MATHEMATICA SOLUTIONS TO THE CHEMICAL ENGINEERING PROBLEM SET[1]

H. Eric Nuttall
Department of Chemical/Nuclear Engineering
Farris Engineering Center, Rm 209
University of New Mexico
Albuquerque, New Mexico 87131-1341

INTRODUCTION

These solutions are for a set of numerical problems in chemical engineering. Professor Michael B. Cutlip of the University of Connecticut developed the problems and Professor Mordechai Shacham of Ben-Gurion University of the Negev for the ASEE Chemical Engineering Summer School held in Snowbird, Utah in August, 1997. The problem statements are provided in another document.[1] Professors Cutlip and Shacham provided a document that showed how to solve the problems using POLYMATH. Professor H. Eric Nuttall of the University of New Mexico provided solutions using Mathematica and Professor J. J. Hwalek provided solutions using Mathcad. After the conference, Professor Ross Taylor provided solutions in Maple, and Edward Rosen provided solution in EXCEL. This paper gives the solutions in MATHEMATICA version 3.0. All documents and solutions are available from http://www.che.utexas.edu/cache/ and via FTP from ftp.engr.uconn.edu/pub/ASEE. The written materials are only readable in Adobe Acrobat 3.0 format and higher; however, this software is free via the Internet from www.adobe.com.

The MATHEMATICA solutions were derived using version 3.0. This version of MATHEMATICA is the same on all platforms; hence, the notebooks should work the same for all users independent of computer model. MATHEMATICA is a very extensive and comprehensive computational tool; hence, there are several possible approaches and various routines in MATHEMATICA available for solving each of the ten problems. The approach chosen here is that of the author which means other solutions may prove to be better. Also please note that in addition to using the routines provided by MATHEMATICA one can easily write there own programs/functions. This programming capability greatly extends the usefulness of MATHEMATICA. Numerous primers and publications are available to instruct the user on the operation of MATHEMATICA. Also the online help system is very comprehensive and includes numerous examples for each command. I have found MATHEMATICA to be an indispensable computational tool for teaching and solving engineering problems. All of our Ch.E. students at the University of New Mexico know and use MATHEMATICA. This prefer this package overall the others which are also equally available to the students.

[1] "The Use of Mathematica Software packages in Chemical Engineering". Michael B. Cutlip, John J. Hwalek, H. Eric Nuttall, Mordechai Shacham, Workshop from Session 12, Chemical Engineering Summer School, Snowbird, Utah, Aug., 1997.

1. Molar Volume and Compressibility Factor from Van Der Waals Equation

Mathematica

Mathematica has its own notebook interface which works somewhat as a wordprocessor but allows mathematical operations directly within the notebook. In addition, it uses the concept of built in or user written functions to perform the mathematical tasks. Note that in these notebooks the gray sections are input and the next line will be output. *Mathematica* has a good on-line help utility but to learn how to effectively use this application it is best to read one of the many help books such as: Nancy Blachman, *Mathematica*: A Practical Approach, Prentice Hall. A new edition for *Mathematica* version 3.0 will be available soon.

■ *Mathematica*--Define equation, paramters and functions

```
(* define all the constants used in the equation and
   the compressibility factor as a function. Note it
   is possible to include units with the constants
   but this was not done in these problems. *)

 R = 0.08206; Tc = 405.5; Pc = 111.3; Pr[P_] := P / Pc;
a = 27 (R^2 Tc^2 / Pc) / 64; b = R Tc / (8 Pc);
Z[P_, V_, T_] := P V / (R T); T =.; P =.;
```

```
(* van der Waals equation *)
eqn = (P + a / V^2) (V - b) == R T
```

$$\left(P + \frac{4.19695}{V^2}\right) (-0.0373712 + V) == 0.08206\, T$$

- **Part (a)--Calculate the molar volume and compressibility factor for gaseous ammonia at a pressure of P=56 atm and a temperature of T= 450 K using the van der Waals equation of state.**

```
(* temperature and pressure *)

T = 450; P = 56;
```

```
(* Solve vad der Waals equation *)

FindRoot[eqn, {V, 0.57}]
```

```
{V → 0.574892}
```

```
(* evaluate compressibility factor *)

Z[P, 0.574892, T]
```

```
0.871827
```

- **Part (b)--Repeat the problem for a series of reduced pressures: Pr= 1,2,4,10, and 20**

```
(* Define partial pressures in the array Pr *)

Pr[0] = 0.503; Pr[1] = 1; Pr[2] = 2; Pr[3] = 4;
Pr[4] = 10; Pr[5] = 20;
```

```
(* Create table of P, Pr V and Z *)

TableForm[Table[{P = Pr[i] Pc,
   Pr[i], V1 = V /. FindRoot[eqn, {V, 0.56}],
   Z[P, V1, T]}, {i, 0, 5}],
 TableHeadings -> {None, {"P(atm)", "Pr", "V", "Z"}},
 TableAlignments -> Center]
```

P(atm)	Pr	V	Z
55.9839	0.503	0.575084	0.871868
111.3	1	0.233509	0.703808
222.6	2	0.0772676	0.465777
445.2	4	0.0606543	0.731261
1113.	10	0.0508753	1.53341
2226.	20	0.046175	2.78348

- **Part (c)--Generate a table and plot Pr versus Z**

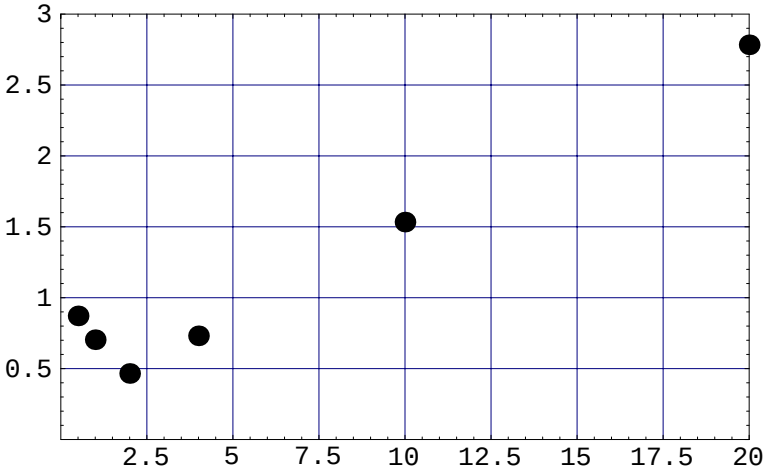
```
plist = Table[{Pr[i],
   Z[P = Pr[i] Pc, V /. FindRoot[eqn, {V, 0.56}], T]},
  {i, 0, 5}]
```

```
{{0.503, 0.871868}, {1, 0.703808}, {2, 0.465777},
 {4, 0.731261}, {10, 1.53341}, {20, 2.78348}}
```

```
TableForm[plist, TableHeadings -> {None, {"Pr", "Z"}},
 TableAlignments -> Center]
```

Pr	Z
0.503	0.871868
1	0.703808
2	0.465777
4	0.731261
10	1.53341
20	2.78348

```
ListPlot[plist, PlotRange -> {{0, 20}, {0, 3}},
 PlotStyle -> PointSize[0.03],
 Frame -> True, GridLines -> Automatic]
```



- Graphics -

2. Steady Steate Material Balances on a Separation Train

Mathematica

- **This set of simultaneous linear algebraic equations can be solved directly using the "Solve" function in *Mathematica*. There are other linear equation solving routines in *Mathematica* that could be used for larger sets of simultaneous equations and of course, matrix algebra approach could be used as well. But the method shown is the most common routine and approach in *Mathematica* for solving equations. Since *Mathematica* is a symbolic program one can assign the equations equal to variables to simplicity and clarification.**

- **Part (a) Solve the set of linear simultaneous equations for the unknowns: D1, D2, B1, B2.**

```
(* Xylene*)
eqnX = 0.07 D1 + 0.18 B1 + 0.15 D2 + 0.24 B2 == 0.15 * 70;
(* Styrene*)
eqnS = 0.04 D1 + 0.24 B1 + 0.1 D2 + 0.65 B2 == 0.25 * 70;
(* Touene *)
eqnT = 0.54 D1 + 0.42 B1 + 0.54 D2 + 0.1 B2 == 0.4 * 70;
(* Benzene*)
eqnB = 0.35 D1 + 0.16 B1 + 0.21 D2 + 0.01 B2 == 0.2 * 70;
```

```
answers =
 Solve[{eqnX, eqnS, eqnT, eqnB}, {D1, B1, D2, B2}]
(* Answers appear in the rule construct. This
  rule construct makes symbolic programs very
  powerful. The rule construct will appear in
  many of the problem solutions. *)
```

```
{{D1 → 26.25, B1 → 17.5, D2 → 8.75, B2 → 17.5}}
```

- **This step shown below reduces the answer list from a list of lists to a simple list for use in Part (b). Notice that there are one less set of braces around the list!**

```
answers = Part[answers, 1]
```

```
{D1 → 26.25, B1 → 17.5, D2 → 8.75, B2 → 17.5}
```

■ Part (b)-Solve for compositions and molar flow rates in streams B & D

- **Part (b)--Equations (again for convenience and clarity the equations are assigned to variables)**

```
eqn1 = D == D1 + B1;
eqn2 = XDx D == 0.07 D1 + 0.18 B1;
eqn3 = XDs D == 0.04 D1 + 0.24 B1;
eqn4 = XDt D == 0.54 D1 + 0.42 B1;
eqn5 = XDb D == 0.35 D1 + 0.16 B1;
eqn6 = B == D2 + B2;
eqn7 = XBx B == 0.15 D2 + 0.24 B2;
eqn8 = XBs B == 0.1 D2 + 0.65 B2;
eqn9 = XBt B == 0.54 D2 + 0.1 B2;
eqn10 = XBb B == 0.21 D2 + 0.01 B2;
```

```
(* Again the "Solve " function is used.
    It is necessary to substitute into the
    equations the solutions from Part (a). *)

answers2 = Solve[
  ReplaceAll[{eqn1, eqn2, eqn3, eqn4, eqn5, eqn6, eqn7,
  eqn8, eqn9, eqn10}, answers],
  {D, B, XDx, XDs, XDt, XDb, XBx, XBs, XBt, XBb}]
```

```
{{XDx → 0.114, XDs → 0.12, XDt → 0.492, XDb → 0.274, XBx → 0.21,
  XBs → 0.466667, XBt → 0.246667, XBb → 0.0766667,
  B → 26.25, D → 43.75}}
```

```
(* The answers from Part (b) are
  listed in the rule construct as a table *)

Part[answers2, 1] // TableForm
```

```
XDx → 0.114
XDs → 0.12
XDt → 0.492
XDb → 0.274
XBx → 0.21
XBs → 0.466667
XBt → 0.246667
XBb → 0.0766667
B → 26.25
D → 43.75
```

3. Vapor Pressure Data Representaton by Polynomials and Equations

Mathematica

In this problem the same temperature versus pressure data is fit using three different models, i.e.,
1) a polynomial with increasing order--linear regression
2) the Clausiu-Clapeyron equation--linear regression of transformed data
3) the Antoine equation--nonlinear regression or curve fitting

In each case, the final model and data are graphed on the same plot for comparison.

