IDL Interface to CSPICE
"Icy"

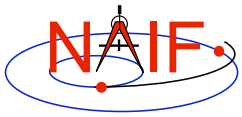
How to Access the CSPICE library Using Interactive Data Language (IDL)©

March 2006

Navigation and Ancillary Information Facility

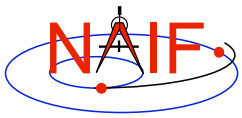
Topics

- How does it work?
- Benefits from Icy use
- Distribution
- Icy Operation
- Vectorization
- Simple Use of Icy Functionality



How Does It Work? (1)

- **IDL includes an intrinsic capability to use external routines.**
  - **Icy functions as an IDL Dynamically Loadable Module. A DLM consists of a shared object library (icy.so/.dll) and a DLM text definition file (icy.dlm).**
    - » **The shared library contains a set of IDL callable C interface routines that wrap a subset of CSPICE wrapper calls.**
    - » **The text definition file lists the routines within the shared library and the format for the routine's call parameters.**



How Does It Work? (2)

When a user invokes a call to a DLM routine:

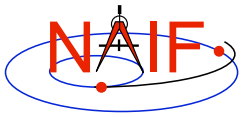
1. IDL calls…

2. the interface routine in the shared object library, linked against…

3. CSPICE, which performs its function and returns the result…

4. to IDL…

… transparent from the user's perspective.

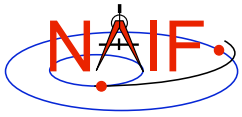


Benefits

- **Benefits from using Icy**
  - **Ease of use: Icy operates as an extension to the IDL language regime.**
  - **Platform independence: the Icy code base requires no modification for ports across supported platforms.* Icy now runs on:**
    - » **OS X (cc/gcc)**
    - » **Solaris in 32 bit mode for cc and gcc compilers****
    - » **Linux (gcc)**
    - » **MS Windows**
    - » **and should run on any platform supporting IDL, CSPICE, and an ANSI C compiler**

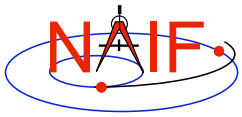
*** CSPICE is widely portable, but not platform independent.**

**** NAIF successfully built an 64 bit Icy using the Solaris cc compiler.**



Icy Distribution

- **NAIF distributes the Icy package as an independent product analogous to SPICELIB and CSPICE.**
- **The package includes:**
  - **The CSPICE source files.**
  - **The Icy interface source code.**
  - **Platform specific build scripts for Icy and CSPICE.**
  - **IDL versions of the SPICE cookbook programs, *states*, *tictoc*, *subpt*, and *simple*.**
  - **An HTML based help system for both Icy and CSPICE, with the Icy help cross-linked to CSPICE.**
  - **The Icy shared library and DLM file. The system is ready for use after installation of the these files.**
    - » The user can recompile the shared library if the appropriate compiler is available.

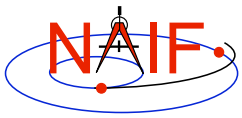


Icy Operation (1)

Navigation and Ancillary Information Facility

- **Icy supports many (335), but not all, CSPICE wrapper functions.**
  - **Icy has some functionality not available in CSPICE.**
    - » **As of Icy 1.1, a subset of calls accept vectorized arguments.* (See Vectorization, pg. 11).**
- **Icy arguments normally match the arguments in the corresponding CSPICE call in type and name, with some exceptions.**
  - **Routines returning vectors do not explicitly return a vector dimension.**
- **CSPICE error messages are returned to Icy in the form usable by the IDL error *catch* handler.**

***Vectorized indicates passing a vector of N items as an argument: a vector of scalars, a vector of vectors, or a vector of matrices, the return value being an N dimensioned version of the non-vectorized output.**



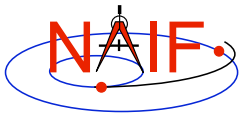
Icy Operation (2)

Navigation and Ancillary Information Facility

- **A user may occasionally encounter an IDL math exception:**

```
% Program caused arithmetic error: Floating underflow
```

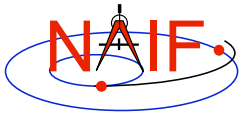
  - **This warning occurs most often as a consequence of CSPICE math operations.**
- **In all known cases, the SIGFPE exceptions caused by CSPICE can be ignored. CSPICE assumes numeric underflow as zero.**
  - **A user can adjust IDL's response to math exceptions by setting to the `!EXCEPT` variable:**
    - » `!EXCEPT = 0` **suppresses the SIGFPE messages.**
    - » `!EXCEPT = 1` **the default, reports math exceptions on return to the interactive prompt.**
    - » `!EXCEPT = 2` **reports exceptions immediately after executing the command.**



