Degradation Parameters Estimation from Interpolated Temperature Data (Version A - Storage).nb

Model by: Micha Peleg Programmed by: Mark D. Normand Last modified: May 14, 2015

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

Theoretically, when a degradation reaction follows a known fixed-order kinetics, n, and the rate constant's temperature-dependence follows a two-parameter model, then these parameters can be determined from two concentration ratios after exposure to two different temperature histories. This is done by numerically solving two simultaneous rate equations. Such a solution, however, may require fairly accurate initial guesses that are difficult to assign intuitively. Nevertheless, they can be estimated by manually matching the two experimental points with their corresponding degradation curves using *Mathematica*'s `Manipulate` function. Once found, they can be used either as initial guesses or as parameters themselves. In this program the degradation reaction rate's temperature-dependence is described by the exponential model $k(T) = k(T_{ref}) * \text{Exp}[c * (T - T_{ref})]$ where T is the temperature, $k(T_{ref})$ is the rate constant at a reference temperature, T_{ref}, and c is a constant. This program allows one to estimate $k(T_{ref})$ and c by moving slider controls on the screen. The user enters the two digitized temperature profile data sets either by reading the data from a file or pasting them from another *Mathematica* notebook. Also entered using sliders are: the two final concentration ratios, the times at which they were determined and the reference temperature, T_{ref}. It is assumed that the reaction follows first-order kinetics or approximately first-order kinetics, in which case the deviation from n=1 can also be entered with a slider. The parameters' settings in this program (Version A) are for storage temperatures. (For heat processing temperatures use Version B.)

Notice that not all allowed entries for concentration ratios and times must result in a match. If there is no match one should suspect that either the model's underlying assumptions are invalid and/or there is a substantial error or errors in the experimental data.

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, A. D. Kim and M. D. Normand, "Predicting anthocyanin's isothermal and non-isothermal degradation with the endpoints method" *Food Chemistry*, 2015 (in press).

[3] 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.

Clear all variables in all contexts.

In[54]:=
```
ClearAll["`*"]
```

Show all of Mathematica's allowed import file formats.

In[55]:=
```
$ImportFormats
```

Out[55]=
```
{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 first set of experimentally determined {time, temperature} data from a tab-separated-values (.TSV) ASCII text file into a matrix called tempData1A. Below 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 called "tempData1A.TSV" (located in your home folder), uncomment the cell below and comment out the cell after that which contains the assignment of the data to the tempData1A matrix.

In[56]:=
```
(*tempData1A=Import["tempData1A.TSV"]*)
```

In[57]:=
```
tempData1A = {{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`}}
```

Out[57]=
```
{{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 in matrix tempData1A and assign it to nPts1.

In[58]:=
```
nPts1 = Length[tempData1A]
```

Out[58]=
```
41
```

Display the minimum temperature value in tempData1A.

In[59]:=
```
Min[tempData1A[[All, 2]]]
```

Out[59]=
```
11.5
```

Display the maximum temperature value in tempData1A.

In[60]:=
```
Max[tempData1A[[All, 2]]]
```

Out[60]=
```
28.3
```

Import the second set of experimentally determined {time, temperature} data from a second tab-separated-values (.TSV) ASCII text file into a matrix called tempData2A. Below 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 called “tempData2A.TSV” (located in your home folder), uncomment the cell below and comment out the cell after that which contains the assignment of the data to the tempData2A matrix.

In[61]:=
```
(*tempData2A=Import["tempData2A.TSV"]*)
```

In[62]:=
```
tempData2A = tempData1A[[1 ;; nPts1 - 5]];
Do[tempData2A[[i, 2]] = tempData1A[[i, 2]] - .2 * tempData1A[[i, 1]], {i, nPts1 - 5}];
tempData2A
```

Out[62]=
```
{{0, 25.}, {3., 25.1}, {6., 20.5}, {9., 18.1}, {12., 19.4}, {15., 16.8},
 {18., 12.2}, {21., 12.5}, {24., 12.5}, {27., 7.9}, {30., 5.5}, {33., 7.3},
 {36., 7.7}, {39., 6.2}, {42., 3.9}, {45., 2.5}, {48., 2.8}, {51., 4.7},
 {54., 6.5}, {57., 6.8}, {60., 5.3}, {63., 3.}, {66., 1.6}, {69., 2.},
 {72., 3.9}, {75., 5.7}, {78., 6.}, {81., 4.4}, {84., 2.2}, {87., 0.8},
 {90., 1.2}, {93., 3.1}, {96., 4.9}, {99., 5.1}, {102., 3.6}, {105., 1.3}}
```

Display the number of time-temperature data values in matrix tempData2A and assign it to nPts2.

