ON-LINE ESTIMATION OF THE OXYGEN-MASS-TRANSFER COEFFICIENT AND OTHER STATE VARIABLES IN A CHEMOSTAT

Joseph J. Vallino

Research Report Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Chemical Engineering

Department of Chemical Engineering
California Institute of Technology
Pasadena, California

May 20, 1985

ABSTRACT

An adaptive Kalman filter is used to estimate the values of the oxygen-mass-transfer coefficient, the biomass, substrate, and dissolved oxygen concentrations, the specific growth rate, and the substrate, and oxygen yield coefficients in a chemostat under steady state and transient conditions from on-line measurements of the oxygen consumption rate, the carbon dioxide production rate, and the dissolved oxygen concentration. An adaptive procedure is incorporated into the filter algorithm to account for modeling errors. The true states and measurements are simulated by a computer so that the performance of the filter can be evaluated.

It is found that the filter algorithm developed works quite well for estimating all states except the substrate concentration. The substrate concentration estimate becomes chaotic at low concentrations due to the nonobservability of this state.

CONTENTS

1. INTRODUCTION 1

2. KALMAN FILTER EQUATIONS 3
  2.1 Discrete State Equation 5
    2.1.1 The State Transition Matrix 7
  2.2 Outline Of The Filter Algorithm 8
  2.3 State Equations Used In The Algorithm 10
    2.3.1 Choice Of The State Equations 11
  2.4 Measurement Equations 12
    2.4.1 Smoothing Of The Measurements 13
  2.5 Observability 14

3. ADAPTIVE STATE-NOISE COVARIANCE MATRIX 16
  3.1 Algorithm For Determing $\mathbf{Q}(t_k)$ 17
    3.1.1 Improvement On $\mathbf{Q}(t_k)$ 19
  3.2 Determination Of The Noise Transition Matrix 19
    3.2.1 Transition Matrix For The Chemostat Equations 20
    3.2.2 Adaptive Ramp Equations 22

4. SIMULATIONS 23
  4.1 Program 23
  4.2 State Equations Used In The Simulations 24
    4.2.1 Simulation 1 24
    4.2.2 Simulation 2 25
    4.2.3 Simulation 3 25
  4.3 Data Used For The Simulations 26

5. RESULTS AND DISCUSSION 28
  5.1 Simulation 1 Results 28
  5.2 Simulation 2 Results 28
  5.3 Simulation 3 Results 30
  5.4 Sensitivity 30
  5.5 Conclusions 31

6. FUTURE RESEARCH 32

7. NOMENCLATURE 33

8. REFERENCES 36

9. APPENDIX I 38
  9.1 Simulation 1 Figures 38
  9.2 Simulation 2 Figures 45
  9.3 Simulation 3 Figures 52
  9.4 Simulated Measured Variables 59

10. APPENDIX II 64
  10.1 Simulation Program 65
  10.2 Kalman Filter Program 71
  10.3 Matrix Exponential Program 80

1. INTRODUCTION

Knowledge of the oxygen-mass-transfer coefficient, k_La, in a biochemical reactor is necessary to determine the point at which the oxygen requirements of the microbes can no longer be meet, and the process should be shut down. However, there exist no accurate methods for the determination of k_La on-line or even off-line, although several papers have been published [1-4]. Most of these current methods incorporate some kind of transient studies, which are not suitable for on-line estimation. Not only is k_La important, but such variables as biomass and substrate concentrations are also required for control purposes. Some of these variables can be measured off-line; however, the turn around time for off-line analysis is too long for control implementation. It is the goal of this study to devise an implementable procedure for the on-line estimation of k_La and other variables in a chemostat under steady state and transient conditions.

Any process can be completely described by a set of parameters, which k_La is a member of, called *state* variables. A procedure for estimating these state variables was devised by Kalman [5] for discrete systems, and later improved by Kalman and Bucy [6] for continuous systems. This procedure is ideally suited for digital computers. Called a Kalman filter, the differential equations that model the process are solved to obtain estimates of the states. Since no system can be modeled exactly, measurements of the system provide the necessary information needed to update the estimates. This updating allows the estimates to converge to their true values even when given poor initial values and in the presence of modeling errors. The filter algorithm also incorporates some adaptive capabilities so that the state equations used will work on practically any chemostat. It should also be noted that the filter acts as an observer. That is, state variables that are not directly measured can still be estimated from information provided by the state equations. Consequently, the Kalman filter is much more informative than simple averaging filters, which can only estimate variables that are directly measurable.

In this study the state variables of interest are: the oxygen-mass-transfer coefficient, $k_L a$, the biomass, substrate, and dissolved oxygen concentrations, b, s and, $[O_2]_d$, the substrate and oxygen yield coefficients, Y_s, and Y_{O_2}, and the specific growth rate, μ. The governing Kalman filter equations are described in the next section.

The filter algorithm developed in this paper has the further advantage of being virtually independent of the microbe. And although the algorithm is devised for a chemostat, the governing equations can easily be changed to accommodate batch and fed batch reactors. It is assumed that only biomass is produced in the chemostat; however, the program can be adjusted to accommodate product formation as well. For a FORTRAN listing of the filter program see Appendix II.

To check the performance of the filter algorithm, the chemostat is simulated by a computer so that the state estimates can easily be compared to their true values. Basically the simulation generates data for the true states for comparison, as well measurement information that would be obtained by on-line analysis for the filter. The program that simulates the chemostat is also presented in Appendix II.

2. KALMAN FILTER EQUATIONS

All state variables can be described by the following nonlinear differential vector equation

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t),t) + \mathbf{G}(t)\mathbf{w}(t). \tag{1}$$

The state, $\mathbf{x}(t)$, is a continuous function of time, where as the measurement vector equation,

$$\mathbf{y}(t_k) = \mathbf{h}(\mathbf{x}(t_k),t_k) + \mathbf{v}(t_k) \tag{2}$$

is discrete in time due to the inherent nature of sampling. Both the state and the measured variables are corrupted by noise, $\mathbf{w}(t)$, and $\mathbf{v}(t_k)$, which is assumed to have a gaussian distribution and a zero mean. The state noise, $\mathbf{w}(t)$, is described by the state-noise spectral density matrix $\mathbf{Q}(t)$ which is given by

$$\varepsilon\{\mathbf{w}(t)\mathbf{w}^T(\tau)\} = \mathbf{Q}(t)\,\delta(t-\tau). \tag{3}$$

The measurement-noise covariance matrix, $\mathbf{R}(t_k)$, describes the measurement noise, $\mathbf{v}(t_k)$, and is given by

$$\varepsilon\{\mathbf{v}(t_k)\mathbf{v}^T(t_l)\} = \mathbf{R}(t_k)\,\delta_{kl} \tag{4}$$

where $\varepsilon\{\ \}$ is the expectation operator, $\delta(t-\tau)$ is the Dirac delta function, and δ_{kl} is the Kronecker delta. It is further assumed that the state and measurement noise inputs are uncorrelated so that

$$\varepsilon\{\mathbf{w}(t)\mathbf{v}^T(t_k)\} = 0 \qquad \text{for all } t \text{ and } k. \tag{5}$$

Since the true state, $\mathbf{x}(t)$, is unknown it is desired to produce an estimate of the state, $\hat{\mathbf{x}}(t)$, which minimizes the cost function J given by

$$J = \left[\mathbf{y}(t_k) - \mathbf{h}(\hat{\mathbf{x}}(t_k),t_k)\right]^T \left[\mathbf{y}(t_k) - \mathbf{h}(\hat{\mathbf{x}}(t_k),t_k)\right]. \tag{6}$$

where J simply represents the sum of the error, in the estimate, squared.

Based on this minimization technique it is possible to obtain a set of recursive equations which produce the best estimate of the state for a continuous-discrete system. The derivation of these equations is too involved to explain here and can be found in Jazwinski [7] or Gelb [8].

The extended Kalman filter equations consist of the prediction between samples

$$\dot{\hat{\mathbf{x}}}(t) = \mathbf{f}(\hat{\mathbf{x}}(t),t) \tag{7}$$

$$\dot{\mathbf{P}}(t) = \mathbf{F}(\hat{\mathbf{x}}(t),t)\,\mathbf{P}(t) + \mathbf{P}(t)\,\mathbf{F}^T(\hat{\mathbf{x}}(t),t) + \mathbf{G}(t)\,\mathbf{Q}(t)\,\mathbf{G}^T(t) \tag{8}$$

and the updating there of at an observation

$$\hat{\mathbf{x}}(t_{k+1}\,|\,t_{k+1}) = \hat{\mathbf{x}}(t_{k+1}\,|\,t_k) + \mathbf{K}(t_{k+1})\left[\mathbf{y}(t_{k+1}) - \mathbf{h}(\hat{\mathbf{x}}(t_{k+1}\,|\,t_k),t_{k+1})\right] \tag{9}$$

$$\mathbf{P}(t_{k+1}\,|\,t_{k+1}) = \left[\mathbf{I} - \mathbf{K}(t_{k+1})\,\mathbf{H}(t_{k+1};\hat{\mathbf{x}}(t_{k+1}\,|\,t_k))\right]\mathbf{P}(t_{k+1}\,|\,t_k) \tag{10}$$

where the Kalman gain is given by

$$\mathbf{K}(t_{k+1}) = \mathbf{P}(t_{k+1}\,|\,t_k)\,\mathbf{H}^T(t_{k+1};\hat{\mathbf{x}}(t_{k+1}\,|\,t_k))$$

$$\times\left[\mathbf{H}(t_{k+1};\hat{\mathbf{x}}(t_{k+1}\,|\,t_k))\,\mathbf{P}(t_{k+1}\,|\,t_k)\,\mathbf{H}^T(t_{k+1};\hat{\mathbf{x}}(t_{k+1}\,|\,t_k)) + \mathbf{R}(t_k)\right]^{-1} \tag{11}$$

and

$$\mathbf{F}(\hat{\mathbf{x}}(t),t) = \left.\frac{\partial \mathbf{f}(\mathbf{x}(t),t)}{\partial \mathbf{x}(t)}\right|_{\mathbf{x}(t)=\hat{\mathbf{x}}(t)} \tag{12}$$

$$\mathbf{H}(t_{k+1};\hat{\mathbf{x}}(t_{k+1}\,|\,t_k)) = \left.\frac{\partial \mathbf{h}(\mathbf{x}(t_{k+1}),t_{k+1})}{\partial \mathbf{x}(t_{k+1})}\right|_{\mathbf{x}(t_{k+1})=\hat{\mathbf{x}}(t_{k+1}\,|\,t_k)}. \tag{13}$$

The P matrix is the covariance of the state-estimate error and is given by

$$\mathbf{P}(t_k\,|\,t_k) = \varepsilon\,\{\tilde{\mathbf{x}}(t_k\,|\,t_k)\,\tilde{\mathbf{x}}^T(t_k\,|\,t_k)\} \tag{14}$$

where

$$\tilde{\mathbf{x}}(t_k \mid t_k) = \hat{\mathbf{x}}(t_k \mid t_k) - \mathbf{x}(t_k). \tag{15}$$

It should be noted that the magnitude of the Kalman gain matrix $\mathbf{K}$ is predominately effected by the state error covariance matrix, $\mathbf{P}(t_k \mid t_k)$. Simply stated $\mathbf{P}$ represents the magnituted of the assumed error in the state estimate with respect to the true state.

The complicated nomenclature used in the Kalman filter equations is necessary to express the relationships between the discrete measurements and the continuous state variables. In this manuscript the following conventions will be used. The expression $\hat{\mathbf{x}}(t_k \mid t_k)$ represents $\hat{\mathbf{x}}$ at time t_k evaluated from the measurements, $\mathbf{y}(t_k)$, taken at time t_k. The expression $\hat{\mathbf{x}}(t_{k+1} \mid t_k)$ represents $\hat{\mathbf{x}}$ at time t_{k+1} but *predicted* from the measurements that were taken at time t_k. These conventions also hold for $\mathbf{P}$.

2.1 Discrete State Equation

It can be seen from equations (7) and (8) that the number of differential equations that must be solved grows as the square of the state dimension since $\mathbf{P}$ is an $M \times M$ matrix. For large systems this can cause convergence problems for numerical integrators as well as an unacceptably long computation time. To increase computational speed and reduce convergence problems, equation (8) can be put in discrete form as follows

$$\mathbf{P}(t_{k+1} \mid t_k) = \Phi[t_{k+1}, t_k]\,\mathbf{P}(t_k \mid t_k)\,\Phi^T[t_{k+1}, t_k]$$
$$+ \Gamma[t_{k+1}, t_k]\,\mathbf{Q}(t_k)\,\Gamma^T[t_{k+1}, t_k] \tag{16}$$

where the state transition matrix, Φ, is given by

$$\Phi[t_{k+1}, t_k] = \exp\left[\mathbf{F}(\hat{\mathbf{x}}(t_k \mid t_k), t_k)\,(t_{k+1} - t_k)\right] \tag{17}$$

and

$$\Gamma[t_{k+1},t_k]\mathbf{Q}(t_k)\Gamma^T[t_{k+1},t_k] = \int_{t_k}^{t_{k+1}} \Phi[t_{k+1},\tau]\mathbf{G}(\tau)\mathbf{Q}(\tau)\mathbf{G}^T(\tau)\Phi^T[t_{k+1},\tau]d\tau. \qquad (18)$$

The discrete noise vector $\mathbf{w}(t_k)$, is described by the state-noise covariance matrix, $\mathbf{Q}(t_k)$, which is given by

$$\varepsilon\{\mathbf{w}(t_k)\mathbf{w}^T(t_l)\} = \mathbf{Q}(t_k)\,\delta_{kl}. \qquad (19)$$

Note that the state-noise covariance matrix, $\mathbf{Q}(t_k)$, is different from the state-noise spectral density matrix, $\mathbf{Q}(t)$, which is used for continuous systems. It is assumed that $\mathbf{F}(\hat{\mathbf{x}}(t_k \mid t_k),t_k)$ is invariant over the sampling period.

The continuous state equation, (1), can also be put in discrete form;

$$\tilde{\mathbf{x}}(t_{k+1}) = \Phi[t_{k+1},t_k]\tilde{\mathbf{x}}(t_k) + \Gamma[t_{k+1},t_k]\mathbf{w}(t_k) \qquad (20)$$

however, this is not usually necessary since the continuous state equation is usually integrated easily and is more accurate. Although less accurate, it turns out that the loss in precision by discretizing $\mathbf{P}(t)$ is warranted by the increase in computational speed. Also, in discretizing the equations there is a further advantage. The state transition matrix, $\Gamma[t_{k+1},t_k]$, distributes the noise to the state variables where as $\mathbf{G}(t)$, in equation (1), distributes the noise to the *derivative* of the state variables. Since noise addition is usually determined by intuition, it turns out that it is easier to construct $\Gamma[t_{k+1},t_k]$.

Consequently, in the filter algorithm prediction across the sampling period is accomplished by

$$\hat{\mathbf{x}}(t_{k+1} \mid t_k) = \hat{\mathbf{x}}(t_k \mid t_k) + \int_{t_k}^{t_{k+1}} \mathbf{f}(\hat{\mathbf{x}}(t,t_k),t)\,dt \qquad (21)$$

and equation (8) is replaced by equation (16).

2.1.1 The State Transition Matrix

To discretize the continuous state and state-error covariance differential equations, (7), (8), the state transition matrix, $\mathbf{\Phi}[t_{k+1},t_k]$, must be calculated. First the Jacobian matrix of $\mathbf{f}(\mathbf{x}(t),t)$ must be determined as given by equation (12), and evaluated at $\hat{\mathbf{x}}(t_k \mid t_k)$. Although not necessarily true, it is a good approximation to assume that the Jacobian matrix is constant over the sampling period so that equation (17) can be applied, otherwise $\mathbf{\Phi}[t_{k+1},t_k]$ would have to integrated across the sampling period which is much more difficult and time consuming. This leaves one with the calculation of the matrix exponential. It can be shown [9] that the matrix exponential can easily be calculated when $\mathbf{F}(\hat{\mathbf{x}}(t_k \mid t_k),t_k)$ is placed in Jordan canonical form. However, this method can cause large errors on finite arithmetic machines because eigenvalues that are distinct may be processed as nondistinct and vice versa due to the finite precision of these machines. Therefore, one must resort to another method.

It can also be shown [9] that the matrix exponential can be approximated by a Taylor

$$\exp(\mathbf{A}) = \mathbf{I} + \mathbf{A} + \frac{\mathbf{A}^2}{2!} + \frac{\mathbf{A}^3}{3!} \cdots \tag{22}$$

where

$$\mathbf{A}^3 = \mathbf{AAA} \quad \text{etc.} \tag{23}$$

This method works well on computers only if the norm of the matrix is not too large. If it is, then the finite precession of the computer can produce significant rounding errors. However, Ward [10] explains an algorithm that can handle matrices with large norms. The main features of this algorithm are explained below.

To reduce the 1-norm of the matrix before exponentiation, it is balanced by similarity transforms which produces a diagonal matrix. This is explained by Parlett and Reinsch [11]. To assure that the 1-norm of the diagonal matrix is less than unity, a integer, j, is found such that $2^{j-1} \leq \|\mathbf{F}_1\|_1 \leq 2^j$. Then the new matrix is given by

$$\mathbf{F}_2 = 2^{-j}\,\mathbf{F}_1 \qquad \therefore \qquad \|F_2\|_1 < 1 \qquad \text{and} \qquad e^{\mathbf{F}_1} = \left[e^{\mathbf{F}_2}\right]^{2^j} . \tag{24}$$

Thus the exponential of $\mathbf{F}_1$ is determined by calculating the exponential of $\mathbf{F}_2$ and squaring the result j times. A Pade' approximation is used to calculated the exponential of $\mathbf{F}_2$. The Pade' series is similar to the Taylor series except that it converges in fewer terms, so the number of matrix multiplications is reduced. This works quite well on a computer since $\mathbf{F}_2$ has a 1-norm less than unity. The final result is obtained by inverting the similarity transformations.

This algorithm has one more advantage in that it gives the number of significant figures in the 1-norm of the result. Thus this routine gives some warning when it fails. Although there are many techniques for calculating the matrix exponential [12], this one seems to work the best. The FORTRAN program for this algorithm is presented in Appendix II.

2.2 Outline Of The Filter Algorithm

Figure 1 illustrates a graphical representation of the filter algorithm. At time t_k where the value of $\hat{\mathbf{x}}(t_k \mid t_k)$ and $\mathbf{P}(t_k \mid t_k)$ have been calculated from the previous sample instance, equations (21) and (16) are used to predict $\hat{\mathbf{x}}(t_{k+1} \mid t_k)$ and $\mathbf{P}(t_{k+1} \mid t_k)$ across the sampling period, respectively. At time t_{k+1} a measurement is taken and the state and its associated uncertainty are updated to $\hat{\mathbf{x}}(t_{k+1} \mid t_{k+1})$ and $\mathbf{P}(t_{k+1} \mid t_{k+1})$ via equations (9) and (10) respectively. The amount $\hat{\mathbf{x}}(t_{k+1} \mid t_k)$ differs from $\hat{\mathbf{x}}(t_{k+1} \mid t_{k+1})$ depends on the magnitude of the Kalman gain matrix $\mathbf{K}(t_{k+1})$ and the size of the residual $\mathbf{r}(t_{k+1} \mid t_k)$

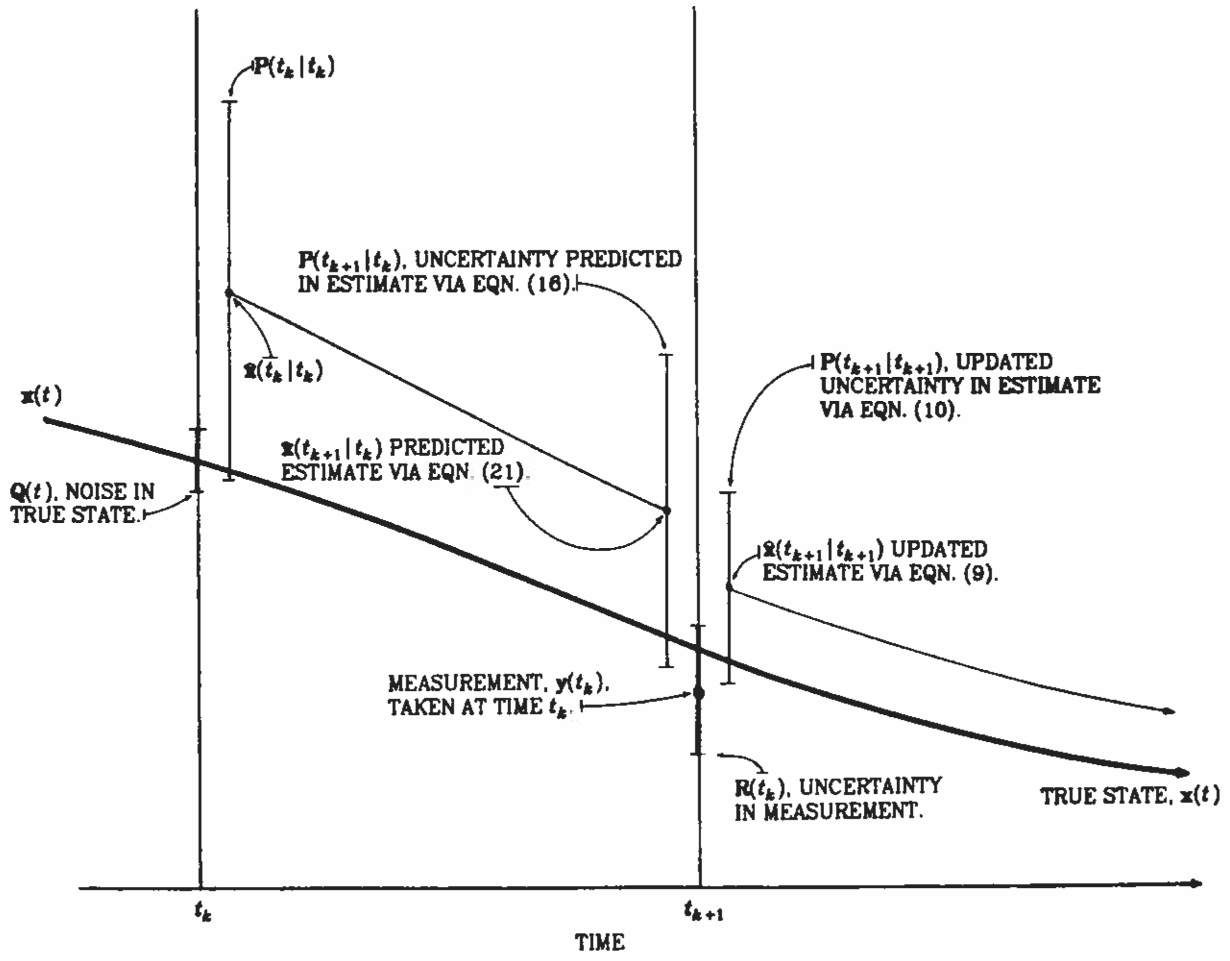
$P(t_k|t_k)$
$P(t_{k+1}|t_k)$, UNCERTAINTY PREDICTED IN ESTIMATE VIA EQN. (16).
$\hat{x}(t_k|t_k)$
$x(t)$
$P(t_{k+1}|t_{k+1})$, UPDATED UNCERTAINTY IN ESTIMATE VIA EQN. (10).
$\hat{x}(t_{k+1}|t_k)$ PREDICTED ESTIMATE VIA EQN. (21).
$Q(t)$, NOISE IN TRUE STATE.
$\hat{x}(t_{k+1}|t_{k+1})$ UPDATED ESTIMATE VIA EQN. (9).
MEASUREMENT, $y(t_k)$, TAKEN AT TIME t_k.
$R(t_k)$, UNCERTAINTY IN MEASUREMENT.
TRUE STATE, $x(t)$
t_k
t_{k+1}
TIME

Figure 1. Qualitative description of the discrete Kalman filter over one sampling period.

$$\mathbf{r}(t_{k+1}|t_k) = \mathbf{y}(t_{k+1}) - \mathbf{h}(\hat{\mathbf{x}}(t_{k+1}|t_k), t_{k+1}) \tag{25}$$

as can be seen from equation (9). The magnitude of the Kalman gain is proportional to the uncertainty in the state, $\mathbf{P}(t_{k+1}|t_{k+1})$, and *inversely* proportional to the uncertainty in the measurement, $\mathbf{R}(t_{k+1})$. Hence as the state-estimate accuracy increases, smaller $\mathbf{P}(t_{k+1}|t_{k+1})$, the weighting on the measurements become smaller. Similarly, if the measurements have a high uncertainty, large $\mathbf{R}(t_{k+1})$, then $\mathbf{K}(t_{k+1})$ is small and again the measurements are not weighted strongly. This relationship is important for understanding why the filter estimate might diverge from the true state. Divergence and counter measures there of will be discussed later in section 3.

2.3 State Equations Used In The Algorithm

In order to model a process a set of variables must be chosen that adequately describe the state of the process at any time. Not only must the variables be determined, but also how the derivative of each variable with respect to time is related to the other state variables. However, due to the finite computational speed of the computer and the limited knowledge of a particular process, a subset of these state variables are used in actual practice. If an adequate estimate is to be obtained, the subset chosen must be those variables that have the greatest influence on the total state of the system.

For a chemostat the following set of differential equations are used to describe the state.

$$\dot{b} = b(\mu - D) \tag{26}$$

$$\dot{s} = D(s_f - s) - \frac{\mu b}{Y_s} \tag{27}$$

$$\dot{\mu} = c_1 \tag{28}$$

$$\dot{c}_1 = 0 \tag{29}$$

$$\dot{Y}_s = 0 \tag{30}$$

$$[\dot{O_2}]_d = k_L a\,([O_2^*]_d - [O_2]_d) - \frac{\mu b}{Y_{O_2}} \tag{31}$$

$$k_L\dot{a} = c_2 \tag{32}$$

$$\dot{c}_2 = 0 \tag{33}$$

$$\dot{Y}_{O_2} = 0 \tag{34}$$

where D, s_f, and $[O_2^*]_d$, are parameters that are known or easily measured, so no filtering is necessary. Since $\Gamma[t_{k+1},t_k]$, not $\mathbf{G}(t)$, is to be determined, the state noise is accounted for in the discrete version of these equations. See section 3.2. Note that these equations are non specific and can be used for any microbe so long as no product is produced.

2.3.1 Choice Of The State Equations

For the biomass, substrate, and dissolved oxygen concentrations, simple mass balance equations are used, but they are quite accurate descriptions. Since biological processes are inherently complicated, it is impossible to derive differential equations that describe the specific growth rate, μ, the substrate or oxygen yield coefficients, Y_s, Y_{O_2}, or the oxygen-mass-transfer coefficient, $k_L a$. If these variables were constant through out the duration of the process there would be no need to incorporate them in the state vector; however, since they do change with time, some form of differential equation must be used to describe them. For the yield coefficients it is assumed that their dynamics are slow enough so that they can be approximated as constants across the sampling period. Thus their derivatives are set to zero. Any change in the estimated value of Y_s or Y_{O_2} comes about during updating at the sampling instant via equation (9). The specific growth rate and $k_L a$ were found to follow the true state more accurately if they are allowed to follow a ramp during the interval between sampling. The slop of the ramp is determined by the size of the residuals, equation (25). This will be discussed in the section on the adaptive **Q** matrix. See Stephanopouls and Ka-Yiu San [13] for other approximations of the state equations.

