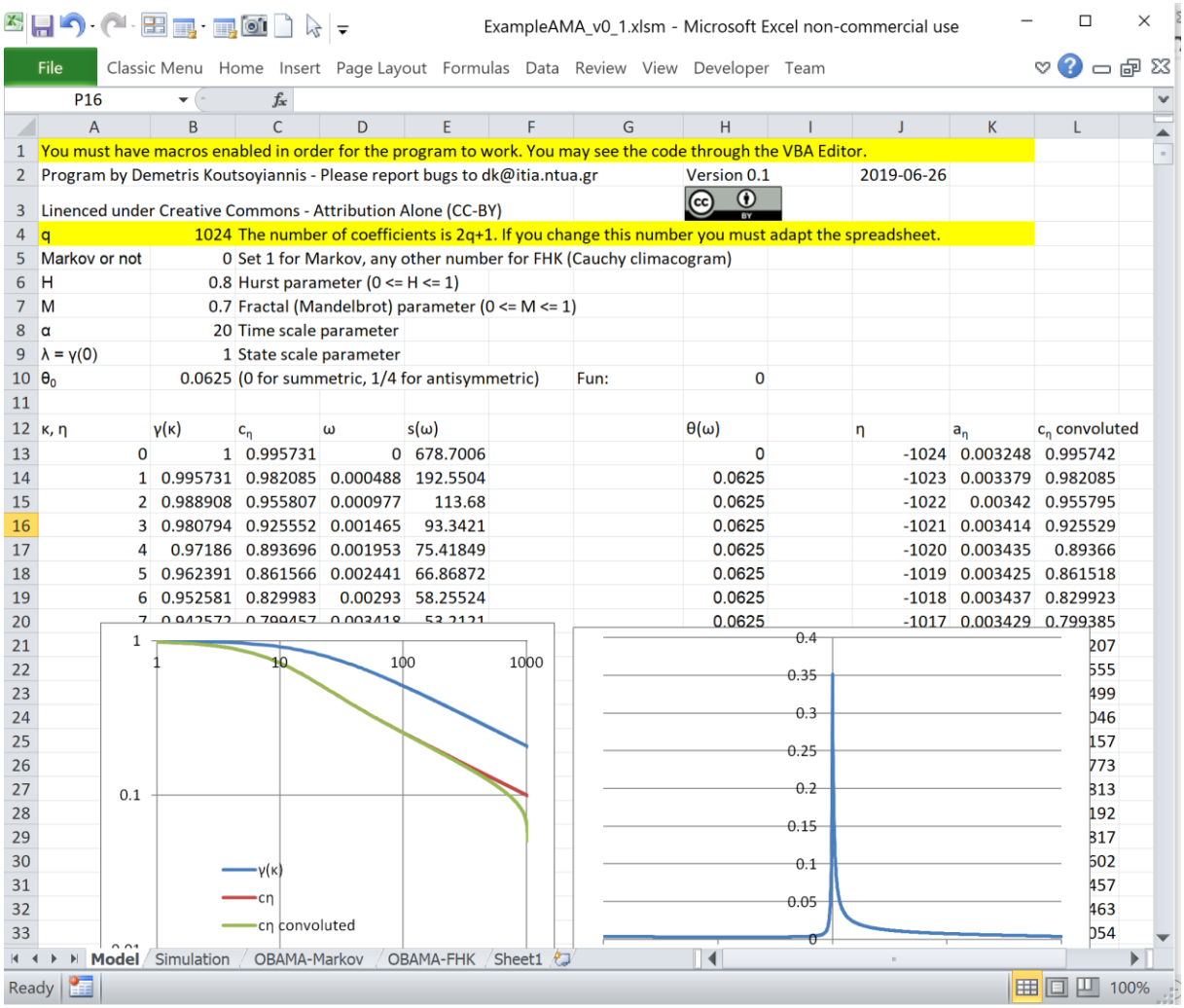
Simple stochastic simulation of time irreversible and reversible processes

Demetris Koutsoyiannis
School of Civil Engineering, National Technical University of Athens, Zographou, Greece
(dk@itia.ntua.gr, http://www.itia.ntua.gr/dk)

Supplementary information

Please download the Excel file from: http://www.itia.ntua.gr/1975/
It contains the required code and examples. A screenshot is shown below. The code in VBA follows but it is better viewed and edited in the Excel environment (VBA editor).

Please report bugs to dk@itia.ntua.gr



Module Models

```
Option Explicit
Option Base 1

Function FHK_Cauchy_(n, M, H, a, la)
Dim i
ReDim List(0 To 2, 0 To n + 1)
For i = 0 To n + 1
  List(0, i) = i
  List(1, i) = la * (1 + (i / a) ^ (2 * M)) ^ ((H - 1) / M)
  Next i
List(2, 0) = List(1, 1)
For i = 1 To n
  List(2, i) = ((i - 1) ^ 2 * List(1, i - 1) + (i + 1) ^ 2 * List(1, i +
1)) / 2 - (i) ^ 2 * List(1, i)
  Next i
ReDim Preserve List(0 To 2, 0 To n)
FHK_Cauchy_ = Application.WorksheetFunction.Transpose(List)
End Function

Function Markov_(n, a, la)
Dim i
ReDim List(0 To 2, 0 To n + 1)
For i = 0 To n + 1
  List(0, i) = i
  If i = 0 Then
    List(1, i) = la
  Else
    List(1, i) = 2 * la * a / i * (1 - (1 - Exp(-i / a)) * a / i)
    End If
  Next i
List(2, 0) = List(1, 1)
For i = 1 To n
  List(2, i) = ((i - 1) ^ 2 * List(1, i - 1) + (i + 1) ^ 2 * List(1, i +
1)) / 2 - (i) ^ 2 * List(1, i)
  Next i
ReDim Preserve List(0 To 2, 0 To n)
Markov_ = Application.WorksheetFunction.Transpose(List)
End Function

Function up2down_(a)
Dim i As Integer, n As Integer, b
n = a.Count
eDim b(n)
For i = 1 To n
  b(i) = a(n - i + 1)
  Next i
up2down_ = Application.WorksheetFunction.Transpose(b)
End Function
```

