Shelf Life Determination by Two Criteria from Digitized Temperature Data.nb

Programmed by: Mark D. Normand Last modified: June 29, 2015

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

This program allows the user to estimate the times, t_{c1} & t_{c2}, at which the degradation curves of two nutrients will cross their respective threshold concentration ratios $Conc_{c1}$ & $Conc_{c2}$, given that the storage temperature profile is entered as a digitized time-temperature data file. The assumptions are that in the pertinent temperature range the degradation of the two nutrients follows fixed order kinetics, $n \geq 0$ [1], and that the temperature-dependence of the corresponding rate constant $k(T)$ follows the exponential model [1, 2], i.e., $k(T(t)) = k_{Tref} * \exp(c*(T(t) - T_{ref}))$. Thus, the other entered values are the assumed kinetic orders, n_1 & n_2, the reference temperatures, T_{ref1} & T_{ref2}, the rate constants at the corresponding reference temperatures k_{Tref1} & k_{Tref2}, the constants c_1 & c_2 and the threshold concentration ratios $Conc_{c1}$ & $Conc_{c2}$.

To assure a numerical solution with the FindRoot function, the reader can move the t_{01} & t_{02} sliders close to the intersection points which will be used as initial guesses of the sought times.

The storage time t_{max} and temperature range, T_{min} & T_{max}, can also be set with sliders.

The Manipulate panel display includes the temperature data in the form of an interpolated function plot (top), the calculated numerical values of the two threshold crossing times, t_{c1} & t_{c2}, (middle) and plots of the two degradation curves with their corresponding threshold levels shown as dashed lines (bottom). The intersection points are plotted as colored dots and the chosen initial guesses in slightly paler colors of the same hue.

For comparison, the program can also be used for isothermal storage by clicking on the isothermal checkbox and then setting the temperature with the T slider below the checkbox.

WARNING: Note that not all possible entry combinations necessarily have a solution within the specified time range.

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.

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, 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.

```
tempDataFileName = "testData.tsv";
```

```
(*testData=Import[tempDataFileName]*)
```

```
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 and assign the maximum time value in testData.

```
tAxisMax = Max[testData[[All, 1]]]
```

```
120.
```

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]]]
```

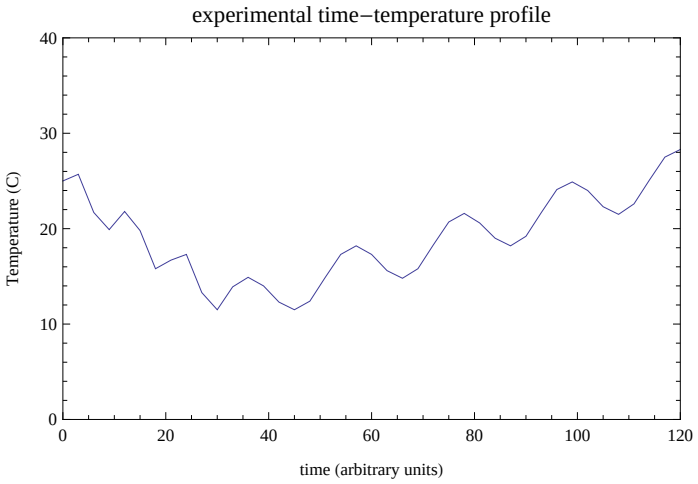
```
28.3
```

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

```
TInterpData = Interpolation[testData]
```

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

Test calls to the interpolating function at various times.

```
TInterpData[0.]
```

```
25.
```

```
TInterpData[1.]
```

```
26.1815
```

```
TInterpData[10.]
```

```
20.4975
```

```
TInterpData[50.]
```

```
13.9938
```

```
TInterpData[100.]
```

