Weibullian Chemical Degradation: Kinetic Parameters Estimation by the Two Endpoints Method: Isothermal and Dynamic (from Two Sets of Interpolated Temperature Profile Data) Version A - Storage.nb

Model by: Micha Peleg Programmed by: Mark D. Normand Last modified: August 30, 2016

Developed as part of a project on vitamin loss kinetics in space foods supported by NASA under Grant NNX14AP32G.

An isothermal chemical degradation reaction may follow Weibullian ("stretched exponential") kinetics with a fixed shape factor ("exponent"), m, i.e., Log[$C(t)$] = -$b(T)$*t^m where $C(t)$ is the momentary concentration ratio at time t and $b(T)$ a temperature dependent rate parameter, the equivalent of $k(T)$ in fixed order kinetics. The rate constant's temperature-dependence, may follow the two-parameter exponential model, $b(T) = b(T_{ref}) * \text{Exp}[c * (T - T_{ref})]$ which can be used interchangeably with the Arrhenius equation. T_{ref} in this equation is an arbitrary reference temperature in a pertinent temperature range. For a known or assumed value of m, the two parameters $b(T_{ref})$ and c can be determined from two concentration ratios C_1 and C_2 determined after exposure to two different temperature histories. This is done by numerically solving two simultaneous rate equations in the form of $\text{dLog}[C_1(t)]/\text{st} = f_1[T_1(t)]$ and $\text{dLog}[C_2(t)]/\text{st} = f_2[T_2(t)]$, for C_1 and C_2 with the boundary condition that $\text{Log}[C_1(0)] = \text{Log}[C_2(0)] = 0$. 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 the parameters' values themselves. This program allows one to estimate $b(T_{ref})$ 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 assumed m value, two final concentration ratios, C_1 and C_2, and the corresponding times of their determinations, $t_1$and t_2, and the reference temperature, T_{ref}.

The program also offers an "isothermal option" whereby instead of entering the digitized temperature profile data, the user enters with sliders two constant temperatures T_1 and T_2, and corresponding C_1 and C_2 and their $t_1$and t_2.

The parameters' settings in the "dynamic option" of this program (Version A) are for storage temperatures. Thus, for heat processing temperatures use Version B. The "isothermal option" covers the whole range from 0 to 100C.

For applying the "successive-points method", that is, having two experimental concentration ratios determined *at two different times during a single non-isothermal temperature history* (see reference [3]), you have to make a slight addition to the code. As before, a single set of temperature profile data gets assigned to tempData1A. After the statement assigning the interpolation function from the tempData1A data to T1 (i.e., after T1=Interpolation[tempData1A]), add the code line T2 = T1. Then, re-evaluate the notebook and attempt to pass the *single reconstructed curve* through the two entered points. When that is successful the $b(T_{ref})$ and c sliders' positions will show these two parameters' sought values.

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

[2] M. Peleg, A. D. Kim and M. D. Normand, "Predicting anthocyanins' isothermal and non-isothermal degradation with the endpoints method. *Food Chemistry,* **187**, 2015 pp. 537-544.

[3] M. Peleg and M. D. Normand, "Predicting chemical degradation during storage from two successive concentration ratios: Theoretical investigation." *Food Research International* **75**, 2015 pp.174-181.

Clear all variables in all contexts.

In[1]:=
```
ClearAll["`*"]
```

Show all of Mathematica's allowed import file formats.

In[2]:=
```
$ImportFormats
```

Out[2]=
```
{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, DAE, DBF, DICOM,
 DIF, DIMACS, Directory, DOT, DXF, EDF, EPS, ExpressionJSON, 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, MESH, MGF, MIDI, MMCIF, MOL, MOL2, MP3, MPS, MTP,
 MTX, MX, NASACDF, NB, NDK, NetCDF, NEXUS, NOFF, OBJ, ODS, OFF, OGG, OpenEXR,
 Package, Pajek, PBM, PCX, PDB, PDF, PGM, PLY, PNG, PNM, PPM, PXR, QuickTime,
 Raw, RawBitmap, RawJSON, 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, WebP, 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[3]:=
```
(*tempData1A=Import["tempData1A.TSV"]*)
```

In[4]:=
```
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[4]=
```
{{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[5]:=
```
nPts1 = Length[tempData1A]
```

Out[5]=
```
41
```

Display the minimum temperature value in tempData1A.

In[6]:=
```
Min[tempData1A[[All, 2]]]
```

Out[6]=
```
11.5
```

Display the maximum temperature value in tempData1A.

In[7]:=
```
Max[tempData1A[[All, 2]]]
```

Out[7]=
```
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[8]:=
```
(*tempData2A=Import["tempData2A.TSV"]*)
```

In[9]:=
```
tempData2A = tempData1A[[1 ;; nPts1 - 5]];
Do[tempData2A[[i, 2]] = tempData1A[[i, 2]] - .2 * tempData1A[[i, 1]], {i, nPts1 - 5}];
tempData2A
```

