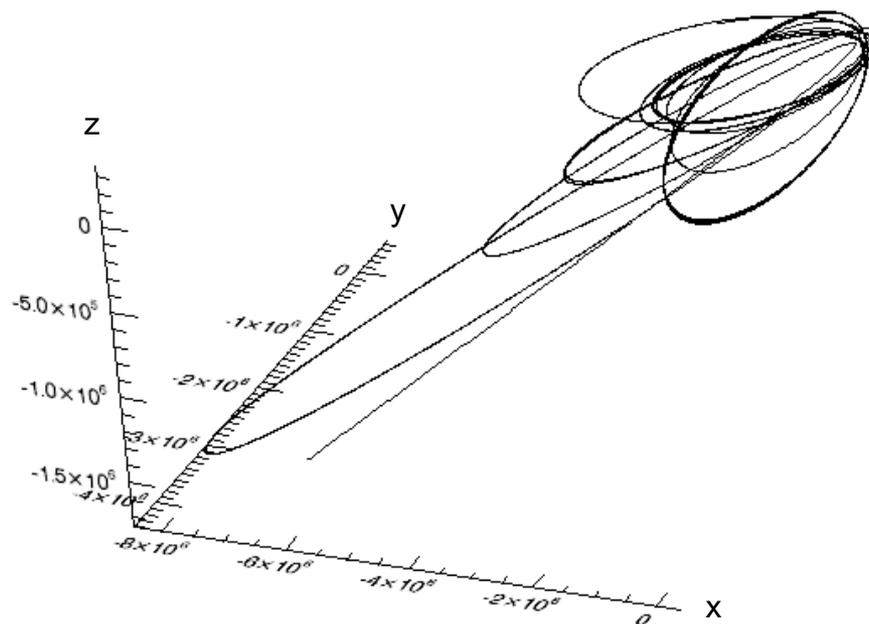
;; Plot the resulting trajectory.
x = pos[0,*]
y = pos[1,*]
z = pos[2,*]
iplot, x, y, z

cspice_kclear
```



Graphic Output

Navigation and Ancillary Information Facility



Trajectory of the Cassini vehicle in the J2000 frame, for June 20, 2004 to Dec 1, 2005



Simple Mice Example

Navigation and Ancillary Information Facility

- **As an example of Mice use with vectorization, calculate and plot the trajectory in the J2000 inertial frame of the Cassini spacecraft from June 20, 2004 to December 1, 2005**

```
% Construct a meta kernel, "standard.tm", which will be used to load the needed
% generic kernels: "naif0009.tls," "de421.bsp," and "pck0009.tpc."

% Load the generic kernels using the meta kernel, and a Cassini spk.

cspice_furnsh( { 'standard.tm', '/kernels/cassini/spk/030201AP_SK_SM546_T45.bsp' } )

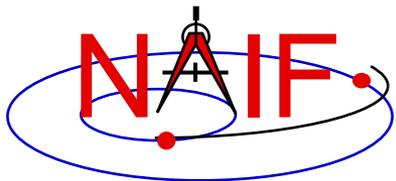
% Define the number of divisions of the time interval and the time interval.
STEP      = 1000;
et        = cspice_str2et( {'Jun 20, 2004', 'Dec 1, 2005'} );
times     = (0:STEP-1) * ( et(2) - et(1) )/STEP + et(1);

[pos, ltime]= cspice_spkpos( 'Cassini', times, 'J2000', 'NONE', 'SATURN BARYCENTER' );

% Plot the resulting trajectory.
x = pos(1,:);
y = pos(2,:);
z = pos(3,:);