■ Part (a)--polynomial fit

▪ Define the regression data--Temperature (C) versus P (mm Hg)

```
data = {{-36.7, 1}, {-19.6, 5}, {-11.5, 10}, {-2.6, 20},
  {7.6, 40}, {15.4, 60}, {26.1, 100}, {42.2, 200},
  {60.6, 400}, {80.1, 760}}
```

```
{{-36.7, 1}, {-19.6, 5}, {-11.5, 10}, {-2.6, 20},
 {7.6, 40}, {15.4, 60}, {26.1, 100}, {42.2, 200},
 {60.6, 400}, {80.1, 760}}
```

```
data // TableForm (* T versus P *)
```

```
-36.7      1
-19.6      5
-11.5      10
-2.6       20
7.6        40
15.4       60
26.1       100
42.2       200
60.6       400
80.1       760
```

```
(* Plot T versus P *)

p1 = ListPlot[data, PlotStyle -> PointSize[0.035],
 PlotRange -> {{-50, 100}, {0, 800}},
  AxesOrigin -> {-50, 0}]
```

```
- Graphics -
```

- **Part (a) Regress the data with a increasing orders of a polynamial in P. Analyze the R squared for best fit.**

```
(* add a standard
  Mathematica package to the kernal *)

Needs["Statistics`LinearRegression`"]
```

```
(* first order polynomial in T,
 RSquared=0.78 *)

T =.;
 (regress = Regress[data, {1, T}, T];
Chop[regress, 10^(-6)])
```

```
{ParameterTable →
        Estimate      SE           TStat        PValue
 1      64.406        42.4524      1.51713      0.167708
 T      5.89072       1.10667      5.32294      0.000708522

 ,
RSquared → 0.779819,
AdjustedRSquared → 0.752296,
EstimatedVariance → 14823.8, ANOVATable →
            DF      SumOfSq      MeanSq       FRatio       PValue
 Model      1       420014.      420014.      28.3337      0.00070
 Error      8       118591.      14823.8
 Total      9       538604.

 }
```

```
(* Second order polynomial in T,
 RSquared=0.98 *)

regress = Regress[data, {1, T, T^2}, T];
Chop[regress, 10^(-6)]
```

```
{ParameterTable →
```

	Estimate	SE	TStat	PValue
1	−0.582065	13.9506	−0.0417234	0.967884
T	2.06715	0.511995	4.03744	0.00494892
T^2	0.0861526	0.00905805	9.51116	0.00002974

```
,
RSquared → 0.984186,
AdjustedRSquared → 0.979668,
EstimatedVariance → 1216.79, ANOVATable →
```

	DF	SumOfSq	MeanSq	FRatio	PValue
Model	2	530087.	265043.	217.823	0
Error	7	8517.5	1216.79		
Total	9	538604.			

```
}
```

```
(* Third order polynomial in T,
 RSquared=0.9996-- This is a very good fit *)

regress = Regress[data, {1, T, T^2, T^3}, T];
Chop[regress, 10^(-6)]
```

{ParameterTable →

	Estimate	SE	TStat	PValue
1	24.4594	2.83402	8.63062	0.0001332
T	1.1981	0.102232	11.7195	0.0000232
T^2	0.0394481	0.0033548	11.7587	0.0000228
T^3	0.000744911	0.0000477255	15.6082	4.3796×1

,
RSquared → 0.99962,
AdjustedRSquared → 0.99943,
EstimatedVariance → 34.1222, ANOVATable →

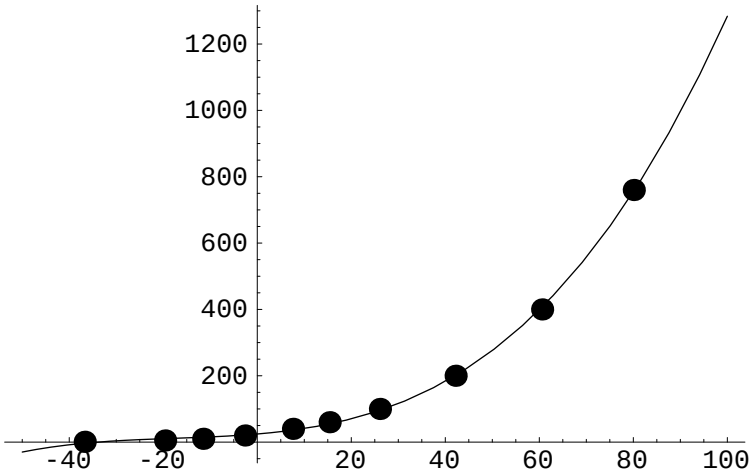
	DF	SumOfSq	MeanSq	FRatio	PValue
Model	3	538400.	179467.	5259.52	0
Error	6	204.733	34.1222		
Total	9	538604.			

}

```
(* In this statement the final polynomial is
  generated in order to graph the function *)

eqn1 = Fit[data, {1, T, T^2, T^3}, T]
```

$24.4594 + 1.1981\ T + 0.0394481\ T^2 + 0.000744911\ T^3$

```
(* This is a fancy way to overlay
  the data onto the polynomial function *)

p2 = Plot[eqn1, {T, -50, 100},
  Epilog → {PointSize[0.03], Point /@ data}]
```



- Graphics -

■ Part (b) Regression with Clausius-Clapeyron Equation

```
data
```

```
{{-36.7, 1}, {-19.6, 5}, {-11.5, 10}, {-2.6, 20},
 {7.6, 40}, {15.4, 60}, {26.1, 100}, {42.2, 200},
 {60.6, 400}, {80.1, 760}}
```

- **In this next operation the data is transformed to temperature versus $\log_{10}$ of pressure**

```
(* Take the log10 of each pressure *)

Transdata =
 Table[{First[data[[i]]],
 Log[10, Last[data[[i]]]]}, {i, 10}] // N
```

```
{{-36.7, 0}, {-19.6, 0.69897}, {-11.5, 1.},
 {-2.6, 1.30103}, {7.6, 1.60206}, {15.4, 1.77815},
 {26.1, 2.}, {42.2, 2.30103}, {60.6, 2.60206},
 {80.1, 2.88081}}
```

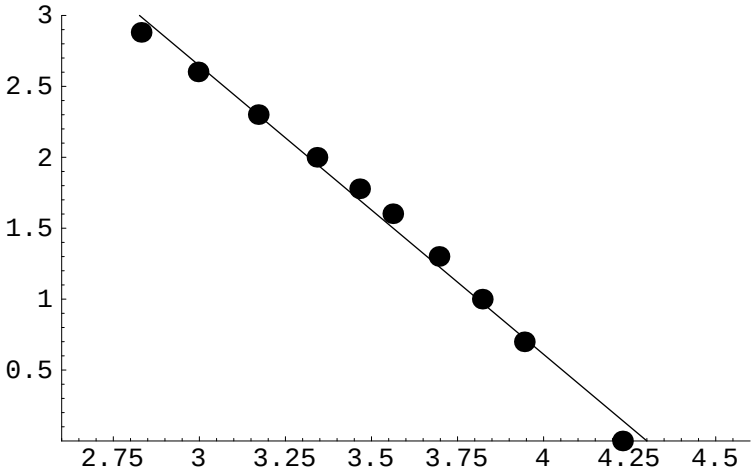
```
(* Regress the transformed
  data and generate an equation model *)

CCmodel = Fit[Transdata, {1, -1 / (T + 273.15)}, T]
```

$$8.75201 - \frac{2035.33}{273.15 + T}$$

```
(* Transform data for
  graphing to 1/(273.15+T) vs. Log10 (P) *)

plotdata = Table[{1000 * 1 / (273.15 + First[data[[i]]]),
    Log[10, Last[data[[i]]]]}, {i, 1, 10}] //
  N
```

```
{{4.22922, 0}, {3.944, 0.69897},
 {3.8219, 1.}, {3.69617, 1.30103}, {3.56189, 1.60206},
 {3.4656, 1.77815}, {3.34169, 2.}, {3.17108, 2.30103},
 {2.99625, 2.60206}, {2.83086, 2.88081}}
```

```
(* Plot Clausius-
  Clapayron model and transformed data *)

Plot[8.7520 - 2035.331 * invTK / 1000,
 {invTK, 2.6, 4.3},  PlotRange -> {{2.6, 4.6}, {0, 3}},
 AxesOrigin -> {2.6, 0},
   Epilog → {PointSize[0.03], Point /@ plotdata}]
```



- Graphics -

■ Part (c) Regression with the Antoine Equation

```
 (* load into Mma kernal the standard package *)

<< Statistics`NonlinearFit`
```

```
T = .;
NonlinearFit[
 Transdata, A - B / (T + C), T, {{A, 5}, {B, 600}, {C, 150}}]
```

$$5.64228 - \frac{637.031}{149.198 + T}$$

```
NonlinearRegress[Transdata, A - B / (T + C), T,
 {{A, 5}, {B, 600}, {C, 150}}]
```

```
{BestFitParameters →
  {A → 5.64228, B → 637.031, C → 149.198}, ParameterCITable →
        Estimate     Asymptotic SE     CI
 A      5.64228      0.133415          {5.3268, 5.95776}
 B      637.031      41.7901           {538.213, 735.849}'
 C      149.198      4.98397           {137.413, 160.983}
 EstimatedVariance → 0.00036253, ANOVATable →
                        DF     SumOfSq       MeanSq
 Model                  3      33.2717       11.0906
 Error                  7      0.0025377     0.00036253,
 Uncorrected Total      10     33.2742
 Corrected Total        9      7.14634

 AsymptoticCorrelationMatrix → {{1., 0.993145, 0.975547},
                                {0.993145, 1., 0.994079},
                                {0.975547, 0.994079, 1.}},

                                                Curvature
 FitCurvatureTable → {Max Intrinsic             0.020214
                      Max Parameter-Effects     2.44175
                      95. % Confidence Region   0.479638}
```

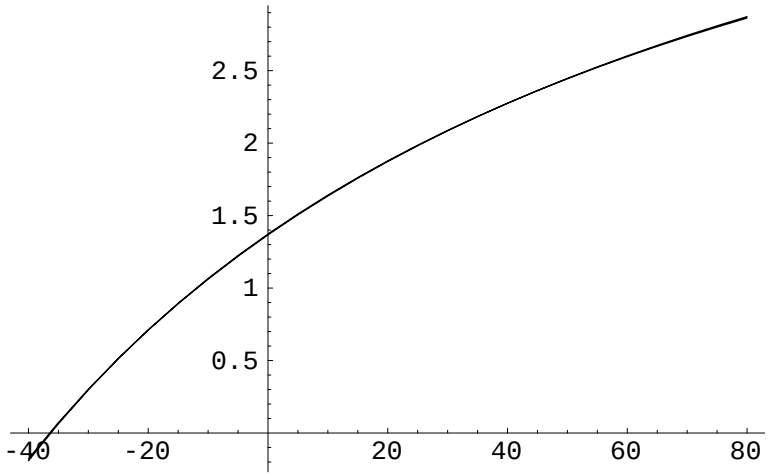
■ The above values for A, B, and C differ from those calculated by Polymath. Below I set C equal to the Polymath value (153.885) and calculated A and B. This procedure gave the same results as Polymath.

```
(* Values for A and B
  the same as calculated from Polymath *)

NonlinearRegress[
 Transdata, A - B / (T + 153.885), T, {A, B}]
```

```
{BestFitParameters → {A → 5.76734, B → 677.091}, ParameterCITable →
        Estimate      Asymptotic SE      CI
 A      5.76734       0.0264636          {5.70631, 5.82836},
 B      677.091       4.22989            {667.337, 686.845}
 EstimatedVariance → 0.000278812, ANOVATable →
                          DF      SumOfSq        MeanSq
 Model                    2       33.272         16.636
 Error                    8       0.0022305      0.000278812,
 Uncorrected Total        10      33.2742
 Corrected Total          9       7.14634

 AsymptoticCorrelationMatrix → ( 1.        0.979892 ),
                               ( 0.979892  1.       )

                                                  Curvature
 FitCurvatureTable → Max Intrinsic                0
                     Max Parameter-Effects        0          }
                     95. % Confidence Region      0.473568
```

■ Below is a plot showing that the two solutions are essentially identical and that they fit the data very well.

