How to make a dll (dynamic link library) from fortran source code and link it to an excel visual basic program

The advantage in compiling a fortran program into a dll form and linking it to excel is you do not have to convert it to visual basic to benefit from the friendly interface available from excel, and the fortran dll will run much faster (my informal tests have it about 3 times faster than a visual basic program or even a compiled visual basic program) than fortran converted to visual basic. For most applications that is not much of an issue. I happen to have written over 160 subroutines for simulating oil and gas processes (using SRK – soave redlich kwong – equation of state). My routines include isothermal and adiabatic two and three phase flashes, distillation columns (two and three phase), hydrate prediction and of course the simple routines like compression, turboexpanders, pumps, valves, etc. When I simulate a large plant a factor of 3 in execution time is more significant (although still most simulations are done in about 30 seconds, even with complex columns- using a Pentium 2.66 ghz) One final possible advantage. A fortran source code distributed as a dll will remain private (you cannot read the source code).

I originally learned FORTRAN as FORTRAN IV (aka FORTRAN 66) and later FORTRAN 77. I do not think I used many or any extensions to the language. FORTRAN 90 and 95 look substantially different but still support the older fortran. Some of the fortran may look different to you as a result. Some day, I may upgrade the fortran to 90 or 95. I almost hate to do that as it is not backwardly compatible and most of the changes I have seen are cosmetic (although it does seem useful not to use do 100 in a do statement and avoid a lot of statement numbers).

The following is the excel visual basic program and the linking required to get the dll.

I have listed comments to the program in red. I did not get real serious about some of the comments.

```
Option Base 1
```
⇦ This tells the program that the arrays start with 1. The default for visual basic is 0
Since I am converting fortran programs, this is convenient for me to do
Also the 0 starting spot for an array doesn't make much sense to me.

The statement below is the important one for linking a fortran dll (dynamic link library) with excel.

I had two areas that initially gave me trouble: I did not capitalize the name of the subroutine (SOLVE in this case). You have to capitalize it. The next parameter is the file location (that has the subroutine in it – note you will have to change this depending on where you save the dll). Finally you have to list out the dummy arguments and tell the type of variable. The nomenclature here of course is visual basics. Single will be the same as a real*4 in fortran (the default real variable type, range is around 1e-38 to 1e+38 and carries 7 significant figures), double is double precision in fortran (also is real*8). The range for double precision is around 1e-300 to 1e+300 with 14 significant figures).The second area I had problems with is I did not initially identify the integers as Long. Long is the same as integer*4 in fortran (the default in most cases). The default in Visual Basic is Integer*2. If you do not address this your program will not work. Both programs must agree in type of numbers.
One final note. I believe passing character strings requires special handling. I need to look that up again. In the following cases I did not pass character strings from the dll. By the way in this case I had only one subroutine in the dll. You could have many, you just have to list a declare statement for each one of them.

```
Declare Sub SOLVE Lib "c:\util\regress\matrix.dll" (ByRef E As Double, _
     ByRef B As Double, ByRef C As Double, ByRef M As Long, ByRef ndim As Long)


Sub lsq2()
'c ****************************************************************************************
'c
'c                    Least square fit program
'c                    fits equation :
'c
'c      y = ao + a1 * x1 + a2 * x2 + a3 * x3 .... an * xn
'c
'c         the variables x1 through xn can be anything. Earlier versions of the least
'c         square program assumed only two variables, x and y. I just hardwired in different
'c         functions, such as y = ao + a1 * x + a2 * x**2      etc
'c         in this version I would set x2 = x**2
```

```
'c          I could just as easily fit:
'c
'c              y = ao + a1*x + a2*ln(x) + a3*x**2
'c
'c
'c
'c ****************************************************************************************
```

