Appendix A  VBA Code

The following pages contain the source code that was used to simulate the model as outlined above.

I have categorised the functions into the following subchapters:

- Functions in the section “Simple Luhmann Economy Model” are the core of the model. I have to apologise that some error messages are in German. The meaning should be obvious by the if-then-else clauses. I tried to keep comments (although few) in English.
- Functions in the section “Fuzzy clustering” refer to the implementation of the fuzzy-c-means clustering algorithm. Special care regarding underflow (as a likely event) had to be taken in the modules.
- Functions of “Exploration” refer to functions that were used in the simulation and exploration part (e.g. the identification of cycles).
- Auxiliary functions like seeking minima, or taking care of the torus are contained in the last section.

Some debugging messages have not been erased but commented out because I regarded them as helpful in understanding the code.

The Beta distribution is given as follows:

\[
    f(x) = \frac{1}{\beta(a,b)} x^{a-1} (1-x)^{b-1} 1_{(0,1)}(x)
\]

\[
    \beta(a,b) = \int_0^1 x^{a-1} (1-x)^{b-1} dx
\]

\[
    a, b > 0
\]

*Formula 1: density function of Beta probability distribution, Beta function*
A.1 Simple Luhmann Economy Model

A.1.1 Displays of Wealth (Show Off)

Sub showoff(n, m, g, pshowoff, ab, xy)
Dim i, j, k, l, maxa As Integer
For i = 1 To n
    For j = 1 To m
        If Rnd() < pshowoff Then
            maxa = 0
            l = 1
            For k = 1 To g
                If ab(i, j, k) > maxa Then
                    maxa = ab(i, j, k)
                    l = k
                End If
            Next k
            xy(i, j, l) = maxa
        End If
    Next j
Next i
End Sub
### A.1.2 Making Proposals

Making proposals includes the following functions:

- The function `propose` is the main loop through all agents
- The function `proposeij` takes care about the actions of an individual agent identified by row and column index i and j
- The functions `mmprice`, `pricerule` and `showoffrule` contain the respective rules to find price minima and maxima, create proposals following the price rule and for creating proposals according to the showoff rule
- The function `convertproposals` and `convertijproposals` are used to ease implementation and exchange (absolute and relative) addresses of proposer and proposee.

```vba
Sub propose(step, n, m, g, ni, mj, ab, ohorizon, o, xy, p, op, sp)
    Dim i, j, k As Integer
    ReDim own(1 To g) As Long
    op = 0
    sp = 0
    For i = 1 To n
        For j = 1 To m
            For k = 1 To g
                own(k) = ab(i, j, k)
            Next k
            Call proposeij(step, n, m, g, ni, mj, i, j, own, ohorizon, o, xy, p, op, sp)
        Next j
    Next i
End Sub

Sub proposeij(step, n, m, g, oi, oj, i, j, own, ohorizon, o, xy, p, op, sp)
    ' n, m rows, columns
    ' g goods
    ' i, j proposing entity
    ' own portfolio owned
    ' o observations of prior deals
    ' xy showoffs
    ' p result
    Dim k, l, omade As Integer
    Dim smax, bmin As Double
    Dim smaxii, smaxjj As Integer
    Dim bminii, bminjj As Integer
    If ohorizon <= 0 Then GoTo pre_bye
    omade = 0
    For k = 1 To g
        For l = k + 1 To g
            Call mmprice(n, m, g, oi, oj, i, j, k, l, o, bmin, bminii, bminjj, smax, smaxii, smaxjj, omade)
            ' prices found for goods combination k, l
            Call pricerule(step, n, m, g, i, j, k, l, own, bmin, bminii, bminjj, smax, smaxii, smaxjj, p, op, sp, omade)
            ' proposals made
        Next l
    Next k
End Sub
```

---

*Page 3 of 37*
Next l
Next k

pre_bye:
If omade = 0 Then
  ' MsgBox("proposeij> no observations made")
  Call showoffrule(n, m, g, oi, oj, i, j, own, xy, p, sp)
End If
bye:
End Sub

Sub mmprice(n, m, g, oi, oj, i, j, k, l, o, bmin, bminii, bminjj, smax, smaxii, smaxjj, omade)
Dim ii, jj As Integer
Dim oA, oB As Long
smax = 0
bmin = 9E+15
For ii = -oi To oi
  For jj = -oj To oj
    ' exclude self, law of the first distinction
    If ii = 0 And jj = 0 Then GoTo jjloop
    For r = 1 To 2
      ' r = 1 role accepter, 2 role proposer
      ' s = 1, a sale of (k) - i.e. a chance for buying - was observed
      oA = o(i, j, ii, jj, r, 1, k)
      oB = o(i, j, ii, jj, r, 1, l)
      If oA = 0 Or oB = 0 Then GoTo rcontinue
      omade = omade + 1
      If Abs(oB / oA) < bmin Then
        bmin = Abs(oB / oA)
        bminii = ii
        bminjj = jj
      End If
    rcontinue:
      ' r = 2, a purchase of (k) - i.e. a chance for selling - was observed
      oA = o(i, j, ii, jj, r, 2, k)
      oB = o(i, j, ii, jj, r, 2, l)
      If oA = 0 Or oB = 0 Then GoTo rloop
      omade = omade + 1
      If Abs(oB / oA) > smax Then
        smax = Abs(oB / oA)
        smaxii = ii
        smaxjj = jj
      End If
    rloop:
      Next r
  jjloop:
    Next jj
  Next ii
  * smax and bmin identified
End Sub
Sub pricerule(step, n, m, g, i, j, l, own, bmin, bminii, bminjj, smax, smaxii, smaxjj, p, op, sp, omade)

Dim buyA, sellA, B2buyA, B2sellA As Long

If smax > bmin And omade <> 0 Then

    B2buyA = own(l)
    buyA = Int(B2buyA / bmin)
    While B2buyA <> bmin * buyA And B2buyA > 0
        B2buyA = B2buyA - 1
        buyA = Int(B2buyA / bmin)
    Wend

'propose to buy A
If B2buyA > 0 Then
    p(i, j, bminii, bminjj, k) = buyA
    p(i, j, bminii, bminjj, l) = -B2buyA
    op = op + 1
    own(l) = own(l) - B2buyA
End If

sellA = own(k)
B2sellA = Int(smax * sellA)
While B2sellA <> smax * sellA And sellA > 0
    sellA = sellA - 1
    B2sellA = Int(smax * sellA)
    Wend

'propose to sell A
If sellA > 0 Then
    p(i, j, smaxii, smaxjj, k) = -sellA
    p(i, j, smaxii, smaxjj, l) = B2sellA
    op = op + 1
    own(k) = own(k) - sellA
End If

End If
End Sub
Sub showoffrule(n, m, g, oi, oj, i, j, own, xy, p, sp)

    Dim ii, jj, k, l As Integer
    Dim a, pA, pB As Long

    ReDim maxg(1 To g) As Long
    ReDim maxwhor(1 To g) As Integer
    ReDim maxwhoc(1 To g) As Integer

    Dim maxM As Long
    Dim maxMg As Integer

    'ReDim minG(1 To g) As Long
    'ReDim minwhor(1 To g) As Integer
    'ReDim minwhoc(1 To g) As Integer

    For k = 1 To g
        '    minG(k) = 1000000000#
        For ii = -oi To oi
            For jj = -oj To oj
                If ii = 0 And jj = 0 Then GoTo jjloop
                a = xy(torus(i + ii, n), torus(j + jj, m), k)
                If a > maxg(k) Then
                    maxg(k) = a
                    maxwhor(k) = ii
                    maxwhoc(k) = jj
                End If
                jjloop:
            Next jj
        Next ii
    Next k

