Predicting Nutrient Degradation from Two Successive Concentrations A.nb

Program A: Using simulated data

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))

This program, A, has two parts: The first simulates a degradation curve for chosen parameters and temperature profiles and calculates the compound's concentration or concentration ratio after chosen times t1 and t2 without added error. The second part recreates the degradation curve from the calculated parameter values obtained after adding error(s) to the initial (generated) concentrations. It then calculates (predicts) the nutrient's concentration or concentration ratio after a chosen time t3 and compares its magnitude with that calculated with the original parameters for the same temperature profile.

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.

PART 1

Clear all variables in all contexts.

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

Assign C0 and Tref. C0 is the initial nutrient’s concentration in the user’s chosen units. *Notice that when C0=1, the program can refer to concentration ratios rather than to absolute concentrations*. Tref is the user-chosen reference temperature for the simulations and calculations.

```
C0 = 1.; Tref = 20.;
```

Assign the three generation kinetic parameters their initial values. We will try to recover the values of kTref and c while assuming that of n.

```
n = .95; kTref = .02; c = .1;
```

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

```
tAxisMax = 120.; yAxisMax = C0 + .01;
```

Assign the 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 or 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 = 60.;
```

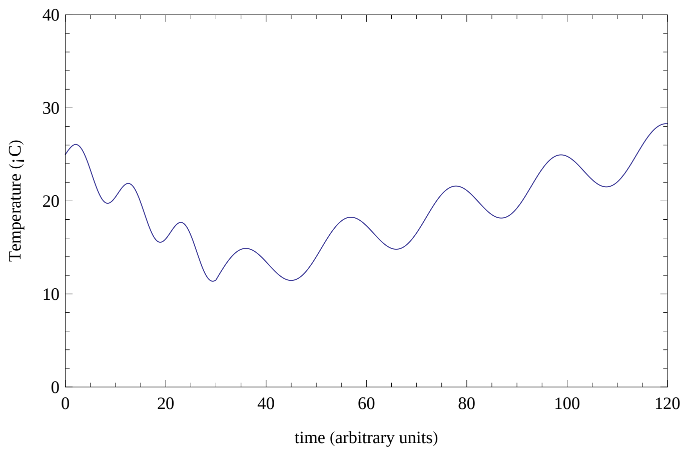
Define your temperature profile function T(t), which can include “If” statements as in the example below. (The example function defined and plotted below is of a fluctuating temperature having a downward followed by an upward trend.)

```
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 function 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 Conc(tFinal), the degradation curve’s equation for the chosen kinetic parameters and temperature profile (concentration or concentration ratio vs. time) from the interpolating function solution returned by NDSolve. *Notice that where n=0, the concentration or concentration ratio can become negative. Or where 0<n<1, the concentration or concentration ratio can become a complex number. Where this occurs the concentration or concentration ratio is assigned a value of zero. However, in this region the method is not viable (see below).* Note that this Conc function has 10 arguments, one more than the ConcRatio function used in the Versions B and C programs. ConcRatio does not have argument C0 which is the 5th argument to Conc.

```
Conc[tFinal_?NumericQ, n_?NumericQ, kTref_?NumericQ,
  c_?NumericQ, C0_?NumericQ, Tref_?NumericQ, T0_?NumericQ,
  a1_?NumericQ, a2_?NumericQ, a3_?NumericQ] := Module[{k1, sfn1, y},
  k1[t_] := kTref * Exp[c * (T[t, T0, a1, a2, a3] - 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] == C0}, y[t], {t, 0., tFinal}][[1, 1, 2]];
  If[sfn1[tFinal] ≤ 0. || Im[sfn1[tFinal]] > 0., 0., sfn1[tFinal]]]
```

Compute the two concentration or concentration ratio values at times t1 and t2 using the entered values of the parameters n, kTref and c.

```
C1 = Conc[t1, n, kTref, c, C0, Tref, T0, a1, a2, a3]
0.755325

C2 = Conc[t2, n, kTref, c, C0, Tref, T0, a1, a2, a3]
0.578259
```