You need to declare the number type to make sure it is compatible with the fortran dll

```
      Dim A(20, 500) As Double
      Dim X(500)   As Double
      Dim Y(500)   As Double
      Dim z(500)   As Double
      Dim r(500)   As Double
      Dim E(20, 20) As Double
      Dim B(20)     As Double
      Dim C(20)     As Double

      ndim = 20   ' this is the dimension for E array in matrix

      ndata = 0
'      read(5,*)nvar                  ! read in number of variables
      i = 1
      nvar = Cells(3, 2).Value
      Do Until (IsEmpty(Cells(i + 3, 3).Value))

'       read(5,*,END=2) (a(j,i),j=1,nvar),r(i)
'         ndata=I
'         j=1
         For j = 1 To nvar
```

```
            A(j, i) = Cells(i + 3, j + 2).Value
          Next j
          r(i) = Cells(i + 3, nvar + 3)
          ndata = i
          i = i + 1
        Loop


'c          Zero Determiment ***************************

        For j = 1 To nvar + 1
          C(j) = 0#
          For K = 1 To nvar + 1
             E(j, K) = 0#
          Next K
        Next j

'c     Do first row ****************************************

        For K = 2 To nvar + 1
          For i = 1 To ndata
            E(1, K) = A(K - 1, i) + E(1, K)
          Next i
        Next K

'c ******** do diagonal ***************************************

        For K = 2 To nvar + 1
          For i = 1 To ndata
            E(K, K) = A(K - 1, i) * A(K - 1, i) + E(K, K)
          Next i
```

```
      Next K

'c ************************************************************

      For j = 2 To nvar + 1
          For K = j + 1 To nvar + 1
            For i = 1 To ndata
              E(j, K) = A(j - 1, i) * A(K - 1, i) + E(j, K)
            Next i
          Next K
       Next j
'c ********* fill in mirror image part of determinant

      For j = 2 To nvar + 1
        For K = 1 To j - 1
           E(j, K) = E(K, j)
        Next K
      Next j

      For i = 1 To ndata
             C(1) = r(i) + C(1)
      Next i

      For j = 2 To nvar + 1
         For i = 1 To ndata
            C(j) = r(i) * A(j - 1, i) + C(j)
         Next i
      Next j

      E(1, 1) = ndata          '! This was tempting to be = 1.0
```

Finally I call the routine that is in the fortran dll. Note the dimension of the array. Normally in visual basic you do Not make E(1,1), you would just list E. It is necessary to pass the first parameter here. It allows the programs to use the Same address for the variables. I know this is not the same as fortran or visual basic, but that is what works.

```
   Call SOLVE(E(1, 1), B(1), C(1), nvar + 1, ndim)     ⇦   This is the call to the dll.

      Cells(ndata + 5, 1).Value = "Y= " & B(1)
      For j = 2 To nvar + 1
        Cells(ndata + 5, 1).Value = Cells(ndata + 5, 1).Value & "+ " & B(j) & " X(" & j - 1 & ") "
      Next j
      Cells(2, nvar + 4) = "Calculated"
      For i = 1 To ndata
         Sum = B(1)
         For j = 1 To nvar
           Sum = B(j + 1) * A(j, i) + Sum
         Next j
       Cells(i + 3, nvar + 4).Value = Sum
'      write(6,'(1h ,2f10.4)')sum,r(i)
      Next i
      End Sub
```

Below is the fortran program that got made into a dll

```
F_stdcall SUBROUTINE SOLVE(A,B,C,M,ndim) ⇦

DIMENSION A(ndim,ndim),B(*),C(*)         ⇦

double precision a,b,c,factor            ⇦
```

Note the F_stdcall. This is what you add to the normal Fortran Subroutine

The array dimensions are the usual, I used the * to pass from the calling program

I used double precision in this routine, just because I wanted to have the greater accuracy. I could have used single in both visual basic and fortran. It is just the integer were the default is not the same

```
C         A IS A SQUARE COEFFICIENT MATRIX,KNOWN VALUES
C         B IS THE SOLUTION VECTOR
C         C IS THE COEFFICIENT VECTOR,KNOWN VALUES
C         M IS THE ORDER OF THE COEFFICIENT MATRIX
      MINOR=M-1
      DO 2 I=1,MINOR
        J=I+1
        DO 2 K=J,M
        FACTOR=A(K,I)/A(I,I)
        DO 1 L=J,M
          A(K,L)=A(K,L)-A(I,L)*FACTOR
    1   CONTINUE
        C(K)=C(K)-C(I)*FACTOR
    2 CONTINUE
      B(M)=C(M)/A(M,M)
      DO 4 I=1,MINOR
        J=M-I
        K=J+1
        DO 3 L=K,M
          C(J)=C(J)-A(J,L)*B(L)
    3   CONTINUE
        B(J)=C(J)/A(J,J)
    4 CONTINUE
      RETURN
      END
```

Now to compile this and make into a dll using the Salford compilier (which is free for personal use):

Ftn95 matrix.for		(note matrix.for is the name of the file that had the solve subroutine in it)
slink	⇦	starts the linker program
dll	⇦	tells it to make a dll
lo matrix.obj	⇦	load the following obj file.
map	⇦	not actually necessary, it just provides a “ map “
file	⇦	the end

You are done. The dll file will be named matrix.dll in this case.

Another compiler is watcom. It is available free from the open watcom project. I found both compilers made executables that had the same execution speed (some compilers are more optimized than others). I initially tried to get the watcom compiler to work, but I was making that mistake in the excel portion where I did not capitalize the subroutine name in the Declare statement. I have not gone back and checked out the watcom program.