Install slalib for python (Ureka)

Working Directory:
<nirc2 pipeline>jlu/nirc2/reduce/slalib_pyf/

Purpose:
I have bundled the necessary SLALIB functions with the pipline. Also IRAF comes with SLALIB code, but it isn't quite ready to be used in python (e.g. refro, refco). To install the python slalib functions that are bundled with the pipeline, you need to do the following steps.

Procedure:

1. Make sure f2py is in your path
2. Get a gfortran compiler
3. Check for *.pyf files
4. Compile the refro and refco modules
5. Test

Step 1: f2py in path
Use *which f2py* to make sure it is installed. It should be in
<Ureka Dir>/python/bin/f2py

Step 2: Fortran compiler
Use *which gfortran* to check if you already have access to one. In Lion, you must

- install Xcode
- install gfortran add on from the GCC Wiki
- it should be located in /usr/local/bin

Step 3: *.pyf files
The necessary SLALIB fortran code and the *.pyf files are included in:

<nirc2 pipeline>/jlu/nirc2/reduce/slalib_pyf/

Check that refro.pyf and refco.pyf are in there along with a number of fortran (*.f) files.

If not, proceed to Appendix A below.

Step 4: Compile
```
f2py --f77exec=gfortran -c refco.pyf refco.f refro.f atms.f atmt.f drange.f
f2py --f77exec=gfortran -c refro.pyf refro.f atms.f atmt.f drange.f
```

Step 5: Check Results
Open ipython and check that you can do the following without error messages.

```
from jlu.nirc2.reduce import dar
```

Appendix A:
If you don't have access to refro.pyf and refco.pyf files, you can make your own from any SLALIB library:

```
f2py refro.f atms.f atmt.f drange.f -m refro -h refro.pyf
f2py refco.f refro.f atms.f atmt.f drange.f -m refco -h refco.pyf
```

Then modify these two *.pyf files in the following manner.

refco.pyf: add intent(out) to refa, refb
FROM:
```
double precision :: refa
double precision :: refb
```
TO:
```
double precision,intent(out) :: refa
double precision,intent(out) :: refb
```

refro.pyf: add intent(out) to ref

FROM:

```
double precision :: ref
```

TO:

```
double precision,intent(out) :: ref
```