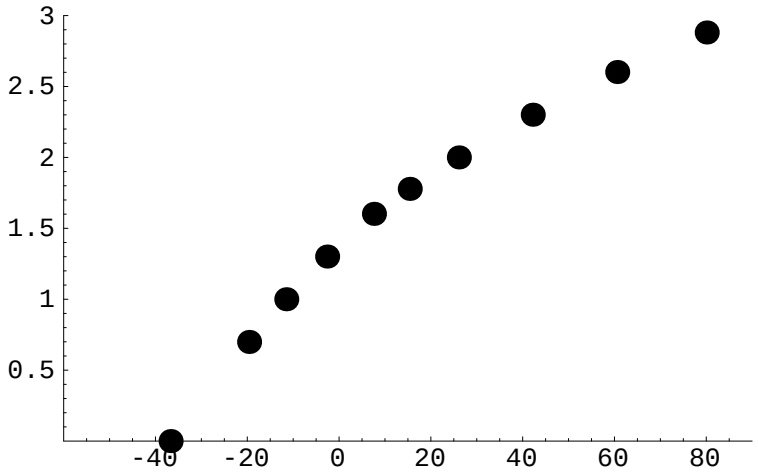
```
(* Note that the Antonine equation with different
  constants gives essentially identical results *)

pc2 = Plot[
  {5.76734 - 677.0908/(153.8849 + Tc), 5.64227 - 637.0310/(149.19784 + Tc)},
  {Tc, -40, 80}]
```



- Graphics -

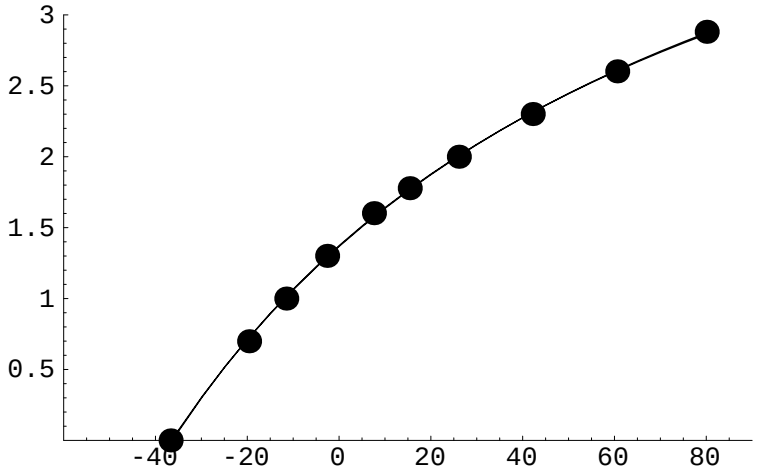
■ **Next the data is overlayed onto the Antoine equation**

```
pc1 = ListPlot[Transdata, PlotStyle -> PointSize[0.035],
 PlotRange -> {{-60, 90}, {0, 3}},
  AxesOrigin -> {-60, 0}]
```



- Graphics -

```
Show[pc1, pc2]
(* plot combined data and model results *)
```



- Graphics -

4. Reaction Equilibrium for Multiple Gas Phase Reactions

Mathematica

This problem requires the solution of simultaneous nonlinear algebraic equations. The *Mathematica* function FindRoot is used to solve these equations. Use the online help to learn more about the built-in FindRoot utility function.

- **For the equations see the problem statement EQUATIONS (12)**

- **Setup equations for solving using Mathematica FindRoot function. Note that the numerical values for certain constants are given and assigned below along with the necessary equations.**

```
f1 = Cc Cd - Kc1 Ca Cb == 0;
f2 = Cx Cy - Kc2 Cb Cc == 0;
f3 = Cz - Kc3 Ca Cx == 0;
eqn4 = Ca == Ca0 - Cd - Cz;
eqn5 = Cb == Cb0 - Cd - Cy;
eqn6 = Cc == Cd - Cy;
eqn7 = Cy == Cx + Cz;
Ca0 = Cb0 = 1.5;
Kc1 = 1.06;
Kc2 = 2.63;
Kc3 = 5;
```

■ Parts-(a), (b) and (c)

■ Solve using the three different starting values for C_D, C_X and C_Z

- **Trial #1**

```
answera =
 FindRoot[{f1, f2, f3, eqn4, eqn5, eqn6, eqn7}, {Ca, 1},
  {Cb, 1}, {Cc, 1}, {Cd, 0}, {Cx, 0}, {Cy, 1}, {Cz, 0}]
```

```
{Ca → 0.420689, Cb → 0.242897, Cc → 0.153565,
 Cd → 0.705334, Cx → 0.177792, Cy → 0.551769, Cz → 0.373977}
```

- **Trial #2**

```
answerb =
 FindRoot[{f1, f2, f3, eqn4, eqn5, eqn6, eqn7}, {Ca, 1},
  {Cb, 1}, {Cc, 1}, {Cd, 1}, {Cx, 1}, {Cy, 1}, {Cz, 1}]
```

```
{Ca → 0.420689, Cb → 0.242897, Cc → 0.153565,
 Cd → 0.705334, Cx → 0.177792, Cy → 0.551769, Cz → 0.373977}
```

- **Trial #3**

```
answerc =
 FindRoot[{f1, f2, f3, eqn4, eqn5, eqn6, eqn7}, {Ca, 1},
  {Cb, 1}, {Cc, 1}, {Cd, 10}, {Cx, 10}, {Cy, 1}, {Cz, 10}]
```

```
{Ca → 0.420689, Cb → 0.242897, Cc → 0.153565,
 Cd → 0.705334, Cx → 0.177792, Cy → 0.551769, Cz → 0.373977}
```

- **The table shown below lists the solutions from each trial. Note that they are all exactly the same.**

```
Table[{answera[[i]],
   answerb[[i]], answerc[[i]]}, {i, 1, 7}] //
 TableForm
```

Ca → 0.420689	Ca → 0.420689	Ca → 0.420689
Cb → 0.242897	Cb → 0.242897	Cb → 0.242897
Cc → 0.153565	Cc → 0.153565	Cc → 0.153565
Cd → 0.705334	Cd → 0.705334	Cd → 0.705334
Cx → 0.177792	Cx → 0.177792	Cx → 0.177792
Cy → 0.551769	Cy → 0.551769	Cy → 0.551769
Cz → 0.373977	Cz → 0.373977	Cz → 0.373977

5. Terminal Velocity of Falling Particles

Mathematica

- ***Mathematica*** **has many special capabilities. For this problem, user written conditional functions are used. The problem is nonlinear because the particle velocity appears both in the equation and embedded in the Reynolds number which appears in the drag coeficient equation. Hence a very nonlinear problem!**

  First, define the Drag Coefficient function for various ranges of Re using the conditional function construct.

```
CD[Re_] := 24 / Re /; Re < 0.1;
CD[Re_] := (24 / Re) (1 + 0.14 Re^0.7) /; 0.1 <= Re <= 1000;
CD[Re_] := 0.44 /; 1000 < Re < 350000;

(* Reference for this special
  conditional function construct: N. Blachman,
 Mathematica: A Practical Approach, pg. 170 *)
```

- **Next define the parameters (given data) using the list & rule construct. This construct preserves the symbolic form of the equations.**

```
data = {ρ -> 994.6 , ρp -> 1800, μ -> 8.93 * 10^-4 ,
  Dp -> 0.208 * 10^-3, g -> 9.80665}
```

```
{ρ → 994.6, ρp → 1800, μ → 0.000893, Dp → 0.000208,
 g → 9.80665}
```

```
Unprotect[Re]; (* if we wish to use the symbol
  "Re" we must unprotect it. In Mathematica
  this symbol is already defined *)
```

- **Define equations: Reynolds number and particle velocity**

```
eqn1 = Re == Dp vt ρ / μ
```

$$\text{Re} == \frac{\text{Dp vt } \rho}{\mu}$$

```
eqn2 = vt == Sqrt[4 g (ρp - ρ) Dp / (3 CD[Re] ρ)]
```

$$\text{vt} == \frac{2\sqrt{\frac{\text{Dp g } (-\rho+\rho\text{p})}{\rho\ \text{CD}[\text{Re}]}}}{\sqrt{3}}$$

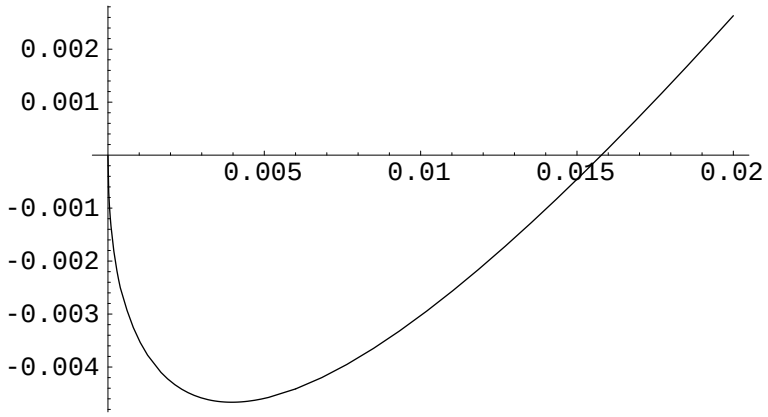
- **Solve the nonlinear equations simultaneously-Part (a)**

```
solution = FindRoot[Evaluate[{eqn1, eqn2} /. data],
 {Re, {1, 1000}}, {vt, {0.01, 0.02}}]
```

$$\{\text{Re} \to 3.65632,\ \text{vt} \to 0.0157828\}$$

- **Plot function velocity equation versus vt to graphically verify the calculated solution.**

```
(* This plot is just a fun and visually instructive
  way to see graphically the correct solution *)

Plot[
 vt - Sqrt[4 g (ρp - ρ) Dp / (3 CD[Dp vt ρ / μ] ρ)] /. data /.
  data, {vt, 0.000, 0.02}]
```



- Graphics -

- **Summary of answers for Part (a)**

```
Join[solution, {CD -> CD[Re] /. solution}] // ColumnForm
```

Re → 3.65632
vt → 0.0157828
CD → 8.8413

■ Part (b)--Recalculate the problem assuming gravity is 30 times greater

```
data2 = {ρ -> 994.6 , ρp -> 1800, μ -> 8.93 * 10^-4 ,
   Dp -> 0.208 * 10^-3, g -> 30 * 9.80665}
```

```
{ρ → 994.6, ρp → 1800, μ → 0.000893, Dp → 0.000208,
 g → 294.199}
```

```
solutionb = FindRoot[Evaluate[{eqn1, eqn2} /. data2],
  {Re, {1, 1000}}, {vt, {0.1, 0.3}}]
```

```
{Re → 47.7299, vt → 0.20603}
```

■ Summary of answers for Part (b)

```
(* This is a fancy way in Mathematica to
   show the answers, i.e., a list of rules *)

Join[solutionb, {CD -> CD[Re] /. solutionb}] //
 ColumnForm
```

```
Re → 47.7299
vt → 0.20603
CD → 1.55649
```

6. Heat Exchange in a Series of Tanks

Mathematica

This problem requires the solution of simultaneous first order differential equations.

