PARAMETER ESTIMATION IN BIOLOGICAL CELL CYCLE MODELS USING DETERMINISTIC OPTIMIZATION

by

Jason W. Zwolak

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science

APPROVED:

Layne T. Watson, co-chair	John J. Tyson, co-chair

Lenwood S. Heath

October 31, 2001
Blacksburg, Virginia

Key words: Cell Cycle Models, Ordinary Differential Equations, Parameter Estimation.

PARAMETER ESTIMATION IN BIOLOGICAL CELL CYCLE MODELS USING DETERMINISTIC OPTIMIZATION

by

Jason W. Zwolak

(ABSTRACT)

Cell cycle models used in biology can be very complex. These models have parameters with initially unknown values. The values of the parameters vastly affect the accuracy of the models in representing real biological cells. Typically people search for the best parameters to these models using computers only as tools to run simulations. In this thesis methods and results are described for a computer program that searches for parameters to a series of related models using well tested algorithms. The code for this program uses ODRPACK for parameter estimation and LSODAR to solve the differential equations that comprise the model.

ACKNOWLEDGMENTS

I would like to thank Dr. Watson and Dr. Tyson for all the time they have spent helping me progress with the research presented in this thesis. Thank you Dr. Tyson for letting me work in your lab, which proved to be an excellent working environment. Thank you Dr. Heath for being on my advisory committee.

Thanks to Laurence Calzone, Andrea Ciliberto, and Kathy Chen for all there support and help as coworkers and friends.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1: INTRODUCTION

Computational models of cell growth and division involve digital representation of a complex network of biochemical reactions within cells. These reactions can be described by a system of nonlinear ordinary differential equations, according to the principles of biochemical kinetics. Rate constants and binding constants enter as parameters in the differential equations, and must be estimated by fitting solutions of the equations to experimental data.

This work concerns some classical experiments on activation of MPF (M-phase promoting factor) in frog egg extracts. MPF is a dimer of cyclin and Cdc2 (a protein kinase that drives egg nuclei into mitosis). In the experimental preparation, a fixed amount of cyclin is added to an extract containing an excess of Cdc2 subunits. If the amount of cyclin added is below a threshold, MPF activity never appears. Above the threshold, MPF is activated but only after a characteristic time lag. The time lag decreases abruptly as total-cyclin-added increases above the threshold. The goal is to fit this data with a reasonable model of the underlying biochemistry, which keeps track of cyclin monomers, Cdc2 monomers, and the phosphorylation state of cyclin/Cdc2 dimers.

ODRPACK, based on the orthogonal distance between experimental data and the model, is used for the nonlinear regression to estimate the unknown rate constants (ODE parameters). The ability of this algorithm to arbitrarily weight data values, and to treat both the abscissa and ordinates as uncertain, is crucial, given the sparsity and uncertainty of available biological data. Constructing the model function values requires simulating MPF activity as a function of time after addition of cyclin. These simulations yield the cyclin threshold for MPF activation, and the time lag (the time necessary for MPF activity to reach one-half of its asymptotic value, for supra-threshold amounts of cyclin added to the extracts).

The complete calculation is expensive, because the ODE's are stiff, and must be solved numerous times for the nonlinear regression. Also, because of local minima, the nonlinear regression must be done from many starting points to adequately explore the parameter space. There are potential sources for parallelism in the ODE solution itself, the estimation of partial derivatives of the ODE solution, and multiple starting points for regression. Numerical results are presented for a relatively simple two-component three-parameter model, as well as a more complex four-component ten-parameter model.

To study realistic models of cell cycle control, more components must be added to the model, and other measurable phenomena incorporated in the cost function. As the modeling fidelity is increased, the mathematical and computational complexities of the problem grow rapidly. Efficient and accurate tools for parameter estimation will be needed to build computational models of the complex control networks operating within cells, which is one of the main goals of bioinformatics in the postgenomic era.

Chapter 2 outlines the initial biological model and provides the experimental data for said model. An overview of the code along with descriptions of the tools (ODRPACK and LSODAR) used by the code can be found in Chapter 3. Chapter 4 contains a more detailed pseudocode for the algorithm. The initial results of the parameter estimation are in Chapter 5. Experimental data relating to thresholds are integrated in Chapter 6. Modifications to the algorithm from Chapter 4 are described in Chapter 7. The results after integrating the threshold data are in Chapter 8. A more accurate ten parameter model is described in Chapter 9. The conclusion and future work are in Chapter 10.

Chapter 2: PROBLEM STATEMENT

The differential equation describing the concentration and rate of change in active MPF in a frog egg with a fixed concentration of total cyclin is

$$\frac{dM}{dt} = -k_{wee}M + (k_{25} + k'_{25}M^2)(C - M),$$

where k_{wee}, k_{25}, and k'_{25} are rate constants, C is concentration of total cyclin, and M is the concentration of active MPF versus time [Tyson et al, 1995].

In cells, MPF is the primary protein that determines when the cell divides. However, MPF does not promote cell division unless MPF is active. Other proteins in the cell inhibit or promote MPF activation. The most dominant proteins in this model are Cdc25 and Wee1. Wee1 inactivates MPF and the rate constant k_{wee} represents Wee1's affect on MPF inactivation. Cdc25 activates MPF and the rate constants k_{25} and k'_{25} represent Cdc25's affect on MPF activation [Tyson et al, 1995].

The first part of this thesis describes the methods and results of estimating the rate constants k_{wee}, k_{25}, and k'_{25}. Estimating k_{wee}, k_{25}, and k'_{25} requires experimental data related to the ODE. The available data is in Table 1.

Table 1. Experimental data [Moore, 1997]

Total Cyclin	Time Lag (min)
0.15	55
0.20	45
0.25	40
0.30	30
0.50	20

The time lag appears in the ODE as the point where the active MPF concentration is half its asymptotic value. For low concentrations of total cyclin, MPF never activates (i.e., the concentration of active MPF never rises above half the concentration of total cyclin). For higher concentrations of total cyclin MPF activates after some time lag. The higher the concentration of total cyclin the smaller the time lag. This behavior can be seen in the experimental data in Table 1 and the plots in Fig. 1.

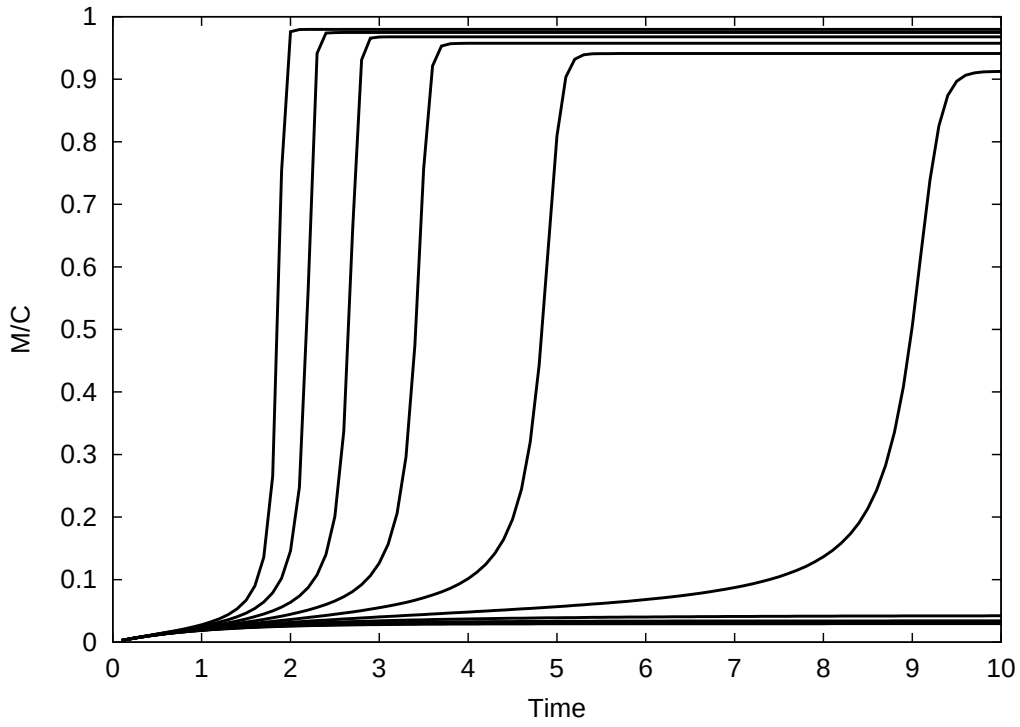

FIG. 1. Percent total cyclin in active MPF M/C versus time t for multiple concentrations of total cyclin

Chapter 3: METHODS

The variables that correspond to the data in Table 1 are not all present in the ODE (total cyclin is in the ODE, time lag is not in the ODE). The first step is to use the ODE to calculate another function f in terms of the variables corresponding to the data in Table 1.

Let $f(x)$ be the time lag for total cyclin x, where time lag is the time for MPF to activate or inactivate (depending on whether MPF was initially inactive or active, respectively). More precisely the time lag is the time where the active MPF concentration is the average of the initial concentration of active MPF and the asymptotic concentration of active MPF.

LSODAR is used to solve the ODE and to find the time lag from the solution to the ODE. The time lag versus total cyclin function $f(x)$ is also a function of the rate constants in the ODE. This function is used by ODRPACK to find the rate constants giving the curve $f(x)$ that best fits the experimental data in Table 1.

3.1 LSODAR

LSODAR is a variant of LSODE [[Radhakrishnan et al, 1993], [Hindmarsh, 1980], [Hindmarsh, 1983]] that automatically switches between stiff and non-stiff methods and has a root finder. LSODAR starts with a non-stiff method and switches to a stiff method if necessary. LSODAR's root finder is used in this application to find the time lag for MPF activation.

For non-stiff problems LSODAR uses Adams-Moulton (AM) of orders 1 to 12. For stiff problems LSODAR uses backward differentiation formulas (BDF) of orders 1 to 5. With both methods LSODAR varies the step size and order. LSODAR switches from AM to BDF when AM is no longer stable for the problem or cannot meet the accuracy requirements efficiently [Petzold, 1983].

The present problem uses LSODAR to solve for $M(t)$ (the concentration of active MPF with respect to time). The tolerances are set to 10^{-12} for both relative and absolute error. A tolerance of 10^{-10} is used when calculating a root for a function of the form

$$M(t) - M_{root},$$

where M_{root} is the value of the function $M(t)$ for which a time, t, is desired.

LSODAR takes, as an argument, a user written function, GEX, that evaluates equations based on the variables involved in the ODE that LSODAR is solving. For this problem GEX evaluates $M - M_{root}$. GEX returns evaluations of its equations to LSODAR and LSODAR looks for roots for those equations. When a sign change is detected LSODAR has bracketed a root and begins an algorithm based on the ROOT function described below. After each iteration of ROOT, LSODAR must evaluate a point on the solution curve of the ODE as requested by ROOT. Each evaluation involves interpolating the ODE solution $M(t)$. This interpolation formula is defined as part of the AM [Shampine et al, 1975] or BDF [Gear, 1971] method (depending on which is currently being used by LSODAR).

3.2 ODRPACK

ODRPACK is used to estimate the rate constants that fit time lag versus total cyclin to the experimental data in Table 1. ODRPACK finds an estimate for the rate constants by minimizing the weighted orthogonal distance between the experimental data and the calculated curve.

The present problem explicitly relates time lag to the total concentration of cyclin in the cell. Precisely,

$$y = f(x; \beta),$$

where y is time lag, x is total cyclin, and β is a vector of the rate constants. ODRPACK takes an equation of this form and experimental data for x and y to minimize

$$E = \min_{\beta,\delta,\epsilon} \left(\sum_{i=1}^{n} w_{\epsilon_i} \epsilon_i^2 + w_{\delta_i} \delta_i^2 \right),$$

where n is the number of experimental data points, ϵ_i is the error in the dependent variable y for point i, δ_i is the error in the explanatory variable x for point i, and w_{ϵ_i} and w_{δ_i} are the weights for ϵ_i and δ_i, respectively. E will be referred to as the weighted sum of squares. β, δ, and ϵ are subject to the constraints

$$y_i = f(x_i + \delta_i; \beta) - \epsilon_i,$$

where $i = 1, \ldots, n$ indexes the experimental data points.

ODRPACK actually minimizes a more general objective function

$$E = \min_{\beta,\delta,\epsilon} \left(\sum_{i=1}^{n} \epsilon_i^T w_{\epsilon_i} \epsilon_i + \delta_i^T w_{\delta_i} \delta_i \right),$$

where ϵ_i and δ_i are vectors for the errors in the dependent variable and errors in the explanatory variable, respectively. w_{ϵ_i} and w_{δ_i} are matrices of weights for ϵ_i and δ_i, respectively [[Boggs et al, 1992], [Boggs et al, 1989]]. Note that x and y, from the previous description of ODRPACK, are vectors and the function f is a vector-valued function in the general case. The present problem can be thought of as using the scalar version of ODRPACK, since the present problem has w_{ϵ_i} and w_{δ_i} as matrices of one element and ϵ_i and δ_i as vectors of one element.

The function $f(x + \delta; \beta)$ is implemented in FORTRAN and used by ODRPACK. Constraints are put on β by setting a flag (when β is invalid) before returning from the user supplied function. This is used to prevent the rate constants from becoming negative, which does not make sense biologically.

ODRPACK uses a trust region Levenberg-Marquardt method with scaling to minimize the objective function [Boggs et al, 1992]. In doing so ODRPACK needs to calculate the Jacobian matrices for β and δ. ODRPACK can calculate the Jacobian matrices by finite differences or by a user supplied routine. Finite differences were used here.

3.3 ROOT

ROOT is based on ZEROIN [Shampine et al, 1973], which is in turn based on code by Dekker [Dekker, 1969]. ROOT uses a combination of the secant and bisection methods where the secant method is used by default. ROOT has two working approximations of the root: A and B. The approximations always satisfy the constraint

$$g(A) * g(B) \leq 0,$$

where $g(t) = M - M_{root}$ and t is time in this problem (note that M is dependent on t). Furthermore, A is the better approximation of the root of $g(t)$. A is replaced in each iteration by a better approximation and B remains the same or changes to the old A, whichever satisfies the above equation. ROOT switches to the bisection method under two circumstances: when the secant method is converging too slowly, or when a large error is introduced because of limitations in machine precision. Notice the bisection method will not suffer from large error because it computes

$$\frac{A+B}{2}$$

for each iteration.

The initial approximations, for A and B, come from LSODAR's evaluation of GEX before and after LSODAR noticed a sign change. ROOT then requests values for g at new times until the approximation for the root of g is within the requested relative and absolute error.

Chapter 4: ALGORITHM

In this chapter the algorithm used is described in some detail using pseudocode. Many of the function arguments used with ODRPACK's subroutine DODRC and ODEPACK's subroutine LSODAR do not appear in the pseudocode. Most of these arguments were set to default values, and others are not relevant to understanding the methods used to solve the present problem.

The main program sets up the input for DODRC and is as follows:

begin

$s := 8$; s is the number of significant digits in the response variable of f. The significant digits here come from the reliability of the computations done by the computer, not by experiments.

$n := 5$; n is the number of experimental data points.

$x := \big(0.15, 0.20, 0.25, 0.30, 0.50\big)$; the vector x contains the total cyclin components of the experimental data.

$y := \big(55, 45, 40, 30, 20\big)$; the vector y contains the time lags corresponding to total cyclin concentrations from above.

$w_\delta := \big(44.44, 25, 16, 11.11, 4\big)$; w_δ contains the weights for the errors in x.

$w_\epsilon := \big(3.305 \cdot 10^{-4}, 4.938 \cdot 10^{-4}, 6.25 \cdot 10^{-4}, 1.111 \cdot 10^{-3}, 2.4 \cdot 10^{-3}\big)$; w_ϵ contains the weights for the errors in y. The weights are the squared reciprocals of the corresponding data values, which makes all the errors in the objective function relative instead of absolute. w_δ uses the same squared reciprocals for the weights.

$\beta := \big(0.5, 0.06, 80\big)$; β contains the initial guess for the rate constants. After DODRC has been called β will contain ODRPACK's best estimate for β given the arguments to DODRC. The parameters in β are k_{wee}, k_{25}, and k'_{25}, respectively.

DODRC(FCN, $n, s, x, y, w_\delta, w_\epsilon, \beta, \ldots$); the ODRPACK subroutine used is DODRC. FCN is defined below.

end

The function FCN takes concentrations of total cyclin and parameters to the ODE and returns time lags for each concentration of total cyclin. ODRPACK does not give FCN the total cyclin concentrations from the experimental data. Instead, ODRPACK gives FCN the total cyclin concentrations plus some error δ. In most cases the time lags returned by FCN will not match the time lags from the experimental data. Errors in measurements in the experimental data contribute to this mismatch. ODRPACK handles this by labeling the output of FCN as $y + \epsilon$. Precisely, let $X = x + \delta$ and $Y = y + \epsilon$. FCN takes arguments β and X and returns Y. The code for FCN follows.

subroutine FCN
  for $i := 1$ **step** 1 **until** n **do**
    $C := X(i)$; (set the total cyclin value)
    $T := 0$; (the initial time)
    $R_{tol} := 10^{-12}$; (relative error tolerance)
    $A_{tol} := 10^{-12}$; (absolute error tolerance)
    $M_{init} := 0$; (initial MPF concentration)
    $T_{out} := 1440$; (solve for the MPF concentration at this time)
    $N_g := 0$; (no roots are desired from LSODAR)
    M_{inf} :=LSODAR(FEX, M_{init}, T, T_{out}, R_{tol}, A_{tol}, N_g, JEX, GEX, ...);
    if $M_{inf} < C/2$ **then**
      $Y(i) := 1440$; (pseudo-infinite-lag)
      cycle;
    endif
    $M_{root} := M_{inf}/2$; (find a root at $M_{inf}/2$)
    $N_g := 1$; (one root is desired from LSODAR)
    LSODAR(FEX, M_{init}, T, T_{out}, R_{tol}, A_{tol}, N_g, JEX, GEX, ...);
    $Y(i) := T_{out}$; (the root is returned in T_{out})
  enddo
end subroutine FCN

The number 1440, construed as a pseudo-infinite-lag, is used to put a limit on how long to search for MPF activation. Effectively, the (computed) curve in Fig. 2 will be flat when it reaches 1440 minutes. The true physical curve continues to increase after 1440 minutes. This modification creates a curve that does not precisely match the actual curve, but this modification does not affect the computation. All the experimental data is well below 1440 minutes (1 day). ODRPACK looks for the point on the calculated curve that is closest to the experimental data when calculating the error. Since the initial guess is not closer to the horizontal line at 1440 minutes than to the real curve, the flat portion will not cause ODRPACK to make wrong estimates for the rate constants.

Subroutine FEX solves for the change in MPF concentration given MPF concentration, time, values for the parameters, and total cyclin concentration. Note that time does not appear directly in the ODE, but M is dependent on time. FEX is used by LSODAR when computing M numerically. FEX takes the MPF concentration M and returns the derivative M_t of MPF concentration with respect to time. JEX computes the partial derivative P of the ODE with respect to the dependent variable M, and takes the same arguments as FEX. LSODAR returns a root for the function g evaluated in GEX. GEX takes the same arguments as FEX. Pseudocode for FEX, JEX, and GEX follows.

subroutine FEX
  $M_t := -\beta_1 M + (\beta_2 + \beta_3 M^2)(C - M)$;
end subroutine FEX

subroutine JEX

$P_{1,1} := -\beta_1 - \beta_2 + 2\beta_3 CM - 3\beta_3 M^2$;

end subroutine JEX

subroutine GEX

$g_1 := M - M_{root}$; M_{root} is set elsewhere to a desired value of the solution M to the ODE defined in FEX.

end subroutine GEX

Chapter 5: INITIAL RESULTS

The best results using the methods and algorithms just described are in Table 2. These results were obtained using the experimental data in Table 1 and Fig. 2.

Table 2. Optimal rate constants from ODRPACK

Rate Constant	Optimal Value
k_{wee}	$1.117 \cdot 10^{-10}$
k_{25}	$3.277 \cdot 10^{-3}$
k'_{25}	$8.719 \cdot 10^{0}$

In Fig. 2 the fitted curve was generated from the rate constants in Table 2. The fit appears good and the weighted sum of squares of the δ's and ϵ's are $3.039 \cdot 10^{-03}$ and $1.810 \cdot 10^{-02}$, respectively. So indeed the curve fits the given data well. However, currently the code used does not take into account the thresholds for MPF activation and inactivation. This curve has a threshold for MPF activation around or below 0.03. Experiments estimate the threshold to be about 0.18. The next step for this problem is to integrate the empirical thresholds for MPF activation and inactivation into the code.

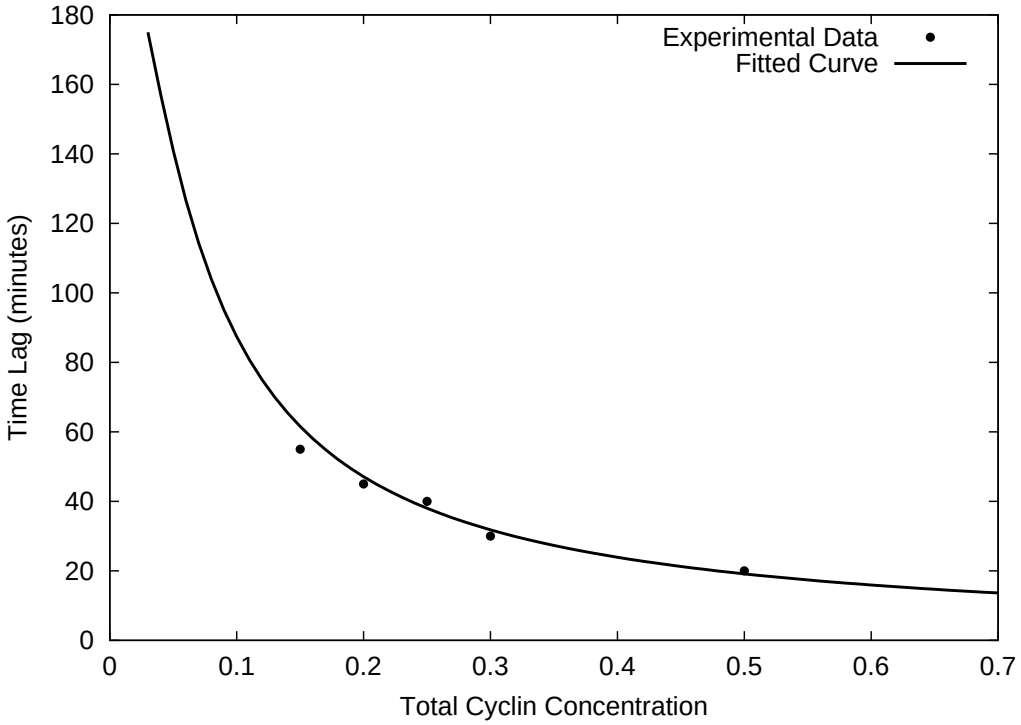

FIG. 2. Time lag for MPF activation versus total cyclin

Chapter 6: INTEGRATING THE THRESHOLDS

There are two thresholds that must be fit for this model. The threshold for MPF activation and the threshold for MPF inactivation. Given that MPF is initially inactive it will remain inactive as long as the total cyclin concentration is below a threshold, the threshold for MPF activation. Similarly, given that MPF is initially active it will remain active as long as the total cyclin concentration is above a threshold, the threshold for MPF inactivation. These thresholds can be seen best in Fig. 3.