Out[9]=
```
{{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[10]:=
```
nPts2 = Length[tempData2A]
```

Out[10]= 36

Display the minimum temperature value in tempData2A.

In[11]:=
```
Min[tempData2A[[All, 2]]]
```

Out[11]= 0.8

Display the maximum temperature value in tempData2A.

In[12]:=
```
Max[tempData2A[[All, 2]]]
```

Out[12]= 25.1

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

In[13]:=
```
tAxisMax1 = Max[tempData1A[[All, 1]]]; tMax1 = tAxisMax1
```

Out[13]= 120.

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

In[14]:=
```
tAxisMax2 = Max[tempData2A[[All, 1]]]; tMax2 = tAxisMax2
```

Out[14]= 105.

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

In[15]:=
```
TAxisMax1 = Max[tempData1A[[All, 2]]]; TMax1 = TAxisMax1
```

Out[15]= 28.3

In[16]:=
```
TAxisMax1 = 40.
```

Out[16]= 40.

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

In[17]:=
```
TAxisMax2 = Max[tempData2A[[All, 2]]]; TMax2 = TAxisMax2
```

Out[17]= 25.1

In[18]:=
```
TAxisMax2 = 40.
```

Out[18]= 40.

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

In[19]:=
```
T1 = Interpolation[tempData1A]
```

Out[19]= InterpolatingFunction[]

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

In[20]:=
```
T2 = Interpolation[tempData2A]
```

Out[20]= InterpolatingFunction[]

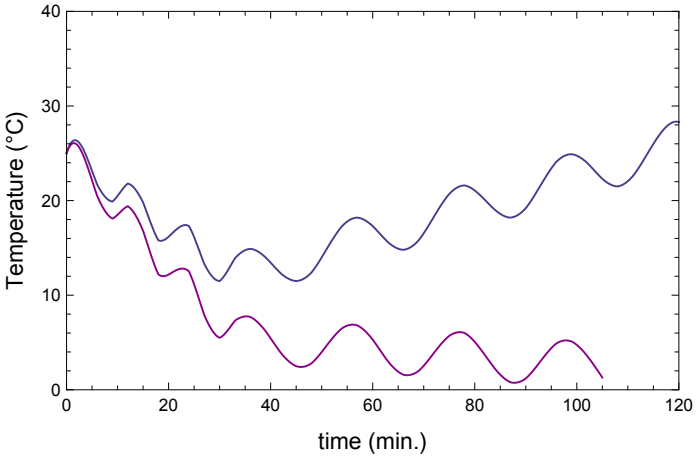
Plot the two interpolating functions T1[t] and T2[t] together from time 0 to Max[tAxisMax1,tAxisMax2].

In[21]:=
```
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 (°C)", 12]}];
```

In[22]:=
```
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 (°C)", 12]}];
```

In[23]:=
```
T1T2Plot = Show[temp1Plot, temp2Plot,
  PlotRange → {{0, Max[tAxisMax1, tAxisMax2]}, {0, Max[TAxisMax1, TAxisMax2]}}]
```

Out[23]=



Define lnConcRatio(t), the natural log of the momentary concentration degradation rate, dC(t)/dt, returned by NDSolve.

In[24]:=
```
lnConcRatio[tFinal_?NumericQ, bTref_?NumericQ, c_?NumericQ,
  m_?NumericQ, Tref_?NumericQ, T_] := Module[{b, sfn, t, y},
  b[t_] := bTref * Exp[c * (T[t] - Tref)];
  sfn[t_] =
   NDSolve[{y'[t] == -b[t] * m * (-y[t] / b[t]) ^ ((m - 1) / m), y[0.0] == -0.0001},
     y[t], {t, .0001, tFinal}, AccuracyGoal → 4][[1, 1, 2]];
  sfn[
   tFinal]]
```

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

In[25]:=
```
Manipulate[Module[{CExpPtsPlot, CPredPlot, CRatio1Plot,
    CRatio2Plot, kTrefcText, T1Plot, T2Plot, TParamText, T1T2Plot,
    TempMax, TempMin, temp1, temp2, tFinalEstConc3Text, tMax, yAxisMax},
  tMax = tAxisMax;
  yAxisMax = 1.;
  tFinalEstConc3Text = Text[Style[Row[{""}], 12]];
  kTrefcText =
   Text[Style[Row[{Spacer[70], Style["m", Italic]as, "= ", Round[mAssumed, .01],
       Spacer[10], Style["T", Italic]ref, "= ", Round[Treference, .01],
       Spacer[10], Style["b", Italic]Trefest, "= ", Round[bTrefEst, .001],
       Spacer[10], Style["c", Italic]est, "= ", Round[cEst, .001]}], Blue, 12]];
  If[tempPro == 1, (* isothermal temperature profile *)
   temp1[t_] := isoT1;
   temp2[t_] := isoT2
   , (* dynamic temperature profile *)
```

