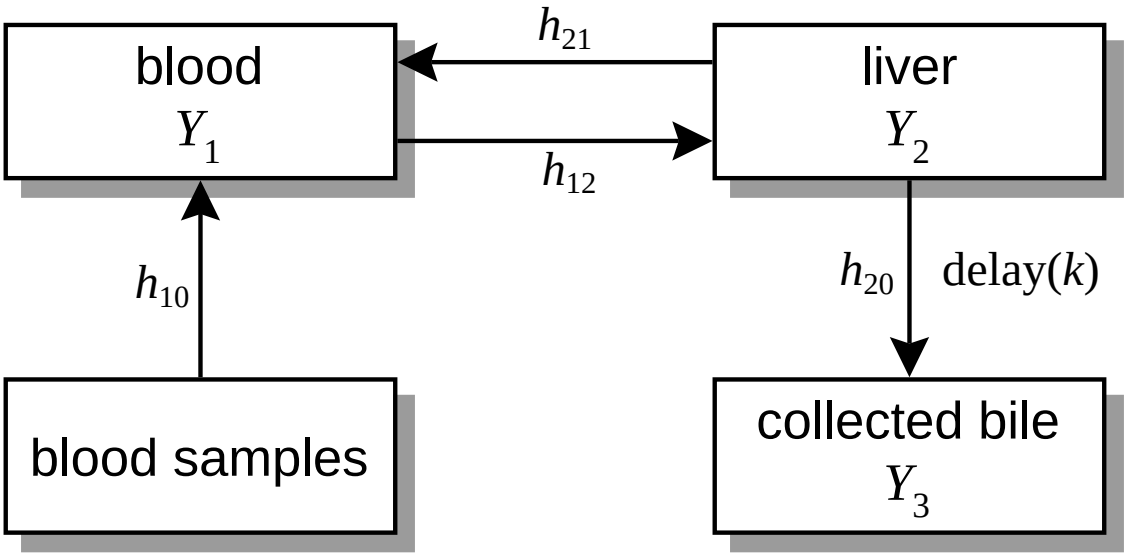
File: /General/MLAB-Text/Papers/delay/delay

Fitting a Pharmacological Model with Delay Using MLAB (data due to Ernest Feytmans)

Gary D. Knott, Ph.D.
Civilized Software, Inc.
12109 Heritage Park Circle
Silver Spring MD 20906 USA
Tel.: (301)-962-3711
email: csi@civilized.com
URL: www.civilized.com

This example shows how to formulate and fit a compartmental model. It also shows how MLAB can handle a delay term in a differential equation. Finally, the EWT function (estimated weights) is exemplified.

At time 0, a dose of radioactive-labeled taurocholate, Y, is injected in a rat. This material, Y, is passed through the liver and enters the bile. The bile is collected from the bile duct and the cumulative amount of Y present is measured at various times over a period of 1202 seconds. The level of Y seen in the bile at time t is actually the amount we measure in the output of a catheter at time $t + k$, since a delay of time k is needed for the bile to travel through the catheter. Also each blood level measurement involves removing a small sample of blood for analysis. Thus a simple model for the incorporation of Y in the bile which includes the effects of measurement takes the following compartmental form.



$Y1(t)$ is the concentration of Y in the blood at time t. $Y2(t)$ is the amount of Y in the liver at time t. $Y3(t)$ is the cumulative amount of Y in the collected bile at time t. V is the volume of blood in milliliters. U is the volume of the liver compartment in milliliters. k is the time-delay of the flow of bile from the liver into the bile-collection vial. We have:

$$\begin{aligned} Y_1'(t) &= (h_{21}Y_2/U - (h_{10}+h_{12})Y_1)/V, \\ Y_2'(t) &= h_{12}Y_1 - (h_{20}+h_{21})Y_2/U, \\ Y_3'(t) &= h_{20}Y_2(t-k)/U, \end{aligned}$$

with $Y_1(0) = 100/V$, $Y_2(0) = 0$, and $Y_3(0) = 0$.