2.4 Measurement Equations

In most processes the variables that can be measured are often some non-linear combination of the state variables. For a chemostat the measured variables are related to the state variables as follows

$$R^m = \mu b + v_1(t_k) \tag{35}$$

$$Y_s^m = Y_s + v_2(t_k) \tag{36}$$

$$Y_{O_2}^m = Y_{O_2} + v_3(t_k) \tag{37}$$

$$[O_2]_d^m = [O_2]_d + v_4(t_k) \tag{38}$$

$$R_{O_2}^m = k_L a \left([O_2^*]_d - [O_2]_d\right) + v_5(t_k) \tag{39}$$

where $v_i(t_k)$ represents the noise in the measurements and R^m and $R_{O_2}^m$ are the rate of biomass production and the rate of oxygen consumption respectively. How these variables are measured is explained fully in sever papers by Stephanopoulos *et. al*. [13-16] and Cooney, Wang, and Wang [17], however, a brief description follows.

In order to obtain an accurate estimate of the state many samples must be taken per unit time. Currently there exist only a few instruments in industry that are capable of sampling at a high enough frequency to meet the requirements of the filter. These instruments are able to measure the O_2 and CO_2 concentrations in the entrance and exit gasses of a chemostat, as well as the dissolved O_2 concentration in the broth. It is possible to determine the values of the above measured variables from these measurements if a stoichiometric equation is assumed for the growth of the microbe. Such a balance looks like

$$\underset{\substack{\text{carbon energy}\\ \text{source}}}{a\,C_xH_yO_z} + \underset{\text{oxygen}}{g\,O_2} + \underset{\text{ammonia}}{c\,NH_3} \rightarrow \underset{\substack{\text{cell}\\ \text{biomass}}}{C_\alpha H_\beta O_\gamma N_\delta} + \underset{\text{water}}{d\,H_2O} + \underset{\substack{\text{carbon}\\ \text{dioxide}}}{e\,CO_2} \tag{40}$$

where it is assumed that the empirical formula for the carbon energy source and the cell biomass are known, and that there is no product formation. Then the only unknowns in this equation are a, c, d, e, and g. These 5 unknowns can be determined from four elemental balances and the measurement of the rate of O_2 consumption and CO_2 production. Once the five unknowns are determined the yield coefficients are given by

$$Y^m_{O_2} = \frac{(MW)_{bio}}{32\,g} \tag{41}$$

$$Y^m_{CO_2} = \frac{(MW)_{bio}}{44\,e} \tag{42}$$

$$Y^m_s = \frac{(MW)_{bio}}{(MW)_s\,a} \tag{43}$$

and the rate of biomass production and oxygen consumption can be determined from

$$R^m = Y^m_{O_2}\,(\text{grams of } O_2 \text{ consumed}) \tag{44}$$

$$R^m_{O_2} = (\text{grams of } O_2 \text{ consumed}). \tag{45}$$

Dissolved oxygen concentration is simply measured by a probe in the broth. Note that the above measurement procedure is the only part of the filter algorithm that is specific for a particular microbe.

2.4.1 Smoothing Of The Measurements

To eliminate spikes produced by noise, the measurement vector is averaged and smoothed before it is fed into the filter algorithm. These two routines are explained below.

Since data can be acquired at a higher frequency than it can be processed by the computer, it would be wasteful to ignore that data which is obtained in this period. Consequently the filter algorithm is set up such that the computer samples data at frequency f_1 while actual measurements are taken at a higher frequency, f_2. All samples taken at the higher frequency, f_2 are averaged over the time period, $1/f_1$ to produce one point every $1/f_1$ time units.

The averaged points are smoothed further by a least squares technique around a given polynomial. This technique uses the moving window approach, that is only m points back from the current measurement are used. In essence a polynomial of desired order is drawn through the current measurement and the previous m points such that the squared error is minimized. The current point is then adjusted to fall on this curve. See Figure 2. This routine has the advantage that once the degree of the polynomial and the size of the window (*i.e.* m) are chosen, the past m points and the current point need only be multiplied by a predetermined constant (*i.e.* convolute). Consequently, computational time is negligible. See Savitzky and Golay [18] and Steinier [19] for a more detailed explanation.

A block diagram of the measurement procedure is shown in Figure 3.

2.5 Observability

When the state and measurement equations are analyzed it can be shown that the substrate concentration is an unobservable state. However, this is not as big a problem as it first might seem. Since the governing equation for the substrate concentration, (27), is well known, the filter gives a reasonable estimation of this state so long as the assumed initial conditions are close to their actual values. Divergence does become a problem at low substrate concentrations (*i.e.* < 0.1 g/l), as will be shown from simulations. Possible solutions to

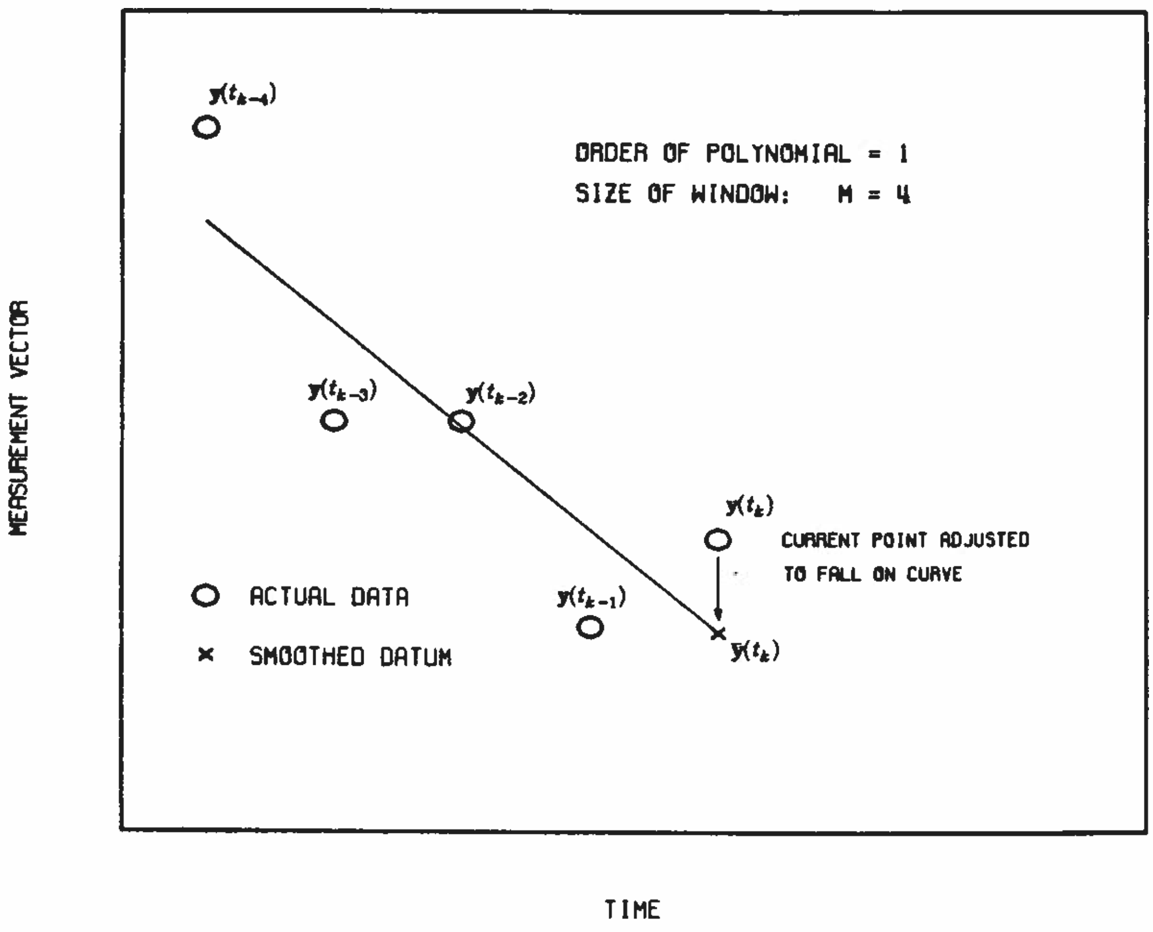

Figure 2. Graphical representation of the least squares smoothing routine for the measured vector.

this problem are discussed later in section 6.

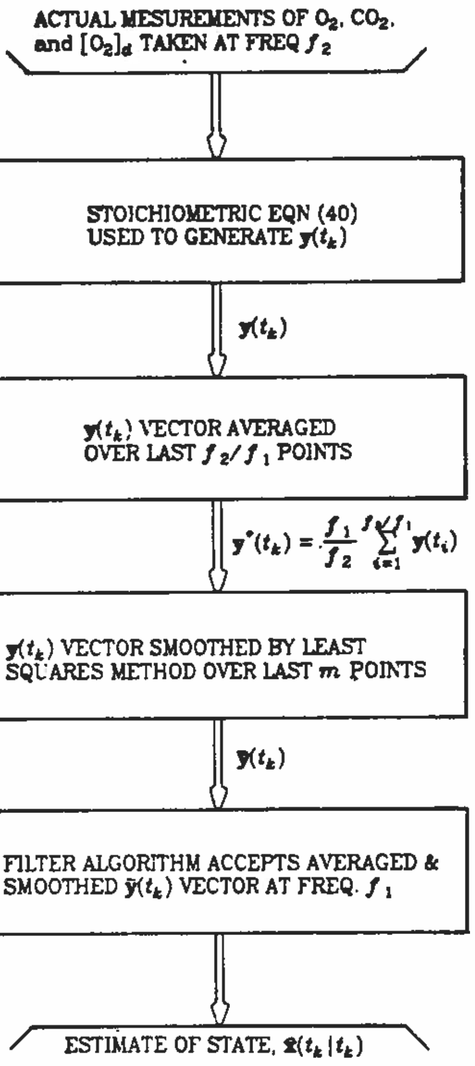
ACTUAL MESUREMENTS OF O_2, CO_2, and $[O_2]_d$ TAKEN AT FREQ f_2
STOICHIOMETRIC EQN (40) USED TO GENERATE $y(t_k)$
$y(t_k)$
$y(t_k)$ VECTOR AVERAGED OVER LAST f_2/f_1 POINTS
$y^*(t_k) = \frac{f_1}{f_2} \sum_{i=1}^{f_2/f_1} y(t_i)$
$y(t_k)$ VECTOR SMOOTHED BY LEAST SQUARES METHOD OVER LAST m POINTS
$y(t_k)$
FILTER ALGORITHM ACCEPTS AVERAGED & SMOOTHED $\bar{y}(t_k)$ VECTOR AT FREQ. f_1
ESTIMATE OF STATE, $\hat{x}(t_k|t_k)$

Figure 3. Block diagram of measurement procedure.

3. ADAPTIVE STATE-NOISE COVARIANCE MATRIX

The Kalman filter equations can be manipulated in in such a way to produce

$$K(t_{k+1}) = P(t_{k+1}\,|\,t_{k+1})\,H^T(t_{k+1};\hat{x}(t_{k+1}\,|\,t_k))\,R(t_{k+1}). \tag{46}$$

which reduces the complexity of the gain equation so it can be evaluated qualitatively. Consider the following. If for any reason the gain becomes small the filter will ignore the measurements no matter how large the residuals are.

Under this situation the Kalman filter reduces to a simple differential equation solver. Consequently if the state equations do not model the system closely the estimate will diverge from the true state because no updating occurs. Since the matrices $\mathbf{H}(t_{k+1};\hat{\mathbf{x}}(t_{k+1}|t_k))$ and $\mathbf{R}(t_{k+1})$ do not change much for a given system, most changes in $\mathbf{K}(t_{k+1})$ are due to changes in the state-error covariance matrix, $\mathbf{P}(t_{k+1}|t_{k+1})$. Divergence can occur when $\mathbf{P}(t_{k+1}|t_{k+1})$ becomes unrealistically small, that is, $\mathbf{P}(t_{k+1}|t_{k+1})$ no longer gives an accurate assessment of the error in the state estimate. This is exactly what is observed to happen when the state equations given above, (26-34), are used to model the chemostat since many of them do not describe the system accurately. To keep the state from diverging, $\mathbf{P}(t_{k+1}|t_{k+1})$ must not be allowed to become unrealistically small. One way of doing this is through the state-noise covariance matrix $\mathbf{Q}(t_k)$.

Ideally the state-noise covariance matrix describes the noise in the equations that represent the true state equation. Unfortunately as mentioned above the true state equations are unknown, consequently, it is very difficult to construct $\mathbf{Q}(t_k)$. The difficulty in constructing $\mathbf{Q}(t_k)$ can be over come by incorporating feed back from the residuals to determine $\mathbf{Q}(t_k)$. In this way filter divergence can be observed and avoided.

3.1 Algorithm For Determing $\mathbf{Q}(t_k)$

To keep $\mathbf{P}(t_k|t_k)$ from becoming unrealistically small, $\mathbf{Q}(t_k)$ can be adjusted in such a way to compensate for model errors. Since the residuals indicate when divergence starts to occur, it is desirable to use them for the determination of $\mathbf{Q}(t_k)$. Jazwinski [20,21] has shown that $\mathbf{Q}(t_k)$ should be chosen in such a way to produce the most probable *predicted* residuals. Based on this idea $\mathbf{Q}(t_k)$ can be determined from the following set of equations.

Let

$$z_{ij} = \Big[\mathbf{r}(t_{k+1} \,|\, t_k)\,\mathbf{r}^T(t_{k+1} \,|\, t_k) - \mathbf{R}(t_{k+1})$$

$$- \mathbf{H}(t_{k+1};\hat{\mathbf{x}}(t_{k+1} \,|\, t_k))\,\Phi[t_{k+1},t_k]\,\mathbf{P}(t_k \,|\, t_k)$$

$$\times\, \Phi^T[t_{k+1},t_k]\,\mathbf{H}^T(t_{k+1};\hat{\mathbf{x}}(t_{k+1} \,|\, t_k))\Big]_{ij}\,\delta_{ij}. \tag{47}$$

Now it is assumed that the measured variables, and consequently the residuals, are uncorrelated, so

$$\mathbf{Q}(t_k) = \text{diag}\,\{\, q_{11}, q_{22}, \cdots, q_{NN}\} \tag{48}$$

where N is the dimension of the measurement vector, $\mathbf{y}(t_k)$. The state-noise covariance matrix is then given by

$$\hat{q}_{ii}(t_k) = \Big[\Big[\mathbf{H}(t_{k+1};\hat{\mathbf{x}}(t_{k+1} \,|\, t_k))\,\Gamma[t_{k+1},t_k]\Big]^{-1}\mathbf{Z}$$

$$\times\,\Big[\Gamma^T[t_{k+1},t_k]\,\mathbf{H}^T(t_{k+1};\hat{\mathbf{x}}(t_{k+1} \,|\, t_k))\Big]^{-1}\Big]_{ii} \tag{49}$$

$$q_{ii}(t_k) = \begin{cases} \hat{q}_{ii}(t_k) & \text{if } z_{ii} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{50}$$

where z_{ij} is an element of $\mathbf{Z}$. These equations are some what ad hoc and will not stand up to rigorous statistical analysis. Nonetheless, they are used in practice since they produce good results. The state-noise covariance matrix now depends on how accurately the current estimate is approximating the true state, and so demonstrates an adaptive behavior.

The adaptive nature of $\mathbf{Q}(t_k)$ is explained as follows. At each measurement the residual vector can be determined from equation (25). Subtracted from this are the current uncertainties in the measurement, $\mathbf{R}(t_k)$, and in the state estimate as given by equation (47). If this value, z_{ij}, is greater than zero, then there exists some noise unaccounted for and hence must be due to either state noise and/or model error. In either case the $\mathbf{Q}(t_k)$ matrix takes on a value proportional to this error via equation (49). Subsequently, the state-error covariance matrix $\mathbf{P}(t_{k+1}|t_k)$ is increased by this amount via equation (16). As mentioned above this increase in $\mathbf{P}(t_{k+1}|t_k)$ will increase the Kalman gain and cause the filter to open to incoming measurements which will adjust the state to its correct value. If z_{ij} is less than zero then the filter is operating satisfactorly and there is no need to adjust $\mathbf{P}(t_{k+1}|t_k)$.

Note that the adaptive feature of $\mathbf{Q}(t_k)$ in the filter algorithm is very important. When simulations were run with out it the state estimate diverged dramatically from its true value. This is mainly due to the poor modeling of the chemostat and not the measurement noise.

3.1.1 Improvement On $\mathbf{Q}(t_k)$

Since only one residual is used to determine $\mathbf{Q}(t_k)$, a spike in the measurement can cause the gain to increase unnecessarily. These wild observations allow noise to pass through the filter. To avoid this the $\mathbf{Q}(t_k)$ matrix is averaged over n points as follows

$$\bar{q}_{ii}^{n}(t_{k+n}) = \frac{1}{n}\sum_{k=1}^{n} \hat{q}_{ii}(t_{k+i}) \qquad (51)$$

The condition given by equation (50) is then applied to $\bar{q}_{ii}^{n}(t_{k+n})$ to obtain $\mathbf{Q}(t_k)$. Another way to elimate spikes from $\mathbf{Q}(t_k)$ is to place an upper bound on $q_{ii}(t_k)$, however, this is not done in the filter program.

3.2 Determination Of The Noise Transition Matrix

As can be seen from equation (20) the noise transition matrix, $\Gamma[t_{k+1},t_k]$, determines how elements of the state-noise vector, $\mathbf{w}(t_k)$, are distributed among the state variables. As mentioned before $\mathbf{G}(t)$ (continuous case) is different from $\Gamma[t_{k+1},t_k]$ (discrete case) in that $\mathbf{G}(t)$ adds noise to the *derivative* of the state variables. Since either matrix must usually be determined by intuition, there is no need to integrate equation (18) to fine $\Gamma[t_{k+1},t_k]$, it can be determined directly.

The adaptive feature of the algorithm discussed in the previous section depends strongly on how $\Gamma[t_{k+1},t_k]$ is constructed. Incorrect modelling of $\Gamma[t_{k+1},t_k]$ will result in noise being added to the wrong variables, and will probably cause the filter to diverge. How $\Gamma[t_{k+1},t_k]$ and $\mathbf{w}(t_k)$ effect the filter are explained as follows. Recall that the diagonal elements of the state-noise covariance matrix, $\mathbf{Q}(t_k)$, are proportional to the residuals squared, equations (47) and (49). From equation (19) it can be seen that the square of the state-noise vector, $\mathbf{w}(t_k)$, is proportional to the diagonal elements of $\mathbf{Q}(t_k)$ if the noise is uncorrelated, which is assumed to be the case. Thus we see that the magnitude of the noise vector is directly related to the magnitude of the residual vector when the adaptive $\mathbf{Q}(t_k)$ matrix routine is incorporated into the filter algorithm. That is, if the filter is operating satisfactorly, which implies small residuals, the state-noise vector will be small and vice versa. Consequently $\Gamma[t_{k+1},t_k]$ determines the relationship between the state variables and the residuals. A large residual will increase the gain on the state variable that $\Gamma[t_{k+1},t_k]$ specifies. The next section describes how $\Gamma[t_{k+1},t_k]$ is constructed for the chemostat state equations.

3.2.1 Transition Matrix For The Chemostat Equations

The construction of $\Gamma[t_{k+1},t_k]$ involves determining how a change in a state variable effects the residual vector. In this way when a residual becomes large it can be determined which state is diverging from its true value. It will be assumed that the biomass differential equation, (26), is exact. Also since the substrate concentration, equation (27), is unobservable, the residuals give no indication of when it diverges. Thus for lack of information it will be assumed that no noise exist in this state equation, (27), either. These two assumptions make the first two rows of $\Gamma[t_{k+1},t_k]$ all zeros. The residuals are directly related to the measured variables via equation (25), so evaluation of these variables ,given by equations (35-39), will determine how the residuals are related to the state variables. The following arguments explain how to determine the structure of $\Gamma[t_{k+1},t_k]$. For simplicity, let

$$\mathbf{w}^T(t_k) = [\, w_1,\, w_2,\, w_3,\, w_4,\, w_5] \tag{52}$$

A large error between the *predicted* rate of biomass production, μb, and the *measured* rate of biomass production, R^m, can be produced by an error in both/either μ and/or b. Since it is assumed that no error exists in b, all error must be associated with μ. Thus w_1 describes the noise in μ, since it is directly related to the residual in the rate of biomass production. Now the state variable c_1 is coupled to μ, so w_1 should account for noise in this variable as well (*i.e.* incorrect slope for μ). Large residuals associate with Y_s^m, $Y_{O_2}^m$, and $[O_2]_d^m$ can only be contributed to errors in the state variables Y_s, Y_{O_2}, and $[O_2]_d$, respectively. Thus w_2, w_3, and w_4 should be added to the state variables Y_s, Y_{O_2}, and $[O_2]_d$ respectively. A large residual associated with the rate of oxygen consumption can be contributed to the state variables k_La and $[O_2]_d$. However, since the noise in $[O_2]_d$ has already been accounted for, w_5 is only added to state variable k_La. From this information and the relationship given by equation (20), $\Gamma[t_{k+1},t_k]$ is given by

$$\Gamma[t_{k+1},t_k] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1/s_f & 0 & 0 & 0 & 0 \\ D/s_f & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1/[O_2^*]_d \\ 0 & 0 & 0 & 0 & D/[O_2^*]_d \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{53}$$

where D, s_f, and $[O_2^*]_d$ are used to get the correct units and magnitudes.

3.2.2 Adaptive Ramp Equations

The specific growth rate and the oxygen-mass-transfer coefficient state variables are indirectly related to the measured variables. Consequently, it was found that more robust state equations than those used for the yield coefficients are necessary for good estimation of k_La and μ. The ramp equations

$$\dot{x} = c + \dot{w} \tag{54}$$

$$\dot{c} = 0 + \dot{w}' \tag{55}$$

proved more adaptive, where x is the state variable, c is a pseudo state, and w is the state noise explained above. The prime on w in equation (55) indicates different units. Integration of these equations produce

$$x = (w' + k_1)t + w + k_2 \tag{56}$$

where k_1 and k_2 are integration constants. The slope of the ramp is determined by the size of the residuals via the noise w. During operation, the filter adjusts the slope of the ramp equation to minimize the error between the estimate and the true state, thus making the equations adaptive.

4. SIMULATIONS

To test the performance of the filter, three different computer simulations were run. All three simulate a chemostat where only biomass is being produce (*i.e.* no products). The simulations differ in that each uses a different set of equations to describe the state of the chemostat. How the simulations are run is explained below.

4.1 Program

Since no actual experiments were run, a computer program was written to simulate the chemostat. Given a set of governing equations for the process, the program generates the true state variables at frequency f_2. From equations (35-39) the measured variables are determined. To each measured variable, noise with a guassian distribution and a zero mean is added. The magnitude of the noise is such that each measurement has a standard deviation of $Sdy\%$ percent of its mean, where $Sdy\%$ is determined by the user. See section 4.3 for the value of $Sdy\%$ and other required inputs to the program. To minimize data file size, the state variables are stored at frequency f_1 and measured variables are averaged as explained before and then stored . Parameters, such as dilution rate, D, are stored at frequency f_1 as well. The averaged measurements are fed to the filter program where they are processed to produce an estimate of the state by the algorithm described above. By comparing the estimate to the true state, performance can be determined. Note that the simulation does not generate O_2 consumption of CO_2 production data. Nor is the stoichiometric microbial growth equation used. Stephanopoulos [13] and Cooney, Wang, and Wang [17] have shown that the algorithm for generating the $\mathbf{y}$ vector works, so there is no need to test it here. For more details on the chemostat simulator program see Appendix II.

4.2 State Equations Used In The Simulations

In the sections below are descriptions of the state equations used in each simulation as well as what aspect of the filter algorithm each simulation is intended to test.

4.2.1 Simulation 1

The main purpose of this simulation is to see if the filter can follow a ramp decrease in the oxygen-mass-transfer coefficient. Such a condition might be induced in actual practice by changing the agitation speed. A Monod model, based on dissolved oxygen concentration, determines the values of μ and Y_s. The simulation was generated by the following equations

$$\dot{b} = b(\mu - D) \tag{57}$$

$$\dot{s} = D(s_f - s) - \frac{\mu b}{Y_s} \tag{58}$$

$$\mu = \frac{0.637[O_2]_d}{1.0\times10^{-3} + [O_2]_d} \tag{59}$$

$$Y_s = \frac{0.7[O_2]_d}{6.67\times10^{-4} + [O_2]_d} \tag{60}$$

$$\dot{[O_2]}_d = k_L a\,([O_2^*]_d - [O_2]_d) - \frac{\mu b}{Y_{O_2}} \tag{61}$$

$$k_L a = \begin{cases} 550 & 0.0 \le t < 6.0 \\ 550 - 33.3(t - 6.0) & 6.0 \le t < 12.0 \\ 350 & 12.0 \le t \le 20.0 \end{cases} \tag{62}$$

$$Y_{O_2} = 0.5 \qquad \text{for all time } t \tag{63}$$

where the parameters have the following values for all time, $D = 0.4$ 1/hr, $s_f = 10.0$ g/l, and $[O_2^*]_d = 7.5$ mg/l.

4.2.2 Simulation 2

This simulation is intended to test how the filter responds to large transients, in this case, step changes in dilution rate, D. The biomass, substrate, and dissolved oxygen concentrations are governed by equations (57), (58), and (61) respectively. Both k_La and Y_{O_2} are held constant through out the simulation at 400 1/hr and 1.5 respectively. To simulate a Crabtree inhibition effect observed in yeast, the substrate yield coefficient and specific growth rate follow curves observed by Wang, Cooney, and Wang [22] and Kaspar Von Meyenburg [23]. No algebraic equation is given for the substrate yield coefficient, so the data was fit to a third order polynomial as shown below

$$\mu = \frac{0.53s}{0.155 + s} \tag{64}$$

$$Y_s = \begin{cases} 8.33\mu & 0.0 \le \mu < 0.06 \\ 0.5 & 0.06 \le \mu \le 0.24 \\ 3.1 - 18.7\mu + 39.5\mu^2 - 27.6\mu^3 & 0.24 < \mu \le 0.6 \end{cases} \tag{65}$$

The dilution rate is stepped as follows

$$D = \begin{cases} 0.10 & 0.0 \le t < 5.0 \\ 0.22 & 5.0 \le t < 10.0 \\ 0.40 & 10.0 \le t \le 20.0 \end{cases} \tag{66}$$

while s_f = 28 g/l, and $[O_2^*]$ is the same as in simulation 1.

4.2.3 Simulation 3

This simulation is the same as simulation 2 except that k_La is a function of biomass, and μ is a function of $[O_2]_d$ as well as s as shown by these equations

$$\mu = \frac{0.53[O_2]_d\, s}{(0.5 + s)(2\times10^{-3} + [O_2]_d)} \tag{67}$$

$$k_L a = 350 + 16.7(14 - b). \tag{68}$$

The parameters and the remaining states are the same as in simulation 2. This simulation is intended to see how the filter responds to increased nonlinearities.

4.3 Data Used For The Simulations

All simulations were run with the following data.

SAMPLING RATES:

$f_1 = 1$ point/min.
$f_2 = 10$ points/min.

To all simulated measurements 5 percent noise is added (*i.e.* $Sdy\% = 5$). For the measurement-noise covariance matrix, $\mathbf{R}(t_k)$, all measurements were assumed uncorrelated with 2 percent standard deviation as show by the following equations

$$\mathbf{R}(t_k) = \mathrm{diag}\{\sigma_1^2, \sigma_2^2, \sigma_3^2, \sigma_4^2, \sigma_5^2, \} \tag{69}$$

where

$$\sigma_i^2 = (SdR\% \;\; y_i(t_k))^2 \qquad \text{and} \qquad SdR\% = 0.02. \tag{70}$$

For the smoothing routine, section 2.4.1, a first order polynomial with $m = 5$ is used. The smoothed measured vector is given by

$$\bar{\mathbf{y}}(t_k) = \frac{11}{21}\mathbf{y}(t_k) + \frac{8}{21}\mathbf{y}(t_{k-1}) + \frac{5}{21}\mathbf{y}(t_{k-2})$$

$$+ \frac{2}{21}\mathbf{y}(t_{k-3}) - \mathbf{y}(t_{k-4}) - \frac{4}{21}\mathbf{y}(t_{k-5}) \tag{71}$$

The state-noise covariance matrix, $\mathbf{Q}(t_k)$, is smoothed over 5 points (*i.e.* $N = 5$).

