```
'program to calculate the conductance between opposite faces of a cube
'cubic array of conductances drawn from a log-normal distribution
'has response model- relative change in conductance that is the same for all elements
'varies the magnitude of the individual element response to get the overall cube response
'activated by command button click
'reads and writes to spreadsheet
'revised version with cyclic boundary conditions on conductivity for non-electrode boundaries

'at the beginning, need to declare all variables as Private so that they are accessible to all subr
outines
Option Base 1            'array indices start at 1
Private V() As Double, conductance() As Double, deltacond() As Double, deltacond2() As Double
Private zerocond() As Double        'this holds the conductances after solution for the 'zero conce
ntration'

Private cubeconductance As Double   'this is the result for the conductance of the cube
' cube size
Private nx As Integer
'relaxation rate parameter and final value for this
Private relax As Double, relax2 As Double, RelaxFinal As Double
'
'random number seed for generating distributions
Private seed As Single
'lognormal mean and sd
Private logmean As Double, logsd As Double
'dummy variable to hold conductance distribution values
Private final_conductance As Double, OldConductance As Double, ChangeConductance As Double, final_c
onductance2 As Double
Private CubeConduct As Double       'calculated conductance of the cube
'counters
Private Rln As Integer ' counter As Integer
Dim i As Integer, j As Integer, k As Integer, colno As Integer
Private outstrtcol As Integer, outstrtrow As Integer, outcol As Integer, outrow As Integer, outstrt
col2 As Integer 'row and column for output
Private RelResponse() As Double   'relative 'response' deltaR/R(final) to model change in distribut
ion
Private RelResponseSD As Double     'arithmetic relative response sd
Private RelResponseMN As Double     'arithmtic mean of relative response
Private LogResponseMean As Double          'lognormal mean for response distribution
Private LogResponseSD As Double            'lognormal sd for response distribution
Private ConcLower As Single, ConcUpper As Single, Conc As Double    'log 'concentration' lower and
upper values, and 'concentration'
Private StepConc As Single, LogConc As Single                       'for 'concentration' loop

Private Sub CommandButton1_Click()
'variables for input are specified here as are the spreadsheet cells for input and output
'calculation is called and results for each realiation written out into the spreadsheet
    nx = 50
    'fix cube size and redimension arrays. Voltage is at every point, Conductance is bonds between
points, assigned to points
    ReDim V(nx, nx, nx), conductance(nx, nx, nx), deltacond(nx, nx, nx), zerocond(nx, nx, nx), RelR
esponse(nx, nx, nx)
    ReDim deltacond2(nx, nx, nx)
    'fix the parameters for the relaxation
    relax = 0.5
    relax2 = 1  'this is the running value for the relaxation parameter
    RelaxFinal = relax ^ 10     'takes 10 steps for relaxation
    Rln = 150
    'number of cubes calculated;
    'specify the cells from which to read in the logmean and logsd
    logmean = Cells(11, 2).Value
    logsd = Cells(12, 2).Value
    'specify the cell from which to read the relative response mean and SD
    LogResponseMean = Cells(9, 2).Value
    LogResponseSD = Cells(9, 4).Value
    'specify the starting cells into which to write the results
    outstrtrow = 18
    outstrtcol = 2
    '
    'specify the 'concentration' range and (logarithmic) step
    ConcLower = -4
    ConcUpper = 2
    StepConc = 0.25

For counter2 = 1 To Rln
'loop, incrementing the realisation number until the required number have been executed
    Call initialise       'initialise arrays and compute final conductance array
```

```
    outrow = outstrtrow + counter2 - 1
    'fix the parameters for the relaxation
    relax = 0.5
    colno = outstrtcol
    Call relaxation(outrow, colno, relax)
    '
    '--------------------------
    'now do the bit about 'response' using the final relaxation
    'in this variation, a log-normal distribution of relative response is assumed
    'the log mean and sd are read in.