In[63]:=
```
nPts2 = Length[tempData2A]
```

Out[63]=
```
36
```

Display the minimum temperature value in tempData2A.

In[64]:=
```
Min[tempData2A[[All, 2]]]
```

Out[64]=
```
0.8
```

Display the maximum temperature value in tempData2A.

In[65]:=
```
Max[tempData2A[[All, 2]]]
```

Out[65]=
```
25.1
```

Assign tAxisMax1 and tMax1 to be the maximum time value found in matrix tempData1A.

In[66]:=
```
tAxisMax1 = Max[tempData1A[[All, 1]]]; tMax1 = tAxisMax1
```

Out[66]=
```
120.
```

Assign tAxisMax2 and tMax2 to be the maximum time value found in matrix tempData2A.

In[67]:=
```
tAxisMax2 = Max[tempData2A[[All, 1]]]; tMax2 = tAxisMax2
```

Out[67]=
```
105.
```

Assign TAxisMax1 and TMax1 to be the maximum Temperature value found in matrix tempData1A.

In[68]:=
```
TAxisMax1 = Max[tempData1A[[All, 2]]]; TMax1 = TAxisMax1
```

Out[68]=
```
28.3
```

In[69]:=
```
TAxisMax1 = 40.
```

Out[69]=
```
40.
```

Assign TAxisMax2 and TMax2 to be the maximum Temperature value found in matrix tempData2A.

In[70]:=
```
TAxisMax2 = Max[tempData2A[[All, 2]]]; TMax2 = TAxisMax2
```

Out[70]=
```
25.1
```

In[71]:=
```
TAxisMax2 = 40.
```

Out[71]=
```
40.
```

Assign T1 to be an interpolating function defined from the tempData1A points.

In[72]:=
```
T1 = Interpolation[tempData1A]
```

Out[72]=
```
InterpolatingFunction[{{0., 120.}}, <>]
```

Assign T2 to be an interpolating function defined from the tempData2A points.

In[73]:=
```
T2 = Interpolation[tempData2A]
```

Out[73]=
```
InterpolatingFunction[{{0., 105.}}, <>]
```

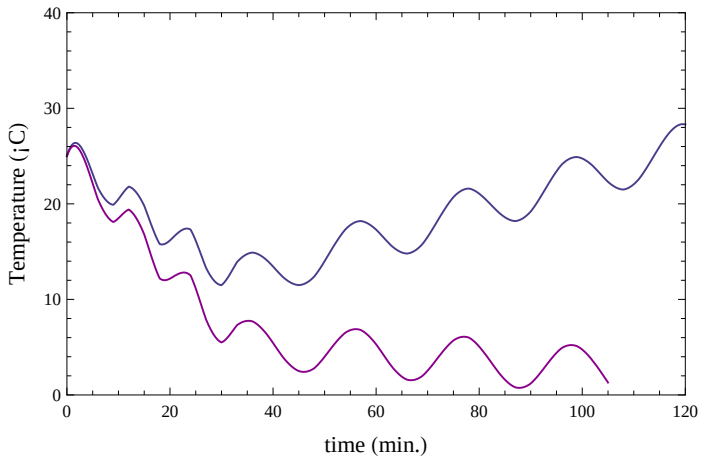
Plot the two interpolating functions T1[t] and T2[t] together from time 0 to Max[tAxisMax1,tAxisMax2].

In[74]:=
```
temp1Plot = Plot[T1[t], {t, 0, tMax1}, PlotRange → {{0, tAxisMax1}, {0, TAxisMax1}},
   PlotStyle → {AbsoluteThickness[1], ColorData["HTML", "DarkSlateBlue"]}, Frame →
    True, FrameLabel → {Style["time (min.)", 12], Style["Temperature (℃)", 12]}];
```

In[75]:=
```
temp2Plot = Plot[T2[t], {t, 0, tMax2}, PlotRange → {{0, tAxisMax2}, {0, TAxisMax2}},
   PlotStyle → {AbsoluteThickness[1], ColorData["HTML", "DarkMagenta"]}, Frame →
    True, FrameLabel → {Style["time (min.)", 12], Style["Temperature (℃)", 12]}];
```

In[76]:=
```
T1T2Plot = Show[temp1Plot, temp2Plot,
  PlotRange → {{0, Max[tAxisMax1, tAxisMax2]}, {0, Max[TAxisMax1, TAxisMax2]}}]
```

Out[76]=



Define ConcRatio(t), the degradation function obtained from the solution returned by NDSolve.

