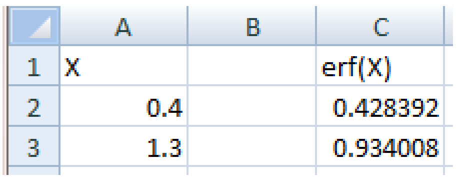
Solutions to Sample Problems

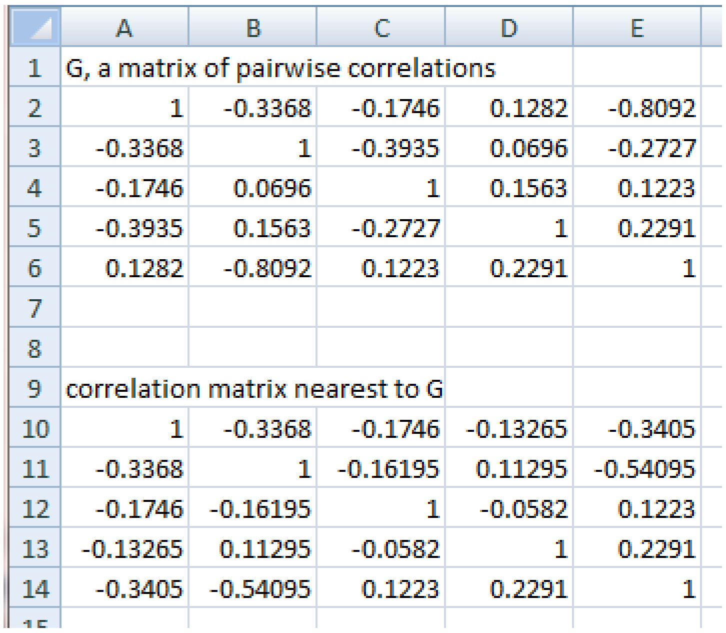
Problem 1:



	A	B	C
1	X		erf(X)
2	0.4		0.428392
3	1.3		0.934008

```
' function declaration, as supplied in VB6 headers
Declare Function S15AEF Lib "FLDLL254M_nag.dll" ( _
  ByRef x As Double, _
  ByRef ifail As Long _
) As Double

Function NAG_erf(x As Variant) As Variant
  ' return the erf function
  Dim RX As Double
  Dim ifail As Long

  ' convert the supplied input to a scalar double
  RX = NAG_GetDoubleScalar(x)

  ' set the NAG error mechanism to a quiet soft exit
  ifail = 1

  ' call the NAG routine
  NAG_erf = S15AEF(RX, ifail)

  ' there are no errors possible from this routine, so
  ' ignore IFAIL
End Function
```

Problem 2:

