

Dependability of the Explicit DMC Algorithm for a Rectification Process

Piotr Gawkowski¹, Maciej Ławryńczuk², Piotr Marusak², Janusz Sosnowski¹,
and Piotr Tatjewski²

¹ Institute of Computer Science

{P.Gawkowski, J. Sosnowski}@ii.pw.edu.pl

² Institute of Control and Computation Engineering

{M.Lawrynczuk, P.Marusak, P.Tatjewski}@ia.pw.edu.pl

Warsaw University of Technology, ul. Nowowiejska 15/19, 00-665 Warsaw, Poland

Abstract. The paper studies dependability of software implementation of the explicit DMC (Dynamic Matrix Control) Model Predictive Control (MPC) algorithm applied for a rectification column. The process with two inputs and two outputs with strong cross-couplings and significant time delays is studied. The algorithm's control law is calculated off-line. Dependability is evaluated experimentally using software implemented fault injection approach. The injected faults influence the quality of rectification process.

1 Introduction

Faults appearing during system operation may be critical for implemented applications. They can result in logical errors and application failure [1, 8]. It is particularly critical in many reactive systems (e.g. nuclear plants, satellites, aircrafts, chemical industry, medicine). Hence, an important practical issue is to evaluate dependability of software applications in the presence of faults. This paper studies the dependability of software implementation of the explicit version of Dynamic Matrix Control (DMC) Model Predictive Control (MPC) algorithm applied to a rectification column [21]. In the research the software implemented fault injector adapted to reactive applications is used [3].

Model Predictive Control is practically the only advanced control technique which have found acceptance in the process industry and is successfully applied practice [10-12, 15, 19, 20]. In the MPC algorithms the control action is calculated using a model of the process. If the model is accurate enough, performance offered by MPC algorithms can be better than the one offered by classical control algorithms, especially for processes with difficult dynamics, e.g. with significant time delay. Moreover, thanks to using the process model, the MPC algorithms contain the decoupling mechanism inside. It is very important for Multi-Input Multi-Output (MIMO) processes with strong cross-couplings. Among different MPC techniques, DMC is often applied in practice because it uses a step-response model of the process which is very easy to obtain in industry [2, 19].

The paper is structured as follows. In the next Section the explicit DMC control algorithm is described in a version for MIMO control processes. In Section 3 fault injection test-bed, experiment set-up and some new aspects of experiment conduction

according to control algorithms specificity are presented. In Section 4 results of conducted experiments are discussed. The last section summarises the paper.

2 DMC Algorithm for MIMO Processes

The distinctive feature of the MPC algorithms is that the on-line calculation of control policy considers the future behaviour of the control process many time-steps ahead. The prediction is made using a dynamic model of the control process. Typically, the future control values are chosen in such a way that the predicted behaviour of the control algorithm minimises a performance function formulated usually as follows:

$$J = \sum_{j=1}^{n_y} \sum_{i=1}^N \psi^j \cdot (\bar{y}_k^j - y_{k+i|k}^j)^2 + \sum_{j=1}^{n_i} \sum_{i=0}^{N_u-1} \lambda^j \cdot (\Delta u_{k+i|k}^j)^2, \quad (1)$$

where \bar{y}_k^j is a set-point value for the j^{th} output, $y_{k+i|k}^j$ is an output value for the $(k+i)^{\text{th}}$ sampling instant predicted at k^{th} sampling instant (the way of its calculation depends on a model used for prediction), $\Delta u_{k+i|k}^j$ are future changes in the manipulated variables, $\psi^j \geq 0$ and $\lambda^j \geq 0$ are weighting coefficients for the predicted control errors of the j^{th} output and for the changes of the j^{th} manipulated variable, respectively, N and N_u denote prediction and control horizons, n_y , n_i denote the number of outputs and inputs, respectively.

