Implementation of Model Predictive Control with Modified Minimal Model on Low-Power RISC Microcontrollers

Binh P. Nguyen, Yvonne Ho, Zimei Wu and Chee-Kong Chui
Department of Mechanical Engineering, National University of Singapore
9 Engineering Drive 1, Singapore 117576
{phubinh, yvonne.yw.ho, g0800347, mpecck}@nus.edu.sg

ABSTRACT

Due to the ability of modeling multivariable systems and handling constraints in the control framework, model predictive control (MPC) has received a lot of interest from both academic and industrial communities. Although it is an established control technique, implementing MPC on small-scale devices is a challenge since we need to handle complicated issues of the control framework using limited computational power and hardware resources. This paper presents our implementation of MPC with constraints on the Texas Instruments MSP430 16-bit microcontroller platform. The MPC operational constraints which are supported in our design include rate of change, amplitude and output constraints, while the associated optimization problem is solved using a primal-dual interior-point algorithm based on predicator-corrector method. Our implementation is demonstrated in a prototype of a real-time close-loop blood glucose regulation system using a modification of the minimal model. Experimental results show that our system is able to achieve desired diabetes management, and the chosen microprocessor is capable of performing the MPC algorithm accurately with high energy-efficiency and in real-time.

Categories and Subject Descriptors

C.3 [**Special-Purpose and Application-Based Systems**]: Process control systems

General Terms

Design

Keywords

Model predictive control, glucose regulation, minimal model, embedded system.

1. INTRODUCTION

SoICT 2012, August 23-24, 2012, Ha Long, Vietnam.

Model predictive control (MPC) is an effective advanced control method since it can model multivariable systems and handle physical constraints in the control frame work. The principle of MPC is to solve a quadratic programming (QP) problem within the sampling period. This is a challenge when applying MPC to complex systems requiring fast response and/or small-scale devices where computational power and memory resources are limited. Recently, there have been research works on implementation of MPC using Application-Specific Integrated Circuit (ASIC) [5], and Field-Programmable Gate Array (FPGA) [11, 8, 3]. Although both of the two platforms can achieve high processing speeds, FPGAs are more flexible and have shorter design cycles than ASICs. The main drawbacks of FPGAs are their high power consumption and high cost. In addition to ASICs and FPGAs, MPC can be also implemented on general purpose microcontrollers [7, 6, 1]. Since a microcontrollers can be considered custom built mini computers in an IC, they are more flexible than ASICs and FPGAs in embedded system design. Other advantages of microcontrollers compared to FPGAs include their low power consumption and low cost. This paper proposes an implementation of MPC on low-power RISC microcontrollers, which can be widely applied to numerous miniaturized control systems.

Diabetes Mellitus is a widespread disease caused by high blood glucose in the patient's body (Diabetes.com). High blood sugar sets off processes that can lead to complications, like heart, kidney, and eye disease, or other serious problems. There are two types of Diabetes: Type 1 and Type 2 Diabetes. In addition to medication and/or insulin shots, Type 1 and some Type 2 patients are required to measure their blood sugar level everyday with the aid of a blood glucose monitor. A blood glucose monitor or glucose meter is a small portable battery-powered electronic device mainly used to determine the deviations of the patient's blood glucose level from the normal healthy level. There are many different types of blood glucose monitors in the market for diabetes management. Patients may be required to use lancets to draw blood from their finger. Recent models allow people to test the glucose level with blood extracted from other areas. The most recent models use sensor pod that is placed underneath the skin to constantly measure the blood sugar via an external monitoring device. Sensors from the latest model could be used continuously for up to a week. There are many interests in the healthcare industry to develop an implantable solution or artificial pancreas that could remove the patient's pains from blood extraction and needle insertion as well as better diabetes management out-

comes. This paper also reports an application of our MPC controller for continuous diabetes management.

2. MODEL PREDICTIVE CONTROL

This section presents our extensions of the MPC design methodology from single-input and single-output (SISO) systems which is described in [15] to multi-input and multi-output (MIMO) systems.

MPC systems are designed based on a state-space model of the plant. Assuming the plant has m inputs, q outputs, and n_1 states, the formulation of the predictive control problem can be stated as below:

$$x_m(k+1) = A_m x_m(k) + B_m u(k), \tag{1a}$$
$$y(k) = C_m x_m(k), \tag{1b}$$

where u is the manipulated or input variable, y is the process output, and x_m is the state variable; A_m, B_m, and C_m have dimensions of $n_1 \times n_1$, $n_1 \times m$, and $q \times n_1$, respectively.