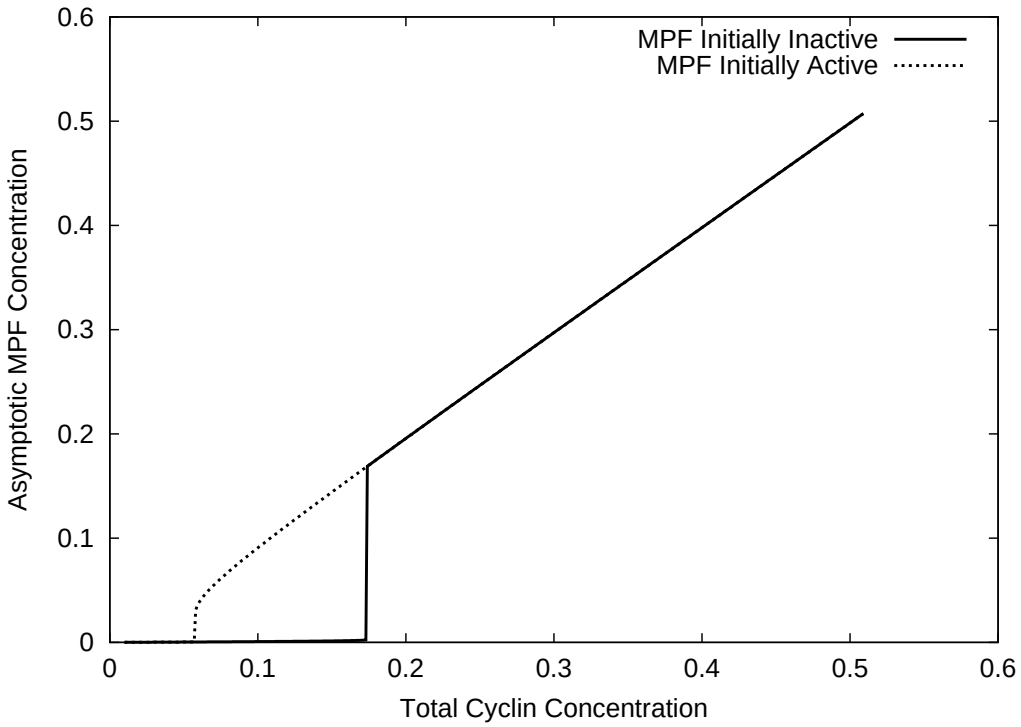

FIG. 3. The asymptotic MPF concentrations versus the total cyclin concentration for both MPF initially active and MPF initially inactive. This is qualitatively what a plot of asymptotic MPF concentration versus total cyclin concentration should look like. The asymptotic MPF concentration is equivalent to the MPF concentration at the steady state of the ODE described earlier.

There are two pieces of data to add: 0.18 as the threshold for MPF activation and the ratio of the threshold for MPF activation to the threshold for MPF inactivation, 3. This data can be added to the data in Table 1 and given to ODRPACK just like the data in Table 1. However, the function f must be modified. The function f was previously described as the time lag for MPF activation versus total cyclin concentration. f must now be a different function when the new data is used. ODRPACK is not affected if f is a

different function depending on which datum is given to f. Also, there is a point at 0.15 total cyclin with a time lag of 55 minutes. This point was recorded as a time that MPF activation was still not seen. The experiment ended before MPF activation was seen with a total cyclin concentration of 0.15. This point was originally misrepresented and is no longer used in favor of the threshold information.

The function used to calculate the thresholds is based on bisection. A total cyclin concentration is found above and below the threshold. The interval is bisected and either the upper or lower bisection of the interval is used as the new interval depending on where the threshold lies. Once the interval is below a tolerance then the value of one side of the interval is returned as the threshold. For the ratio of the thresholds, the two thresholds are simply calculated and the inactivation threshold is divided into the activation threshold. These are the two other functions used when calculating f.

The main program that sets up the parameters to DODRC is now as follows:

begin

$s := 8$;

$n := 6$;

$x := \big(0.20, 0.25, 0.30, 0.50, 0, 0\big)$; The threshold data does not have an explanatory variable. Any value will do, so 0 was chosen.

$y := \big(45, 40, 30, 20, 0.18, 3\big)$; The activation threshold and the threshold ratio are the last two data, respectively.

$w_\delta := \big(25, 16, 11.11, 4, 1, 1\big)$; The weights of the explanatory variables for the threshold information do not matter. Values of 1 were chosen.

$w_\epsilon := \big(4.938 \cdot 10^{-4}, 6.25 \cdot 10^{-4}, 1.111 \cdot 10^{-3}, 2.4 \cdot 10^{-3}, 30.86, 0.1111\big)$;

$\beta := \big(0.5, 0.06, 80\big)$;

$I_x := \big(1, 1, 1, 1, 0, 0\big)$; I_x specifies which x to vary when finding the orthogonal distance. A value of 1 tells ODRPACK to vary the corresponding x, a value of 0 tells ODRPACK not to vary the corresponding x. This will ensure no error is introduced in the explanatory variables for the threshold information and no time will be wasted finding the orthogonal distance for the threshold information.

DODRC(FCN, $n, s, x, y, w_\delta, w_\epsilon, \beta, I_x, \ldots$);

end

The FCN function is similar with a different means of calculating the last two data (the threshold information):

subroutine FCN

for $i := 1$ **step** 1 **until** $n - 2$ **do**

$Y(i)$:=TIMELAG$\big(0, \beta, X(i)\big)$; The TIMELAG function contains the code that is in the for loop of the FCN function described earlier. The only difference is when the first parameter to TIMELAG is 0, MPF is initially inactive. If the first parameter to TIMELAG is 1, MPF is initially active.

enddo

$Y(n-1)$:=THRESHOLD$\big(0, \beta\big)$; Calculate the threshold for MPF activation.

$Y(n) := Y(n-1)/$THRESHOLD$\big(1, \beta\big)$; Divide the threshold for MPF activation by the threshold for MPF inactivation.

end subroutine FCN

The THRESHOLD function calculates the threshold for MPF activation or inactivation for a given set of rate constants β. If the first parameter, a, to THRESHOLD is 1 then MPF is initially active. If the first parameter to THRESHOLD is 0 then MPF is initially inactive. The estimate for the threshold is returned in h.

subroutine THRESHOLD
  $B := 0.01$; Initialize the lower bound for the threshold. The initial value of B may not be a lower bound. A more realistic lower bound will be found later, if B is not a lower bound.
  $C := 1$; Initialize the upper bound for the threshold. The initial value of C may not be an upper bound. A more realistic upper bound will be found later, if C is not an upper bound.
  $e_{tol} := 10^{-10}$; The error tolerance for calculating the threshold.
  Find the lower bound on the threshold.
  while $((a = 0$ **and** TIMELAG$\big(a, \beta, b\big) < 1440)$ **or** $(a = 1$ **and** TIMELAG$\big(a, \beta, b\big) \geq 1440))$ **and** $(b > e_{tol})$ **do**
    $b := b/2$;
  enddo
  if $b \leq e_{tol}$ **then**
    $h := b$;
    return;
  endif
  Find the upper bound on the threshold.
  while $((a = 0$ **and** TIMELAG$\big(a, \beta, c\big) \geq 1440)$ **or** $(a = 1$ **and** TIMELAG$\big(a, \beta, c\big) < 1440))$ **and** $(c < 1/e_{tol})$ **do**
    $c := c * 2$;
  enddo
  if $c \geq 1/e_{tol}$ **then**
    $h := c$;
    return;
  endif
  while $(c - b)/b > e_{tol}$ **do**
    $next = (c + b)/2$; Bisect the interval.
    if $(a = 0$ **and** TIMELAG$\big(a, \beta, next\big) \geq 1440)$ **or** $(a = 1$ **and** TIMELAG$\big(a, \beta, next\big) < 1440)$ **then**
      $b := next$;
    else
      $c := next$;
    endif
  enddo
  $h := b$;
end subroutine THRESHOLD

Chapter 8: RESULTS WITH THRESHOLD DATA

With the added information about the thresholds and the new code to integrate that information the rate constants change significantly. The optimal rate constants can be found in Table 3.

Table 3. Optimal rate constants from ODRPACK after threshold information was added

Rate Constant	Optimal Value
k_{wee}	$9.096 \cdot 10^{-2}$
k_{25}	$6.591 \cdot 10^{-4}$
k'_{25}	$1.078 \cdot 10^{2}$

The error for the first four data points and the activation threshold can be seen in Fig. 4 and Fig. 5, respectively. The ratio of the activation threshold to the inactivation threshold is 2.9939. The total weighted sum of squares of the errors is $8.22 \cdot 10^{-2}$.

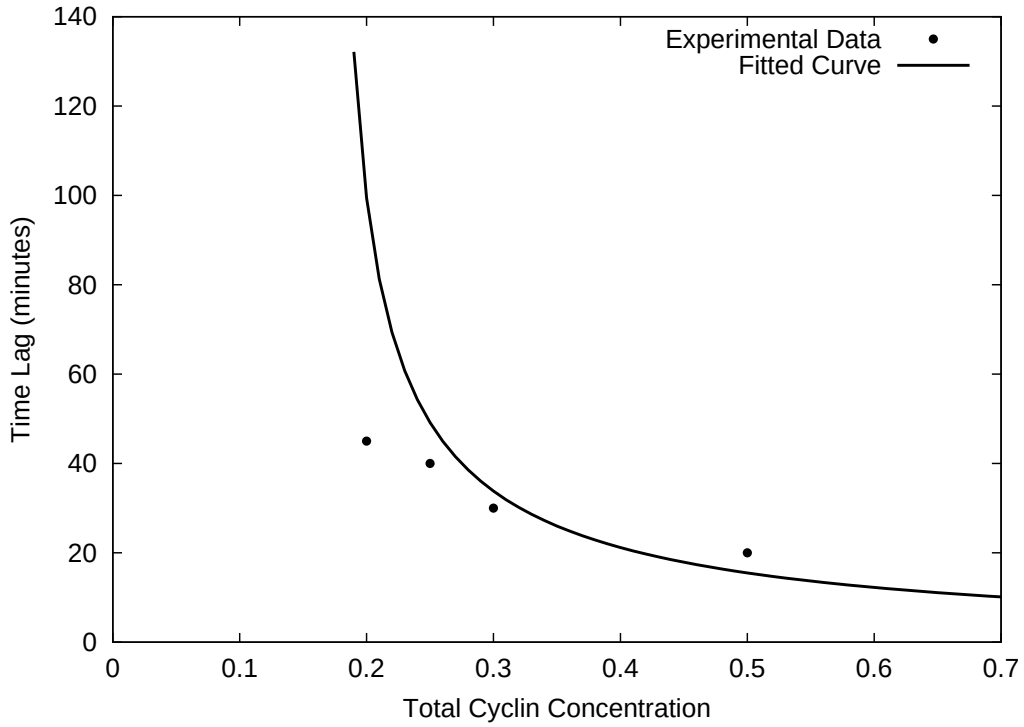

FIG. 4. Time lag for MPF activation versus total cyclin (as calculated using the parameters estimated while considering the thresholds)

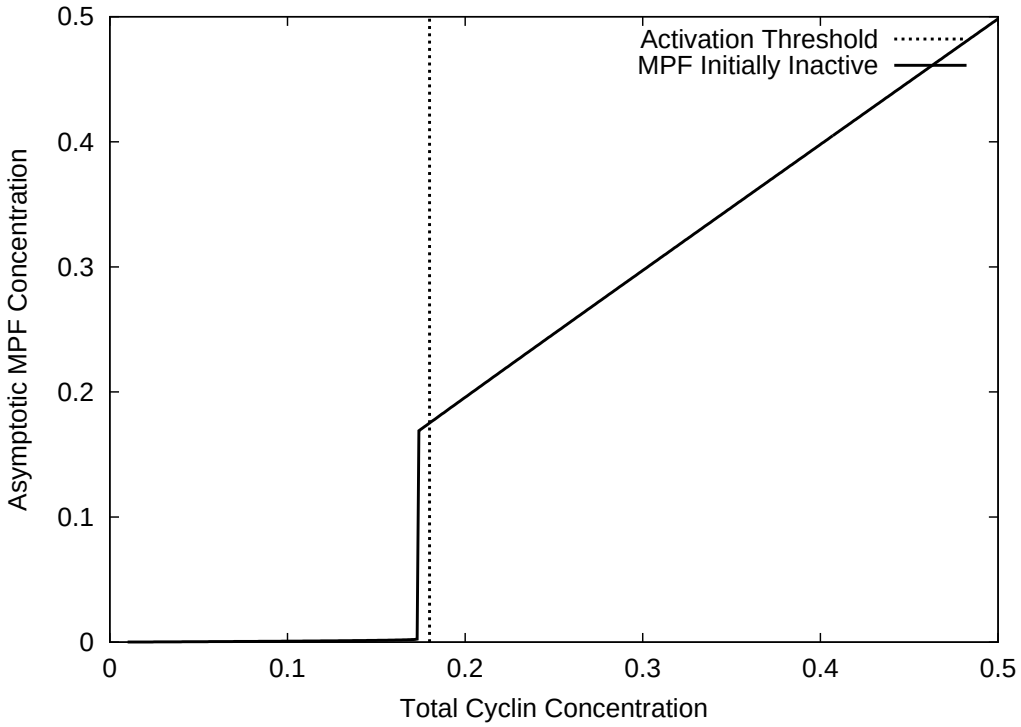

FIG. 5. Asymptotic MPF concentration versus total cyclin concentration. The experimentally measured threshold is also present as a vertical line.

Chapter 9: TEN PARAMETER MODEL

The model used so far has the simplest form that will still yield curves that can reasonably fit the data. The biological system being modelled has two feedback loops. One loop has active MPF activating Cdc25, which in turn activates MPF. The other loop has active MPF inactivating Wee1 inhibiting Wee1 from inactivating MPF. The Cdc25 feedback loop is modelled by $(k_{25} + k'_{25}M^2)(C - M)$. However, the term modelling the Cdc25 feedback is not the accepted way to model such a feedback loop. The Wee1 feedback loop is not modelled at all; only the effect of active Wee1 on inactivating MPF is modelled. Currently the data does not enforce modelling the Cdc25 feedback correctly or modelling the Wee1 feedback loop. The model may more accurately represent new data by using Michaelis-Menten kinetics in the feedback loops.

The equations for the more complicated model [Marlovits et al, 1998] are

$$\begin{aligned}
\frac{dM}{dt} &= k_d(C - M) - k_w M, \\
k_d &= v'_d(1 - d) + v''_d d, \\
k_w &= v'_w(1 - w) + v''_w w, \\
d &= G(M, v'''_d, K_{md}, K_{mdr}), \\
w &= G(v'''_w, M, K_{mw}, K_{mwr}), \\
G(v_1, v_2, K_1, K_2) &= \frac{2v_1K_2}{\beta + \sqrt{\beta^2 - 4v_1K_2(v_2 - v_1)}}, \\
\beta &= v_2 - v_1 + v_1K_2 + v_2K_1.
\end{aligned}$$

k_d and k_w represent the effect of the feedback loops on MPF from Cdc25 and Wee1, respectively. In the equations for k_d and k_w both active (d and w) and inactive ($(1 - d)$ and $(1 - w)$) forms of Cdc25 and Wee1 appear because the inactive forms may still have some activity. The concentrations of active Cdc25 and Wee1 are scaled so they represent the fraction of active concentration over total concentration; therefore the concentration of inactive Cdc25 and Wee1 can be written as $(1 - d)$ and $(1 - w)$.

The equations for active Cdc25 d and active Wee1 w come from differential equations using Michaelis-Menten kinetics. The equations reach a steady state much faster than the differential equation for MPF. Therefore, the model can be approximated by setting the time rate of change of d and w to zero. The resulting equations can be solved for d and w. The solutions for d and w have the form of G.

There are now ten parameters in the model and they are v'_d, v''_d, v'''_d, v'_w, v''_w, v'''_w, K_{md}, K_{mdr}, K_{mw}, and K_{mwr}. There are still six data points to be fit. Therefore not all the parameters can be estimated. The parameter estimates from Marlovits [Marlovits et al, 1998] are used as the parameter values for some parameters and the initial guess for

Table 4. Initial and final estimated parameter values for the ten parameter model

Rate Constant	Initial	Final
v'_d	0.017	0.00123
v''_d	0.17	0.0475
v'''_d, v'''_w	0.05	0.0363
v'_w	0.01	0.000502
v''_w	1.0	0.305
K_{md}	0.1	0.1
K_{mdr}	1.0	1.0
K_{mw}	1.0	1.0
K_{mwr}	0.1	0.1

the remaining parameters. K_{md}, K_{mdr}, K_{mw}, and K_{mwr} are set to the estimates from Marlovits [Marlovits et al, 1998] and v'''_w is set equal to v'''_d. The number of parameters being estimated is then five.

The algorithm for parameter estimation remains the same. The function FEX evaluates the equations described above. The function JEX is empty; the Jacobian matrix is now evaluated numerically. The new initial values of the parameters (and guessed values for the fixed parameters) are in Table 4.

The parameter estimates can be found in Table 4. The experimental data along with the fitted curves are in Fig. 6 and Fig. 7. The ratio of the inactivation threshold to the activation threshold is 2.9993 The total weighted sum of squares is $1.54 \cdot 10^{-2}$. A few thousand points were checked in the neighborhood of the parameter estimate in Table 4 to ensure that the sum of squares is really a minimum.

A sensitivity analysis was done against each of the parameters. Table 5 contains the partial derivative of the weighted sum of squares with respect to each parameter. These derivatives are all small with the exception of the derivative with respect to v'_d. The derivatives were calculated using a center difference formula with a step size of 10^{-4} relative to the parameter.

Table 6 contains partial derivatives of the weighted orthogonal distance squared with respect to each parameter. For every combination of one data point with one parameter there is a partial derivative of the weighted orthogonal distance of the error with respect to the parameter. Table 6 only reports the partial derivatives for one data point for each parameter. The maximum partial for each parameter is reported. j represents the data point that yielded the largest partial. Indexing for j starts at 1. Refer to the pseudocode in Chapter 7 for the order of the data.

Table 5. The partial derivatives of the weighted sum of squares with respect to each parameter at the estimated point in Table 4.

Rate Constant	$\partial E/\partial\beta_i$
v'_d	0.1393
v''_d	0.0051
v'''_d, v'''_w	−0.0089
v'_w	0.0080
v''_w	−0.0005
K_{md}	−0.0034
K_{mdr}	0.0013
K_{mw}	−0.0007
K_{mwr}	0.0022

Table 6. The maximum partial derivatives of the weighted orthogonal distances with respect to each parameter at the estimated point in Table 4.

Rate Constant	j	$\max_j \frac{\partial(w_{\epsilon_j} * \epsilon_j^2 + w_{\delta_j} * \delta_j^2)}{\partial\beta_i}$
v'_d	1	32.84
v''_d	1	138.5
v'''_d, v'''_w	1	155.7
v'_w	4	0.520
v''_w	1	111.2
K_{md}	1	15.24
K_{mdr}	1	114.9
K_{mw}	1	18.55
K_{mwr}	1	3.100

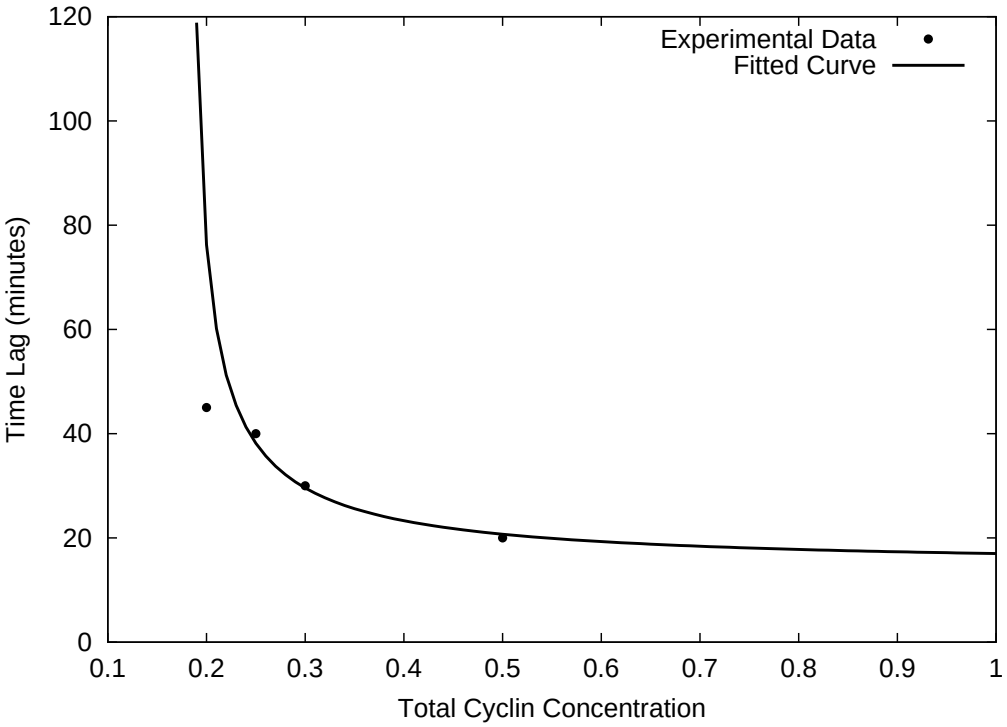

FIG. 6. Time lag for MPF activation versus total cyclin (as calculated using the parameters estimated for the ten parameter model)

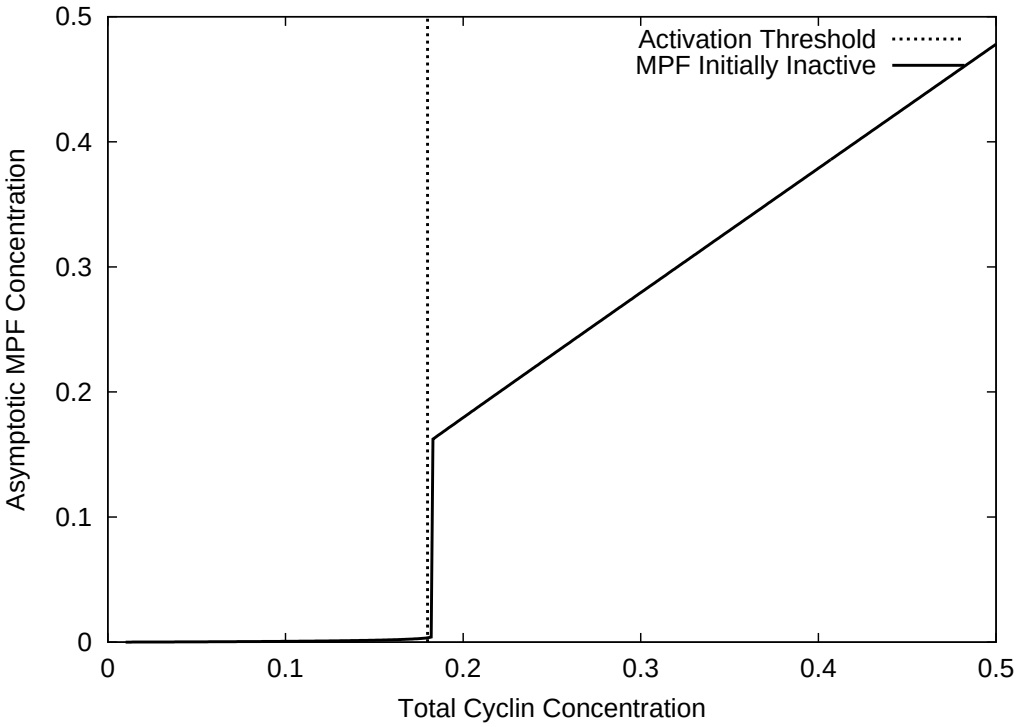

FIG. 7. Asymptotic MPF concentration versus total cyclin concentration for the ten parameter model. The experimentally measured threshold is also present as a vertical line.