- **Define in *Mathematica* the differentional equations (20), (21), and (22) from problem statement. The three equations are assigned for convenience to the variables eqn1, eqn2, and eqn3.**

```
eqn1 =
 T1'[t] == (W Cp (To - T1[t]) + UA (Ts - T1[t])) / (M Cp)
```

$$T1'[t] == \frac{Cp\ W\ (To - T1[t]) + UA\ (Ts - T1[t])}{Cp\ M}$$

```
eqn2 =
 T2'[t] == (W Cp (T1[t] - T2[t]) + UA (Ts - T2[t])) / (M Cp)
```

$$T2'[t] == \frac{UA\ (Ts - T2[t]) + Cp\ W\ (T1[t] - T2[t])}{Cp\ M}$$

```
eqn3 =
 T3'[t] == (W Cp (T2[t] - T3[t]) + UA (Ts - T3[t])) / (M Cp)
```

$$T3'[t] == \frac{UA\ (Ts - T3[t]) + Cp\ W\ (T2[t] - T3[t])}{Cp\ M}$$

- **Steady-state equations**

```
eqn1s = 0 == (W Cp (To - T1[t]) + UA (Ts - T1[t])) / (M Cp)
```

$$0 == \frac{Cp\ W\ (To - T1[t]) + UA\ (Ts - T1[t])}{Cp\ M}$$

```
eqn2s = 0 == (W Cp (T1[t] - T2[t]) + UA (Ts - T2[t])) / (M Cp)
```

$$0 == \frac{UA\,(Ts - T2[t]) + Cp\,W\,(T1[t] - T2[t])}{Cp\,M}$$

```
eqn3s = 0 == (W Cp (T2[t] - T3[t]) + UA (Ts - T3[t])) / (M Cp)
```

$$0 == \frac{UA\,(Ts - T3[t]) + Cp\,W\,(T2[t] - T3[t])}{Cp\,M}$$

■ Define parameters

```
data = {W -> 100, Cp -> 2,
  To -> 20, UA -> 10, Ts -> 250, M -> 1000,
  T1o -> 20, T2o -> 20, T3o -> 20}
```

```
{W → 100, Cp → 2, To → 20, UA → 10, Ts → 250, M → 1000,
 T1o → 20, T2o → 20, T3o → 20}
```

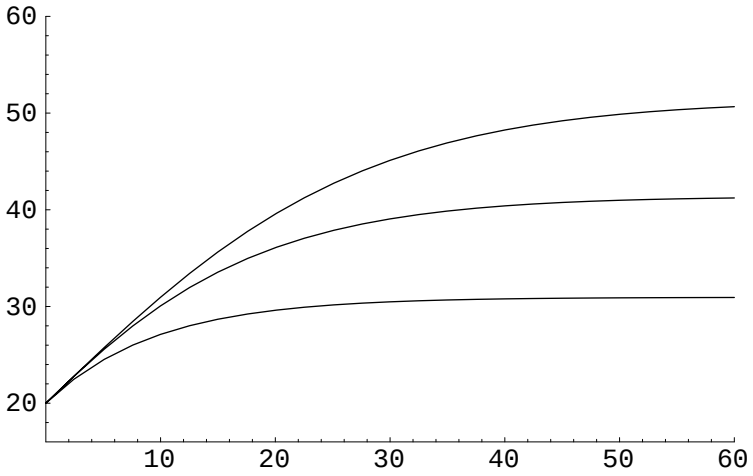
■ Solve the differential equations

```
solutions = NDSolve[Evaluate[{eqn1, T1[0] == T1o, eqn2,
     T2[0] == T2o, eqn3, T3[0] == T3o} /. data],
  {T1, T2, T3}, {t, 0, 100}]
```

```
{{T1 → InterpolatingFunction[{{0., 100.}}, <>],
  T2 → InterpolatingFunction[{{0., 100.}}, <>],
  T3 → InterpolatingFunction[{{0., 100.}}, <>]}}
```

■ Plot results

```
Plot[Evaluate[
  {T1[t], T2[t], T3[t]} /. solutions], {t, 0, 60},
 PlotRange -> {{0, 60}, {16, 60}}, AxesOrigin -> {0, 16}]

Note that the top curve is for tank 3,
tank 2 and the lowest curve is tank1 .
```



- Graphics -

■ Solve for the Steady State Temperatures

```
(* Before solveing the three steady state equations,
 I check to see that they are correct *)

Evaluate[{eqn1s, eqn2s, eqn3s} /. data]
```

$$\left\{0 == \frac{200\,(20 - T1[t]) + 10\,(250 - T1[t])}{2000},\right.$$
$$0 == \frac{10\,(250 - T2[t]) + 200\,(T1[t] - T2[t])}{2000},$$
$$\left.0 == \frac{10\,(250 - T3[t]) + 200\,(T2[t] - T3[t])}{2000}\right\}$$

```
(* This statement
  solves the three steady state equations *)

NSolve[Evaluate[{eqn1s, eqn2s, eqn3s} /. data],
 {T1[t], T2[t], T3[t]}]
```

```
{{T1[t] → 30.9524, T2[t] → 41.3832, T3[t] → 51.3174}}
```

- **Calculate the time required to reach 99% of the final T3 temperature. First calcuate the temperature of tank 3 at the 99% level.**

```
T3f99 = 0.99 * 51.3174
```

```
50.8042
```

```
(* Create a table of time versus
  T3. This table verifies that the solution
  approaches SS about about 60 minutes *)

Table[
  {t, Evaluate[T3[t] /. solutions]}, {t, 0, 100, 10}] //
 TableForm
```

```
0       20.
10      30.9591
20      39.5629
30      45.1144
40      48.2512
50      49.8736
60      50.6623
70      51.0287
80      51.193
90      51.2648
100     51.2955
```

■ Part (b)--Answer 63 minutes

```
(* Solve for the
  time at the 99 % level using FindRoot *)

FindRoot[Evaluate[0 == 50.804 - T3[t] /. solutions],
 {t, {0, 70}}]
```

```
{t → 63.0087}
```

7. Diffusion with Chemical Reaction in a One Dimensional Slab

Mathematica

This problem requires the solution of a second order boundary value problem. *Mathematica* doesn't come with a built-in routine for numerically solving boundary value problems; however, *Mathematica* can analytically solve this BVP. Hence to numerically solve the problem I wrote a small program within *Mathematica*. As you will see writing programs in *Mathematica* requires very few statements since one has access to numerous *Mathematica* routines. I assume that others have already written excellent boundary value solution routines for *Mathematica* and I assume these are available on the WEB.

- **Define equation**

```
eqn = Ca''[z] == (k / Dab) Ca[z]
```

$$\text{Ca}''[z] == \frac{k\,\text{Ca}[z]}{\text{Dab}}$$

- **Define data**

```
data =
 {Cao -> 0.2, k -> 0.001, Dab -> 1.2 * 10^-9, L -> 10^-3}
```

$$\left\{\text{Cao} \to 0.2,\ \text{k} \to 0.001,\ \text{Dab} \to 1.2 \times 10^{-9},\ \text{L} \to \frac{1}{1000}\right\}$$

■ Note one needs to write a small program in Mathematica to solve boundary value problem.

- **Test to see if *Mathematica* (Mma) can correctly solve the second order bound value equation as an initial value ODE.**

```
temp = NDSolve[
  Evaluate[{eqn, Ca[0] == Cao, Ca'[0] == -132} /. data],
  Ca[z], {z, 0, 10^-3}]
```

```
{{Ca[z] → InterpolatingFunction[{{0., 0.001}}, <>][z]}}
```

- **Check to see if Mma can calculate the derivative of the solution at Lmax.**

```
Cz = Ca[z] /. temp
```

```
{InterpolatingFunction[{{0., 0.001}}, <>][z]}
```

```
N[D[Cz, z] /. z -> 10^-3]
```

```
{-0.128436}
```

- **Write a function of initial slope that returns the slope of Ca at Lmax.**

```
Fun[DCa_] := Block[{},
  temp = NDSolve[
    Evaluate[{eqn, Ca[0] == Cao, Ca'[0] == DCa} /. data],
    Ca[z], {z, 0, 10^-3}];
  Cz = Ca[z] /. temp;
  First[N[D[Cz, z] /. z -> 10^-3]]
 ]
```

■ Test this function

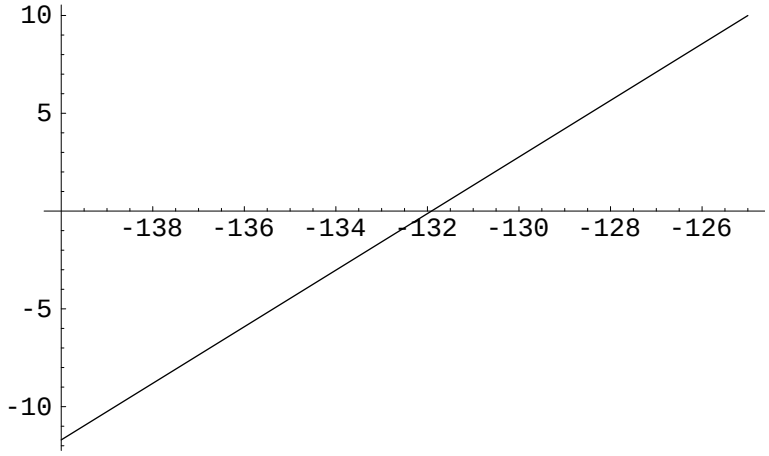
```
Fun[-140]
```

```
-11.6998
```

```
FindRoot[Fun[DCa], {DCa, {-125, -140}}]
```

```
{DCa → -131.911}
```

```
Plot[Fun[DCa], {DCa, -125, -140}]
```




```
- Graphics -
```