Denoting $\Delta x_m(k) = x_m(k) - x_m(k-1)$, $\Delta u(k) = u(k) - u(k-1)$, and $x(k) = \begin{bmatrix}\Delta x_m(k)^T & y(k)\end{bmatrix}^T$, with a difference operation on both sides of (1a) and (1b), we obtain the following state-space model:

$$\overbrace{\begin{bmatrix}\Delta x_m(k+1)\\ y(k+1)\end{bmatrix}}^{x(k+1)} = \overbrace{\begin{bmatrix}A_m & o_m^T\\ C_m A_m & I_{q\times q}\end{bmatrix}}^{A}\overbrace{\begin{bmatrix}\Delta x_m(k)\\ y(k)\end{bmatrix}}^{x(k)} + \underbrace{\begin{bmatrix}B_m\\ C_m B_m\end{bmatrix}}_{B}\Delta u(k), \tag{2a}$$

$$y(k) = \underbrace{\begin{bmatrix}o_m & I_{q\times q}\end{bmatrix}}_{C}\underbrace{\begin{bmatrix}\Delta x_m(k)\\ y(k)\end{bmatrix}}_{x(k)}, \tag{2b}$$

where o_m is a $q \times n_1$ zero matrix, and $I_{q\times q}$ is the identity matrix with dimensions of $q \times q$.

2.1 Solution of predictive control

At the sampling instant k_i, we assume that the state variable $x(k_i)$ is available to provide the current information of the plant. Then, the future state variables are predicted for N_p samples: $x(k_i+1|k_i), x(k_i+2|k_i), ..., x(k_i+N_p|k_i)$, and the optimal future control trajectory is described within N_c samples ($N_c \le N_p$): $\Delta u(k_i), \Delta u(k_i+1), ..., \Delta u(k_i+N_c-1)$. By defining

$$\Delta U = \begin{bmatrix}\Delta u(k_i)^T & \Delta u(k_i+1)^T & \cdots & \Delta u(k_i+N_c-1)^T\end{bmatrix}^T \tag{3a}$$

$$Y = \begin{bmatrix}y(k_i+1|k_i)^T & y(k_i+2|k_i)^T & \cdots & y(k_i+N_p|k_i)^T\end{bmatrix}^T, \tag{3b}$$

and sequentially calculating the predicted future state and output variables using the future control parameters, we obtain

$$Y = Fx(k_i) + \Phi\Delta U, \tag{4}$$

where

$$F = \begin{bmatrix}CA & CA^2 & \cdots & CA^{N_p}\end{bmatrix}^T \tag{5}$$

and

$$\Phi = \begin{bmatrix} CB & 0 & \cdots & 0\\ CAB & CB & \cdots & 0\\ \vdots & \vdots & \cdots & \vdots\\ CA^{N_p-1}B & CA^{N_p-2}B & \cdots & CA^{N_p-N_c}B\end{bmatrix}. \tag{6}$$

For a given set-point signal $r(k_i) = [r_1(k_i)\ r_2(k_i) \cdots r_q(k_i)]^T$ at sample time k_i, we need to find the optimal control parameter vector ΔU so that within the prediction horizon, the predicted output variables are as close as possible to the set-points. The cost function that reflects this control objective is defined as

$$J = (R_s - Y)^T(R_s - Y) + \Delta U^T \bar{R}\Delta U, \tag{7}$$

where $R_s = \begin{bmatrix}I_{q\times q} & I_{q\times q} & \cdots & I_{q\times q}\end{bmatrix}^T r(k_i) = \bar{R}_s r(k_i)$, and $\bar{R}$ is a block matrix with $m \times m$ blocks of $r_\omega I_{N_c\times N_c}$ ($r_\omega \ge 0$), in which r_ω is a tuning parameter, and $I_{N_c\times N_c}$ is the $N_c \times N_c$ identity matrix.

From (4) and (7), by solving $\frac{\partial J}{\partial \Delta U} = 0$ for the minimum of J, the optimal solution is:

$$\Delta U = (\Phi^T\Phi + \bar{R})^{-1}\Phi^T(R_s - Fx(k_i)). \tag{8}$$

Note that only the first m elements of ΔU, i.e. $\Delta u(k_i)$, will be utilized. At next sample time, the new state variable $x(k_i+1)$ is measured and used to calculate the new control trajectory.

2.2 State estimate predictive control

At time k_i, the state variable $x(k_i)$ is assumed to be available. However, in practice, not all state variables are available or measurable. A solution to this issue is to estimate the state variable $x(k)$ using a *state observer*. We calculate the state variable $\hat{x}_m(k)$ based on the plant model in (1), and use the error signal of output as a feedback to improve the estimation as below

$$\hat{x}_m(k+1) = \underbrace{A_m\hat{x}_m(k) + B_m u(k)}_{\text{model}} + \underbrace{K_{ob}(y(k) - C_m\hat{x}_m(k))}_{\text{correctionterm}}, \tag{9}$$

where K_{ob} is the observer gain matrix.

By substituting (1b) into (9), we deduce that the error $\tilde{x}_m(k) = x_m(k) - \hat{x}_m(k)$ satisfies the difference equation:

$$\begin{aligned}\tilde{x}_m(k+1) &= A_m\tilde{x}_m(k) - K_{ob}C_m\tilde{x}_m(k)\\ &= (A_m - K_{ob}C_m)\tilde{x}_m(k).\end{aligned} \tag{10}$$

For a given initial error state $\tilde{x}_m(0) \ne 0$, we have

$$\tilde{x}_m(k) = (A_m - K_{ob}C_m)^k\tilde{x}_m(0). \tag{11}$$

If the observer gain K_{ob} is chosen so that the error system matrix $A_m - K_{ob}C_m$ has all eigenvalues inside the unit circle, then the error system (11) is stable and $\|\tilde{x}_m(k)\| \to 0$ as $k \to \infty$. This means that the estimated state variable $\hat{x}_m(k)$ converges to $x_m(k)$.