Module Spectra

```
Option Base 0

Private Sub DFT__(f(), t)
Pi = 4 * Atn(1)
n = UBound(f()) + 1
ReDim df(n, 1)
w = 2 * Pi / n
For k = 0 To n - 1
  For M = 0 To n - 1
    c = Cos(k * w * M)
    S = Sin(k * w * M) * t
    df(k, 0) = df(k, 0) + f(M, 0) * c + f(M, 1) * S
    df(k, 1) = df(k, 1) + f(M, 1) * c - f(M, 0) * S
  Next M
  If t = 1 Then
    df(k, 0) = df(k, 0) / n
    df(k, 1) = df(k, 1) / n
  End If
Next k
For k = 0 To n - 1
  f(k, 0) = df(k, 0)
  f(k, 1) = df(k, 1)
Next k
End Sub

Private Sub FFT__(f(), t)
Const tol = 0.000001
Pi = 4 * Atn(1)
n = UBound(f()) + 1
M = Log(n) / Log(2)
mm = Round(M)
If Abs(M - mm) > tol Then
  Call DFT__(f(), t)
Else
  For j = 1 To M
    l = n / 2 ^ j
    For k = 0 To n - 1 Step 2 * l
    w = Pi / l
    For i = 0 To l - 1
      c = Cos(i * w)
      S = Sin(i * w) * t
      x = k + i
      y = k + i + l
      f(x, 0) = f(x, 0) + f(y, 0)
      f(x, 1) = f(x, 1) + f(y, 1)
      tr = f(x, 0) - f(y, 0) - f(y, 0)
      ti = f(x, 1) - f(y, 1) - f(y, 1)
      f(y, 0) = tr * c + ti * S
      f(y, 1) = ti * c - tr * S
    Next i
    Next k
```

```
  Next j
  Bit_Reverse f(), n
  If t = 1 Then
    For i = 0 To n - 1
      f(i, 0) = f(i, 0) / n
      f(i, 1) = f(i, 1) / n
    Next i
  End If
End If
End Sub
Private Sub Bit_Reverse(f(), n)

j = 0
For i = 0 To n - 2
  If i < j Then
    Swap f(i, 0), f(j, 0)
    Swap f(i, 1), f(j, 1)
  End If
  k = n / 2
  Do
    If k >= j + 1 Then Exit Do
    j = j - k
    k = k / 2
  Loop
  j = j + k
Next

End Sub
Private Sub Swap(a, b)
t = a
a = b
b = t
End Sub

'Should be called with a range of covariances gamma(0) to gamma(n-1),
'where gamma(0) is the variance.
'To utilise FFT, n - 1 (and not n as in typical FFT) should be a power of 2

Function FFTPowSpec_(a)
n = a.Count
ReDim f(2 * n - 3, 1)
ReDim df(n - 1, 1)
For i = 0 To n - 1
  f(i, 0) = 4 * (n - 1) * a(i + 1)
  Next i
For i = n To 2 * n - 3
  f(i, 0) = f(2 * n - i - 2, 0)
  Next i
FFT__ f(), 1
For i = 0 To n - 1
  df(i, 0) = i / 2 / (n - 1)
  df(i, 1) = f(i, 0)
Next i
```

```
FFTPowSpec_ = df
End Function

Function FFT_AMACoef_(spectrum, theta)
Pi = 4 * Atn(1)
n = spectrum.Count
ReDim ss(n - 1) As Double
ReDim th(n - 1) As Double
For i = 0 To n - 1
  If IsMissing(theta) Then
      th(i) = 0
    Else
      th(i) = theta(i + 1)
    End If
  ss(i) = (2 * spectrum(i + 1)) ^ 0.5 * Cos(2 * Pi * th(i)) / (4 * (n - 1))
  Next i
ReDim f(2 * n - 3, 1)
  For i = 0 To n - 1
    f(i, 0) = ss(i)
    f(i, 1) = ss(i) * Tan(2 * Pi * th(i))
    Next i
  f(n - 1, 1) = 0
  For i = n To 2 * n - 3
    f(i, 0) = f(2 * n - i - 2, 0)
    f(i, 1) = -f(2 * n - i - 2, 1)
    Next i
FFT__ f(), -1
ReDim df(2 * (n - 1) + 1, 1)
For i = 0 To n - 1
  df(i, 0) = i - n + 1
  df(i + n - 1, 0) = i
  df(i, 1) = f(n - 1 - i, 0)
  If i = 0 Then
    df(i + n - 1, 1) = f(n - 1, 0)
  Else
    df(i + n - 1, 1) = f(2 * (n - 2) + 2 - i, 0)
  End If
  Next i
FFT_AMACoef_ = df
End Function
```