    'first store the 'zero' start remaining after the relaxation calculation

    For i = 1 To nx
        For j = 1 To nx
            For k = 1 To nx
                zerocond(i, j, k) = conductance(i, j, k)
            Next k
        Next j
    Next i
'
    'conductance and voltage arrays are left at the values following the relaxation
    'outrow is still set from the previous relaxation; colno has been returned from the relaxation
subroutine

        colno = colno + 2
        outstrtcol2 = colno

    'now calculate the 'relative response' taken from a lognormal distribution
    'find the 'relative response' coefficient values for each element
    Call Randomize
    For i = 1 To nx
        For j = 1 To nx
            For k = 1 To nx
                seed = Rnd() * 0.9999999 + 0.00000001    'cumulative probability value chosen at ra
ndom but very low and very high values avoided to avoid error in getting inverse distribution
                RelResponse(i, j, k) = Application.WorksheetFunction.LogInv(seed, LogResponseMean,
LogResponseSD)
                'RelResponse(i, j, k) = 1        'making things simple to debug
            Next k
        Next j
    Next i
    '
    'now step up the 'concentration' from the lower to the upper limit, recalculating the cube cond
uctance in each step, talking the previous one as the starting point
    'logarithmic steps
    For LogConc = ConcLower To ConcUpper Step StepConc
        Conc = 10 ^ LogConc
        For i = 1 To nx
            For j = 2 To nx - 1
                For k = 2 To nx - 1

                   'OldConductance = zerocond(i, j, k)      'this is the conductance array left aft
er the 'zero' calculation
                    final_conductance2 = zerocond(i, j, k) / (1 + RelResponse(i, j, k) * Conc) 'thi
s is for a response which is a conductivity decrease
                                                                         ' the relative response is
deltaR/R0
                    deltacond2(i, j, k) = final_conductance2 - zerocond(i, j, k)  'recalculates the
 conductance array
                    conductance(i, j, k) = zerocond(i, j, k) + deltacond2(i, j, k)
                 Next k
            Next j
        Next i
    'recalculate the cube conductance
        relax = 0.5
        'Call relaxation(outrow, colno, relax)
        Call VoltageArray   'ideally 'relaxation' would be called but the way the variables are set
 up leads to negative conductances if this is done
                            'should pass deltacond array to relaxation as a parameter. too lazy to
reorganise it! Happy to accept the extra processing time
                            ' the conductances are changing by small increments in this stage so th
e relaxation procedure is not needed
        Call conduct
    'now write the results into the spreadsheet

        Cells(outrow, colno).Value = cubeconductance
```

```
        '
        colno = colno + 1
    Next LogConc




    'reset the out start column for the next realisation
    outstrtcol = 2
    '
Next counter2
'
'finally, tidy up by writing into the spreadshhet the logconc numbers
    outrow = outrow + 2
    colno = outstrtcol2
    For LogConc = ConcLower To ConcUpper Step StepConc
        Cells(outrow, colno).Value = LogConc
        colno = colno + 1
    Next LogConc
'
End Sub
'-------------------------------
Sub initialise()
'initialise conductance and voltage and calculates total conductance change for each element

Dim i As Integer, j As Integer, k As Integer
    Call initconductance
    'initialise conductance values (at 1/exp^logmean)
    Call initvoltage
    'initialise voltage array as linear gradient between faces V =0 at i = 1 and V = 1 at i= nx
    'this also defines the boundary conditions at i = 0 and i = 1

    'set up total conductance change array: use spreadsheet function to calculate final conductance
 for each element
    'then calculate the change from the initial value and store this in the array
    Call Randomize

    For i = 1 To nx
        For j = 1 To nx
            For k = 1 To nx
            seed = Rnd() * 0.9999999 + 0.00000001    'cumulative probability value chosen at random
 but very low and very high values avoided to avoid error in getting inverse distribution
            final_conductance = 1 / Application.WorksheetFunction.LogInv(seed, logmean, logsd)
                'value taken from cumulative probability distribution with random probability
                'distribution is of resistance. Conductance taken as inverse