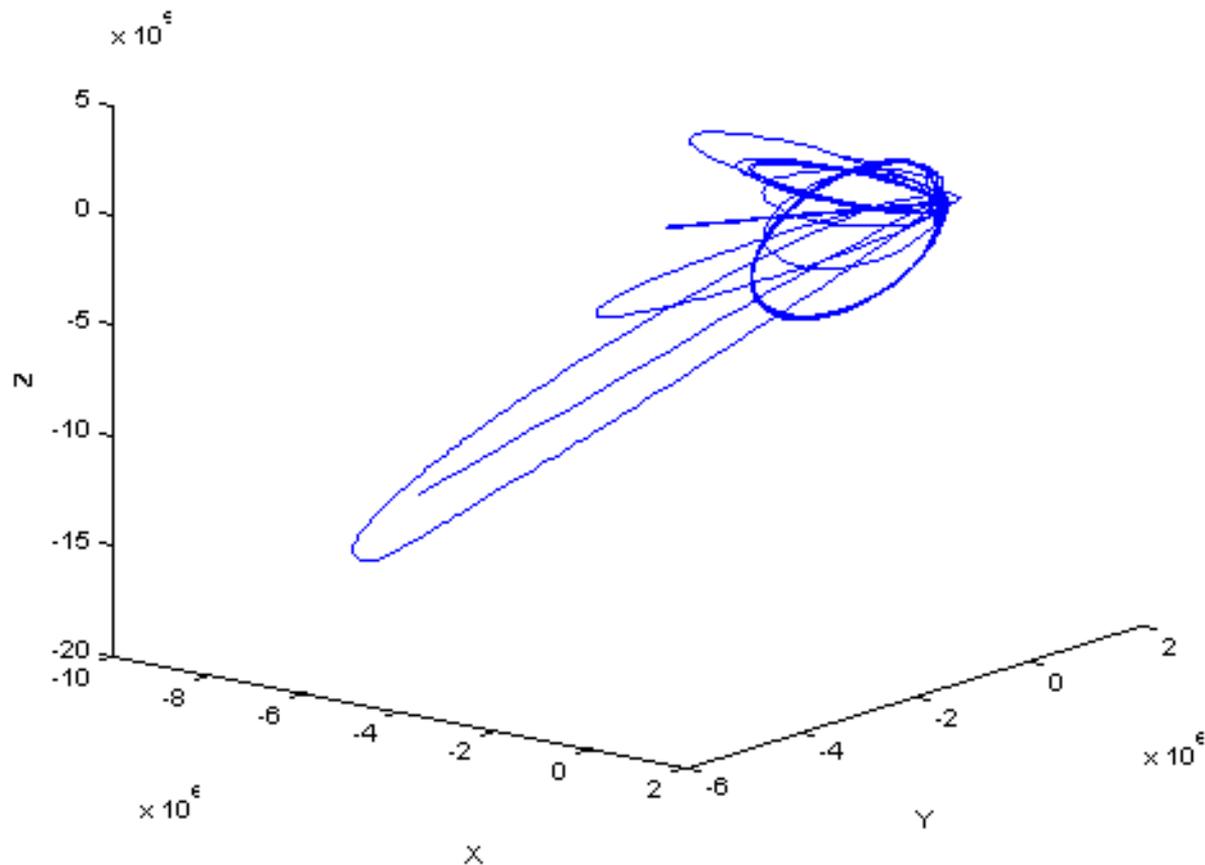
plot3(x,y,z)

cspice_kclear
```

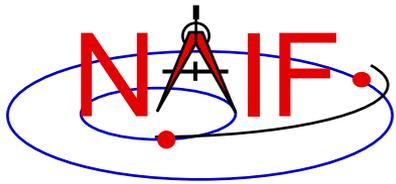


Graphic Output

Navigation and Ancillary Information Facility



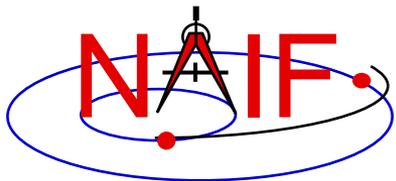
Trajectory of the Cassini vehicle in the J2000 frame, for June 20, 2004 to Dec 1, 2005



References

Navigation and Ancillary Information Facility

- **NAIF documents providing more information concerning SPICE programing:**
 - **“icy.req,” Icy Required Reading**
 - » **icy/doc/icy.req**
 - » **icy/doc/html/req/icy.html**
 - **“mice.req,” Mice Required Reading**
 - » **mice/doc/mice.req**
 - » **mice/doc/html/req/mice.html**
 - **“cspice.req,” CSPICE Required Reading**
 - » **cspice/doc/cspice.req**
 - » **cspice/doc/html/req/cspice.html**
 - **“Introduction to the Family of SPICE Toolkits” tutorial**



Matlab 2016a MEX Change

Navigation and Ancillary Information Facility

- **Mathworks changed the operation of the MEX utility in the Matlab 2016a. Use of the `mkprodct.csh/mkprodct.bat` build script included with SPICE Toolkit N66 or earlier will fail to build Mice against 2016a or later. NAIF will include a modified version of the build scripts in the version N67 Toolkits.**
- **Most Mice users should **NOT** rebuild the Mice Toolkit.**