TK Solver Case Study: Pharmacokinetics

The content for this example was provided by Professor Prasad Dhurjati of the University of Delaware. It is used there in the class Applied Mathematics for Chemical Engineers (MATH 305). Professor Dhurjati uses this as the semester laboratory project. The students were asked to use the software programs Matlab and Simulink to solve this problem. This case study shows how TK Solver can be used to model and solve the problem.

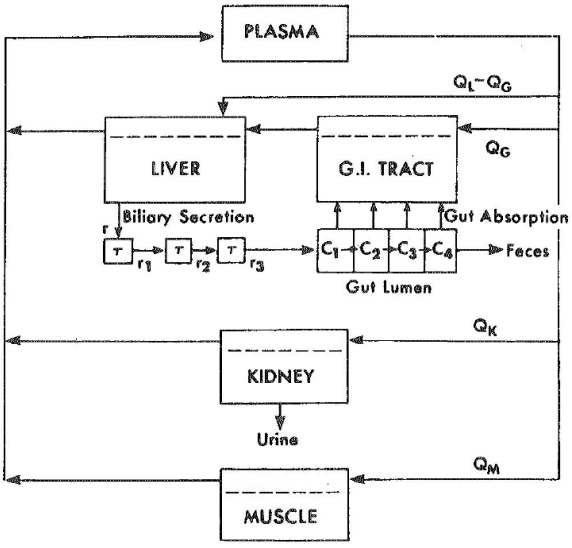
TK Solver is a powerful and efficient tool for solving systems of differential equations as part of a mathematical model. Such models can be solved (or backsolved) for any variables of interest. Results are concisely summarized on the TK variable sheet and in tables and plots.

Background

Dosage levels of drugs are usually prescribed on the basis of the patient's weight, with the standard dosage level often specified in units of mg/kg, from which the physician then calculates the actual dose for a particular patient. Implicit in this approach is an assumption that the response of the body to the drug is linear, and that the concentration of the drug in the blood or in the target tissue would then be above a minimum level for therapeutic efficacy for an adequate time. Whether this assumption is valid is usually determined empirically, using animal experiments and clinical trials. A more rigorous theoretical basis on which dosage levels might be calculated is provided by the science of pharmacokinetics, in which the body is modeled as an assembly of continuously stirred tank reactions (CSTRs) with different effective volumes and flow rates. This case study shows how TK Solver can be used to solve a set of pharmacokinetic equations in order to model the distribution of methotrexate (MTX), a drug used mainly to treat cancer.

Model

The pharmacokinetic model that will be used is that published by Bischoff et al. (J. Pharm. Sci., **60**, 1128 (1971)). The overall scheme is shown in the adjacent diagram above, where each block represents a CSTR, the Q's are flow rates, the C's are concentrations, the r's are rates and the subscripts denote the different organs/ "compartments". Plasma is the cell-free portion of blood, in which the drug is carried in solution in the blood and distributed to the organs. MTX is eliminated from the body in the urine and by metabolic processing in the liver, from where it is secreted into the gut in bile and then excreted in the feces.



The equations describing the distribution of MTX with time are then a set of first-order ODEs, one for each of the compartments shown in the diagram:

Plasma:

$$V_P \frac{dC_P}{dt} = Q_L \frac{C_L}{R_L} + Q_K \frac{C_K}{R_K} + Q_M \frac{C_M}{R_M} - (Q_L + Q_K + Q_M)C_P$$

Muscle:

$$V_M \frac{dC_M}{dt} = Q_M (C_P - \frac{C_M}{R_M})$$

Kidney:

$$V_K \frac{dC_K}{dt} = Q_K (C_P - \frac{C_K}{R_K}) - k_K \frac{C_K}{R_K}$$

Liver:

$$V_L \frac{dC_L}{dt} = (Q_L - Q_G)(C_P - \frac{C_L}{R_L}) + Q_G (\frac{C_G}{R_G} - \frac{C_L}{R_L}) - \frac{k_L}{K_L} \frac{C_L}{R_L}$$