The filter program requires an initial value for the state variables and the state-error covariance matrix, $\mathbf{P}(t_k \mid t_k)$. For all three simulations a 10 percent error is incorporated in the initial guesses for the state variables and only the diagonal elements of $\mathbf{P}(t_k \mid t_k)$ are initialized.

5. RESULTS AND DISCUSSION

In Appendix I, the results of simulations 1, 2, and 3 are presented in Figures 4a-4g, Figures 5a-5g, and Figures 6a-6g, respectively. In each figure the solid line is produced by the true state, generated by the simulation program, while the dashed line is produced by the estimated state, generated by the Kalman filter program. Also in Appendix I, Figures 7a-7e, are examples of what the measured variable data, generated by the simulation program, looks like. The specific results of each simulation are discussed below.

5.1 Simulation 1 Results

As can be seen in Figure 4f the estimate of the oxygen-mass-transfer coefficient follows the true state quite closely. All other state variables are also estimated well; however, high frequency noise appears in the dissolved oxygen concentration estimate. The noise is not due to the measurement noise of $[O_2]_d^m$ as one might expect. Instead it was found that the noise in $[O_2]_d$ is produced by fluctuations in the state variables that $[O_2]_d$ is a function of, see equation (31). These fluctuations do not effect the biomass or substrate concentrations because the dynamics of these equations are much slower than the $[O_2]_d$ equation. A quasi-steady-state approximation was used for the $[O_2]_d$ state filter equation in hopes of reducing the noise. Although this idea did reduce some of the noise, it was not a great enough improvement to warrant the loss in generality. The noise in the $[O_2]_d$ estimate was only observed at low concentrations (< 2 mg/l), and is only considered a problem if the signal is used for control purposes. Also the problem can probably be solved by a post filter, smoother.

5.2 Simulation 2 Results

This simulation also exhibits good k_La estimation; however, it also introduces two new problems. The first and most disturbing result can be seen in

Figure 5b. The estimate of the substrate concentration fluctuates violently around the true state. These oscillations are only observed at low concentrations, but they can be explained qualitatively. Consider the steady state substrate concentration equation

$$s = s_f - \frac{\mu b}{DY_s} \tag{72}$$

and remember that μ, b, and Y_s are only estimates so they can fluctuate. At high substrate concentrations s_f is much greater than $\frac{\mu b}{DY_s}$, as can be seen from equation (72). Consequently fluctuations in $\frac{\mu b}{DY_s}$ produce only *small* relative fluctuations in s. However, at low substrate concentrations $\frac{\mu b}{DY_s}$ is almost equal to s_f, so fluctuations in $\frac{\mu b}{DY_s}$ produce *large* relative fluctuations in s. Since s is nonobservable, and so cannot be updated, these oscillations can cause the state to diverge from its true value. The only real solution to this problem is to make s observable. This idea might also explain some of the noise observed in $[O_2]_d$ at low concentrations.

The second problem involves the estimation of the biomass concentration and the specific growth rate. As can be seen from Figures 5a and 5c, the biomass concentration over shoots its true value while the specific growth under shoots its true value. To account for this behavior, recall that there is only one measurement, R^m, yet there two unknowns, μ, and b. Consequently, the evaluation of μ and b is underdetermined and there are and infinite number of solutions that can meet these constraints. Yet this is not a problem since the necessary information required to close the system is obtained from the state equations and through minimizing the cost function J, equation (6). However, the quality of the solution depends on how accurate the state model is. Consequently the accuracy of μ and b depends on the amount of noise in the measured variable and how well μ is modeled. The only solution to this problem is to model μ more accurately. Even so, this is not much of a problem since it was induced by a large transient and both estimates converge back to their true states rapidly.

5.3 Simulation 3 Results

This simulation is very similar to simulation 2. The main differences are the increased enter dependence of the state variables and the higher substrate concentration. The increase complexity of the model does not effect the accuracy of the estimate. Also the improved estimate of the substrate concentration, Figure 6b, confirms the hypothesis about the poor estimation at low substrate concentrations. However, this simulation still exhibits overshoots and undershoots of b and μ respectively. This is understandable since the filter uses the same model for the specific growth rate equation.

5.4 Sensitivity

It has been observed that the sensitivity of the filter resides in the choice of $SdR\%$ in the measurement-noise covariance matrix $\mathbf{R}(t_k)$. See equations (69) and (70). Qualitatively, the $\mathbf{R}(t_k)$ matrix determines how heavily to weight the measurements with respect to the estimated values. If $SdR\%$ is chosen too small then the mean of the estimates follow their true values closely; however, much more measurement noise passes through the filter. If $SdR\%$ is chosen to large, then the measurements are basically ignored, in which case the state is completely determined via integration of equations (26-34) only. This produces very smooth estimates and works quite well if the model equations accurately describe the system. However, if the model does not describe the state accurately, as it does not in this case, then the estimates can wildly diverge from their true values. Consequently, there is a trade off between a noisy accurate estimate and a smooth inaccurate estimate. As stated above, a value of 2% for *SdR%* produces a nice compromise between the two. This value of course depends on the magnitude of the measurement noise and the accuracy of the model, which will be different for each application.

5.5 Conclusions

Although not tested with experimental data, the simulations imply that the filter algorithm works quite well for estimating $k_L a$ as well as the other state variables. The algorithm handles transients very well and the state equations do adapt to the prevailing conditions of the system. This information suggests that $k_L a$ can be estimated accurately on-line with the filter algorithm presented in this paper.

6. FUTURE RESEARCH

So far the filter algorithm has only been tested theoretically. Although it appears to work, is must be confirmed experimentally. The experiment must be set up such that all important state variables can be determined. Variables such as biomass and substrate concentrations can be measured off-line quite easily so they present no problems. The oxygen-mass-transfer coefficient is another case though. Some technique must be developed to measure k_La so that the performance of the filter can be evaluated. This problem will be investigated next.

Currently the adaptive feature of the filter predominately resides in the calculation of the $\mathbf{Q}(t_k)$ matrix. It is believed that the performance of the filter can be dramatically improved if the adaptive state equations, such as the ramp equations, can be designed to adapt to the process faster and with better prediction capabilities. With better prediction, the measurements need not be weighted heavily, so the noise that passes through the filter will be greatly reduced. This is important enough to warrant further research.

Observability of the substrate concentration is still a problem. One solution, however, is to incorporate off-line measurements into the filter algorithm. Although off-line measurements have a sampling period of around one hour, these updates might be enough to keep the substrate concentration from diverging from its true value. However, this must be confirmed.

Finally, research must be done to incorporate the estimation algorithm into a control scheme.

7. NOMENCLATURE

In this manuscript all bold capital letters are matrices, all bold lower case letters are vectors and the rest are simple variables or operators.

VARIABLE	DEFINITION	DIMENSIONS OR UNITS
$\mathbf{A}$	As defined in equation (22)	
a	Stiochiometric coefficient, equation(40)	
b	Biomass concentration	g/l
c_1	Pseudo state variable	$1/hr^2$
c_2	Pseudo state variable	$1/hr^2$
c	Stiochiometric coefficient equation (40)	
D	Dilution rate	1/hr
d	Stiochiometric coefficient equation (40)	
e	Stiochiometric coefficient equation (40)	
$\mathbf{F}(\hat{\mathbf{x}}(t),t)$	Jacobian matrix of $\mathbf{f}(\hat{\mathbf{x}}(t),t)$	$M \times M$
$\mathbf{F}_1$	As defined in equation (24)	
$\mathbf{F}_2$	As defined in equation (24)	
$\mathbf{f}(\mathbf{x}(t),t)$	Nonlinear state vector function	M
f_1	Computer sampling frequency	1/hr
f_2	Measurement sampling frequency	1/hr
$\mathbf{G}(t)$	As defined in equation (1)	$M \times L$
g	Stiochiometric coefficient equation (40)	
$\mathbf{H}(t_{k+1};\hat{\mathbf{x}}(t_{k+1}\|t_k))$	Jacobian matrix of $\mathbf{h}(\hat{\mathbf{x}}(t_{k+1}),t_{k+1})$ evaluated at $\hat{\mathbf{x}}(t_{k+1}\|t_k)$	$N \times M$
$\mathbf{h}(\mathbf{x}(t_k),t_k)$	Nonlinear measurement vector function	N
$\mathbf{I}$	Identity matrix	
J	Cost function	
j	As defined in equation (24)	
$\mathbf{K}(t_k)$	Kalman gain matrix	$M \times N$
$k_L a$	Oxygen-mass-transfer coefficient	1/hr
k_1	Integration constant	
k_2	Integration constant	
L	Dimension of $\mathbf{w}(t)$	
M	Dimension of state	

VARIABLE	DEFINITION	DIMENSIONS OR UNITS
$(MW)_{bio}$	Molecular weight of the biomass	
$(MW)_s$	Molecular weight of the substrate	
m	Size of smoothing window	
N	Dimension of measurement vector	
n	Number of points for $\hat{q}_{ii}$ smoothing	
$[O_2]_d$	Dissolved oxygen concentration	g/l
$[O_2^*]_d$	Equilibrium $[O_2]_d$ concentration	g/l
$\mathbf{P}(t)$	State-error covariance matrix	$M \times M$
$\mathbf{P}(t_k \mid t_l)$	State-error covariance matrix at time t_k assessed from measurements taken at time t_l	$M \times M$
$\mathbf{Q}(t)$	State-noise spectral density matrix	$L \times L$
$\mathbf{Q}(t_k)$	State-noise covariance matrix	$N \times N$
$q_{ii}(t_k)$	Diagonal element of $\mathbf{Q}(t_k)$	
$\hat{q}_{ii}(t_k)$	As defined in equation (49)	
$\bar{q}_{ii}^n(t_k)$	Smoothed element of $\mathbf{Q}(t_k)$	
$\mathbf{R}(t_k)$	Measurement-noise covariance matrix	$N \times N$
$\mathbf{r}(t_{k+1} \mid t_k)$	Residual vector	N
s	Substrate concentration	g/l
s_f	Substrate feed concentration	g/l
t	Time (continuous)	hr
t_k	Time (discrete)	hr
$\mathbf{v}(t_k)$	Measurement vector	N
$v_i(t_k)$	Element of $\mathbf{v}(t_k)$	
$\mathbf{w}(t)$	State-noise vector, continuous	L
$\mathbf{w}(t_k)$	State-noise vector, discrete	N
w_i	Element of $\mathbf{w}(t_k)$	
$\mathbf{x}(t)$	True state vector	M
$\tilde{\mathbf{x}}(t)$	Error in estimated state vector	M
$\hat{\mathbf{x}}(t_k \mid t_l)$	State estimate at time t_k assessed from measurements taken at time t_l	M
Y_s	Substrate yield coefficient	
Y_{O_2}	Oxygen yield coefficient	
$\mathbf{y}(t_k)$	Measurement vector	N
$\mathbf{Z}$	As defined in equation (47)	N
z_{ij}	Element of $\mathbf{Z}$	

GREEK LETTERS

VARIABLE	DEFINITION	DIMENSIONS OR UNITS
$\Gamma[t_{k+1},t_k]$	Noise transition matrix	$M \times N$
$\Phi[t_{k+1},t_k]$	State transition matrix	$M \times M$
μ	Specific growth rate	1/hr
σ_{ii}^2	Diagonal element of $\mathbf{R}(t_k)$	

SUPERSCIPTS

m	Measured variable
T	Transpose of a matrix
-1	Inverse of a matrix

SPECIAL FUNCTIONS

$(\dot{\ })$	Derivative with respect to time
$\|\ \|_1$	1-norm of a matrix
$\varepsilon\{\ \}$	Expectation operator
δ_{ij}	Kronecker delta
$\delta(t-\tau)$	Dirac delta function

8. REFERENCES

[1] T. Yoshida, S. Ohasa, and H. Taguchi, "On-Line Estimation of $k_L a$ by Analog Computer," *Biotech. Bioeng.*, **22**, 201-214 (1980).

[2] J. Koizumi, and S. Aiba, "Reassessment of the Dynamic $k_L a$ Methods," *Biotech. Bioeng.*, **26**, 1131-1133 (1984).

[3] J.A. Spriet, J. Botterman, D.R. De Buyser, P.L. Visscher, and E.J. Vandamme, "A Computer-Aided Noninterfering On-Line Technique for Monitoring Oxygen-Transfer Characteristics during Fermentation Processes," *Biotech. Bioeng.*, **24**, 1605-1621 (1982).

[4] F. Vardar, and M.D. Lilly, "The Measurement of Oxygen-Transfer Coefficients in Fermentors by Frequency Responce Techniques," *Biotech. Bioeng.*, **24**, 1711-1719 (1982).

[5] R.E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *J. Basic Eng.*, **82D**, 34-35 (1960).

[6] R.E. Kalman, and R.S. Bucy, "New Results in Linear Filtering and Prediction Theory," *J. Basic Eng.*, **83D**, 95-107 (1961).

[7] A.H. Jazwinski, *Stochastic Processes* and *Filtering Theory*, Academic Press, New York, 1971.

[8] A. Gelb, Ed., *Applied Optimal Estimation*, The M.I.T. Press, Massachusetts, 1974.

[9] B. Nobe, and J.W. Danieal, *Applied linear Algebra*, Second ed., Prentice-Hall, Inc., New Jersey, 1977.

[10] R.C. Ward, "Numerical Computation of the Matrix Exponential with Accuracy Estimate," *SIAM J. Numer. Anal.*, **14**, 600-610 (1977).

[11] B.H. Parlett, and C. Reinsch, "Balancing a Matrix for Calculation of Eigenvalues and Eigenvectors," *Numer. Math.*, **13**, 293-304 (1969).

[12] C. Moler, and C. Van Loan, "Nineteen Dubious Ways to Compute the Exponential of a Matrix," *SIAM Review*, **20**, 801-836 (1978).

[13] G. Stephanopoulos, and K.-Y. San, "Studies On On-Line Bioreactor Identification. I. Theory," *Biotech. Bioeng.*, **26**, 1176-1188 (1984).

[14] K.-Y. San, and G. Stephanopoulos, "Studies On On-line Bioreactor Identification. II. Numerical and Experimental Results," *Biotech. Bioeng.*, **26**, 1189-1197 (1984).

[15] R. Gross, G. Stephanopoulos, and K.-Y. San, "Studies On On-Line Bioreactor Identification. III. Sensitivity Problems with Respiratory and Heat Evolution Measurements," *Biotech. Bioeng.*, **26**, 1198-1208 (1984).

[16] K.-Y. San, and G. Stephanopoulos, "Studies On On-Line Bioreactor Identification. IV. Utilization of pH Measurements for Product Estimation," *Biotech. Bioeng.*, **26**, 1209-1218 (1984).

[17] C.L. Cooney, H.Y. Wang, and D.I.C. Wang, "Computer-Aided Material Balancing for Prediction of Fermentation Parameters," *Biotech. Bioeng.*, **14**, 55-67 (1977).

[18] A. Savitzky, and M.J.E. Golay, "Smoothing and Differentiation of Data by Simplified Least Squares Procedure," *Anal. Chem.*, **36**, 1627-1639 (1964).

[19] J. Steinier, Y. Termonia, and J. Deltour, "Comments on Smoothing and Differentiation of Data by Simplified Least Squares Procedure," *Anal. Chem.*, **44**, 1906-1909 (1972).

[20] A.H. Jazwinski, "Adaptive Filtering," Analytical Mechanics Associates, Inc., Interm Report 67-6, 1967.

[21] A.H. Jazwinski, "Adaptive Filtering," *Automatica*, **5**, 475-485 (1969).

[22] H. Wang, C.L. Cooney, and D.I.C. Wang, "Computer Control of Bakers' Yeast Production," *Biotech. Bioeng.*, **21**, 975-995 (1979).

[23] H. Kaspar Von Meyenburg, "Energetics of the Budding Cycle of *Saccharomyces cerevisial* During Glucose Limited Aerobic Growth," *Arch. Mikrobiol.*, **66**, 289-303 (1969).

9. APPENDIX I

This appendix contains the results of all simulations. In each figure the solid line represents the true state, generated by the simulation program, and the dotted line represents the estimate, produced by the filter program. All state estimates are started with a 10 percent error. In section 9.4 are the figures of the measured variables typically generated by the simulation program. These are the actual measured variables used for simulation 3.

9.1 Simulation 1 Figures

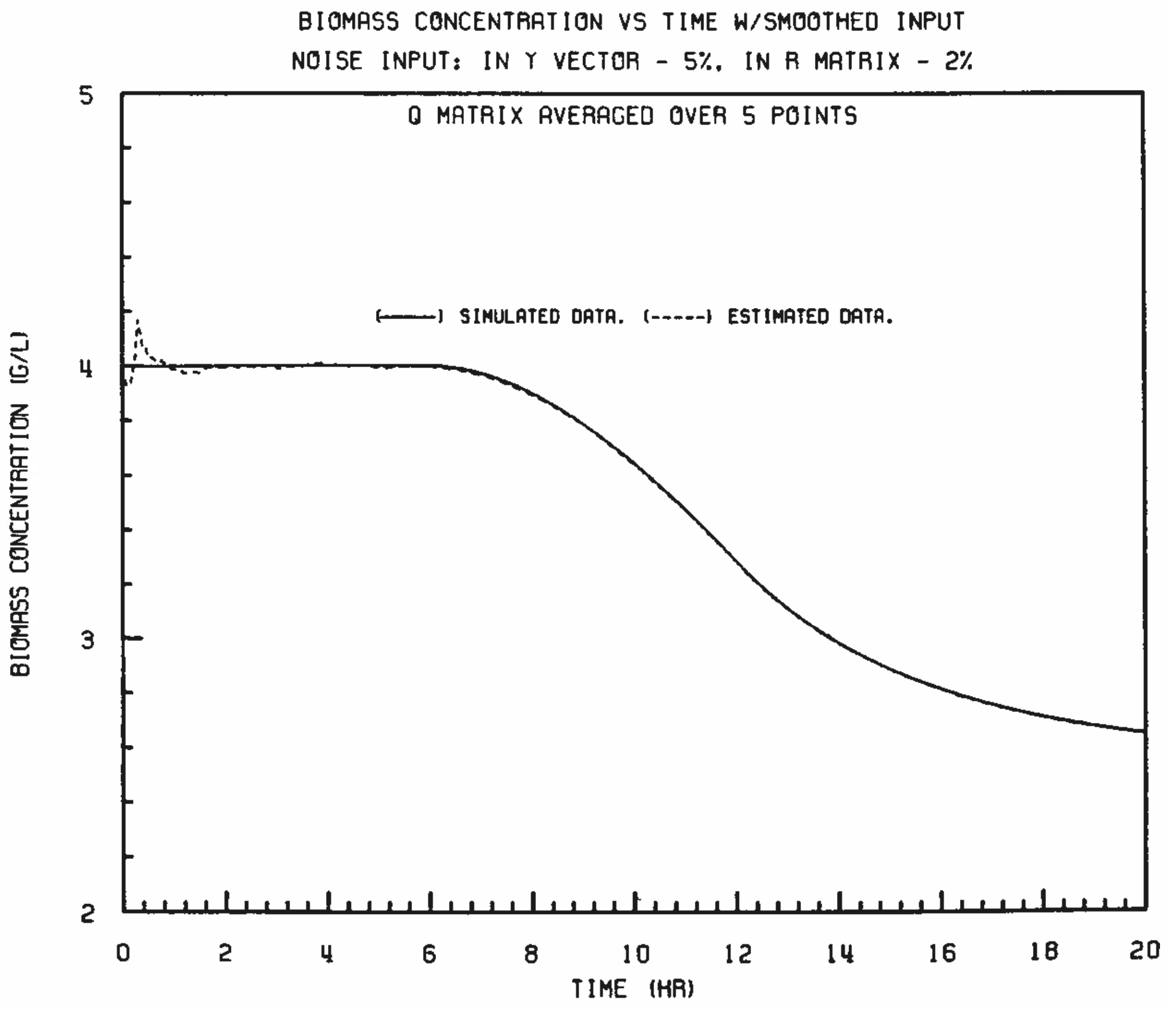

Figure 4a. Biomass concentration vs time for simulation 1. Note the rapid convergence of the estimate.

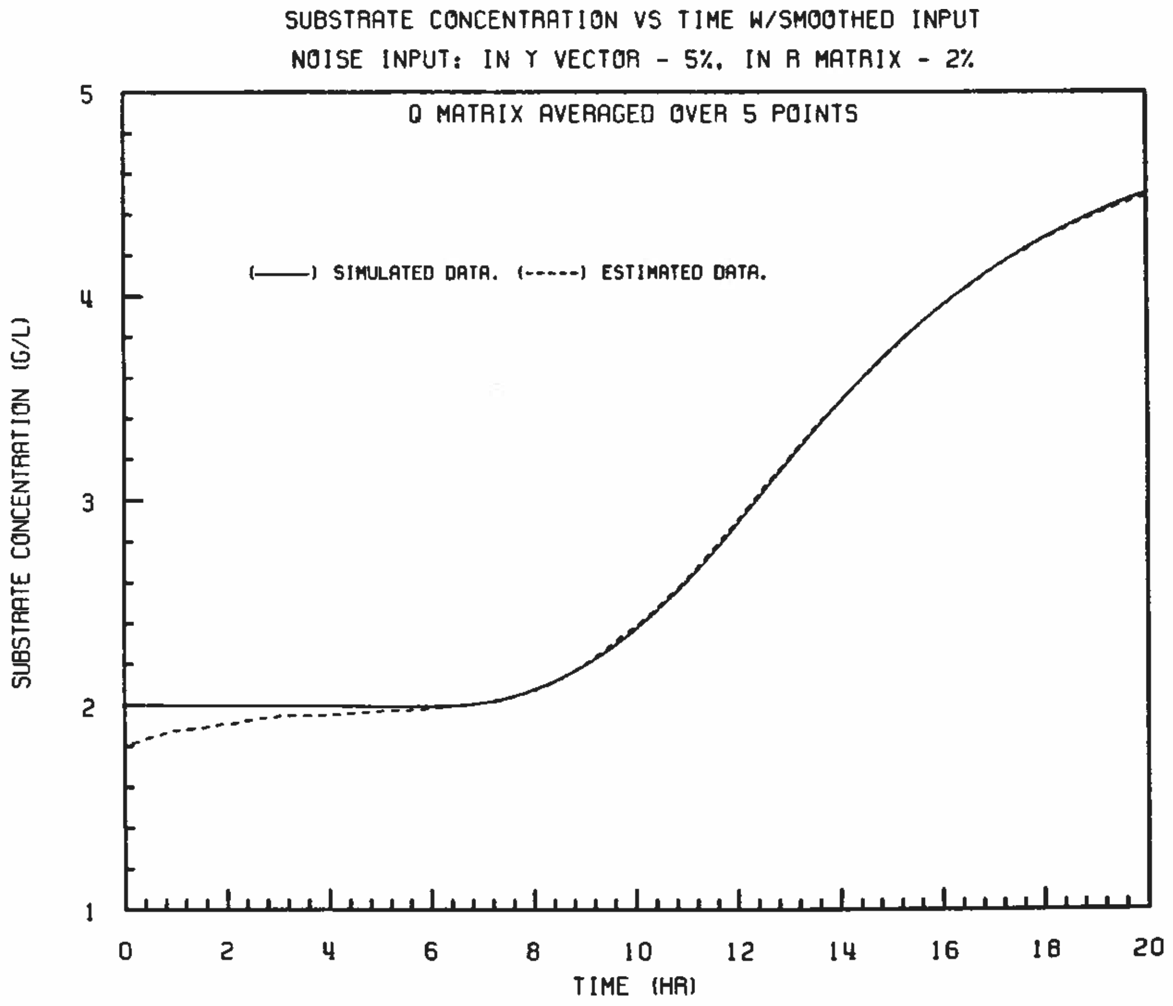

Figure 4b. Substrate concentration vs time for simulation 1. Since *s* is unobservable it takes 6 hours before convergence occurs.