At sample time k_i, with the information of $\hat{x}(k_i)$ replacing $x(k_i)$, the estimation of state variable in (2) is modified as

$$\hat{x}(k_i+1) = A\hat{x}(k_i) + B\Delta u(k_i) + K_{ob}(y(k_i) - C\hat{x}(k_i)). \tag{12}$$

The cost function and optimal solution are similar to that of (7) and (8), respectively, except that the term $x(\cdot)$ is replaced by $\hat{x}(\cdot)$.

2.3 MPC with constraints

We need to modify the above solution if operational constraints are introduced to the model. Typical constraints are:

(1) Constraints on rate of change of all the control variables within the future control trajectory started from timestep k_i:

$$\Delta U^{\min} \leq \Delta U \leq \Delta U^{\max}, \tag{13}$$

where ΔU was defined in (3a); $\Delta U^{\min}$ and $\Delta U^{\max}$ are column vectors with N_c elements of the lower limits $\Delta u^{\min} = \begin{bmatrix}\Delta u_1^{\min} & \Delta u_2^{\min} & \cdots & \Delta u_m^{\min}\end{bmatrix}$ and the upper limits $\Delta u^{\max} = \begin{bmatrix}\Delta u_1^{\max} & \Delta u_2^{\max} & \cdots & \Delta u_m^{\max}\end{bmatrix}$, respectively.

(2) Constraints on amplitude of all the control variables within the future control trajectory:

$$U^{\min} \leq U \leq U^{\max}, \tag{14}$$

where $U = \begin{bmatrix}u(k_i)^T & u(k_i+1)^T & \cdots & u(k_i+N_c-1)^T\end{bmatrix}^T$; $U^{\min}$ and $U^{\max}$ are column vectors with N_c elements of the lower limits $u^{\min} = \begin{bmatrix}u_1^{\min} & u_2^{\min} \cdots & u_m^{\min}\end{bmatrix}$ and the upper limits $u^{\max} = \begin{bmatrix}u_1^{\max} & u_2^{\max} & \cdots & u_m^{\max}\end{bmatrix}$, respectively.

(3) Constraints on the outputs within the prediction trajectory:

$$Y^{\min} \leq Y \leq Y^{\max}, \tag{15}$$

where Y was defined in (3b); $Y^{\min}$ and $Y^{\max}$ are column vectors with N_p elements of the lower limits $y^{\min} = \begin{bmatrix}y_1^{\min} & y_2^{\min} & \cdots & y_q^{\min}\end{bmatrix}$ and the upper limits $y^{\max} = \begin{bmatrix}y_1^{\max} & y_2^{\max} & \cdots & y_q^{\max}\end{bmatrix}$, respectively.

The constraints in (13) can be rewritten in a matrix form:

$$\begin{bmatrix}-I \\ I\end{bmatrix} \Delta U \leq \begin{bmatrix}-\Delta U^{\min} \\ \Delta U^{\max}\end{bmatrix}. \tag{16}$$

In (14), $U = \begin{bmatrix}u(k_i)^T & u(k_i+1)^T & \cdots & u(k_i+N_c-1)^T\end{bmatrix}^T$, which can be expressed as

$$U = \underbrace{\begin{bmatrix}I \\ I \\ \vdots \\ I\end{bmatrix}}_{C_1} u(k_i-1) + \underbrace{\begin{bmatrix}I & O & \cdots & O \\ I & I & \cdots & O \\ \vdots & \vdots & \ddots & \vdots \\ I & I & \cdots & I\end{bmatrix}}_{C_2} \underbrace{\begin{bmatrix}\Delta u(k_i) \\ \Delta u(k_i+1) \\ \vdots \\ \Delta u(k_i+N_c-1)\end{bmatrix}}_{\Delta U}, \tag{17}$$

where I and O are the identity and zero matrices with dimension of $m \times m$. From (17), we can represent (14) as

$$\begin{bmatrix}-C_2 \\ C_2\end{bmatrix} \Delta U \leq \begin{bmatrix}-U^{\min} + C_1 u(k_i-1) \\ U^{\max} - C_1 u(k_i-1)\end{bmatrix}. \tag{18}$$

From (4), the output constraints can be rewritten as

$$\begin{bmatrix}-\Phi \\ \Phi\end{bmatrix} \Delta U \leq \begin{bmatrix}-Y^{\min} + Fx(k_i) \\ Y^{\max} - Fx(k_i)\end{bmatrix}. \tag{19}$$

Finally, solving the MPC with constraints is to find ΔU that minimizes the cost function in (7) subject to the inequality constraints in (16), (18), and (19). This is a quadratic programming problem with linear inequality constraints:

$$J = \frac{1}{2} x^T E x + x^T F, \tag{20a}$$

$$Mx \leq \gamma, \tag{20b}$$

where E and F can be inferred from (7); M and γ are a matrix and a vector reflecting the constraints. In the case that the constraints are fully imposed, M and γ have $4mN_c + 2qN_p$ rows.