The initial value $100/V$ for $Y_1(0)$ denotes the concentration of 100 percent of the dose of Y, and all measurements are given in terms of percent of the initial dose. Y_1 is in units of percent-of-dose per V milliliters and Y_2 and Y_3 are in percent-of-dose units.

Let $k_{21} = h_{21}/U$, $k_{12} = h_{12}/V$, $k_{20} = h_{20}/U$, and $k_{10} = h_{10}/V$. Then we have

$$\begin{aligned} Y_1'(t) &= k_{21}Y_2/V - (k_{10}+k_{12})Y_1, \\ Y_2'(t) &= k_{12}V \cdot Y_1 - (k_{20}+k_{21})Y_2, \\ Y_3'(t) &= k_{20}Y_2(t-k), \end{aligned}$$

with $Y_1(0) = 100/V$, $Y_2(0) = 0$, and $Y_3(0) = 0$.

We may guess the following values (these are fairly good guesses, obtained by prior study with `MLAB`.)

$V \approx 15\,\mathrm{ml}$; $k \approx 1.7\,\mathrm{hsec}$; $k_{12} \approx 1.7\,\mathrm{hsec}^{-1}$; $k_{21} \approx .1\,\mathrm{hsec}^{-1}$; $k_{10} \approx .25\,\mathrm{hsec}^{-1}$; $k_{20} \approx .5\,\mathrm{hsec}^{-1}$;

The time units used are hectoseconds, denoted hsec; one hsec is 100 seconds. The data below is given in units of hectoseconds.

The blood data is in a text-file called $D1$, as follows.

time	Y_1
.2	3.972
.3	3.484
.4	2.928
.5	2.317
.6	1.988
1	.2 .481
1	.8 .321
2	.4 .217
3	.6 .146
4	.8 .086
6	.0 .066
7	.2 .047
9	.0 .033
12	.0 .047

The bile data is in a text-file called $D2$, as follows.

t	Y_3	t	Y_3	t	Y_3
.35	0	.60	0	.90	0
1.17	.004	1.44	.114	1.78	1.095
2.05	2.248	2.32	4.689	2.62	6.542
2.91	7.652	3.12	5.937	3.40	6.580
3.60	4.687	3.85	6.663	4.20	4.587
4.44	3.687	4.77	3.215	5.06	3.378
5.30	2.458	5.60	2.679	5.84	1.854
6.10	2.059	6.42	1.691	6.63	1.181
6.97	1.648	7.25	1.235	7.56	1.183
7.83	.955	8.12	.976	8.37	.706
8.60	.498	8.82	.665	9.13	.792
9.32	.395	9.55	.462	9.80	.470
10.00	.478	10.27	.407	10.60	.426
10.84	.285	11.15	.438	11.46	.396
11.74	.273	12.02	.318		

The bile data in $D2$ is the incremental percentage-of-dose amounts of Y found in drops of bile collected at the specified times. To obtain cumulative bile measurements we must compute the successive partial sums of the given values.

We wish to fit our model to the data by estimating values for k_{12}, k_{21}, k_{20}, k_{10}, V, and k.

We may run MLAB and proceed as follows. This example shows the use of MLAB interactively. In practice, we would construct a do-file and then execute it repetitively, probably with modifications; using a do-file is the most convenient way to do modeling with MLAB.

```
* M1 = READ(D1,200,2)
* M2 = READ(D2,200,2)
```

First we must compute the partial sums of the incremental bile data.

```
* FUNCTION PS(j) = (IF j=1 THEN 0 ELSE PS) + M2[j,2]
* M2 COL 2 = PS ON 1:NROWS(M2)
```

