Predicting Nutrient Degradation from Two Successive Concentrations C.nb

Program C: Using experimental concentrations and interpolated time-temperature data

Programmed by: Micha Peleg & Mark D. Normand Last modified: March 10, 2015

Developed as part of a project on vitamin loss kinetics in space foods supported by NASA under project SA-14-042.

This program predicts chemical degradation of compounds in food such as nutrients or pigments from two successive concentrations (or concentration ratios) determined experimentally during storage. It is based on the assumptions that in the pertinent temperature range the degradation follows fixed order kinetics, $n \geq 0$ [1], and that the temperature-dependence of the corresponding rate constant k(T) follows the exponential model [2], i.e., k(T(t)) = kTref*Exp(c*(T(t) - Tref))

This program, C, has two parts: The first starts with conversion of user-entered experimental time-temperature data to a continuous model using *Mathematica*'s Interpolation function. This model in the form of an interpolation function (T) is now used as the temperature profile equation replacing the algebraic terms in program versions A and B. The second part is using T and two entered experimentally determined concentrations, C1 and C2, at times t1 and t2, respectively, to extract the reaction's kinetic parameters, kTrefEst and cEst, to reconstruct the entire degradation curve and to predict a third concentration on it at time t3.

The experimentally determined {time,Temperature} data are imported into this program from a tab-separated-values (.TSV) ASCII text file.

WARNING: Note that not all possible experimental concentration entries (C1 at t1 and C2 at t2) have a solution. The reasons can be experimental error(s), the reaction order differs from that assumed or that the reaction follows nonlinear kinetics.

References

[1] M. Peleg, M. D. Normand and A. D. Kim, "Estimating Nutrients' Thermal Degradation Kinetic Parameters with the Endpoints Method," *Food Research International*, **66**, 2014 pp. 313-324.

[2] M. Peleg, M. D. Normand and M. G. Corradini, "The Arrhenius equation revisited," *Critical Reviews in Foods Science and Nutrition,* **52**, 2012 pp. 830-851.

PART 1

Clear all variables in all contexts.

```
ClearAll["`*"]
```

C0 is the initial nutrient's concentration in the user's chosen units. *Notice that when C0=1, the program can refer to concentration ratios rather than to absolute concentrations*. Tref is the user-chosen reference temperature for the calculations. It should be in the same temperature range as the data.

```
C0 = 1.; Tref = 25.;
```

Assign tAxisMax and yAxisMax, the maximum values of the plot's x- and y-axes, respectively.

```
tAxisMax = 120.; yAxisMax = C0 + .01;
```

Assign the time values, t1 and t2, in the pertinent time units, e.g., days, weeks, months.

```
t1 = 15.; t2 = 45.;
```

Enter C1 and C2, the experimentally measured concentration values at times t1 and t2.

```
C1 = 0.788; C2 = 0.635;
```

Assign t3, the time at which we want to predict the concentration or concentration ratio in the pertinent time units, e.g., days, weeks, months. (Note that there are conditions where this value might be lowered by the program.)

```
t3 = 80.;
```

Show all of Mathematica's allowed import file formats.

```
$ImportFormats
```

```
{3DS, ACO, Affymetrix, AgilentMicroarray, AIFF, ApacheLog, ArcGRID, AU, AVI,
 Base64, BDF, Binary, Bit, BMP, Byte, BYU, BZIP2, CDED, CDF, Character16,
 Character8, CIF, Complex128, Complex256, Complex64, CSV, CUR, DBF, DICOM,
 DIF, DIMACS, Directory, DOT, DXF, EDF, EPS, ExpressionML, FASTA, FASTQ, FCS,
 FITS, FLAC, GenBank, GeoTIFF, GIF, GPX, Graph6, Graphlet, GraphML, GRIB,
 GTOPO30, GXL, GZIP, HarwellBoeing, HDF, HDF5, HIN, HTML, ICC, ICNS, ICO, ICS,
 Integer128, Integer16, Integer24, Integer32, Integer64, Integer8, JCAMP-DX,
 JPEG, JPEG2000, JSON, JVX, KML, LaTeX, LEDA, List, LWO, MAT, MathML, MBOX,
 MDB, MGF, MIDI, MMCIF, MOL, MOL2, MPS, MTP, MTX, MX, NASACDF, NB, NDK, NetCDF,
 NEXUS, NOFF, OBJ, ODS, OFF, OpenEXR, Package, Pajek, PBM, PCX, PDB, PDF, PGM,
 PLY, PNG, PNM, PPM, PXR, QuickTime, RawBitmap, Real128, Real32, Real64, RIB,
 RSS, RTF, SCT, SDF, SDTS, SDTSDEM, SFF, SHP, SMILES, SND, SP3, Sparse6, STL,
 String, SurferGrid, SXC, Table, TAR, TerminatedString, Text, TGA, TGF, TIFF,
 TIGER, TLE, TSV, UnsignedInteger128, UnsignedInteger16, UnsignedInteger24,
 UnsignedInteger32, UnsignedInteger64, UnsignedInteger8, USGSDEM, UUE, VCF, VCS,
 VTK, WAV, Wave64, WDX, XBM, XHTML, XHTMLMathML, XLS, XLSX, XML, XPORT, XYZ, ZIP}
```

