Wrapping Fortran libraries

Arjen Markus

June 2010

Wrapping Fortran libraries

- Examples of numerical libraries
- Designing the interface
- Generating the wrapper code
- Further developments

A strategy

- Wrapping such libraries makes this functionality available to Tcl programmers
- It also makes Tcl's flexibility available to engineers and scientists

Examples of numerical libraries

- LAPACK: Linear algebra problems
- Specfunc: Special mathematical functions
- MINPACK: least-squares minimisation
- Evaluating multidimensional integrals (Stroud)
- FFTW: Fourier transforms

Designing the interfaces

- Some typical examples
- Good fit to Tcl?

```
 SUBROUTINE AIRYA(X,AI,BI,AD,BD)
C
C       ===================================================
C       Purpose: Compute Airy functions and their derivatives
C       Input:   x  --- Argument of Airy function
C       Output:  AI --- Ai(x)
C                BI --- Bi(x)
C                AD --- Ai'(x)
C                BD --- Bi'(x)
C       Routine called:
C                AJYIK for computing Jv(x), Yv(x), Iv(x) and
C                Kv(x) with v=1/3 and 2/3
C       ===================================================
C
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
```

Interface: fit to Tcl? - part 1

Option 1:

```
proc airy {x ai_ bi_ ad_ bd_} {
    upvar 1 $ai_ ai
    upvar 1 $bi_ bi
    upvar 1 $ad_ ad
    upvar 1 $bd_ bd

    ...
}
```

Pass variable names – rather unusual

Interface: fit to Tcl? – part 2

Option 2: Return a list of values

```
proc airy {x} {
    ...
    return [list $ai $bi $ad $bd]
}
```

Option 3: Separate routines

```
proc airyA {x} {
    ...
    return $ai
}
```

```
proc airyB {x}
    ...
    return $bi
}
```

```
...
```

LAPACK routines – part 1

```
      SUBROUTINE DBDSDC( UPLO, COMPQ, N, D, E, U, LDU, VT, LDVT, Q, IQ,
     $                   WORK, IWORK, INFO )
*
*  -- LAPACK routine (version 3.2) --
*  -- LAPACK is a software package provided by Univ. of Tennessee,    --
*  -- Univ. of California Berkeley, Univ. of Colorado Denver and NAG
   Ltd..--
*     November 2006
*
*     .. Scalar Arguments ..
      CHARACTER          COMPQ, UPLO
      INTEGER            INFO, LDU, LDVT, N
*     ..
*     .. Array Arguments ..
      INTEGER            IQ( * ), IWORK( * )
      DOUBLE PRECISION   D( * ), E( * ), Q( * ), U( LDU, * ),
     $                   VT( LDVT, * ), WORK( * )
*     ..
*
```

LAPACK routines – part 2

- Long argument list
- Arguments that are array dimensions
- Arguments that provide workspace
- Multiple output arguments

Needed in Tcl?

Not if we use lists to store the data

LAPACK: final interface

Hiding the unnecessary arguments:

```
proc dbdsdc { uplo compq d e } {

    ...

    return [list $dnew $u $vt $q $iq $info]  ;# Leaving out "e"
}
```

Data structures

Main types of data:

- Scalars (mostly floating-point and integer)
- One-dimensional arrays (vectors)
- Two-dimensional arrays (matrices)

Currently: mapped onto Tcl lists and nested lists

Generating the wrapper code

Distinguish the various roles of the arguments:

- Input data (scalar/array)
- Output arguments (scalar/array)
- Sizes of arrays
- Workspace
- Options
- Function names

Handling the roles

Boilerplate code (integer array v):

Declaration: `long *v; int size__v;`

Initialisation:

```
if ( WrapCopyIntListToArray( interp, objv[1], &v, &size__v ) != TCL_OK ) {
    Tcl_SetResult( interp, "Argument 1 must be a list of integers", NULL );
    return TCL_ERROR;
}
```

Clean-up: `ckfree((char *)v);`

Definition of arguments

Fortran routine (arrays x and w are returned):

```
     subroutine lagzo( n, x, w )
         integer n
         double precision x(n), w(n)
         ...
     end
```

Tcl wrapper:

```
Wrapfort::fproc ::Specfunc::laguerreZeros lagzo {
    integer       n  input
    double-array  x  {allocate n}
    double-array  w  {allocate n}
    code {} {
        lagzo( &n, x, w );
        … additional code: return arrays x and w …
    }
}
```

Technical issues

- Role of arguments can not be deduced automatically
- Some knowledge of C and Fortran and interfacing the two is required
- Returning more than one result (as a list) requires extra code – not handled automatically yet

LAPACK - wrapping

- Hundreds of routines
- Of most interest: the “driver” routines
- Arguments are well described – automation possible
- Some handcrafting still required (to make the interface more Tcl-like)

MINPACK: user-defined functions

Finding zeros for a vector-valued function:

```
subroutine hybrd1(fcn,n,x,fvec,tol,info,wa,lwa)
integer n,info,lwa
double precision tol
double precision x(n),fvec(n),wa(lwa)
external fcn
...
end
```

The function is implemented via a user-defined subroutine:

```
subroutine fcn(n,x,fvec,iflag)
integer n,iflag
double precision x(n),fvec(n)
----------
calculate the functions at x and
return this vector in fvec.
---------
return
end
```

MINPACK – part 2

The corresponding Tcl procedure:

```
proc fcn {x} {
    ...
    if { ... $x too far away ... } {
        return -code error "Unacceptable solution"
    } else {
        ...
        return $fvec ;# Vector-valued function
    }
}
```

MINPACK - wrapping

Definition via Wrapfort: subroutine fcn(n,x,fvec,iflag)

```
Wrapfort::fexternal fcn {
    fortran {
        integer      n      input
        double-array x      {input n}
        double-array fvec   {output n}
        integer      iflag  output
    }
    toproc {
        x      input
        fvec   result
    }
    onerror {
        *iflag = -1;
    }
}
```

More about Ftcl and Wrapfort

- Ftcl is a project on SourceForge to combine Tcl and Fortran
- Wrapfort is part of that project, focusing on generating interfaces
- You can avoid all C programming

http://ftcl.sf.net

Further developments

- Performance: Tcl lists may not be the best representation in all cases
- Wrapping C libraries (akin to Critcl)

- Combining whole programs – develop a strategy