■ Second solution technique

■ Modify the problem into a minimization problem

```
FunM[DCa_] := Block[{},
    temp = NDSolve[
    Evaluate[{eqn, Ca[0] == Cao, Ca'[0] == DCa} /. data],
    Ca[z], {z, 0, 10^-3}];
    Cz = Ca[z] /. temp;
    Abs[First[N[D[Cz, z] /. z -> 10^-3]]]
 ]
```

```
FindMinimum[FunM[DCa], {DCa, {125, 140}}]
```

$\{2.22874 \times 10^{-14}, \{DCa \to -131.911\}\}$

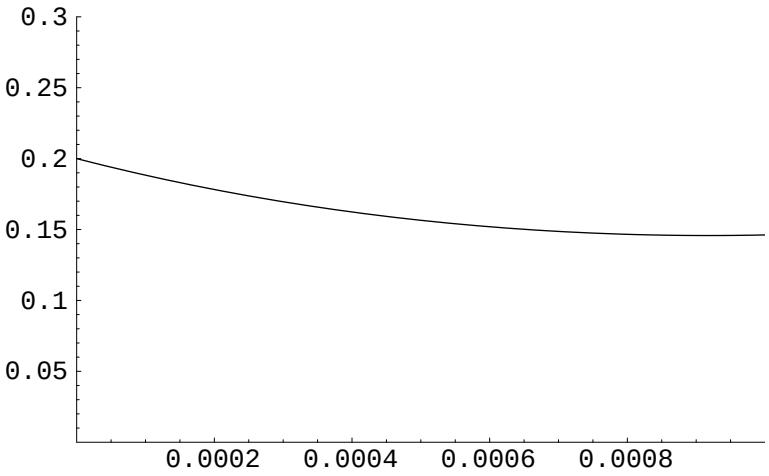
■ Plot the minimization function, FunM

```
Plot[FunM[DCa], {DCa, -125, -140}]
```

- Graphics -

- **Plot the solution over the variable z (This completes Part (a))**

```
(* Plotted here is the
  numerical solution to the BVP problem. *)

Plot[Ca[z] /. temp, {z, 0, 10^-3},
 PlotRange -> {{0, 10^-3}, {0, 0.3}}]
```

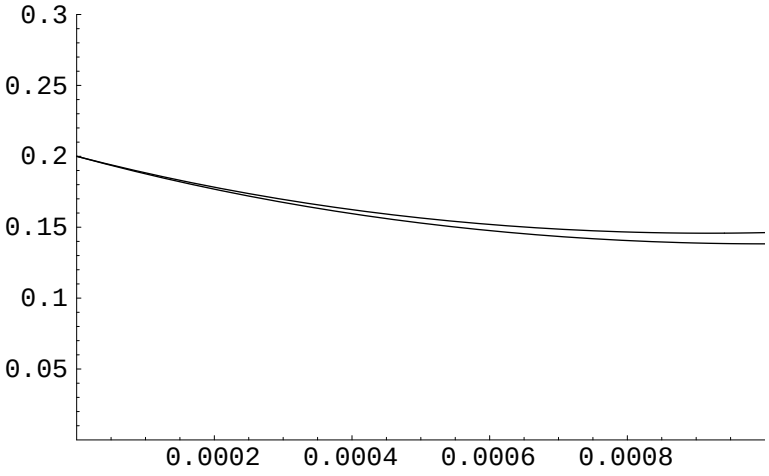


- Graphics -

- **Part (b) Comparison of numerical and analytical solutions**

```
Canalytical[z_] :=
 Evaluate[Cao * Cosh[L (Sqrt[k / Dab] (1 - z / L))] /
     Cosh[L (Sqrt[k / Dab])] /.
   data]
```

```
Plot[{Ca[z] /. temp, Canalytical[z]}, {z, 0, 10^-3},
 PlotRange -> {{0, 10^-3}, {0, 0.3}}]
```



- Graphics -

```
(* Note that the numerical and analytical
  solutions at nearly identical *)
```

- **Table comparsion of the numerical and analytical solutions**

```
TableForm[Table[{First[Ca[z] /. temp], Canalytical[z]},
  {z, 0, 10^-3, 0.0001}],
 TableHeadings -> {None, {"Numerical", "Analytical"}},
 TableAlignments -> Center]
```

Numerical	Analytical
0.2	0.2
0.188317	0.187624
0.178204	0.176814
0.169577	0.167477
0.162364	0.159537
0.156505	0.152928
0.151951	0.147594
0.148664	0.14349
0.146617	0.140584
0.145793	0.138849
0.146185	0.138273

8. Binary Batch Distillation

Mathematica

This problem requires the simultaneous solution of an ordinary differential equaiton with nonlinear algebraic equations. My approach was to write the nonlinear algebraic equations as a user written *Mathematica* function that could then be called from *Mathematica*'s numerical ordinary differential equaiton solver.

- **In this problem, k2 is a function of x2. A special function is written and the problem is solved in *Mathematica*'s NDSolve (numerical differential equation solver).**

- **The special function defining k2 as a function of x2 is written below as kn2.**

```
kn2[xn2_?NumberQ] :=
 Module[
 {data = {A1 -> 6.90565, B1 -> 1211.033, C1 -> 220.79,
    A2 -> 6.95464, B2 -> 1344.8, C2 -> 219.482}},
 k1 = ((10^(A1 - B1 / (T + C1))) / (760 * 1.2)) /. data;
 k2 = ((10^(A2 - B2 / (T + C2))) / (760 * 1.2)) /. data;
 (* determine equilibrium temperature *)
 temp =
  FindRoot[(1 - k1 (1 - xn2) - k2 xn2), {T, {90, 110}}] ;
 T1 = T /. temp;
(* calculate
   value of k2 at the equilibrium temperature *)
 ((10^(A2 - B2 / (T1 + C2))) / (760 * 1.2)) /. data
]
```

- **Once the special function is defined, the problem is solved using Mma's built-in numerical ODE solver, NDSolve .**

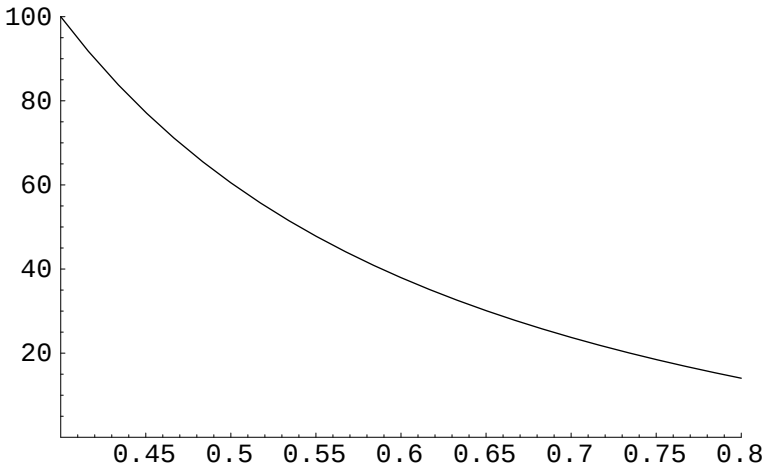
```
eqn2 = eqn = L'[x2] == L[x2] / (x2 (kn2[x2] - 1))
```

$$L'[x2] == \frac{L[x2]}{x2\,(-1 + kn2[x2])}$$

```
solution =
 NDSolve[{eqn2, L[0.4] == 100}, L[x2], {x2, 0.4, 0.8}]
```

```
{{L[x2] → InterpolatingFunction[{{0.4, 0.8}}, <>][x2]}}
```

```
Plot[L[x2] /. solution, {x2, 0.4, 0.8},
 PlotRange -> {{0.4, 0.8}, {0, 100}}]
```




```
- Graphics -
```

```
solution /. x2 -> 0.8 (* value of L at x2=0.8 *)
```

```
{{L[0.8] → 14.0417}}
```

```
T1 (* final temperature *)
```

```
108.572
```

9. Reversible, Exothermic, Gas Phase Reaction in a Catalytic Reactor

Mathematica

This problem requires the numerical solution of three nonlinear simultaneous differential equations. The advantage of the symbolic routines for this type of problem is that they can help the user with the book keeping of all the equations and variables. In this problem to make it easier to check the equations and avoid typing errors the data is treated as rules and the symbolic forum of the equations are preserved. If needed *Mathematica* could also track and process units; however, to streamline these calculations this is not done here.

- **In this case we preserve the symbolic forum of the equations by using the rule and replace capability in Mathematica**

- **Define data as a list of rules (the parameters in these equations are expressions of temperature and concentration)**

```
data = {k -> 0.5 Exp[5032 (1 / 450 - 1 / T[w])],
  CA -> 0.271 * (1 - x[w])
    (450 / T[w]) / (1 - 0.5 * x[w]) * y[w], CC ->
   0.271 * 0.5 x[w] * (450 / T[w]) / (1 - 0.5 * x[w]) * y[w],
  Kc -> 25000 * Exp[delH / 8.314 * (1 / 450 - 1 / T[w])],
  rA -> -k * (CA^2 - CC / Kc), Ta -> 500
    delH -> -40000, CPA -> 40, FA0 -> 5}
```

$$\left\{k \to 0.5\, E^{5032\left(\frac{1}{450}-\frac{1}{T[w]}\right)},\ CA \to \frac{121.95\,(1-x[w])\,y[w]}{T[w]\,(1-0.5\,x[w])},\right.$$
$$CC \to \frac{60.975\,x[w]\,y[w]}{T[w]\,(1-0.5\,x[w])},\ Kc \to 25000\,E^{0.120279\,delH\left(\frac{1}{450}-\frac{1}{T[w]}\right)},$$
$$\left.rA \to -k\left(CA^2-\frac{CC}{Kc}\right),\ Ta \to 500,\ delH \to -40000,\ CPA \to 40,\ FA0 \to 5\right\}$$

- **Define differential equations . The three equations are assigned to the variable names eqn1, eqn2, and eqn3. This simplifies the users task of checking for typos and avoids re-typing.**

```
eqn1 = x'[w] == -rA / FA0 //. data;
eqn2 = T'[w] ==
   ((0.8 (Ta - T[w]) + rA * delH) / (CPA * FA0)) //. data;
eqn3 = y'[w] ==
   -0.015 * (1 - 0.5 x[w]) (T[w] / 450) / (2 * y[w]) //.
  data;
```

- **Check eqn1**

```
eqn1 (* Check Equation 1 *)
```

$$x'[w] == 0.1\, E^{5032\left(\frac{1}{450}-\frac{1}{T[w]}\right)}\left(-\frac{0.002439\, E^{4811.16\left(\frac{1}{450}-\frac{1}{T[w]}\right)}\, x[w]\, y[w]}{T[w]\,(1-0.5\,x[w])}+\frac{14871.8\,(1-x[w])^2\, y[w]^2}{T[w]^2\,(1-0.5\,x[w])^2}\right)$$