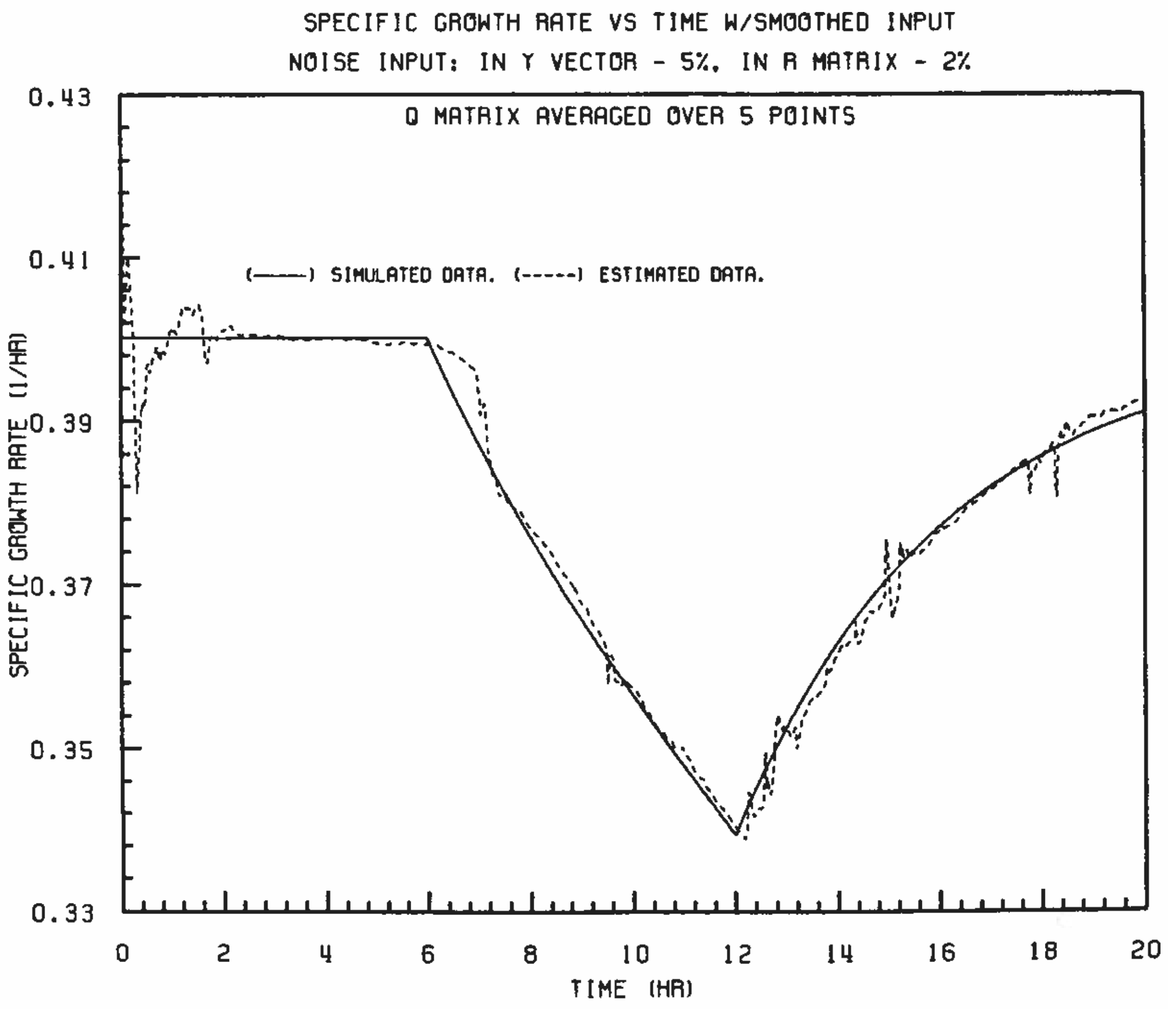

Figure 4c. Specific growth rate vs time for simulation 1. Note that the ordinate is greatly expanded

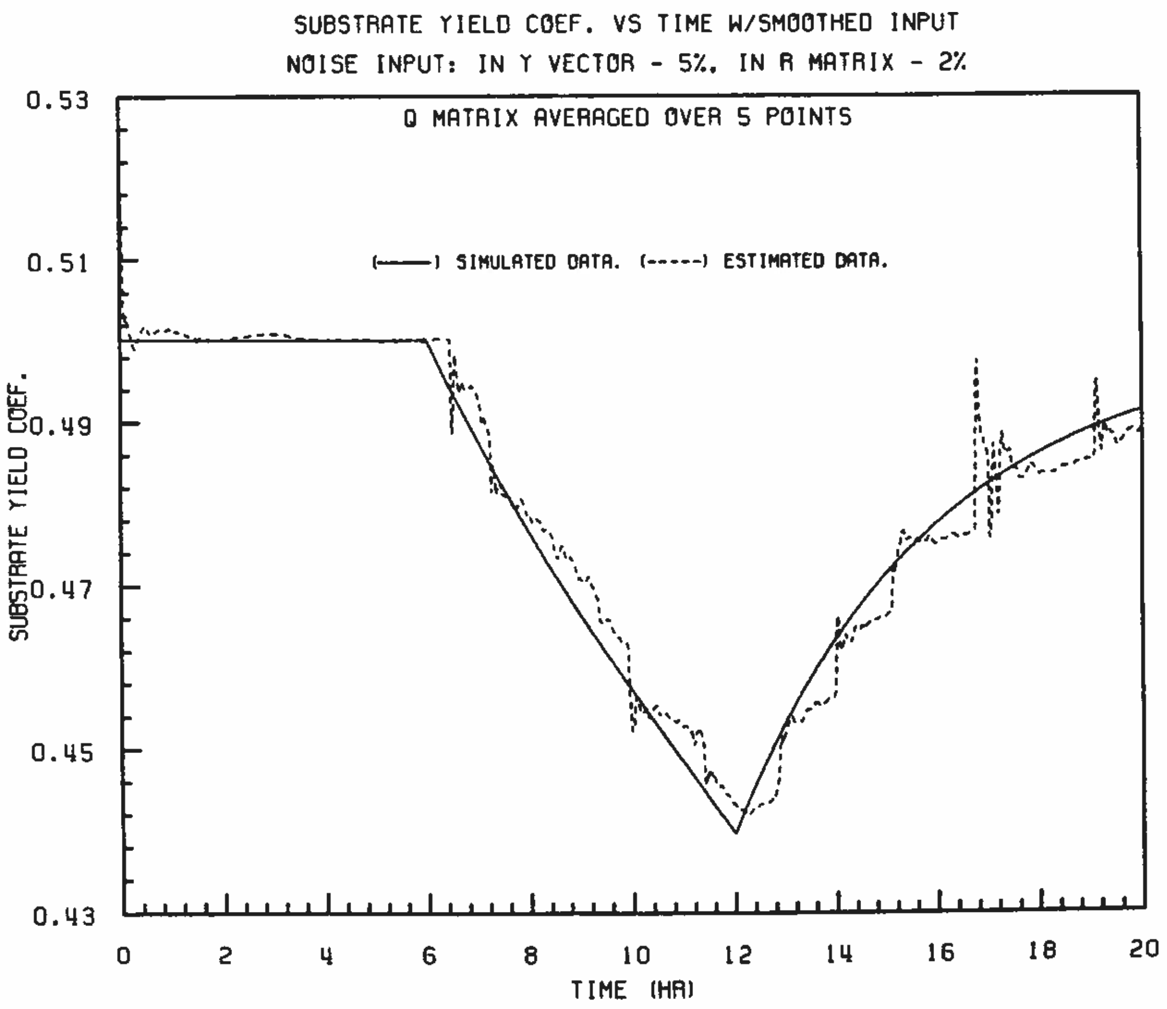

Figure 4d. Substrate yield coefficient vs time for simulation 1. Again the ordinate is expanded, since Y_s only changes about 15 percent.

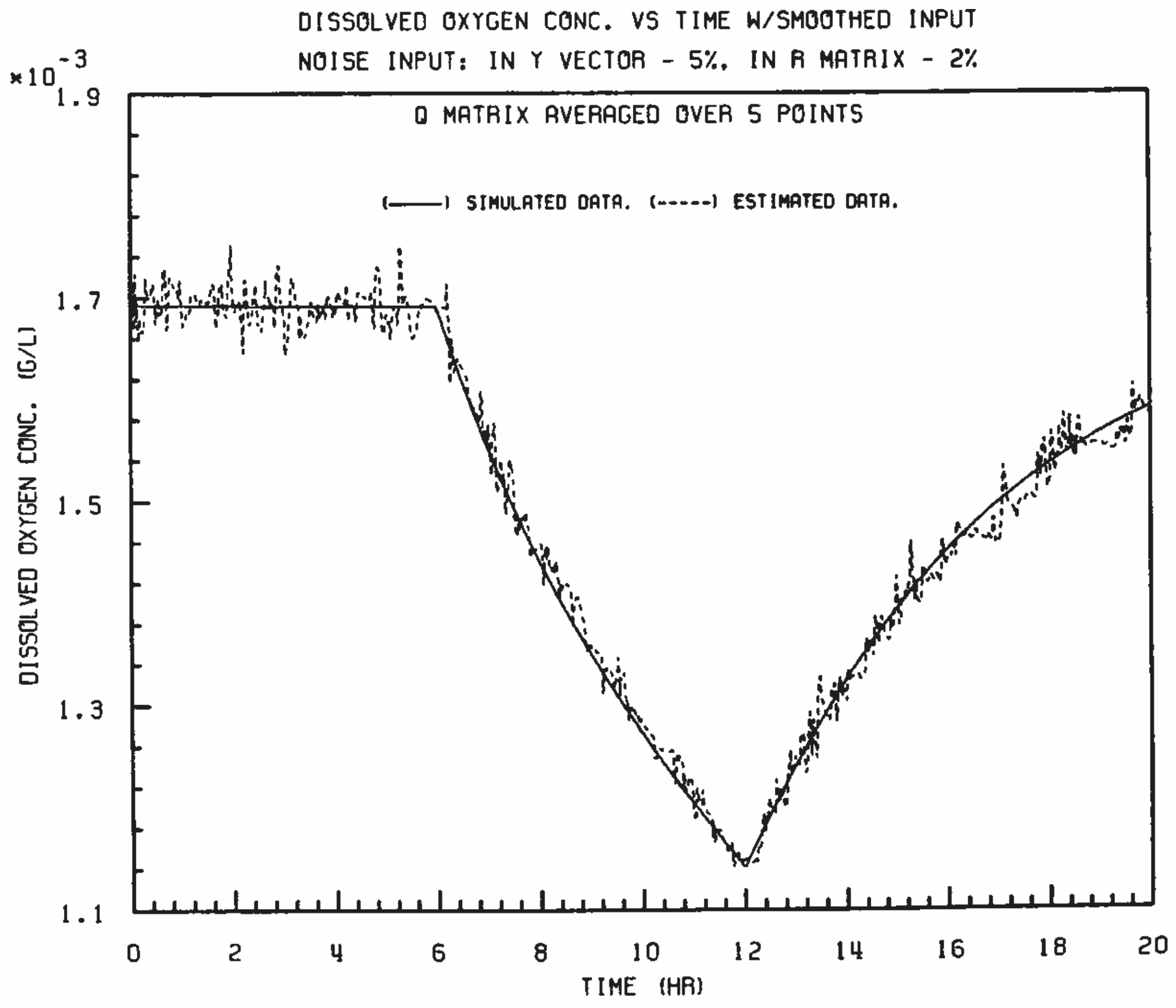

Figure 4e. Dissolved oxygen concentration vs time for simulation 1. High frequency noise passes through the filter in this case.

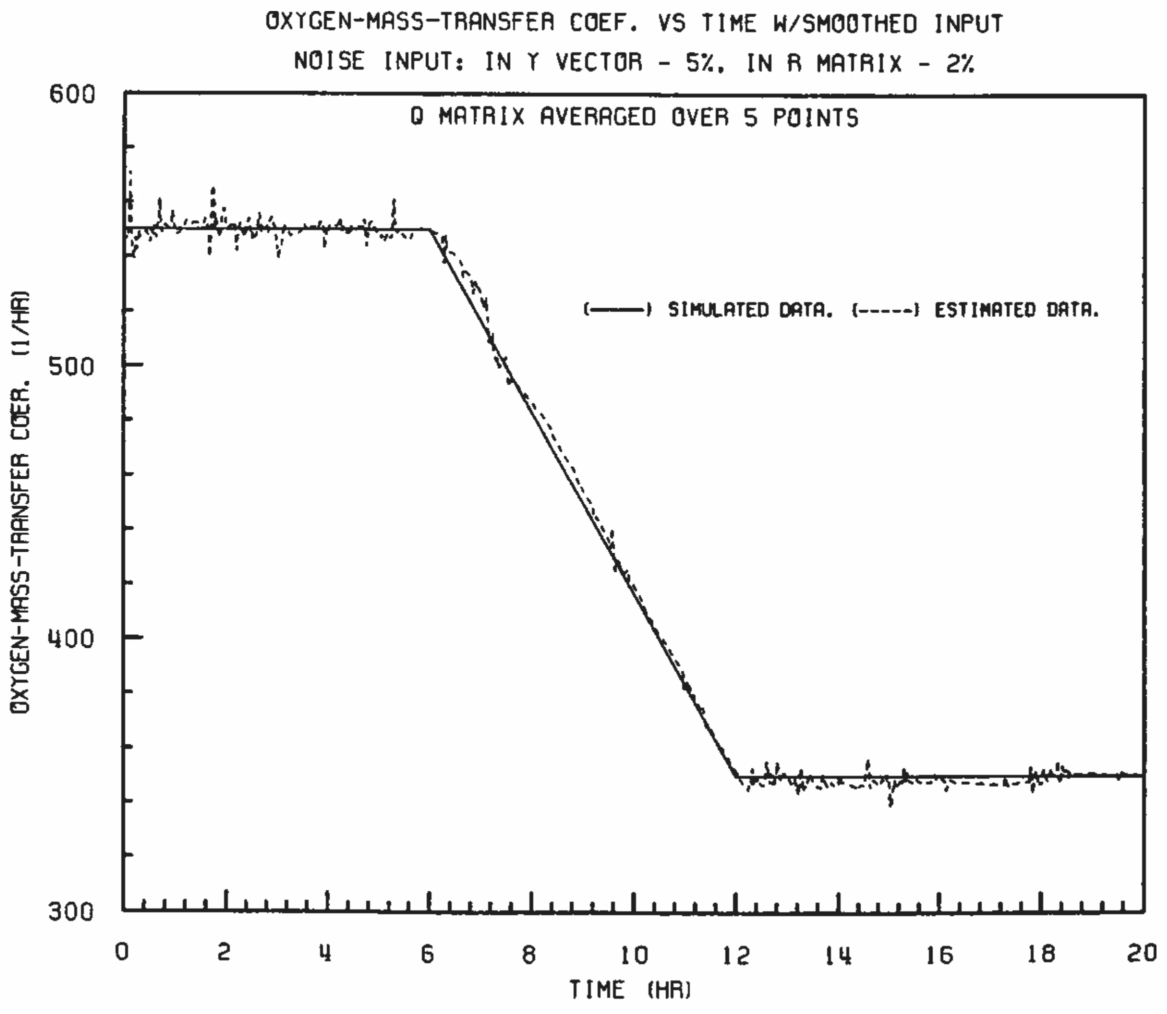

Figure 4f. Oxygen-mass-transfer coefficient vs time for simulation 1. The adaptive ramp equation works very well here.

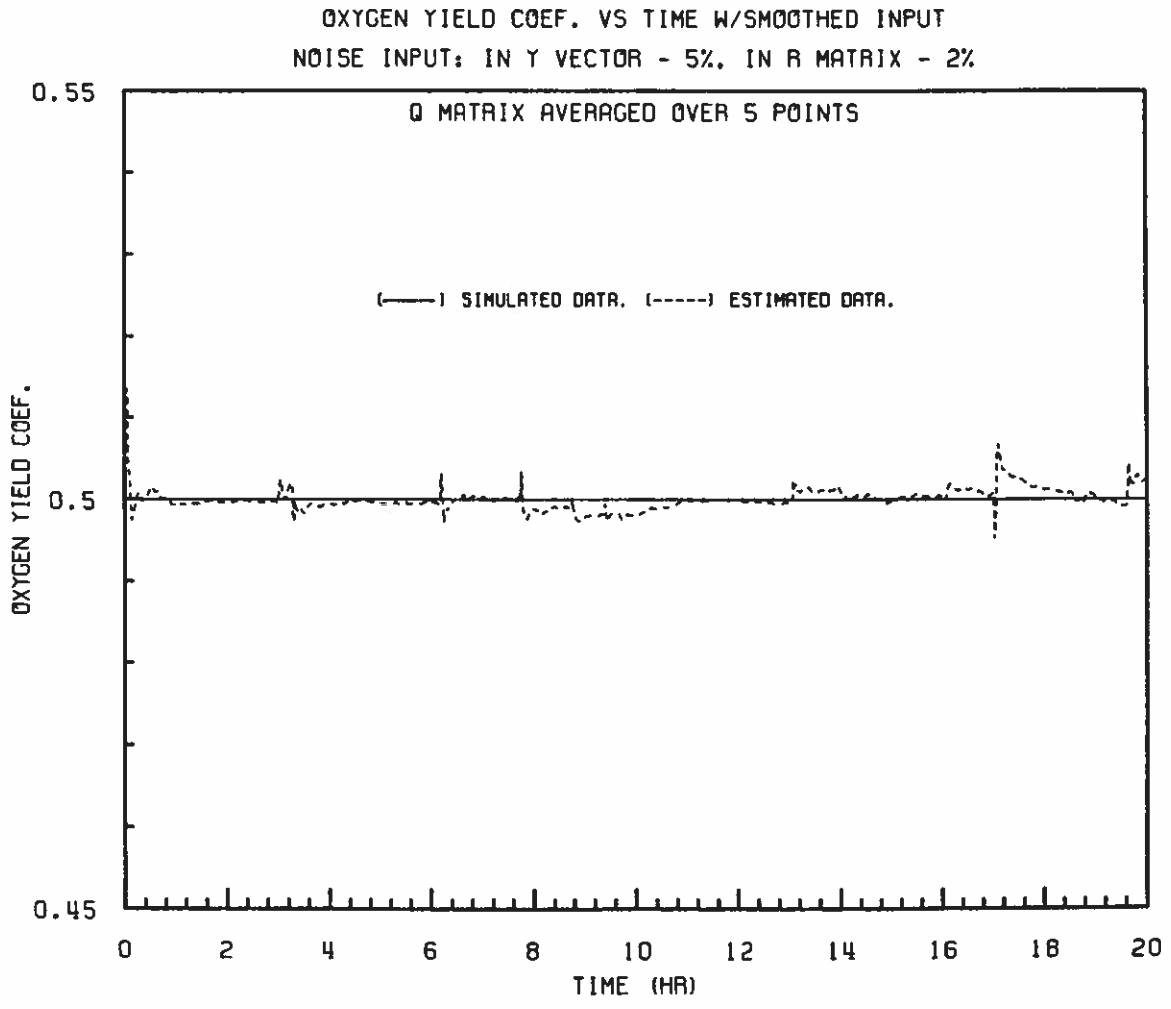

Figure 4g. Oxygen yield coefficient vs time for simulation 1.

9.2 Simulation 2 Figures

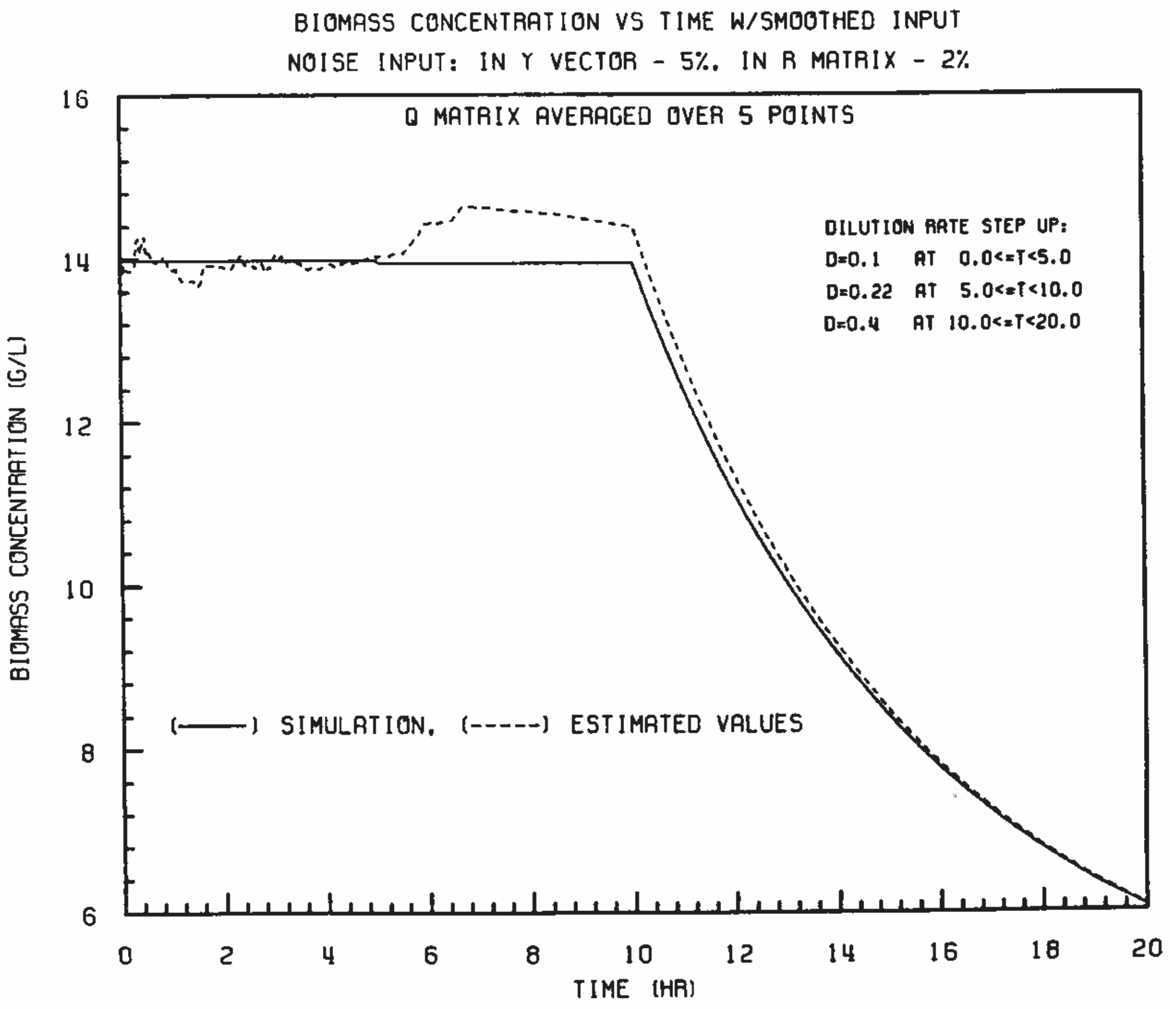

Figure 5a. Biomass concentration vs time for simulation 2. Note the overshoot induced by the step in dilution rate at 6 hours.

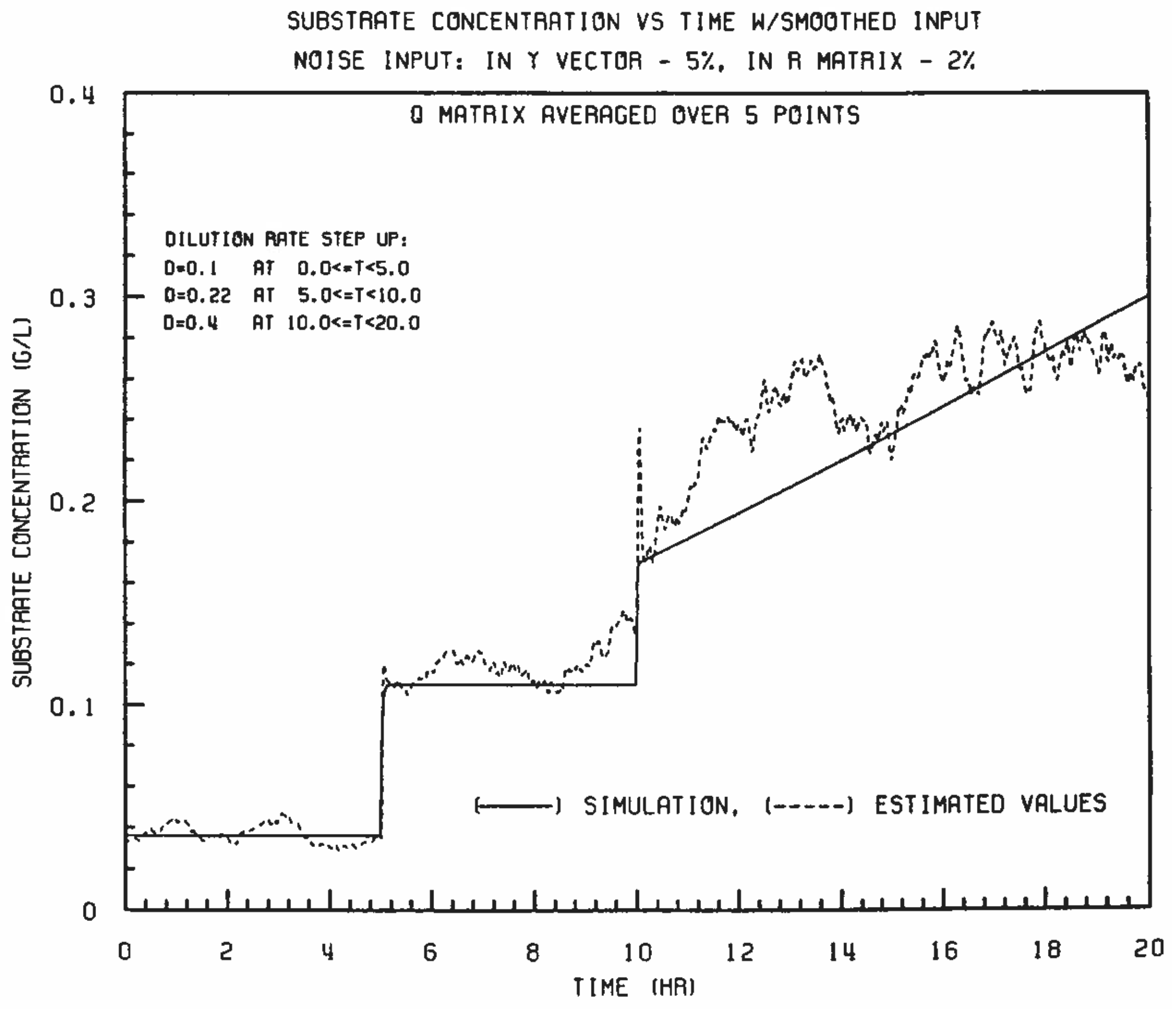

Figure 5b. Substrate concentration vs time for simulation 2. The poor estimate is due to the low substrate concentration and the nonobservability problem.

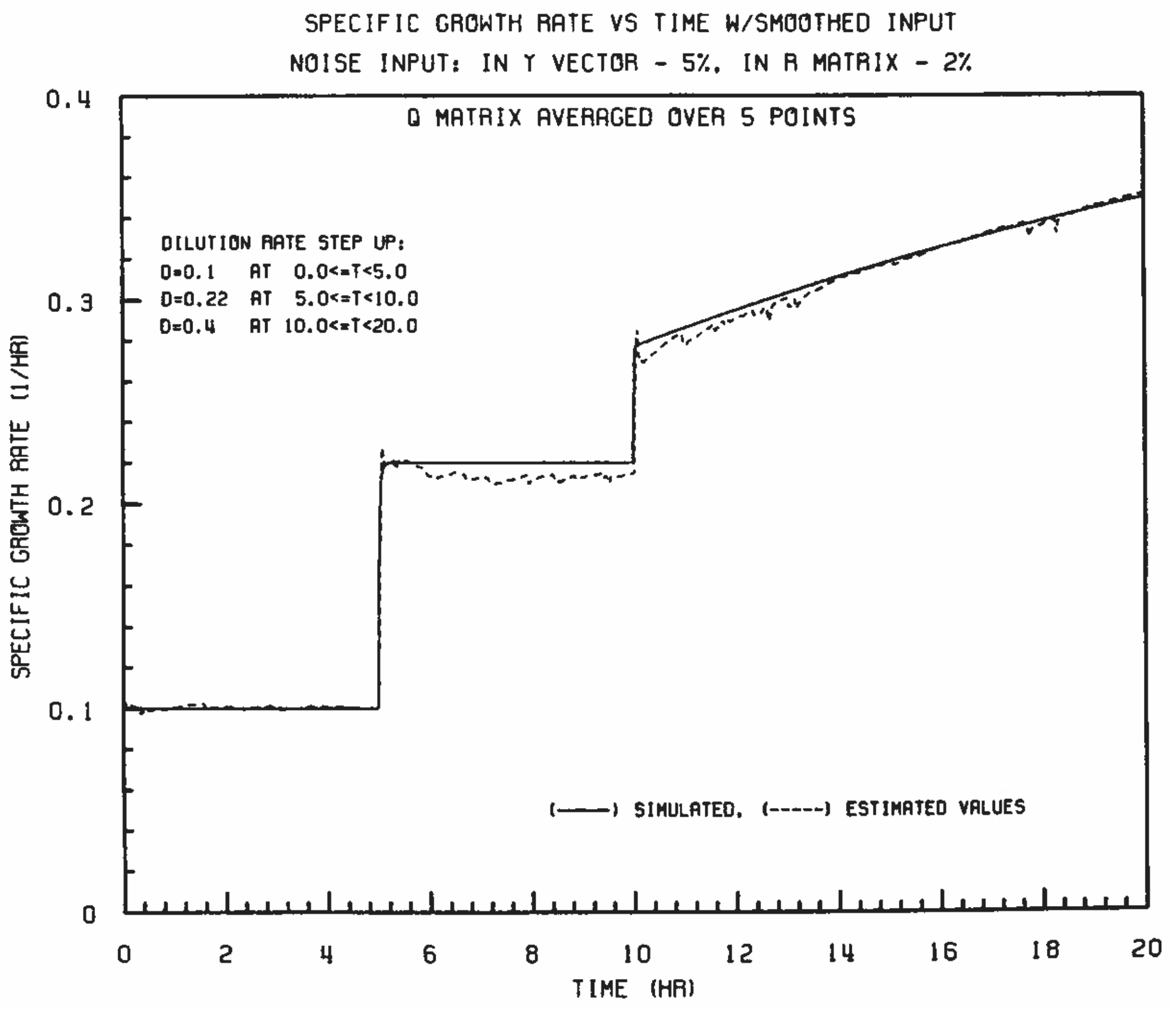

Figure 5c. Specific growth rate vs time for simulation 2. Note undershoot induced by the step in dilution rate.