In general, the weight associated with a given observation should be the reciprocal of the variance of the random variable of which the observation is a sample. When using correct weights, we not only account for differing amounts of error, but also automatically compensate for differing scales which may be used in distinct sets of data being fit simultaneously by model functions with shared parameters. We can estimate the standard deviations, and hence the desired weight values, non-parametrically, using the EWT operator.

```
* W1 = EWT(M1)
* W2 = EWT(M2)
```

Now we normalize both $W1$ and $W2$ to sum to $1/2$, since we wish to give each curve the same total weight.

```
* W1 = W1/ROWSUM(W1)/2
* W2 = W2/ROWSUM(W2)/2
```

Now $M1$ and $M2$ are our data matrices for the blood and cummulative bile data respectively, and $W1$ and $W2$ are the associated weight vectors. The differential equations which define the model functions Y_1, Y_2, and Y_3 are given below. Note one differential equation contains a delay term.

```
* FUNCTION Y1'T(T) = K21*Y2/V - (K10 + K12)*Y1
* FUNCTION Y2'T(T) = K12*V*Y1 - (K20 + K21)*Y2
* FUNCTION Y3'T(T) = K20*Y2(T-K)
* INITIAL Y1(0)= 100/V
* INITIAL Y2(0)= 0
* INITIAL Y3(0)= 0
* V=15; K=1.7; K12=1.7; K21=.1; K10=.25; K20=.5
```

Since we are using MLAB interactively, it is wise to save our data and model in an MLAB save-file for later reuse.

```
* SAVE IN FEYT
```

At this point we may start fitting our model. However, first let us consider what the delay term implies. In MLAB, delay terms in differential equations are evaluated during the numerical solution process by looking back in the table of previously-computed results and obtaining a value for the delay term by interpolation. If the previously-generated results do not extend far enough into the past, the earliest available value is used! In order for this to be the initial value, it is necessary for the initial time to be present in the vector of time values at which we are maintaining results. It is also necessary that the time vector consist of sufficiently closely-spaced values so that the interpolation process is adequately accurate. In our case, our time values are reasonably closely-spaced, but, we must add the initial time value (with weight 0) as follows.

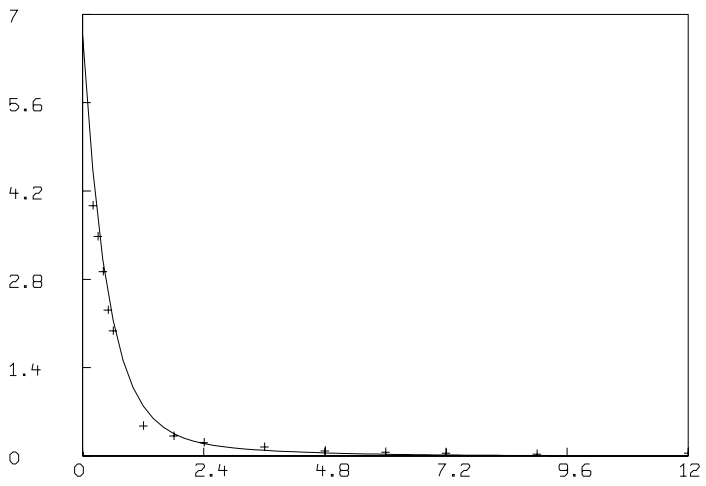
```
* M1 = 0&M1
* M2 = 0&M2
* W1 = 0&W1
* W2 = 0&W2
```