In[77]:=
```
ConcRatio[tFinal_?NumericQ, kTref_?NumericQ, c_?NumericQ,
  n_?NumericQ, Tref_?NumericQ, T_] := Module[{k, sfn, t, y},
  k[t_] := kTref * Exp[c * (T[t] - Tref)];
  sfn[t_] = NDSolve[{y'[t] == -k[t] * y[t]^n, y[0] == 1.},
     y[t], {t, 0., tFinal}, AccuracyGoal → 3][[1, 1, 2]];
  sfn[
   tFinal]]
```

The following Manipulate panel assists in selecting initial guesses for the kTref and c parameters of the ConcRatio(t) degradation function.

In[79]:=
```
Manipulate[Module[
  {CExpPtsPlot, CPredPlot, CRatio1Plot, CRatio2Plot, kTrefcText, T1Plot, T2Plot,
   TParamText, T1T2Plot, TempMax, TempMin, tFinalEstConc3Text, tMax, yAxisMax},
  tMax = tAxisMax;
  yAxisMax = 1.;
  tFinalEstConc3Text = Text[Style[Row[{""}], 12]];
  kTrefcText =
   Text[Style[Row[{Style["n", Italic]as, "= ", Round[nAssumed, .01], Spacer[10],
       Style["T", Italic]ref, "= ", Round[Treference, .01], Spacer[10],
       Style["k", Italic]Trefest, "= ", Round[kTrefEst, .001], Spacer[10],
       Style["c", Italic]est, "= ", Round[cEst, .001]}], Blue, 12]];
  T1Plot = Plot[T1[t], {t, 0., tMax1}, PlotRange → {{0., tAxisMax}, {0., TAxisMax}},
    PlotStyle → {AbsoluteThickness[1], ColorData["HTML", "DarkSlateBlue"]},
    Frame → True, FrameLabel → {{Style["Temperature (℃)", 12], ""},
      {Style["time (arbitrary units)", 12], Style["temperature profiles", 12]}},
    ImagePadding → {{45, 10}, {35, 35}}, ImageSize → {360, 222}];
  T2Plot = Plot[T2[t], {t, 0., tMax2}, PlotRange → {{0., tAxisMax}, {0., TAxisMax}},
    PlotStyle → {AbsoluteThickness[1], ColorData["HTML", "DarkMagenta"]},
    Frame → True, FrameLabel → {{Style["Temperature (℃)", 12], ""},
      {Style["time (arbitrary units)", 12], Style["temperature profiles", 12]}},
    ImagePadding → {{45, 10}, {35, 35}}, ImageSize → {360, 222}];
  CRatio1Plot = Plot[ConcRatio[t, kTrefEst, cEst, nAssumed, Treference, T1],
    {t, 0.001, tMax1}, PlotRange → {{0., tAxisMax}, {-.05, yAxisMax}},
    PlotStyle → {AbsoluteThickness[1], ColorData["HTML", "DarkSlateBlue"]},
    Frame → True, PlotLabel -> kTrefcText, FrameLabel →
     {{Style["concentration ratio(t)", 12], ""}, {Style["time (arbitrary units)",
        12], Style["reconstructed degradation curves", 12]}},
    ImagePadding → {{45, 10}, {35, 35}}, ImageSize → {360, 222}];
  CRatio2Plot = Plot[ConcRatio[t, kTrefEst, cEst, nAssumed, Treference, T2],
    {t, 0.001, tMax2}, PlotRange → {{0., tAxisMax}, {-.05, yAxisMax}},
    PlotStyle → {AbsoluteThickness[1], ColorData["HTML", "DarkMagenta"]},
    Frame → True, PlotLabel -> kTrefcText, FrameLabel →
     {{Style["concentration ratio(t)", 12], ""}, {Style["time (arbitrary units)",
        12], Style["reconstructed degradation curves", 12]}},
    ImagePadding → {{45, 10}, {35, 35}}, ImageSize → {360, 222}];
  CExpPtsPlot = ListPlot[{{{tFinal1, Cexper1}}, {{tFinal2, Cexper2}}},
    PlotRange → {{0., tAxisMax}, {-.05, yAxisMax}},
    PlotStyle → {{AbsolutePointSize[6], ColorData["HTML", "DarkSlateBlue"]},
      {AbsolutePointSize[6], ColorData["HTML", "DarkMagenta"]}},
    Frame → True, FrameLabel → {{Style["concentration ratio(t)", 12], ""},
      {Style["time (arbitrary units)", 12]}},
    ImagePadding → {{45, 10}, {35, 35}}, ImageSize → {360, 222}];

  T1T2Plot = Show[T1Plot, T2Plot];
```

```
 Column[
   {tFinalEstConc3Text, T1T2Plot, Show[CRatio1Plot, CRatio2Plot, CExpPtsPlot]}]
],
Style["kinetic parameter estimation", Bold, 9],
{{tFinal1, tMax1, Style["t", Italic]final1},
 10., 600., Appearance → "Labeled", ImageSize → Small},
{{Cexper1, .5, Style["C", Italic]exper1}, .1, .95,
 Appearance → "Labeled", ImageSize → Small},
"\n",
{{tFinal2, tMax2, Style["t", Italic]final2},
 10., 600., Appearance → "Labeled", ImageSize → Small},
{{Cexper2, .85, Style["C", Italic]exper2}, .1, .95,
 Appearance → "Labeled", ImageSize → Small},
Delimiter,
Style["assumed kinetic order", Bold, 9],
{{nAssumed, 1., Style["n", Italic]assumed},
 .95, 1.05, .01, Appearance → "Labeled", ImageSize → Small},
Delimiter,
Style["reference temperature", Bold, 9],
{{Treference, 20., Style["T", Italic]ref},
 20., 80., Appearance → "Labeled", ImageSize → Small},
Delimiter,
Style["estimated degradation's parameters", Bold, 9],
{{kTrefEst, .0055, Style["k", Italic]Trefest},
 .0001, .2, Appearance → "Labeled", ImageSize → Small},
{{cEst, .149, Style["c", Italic]est}, .0001, .2, Appearance → "Labeled",
 ImageSize → Small},
Delimiter,
Style["time axis maximum", Bold, 9],
{{tAxisMax, 140., Style["t", Italic]max},
 10., 600., Appearance → "Labeled", ImageSize → Small},
Style["Temperature axis maximum", Bold, 9],
{{TAxisMax, 40., Style["T", Italic]max},
 10., 150., Appearance → "Labeled", ImageSize → Small},
ControlPlacement → Left, SaveDefinitions → True, AutorunSequencing → Automatic,
TrackedSymbols :> {tFinal1, Cexper1, tFinal2, Cexper2,
  nAssumed, Treference, kTrefEst, cEst, tAxisMax, TAxisMax}]
```