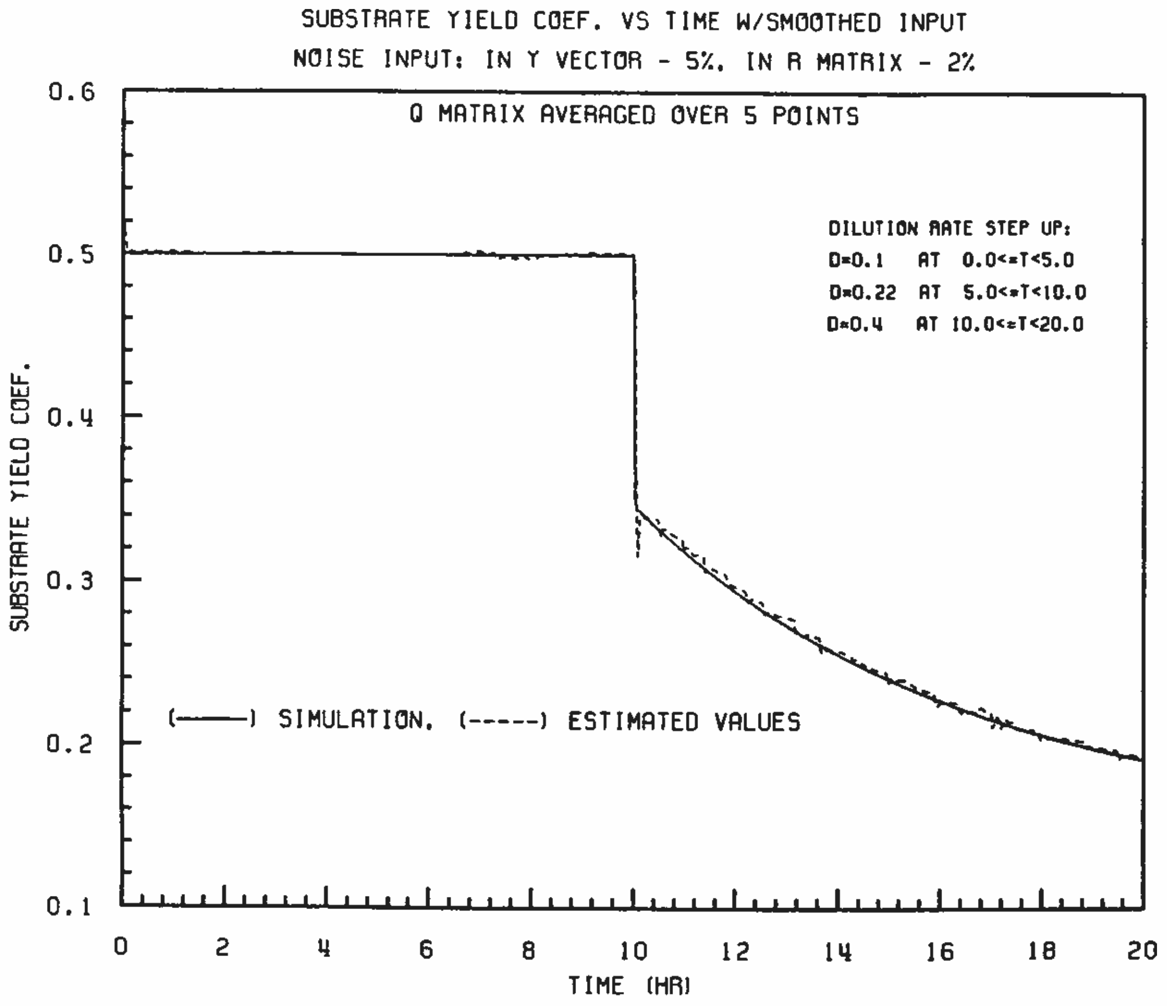

Figure 5d. Substrate yield coefficient vs time for simulation 2.

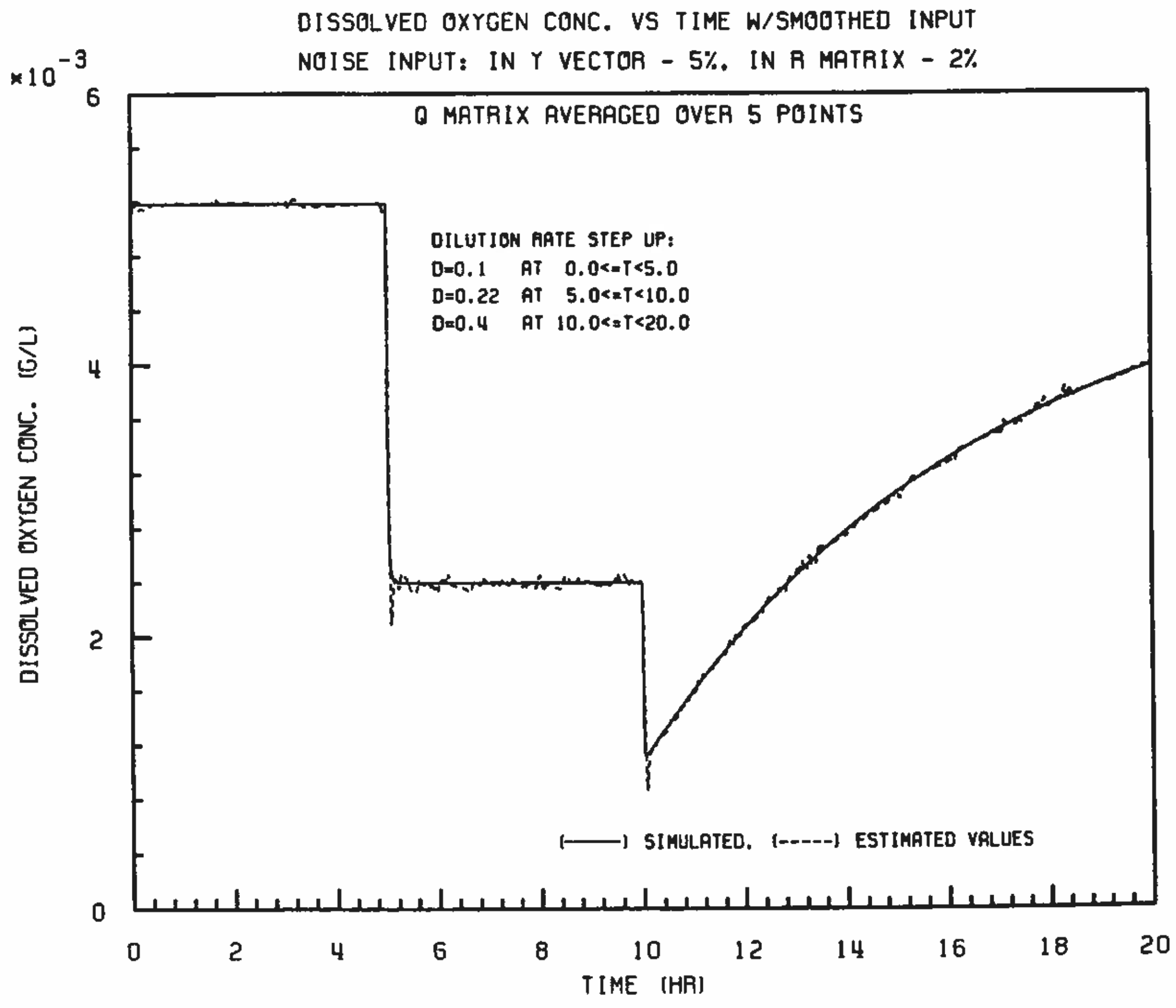

Figure 5e. Dissolved oxygen concentration vs time for simulation 2. This simulation is run at a higher $[O_2]_d$ concentration. Note that less noise passes through the filter.

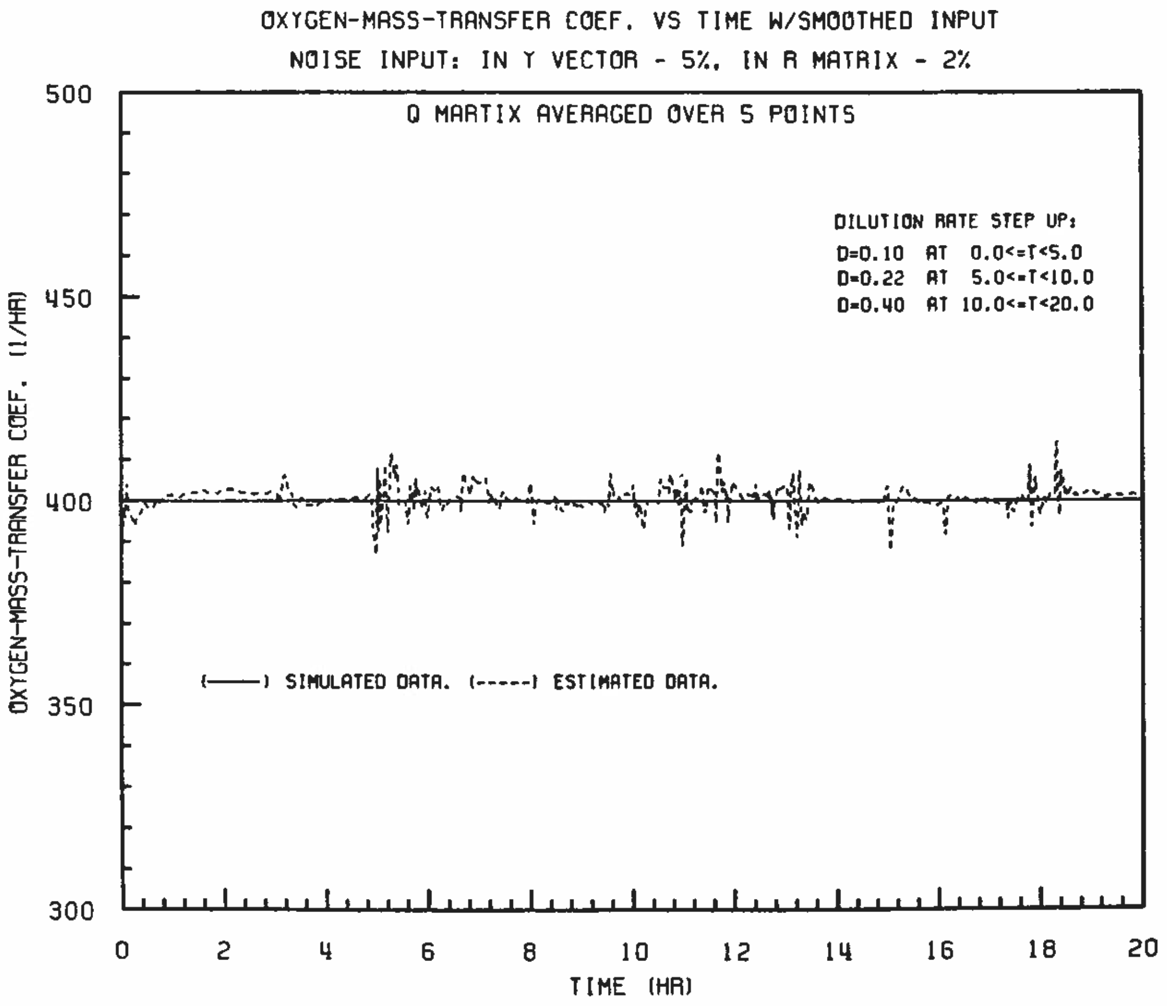

Figure 5f. Oxygen-mass-transfer coefficient vs time for simulation 2. Some noise gets through the filter.

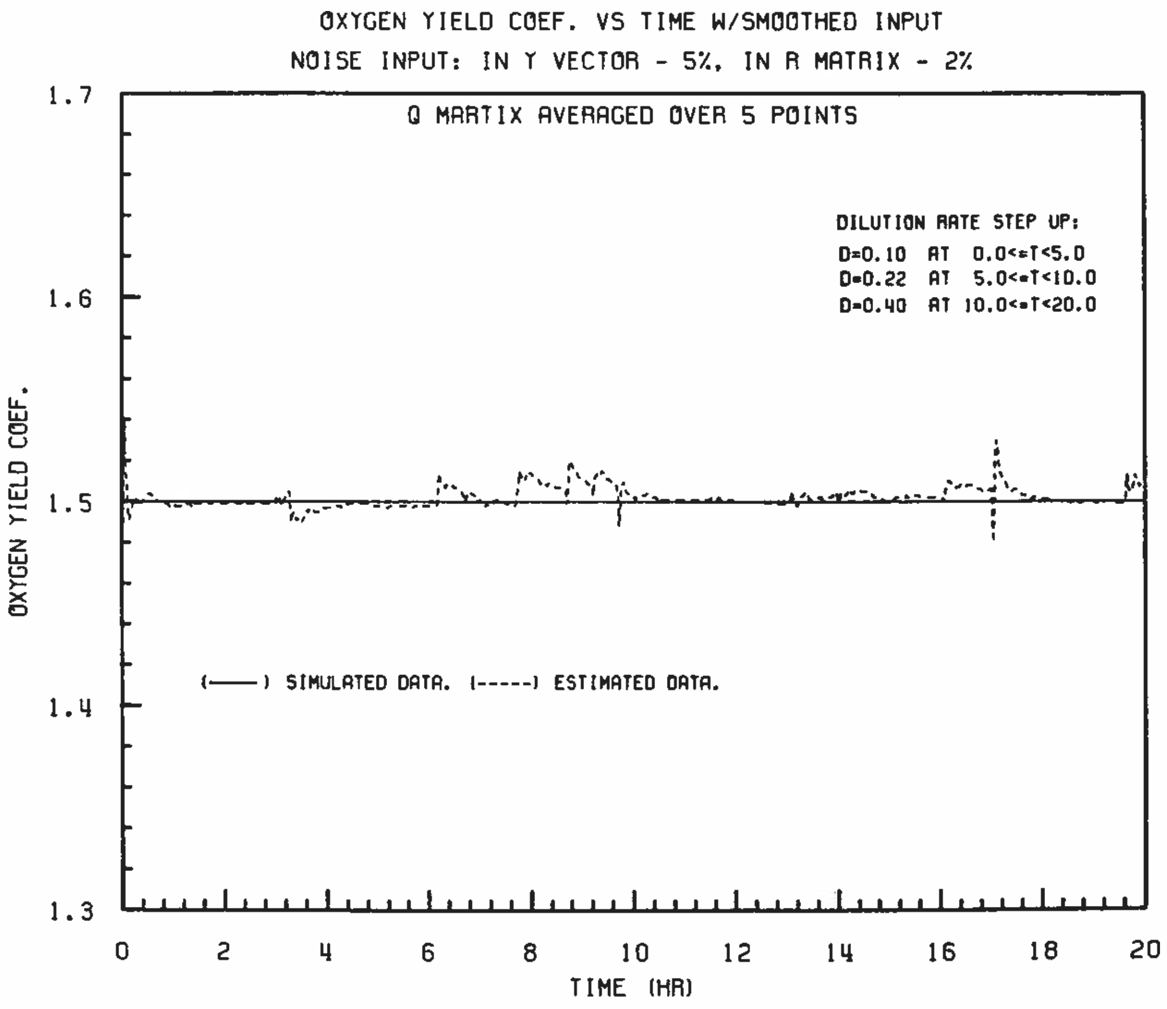

Figure 5g. Oxygen yield coefficient vs time for simulation 2.

9.3 Simulation 3 Figures

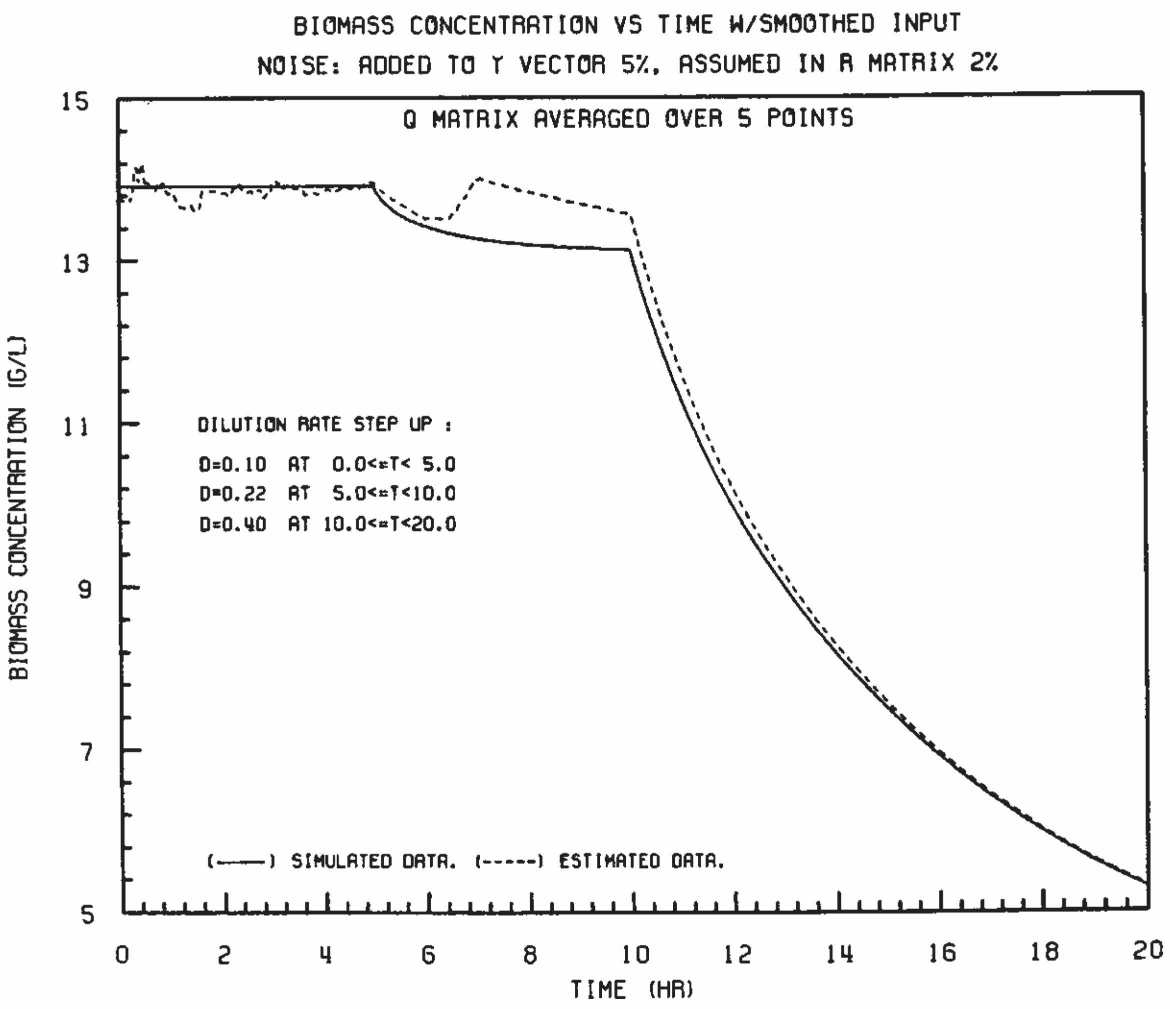

Figure 6a. Biomass concentration vs time for simulation 3. Overshoot is still present.

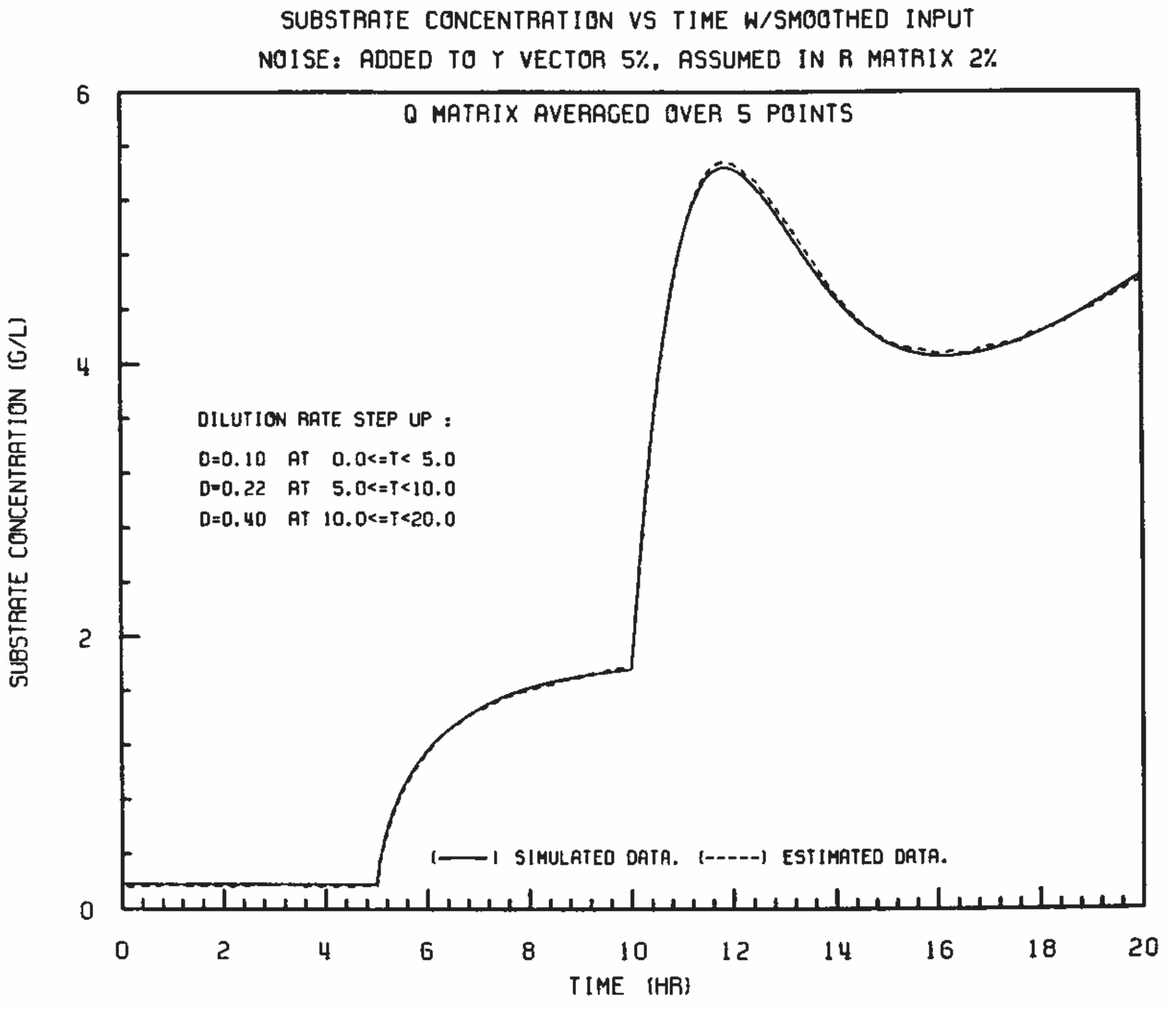

Figure 6b. Substrate concentration vs time for simulation 3. Since the substrate concentration is higher, the noise is much less.

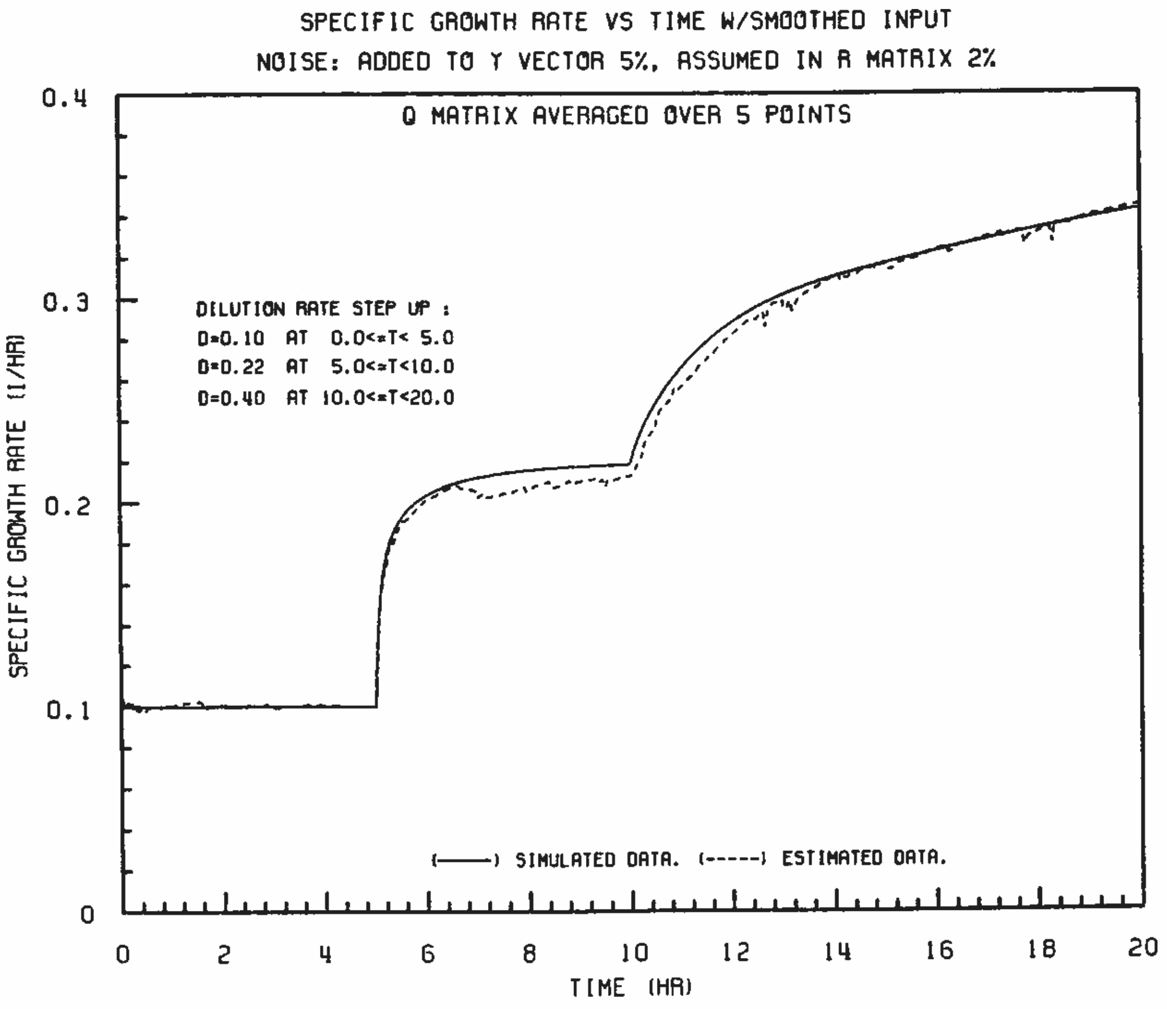

Figure 6c. Specific growth rate vs time for simulation 3. Undershoot still exists.