Chapter 10: EXPANDING THE TEN PARAMETER MODEL

The ten parameter model was a reasonable model but lacks the ability to model many experiments. Marlovits et al, [1998]has the full model that the ten parameter model was derived from. The Marlovits model can explain a wide range of data. The model described in this chapter is closer to the Marlovits model. One of the simplifying assumptions in the ten parameter model has been removed: the steady state assumption for Wee1 and Cdc25. Also, a dilution factor was added to compensate for dilution in experiments. New experimental data was added to the parameter estimation. The data and the experiments the data come from are described in Appendix F. The results of the parameter estimation is discussed at the end of the chapter. Plots of the experimental data and model data can be seen in Appendix E.

10.1 THREE EQUATION MODEL

The equations for the new model are

$$\begin{aligned}
\frac{dM}{dt} &= k_d * (C - M) - k_w * M, \\
\frac{dD}{dt} &= v_d * \left(\frac{M * (1 - D)}{K_{md} + (1 - D)} - \frac{v_d''' * D}{K_{mdr} + D} \right), \\
\frac{dW}{dt} &= v_w * \left(-\frac{M * W}{K_{mw} + W} + \frac{v_w''' * (1 - W)}{K_{mwr} + (1 - W)} \right), \\
k_d &= v_d'(1 - D) + v_d'' D, \\
k_w &= v_w'(1 - W) + v_w'' W,
\end{aligned}$$

where all the variables and parameters remain the same as in the ten parameter model except D and W, which now represent the concentrations of Cdc25 and Wee1, respectively. The dilution factor is applied to the parameters before these equations are used. These equations are already scaled according to the total concentrations of Cdc2, Cdc25, and Wee1.

10.2 DILUTION FACTOR

The dilution factor was added to compensate for the change in concentrations when solutions are combined in an experiment. In particular, these experiments typically have an extract for which a solution is added containing some protein (for example Cdc25, Wee1, or Cyclin). The equations have all been scaled so that the total concentrations of Wee1, Cdc25, and Cdc2 are all 1. In one experiment a concentration of 1 is different than another experiment, because of the solution with more Wee1, Cdc25, or Cdc2 that was added. The parameters to the ODE's can be scaled with the total concentrations to maintain consistency between experiments.

The scaling is done to the parameters before the ODE's are solved. See the MODEL module in Appendix A for how each parameter is scaled. Here is a brief explanation for how the scaling was done for one equation, the equation for k_d. The unscaled equation for k_d is

$$k_{d_{unscaled}} = v'_d * D_{inactive} + v''_d * D_{active}.$$

$D_{inactive}$ can be substituted with $D_{total} - D_{active}$. Then the equation can be divided by D_{total} making 1 the scaled total concentration of Cdc25. Then we have

$$k_d = v'_d(1 - D) + v''_d D,$$

where D is now the scaled active Cdc25. However, for a different D_{total} the parameters will take on different values even though the scaled equation remains unchanged. Before the scaled equation can be evaluated the parameters must be scaled against the D_{total}. v'_d becomes $v'_d * D_{total}$ and v''_d becomes $v''_d * D_{total}$.

The dilution factor is used when the initial concentration of D_{total} is the same across experiments and some solution is added which dilutes D_{total}. In this case for each experiment D_{total} becomes $\mu * D_{total}$, where μ is the dilution factor.

The final result is that v'_d is substitute with $v'_d * D_{total} * \mu$ and likewise for v''_d. This same technique is used for all the parameters. The substitutions can be seen in the MODEL module in Appendix A.

10.3 RESULTS

The values of the parameters (along with their initial guesses) are in Table 7. The parameter estimation yielded parameter values relatively close to the initial guess. The initial guess was done by hand and therefore cannot be as accurate or precise. However, receiving similar results from the parameter estimation is encouraging.

The graphs of the model and experimental data are in Appendix E. All the graphs show the model and experiments agree with the exception of Figure 16. Some more research should be done into Figure 16 to determine why Wee1 inactivation does not agree between model and experiment. The model and experiment have the same qualitative behavior.

Table 7. Initial and final estimated parameter values for the 15 parameter model

Rate Constant	Fixed	Initial	Final
v'_d	no	0.017	0.0165
v''_d	no	0.17	0.182
v'''_d, v'''_w	no	0.05	0.0709
v'_w	no	0.01	0.0000003
v''_w	no	1.0	0.736
K_{md}	yes	0.1	0.1
K_{mdr}	yes	1.0	1.0
K_{mw}	yes	1.0	1.0
K_{mwr}	yes	0.1	0.1
v_d	yes	1.0	1.0
v_w	yes	1.0	1.0
dilution	no	1.0	2.096
Cdc25 total	no	2.0	0.54
Wee1 total	no	2.0	0.48

Chapter 11: CONCLUSION

Four related models for frog eggs were presented. Each model made progress towards a better approximation for frog eggs. The number of experiments fitted increased. The errors between experiments and the models remained low. For these reasons the parameter estimation was successful.

The final model included concentration levels for three protiens which interacted with each other involving feedback loops. The feedback loops are modelled using Michaelis-Menton kinetics. The three equation model is reasonable and fits many experiments.

More models will be used with this parameter estimator in the future. The long term goal is to integrate the parameter estimator with a larger project providing tools to theoretical biologists to build and refine models.

BIBLIOGRAPHY

Boggs, P.T., Byrd, R.H., Donaldson, J.R., and Schnabel, R.B. 1989. Algorithm 676 — ODRPACK: Software for Weighted Orthogonal Distance Regression. *ACM Trans. Math. Software 15*, 4, 348–364.

Boggs, P.T., Byrd, R.H., Rogers, J.E., and Schnabel, R.B. 1992. *User's Reference Guide for ODRPACK Version 2.01: Software for Weighted Orthogonal Distance Regression*. Center for Computing and Applied Mathematics, U.S. Department of Commerce, Gaithersburg, MD

Dekker, T.J. 1969. Finding a zero by means of successive linear interpolation. In *Constructive Aspects of the Fundamental Theorem of Algebra*, Dejon, B., and Henrici, P., Eds., Wiley-Interscience, London

Gear, C.W. 1971. *Numerical Initial Value Problems in Ordinary Differential Equations*. Prentice-Hall, Englewood Cliffs, NJ

Hindmarsh, A.C. 1980. LSODE and LSODI, Two New Initial Value Ordinary Differential Equation Solvers. *ACM SIGNUM Newsletter 15*, 4, 10–11.

Hindmarsh, A.C. 1983. ODEPACK: A Systematized Collection of ODE Solvers, 55–64. In *Scientific Computing*, Stepleman, R.S., et al., Eds., North Holland Publishing Co., New York

Kahaner, D., Moler, C., and Nash, S. 1989. *Numerical Methods and Software*. Prentice-Hall, Inc., Englewood Cliffs, NJ

Kumagai, A., and Dunphy, W. G. 1992. Regulation of the cdc25 Protein during the Cell Cycle in Xenopus Extracts. *Cell*, 70, 139–151.

Kumagai, A., and Dunphy, W. G. 1995. Control of the Cdc2/Cyclin B Complex in Xenopus Egg Extracts Arrested at a G2/M Checkpoint with DNA Synthesis Inhibitors. *Mol. Bio. of the Cell*, 6, 199–213.

Marlovits, G., Tyson, C.J., Novak, B., and Tyson, J.J. 1998. Modeling M-phase control in Xenopus oocyte extracts: the surveillance mechanism for unreplicated DNA. *Biophys. Chem.*, 72, 169–184.

Moore, J. 1997. Private Communication, Aug.

Petzold, L. 1983. Automatic Selection of Methods for Solving Stiff and Nonstiff Systems of Ordinary Differential Equations. *SIAM Journal on Scientific and Statistical Computing*, 4, 136–148.

Radhakrishnan, K., and Hindmarsh, A.C. 1993. *Description and Use of LSODE, the Livermore Solver for Ordinary Differential Equations*. NASA Reference Publication 1327. Lawrence Livermore National Laboratory, Livermore, CA, Dec

Shampine, L.F., and Allen, R.C. 1973. *Numerical Computing: An Introduction*. W. B. Saunders Company, Philadelphia, PA

Shampine, L.F., and Gordon, M.K. 1975. *Computer Solution of Ordinary Differential Equations, The Initial Value Problem*. W. H. Freeman, San Francisco, CA

Tang, Z., Coleman, T. R., and Dunphy, W. G. 1993. Two distinct mechanisms for negative regulation of the Wee1 protein kinase. *EMBO J.*, 12, 9:3427–3436.

Tyson, J.J., Novak, B., Chen, K., and Val, J. 1995. Checkpoints in the cell cycle from a modeler's perspective. *Progress in Cell Cycle Research*, 1, 1–8.

