Predicting Nutrient Degradation from Two Successive Concentrations E.nb

Program E: Manual estimation of the kinetic parameters using the Manipulate function and imported experimental time-temperature data

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

This program allows the user to estimate kinetic parameters kTref and c from two entered concentration ratios Cexp1 and Cexp2 at times t1 and t2, respectively, given the temperature profile equation T(t) and assumed kinetic order nAssumed. This is done by attempting to make the generated degradation curve pass through the two points by manual adjustment of these parameters' controls. It is based on the assumptions that in the pertinent temperature range the degradation follows fixed order kinetics, n≥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))

The obtained values of the parameters kTref and c can serve as initial guesses for the FindRoot in the B program or can be used as estimates of the parameters themselves.

The two experimentally determined {time, concentration ratio} data points are entered into the program manually. The temperature profile function in the example below is interpolated from a set of imported experimental data. If the temperature profile function can be expressed algebraically, use program D.

WARNING: Note that not all possible experimental concentration entries (Cexp1 at t1 and Cexp2 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.

Clear all variables in all contexts.

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

Tref is the user-chosen reference temperature for the calculations. It should be in the same temperature range as the data.

```
Tref = 25.;
```

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

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

Enter the nutrient's experimental concentration ratios, Cexp1 and Cexp2, at times t1 and t2.

```
Cexp1 = 0.79
```

```
0.79
```

```
Cexp2 = 0.64
```

```
0.64
```

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 in 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.;
```

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

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

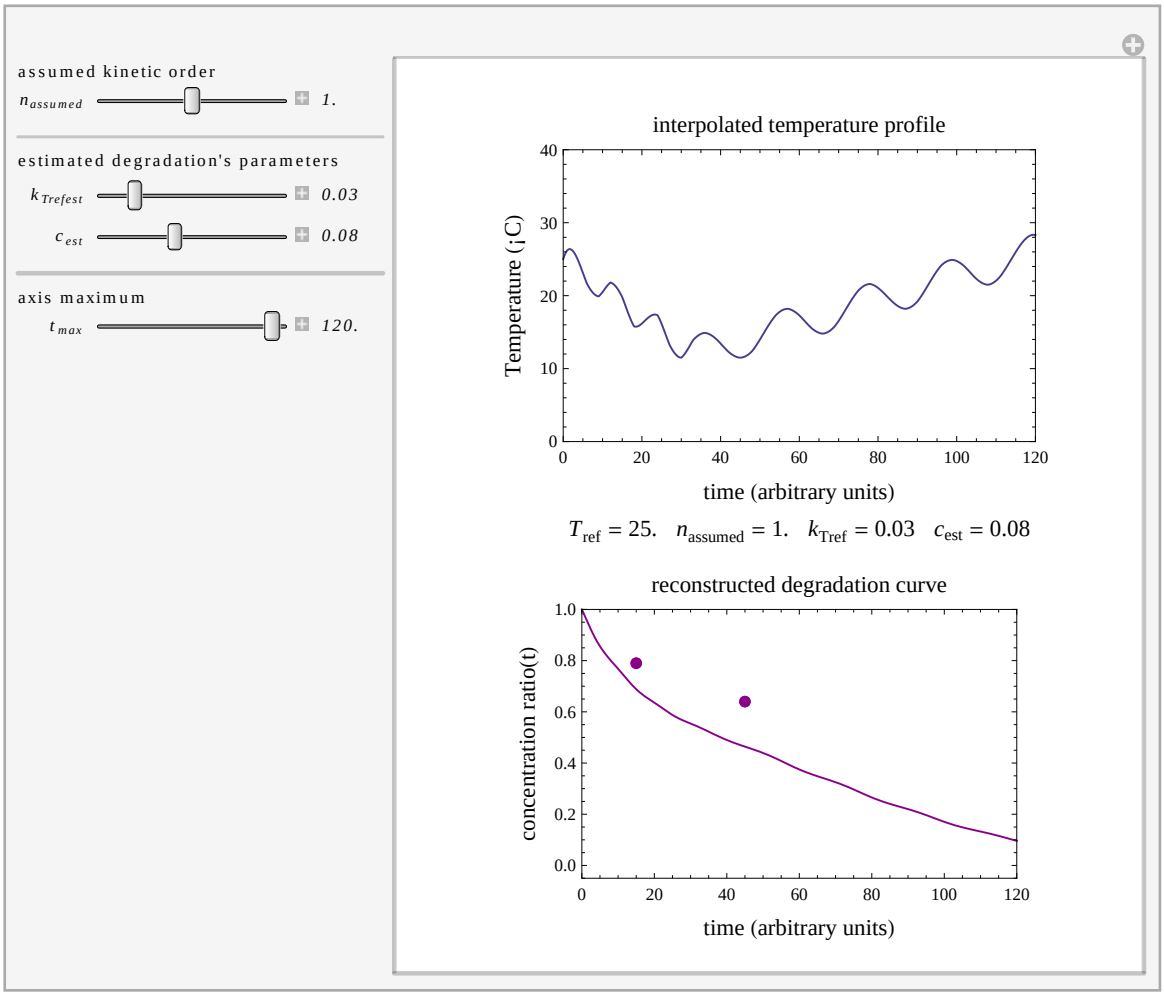
```
InterpolatingFunction[{{0., 120.}}, <>]
```

Define ConcRatio(tFinal), the degradation curve's equation for the chosen kinetic parameters and temperature profile (concentration ratio vs. time) 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).*

```
ConcRatio[tFinal_?NumericQ, nEst_?NumericQ, kTrefEst_?NumericQ,
  cEst_?NumericQ, Tref_?NumericQ, T_] := Module[{k1, sfn1, y},
  k1[t_] := kTrefEst * Exp[cEst * (T[t] - Tref)]; sfn1[t_] = NDSolve[
     {y'[t] == If[nEst == 0, -k1[t], If[nEst == 1, -k1[t] * y[t] , -k1[t] * y[t]^nEst]],
      y[0] == 1.}, y[t], {t, 0., tFinal}][[1, 1, 2]];
  If[sfn1[tFinal] ≤ 0. || Im[sfn1[tFinal]] > 0., 0., sfn1[tFinal]]]
```

The following Manipulate panel assists in selecting initial guesses for the kTrefEst and cEst parameters of the ConcRatio function.