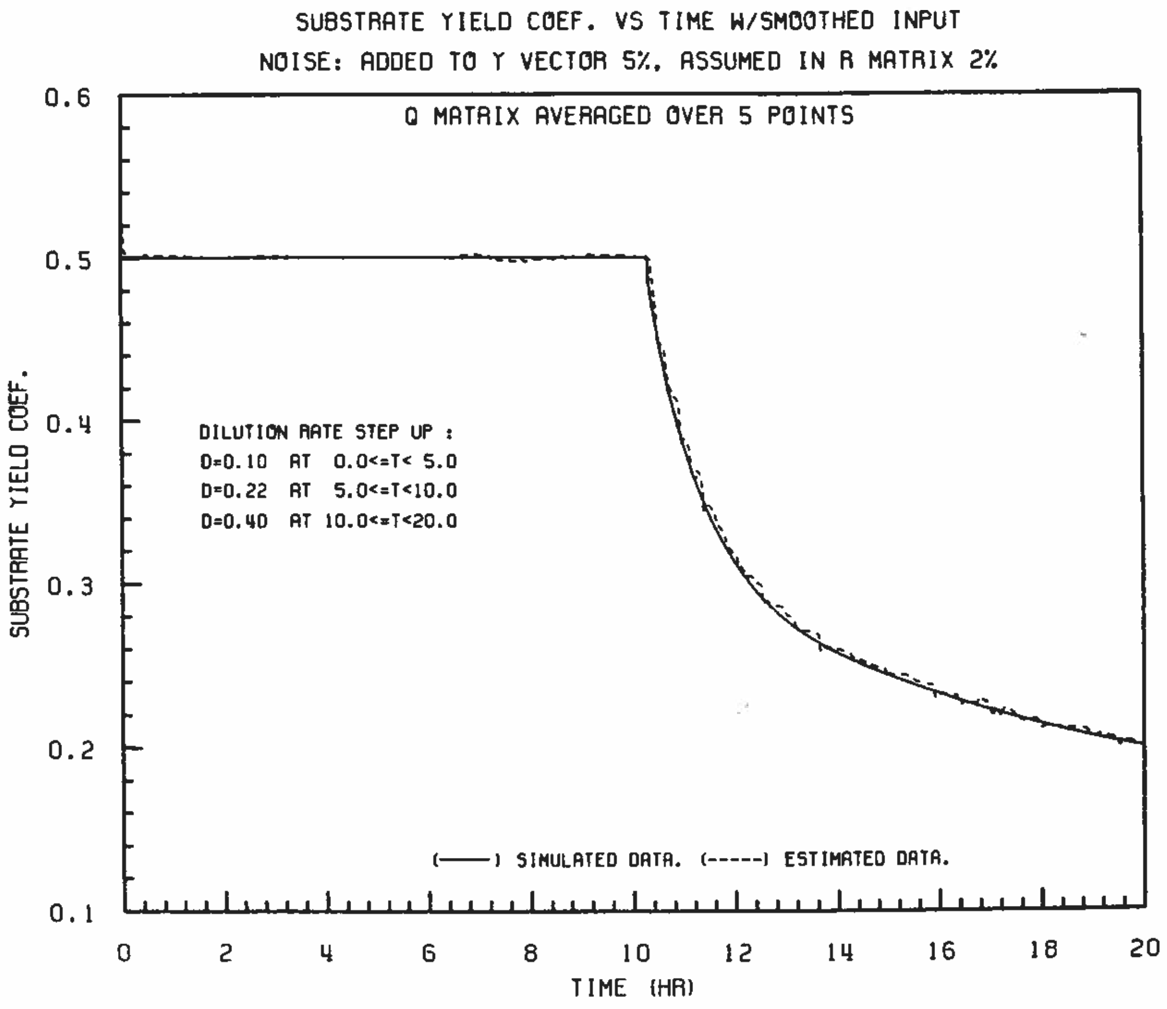

Figure 6d. Substrate yield coefficient vs time for simulation 3.

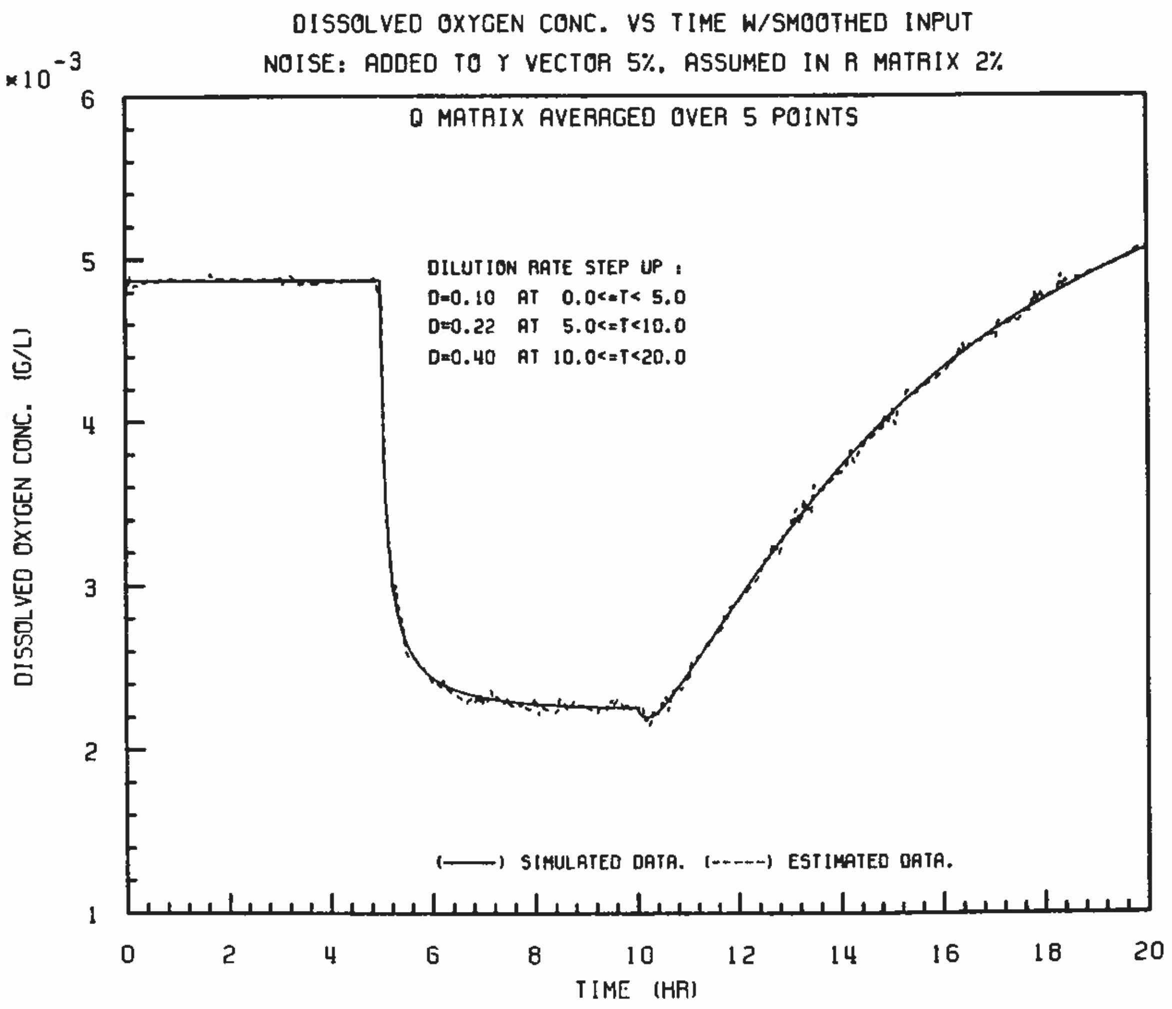

Figure 6e. Dissolved oxygen concentration vs time for simulation 3.

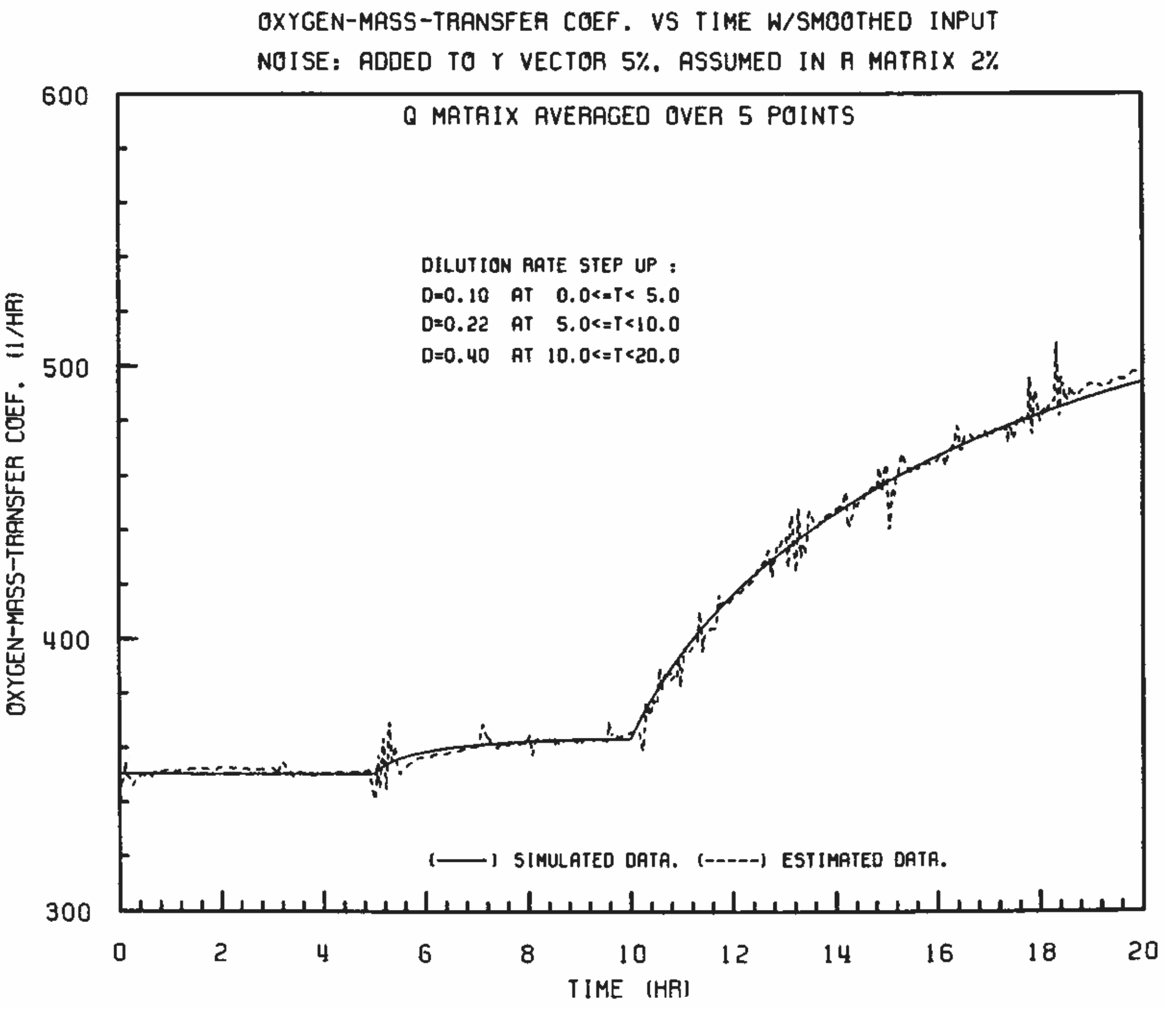

Figure 6f. Oxygen-mass-transfer coefficient vs time for simulation 3. Eventhough the function is not a ramp, the estimate is still quite good.

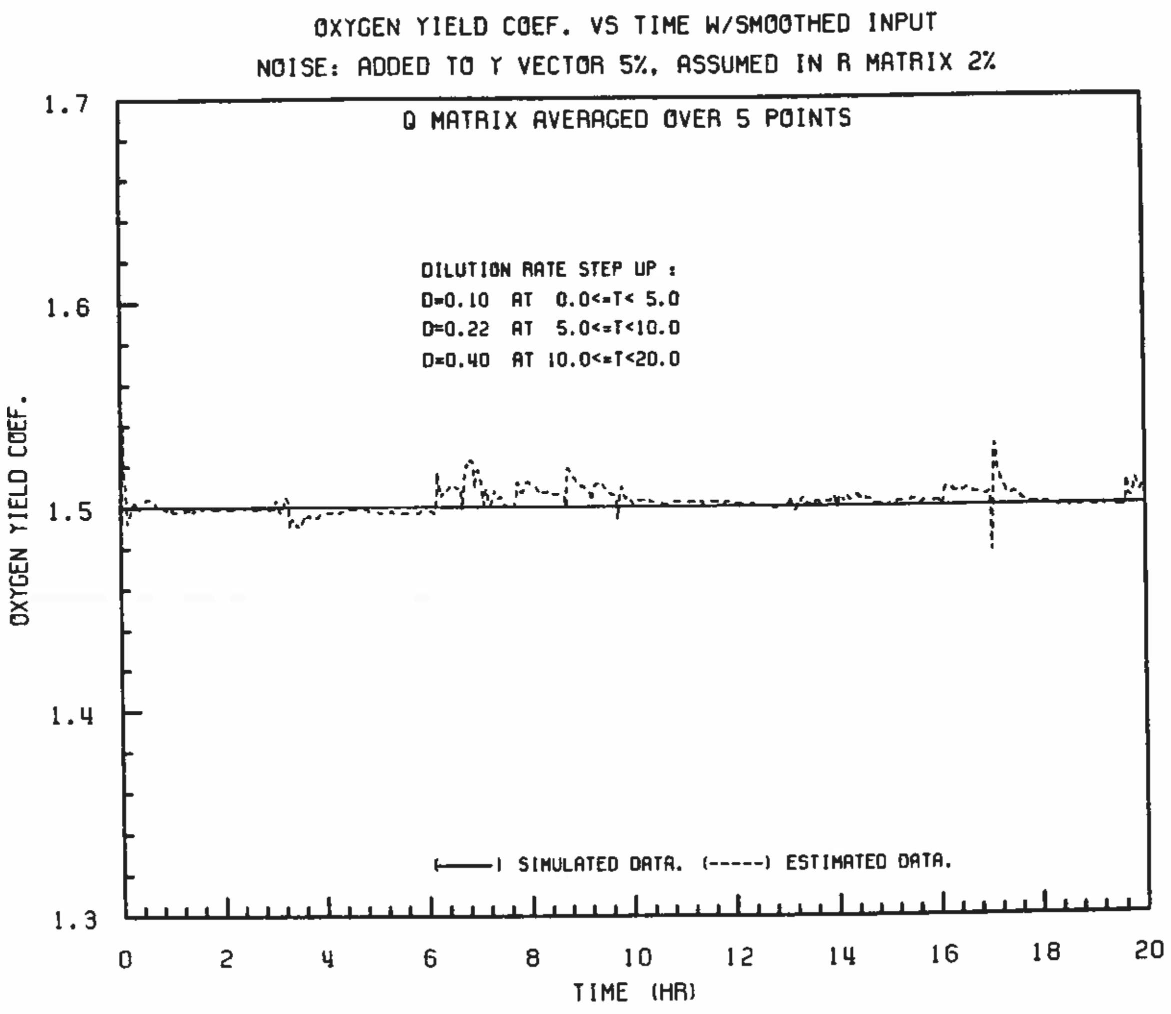

Figure 6g. Oxygen yield coefficient vs time for simulation 3.

9.4 Simulated Measured Variables

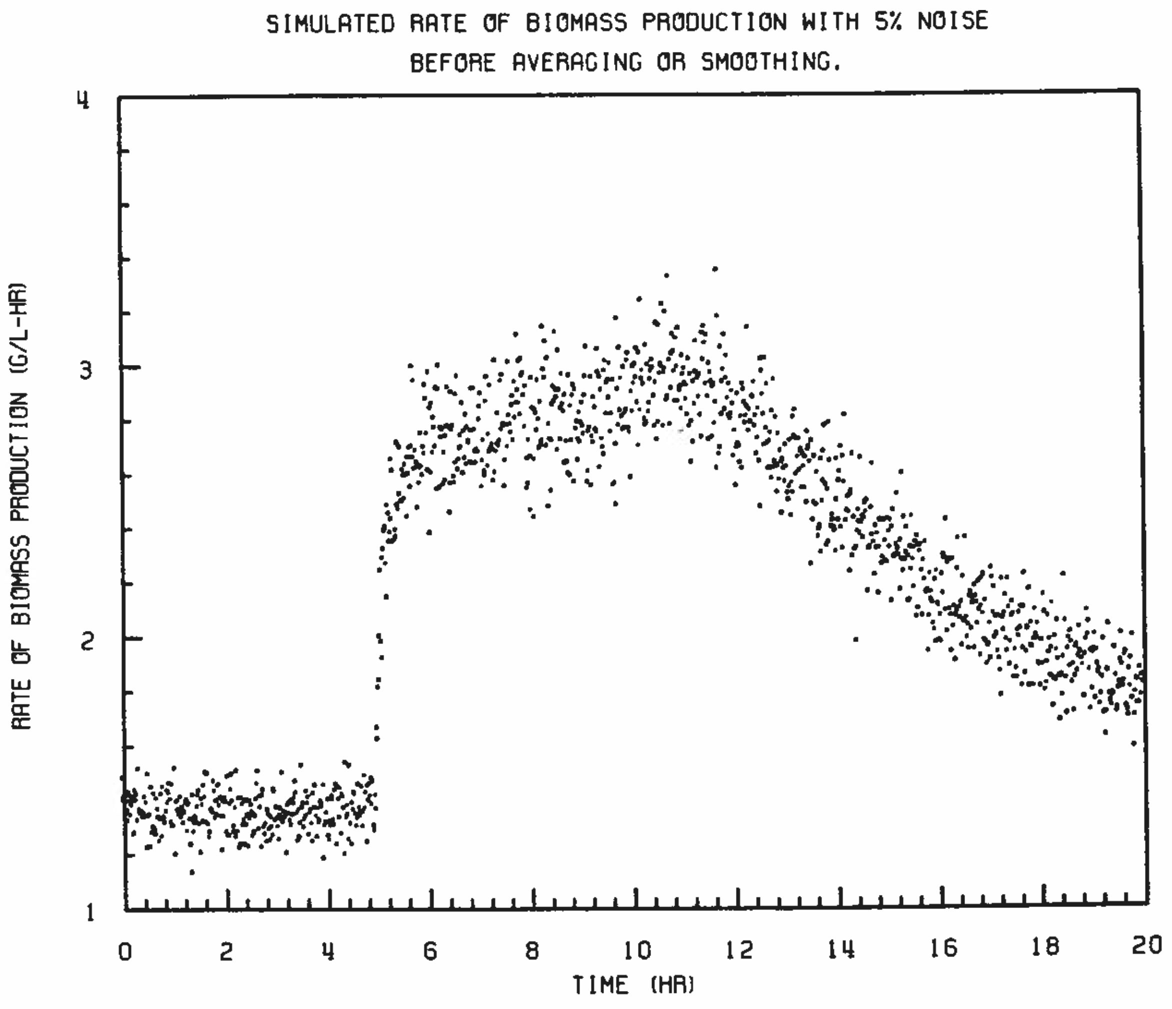

Figure 7a. Rate of biomass production vs time. Actual input to filter for simulation 3. The filter uses this measurement to update μ and b.

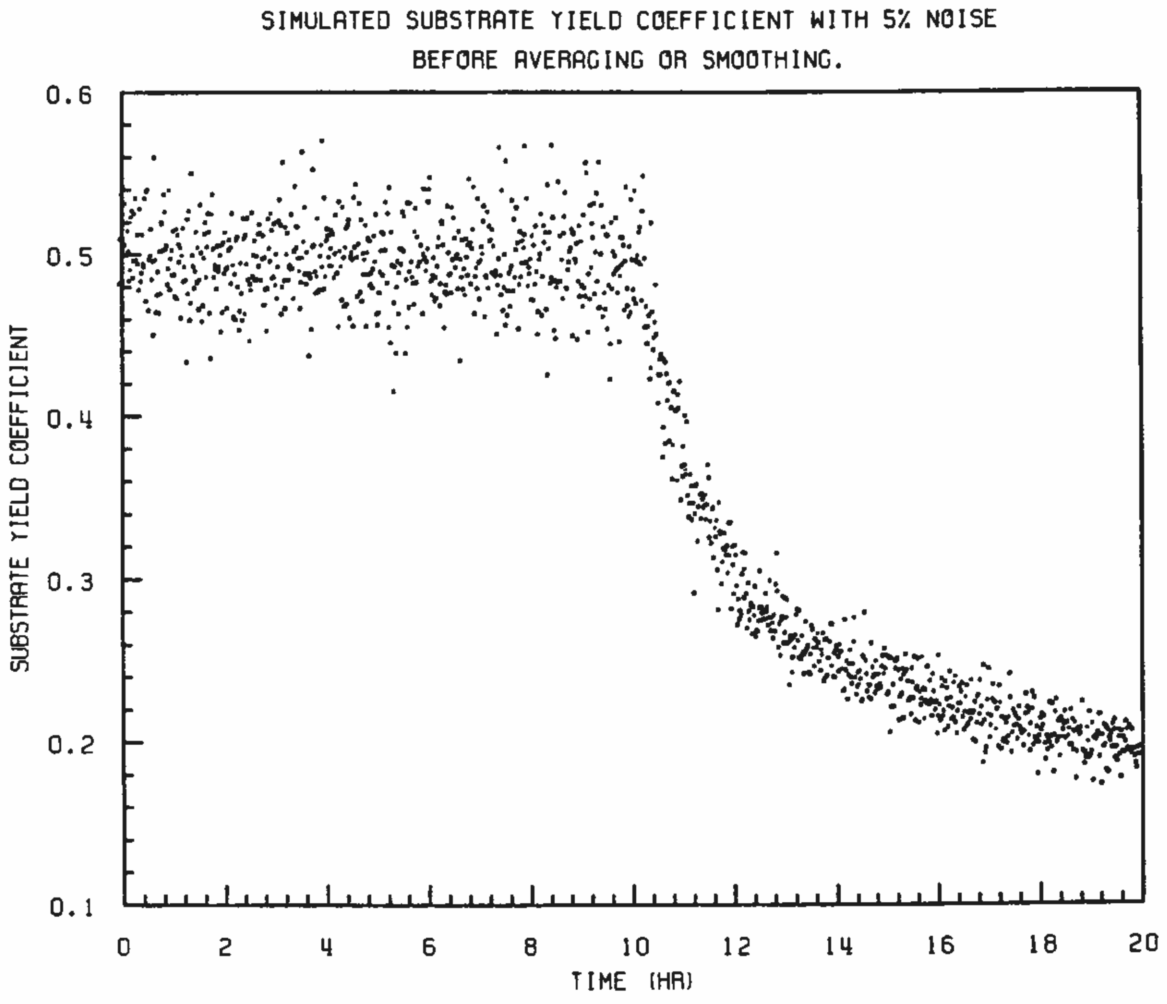

Figure 7b. Substrate yield coefficient vs time. Actual input to filter for simulation 3. The filter uses this measurement to update Y_s. Compare to the filtered value, Figure 6d.

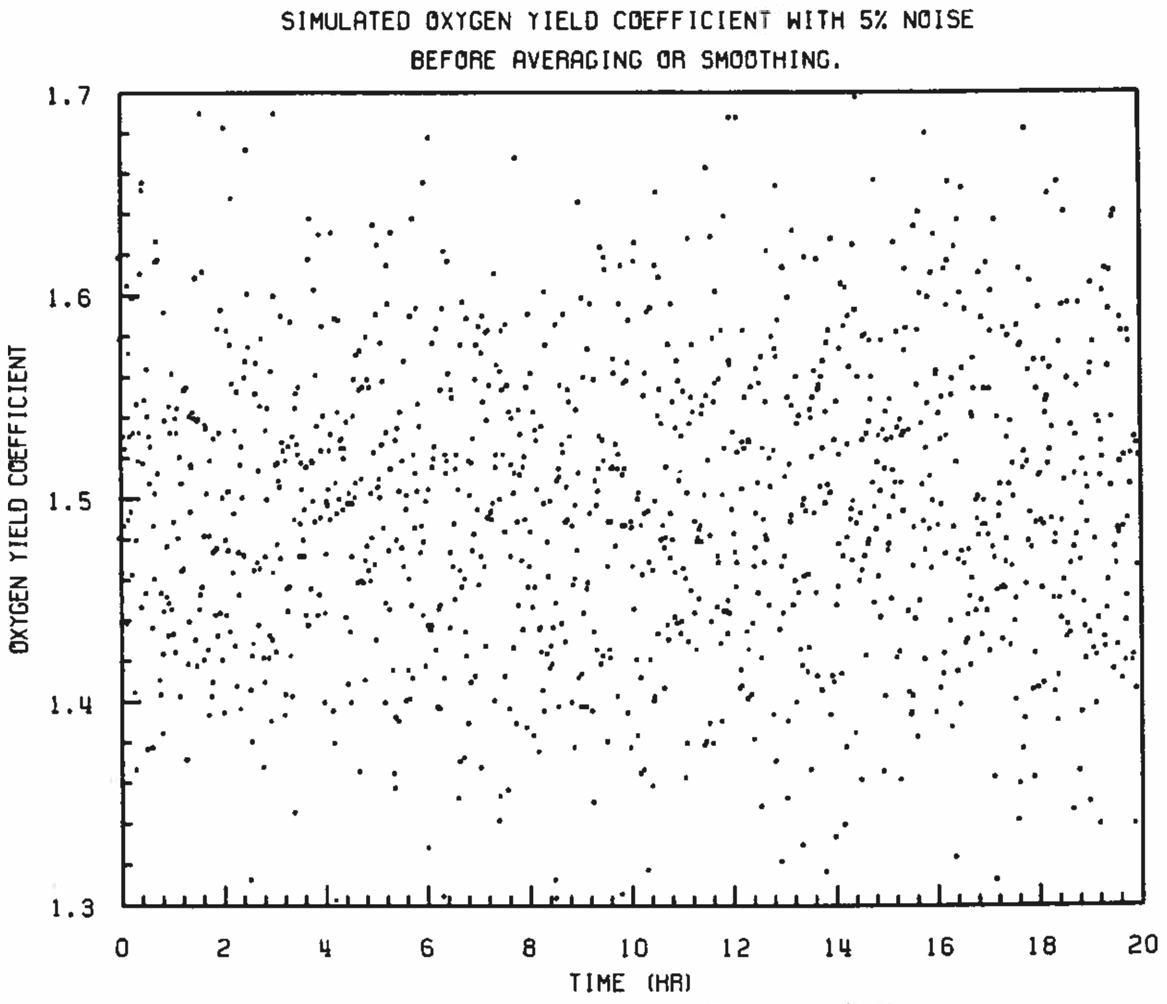

Figure 7c. Oxygen yield coefficient vs time. Actual input to filter for simulation 3. The filter uses this measurement to update Y_{O_g}. The ordinate is the same as in Figure 6g for comparison.

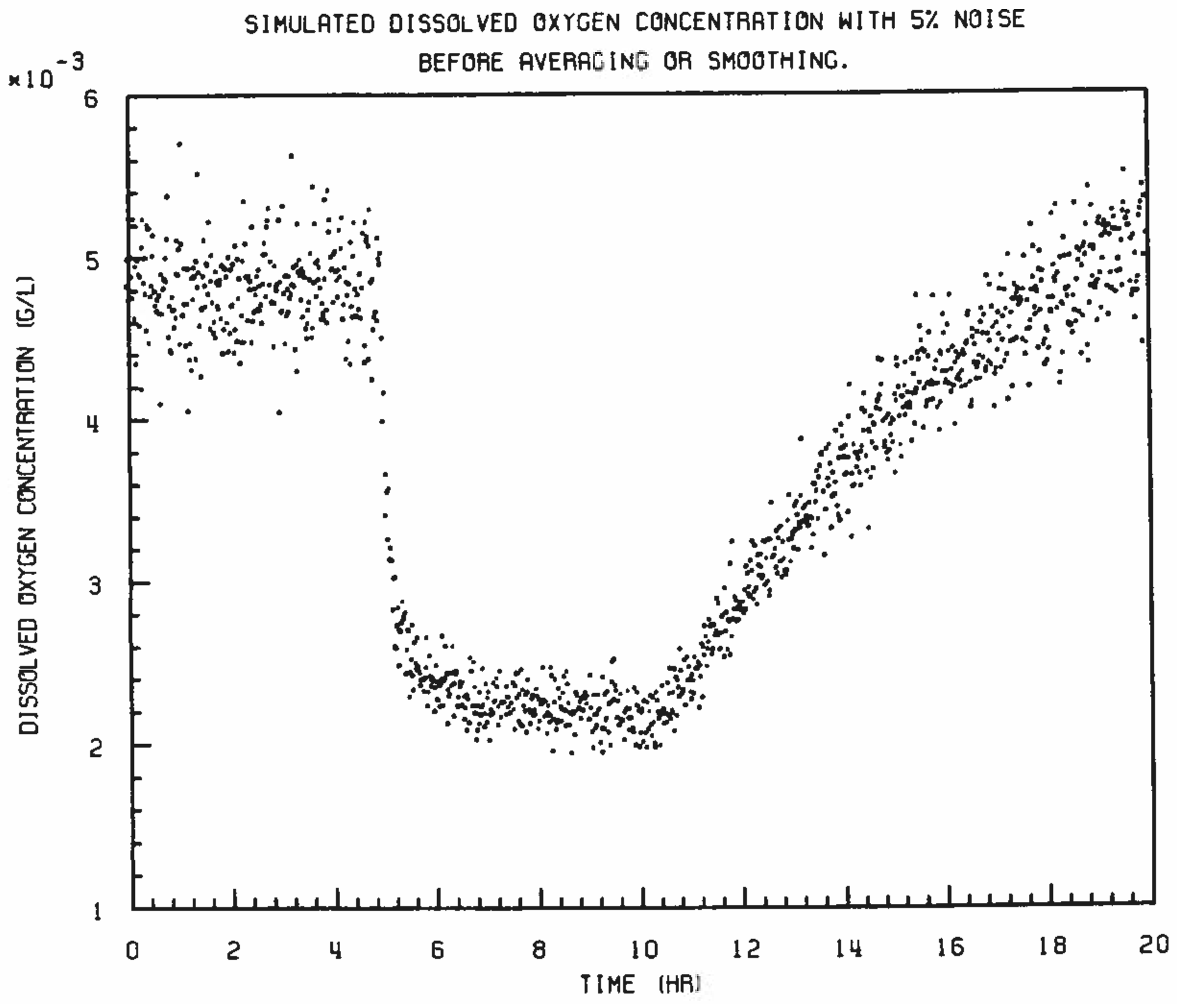

Figure 7d. Dissolved oxygen concentration vs time. Actual input to filter for simulation 3. The filter uses this measurement to update $[O_2]_d$. Compare to the filtered value, Figure 6f.