            deltacond(i, j, k) = final_conductance - conductance(i, j, k)
            'deltacond(i, j, k) = 0 'debug: take out the variation
            Next k
        Next j
    Next i

End Sub
'-------------------------------------------
Sub relaxation(outrow As Integer, colno As Integer, relax As Double)
'conductance values change incrementally to the final value
Dim counter1 As Integer         'counts the relaxation
'start relaxation loop
    'loop, decreasing the relaxation parameter until the final value is attained: all conductances
arbitrarily close to final value
'reset the relaxation parameter
relax2 = 1
'counter1 = 0

    Do Until relax2 < RelaxFinal

        relax2 = relax2 * relax  'at each step the conductance increment decreases
        'calculate the new conductance values - incrementally approaching the final value
        For i = 1 To nx
           For j = 1 To nx
                For k = 1 To nx
                    'OldConductance = conductance(i, j, k)
                    ChangeConductance = deltacond(i, j, k) * relax2
                    conductance(i, j, k) = conductance(i, j, k) + ChangeConductance 'careful about
sign: based on sign of deltacond
              If conductance(i, j, k) < 0 Then Stop
                Next k
```

```
            Next j
        Next i
        'solve for voltage and then for cube conductance
       Call VoltageArray
       Call conduct

    'write out cube conductance to spreadsheet for each stage:relaxation stages across, realisation
s down
    'colno = outstrtcol + counter1

  '
    'counter1 = counter1 + 1
    Loop

    Cells(outrow, colno).Value = cubeconductance
End Sub
'----------------------------------------------
Sub VoltageArray()
'
      'iterates the potential field to a self-consistent solution
      '
      'define local conductance variables
      Dim old As Double, sum1 As Double, sum2 As Double
      Dim a1 As Double, a2 As Double, a3 As Double, a4 As Double
      Dim a5 As Double, a6 As Double, a7 As Double, r As Double

      'define convergence criterion
      Dim condition As Double, resid As Double
       condition = 0.00005 * nx * nx * nx
 'iterate until less than 0.00005 relative change in values overall .
 'residual is the relative change for each value at each iteration and is summed over all values


' main loop start
resid = 10 * condition 'dummy value to start off
Do Until resid < condition      ' test occurs here; resid is reset afer this statement
 resid = 0
         'residual to compare with condition:reset at each iteration and accumulated
     'leave boundary values at i=1,nx unchanged - electrodes
    For i = 2 To nx - 1
    'voltage is fixed at i = 1 and nx
        'apply cyclic bc's on conductance for other edges
        'conductance values are assigned to lattice points.
        'conductance between lattice points is taken as the average of these values
        For j = 2 To nx - 1
            For k = 2 To nx - 1
                sum1 = 0
                sum2 = 0
                old = V(i, j, k)

        sum1 = conductance(i, j - 1, k) + conductance(i, j + 1, k)
        sum1 = sum1 + conductance(i, j, k + 1) + conductance(i, j, k - 1)
        sum1 = sum1 + conductance(i - 1, j, k) + conductance(i + 1, j, k)
        sum1 = sum1 + 6 * conductance(i, j, k)
                       a1 = V(i, j + 1, k) * conductance(i, j + 1, k)
                       a2 = V(i, j - 1, k) * conductance(i, j - 1, k)
                       a3 = V(i, j, k + 1) * conductance(i, j, k + 1)
                       a4 = V(i, j, k - 1) * conductance(i, j, k - 1)
                       a5 = V(i - 1, j, k) * conductance(i - 1, j, k)
                       a6 = V(i + 1, j, k) * conductance(i + 1, j, k)
                       a7 = V(i, j - 1, k) + V(i, j + 1, k) + V(i, j, k - 1) + V(i, j, k + 1) + V(i
 - 1, j, k) + V(i + 1, j, k)
                       a7 = a7 * conductance(i, j, k)
                       sum2 = a1 + a2 + a3 + a4 + a5 + a6 + a7
                        r = sum2 / sum1
                       V(i, j, k) = r
                       resid = resid + Abs((old - r) / r)

                    Next k

                Next j
        'the bc for i = 0 and i = nx are set up in the initialisation and not changed
        'now have to flip across the boundaries for j and k
        'exactly the same calculation as above except that when j or k = nx, j+1 or k+1 is taken as
 1
        'when j or k = 1,j+1 or k+1 is taken as 1
        'do j first
```

```
            For k = 1 To nx
                Call CyclicBC(i, 1, k, resid)