Secretion/transport kinetics:

$$\tau \frac{dr_1}{dt} = \frac{k_L}{K_L} \frac{C_L}{R_L} - r_1 \qquad \tau \frac{dr_i}{dt} = r_{i-1} - r_i \qquad (i = 2,3)$$

Gut tissue (i.e., intestinal walls, etc.):

$$V_G \frac{dC_G}{dt} = Q_G (C_P - \frac{C_G}{R_G}) + \sum_{i=1}^{4} \frac{1}{4} \left(\frac{k_G C_i}{K_G + C_i} + bC_i \right)$$

Gut lumen (i.e., contents of intestines):

$$\frac{dC_{GL}}{dt} = \frac{1}{4} \sum_{i=1}^{4} \frac{dC_i}{dt}$$

$$\frac{V_{GL}}{4} \frac{dC_1}{dt} = r_3 - k_F V_{GL} C_1 - \frac{1}{4} \left(\frac{k_G C_1}{K_G + C_1} + bC_1 \right)$$

$$\frac{V_{GL}}{4} \frac{dC_i}{dt} = k_F V_{GL} \left(C_{i-1} - C_i \right) - \frac{1}{4} \left(\frac{k_G C_i}{K_G + C_i} + bC_i \right) \qquad (i = 2,3,4)$$

Initially no MTX is present anywhere except in the plasma, where the concentration $c_P = c_0$.

Required data (volumes, flow rates, and other parameters) is available for several different species of test subjects and is shown in the following table.

Table 1 – Model Parameters for Methotrexate in Several Species

Parameter	**Mouse**	**Rat**	**Dog/monkey 5 kg**	**Dog 17 kg**	**Man**
Body weight, g	22	200	5000	17,000	70,000
Volume, ml					
V_P	1.0	9.0	220	650	3,000
V_M	10.0	100	2500	7,500	35,000
V_K	0.34	1.9	30	76	280
V_L	1.3	8.3	135	360	1,350
V_G	1.5	11.0	230	640	2,100
V_{GL}	1.5	11.0	230	640	2,100
Plasma flow rate, ml/min					
Q_M	0.5	3.0	50	140	420
Q_K	0.8	5.0	74	190	700
Q_L	1.1	6.5	92	220	800
Q_G	0.9	5.3	75	190	700
Tissue/plasma equilibrium distribution ratio for linear binding					
R_M	0.15	0.15	0.15	0.15	0.15
R_K	3.0	3.0	14	14	3.0
R_L	10	3.0	2.0	2.0	3.0
R_G	1.0	1.0	1.0	1.0	1.0
Kidney clearance					
k_K, ml/min	0.2	1.1	20	56	190
Bile secretion parameters					
Clearance k_L/K_L,. ml/min	0.4	3.0	2.0	8	200
τ, min	2.0	2.0	6	8.0	10
Gut-lumen parameters					
k_F, min^{-1}	0.01	0.01	0.0022	0.0015	0.001
k_G, µg/min	0.20	20	340	1,000	1,900
K_G, µg/ml	6.0	200	200	200	200
b, ml/min	0.001				

The TK Solver Model

The first step is to create a procedure function defining the system of differential equations. We call the function f. Function f will be processed by one of TK's built-in integrators which require that the procedure must have three input variables representing the derivative matrix (y`), dependent variables matrix (y), and independent variable list (t), in that order. All other variables must be passed into the procedure as parameter variables or list elements.

We start by assigning variable names to the solution matrix columns. The first column will contain the CP values. The second column will contain the CM values. In total, there are 13 differential equations to be solved.

Statement
CP = y[1]
CM = y[2]
CK = y[3]
CL = y[4]
r1 = y[5]
r2 = y[6]
r3 = y[7]
CG = y[8]
CGL = y[9]
C1 = y[10]
C2 = y[11]
C3 = y[12]
C4 = y[13]

The expression y[1] represents the first column of the solution matrix or "the first function of the independent variable t". The differential equations can now be entered using these variable names. Here are the equations for the plasma, muscle, kidney, and liver.