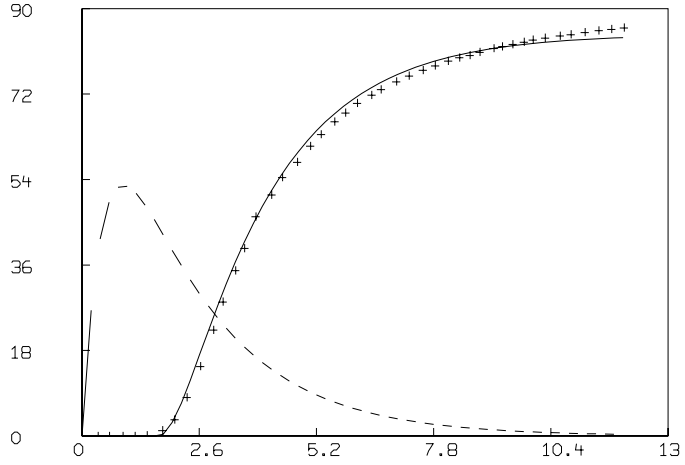
In MLAB, the delay expression $Y2(t-k)$ is effectively treated as $Y2$(if $t < k$ then 0 else $t-k$). Thus the delay enters gradually. We start with zero delay, and the delay increases until time $t = k$, whereupon the delay remains constant. This interpretation of a delay is forced by MLAB, but it is generally the appropriate way to handle delays, given initial conditions only.

Before fitting, it is a good idea to check the model and the quality of our guesses.

```
* R=INTEGRATE (Y1'T, Y2'T, Y3'T, 0:12:.2)
t = 1.7, errfac = 0.001, eqn # 3, truncation error = 111.111
tolerance will be increased in order to proceed with the deq solution.
t = 1.7, errfac = 0.01, eqn # 3, truncation error = 11.1111
tolerance will be increased in order to proceed with the deq solution.
solution of deq system forced past a possible singularity near 1.7.
accuracy may be lost
dy[1]: -0.486184
dy[2]: -16.1942
dy[3]: 0
* DRAW M1, LINE NONE PT "+"
* DRAW R COL 1:2
* VIEW
* DELETE W
* DRAW R COL (1,4) LINETYPE ALTERNATE
* DRAW M2, LINETYPE NONE POINTTYPE "o"
* DRAW R COL (1,6)
* VIEW
* DELETE W
```

These graphs are shown below. Note how the delay effects $Y3$; it remains zero even when $Y2$ is positive.





The tolerance violations which occurred during the integration above are harmless. What happened is that for the chosen value of k, the particular time points where the differential-equation-solver obtained solution values included a time, t_1, which was very close to k. When a step, s, from t_1 to $t_1 + s$ was attempted, the value of $Y2(t - k)$, and hence of $Y3't$, jumped from zero to a positive value. The change in a derivative value must be sufficiently gentle or the differential-equation-solver will reduce the step-size s until it is! But in this case, t_1 was so close to k that $t_1 + s$ could not achieve a value at which $Y3't$ was sufficiently close to zero without s being too small! The tolerance violation is complaining of this fact. The effect of a tolerance violation is to advance t and assume the current value of $Y1$, $Y2$, and $Y3$ hold at the new time. This introduces an error which we hope is small and whose effect dies out over time.

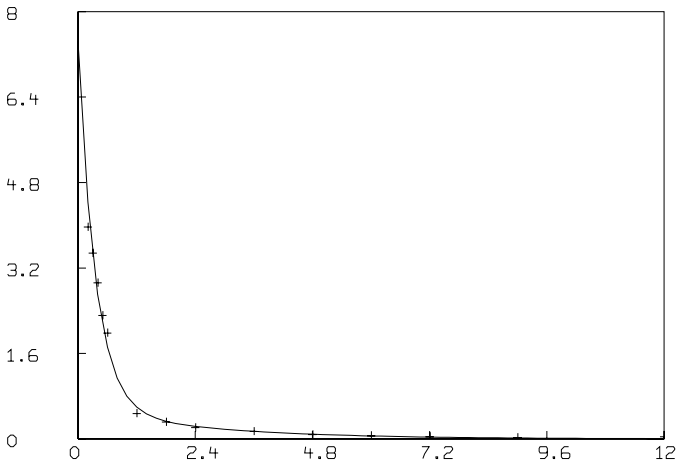
Now we may fit our data. In order to see the progress of the curve-fitting process, we set the `MLAB` control variable `lsqrpt` to 9. Note the parameter V occurs in the initial condition for $Y1$ as well as in the derivative functions. In order to avoid tolerance violation messages, we shall set the `MLAB` control variable disastersw to -2.

```
* lsqrpt = 9
* disastersw = -2
* FIT (K12, K21, K20, K10, K, V), Y1 TO M1 WITH WEIGHT W1,Y3 TO M2 WITH WEIGHT W2
Begin iteration 1 bestsosq=6.06291e-01
```