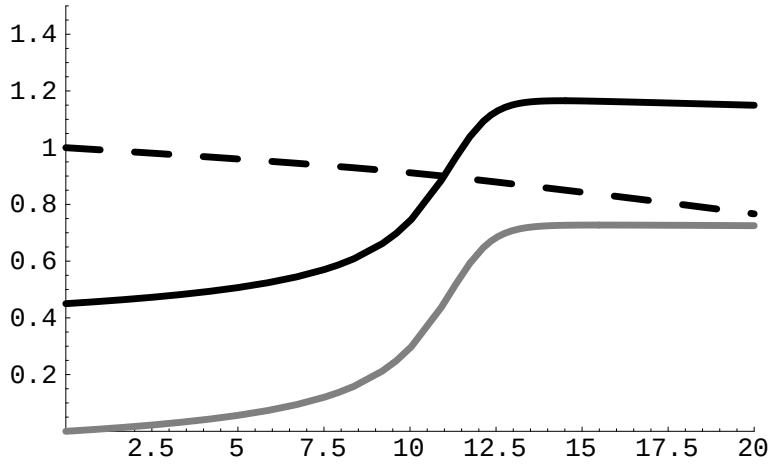
- **Solve the three ODE equations**

```
(* Mathematica gives numerical solutions
  to ODEs are interpolating polynomials
  as shown below. This makes it easy in
  Mathematica to view and use the solutions *)

solutions = NDSolve[{eqn1, x[0] == 0,
   eqn2, T[0] == 450, eqn3, y[0] == 1}, {x[w],
 T[w], y[w]}, {w, 0, 20}]
```

```
{{x[w] → InterpolatingFunction[{{0., 20.}}, <>][w],
  T[w] → InterpolatingFunction[{{0., 20.}}, <>][w],
  y[w] → InterpolatingFunction[{{0., 20.}}, <>][w]}}
```

- **Plot the three solutions versus w**

```
(* Plot of profiles for x, T, and y versus w *)

Plot[Evaluate[
  {x[w], T[w] / 1000, y[w]} /. solutions], {w, 0, 20},
 PlotRange -> {{0, 20}, {0, 1.5}},
  PlotStyle -> {{Thickness[0.01],
    GrayLevel[0.5]}, {Thickness[0.01]},
   {Thickness[0.01], Dashing[{0.05, 0.05}]}}]

(* x=dashed, T=black, y=gray *)
```

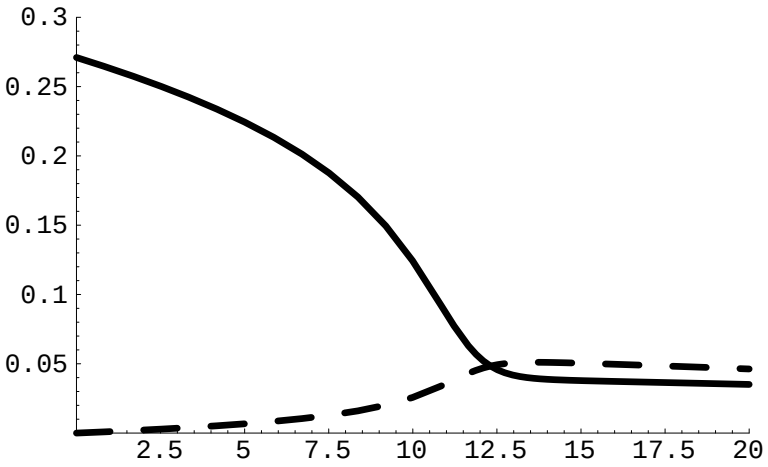


- Graphics -

- **Part (b) The rapid increase is due to the exotermic reation quickly accelerating due to the increasing temperature even though the reatant concentration is falling. Equilibrium is rapidly achieved after this hot spot appears. The temperature and conversion reduce only slightly due to the external heat transfer which tends to slightly cool the reator as the reating mixture contiues toward the reactor exit.**

- **Part (c) Plot CA and CC from 0<w<20.**

```
Plot[
 Evaluate[{{CA, CC} /. data} /. solutions], {w, 0, 20},
 PlotRange -> {{0, 20}, {0, 0.3}},
  PlotStyle -> {{Thickness[0.01]},
   {Thickness[0.01], Dashing[{0.05, 0.05}]}}]

(* CC=dashed, CA=solid black *)
```



- Graphics -

10. Dynamics of a Heated Tank with PI Temperature Control

Mathematica

This problem requires the solution of several simultaneous ordinary differential equaitons and the solution of a PI controller with step input. Problem 10 is the longest and most complicated of the set. *Mathematica* v3.0 gives the user the ability to use special symbols so one can match the actual mathematical equations. Symbol will match those of the problem statement.

■ Part (a) Demonstrate the open loop system with Kc = 0. At t = 10 minutes the inlet temperature T_i is changed from 80 ^oC to 40 ^oC. The Pade approximation is not necessary in *Mathematica* since a delay function is easy to write directly and is 100% accurate. However the Pade function will also be demonstrated for completeness.

■ Define parameters and other relations

```
data = {WCp -> 500 (*flow rate x heat capacity *),
  ρVCp -> 4000 (* denisty x volume x heat capacity *),
  τd -> 1 (* dead time *),
  τm -> 5 (* measured temperature *),
  Tr -> 80 (* set point temperature *),
  Kc -> 0 (* controller gain *),
  τI -> 2 (* integral time *),
  Δ -> If[t < 10, 0, 1] (* step function *),
  Ti -> 60 + Δ (-20) (* inlet temperature *),
  q -> 10000 + Kc (Tr - Tm[t]) + (Kc/τI) errsum[t],
  dTdt -> (WCp (Ti - T[t]) + q)/(ρVCp)};
```

```
data // ColumnForm (* display the inputs *)
```

$$WC_p \to 500$$
$$\rho VC_p \to 4000$$
$$\tau_d \to 1$$
$$\tau_m \to 5$$
$$T_r \to 80$$
$$K_c \to 0$$
$$\tau_I \to 2$$
$$\triangle \to \text{If}[t < 10, 0, 1]$$
$$T_i \to 60 - 20\,\triangle$$
$$q \to 10000 + \frac{\text{errsum}[t]\,K_c}{\tau_I} + K_c\,(T_r - \text{Tm}[t])$$
$$\text{dTdt} \to \frac{q + WC_p\,(T_i - T[t])}{\rho VC_p}$$

■ Define differential equations

```
eqn1 = T'[t] == (WCp (Ti - T[t]) + q) / ρVCp //. data
eqn2 = To'[t] == (T[t] - To[t] - (τd / 2) dTdt) 2 / τd //. data
eqn3 = Tm'[t] == ((To[t] - Tm[t]) / τm) //. data
eqn4 = errsum'[t] == Tr - Tm[t] //. data
```

$$T'[t] == \frac{10000 + 500\,(60 - 20\,\text{If}[t < 10, 0, 1] - T[t])}{4000}$$

$$To'[t] == 2\left(\frac{-10000 - 500\,(60 - 20\,\text{If}[t < 10, 0, 1] - T[t])}{8000} + T[t] - To[t]\right)$$

$$Tm'[t] == \frac{1}{5}\,(-Tm[t] + To[t])$$

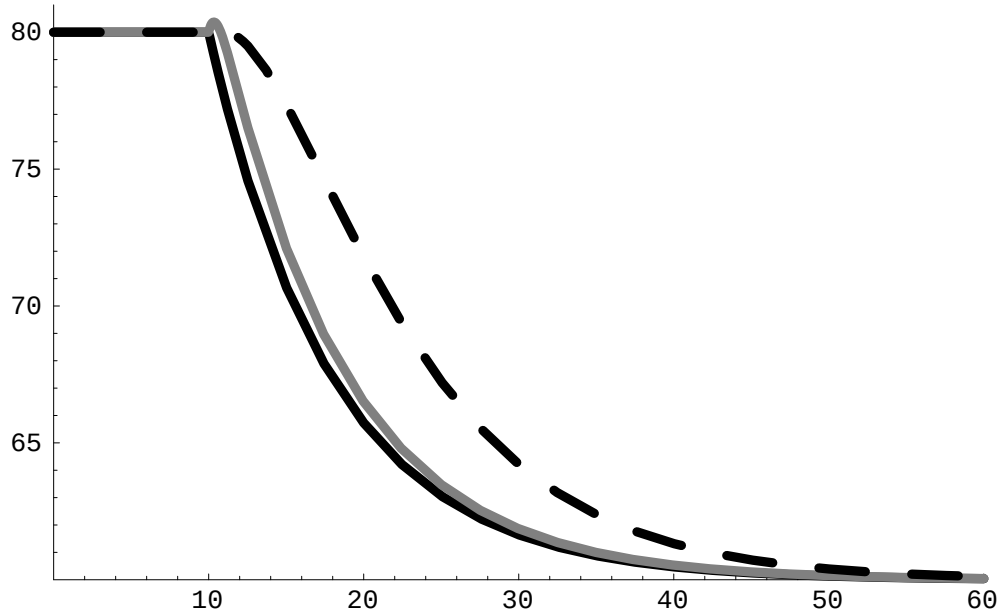
$$\text{errsum}'[t] == 80 - Tm[t]$$

- **Solve the differential equations**

```
solutions = NDSolve[{eqn1, T[0] == 80,
   eqn2, To[0] == 80,
   eqn3, Tm[0] == 80,
   eqn4, errsum[0] == 0},
   {T[t], To[t], Tm[t], errsum[t]}, {t, 0, 60}]
```

```
{{T[t] → InterpolatingFunction[{{0., 60.}}, <>][t],
  To[t] → InterpolatingFunction[{{0., 60.}}, <>][t],
  Tm[t] → InterpolatingFunction[{{0., 60.}}, <>][t],
  errsum[t] → InterpolatingFunction[{{0., 60.}}, <>][t]}}
```

```
Plot[Evaluate[
   {T[t], To[t], Tm[t]} /. solutions], {t, 0, 60},
 PlotRange -> {{0, 60}, {60, 81}},
 PlotStyle -> {{Thickness[0.01]},
    {Thickness[0.01], GrayLevel[0.5]},
    {Thickness[0.01], Dashing[{0.05, 0.05}]}}]

(* T is thin line, To is gray, and
 Tm is dashed *)
```



