Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 50-1 (2017) 14412–14417

Programmable Logic Controller for Embedded Implementation of Input-Constrained Systems ⋆

Richard M. Levenson * **Zachary E. Nelson** ** **Ambrose A. Adegbege***

* *Department of Electrical and Computer Engineering, The College of New Jersey, Ewing, NJ 08628-0718, USA (e-mail: levensr1@tcnj.edu, adegbega@tcnj.edu).*

** *Department of Electrical and Computer Engineering, The Johns Hopkins University, Baltimore, MD 21218, USA (e-mail: znelson2@jhu.edu).*

Abstract: Programmable logic controllers (PLCs) have the benefit of being able to withstand a variety of environments while maintaining high reliability compared to computers and other controllers. Traditionally, PLCs have been used for simple ON-OFF control schemes or for proportional-integral-derivative (PID) control. However, recent developments in technology have allowed even low-cost PLCs to implement more advanced and efficient algorithms. This paper investigates a PLC implementation of the specialized Projected Gauss-Seidel (PGS) algorithm. The PGS algorithm is able to quickly and efficiently solve both symmetric and asymmetric mathematical programming problems. The algorithm can be further utilized to solve more complicated problems such as model predictive control (MPC). The focus of this paper is to solve a general constrained optimization problem on a PLC and to expand the capability to solve a MPC example. The DirectLogic Do-More PLC is exploited both for computational efficiency and for memory utilization.

Keywords: Projected Gauss-Seidel, Programmable Logic Controller, Input-Constrained Systems, Model Predictive Control

1. INTRODUCTION

Programmable Logic Controllers (PLC) are the de-facto industrial platform for implementing control algorithms. Due to their limited storage and computational capabilities, PLCs are often relegated for implementing simple ON-OFF control and proportional-integral-derivative (PID) controls. However, recent advances both in algorithms and in hardware for embedded control technology (e.g. Wang and Boyd (2008); Wills et al. (2010); Adegbege and Nelson (2016)) have allowed the deployment of PLCs for advanced control applications such as model predictive control (MPC) (Pereira et al., 2015).

In real-time applications, an approximate solution of the MPC or an associated optimization problem must be obtained relatively quickly during each PLC scan time. The computational burden and storage requirements for such problems can still be challenging and often times beyond the capabilities of many low-end PLCs. Recent works have favored high-level code generation for high-end PLCs which allows for incorporating C code as part of the PLC program (Kufoalor et al., 2014; Bonne et al., 2017). Such high-end PLCs are often referred to as programmable automation controllers (PACs) (Kufoalor et al., 2014). Code generation requires the handling of libraries for linear algebra which must also be embedded on the hardware.

In this work, we take a different approach by exploring the traditional IEC 61131-3 programming standard for implementing advanced control algorithms on low-cost, low-end PLCs. Specifically, we employ a combination of ladder logic and function blocks to implement a fast algorithm for application in MPC. The algorithm is based on a recently developed Projected Gauss-Seidel method (Adegbege and Nelson, 2016). We demonstrate that this algorithm is particularly suitable for PLC implementation since the mode of its execution is sequential.

2. PROBLEM SETUP

We consider the linear complementarity problem (LCP) of finding vectors $u \in \mathbb{R}^m$, $0 \leq w \in \mathbb{R}^m$ and $0 \leq v \in \mathbb{R}^m$ such that:

$$Hu + q = w - v, \tag{1a}$$
$$\underline{u} \leq u \leq \overline{u} \tag{1b}$$
$$(u - \underline{u})^T w = 0;\ w \geq 0, \tag{1c}$$
$$(\overline{u} - u)^T v = 0;\ v \geq 0, \tag{1d}$$

where $H \in \mathbb{R}^{m \times m}$ is a positive definite matrix (not necessarily symmetric), $q \in \mathbb{R}^m$ is an input vector, $u \in \mathbb{R}^m$(free) is the primal variable of particular interest, $w \in \mathbb{R}^m$ and

⋆ This work began while Zachary Nelson was a student of the Department of Electrical and Computer Engineering at The College of New Jersey.