The optimization problem in (20) can be solved using various methods. This work uses a *primal-dual interior-point* algorithm [13] employing the *predictor-corrector* method [16] which is computational and resource efficient.

3. LINEAR INEQUALITY CONSTRAINED QUADRATIC PROGRAMMING

This section describes a primal-dual interior-point method to solve the quadratic program shown in (20) where x is a $n \times 1$ vector (in our problem, x is ΔU, and $n = m \times N_c$), E is a $n \times n$ matrix, F is a $n \times 1$ vector, M is a $l \times n$ matrix, where l is the number of inequality constraints, and γ is a $l \times 1$ vector.

Denoting λ as the Lagrange multipliers vector of the optimization problem, and introducing the *slack vector* $s = \gamma - Mx, s \geq 0$, then the Karush-Kuhn-Tucker (KKT) conditions for this problem can be represented as

$$\Psi(x,\lambda,s) = \begin{bmatrix}Ex + F + M^T\lambda \\ s + Mx - \gamma \\ \Lambda S e\end{bmatrix} = 0, \tag{21a}$$

$$\lambda, s \geq 0, \tag{21b}$$

where $\Lambda = diag(\lambda_1, \lambda_1, \ldots, \lambda_l)$, $S = diag(s_1, s_2, \ldots, s_l)$, and $e = [1, 1, \ldots, 1]^T$.

In order to find the solutions (x^*, λ^*, s^*), primal-dual interior-point methods apply variants of Newton's method to (21a) and modify the search directions and step lengths so that the inequalities (21b) satisfied strictly, i.e., $\lambda > 0$ and $s > 0$, at each iteration. From the current point (x, λ, s), we can obtain the search direction $(\Delta x, \Delta\lambda, \Delta s)$ by solving

$$\Psi'(x,\lambda,s) \begin{bmatrix}\Delta x & \Delta\lambda & \Delta s\end{bmatrix}^T = -\Psi(x,\lambda,s), \tag{22}$$

where $\Psi' = \begin{bmatrix}E & M^T & 0 \\ M & 0 & I \\ 0 & S & \Lambda\end{bmatrix}$ is the Jacobian of Ψ. After $(\Delta x, \Delta\lambda, \Delta s)$ is obtained, the new iterate is calculated as

$$(x,\lambda,s) + \alpha(\Delta x, \Delta\lambda, \Delta s), \tag{23}$$

where the *step length* $\alpha \in [0, 1]$ is determined so that the condition (21b) is hold.

However, in practice, if following the pure Newton direction (also known as the *affine-scaling direction*) in (22), we only can obtain a small step length, i.e., $\alpha << 1$, before the condition $(\lambda, s) > 0$ is violated. Hence, the convergence speed is rather slow. To overcome this issue, primal-dual interior-point methods bias the Newton direction toward the interior of the orthant $(\lambda, s) \geq 0$ and keep the components of (λ, s) away from the boundary of the orthant. Typically, the search direction is aimed to a point whose pairwise products $\lambda_i s_i$ are reduced to a lower average value. The *duality measure* $\mu = \frac{\lambda^T s}{l}$ is used to estimate the reduction speed of $\lambda^T s$. The Newton search direction is then modified toward a point for which $\lambda_i s_i = \sigma\mu$, where μ is the current duality measure and $\sigma \in [0, 1]$ is termed a *centering parameter*.

The values of σ and α are two important factors to the performance of the method. Different strategies of choosing these parameters lead to a wide range of primal-dual interior-point methods. Our method is based on the *predictor-corrector method* proposed by Mehrotra [13], which was proven

to be efficient in practice. Details of the algorithm are presented in Algorithm 1.

Algorithm 1 Predictor-Corrector Algorithm

Require: $E, F, M, \gamma, (x^0, \lambda^0 > 0, s^0 > 0)$, $k_{\max}$, ϵ_d, ϵ_p, ϵ
$(x, \lambda, s) \Leftarrow (x^0, \lambda^0, s^0)$
while $(k \leq k_{\max})\&(\|\psi_d\| \geq \epsilon_d)\&(\|\psi_p\| \geq \epsilon_p)\&(\mu \geq \epsilon)$ **do**
$\psi_d \Leftarrow Ex + F + M^T\lambda$
$\psi_p \Leftarrow s + Mx - \gamma$
$\mu \Leftarrow \frac{\lambda^T s}{l}$
Solve the following linear system for $(\Delta x^{aff}, \Delta\lambda^{aff}, \Delta s^{aff})$:

$$\begin{bmatrix} E & M^T & 0 \\ M & 0 & I \\ 0 & S & \Lambda \end{bmatrix} \begin{bmatrix} \Delta x^{aff} \\ \Delta\lambda^{aff} \\ \Delta s^{aff} \end{bmatrix} = \begin{bmatrix} -\psi_d \\ -\psi_p \\ -\Lambda Se \end{bmatrix}$$