- Graphics -

■ Part (b) Change to PI control by making Kc->50

■ Data definition

```
data2 = {WCp -> 500 (*flow rate x heat capacity *),
  ρVCp -> 4000 (* denisty x volume x heat capacity *),
  τd -> 1 (* dead time *),
  τm -> 5 (* measured temperature *),
  Tr -> 80 (* set point temperature *),
  Kc -> 50 (* controller gain *),
  τI -> 2 (* integral time *),
  Δ -> If[t < 10, 0, 1] (* step function *),
  Ti -> 60 + Δ (-20) (* inlet temperature *),
  q -> 10000 + Kc (Tr - Tm[t]) + (Kc / τI) errsum[t],
  dTdt -> (WCp (Ti - T[t]) + q) / (ρVCp)};
```

```
data2 // ColumnForm (* display the inputs *)
```

$$WC_p \to 500$$

$$\rho VC_p \to 4000$$

$$\tau_d \to 1$$

$$\tau_m \to 5$$

$$T_r \to 80$$

$$K_c \to 50$$

$$\tau_I \to 2$$

$$\Delta \to \text{If}[t < 10, 0, 1]$$

$$T_i \to 60 - 20\,\Delta$$

$$q \to 10000 + \frac{\text{errsum}[t]\,K_c}{\tau_I} + K_c\,(T_r - \text{Tm}[t])$$

$$\text{dTdt} \to \frac{q + WC_p\,(T_i - T[t])}{\rho VC_p}$$

- **Place data into equations**

```
eqn1 = T'[t] == (WCp (Ti - T[t]) + q) / (ρVCp) //. data2
eqn2 = To'[t] == (T[t] - To[t] - (τd/2) dTdt) 2/τd //. data2
eqn3 = Tm'[t] == ((To[t] - Tm[t])/τm) //. data2
eqn4 = errsum'[t] == Tr - Tm[t] //. data2
```

$$T'[t] == \frac{1}{4000}\,(10000 + 25\,\text{errsum}[t] + 500\,(60 - 20\,\text{If}[t < 10,\ 0,\ 1] - T[t]) + 50\,(80 - Tm[t]))$$

$$To'[t] == 2\left(T[t] + \frac{1}{8000}\,(-10000 - 25\,\text{errsum}[t] - 500\,(60 - 20\,\text{If}[t < 10,\ 0,\ 1] - T[t]) - 50\,(80 - Tm[t])) - To[t]\right)$$

$$Tm'[t] == \frac{1}{5}\,(-Tm[t] + To[t])$$

$$\text{errsum}'[t] == 80 - Tm[t]$$

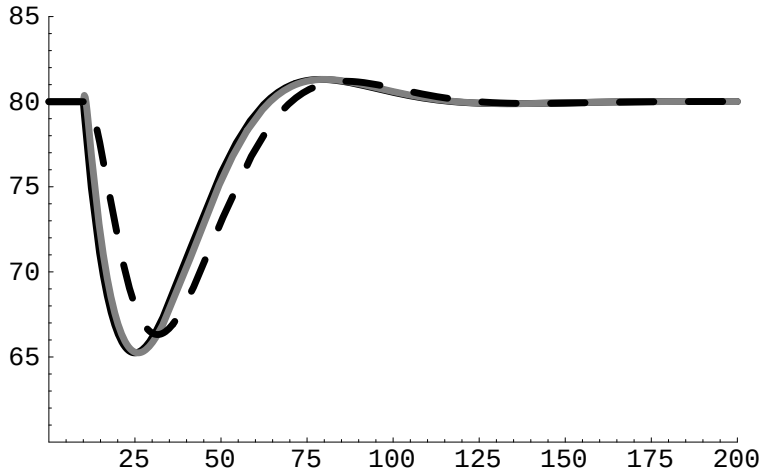
- **Solve equations**

```
solutions2 = NDSolve[{eqn1, T[0] == 80, eqn2,
   To[0] == 80, eqn3, Tm[0] == 80, eqn4, errsum[0] == 0},
  {T[t], To[t], Tm[t], errsum[t]}, {t, 0, 200}]
```

```
{{T[t] → InterpolatingFunction[{{0., 200.}}, <>][t],
  To[t] → InterpolatingFunction[{{0., 200.}}, <>][t],
  Tm[t] → InterpolatingFunction[{{0., 200.}}, <>][t],
  errsum[t] →
   InterpolatingFunction[{{0., 200.}}, <>][t]}}
```

■ Plot results

```
Plot[Evaluate[
  {T[t], To[t], Tm[t]} /. solutions2], {t, 0, 200},
 PlotRange -> {{0, 200}, {60, 85}},
 PlotStyle -> {{Thickness[0.01]},
   {Thickness[0.01], GrayLevel[0.5]},
   {Thickness[0.01], Dashing[{0.05, 0.05}]}}]

(* T is black line, To is gray, and Tm is dashed *)
```



- Graphics -

■ Part (c) Change to PI control by making Kc->500

■ Data definition

```
data3 = {WCp -> 500 (*flow rate x heat capacity *),
  ρVCp -> 4000 (* denisty x volume x heat capacity *),
  τd -> 1 (* dead time *),
  τm -> 5 (* measured temperature *),
  Tr -> 80 (* set point temperature *),
  Kc -> 500 (* controller gain *),
  τI -> 2 (* integral time *),
  Δ -> If[t < 10, 0, 1] (* step function *),
  Ti -> 60 + Δ (-20) (* inlet temperature *),
  q -> 10000 + Kc (Tr - Tm[t]) + (Kc/τI) errsum[t],
  dTdt -> (WCp (Ti - T[t]) + q)/(ρVCp)};
```

```
data3 // ColumnForm (* display the inputs *)
```

$$WC_p \to 500$$

$$\rho VC_p \to 4000$$

$$\tau_d \to 1$$

$$\tau_m \to 5$$

$$T_r \to 80$$

$$K_c \to 500$$

$$\tau_I \to 2$$

$$\Delta \to \text{If}[t < 10, 0, 1]$$

$$T_i \to 60 - 20\,\Delta$$

$$q \to 10000 + \frac{\text{errsum}[t]\,K_c}{\tau_I} + K_c\,(T_r - \text{Tm}[t])$$

$$dTdt \to \frac{q + WC_p\,(T_i - T[t])}{\rho VC_p}$$

■ Place data into equations

```
eqn1 = T'[t] == (WCp (Ti - T[t]) + q) / (ρVCp) //. data3
eqn2 = To'[t] == (T[t] - To[t] - (τd/2) dTdt) 2/τd //. data3
eqn3 = Tm'[t] == ((To[t] - Tm[t])/τm) //. data3
eqn4 = errsum'[t] == Tr - Tm[t] //. data3
```

$$T'[t] == \frac{1}{4000}\,(10000 + 250\,\text{errsum}[t] + 500\,(60 - 20\,\text{If}[t < 10,\ 0,\ 1] - T[t]) + 500\,(80 - Tm[t]))$$

$$To'[t] == 2\left(T[t] + \frac{1}{8000}\,(-10000 - 250\,\text{errsum}[t] - 500\,(60 - 20\,\text{If}[t < 10,\ 0,\ 1] - T[t]) - 500\,(80 - Tm[t])) - To[t]\right)$$

$$Tm'[t] == \frac{1}{5}\,(-Tm[t] + To[t])$$

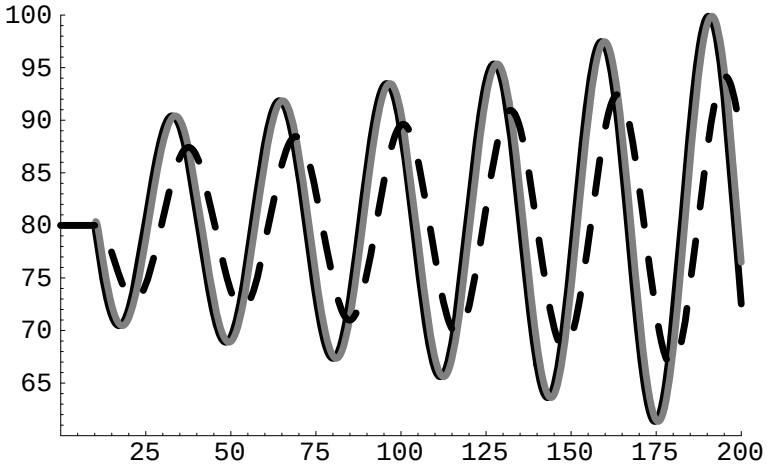
$$\text{errsum}'[t] == 80 - Tm[t]$$

```
solutions3 = NDSolve[{eqn1, T[0] == 80, eqn2,
   To[0] == 80, eqn3, Tm[0] == 80, eqn4, errsum[0] == 0},
  {T[t], To[t], Tm[t], errsum[t]}, {t, 0, 200}]
```

```
{{T[t] → InterpolatingFunction[{{0., 200.}}, <>][t],
  To[t] → InterpolatingFunction[{{0., 200.}}, <>][t],
  Tm[t] → InterpolatingFunction[{{0., 200.}}, <>][t],
  errsum[t] →
   InterpolatingFunction[{{0., 200.}}, <>][t]}}
```

■ Plot results

```
Plot[Evaluate[
  {T[t], To[t], Tm[t]} /. solutions3], {t, 0, 200},
 PlotRange -> {{0, 200}, {60, 100}},
 PlotStyle -> {{Thickness[0.01]},
   {Thickness[0.01], GrayLevel[0.5]},
   {Thickness[0.01], Dashing[{0.05, 0.05}]}}]

(* T is black line, To is gray, and Tm is dashed *)
```



- Graphics -

■ Part (d) Closed loop with proportional control only

■ Data definition

```
data4 = {WCp -> 500 (*flow rate x heat capacity *),
  ρVCp -> 4000 (* denisty x volume x heat capacity *),
  τd -> 1 (* dead time *),
  τm -> 5 (* measured temperature *),
  Tr -> 80 (* set point temperature *),
  Kc -> 500 (* controller gain *),
  τI -> 2 (* integral time *),
  Δ -> If[t < 10, 0, 1] (* step function *),
  Ti -> 60 + Δ (-20) (* inlet temperature *),
  q -> 10000 + Kc (Tr - Tm[t]), dTdt -> (WCp (Ti - T[t]) + q)/(ρVCp)};
```

```
data4 // ColumnForm (* display the inputs *)
```

$WC_p \to 500$

$\rho VC_p \to 4000$

$\tau_d \to 1$

$\tau_m \to 5$

$T_r \to 80$

$K_c \to 500$

$\tau_I \to 2$

$\Delta \to \text{If}[t < 10, 0, 1]$

$T_i \to 60 - 20\,\Delta$

$q \to 10000 + K_c\,(T_r - \text{Tm}[t])$

$\text{dTdt} \to \frac{q + WC_p\,(T_i - T[t])}{\rho VC_p}$

■ Place data into equations

```
eqn1 = T'[t] == (WCp (Ti - T[t]) + q) / (ρVCp) //. data4
eqn2 = To'[t] == (T[t] - To[t] - (τd/2) dTdt) 2/τd //. data4
eqn3 = Tm'[t] == ((To[t] - Tm[t])/τm) //. data4
eqn4 = errsum'[t] == Tr - Tm[t] //. data4
```

$$T'[t] == \frac{1}{4000}(10000 + 500\,(60 - 20\,\text{If}[t < 10, 0, 1] - T[t]) + 500\,(80 - Tm[t]))$$

$$To'[t] == 2\left(T[t] + \frac{1}{8000}(-10000 - 500\,(60 - 20\,\text{If}[t < 10, 0, 1] - T[t]) - 500\,(80 - Tm[t])) - To[t]\right)$$

$$Tm'[t] == \frac{1}{5}(-Tm[t] + To[t])$$

$$errsum'[t] == 80 - Tm[t]$$

■ Solve equations

```
solutions4 = NDSolve[{eqn1, T[0] == 80, eqn2,
   To[0] == 80, eqn3, Tm[0] == 80, eqn4, errsum[0] == 0},
  {T[t], To[t], Tm[t], errsum[t]}, {t, 0, 60}]
```

```
{{T[t] → InterpolatingFunction[{{0., 60.}}, <>][t],
  To[t] → InterpolatingFunction[{{0., 60.}}, <>][t],
  Tm[t] → InterpolatingFunction[{{0., 60.}}, <>][t],
  errsum[t] → InterpolatingFunction[{{0., 60.}}, <>][t]}}
```