Import the experimentally determined {time, temperature} data from a tab-separated-values (.TSV) ASCII text file into the testData matrix. Here we have pasted the data into this notebook so that no external file needs to be imported. If you want to import your own data from a file, uncomment the cell below and delete the cell after that which contains the assignment of data to testData.

```
(*testData=Import["testData.tsv"]*)
```

```
testData = {{0, 25.`}, {3.`, 25.7`}, {6.`, 21.7`}, {9.`, 19.9`},
   {12.`, 21.8`}, {15.`, 19.8`}, {18.`, 15.8`}, {21.`, 16.7`},
   {24.`, 17.3`}, {27.`, 13.3`}, {30.`, 11.5`}, {33.`, 13.9`}, {36.`, 14.9`},
   {39.`, 14.`}, {42.`, 12.3`}, {45.`, 11.5`}, {48.`, 12.4`}, {51.`, 14.9`},
   {54.`, 17.3`}, {57.`, 18.2`}, {60.`, 17.3`}, {63.`, 15.6`}, {66.`, 14.8`},
   {69.`, 15.8`}, {72.`, 18.3`}, {75.`, 20.7`}, {78.`, 21.6`}, {81.`, 20.6`},
   {84.`, 19.`}, {87.`, 18.2`}, {90.`, 19.2`}, {93.`, 21.7`}, {96.`, 24.1`},
   {99.`, 24.9`}, {102.`, 24.`}, {105.`, 22.3`}, {108.`, 21.5`},
   {111.`, 22.6`}, {114.`, 25.1`}, {117.`, 27.5`}, {120.`, 28.3`}}
```

```
{{0, 25.}, {3., 25.7}, {6., 21.7}, {9., 19.9}, {12., 21.8}, {15., 19.8},
 {18., 15.8}, {21., 16.7}, {24., 17.3}, {27., 13.3}, {30., 11.5}, {33., 13.9},
 {36., 14.9}, {39., 14.}, {42., 12.3}, {45., 11.5}, {48., 12.4}, {51., 14.9},
 {54., 17.3}, {57., 18.2}, {60., 17.3}, {63., 15.6}, {66., 14.8}, {69., 15.8},
 {72., 18.3}, {75., 20.7}, {78., 21.6}, {81., 20.6}, {84., 19.}, {87., 18.2},
 {90., 19.2}, {93., 21.7}, {96., 24.1}, {99., 24.9}, {102., 24.}, {105., 22.3},
 {108., 21.5}, {111., 22.6}, {114., 25.1}, {117., 27.5}, {120., 28.3}}
```

Display testData in matrix form.

```
MatrixForm[testData]
```

0	25.
3.	25.7
6.	21.7
9.	19.9
12.	21.8
15.	19.8
18.	15.8
21.	16.7
24.	17.3
27.	13.3
30.	11.5
33.	13.9
36.	14.9
39.	14.
42.	12.3
45.	11.5
48.	12.4
51.	14.9
54.	17.3
57.	18.2
60.	17.3
63.	15.6
66.	14.8
69.	15.8
72.	18.3
75.	20.7
78.	21.6
81.	20.6
84.	19.
87.	18.2
90.	19.2
93.	21.7
96.	24.1
99.	24.9
102.	24.
105.	22.3
108.	21.5
111.	22.6
114.	25.1
117.	27.5
120.	28.3

