Application of Dynamic Optimization-Based Parameter Estimation to a Diabetes Mellitus Patient Model

A. A. Rivera-Montalvo, A. T. Stamps and E. P. Gatzke

Abstract— **This work presents a method for modeling the effect of insulin infusion and meal disturbances in a patient with diabetes mellitus. The model used was presented by Bergman and later modified by Lynch and Bequette. This modified Bergman model is a simple set of four differential equations which account for blood glucose, blood insulin, free insulin and subcutaneous glucose concentration profiles in the body of a person with diabetes mellitus. Using subcutaneous blood sugar measurements and data for the response to insulin of a patient, it is possible to develop an accurate dynamic model, leading to the possibility of applying model-predictive control to help regulate the blood glucose levels of a patient. The initial parameters for the modified Bergman model were recovered from simulated data by performing dynamic optimizations using MAPLE, MATLAB® and IPOPT. In the future, this procedure could be tested using real data for a patient.**

I. INTRODUCTION

Diabetes mellitus is a metabolic disease characterized by irregular glucose metabolism. Patients with diabetes mellitus usually develop serious long-term effects, which increase their risk of developing serious cardiovascular, renal, and neural complications [1], [2]. This work focuses on a diabetes model presented by Bergman [3], [4], [5], in which the dynamic concentration profiles of glucose and insulin are modeled.

The dynamic model considered here includes four states: $G(t)$, $X(t)$, $I(t)$, and $G_{SC}(t)$. Equation 1 describes the change in glucose concentration, $\frac{dG}{dt}$ [mg/dL/min], with respect to the current glucose concentration, $G(t)$ [mg/dL], the plasma insulin concentration, $X(t)$ [mU/L], the basal glucose value for a healthy human subject, G_b [mg/dL], and the rate of exogenous glucose (the glucose intake from meals and other sources) $D(t)$ [mg/L/min].

$$\frac{dG}{dt} = -P_1 G - X\Big(G + G_b\Big) + D(t) \tag{1}$$

Equation 2 in the model describes the change in the plasma insulin concentration, $\frac{dX}{dt}$ [mU/L/min], with respect to the current plasma insulin concentration $X(t)$ and the free plasma insulin concentration above basal value, $I(t)$ [mU/L].

Corresponding author: gatzke@sc.edu, (803) 777-1159

$$\frac{dX}{dt} = -P_2 X + P_3 I \tag{2}$$

Equation 3 in the model describes the change in the free plasma insulin concentration above basal value, $\frac{dI}{dt}$ [mU/L/min], with respect to the free plasma insulin concentration above basal value $I(t)$ [mU/L], the basal value of free plasma insulin, I_b (in [mU/L], the typical free insulin level for controlled diabetic patients), the insulin distribution volume V_I [L], the fractional insulin dissappearance rate, n, and the rate of exogenous (administered) insulin, $U(t)$ [mU/min]. Fisher et al [6] modified this equation of the Bergman model, adding an insulin infusion term and omitting an insulin secretion term:

$$\frac{dI}{dt} = -n(I + I_b) + \frac{U(t)}{V_I} \tag{3}$$

Lynch and Bequette [7] showed an additional variation to the Bergman model, using changes in subcutaneous glucose measurements $\frac{dG_{SC}}{dt}$ [mg/dL/min] and relating them to the subcutaneous glucose measurements $G_{sc}(t)$ [mg/dL], the rate of tissue utilization R_{ut} [mg/dL/min], and the blood glucose concentration $G(t)$. This model also assumes a first order lag θ of 5 min.

$$\frac{dG_{SC}}{dt} = \frac{(G - G_{sc})}{\theta} - R_{ut} \tag{4}$$

All constants in this Bergman model are known, but parameters P_1, P_2, P_3, V_I and n might vary in every patient. Using dynamic patient data, it is possible to extract the values of these parameters using parameter estimation. Parameter estimation begins with a set of initial parameter values. The model is evaluated at these values, then the error between the model values and the actual data is calculated. After this step, other parameter values near the original are evaluated. If the calculated error is less than the error of the last parameter set evaluated, that new parameter set becomes the next point, essentially moving through the iterative process toward a group of parameters that results in a minimum of error. When optimizing, it is important to select the initial parameter values carefully. When working with an oscillatory or otherwise non-linear system, it might be possible to find a point where the error is a minima for that region (*e.g.* a local minima), but not a global minima as desired. In other cases, the estimation routine may diverge completely.