Statement
y`[1] = (QL*CL/RL + QK*CK/RK + QM*CM/RM - (QL+QK+QM)*CP)/VP
y`[2] = QM*(CP-CM/RM)/VM
y`[3] = (QK*(CP-CK/RK) - kK*CK/RK)/VK
y`[4] = ((QL-QG)*(CP-CL/RL) + QG*(CG/RG-CL/RL) - kLratio*CL/RL)/VL

The expression y`[1] represents "the derivative of the first function with respect to t". Variables such as QL, RL and QK must be assigned as parameter variables of the function and their values will come from the variable sheet.

The next three equations involve the secretion/transport kinetics.

Statement
y`[5] = (kLratio*CL/RL - r1)/tau
y`[6] = (r1 - r2)/tau
y`[7] = (r2 - r3)/tau

The gut tissue calculations require a series of steps, summing four stages of absorption.

Statement
s1 = kG*C1/(KG+C1) + b*C1
s2 = kG*C2/(KG+C2) + b*C2
s3 = kG*C3/(KG+C3) + b*C3
s4 = kG*C4/(KG+C4) + b*C4
s = (s1+s2+s3+s4)/4
y`[8] = (QG*(CP-CG/RG) + s)/VG

Similarly, the gut lumen calculations involve a series of steps.

Statement
y`[10] = (r3 - kF*VGL*C1 - 0.25*(kG*C1/(KG+C1) + b*C1))*4/VGL
y`[11] = (kF*VGL*(C1-C2) - 0.25*(kG*C2/(KG+C2) + b*C2))*4/VGL
y`[12] = (kF*VGL*(C2-C3) - 0.25*(kG*C3/(KG+C3) + b*C3))*4/VGL
y`[13] = (kF*VGL*(C3-C4) - 0.25*(kG*C4/(KG+C4) + b*C4))*4/VGL
y`[9] = (y`[10]+y`[11]+y`[12]+y`[13])/4

The sequence of calculations is important in a procedure function. Since y`[9] requires values of y`[10], y`[11], y`[12], and y`[13], those values must be computed first.

Here is the complete list of Parameter Variables for procedure f:
VP,VM,VK,VL,VG,VGL,QM,QK,QL,QG,RM,RK,RL,RG,kK,kLratio,tau,kF,kG,KG,b .
Function f is now ready for integration.

The next step is to create the solution matrix on the list sheet. We call the matrix y. The name is arbitrary at this point. Then we open the list subsheet and enter the names of the solution lists in the same order we defined them in the procedure function f.

Value
CP
CM
CK
CL
r1
r2
r3
CG
CGL
C1
C2
C3
C4

We also create a table to display the independent variable list t and the solution matrix lists. Before solving, we include the initial conditions for each of the lists, based on the sample plots from the Bischoff paper which are also shown at the end of this case study . Here is the table.

Element	t	CP	CM	CK	CL	r1	r2	r3	CG	CGL	C1	C2	C3	C4
1	0	220	22	220	220	0	0	0	0	0	0	0	0	0

Next, we go to the rule sheet to call an integrator function.

Rule
call ODE_STIFFR('f,0,240,'y,'t)

We use the ODE_STIFFR integrator which requires five inputs – the name of the function containing the differential equations (‘f), the start (0) and end (240) points of integration, the name of the solution matrix list (‘y) and the name of the independent variable list (‘t). ODE_STIFFR uses self-adjusting step sizes in producing an accurate solution.