Display the number of time-temperature data values imported into matrix testData.

```
nPts = Length[testData]
```

```
41
```

Display the minimum temperature value in testData.

```
TAxisMin = Min[testData[[All, 2]]]
```

```
11.5
```

Assign the minimum value of the temperature axis.

```
TAxisMin = 0.;
```

Display the maximum temperature value in testData.

```
TAxisMax = Max[testData[[All, 2]]]
```

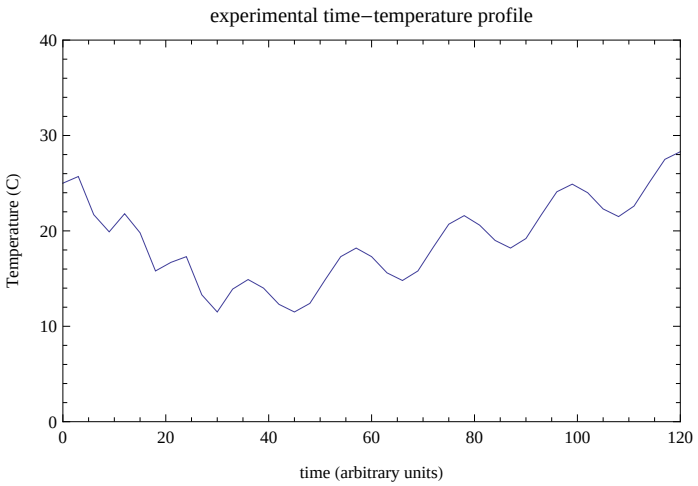
```
28.3
```

Assign the maximum value of the temperature axis.

```
TAxisMax = 40.;
```

ListPlot the time-temperature data with the points connected.

```
dataPlot =
 ListPlot[testData, PlotRange → {{0., tAxisMax}, {TAxisMin, TAxisMax}}, Joined → True,
  Frame → True, FrameLabel → {"time (arbitrary units)", "Temperature (C)"},
  PlotLabel → "experimental time-temperature profile"]
```



Assign T to be an interpolating function defined from the nPts testData points.

```
T = Interpolation[testData]
```

```
InterpolatingFunction[{{0., 120.}}, <>]
```

Define the degradation curve's equation, Conc(tFinal), for the chosen kinetic parameters and temperature profile (now in the form of an interpolation function, T) as returned by NDSolve. *Notice that where n=0, the concentration or concentration ratio can become negative. Or where 0<n<1, the concentration or concentration ratio can become a complex number. Where this occurs the concentration or concentration ratio is assigned a value of zero. However, in this region the method is not viable (see below). Note that T[t] refers to the interpolation function defined from the testData values.*

```
Conc[tFinal_?NumericQ, n_?NumericQ, kTref_?NumericQ,
  c_?NumericQ, C0_?NumericQ, Tref_] := Module[{k1, sfn1, y},
  k1[t_] := kTref * Exp[c * (T[t] - Tref)]; sfn1[t_] =
   NDSolve[{y'[t] == If[n == 0, -k1[t], If[n == 1, -k1[t] * y[t] , -k1[t] * y[t]^n]],
      y[0] == C0}, y[t], {t, 0., tFinal}][[1, 1, 2]];
  If[sfn1[tFinal] ≤ 0. || Im[sfn1[tFinal]] > 0., 0., sfn1[tFinal]]]
```

PART 2

Enter the assumed degradation kinetics order, nEst.

```
nEst = 1.;
```

Use FindRoot to solve for the parameters kTrefEst and cEst from the actual concentrations, C1 and C2, at times t1 and t2, respectively, assuming the reaction's kinetic order nEst. *In case the FindRoot fails with the automatically assigned initial guesses for kTrefEst and cEst, use the attached Mathematica notebook "*PredictingNutrientDegradationFromTwoSuccessiveConcentrationsD.nb*" to find better guesses.*

```
Clear[kTrefEst, cEst]
```

```
theRoot = FindRoot[{Conc[t1, nEst, kTrefEst, cEst, C0, Tref] == C1,
    Conc[t2, nEst, kTrefEst, cEst, C0, Tref] == C2},
  {{kTrefEst, .015, .02}, {cEst, .05, .1}}, MaxIterations → 50]
```

```
{kTrefEst → 0.0199115, cEst → 0.0990813}
```