The modified Berman model was used to generate simulated data. Next, the simulated data was used with the estimation routine to find the original parameter values specified. Using dynamic optimization methods, one may use data to extract the parameters in the model that best fit the data, giving some insight into how the real system behaves. Using this method, given a set of data from a patient, it is possible to understand the metabolism of a patient by finding these values. It is important to fit this data to the model so advanced control of the glucose metabolism can be performed.

II. Numerical Methods

Given the need to work with a dynamic process model, the optimal parameter estimation problem becomes more difficult to formulate and solve. For a traditional numerical optimization problem using a steady-state model, the optimization is usually formulated as

$$\begin{aligned} \min_{x,p} \quad & \varphi(x,p) \\ \text{s.t.} \quad & f(x,p) = 0 \\ & g(x,p) \le 0 \\ & x^L \le x \le x^U \\ & p^L \le p \le p^U \end{aligned} \tag{5}$$

The objective function φ depends on the system states x and the design parameters p. It typically corresponds to system costs or lost profit. The behavior of the process is enforced by the equality constraints $f(x,p)$ obtained from the process model. Furthermore, it is common to add additional path constraints for performance reasons or safety/equipment protection reasons. The states x and design parameters p are generally expected to lie within allowable ranges $\left[x^L, x^U\right]$ and $\left[p^L, p^U\right]$, respectively. Generally, these problems fall in a class of optimizations known as constrained Nonlinear Programming problems (NLPs) [8], [9]. Under certain conditions, these problems satisfy more restrictive categories such as convex NLPs, Quadratic Programs (QPs), or Linear Programs (LPs). Regardless of the exact classification, a number of well-established methods exist for solving these design problems.

However, when the dynamic modeling equations must be used, the problem becomes considerably more difficult. There are several classes of dynamic models and a variety of ways to represent them mathematically, but for the purpose of this work it is assumed that the system can be described as a set of linearly-implicit DAEs

$$M\frac{dx}{dt} = f(t,x,p) \tag{6}$$

whose dynamic behavior depends on the design variables p. The linearly-implicit differential algebraic (DAE) form also includes the so-called *mass matrix M* that is assumed to have constant coefficients. When M has full rank, the equations comprise a set of ordinary differential equations (ODEs); a singular matrix denotes a DAE system. Making the corresponding substitution into the optimal design problem (5) yields the following *Dynamic Optimization* problem:

$$\begin{aligned} \min_{x,p} \quad & \varphi(x(t),p) & \\ \text{s.t.} \quad & M\tfrac{dx}{dt} - f(t,x(t),p) = 0 & \forall t \in \left[t_0, t_f\right] \\ & g(x(t),p) \le 0 & \forall t \in \left[t_0, t_f\right] \\ & x^L \le x(t) \le x^U & \forall t \in \left[t_0, t_f\right] \\ & p^L \le p \le p^U & \end{aligned} \tag{7}$$

The primary difference in this optimization problem (7) is that the finite number of equality and inequality constraints present in problem (5) must now be satisfied at an infinite number of points in the continuous interval $\left[t_0, t_f\right]$. This cannot be solved directly, but different approaches have been developed to approximate this class of problems with finite-dimensional representations that can be solved. These approaches include *variational* methods, *sequential* or *control variable parameterization* methods, *multiple shooting* and *quasi-sequential* methods, and *simultaneous* or *direct transcription* methods.

A. Simultaneous Dynamic Optimization

Taking into consideration each methods strengths and flaws, a simultaneous approach was selected for this work for a variety of reasons. Unlike many of the other methods, the simultaneous approach applies a discretization method in the time domain to the state variables, which converts the continuous DAE equations into a large set of coupled algebraic constraint equations. Collocation on finite elements is the preferred discretization as it has a number of desirable stability and convergence properties. The collocation method has been used successfully for engineering applications for quite some time [10] and is most commonly used as a technique for solving two-point boundary value problems with the basic formulation covered in numerous locations including [11], [12], [13]. However, it can be used effectively for initial value problems as well, particularly when the collocation points are chosen to be the Radau quadrature points. The basic procedure is described here.