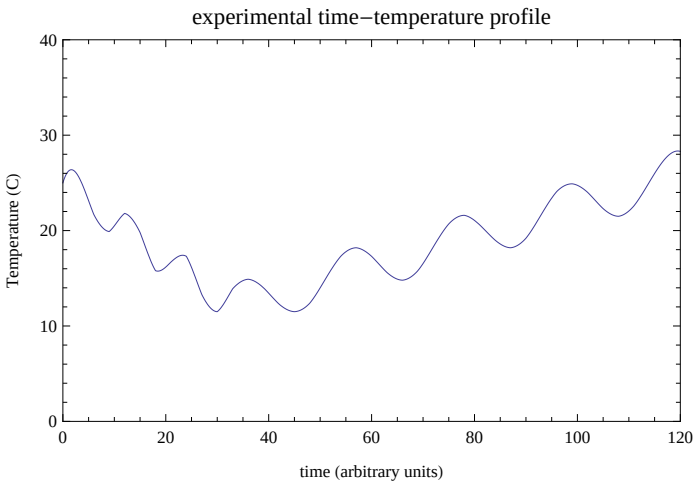
```
24.7444
```

```
TInterpData[120.]
```

```
28.3
```

Plot the interpolating function defined from the nPts testData points.

```
TInterpDataPlot = Plot[TInterpData[t], {t, 0., 120.},
  PlotRange → {{0., tAxisMax}, {TAxisMin, TAxisMax}}, Frame → True,
  FrameLabel → {"time (arbitrary units)", "Temperature (C)"},
  PlotLabel → "experimental time-temperature profile"]
```



Define concRatioAnalytic(t) (concentration ratio vs. time), the degradation curve's equation for the chosen kinetic parameters and isothermal temperature profile. *Note that* Temp *refers to the isothermal value of the T slider in the Manipulate panel.*

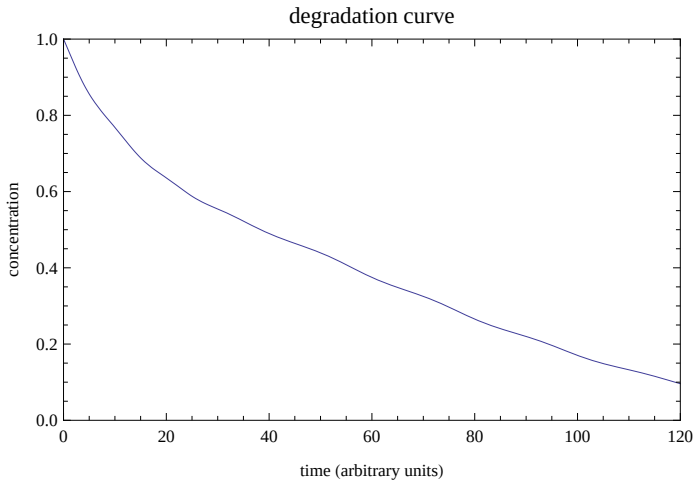
```
concRatioAnalytic[t_?NumericQ, n_?NumericQ,
  Tref_?NumericQ, kTref_?NumericQ, c_?NumericQ, Temp_] :=
 Module[{C0 = 1., k1, fn}, k1[tt_] := kTref * Exp[c * (Temp[tt] - Tref)];
  fn[tt_] := Piecewise[{{C0 - k1[t] * t, n == 0 && C0 - k1[t] * t > 0},
     {0, n == 0 && C0 - k1[t] * t ≤ 0}, {C0 * Exp[-k1[t] * t], n == 1},
     {(C0^(1-n) + k1[t] (-1 + n) t)^(1/(1-n)), n > 1}, {(C0^(1-n) + k1[t] (-1 + n) t)^(1/(1-n)),
      n < 1 && t < C0^(1-n)/(k1[t] (1 - n))}, {0, n < 1 && t ≥ C0^(1-n)/(k1[t] (1 - n))}}];
  If[Re[fn[t]] ≤ .001 || Im[fn[t]] > 0., 0., fn[t]]]
```

Define concRatio(time) (concentration ratio vs. time), the degradation curve's equation for the chosen kinetic parameters and nonisothermal temperature profile from the interpolating function solution returned by NDSolve. *Notice that if the concentration ratio where n=0 becomes negative or where 0<n<1 the ratio becomes a complex number, the concentration is assigned a value of zero. However, in this region the method is not viable (see below).* *Note that* Temp *refers to the global interpolation*

function defined from the testData values.

```
concRatio[time_?NumericQ, n_?NumericQ, Tref_?NumericQ,
  kTref_?NumericQ, c_?NumericQ, Temp_] := Module[{k1, sfn1, y},
  k1[t_] := kTref * Exp[c * (Temp[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] == 1.}, y[t], {t, 0., time}][[1, 1, 2]];
  If[sfn1[time] ≤ 0. || Im[sfn1[time]] > 0., 0., sfn1[time]]]
```

```
Plot[concRatio[t, 1., 25., .03, .08, TInterpData],
 {t, 0., 120.}, PlotRange → {{0., tAxisMax}, {0., 1.}}, Frame → True,
 FrameLabel → {"time (arbitrary units)", "concentration"},
 PlotLabel → "degradation curve"]
```