$\alpha^{aff} \Leftarrow \min\left(\min_{i:\Delta\lambda_i<0}(1, \min\frac{-\lambda_i}{\Delta\lambda_i^{aff}}), \min_{i:\Delta s_i<0}(1, \min\frac{-s_i}{\Delta s_i^{aff}})\right)$
$\mu^{aff} \Leftarrow \frac{(\lambda+\alpha^{aff}\Delta\lambda^{aff})^T(s+\alpha^{aff}\Delta s^{aff})}{l}$
$\sigma \Leftarrow \left(\frac{\mu^{aff}}{\mu}\right)^3$
Solve the following linear system for $(\Delta x, \Delta\lambda, \Delta s)$:

$$\begin{bmatrix} E & M^T & 0 \\ M & 0 & I \\ 0 & S & \Lambda \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \end{bmatrix} = \begin{bmatrix} -\psi_d \\ -\psi_p \\ -\Lambda Se - \Delta\Lambda^{aff}\Delta S^{aff}e + \sigma\mu e \end{bmatrix}$$

$\alpha \Leftarrow \min\left(\min_{i:\Delta\lambda_i<0}(1, \min\frac{-\lambda_i}{\Delta\lambda_i}), \min_{i:\Delta s_i<0}(1, \min\frac{-s_i}{\Delta s_i})\right)$
$(x, \lambda, s) \Leftarrow (x, \lambda, s) + \alpha(\Delta x, \Delta\lambda, \Delta s)$
$k \Leftarrow k + 1$
end while

It should be noted that the two linear systems in Algorithm 1 have a same coefficient matrix. Thus, the matrix factorization needs to be computed only once, leading to a relatively low cost of solving the second system.

4. MODIFIED MINIMAL MODEL

There are many mathematical blood glucose models proposed in the literature [2]. Among them, the *minimal model* proposed by Bergman et al. (1981) [4] is the most popular model due to its simplicity and physiological accuracy. The blood glucose model in this work is based on the Fisher model [9], which is a modified version of the minimal model.

In addition to changes on the third equation of the minimal model made by Fisher (1991) in order to adapt the model to type I diabetic, that is removing the insulin secretion and adding an insulin infusion term, we add one more equation which is adapted from [12] to represent a relationship between plasma glucose concentration and subcutaneous glucose. This is due to in continuous glucose monitoring in practice, most glucose measurements are obtained via the subcutaneous layer.

The model equations used in our work are

$$\frac{dG}{dt} = -p_1(G(t) - G_b) - X(t)G(t) + P(t), \tag{24a}$$
$$\frac{dX}{dt} = -p_2X(t) + p_3(I(t) - I_b), \tag{24b}$$
$$\frac{dI}{dt} = -nI(t) + U(t)/V_I, \tag{24c}$$
$$\frac{dG_{sc}}{dt} = \frac{G(t) - G_{sc}(t)}{5} - R_{utln}, \tag{24d}$$

where the model parameters are described in Table 4.

The following discrete model is obtained by linearizing the model with the steady-state values of $(G, X, I, G_{sc}) = (G_b, 0, I_b, G_{bsc})$:

$$x_{k+1} = Ax_k + Bu_k + Dd_k, \tag{25a}$$
$$y_k = Cx_k, \tag{25b}$$

where x, u and y represent the state variable, input and output of the system, respectively; k is the discrete time-step.

With the model parameters adopted from [12]:

$p_1 = 0.028735$ $p_2 = 0.028344$ $p_3 = 5.035 \times 10^{-5}$
$G_b = 81.3$ $G_{bsc} = 77.6$ $I_b = 15$
$V_I = 12$ $n = 5/54$,

and the sampling time of 5 minutes, the parameter matrices in (25) are:

$$A = \begin{bmatrix} 0.8662 & -352.4422 & -0.0400 & 0 \\ 0 & 0.8679 & 0.0002 & 0 \\ 0 & 0 & 0.6294 & 0 \\ 0.5819 & -135.0405 & -0.0111 & 0.3679 \end{bmatrix},$$

$$B = \begin{bmatrix} -0.0059 \\ 0.0000 \\ 0.3335 \\ -0.0013 \end{bmatrix}, C = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}, D = \begin{bmatrix} 5 & 0 & 0 & 0 \end{bmatrix}^T.$$

The term $P(t)$ in (24) represents the rate of glucose entering the blood from intestinal absorption after a meal, and can be estimated based on the following meal model [10]:

$$U_G(t) = \frac{D_G A_G t e^{-t/t_{\max,G}}}{t^2_{\max,G}}, \tag{26}$$

where D_G (mg) is the amount of carbohydrates digested, A_G represents the carbohydrate bioavailability, and $t_{max,G}$ (min) is the time from the beginning of the meal consumption until the absorption rate reaches its maximum. A_G and $t_{max,G}$ are chosen as 0.8 and 40 minutes, respectively, as in [10]. Hence, the term d_k in (25) is determined as

$$d_k = \begin{cases} 0.00002D_G(k - k_G)e^{-0.125(k-k_G)} & \text{if } k \geq k_G, \\ 0 & \text{otherwise,} \end{cases} \tag{27}$$

where k_G is the time-step at which the meal disturbance is implemented.