                Call CyclicBC(i, nx, k, resid)
            Next k
        ' then k
            For j = 1 To nx
                Call CyclicBC(i, j, 1, resid)
                Call CyclicBC(i, j, nx, resid)
            Next j

   Next i
Loop       'end of iteration loop. When condtion is satisfied, subroutine exits
Exit Sub
'
End Sub

'---------------------------------------------------------
Sub conduct()
'calculates conductance between cube faces

'compute resistance by computing current flow through electrodes
      'pd across device = 1 ie (nx-1) boxes
      'potential gradient is therefore 1/(nx-1)
      'current through each electrode box is therefore potential gradient*conductivity
      'again, conductivity assigned to the link from electrode to conductor is the average
      'of that between the electrode and the adjacent point
      'total current is the sum
      'conductivity is total current/(potential gradient*area) therefore sum(condy)*nx/(nx)^2
      'do both electrodes - should be the same. Take the average
Dim conductor As Double, conductor2 As Double
       conductor1 = 0
       conductor2 = 0
    For j = 2 To nx - 1
           'ignoring the edges
        For k = 2 To nx - 1
               conductor1 = conductor1 + V(2, j, k) * (conductance(1, j, k) + conductance(2, j, k))
 / 2
               conductor2 = conductor2 + (1 - V(nx - 1, j, k)) * (conductance(nx, j, k) + conductan
ce(nx - 1, j, k)) / 2
        Next k
    Next j
        'conductor is the total current through the electrodes
       cubeconductance = (conductor1 + conductor2) * (nx - 1) / (2 * (nx - 2) * (nx - 2))
       'cubeconductance = conductor  'debugging

End Sub
'----------------------------------------------------------
Sub initconductance()
'sets up starting values for the conductance
Dim i As Integer, j As Integer, k As Integer
    'initialise conductance values at 10^logmean
    'Conductances are assigned to points of the array and
    'conductances between points are calculated as the average
    For i = 1 To nx
        For j = 1 To nx
         For k = 1 To nx
          conductance(i, j, k) = 1 / Exp(logmean)   'distribution of resistance is determined
        Next k
        Next j
    Next i
End Sub

Sub initvoltage()
'sets up starting voltage array and boundry conditions at electrode faces
'linear variation along i - axis
Dim i As Integer, j As Integer, k As Integer
For i = 1 To nx
        For j = 1 To nx
            For k = 1 To nx
                V(i, j, k) = (i - 1) / (nx - 1)
            Next k
        Next j
    Next i
End Sub

Sub CyclicBC(i As Integer, j As Integer, k As Integer, resid As Double)
'does the sums for the voltage array cyclic bc iteration. Put into a subroutine to keep things tidy
```

```
Dim j_up As Integer, j_down As Integer, k_up As Integer, k_down As Integer
'now set up the cycling part. This is set up assuming that only j and k cycle
'also assumes that this clculation occurs only for the boundary values so the 'if' statements dont have to include anything else
'first set up the index values, otherwise when subroutine enters thet will have value zero,then test condition and reset
j_up = j + 1
j_down = j - 1
k_up = k + 1
k_down = k - 1

If j = 1 Then
    j_down = nx
End If
If j = nx Then
    j_up = 1
End If
If k = 1 Then
    k_down = nx
End If
If k = nx Then
    k_up = 1
End If

                sum1 = 0
                sum2 = 0
                old = V(i, j, k)


        sum1 = conductance(i, j_down, k) + conductance(i, j_up, k)
        sum1 = sum1 + conductance(i, j, k_up) + conductance(i, j, k_down)
        sum1 = sum1 + conductance(i - 1, j, k) + conductance(i + 1, j, k)
        sum1 = sum1 + 6 * conductance(i, j, k)
                       a1 = V(i, j_up, k) * conductance(i, j_up, k)
                       a2 = V(i, j_down, k) * conductance(i, j_down, k)
                       a3 = V(i, j, k_up) * conductance(i, j, k_up)
                       a4 = V(i, j, k_down) * conductance(i, j, k_down)
                       a5 = V(i - 1, j, k) * conductance(i - 1, j, k)
                       a6 = V(i + 1, j, k) * conductance(i + 1, j, k)
                       a7 = V(i, j_down, k) + V(i, j_up, k) + V(i, j, k_down) + V(i, j, k_up) + V(i - 1, j, k) + V(i + 1, j, k)
                       a7 = a7 * conductance(i, j, k)
                       sum2 = a1 + a2 + a3 + a4 + a5 + a6 + a7
                        r = sum2 / sum1
                       V(i, j, k) = r
                       resid = resid + Abs((old - r) / r)
End Sub
```