Appendix A: Fortran 90 Source Code

The Fortran 90 modules and subroutines `EST`, `INPUTMOD`, `FCN`, `TIMELAGMOD`, and `MODEL` which are discussed in Chapters 4, 7, and 9 are listed here.

```
      PROGRAM EST

!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
!  NAME: EST
!  DESCRIPTION:
!      EST is used for parameter estimation in cell cycle models.  EST
!      uses ODRPACK to minimize an objective function consisting of the weighted
!      sum of squares of the residuals between model and experimental data.  The
!      high level psuedo code for EST is:
!
!          read input
!            (experimental data, initial parameters, some ODRPACK parameters,
!             etc.)
!          call ODRPACK
!
!      ODRPACK then calls FCN to get data from the model to compare to the
!      experimental data read.
!
!      FCN is a vector function of the parameters to the model and the
!      explanatory variables f_i(x_i;B).  FCN calculates the dependent variable
!      y_i.  ODRPACK uses FCN to minimize the objective function
!
!          min    ( sum i=1 to n of ( w_e_i * e_i^2 + w_d_i * d_i^2 ) )
!          B,d,e
!
!      where e_i is the error for the ith dependent variable, d_i is the error
!      for the ith explanatory variable, w_* are the weights for their
!      respective subscripts.  While minimizing the sum of squares ODRPACK
!      works under the constraint
!
!          y_i = f_i(x_i + d_i; B) - e_i.
!
!      FCN takes (x_i + d_i) and B to calculate y_i.  FCN does so by simulating
!      biological experiments using a biological model composed of ordinary
!      differential equations.  The exact process FCN uses to simulate
!      experiments varies from experiment to experiment and model to model.  The
!      simulations usually involve setting up initial conditions and running
!      LSODAR to solve the ODE's.
!
!      LSODAR is a numerical integrater for solving ordinary differential
!      equations.  LSODAR automatically switches between a non-stiff
!      (Adams-Moulton) and stiff (Backwards Differentiation Formulas) methods
!      for solving ODE's.  LSODAR also has a root finder that can be used by FCN
!      to find a time given a value of one of the ODE's.
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

      USE INPUTMOD
      USE REAL_PRECISION
```

```
      INTERFACE

        SUBROUTINE FCN(N,M,NP,NQ,            &
                       LDN,LDM,LDNP,         &
                       BETA,XPLUSD,          &
                       IFIXB,IFIXX,LDIFX,    &
                       IDEVAL,F,FJACB,FJACD, &
                       ISTOP)
        USE REAL_PRECISION
        !  INPUT ARGUMENTS, NOT TO BE CHANGED BY THIS ROUTINE:
        INTEGER          :: IDEVAL,LDIFX,LDM,LDN,LDNP,M,N,NP,NQ
        REAL (KIND=R8)   :: BETA(NP),XPLUSD(LDN,M)
        INTEGER          :: IFIXB(NP),IFIXX(LDIFX,M)
        !  OUTPUT ARGUMENTS:
        REAL (KIND=R8)   :: F(LDN,NQ),FJACB(LDN,LDNP,NQ),FJACD(LDN,LDM,NQ)
        INTEGER          :: ISTOP
        END SUBROUTINE FCN

      END INTERFACE



      CALL INPUT()

!  SET UP ODRPACK REPORT FILES
      LUNERR  =   9
      LUNRPT  =   9
      OPEN (UNIT=9,FILE='REPORT')



      WRITE (6,*)"STARTING DODRC "
      CALL DODRC(FCN,                          &
                 N,M,NP,NQ,                    &
                 BETA,                         &
                 Y,LDY,X,LDX,                  &
                 WE,LDWE,LD2WE,WD,LDWD,LD2WD,  &
                 IFIXB,IFIXX,LDIFX,            &
                 JOB,NDIGIT,TAUFAC,            &
                 SSTOL,PARTOL,MAXIT,           &
                 IPRINT,LUNERR,LUNRPT,         &
                 STPB,STPD,LDSTPD,             &
                 SCLB,SCLD,LDSCLD,             &
                 WORK,LWORK,IWORK,LIWORK,      &
                 INFO)
      WRITE (6,*)INFO
      WRITE (6,*)"DONE WITH DODRC"

      CLOSE (9)

      END PROGRAM EST
```

```
      MODULE INPUTMOD

!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
!  INPUTMOD CONTAINS A SUBROUTINE INPUT WHICH READS A NAMELIST FORMATED FILE
!  CONTAINING ODRPACK PARAMETERS.  THE MODULE CONTAINS PUBLIC
!  VARIABLES TO HOLD THE ODRPACK PARAMATERS FOR USE BY THE CALLING PROGRAM.
!
!  SEE THE ODRPACK USER'S GUIDE FOR DETAILED DESCRIPTION OF THE PARAMETERS:
!      Boggs, P.T., Byrd, R.H., Rogers, J.E., and Schnabel, R.B. 1992.
!      User's Reference Guide for ODRPACK Version 2.01: Software for Weighted
!      Orthogonal Distance Regression.  Center for Computing and Applied
!      Mathematics, U.S. Department of Commerce, Gaithersburg, MD
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC




      USE REAL_PRECISION





!  PARAMETER DECLARATIONS AND SPECIFICATIONS,  THESE ARE THE CONSTANTS USED WITH
!  ODRPACK OR FOR ARRAY DECLARATIONS.
      INTEGER :: LDIFX,LDSCLD,LDSTPD,LDWD,LDWE,LDX,LDY,LD2WD,LD2WE,      &
                 LIWORK,LWORK,MAXM,MAXN,MAXNP,MAXNQ

      PARAMETER (MAXM=25,MAXN=125,MAXNP=125,MAXNQ=25,                    &
                 LDY=MAXN,LDX=MAXN,                                      &
                 LDWE=MAXN,LD2WE=1,LDWD=MAXN,LD2WD=1,                    &
                 LDIFX=MAXN,LDSTPD=1,LDSCLD=1,                           &
                 LWORK=18 + 11*MAXNP + MAXNP**2 + MAXM + MAXM**2 +       &
                       4*MAXN*MAXNQ + 6*MAXN*MAXM + 2*MAXN*MAXNQ*MAXNP + &
                       2*MAXN*MAXNQ*MAXM + MAXNQ**2 +                    &
                       5*MAXNQ + MAXNQ*(MAXNP+MAXM) + LDWE*LD2WE*MAXNQ,  &
                 LIWORK=20+MAXNP+MAXNQ*(MAXNP+MAXM)                      &
                 )

!  VARIABLE DECLARATIONS, THESE ARE THE VARIABLES PASSED TO ODRPACK.
      INTEGER           :: INFO,IPRINT,J,JOB,L,LUNERR,LUNRPT,M,MAXIT,N,   &
                           NDIGIT,NP,NQ
      INTEGER           :: IFIXB(MAXNP),IFIXX(LDIFX,MAXM),IWORK(LIWORK)
      REAL (KIND=R8)    :: PARTOL,SSTOL,TAUFAC
      REAL (KIND=R8)    :: BETA(MAXNP),SCLB(MAXNP),SCLD(LDSCLD,MAXM),     &
                           STPB(MAXNP),STPD(LDSTPD,MAXM),                &
                           WD(LDWD,LD2WD,MAXM),WE(LDWE,LD2WE,MAXNQ),     &
                           WORK(LWORK),X(LDX,MAXM),Y(LDY,MAXNQ)
```

```
      CONTAINS




!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
!  NAME: INPUT
!
!  DESCRIPTION:
!      SET ODRPACK PARAMETERS TO DEFAULTS THEN READ PARAMETERS FROM A FILE.
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE INPUT ( )
      NAMELIST / ODRPACKPARAMS / &
        MAXIT, NP, BETA, IFIXB, N, X, Y, WD, WE, IFIXX, &
        JOB, WORK, NDIGIT, TAUFAC, SSTOL, PARTOL, IPRINT, LUNERR, &
        STPB, STPD, SCLB, SCLD, NQ, M

      !  Set defaults

      IFIXB(1)   = -1
      IFIXX(1,1) = -1
      JOB        = -1
      NDIGIT     =  6
      TAUFAC     = -1_R8
      SSTOL      = -1_R8
      PARTOL     = -1_R8
      IPRINT     = -1
      LUNERR     = -1
      LUNRPT     = -1
      STPB(1)    = -1.0_R8
      STPD(1,1)  = -1.0_R8
      SCLB(1)    = -1.0_R8
      SCLD(1,1)  = -1.0_R8

      M  = 1
      NQ = 1

      IPRINT  = 6666

      OPEN  ( 5, FILE="odrpackparams.nml" )
      READ  ( 5, NML=ODRPACKPARAMS )
      CLOSE ( 5 )

      END SUBROUTINE INPUT




!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
!
!  NAME: OLDINPUT
!
!  THIS ROUTINE IS NO LONGER USED, IT IS KEPT IN CASE OLD RUNS WISH TO BE
!  DUPLICATED.
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE OLDINPUT()
      INTEGER :: I,J

      IFIXB(1)   = -1
      IFIXX(1,1) = -1
      JOB        = -1
      NDIGIT     =  6
      TAUFAC     = -1_R8
      SSTOL      = -1_R8
      PARTOL     = -1_R8
      IPRINT     = -1
      LUNERR     = -1
      LUNRPT     = -1
      STPB(1)    = -1.0_R8
      STPD(1,1)  = -1.0_R8
      SCLB(1)    = -1.0_R8
      SCLD(1,1)  = -1.0_R8

      M  = 1
      NQ = 1

      IPRINT  = 6666

!  READ DATA AND PARAMETERS FROM FILE

      OPEN (UNIT=5,FILE='input')
      READ (5,FMT=*) MAXIT,NP
      READ (5,FMT=*) (BETA(I),I=1,NP)
      READ (5,FMT=*) (IFIXB(I),I=1,NP)
      READ (5,FMT=*) N
      READ (5,FMT=*) (X(I,1),I=1,N)
      READ (5,FMT=*) (Y(I,1),I=1,N)
      READ (5,FMT=*) (WD(I,1,1),I=1,N)
      READ (5,FMT=*) (WE(I,1,1),I=1,N)
      READ (5,FMT=*) (IFIXX(I,1),I=1,N)
      CLOSE(5)




      END SUBROUTINE OLDINPUT

      END MODULE INPUTMOD
```

```
      MODULE TIMELAGMOD

!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
!  THE TIMELAGMOD MODULE HAS FUNCTIONS TO CALCULATE THE TIMELAG FUNCTION,
!  THRESHOLD FUNCTIONS, TIMECOURSE FUNCTIONS, AND AUXILARY FUNCTIONS FOR THE
!  FORMER.
!
!  THESE FUNCTIONS ARE USED BY FCN WHEN CALCULATING MODEL DATA TO BE COMPARED TO
!  EXPERIMENTAL DATA.
!
!  THROUGHOUT THIS MODULE 1440 MINUTES IS USED AS THE UPPER BOUND WHEN AN
!  INFINITE TIME IS BEING SOUGHT.
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC



      USE MODEL
      USE REAL_PRECISION



      CONTAINS




!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
!  NAME:  REPORTLSODAR
!
!  REPORTS THE STATE OF LSODAR AFTER IT RETURNED.  TAKES THE ISTATE VARIABLE
!  AND TURNS IT TO A TEXT ERROR MESSAGE (IF AN ERROR OCCURRED).
!
!  ARGUMENTS:
!
!      MESSAGE    A PREFIX FOR THE ERROR MESSAGE
!      ISTATE     THE RETURN STATE OF LSODAR
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

      SUBROUTINE REPORTLSODAR (MESSAGE,ISTATE)
      INTEGER :: ISTATE
      CHARACTER(8) :: MESSAGE

      IF (ISTATE .EQ. -1) THEN
        WRITE(6,*) MESSAGE, &
        'LSODAR: excess work done on this call (perhaps wrong jt)'
      ELSE IF (ISTATE .EQ. -2) THEN
        WRITE(6,*) MESSAGE, &
```

```
        'LSODAR: excess accuracy requested (tolerances too small)'
      ELSE IF (ISTATE .EQ. -3) THEN
        WRITE(6,*) MESSAGE, &
        'LSODAR: illegal input detected (see printed message).'
      ELSE IF (ISTATE .EQ. -4) THEN
        WRITE(6,*) MESSAGE, &
        'LSODAR: repeated error test failures (check all inputs).'
      ELSE IF (ISTATE .EQ. -5) THEN
        WRITE(6,*) MESSAGE, &
        "LSODAR: repeated convergence failures (perhaps bad jacobian    &
     & supplied or wrong choice of jt or tolerances)."
      ELSE IF (ISTATE .EQ. -6) THEN
        WRITE(6,*) MESSAGE, &
        'LSODAR: error weight became zero during problem. (solution     &
     &        component i vanished, and atol or atol(i) = 0.)'
      ELSE IF (ISTATE .EQ. -7) THEN
        WRITE(6,*) MESSAGE, &
        'LSODAR: work space insufficient to finish (see messages).'
      ELSE IF (ISTATE .LT. 0) THEN
        WRITE(6,*) MESSAGE, &
        'LSODAR: some unknown error'
      END IF

      END SUBROUTINE REPORTLSODAR




!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
!  NAME:  THRESHOLD
!
!  DESCRIPTION:
!      FINDS THE ACTIVATION OR INACTIVATION THRESHOLD GIVEN VALUES FOR THE
!      PARAMETERS TO THE MODEL.
!
!  ARGUMENTS:
!      ACTIVATION   0 IF THE ACTIVATION THRESHOLD IS DESIRED
!                   1 IF THE INACTIVATION THRESHOLD IS DESIRED
!      PSIZE        SIZE OF PARAMS
!      PARAMS       ARRAY OF PARAMETERS (EQUIVALENT TO ODRPACK'S BETA)
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

      REAL (KIND=R8)   FUNCTION THRESHOLD (ACTIVATION,PSIZE,PARAMS)
      INTEGER :: PSIZE
      REAL (KIND=R8)   :: PARAMS(PSIZE)
      INTEGER :: ACTIVATION


!  B     THE LOWER BOUND OF THE THRESHOLD BEING CALCULATED
!  C     THE UPPER BOUND OF THE THRESHOLD BEING CALCULATED
!  TOL   THE TOLERENCE FOR THE THRESHOLD BEING CALCULATED, THIS IS THE ALLOWED
!        RELATIVE ERROR IN THE THRESHOLD CALCULATION.
```

```
!  NEXT  THE BISECTION POINT OF B AND C.  AFTER EACH ITERATION EITHER B OR C
!        WILL BE ASSIGNED TO NEXT AND NEXT WILL BE REASSIGNED (C-B)/2.
      REAL (KIND=R8)   :: B,C,TOL,NEXT,TEMPTL

!  INITIAL GUESS FOR LOWER AND UPPER BOUNDS.
      B=0.0010_R8
      C=1_R8

!  TOLERANCE FOR THRESHOLD
      TOL=1E-10_R8

!  FIND LOWER BOUND
!  IF B IS NOT A LOWER BOUND FOR THE THRESHOLD THEN DECREASE B UNTIL IT IS A
!  LOWER BOUND OR IS BELOW TOL.
!
!  THE CONDITION FOR THIS WHILE LOOP SAYS THAT IF THE ACTIVATION THRESHOLD IS
!  DESIRED (ACTIVATION=0) THEN THE LOWER BOUND HAS THE PROPERTY MPF STARTS
!  INITIALLY INACTIVE AND NEVER ACTIVATES.  IF THE INACTIVATION THRESHOLD IS
!  DESIRED (ACTIVATION=1) THEN THE LOWER BOUND HAS THE PROPERTY MPF STARTS
!  ACTIVE AND INACTIVATES AFTER A TIMELAG.
!
      TEMPTL=TIMELAG(ACTIVATION,PSIZE,PARAMS,B)
      DO WHILE (((                    &
                  ACTIVATION .EQ. 0 &
                  .AND.             &
                  TEMPTL<1440_R8     &
                ) .OR. (            &
                  ACTIVATION .EQ. 1 &
                  .AND.             &
                  TEMPTL>=1440_R8    &
                )                   &
               )                    &
               .AND. B>TOL)
        B=B/2_R8;
        WRITE(6,*)"Halving B ",B
        TEMPTL=TIMELAG(ACTIVATION,PSIZE,PARAMS,B)
      END DO

      IF (B<=TOL) THEN
        THRESHOLD=B
        RETURN
      END IF

!  FIND UPPER BOUND
!  IF C IS NOT AN UPPER BOUND FOR THE THRESHOLD THEN INCREASE C UNTIL IT IS AN
!  UPPER BOUND OR ABOVE 1/TOL.
      TEMPTL=TIMELAG(ACTIVATION,PSIZE,PARAMS,C)
      DO WHILE (((                    &
                  ACTIVATION .EQ. 0 &
                  .AND.             &
                  TEMPTL>=1440_R8   &
                ) .OR. (            &
                  ACTIVATION .EQ. 1 &
                  .AND.             &
                  TEMPTL<1440_R8    &
```

```
              )                   &
             )                    &
             .AND. C<1_R8/TOL)
        C=C*2_R8
        WRITE(6,*)"Doubling C ",C
        TEMPTL=TIMELAG(ACTIVATION,PSIZE,PARAMS,C)
      END DO

      IF (C>=1_R8/TOL) THEN
        THRESHOLD=C
        RETURN
      END IF


!  NARROW THE BOUNDS
!  BISECT THE INTERVAL (B,C) UNTIL (C-B)/B .LE. TOL.
      DO WHILE ((C-B)/B>TOL)
        NEXT=(C+B)/2
        TEMPTL=TIMELAG(ACTIVATION,PSIZE,PARAMS,NEXT)
        IF ((                   &
              ACTIVATION .EQ. 0 &
              .AND.             &
              TEMPTL>=1440_R8    &
            ) .OR. (            &
              ACTIVATION .EQ. 1 &
              .AND.             &
              TEMPTL<1440_R8     &
            )                   &
           )                    &
        THEN
          B=NEXT
        ELSE
          C=NEXT
        END IF
      END DO

      THRESHOLD=B

      RETURN
      END FUNCTION THRESHOLD




!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
!  NAME:  TIMELAG
!
!  RETURNS A TIME LAG FOR MPF ACTIVATION OR INACTIVATION FOR A SPECIFIC
!  TOTALCYCLIN.  THERE IS AN ASYMPTOTE FOR THE TIMELAGS AT THE THRESHOLD FOR MPF
!  ACTIVATION OR INACTIVATION (DEPENDING ON ACTIVATED).  TO THE LEFT OF THE
!  ASYMPTOTE IT BECOMES DIFICULT TO HANDLE THE OPTIMIZATION PROBLEM.  TO RESOLVE
!  THIS TIMELAG CREATES AN ARTIFICIAL FUNCTION TO THE LEFT OF THE ASYMPTOTE.
```

```
!  THE FUNCTION IS A STRAIGHT LINE WITH A NEGATIVE SLOPE AND A LARGE Y
!  INTERCEPT.  THE PRECISE EQUATION IS:
!
!              -1440 * TOTALCYCLIN + 2880
!
!  ARGUMENTS:
!
!  ACTIVATED   0       CALCULATIONS ARE MADE WITH MPF INITIALLY 0
!              1       CALCULATIONS ARE MADE WITH MPF INITIALLY TOTALCYCLIN
!              !0 !1   UNDEFINED
!
!  PARAMS      PARAMETERS TO THE ODES
!
!  TOTALCYCLIN         THE TOTAL CYCLIN IN THE SYSTEM.  THIS IS A CONSTANT TO
!                      THE ODES.
!
!  BUGS        I SUSPECT THIS ROUTINE DOES NOT PROPERLY CALCULATE THE TIME LAG
!              WHEN IT IS CLOSE TO THE THRESHOLD.  I SHOULD LOOK INTO THIS IN
!              MORE DETAIL LATER.
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

      REAL (KIND=R8)   FUNCTION TIMELAG (ACTIVATED,PSIZE,PARAMS, &
                                    TOTALCYCLIN_)

      INTEGER :: PSIZE
      REAL (KIND=R8)   :: PARAMS(PSIZE), TOTALCYCLIN_
      INTEGER :: ACTIVATED

!  MINF        THE ASYMPTOTIC VALUE OF MPF
!  MT          THE VALUE OF MPF AT THE (IN)ACTIVATION THRESHOLD
!  T           TIME LAG TO MT
!  FT          F(T) FOR ROOT, MPF VALUE AT T
!  MINFPD      THE MINF PLUS DELTA.  WHEN MINFPD-MINF<EPSILON MINF IS ASYMTOTE
!  AT          ASYMPTOTIC TIME

      REAL (KIND=R8)   ::  MINF, MT, T, FT, MINFPD, AT


!  THE THRESHOLD CAN NEVER BE NEGATIVE, SO RETURN THE ARTIFICIAL FUNCTION VALUE
      IF (TOTALCYCLIN_ .LE. 0) THEN
        TIMELAG=-1440_R8*TOTALCYCLIN_+2880_R8
        RETURN
      END IF

!  CALCULATE ASYMPTOTIC MPF

      MINF=MPF_INF(ACTIVATED,PSIZE,PARAMS,TOTALCYCLIN_)

!  RETURN THE VALUE OF THE ARTIFICIAL FUNCTION IF MPF NEVER (IN)ACTIVATES.

      IF ((MINF .GT. TOTALCYCLIN_/2 .AND. ACTIVATED .EQ. 1) .OR. &
          (MINF .LT. TOTALCYCLIN_/2 .AND. ACTIVATED .EQ. 0)) THEN
        TIMELAG=-1440_R8*TOTALCYCLIN_+2880_R8
```

```
        RETURN
      END IF

      IF (ACTIVATED .EQ. 1) THEN
        MT=(TOTALCYCLIN_-MINF)/2+MINF
      ELSE
        MT=MINF/2
      END IF

!  USE LSODAR TO CALCULATE THE ROOT OF THE ODE TO FIND THE TIMELAG FOR MPF
!  ACTIVATION OR INACTIVATION.
      TIMELAG=MPFINV(ACTIVATED,PSIZE,PARAMS,TOTALCYCLIN_,MT)

      RETURN

      END FUNCTION TIMELAG





!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
!  NAME:        TIMELAGINV
!
!  RETURNS A TOTALCYCLIN FOR MPF ACTIVATION OR INACTIVATION FOR A SPECIFIC
!  TIME LAG.  THIS IS THE INVERSE TIMELAG FUNCTION.
!
!  ACTIVATED    0       CALCULATIONS ARE MADE WITH MPF INITIALLY 0
!               1       CALCULATIONS ARE MADE WITH MPF INITIALLY TOTALCYCLIN
!               !0 !1   UNDEFINED
!
!  PARAMS       PARAMETERS TO THE ODES
!
!  TIMELAG_     THE TIME LAG FOR MPF INACTIVATION OR ACTIVATION.
!
!  THERE ARE TWO POTENTIAL PROBLEMS WITH THIS FUNCTION.  IF TIMELAG_ IS TOO
!  HIGH (APPROXIMATELY >= 1440) THEN THERE IS NO SOLUTION AND THIS FUNCTION
!  WILL NOT BEHAVE CORRECTLY.  ALSO, B NEEDS TO BE SUFFICIENTLY FAR FROM THE
!  ASYMPTOTE SO THAT ROOT CAN TAKE FINITE STEPS TOWARDS THE ASYMPTOTE (IF ROOT
!  DOES SUCH THINGS).
!
!  NOTE: THE TERM PLATEAU IS USED THROUGHOUT THE DOCUMENATION IN THIS FUNCTION.
!  WHAT IS MEANT IS THE ARTIFICIAL FUNCTION TO THE LEFT OF THE ASYMPTOTE AS
!  DESCRIBED IN THE DOCUMENTATION FOR TIMELAG.
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

      REAL (KIND=R8)   FUNCTION TIMELAGINV &
             (ACTIVATED,PSIZE,PARAMS,TIMELAG_)

      INTEGER :: PSIZE
      REAL (KIND=R8)   :: PARAMS(PSIZE), TOTALCYCLIN_, TIMELAG_
      INTEGER :: ACTIVATED
```

```
!  TL           THE CURRENT TIME LAG, THIS NUMBER SHOULD APPROACH TIMELAG AS
!               THE ROOT IS CLOSER.
!  B_PREV       USED TO BOUND B WHILE FINDING THE LOWER BOUND OF TOTALCYCLIN

      REAL (KIND=R8)   :: TL,B_PREV


!  ROOT PARAMETERS
!  B            CLOSER APPROXIMATION TO THE ROOT
!  C            APPROXIMATION ON THE OPOSITE SIDE OF THE ROOT AS B
!  RELERR       RELATIVE ERROR (IN B )
!  ABSERR       ABSOLUTE ERROR (IN B )
!  IFLAG        ZERO TO START, ROOT RETURNS NEGATIVE IF IT NEEDS ANOTHER FT

      REAL (KIND=R8)   :: B,C,RELERR,ABSERR
      INTEGER :: IFLAG

      B      = 1E-2_R8
      C      = 1_R8
      RELERR = 1E-10_R8
      ABSERR = 0_R8
      IFLAG  = 1


      IF (ACTIVATED .EQ. 1) THEN
        WRITE(6,*)"TIMELAGINV NOT IMPLEMENTED FOR MPF INIT = &
     &TOTALCYCLIN"
        RETURN
      END IF


!  BRACKET TOTALCYCLIN TO THE RIGHT
!  D18 IS ROUGHLY 2**63
      TL=TIMELAG(ACTIVATED,PSIZE,PARAMS,C)
      DO WHILE (TL>=TIMELAG_ .AND. C<1E8_R8)
        C=2*C
        TL=TIMELAG(ACTIVATED,PSIZE,PARAMS,C)
      END DO


!  BRACKET TOTALCYCLIN TO THE LEFT
      B_PREV=0
      TL=TIMELAG(ACTIVATED,PSIZE,PARAMS,B)
!  ENSURE THAT B IS NOT ON THE PLATEAU
      DO WHILE (TL>=1440 .AND. B<C)
        B_PREV=B
        B=2*B
        TL=TIMELAG(ACTIVATED,PSIZE,PARAMS,B)
      END DO
      IF (B > C) THEN
```

```
        B=C
      END IF
!  THE END OF THE PLATEAU AND TIMELAG_ ARE BRACKETED BETWEEN B_PREV AND B IF TL
!  IS GREATER THAN 1440 OR LESS THAN TIMELAG_.  ANY POINT BETWEEN THE END OF
!  THE PLATEAU AND TIMELAG_ IS ACCEPTABLE FOR B.  THE FOLLOWING LOOP SEARCHES
!  FOR SUCH A POINT.
      DO WHILE ((TL>=1440 .OR. TL<=TIMELAG_) .AND. B>1E-8_R8)
        TL=TIMELAG(ACTIVATED,PSIZE,PARAMS,(B_PREV+B)/2)
        IF (TL>=1440) THEN
          B_PREV=(B_PREV+B)/2
        ELSE
          B=(B_PREV+B)/2
        END IF
      END DO



      IF (TL>=1440 .OR. TL<=TIMELAG_) THEN
        TIMELAGINV=B
        RETURN
      END IF



      TOTALCYCLIN_=B
      CALL ROOT(TOTALCYCLIN_,TL-TIMELAG_,B,C,RELERR,ABSERR,IFLAG)
      DO WHILE (IFLAG<0)
        TL=TIMELAG(ACTIVATED,PSIZE,PARAMS,TOTALCYCLIN_)
        CALL ROOT(TOTALCYCLIN_,TL-TIMELAG_,B,C,RELERR,ABSERR,IFLAG)
      END DO

      IF (IFLAG .NE. 1 .AND. IFLAG .NE. 2) THEN
        WRITE(6,*)"ROOT RETRURNED IFLAG ",IFLAG
      END IF

      TIMELAGINV=B
      RETURN

      END FUNCTION TIMELAGINV


!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
!  NAME:         MPF
!
!  RETURNS THE VALUE OF MPF AT THE SPECIFIED PARAMETERS AND TIME WITH THE
!  SPECIFIED TOTALCYCLIN.  THIS WORKS WITH AN INITIAL VALUE OF MPF ACTIVE OR
!  INACTIVE DEPENDING ON THE ACTIVATED PARAMETER TO THIS FUNCTION.
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

      REAL (KIND=R8)   FUNCTION MPF &
             (ACTIVATED,PSIZE,PARAMS,TOTALCYCLIN_,TOUT)

      REAL (KIND=R8)   :: PARAMS(PSIZE), TOTALCYCLIN_
```

```
      INTEGER :: ACTIVATED, PSIZE

!  LSODAR PARAMETERS
      INTEGER ::        NEQ, ITOL, ITASK, ISTATE, IOPT, LRW, &
                        IWORK, LIW, JT, NG, JROOT, MF
      REAL (KIND=R8)   ::  Y, T, TOUT, RTOL, ATOL, RWORK
      DIMENSION ::      NEQ(1), Y(5), RTOL(1), ATOL(1), &
                        RWORK(215), IWORK(30), JROOT(0)

!  SET ODE PARAMETERS
      CALL MAPBTOPARAMS ( PSIZE, PARAMS )
      CALL SETTOTALCYCLIN ( TOTALCYCLIN_ )

      NEQ     = 5
      T       = 0.E0_R8
      ITOL    = 1
      RTOL    = 1.E-10_R8
      ATOL    = 1.E-10_R8
      ITASK   = 1
      ISTATE  = 1
      IOPT    = 1
      LRW     = 215 ! enough for 10 eqs
      LIW     = 30  ! enough for 10 eqs
      JT      = 2
      NG      = 0
      MF      = 21

!  MAXIMUM STEPS IN ONE CALL TO SOLVER (DEFAULT 500) IWORK(6)
!  THESE ARE OPTIONAL INPUTS, 0 VALUES REPRESENT DEFAULTS
      IWORK( 5) = 0
      IWORK( 6) = 1000000
      IWORK( 7) = 0
      IWORK( 8) = 0
      IWORK( 9) = 0
      IWORK(10) = 0

      RWORK( 5) = 0.0
      RWORK( 6) = 0.0
      RWORK( 7) = 0.0
      RWORK( 8) = 0.0
      RWORK( 9) = 0.0
      RWORK(10) = 0.0

      IF (ACTIVATED .EQ. 1) THEN
        Y(1) = TOTALCYCLIN_
        Y(2) = 1.0
        Y(3) = 0.0
        Y(4) = 0.0
        Y(5) = 0.0
      ELSE
        Y(1) = 0.E0_R8
        Y(2) = 0.0
        Y(3) = 1.0
        Y(4) = 0.0
```

```
        Y(5) = 0.0
      END IF


!  SOLVE ODE FOR SPECIFIC POINT

      CALL lsodar(FEX,NEQ,Y,T,TOUT,ITOL,RTOL,ATOL,ITASK,ISTATE, &
                  IOPT,RWORK,LRW,IWORK,LIW,JEX,JT,GEX,NG,JROOT)
      CALL REPORTLSODAR('MPF:    ',ISTATE)

      MPF=Y(1)

      RETURN
      END FUNCTION MPF


!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
!  NAME:        MPFINV
!
!  RETURNS THE VALUE OF T AT THE SPECIFIED PARAMETERS AND MPF WITH THE
!  SPECIFIED TOTALCYCLIN.  THIS WORKS WITH AN INITIAL VALUE OF MPF ACTIVE OR
!  INACTIVE DEPENDING ON THE ACTIVATED PARAMETER TO THIS FUNCTION.
!
!  THIS IS THE INVERSE FUNCTION TO MPF.  IT UTILIZES THE ROOT FINDING
!  CAPABILITIES OF LSODAR.
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

      REAL (KIND=R8)   FUNCTION MPFINV &
             (ACTIVATED,PSIZE,PARAMS,TOTALCYCLIN_,MPFROOT_)

      REAL (KIND=R8)   :: PARAMS(PSIZE), TOTALCYCLIN_, MPFROOT_
      INTEGER ::          ACTIVATED, PSIZE

!  LSODAR PARAMETERS
      INTEGER ::         NEQ, ITOL, ITASK, ISTATE, IOPT, LRW, &
                         IWORK, LIW, JT, NG, JROOT, MF
      REAL (KIND=R8)   :: Y, T, TOUT, RTOL, ATOL, RWORK
      DIMENSION ::       NEQ(1), Y(5), RTOL(1), ATOL(1), &
                         RWORK(215), IWORK(30), JROOT(1)

!  SET ODE PARAMETERS
      CALL MAPBTOPARAMS(PSIZE,PARAMS)
      CALL SETTOTALCYCLIN ( TOTALCYCLIN_ )
      CALL SETMPFROOT ( MPFROOT_ )

      NEQ     = 5
      T       = 0.E0_R8
      ITOL    = 1
      RTOL    = 1.E-10_R8
      ATOL    = 1.E-10_R8
      ITASK   = 1
```

```
      ISTATE  = 1
      IOPT    = 1
      LRW     = 215 ! enough for 10 eqs
      LIW     = 30  ! enough for 10 eqs
      JT      = 2
      NG      = 1
      MF      = 21
      TOUT    = 1440_R8

!  MAXIMUM STEPS IN ONE CALL TO SOLVER (DEFAULT 500) IWORK(6)
!  THESE ARE OPTIONAL INPUTS, 0 VALUES REPRESENT DEFAULTS
      IWORK( 5) = 0
      IWORK( 6) = 1000000
      IWORK( 7) = 0
      IWORK( 8) = 0
      IWORK( 9) = 0
      IWORK(10) = 0

      RWORK( 5) = 0.0
      RWORK( 6) = 0.0
      RWORK( 7) = 0.0
      RWORK( 8) = 0.0
      RWORK( 9) = 0.0
      RWORK(10) = 0.0

      IF (ACTIVATED .EQ. 1) THEN
        Y(1) = TOTALCYCLIN_