    For k = 1 To g
        If maxg(k) - own(k) > 2 Then
            maxM = 0
            For l = 1 To g
                If own(l) > maxM And l <> k Then
                    maxM = own(l)
                    maxMg = l
                End If
            Next l
            If maxM > 0 Then
                pA = Int((maxg(k) - own(k)) / 2)
                p(i, j, maxwhor(k), maxwhoc(k), k) = pA
                pB = Int((minG(l) - own(l)) / 2)
                pB = -1
                ' greedy ... no more than 1 or other
                p(i, j, maxwhor(k), maxwhoc(k), maxMg) = pB
                ' do adapt ownership of maxMg not k
                own(maxMg) = own(maxMg) + pB
                sp = sp + 1
            End If
        End If
    Next k
End Sub
Sub convertproposals(n, m, g, ni, mj, z, d)
Dim i, j As Integer
For i = 1 To n
    For j = 1 To m
        Call convertijproposals(n, m, g, ni, mj, i, j, z, d)
    Next j
Next i
End Sub

Sub convertijproposals(n, m, g, ni, mj, i, j, z, ByRef d)
Dim ii, jj, k, l As Integer
Dim v1, v2 As Long
For ii = -ni To ni
    For jj = -mj To mj
        For k = 1 To g
            v1 = z(i, j, ii, jj, k)
            If v1 = 0 Then GoTo kloop
            For l = k + 1 To g
                v2 = z(i, j, ii, jj, l)
                If v2 = 0 Then GoTo lloop
                d(torus(i + ii, n), torus(j + jj, m), -ii, -jj, k) = -v1
                d(torus(i + ii, n), torus(j + jj, m), -ii, -jj, l) = -v2
            Next l
        Next k
    Next jj
Next ii
End Sub
A.1.3 Accepting Deals

Accepting Deals includes the following functions:

- The function `acceptijdeals` is the main function. It loops through all combinations of tradeable goods.
- The functions `bestprice` and `acceptbestdeal` contain the identification of the best prices and the accepting of deals

```vba
Sub acceptijdeals(step, g, i, j, ni, mj, own, d, dd)
    Dim ii, jj, k, l As Integer
    Dim dA, dB As Long
    Dim smax, bmin As Double
    Dim smaxii, s杰xjj As Integer
    Dim bminii, b杰njj As Integer
    For k = 1 To g
        For l = k + 1 To g
            Call bestprice(i, j, ni, mj, k, l, own, d, smax, smaxii, s杰xjj, b杰min, b杰minii, b杰minjj)
                'best price selected
            Call acceptbestdeal(step, i, j, k, l, own, d, smax, s杰maxii, s杰xjj, b杰min, b杰minii, b杰minjj, dd)
                'best deal accepted
        Next l
    Next k
End Sub
```

```vba
Sub bestprice(i, j, ni, mj, k, l, own, d, smax, s杰maxii, s杰xjj, b杰min, b杰minii, b杰minjj)
    Dim ii, jj As Integer
    Dim dA, dB As Long
    smax = 0
    b杰min = 9E+15
    For ii = -ni To ni
        For jj = -mj To mj
            dA = d(i, j, ii, jj, k)
            dB = d(i, j, ii, jj, l)
            If dA = 0 Or dB = 0 Then GoTo jjloop
            If dA < 0 And dB > 0 Then
                If own(k) + dA > 0 And Abs(dB / dA) > smax Then
                    smax = Abs(dB / dA)
                    s杰maxii = ii
                    s杰xjj = jj
                End If
            End If
            If dA > 0 And dB < 0 Then
                If own(l) + dB > 0 And Abs(dB / dA) < b杰min Then
                    b杰min = Abs(dB / dA)
                    b杰minii = ii
                    b杰minjj = jj
                End If
            End If
        Next jj
    Next ii
End Sub
```

Sub acceptbestdeal(step, i, j, k, l, own, d, smax, smaxii, smaxjj, bmin, bminii, bminjj, dd)

Dim qK, qL As Long

If smax = 0 Then
    If bmin < 9E+15 Then
        qK = d(i, j, bminii, bminjj, k)
        qL = d(i, j, bminii, bminjj, l)
        dd(i, j, bminii, bminjj, k) = qK
        dd(i, j, bminii, bminjj, l) = qL
        own(k) = own(k) + qK
        own(l) = own(l) + qL
    End If
    GoTo bye
End If

If bmin >= 9E+15 Then
    If smax > 0 Then
        qK = d(i, j, smaxii, smaxjj, k)
        qL = d(i, j, smaxii, smaxjj, l)
        dd(i, j, smaxii, smaxjj, k) = qK
        dd(i, j, smaxii, smaxjj, l) = qL
        own(k) = own(k) + qK
        own(l) = own(l) + qL
    End If
    GoTo bye
End If

If smax > bmin Then
    'If ni > 1 Then MsgBox("double deal accepted")
    qK = d(i, j, bminii, bminjj, k)
    qL = d(i, j, bminii, bminjj, l)
    dd(i, j, bminii, bminjj, k) = qK
    dd(i, j, bminii, bminjj, l) = qL
    own(k) = own(k) + qK
    own(l) = own(l) + qL
    qK = d(i, j, smaxii, smaxjj, k)
    qL = d(i, j, smaxii, smaxjj, l)
    dd(i, j, smaxii, smaxjj, k) = qK
    dd(i, j, smaxii, smaxjj, l) = qL
    own(k) = own(k) + qK
    own(l) = own(l) + qL
End If

'accepted deals filled to dd
'ownership adapted

bye:
End Sub
A.1.4 Observing Deals

Sub observedeals(n, m, g, oii, ojj, dd, o)
    ' ReDim o(1 To n, 1 To m, -1 To 1, -1 To 1, 1 To 2, 1 To 2, 1 To 2) As Integer
    ' observations( observer(row,column), observee(relrow,relcol), _
    '    role(accepter,proposer), signA(sellA,buyA), scarce good) qty
    Dim i, j, ii, jj, k, l, ni, nj, role, saleA As Integer
    Dim ddA, ddB As Long
    For i = 1 To n
        For j = 1 To m
            '       dd(i,j,...) has accepted deal
            For ii = -oii To oii
                For jj = -ojj To ojj
                    For k = 1 To g
                        ddA = dd(i, j, ii, jj, k)
                        If ddA = 0 Then GoTo kloop
                        For l = k + 1 To g
                            ddB = dd(i, j, ii, jj, l)
                            If ddB = 0 Then GoTo lloop
                            '                       inner llop
                            saleA = 1
                            If ddA > 0 Then saleA = 2
                            '                       saleA=1 it's a sale of A from the viewpoint of the accepter (and
                            '                       proposer as *-1)
                            '                       saleA=2 it's a purchase of A from the viewpoint of the accepter (and
                            '                       proposer as * -1)
                            '                       walk through all neighbours of accepter
                            For ni = -oii To oii
                                For nj = -ojj To ojj
                                    o(torus(i + ni, n), torus(j + nj, m), -ni, -nj, 1, saleA, k) = ddA
                                    o(torus(i + ni, n), torus(j + nj, m), -ni, -nj, 1, saleA, l) = ddB
                                Next nj
                            Next ni
                        Next l
                    Next k
                Next ii
            Next jj
        Next j
    Next i
End Sub
A.1.5 Clearing Deals

The function for clearing (additionally) verifies the bookkeeping (no short selling) rules.