```
Manipulate[
 Module[{CExpPtsPlot, ConcRatioPlot, kTrefcText, TProfilePlot, yAxisMax},
  If[t2 < t1 + 10., t2 = t1 + 10.];
  If[tAxisMax < t2 + 10., tAxisMax = t2 + 10.];
  If[Cexp2 > Cexp1 - .05, Cexp2 = Cexp1 - .05];
  If[Cexp2 < 0., Cexp2 = 0.];
  yAxisMax = 1.;
  kTrefcText =
   Text[Style[Row[{Subscript[Style["T", Italic], "ref"], " = ", Round[Tref, .01],
       Spacer[10], Subscript[Style["n", Italic], "assumed"], " = ",
       Round[nAssumed, .01], Spacer[10], Subscript[Style["k", Italic], "Tref"],
       " = ", Round[kTrefEst, .001], Spacer[10],
       Subscript[Style["c", Italic], "est"], " = ", Round[cEst, .001]}], 12]];
  TProfilePlot = Plot[ T[t], {t, 0., tAxisMax},
    PlotRange → {{0., tAxisMax}, {TAxisMin, TAxisMax}},
    PlotStyle → {AbsoluteThickness[1], ColorData["HTML", "DarkSlateBlue"]},
    Frame → True, FrameLabel →
     {{Style["Temperature (℃)", 12], ""}, {Style["time (arbitrary units)", 12],
       Style["interpolated temperature profile", 12]}},
    ImagePadding → {{45, 10}, {35, 35}}, ImageSize → {360, 222}];
  ConcRatioPlot = Plot[ConcRatio[t, nAssumed, kTrefEst, cEst, Tref, T],
    {t, 0.001, tAxisMax}, 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 curve", 12]}},
    ImagePadding → {{45, 10}, {35, 35}}, ImageSize → {360, 222}];
  CExpPtsPlot = ListPlot[{{{t1, Cexp1}}, {{t2, Cexp2}}},
    PlotRange → {{0., tAxisMax}, {-.05, yAxisMax}},
    PlotStyle → {{AbsolutePointSize[6], ColorData["HTML", "DarkMagenta"]},
      {AbsolutePointSize[6], ColorData["HTML", "DarkMagenta"]}}, Frame → True,
    PlotLabel -> kTrefcText, FrameLabel → {{Style["concentration ratio(t)", 12], ""},
      {Style["time (arbitrary units)", 12]}},
    ImagePadding → {{45, 10}, {35, 35}}, ImageSize → {360, 222}];
  Column[{TProfilePlot, Show[ConcRatioPlot, CExpPtsPlot]}]
 ],
 Style["assumed kinetic order", Bold, 9],
 {{nAssumed, 1., Subscript[Style["n", Italic], "assumed"]},
  .8, 1.2, .05, Appearance → "Labeled", ImageSize → Small},
 Delimiter,
 Style["estimated degradation's parameters", Bold, 9],
 {{kTrefEst, .03, Subscript[Style["k", Italic], "Trefest"]},
  .001, .2, Appearance → "Labeled", ImageSize → Small},
 {{cEst, .08, Subscript[Style["c", Italic], "est"]},
  .001, .2, Appearance → "Labeled",
  ImageSize → Small},
 Delimiter,
 Style["axis maximum", Bold, 9],
 {{tAxisMax, 120., Subscript[Style["t", Italic], "max"]},
  t2 + 10., 120., Appearance → "Labeled", ImageSize → Small},
 ControlPlacement → Left, SaveDefinitions → True, AutorunSequencing → Automatic,
 TrackedSymbols :⇢ {nAssumed, kTrefEst, cEst, tAxisMax}]
```



The following .TIFF image shows the manual match between the two experimental points and the reconstructed degradation curve obtained by moving the kTrfEst and cEst sliders. Use the values of the sliders as the initial guesses for the FindRoot in the B program.

assumed kinetic order

$n_{assumed}$ 1.

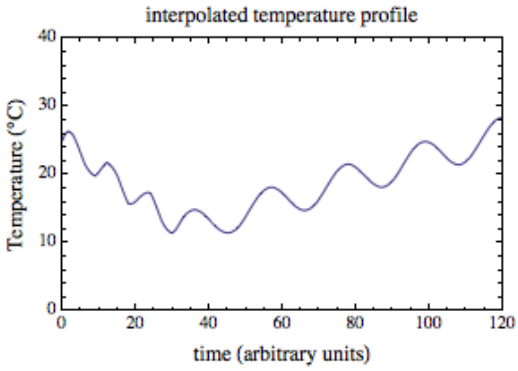
estimated degradation's parameters

$k_{Trefest}$ 0.018

c_{est} 0.08

axis maximum

t_{max} 120.



$T_{ref} = 25.$ $n_{assumed} = 1.$ $k_{Tref} = 0.018$ $c_{est} = 0.08$