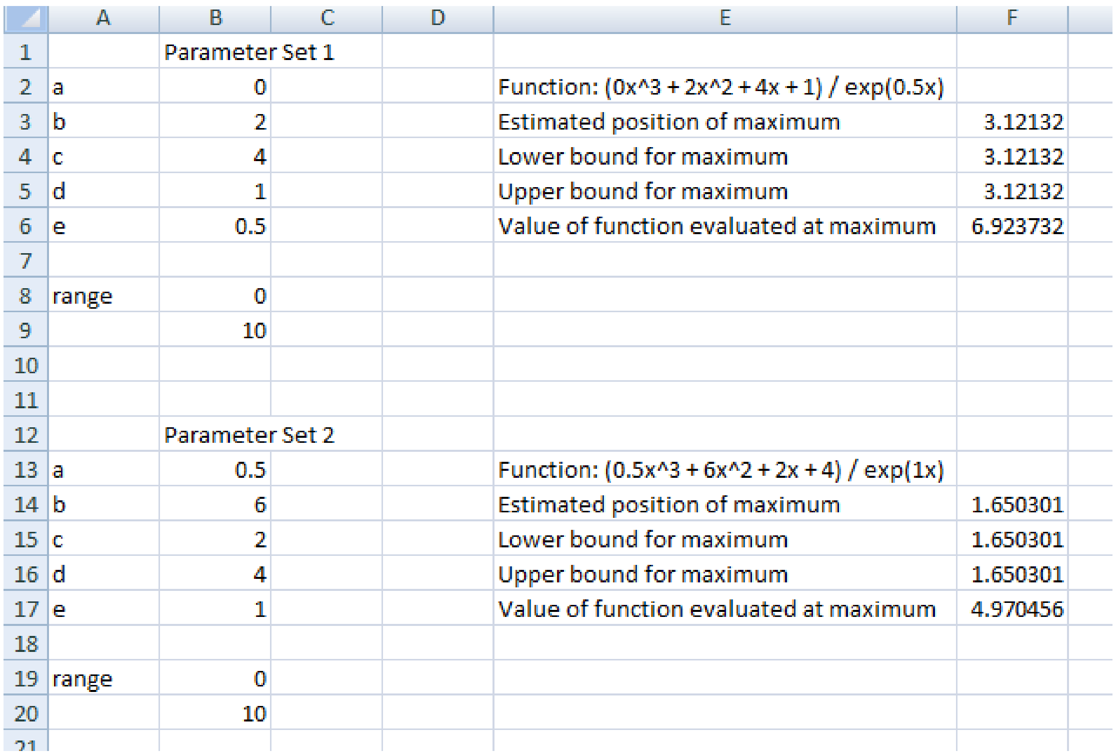


	A	B	C	D	E
1	G, a matrix of pairwise correlations				
2	1	-0.3368	-0.1746	0.1282	-0.8092
3	-0.3368	1	-0.3935	0.0696	-0.2727
4	-0.1746	0.0696	1	0.1563	0.1223
5	-0.3935	0.1563	-0.2727	1	0.2291
6	0.1282	-0.8092	0.1223	0.2291	1
7					
8					
9	correlation matrix nearest to G				
10	1	-0.3368	-0.1746	-0.13265	-0.3405
11	-0.3368	1	-0.16195	0.11295	-0.54095
12	-0.1746	-0.16195	1	-0.0582	0.1223
13	-0.13265	0.11295	-0.0582	1	0.2291
14	-0.3405	-0.54095	0.1223	0.2291	1

```
' function declaration, as supplied in VB6 headers
Declare Sub G02AAF Lib "FLDLL254M_nag.dll" ( _
  ByRef g As Double, _
  ByRef ldg As Long, _
  ByRef n As Long, _
  ByRef errtol As Double, _
  ByRef maxits As Long, _
  ByRef maxit As Long, _
  ByRef x As Double, _
  ByRef ldx As Long, _
  ByRef iter As Long, _
  ByRef feval As Long, _
  ByRef nrmgrd As Double, _
  ByRef ifail As Long _
)
```

```
Function NAG_nearestCorrelation(rg As Range) As Variant
  ' calculate the nearest correlation matrix to the matrix
  ' supplied in rg
  Dim nrow As Long, ncol As Long, n As Long, ifail As Long
  Dim ldg As Long, ldx As Long, iter As Long, feval As Long
  Dim maxit As Long, maxits As Long, i As Long, j As Long
  Dim nrmgrd As Double, errtol As Double
  Dim g() As Double, x() As Double
  Dim vntOutput As Variant

  Call NAG_GetDoubleMatrixFromRange(rg, nrow, ncol, g)

  ' expect the supplied matrix to be square, but use
  ' the smallest of nrow and ncol just incase
  n = IIf(nrow < ncol, nrow, ncol)

  ' g is nrow by ncol
  ldg = nrow

  ' use default values
  errtol = 0#
  maxits = 0
  maxit = 0

  ' allocate memory for output
  ldx = n
  ReDim x(ldx, n)

  ' set the NAG error mechanism to a quiet soft exit
  ifail = 1

  ' call the NAG routine
  Call G02AAF(g(1, 1), ldg, n, errtol, maxits, maxit, x(1, 1), _
              ldx, iter, feval, nrmgrd, ifail)

  If (ifail <> 0) Then
    ' handle any errors
    ReDim vntOutput(1, 1)
    vntOutput(1, 1) = "G02AAF returned with IFAIL = " + CStr(ifail)

  Else
    ' output the result
    ReDim vntOutput(n, n)
    For j = 1 To n
      For i = 1 To n
        vntOutput(i, j) = x(i, j)
      Next i
    Next j
  End If

  NAG_nearestCorrelation = vntOutput
End Function
```

Problem 3:



	A	B	C	D	E	F
1		Parameter Set 1				
2	a	0			Function: (0x^3 + 2x^2 + 4x + 1) / exp(0.5x)	
3	b	2			Estimated position of maximum	3.12132
4	c	4			Lower bound for maximum	3.12132
5	d	1			Upper bound for maximum	3.12132
6	e	0.5			Value of function evaluated at maximum	6.923732
7						
8	range	0				
9		10				
10						
11						
12		Parameter Set 2				
13	a	0.5			Function: (0.5x^3 + 6x^2 + 2x + 4) / exp(1x)	
14	b	6			Estimated position of maximum	1.650301
15	c	2			Lower bound for maximum	1.650301
16	d	4			Upper bound for maximum	1.650301
17	e	1			Value of function evaluated at maximum	4.970456
18						
19	range	0				
20		10				
21						

```
' function declaration, as supplied in VB6 headers
Declare Sub E04ABA Lib "FLDLL254M_nag.dll" ( _
  ByVal funct As Long, _
  ByRef e1 As Double, _
  ByRef e2 As Double, _
  ByRef a As Double, _
  ByRef b As Double, _
  ByRef maxcal As Long, _
  ByRef x As Double, _
  ByRef f As Double, _
  ByRef iuser As Long, _
  ByRef ruser As Double, _
  ByRef ifail As Long _
)

' Copies memory from pointer
Declare Sub CopyMemFromPtr Lib "kernel32" Alias "RtlMoveMemory" ( _
 ByRef hpvDest As Any, ByVal hpvSource As Any, ByVal cbCopy As Long)

' Copies memory to pointer
Declare Sub CopyMemToPtr Lib "kernel32" Alias "RtlMoveMemory" ( _
 ByVal hpvDest As Long, ByRef hpvSource As Any, ByVal cbCopy As Long)
```

```
Function NAG_maxFX(rrange As Range, rparam As Range) As Variant
  ' example of calling E04ABA
  ' rparam is expected to hold 5 values
  Dim lenab As Long, maxcal As Long, ifail As Long
  Dim lenrparam As Long, iuser(1) as long
  Dim e1 As Double, e2 As Double, a As Double, b As Double
  Dim x As Double, f As Double, ruser() As Double, ab() As Double
  Dim vntOutput As Variant

  ' get the parameters
  Call NAG_GetDoubleVectorFromRange(rrange, lenab, ab)
  Call NAG_GetDoubleVectorFromRange(rparam, lenrparam, ruser)

  ' set the NAG error mechanism to a quiet soft exit
  ifail = 1

  ' use default values for e1 and e2
  e1 = 0#
  e2 = 0#

  ' get the limits
  a = ab(1)
  b = ab(2)

  ' set maxcal to a large value
  maxcal = 1000

  ' call the NAG routine
  Call E04ABA(AddressOf Exercise3_UserCallableFunction, e1, e2, a, _
            b,  maxcal, x, f, iuser(1), ruser(1), ifail)

  If (ifail <> 0) Then
    ' handle any errors
    ReDim vntOutput(1, 1)
    vntOutput(1, 1) = "E04ABA returned with IFAIL = " + CStr(ifail)

  Else
    ' output the result
    ReDim vntOutput(5, 2)

    vntOutput(1, 1) = "Function: " + _
        "(" + CStr(ruser(1)) + "x^3 + " + CStr(ruser(2)) + _
        "x^2 + " + CStr(ruser(3)) + "x + " + _
        CStr(ruser(4)) + ") / exp(" + CStr(ruser(5)) + "x)"
    vntOutput(1, 2) = ""
    vntOutput(2, 1) = "Estimated position of maximum"
    vntOutput(2, 2) = x
    vntOutput(3, 1) = "Lower bound for maximum"
    vntOutput(3, 2) = a
    vntOutput(4, 1) = "Upper bound for maximum"
    vntOutput(4, 2) = b
    vntOutput(5, 1) = "Value of function evaluated at maximum"
    ' note, we multiply the returned value by -1 as we E04ABA
    ' minimises the function
    vntOutput(5, 2) = -f
  End If

  NAG_maxFX = vntOutput
End Function
```

```
Sub Exercise3_UserCallableFunction( _
    ByRef xc As Double, _
    ByRef fc As Double, _
    ByVal iuser_iptr As Long, _
    ByVal ruser_rptr As Long _
)
  Dim lruser As Long
  Dim ruser() As Double

  ' we are using 5 elements of ruser
  lruser = 5

  ' allocate memory for ruser
  ReDim ruser(lruser)

  ' copy array input from ruser_rptr into local arrays
  ' we are not using iuser_rptr
  Call CopyMemFromPtr(ruser(1), ruser_rptr, lruser * Len(ruser(1)))

  ' calculate the value we are trying to minimise
  fc = (ruser(1) * xc ^ 3 + ruser(2) * xc ^ 2 + ruser(3) * xc + _
       ruser(4)) / Exp(ruser(5) * xc)

  ' want to maximise fc, so multiply by -1
  fc = -fc

  ' if we were altering anything in RUSER we would now have to copy
  ' ruser back into ruser_ptr via a call to CopyMemToPtr
End Sub
```