2405-8963
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2017.08.2031

$v \in \mathbb{R}^m$ (restricted) are internal variables also known as multipliers. The vectors $\underline{u}$ and $\overline{u}$ are respectively the lower and the upper bounds on variable u. The LCP problem (1)

Algorithm 1 Projected Gauss-Seidel (PGS) Algorithm

1: Let $u^0 \in [\underline{u}, \overline{u}]$ ▷ Setting a starting point
2: $k = 0$
3: **do**
4: **for** $i = 1$ to m **do**
5: $\Delta u_i = \left([H_L(u^{k+1} - u^k)]_i + [Hu^k + q)]_i\right) / H_{ii}$
6: $u_i^{k+1} = \phi(u_i^k - \alpha \Delta u_i)$
7: **end for**
8: $k = k + 1$
9: **while** $|u_i^{k+1} - u_i^k| > tol$ & $k < maxItrs$

has widespread application in the solution of quadratic programs and variational inequalities (Robinson et al., 2013). In particular, when H is symmetric and positive definite, the LCP (1) describes exactly the Karush-Kuhn-Tucker (KKT) optimality conditions for the strictly convex quadratic program:

$$u^* = \arg\min_u \frac{1}{2} u^T H u + u^T q \tag{2a}$$

$$\text{subject to } \underline{u} \leq u \leq \overline{u}. \tag{2b}$$

It follows that if u^* is the unique solution for (2), then there exist non-negative vectors w^* and v^* such that (u^*, w^*, v^*) solves the LCP problem (1). There are very efficient iterative algorithms with light computational footprints and very attractive convergence properties for solving the LCP problem (1). In fact, there is a very large class of matrices H for which the LCP is guaranteed to be solvable with unique solutions. However, when H is asymmetric, the link between the LCP (1) and the QP (2) is broken as (1) ceases to be the KKT conditions for (2).

Quadratic programs with simple bounds as in (2) occur frequently in input-constrained control applications such as in anti-windup control and in model predictive control (Adegbege and Heath, 2015). The LCP formulation provides an attractive framework for addressing the computational aspects of these constrained systems. Recent interests have been on developing fast gradient-based algorithms with proven convergence bounds for embedded control applications. We discuss in section 3 one of such algorithms and in sections 4 and 5, we discuss how the algorithm can be efficiently implemented on a low-end, low-cost PLC using the IEC 61131-3 programming standard. Finally, we discuss in section 6 the computational and storage implications for model predictive control on a low-end, low-cost PLC.

3. PROJECTED GAUSS-SEIDEL ALGORITHM

Earlier methods for solving the LCP problem are finite-methods capable of attaining a solution in a finite number of arithmetic operations, often with pivots (Cottle et al., 2009). Such methods are known not to preserve sparsity and are usually employed for small-size problems. On the other hand, iterative methods are capable of preserving sparsity and are generally deployed for large-size problems especially where approximate solutions must be computed relatively fast. Due to these features, projected iterative methods such as Projected Gauss-Seidel (PGS) and Projected Successive Over Relaxation (PSOR) have become popular for solving large constrained optimization problems (Robinson et al., 2013; Morales et al., 2008). In Adegbege and Nelson (2016), we exploited the structure and the convergence properties of the PGS for applications in input-constrained systems. We develop here a version of the PGS algorithm optimized for PLC implementation.

Associated with problem (1) or (2) are the gradient equation

$$Hu + q = 0 \tag{3}$$

and the projection operation

$$\phi_i(u_i) = \begin{cases} \overline{u}_i & \text{if } u_i \geq \overline{u}_i, \\ u_i & \text{if } \underline{u}_i < u_i < \overline{u}_i, \\ \underline{u}_i & \text{if } u_i \leq \underline{u}_i. \end{cases} \tag{4}$$