```
  temp1[t_] := T1[t];
  temp2[t_] := T2[t]
 ];
 T1Plot = Plot[temp1[t], {t, 0., tMax1}, PlotRange → {{0., tAxisMax}, {0., TAxisMax}},
   PlotStyle → {AbsoluteThickness[1], ColorData["HTML", "DarkSlateBlue"]},
   Frame → True, FrameLabel → {{Style["Temperature (°C)", 12], ""},
     {Style["time (arbitrary units)", 12], Style["temperature profiles", 12]}},
   ImagePadding → {{45, 10}, {35, 35}}, ImageSize → {360, 222}];
 T2Plot = Plot[temp2[t], {t, 0., tMax2}, PlotRange → {{0., tAxisMax}, {0., TAxisMax}},
   PlotStyle → {AbsoluteThickness[1], ColorData["HTML", "DarkMagenta"]},
   Frame → True, FrameLabel → {{Style["Temperature (°C)", 12], ""},
     {Style["time (arbitrary units)", 12], Style["temperature profiles", 12]}},
   ImagePadding → {{45, 10}, {35, 35}}, ImageSize → {360, 222}];
 CRatio1Plot = Plot[Exp[lnConcRatio[t, bTrefEst, cEst, mAssumed, Treference,
     temp1]], {t, 0.001, tMax1}, PlotRange → {{0., tAxisMax}, {-.05, yAxisMax}},
   PlotStyle → {AbsoluteThickness[1], ColorData["HTML", "DarkSlateBlue"]},
   Frame → True, 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[Exp[lnConcRatio[t, bTrefEst, cEst, mAssumed, Treference,
     temp2]], {t, 0.001, tMax2}, PlotRange → {{0., tAxisMax}, {-.05, yAxisMax}},
   PlotStyle → {AbsoluteThickness[1], ColorData["HTML", "DarkMagenta"]},
   Frame → True, 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,
   kTrefcText, Show[CRatio1Plot, CRatio2Plot, CExpPtsPlot]}]
],
Row[{"temperature profile",
  Control[{{tempPro, 2, ""}, {1 → "isothermal", 2 → "dynamic"}}]}],
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["constant temperatures", Bold],
{{isoT1, 30., Row[{Style["T", Italic]1, " (°C)"}]}, 0., 100., .01,
 Appearance → "Labeled", ImageSize → Small, Enabled → If[tempPro == 1, True, False]},
```

```
{{isoT2, 20., Row[{Style["T", Italic]₂, " (°C)"}]}, 0., 100., .01,
 Appearance → "Labeled", ImageSize → Small, Enabled → If[tempPro == 1, True, False]},
Delimiter,
Style["assumed shape factor", Bold, 9],
{{mAssumed, 0.9, Style["m", Italic]assumed},
 .12, 1.5, .01, Appearance → "Labeled", ImageSize → Small},
Delimiter,
Style["reference temperature", Bold, 9],
{{Treference, 20., Style["T", Italic]ref},
 15., 80., Appearance → "Labeled", ImageSize → Small},
Delimiter,
Style["estimated degradation's parameters", Bold, 9],
{{bTrefEst, .0082, Style["b", Italic]Trefest},
 .0001, .2, Appearance → "Labeled", ImageSize → Small},
{{cEst, .158, 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 :→ {tempPro, tFinal1, Cexper1, tFinal2, Cexper2,
  mAssumed, isoT1, isoT2, Treference, bTrefEst, cEst, tAxisMax, TAxisMax}]
```

Out[25]=

temperature profile isothermal dynamic

kinetic parameter estimation

t_{final1} 120.

C_{exper1} 0.5

t_{final2} 105.

C_{exper2} 0.85

constant temperatures

T_1 (°C) 30.

T_2 (°C) 20.

assumed shape factor

$m_{assumed}$ 0.9

reference temperature

T_{ref} 20.

estimated degradation's parameters

$b_{Trefest}$ 0.0082

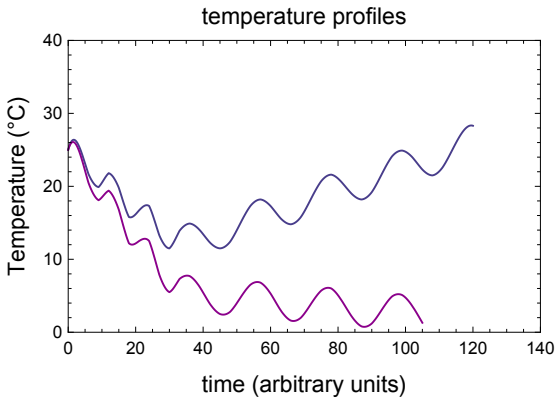
c_{est} 0.158

time axis maximum

t_{max} 140.

Temperature axis maximum

T_{max} 40.



m_{as}= 0.9 T_{ref}= 20. $b_{Trefest}$= 0.008 c_{est}= 0.158