Sub cleardeals(step, n, m, g, oii, ojj, ab, dd, ndeals, flow, fstat, delta)
Dim i, j, k, l, ii, jj As Integer
Dim oA, oB As Long
Dim fA, fB As Double
delta = 0
ndeals = 0
For i = 1 To n
    For j = 1 To m
        For ii = -oii To oii
            For jj = -ojj To ojj
                If ii = 0 And jj = 0 Then GoTo jjloop
                For k = 1 To g
                    oA = dd(i, j, ii, jj, k)
                    If oA <> 0 Then GoTo kbreak
                Next k
                kbreak:
                If oA = 0 Then GoTo jjloop
                For l = k + 1 To g
                    oB = dd(i, j, ii, jj, l)
                    If oB <> 0 Then GoTo lbreak
                Next l
                lbreak:
                If oB = 0 Then GoTo jjloop
            Next jj
        Next ii
    Next j
End For

' Assertions
If ab(i, j, k) + oA < 0 Then
    GoTo jjloop
End If
If ab(torus(i + ii, n), torus(j + jj, m), k) - oA < 0 Then
    GoTo jjloop
End If
If ab(i, j, l) + oB < 0 Then
    GoTo jjloop
End If
If ab(torus(i + ii, n), torus(j + jj, m), l) - oB < 0 Then
    GoTo jjloop
End If

' Inner loop
delta = delta + Abs(oA) + Abs(oB)
ndeals = ndeals + 1
fA = Abs(oA)
fB = Abs(oB)
ab(i, j, k) = ab(i, j, k) + oA
If fstat = 1 Or (fstat = 2 And oA > 0) Or (fstat = 3 And oA < 0) Then
    flow(i, j, k) = flow(i, j, k) + fA
End If
ab(torus(i + ii, n), torus(j + jj, m), k) = ab(torus(i + ii, n), torus(j + jj, m), k) + fA
If fstat = 1 Or (fstat = 2 And oA < 0) Or (fstat = 3 And oA > 0) Then
    flow(torus(i + ii, n), torus(j + jj, m), k) = flow(torus(i + ii, n), torus(j + jj, m), k) + fA
End If
ab(i, j, l) = ab(i, j, l) + oB
If fstat = 1 Or (fstat = 2 And oB > 0) Or (fstat = 3 And oB < 0) Then
    flow(i, j, l) = flow(i, j, l) + fB
End If
ab(torus(i + ii, n), torus(j + jj, m), l) = ab(torus(i + ii, n), torus(j + jj, m), l) + fB
If fstat = 1 Or (fstat = 2 And oB < 0) Or (fstat = 3 And oB > 0) Then
    flow(torus(i + ii, n), torus(j + jj, m), l) = flow(torus(i + ii, n), torus(j + jj, m), l) + fB
Appendix A – VBA Code

End If
jjloop:
    Next jj
    Next ii
Next j
Next i

'If Abs(delta) < 0.00001 Then MsgBox ("cleardeals" + Str(step) + " no delta")

End Sub

A.1.6 Trade Runs

To compute trade runs, the following functions are used:

- The function ngoodsrun takes care of the interface to an .xls spreadsheet. The output depends on the last two parameters. By setting them appropriately the function either delivers stocks, flows, or trade run statistics.
- The function dorus is the main module that guides the calculation of trade runs. The main loop through all iterations given by an input parameter is located in that function.
- To be able to extract detailed information or statistics runs at a later stage (with a later call of the same function) the method of static variables is used.
- The function iteration takes care about one single trade run

Function ngoodsrun(g, numbiter, ohorizon, pshowoff, a As Range, Optional fstat = 0, Optional showstat = 0)
If pshowoff < 0 Or pshowoff > 1 Then
    MsgBox ("wrong showoff probability")
End If
Dim i, j, k, l, ii, jj, n, m As Integer
n = a.Rows.Count
m = a.Columns.Count
If g < 2 Then
    MsgBox ("Anzahl Güter < 2 oder nicht ganzzahlig")
    Exit Function
End If
If Int(n / g) < 1 Then
    MsgBox ("Anzahl Zeilen zu klein")
    Exit Function
End If
If Int(n / g) * Int(g) <> n Then
    MsgBox ("Anzahl Güter " + Str(Int(n / g)) + " inkonsistent zu Zeilenzahl")
    Exit Function
End If
n = Int(n / g)
ReDim ab(1 To n, 1 To m, 1 To g) As Long                                        'stock
ReDim flow(1 To n, 1 To m, 1 To g) As Double                                    'flow
For k = 1 To g
    For i = 1 To n
        For j = 1 To m
            ab(i, j, k) = Int(a((k - 1) * n + i, j).Cells.Value)
        Next j
    Next i
Next k
ii = oh(ohorizon, n)
jj = oh(ohorizon, m)
ReDim z(1 To n, 1 To m, -ii To ii, -jj To jj, 1 To g) As Long  'proposals
ReDim d(1 To n, 1 To m, -ii To ii, -jj To jj, 1 To g) As Long  'proposed deals (converted)
ReDim dd(1 To n, 1 To m, -ii To ii, -jj To jj, 1 To g) As Long  'deals
ReDim o(1 To n, 1 To m, -ii To ii, -jj To jj, 1 To 2, 1 To 2, 1 To g) As Long  'observations

If showstat <> 0 Then GoTo statistics

ReDim xs(1 To 2, 1 To 18) As Double
Call dorun(1, numbiter, n, m, g, ii, jj, pshowoff, ohorizon, ab, z, d, dd, o, flow, fstat, xs)
' display

ReDim s(1 To g * n, 1 To m) As Double
If fstat = 0 Then
For k = 1 To g
  For i = 1 To n
    For j = 1 To m
      s((k - 1) * n + i, j) = ab(i, j, k)
    Next j
  Next i
Next k
Else
  For k = 1 To g
    For i = 1 To n
      For j = 1 To m
        s((k - 1) * n + i, j) = flow(i, j, k)
      Next j
    Next i
  Next k
End If
GoTo bye

statistics:
ReDim s(1 To minlong(numbiter, 5000), 1 To 18) As Double
Call dorun(0, numbiter, n, m, g, ii, jj, pshowoff, ohorizon, ab, z, d, dd, o, flow, fstat, s)

bye:
ngoodsrun = s
End Function

Sub dorun(do_show, numbiter, n, m, g, ii, jj, pshowoff, ohorizon, ab, zz, d, dd, o, flow, fstat, xstat)
  Dim ps As Double  'percentage showoff proposals
  Dim op, sp, ndeals, lnod, nnod As Long
  Dim delta, deltaold, mm, x, y, z As Double
  Dim i, j, k, l As Long
  Static s(1 To 5000, 1 To 18) As Double
  If do_show = 1 Then
    lnod = 0
    mm = 0
    ReDim mab(1 To g) As Double
    For k = 1 To g
      x = gmean(n, m, k, ab)
      mab(k) = x
      mm = mm + x
    Next k
    Call inits(5000, 18, s)
  End If
Dim i, j, k, l As Long
Static s(1 To 5000, 1 To 18) As Double
If do_show = 1 Then
  inod = 0
  mm = 0
  ReDim mab(1 To g) As Double
  For k = 1 To g
    x = gmean(n, m, k, ab)
    mab(k) = x
    mm = mm + x
  Next k
  Call inits(5000, 18, s)
  Dim c, cc As Integer  'cycle detection
  c = 5
  ReDim abold(0 To c, 1 To n, 1 To m, 1 To g) As Double
  ReDim citer(0 To c) As Long
  cc = 0
  Call savesab(n, m, g, ab, abold, cc)
  citer(cc) = 0
' y = doobserveablex(n, m, g, ohorizon, ab, mab) 
' z = doobserveablea(n, m, g, ohorizon, ab) 
' z = dfuzzycluster(n, m, g, 2, 1.5, ab) 

For i = 1 To numbiter 
    Call iteration(i, n, m, g, ii, jj, pshowoff, ohorizon, ab, zz, d, dd, o, op, sp, 
    ndeals, flow, fstat, delta) 
    If Abs(delta) < 0.00001 And Abs(deltaold) < 0.00001 Then 
        MsgBox ("dorun> 2 phase no delta, step: " + Str(i) + " last no deals: " + Str(lnod)) 
        GoTo ibreak 
        deltaold = delta 
        If ndeals = 0 Then lnod = i