By applying the matrix splitting

$$H = H_L + H_D + H_U \tag{5}$$

where H_L is the strictly lower triangular matrix part of H, H_D is the diagonal part of H, and H_U is the strictly upper triangular part of H to (3) and combining with (4) gives the PGS iterative method stated in Algorithm 1. Note that the PGS formulation does not require the symmetry of H, but the diagonal entries of H must be positive. This is always true since by assumption, H is positive definite. Hence the computation in step 5 of Algorithm 1 is well-defined. Also the matrix-vector multiplications in this step are carried out in sequential fashion due to the structure of the matrix splitting. This sequential manner of solution updates makes the PGS algorithm suitable for efficient PLC implementation where program execution also takes place sequentially. In addition, the projection in step 6 is easy to carry out and hence, the computational burden of the PGS is expected to be light for the category of problems in the form of (1) or (2). The algorithm will terminate when a certain tolerance is met or a maximum number of iterations is exceeded. Note that every iterate is a feasible solution, so the algorithm can be terminated prematurely.

4. PROGRAMMABLE LOGIC CONTROLLER

4.1 PLC Set-Up

We used the DirectLogic Do-More PLC for our implementation. A general configuration of this PLC is shown in Fig. 1. The base unit consists of the power wiring connections, CPU slot, and removable I/O modules. We used an 8-channel analog input module and 2-channel analog output module.

The set-up of the PLC starts with running a system task called $FirstScan. As the name implies, this task runs whenever the PLC goes from the STOP mode to RUN mode. A memory clear function named MEMCLEAR is then called to clear all the elements of memory to ensure that no data is left over from a previous program installed on the PLC. The next function in the $FirstScan task is to initialize data with the INIT function to input the problem parameters. The input parameters for the problem include:

(1) Hessian matrix H

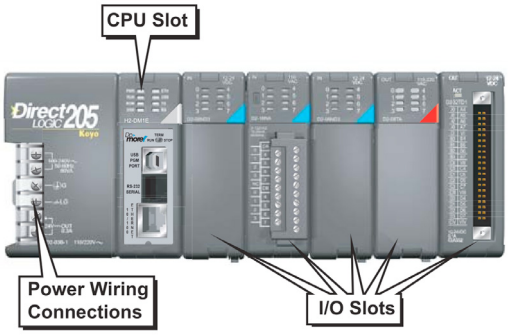

Fig. 1. General PLC Base Unit Configuration

(2) constraint limits $\underline{u}$ and $\overline{u}$
(3) relaxation parameter α
(4) maximum number of iterations
(5) stopping tolerance
(6) problem dimension m

Based on these input parameters, the task calculates how much memory it will need to use for the given problem size.

4.2 Memory Requirements

The DirectLogic Do-More PLC has up to 262.1 kilobytes of memory. The low amount of memory on the PLC is one of the major differences between a PLC and a PC and is why many PLCs can be purchased at a much lower cost than a PC. In order to better understand the memory requirements, Table 1 shows the important memory blocks that were used for this program.

Table 1. PLC Memory Blocks Used

Memory Block	Data Type	Size (Bytes)
R	Floating Point	250016
V	Unsigned Word	4
C	Bit	4
WX	Signed Word	496
WY	Signed Word	496
CT	Counter structure	24

A memory block is called in the ladder logic by writing the memory block followed by the location within the block. Table 2 shows the memory configuration for the PLC that we used.

It can be derived from Table 2 that the PLC is able to solve a quadratic program of form (2) with problem dimension up to $m = 248$. Fig. 2 shows the percentage of free and used memory versus the problem dimension.

5. LADDER LOGIC IMPLEMENTATION

The ladder logic for the DiretLogic Do-more Designer PLC was developed in the Do-more Designer 1.4.1 software environment. This software does not fully support the IEC 61131-1 programming standard, but does support ladder logic and function blocks. While there are high-end PLCs that support more powerful languages such as C and C++, many of the preexistent PLCs implemented in industrial controls systems do not have this ability. Additionally, many of the technicians that work with industrial control