Icy Operation (3)

Navigation and Ancillary Information Facility

- **An operational irritant exists when using the `cspice_furnsh` call.**
  - **The IDL program interprets .pro files, so use of Icy's `cspice_furnsh` module loads kernels to IDL, not the calling script. Therefore, kernels and pool variables persist in memory while IDL runs.**
  - **Possible solutions:**
    - » **execute a single `cspice_furnsh` call to load all needed kernels at the start of an IDL run**
    - » **balance every `cspice_furnsh` call with a `cspice_unload`**
- **Please refer to the Icy system required reading, icy.req, for further information.**

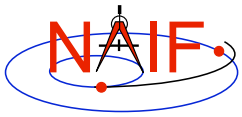


Icy Vectorization (1)

Navigation and Ancillary Information Facility

- **34 Icy interfaces now accept and return vectorized arguments. Use of such arguments can eliminate the need for explicit loops (slow).**
- **Calls vectorized as of Icy 1.2:**

```
cspice_cylrec          cspice_reccyl       cspice_spkezr
cspice_deltet          cspice_recgeo       cspice_spkpos
cspice_et2lst          cspice_reclat       cspice_srfrec
cspice_et2utc          cspice_removd       cspice_srfxpt
cspice_georec          cspice_removi       cspice_subpt
cspice_illum           cspice_recpgr       cspice_str2et
cspice_insrtd          cspice_recrad       cspice_sxform
cspice_insrti          cspice_recsph       cspice_timout
cspice_latrec          cspice_scdecd
cspice_oscelt          cspice_scencd
cspice_pxform          cspice_sce2c
cspice_pgrrec          cspice_scs2e
cspice_radrec          cspice_sphrec
```



Icy Vectorization (2)

Navigation and Ancillary Information Facility

- **Example: use Icy to retrieve state vectors and light-time values for 1000 ephemeris times.**
  - **Create the array of 1000 ephemeris times in steps of 10 hours, keyed on July 1, 2005:**

```
cspice_str2et, 'July 1, 2005', start
et = dindgen( 1000 )*36000.d + start
```

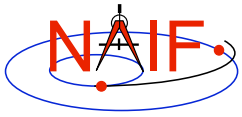
  - **Retrieve the state vectors from Mars to earth at each `et` in the J2000 frame with LT+S aberration correction:**

```
cspice_spkezr, 'Earth', et, 'J2000', 'LT+S', 'MARS', state, ltime
```

  - **Access the *ith* state 6-vector corresponding to the *ith* ephemeris time with the expression**

```
state_i = state[*,i]
```


continues on next page



Icy Vectorization (3)

Navigation and Ancillary Information Facility

  - **Convert the ephemeris time vector to UTC calendar strings with three decimal places accuracy.**

```
format = 'C'
prec   = 3
cspice_et2utc, et, format, prec, utcstr
```

  - **The call returns `utcstr`, a vector of 1000 strings, each *ith* string the calendar date corresponding to `et[i]`.**
  - **Convert the position components of the N state vectors to latitudinal coordinates (the first three components of a state vector - IDL uses a zero based vector index).**

```
cspice_reclat, state[0:2,*], radius, latitude, longitude
```

  - **The call returns three double precision 1000-vectors: `radius`, `latitude`, and `longitude`.**

Simple Use of Icy Functionality

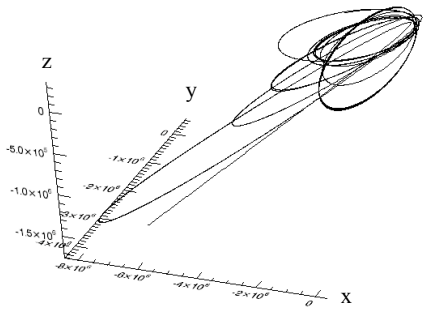
Navigation and Ancillary Information Facility

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

```
;; Define the number of divisions of the time interval and the time interval.
STEP = 10000
utc  = [ 'Jun 20, 2004', 'Dec 1, 2005' ]

;; Load the needed kernels
cspice_furnsh, 'standard.ker'
cspice_furnsh, '/kernels/cassini/spk/T18-5TDJ5.bsp'

;; Create an array of ephemeris times, then retrieve position for each time value
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_unload, 'standard.ker'
cspice_unload, '/kernels/cassini/spk/T18-5TDJ5.bsp'
```


Graphic Output using IDL iTool

Navigation and Ancillary Information Facility



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