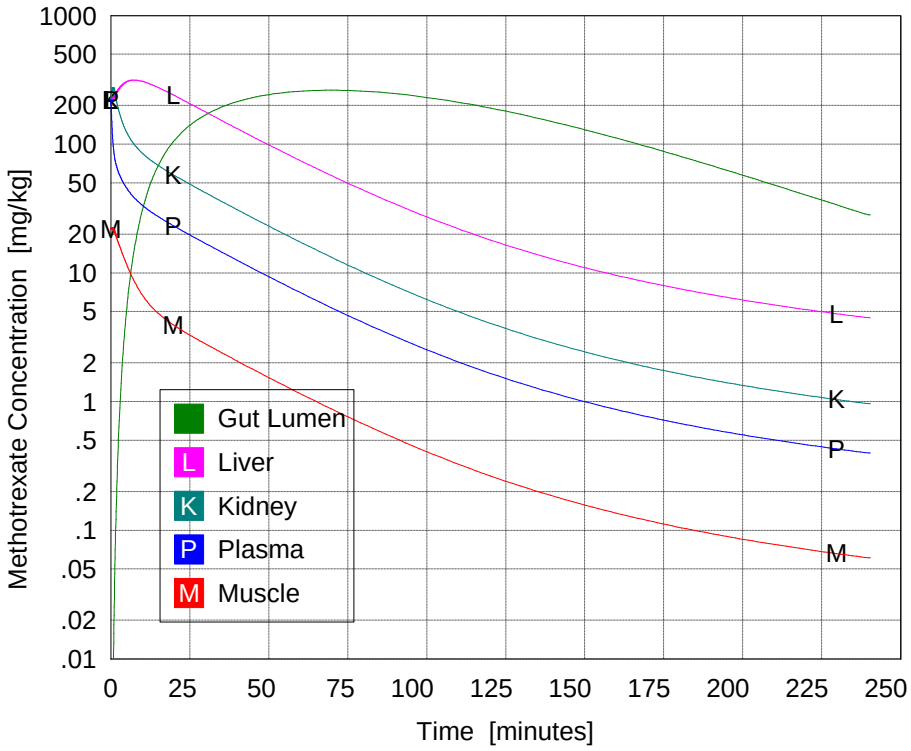
The next step is to organize the variable sheet and provide inputs. Here are the inputs for the mouse data. We also applied a global format to the TK model, using 4 decimal places and leading zeros.

Status	Input	Name	Output	Unit	Comment
					Volumes
	1	VP		ml	Plasma
	10	VM		ml	Muscle
	0.34	VK		ml	Kidney
	1.3	VL		ml	Liver
	1.5	VG		ml	Gut tissue
	1.5	VGL		ml	Gut lumen
					Flow Rates
	0.5	QM		ml/min	Muscle
	0.8	QK		ml/min	Kidney
	1.1	QL		ml/min	Liver
	0.9	QG		ml/min	Gut tissue
					Tissue/Plasma Equilibrium Ratios
	0.15	RM			Muscle
	3	RK			Kidney
	10	RL			Liver
	1	RG			Gut tissue
	0.2	kK		ml/min	Kidney clearance
	0.4	kLratio		ml/min	Bile secretion clearance
	2	tau		min	Bile secretion time constant
	0.01	kF		1/min	Gut lumen parameter
	0.2	kG		ug/ml	Gut lumen parameter
	6	KG		ug/ml	Gut lumen parameter
	0.001	b		ml/min	Gut lumen parameter

The model is now ready to solve. Solving, the solution table contains 290 time steps. Here are the first few rows of the table.

Element	t	CP	CM	CK	CL	r1	r2	r3	CG	CGL	C1	C2	C3	C4
1	0	220	22	220	220	0	0	0	0	0	0	0	0	0
2	1E-4	220	22	220	220	6E-4	2E-8	4E-13	0.017	8E-18	3E-17	8E-23	6E-28	4E-33
3	3E-4	219.9	22	220.1	220	0.001	1E-7	6E-12	0.042	3E-16	1E-15	4E-21	2E-26	1E-31
4	6E-4	219.8	22	220.2	220	0.003	4E-7	4E-11	0.08	4E-15	2E-14	8E-20	4E-25	4E-30
5	0.001	219.6	22	220.3	220	0.005	1E-6	2E-10	0.136	3E-14	1E-13	1E-18	9E-24	1E-28

We can also plot the solution lists. Here is a line chart with t on the horizontal axis and CP, CM, CK, CL, and CGL on the vertical axis. The plot uses a log scale for the vertical axis with the y-axis limits set to 0.01 and 1000 to avoid an error in taking the log of 0 at the initial condition. This plot compares well with Figure 2 from the Bischoff paper. Note the difference in units.