Table 2. PLC Memory Configuration

Memory Locations	Data
R0-R661503	Hessian Matrix H
R61504-R61751	Lower Constraints $\underline{u}$
R61752-R61999	Upper Constraints $\overline{u}$
R62000-R62247	Temporary Values of v
R62248	Δv
R62249	$v + \Delta v$
R62250	Lower Constraint Index
R62251	Upper Constraint Index
R62252	Relaxation Parameter α
R62253	Maximum Number of Iterations Allowed
R62254-R62501	Previous value of v
R62502	Error
R62503	Tolerance
V0	Number of Elements
V1	Problem Dimension
C0	Saturation Indicator
C1	Repeat Indicator
CT0-CT2	Counters
WX0-WX247	Analog Input u
WY0-WY247	Analog Output v

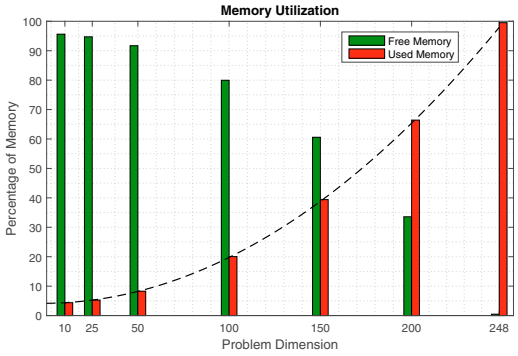

Fig. 2. PLC Memory Requirments for Solving PGS

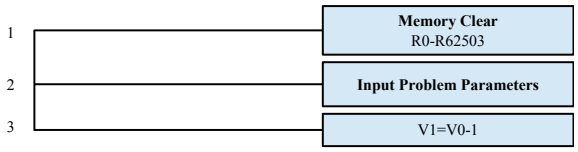

Fig. 3. First Scan Ladder Logic

systems are more comfortable with ladder logic than high-level programming. Therefore, the authors believe that using ladder logic to solve complex control problems offers a unique and practical look on the role of PLCs in control.

The ladder logic program for the PGS algorithm consists of three modules: the $FirstScan system task, the $Main system task, and a program called PGS. As previously discussed, the $FirstScan task is used to input all of the problem parameters. The $Main system task is the ladder logic that gets executed during every scan. Lastly, the program PGS is called by $Main to run the PGS algorithm. A description of $Main and the PGS program will now be given.

There is only one rung of ladder logic in $Main and it is only executed when a change is detected in one of

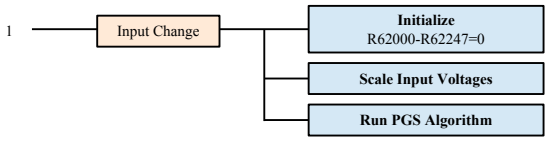

Fig. 4. Main Scan Ladder Logic

the analog inputs WX0-WX247. There are three types of function blocks used on this rung:

(1) INIT: sets the discrete solution v as 0
(2) SCALE: scales the analog inputs into discrete values
(3) RUN: calls the PGS algorithm

The PGS program starts off by setting the iteration counter (CT0) to 1 and saturation indicator (C0) to 0. The program then uses a function called WHILE to loop through the code until the stopping condition is met. We let this loop run until a maximum number of iterations is reached or until a desired tolerance condition is met. Note that including the tolerance stopping condition requires more memory and thus reduces the maximum problem size that can be solved on the PLC. The next step was to calculate the next iteration value v^{k+1} by using two FOR loops. Multiple loops were needed because the PLC is not capable of performing vector multiplication and instead must performed the operation element-by-element. Within the outer FOR loop, the ladder logic projects the solution onto the saturation region and turns on the saturation bit if saturation has occurred. The FOR loop function stops looping when the NEXT function is executed. The last step of the PGS program is to use the SCALE function to convert the discrete output values into analog ones. The program then exits and waits for another change in an analog input value. Details of the ladder logic implementation of the PGS algorithm are shown in Fig. 5.