Compute the two perturbed concentration values at times t1 and t2 using the entered values of the parameters n, kTref and c with some user-entered added noise, eps1 and eps2.

```
eps1 = .010;

perturbedConct1 = C1 + eps1
0.765325

eps2 = .015;

perturbedConct2 = C2 + eps2
0.593259
```

In order to avoid a very small residual concentration where the curve’s slope can become too small for the numerical calculation, we limit the method to concentration ratios above 0.1 when n<1. kneeRoot is the calculated time in the pertinent units where this ratio is reached.

```
kneeRoot = FindRoot[If[n >= 1., t == 1000.,
   Conc[t, n, kTref, c, C0, Tref, T0, a1, a2, a3] == 0.1], {t, t2 + 1., t2 + 2.}]
{t → 115.841}
```

Assign the final time value, tModelMax.

```
tModelMax = kneeRoot[[1, 2]]
115.841
```

These tests set the time scale for the calculations so that the chosen duration of the simulation will not exceed that permitted. Assign the final time value, tModelMax.

```
If[tModelMax < tAxisMax, tAxisMax = tModelMax]
115.841

If[tModelMax > tAxisMax, tModelMax = tAxisMax]
```

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

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

Plot the concentration curve Conc(t) using the entered values for the parameters n, kTref and c with t varying from 0.001 to tModelMax.

```
CPlot = Plot[Conc[t, n, kTref, c, C0, Tref, T0, a1, a2, a3], {t, 0.001, tModelMax},
   PlotRange → {{0., tAxisMax}, {-.05, yAxisMax}}, PlotStyle → Thick, Frame → True,
   FrameLabel → {"time (arbitrary units)", "Concentration ratio(t)", "", ""}];
```

PART 2

Enter the assumed degradation kinetics order, nAssumed. *Notice that as in this example, it need not be exactly the same as the order used to generate the data.*

```
nAssumed = 1.;
```

Use FindRoot to solve for the parameters kTrefEst and cEst from the actual concentrations at t1 and t2 assuming the reaction order nAssumed. In this program they are the generated values to which errors (i.e., eps1 and/or eps2) have been introduced or not (i.e., eps1=eps2=0). *In case the FindRoot fails with the automatically assigned initial guesses for kTrefEst and cEst (*kTref-.01 & kTref+.01 *and* c-.01 & c+.01*), use the attached Mathematica notebook* “PredictingNutrientDegradationFromTwoSuccessiveConcentrationsD.nb” *to find better guesses.*

```
Clear[kTrefEst, cEst]

theRoot = FindRoot[
  {Conc[t1, nAssumed, kTrefEst, cEst, C0, Tref, T0, a1, a2, a3] == perturbedConct1,
   Conc[t2, nAssumed, kTrefEst, cEst, C0, Tref, T0, a1, a2, a3] == perturbedConct2},
  {{kTrefEst, kTref - .01, kTref + .01}, {cEst, c - .01, c + .01}}, MaxIterations → 50]

{kTrefEst → 0.0194019, cEst → 0.0970485}
```

Assign the retrieved values of the parameters kTrefEst and cEst.

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

kTrefEst = 0.0194019

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

cEst = 0.0970485
```

Compute the concentrations at times t1 and t2 using the retrieved (newly estimated) values for the parameters, kTrefEst and cEst.

```
Conc[t1, nAssumed, kTrefEst, cEst, C0, Tref, T0, a1, a2, a3]

0.765325

Conc[t2, nAssumed, kTrefEst, cEst, C0, Tref, T0, a1, a2, a3]