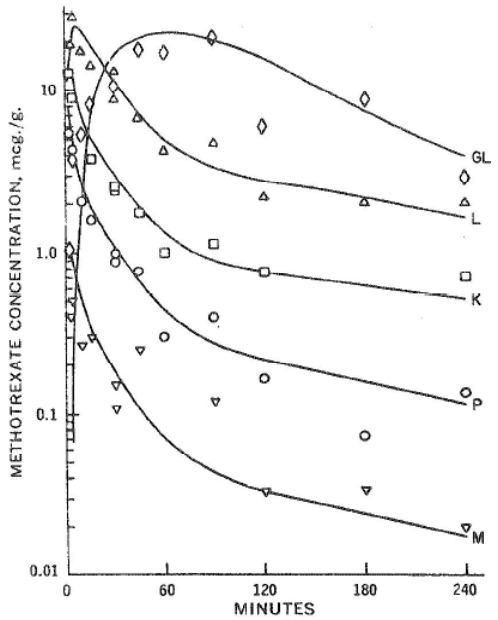


Figure 2—*Model prediction* versus *experimental results, mice, 3 mg./kg. i.v. Solid lines are model predictions; symbols are experimental data. Key:* GL (◇), *small intestine;* L (△), *liver;* K (□), *kidney;* P (○), *plasma; and* M (▽), *muscle.*

Enhancing the TK Model

There are several simple enhancements that can be made to the TK model to greatly expand its value. (1) We can create variables for the initial condition values and automate their placement in the lists. (2) We can create TK list functions to interpolate within the solution matrix, allowing for solutions at any time point. (3) We can improve the accuracy by interpolating on the logs of the solutions. (4) We can backsolve the model for initial conditions or parameter values that result in a particular solution. (5) We can create an input table for each of the five species discussed in the Bischoff paper to automate the task of providing parameter values.

The following rules can be used to place initial conditions into the solution matrix.

Rule
place('CP,1) = CP_0
place('CM,1) = CM_0
place('CK,1) = CK_0
place('CL,1) = CL_0
place('CG,1) = CG_0
place('CGL,1) = CGL_0

The new variables, CP_0 through CGL_0, appear on the variable sheet and we can assign input values to them there.

Status	Input	Name	Output	Unit	Comment
	220	CP_0		mg/kg	Plasma concentration
	22	CM_0		mg/kg	Muscle concentration
	220	CK_0		mg/kg	Kidney concentration
	220	CL_0		mg/kg	Liver concentration
	0	CG_0		mg/kg	Gut tissue concentration
	0	CGL_0		mg/kg	Gut lumen concentration

It is simple enough to create TK list functions relating the time (list t) with each of the solution matrix lists, using cubic interpolation. However, it is better to interpolate these values on a log scale as shown by the plot. So, we create a procedure function to loop through the solution matrix and generate the logs of the lists to be used in the list functions. Here is the procedure function code. Note that we loop from the 2nd element, avoiding a log(0) error condition.

Statement
call blankm('LCP,'LCM,'LCK,'LCL,'LCG,'LCGL
n = length('t)
for i = 2 to n
'LCP[i] = log('CP[i])
'LCM[i] = log('CM[i])
'LCK[i] = log('CK[i])
'LCL[i] = log('CL[i])
'LCG[i] = log('CG[i])
'LCGL[i] = log('CGL[i])
next i

This function requires no inputs or outputs and is called from the rule sheet. To assure that it is not called until the solution matrix has been generated, we can check the length first. We insert a

simple rule at the top of the rule sheet – Call Blank(‘t) – to blank the list. Then, after the call to the integrator, we insert a rule to verify that the t list has been filled.

tcheck = (length(‘t)>0)

Then we can call our logger procedure…

If tcheck then call logger()

Then we can create and call the list functions relating t with list LCP through LCGL. The interpolations will be done using the logs of the original values, so we must reverse the log transformation when evaluating the list functions. For example,

CP = 10^CP(t)

Function CP is the list function mapping t with the log(CP) values and interpolating if necessary. The result of the CP function is the log of the interpolated CP value. This equation then returns the CP value.