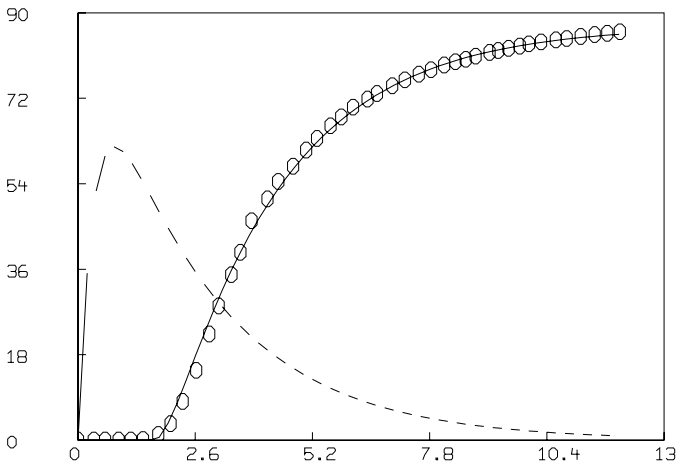
```
Begin iteration 2 bestsosq=3.37174e-01
Begin iteration 3 bestsosq=2.80705e-01
Begin iteration 4 bestsosq=6.13384e-02
Begin iteration 5 bestsosq=6.01915e-02
Begin iteration 6 bestsosq=6.01306e-02
Begin iteration 7 bestsosq=6.00476e-02
Begin iteration 8 bestsosq=5.95795e-02
Begin iteration 9 bestsosq=5.94568e-02
Begin iteration 10 bestsosq=5.91193e-02
Begin iteration 11 bestsosq=5.41983e-02
Begin iteration 12 bestsosq=5.39791e-02
Begin iteration 13 bestsosq=5.31954e-02
final parameter values
      value              error               dependency    parameter
    2.404830506        0.5089934386          0.9996545161   K12
   0.1788429902       0.03364502419          0.9984644263   K21
   0.4046451617       0.01567822569           0.995752869   K20
   0.2352243999         0.050102745          0.9988267993   K10
    1.667994505       0.06660192075          0.9643014847   K
    13.44151307         2.411338814           0.855048444   V
13 iterations
CONVERGED
best weighted sum of squares = 5.31954e-02
weighted root mean square error = 3.13863e-02
weighted deviation fraction = 2.74048e-03
R squared = 9.98708e-01
```

We may observe the result below.

```
* R = INTEGRATE(Y1'T,Y2'T,Y3'T,0:12:.2)
* DRAW M1,LINETYPE 0,POINTTYPE "+"
* DRAW R COL 1:2; VIEW
```




```
* DELETE W
* DRAW M2,LINETYPE NONE POINTTYPE "o"
* DRAW R COL (1,4),LINETYPE ALTERNATE
* DRAW R COL (1,6); VIEW;
```




```
* DELETE W
```

This is a good fit. In this case we did not need to impose constraints to keep the parameter values within reasonable limits, but such constraints are often necessary.

We can get a slightly better fit by using the explicit formulas for $Y1$ and $Y3$, which are obtainable in this case since $Y1$ and $Y2$ do not depend on $Y3$.