    ' cycle check when no deals are made 
    If ndeals = 0 Then 
        nnod = nnod + 1 
        j = cyclefound(n, m, g, ab, abold, citer, c) 
        If j <> 0 Then 
            MsgBox ("dorun> cycle found at " + Str(i + 1) + " same as " + Str(citer(j)) + " last no deals: " + Str(lnod)) 
            GoTo ibreak 
        End If 
        cc = cc + 1 
        If cc > c Then 
            cc = 1 
        End If 
        Call saveab(n, m, g, ab, abold, cc) 
        citer(cc) = i

End If 

' j = i 
' If numbiter > 500 Then 
'    If i <= numbiter - 500 Then 
'        GoTo iloop 
'    Else 
'        j = i - numbiter + 500 
'    End If 
' End If 

' statistics collection 
' only zero deals 
' or 
' j = torus(i, 5000) 
' s(j, 1) = i 
's(j, 2) = dgmean(n, m, ab, mab(1), 1) 
's(j, 3) = dgmean(n, m, ab, mab(2), 2) 
's(j, 4) = dgmean(n, m, ab, mab(3), 3) 

' s(j, 5) = dall(n, m, g, ab, mab) 
's(j, 6) = dweighted(n, m, g, ab, mab, mm) 
's(j, 7) = op 
's(j, 8) = sp 
's(j, 9) = ndeals 

For k = 1 To minlong(3, g) 
    For l = k + 1 To minlong(3, g) 
        If k = l Then GoTo lloop 
        x = maxAprice(n, m, oh(ohorizon, n), oh(ohorizon, m), k, l, dd) 
        y = minAprice(n, m, oh(ohorizon, n), oh(ohorizon, m), k, l, dd) 
        z = avgAprice(n, m, oh(ohorizon, n), oh(ohorizon, m), k, l, dd) 
        MsgBox ("dorun maxminstat> k " + Str(k) + " l " + Str(l) + " max(" + Str(x) + " min(" + Str(y) + " avg(" + Str(z) + "))") 
        s(j, 9 + (k - 1) * 2 + l - k) = x
s(j, 12 + (k - 1) * 2 + 1 - k) = y
s(j, 15 + (k - 1) * 2 + 1 - k) = z

iloop:
    Next l
    Next k

iloop:
    Next i

ibreak:
    MsgBox ("dorun> No deals encountered " + Str(nnod) + " times, last at " + Str(citer(cc)))

' if any inbetween step was reached call last one back
If citer(cc) <> 0 Then Call abback(n, m, g, ab, abold, cc)
Else
    For k = 1 To minlong(5000, numbiter)
        If s(k, 1) = 0 Then GoTo kbreak
    Next k

kbreak:
    If k <= minlong(5000, numbiter) Then
        For i = 1 To k - 1
            For j = 1 To 18
                xstat(i, j) = s(k - i, j)
            Next j
        Next i
    Else
        For i = 1 To minlong(5000, numbiter)
            For j = 1 To 18
                xstat(i, j) = s(minlong(numbiter, 5000) - i + 1, j)
            Next j
        Next i
    End If
End If
End Sub
Appendix A – VBA Code

Sub iteration(i, n, m, g, ni, mj, pshowoff, ohorizon, ByRef ab, ByRef z, ByRef d, ByRef dd, ByRef o, op, sp, ndeals, flow, fstat, delta)
    ReDim xy(1 To n, 1 To m, 1 To g) As Integer
    Call showoff(n, m, g, pshowoff, ab, xy)
    Call initz(n, m, g, ni, mj, z)
    Call propose(i, n, m, g, ni, mj, ab, ohorizon, o, xy, z, op, sp)
    Call initz(n, m, g, ni, mj, d)
    Call convertproposals(n, m, g, ni, mj, z, d)
    Call acceptdeals(i, n, m, g, ni, mj, ab, ohorizon, o, xy, z, d, dd)
    Call inito(n, m, g, ni, mj, o)
    Call observedeals(n, m, g, ni, mj, d, o)
    Call cleardeals(i, n, m, g, ni, mj, ab, dd, ndeals, flow, fstat, delta)
    If i > 171 Then MsgBox ("iteration Z> " + Str(i))
End Sub

Sub acceptdeals(step, n, m, g, ni, mj, ab, z, d, dd)
    Dim i, j, k, ii, jj As Integer
    Dim oA As Long
    ReDim own(1 To g) As Long
    For i = 1 To n
        For j = 1 To m
            For k = 1 To g
                own(k) = ab(i, j, k)
                If own(k) < 0 Then MsgBox ("acceptdeals A step" + Str(step) + "> i=" + Str(i) + "> j=" + Str(j) + "> own(" + Str(k) + ")= " + Str(own(k)))
                Next k
            ' reduce for proposals made
            For ii = -ni To ni
                For jj = -mj To mj
                    For k = 1 To g
                        oA = z(i, j, ii, jj, k)
                        If oA < 0 Then own(k) = own(k) + oA
                    Next k
                Next jj
            Next ii
            For k = 1 To g
                If own(k) < 0 Then MsgBox ("acceptdeals B step" + Str(step) + "> i=" + Str(i) + "> j=" + Str(j) + "> own(" + Str(k) + ")= " + Str(own(k)))
                Next k
            Call acceptijdeals(step, g, i, j, ni, mj, own, d, dd)
            For k = 1 To g
                If own(k) < 0 Then MsgBox ("acceptdeals C step" + Str(step) + "> i=" + Str(i) + "> j=" + Str(j) + "> own(" + Str(k) + ")= " + Str(own(k)))
                Next k
            Next j
        Next j
    Next i
End Sub
A.2  **Fuzzy Clustering**

The following modules implement the fuzzy-c-means clustering algorithm.

A.2.1  **Clustering**

For clustering the following four functions are used:

- The function `showgfuzzycluster` takes care of the interface to an .xls spreadsheet
- The function `fuzzycluster` is the main module that guides the calculation
- The functions `nextu`, `nextv` calculate next approximation steps

```vba
Function showgfuzzycluster(g, ww, nnc, a As Range)
    Dim n, m, i, j, k, nstocks As Integer
    nstocks = Int(g)
    n = a.Rows.Count
    m = a.Columns.Count

    If g < 2 Or nstocks <> g Then
        MsgBox ("Anzahl Güter < 2 oder nicht ganzzahlig")
        Exit Function
    End If
    If Int(n / g) < 1 Then
        MsgBox ("Anzahl Zeilen zu klein")
        Exit Function
    End If
    If Int(n / g) * Int(g) <> n Then
        MsgBox ("Anzahl Güter " + Str(Int(n / g)) + " inkonsistent zu Zeilenzahl")
        Exit Function
    End If
    n = Int(n / g)

    Dim nc As Integer
    nc = Int(nnc.Cells.Value)
    If nnc.Cells.Value <> nc Then
        MsgBox ("Anzahl Cluster nicht ganzzahlig")
        Exit Function
    End If
    If nc < 1 Or nc > n * m Then
        MsgBox ("Anzahl Cluster < 1 oder >" + Str(n * m))
        Exit Function
    End If

    Dim w As Double
    w = ww.Cells.Value
    If w <= 1 Then
        MsgBox ("w muss > 1")
        Exit Function
    End If

    ReDim abc(1 To n, 1 To m, 1 To nstocks) As Double

    For k = 1 To nstocks
        For i = 1 To n
            For j = 1 To m
                abc(i, j, k) = a((k - 1) * n + i, j).Cells.Value
            Next j
        Next i
```
Next k
ReDim c(1 To n, 1 To m, 1 To nc) As Double
'MsgBox ("g cluster> n" + Str(n) + " m" + Str(m) + " g" + Str(nstocks) + " nc" + Str(nc))
Call fuzzycluster(w, n, m, nstocks, nc, abc, c)

show:
ReDim cc(1 To n * nc, 1 To m) As Double
For k = 1 To nc
    For i = 1 To n
        For j = 1 To m
            cc((k - 1) * n + i, j) = c(i, j, k)
        Next j
    Next i
Next k

showgfuzzycluster = cc
End Function
Sub fuzzycluster(ByVal w, n, m, nstocks, nc, ByRef ab, ByRef c)