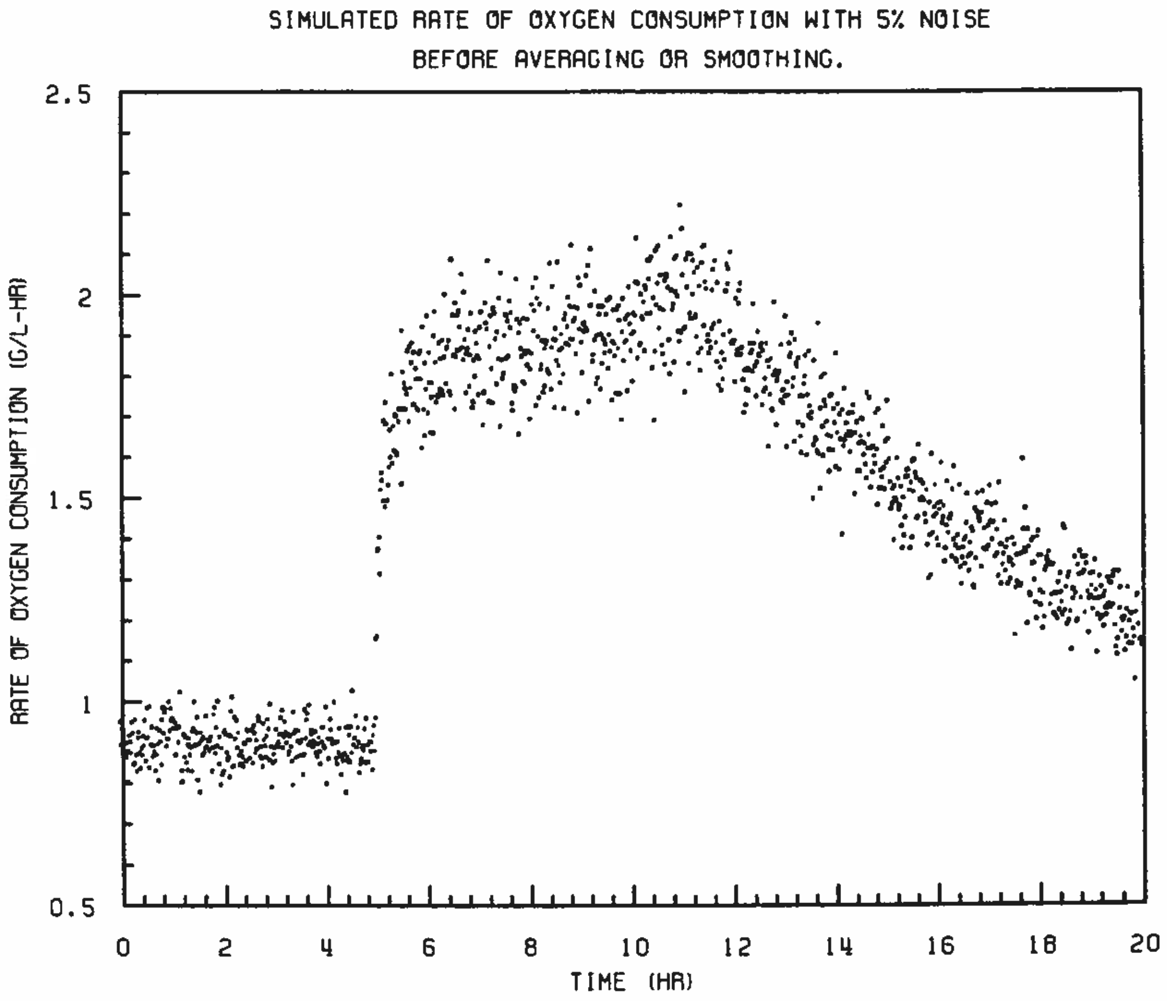

Figure 7e. Rate of oxygen consumption vs time. Actual input to filter for simulation 3. The filter uses this measurement to update $k_L a$ and $[O_2]_d$.

10. APPENDIX II

In this appendix are the FORTRAN listings of all programs used to generate the data presented in Appendix I.

The first listing is the program that is used to simulate the chemostat. This program generates the true state variables from state equations provided by the user. From these state variables the measured variables are produced which the Kalman filter program uses as input. The measurement equations must also be provided by the user.

The Kalman filter program reads the measured variables and produces the estimated state variables. The user must provide the state equations.

The third listing calculates the exponential of a matrix and is called by the filter program to calculate the state transition matrix.

All three programs were run on a VAX-11/780 main frame computer made by Digital Equipment Corporation. The programs are written in VAX-11 FORTRAN version 3.0. Also the following subroutines from The IMSL Library are called

DGEAR:	Differential equation solver for stiff systems
GGNQF:	Normal random diviate generator
LINV2F:	Inversion of a matrix, high accuracy solution
VMULFF:	Matrix multiplier $\mathbf{C} = \mathbf{AB}$
VMULFM:	Matrix multiplier $\mathbf{C} = \mathbf{A}^T\mathbf{B}$
VMULFP:	Matrix multiplier $\mathbf{C} = \mathbf{AB}^T$

10.1 Simulation Program

```
C This program simulates a chemostat producing state variables
C and output w/noise (measured) variables
C This program is capable of generating 20 state variable and 20 measured
C variables for a chemostat.
C
C This program is set up for batch operation.
C The following information must be stored in file FOR006
C in the order shown:
C               TSTOP            The stopping time of the program (hrs)
C               DELT             Delta time between stored data   (hrs)
C               NOSAMP           The no. of samples taken between stored data
C                                which are averaged to produce one point every
C                                DELT hours
C               ERROR(1-NY)      Relative error for each Y vector
C
C Also the files X1TO10 and PARAMETER must be initialized with the
C initial values of X and PARAM
C
C The state equations that specify the system are stored by the user in
C the subprogram TRUE.  See this subroutine for further comments.
C
C The true states generated in this program are stored in files TX1TO10.DAT
C and TX11TO20.DAT. The output with noise is stored in files Y1TO10.DAT
C and Y11TO20.DAT.
C To each measured variable white noise with a gaussian distribution and
C a zero mean is added.  The magnitude of the noise is such that the
C standard deviation of the variable is ERROR(I)% of its mean.
C
C The variable DELT corresponds to the sample frequency of the filter f sub 1
C while the expression DELT/NOSAMP corresponds to the sample frequency of the
C measuring device, f sub 2.
C
C The important variables are as follows:
C               NX               :==     State dimension
C               NY               :==     Measurement dimension
C               Y(I)             :==     Measurement vector
C               AVEY(I)          :==     Y vector averaged over NOSAMP points
C               PARAM(I)         :==     Vector which contains the parameters
C                                        used by the state equations
C               NP               :==     No. of parameters
C               AVEPAR(I)        :==     PARAM vector averaged over NOSAMP
C                                        points
C               X(I)             :==     Vector for those state which require
C                                        integration.
C               STATE(I)         :==     State vector including X(I) states.
C               NDIF             :==     No. of state variables the require
C                                        integration.
C
C
C
      IMPLICIT REAL*8 (A-H,O-Z)
      EXTERNAL DIFEQN,JACOB
      DIMENSION X(20),STATE(20),Y(20),C(800),IWK(20),PARAM(10)
      DIMENSION PD(20,20),AVEPAR(20),AVEY(20)
      CHARACTER*3 YESNO
      COMMON PARAM,STATE
      MS=20
C
C STEP 1.  Initialize system.
C
      READ (5,*) TSTOP
      READ (5,*) DELT
      READ (5,*) NOSAMP
      CALL TRUE (X,MS,NX,NY,NDIF,NP)
C
C Output information to FOR006 for conformation
```

```
      WRITE(6,1) TSTOP,DELT,NOSAMP
 1    FORMAT(' STOPPING TIME = ',F8.5,/,' DELTA TIME = ',F7.5,
     & /,' NO. OF SAMPLES = ',I3)
C
      TINT=Ø.Ø          !
      METH=2            !
      TOL=Ø.ØØØØ1       !  All information required by the
      STEP=Ø.ØØØØØ1     !  numerical integrator DGEAR.
      INDEX=1           !
      MITER=2           !
      SAMPLE=DELT/NOSAMP
      OPEN(UNIT=1,FILE='TX1TO1Ø',STATUS='OLD')
      OPEN(UNIT=9,FILE='Y1TO1Ø',STATUS='NEW')
      OPEN(UNIT=3,FILE='PARAMETER',STATUS='OLD')
      IF (NX .GT. 1Ø) THEN
         OPEN(UNIT=4,FILE='TX11TO2Ø',STATUS='OLD')
      END IF
      IF (NY .GT. 1Ø) THEN
         OPEN(UNIT=7,FILE='Y11TO2Ø',STATUS='NEW')
      END IF
      DO 2 I=1,NP
         AVEPAR(I)=PARAM(I)
 2    CONTINUE
C
C STEP 2.  Store state and parameters.
C
 5    IF (NX .LE. 1Ø) THEN
         WRITE(1,1Ø) TINT,(STATE(I),I=1,NX)
      ELSE
         WRITE(1,1Ø) TINT,(STATE(I),I=1,1Ø)
         WRITE(4,1Ø) TINT,(STATE(I),I=11,NX)
      END IF
      WRITE(3,2Ø) (AVEPAR(I),I=1,NP)
 1Ø   FORMAT(1X,F8.5,1Ø(1X,E11.4))
 2Ø   FORMAT(1X,1Ø(E11.4,1X))
C
C STEP 3.  Check stopping time.
C
      IF (TINT .GE. TSTOP) GOTO 4Ø
      ISAMPLE=Ø
      DO 22 I=1,NP
         AVEPAR(I)=Ø.Ø
 22   CONTINUE
      DO 23 I=1,NY
         AVEY(I)=Ø.Ø
 23   CONTINUE
C
C STEP 4.  Solve differential eqns.
C
 24   TEND=TINT + SAMPLE
      IF (NDIF .EQ. Ø) GOTO 35
      CALL DGEAR (NDIF,DIFEQN,JACOB,TINT,STEP,X,TEND,TOL,METH,MITER,
     * INDEX,IWK,C,IER)
      IF (IER .GT. 128) THEN
         WRITE(6,25) IER,STEP,TOL,INDEX,TINT
 25      FORMAT(' FAILURE IN DGEAR AT STEP 4. IER=',I3,/,' STEP='
     *   ,F8.6,/,' TOL=',F8.6,/,' INDEX=',I2,/,' TIME=',F8.6)
         STOP
      END IF
      IF (IER .GT. Ø) THEN
         WRITE(6,32) IER,STEP,TINT
 32      FORMAT(' **WARNING IN DGEAR AT STEP 4.**   IER=',I3,
     *   /,' STEP=',F8.6,/,' TIME=',F8.6)
      END IF
 35   CONTINUE
C
```

```
C STEP 5. Input data into state vector and parameters.
C
        CALL ALGEQN (TEND,X)
C
C STEP 6. Calculate output vector w/noise.
C
        CALL OUTPUT (Y)
C
C STEP 7.  Calculate the average values of the Y vector and PARAM
C             and check to see if enough samples have been taken
C
        DO 37 I=1,NY
           AVEY(I)=AVEY(I)+Y(I)/NOSAMP
 37     CONTINUE
        DO 38 I=1,NP
           AVEPAR(I)=AVEPAR(I)+PARAM(I)/NOSAMP
 38     CONTINUE
        ISAMPLE=ISAMPLE+1
        IF (ISAMPLE .EQ. NOSAMP) GOTO 39
        GOTO 24
C
C STEP 8. Store outpur data vector.
C
 39     IF (NY .LE. 1Ø) THEN
           WRITE(9,1Ø) TEND,(AVEY(I),I=1,NY)
        ELSE
           WRITE(9,1Ø) TEND,(AVEY(I),I=1,1Ø)
           WRITE(7,1Ø) TEND,(AVEY(I),I=11,NY)
        END IF
        GOTO 5
C
C EXIT PROGRAM.
C
 4Ø     WRITE(6,5Ø) TINT
 5Ø     FORMAT(' STOPPED AT TIME =',F8.5)
        CLOSE(UNIT=1)
        CLOSE(UNIT=9)
        CLOSE(UNIT=3)
        IF (NX .GT. 1Ø) CLOSE(UNIT=4)
        IF (NY .GT. 1Ø) CLOSE(UNIT=7)
        STOP
        END



C This subroutine is used to generate the necessary eqns for the
C simulation program.
C
C A discription of each entry section is described as follows
C
C ENTRY TRUE:
C       This routine is used to initialize the main program variables
C       such as NX, NY, NP, STATE(i), and PARAM(i).
C
C ENTRY DIFEQN:
C       In this routine all state equation that need to be integrated are
C       stored here.  XDOT is the derivative of X.
C       Note if NDIF is zero then this subroutine is by passed
C
C ENTRY JACOB:
C       This is a dummy routine used by DGEAR.
C
C ENTRY ALGEQN:
C       This routine has all algebraic equation used to describe the state.
```

```
C       If NDIF equals zero then the state of the system is completely
C       generated by a set of "algebraic" equations
C
C ENTRY OUTPUT:
C       In this subroutine all of the measurement equation are stored.
C       These equations are used to generate the measurement vector
C       from the true state varibles.  Also in this routine, the noise
C       is added to corrupt the measurements.
C
C **************** PROGRAM STATS HERE ********************


        SUBROUTINE TRUE (X,MS,NX,NY,NDIF,NP)
        IMPLICIT REAL*8 (A-H,O-Z)
        REAL*8 KLA,KS,KO
        DIMENSION X(1),STATE(2Ø),PARAM(1Ø),XDOT(1),Y(1),PD(MS,MS)
        DIMENSION ERROR(2Ø)
        COMMON PARAM,STATE
        IFLAG=Ø.Ø
        NX=9
        NY=5
        NDIF=3
        NP=3
        DSEED=123579.Ø
C
C Read relative error for Y vector from FORØØ5
        READ(5,*) (ERROR(I),I=1,NY)

C Output data to file FORØØ6 for conformation
        WRITE(6,*) 'THE FOLLOWING ERRORS ARE ASSUMED FOR THE Y VECTOR:'
        DO 2 I=1,NY
           WRITE(6,1) I,ERROR(I)
 1         FORMAT(' ERROR(',I2,') = ',E9.2)
 2      CONTINUE


        OPEN (UNIT=1,FILE='TX1TO1Ø',STATUS='OLD')
        OPEN (UNIT=7,FILE='PARAMETER',STATUS='OLD')
        READ(1,*) (STATE(I),I=1,NX)
        READ(7,*) (PARAM(I),I=1,NP)
        CLOSE(UNIT=1)
        CLOSE(UNIT=7)
        B=STATE(1)
        S=STATE(2)
        U=STATE(3)
        C1=STATE(4)
        YS=STATE(5)
        O2D=STATE(6)
        KLA=STATE(7)
        C2=STATE(8)
        YO2=STATE(9)
C
        D=PARAM(1)
        SF=PARAM(2)
        O2MAX=PARAM(3)
C
        X(1)=B
        X(2)=S
        X(3)=O2D
C
        M=NX
        N=NY
        RETURN
C
C This part is used to solve any differential eqns in the simulation.
C
```

```
        ENTRY DIFEQN (NDIF,T,X,XDOT)
        IFLAG=1
        GOTO 7
 5      CONTINUE
C
C Three dif eqns: one for B, one for S, and one for [O2].
C
        XDOT(1)=X(1)*(STATE(3)-PARAM(1))
        XDOT(2)=PARAM(1)*(PARAM(2)-X(2))-STATE(3)*X(1)/STATE(5)
        XDOT(3)=STATE(7)*(PARAM(3)-X(3))-STATE(3)*X(1)/STATE(9)
        IFLAG=Ø
        RETURN
C
C This is a dummy subroutine used by DGEAR.
C
        ENTRY JACOB (MS,T,X,PD)
        RETURN
C
C This part is used to solve any algabraic eqns in the simulation.
C
        ENTRY ALGEQN (T,X)
 7      CONTINUE
C
C Set parameters to desired value
C
        PARAM(1)=Ø.1
        IF (T .GE. 5.Ø .AND. T .LT. 1Ø.Ø) PARAM(1)=Ø.22
        IF (T .GE. 1Ø.Ø) PARAM(1)=Ø.4
C
C Set state variables
        UMAX=Ø.53
        KS=Ø.5
        KO=2.ØE-3
C  SET BIOMASS CONC.
        STATE(1)=X(1)
C  SET SUBSTRATE CONC.
        STATE(2)=X(2)
C  SET SPECIFIC GROWTH RATE
        STATE(3)=UMAX*X(2)*X(3)/((KS+X(2))*(KO+X(3)))
C  SET SUBSTRATE YIELD COEF.
        U=STATE(3)
        IF (U .LE. Ø.Ø6) STATE(5)=8.333333*U
        IF (U .GT. Ø.24) THEN
           STATE(5)=3.Ø9Ø4-18.732Ø*U+39.5Ø59*U**2-27.588*U**3
        ELSE
           STATE(5)=Ø.5
        END IF
C  SET DISSOLVED OXYGEN CONC.
        STATE(6)=X(3)
C  SET OXYGEN MASS TRANSFER COEF.
        STATE(7)=35Ø.Ø+15Ø.Ø*((13.936-X(1))/9)
C  SET OXYGEN YIELD COEF.
        STATE(9)=1.5
C
C  IF ENTERRED FROM DGEAR THEN RETURN
        IF (IFLAG .EQ. 1) GOTO 5
        RETURN
C
C This part set output eqns and adds noise to the output.
C
        ENTRY OUTPUT (Y)
        Y(1)=STATE(1)*STATE(3)
        Y(2)=STATE(5)
        Y(3)=STATE(9)
        Y(4)=STATE(6)
        Y(5)=STATE(7)*(PARAM(3)-STATE(6))
```

```
C
C Now add white noise to the output
C
        DO 1Ø I=1,N
           SDEV=ERROR(I)*Y(I)
           Y(I)=Y(I) + GGNQF(DSEED) * SDEV
 1Ø     CONTINUE
        RETURN
        END
```