A polynomial approximation for each unknown state is constructed on k intervals (finite elements) using the trial function $\hat{x}_k$ defined below:

$$\hat{x}_k(t) = \sum_{i=0}^{NC} x_{k,i}\,\ell_i(t) \tag{8}$$

where $x_{k,i}$ are the unknown state values at $NC+1$ points in the finite element and $\ell_i(t)$ is the Lagrange interpolating polynomial corresponding to the point t_i in the interval as given by

$$\ell_i(t) = \prod_{j \neq i} \frac{(t-t_j)}{(t_i-t_j)} \tag{9}$$

which has the useful property that

$$\ell_i(t_j) = \delta_{ij} \tag{10}$$

where δ_{ij} is the Kronecker delta function. The derivative of the trial function is found by differentiating the basis polynomials:

$$\frac{d\hat{x}_k}{dt} = \sum_{i=0}^{NC} x_{k,i}\,\dot{\ell}_i(t) \tag{11}$$

When evaluating the derivative at the points t_j within the collocation element, which include the NC Radau quadrature points and also the left endpoint, it can be shown that the value is a weighted linear combination of the unknown state values $x_{k,j}$ scaled by the inverse length of the interval Δt_k as shown in Equation 12.

$$\left.\frac{d\hat{x}_k}{dt}\right|_{t=t_{k,j}} = \frac{1}{\Delta t_k}\sum_{i=0}^{NC} c_{j,i}\,x_{k,i} \tag{12}$$

Using this trial formulation, it is then straightforward to apply collocation to a system of DAEs described by Equation 6. The result is a large set of coupled nonlinear *algebraic* equations in the form

$$\mathbf{M}\sum_{i=0}^{NC} c_{j,i}\mathbf{x}_{k,i} - \Delta t_k \mathbf{f}(t_{k,j}, \mathbf{x}_{k,j}, \mathbf{p}) = 0$$
$$j = 1 \ldots NC,\; k = 1 \ldots NE \tag{13}$$

This results in a set of $N \times NC \times NE$ equations in $N \times (NC+1) \times NE$ unknowns where N is the number of states and NE is the number of elements, so an additional $N \times NE$ equations are required to obtain a unique solution. Given the nature of DAE systems, the state variables are expected to be continuous at interval boundaries

$$\mathbf{x}_{k,0} = \mathbf{x}_{k-1,NC} \tag{14}$$

and it is assumed that the initial conditions are known at least as a function of the initial time $t_{0,0}$ and parameters $\mathbf{p}$:

$$\mathbf{x}_{0,0} = \mathbf{x}_{IC}(t_{0,0}, \mathbf{p}) \tag{15}$$

Thus, the DAE governing equations are converted to a very large set of algebraic equations. By replacing the continuous constraints in (7) with the collocations equations 13, 14, and 15, evaluating path constraints at the collocation points, and including the state values $\mathbf{x}_{k,j}$ directly in the optimization formulation, the dynamic optimization problem is converted to a standard NLP of the form given by (5). In this work, three-point collocation was selected, resulting in piecewise cubic trial solutions. Traditionally, Radau quadrature includes the left endpoint of the interval as a quadrature point. In this work, it is useful to invert the ordering and include the right endpoint instead. In general, the locations for the $n-1$ free collocation points in n-point Radau collocation can be found as the roots to the polynomial

$$RP(t) = \frac{P_{n-1}(t) + P_n(t)}{1+t} \tag{16}$$

where $P_n(t)$ is the n^{th} Legendre polynomial. This polynomial returns points on the interval $[-1, 1]$ and assumes that the left endpoint (-1) is the included point. These points can be inverted and then scaled to the more useful interval $[0, 1]$ for use in collocation. Ultimately, this procedure produces a numerically stable system of equations and exhibits favorable properties for the solution of stiff ODEs or DAEs [14]. Moreover, with appropriate assumptions, it can be shown that the necessary conditions for optimality of the reformulated problem approach those for the original problem generated using variational methods as the number of collocation points increases [15]. Thus, one can be confident that the optimal solution obtained by the simultaneous method is near to the solution of the original dynamic optimization problem.