        Y(2) = 1.0
        Y(3) = 0.0
        Y(4) = 0.0
        Y(5) = 0.0
      ELSE
        Y(1) = 0.E0_R8
        Y(2) = 0.0
        Y(3) = 1.0
        Y(4) = 0.0
        Y(5) = 0.0
      END IF


!  SOLVE ODE FOR SPECIFIC POINT

      CALL lsodar(FEX,NEQ,Y,T,TOUT,ITOL,RTOL,ATOL,ITASK,ISTATE, &
                  IOPT,RWORK,LRW,IWORK,LIW,JEX,JT,GEX,NG,JROOT)
      CALL REPORTLSODAR('MPFINF: ',ISTATE)



      IF (JROOT(1) .EQ. 1) THEN
        MPFINV=T
      ELSE
        MPFINV=-1440*TOTALCYCLIN_+2880
      END IF

      RETURN
```

```
      END FUNCTION MPFINV




!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
!  NAME:        MPF_INF
!
!  DESCRIPTION:
!      RETURNS THE ASYMPTOTIC (STEADY STATE) CONCENTRATION OF ACTIVE MPF.  THE
!      STEADY STATE CONCENTRATION IS DEFINED BY THIS FUNCTION AS THE
!      CONCENTRATION WHERE THE CHANGE IN MPF ACTIVITY IS LESS THAN 10^-8 AND
!      TIME .GT. 1440.  THIS APPROXIMATION ALSO STOPS WHEN THE TIME IS .GT.
!      10^5.
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      REAL (KIND=R8)   FUNCTION MPF_INF(ACTIVATED,PSIZE,PARAMS, &
                                        TOTALCYCLIN_)
      REAL (KIND=R8)   :: PARAMS(PSIZE),TOTALCYCLIN_,AT,MINF,MINFPD
      INTEGER :: ACTIVATED, PSIZE

      AT=1440_R8
      MINF   = MPF(ACTIVATED,PSIZE,PARAMS,TOTALCYCLIN_,AT)
      MINFPD = MPF(ACTIVATED,PSIZE,PARAMS,TOTALCYCLIN_,AT*1.01)
      DO WHILE ((MINF-MINFPD)/MINF > 1E-8_R8 .AND. AT<1E5_R8)
        AT=AT*2
        MINF   = MPF(ACTIVATED,PSIZE,PARAMS,TOTALCYCLIN_,AT)
        MINFPD = MPF(ACTIVATED,PSIZE,PARAMS,TOTALCYCLIN_,AT*1.01)
      END DO

      MPF_INF=MINF

      RETURN
      END FUNCTION MPF_INF




!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
!  NAME: KEDFIG4B
!
!  CALCULATES THE TIME COURSE FOR THREE POINTS TO BE COMPARED WITH THE DATA FROM
!  K AND D FIGURE 4B.
!
!  ACTIVATED      1 IF MPF IS INITIALLY ACTIVE
!                 0 IF MPF IS INITIALLY INACTIVE
!  TIME           THE THREE TIME POINTS FOR WHICH L IS DESIRED.
!  LOUT           THE TIME COURSE AT THE POINTS IN TIME
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
      SUBROUTINE TIMECOURSE ( ACTIVATED, NEQ, Y, PSIZE, PARAMS, TSIZE,  &
                              TIME, LDSIZE, LSIZE, LOUT )
      INTEGER          :: ACTIVATED, PSIZE, TSIZE, LDSIZE, LSIZE
      REAL (KIND=R8)   :: TIME(TSIZE), PARAMS(PSIZE), LOUT(LDSIZE,LSIZE)

!  LSODAR PARAMETERS
      INTEGER ::        NEQ, ITOL, ITASK, ISTATE, IOPT, LRW, &
                        IWORK, LIW, JT, NG, JROOT, MF
      REAL (KIND=R8)   ::  Y, T, TOUT, RTOL, ATOL, RWORK
      DIMENSION ::      NEQ(1), Y(NEQ(1)), RTOL(1), ATOL(1), &
                        RWORK(215), IWORK(30), JROOT(0)

!  LOCAL VARS
      INTEGER :: I

      CALL MAPBTOPARAMS(PSIZE, PARAMS)

      T       = 0.E0_R8
      ITOL    = 1
      RTOL    = 1.E-10_R8
      ATOL    = 1.E-10_R8
      ITASK   = 1
      ISTATE  = 1
      IOPT    = 1
      LRW     = 215 ! enough for 10 eqs
      LIW     = 30  ! enough for 10 eqs
      JT      = 2
      NG      = 0
      MF      = 21

!  MAXIMUM STEPS IN ONE CALL TO SOLVER (DEFAULT 500) IWORK(6)
!  THESE ARE OPTIONAL INPUTS, 0 VALUES REPRESENT DEFAULTS
      IWORK( 5) = 0
      IWORK( 6) = 1000000
      IWORK( 7) = 0
      IWORK( 8) = 0
      IWORK( 9) = 0
      IWORK(10) = 0

      RWORK( 5) = 0.0
      RWORK( 6) = 0.0
      RWORK( 7) = 0.0
      RWORK( 8) = 0.0
      RWORK( 9) = 0.0
      RWORK(10) = 0.0

      IF ( ACTIVATED .EQ. 1 ) THEN
        CALL SETTOTALCYCLIN ( 1.0_R8 )
        Y(1) = 1.0
        Y(2) = 1.0
        Y(3) = 0.0
        Y(4) = 1.0
        Y(5) = 0.0
      ELSE IF ( ACTIVATED .EQ. 0 ) THEN
```

```
  CALL SETTOTALCYCLIN ( 0.0_R8 )
  Y(1) = 0.0
  Y(2) = 0.0
  Y(3) = 1.0
  Y(4) = 1.0
  Y(5) = 0.0
ELSE IF ( ACTIVATED .EQ. -1 ) THEN
  !  Y already set by calling program
ELSE
  WRITE ( 6, * ) "ERROR, ACTIVATED HAS INVALID VALUE: ",ACTIVATED
END IF

DO I = 1, TSIZE
  TOUT = TIME(I)
  CALL lsodar(FEX,NEQ,Y,T,TOUT,ITOL,RTOL,ATOL,ITASK,ISTATE, &
              IOPT,RWORK,LRW,IWORK,LIW,JEX,JT,GEX,NG,JROOT)
  CALL REPORTLSODAR('KED:    ',ISTATE)
  LOUT(I,:) = Y(:)
END DO

END SUBROUTINE TIMECOURSE




END MODULE TIMELAGMOD





!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
!  NAME: FCN
!
!  DESCRIPTION:
!      ODRPACK CALLS FCN TO EVALUATE THE MODEL GIVEN VALUES FOR THE EXPLANATORY
!      VARIABLES AND PARAMETERS.  VALUES FOR THE RESPONSE VARIABLES ARE RETURNED
!      AND ODRPACK USES THIS TO DETERMINE THE WEIGHTED SUM OF SQUARES AND THE
!      JACOBIAN.  AT THE MOMENT THE JACOBIAN IS CALCULATED NUMERICALLY USING
!      DIFFERENCE FORMULAS, BUT THIS FUNCTION IS STILL USED IN THOSE
!      CALCULATIONS.
!
!  SEE ODRPACK DOCUMENTATION FOR A DETAILED DESCRIPTION OF THE ARGUMENTS TO THIS
!  FUNCTION:
!      Boggs, P.T., Byrd, R.H., Rogers, J.E., and Schnabel, R.B. 1992.
!      User's Reference Guide for ODRPACK Version 2.01: Software for Weighted
!      Orthogonal Distance Regression.  Center for Computing and Applied
!      Mathematics, U.S. Department of Commerce, Gaithersburg, MD
!
!  N         NUMBER OF EXPERIMENTAL DATA, REFERED TO AS NUMBER OF OBSERVATIONS
!  M         NUMBER OF ELEMENTS PER OBSERVATION OF THE EXPLANATORY VARIABLE
!  NP        NUMBER OF PARAMETERS
!  NQ        NUMBER OF ELEMENTS PER OBSERVATION OF THE RESPONSE VARIABLE
```

```
!  LDN      .GE. N, USED TO DECLARE SIZE OF ARRAYS
!  LDM      .GE. M, USED TO DECLARE SIZE OF ARRAYS
!  LDNP     .GE. NP, USED TO DECLARE SIZE OF ARRAYS
!  BETA     PARAMETERS, THIS WILL BE DIFERENT FOR EVERY CALL TO FCN AS ODRPACK
!           VARIES THE PARAMETERS TO FIND THE OPTIMUM.
!  XPLUSD   CONTAINS A SET OF VALUES FOR THE EXPLANATORY VARIABLES PLUS SOME
!           UNKNOWN BUT REAL ERROR DELTA.  NOTE: ODRPACK VARIES DELTA TO FIND
!           THE ORTHOGONAL DISTANCE.
!  IFIXB    NOT USED, TELLS FCN WHICH BETAS ARE FIXED
!  IFIXX    NOT USED, TELLS FCN WHICH DELTAS ARE FIXED (EQUIVALENT TO WHICH
!           XPLUSD'S ARE FIXED)
!  LDIFX    LEADING DIMENSION OF IFIXX
!  IDEVAL   1 IF YOU WANT TO CALCULATE F.
!  F        THE RESPONSE VARIABLES AS CALCULATED BY FCN.
!  FJACB    NOT USED, OUTPUT FOR USER DEFINED JACOBIAN
!  FJACD    NOT USED, OUTPUT FOR USER DEFINED JACOBIAN
!  ISTOP    SET TO 1 AND RETURN IF PARAMETERS ARE INVALID.
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

      SUBROUTINE FCN(N,M,NP,NQ,             &
                     LDN,LDM,LDNP,          &
                     BETA,XPLUSD,           &
                     IFIXB,IFIXX,LDIFX,     &
                     IDEVAL,F,FJACB,FJACD,  &
                     ISTOP)
      USE TIMELAGMOD
      USE REAL_PRECISION

!  INPUT ARGUMENTS, NOT TO BE CHANGED BY THIS ROUTINE:
      INTEGER          :: IDEVAL,LDIFX,LDM,LDN,LDNP,M,N,NP,NQ
      REAL (KIND=R8)   :: BETA(NP),XPLUSD(LDN,M)
      INTEGER          :: IFIXB(NP),IFIXX(LDIFX,M)

!  OUTPUT ARGUMENTS:
      REAL (KIND=R8)   :: F(LDN,NQ),FJACB(LDN,LDNP,NQ),FJACD(LDN,LDM,NQ)
      INTEGER          :: ISTOP

!  LOCAL VARIABLES
      INTRINSIC EXP
      LOGICAL          :: BETAGTZERO
      INTEGER          :: L,I,LDSIZE,LSIZE,YNEQ(1)
      PARAMETER           (LDSIZE=5,LSIZE=5,YNEQ=(/5/))
      REAL (KIND=R8)   :: TEMPT, LOUT(LDSIZE,LSIZE)
      REAL (KIND=R8)   :: PARAMS(NP)
      REAL (KIND=R8)   :: YINIT(YNEQ(1))




!  CHECK FOR UNACCEPTABLE PARAMETER VALUES FOR THIS PROBLEM
      BETAGTZERO = .TRUE.
      DO I=1,NP
        BETAGTZERO = BETAGTZERO .AND. (BETA(I) .GE. 0.0_R8)
```

```
      END DO
      IF (.NOT. BETAGTZERO) THEN
         ISTOP = 1
         RETURN
      ELSE
         ISTOP = 0
      END IF


!  COMPUTE MODEL PREDICTED VALUES
      IF (MOD(IDEVAL,10).GE.1) THEN
         DO L = 1,NQ
```

```
!-----------------------------------------------------------------------------
!
!  Experiment:  J. Moore
!  Description: Time lags for MPF activation.
!
            PARAMS=BETA
            PARAMS ( 14 ) = 1.0_R8  ! CDC25TPARAM not used
            PARAMS ( 15 ) = 1.0_R8  ! WEETPARAM not used
            DO I = 1,4
               F(I,L) = TIMELAG(0,NP,PARAMS,XPLUSD(I,1))
            END DO
!
!-----------------------------------------------------------------------------
```

```
!-----------------------------------------------------------------------------
!
!  Experiment:  J. Moore
!  Description: Thresholds for MPF activation and inactivation.
!
            PARAMS=BETA
            PARAMS ( 14 ) = 1.0_R8  ! CDC25TPARAM not used
            PARAMS ( 15 ) = 1.0_R8  ! WEETPARAM not used
            TEMPT = THRESHOLD(0,NP,PARAMS)
            F(5,L) = TEMPT
            F(6,L) = TEMPT/ &
                     THRESHOLD(1,NP,PARAMS)
!
!-----------------------------------------------------------------------------
```

```
!-------------------------------------------------------------------------
!
!  Experiment:  Kumagai & Dumphy 1995 Fig 4b
!  Description: Timecourse data for MPF activation during mphase
!
            PARAMS=BETA
            PARAMS ( 13 ) = 1.0_R8  ! Dilution factor known for this exp
            PARAMS ( 14 ) = 1.0_R8  ! CDC25TPARAM not used
            PARAMS ( 15 ) = 1.0_R8  ! WEETPARAM not used
            CALL TIMECOURSE ( 1, YNEQ, YINIT, NP, PARAMS, 3, &
                              XPLUSD (  7: 9, 1 ), LDSIZE, LSIZE, LOUT )
            F (  7: 9, L ) = LOUT ( 1:3, 4 )
!
!-------------------------------------------------------------------------
```

```
!-------------------------------------------------------------------------
!
!  Experiment:  Kumagai & Dumphy 1995 Fig 4b
!  Description: Timecourse data for MPF activation during interphase
!
            PARAMS=BETA
            PARAMS ( 13 ) = 1.0_R8  ! Dilution factor known for this exp
            PARAMS ( 14 ) = 1.0_R8  ! CDC25TPARAM not used
            PARAMS ( 15 ) = 1.0_R8  ! WEETPARAM not used
            CALL TIMECOURSE ( 0, YNEQ, YINIT, NP, PARAMS, 3, &
                              XPLUSD ( 10:12, 1 ), LDSIZE, LSIZE, LOUT )
            F ( 10:12, L ) = LOUT ( 1:3, 4 )
!
!-------------------------------------------------------------------------
```

```
!-------------------------------------------------------------------------
!
!  Experiment:  Kumagai & Dumphy 1995 Fig 3c
!  Description: Timecourse data for MPF inactivation during mphase
!
            PARAMS=BETA
            PARAMS ( 13 ) = 1.0_R8  ! Dilution factor known for this exp
            PARAMS ( 14 ) = 1.0_R8  ! CDC25TPARAM not used
            PARAMS ( 15 ) = 1.0_R8  ! WEETPARAM not used
            CALL TIMECOURSE ( 1, YNEQ, YINIT, NP, PARAMS, 2, &
                              XPLUSD ( 13:14, 1 ), LDSIZE, LSIZE, LOUT )
            F ( 13:14, L ) = LOUT ( 1:2, 5 )
!
!-------------------------------------------------------------------------
```

```
!-------------------------------------------------------------------------------
!
!  Experiment:  Kumagai & Dumphy 1995 Fig 3c
!  Description: Timecourse data for MPF inactivation during interphase
!
            PARAMS=BETA
            PARAMS ( 13 ) = 1.0_R8  ! Dilution factor known for this exp
            PARAMS ( 14 ) = 1.0_R8  ! CDC25TPARAM not used
            PARAMS ( 15 ) = 1.0_R8  ! WEETPARAM not used
            CALL TIMECOURSE ( 0, YNEQ, YINIT, NP, PARAMS, 3, &
                              XPLUSD ( 15:17, 1 ), LDSIZE, LSIZE, LOUT )
            F ( 15:17, L ) = LOUT ( 1:3, 5 )
!
!-------------------------------------------------------------------------------
```

```
!-------------------------------------------------------------------------------
!
!  Experiment:  Kumagai & Dumphy 1992 Fig 10
!  Description: Cdc25 timecourse data for activation.
!
            PARAMS=BETA
            PARAMS ( 13 ) = 0.83_R8  ! Dilution factor known for this exp
            PARAMS ( 15 ) = 1.0_R8   ! WEETPARAM not used
            CALL SETTOTALCYCLIN ( 0.25_R8 )
            YINIT = (/ 0.25_R8, 0.0_R8, 0.0_R8, 1.0_R8, 0.0_R8 /)
            CALL TIMECOURSE ( -1, YNEQ, YINIT, NP, PARAMS, 4, &
                              XPLUSD ( 18:21, 1 ), LDSIZE, LSIZE, LOUT )
            F ( 18:21, L ) = LOUT ( 1:4, 2 )
!
!-------------------------------------------------------------------------------
```

```
!-------------------------------------------------------------------------------
!
!  Experiment:  Kumagai & Dumphy 1992 Fig 10
!  Description: Cdc25 timecourse data for inactivation.
!
            PARAMS=BETA
            PARAMS ( 13 ) = 0.83_R8  ! Dilution factor known for this exp
            PARAMS ( 15 ) = 1.0_R8   ! WEETPARAM not used
            CALL SETTOTALCYCLIN ( 0.0_R8 )
            YINIT = (/ 0.0_R8, 1.0_R8, 0.0_R8, 1.0_R8, 0.0_R8 /)
            CALL TIMECOURSE ( -1, YNEQ, YINIT, NP, PARAMS, 4, &
                              XPLUSD ( 22:25, 1 ), LDSIZE, LSIZE, LOUT )
            F ( 22:25, L ) = LOUT ( 1:4, 2 )
!
```

```
!-------------------------------------------------------------------------------




!-------------------------------------------------------------------------------
!
!  Experiment:  Tang Coleman Dumphy 1993 Fig 2
!  Description: Wee1 timecourse data for inactivation.
!
            PARAMS=BETA
            PARAMS ( 13 ) = 0.67_R8  ! Dilution factor known for this exp
            PARAMS ( 14 ) = 1.0_R8   ! CDC25TPARAM not used
            CALL SETTOTALCYCLIN ( 0.2537_R8 )
            YINIT = (/ 0.2537_R8, 1.0_R8, 1.0_R8, 1.0_R8, 0.0_R8 /)
            CALL TIMECOURSE ( -1, YNEQ, YINIT, NP, PARAMS, 4, &
                              XPLUSD ( 26:29, 1 ), LDSIZE, LSIZE, LOUT )
            F ( 26:29, L ) = LOUT ( 1:4, 3 )
!
!-------------------------------------------------------------------------------




!-------------------------------------------------------------------------------
!
!  Experiment:  Tang Coleman Dumphy 1993 Fig 2
!  Description: Wee1 timecourse data for activation.
!
            PARAMS=BETA
            PARAMS ( 13 ) = 0.67_R8  ! Dilution factor known for this exp
            PARAMS ( 14 ) = 1.0_R8   ! CDC25TPARAM not used
            CALL SETTOTALCYCLIN ( 0.0_R8 )
            YINIT = (/ 0.0_R8, 1.0_R8, 0.0_R8, 1.0_R8, 0.0_R8 /)
            CALL TIMECOURSE ( -1, YNEQ, YINIT, NP, PARAMS, 2, &
                              XPLUSD ( 30:31, 1 ), LDSIZE, LSIZE, LOUT )
            F ( 30:31, L ) = LOUT ( 1:2, 3 )
!
!-------------------------------------------------------------------------------


         END DO
      END IF

      RETURN
      END SUBROUTINE FCN



MODULE MODEL
```

```
USE REAL_PRECISION


PRIVATE





!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
!  USER DEFINES PARAMETERS, MODEL VARIABLES, AND OTHER VARIABLES NEED BY THE
!  USER ALL THESE ARE PRIVATE TO THIS CODE.  THE NAMES CAN BE ANY VALID
!  FORTRAN 90 IDENTIFIER.  BE SURE TO DEFINE VARIABLES FOR THE DERIVATIVES OF
!  THE MODEL VARIABLES, CALL THEM <MODEL_VAR_NAME>DOT IF YOU WISH.
!
!

! PARAMS
REAL (KIND=R8)   :: VCP, VCPP, VCPPP, &
                    VWP, VWPP, VWPPP, &
                    KMC, KMCR,        &
                    KMW, KMWR,        &
                    VC,  VW,          &
                    DILUTION,         &
                    CDC25TPARAM,      &
                    WEETPARAM

! VARS
REAL (KIND=R8)   :: M, C, W, L, L2

! VAR DERIVATIVES
REAL (KIND=R8)   :: MDOT, CDOT, WDOT, LDOT, L2DOT

! INDEPENDENT VARS
REAL (KIND=R8)   :: TOTALCYCLIN, MPFROOT

!
!
!  END USER DEFINED VARIABLES
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
!  THE USER DEFINES THE NAME LISTS.
!
!  THE NAMELISTS ARE USED TO READ THE MODEL INITIAL CONDITIONS AND PARAMETERS
!  FROM A NAMELIST FORMATED FILE
!
!

NAMELIST / MODELINIT   / M, C, W, L, L2 ! USER LIST MODEL VARIABLES HERE
NAMELIST / MODELPARAMS / VCP, VCPP, VCPPP, &
                         VWP, VWPP, VWPPP, &
                         KMC, KMCR,        &
                         KMW, KMWR,        &
                         VC,  VW,          &
                         TOTALCYCLIN,      &
                         DILUTION,         &
                         CDC25TPARAM,      &
                         WEETPARAM
                         ! USER LIST MODEL PARAMETERS HERE
!
!  END USER DEFINES NAME LISTS
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC




PUBLIC :: INITMODEL, MAPTOYS, MAPTOYDOTS, MAPTOLOCALS, MAPTOLOCALDOTS, &
          MAPBTOPARAMS, MAPPARAMSTOB, &
          FOUNDROOT, FEX, JEX, GEX


!  User defined public routines
PUBLIC :: SETTOTALCYCLIN, SETMPFROOT




CONTAINS




!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
!  INITIALIZE PARAMETERS TO MODEL AND INITIAL CONDITIONS OF MODEL
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
      SUBROUTINE INITMODEL ( FILEUNIT )
      INTEGER :: FILEUNIT

      READ ( UNIT=FILEUNIT, NML=MODELPARAMS )
      READ ( UNIT=FILEUNIT, NML=MODELINIT   )

      RETURN
      END SUBROUTINE INITMODEL









!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
!  USER DEFINES REVERSE MAPPING OF YDOT TO MODEL VARIABLE DERIVATIVES
!
!

      SUBROUTINE MAPTOYDOTS ( NEQ, YDOT )
      INTEGER          :: NEQ
      REAL (KIND=R8)   :: YDOT(NEQ)

      YDOT ( 1 ) = MDOT
      YDOT ( 2 ) = CDOT
      YDOT ( 3 ) = WDOT
      YDOT ( 4 ) = LDOT
      YDOT ( 5 ) = L2DOT

      RETURN
      END SUBROUTINE

!
!
!  END USER DEFINES REVERSE MAPPING OF YDOT
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC









!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
!
!  USER DEFINES MAPPING OF YDOT TO MODEL VARIABLS DERIVATIVES
!
!

      SUBROUTINE MAPTOLOCALDOTS ( NEQ, YDOT )
      INTEGER           :: NEQ
      REAL (KIND=R8)    :: YDOT(NEQ)

      MDOT  = YDOT ( 1 )
      CDOT  = YDOT ( 2 )
      WDOT  = YDOT ( 3 )
      LDOT  = YDOT ( 4 )
      L2DOT = YDOT ( 5 )

      RETURN
      END SUBROUTINE

!
!
!  END USER DEFINES MAPPING OF YDOT
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
!  USER DEFINES REVERSE MAPPING OF Y TO MODEL VARIABLES
!
!

      SUBROUTINE MAPTOYS ( NEQ, Y )
      INTEGER           :: NEQ
      REAL (KIND=R8)    :: Y(NEQ)

      Y ( 1 ) = M
      Y ( 2 ) = C
      Y ( 3 ) = W
      Y ( 4 ) = L
      Y ( 5 ) = L2

      RETURN
      END SUBROUTINE

!
```

```
!
!  END USER DEFINES REVERSE MAPPING OF Y
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
!  USER DEFINES MAPPING OF Y TO MODEL VARIABLES
!
!

      SUBROUTINE MAPTOLOCALS ( NEQ, Y )
      INTEGER          :: NEQ
      REAL (KIND=R8)   :: Y(NEQ)

      M  = Y ( 1 )
      C  = Y ( 2 )
      W  = Y ( 3 )
      L  = Y ( 4 )
      L2 = Y ( 5 )

      RETURN
      END SUBROUTINE

!
!
!  END USER DEFINES MAPPING OF Y
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
!  USER DEFINES MAPPING OF PARAMETERS TO BETA
!
!
```

```
      SUBROUTINE MAPPARAMSTOB ( BSIZE, B )
      INTEGER          :: BSIZE
      REAL (KIND=R8)   :: B(BSIZE)

      B (  1 ) = VCP
      B (  2 ) = VCPP
      B (  3 ) = VCPPP
      B (  4 ) = VWP
      B (  5 ) = VWPP
      B (  6 ) = VWPPP
      B (  7 ) = KMC
      B (  8 ) = KMCR
      B (  9 ) = KMW
      B ( 10 ) = KMWR
      B ( 11 ) = VC
      B ( 12 ) = VW
      B ( 13 ) = DILUTION
      B ( 14 ) = CDC25TPARAM
      B ( 15 ) = WEETPARAM

      END SUBROUTINE MAPPARAMSTOB

!
!
!  END USER DEFINES MAPPING OF PARAMETERS TO BETA
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
!  USER DEFINES MAPPING OF BETA TO PARAMETERS
!
!

      SUBROUTINE MAPBTOPARAMS ( BSIZE, B )
      INTEGER          :: BSIZE
      REAL (KIND=R8)   :: B(BSIZE)

      VCP         = B (  1 )
      VCPP        = B (  2 )
      VCPPP       = B (  3 )
      VWP         = B (  4 )
      VWPP        = B (  5 )
      VWPPP       = B (  6 )
```

```fortran
      KMC        = B (  7 )
      KMCR       = B (  8 )
      KMW        = B (  9 )
      KMWR       = B ( 10 )
      VC         = B ( 11 )
      VW         = B ( 12 )
      DILUTION   = B ( 13 )
      CDC25TPARAM = B ( 14 )
      WEETPARAM  = B ( 15 )

      END SUBROUTINE MAPBTOPARAMS

!
!
!  END USER DEFINES MAPPING OF BETA TO PARAMETERS
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```fortran
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
!  USER DEFINES HANDLING OF ROOT FINDING
!  THIS ROUTINE IS CALLED WHEN A ROOT (AS DESCRIBED IN GEX) IS FOUND.  THE
!  CALLER OF THIS CODE WILL CALL READY AND READYDOT TO DEFINE ALL MODEL
!  VARIABLES AND DERIVATIVES BEFORE CALLING THIS ROUTINE AND THEN IT WILL CALL
!  WRITEY AFTERWORDS TO DEFINE Y FOR THE ODE SOLVER.  NOTE THAT THE YDOTS WILL
!  NOT BE DEFINED FOR THE ODE SOLVER BY WRITEYDOT, PARTLY BECAUSE THE SOLVER CAN
!  CALL FEX TO GET THE YDOTS.
!
!

      SUBROUTINE FOUNDROOT ( TIME, NG, ROOTS)
      REAL (KIND=R8)   :: TIME      ! TIME ROOT EXISTS AT
      INTEGER          :: NG
      INTEGER          :: ROOTS(NG) ! WHICH ROOT EXISTS, SAME INDEX AS IN GEX

      END SUBROUTINE

!
!
!  END USER DEFINES HANDLING OF ROOT FINDING
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
!  USER DEFINES THE FUNCTIONS HE NEEDS
!
!



!
!
!  END USER DEFINED FUNCTIONS
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
!  USER DEFINES THE MODEL IN FEX
!
!

      SUBROUTINE FEX (NEQ, T, Y, YDOT)

      INTEGER :: NEQ
      REAL (KIND=R8)   :: T, Y(NEQ), YDOT(NEQ)

      REAL (KIND=R8)   :: KW, KC

      !  Scaled parameters:
      REAL (KIND=R8)   :: VCP_, VCPP_, VCPPP_, &
                          VWP_, VWPP_, VWPPP_, &
                          KMC_, KMCR_,         &
                          KMW_, KMWR_,         &
                          VC_,  VW_
      !  Concentration totals
      REAL (KIND=R8)   :: CDC2T, CDC25T, WEET

      CALL MAPTOLOCALS ( NEQ, Y )
```

```
      ! Constrain parameters
      VWPPP=VCPPP
      VW=VC

      ! Dilute concentrations
      CDC2T  = DILUTION
      CDC25T = DILUTION * CDC25TPARAM
      WEET   = DILUTION * WEETPARAM

      ! Scale parameters
      VCP_   = VCP   * CDC25T
      VCPP_  = VCPP  * CDC25T
      VCPPP_ = VCPPP / CDC25T
      VWP_   = VWP   * WEET
      VWPP_  = VWPP  * WEET
      VWPPP_ = VWPPP / WEET
      KMC_   = KMC   / CDC25T
      KMCR_  = KMCR  / CDC25T
      KMW_   = KMW   / WEET
      KMWR_  = KMWR  / WEET
      VC_    = VC    * CDC2T  / CDC25T
      VW_    = VW    * CDC2T  / WEET

      ! DEFINE AUXILARY FUNCTIONS
      KC = VCP_*(1-C) + VCPP_*C
      KW = VWP_*(1-W) + VWPP_*W

      ! DEFINE ODES
      MDOT = KC*(TOTALCYCLIN-M) - KW*M
      CDOT = VC_ * (                                          &
                 + ( M      * ( 1 - C ) ) / ( KMC_  + ( 1 - C ) ) &
                 - ( VCPPP_ *       C   ) / ( KMCR_ +       C   ) &
                 )
      WDOT = VW_ * (                                          &
                 - ( M      *       W   ) / ( KMW_  +       W   ) &
                 + ( VWPPP_ * ( 1 - W ) ) / ( KMWR_ + ( 1 - W ) ) &
                 )
      LDOT = -KC*L
      L2DOT = KW * ( 1 - L2 )

      !  END USER DEFINE MODEL

      CALL MAPTOYDOTS ( NEQ, YDOT )

      RETURN
      END SUBROUTINE FEX

!
!
!  END USER DEFINES THE MODEL
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
!  USER DEFINES THE JACOBIAN
!  IF YOU WANT YOU CAN SUPPLY ANALYTIC FORMULAS FOR THE JACOBIAN.  THIS WILL
!  MAKE THINGS MORE FAST.  BUT THERE ARE OTHER THINGS THAT NEED TO BE DONE TO
!  MAKE THIS FUNCTION WORK, SO I RECOMMEND LEAVING IT UNTIL LATER.
!
!

      SUBROUTINE JEX (NEQ, T, Y, ML, MU, PD, NRPD)

      INTEGER          :: NRPD, NEQ, ML, MU
      REAL (KIND=R8)   :: PD(NRPD,1), T, Y(NEQ)



      RETURN
      END SUBROUTINE JEX

!
!
!  END USER DEFINES THE JACOBIAN
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
!  USER DEFINES THE CONSTRAINTS FUNCTION
!  CURRENTLY UNSUPPORTED BY THE SIMULATOR.  IT ACTUALLY IS HALF SUPPORTED, BUT
!  THERE ARE BUGS WITH USING IT THAT MAKE IT COMPLETELY NON-FUNCTIONAL
!
!

      SUBROUTINE GEX (NEQ, T, Y, NG, GOUT)

      INTEGER          :: NG, NEQ
      REAL (KIND=R8)   T, Y, GOUT
```

```
      DIMENSION Y(NEQ), GOUT(NG)

      CALL MAPTOLOCALS ( NEQ, Y )

      ! USER DEFINES ROOTS
      ! DEFINE ROOTS OF THE ODES THAT THE USER WISHES TO FIND

      GOUT(1)=Y(1)-MPFROOT
      !  Without this if statement lsodar will not return when it finds a root.
      IF (ABS(GOUT(1)) .LE. 1.D-10) THEN
        GOUT(1)=0
      END IF


      ! END USER DEFINES ROOTS

      RETURN
      END SUBROUTINE GEX

!
!
!  END USER DEFINES THE CONSTRAINTS FUNCTION
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
!  User defined public subroutines and functions
!

      SUBROUTINE SETTOTALCYCLIN ( TC )
      REAL (KIND=R8)   :: TC
      TOTALCYCLIN=TC
      END SUBROUTINE SETTOTALCYCLIN


      SUBROUTINE SETMPFROOT ( MR )
      REAL (KIND=R8)   :: MR
      MPFROOT=MR
      END SUBROUTINE SETMPFROOT

!
!  End user defined public subroutines and functions
!
```

```
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

END MODULE MODEL

MODULE REAL_PRECISION
!  This module is for 64-bit floating point arithmetic
INTEGER, PARAMETER :: R8=SELECTED_REAL_KIND(13)
END MODULE REAL_PRECISION
```