Initialize the global error messages lists for use in debugging.

```
errMsgs1 = {}; errMsgs2 = {};
```

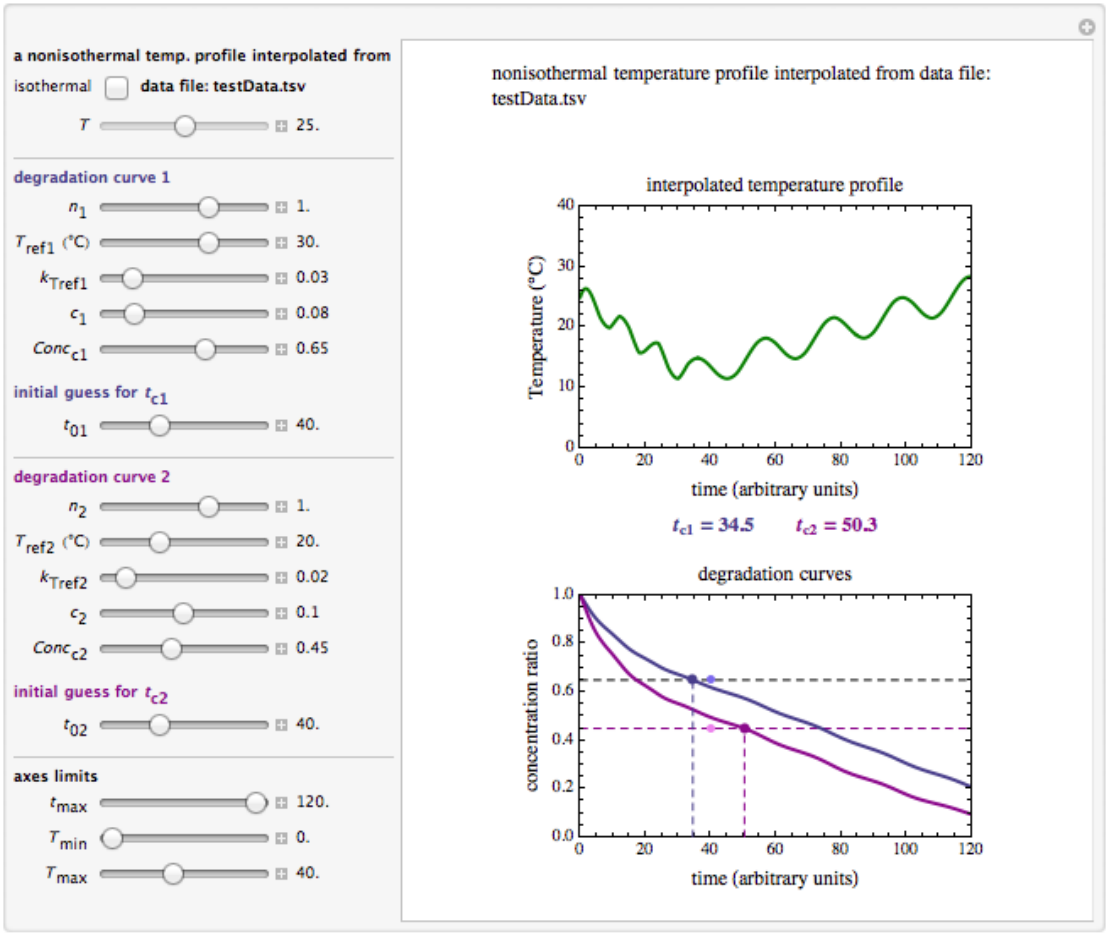
The following Manipulate panel compares two markers for shelf life determination using the concRatioAnalytic (isothermal) and concRatio (nonisothermal) functions.

```
Manipulate[Module[{conc, concRatio1Plot, concRatio2Plot, root1, root2, tc1Text,
    tc2Text, tc1tc2Text, Temp, TEqnText, TParamText, TProfilePlot, tempTitle},
  If[isoTemp,
   tempTitle = "storage temperature"; TEqnText = "";
   TParamText = Text[Style[Row[{Style["T", Italic], " = ", TT}], Black, 12]];
   (* The isothermal Temp temperature
    profile function is the current value of the TT slider. *)
   Temp[t_] := TT;
   conc[tc_, n_, Tref_, kTref_, c_, T_] :=
    concRatioAnalytic[tc, n, Tref, kTref, c, T];
   tc1 = .; errMsgs1 = {}; root1OK = True;
   Quiet[root1 = FindRoot[conc[tc1, n1, Tref1, kTref1, c1, Temp] == Cc1, {tc1, t01}];
    If[Length[$MessageList] == 0, root1OK = True; errMsgs1 = {},
     root1OK = False; errMsgs1 = $MessageList; $MessageList = {}],
    All];
   If[root1OK && ValueQ[root1],
    tc1 = root1[[1, 2]]; tc1Text =
     Style[Row[{Subscript[Style["t", Italic], "c1"], " = ", Round[tc1, .1]}],
      ColorData["HTML", "DarkSlateBlue"], Bold],
    tc1 = .; tc1Text = Style[Row[{Subscript[Style["t", Italic], "c1"], " = "}],
```

```
    Red, Bold];
   Beep[]
  ];
  tc2 =.; errMsgs2 = {}; root2OK = True;
  Quiet[root2 = FindRoot[conc[tc2, n2, Tref2, kTref2, c2, Temp] == Cc2, {tc2, t02}];
   If[Length[$MessageList] == 0, root2OK = True; errMsgs2 = {},
    root2OK = False; errMsgs2 = $MessageList; $MessageList = {}],
   All];
  If[root2OK && ValueQ[root2],
   tc2 = root2[[1, 2]];
   tc2Text =
    Style[Row[{Subscript[Style["t", Italic], "c2"], " = ", Round[tc2, .1]}],
     ColorData["HTML", "DarkMagenta"], Bold],
   tc2 =.; tc2Text = Style[Row[{Subscript[Style["t", Italic], "c2"], " = "}],
     Red, Bold];
   Beep[]; Beep[]
  ]
  ,
  tempTitle = "interpolated temperature profile"; TParamText = "";
  TEqnText = Text@Style[Row[{Spacer[40],
       "nonisothermal temperature profile interpolated from data file:\n",
       Spacer[40], tempDataFileName}], Black, 12];
  (* Temp is the temperature profile function interpolated from