The set-point is chosen as G_{bsc} $(77.6mg/dL)$ in our experiments. In addition, due to limitations on the range and possible changes of insulin infusion rate that can be enforced by a common insulin pump, and conditions of the output to avoid hypoglycemia and extreme hyperglycemia, the follow-

Table 1: Model Parameters

Parameter	Unit	Description
$G(t)$	mg/dL	Plasma glucose concentration
$G_{sc}(t)$	mg/dL	Subcutaneous glucose concentration
$X(t)$	min^{-1}	Insulin variable for the remote compartment
$I(t)$	mU/L	Plasma insulin concentration
G_b	mg/dL	Basal value of plasma glucose concentration
G_{bsc}	mg/dL	Basal value of subcutaneous glucose concentration
I_b	mU/L	Basal value of plasma insulin concentration
$P(t)$	$(mg/dL)min^{-1}$	Climbing rate of blood glucose due to a meal disturbance
$U(t)$	mU/min	Exogenous insulin infusion rate
V_I	L	Insulin distribution volume
p_1	min^{-1}	Insulin-independent constant rate of glucose uptake
p_2	min^{-1}	Rate of clearance of active insulin
p_3	$L/(min^2 mU)$	Increase in uptake ability caused by insulin
R_{utln}	$(mg/dL)min^{-1}$	Tissue rate of utilization

ing constraints are also imposed:

$$0 \le u \le 80 \ \ mU/min, \tag{28a}$$

$$-16.7 \le \Delta u \le 16.7 \ \ mU/min, \tag{28b}$$

$$60 \le y \le 180 \ \ mg/dL. \tag{28c}$$

Since measurements of X are physically unavailable, and those of I are inaccessible in practice, we need to estimate the state variable x using an observer. Hence, the optimization problem is listed in (20), where all the parameter matrices can be easily inferred from (7), (12), (25), (27), and (28).

5. MPC-ON-A-CHIP IMPLEMENTATION

We implemented the constrained MPC algorithm on a Texas Instruments (TI) MSP430FG4618 microcontroller. This is a ultralow-power 16-bit RISC-architecture microcontroller consisting of 116KB+256B Flash ROM, 8KB RAM, and five low-power modes, which is optimized to achieve extended battery life in portable measurement applications. It should be noted that our implementation can be applied to any generic RISC microcontroller. In addition, it is easily to port our code to other TI series or other platforms since it was developed using the standard C language.

Main functions. As seen from Sections 2 and 3, most of the calculations are based on matrix operations and solving systems of linear equations. We implemented these calculations as functions which are described in Table 5.

Dynamic memory-allocation. Since the dimensions of the matrices vary based on a specific application, the functions in Table 5 should use dynamic memory-allocation to store matrices and intermediate values. However, due to limited memory of microcontrollers, "dynamic heap memory allocation shall not be used" as indicated in [14] (rule 118). Hence, we should not use the built-in memory-allocation routines in C, e.g, `malloc`, `realloc`, `calloc` or `free`. Instead, we propose the use of a global array as a buffer shared between the main function and all other sub-routines to store dynamic variables and intermediate values. For any function which requires a dynamic memory-allocation, it will be provided by the caller function a part of the global buffer through an input pointer. The caller function needs to determine the address of the buffer allocated to the called function. The size of the global array is set based on the implementation of all the functions and the size of the physical SRAM memory of the microcontroller.

Listing 1: Dynamic Memory-Allocation Illustration

```
1  // garbage is shared by all functions in the program
2  double garbage[MAX_BUFFER_SIZE];
3
4  double QP(double *x, int m, int n, double *H, double *c,
5            double *A, double *b, double *x0, double *buffer)
6  {
7      double      *M = &buffer[0];
8      double      *N = &M[n*n];
9      double      *df = &N[n];
10     double      *Lx = &df[n];
11     double      *dxdlam = &Lx[n];
12     double      *h = &dxdlam[n];
13     double      *z = &h[m];
14     double      *dz = &z[m];
15     double      *mu = &dz[m];
16     double      *dmu = &mu[m];
17     double      *dh_zinv = &dmu[m];
18     double      *buf_tmp = &dh_zinv[m*n];
19
20     // ... other declarations and statements
21
22     // The following two statements calculate h = A*x - b
23     mult(A, x, m, n, 1, h);
24     sub(h, b, m, 1, h);
25
26     // ... other statements
27
28     // dxdlam = M \ (-N)
29     solve(M, n, n, N, dxdlam, -1, buf_tmp);
30
31     // ... other statements
32 }
33
34 void main(void)
35 {
36     // ... other statements
37     QP(du, N_ROWS, Nc, H, f, A_cons, b_cons, NULL, garbage);
38     // ... other statements
39 }
```

Listing 1 illustrates our dynamic memory-allocation mechanism. As in Table 5, `solve` and `QP` are two functions requiring buffers to store intermediate values. Dynamic variables in `QP` are allocated through the input address *buffer* and the size parameters m and n as from lines 7 to 18 in the listing. When `solve` is called by `QP`, a buffer is provided to `solve` through the array *buf_tmp* which has already been allocated in `QP` (line 29). When `QP` is called by the main function, its buffer is assigned as the global buffer *garbage* (line 37).