Appendix B: Plotting Programs

The Fortran 90 code used for generating data for plots is in this section. This code was used to generate data for all figures in this thesis.

```
MODULE LSODARMOD




! LSODAR PARAMS, BACKUPS
INTEGER          :: NEQ_, ITOL_, ITASK_, ISTATE_, IOPT_, LRW_, &
                    IWORK_(:), LIW_, JT_, NG_, JROOT_(:)
DOUBLE PRECISION :: Y_(:), T_, TOUT_, RTOL_, ATOL_, RWORK_(:)
ALLOCATABLE      :: Y_, IWORK_, JROOT_, RWORK_
PRIVATE          :: NEQ_, ITOL_, ITASK_, ISTATE_, IOPT_, LRW_, &
                    IWORK_, LIW_, JT_, NG_, JROOT_,            &
                    Y_, T_, TOUT_, RTOL_, ATOL_, RWORK_



! LSODAR subroutine params
INTEGER          :: NEQ, ITOL, ITASK, ISTATE, IOPT, LRW, &
                    IWORK(:), LIW, JT, NG, JROOT(:)
DOUBLE PRECISION :: Y(:), T, TOUT, RTOL, ATOL, RWORK(:)
ALLOCATABLE      :: Y, IWORK, JROOT, RWORK

! Simulator related params
DOUBLE PRECISION :: TFINAL, TSTEP
INTEGER          :: MAXNG




CONTAINS




!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
!  NAME: READLSODARPARAMS
!
!  READS LSODAR PARAMS FROM A FILE.  THIS ROUTINE SHOULD ONLY BE CALLED ONCE
!  DURING EXECUTION OF A PROGRAM THAT USES THIS MODULE.  THE REASON BEING THAT
!  IT ALLOCATES THE LSODAR VARIABLES BLINDLY.
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

SUBROUTINE READLSODARPARAMS ( FILEUNIT )
```

```
  INTEGER :: FILEUNIT
  NAMELIST / LSODARPARAMS / NEQ, TFINAL, TSTEP, MAXNG, NG

  INTEGER :: I


  NEQ     = 0
  T       = 0.D0
  ITOL    = 1
  RTOL    = 1.D-6
  ATOL    = 1.D-6
  ITASK   = 1
  ISTATE  = 1
  IOPT    = 0
  LRW     = 215 ! enough for 10 eqs
  LIW     = 30  ! enough for 10 eqs
  JT      = 2
  NG      = 0
  MAXNG   = 0


  READ  ( UNIT=FILEUNIT, NML=LSODARPARAMS )

  MAXNG   = MAX ( NG, MAXNG )  ! USER MAY ALWAYS WANT TO LOOK FOR ROOTS

  LRW     = 22 + NEQ * MAX ( 16, NEQ + 9 ) + 3 * MAXNG
  LIW     = 20 + NEQ

  ALLOCATE ( Y (NEQ), IWORK (LIW), RWORK (LRW), JROOT (MAXNG) )
  ALLOCATE ( Y_(NEQ), IWORK_(LIW), RWORK_(LRW), JROOT_(MAXNG) )

  DO I = 1, NEQ
    Y(I) = 0.0
  END DO

  CALL SAVELSODARPARMS()

END SUBROUTINE READLSODARPARAMS




!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
!  NAME: SAVELSODARPARMS
!
!  PUTS THE PUBLIC LSODAR PARAMS IN THE PRIVATE PARAMS FOR LATER RECAL/RESET
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

SUBROUTINE SAVELSODARPARMS ()
```

```
NEQ_          = NEQ
ITOL_         = ITOL
ITASK_        = ITASK
ISTATE_       = ISTATE
IOPT_         = IOPT
LRW_          = LRW
IWORK_(1:LIW) = IWORK(1:LIW)
LIW_          = LIW
JT_           = JT
NG_           = NG
JROOT_(1:NG)  = JROOT(1:NG)
Y_(1:NEQ)     = Y(1:NEQ)
T_            = T
TOUT_         = TOUT
RTOL_         = RTOL
ATOL_         = ATOL
RWORK_(1:LRW) = RWORK(1:LRW)

END SUBROUTINE




!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
!  NAME: LOADLSODARPARMS
!
!  PUTS THE PRIVATE LSODAR PARAMS IN THE PUBLIC PARAMS TO RESET FOR A NEW RUN
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

SUBROUTINE LOADLSODARPARMS ()

NEQ           = NEQ_
ITOL          = ITOL_
ITASK         = ITASK_
ISTATE        = ISTATE_
IOPT          = IOPT_
LRW           = LRW_
IWORK (1:LIW) = IWORK_(1:LIW)
LIW           = LIW_
JT            = JT_
NG            = NG_
JROOT (1:NG)  = JROOT_(1:NG)
Y (1:NEQ)     = Y_(1:NEQ)
T             = T_
TOUT          = TOUT_
RTOL          = RTOL_
ATOL          = ATOL_
RWORK (1:LRW) = RWORK_(1:LRW)

END SUBROUTINE
```

```
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
!  NAME: RUNLSODAR
!
!  RUN LSODAR AND REPORT ANY ERROR MESSAGE.
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

SUBROUTINE RUNLSODAR ()
  USE MODEL
  EXTERNAL lsodar

  CALL lsodar(FEX,NEQ,Y,T,TOUT,ITOL,RTOL,ATOL,ITASK,ISTATE, &
              IOPT,RWORK,LRW,IWORK,LIW,JEX,JT,GEX,NG,JROOT)

  IF (ISTATE .EQ. -1) THEN
    WRITE(6,*) 'LSODAR: excess work done on this call (perhaps wrong jt)'
  ELSE IF (ISTATE .EQ. -2) THEN
    WRITE(6,*) 'LSODAR: excess accuracy requested (tolerances too small)'
  ELSE IF (ISTATE .EQ. -3) THEN
    WRITE(6,*) 'LSODAR: illegal input detected (see printed message).'
  ELSE IF (ISTATE .EQ. -4) THEN
    WRITE(6,*) 'LSODAR: repeated error test failures (check all inputs).'
  ELSE IF (ISTATE .EQ. -5) THEN
    WRITE(6,*) "LSODAR: repeated convergence failures (perhaps bad jacobian &
               &supplied or wrong choice of jt or tolerances)."
  ELSE IF (ISTATE .EQ. -6) THEN
    WRITE(6,*) 'LSODAR: error weight became zero during problem. (solution &
               &component i vanished, and atol or atol(i) = 0.)'
  ELSE IF (ISTATE .EQ. -7) THEN
    WRITE(6,*) 'LSODAR: work space insufficient to finish (see messages).'
  ELSE IF (ISTATE .LT. 0) THEN
    WRITE(6,*) 'LSODAR: some unknown error'
  END IF

END SUBROUTINE




END MODULE LSODARMOD




PROGRAM SIM

USE MODEL
```

```
USE LSODARMOD

INTEGER          :: I, MAXI
INTEGER          :: FILEUNIT
DOUBLE PRECISION :: YDOTTEMP(:)
DOUBLE PRECISION :: TTEMP     ! TTEMP IS NOT NECESSARY NOW, BUT IF I WANT TO
                              ! RESUME ON ISTATE .EQ. 2 THEN I NEED IT
ALLOCATABLE      :: YDOTTEMP



FILEUNIT = 5
OPEN ( UNIT=FILEUNIT, FILE='model.nml' )



CALL READLSODARPARAMS ( FILEUNIT )
CALL INITMODEL        ( FILEUNIT )
CALL MAPTOYS ( NEQ, Y )
ALLOCATE ( YDOTTEMP(NEQ) )



CLOSE ( UNIT=FILEUNIT )



MAXI  = TFINAL/TSTEP
TOUT  = 0.0
TTEMP = 0.0

CALL WRITEVARS ( TOUT, NEQ, Y )

DO I = 1, MAXI
  TTEMP = TTEMP + TSTEP
  TOUT  = TTEMP
  CALL RUNLSODAR ( )
! DO WHILE ( ISTATE .EQ. 3  )
!   CALL FEX ( NEQ, T, Y, YDOTTEMP )
!   CALL MAPTOLOCALDOTS ( NEQ, YDOTTEMP )
!   CALL MAPTOLOCALS    ( NEQ, Y        )
!   CALL FOUNDROOT ( T, NG, JROOT )
!   CALL MAPTOYS ( NEQ, Y )
!   Y(1)=0.05
!   WRITE ( 6, * ) "RESETTING ISTATE"
!   CALL WRITEVARS ( T, NEQ, Y )
!   ISTATE = 2
!   CALL RUNLSODAR ( )
! END DO
  CALL WRITEVARS ( TOUT, NEQ, Y )
END DO

IF ( TFINAL .GT. TOUT ) THEN
  TOUT = TFINAL
  CALL RUNLSODAR ( )
  CALL WRITEVARS ( TOUT, NEQ, Y )
END IF
```

```
CONTAINS




SUBROUTINE WRITEVARS ( TOUT, NEQ, Y )

  INTEGER          :: NEQ
  DOUBLE PRECISION :: TOUT, Y(NEQ)

  INTEGER          :: I

  WRITE ( 6, '(E)', ADVANCE='NO' ) TOUT
  DO I = 1, NEQ
    WRITE ( 6, '(E)', ADVANCE='NO' ) Y(I)
  END DO
  WRITE ( 6, * )

END SUBROUTINE




END PROGRAM SIM




!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
!  CALCULATES ACTIVE MPF VS TIME FOR SPECIFIC PARAMETERS.  THIS CAN BE USED TO
!  GENERATE DATA TO PLOT (INSTEAD OF MATHEMATICA).
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      USE TIMELAGMOD

      INTEGER, PARAMETER :: PSIZE=15
      DOUBLE PRECISION PARAMS
      DIMENSION PARAMS(PSIZE)
      INTEGER I
      DOUBLE PRECISION TOUT, M

!  READ PARAMETERS FROM FILE

      OPEN (UNIT=5,FILE='params')
```

```
      READ (5,FMT=*) (PARAMS(I),I=1,PSIZE)

      TOUT=0.001D0
      DO I=1,100
        M=MPF(0,PSIZE,PARAMS,0.2D0,TOUT)
        WRITE(6,21)TOUT,M
        TOUT=TOUT+1D-2
      END DO

   21 FORMAT(2E16.6)
      END
```

```
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
!  CALCULATES TIMELAG VS CYCLIN FOR SPECIFIC PARAMETERS.  THIS CAN BE USED TO
!  GENERATE DATA TO PLOT (INSTEAD OF MATHEMATICA).
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      USE TIMELAGMOD

      INTEGER, PARAMETER :: PSIZE=15
      DOUBLE PRECISION PARAMS
      DIMENSION PARAMS(PSIZE)
      INTEGER I
      DOUBLE PRECISION TC,MA,MI

!  READ PARAMETERS FROM FILE

      OPEN (UNIT=5,FILE='params')
      READ (5,FMT=*) (PARAMS(I),I=1,PSIZE)

      TC=0.01D0
      DO I=1,500
        MI=MPF_INF(0,PSIZE,PARAMS,TC)
        MA=MPF_INF(1,PSIZE,PARAMS,TC)
        WRITE(6,21)TC,MI,MA
        TC=TC+1D-3
      END DO

   21 FORMAT(3E16.6)
      END
```

```
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
!  CALCULATES TIMELAG VS CYCLIN FOR SPECIFIC PARAMETERS.  THIS CAN BE USED TO
!  GENERATE DATA TO PLOT (INSTEAD OF MATHEMATICA).
```

```
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      USE TIMELAGMOD

      INTEGER, PARAMETER :: PSIZE=15
      DOUBLE PRECISION Y,PARAMS,T
      DIMENSION PARAMS(PSIZE)
      INTEGER I
      DOUBLE PRECISION TC,M

!  READ PARAMETERS FROM FILE

      OPEN (UNIT=5,FILE='params')
      READ (5,FMT=*) (PARAMS(I),I=1,PSIZE)

      Y=0.01D0
      DO I=1,100
        T=TIMELAG(0,PSIZE,PARAMS,Y)
        IF (T .LT. 180D0) THEN
          WRITE(6,21)Y,T
        ENDIF
        Y=Y+1D-2
      END DO

  21  FORMAT(E14.6,' ',E20.12)
      END
```

Appendix C: ODRPACK Report File

The ODRPACK report file for the final run reported in this thesis. The parameter estimates in Table 4 come from this file.

```
*****************************************************
* ODRPACK VERSION 2.01 OF 06-19-92 (DOUBLE PRECISION) *
*****************************************************


*** INITIAL SUMMARY FOR FIT BY METHOD OF ODR ***

--- PROBLEM SIZE:
          N =    31          (NUMBER WITH NONZERO WEIGHT =    31)
         NQ =     1
          M =     1
         NP =    15          (NUMBER UNFIXED =     8)

--- CONTROL VALUES:
        JOB = 00000
            = ABCDE, WHERE
                     A=0 ==> FIT IS NOT A RESTART.
                     B=0 ==> DELTAS ARE INITIALIZED TO ZERO.
                     C=0 ==> COVARIANCE MATRIX WILL BE COMPUTED USING
                             DERIVATIVES RE-EVALUATED AT THE SOLUTION.
                     D=0 ==> DERIVATIVES ARE ESTIMATED BY FORWARD DIFFERENCES.
                     E=0 ==> METHOD IS EXPLICIT ODR.
     NDIGIT =     6          (SUPPLIED BY USER)
     TAUFAC =     1.00E+00

--- STOPPING CRITERIA:
      SSTOL =     1.49E-08   (SUM OF SQUARES STOPPING TOLERANCE)
     PARTOL =     3.67E-11   (PARAMETER STOPPING TOLERANCE)
      MAXIT = 10000          (MAXIMUM NUMBER OF ITERATIONS)

--- INITIAL WEIGHTED SUM OF SQUARES        =                    1.10253720E+02
       SUM OF SQUARED WEIGHTED DELTAS     =   0.00000000E+00
       SUM OF SQUARED WEIGHTED EPSILONS   =   1.10253720E+02
```

--- FUNCTION PARAMETER SUMMARY:

INDEX (K)	BETA(K)	FIXED (IFIXB)	SCALE (SCLB)	DERIVATIVE STEP SIZE (STPB)
1	1.70000000E-02	NO	5.88235294E+01	1.00000E-05
2	1.70000000E-01	NO	5.88235294E+00	1.00000E-05
3	5.00000000E-02	NO	2.00000000E+01	1.00000E-05
4	1.00000000E-02	NO	1.00000000E+02	1.00000E-05
5	1.00000000E+00	NO	1.00000000E+00	1.00000E-05
6	5.00000000E-02	YES	2.00000000E+01	1.00000E-05
7	1.00000000E-01	YES	1.00000000E+01	1.00000E-05
8	1.00000000E+00	YES	1.00000000E+00	1.00000E-05
9	1.00000000E-01	YES	1.00000000E+01	1.00000E-05
10	1.00000000E+00	YES	1.00000000E+00	1.00000E-05
11	1.00000000E+00	YES	1.00000000E+00	1.00000E-05
12	1.00000000E+00	YES	1.00000000E+00	1.00000E-05
13	1.00000000E+00	NO	1.00000000E+00	1.00000E-05
14	2.00000000E+00	NO	5.00000000E-01	1.00000E-05
15	2.00000000E+00	NO	5.00000000E-01	1.00000E-05

--- EXPLANATORY VARIABLE AND DELTA WEIGHT SUMMARY:

INDEX (I,J)	X(I,J)	DELTA(I,J)	FIXED (IFIXX)	SCALE (SCLD)	WEIGHT (WD)	DERIVATIVE STEP SIZE (STPD)
1,1	2.000E-01	0.000E+00	NO	5.00E+00	2.50E+01	1.00000E-05
N,1	1.500E+01	0.000E+00	NO	6.67E-02	4.40E-03	1.00000E-05

--- RESPONSE VARIABLE AND EPSILON ERROR WEIGHT SUMMARY:

INDEX (I,L)	Y(I,L)	WEIGHT (WE)
1,1	4.500E+01	4.938E-04
N,1	1.000E+00	1.000E+00

*** ITERATION REPORTS FOR FIT BY METHOD OF ODR ***

IT. NUM.	CUM. NO. FN EVALS	WEIGHTED SUM-OF-SQS	ACT. REL. SUM-OF-SQS REDUCTION	PRED. REL. SUM-OF-SQS REDUCTION	TAU/PNORM	G-N STEP	BETA INDEX ---------->	VALUE		
1	13	9.06121E+00	9.1781E-01	7.8548E-01	5.282E-01	NO	1 TO 3	1.70606613E-02	1.99601934E-01	7.52592226E-02
							4 TO 6	9.89197381E-03	6.24492080E-01	5.00000000E-02
							7 TO 9	1.00000000E-01	1.00000000E+00	1.00000000E-01
							10 TO 12	1.00000000E+00	1.00000000E+00	1.00000000E+00
							13 TO 15	1.21796964E+00	1.93968700E-02	6.56670828E-01
2	23	5.73115E+00	3.6751E-01	5.5967E-01	4.422E-01	NO	1 TO 3	2.11511992E-02	1.79723943E-01	7.64097792E-02
							4 TO 6	5.04960648E-03	8.25416845E-01	5.00000000E-02
							7 TO 9	1.00000000E-01	1.00000000E+00	1.00000000E-01
							10 TO 12	1.00000000E+00	1.00000000E+00	1.00000000E+00
							13 TO 15	2.24089707E+00	7.74889315E-02	2.97277901E-01
3	34	4.31942E+00	2.4633E-01	3.1834E-01	4.677E-02	NO	1 TO 3	2.13336576E-02	1.83237213E-01	7.70109904E-02
							4 TO 6	5.01355955E-03	7.92638476E-01	5.00000000E-02
							7 TO 9	1.00000000E-01	1.00000000E+00	1.00000000E-01
							10 TO 12	1.00000000E+00	1.00000000E+00	1.00000000E+00
							13 TO 15	2.24739529E+00	2.07841519E-01	5.46129357E-01
4	45	2.28829E+00	4.7023E-01	3.5280E-01	5.072E-02	NO	1 TO 3	2.12166113E-02	1.83455263E-01	7.61256380E-02
							4 TO 6	4.97997219E-03	7.79412437E-01	5.00000000E-02
							7 TO 9	1.00000000E-01	1.00000000E+00	1.00000000E-01
							10 TO 12	1.00000000E+00	1.00000000E+00	1.00000000E+00
							13 TO 15	2.24843163E+00	5.01384036E-01	3.85140644E-01
5	55	2.19405E+00	4.1186E-02	2.5529E-01	2.541E-02	NO	1 TO 3	2.10563244E-02	1.83199555E-01	7.50505958E-02
							4 TO 6	4.83398377E-03	7.91832332E-01	5.00000000E-02
							7 TO 9	1.00000000E-01	1.00000000E+00	1.00000000E-01
							10 TO 12	1.00000000E+00	1.00000000E+00	1.00000000E+00
							13 TO 15	2.24530336E+00	5.75884116E-01	5.80519111E-01
6	65	2.02301E+00	7.7956E-02	2.6049E-01	2.551E-02	NO	1 TO 3	2.09653314E-02	1.83166264E-01	7.48056140E-02
							4 TO 6	4.80944317E-03	7.83388651E-01	5.00000000E-02
							7 TO 9	1.00000000E-01	1.00000000E+00	1.00000000E-01
							10 TO 12	1.00000000E+00	1.00000000E+00	1.00000000E+00

							13 TO 15	2.24436821E+00	5.69327876E-01	4.00073472E-01
7	76	1.92001E+00	5.0914E-02	8.8383E-02	1.158E-02	NO	1 TO 3	2.09419086E-02	1.82976317E-01	7.46100203E-02
							4 TO 6	4.80134898E-03	7.86515963E-01	5.00000000E-02
							7 TO 9	1.00000000E-01	1.00000000E+00	1.00000000E-01
							10 TO 12	1.00000000E+00	1.00000000E+00	1.00000000E+00
							13 TO 15	2.24401316E+00	5.68085775E-01	4.79521962E-01
8	86	1.90341E+00	8.6429E-03	1.2542E-02	1.164E-02	NO	1 TO 3	2.07590253E-02	1.82433811E-01	7.35132841E-02
							4 TO 6	4.74872439E-03	7.85324717E-01	5.00000000E-02
							7 TO 9	1.00000000E-01	1.00000000E+00	1.00000000E-01
							10 TO 12	1.00000000E+00	1.00000000E+00	1.00000000E+00
							13 TO 15	2.23973219E+00	5.61819565E-01	4.57018957E-01
9	96	1.89333E+00	5.2984E-03	1.1201E-02	1.169E-02	NO	1 TO 3	2.05833689E-02	1.81968944E-01	7.26668683E-02
							4 TO 6	4.68637129E-03	7.87674468E-01	5.00000000E-02
							7 TO 9	1.00000000E-01	1.00000000E+00	1.00000000E-01
							10 TO 12	1.00000000E+00	1.00000000E+00	1.00000000E+00
							13 TO 15	2.23323840E+00	5.56494997E-01	4.87806894E-01
10	106	1.89087E+00	1.3001E-03	1.3623E-02	5.875E-03	NO	1 TO 3	2.03898425E-02	1.81595680E-01	7.19279522E-02
							4 TO 6	4.62440249E-03	7.86234650E-01	5.00000000E-02
							7 TO 9	1.00000000E-01	1.00000000E+00	1.00000000E-01
							10 TO 12	1.00000000E+00	1.00000000E+00	1.00000000E+00
							13 TO 15	2.22577766E+00	5.51601037E-01	4.44946759E-01
11	116	1.88101E+00	5.2136E-03	1.4070E-02	5.877E-03	NO	1 TO 3	2.03566266E-02	1.81494766E-01	7.18186914E-02
							4 TO 6	4.61032267E-03	7.88161161E-01	5.00000000E-02
							7 TO 9	1.00000000E-01	1.00000000E+00	1.00000000E-01
							10 TO 12	1.00000000E+00	1.00000000E+00	1.00000000E+00
							13 TO 15	2.22408280E+00	5.50748687E-01	4.82855301E-01
12	126	1.87896E+00	1.0873E-03	6.4275E-03	2.947E-03	NO	1 TO 3	2.02518390E-02	1.81377861E-01	7.14893074E-02
							4 TO 6	4.57243460E-03	7.86670474E-01	5.00000000E-02
							7 TO 9	1.00000000E-01	1.00000000E+00	1.00000000E-01
							10 TO 12	1.00000000E+00	1.00000000E+00	1.00000000E+00
							13 TO 15	2.21934189E+00	5.48438431E-01	4.54390587E-01
13	136	1.87415E+00	2.5602E-03	4.6937E-03	2.948E-03	NO	1 TO 3	2.02309050E-02	1.81321687E-01	7.14305205E-02
							4 TO 6	4.56367460E-03	7.87625286E-01	5.00000000E-02
							7 TO 9	1.00000000E-01	1.00000000E+00	1.00000000E-01
							10 TO 12	1.00000000E+00	1.00000000E+00	1.00000000E+00
							13 TO 15	2.21823985E+00	5.47968596E-01	4.72984065E-01
14	151	1.84880E+00	1.3530E-02	1.8643E-02	9.284E-02	NO	1 TO 3	1.71126964E-02	1.82395685E-01	6.95424905E-02
							4 TO 6	2.91054181E-03	7.55005874E-01	5.00000000E-02
							7 TO 9	1.00000000E-01	1.00000000E+00	1.00000000E-01

							10 TO 12	1.00000000E+00	1.00000000E+00	1.00000000E+00
							13 TO 15	2.10373162E+00	5.33593660E-01	4.63585356E-01
15	161	1.84702E+00	9.5971E-04	1.1674E-02	4.656E-02	NO	1 TO 3	1.66549294E-02	1.82972635E-01	7.14248382E-02
							4 TO 6	2.33119444E-04	7.36632212E-01	5.00000000E-02
							7 TO 9	1.00000000E-01	1.00000000E+00	1.00000000E-01
							10 TO 12	1.00000000E+00	1.00000000E+00	1.00000000E+00
							13 TO 15	2.10164949E+00	5.43602335E-01	5.10471223E-01
16	171	1.83523E+00	6.3823E-03	1.1663E-02	4.658E-03	NO	1 TO 3	1.66470030E-02	1.82953472E-01	7.13981974E-02
							4 TO 6	2.26875076E-04	7.35361857E-01	5.00000000E-02
							7 TO 9	1.00000000E-01	1.00000000E+00	1.00000000E-01
							10 TO 12	1.00000000E+00	1.00000000E+00	1.00000000E+00
							13 TO 15	2.10164655E+00	5.43394339E-01	4.80846607E-01
17	182	1.83279E+00	1.3309E-03	1.4569E-03	8.948E-03	NO	1 TO 3	1.65821382E-02	1.82513691E-01	7.10341539E-02
							4 TO 6	2.85528423E-05	7.36214336E-01	5.00000000E-02
							7 TO 9	1.00000000E-01	1.00000000E+00	1.00000000E-01
							10 TO 12	1.00000000E+00	1.00000000E+00	1.00000000E+00
							13 TO 15	2.09821615E+00	5.41127348E-01	4.85347543E-01
18	192	1.83264E+00	8.0893E-05	1.6638E-04	8.949E-04	NO	1 TO 3	1.65770563E-02	1.82495188E-01	7.10135605E-02
							4 TO 6	1.78680313E-05	7.35977707E-01	5.00000000E-02
							7 TO 9	1.00000000E-01	1.00000000E+00	1.00000000E-01
							10 TO 12	1.00000000E+00	1.00000000E+00	1.00000000E+00
							13 TO 15	2.09800080E+00	5.40998974E-01	4.81470382E-01
19	202	1.83251E+00	7.2623E-05	1.0289E-04	8.951E-04	NO	1 TO 3	1.65691097E-02	1.82456188E-01	7.09884325E-02
							4 TO 6	8.00497759E-06	7.36084925E-01	5.00000000E-02
							7 TO 9	1.00000000E-01	1.00000000E+00	1.00000000E-01
							10 TO 12	1.00000000E+00	1.00000000E+00	1.00000000E+00
							13 TO 15	2.09746770E+00	5.40840088E-01	4.84371181E-01
20	214	1.83245E+00	3.1675E-05	4.5252E-05	3.242E-04	NO	1 TO 3	1.65676027E-02	1.82452439E-01	7.09840114E-02
							4 TO 6	6.35119381E-06	7.35985744E-01	5.00000000E-02
							7 TO 9	1.00000000E-01	1.00000000E+00	1.00000000E-01
							10 TO 12	1.00000000E+00	1.00000000E+00	1.00000000E+00
							13 TO 15	2.09738621E+00	5.40813726E-01	4.82727594E-01
21	226	1.83243E+00	1.0939E-05	1.4523E-05	1.313E-04	NO	1 TO 3	1.65669463E-02	1.82449100E-01	7.09821270E-02
							4 TO 6	2.80164930E-06	7.35990880E-01	5.00000000E-02
							7 TO 9	1.00000000E-01	1.00000000E+00	1.00000000E-01
							10 TO 12	1.00000000E+00	1.00000000E+00	1.00000000E+00
							13 TO 15	2.09734891E+00	5.40802126E-01	4.82804720E-01
22	236	1.83243E+00	9.4366E-07	4.9122E-06	6.567E-06	NO	1 TO 3	1.65669272E-02	1.82449006E-01	7.09820716E-02
							4 TO 6	2.39832126E-06	7.35990933E-01	5.00000000E-02

							7 TO 9	1.00000000E-01	1.00000000E+00	1.00000000E-01
							10 TO 12	1.00000000E+00	1.00000000E+00	1.00000000E+00
							13 TO 15	2.09734780E+00	5.40801784E-01	4.82805897E-01
23	246	1.83243E+00	4.8768E-07	2.6431E-06	3.284E-06	NO	1 TO 3	1.65669175E-02	1.82448958E-01	7.09820439E-02
							4 TO 6	2.18896757E-06	7.35990970E-01	5.00000000E-02
							7 TO 9	1.00000000E-01	1.00000000E+00	1.00000000E-01
							10 TO 12	1.00000000E+00	1.00000000E+00	1.00000000E+00
							13 TO 15	2.09734725E+00	5.40801614E-01	4.82806472E-01
24	256	1.83243E+00	2.5624E-07	6.3944E-07	3.284E-06	NO	1 TO 3	1.65669080E-02	1.82448910E-01	7.09820168E-02
							4 TO 6	2.08978812E-06	7.35990999E-01	5.00000000E-02
							7 TO 9	1.00000000E-01	1.00000000E+00	1.00000000E-01
							10 TO 12	1.00000000E+00	1.00000000E+00	1.00000000E+00
							13 TO 15	2.09734670E+00	5.40801445E-01	4.82807029E-01
25	266	1.83243E+00	4.7951E-08	1.9856E-07	1.642E-06	NO	1 TO 3	1.65668786E-02	1.82448764E-01	7.09819321E-02
							4 TO 6	2.13927665E-06	7.35991085E-01	5.00000000E-02
							7 TO 9	1.00000000E-01	1.00000000E+00	1.00000000E-01
							10 TO 12	1.00000000E+00	1.00000000E+00	1.00000000E+00
							13 TO 15	2.09734500E+00	5.40800920E-01	4.82808702E-01
26	286	1.83232E+00	5.6859E-05	6.0215E-05	1.363E-03	NO	1 TO 3	1.65494748E-02	1.82390748E-01	7.09441560E-02
							4 TO 6	7.29153170E-07	7.35840471E-01	5.00000000E-02
							7 TO 9	1.00000000E-01	1.00000000E+00	1.00000000E-01
							10 TO 12	1.00000000E+00	1.00000000E+00	1.00000000E+00
							13 TO 15	2.09628372E+00	5.40568948E-01	4.81930046E-01
27	296	1.83232E+00	1.0755E-06	6.0845E-06	6.813E-06	NO	1 TO 3	1.65494604E-02	1.82390666E-01	7.09441313E-02
							4 TO 6	3.04135678E-07	7.35841249E-01	5.00000000E-02
							7 TO 9	1.00000000E-01	1.00000000E+00	1.00000000E-01
							10 TO 12	1.00000000E+00	1.00000000E+00	1.00000000E+00
							13 TO 15	2.09628283E+00	5.40568750E-01	4.81943294E-01
28	316	1.83232E+00	-2.6654E-10	8.9080E-09	2.337E-09	NO	1 TO 3	1.65494604E-02	1.82390666E-01	7.09441313E-02
							4 TO 6	3.04135678E-07	7.35841249E-01	5.00000000E-02
							7 TO 9	1.00000000E-01	1.00000000E+00	1.00000000E-01
							10 TO 12	1.00000000E+00	1.00000000E+00	1.00000000E+00
							13 TO 15	2.09628283E+00	5.40568750E-01	4.81943294E-01

*** FINAL SUMMARY FOR FIT BY METHOD OF ODR ***

--- STOPPING CONDITIONS:
        INFO =   1 ==> SUM OF SQUARES CONVERGENCE.
       NITER =   28   (NUMBER OF ITERATIONS)

```
        NFEV =   325          (NUMBER OF FUNCTION EVALUATIONS)
       IRANK =     0          (RANK DEFICIENCY)
       RCOND =     5.00E-02   (INVERSE CONDITION NUMBER)
       ISTOP =     0          (RETURNED BY USER FROM SUBROUTINE FCN)

--- FINAL WEIGHTED SUMS OF SQUARES       =                    1.83232175E+00
        SUM OF SQUARED WEIGHTED DELTAS   =    4.03577980E-01
        SUM OF SQUARED WEIGHTED EPSILONS =    1.42874377E+00

--- RESIDUAL STANDARD DEVIATION          =                    2.82251950E-01
        DEGREES OF FREEDOM               =   23

--- ESTIMATED BETA(J), J = 1, ..., NP:

                   BETA      S.D. BETA    ---- 95%  CONFIDENCE INTERVAL ----

      1   1.65494604E-02     2.0361E-02    -2.55711989E-02 TO  5.86701198E-02
      2   1.82390666E-01     4.6904E-02     8.53589459E-02 TO  2.79422387E-01
      3   7.09441313E-02     2.1818E-02     2.58084963E-02 TO  1.16079766E-01
      4   3.04135678E-07     5.7798E-03    -1.19563844E-02 TO  1.19569927E-02
      5   7.35841249E-01     4.1150E-01    -1.15442357E-01 TO  1.58712486E+00
      6   5.00000000E-02          FIXED
      7   1.00000000E-01          FIXED
      8   1.00000000E+00          FIXED
      9   1.00000000E-01          FIXED
     10   1.00000000E+00          FIXED
     11   1.00000000E+00          FIXED
     12   1.00000000E+00          FIXED
     13   2.09628283E+00     9.0929E-01     2.15234139E-01 TO  3.97733152E+00
     14   5.40568750E-01     1.6307E-01     2.03233382E-01 TO  8.77904119E-01
     15   4.81943294E-01     1.5827E-01     1.54520523E-01 TO  8.09366064E-01

--- ESTIMATED EPSILON(I) AND DELTA(I,*), I = 1, ..., N:

        I    EPSILON(I,1)      DELTA(I,1)

        1  1.16072434E+01  7.44459971E-02
        2  7.11419794E+00  6.01962188E-02
```

3	6.31155464E+00	7.66861544E-02
4	2.70771306E+00	6.76445668E-02
5	-4.13612016E-02	0.00000000E+00
6	-5.74496110E-02	0.00000000E+00
7	-4.91588978E-02	-4.48425078E-02
8	-1.78747712E-02	-9.99018796E-02
9	7.12329168E-03	4.01621387E-01
10	-3.25231901E-02	-2.12075620E-03
11	-6.38049989E-02	-1.60943042E-02
12	2.55218584E-02	3.25978831E-02
13	1.05071028E-02	-6.55130450E-04
14	4.23479137E-02	-4.05696889E-02
15	-2.08907381E-01	1.27991980E-01
16	-5.15104616E-02	3.07204241E-02
17	-7.70638693E-06	9.82536799E-09
18	-3.25516721E-01	2.14511159E-01
19	-1.54500625E-01	1.99440319E-01
20	-7.03866721E-02	3.40450471E-02
21	-5.75822756E-02	-2.97667602E-03
22	-1.12810722E-01	-3.25567461E-01
23	-7.79681415E-02	-1.45081991E+00
24	1.06326038E-03	5.37493554E-01
25	8.04761673E-03	1.36001889E-01
26	-2.41830827E-02	-5.59955147E-02
27	1.51465000E-01	9.32553834E-01
28	1.25213209E-01	5.80376526E-01
29	1.10826860E-01	3.02756215E-01
30	5.94488603E-02	-7.04167144E-01
31	-1.33397541E-01	5.85475781E-01

Appendix D: Input to EST

The input used for EST.

```
&ODRPACKPARAMS

MAXIT = 10000

IPRINT = 6616

NP = 15
BETA  (1:15) = 0.017, 0.17, 0.05, 0.01, 1, 0.05, 0.1, 1, 0.1, 1, 1, 1, 1, 2, 2
IFIXB (1:15) =     1,    1,    1,    1, 1,    0,   0, 0, 0,   0, 0, 0, 1, 1, 1

N = 31

X (1:31,1) = 0.20, 0.25, 0.30, 0.50,    0, 0,    2,    4,    8,   2,   4,    8,
4, 16, 2, 4, 16, 1.25, 2.5, 5, 10,    5,  10,  20, 40,   2, 5, 7, 10, 7.5, 15

Y (1:31,1) =   45,   40,   30,   20, 0.18, 3, 0.75, 0.51, 0.21, 1.0, 1.0, 0.85,
0,  0, 1, 1,  1,  0.8, 0.9, 1,  1, 0.75, 0.5, 0.1,  0, 0.5, 0, 0,  0, 0.5,  1

WD (1:31,1,1) =       25,      16,    11.11,      4,      1,      1, 0.25,
0.0625, 0.015625, 0.25, 0.0625, 0.015625, 0.0625, 0.0039, 0.25, 0.0625, 0.0039,
0.64, 0.16, 0.04, 0.01, 0.04, 0.01, 0.0025, 6.25E-4, 0.25, 0.04, 0.0204, 0.01,
0.01778, 0.0044

WE (1:31,1,1) = 4.938E-4, 6.25E-4, 1.111E-3, 2.4E-3,240, 0.8888, 1.78, 3.8447,
22.67574,    1,      1, 1.384083,      1,      1,    1,      1,      1, 1.5625,
1.2346, 1, 1, 1.778, 4, 100, 10, 4, 10, 10, 10, 4, 1

IFIXX (1:31,1) = 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1

/
```

Appendix E: Plots for the Three Equation Model

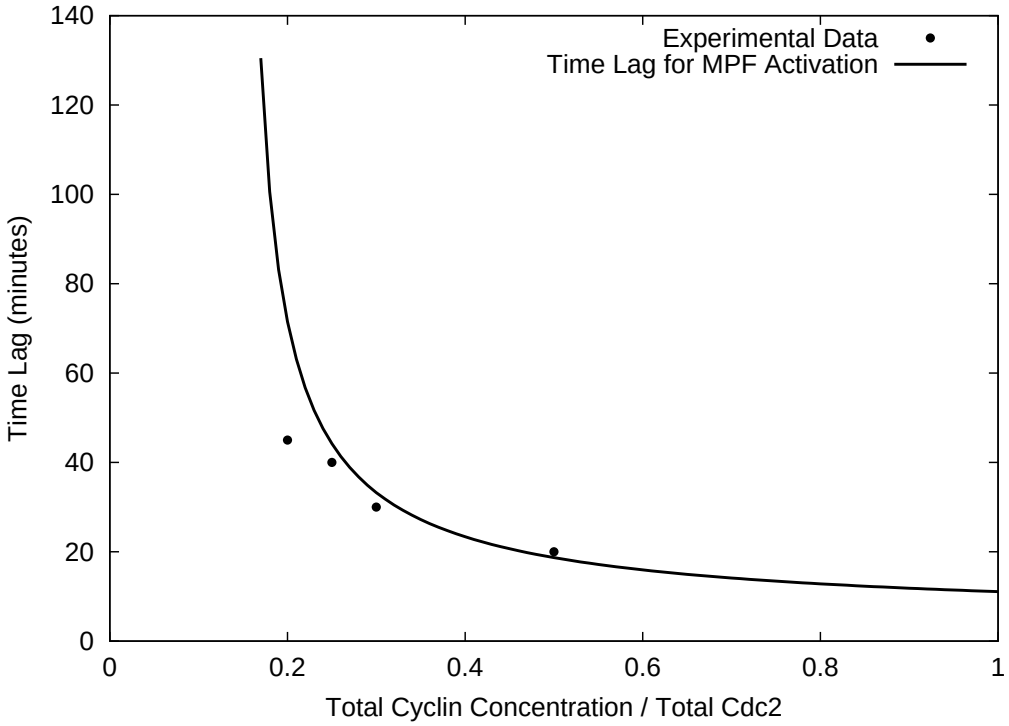

FIG. 8. Time lag for MPF activation versus total cyclin [Moore, 1997].

