# Automatic Vectorization in SLIRP

Michael S. Noble

MIT Kavli Institute for Astrophysics

mnoble@space.mit.edu

May 25, 2006

# "All Science is Interdisciplinary"

-Me, 2006

Until 1940s ...

science impossible w/out mathematics

Today ...

impossible w/out math AND computers

***BUT, must you suffer?***

# SLIRP Primer

- Wrapper generator for C / C++ / FORTRAN
- Greatly simplifies use of external codes within S-Lang

```
unix% slirp –make mffjet.f
    Starter make file generated to 'Makefile'

unix% make
    gcc . . . –O2 –c mffjet_glue.c
    . . .

unix% isis
isis>  import("mffjet")
isis>  mfffunc
Usage: mfffunc( float[], int, float[], int, float[], float[])
```

# SLIRP Primer ...

- This easy for XSPEC, IDL, Matlab, ... ?
- You concentrate on *SCIENCE* ...
- ... tool takes care of *PLUMBING*
- Many other features ... in kindred spirit of

*Where's the Real Bottleneck in Scientific Computing?*

G. Wilson, American Scientist Online, Jan/Feb 2006

# Vectorization

Scalar Usage

```
slsh>  cos(PI)
-1
```

Vectored Usage

```
slsh>  cos( [ 1 : 10000 * PI : PI / 2 ] )
Double_Type[20000]
```

Explicit S-Lang loop avoided

Implied loop executed in compiled C scope

Yielding ***significantly*** greater performance

This s/b reasonably familiar to IDL, Matlab, etc users

But, how do you ***vectorize your own codes?***

# Example: *strlen()* function

Big string array

```
slsh>   strings =  array_map( ... , &sprintf, "string−%d",  [ 1: 500000])
```

Built−in strlen

```
slsh>   tic ;  array_map(Int_Type, &strlen, strings);   toc;
Integer_Type[500000]
3.43534
```

Vectorized

```
unix%   slirp −vec  −rename strlen vstrlen strlen.h
slsh>   import("strlen")
slsh>   vstrlen
Usage:   int = vstrlen(string)
This function has been vectorized.

slsh>   tic ;   vstrlen(strings);   toc;
Integer_Type[500000]
0.062716
```

**Cleaner syntax, and 55 times faster!**

# Example: Vector Multiplication

C function to multiply 2 1D vectors

```
void   vmult(double *x,   double  *y,  double *result, int len)
{
        while (len--)
          result[len] = x[len] * y[len];
}
```

Custom SLIRP vectorization (S–Lang script)

```
#vectorize
    void vmult( double *x,  double  *y,  double  *OUT,  int DIM1)
#end
```

Now can multiply 2D, 3D, ...

```
slsh>  arr2d_1 = array( [5,5,5], [100, 100, 100])
slsh>  arr2d_2 = array([100,100,100], [5,5,5])
slsh>   print ( vmult( arr2d_1  ,  arr2d_2 ) )
 500   500  500
 500   500  500
```

Or 1D * 2D, etc

```
slsh>   print ( vmult( arr2d_1 ,  [9, 9, 9] ) )
45    45    45
900  900  900
```

NB: native S–Lang does not support 1D*2D, 2D*3D, etc!

# Performance (Solaris8)