Weibullian Chemical Degradation: Kinetic Parameters Estimation by the Two Endpoints Method (from Two Sets of Interpolated Temperature Profile Data) Version B - Heat Processing.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 $d\text{Log}[C_1(t)]/dt = f_1[T_1(t)]$ and $d\text{Log}[C_2(t)]/dt = f_2[T_2(t)]$, 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, the corresponding times of their determinations, t_1 and t_2, and the reference temperature, T_{ref}.

The parameters' settings in this program (Version B) are for heat processing temperatures. For storage temperatures use Version A.

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.

Clear all variables in all contexts.

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

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, 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 tempData1B. 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 “tempData1.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 tempData1B matrix.

```
(*tempData1B=Import["tempData1B.TSV"]*)
```

```
tempData1B = {{0.`, 60.`}, {0.5`, 60.21442467914263`}, {1.`, 60.852695964467564`},
  {1.5`, 61.899999622921136`}, {2.`, 63.332281480947934`},
  {2.5`, 65.11716561502224`}, {3.`, 67.21517009036855`}, {3.5`, 69.58115622350573`},
  {4.`, 72.16593667345282`}, {4.5`, 74.91796184500052`}, {5.`, 77.78500333132342`},
  {5.5`, 80.71575726129626`}, {6.`, 83.66129892015448`}, {6.5`, 86.57633205322986`},
  {7.`, 89.42019079896974`}, {7.5`, 92.15756806990885`}, {8.`, 94.75896023509617`},
  {8.5`, 97.20083306434333`}, {9.`, 99.4655271496914`}, {9.5`, 101.54093172744257`},
  {10.`, 103.4199635551357`}, {10.5`, 105.09989209785462`},
  {11.`, 106.58155385433724`}, {11.5`, 107.86849753860666`},
  {12.`, 108.9660985364406`}, {12.5`, 109.88067620527615`},
  {13.`, 110.61864186754536`}, {13.5`, 111.18569945206035`},
  {14.`, 111.58611531615975`}, {14.5`, 111.82206940778707`},
  {15.`, 111.89309707728044`}, {15.5`, 111.79562988450462`},
  {16.`, 111.52264490204746`}, {16.5`, 111.06343538057601`},
  {17.`, 110.40352113872767`}, {17.5`, 109.52472436992444`},
  {18.`, 108.40544513267139`}, {18.5`, 107.02117962187657`},
  {19.`, 105.34533188906003`}, {19.5`, 103.35037379937593`},
  {20.`, 101.00940570753004`}, {20.5`, 98.29815782355026`},
  {21.`, 95.19744512311841`}, {21.5`, 91.6960424426066`},
  {22.`, 87.79387753520669`}, {22.5`, 83.5053474282347`}, {23.`, 78.8624514783646`},
  {23.5`, 73.91731493125714`}, {24.`, 68.74357178588501`},
  {24.5`, 63.43601901481958`}, {25.`, 58.10798849438553`},
  {25.5`, 52.88605241876191`}, {26.`, 47.90201302037292`},
  {26.5`, 43.28262410293211`}, {27.`, 39.138089722127816`},
  {27.5`, 35.55095470115728`}, {28.`, 32.56735476501829`},
  {28.5`, 30.192531626698134`}, {29.`, 28.391911335342762`},
  {29.5`, 27.09792698630967`}, {30.`, 26.22139327151459`}}
```

```
{{0., 60.}, {0.5, 60.2144}, {1., 60.8527}, {1.5, 61.9}, {2., 63.3323}, {2.5, 65.1172},
 {3., 67.2152}, {3.5, 69.5812}, {4., 72.1659}, {4.5, 74.918}, {5., 77.785},
 {5.5, 80.7158}, {6., 83.6613}, {6.5, 86.5763}, {7., 89.4202}, {7.5, 92.1576},
 {8., 94.759}, {8.5, 97.2008}, {9., 99.4655}, {9.5, 101.541}, {10., 103.42},
 {10.5, 105.1}, {11., 106.582}, {11.5, 107.868}, {12., 108.966}, {12.5, 109.881},
 {13., 110.619}, {13.5, 111.186}, {14., 111.586}, {14.5, 111.822}, {15., 111.893},
 {15.5, 111.796}, {16., 111.523}, {16.5, 111.063}, {17., 110.404}, {17.5, 109.525},
 {18., 108.405}, {18.5, 107.021}, {19., 105.345}, {19.5, 103.35}, {20., 101.009},
 {20.5, 98.2982}, {21., 95.1974}, {21.5, 91.696}, {22., 87.7939}, {22.5, 83.5053},
 {23., 78.8625}, {23.5, 73.9173}, {24., 68.7436}, {24.5, 63.436}, {25., 58.108},
 {25.5, 52.8861}, {26., 47.902}, {26.5, 43.2826}, {27., 39.1381}, {27.5, 35.551},
 {28., 32.5674}, {28.5, 30.1925}, {29., 28.3919}, {29.5, 27.0979}, {30., 26.2214}}
```