10.2 Kalman Filter Program

```
C Adaptive-Extended Kalman Filter Program
C For a batch process
C
C This program can except up to 2Ø state differential eqns, 2Ø measured
C eqns and 1Ø parameters in those differential eqns.
C The state and measured eqns along with their associated matrices are
C place in subroutine SYSTEM, see that subroutine for details.
C The following defines program variables:
C
C       M               :==     Actual dimension of the state vector
C       N               :==     Actual dimension of the measured vector
C       X(i)            :==     State after updated from measurement.
C       XPRED(i)        :==     State predicted by state D.E eqns.
C       P(i,j)          :==     State error covariance matrix after updating
C       PPRED(i,j)      :==     State error covariance matrix predicted
C                               by difference eqn.
C       Y(i)            :==     Measured vector.
C       YMAT(i,j)       :==     Matrix used in smoothing of Y vector
C       YHAT(i)         :==     Measured vector predicted via XPRED(i)
C       YMAX            :==     If the residuals exceed YMAX*YHAT(i) then the
C                               residual are set to YMAX*YHAT(i) in the
C                               determination of Q(i,j) matrix
C       F(i,j)          :==     Jacobian of nonlinear state eqn evaluated at
C                               X(i)
C       H(i,j)          :==     Jacobian of nonlinear measured eqn evaluated
C                               at XPRED(i)
C       PHI(i,j)        :==     State transition matrix
C       G(i,j)          :==     Noise transition matrix
C       Q(i,j)          :==     State noise covariance matrix
C       QMAT(i,j)       :==     Matrix used to average Q matrix
C       R(i,j)          :==     Measured noise covarance matrix
C       ERROR(i)        :==     Assumed relative error in Y vector. When
C                               when multiplied by Y vector gives standard
C                               deviation in Y vector. Used for determining
C                               value of R matrix.
C       RES(i)          :==     Residuals between measurement and predicted
C                               measurement
C       K(i,j)          :==     Kalman gain matrix
C       PARAM(i)        :==     Vector which contains parameters used in
C                               state eqns
C       CONV(i)         :==     Vector which contains convolutes used in
C                               smoothing the Y vector
C
C The following information must be entered in the file assigned to FORØØ5
C in the following order and on each line:
C
C     Rel error in Y vector:                    Y(1)-Y(N)
C     Maximum rel error in Y vector:            YMAX
C     Number of points to averavge Q over:      NQPT
C
C Also the files IXiTOj.DAT AND IPiTOj.DAT must be renamed to XiTOj.DAT
C and PiTOj.DAT
C
        IMPLICIT REAL*8 (A-H,O-Z)
        EXTERNAL STEQNS,JACOB
        REAL*8    K(2Ø,2Ø)
        DIMENSION X(2Ø),XPRED(2Ø),P(2Ø,2Ø),PPRED(2Ø,2Ø),H(2Ø,2Ø)
        DIMENSION F(2Ø,2Ø),Y(2Ø),YHAT(2Ø),TEMP1(2Ø,2Ø),C(8ØØ),RES(2Ø)
        DIMENSION Q(2Ø,2Ø),PARAM(1Ø),FPFT(2Ø,2Ø),R(2Ø,2Ø),QMAT(2Ø,2Ø)
        DIMENSION PHI(2Ø,2Ø),G(2Ø,2Ø),IWK(2Ø),YMAT(2Ø,2Ø),CONV(2Ø)
        COMMON PARAM
C
C STEP 1.
C Read state and measured variable dimensions and intialize system
C by calling SYSTEM subroutine
```

```
C
        MS=2Ø
        TINT=Ø.Ø
        ICOUNT=Ø
        IQCNT=Ø
        CALL SYSTEM (MS,M,N,NPARAM,X,P)
C
C   Open files to be used for storage and input data.
C
        OPEN (UNIT=1Ø,FILE='PARAMETER',STATUS='OLD')
        OPEN (UNIT=7,FILE='Y1TO1Ø',STATUS='OLD')
        OPEN (UNIT=1,FILE='X1TO1Ø',STATUS='OLD')
        OPEN (UNIT=9,FILE='P1TO1Ø',STATUS='OLD')
        OPEN (UNIT=11,FILE='CONV',STATUS='OLD')
        OPEN (UNIT=13,FILE='IP1TO1Ø',STATUS='NEW')
        OPEN (UNIT=15,FILE='IX1TO1Ø',STATUS='NEW')
        IF (M .GT. 1Ø) THEN
           OPEN (UNIT=3,FILE='X11TO2Ø',STATUS='OLD')
           OPEN (UNIT=4,FILE='P11TO2Ø',STATUS='OLD')
           OPEN (UNIT=14,FILE='IP11TO2Ø',STATUS='NEW')
           OPEN (UNIT=16,FILE='IX11TO2Ø',STATUS='NEW')
        END IF
        IF (N .GT. 1Ø) THEN
           OPEN (UNIT=8,FILE='Y11TO2Ø',STATUS='OLD')
        END IF
C
C   Input maximum relative error for Y vector.
        READ (5,*) YMAX
C   Input number of points to average Q over.
        READ (5,*) NQPT
C
C Convolute points are stored in file CONV.DAT with the first data NPT
C the second DNORM  and the remainder CONV(NPT)
C
        READ(11,*) NPT,DNORM,(CONV(I),I=1,NPT)
C
C Output inputted data to FORØØ6 for conformation.
C
        WRITE(6,1) YMAX,NQPT
 1      FORMAT(' YMAX = ',F7.2,/,' Q MATRIX AVE. OVER:',I3,' PTS')
        WRITE(6,2) NPT,DNORM
 2      FORMAT(' NO. OF CONVOLUTE PIONTS = ',I2
     + ,/,' NORM = ',F9.Ø)
        DO 4 I=1,NPT
           WRITE(6,3) I,-I+1,CONV(I)
 3         FORMAT(' C(',I2,') * Y(',I3,') = ',F8.Ø)
 4      CONTINUE
C
C Create starting files for next run at same initial conditions
C
        IF (M .LE. 1Ø) THEN
           WRITE(15,7) (X(I),I=1,M)
           WRITE(13,7) (P(I,I),I=1,M)
           CLOSE(UNIT=13)
           CLOSE(UNIT=15)
        ELSE
           WRITE(15,7) (X(I),I=1,1Ø)
           WRITE(16,7) (X(I),I=11,M)
           WRITE(13,7) (P(I,I),I=1,1Ø)
           WRITE(14,7) (P(I,I),I=11,M)
           CLOSE(UNIT=13)
           CLOSE(UNIT=14)
           CLOSE(UNIT=15)
           CLOSE(UNIT=16)
        END IF
 7      FORMAT(1X,1Ø(1X,E11.4))
```

```
C
C STEP 2.
C Store data.
C
 5       IF (M .LE. 1Ø) THEN
            WRITE(1,1Ø) TINT,(X(I),I=1,M)
            WRITE(9,1Ø) TINT,(P(I,I),I=1,M)
         ELSE
            WRITE(1,1Ø) TINT,(X(I),I=1,1Ø)
            WRITE(3,1Ø) TINT,(X(I),I=11,M)
            WRITE(9,1Ø) TINT,(P(I,I),I=1,1Ø)
            WRITE(4,1Ø) TINT,(P(I,I),I=11,M)
         END IF
 1Ø      FORMAT (1X,F7.4,1Ø(1X,E11.4))
C
C STEP 3.
C Read measured (output) vector, parameters and time.
C
         IF (N .LE. 1Ø) THEN
            READ(7,*,END=16Ø) TEND,(Y(I),I=1,N)
         ELSE
            READ(7,*,END=16Ø) TEND,(Y(I),I=1,1Ø)
            READ(8,*) TEND,(Y(I),I=11,N)
         END IF
C
C Now smooth data.
C Use polynomial least squares fit of NPT points (see text)
C
         ICOUNT=ICOUNT+1
         IFLAG=Ø
         IF(ICOUNT .LT. NPT) IFLAG=1
         DO 12 I=1,NPT
            DO 12 J=1,N
               YMAT(NPT+2-I,J)=YMAT(NPT+1-I,J)
 12      CONTINUE
         DO 15 I=1,N
            YMAT(1,I)=Y(I)
 15      CONTINUE
         IF(IFLAG .EQ. 1) GOTO 17
         DO 17 I=1,N
            Y(I)=Ø.Ø
            DO 17 J=1,NPT
               Y(I)=Y(I)+(CONV(J)*YMAT(J,I)/DNORM)
 17      CONTINUE
C Read parameters for uses by state eqns
         READ(1Ø,*) (PARAM(I),I=1,NPARAM)
C
C STEP 4.
C Calculate X(t+1|t) using DGEAR from IMSL library.
C DGEAR subroutine can handle stiff eqns
C
         METH=2
         TOL=Ø.ØØØØ1
         INDEX=1
         STEP=Ø.ØØØØ1
         MITER=1
         DO 2Ø I=1,M
            XPRED(I)=X(I)
 2Ø      CONTINUE
         T=TINT
         CALL DGEAR (M,STEQNS,JACOB,T,STEP,XPRED,TEND,TOL,METH,MITER,
     *   INDEX,IWK,C,IER)
         IF (IER .GT. 128) THEN
            WRITE(6,25) IER,STEP,TOL,INDEX,T
 25         FORMAT(' FAILURE IN DGEAR AT STEP 4. IER=',I3,/,' STEP='
     *      ,F8.6,/,' TOL=',F8.6,/,' INDEX=',I2,/,' TIME=',F8.6)
```

```
            STOP
         END IF
         IF (IER .GT. Ø) THEN
            WRITE(6,3Ø) IER,STEP,T
 3Ø         FORMAT(' **WARNING IN DGEAR AT STEP 4.**   IER=',I3,
     *      /,' STEP=',F8.6,/,' TIME=',F8.5)
         END IF
C
C STEP 5.
C Obtain values for H,G,R,F,YHAT.
C
         CALL SYSMAT (MS,X,XPRED,Y,H,G,R,YHAT,TINT,TEND)
         CALL JACOB (MS,TINT,X,F)
C
C STEP 6.
C Calculate residuals
C
         DO 4Ø I=1,N
            RES(I)=Y(I)-YHAT(I)
 4Ø      CONTINUE
C
C STEP 7.
C Calculate state transition matrix PHI
C
         DO 5Ø I=1,M
            DO 5Ø J=1,M
               F(I,J)=F(I,J)*(TEND-TINT)
 5Ø      CONTINUE
         B=2.Ø
         MACHDIG=55
         CALL SUBEXP (F,MS,M,PHI,B,MACHDIG,SIGDIG)
         IF (SIGDIG .LE. 5.Ø) THEN
            WRITE(6,6Ø) SIGDIG,TEND
 6Ø         FORMAT(' **WARNING IN STEP 7.**   NO. OF SIG DIG IN PHI='
     *      ,F2.Ø,/,' AT TIME=',F7.4)
         END IF
C
C STEP 8.
C Calculate state-noise covariance matrix by Adaptive Filter.
C Note the matrices TEMP1,K and PHI are used as work space in this step.
C
         CALL VMULFF (PHI,P,M,M,M,MS,MS,TEMP1,MS,IR)
         CALL VMULFP (TEMP1,PHI,M,M,M,MS,MS,FPFT,MS,IR)
         CALL VMULFF (H,FPFT,N,M,M,MS,MS,TEMP1,MS,IR)
         CALL VMULFP (TEMP1,H,N,M,N,MS,MS,PHI,MS,IR)
         DO 7Ø I=1,N
            Y(I)=RES(I)
            IF(DABS(RES(I)) .GT. DABS(YMAX*YHAT(I)))
     *      Y(I)=YMAX*YHAT(I)
            TEMP1(I,I)=Y(I)**2 - R(I,I) - PHI(I,I)
 7Ø      CONTINUE
         CALL VMULFF (H,G,N,M,N,MS,MS,PHI,MS,IR)
         IDGT=Ø
         CALL LINV2F (PHI,N,MS,K,IDGT,C,IER)
         IF (IER .GT. Ø) THEN
            WRITE(6,8Ø) IER
 8Ø         FORMAT(1X,'INVERSE FAILURE IN ADAPTIVE FILTER STEP 8. IER='
     *   ,I3)
            STOP
         END IF
         CALL VMULFF (K,TEMP1,N,N,N,MS,MS,PHI,MS,IR)
         CALL VMULFP (PHI,K,N,N,N,MS,MS,TEMP1,MS,IR)
C
C STEP 9.
C Set Q and make sure it's postive and average Q over NQPT points.
C
```

```
        IQCNT=IQCNT+1
        IFLAG2=Ø
        IF (IQCNT .LT. NQPT) IFLAG2=1
        DO 81 I=1,NQPT
           DO 81 J=1,N
              QMAT(NQPT+2-I,J)=QMAT(NQPT+1-I,J)
 81     CONTINUE
        DO 82 I=1,N
           QMAT(1,I)=TEMP1(I,I)
 82     CONTINUE
        NAVE=NQPT
        IF (IFLAG2 .EQ. 1) NAVE=IQCNT
        DO 83 I=1,N
           TEMP1(I,I)=Ø.Ø
           DO 83 J=1,NAVE
              TEMP1(I,I)=TEMP1(I,I)+QMAT(J,I)/NQPT
 83     CONTINUE
        DO 85 I=1,N
           DO 85 J=1,N
              Q(I,J)=Ø.Ø
 85     CONTINUE
        DO 9Ø I=1,N
           Q(I,I)=TEMP1(I,I)
           IF (Q(I,I) .LT. Ø.Ø) Q(I,I)=Ø.Ø
 9Ø     CONTINUE
C
C STEP 1Ø.
C Compute state-error covariance matrix.
C
        CALL VMULFF (G,Q,M,N,N,MS,MS,TEMP1,MS,IR)
        CALL VMULFP (TEMP1,G,M,N,M,MS,MS,Q,MS,IR)
        DO 11Ø I=1,M
           DO 11Ø J=1,M
              PPRED(I,J)=FPFT(I,J)+Q(I,J)
 11Ø    CONTINUE
C
C STEP 11.
C  Calculate Kalman Gain.
C  Matrices TEMP1,FPFT, and PHI are work space in this step.
C
        CALL VMULFF (H,PPRED,N,M,M,MS,MS,TEMP1,MS,IR)
        CALL VMULFP (TEMP1,H,N,M,N,MS,MS,PHI,MS,IR)
        DO 12Ø I=1,N
           DO 12Ø J=1,N
              TEMP1(I,J)=PHI(I,J) + R(I,J)
 12Ø    CONTINUE
        IDGT=1Ø
        CALL LINV2F (TEMP1,N,MS,PHI,IDGT,C,IER)
        IF (IER .GT. 34) THEN
           WRITE (6,13Ø) IER
 13Ø       FORMAT (1X,'INVERSE FAILURE IN KALMAN GAIN STEP 11. IER=',I3)
           STOP
        END IF
        IF (IER .GT. Ø) THEN
           WRITE(6,135) IER,TINT,IDGT
 135       FORMAT(' ** INVERSE WARNING IN KALMAN GAIN STEP 11. **',/,
     *     ' IER= ',I2,/,' TIME = ',F8.4,/,' IDGT = ',I2)
        END IF
        CALL VMULFM (H,PHI,N,M,N,MS,MS,TEMP1,MS,IR)
        CALL VMULFF (PPRED,TEMP1,M,M,N,MS,MS,K,MS,IR)
C
C STEP 12.
C  Update state vector.
C  Y is used as work space.
C
        CALL VECMAT (K,MS,M,N,RES,Y)
        DO 14Ø I=1,M
           X(I)=XPRED(I)+Y(I)
 14Ø    CONTINUE
C
C STEP 13.
C Update error covariance matrix
C PHI is used as work space.
C
        CALL VMULFF (K,H,M,N,M,MS,MS,PHI,MS,IR)
        CALL VMULFF (PHI,PPRED,M,M,M,MS,MS,TEMP1,MS,IR)
        DO 15Ø I=1,M
           DO 15Ø J=1,M
              P(I,J)=PPRED(I,J)-TEMP1(I,J)
 15Ø    CONTINUE
        TINT=TEND
        GOTO 5
C
C Now exiting program.
C
 16Ø    WRITE (6,17Ø) TEND
 17Ø    FORMAT (1X,'END OF FILE HIT AT TIME =',F7.3,'hrs.')
        CLOSE (UNIT=1)
        CLOSE (UNIT=9)
        CLOSE (UNIT=7)
        CLOSE (UNIT=1Ø)
        CLOSE (UNIT=11)
        IF (M .GT. 1Ø) THEN
           CLOSE (UNIT=3)
           CLOSE (UNIT=4)
        END IF
        IF (M .GT. 1Ø) CLOSE(UNIT=8)
        STOP
        END




        SUBROUTINE VECMAT (A,IA,M,N,X,Y)
C This routine compute Ax and places the result in vector y.  A is a
C M by N matrix.
        REAL*8 A(IA,1),X(1),Y(1)
        DO 2Ø I=1,M
           Y(I)=Ø.Ø
           DO 1Ø J=1,N
              Y(I)=Y(I)+A(I,J)*X(J)
 1Ø        CONTINUE
 2Ø     CONTINUE
        RETURN
        END


C This subroutine is used to specialize the program to hand a specified
C set of state and measured eqns.  Changes need only be made to this subroutine
C to handle a different set of state and measured eqns.  No changes need be
C made to the main program.
C This subroutine has several entry point which are explained below.
C
C SYSTEM ENTRY:
C This section is used to inialize the main program.  The dimensions of the
C state and measured vectors are set here along with the no. of parameters
C used in the state eqns.
C
C STEQNS ENTRY:
```

```
C This section contains the state differential eqns that are specific to
C a particular user
C
C JACOB ENTRY:
C This section contains the jacobian of the state eqns. It is used by the
C filter routine as well as DGEAR.
C
C SYSMAT ENTRY:
C This section set all the matrices that are required by the main program


        SUBROUTINE SYSTEM (MS,M,N,NPARAM,X,P)
        IMPLICIT REAL*8 (A-H,O-Z)
        REAL*8 KLA
        DIMENSION X(1),XPRED(1),P(MS,MS),H(MS,MS),G(MS,MS),R(MS,MS)
        DIMENSION F(MS,MS),Y(1),YHAT(1),XDOT(1),PARAM(1Ø),ERROR(2Ø)
        COMMON PARAM
C
C System intializes X and P and sets the state space dimension to M
C and the output space to N.
C
        M=9
        N=5
C
C  Read from FORØØ5 relative errors assumed for Y vector.
        READ(5,*) (ERROR(I),I=1,N)
        DO 5 I=1,M
           DO 5 J=1,M
              P(I,J)=Ø.Ø
 5      CONTINUE
        NPARAM=3
        NX=M
        NY=N
        OPEN (UNIT=1,FILE='X1TO1Ø',STATUS='OLD')
        OPEN (UNIT=9,FILE='P1TO1Ø',STATUS='OLD')
        READ(1,*) (X(I),I=1,M)
        READ(9,*) (P(I,I),I=1,M)
        CLOSE(UNIT=1)
        CLOSE(UNIT=9)
C
C Output relative errors in Y vector to FORØØ6 for conformation.
        DO 7 I=1,NY
           WRITE(6,6) I,ERROR(I)
 6         FORMAT(' REL  IN Y(',I2,') = ',F6.3)
 7      CONTINUE
        RETURN
C
C This part of the program is used by DGEAR.
C
        ENTRY STEQNS (M,TIME,X,XDOT)
        D=PARAM(1)
        SF=PARAM(2)
        O2MAX=PARAM(3)
C
        B=X(1)
        S=X(2)
        U=X(3)
        C1=X(4)
        YS=X(5)
        O2D=X(6)
        KLA=X(7)
        C2=X(8)
        YO2=X(9)
C
C These are the governing differintial eqns for this system.
C
```

```
          XDOT(1)=B*(U-D)
          XDOT(2)=D*(SF-S)-U*B/YS
          XDOT(3)=C1
          XDOT(4)=Ø.Ø
          XDOT(5)=Ø.Ø
          XDOT(6)=KLA*(O2MAX-O2D)-U*B/YO2
          XDOT(7)=C2
          XDOT(8)=Ø.Ø
          XDOT(9)=Ø.Ø
C
          RETURN
C
C This routine calculates the Jacobian for DGEAR and SYSMAT.
C
          ENTRY JACOB (MS,T,X,F)
          DO 1Ø I=1,NX
             DO 1Ø J=1,NX
                F(I,J)=Ø.Ø
 1Ø       CONTINUE
          D=PARAM(1)
          SF=PARAM(2)
          O2MAX=PARAM(3)
C
          B=X(1)
          S=X(2)
          U=X(3)
          C1=X(4)
          YS=X(5)
          O2D=X(6)
          KLA=X(7)
          C2=X(8)
          YO2=X(9)
C
C Set F matrix. F is the Jacobian of the state eqns xdot=f(x,t).
C
          F(1,1)=X(3)-D
          F(1,3)=X(1)
          F(2,1)=-X(3)/X(5)
          F(2,2)=-D
          F(2,3)=-X(1)/X(5)
          F(2,5)=X(1)*X(3)/X(5)**2
          F(3,4)=1.Ø
          F(4,4)=Ø.Ø
          F(6,1)=-X(3)/X(9)
          F(6,3)=-X(1)/X(9)
          F(6,6)=-X(7)
          F(6,7)=O2MAX-X(6)
          F(6,9)=X(1)*X(3)/X(9)**2
          F(7,8)=1.Ø
          F(8,8)=Ø.Ø
C
          RETURN
C
C This part of the program sets the state and output eqns matrices and vectors
C
          ENTRY SYSMAT (MS,X,XPRED,Y,H,G,R,YHAT,TINT,TEND)
          DO 15 I=1,NX
             DO 15 J=1,NX
                R(I,J)=Ø.Ø
                G(I,J)=Ø.Ø
                H(I,J)=Ø.Ø
 15       CONTINUE
C
C Set H matrix
C
          H(1,1)=XPRED(3)
```

1

```
      H(1,3)=XPRED(1)
      H(2,5)=1.Ø
      H(3,9)=1.Ø
      H(4,6)=1.Ø
      H(5,6)=-XPRED(7)
      H(5,7)=O2MAX-XPRED(6)
C
C Set G matrix.
C
      G(3,1)=1/SF
      G(4,1)=D/SF
      G(5,2)=1.Ø
      G(6,4)=1.Ø
      G(7,5)=1/O2MAX
      G(8,5)=D/O2MAX
      G(9,3)=1.Ø
C
C Next is the nonlinear output (measurement) eqns.
C
      YHAT(1)=XPRED(1)*XPRED(3)
      YHAT(2)=XPRED(5)
      YHAT(3)=XPRED(9)
      YHAT(4)=XPRED(6)
      YHAT(5)=XPRED(7)*(O2MAX-XPRED(6))
C
C Set output noise covariance matrix R
C
      DO 2Ø I=1,NY
         R(I,I)=(ERROR(I)*YHAT(I))**2
 2Ø   CONTINUE
C
      RETURN
      END
```

10.3 Matrix Exponential Program

```
C  SUBEXP computes the exponential of matrix A and places the result in
C  matrix EXPA. B denotes the radix base of the computer (2 for vax) and
C  MACHDIG denotes the number of digits (base B) used to reprsent the
C  mantissa of a floating point number in the computer (MACHDIG=55 for
C  REAL*8 representation on the vax).  SIGDIG gives the number of
C  significant digits in the norm of exp(a).  Note matrices A and EXPA
C  must occupi different locations in memory.
C  This program also call the subroutine BALANCE to balance A with respect
C  to the 1-norm, and subroutine ANORM to compute the 1-norm of A.
C
C  Definitions of important variables:
C
C       IA       :==     Row dimension of matrix A as it appears in the
C                        dimension statement of the calling program.
C       N        :==     Used dimensions of matrix A
C       P        :==     Order of Pade approximation used.
C       AVLAM    :==     Average of eigenvalues of matrix A

C  See R. Ward, SIAM J. NUMER. ANAL. Vol. 14, NO. 4, Sept 1977.
C
       SUBROUTINE SUBEXP (A,IA,N,EXPA,B,MACHDIG,SIGDIG)
       IMPLICIT REAL*8 (A-H,O-Z)
       DIMENSION A(IA,1),EXPA(IA,1),D(2Ø),EXPAK(2Ø,2Ø),QPA(2Ø,2Ø),AK(2Ø,2Ø)
       DIMENSION QPNA(2Ø,2Ø),WK(5ØØ)
C
C      TO MAKE THIS SUBROUTINE HANDLE MATRICES LARGER THAN 2Ø BY 2Ø
C      CHANGE IAK=# AND ALL MATRICES THAT ARE 2Ø BY 2Ø TO # BY #.
C      CHANGE WK MATRIX TO (#**2)+3#.
C
       INTEGER HI,P,FLAG
       IAK=2Ø
       FLAG=Ø
       DO 1Ø I=1,N
          DO 1Ø J=1,N
             EXPA(I,J)=A(I,J)
 1Ø    CONTINUE
C
C  STEP 1. COMPUTE AVERAGE OF EIGENVALUES
C
       AVLAM=Ø.DØØ
       DO 15 I=1,N
          AVLAM=AVLAM+EXPA(I,I)/N
 15    CONTINUE
C
C  STEP 2. SUBTRACT AVLAM FROM DIAGANAL ELEMENTS OF EXPA
C
       DO 2Ø I=1,N
          EXPA(I,I)=EXPA(I,I)-AVLAM
 2Ø    CONTINUE
C
C  STEP 3. NOW BALANCE EXPA MATRIX
C
       DO 25 I=1,N
          DO 25 J=1,N
             EXPAK(I,J)=EXPA(I,J)
 25    CONTINUE
       CALL BALANCE (EXPAK,IAK,N,EXPA,IA,D,HI,LOW,B)
C
C  STEP 4. CHECH NORM OF MATRIX EXPA
C
       ENORM=ANORM(EXPA,IA,N)
       IF (ENORM .LE. 1.DØØ) THEN
          FLAG=1
          GOTO 5Ø
       END IF
```

```
C
C  STEP 5. DETERMINE VALUE OF M SUCH THAT 2**(-M)|EXPA|<1.
C
       M=Ø
 3Ø    M=M+1
       IF ((2.DØØ**(-M))*ENORM .GT. 1.DØØ) GOTO 3Ø
C
C  STEP 6. REDUCE NORM OF EXPA IF ENORM>1
C
       DO 4Ø I=1,N
          DO 4Ø J=1,N
             EXPA(I,J)=EXPA(I,J)*(2.DØØ**(-M))
 4Ø    CONTINUE
C
C  STEP 7. COMPUTE PADE APPROXIMATION TO THE EXPONENTIAL OF EXPA
C
 5Ø    P=8
C      NOTE THAT P MUST BE EQUAL TO OR GREATER THAN 2
       CK=Ø.5DØØ
       DO 8Ø I=1,N
          DO 7Ø J=1,N
             QPA(I,J)=CK*EXPA(I,J)
             QPNA(I,J)=-CK*EXPA(I,J)
             EXPAK(I,J)=EXPA(I,J)
 7Ø       CONTINUE
          QPA(I,I)=QPA(I,I)+1.DØØ
          QPNA(I,I)=QPNA(I,I)+1.DØØ
 8Ø    CONTINUE
       DO 1ØØ K=2,P
          CK=CK*(P-K+1)/(K*(2*P-K+1))
          CALL VMULFF (EXPA,EXPAK,N,N,N,IA,IAK,AK,IAK,IERR)
          DO 9Ø I=1,N
             DO 9Ø J=1,N
                EXPAK(I,J)=AK(I,J)
                QPA(I,J)=QPA(I,J)+CK*AK(I,J)
                QPNA(I,J)=QPNA(I,J)+((-1)**K)*CK*AK(I,J)
 9Ø       CONTINUE
 1ØØ   CONTINUE
C
C  STEP 8. COMPUTE EXP(B)
C
       IDGT=15
       CALL LINV2F (QPNA,N,IAK,AK,IDGT,WK,IER)
       CALL VMULFF (AK,QPA,N,N,N,IAK,IAK,EXPA,IA,IERR)
       IF ((IER .EQ. 129) .OR. (IER .EQ. 131) .OR. (IER .EQ. 34)) THEN
          WRITE(6,11Ø) IER
 11Ø      FORMAT(1X,'MATRIX INVERSION FAILED, IER=',I3)
          STOP
       END IF
       EXPBN=ANORM(EXPA,IA,N)
       G=(84*N+73)*B**(-MACHDIG)*EXPBN
C
C  STEP 9. CHECK FLAG. IF FLAG=1 THEN GOTO STEP 11.
C
       IF (FLAG .EQ. 1) GOTO 13Ø
C
C  STEP 1Ø SQUARE EXP(B) M TIMES
C
       DO 13Ø K=1,M
          CALL VMULFF (EXPA,EXPA,N,N,N,IA,IA,AK,IAK,IERR)
          DO 12Ø I=1,N
             DO 12Ø J=1,N
                EXPA(I,J)=AK(I,J)
 12Ø      CONTINUE
          G=2.DØØ*EXPBN*G+G**2+N*(B**(-MACHDIG))*(EXPBN**2)
          EXPBN=ANORM(EXPA,IA,N)
 13Ø    CONTINUE
C
C  STEP 11. COMPUTE EXP(A)
C
C       FIRST MULTIPLY BY    D EXP(B) DINV  IF MATRIX WAS BALANCED
C
        IF (HI .EQ. 1) GOTO 155
        DO 15Ø K=LOW,HI
           DO 14Ø I=1,N
              EXPA(K,I)=EXPA(K,I)*D(K)
              EXPA(I,K)=EXPA(I,K)/D(K)
 14Ø       CONTINUE
 15Ø    CONTINUE
C
C       APPLY PERMUTATION MATRIX
C
        I=LOW-1
        DO WHILE (I .GE. 1)
           J1=JIDINT(D(I))
           CALL EXCHANGE (EXPA,IA,N,I,J1)
           I=I-1
        END DO
        I=HI+1
        DO WHILE (I .LE. N)
           J1=JIDINT(D(I))
           CALL EXCHANGE (EXPA,IA,N,I,J1)
           I=I+1
        END DO
 155    AVLAM=DEXP(AVLAM)
        DO 16Ø I=1,N
           DO 16Ø J=1,N
              EXPA(I,J)=AVLAM*EXPA(I,J)
 16Ø    CONTINUE
C
C  STEP 12. COMPUTE NO. OF SIGNIFICANT FIGURES IN ANSWER.
C
        IF (HI .EQ. 1) THEN
           DNORM=1.DØØ
           DINVN=1.DØØ
           GOTO 17Ø
        END IF
        DMAX=1.DØØ
        IF ((LOW .EQ. 1) .AND. (HI .EQ. N)) DMAX=Ø.DØØ
        DNORM=DMAX
        DINVN=DMAX
        DO 17Ø I=LOW,HI
           IF (D(I) .GT. DNORM) DNORM=D(I)
           IF (1.DØØ/D(I) .GT. DINVN) DINVN=1.DØØ/D(I)
 17Ø    CONTINUE
        SIGDIG=-DLOG1Ø(AVLAM*DNORM*G*DINVN/ANORM(EXPA,IA,N))
        RETURN
        END




C   ANORM computes the 1 norm of matrix A.
C              IA       :==     Row dimension of matrix A as in dimension
C                               statement of calling program.
C              N        :==     Used dimensions of matrix A
C
        REAL FUNCTION ANORM*8 (A,IA,N)
        REAL*8 A(IA,1),COLA
        ANORM=Ø.DØØ
        DO 2Ø J=1,N
```

```
          COLA=Ø.DØØ
          DO 1Ø I=1,N
             COLA=COLA+DABS(A(I,J))
 1Ø       CONTINUE
          IF (COLA .GT. ANORM) ANORM=COLA
 2Ø     CONTINUE
        RETURN
        END


C  This program balances matix A and places the result in matrix BAL.
C  The diagonal elements of D are stored in the D matrix from LOW to
C  HI.  The information for the permutation matrix is stored in the
C  D matrix from 1-(LOW-1) and (HI+1)-N. B is the radix base of the
C  computer (B=2 for vax)
C  This program call subroutine EXCHANGE
C  See B. N. Parlett, NUMER. MATH. 13, 293-3Ø4 (1969).
C
C  WHEN CALLING "BALANCE", MARICES A AND BAL MUST OCCUPI DIFFERENT LOCATIONS
C  IN MEMORY.  IF ON RETURN, HI=1 THEN NO MANIPULATION WAS DONE
C  ON EITHER MATRIX.
C
        SUBROUTINE BALANCE(A,IA,N,BAL,IB,D,HI,LOW,B)
        IMPLICIT REAL*8 (A-H,O-Z)
        DIMENSION A(IA,IA),BAL(IB,IB),D(1)
        INTEGER LOW,HI,FLAG
        DO 1Ø I=1,N
          DO 1Ø J=1,N
            BAL(I,J)=A(I,J)
 1Ø     CONTINUE
C SEARCH FOR ROWS ISOLATING AN EIGENVALUE AND PUSH THEM DOWN
        B2=B*B
        K=N
        L=1
 12     J=K
        DO  WHILE (J .GE. 1)
           R=Ø.DØØ
           DO 15 I=1,K
              IF (I .EQ. J) GOTO 15
              R=R+DABS(BAL(J,I))
 15        CONTINUE
           IF (R .EQ. Ø.DØØ) THEN
              D(K)=J
              CALL EXCHANGE (BAL,IB,N,K,J)
              K=K-1
              GOTO 12
           END IF
           J=J-1
        END DO
C
C       CHECK TO SEE IF MATRIX IS BALANCEABLE  IF NOT RESET MATRIX & RETURN
C
        IF (K .EQ. Ø) THEN
           HI=1
           LOW=1
           DO 16 I=1,N
              DO 16 J=1,N
                 BAL(I,J)=A(I,J)
 16        CONTINUE
           RETURN
        END IF
C SEARCH FOR COLUMNS ISOLATING AN EIGENVALUE AND PUSH THEM LEFT
 17     DO 2Ø J=L,K
           C=Ø.DØØ
           DO 25 I=L,K
```

```
               IF (I .EQ. J) GOTO 25
               C=C+DABS(BAL(I,J))
 25         CONTINUE
            IF (C .EQ. Ø.DØØ) THEN
               D(L)=J
               CALL EXCHANGE (BAL,IB,N,L,J)
               L=L+1
               GOTO 17
            END IF
 2Ø      CONTINUE
C
C        NOW DETERMINE ELEMENTS FOR DIAGONAL MATRIX D
C
         LOW=L
         HI=K
         DO 26 I=L,K
            D(I)=1.DØØ
 26      CONTINUE
 27      FLAG=Ø
         DO 5Ø I=L,K
            C=Ø.DØØ
            R=Ø.DØØ
            DO 3Ø J=L,K
               IF (J .EQ. I) GOTO 3Ø
               C=C+DABS(BAL(J,I))
               R=R+DABS(BAL(I,J))
 3Ø         CONTINUE
            G=R/B
            F=1.DØØ
            S=C+R
 35         IF (C .LT. G) THEN
               F=F*B
               C=C*B2
               GOTO 35
            END IF
            G=R*B
 4Ø         IF (C .GE. G) THEN
               F=F/B
               C=C/B2
               GOTO 4Ø
            END IF
            IF (((C+R)/F) .LT. (Ø.95DØØ*S)) THEN
               G=1/F
               D(I)=D(I)*F
               FLAG=1
               DO 45 J=L,N
 45               BAL(I,J)=BAL(I,J)*G
               DO 47 J=1,K
 47               BAL(J,I)=BAL(J,I)*F
            END IF
 5Ø      CONTINUE
         IF (FLAG .EQ. 1) GOTO 27
         RETURN
         END


C   EXCHANGE exchanges row and column M with row and column J of matrix A.
C   IA is the dimension of A in the dimension statement of the calling prog.
C
         SUBROUTINE EXCHANGE (A,IA,N,M,J)
         REAL*8 A(IA,IA),F
         IF (J .NE. M) THEN
            DO 1Ø I=1,N
               F=A(I,J)
```

```
                 A(I,J)=A(I,M)
                 A(I,M)=F
 10            CONTINUE
             DO 15 I=1,N
                 F=A(J,I)
                 A(J,I)=A(M,I)
                 A(M,I)=F
15           CONTINUE
          END IF
          RETURN
          END
```