B. Selection of Optimization Algorithm

As stated previously, the collocation-based simultaneous dynamic optimization approach results in an NLP with $N \times NE \times (NC+1)$ equality constraints. Clearly, the problem size grows quickly and requires solution methods specifically designed for large-scale optimization. Therefore, the IPOPT package [16], [17] was chosen as the candidate solver due to its successful use in other dynamic programming applications with as many as $10^5 - 10^6$ variables [18], [19]. Its solving capabilities were crucial to the feasibility and utility of this approach. Specifically, IPOPT is an interior-point implementation of the Sequential Quadratic Programming (SQP) approach to solving constrained NLPs. It utilizes exact second derivative information (when available) to compute Newton-like search directions and explores these directions using a filter line search to ensure convergence of the algorithm. The FORTRAN 77 version of IPOPT solves general constrained NLPs of the form

$$\begin{aligned} \min_{x} \quad & \varphi(x) \\ s.t. \quad & c(x) = 0 \\ & x^L \le x \le x^U \end{aligned} \tag{17}$$

Since general nonlinear inequalities are not handled directly, it is the user's responsibility to incorporate slack variables s manually as shown in Eq. 18:

$$g(x) \le 0 \quad \Rightarrow \quad g(x) + s = 0 \quad s.t. s \ge 0 \tag{18}$$

C. Implementation

The task of formulating the optimization problem itself would be challenging, time-consuming, and error-prone due to its large size. IPOPT generally requires five user-provided functions: an *objective* function, the *gradient of the objective* function, the *constraint* functions, the *gradient of the constraints*, and the *Hessian of the Lagrangian*. For detailed models, calculating the necessary partial derivatives is cumbersome. Moreover, the collocation procedure is intricate and repetitive. Thus, the utility of this method is dependent on the ease with which the problems can be formulated; the greater the extent that the process can be automated, the more useful this optimization approach becomes.

Consequently, a multi-stage procedure was developed including pre-processing that is accomplished using the symbolic capabilities of the MAPLE [20] to generate first and second derivatives using the `GRADIENT` and `JACOBIAN` commands in the `codegen` package. In particular, the gradients of the functions needed are with respect to both the state variables and the design parameters, as well as the state, parameter, and mixed Hessians. After these procedures have been created, the corresponding MATLAB® function code can be generated automatically by utilities in MAPLE's `CodeGeneration` package. Once the MATLAB® functions are created, the user develops a script that populates a structure with the problem information: function names, number of states, number of constraints, number of time steps, size of time step, initial parameters, initial condition guesses, state/parameter bounds, etc. Once the structure is fully populated, a function is executed that calls a DAE solution algorithm to perform a forward simulation to determine consistent initial conditions for all state variables in the optimization. An appropriate number of slack variables are created and initialized. Finally, the IPOPT wrapper is called to solve the optimization problem. All function evaluations needed by IPOPT are performed by callbacks to MATLAB® where the functions named in the problem structure are used to generate the correct objective and constraint functions based. Assuming successful termination of IPOPT, the optimal state and parameter values are returned to the user along with the final objective value and optionally a significant amount of information related to the solution of the optimization problem. Solving different optimization problems is now reduced primarily to a matter of changing values within the specialized problem structure. For this work, it is assumed that the optimization problem depends on the system states x and the model parameters p. Generally, a user will need to supply information about three different functions: the objective function, the dynamic constraint (Right-Hand Side (RHS), see Eq. 6) function, and the optional path constraint function.

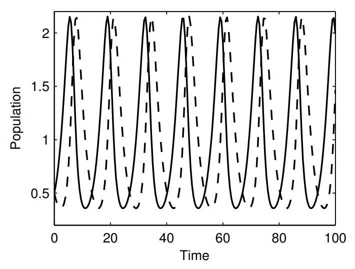

Fig. 1. The prey (*solid*) and predator (*dashed*) data used in the parameter regression problem.