Assign the estimated (calculated) values of the parameters kTrefEst and cEst.

```
kTrefEst = theRoot[[1, 2]]; Print[Style[Row[{"kTrefEst = ", kTrefEst}], Red, 14]]
```

```
kTrefEst = 0.0199115
```

```
cEst = theRoot[[2, 2]]; Print[Style[Row[{"cEst = ", cEst}], Red, 14]]
```

```
cEst = 0.0990813
```

Assign the final time value, tModelMax.

```
tModelMax = tAxisMax
```

```
120.
```

The following test may lower the user-entered value of t3, which is displayed.

```
If[t3 > tModelMax, t3 = tModelMax - 1.]; t3
```

```
80.
```

Plot the predicted (reconstructed) degradation curve Conc(t) using the chosen nEst and retrieved parameter values, kTrefEst and cEst, returned by FindRoot for t varying from 0.001 to tModelMax.

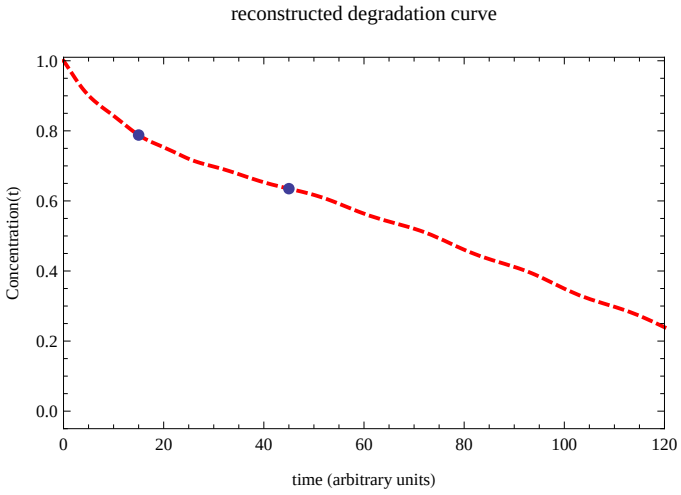
```
predictedPlot = Plot[Conc[t, nEst, kTrefEst, cEst, C0, Tref],
    {t, 0.001, tModelMax}, PlotRange → {{0., tAxisMax}, {-.05, yAxisMax}},
    PlotStyle → {Red, Dashed, Thick},
    PlotStyle → {AbsoluteThickness[1], AbsoluteThickness[1]}, Frame → True,
    FrameLabel → {"time (arbitrary units)", "Concentration(t)", "", ""},
    PlotLabel → "reconstructed degradation curve"];
```

Plot the two points' concentration values, C1 and C2, at times t1 and t2, respectively.

```
kTrefcPtPlot =
  ListPlot[{{t1, C1}, {t2, C2}}, PlotRange → {{0., tAxisMax}, {-.05, yAxisMax}},
   PlotStyle → AbsolutePointSize[6], Frame → True,
   FrameLabel → {"time (arbitrary units)", "Concentration(t)", "", ""}];
```

Show the reconstructed curve (dashed in red) and the two experimental concentration values, C1 and C2, at times t1 and t2, respectively, on the same graph.

```
Show[predictedPlot, kTrefcPtPlot]
```



Compute (predict) the concentration at time t3 using the assumed nEst and the estimated parameters kTrefEst and cEst.

```
predictedC3 = Conc[t3, nEst, kTrefEst, cEst, C0, Tref];
Print[Style[Row[{"predictedC3 = ", predictedC3}], Magenta, 14]]
```

```
predictedC3 = 0.460524
```

Plot the predicted concentration predictedC3 at time t3.

```
predictedC3Plot =
  ListPlot[{{t3, predictedC3}}, PlotRange → {{0., tAxisMax}, {-.05, yAxisMax}},
   PlotStyle → {AbsolutePointSize[6], Red}, Frame → True,
   FrameLabel → {"time (arbitrary units)", "Concentration(t)", "", ""}];
```

Show the reconstructed (predicted) degradation curve together with the three points

```
Show[predictedPlot, kTrefcPtPlot, predictedC3Plot]
```