6. RESULTS AND DISCUSSION

A *Virtual Patient* using the modified minimal model was implemented on a computer to verify the accuracy of the embedded MPC algorithm. The Virtual Patient communicates with the MPC microcontroller through serial communica-

Table 2: Main Functions

Function prototype	Description	Parameters
void **zeros**(double *A, int n);	Set a square matrix to a zero matrix	A: [out] an $n \times n$ matrix to be set, n: [in] size of A
void **eye**(double *A, int n);	Set a square matrix to identity matrix	A: [out] an $n \times n$ matrix to be set, n: [in] size of A
void **copy**(double *src, double *dst, int size);	Copy elements between two matrices	src: [in] matrix to copy from, dst: [out] destination matrix, size: [in] number of elements to copy
void **transpose**(double *src, int m, int n, double *dst);	Compute matrix transpose	src: [in] an $m \times n$ input matrix, dst: [out] the $n \times m$ output matrix
void **add**(double *src1, double *src2, int m, int n, double *dst);	Add two matrices	src1, src2: [in] two input matrices, dst: [out] the sum matrix, $m \times n$: size of the three matrices
void **sub**(double *src1, double *src2, int m, int n, double *dst);	Subtract two matrices	src1: [in] minuend matrix, src2: [in] subtrahend matrix, dst: [out] the difference matrix, $m \times n$: size of the three matrices
void **mult**(double *src1, double *src2, int m, int n, int p, double *dst);	Multiply two matrices	src1: [in] an $m \times n$ input matrix, src2: [in] an $n \times p$ input matrix, dst: [out] the $m \times p$ product matrix
void **solve**(double *A, int m, int n, double *b, double *x, double thresh, double *buffer);	Solve a system of linear equations: $Ax = b$	A: [in] an $m \times n$ matrix, b: [in] a $m \times 1$ vector, x: [out] the $n \times 1$ solution vector, thresh: [in] a threshold for checking singular values, buffer: [in] an array (of $m \times (n+1) + n \times (n+2)$ elements) to store intermediate values
double **QP**(double *x, int m, int n, double *H, double *c, double *A, double *b, double *x0, double *buffer);	Solve a quadratic programming problem as minimizing $J = \frac{1}{2}x^T Hx + x^T c$ subject to $Ax \leq b$.	x: [out] the $n \times 1$ solution vector, H: [in] an $n \times n$ matrix, c: [in] an $n \times 1$ vector, A: [in] an $m \times n$ matrix, b: [in] an $m \times 1$ vector, x0: [in] an initial solution, buffer: [in] an array to store intermediate values

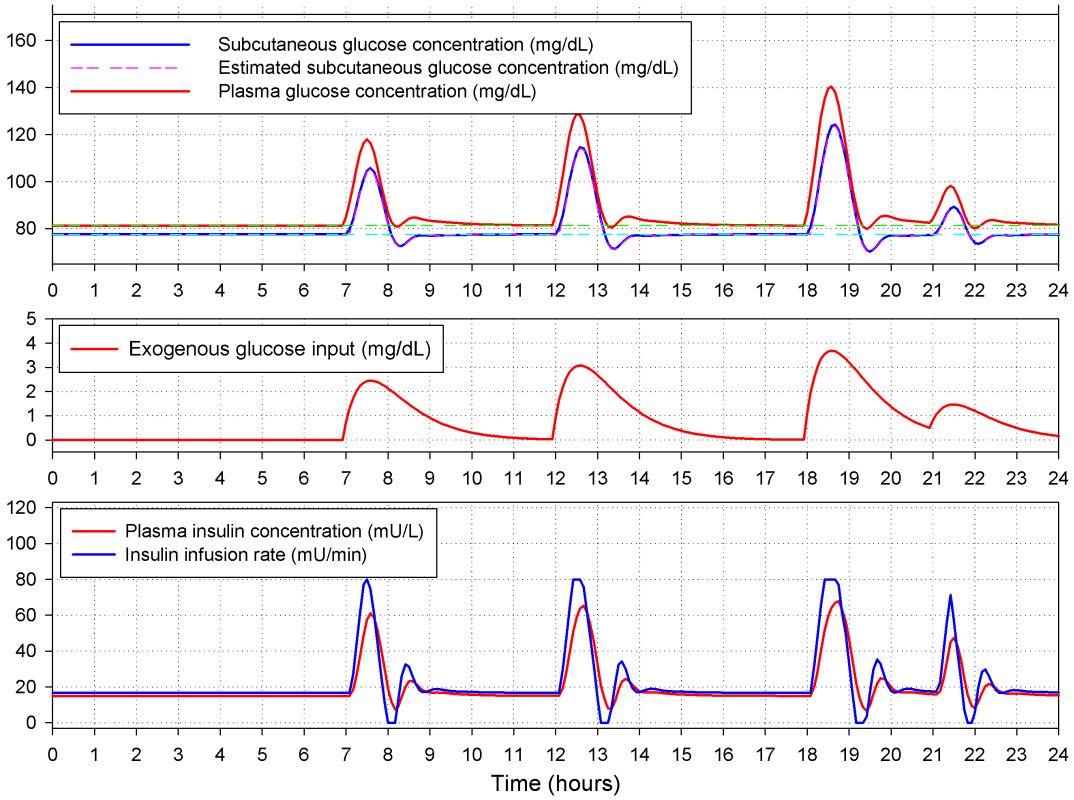

Figure 1: Simulation Results.