- **Plot results**

```
Plot[Evaluate[
  {T[t], To[t], Tm[t]} /. solutions4], {t, 0, 60},
 PlotRange -> {{0, 60}, {65, 82}},
 PlotStyle -> {{Thickness[0.01]},
   {Thickness[0.01], GrayLevel[0.5]},
   {Thickness[0.01], Dashing[{0.05, 0.05}]}}]

(* T is black line, To is gray, and Tm is dashed *)
```

```
- Graphics -
```

■ Part (e) Closed loop with peformance limits on q

- **Data definition**

```
data5 = {WCp -> 500 (*flow rate x heat capacity *),
  ρVCp -> 4000 (* denisty x volume x heat capacity *),
  τd -> 1 (* dead time *),
  τm -> 5 (* measured temperature *),
  Tr -> 80 + Δ * 10(* set point temperature *),
  Kc -> 5000 (* controller gain *),
  τI -> 2 (* integral time *),
  Δ -> If[t < 10, 0, 1] (* step function *),
  Ti -> 60(* inlet temperature *),
  q -> 10000 + Kc (Tr - Tm[t]),
  qlim -> If[q < 0, 0, If[q >= 2.6 * 10000, 2.6 * 10000, q]]
  (* calculate the limits on q *),
  dTdt -> (WCp (Ti - T[t]) + qlim) / (ρVCp)};
```

```
data5 // ColumnForm (* display the inputs *)
```

$$WC_p \to 500$$
$$\rho VC_p \to 4000$$
$$\tau_d \to 1$$
$$\tau_m \to 5$$
$$T_r \to 80 + 10\,\triangle$$
$$K_c \to 5000$$
$$\tau_I \to 2$$
$$\triangle \to \text{If}[t < 10,\ 0,\ 1]$$
$$T_i \to 60$$
$$q \to 10000 + K_c\,(T_r - \text{Tm}[t])$$
$$\text{qlim} \to \text{If}[q < 0,\ 0,\ \text{If}[q \geq 2.6\,10000,\ 2.6\,10000,\ q]]$$
$$\text{dTdt} \to \frac{\text{qlim} + WC_p\,(T_i - T[t])}{\rho VC_p}$$

```
data4 = {WC -> 500, rhoVCp -> 4000, taud -> 1, taum -> 5,
  Tr -> 80 + step * 10, Kc -> 5000, step -> If[t < 10, 0, 1],
  Ti -> 60, q -> 10000 + Kc (Tr - Tm[t]), qlim ->
   If[q < 0, 0, If[q >= 2.6 * 10000, 2.6 * 10000, q]],
  dTdt -> (WC (Ti - T[t]) + qlim) / rhoVCp}
```

$$\{WC \to 500,\ \text{rhoVCp} \to 4000,$$
$$\text{taud} \to 1,\ \text{taum} \to 5,\ Tr \to 80 + 10\,\text{step},\ Kc \to 5000,$$
$$\text{step} \to \text{If}[t < 10,\ 0,\ 1],\ Ti \to 60,\ q \to 10000 + Kc\,(Tr - \text{Tm}[t]),$$
$$\text{qlim} \to \text{If}[q < 0,\ 0,\ \text{If}[q \geq 2.6\,10000,\ 2.6\,10000,\ q]],$$
$$\text{dTdt} \to \frac{\text{qlim} + WC\,(Ti - T[t])}{\text{rhoVCp}}\}$$

■ Place data into equations

$$\text{eqn1} = \text{T}'[\text{t}] == \frac{(\text{WC}_\text{p}\,(\text{T}_\text{i} - \text{T}[\text{t}]) + \text{qlim})}{\rho \text{VC}_\text{p}} \;//.\; \text{data5}$$

$$\text{eqn2} = \text{To}'[\text{t}] == \left(\text{T}[\text{t}] - \text{To}[\text{t}] - \left(\frac{\tau_\text{d}}{2}\right) \text{dTdt}\right) \frac{2}{\tau_\text{d}} \;//.\; \text{data5}$$

$$\text{eqn3} = \text{Tm}'[\text{t}] == \left(\frac{(\text{To}[\text{t}] - \text{Tm}[\text{t}])}{\tau_\text{m}}\right) \;//.\; \text{data5}$$

$$\text{eqn4} = \text{errsum}'[\text{t}] == \text{T}_\text{r} - \text{Tm}[\text{t}] \;//.\; \text{data5}$$

```
T'[t] ==
  1/4000 (If[10000 + 5000 (80 + 10 If[t < 10, 0, 1] - Tm[t]) < 0,
      0, If[10000 + 5000 ((80 + 10 If[t < 10, 0, 1]) - Tm[t]) ≥
        2.6 10000, 2.6 10000,
       10000 + 5000 ((80 + 10 If[t < 10, 0, 1]) - Tm[t])]] +
    500 (60 - T[t]))

To'[t] == 2 (1/8000 (-If[10000 +
             5000 (80 + 10 If[t < 10, 0, 1] - Tm[t]) < 0, 0, If[
            10000 + 5000 ((80 + 10 If[t < 10, 0, 1]) - Tm[t]) ≥
             2.6 10000,
            2.6 10000, 10000 +
             5000 ((80 + 10 If[t < 10, 0, 1]) - Tm[t])]] -
        500 (60 - T[t])) +
    T[t] - To[t])

Tm'[t] == 1/5 (-Tm[t] + To[t])

errsum'[t] == 80 + 10 If[t < 10, 0, 1] - Tm[t]
```

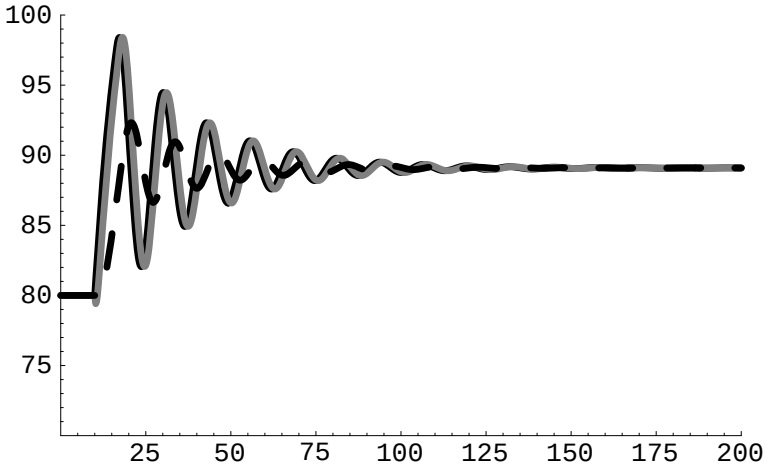
■ Solve equations

```
solutions5 = NDSolve[{eqn1, T[0] == 80, eqn2,
   To[0] == 80, eqn3, Tm[0] == 80, eqn4, errsum[0] == 0},
  {T[t], To[t], Tm[t], errsum[t]}, {t, 0, 200}]
```

```
{{T[t] → InterpolatingFunction[{{0., 200.}}, <>][t],
  To[t] → InterpolatingFunction[{{0., 200.}}, <>][t],
  Tm[t] → InterpolatingFunction[{{0., 200.}}, <>][t],
  errsum[t] →
   InterpolatingFunction[{{0., 200.}}, <>][t]}}
```

■ Plot results

```
Plot[Evaluate[
  {T[t], To[t], Tm[t]} /. solutions5], {t, 0, 200},
 PlotRange -> {{0, 200}, {70, 100}},
 PlotStyle -> {{Thickness[0.01]},
   {Thickness[0.01], GrayLevel[0.5]},
   {Thickness[0.01], Dashing[{0.05, 0.05}]}}]

(* T is black line, To is gray, and Tm is dashed *)
```



- Graphics -

■ Generate the plot of q and qlim versus time. Define the q and qlim time dependent functions.

```
temp1 = (10000 + Kc (Tr - Tm[t]) //. data5) /. solutions5
```

```
{10000 + 5000 (80 + 10 If[t < 10, 0, 1] -
     InterpolatingFunction[{{0., 200.}}, <>][t])}
```

```
qfun[t_] := temp1 (* q time dependent function *)
```

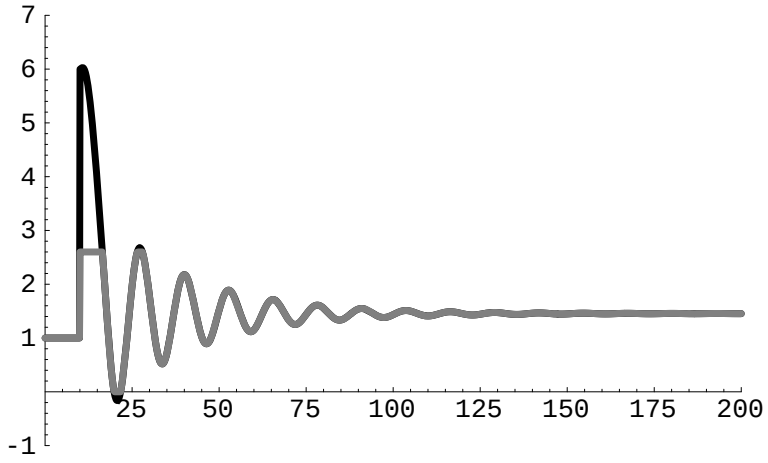
```
temp2 = (If[q < 0, 0,
     If[q >= 2.6 * 10000, 2.6 * 10000, q]] //. data5) /.
  solutions5
```

```
{If[10000 + 5000 (80 + 10 If[t < 10, 0, 1] -
       InterpolatingFunction[{{0., 200.}}, <>][t]) < 0,
  0,
  If[10000 + 5000 ((80 + 10 If[t < 10, 0, 1]) -
       InterpolatingFunction[{{0., 200.}}, <>][t]) ≥
    2.6 10000, 2.6 10000,
   10000 + 5000 ((80 + 10 If[t < 10, 0, 1]) -
       InterpolatingFunction[{{0., 200.}}, <>][t])]]}
```

```
qlimfun[t_] := temp2 (* qlim time dependent function *)
```

```
(* Note that the plot is scaled by
    dividing both "q             " and "qlim
  " by 10^4 *)
```

```
Plot[{qfun[t] / 10^4, qlimfun[t] / 10^4},
 {t, 0, 200}, PlotPoints → 25,
 PlotRange -> {{0, 200}, {-1, 7}},
 PlotStyle -> {{Thickness[0.01]},
   {Thickness[0.01], GrayLevel[0.5]}}]

(* Black line is "q" and gray line is "qlim" *)
```



- Graphics -