The explicit solution functions are given below. It is important to use the gradual delay expression (if $t < k$ then t else k) in solving the differential equations. Otherwise the solutions will not correspond to the results obtained by numerical integration.

$$\begin{aligned}
Y_1(t) &= A_1 \exp(-A_2 t) + B_1 exp(-B_2 t), \\
Y_2(t) &= A_3[\exp(-A_2 t) - \exp(-B_2 t)], \quad \text{and} \\
Y_3(t) &= \text{if } t < k \text{ then } 0 \text{ else } A_4[1 - \exp(-A_2(t-k))] + A_5[1 - \exp(-B_2(t-k))],
\end{aligned}$$

where

$$\begin{aligned}
A_2 &= (S+Q)/2, \\
B_2 &= (S-Q)/2, \\
S &= K_{12} + K_{21} + K_{20} + K_{10}, \\
Q &= [S_2 - 4(K_{12}K_{20} + K_{10}K_{21} + K_{10}K_{20})]^{1/2}, \\
A_1 &= (100/V)[A_2 - K_{21} - K_{20}]/(A_2 - B_2), \\
B_1 &= (100/V)[K_{21} + K_{20} - B_2]/(A_2 - B_2), \\
A_3 &= 100K_{20}K_{12}/(B_2 - A_2), \\
A_4 &= 100K_{20}K_{12}/[A_2(B_2 - A_2)], \quad \text{and} \\
A_5 &= 100K_{20}K_{12}/[B_2(A_2 - B_2)].
\end{aligned}$$

Now, an important point arises. How are these functions to be defined in `MLAB`? The straightforward process of substituting to eliminate the auxilary variables A_1, A_2, A_3, A_4, A_5, B_1, and B_2 results in huge formulas, which, when defined in `MLAB` would cause the required symbolic derivative functions $Y1'K_{12}$, $Y1'K_{21}$, ..., etc. to be so large they might not all fit in memory!

One approach to representing these functions is to make A_1, A_2, A_3, A_4, A_5, B_1, B_2, S, and Q functions of no arguments which have the appropriate parameters merely by virtue of their appearence in the definitions. Thus we could type:

```
FUNCTION Y1(T) = A1()*EXP(-A2()*T)+B1()*EXP(-B2()*T)
```

```
FUNCTION A1() = (100/V)*(A2()-K12-K20)/(A2()-B2())
FUNCTION A2() = (S() + Q())/2
FUNCTION S() = K12 + K21 + K20 + K10 ..., etc.
```

This process results in functions which waste much time computing the same values many times during one invocation. For example computing $Y1(2)$ involves computing $A1()$, $A2()$, $B1()$, and $B2()$; but computing $A1()$ also involves computing $A2()$ and $B2()$ (again!) and so on.

A better approach is to discover common sub-expressions, and when they occur, make them actual arguments to be passed to a function which returns the value of the formula in which they occur. This process results in the following formulation for defining our model. This example merits careful study since the principle involved should usually be employed in using MLAB!

```
* DELETE Y1, Y2, Y3, Y1'T, Y2'T, Y3'T
* FUNCTION F3(J,T,A2,B2) = (100/(A2-B2))*(IF J=1 THEN \
          ((A2-K21-K20)*EXP(A2*T)+(K21+K20-B2)*EXP(B2*T))/V \
          ELSE K20*K12*(IF J=2 THEN (EXP(B2*T)-EXP(A2*T)) ELSE \
          ((1-EXP(B2*T))/B2-(1-EXP(A2*T))/A2)))
* FUNCTION F2(J,T,S,Q) = F3(J, -T, .5*(S+Q), .5*(S-Q))
* FUNCTION F1(J,T,S) = F2(J,T,S,SQRT(S*S-4*(K12*K20+K10*K21+K10*K20)))
* FUNCTION Y1(T) = F1(1, T, K12+K21+K10+K20)
* FUNCTION Y2(T) = F1(2, T, K12+K21+K10+K20)
* FUNCTION Y3(T) = IF T<K THEN 0 ELSE F1(3, T-K, K12+K21+K10+K20)
```