6. COMPUTATIONAL EXAMPLES

6.1 Linear Complementarity Problems

In order to demonstrate the trade-off between number of iterations, speed, and accuracy, we solve several problems on the PLC and in Matlab using the PGS algorithm. The following tridiagonal matrix taken from literature was used for the symmetric feedback gain testing (Byong-Hun, 1983):

$$H = \begin{bmatrix} 6 & -1 & & & \\ -1 & 6 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & -1 & 6 & -1 & \\ & & -1 & 6 & \end{bmatrix}. \tag{6}$$

The following tridiagonal matrix was used for the asymmetric feedback gain testing:

$$H = \begin{bmatrix} 4 & -2 & & & \\ 1 & 4 & -2 & & \\ & \ddots & \ddots & \ddots & \\ & 1 & 4 & -2 & \\ & & 1 & 4 & \end{bmatrix}. \tag{7}$$

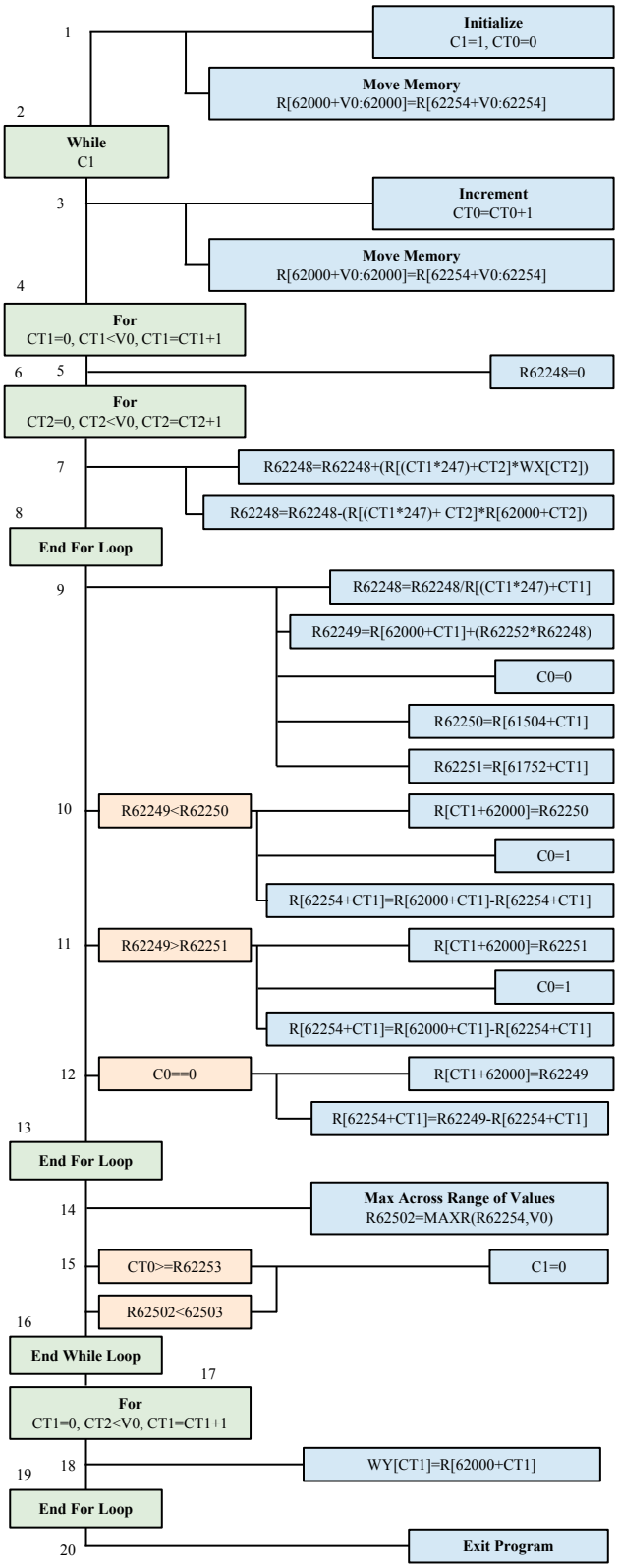

Fig. 5. PGS Algorithm Ladder Logic

Both of the resulting problems using matrices (6) and (7) in (1) with $\underline{u} = -1$ and $\overline{u} = 1$ were successfully solved using the PGS algorithm in both Matlab and ladder logic. The results were additionally verified using the Matlab quadprog function for the symmetric case and a Matlab implementation of the Lemke algorithm for the asymmetric case. For the PLC hardware implementation, the problem dimension for each case was limited to four in order to keep the number of power source inputs feasible in the laboratory environment. The number of PGS iterations required to reach a solution with error tolerance 10^{-5} on the PLC hardware for each of the cases is noted in Table 3. The input vectors, $q^{(m)}$, for this example were set as $q^{(2)} = [1.2\ 0.5]^T$, $q^{(3)} = [1.2\ 0.5\ 1.2]^T$, and $q^{(4)} = [1.2\ 0.5\ 1.2\ 0.5]^T$. As expected, the asymmetric case required more iterations to reach an optimal solution than the symmetric case for similar problem dimensions. The

Table 3. Number of Iterations for Symmetric vs. Asymmetric Cases

m	Symmetric Iterations	Asymmetric Iterations	Output u^*