    Dim i, j, k, l As Integer
    Dim x, xvi As Double
    Dim delta As Double
    delta = 0.0001

    'initialize u(ij)k=c(ij)k
    For i = 1 To n
        For j = 1 To m

            'no zero begin
            xvi = 0
            For l = 1 To nstocks
                xvi = xvi + Abs(ab(i, j, l))
                Next l
            If xvi = 0 Then GoTo jloop

            'no zero end
            xvi = 0
            For k = 1 To nc - 1
                x = Rnd()
                If xvi + x < 1 Then
                    c(i, j, k) = x
                Else
                    c(i, j, k) = 1 - xvi
                End If
                xvi = xvi + c(i, j, k)
            Next k
            c(i, j, nc) = 1 - xvi

            'smooth
            xvi = 0
            For k = 1 To nc
                If c(i, j, k) < 1 / (1.5 * nc) Then
                    c(i, j, k) = 1 / (1.5 * nc)
                xvi = xvi + c(i, j, k)
            Next k
            For k = 1 To nc
                c(i, j, k) = c(i, j, k) / xvi
            Next k

            jloop:
            Next j
        Next i

        'cluster centers
        ReDim v(1 To nc, 1 To nstocks) As Double
        Dim deltav As Double
deltav = 0
l = 0
Call nextv(w, n, m, nc, nstocks, ab, c, v, deltav)

While deltav > delta And l < 300
    l = l + 1
    Call nextu(w, n, m, nc, nstocks, ab, c, v)
    Call nextv(w, n, m, nc, nstocks, ab, c, v, deltav)
Wend

If deltav > delta Then
    MsgBox ("fuzzycluster> bad convergence, deltav: " + Str(deltav))
    GoTo bye
End If

    'if zeroes excluded
    If xvi <> 0 And Abs(xvi - 1) > delta Then
        MsgBox ("fuzzycluster> restriction violated xvi: " + Str(xvi) + " i: " + Str(i) + " j: " + Str(j))
    End If

Next j
Next i
bye:
End Sub
Sub nextv(ByVal w, n, m, nc, nstocks, ByRef ab, ByRef c, ByRef v, ByRef deltav)
On Error GoTo Sorry

Dim eps As Double
eps = 0.000000000000001

Dim k, l, ll As Integer
Dim void, vnew, sux, su, xvi As Double
Dim i, j As Integer
deltav = 0
For k = 1 To nc
    For l = 1 To nstocks
        MsgBox("nextv> cluster" + Str(k) + " stock" + Str(l))
        sux = 0
        su = 0
        MsgBox("nextv> cluster" + Str(k) + " stock" + Str(l) + " sux,su init")
        MsgBox("nextv> again n" + Str(n) + " m" + Str(m))
        For i = 1 To n
            MsgBox("nextv cluster" * Str(k) + " stock" + Str(l) + " i" + Str(i) + " i pre xvi")
            For j = 1 To m
                MsgBox("nextv cluster" * Str(k) + " stock" + Str(l) + " i" + Str(i) + " j" + Str(j) + " pre xvi")
                xvi = 0
                For ll = 1 To nstocks
                    xvi = xvi + ab(i, j, ll)
                Next ll
                If Abs(xvi) > eps Then
                    MsgBox("nextv cluster" * Str(k) + " stock" + Str(l) + " i" + Str(i) + " j" + Str(j) + " completed")
            Next j
        Next i
        MsgBox("nextv cluster" + Str(k) + " stock" + Str(l) + " pre completion")
        If Abs(su) > eps Then
            vnew = sux / su
            void = v(k, l)
            deltav = deltav + Abs(void - vnew)
            v(k, l) = vnew
        Else
            MsgBox("nextv> su=0")
            GoTo bye
        End If
    Next l
Next k
bye:
Exit Sub
Sorry:
If Err.Number = 6 Then
    Resume Next
Else
    MsgBox("nextv> " & Err.Number & vbCrLf & vbCrLf & Err.Description)
End If
End Sub
Sub nextu(ByVal w, n, m, nc, nstocks, ByRef ab, ByRef c, ByRef v)
On Error GoTo Sorry

Dim eps As Double
eps = 0.000000000000001

Dim i, j, k, l, ll As Integer
Dim xv, xvi, xvj, x, e As Double
ReDim singularity(1 To n, 1 To m) As Boolean
For i = 1 To n
    For j = 1 To m
        singularity(i, j) = False
    Next j
Next i
Dim nsing As Integer

For ll = 1 To nc
    ' for all i<=>ll among clusters seek uik
    For i = 1 To n
        For j = 1 To m
            ' no zero begin
            xvi = 0
            For l = 1 To nstocks
                xvi = xvi + Abs(ab(i, j, l))
            Next l
            If xvi = 0 Then GoTo jloop
            ' no zero end
            '           for all k<=>i,j
            Among data seek uik
            xvi = 0
            For l = 1 To nstocks
                x = ab(i, j, l) - v(ll, l)
                xvi = xvi + x * x
            Next l
            xv = 0
            For k = 1 To nc
                xvj = 0
                For l = 1 To nstocks
                    x = ab(i, j, l) - v(k, l)
                    xvj = xvj + x * x
                Next l
                If Abs(xvj) > eps Then
                    x = 0
                    x = xvi / xvj
                    e = 1 / (2 * (w - 1))
                    If Abs(x) > eps Then
                        xv = xv + Exp(e * Log(x))
                    Else
                        MsgBox ("nextu> xv = 0")
                        GoTo bye
                    End If
                    c(i, j, ll) = 1 / xv
                    MsgBox ("nextu> c(i, j, ll) = 1 / xv")
                Else
                    C(i, j, ll) = 0
                    singularity(i, j) = True
                End If
            Next k
            '           new uik computed
            For ll = 1 To nc
                If Abs(xv) > eps Then
                    c(i, j, ll) = 1 / xv
                    MsgBox ("nextu> c(i, j, ll) = 1 / xv")
                Else
                    c(i, j, ll) = 0
                    singularity(i, j) = True
                End If
            Next ll
        Next j
    Next i
Next ll

Next ll
'repair singularities
For i = 1 To n
    For j = 1 To m
        If singularity(i, j) Then
            MsgBox ("repair i" + Str(i) + " j" + Str(j))
            xvi = 0
            nsing = 0
            For k = 1 To nc
                If Abs(c(i, j, k)) <= eps Then
                    nsing = nsing + 1
                Else
                    xvi = xvi + c(i, j, k)
                End If
            Next k
            k = 1
            l = 1
            While k < nsing
                If c(i, j, l) = 0 Then
                    k = k + 1
                Else
                    x = Rnd()
                    x = 2 * Rnd() / nc
                    If xvi + x < 1 Then
                        c(i, j, l) = x
                    Else
                        c(i, j, l) = 1 - xvi
                    End If
                    xvi = xvi + c(i, j, l)
            End If
            l = l + 1
            While l <= nc
                If c(i, j, l) = 0 Then
                    c(i, j, l) = 1 - xvi
                    GoTo jloop2
                End If
                l = l + 1
            Wend
        End If
    Next j
Next i
bye:
Exit Sub
Sorry:
If Err.Number = 6 Then
    Resume Next
Else
    MsgBox "nextu> " & Err.Number & vbCrLf & vbCrLf & Err.Description
End If
End Sub
A.2.2 Visualisation (Conditional Formatting)

To aid the visualisation of clusters the following functions are used:

- The function `showgcenter` takes care of the interface to an .xls spreadsheet
- The function `fuzzyclustercenter` computes the cluster centres

```vba
Function showgcenter(nstocks, ww, nnc, d As Range, a As Range)
Dim n, n2, m, i, j, k, l, g As Integer
Dim eps As Double
eps = 0.00001

    g = Int(nstocks.Cells.Value)
    n = a.Rows.Count
    n2 = d.Rows.Count
    m = a.Columns.Count
    If m <> d.Columns.Count Then
        MsgBox("showgcenter> Anzahl Spalten stimmen nicht überein")
        Exit Function
    End If
    If Int(n2 / g) < 1 Then
        MsgBox("Anzahl (Daten) Zeilen zu klein")
        Exit Function
    End If
    If Int(n2 / g) * Int(g) <> n2 Then
        MsgBox("Anzahl Güter " + Str(Int(n / g)) + " inkonsistent zu Zeilenzahl Daten")
        Exit Function
    End If
    n2 = Int(n2 / g)
    If g < 2 Then
        MsgBox("Anzahl Güter < 2 oder nicht ganzzahlig")
        Exit Function
    End If
    Dim nc As Integer
    nc = Int(nnc.Cells.Value)
    If nnc.Cells.Value <> nc Then
        MsgBox("Anzahl Cluster nicht ganzzahlig")
        Exit Function
    End If
    If nc < 1 Or nc > n * m Then
        MsgBox("Anzahl Cluster < 1 oder >" + Str(n * m))
        Exit Function
    End If
    If Int(n / nc) < 1 Then
        MsgBox("Anzahl (Cluster) Zeilen zu klein")
        Exit Function
    End If
    If Int(n / nc) * Int(nc) <> n Then
        MsgBox("Anzahl Cluster " + Str(Int(n / nc)) + " inkonsistent zu Zeilenzahl Cluster")
        Exit Function
    End If
    n = Int(n / nc)

Dim w As Double
w = ww.Cells.Value
If w <= 1 Then
    MsgBox("w muss > 1")
    Exit Function
End If
End Function
```
If n <> n2 Then
  MsgBox ("showgcenter> Zeilen Daten und Zeilen Cluster inkonsistent")
  Exit Function
End If

' MsgBox ("showgcenter> n=" + Str(n) + ", m=" + Str(m) + ", nc=" + Str(nc) + ", g=" + Str(g))

ReDim c(1 To n, 1 To m, 1 To nc) As Double
For k = 1 To nc
  For i = 1 To n
    For j = 1 To m
      c(i, j, k) = a((k - 1) * n + i, j).Cells.Value
    Next j
  Next i
Next k

ReDim abc(1 To n, 1 To m, 1 To g) As Double
For k = 1 To g
  For i = 1 To n
    For j = 1 To m
      ' MsgBox ("showgcenter> step A.(" + Str(i) + "," + Str(j) + ")")
      abc(i, j, k) = d((k - 1) * n + i, j).Cells.Value
    Next j
  Next i
Next k

ReDim cc(1 To nc, 1 To g) As Double
Call fuzzyclustercenter(w, n, m, g, nc, abc, c, cc)
showgcenter = cc
Exit Function
End Function

Sub fuzzyclustercenter(w, n, m, g, nc, abc, c, cc)
  Dim i, j, k As Integer
  Dim x, mu, mm, eps As Double
  eps = 0.000001
  For i = 1 To n
    For j = 1 To m
      x = 0
      For k = 1 To nc
        x = x + c(i, j, k)
      Next k
      If Abs(x - 1) > eps Then
        ' MsgBox ("showgcenter> exclusion at x(" + Str(i) + "," + Str(j) + ")=" + Str(x))
        ' Exit Function
      End If
    Next j
  Next i

  For k = 1 To nc
    For l = 1 To g
      mu = 0
      mm = 0
      For i = 1 To n
        For j = 1 To m
          x = c(i, j, k) ^ w
          mu = mu + x * abc(i, j, l)
          mm = mm + x
        Next j
      Next i
      If Abs(mm) > eps Then cc(k, l) = mu / mm
    Next l
    Next k
  End Sub
A.3 Exploration

The following modules were used for the exploration of the model

A.3.1 Simulation Runs

The following functions are used to generate initial distributions of wealth and to obtain statistics regarding the overall behaviour.

In the main function `nsimrun` the same technique (as in `dorun`) of using static variables to extract further details by a second call is applied.

```vba
Sub newab(g, n, m, a, b, c, a1, a2, b1, b2, c1, c2, ab)
    Dim i, j, k, l As Integer
    For i = 1 To n
        For j = 1 To m
            l = 1
            If g >= 3 Then
                ab(i, j, 1) = Int(Application.WorksheetFunction.BetaInv(Rnd(), a1, a2, 0, a))
                ab(i, j, 2) = Int(Application.WorksheetFunction.BetaInv(Rnd(), b1, b2, 0, b))
                ab(i, j, 3) = Int(Application.WorksheetFunction.BetaInv(Rnd(), c1, c2, 0, c))
                l = 4
            End If
            For k = l To g
                ab(i, j, k) = Int(Rnd() * a)
            Next k
        Next j
    Next i
End Sub
```
Function nsimrun(Optional r = 10, Optional g = 3, Optional n = 6, Optional m = 6, Optional a = 7, Optional b = 11, Optional c = 5, Optional a1 = 1, Optional a2 = 1, Optional b1 = 2, Optional b2 = 1, Optional c1 = 1, Optional c2 = 2, Optional save = 0)
Dim i, i2, j, k, l, ii, jj As Integer
Dim pshowoff As Double
pshowoff = 1
Dim ohorizon As Integer
ohorizon = 1
ii = oh(ohorizon, n)
jj = oh(ohorizon, m)
ReDim z(1 To n, 1 To m, -ii To ii, -jj To jj, 1 To g) As Long  'proposals
ReDim d(1 To n, 1 To m, -ii To ii, -jj To jj, 1 To g) As Long  'proposed deals (converted)
ReDim dd(1 To n, 1 To m, -ii To ii, -jj To jj, 1 To g) As Long  'deals
ReDim o(1 To n, 1 To m, -ii To ii, -jj To jj, 1 To 2, 1 To 2, 1 To g) As Long  'observations
ReDim ab(1 To n, 1 To m, 1 To g) As Long  'stock
Static absave(1 To 3, 1 To 6, 1 To 6, 1 To 3) As Long  'stock save
ReDim flow(1 To n, 1 To m, 1 To g) As Double  'flow
Dim ns As Integer
ns = 10
ReDim s(1 To r, 1 To ns) As Double  'simulation statistics
ReDim xs(1 To ns) As Double
Dim x As Double
If save <> 0 Then
    ReDim s(1 To g * n, 1 To m) As Double
    For k = 1 To g
        For i = 1 To n
            For j = 1 To m
                s((k - 1) * n + i, j) = absave(1, i, j, k)
                If save = 2 Then s((k - 1) * n + i, j) = absave(2, i, j, k)
            Next j
        Next i
    Next k
    GoTo bye
End If
For i = 1 To r
    Call newab(g, n, m, a, b, c, a1, a2, b1, b2, c1, c2, ab)
    For k = 1 To g
        x = 0
        For i2 = 1 To n
            For j = 1 To m
                x = x + ab(i2, j, k)
            Next j
        Next i2
        If k <= ns - 7 Then s(i, 7 + k) = x
    Next k
    Call simrun(2000, n, m, g, ii, jj, pshowoff, ohorizon, ab, z, d, dd, o, flow, xs)
    s(i, 1) = i
    For j = 2 To 7
        s(i, j) = xs(j)
    Next j
    If s(i, 2) > 2000 Then
        MsgBox ">2000 ..." + Str(absave(1, 6, 6, 3)) + "..2.." + Str(absave(3, 6, 6, 3))
    End If
    If s(i, 5) = 2 Then
        MsgBox ">cycle 2..." + Str(absave(2, 6, 6, 3)) + "..2.." + Str(absave(3, 6, 6, 3))
    End If
Next i
bye:
nsimrun = s
End Function
Sub simrun(numbiter, n, m, g, ii, jj, pshowoff, ohorizon, ab, zz, d, dd, o, flow, s)