Again, to avoid processing the list functions until the integrator has completed its work, we bundle them in a function C and call C conditional on tcheck being true. We also make sure that the sample t value is less than the maximum t value.

Here is the new function C with calls to the list functions. This function has a single input t and six outputs, CP through CGL.

Rule
if t<'t[2] then call errmsg("The sample time, t, is less than the first time step. Interpolation is not poss
CP = 10^CP(t)
CM = 10^CM(t)
CK = 10^CK(t)
CL = 10^CL(t)
CG = 10^CG(t)
CGL = 10^CGL(t)

Here is the rule calling the C function.

Rule
if and(tcheck,t<=maxtime) then call C(t;CP,CM,CK,CL,CG,CGL

The model is now set to solve, returning a solution at any time we choose in addition to the times presented in the solution matrix.

Here is a sample solution on the variable sheet, assuming the mouse species parameters. At the bottom are the solutions at the time of 60 minutes.

Status	Input	Name	Output	Unit	Comment
					Volumes
	1	VP		ml	Plasma
	10	VM		ml	Muscle
	0.34	VK		ml	Kidney
	1.3	VL		ml	Liver
	1.5	VG		ml	Gut tissue
	1.5	VGL		ml	Gut lumen
					Flow Rates
	0.5	QM		ml/min	Muscle
	0.8	QK		ml/min	Kidney
	1.1	QL		ml/min	Liver
	0.9	QG		ml/min	Gut tissue
					Tissue/Plasma Equilibrium Ratios
	0.15	RM			Muscle
	3	RK			Kidney
	10	RL			Liver
	1	RG			Gut tissue
	0.2	kK		ml/min	Kidney clearance
	0.4	kLratio		ml/min	Bile secretion clearance
	2	tau		min	Bile secretion time constant
	0.01	kF		1/min	Gut lumen parameter
	0.2	kG		ug/ml	Gut lumen parameter
	6	KG		ug/ml	Gut lumen parameter
	0.001	b		ml/min	Gut lumen parameter
					Initial Conditions:
	240	maxtime		min	Max. time for solution
	220	CP_0		mg/kg	Plasma concentration
	22	CM_0		mg/kg	Muscle concentration
	220	CK_0		mg/kg	Kidney concentration
	220	CL_0		mg/kg	Liver concentration
	0	CG_0		mg/kg	Gut tissue concentration
	0	CGL_0		mg/kg	Gut lumen concentration
	60	t		min	Sample time
		CP	7.0493	mg/kg	Plasma concentration at t
		CM	1.1556	mg/kg	Muscle concentration at t
		CK	17.418	mg/kg	Kidney concentration at t
		CL	74.7617	mg/kg	Liver concentration at t
		CG	7.8989	mg/kg	Gut tissue concentration at t
		CGL	259.5113	mg/kg	Gut lumen concentration at t

The model can now be backsolved as desired. Suppose we know that the plasma concentration after one hour is actually 7.8 mg/kg and suspect that there may be a difference in the flow rate to the kidney causing the change. We input CP as 7.8, try a guess of 0.1 for QK and solve. After a few iterations, TK reports QK as 3.4539E-1; less than half of the original 0.8 value.

A species table is made by entering the values from Table 1. Then if each of the parameter variables on the variable sheet is associated with those lists, we can use the Commands menu, selecting Get Values from Lists, to fill the variable sheet. Of course, those values come to the variable sheet as inputs but we can backsolve for them as necessary.