0.593259
```

Plot the new (predicted/estimated) degradation curve Conc(t) using the chosen nAssumed and retrieved parameter values (kTrefEst and cEst) returned by FindRoot for t varying from 0.001 to tModelMax.

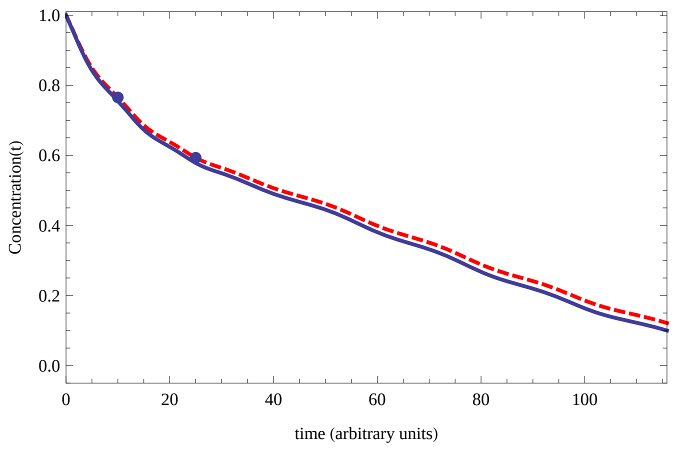
```
predictedPlot = Plot[Conc[t, nAssumed, kTrefEst, cEst, C0, Tref, T0, a1, a2, a3],
   {t, 0.001, tModelMax}, PlotRange → {{0., tAxisMax}, {-.05, yAxisMax}},
   PlotStyle → {Red, Dashed, Thick},
   PlotStyle → {AbsoluteThickness[1], AbsoluteThickness[1]}, Frame → True,
   FrameLabel → {"time (arbitrary units)", "Concentration(t)", "", ""}];
```

Plot the two points' values, perturbedConct1 and perturbedConct2, (which may or may not include added noise) at times t1 and t2.

```
kTrefcPtPlot = ListPlot[{{t1, perturbedConct1}, {t2, perturbedConct2}},
   PlotRange → {{0., tAxisMax}, {-.05, yAxisMax}},
   PlotStyle → AbsolutePointSize[6], Frame → True,
   FrameLabel → {"time (arbitrary units)", "Concentration(t)", "", ""}];
```

Show the original simulated curve (in blue) and the one estimated (predicted) from perturbedConct1 and perturbedConct2 at times t1 and t2 (dashed in red) together with the 2 points on the same graph.

```
Show[predictedPlot, CPlot, kTrefcPtPlot]
```



Compute the generated concentration at time t3 with the original generation parameters n, kTref and c.

```
generatedValue = Conc[t3, n, kTref, c, C0, Tref, T0, a1, a2, a3];
Print[Style[Row[{"generatedValue = ", generatedValue}], Magenta, 14]]
```

```
generatedValue = 0.380482
```

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

```
predictedValue = Conc[t3, nAssumed, kTrefEst, cEst, C0, Tref, T0, a1, a2, a3];
Print[Style[Row[{"predictedValue = ", predictedValue}], Red, 14]]
```

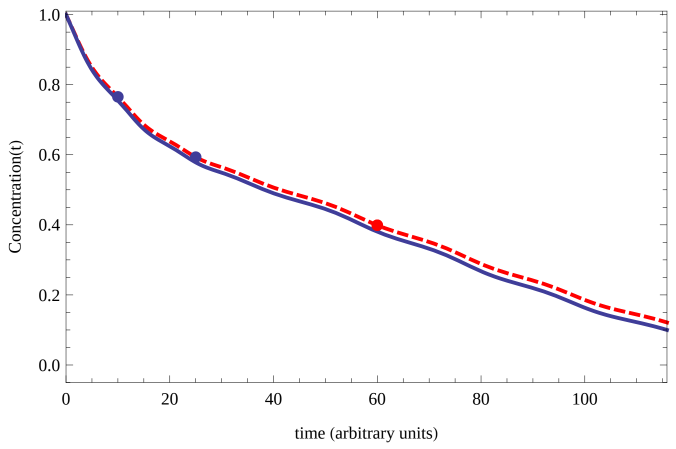
```
predictedValue = 0.398819
```

Plot the predicted concentration 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 original simulated curve (in blue) and the predicted degradation curve (dashed in red) together with the three points

```
Show[predictedPlot, CPlot, kTrefcPtPlot, predictedPtPlot]
```


Concentration(t)
time (arbitrary units)