III. Results and Discussion

A. An Illustrative Example

Consider a simple parameter regression problem using the well-known Lotka-Volterra predator-prey model [21], [22]. The populations of the prey x and predators y are governed by the following set of nonlinear ordinary differential equations:

$$\begin{aligned} \dot{x} &= \alpha x - \beta xy \\ \dot{y} &= \delta xy - \gamma y \end{aligned} \tag{19}$$

Synthetic data was created by numerically integrating these equations with the following parameter values: $\alpha = \beta = \delta = \gamma = 0.5$. The initial conditions were selected to be $x(0) = y(0) = 0.5$. The model was sampled at 0.5 unit intervals to a time of 100 units creating a data record with 201 points as shown in Figure 1.

For simplicity, it was assumed that the initial conditions $x(0)$ and $y(0)$ and parameters α and γ were known exactly, leaving two unknown parameters β and δ to be determined through parameter estimation. Two different optimization studies were conducted to obtain the values of β and δ. The first used a traditional nonlinear least squares approach. The second optimization study utilized the simultaneous optimization strategy described above to minimize the same Sum-Squared Error (SSE) objective function given by Eq. 20. However, since

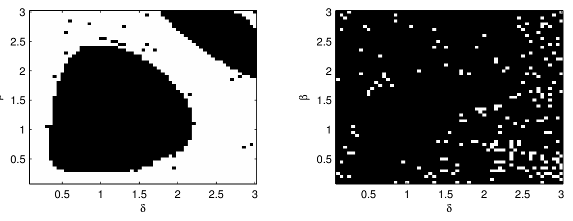

Fig. 2. Regions of convergence for the predator-prey problem using standard nonlinear least squares regression (left) and simultaneous dynamic optimization to minimize SSE (right). Dark areas indicate starting points that converge to the global solution.

the dynamic equations are discretized in time and the state variables included within the optimization problem itself, it is possible to include bounds on the state variables within the optimization problem itself. It was hypothesized that placing bounds on the states to tightly enclose the data would provide extra information to the optimization algorithm, increasing the likelihood of converging to the optimal solution.

During each iteration k of the optimization, the dynamic solver DASSL [23] was used to compute the state profiles $\hat{x}$ and $\hat{y}$ based on the given parameter values $[\beta_k, \delta_k]$. The objective function was then computed according to Eq. 20.

$$\varphi(\beta, \delta) = \sum_{i=0}^{200} \left[(\hat{x}_i - x_i)^2 + (\hat{y}_i - y_i)^2 \right] \tag{20}$$

Based on initial parameter guesses $[\beta_0, \delta_0]$, the gradient-based nonlinear program (NLP) solver IPOPT [17] was used to minimize Eq. 20. Bounds of $[0.001, 10]$ were specified for both parameters. Since the exact parameter values were known — $\{0.5, 0.5\}$ — it was desired to determine the region of initial parameter space that would lead to the global solution when using this optimization strategy. Initial values for each parameter were selected in the range $[0.1, 3]$ in increments of 0.05, resulting in a total of 3481 optimization cases. The regions of convergence for this method are presented in Figure 2. Regions of parameter space that produced profiles that exceeded the range of the data would be automatically excluded from the possible solutions.

As seen in the figure, the initial points that tend to converge predominantly lie in two disjoint regions. In all, 1634 cases converged to the global solution or about 46.9% of the total number for the first study. Also, cases in which either of the two parameters were initialized to a value lower than the true value were unlikely to converge to the correct answer. Using the simultaneous approach, the global solution was found for 3277 of 3481 cases, about 94.1%. Not only is this a vast improvement, but it should also be noted that when using the simultaneous approach the cases that did not converge to the global did not fail by converging to a local minimum; the NLP algorithm terminated with an error condition after exceeding a maximum number of iterations. Many more of the trials converge to the global solution, and that convergence is more uniformly distributed throughout the entire region that was considered. Hence, not only does the simultaneous approach offer a greater chance of success than traditional methods, it also has a more useful failure mode. Termination due to exceeding the maximum number of iterations is an automatic flag to the user that the results should not be considered optimal. However, the user may have difficulty recognizing the problem if the algorithm converges to a local minimum instead of the global minimum. This is especially applicable when regressing experimental data as the "true" values of the parameters — and thus the global objective value — are unknown.