Display the number of time-temperature data values in matrix tempData1B and assign it to nPts1.

```
nPts1 = Length[tempData1B]
```

```
61
```

Display the minimum temperature value in tempData1B.

```
Min[tempData1B[[All, 2]]]
```

```
26.2214
```

Display the maximum temperature value in tempData1B.

```
Max[tempData1B[[All, 2]]]
```

```
111.893
```

Import the second set of experimentally determined {time, temperature} data from a second tab-separated-values (.TSV) ASCII text file into a matrix called tempData2B. 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 "tempData2.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 tempData2B matrix.

```
(*tempData2B=Import["tempData2B.TSV"]*)
```

```
tempData2B = {{0.`, 60.`}, {0.5`, 60.746262468757905`}, {1.`, 62.940792048771854`},
  {1.5`, 66.45516069010628`}, {2.`, 71.08921122381109`}, {2.5`, 76.58991055433475`},
  {3.`, 82.67412789871628`}, {3.5`, 89.05245763889086`}, {4.`, 95.4512666554964`},
  {4.5`, 101.63061713355084`}, {5.`, 107.3964824300489`},
  {5.5`, 112.6065768276101`}, {6.`, 117.16999377404103`},
  {6.5`, 121.0415527437714`}, {7.`, 124.21220571847448`},
  {7.5`, 126.69702724717582`}, {8.`, 128.52224333415055`},
  {8.5`, 129.71252464802697`}, {9.`, 130.27948308267923`},
  {9.5`, 130.21207480873676`}, {10.`, 129.46952042725965`},
  {10.5`, 127.9774673738314`}, {11.`, 125.62845911412062`},
  {11.5`, 122.28828419943956`}, {12.`, 117.81027998025101`},
  {12.5`, 112.05980016278646`}, {13.`, 104.95022463452177`},
  {13.5`, 96.48930255348303`}, {14.`, 86.82958784518725`},
  {14.5`, 76.30940340112963`}, {15.`, 65.46333353412646`},
  {15.5`, 54.97890688816837`}, {16.`, 45.58631797528321`},
  {16.5`, 37.895505554978335`}, {17.`, 32.23291378606156`},
  {17.5`, 28.554674260686088`}, {18.`, 26.492812316623585`},
  {18.5`, 25.520433206310024`}, {19.`, 25.14569472852294`},
  {19.5`, 25.0315296466186`}, {20.`, 25.005050636811006`}}
```

```
{{0., 60.}, {0.5, 60.7463}, {1., 62.9408}, {1.5, 66.4552}, {2., 71.0892},
 {2.5, 76.5899}, {3., 82.6741}, {3.5, 89.0525}, {4., 95.4513}, {4.5, 101.631},
 {5., 107.396}, {5.5, 112.607}, {6., 117.17}, {6.5, 121.042}, {7., 124.212},
 {7.5, 126.697}, {8., 128.522}, {8.5, 129.713}, {9., 130.279}, {9.5, 130.212},
 {10., 129.47}, {10.5, 127.977}, {11., 125.628}, {11.5, 122.288},
 {12., 117.81}, {12.5, 112.06}, {13., 104.95}, {13.5, 96.4893}, {14., 86.8296},
 {14.5, 76.3094}, {15., 65.4633}, {15.5, 54.9789}, {16., 45.5863},
 {16.5, 37.8955}, {17., 32.2329}, {17.5, 28.5547}, {18., 26.4928},
 {18.5, 25.5204}, {19., 25.1457}, {19.5, 25.0315}, {20., 25.0051}}
```

Display the number of time-temperature data values in matrix tempData2B and assign it to nPts2.

```
nPts2 = Length[tempData2B]
```

```
41
```

Display the minimum temperature value in tempData2B.

```
Min[tempData2B[[All, 2]]]
```

```
25.0051
```

Display the maximum temperature value in tempData2B.

```
Max[tempData2B[[All, 2]]]
```

```
130.279
```

Assign tAxisMax1 and tMax1 to be the maximum time value found in matrix tempData1B.

```
tAxisMax1 = Max[tempData1B[[All, 1]]]; tMax1 = tAxisMax1
```

```
30.
```