   the data file imported into the testData matrix. *)
  Temp = TInterpData;
  conc[tc_, n_, Tref_, kTref_, c_, T_] := concRatio[tc, n, Tref, kTref, c, Temp];
  tc1 =.; errMsgs1 = {}; root1OK = True;
  Quiet[root1 = FindRoot[conc[tc1, n1, Tref1, kTref1, c1, Temp] == Cc1, {tc1, t01}];
   If[Length[$MessageList] == 0, root1OK = True; errMsgs1 = {},
    root1OK = False; errMsgs1 = $MessageList; $MessageList = {}],
   All];
  If[root1OK && ValueQ[root1],
   tc1 = root1[[1, 2]]; tc1Text =
    Style[Row[{Subscript[Style["t", Italic], "c1"], " = ", Round[tc1, .1]}],
     ColorData["HTML", "DarkSlateBlue"], Bold],
   tc1 =.; tc1Text = Style[Row[{Subscript[Style["t", Italic], "c1"], " = "}],
     Red, Bold];
   Beep[]
  ];
  tc2 =.; errMsgs2 = {}; root2OK = True;
  Quiet[root2 = FindRoot[conc[tc2, n2, Tref2, kTref2, c2, Temp] == Cc2, {tc2, t02}];
   If[Length[$MessageList] == 0, root2OK = True; errMsgs2 = {},
    root2OK = False; errMsgs2 = $MessageList; $MessageList = {}],
   All];
  If[root2OK && ValueQ[root2],
   tc2 = root2[[1, 2]];
   tc2Text =
    Style[Row[{Subscript[Style["t", Italic], "c2"], " = ", Round[tc2, .1]}],
     ColorData["HTML", "DarkMagenta"], Bold],
   tc2 =.; tc2Text = Style[Row[{Subscript[Style["t", Italic], "c2"], " = "}],
     Red, Bold];
   Beep[]; Beep[]
  ]
 ];
 tc1tc2Text = Text[Style[Row[{tc1Text, Spacer[25], tc2Text}], 12]];
```

```
 TProfilePlot = Plot[ Temp[t], {t, 0., tAxisMax},
   PlotRange → {{0., tAxisMax}, {tempAxisMin, tempAxisMax}},
   PlotStyle → {AbsoluteThickness[2], ColorData["HTML", "Green"]},
   Frame → True, FrameStyle → Directive[Thin, Black], PlotLabel -> TParamText,
   FrameLabel → {{Style["Temperature (℃)", 12], ""},
     {Style["time (arbitrary units)", 12], Style[tempTitle, 12]}},
   ImagePadding → {{45, 10}, {35, 35}}, ImageSize → {380, 225}];
 concRatio1Plot = Plot[conc[t, n1, Tref1, kTref1, c1, Temp],
   {t, 0.001, tAxisMax}, PlotRange → {{0., tAxisMax}, {0., 1.}},
   PlotStyle → {AbsoluteThickness[2], ColorData["HTML", "DarkSlateBlue"]},
   Frame → True, FrameStyle → Directive[Thin, Black],
   ImagePadding → {{45, 10}, {35, 35}}, ImageSize → {380, 225},
   Prolog → {Dashed, Line[{{0., Cc1}, {tAxisMax, Cc1}}],
     ColorData["HTML", "DarkMagenta"], Dashed, Line[{{0., Cc2}, {tAxisMax, Cc2}}]},
   Epilog → {ColorData["HTML", "MediumSlateBlue"], AbsolutePointSize[5],
