Predicting Nutrient Degradation from Two Successive Concentrations B.nb

Program B: Using experimental data and a temperature profile expressed algebraically

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

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

This program predicts chemical degradation of compounds in food such as nutrients or pigments from two successive concentrations or concentration ratios determined experimentally during storage. 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))

In this program, B, the user enters two experimentally determined concentration ratios, CExperimental1 and CExperimental2, taken at two times, t1 and t2, during storage. The program reconstructs the corresponding degradation curve and calculates (predicts) the nutrient's concentration ratio after a chosen time t3 for the same temperature profile.

WARNING: Note that not all possible experimental concentration entries (C1 at t1 and C2 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.

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["`*"]
```

Enter Tref, the user-chosen reference temperature for the calculations.

```
Tref = 20.;
```

Assign nAssumed, the assumed kinetic order.

```
nAssumed = 1.;
```

Assign tAxisMax and yAxisMax, the maximum values of the plot's x- and y-axes, respectively.

```
tAxisMax = 120.; yAxisMax = 1 + .01;
```

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

```
t1 = 10.; t2 = 25.;
```

Assign t3, the time at which we want to predict the concentration ratio in the pertinent time units, e.g., days, weeks, months. (Note that there are conditions where this value might be lowered by the program.)

```
t3 = 85.;
```

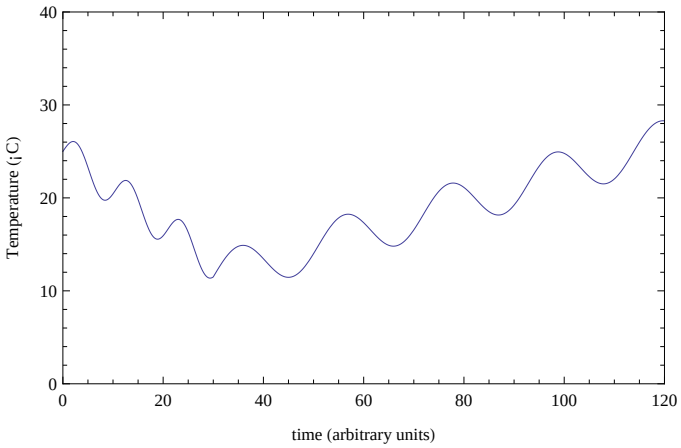
Define your temperature profile function T(t), which can include "If" statements as in the example below. (The example function defined and plotted below is of a fluctuating temperature having a downward followed by an upward trend.)

```
T0 = 25.; a1 = .4; a2 = 2.; a3 = .6;
```

```
T[t_, T0_, a1_, a2_, a3_] :=
  If[t ≤ 30., T0 - a1 * t + a2 * Sin[a3 * t], T0 - 30. * a1 + a2 * Sin[30. * a3] +
    .4 * a1 * (t - 30.) + 1.25 * a2 * Sin[.5 * a3 * (t - 30.)]];
```

Plot the temperature profile T(t) with t varying from 0 to tAxisMax.

```
Plot[ T[t, T0, a1, a2, a3], {t, 0., tAxisMax}, PlotRange → {{0., tAxisMax}, {0., 40.}},
 Frame → True, FrameLabel → {"time (arbitrary units)", "Temperature (℃)", "", ""}]
```



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 where n=0, the concentration ratio can become negative. Or where 0<n<1, the concentration ratio can become a complex number. Where this occurs the concentration ratio is assigned a value of zero. However, in this region the method is not viable (see below).* Note that this ConcRatio function has 9 arguments, one fewer than the Conc function used in the Version A program. ConcRatio does not have argument C0 which is the 5th argument to Conc.

```
ConcRatio[tFinal_?NumericQ, nEst_?NumericQ,
  kTrefEst_?NumericQ, cEst_?NumericQ, Tref_?NumericQ, T0_?NumericQ,
  a1_?NumericQ, a2_?NumericQ, a3_?NumericQ] := Module[{k1, sfn1, y},
  k1[t_] := kTrefEst * Exp[cEst * (T[t, T0, a1, a2, a3] - 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]]]
```

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

```
CExperimental1 = 0.75
```

```
0.75
```

```
CExperimental2 = 0.6
```

```
0.6
```

Estimate the kinetic parameters kTrefEst and cEst. *To find the user-entered initial guesses for kTrefEst and cEst use the attached Mathematica notebook "PredictingNutrientDegradationFromTwoSuccessiveConcentrationsD.nb".*

```
Clear[kTrefEst, cEst]
```

```
theRoot = FindRoot[
  {ConcRatio[t1, nAssumed, kTrefEst, cEst, Tref, T0, a1, a2, a3] == CExperimental1,
   ConcRatio[t2, nAssumed, kTrefEst, cEst, Tref, T0, a1, a2, a3] == CExperimental2},
  {{kTrefEst, 0.03, 0.04}, {cEst, .015, .02}}, MaxIterations → 50]
```

```
{kTrefEst → 0.0177662, cEst → 0.140358}
```

Assign the retrieved values of the parameters kTrefEst and cEst.