Out[79]=

kinetic parameter estimation

t_{final1} 120.

C_{exper1} 0.5

t_{final2} 105.

C_{exper2} 0.85

assumed kinetic order

$n_{assumed}$ 1.

reference temperature

T_{ref} 20.

estimated degradation's parameters

$k_{Trefest}$ 0.0055

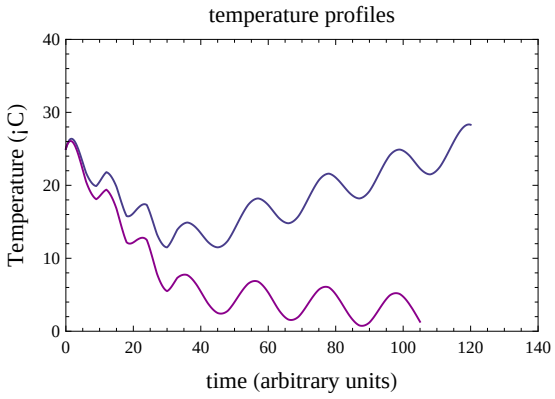
c_{est} 0.149

time axis maximum

t_{max} 140.

Temperature axis maximum

T_{max} 40.



n_{as}= 1. T_{ref}= 20. $k_{Trefest}$= 0.006 c_{est}= 0.149