Assign tAxisMax2 and tMax2 to be the maximum time value found in matrix tempData2B.

```
tAxisMax2 = Max[tempData2B[[All, 1]]]; tMax2 = tAxisMax2
```

```
20.
```

Assign TAxisMax1 and TMax1 to be the maximum Temperature value found in matrix tempData1B.

```
TAxisMax1 = Max[tempData1B[[All, 2]]]; TMax1 = TAxisMax1
```

```
111.893
```

```
TAxisMax1 = 120.
```

```
120.
```

Assign TAxisMax2 and TMax2 to be the maximum Temperature value found in matrix tempData2B.

```
TAxisMax2 = Max[tempData2B[[All, 2]]]; TMax2 = TAxisMax2
```

```
130.279
```

```
TAxisMax2 = 140.
```

```
140.
```

Assign T1 to be an interpolating function defined from the tempData1B points.

```
T1 = Interpolation[tempData1B]
```

```
InterpolatingFunction[ Domain: {{0., 30.}}
                       Output: scalar ]
```

Assign T2 to be an interpolating function defined from the tempData2B points.

```
T2 = Interpolation[tempData2B]
```

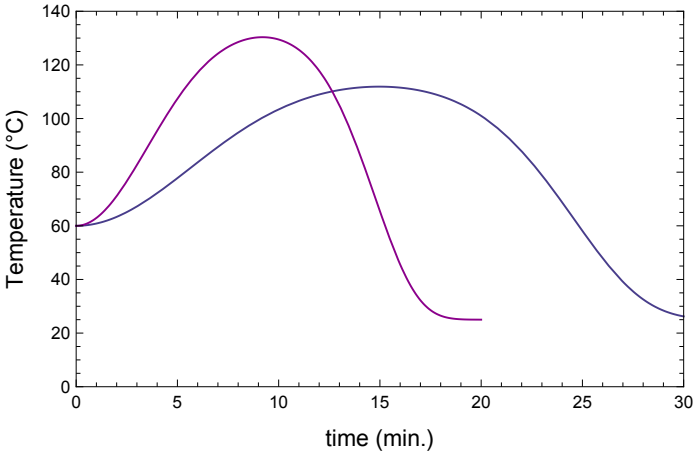
```
InterpolatingFunction[ Domain: {{0., 20.}}
                       Output: scalar ]
```

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

```
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]}];
```

```
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]}];
```

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



Define lnConcRatio(t), the natural log of the momentary concentration of the degrading compound, C(t), returned by NDSolve.

```
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] == -0.000001},
     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.

```
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[{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]];
  T1Plot = Plot[T1[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[T2[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, T1]],
```

```
   {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, T2]],
   {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]}]
],
Style["kinetic parameter estimation", Bold, 9],
{{tFinal1, tMax1, Style["t", Italic]final1},
 10., 60., Appearance → "Labeled", ImageSize → Small},
{{Cexper1, .8, Style["C", Italic]exper1}, .1, .95,
 Appearance → "Labeled", ImageSize → Small},
"\n",
{{tFinal2, tMax2, Style["t", Italic]final2},
 10., 60., Appearance → "Labeled", ImageSize → Small},
{{Cexper2, .6, Style["C", Italic]exper2}, .1, .95,
 Appearance → "Labeled", ImageSize → Small},
Delimiter,
Style["assumed shape factor", Bold, 9],
{{mAssumed, .9, Style["m", Italic]assumed},
 .2, 1.5, .01, Appearance → "Labeled", ImageSize → Small},
Delimiter,
Style["reference temperature", Bold, 9],
{{Treference, 100., Style["T", Italic]ref},
 50., 110., Appearance → "Labeled", ImageSize → Small},
Delimiter,
Style["estimated degradation's parameters", Bold, 9],
{{bTrefEst, .0112, Style["b", Italic]Trefest},
 .0001, .2, Appearance → "Labeled", ImageSize → Small},
{{cEst, .078, Style["c", Italic]est}, .0001, .2, Appearance → "Labeled",
 ImageSize → Small},
Delimiter,
Style["time axis maximum", Bold, 9],
{{tAxisMax, 40., Style["t", Italic]max},
 10., 120., Appearance → "Labeled", ImageSize → Small},
Style["Temperature axis maximum", Bold, 9],
{{TAxisMax, 140., Style["T", Italic]max},
```

```
 20., 150., Appearance → "Labeled", ImageSize → Small},
ControlPlacement → Left, SaveDefinitions → True, AutorunSequencing → Automatic,
TrackedSymbols :→ {tFinal1, Cexper1, tFinal2, Cexper2,
  mAssumed, Treference, bTrefEst, cEst, tAxisMax, TAxisMax}]
```