NAG & Excel

David Sayers (Numerical Algorithms Group)
Mike Croucher (University of Manchester)

Part 1
The basics
Who am I?

Michael Croucher

- Science and Engineering software specialist
- IT Department, faculty of Engineering and Physical Sciences, University of Manchester
- Technical contact between Manchester and NAG.
- NAG Licensing, installation, deployment
- User queries
This Talk

Part 1 – Excel and the NAG Fortran Library
Michael Croucher

Part 2 – Excel and the NAG C Library, Debugging and tips on VBA.
David Sayers
The full tutorial

*Using the NAG Fortran Library with Excel  2003*

[www.nag.co.uk/doc/techrep/index.asp#tr0208](http://www.nag.co.uk/doc/techrep/index.asp#tr0208)
Agenda

• How to access NAG functions in Excel
• The IFAIL parameter
• Callback functions
• Arrays as input arguments
Why this combination of NAG/Excel?

• Everyone knows how to use Excel!
Why this combination of NAG/Excel?

- For serious numerical work Excel is somewhat lacking.
Why this combination of NAG/Excel?

• For serious numerical work Excel is somewhat lacking.

Friends Don’t Let Friends Use Excel for Statistics!

(Jon Cryer, 2001 – University of Iowa)
Why this combination of NAG/Excel?

- For serious numerical work Excel is somewhat lacking.

Friends Don’t Let Friends Use Excel for Serious Numerical Work!

(Mike Croucher, 2009, University of Manchester)
Why this combination of NAG/Excel?

- NAG is great at serious numerical work but the main library is written in FORTRAN.

- FORTRAN is extremely fast but requires a lot of work from the developer.
Why this combination of NAG/Excel?

- Excel – User Interface
Why this combination of NAG/Excel?

• Excel – User Interface

• NAG – Numerical heavy lifting
Why this combination of NAG/Excel?

NAG program in FORTRAN so you don’t have to!
Example – Bessel Function

- Shows how to access the NAG Library in Excel
- Demonstrates basic structure of almost all NAG-Excel programs
- S17AEF returns the value of the Bessel Function $J_0(x)$.
- S17AEF(x,IFAIL)
'Listing 1 - basic interface for S1AEF
Option Explicit
Option Base 1

Declare Function S17AEF Lib "FLDLL214M_nag.dll" ( _
    ByRef X As Double, _
    ByRef IFAIL As Long _
) As Double
Vb6.txt

• Contains VBA function declarations for ALL NAG routines.

• On Start Menu

• http://www.nag.co.uk/Market/training/Manchester_Finance_Feb09/
To Excel

• Show Listing 1
IFAIL

• S17AEF(x,IFAIL)

• In example spreadsheet IFAIL=1

• IFAIL determines what happens if the NAG routine detects an error of some kind.
IFAIL – Error Handling

• Input
  -1 Noisy Return (Output error and continue)
  0 Hard Return (Output error and stop)
  1 Quiet Return (Do nothing)
IFAIL – Error Handling

• Input
  -1 Noisy Return (Output error and continue)
  0 Hard Return (Output error and stop)
  1 Quiet Return (Do nothing)

• Output
  Varies according to function. See individual function documentation for details.
IFAIL – Output

• For S17AEF (Bessel Function)

• Gets set to 0 if successful

• Get’s set to -1 if there is a problem
Example – Better Bessel

'Listing 2 - Improved interface to S17AEF
Option Explicit
Option Base 1

Declare Function S17AEF Lib "FLDLL214M_nag.dll" ( _
    ByRef x As Double, _
    ByRef IFAIL As Long _
) As Double

Function Bessel(x As Double) As Variant
    Dim IFAIL As Long
    IFAIL = 1
    Bessel = S17AEF(x, IFAIL)
    If (IFAIL <> 0) Then
        Bessel = "Argument too large"
    End If
End Function
Callback functions

• A callback function is executable code that is passed as an argument to other code.

• Example – A general purpose quadrature routine.