    Dim ps As Double   'percentage showoff proposals
    Dim op, sp, ndeals, lnod, nnod As Long
    Dim delta, deltaold, mm, x, y, z As Double
    Dim i, j, k, l As Long

    lnod = 0
    mm = 0
    ReDim mab(1 To g) As Double
    For k = 1 To g
        x = gmean(n, m, k, ab)
        mab(k) = x
        mm = mm + x
    Next k

    Dim c, cc As Integer    'cycle detection
    c = 20  'depth of comparison
    ReDim abold(0 To c, 1 To n, 1 To m, 1 To g) As Double
    ReDim citer(0 To c) As Long
    cc = 0
    Call saveab(n, m, g, ab, abold, cc)
    citer(cc) = 0
    For i = 1 To numbiter
        Call iteration(i, n, m, g, ii, jj, pshowoff, ohorizon, ab, zz, d, dd, o, op, sp, ndeals, flow, fstat, delta)
        If Abs(delta) < 0.00001 And Abs(deltaold) < 0.00001 Then
            s(2) = i
            s(3) = lnod
            s(4) = 0
            s(5) = 0
            GoTo ibreak
        End If
        deltaold = delta
        If ndeals = 0 Then lnod = i
            nnod = nnod + 1
            j = cyclefound(n, m, g, ab, abold, citer, c)
            If j <> 0 Then
                s(2) = i
                s(3) = lnod
                s(4) = citer(j)
                If j <= cc Then
                    s(5) = cc - j
                Else
                    s(5) = c + cc - j
                End If
                GoTo ibreak
            End If
            cc = cc + 1
            If cc > c Then
                cc = 1
            End If
            If cc > c Then
                cc = 1
            End If
            Call saveab(n, m, g, ab, abold, cc)
            citer(cc) = i
        End If
    Next i
    ibreak:
    s(6) = nnod
    s(7) = citer(cc)
    If i > numbiter Then
        s(2) = i
        s(3) = lnod
        s(4) = 0
        s(5) = 0
    End If
End Sub
A.3.2 Model (Trade Run) Statistics

The following functions are used during trade runs, to compute and collect various statistics, e.g. average prices, number of deals, exchanged quantities etc.

Function qtyijAbuys(oii, ojj, i, j, dd)
Dim ii, jj As Integer
Dim noA As Double
For ii = -oii To oii
    For jj = -ojj To ojj
        If dd(i, j, ii, jj, 1) > 0 Then noA = noA + dd(i, j, ii, jj, 1)
    Next jj
Next ii
qtyijAbuys = noA
End Function

Function qtyijAsales(oii, ojj, i, j, dd)
Dim ii, jj As Integer
Dim noA As Double
For ii = -oii To oii
    For jj = -ojj To ojj
        If dd(i, j, ii, jj, 1) < 0 Then
            noA = noA - dd(i, j, ii, jj, 1)
        End If
    Next jj
Next ii
qtyijAsales = noA
End Function

Function qtyAsales(n, m, oii, ojj, dd)
Dim i, j, ii, jj As Integer
Dim noA As Double
For i = 1 To n
    For j = 1 To m
        For ii = -oii To oii
            For jj = -ojj To ojj
                If dd(i, j, ii, jj, 1) < 0 Then
                    noA = noA - dd(i, j, ii, jj, 1)
                End If
            Next jj
        Next ii
    Next j
Next i
qtyAsales = noA
End Function
Function qtyAbuys(n, m, oii, ojj, dd)
    Dim i, j, ii, jj As Integer
    Dim noA As Double
    For i = 1 To n
        For j = 1 To m
            For ii = -oii To oii
                For jj = -ojj To ojj
                    If dd(i, j, ii, jj, 1) > 0 Then
                        noA = noA + dd(i, j, ii, jj, 1)
                    End If
                Next jj
            Next ii
        Next j
    Next i
    qtyAbuys = noA
End Function

Function noAbuys(n, m, oii, ojj, dd)
    Dim i, j, ii, jj, noA As Integer
    For i = 1 To n
        For j = 1 To m
            For ii = -oii To oii
                For jj = -ojj To ojj
                    If dd(i, j, ii, jj, 1) > 0 Then noA = noA + 1
                Next jj
            Next ii
        Next j
    Next i
    noAbuys = noA
End Function

Function noAsales(n, m, oii, ojj, dd)
    Dim i, j, ii, jj, k, noA As Integer
    For i = 1 To n
        For j = 1 To m
            For ii = -oii To oii
                For jj = -ojj To ojj
                    If dd(i, j, ii, jj, 1) < 0 Then noA = noA + 1
                Next jj
            Next ii
        Next j
    Next i
    noAsales = noA
End Function
Function minAprice(n, m, oii, ojj, k, l, dd)
Dim i, j, ii, jj, a As Integer
Dim b, bmin As Double
bmin = 9E+15
For i = 1 To n
    For j = 1 To m
        For ii = -oii To oii
            For jj = -ojj To ojj
                a = dd(i, j, ii, jj, k)
                If a <> 0 Then
                    b = Abs(dd(i, j, ii, jj, l) / a)
                    If b <> 0 And b < bmin Then bmin = b
                End If
            Next jj
        Next ii
    Next j
Next i
If bmin >= 9E+15 Then bmin = 0
minAprice = bmin
End Function

Function maxAprice(n, m, oii, ojj, k, l, dd)
Dim i, j, ii, jj, a As Integer
Dim b, bmax As Double
bmax = 0
For i = 1 To n
    For j = 1 To m
        For ii = -oii To oii
            For jj = -ojj To ojj
                a = dd(i, j, ii, jj, k)
                If a <> 0 Then
                    b = Abs(dd(i, j, ii, jj, l) / a)
                    If b <> 0 And b > bmax Then bmax = b
                End If
            Next jj
        Next ii
    Next j
Next i
maxAprice = bmax
End Function

Function avgAprice(n, m, oii, ojj, k, l, dd)
Dim i, j, ii, jj As Integer
Dim a, ak, b, bl, aprice As Double
For i = 1 To n
    For j = 1 To m
        For ii = -oii To oii
            For jj = -ojj To ojj
                ak = Abs(dd(i, j, ii, jj, k))
                bl = Abs(dd(i, j, ii, jj, l))
                If ak <> 0 And bl <> 0 Then
                    a = a + ak
                    b = b + bl
                End If
            Next jj
        Next ii
    Next j
Next i
If a <> 0 Then aprice = b / a
avgAprice = aprice
End Function
A.3.3 Finding Cycles

Function cyclefound(n, m, g, ab, abold, citer, c)
Dim cc, j As Integer
'MsgBox ("cyclefound> start")
j = 0
For cc = 0 To c
  If citer(cc) <> 0 Then
    If cyclefoundcc(n, m, g, ab, abold, citer, c, cc) Then
      j = cc
      GoTo ccbreak
    End If
  End If
Next cc
ccbreak:
cyclefound = j
'MsgBox ("cyclefound> passed")
End Function