B. Diabetic Patient Example

Using MATLAB®, various different random patient parameter values were used to generate data. In all cases, the program was able to obtain the parameters given to it from fitting the Bergman model to the generated data. All data was recorded with initial values of zero, and in the time range between 0 and 1300 min, with time intervals of 1 min. Figure 3 shows that both the original data and the model data are very similar, so difference between them is not visually noticeable.

The peaks at 200, 450 and 700 min correspond approximately to three meals. An afternoon snack was inserted at time 550 min. The smaller peaks are insulin levels. The patient is assumed to have administered some insulin after the meal, causing the glucose concentration to rapidly drop. This procedure was repeated for twenty different parameter sets. Table I shows the parameter values used for generating data and the fitting error after the optimization. Some parameter sets did not reach an optimal solution in the number of maximum iterations specified (although the error was being minimized with each iteration), and thus were not included in the table.

IV. Conclusions

The program converged easily by placing bounds on the states near the maximum and minimum values found in the data. In the future, it could be advantageous to find certain time-varying bounds that follow the data, acting as a convergence band, so to speak, so the model could potentially converge with even greater ease.

Using the diabetic patient model, the method was able to find the original parameters robustly and accurately. The vast difference between the parameter sets in Table I may suggest the ability to describe the

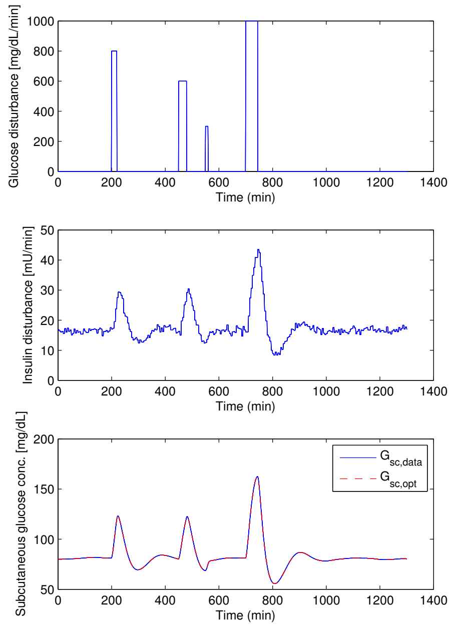

Fig. 3. Glucose intake rate $D(t)$[mg/L/min] (top), insulin disturbance $U(t)$[mU/min] (middle) and simulated subcutaneous glucose concentration measurements [mg/dL] (bottom) vs time (min). This data was generated using the Bergman model, with $P_1 = 2.8735 \times 10^{-2}$, $P_2 = 2.8344 \times 10^{-2}$, $P_3 = 5.035 \times 10^{-5}$, $n = \frac{5}{54}$, $G_b = 81mg/dL$, $V_I = 12L$, $I_b = 15mU/L$, $R_{ut} = 0$, $\theta = 5min$.

TABLE I

VARIOUS CONVERGING PARAMETER SETS AND THEIR RESPECTIVE FITTING ERRORS.

$P_1 \times 10^2$	$P_2 \times 10^2$	$P_3 \times 10^5$	error$\times 10^{10}$
2.8735	2.8344	5.035	4.2109
3.8596	0.6305	9.7059	2.0811
3.8287	1.9415	8.0028	1.4373
0.5675	1.6870	9.1574	3.0970
3.1688	3.8380	6.5574	1.9462
0.1428	3.3965	9.3399	24.3243
2.7149	3.0310	7.4313	2.0359
1.5689	2.6219	1.7119	4.1128
2.8242	0.1273	2.7692	1.5977
0.1847	0.3885	8.2346	6.1748
2.7793	1.2684	9.5022	3.2828

glucose metabolism of a wide variety of patients using this method. However, because the data was computer-generated, it is still unknown whether this can be done with real patient data. In order to follow up on this method, it would be useful to use actual patient data and attempt to fit it to this model. In addition, any diseases or conditions that are described by a mathematical model could also be subjected to the procedure presented, in order to find parameters that best describe the patient, leading toward model-predictive control as a possible treatment option.