     Point[{t01, Cc1}], ColorData["HTML", "Violet"],
     Point[{t02, Cc2}], ColorData["HTML", "DarkSlateBlue"],
     If[root1OK, {ColorData["HTML", "DarkSlateBlue"], Dashed,
       Line[{{tc1, 0.}, {tc1, Cc1}}], AbsolutePointSize[6], Point[{tc1, Cc1}]}],
     If[root2OK, {ColorData["HTML", "DarkMagenta"], Dashed,
       Line[{{tc2, 0.}, {tc2, Cc2}}], AbsolutePointSize[6], Point[{tc2, Cc2}]}]}];
 concRatio2Plot = Plot[conc[t, n2, Tref2, kTref2, c2, Temp],
   {t, 0.001, tAxisMax}, PlotRange → {{0., tAxisMax}, {0., 1.}},
   PlotStyle → {AbsoluteThickness[2], ColorData["HTML", "DarkMagenta"]},
   Frame → True, FrameStyle → Directive[Thin, Black],
   ImagePadding → {{45, 10}, {35, 35}}, ImageSize → {380, 225}];
 Column[{Pane[TEqnText, ImageSize → {380, 32}], TProfilePlot,
   Show[concRatio1Plot, concRatio2Plot, PlotLabel -> tc1tc2Text,
    FrameLabel → {{Style["concentration ratio", 12], ""},
      {Style["time (arbitrary units)", 12], Style["degradation curves", 12]}}]}]
],
PaneSelector[{True ->
   Row[{Spacer[35], Style["set the isothermal temperature", Bold, 9], Spacer[5]}],
  False -> Row[{Style["a nonisothermal temp. profile interpolated from", Bold, 9],
     Spacer[2]}]}, isoTemp],
Row[{"isothermal", Control[{{isoTemp, False, ""}, {True, False}}],
  Spacer[5], PaneSelector[{True → Null,
    False → Style[Row[{"data file: ", tempDataFileName}], Bold, 9]}, isoTemp]}],
{{TT, 25., Style["T", Italic]}, 0., 50., Appearance → "Labeled",
 ImageSize → Small, Enabled → If[isoTemp, True, False]},
Delimiter,
Style["degradation curve 1", ColorData["HTML", "DarkSlateBlue"], Bold, 9],
{{n1, 1., Subscript[Style["n", Italic], "1"]},
 0., 1.5, Appearance → "Labeled", ImageSize → Small},
{{Tref1, 30., Row[{Subscript[Style["T", Italic], "ref1"], " (℃)"}]},
 10., 40., Appearance → "Labeled", ImageSize → Small},
{{kTref1, .03, Subscript[Style["k", Italic], "Tref1"]},
 .001, .2, Appearance → "Labeled", ImageSize → Small},
{{c1, .08, Subscript[Style["c", Italic], "1"]}, .001, .5, Appearance → "Labeled",
 ImageSize → Small},
{{Cc1, .65, Subscript[Style["Conc", Italic], "c1"]},
 .1, .95, Appearance → "Labeled",
 ImageSize → Small},
"\n",
Style[Row[{"initial guess for ", Subscript[Style["t", Italic], "c1"]}],
```

```
 ColorData["HTML", "DarkSlateBlue"], Bold, 9],