• The function to be integrated is passed as a callback to the quadrature routine.
Example – Numerical Integration

'Listing 3 - Demonstration of callback functions
Option Explicit
Option Base 1

Declare Function D01AHF Lib "FLDLL214M_nag.dll" ( _
    ByVal a As Double, _  
    ByVal b As Double, _  
    ByVal epsr As Double, _  
    ByVal npts As Long, _  
    ByVal relerr As Double, _  
    ByVal F As Long, _  
    ByVal nlimit As Long, _  
    ByVal IFAIL As Long _  
) As Double

Function funcy(x As Double) As Double
    funcy = 4 / (1 + x * x)
End Function

Function integrate(a As Double, b As Double, epsr As Double)
    Dim relerr As Double
    Dim nlimit As Long, IFAIL As Long, npts As Long
    nlimit = 0
    integrate = D01AHF(a, b, epsr, npts, relerr, AddressOf funcy, nlimit, IFAIL)
End Function
Example – Numerical Integration

Function funcy(x As Double) As Double
    funcy = 4 / (1 + x * x)
End Function

Function integrate(a As Double, b As Double, epsr As Double)
    Dim relerr As Double
    Dim nlimit As Long, IFAIL As Long, npts As Long
    nlimit = 0
    integrate = D01AHF(a, b, epsr, npts, relerr, AddressOf funcy, nlimit, IFAIL)
End Function
Arrays as Input Arguments

- F03AAF – Determinant of a real matrix
- Go to Excel – Listing 5.
Matrix Determinant Part 1

'Listing 5 - More advanced array example using F03AAF
Option Explicit
Option Base 1

Declare Sub F03AAF Lib "FLDLL214M_nag.dll" ( _
    ByVal a As Double, _
    ByVal IA As Long, _
    ByVal n As Long, _
    ByVal det As Double, _
    ByVal WKSPCE As Double, _
    ByVal IFAIL As Long _
)
Function determinant(a As Range)

Dim myArray() As Double
Dim x As Long
Dim y As Long
Dim result As Double
Dim WKSPCE() As Double
Dim IFAIL As Long

Call Assemble(a, myArray) 'Takes the range a and puts it into the vba array myArray

'get array dimensions
x = UBound(myArray, 1)
y = UBound(myArray, 2)

If (x <> y) Then
    determinant = "Error: Given range is not square"
    Exit Function
End If

'create workspace
ReDim WKSPCE(x)

IFAIL = 1
Call F03AAF(myArray(1, 1), x, x, result, WKSPCE(1), IFAIL)
determinant = result
End Function
Sub Assemble(x As Range, a As Variant) ' Takes an argument X and form a VB array A (ReDimmed)
Dim i As Long, j As Long
Dim m As Long, n As Long

' Now get the overall dimensions of the matrix
m = x.Rows.Count
n = x.Columns.Count

ReDim a(m, n) ' VB Array redimensioned

' Assemble the matrix A
For j = 1 To n
    For i = 1 To m
        a(i, j) = x.Cells(i, j).Value
    Next i
Next j

End Sub
Conclusions

• Calling many NAG Fortran functions from Excel is simple.

• More detail in my NAG-Excel paper.

• Ask if you find yourself needing something that isn’t covered

• Michael.Croucher@manchester.ac.uk
NAG & Excel

Part 2
Agenda

• Calling the C DLL
• Debugging Excel and NAG
• Tips on exploring VBA
• Getting help
Calling the C DLL

• ‘Essentially’ the same as Fortran DLL
  – Need to define the C types in VBA *
  – C arrays start with 0
  – C arrays need transposing

* NAG provide these!
Let’s See an Example
Debugging Excel and NAG

- NAG routines do NOT accept Variants
- NAG routines use storage association to accept array arguments i.e. first element of array.
- Always use the ‘Declare Statements’ provided by NAG – note string handling.
- Check that array sizes are large enough by looking at NAG DOC
Debugging Excel and NAG

Don't Forget
Tips on Exploring VBA

• Use the VBA Help system
• Record a MACRO and see what VBA is used.
• Look at the Object Browser
Tips on Exploring VBA

Reminder!
Getting Help

- applicationsupport-eps@manchester.ac.uk
- support@nag.co.uk
- david.sayers@nag.co.uk

- This email message has been sent to you on behalf of NAG Technical Support.
- So that we can provide you with good service, please ensure that you copy any replies to support@nag.co.uk. This will help to ensure that you get a response.
- even if the staff member to which you directly address your message is absent.

- ************************************************** ***************************