REFERENCES

[1] D. M. Nathan, “Long-Term Complications of Diabetes Mellitus,” *The New England Journal of Medicine*, vol. 328, no. 23, pp. 1676–1685, 1993.

[2] R. J. Rubin, W. M. Altman, and D. N. Mendelson, “Health Care Expenditures for People with Diabetes Mellitus, 1992,” *Journal of Clinical Endocrinology and Metabolism*, vol. 74, no. 4, pp. 809A–809F, 1994.

[3] R. Bergman, D. Finegood, and M. Ader, “Assessment of insulin secretion in vivo,” *Endocrine Rev.*, vol. 6, pp. 45–86, 1985.

[4] R. Bergman, Y. Ider, C. Bowden, and C. Cobelliar, “Quantitative estimation of insulin sensitivity,” *Am. J. Physiol.*, vol. 236, pp. 667–677, 1979.

[5] R. N. Bergman, L. S. Philips, and C. Cobelli, “"physiological evaluation of factors controlling glucose tolerance in man",” *J.Clin.Invest*, vol. 68, pp. 1456–1467, 1981.

[6] M. E. Fisher, “A semi-closed loop algorithm for the control of blood-glucose levels in diabetes.” *IEEE.Trans.Biomed.Eng*, vol. 38, pp. 57–61, 1991.

[7] S. Lynch and B. W. Bequette, “Model predictive control of blood glucose in type i diabetics using subcutaneous glucose measurements,” *Proceeding of the American Controls Conference*, no. 4039-4043, 2002.

[8] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming*. Wiley, 1992.

[9] C. A. Floudas, “Global Optimization in Design and Control of Chemical Process Systems,” *J. Proc. Cont.*, vol. 10, no. 125-134, 2000.

[10] J. C. Slater, “Electronic Energy Bands in Metals,” *Phys. Rev.*, vol. 45, pp. 794–801, 1934.

[11] L. Lapidus and G. F. Pinder, *Numerical Solution of Partial Differential Equations in Science and Engineering*. New York: John Wiley & Sons, 1982.

[12] M. E. Davis, *Numerical Methods and Modeling for Chemical Engineers*. New York: John Wiley & Sons, 1984.

[13] R. G. Rice and D. D. Do, *Applied Mathematics and Modeling for Chemical Engineers*. New York: John Wiley & Sons, 1995.

[14] S. Kameswaran and L. T. Biegler, “Simultaneous Dynamic Optimization Strategies: Recent Advances and Challenges,” *In Preparation*, vol. http://tinyurl.com/2m43s3, 2007.

[15] ——, “Convergence Rates for Direct Transcription of Optimal Control Problems Using Collocation at Radau Points,” *In Press*, vol. http://tinyurl.com/2npr89, 2007.

[16] A. Wächter, “An Interior Point Algorithm for Large-Scale Nonlinear Optimization with Applications in Process Engineering,” Ph.D. dissertation, Carnegie Mellon University, January 2002.

[17] A. Wächter and L. T. Biegler, “On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming,” *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.

[18] T. Jockenhövel, L. T. Biegler, and A. Wächter, “Dynamic Optimization of the Tennessee Eastman Process Using the OptControlCentre,” *Comput. Chem. Eng.*, vol. 27, pp. 1513–1531, 2003.

[19] C. D. Laird, L. T. Biegler, B. G. van Bloemen Waanders, and R. A. Bartlett, “Contamination Source Determination for Water Networks,” *Journal of Water Resources Planning and Management*, vol. 131, no. 2, pp. 125–134, 2005.

[20] Maplesoft, *Maple Reference Guide*. Springer Verlag, 2000.

[21] A. J. Lotka, *Elements of Physical Biology*. Baltimore: Williams & Wilkins Co., 1925.

[22] V. Volterra, “Variazioni e Fluttuazioni del Numero D’Individui in Specie Animali Conviventi,” *Mem. R. Accad. Naz. dei Lincei*, vol. VI, no. 2, 1926.

[23] L. R. Petzold, “A Description of DASSL: A Differential / Algebraic System Solver,” Sandia National Laboratories, Tech. Rep. SAND82-8637, September 1982. [Online]. Available: www.netlib.org/ode/ddassl.f