tions. Figure 1 shows the simulation results for a patient who has breakfast, lunch, dinner, and supper at 7:00, 12:00, 18:00, and 21:00 respectively. The corresponding meal sizes are $40g$, $50g$, $60g$, and $20g$ of carbohydrates, respectively. As seen in Figure 1, the patient blood glucose reaches the maximum level after the meal starts about 30–40 minutes. However, with a prediction window of 300 minutes ($N_P = 60$) and appropriate meal size, the MPC-based artificial pancreas is able to bring down the glucose level to the basal value 3–3.5 hours after the meal.

The binary code of the MPC and serial communications on the MSP430F5438A microcontroller is 24KB, which is smaller than that of other implementations [7, 6]. The time for computation and communications during each time-step was less than 1 second on average. With longer prediction and control windows ($N_P = 80$, $N_C = 4$), the computation speed of our system is comparable with that of the other designs [7, 6, 1]. Other advantages of our implementation compared to these designs are that (1) all typical operational constraints, and (2) the estimation of unmeasurable states are supported. In addition, this system is highly energy efficient since most of the time the processor is on standby mode with the power consumption rate of $2.1\mu A$ at $3.0V$. The power consumption rate in active mode is about $230\mu A/MHz$ at $8MHz, 3.0V$. The microcontroller takes less than $5\mu s$ to be active from standby mode.

7. CONCLUSION

We have proposed an implementation of MPC with constraints on a microcontroller, and applied this implementation on the development of a real-time implantable close-loop glucose control system. The experimental results show that our embedded MPC microcontroller has low power consumption and is capable of performing the computational efficient MPC algorithm accurately and in real-time.

8. REFERENCES

[1] A. K. Abbes, F. Bouani, and M. Ksouri. A microcontroller implementation of constrained model predictive control. *I. J. Electr. Electron. Eng.*, 5(3):199–206, 2006.

[2] N. P. Balakrishnan, G. P. Rangaiah, and L. Samavedham. Review and analysis of blood glucose (BG) models for type 1 diabetic patients. *Ind. Eng. Chem. Res*, 50(21):12041–12066, Sep 2011.

[3] K. Basterretxea and K. Benkrid. Embedded high-speed model predictive controller on a FPGA. In *Proc. AHS 2011*, pages 327–335, Jun 2011.

[4] R. N. Bergman, L. S. Phillips, and C. Cobelli. Physiologic evaluation of factors controlling glucose tolerance in man: measurement of insulin sensitivity and beta-cell glucose sensitivity from the response to intravenous glucose. *J. Clin. Invest*, 68(6):1456–1467, Dec 1981.

[5] L. G. Bleris, J. Garcia, M. V. Kothare, and M. G. Arnold. Towards embedded model predictive control for System-on-a-Chip applications. *J. Process. Contr.*, 16(3):255–264, Mar 2006.

[6] L. G. Bleris and M. V. Kothare. Implementation of model predictive control for glucose regulation on a general purpose microprocessor. In *Proc. CDC-ECC 2005*, pages 5162–5167, Dec 2005.

[7] L. G. Bleris and M. V. Kothare. Real-time implementation of model predictive control. In *Proc. ACC 2005*, volume 6, pages 4166–4171, Aug 2005.

[8] L. G. Bleris, P. D. Vouzis, M. G. Arnold, and M. V. Kothare. A co-processor FPGA platform for the implementation of real-time model predictive control. In *Proc. ACC 2006*, pages 1912–1917, Jun 2006.

[9] M. E. Fisher. A semiclosed-loop algorithm for the control of blood glucose levels in diabetics. *IEEE Trans. Biomed. Eng*, 38(1):57–61, Jan 1991.

[10] R. Hovorka, V. Canonico, L. J. Chassin, U. Haueter, W. Massi-Benedetti, M. O. Federici, T. R. Pieber, H. C. Schaller, L. Schaupp, T. Vering, and M. E. Wilinska. Nonlinear model predictive control of glucose concentration in subjects with type 1 diabetes. *Physiol. Meas.*, 25(4):905–920, Aug 2004.

[11] K. V. Ling, S. P. Yue, and J. M. Maciejowski. A FPGA implementation of model predictive control. In *Proc. ACC 2006*, pages 1930–1935, Jun 2006.

[12] S. M. Lynch and B. W. Bequette. Model predictive control of blood glucose in type I diabetics using subcutaneous glucose measurements. In *Proc. ACC 2002*, pages 4039–4043, May 2002.

[13] S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM J. Optim.*, 2(4):575–601, 1992.

[14] MISRA. *Guidelines for the use of the C language in critical systems*. MIRA Ltd., Oct 2004.

[15] L. Wang. *Model predictive control system design and implementation using MATLAB*. Advances in Industrial Control. Springer-Verlag London, 2009.

[16] S. J. Wright. *Primal-dual interior-point methods*. SIAM, 1997.