Now given these functions, you may see the derivative forms which result by typing them out; although they are very large, they are easily handled within MLAB. For example:

```
* TYPE Y1'K12
FUNCTION Y1'K12(T) = F1'K12(1,T,K12+K21+K10+K20)+ \
      F1'S(1,T,K12+K21+K10+K20)
* TYPE F1'S
FUNCTION F1'S(J,T,S) = F2' S(J,T,S, \
       SQRT(S*S-4*(K12*K20+K10*K21+K10*K20))) \
       +F2' Q(J,T,S,SQRT(S*S-4*(K12*K20+K10*K21+K10*K20))) \
       *((S+S)/(2*SQRT(S*S-4*(K12*K20+K10*K21+K10*K20))))
* TYPE F2'S
 FUNCTION F2'S(J,T,S,Q) = F3'A2(J,-T,.5*(S+Q),.5*(S-Q))*.5 \
        +F3'B2(J,-T,.5*(S+Q),.5*(S-Q))*.5
```

```
* TYPE F3'A2
 FUNCTION F3 DIFF A2(J,T,A2,B2) = (100/(A2-B2))*( IF J=1 \
       THEN (((A2-K21)-K20)*T*EXP(A2*T)+EXP(A2*T))/V \
       ELSE K20*K12*( IF J=2 THEN -T*EXP(A2*T)  \
       ELSE -((-T*EXP(A2*T))/A2+(1-EXP(A2*T))*(-A2^(-2))))  \
       +( IF J=1 THEN (((A2-K21)-K20)*EXP(A2*T)+((K21+K20)-B2)*EXP(B2*T))/V \
       ELSE K20*K12*( IF J=2 THEN EXP(B2*T)-EXP(A2*T) \
       ELSE (1-EXP(B2*T))/B2-(1-EXP(A2*T))/A2))*100*(-(A2-B2)^(-2))
```

Now to do our fit, we may use the following MLAB fit-statement.

```
FIT(K12, K21, K20, K10, K), Y1 TO M1 WITH WEIGHT W1,Y3 TO M2 WITH WEIGHT W2
Begin iteration 1 bestsosq=1.03081e-01
Begin iteration 2 bestsosq=6.66487e-02
Begin iteration 3 bestsosq=6.45033e-02
Begin iteration 4 bestsosq=6.39085e-02
final parameter values
      value              error               dependency    parameter
    2.486325129        0.2919415767          0.9986809093   K12
   0.2181076867         0.175390433          0.9997631512   K21
   0.4060674476       0.01517223395          0.9949641321   K20
   0.2362370486       0.04912624607          0.9993555185   K10
    1.636370209       0.05358085176          0.9369202287   K
4 iterations
CONVERGED
best weighted sum of squares = 6.38532e-02
weighted root mean square error = 3.40730e-02
weighted deviation fraction = 2.94842e-03
R squared = 9.98282e-01
```

We can do even better if we account for the laminar flow in the catheter which is sampling the bile. If this tube is of length L with interior diameter $2R$, then

$$Y_3(t) = \text{if } t = 0 \text{ then } 0 \text{ else } (1/t)\int_0^t g(u)\,du,$$

where $g(u)$ is the concentration of labeled taurocholate entering compartment Y_3 at time u. This is just an average of earlier concentrations which have entered the catheter; the following function g expresses the delayed laminar flow and we have:

$$g(u) = \int_0^R Y_2(u - L/(a(R-x)^2)) \cdot (2x/R^2)\,dx,$$

where $Y2(t) = 0$ for $t < 0$, and a is defined by the equation

$$t \int_0^R a(R-z)^2 2\pi x \, dx - \pi R^2 L = V_t,$$

where V_t is the total volume of fluid accumulated in t seconds of flow through the catheter, with to be completely filled.

We can thus reformulate our model using the MLAB integral operator:

```
fct Y3(T) = if T=0 then 0 else \
     INTEGRAL(U,0,T,INTEGRAL(X,0,R,Y2(U-L/(A*(R-X)^2))*(2*X/R^2)))/T
```

where R and L are known values, and A is computed initially (before fitting) using a function which employs the MLAB ROOT operator.