2	2	3	$[0.999\ 0.433\ 0.999\ 0.466]$
3	3	4	$[0.999\ 0.433\ 0.999]$
4	3	4	$[0.999\ 0.466]$

results confirmed that the PGS algorithm implemented on the PLC is capable of solving constrained control problems for cases when H is either symmetric or asymmetric, so long as H is positive definite.

6.2 Application to Model Predictive Control (MPC)

We consider linear MPC with quadratic cost function and linear constraints of the form:

$$\min_{U,x_t} \left\{ \frac{1}{2}\left[\sum_{k=0}^{N-1} x_{t+k}^T Q x_{t+k} + \sum_{k=0}^{N_u-1} u_{t+k}^T R u_{t+k}\right] + \frac{1}{2} x_{t+N}^T P x_{k+N} \right. \tag{8a}$$

$$\text{subject to } x_{t+k+1} = Ax_{t+k} + Bu_{t+k}, k = 0 \cdots N-1 \tag{8b}$$

$$u_{min} \le u_{t+k} \le u_{max}, k = 0 \cdots N_u - 1 \tag{8c}$$

where $x_t \in \mathbb{R}^n$ and $u_t \in \mathbb{R}^m$ are the state and the input respectively. The weighting matrices are such that $Q = Q^T \ge 0$, $R = R^T > 0$ and P is the solution of the discrete algebraic Riccati equation

$$P = A^T PA - A^T PB(B^T PB - R)^{-1} B^T PA + Q. \tag{9}$$

The prediction and control horizons denoted as N and N_u respectively with $N \ge N_u$. We assume that the state x_t is accessible (or can be observed) for control.

By condensing, the linear MPC problem (8) can be expressed as the following quadratic program (e.g. see Bemporad et al. (2002); Adegbege and Nelson (2016))

$$U^* = \arg\min_U \frac{1}{2} U^T MU - U^T MFx_t \tag{10a}$$

$$\text{subject to } \underline{U} \le U \le \overline{U} \tag{10b}$$

where $U = \left[u_t^T \cdots u_{t+N_u-1}^T\right]^T$ and the implemented control law at current instant is obtained as

$$u_t^* = [I\ 0 \cdots 0]\, U^* = EU^*. \tag{11}$$

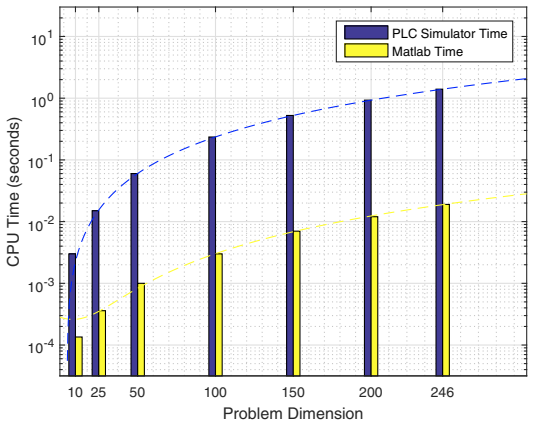

Fig. 6. MPC Computation Time

In this case, we consider a system with

$$A = \begin{bmatrix} 0.7326 & -0.0861 \\ 0.1722 & 0.9909 \end{bmatrix}, B = \begin{bmatrix} 0.0609 \\ 0.0064 \end{bmatrix}, C = [0\ 1.4142].$$