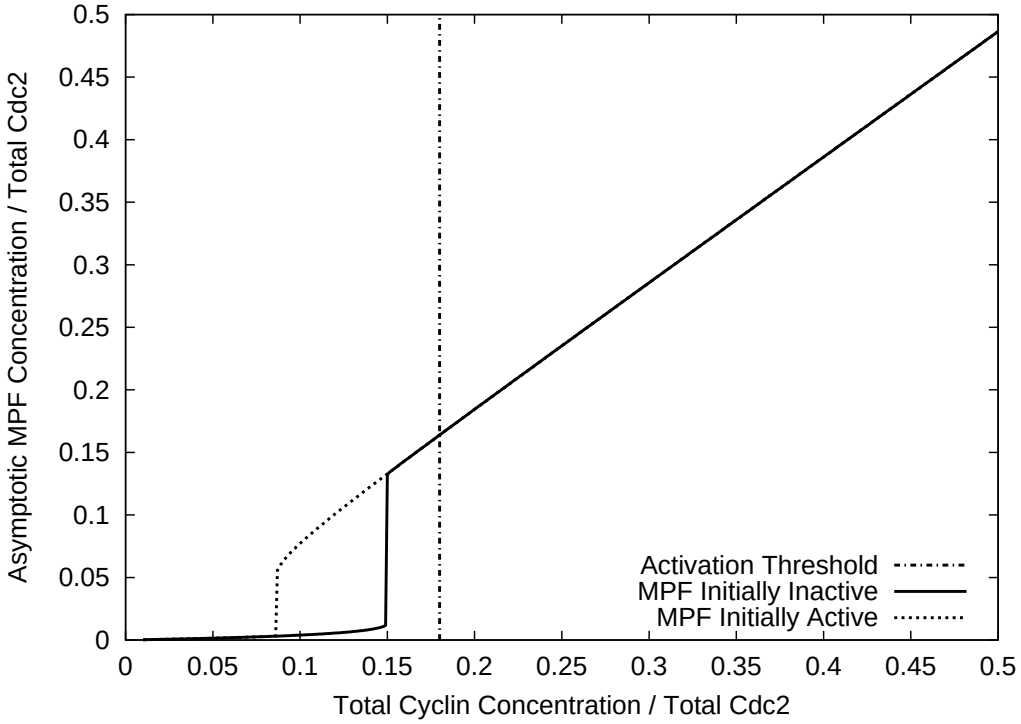

FIG. 9. Thresholds for MPF activation [Moore, 1997].

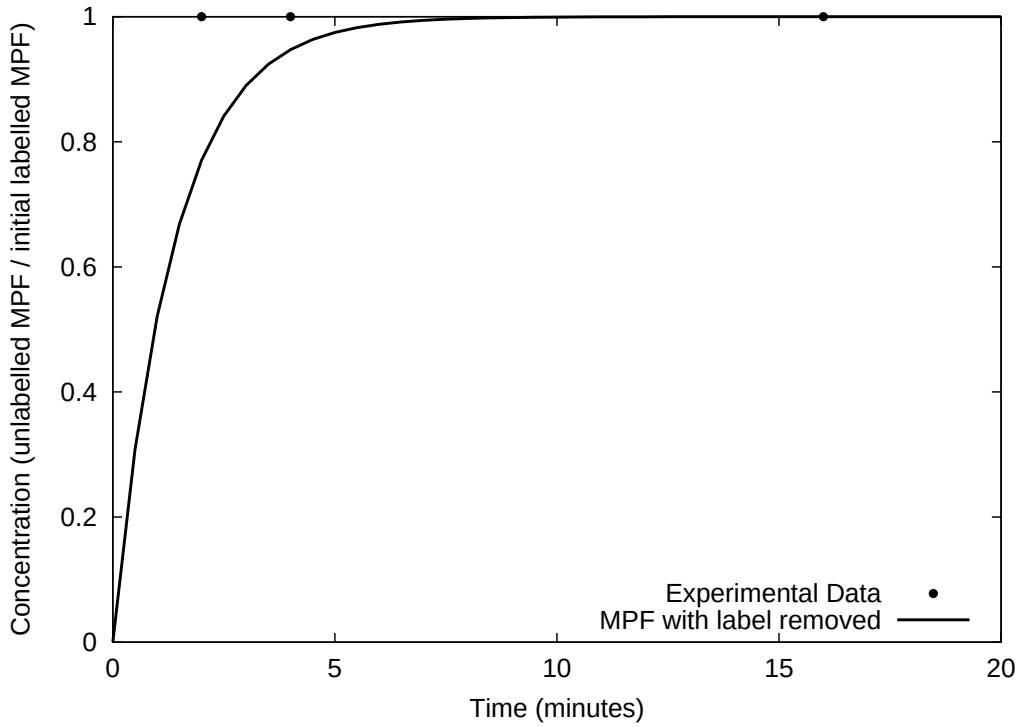

FIG. 10. MPF activation during interphase [Kumagai and Dunphy, 1995].

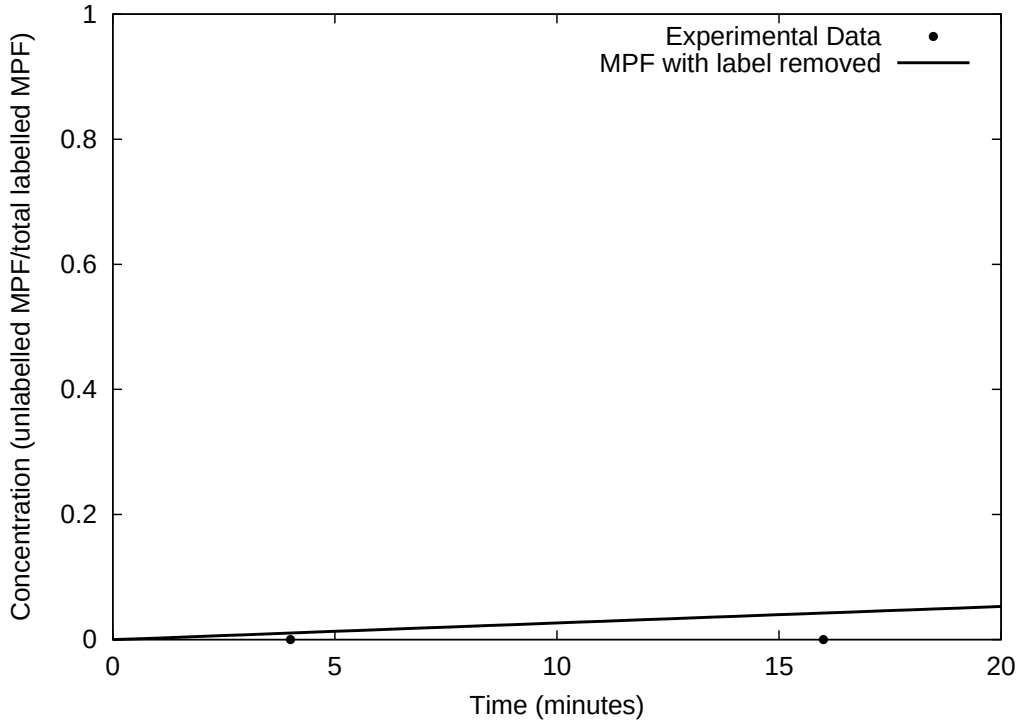

FIG. 11. MPF activation during M-phase [Kumagai and Dunphy, 1995].

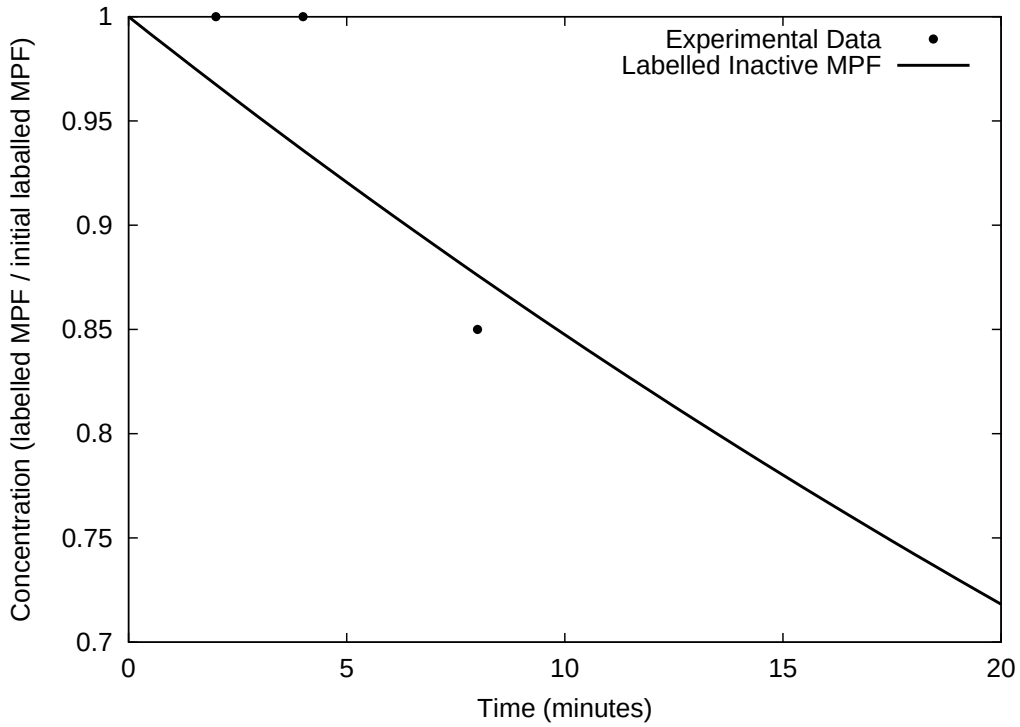

FIG. 12. MPF inactivation during interphase [Kumagai and Dunphy, 1995].

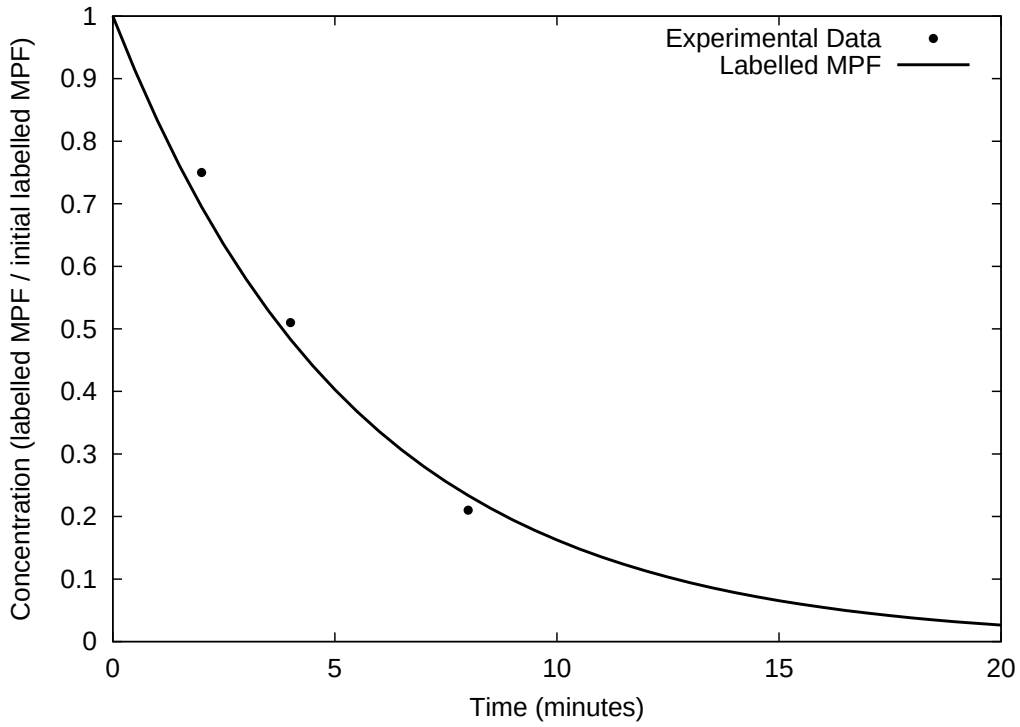

FIG. 13. MPF inactivation during M-phase [Kumagai and Dunphy, 1995].

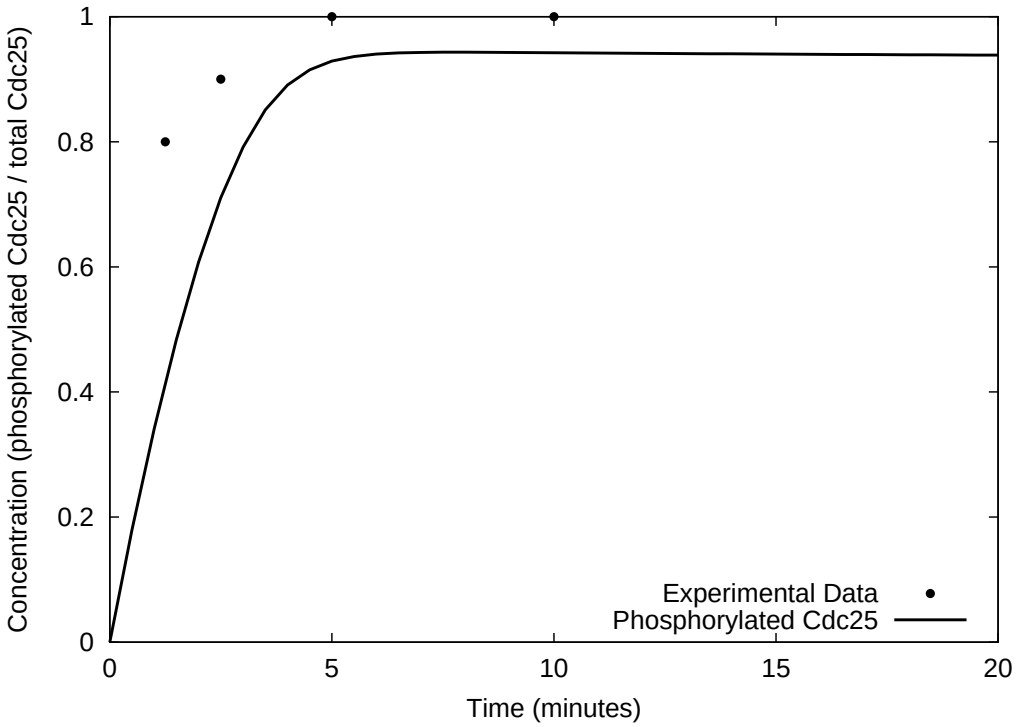

FIG. 14. Cdc25 activation [Kumagai and Dunphy, 1992].

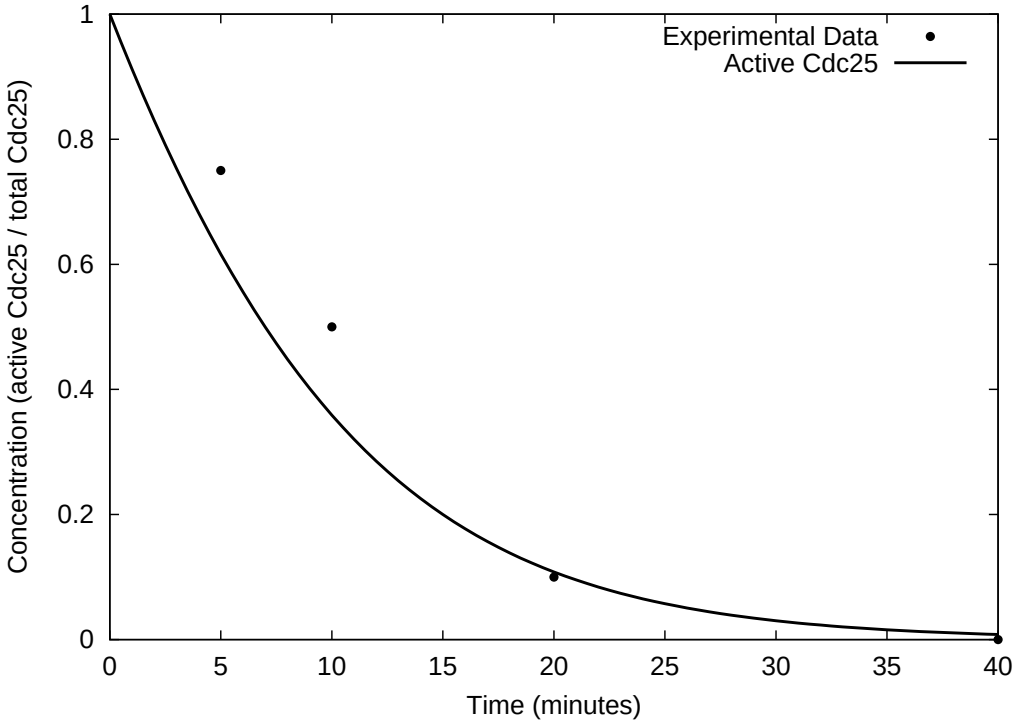

FIG. 15. Cdc25 inactivation [Kumagai and Dunphy, 1992].

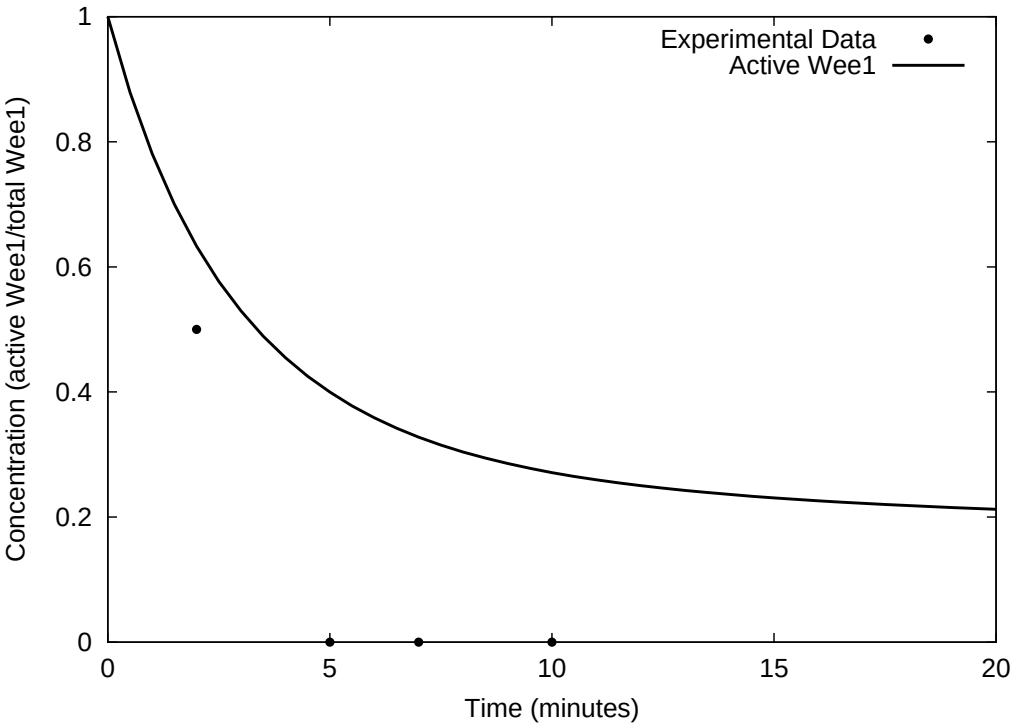

FIG. 16. Wee1 inactivation [Tang et al, 1993].

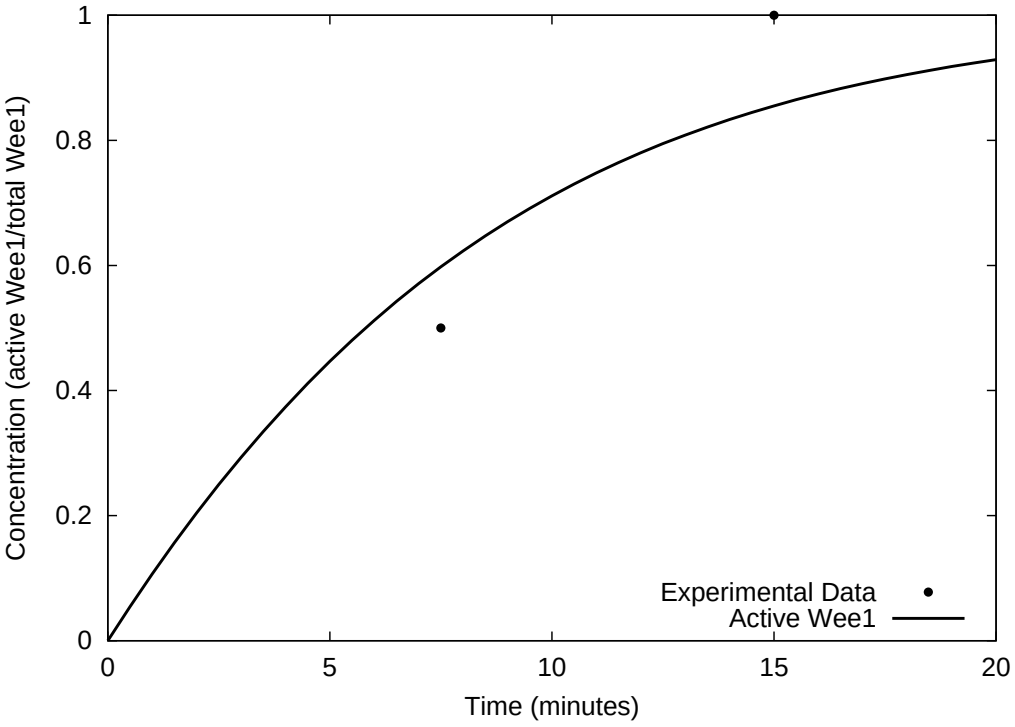

FIG. 17. Wee1 activation [Tang et al, 1993].

Appendix F: Experimental Data for the Three Equation Model.

This table contains the data used for the parameter estimation of the three equation model.

Table 8. Experimental data used in the three equation model.

Data	Reference	Description
(0.20,45) (0.25,40) (0.30,30) (0.50,20)	Private communication with Moore, [1997]	(Total Cyclin, Time Lag) Time lags for MPF activation.
(0,0.18) (0,3)	Private communication with Moore, [1997]	(Dummy,Threshold/Threshold Ratio) Thresholds for MPF activation and inactivation
(2,0.75) (4,0.51) (8,0.21)	Figure 4.b. from Kumagai and Dunphy, [1995]	(Time, Inactive MPF) MPF activation during M-phase
(2,1.0) (4,1.0) (8,0.85)	Figure 4.b. from Kumagai and Dunphy, [1995]	(Time, Inactive MPF) MPF activation during interphase
(4,0) (16,0)	Figure 3.c. from Kumagai and Dunphy, [1995]	(Time, Inactive MPF) MPF inactivation during M-phase
(2,1) (4,1) (16,1)	Figure 3.c. from Kumagai and Dunphy, [1995]	(Time, Inactive MPF) MPF inactivation during interphase
(1.25,0.8) (2.5,0.9) (5,1) (10,1)	Figure 10. from Kumagai and Dunphy, [1992]	(Time, Active Cdc25) Cdc25 activation
(5,0.75) (10,0.5) (20,0.1) (40,0)	Figure 10. from Kumagai and Dunphy, [1992]	(Time, Active Cdc25) Cdc25 inactivation

Data	Reference	Description
(2,0.5) (5,0) (7,0) (10,0)	Figure 2. from Tang et al, [1993]	(Time, Active Wee1) Wee1 inactivation
(7.5,0.5) (15,1)	Figure 2. from Tang et al, [1993]	(Time, Active Wee1) Wee1 activation

VITA.

Jason W. Zwolak was born on May 14, 1977, in Hartford, Connecticut. He earned a Bachelor of computer engineering degree from Virginia Tech in 1999. In August 1999 he began graduate work in the Computer Science department at Virginia Tech. He completed his masters in October of 2001 and plans to continue graduate school until he receives a PhD.