{{t01, 40., Subscript[Style["t", Italic], "01"]}, 0.,
 tAxisMax, Appearance → "Labeled", ImageSize → Small},
Delimiter,
Style["degradation curve 2", ColorData["HTML", "DarkMagenta"], Bold, 9],
{{n2, 1., Subscript[Style["n", Italic], "2"]},
 0., 1.5, Appearance → "Labeled", ImageSize → Small},
{{Tref2, 20., Row[{Subscript[Style["T", Italic], "ref2"], " (℃)"}]},
 10., 40., Appearance → "Labeled", ImageSize → Small},
{{kTref2, .02, Subscript[Style["k", Italic], "Tref2"]},
 .001, .2, Appearance → "Labeled", ImageSize → Small},
{{c2, .1, Subscript[Style["c", Italic], "2"]}, .001, .2, Appearance → "Labeled",
 ImageSize → Small},
{{Cc2, .45, Subscript[Style["Conc", Italic], "c2"]},
 .1, .95, Appearance → "Labeled",
 ImageSize → Small},
"\n",
Style[Row[{"initial guess for ", Subscript[Style["t", Italic], "c2"]}],
 ColorData["HTML", "DarkMagenta"], Bold, 9],
{{t02, 40., Subscript[Style["t", Italic], "02"]}, 0.,
 tAxisMax, Appearance → "Labeled", ImageSize → Small},
Delimiter,
Style["axes limits", Bold, 9],
{{tAxisMax, 120., Subscript[Style["t", Italic], "max"]},
 10., 120., Appearance → "Labeled", ImageSize → Small},
{{tempAxisMin, 0., Subscript[Style["T", Italic], "min"]},
 0., 20., Appearance → "Labeled", ImageSize → Small},
{{tempAxisMax, 40., Subscript[Style["T", Italic], "max"]},
 25., 60., Appearance → "Labeled", ImageSize → Small},
{tc1, 0., 0., ControlType → None},
{root1OK, True, True, ControlType → None},
(*{errMsgs1,{},{},ControlType→None},*)
{tc2, 0., 0., ControlType → None},
{root2OK, True, True, ControlType → None},
(*{errMsgs2,{},{},ControlType→None},*)
ControlPlacement → Left, SaveDefinitions → True, AutorunSequencing → {1, 2, 8},
TrackedSymbols :→ {isoTemp, TT, n1, Tref1, kTref1, c1, Cc1, t01, n2,
  Tref2, kTref2, c2, Cc2, t02, tAxisMax, tempAxisMin, tempAxisMax}]
```



errMsgs1

{}

errMsgs2

{}

The default panel display for an isothermal case.

set the isothermal temperature

isothermal ☑

T 25.

degradation curve 1

n_1 1.

T_{ref1} (°C) 30.

k_{Tref1} 0.03

c_1 0.08

$Conc_{c1}$ 0.65

initial guess for t_{c1}

t_{01} 40.

degradation curve 2

n_2 1.

T_{ref2} (°C) 20.

k_{Tref2} 0.02

c_2 0.1

$Conc_{c2}$ 0.45

initial guess for t_{c2}

t_{02} 40.

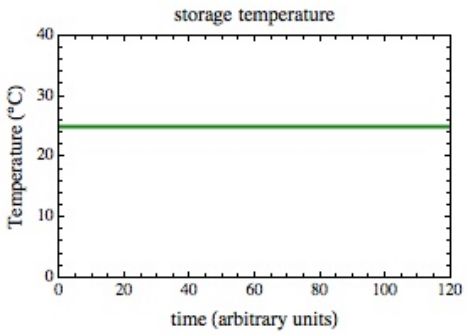
axes limits

t_{max} 120.

T_{min} 0.

T_{max} 40.

$T = 25.$



$t_{c1} = 21.4$ $t_{c2} = 24.2$