```
kTrefEst = theRoot[[1, 2]]; Print[Style[Row[{"kTrefEst = ", kTrefEst}], Red, 14]]
```

```
kTrefEst = 0.0177662
```

```
cEst = theRoot[[2, 2]]; Print[Style[Row[{"cEst = ", cEst}], Red, 14]]
```

```
cEst = 0.140358
```

Assign the final time value, tModelMax.

```
tModelMax = tAxisMax
```

```
120.
```

The following test may lower the user-entered value of t3.

```
If[t3 > tModelMax, t3 = tModelMax - 1.]; t3
```

```
85.
```

Plot the reconstructed degradation curve's ConcRatio(t) using the original values for the parameters nAssumed, kTrefEst and cEst with t varying from 0.001 to tModelMax.

```
predictedCPlot = Plot[ConcRatio[t, nAssumed, kTrefEst, cEst, Tref, T0, a1, a2, a3],
    {t, 0.001, tModelMax}, PlotRange → {{0., tAxisMax}, {-.05, yAxisMax}},
    Frame → True, PlotStyle → {Red, Dashed, Thick},
    FrameLabel → {"time (arbitrary units)", "Concentration ratio(t)", "", ""}];
```

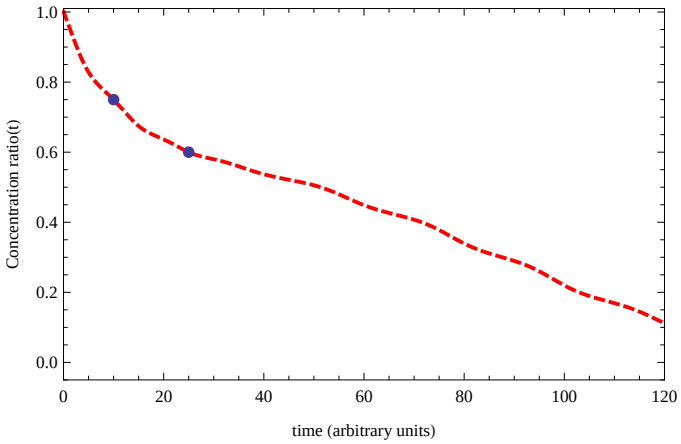
Plot the two experimental point's values (CExperimental1 and CExperimental2, the actual measure-

ments) at times t1 and t2.

```
twoExperimentalPointsPlot = ListPlot[{{t1, CExperimental1}, {t2, CExperimental2}},
    PlotRange → {{0., tAxisMax}, {-.05, yAxisMax}},
    PlotStyle → AbsolutePointSize[6], Frame → True,
    FrameLabel → {"time (arbitrary units)", "Concentration Ratio(t)", "", ""}];
```

Show the reconstructed degradation curve (dashed in red) and the experimental values CExperimental1 and CExperimental2 at times t1 and t2 together on the same plot.

```
Show[predictedCPlot, twoExperimentalPointsPlot]
```



Compute the predicted concentration ratio at t3 using nAssumed and the retrieved parameters, kTrefEst and cEst.

```
predictedValue = ConcRatio[t3, nAssumed, kTrefEst, cEst, Tref, T0, a1, a2, a3];
Print[Style[Row[{"predictedValue = ", predictedValue}], Red, 14]]
```

```
predictedValue = 0.31098
```

Plot the predicted concentration ratio at time t3.

```
predictedPtPlot =
  ListPlot[{{t3, predictedValue}}, PlotRange → {{0., tAxisMax}, {-.05, yAxisMax}},
   PlotStyle → {AbsolutePointSize[6], Red}, Frame → True,
   FrameLabel → {"time (arbitrary units)", "Concentration(t)", "", ""}];
```

Show the reconstructed degradation curve (dashed in red) together with the two experimental points and the one predicted point.

```
Show[predictedCPlot, twoExperimentalPointsPlot, predictedPtPlot]
```