The input is constrained as $-1 \le u_{t+k} \le 1$, $k = 0 \ldots N_u$. The above problem can be reformulated into the form of (10). Following Bemporad et al. (2002), we set the control parameters as follows: $Q = I$, $R = 0.01$ and an initial condition of $x_t = [1\ \ 1]^T$. The prediction and the control horizons were set equal to each other for all implementations. In order to use the PLC to solve this MPC problem, the PLC memory configuration needed to be modified to allow storage for the additional matrix F and the solution of the multiplication of that matrix with the state x_t. The PGS Algorithm ladder logic of Fig. 5 also needed to be appended with an additional program to compute this matrix multiplication prior to running the rest of the algorithm. With the resulting memory configurations, the maximum calculated MPC horizon and thus problem dimension was determined to be 246.

The matrices M and F were pre-calculated in Matlab from the problem data and were loaded on to the PLC via the Do-more Designer software. The problem (10) was solved using the PGS algorithm in the Do-More Designer PLC Simulator, on Matlab running on a 3.3 GHz processor, and on the DirectLogic Do-More H2-DM1 PLC hardware. A comparison of computation times for the Do-More Designer PLC Simulator and Matlab PGS implementations is shown in Fig. 6. The H2-DM1 hardware was unable to process the computation for the MPC problem with a horizon greater than 25 within its maximum scan time, but the Do-More Designer PLC Simulator, limited to the same amount of memory as the hardware, was able to show that a horizon of 246 is possible within the 262.1 kilobyte memory limitation.

A DirectLogic Do-More H2-DM1 PLC hardware implementation setup is shown in Fig. 7. Here, a two-channel DC power supply was used to supply the input voltage corresponding to the initial state of the MPC problem. The computation time of the PLC is measured separately through an external computer running the the Do-More Designer software. The Do-More H2-DM1 PLC extends its scan-time to allow the entire ladder logic program to complete in one scan. Thus, the computation time was attained by viewing the scan-time of the PLC as it solved

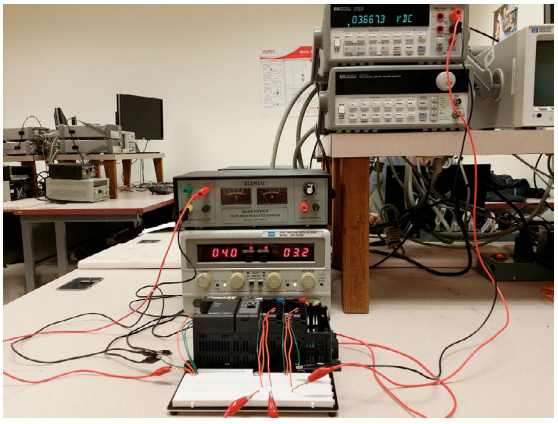

Fig. 7. PLC Hardware Testing Setup

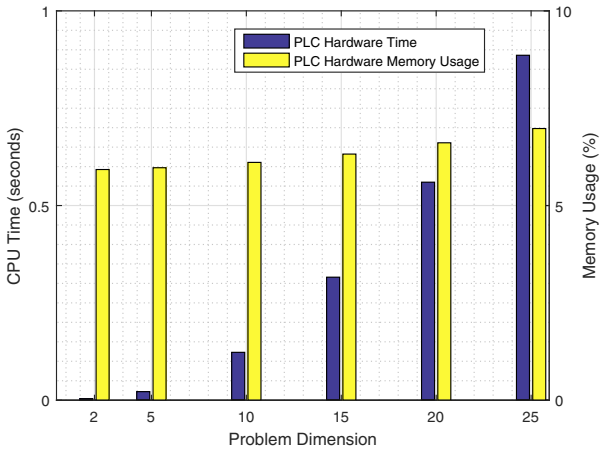

Fig. 8. PLC Hardware Computation Time and Memory Usage

the MPC problem at different problem dimensions. Results demonstrating the computation time and memory usage for problem dimensions between 2 and 25 are shown in Fig. 8. This example demonstrates that PLCs have the capacity to solve model predictive control problems.