Function cyclefoundcc(n, m, g, ab, abold, citer, c, cc)
Dim i, j, k As Integer
Dim starti, startj As Integer
Dim startok As Boolean
Dim a, maxg As Long
'MsgBox ("cyclefoundcc> start")
starti = 0
startj = 0
startok = False
For i = 1 To n
  For j = 1 To m
    startok = True
    maxg = 0
    For k = 1 To g
      a = ab(i, j, k)
      If a > maxg Then maxg = a
      If a <> abold(cc, i, j, k) Then startok = False
    Next k
    If startok Then
      If maxg = 0 Then GoTo jcontinue
      starti = i - 1
      startj = j - 1
      GoTo ibreak
    End If
  Next j
jcontinue:
Next i
'MsgBox ("cyclefoundcc> no startij identified")
GoTo bye
ibreak:
'assert.debug startok = True
'MsgBox ("cyclefoundcc> startij identified" + Str(starti) + ", " + Str(startj))
For i = 1 To n
  For j = 1 To m
    For k = 1 To g
      'MsgBox ("cyclefoundcc> run" + Str(i) + ", " + Str(j) + ", " + Str(k))
      If ab(torus(starti + i, n), torus(startj + j, m), k) <> abold(cc, i, j, k) Then
        'MsgBox ("cyclefoundcc> inequality identified")
        startok = False
      GoTo bye
    End If
  Next j
Next i
GoTo bye
Appendix A – VBA Code

Next k
   Next j
Next i
bye:
cyclefoundcc = startok
'MsgBox ("cyclefoundcc> passed")
End Function

A.3.4 Computing Distances

Function dobserveablex(n, m, g, ohorizon, ab, mab)
   Dim i, ii, j, jj, k As Integer
   Dim x, y, z, mk, mm As Double
   ReDim xy(1 To n, 1 To m, 1 To g) As Long
   Call showoff(n, m, g, 1, ab, xy)
   mm = 0
   For k = 1 To g
      mm = mm + mab(k)
   Next k
   y = 0
   For i = 1 To n
      For j = 1 To m
         For k = 1 To g
            z = ab(i, j, k)
            mk = mab(k)
            For ii = -oh(ohorizon, n) To oh(ohorizon, n)
               For jj = -oh(ohorizon, m) To oh(ohorizon, m)
                  If ii = 0 And jj = 0 Then GoTo jjloop
                  x = ab(torus(i + ii, n), torus(j + jj, m), k)
                  If x <> 0 Then y = y + Abs(z - x) * mk
               jjloop:
               Next jj
            Next ii
         Next k
      Next j
   Next i
   x = (2 * oh(ohorizon, n) + 1) * (2 * oh(ohorizon, m) + 1) - 1
   dobserveablex = y / (n * m * x * mm)
End Function
Function dfuzzycluster(n, m, g, nc, w, ab)

ReDim c(1 To n, 1 To m, 1 To nc) As Double
ReDim cc(1 To nc, 1 To g) As Double
ReDim x(1 To nc) As Double
Dim i, j As Integer
Dim y As Double
Call fuzzycluster(w, n, m, g, nc, ab, c)
Call fuzzyclustercenter(w, n, m, g, nc, ab, c, cc)
For i = 1 To nc
    x(i) = 0
    For j = 1 To g
        y = y + cc(i, j) ^ 2
    Next j
    x(i) = y ^ 0.5
Next i
y = 0
For i = 1 To nc - 1
    y = y + Abs(x(i) - x(i + 1))
Next i
dfuzzycluster = y
End Function

Function gmean(n, m, k, ab)
Dim i, j, a As Integer
Dim x, y As Double
x = 0
a = 0
For i = 1 To n
    For j = 1 To m
        y = ab(i, j, k)
        If y <= 0 Then GoTo jloop
        x = x + y
        a = a + 1
    jloop:
    Next j
Next i
If a > 0 Then gmean = x / a
End Function

Function dgmean(n, m, ab, mm, k)
Dim i, j, a As Integer
Dim x, y As Double
x = 0
y = 0
For i = 1 To n
    For j = 1 To m
        x = ab(i, j, k)
        If x <= 0 Then GoTo jloop
        y = y + Abs(mm - x)
        a = a + 1
    jloop:
    Next j
Next i
If a > 0 Then dgmean = y / a
End Function
Function dall(n, m, g, ab, mab)
Dim i, j, k As Integer
Dim x, y As Double
x = 0
For i = 1 To n
    y = 0
    For j = 1 To m
        y = y + Abs(ab(i, j, k) - mab(k))
        Next k
    x = x + y
    Next j
Next i
dall = x / (n * m)
End Function

Function dweighted(n, m, g, ab, mab, mm)
Dim i, j, k As Integer
Dim x, y As Double
x = 0
For i = 1 To n
    y = 0
    For j = 1 To m
        y = y + Abs(ab(i, j, k) - mab(k)) * mab(k)
        Next k
    x = x + y / mm
    Next j
Next i
dweighted = x / (n * m)
End Function

Function dobserveablea(n, m, g, ohorizon, ab)
Dim i, ii, j, jj, k As Integer
Dim x, y, z As Double
y = 0
For i = 1 To n
    y = 0
    For j = 1 To m
        For k = 1 To g
            z = ab(i, j, k)
            For ii = -oh(ohorizon, n) To oh(ohorizon, n)
                For jj = -oh(ohorizon, m) To oh(ohorizon, m)
                    If ii = 0 And jj = 0 Then GoTo jjloop
                    y = y + Abs(z - ab(torus(i + ii, n), torus(j + jj, m), k))
                jjloop:
                Next jj
            Next ii
        Next k
    Next j
Next i
x = (2 * oh(ohorizon, n) + 1) * (2 * oh(ohorizon, m) + 1) - 1
dobserveablea = y / (n * m * g * x)
End Function
A.4 Auxiliary Functions

Function wealth(x, n)
Dim y As Double
y = x * n
wealth = Int(y + 0.49)
End Function

Sub saveab(n, m, g, ab, abold, cc)
Dim i, j, k As Integer
For i = 1 To n
    For j = 1 To m
        For k = 1 To g
            abold(cc, i, j, k) = ab(i, j, k)
        Next k
    Next j
Next i
End Sub

Sub abback(n, m, g, ab, abold, cc)
Dim i, j, k As Integer
For i = 1 To n
    For j = 1 To m
        For k = 1 To g
            ab(i, j, k) = abold(cc, i, j, k)
        Next k
    Next j
Next i
End Sub

Function ldistg(g, p)
Dim x As Double
Dim i, j As Integer
i = p
x = Log(g)
j = g
While j > 1 And i > 1
    j = j - 1
    i = i - 1
    x = x + Log(j)
Wend
ldistg = x
End Function

Function torus(ByVal x, n)
While x < 1
    x = x + n
Wend
While x > n
    x = x - n
Wend
torus = x
End Function
Function oh(ByVal x, n)
    If x < 1 Then x = 1
    While (2 * x) > (n - 1) And x > 1
        x = x - 1
    Wend
    oh = x
End Function

Sub initz(n, m, g, ni, mj, ByRef z)
    Dim i, j, ii, jj, k As Integer
    For i = 1 To n
        For j = 1 To m
            For ii = -ni To ni
                For jj = -mj To mj
                    For k = 1 To g
                        z(i, j, ii, jj, k) = 0
                    Next k
                Next jj
            Next ii
        Next j
    Next i
End Sub

Sub inito(n, m, g, ni, mj, ByRef o)
    Dim i, j, ii, jj, r, s, k As Integer
    For i = 1 To n
        For j = 1 To m
            For ii = -ni To ni
                For jj = -mj To mj
                    For r = 1 To 2
                        For s = 1 To 2
                            For k = 1 To g
                                o(i, j, ii, jj, r, s, k) = 0
                            Next k
                        Next s
                    Next r
                Next jj
            Next ii
        Next j
    Next i
End Sub

Sub inits(a, b, s)
    Dim i, j As Integer
    For i = 1 To a
        For j = 1 To b
            s(i, j) = 0
        Next j
    Next i
End Sub

Function minlong(a, b)
    Dim x As Long
    x = a
    If b < x Then x = b
    minlong = x
End Function
Sub moveab(n, m, g, abold, f, t)
    Dim i, j, k As Integer
    For i = 1 To n
        For j = 1 To m
            For k = 1 To g
                abold(t, i, j, k) = abold(f, i, j, k)
            Next k
        Next j
    Next i
End Sub