7. CONCLUSION

This paper has demonstrated that a low-cost PLC with limited computational capacity has the ability to solve complex control problems. Despite the limited amount of memory, the PGS algorithm was efficiently implemented in ladder logic and executed in simulation for MPC problems with problem dimensions up to 246. Running on the H2-DM1 PLC hardware, the PGS algorithm was able to solve MPC problems with problem dimensions up to 25 in less than one second. The ladder logic implementation of this algorithm is especially noteworthy because the standard is still used in many industrial control systems. These industrial controllers are often used for controlling slow systems where a time constant on the order of one second would be feasible. Further work includes using a PLC to control a quadruple coupled tank system and making modifications to improve upon the convergence rate of the PGS algorithm. An improved convergence rate may be possible by incorporating adaptive step length computation (Moré and Toraldo, 1989) or by incorporating subspace optimization (Robinson et al., 2013). The authors also have plans to implement the algorithm on other forms of hardware, such as field-programmable gate arrays, to make it feasible for use in faster systems.

ACKNOWLEDGEMENTS

The authors would like to thank Automation Direct for providing the Do-More PLC for this research.

REFERENCES

Adegbege, A.A. and Nelson, Z.E. (2016). A gauss–seidel type solver for the fast computation of input-constrained control systems. *Systems & Control Letters*, 97, 132–138.

Adegbege, A.A. and Heath, W.P. (2015). Multivariable algebraic loops in linear anti-windup implementations. In *Proc. 23rd Mediterranean Conference on Control and Automation*, 514–519. Torremolinos, Spain.

Bemporad, A., Morari, M., Dua, V., and Pistikopoulos, E.N. (2002). The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1), 3 – 20.

Bonne, F., Alamir, M., and Bonnay, P. (2017). Experimental investigation of control updating period monitoring in industrial plc-based fast mpc: Application to the constrained control of a cryogenic refrigerator. *Journal of Control Theory and Technology*.

Byong-Hun, A. (1983). Iterative methods for linear complementarity problems with upperbounds on primary variables. *Mathematical Programming*, 26(3), 295–315.

Cottle, R.W., Pang, J., and Stone, R.E. (2009). *The Linear Complementarity Problem*. Society for Industrial and Applied Mathematics, Philadelphia, USA.

Kufoalor, D., Richter, S., Imsland, L., Johansen, T., Morari, M., and Eikrem, G. (2014). Embedded model predictive control on a plc using a primal-dual first-order method for a subsea separation process. In *Control and Automation (MED), 2014 22nd Mediterranean Conference of*, 368–373.

Morales, J.L., Nocedal, J., and Smelyanskiy, M. (2008). An algorithm for the fast solution of symmetric linear complementarity problems. *Numerische Mathematik*, 111(2), 251–266.

Moré, J.J. and Toraldo, G. (1989). Algorithms for bound constrained quadratic programming problems. *Numerische Mathematik*, 55(4), 377–400.

Pereira, M., Limon, D., de la Peña, D.M., and Alamo, T. (2015). Mpc implementation in a plc based on nesterov's fast gradient method. In *Control and Automation (MED), 2015 23th Mediterranean Conference on*, 371–376.

Robinson, D.P., Feng, L., Nocedal, J.M., and Pang, J. (2013). Subspace accelerated matrix splitting algorithms for asymmetric and symmetric linear complementarity problems. *SIAM Journal of Optimization*, 23(3), 1371–1397.

Wang, Y. and Boyd, S. (2008). Fast model predictive control using online optimization. In *17th IFAC World Congress*, 6974–6979. Seoul.

Wills, A., Knagge, G., and Ninness, B. (2010). Fast linear model predictive control via custom integrated circuit architecture. *IEEE Transactions on Control Systems Technology*.