The performance function (1) can be rewritten in the following way:

$$J = (\bar{\mathbf{y}} - \mathbf{y})^T \cdot \boldsymbol{\Psi} \cdot (\bar{\mathbf{y}} - \mathbf{y}) + \Delta \mathbf{u}^T \cdot \mathbf{A} \cdot \Delta \mathbf{u}, \quad (2)$$

where $\boldsymbol{\Psi}$ is a weighting matrix of dimensionality $n_y N \times n_y N$, \mathbf{A} is a weighting matrix of dimensionality $n_i N_u \times n_i N_u$, $\Delta \mathbf{u}$ is a vector of dimensionality $n_i N_u$ composed of the future increments of control values, $\bar{\mathbf{y}}$ is a vector of dimensionality $n_y N$ composed of set-point values \bar{y}_k^j , \mathbf{y} is a vector of dimensionality $n_y N$ composed of output values $y_{k+i|k}^j$ predicted using a control process model. If the prediction is performed using a linear process model then the superposition principle can be used and the vector \mathbf{y} can be decomposed as:

$$\mathbf{y} = \mathbf{y}^0 + \mathbf{G} \cdot \Delta \mathbf{u}, \quad (3)$$

where \mathbf{G} is a matrix of dimensionality $n_y N \times n_i N_u$ called a dynamic matrix composed of the elements of the process step response, \mathbf{y}^0 is a vector of dimensionality $n_y N$ called a free response because it contains elements equal to the values of the process outputs in the future in the situation if manipulated signals are frozen at the k^{th} sampling instant [11, 15, 19].

If the performance function (2) is minimised without constraints, the optimal unique solution is:

$$\Delta \mathbf{u} = \mathbf{K} \cdot (\bar{\mathbf{y}} - \mathbf{y}^0), \quad (4)$$

where

$$\mathbf{K} = (\mathbf{G}^T \cdot \boldsymbol{\Psi} \cdot \mathbf{G} + \mathbf{A})^{-1} \mathbf{G}^T \cdot \boldsymbol{\Psi} \cdot \quad (5)$$

Only the elements $\Delta u_{k|k}^j$ of the vector $\Delta \mathbf{u}$ are applied to the process and then the procedure is repeated in the next sampling instant. The step-response model of the controlled process used by the DMC algorithm is following:

$$y_k^j = \sum_{m=1}^{n_i} \sum_{i=1}^{D-1} s_i^{j,m} \cdot \Delta u_{k-i}^m + s_D^{j,m} \cdot u_{k-D}^m, \quad (6)$$

where Δu_k^m is a change in the m^{th} manipulated variable at the k^{th} sampling instant, $s_i^{j,m}$ ($i=1, \dots, D$) are step response coefficients of the controlled process describing influence of the m^{th} input on the j^{th} output, D is equal to the number of time instants after which the coefficients of the step responses can be assumed as settled, u_{k-D}^m is a value of the m^{th} manipulated variable at the $(k-D)^{\text{th}}$ sampling instant.

The DMC control law (4) can be formulated:

$$\begin{bmatrix} \Delta u_{k|k}^1 \\ \Delta u_{k|k}^2 \\ \vdots \\ \Delta u_{k|k}^{n_i} \end{bmatrix} = \mathbf{K}^e \begin{bmatrix} \bar{y}_k^1 - y_k^1 \\ \bar{y}_k^2 - y_k^2 \\ \vdots \\ \bar{y}_k^{n_y} - y_k^{n_y} \end{bmatrix} - \sum_{j=1}^{D-1} \mathbf{K}_j^u \begin{bmatrix} \Delta u_{k-j}^1 \\ \Delta u_{k-j}^2 \\ \vdots \\ \Delta u_{k-j}^{n_i} \end{bmatrix}, \quad (7)$$

where \mathbf{K}^e is a matrix of dimensionality $n_i \times n_y$ and \mathbf{K}_j^u , $j=1, \dots, D-1$ are matrices of dimensionality $n_i \times n_i$ composed of controller coefficients. Because these matrices and the resulting control law are calculated off-line, the discussed MPC algorithm is named the explicit one. The detailed description of the explicit DMC algorithm derivation can be found in [14, 19].

3 Experiment Set-up

The concept of the SoftWare Implemented Fault Injector (SWIFI) is based on the software emulation of a fault during the run-time of the application under test. In this research FITS fault injector is used [4, 17]. It uses standard Win32 *Debugging API* to control the execution of the software application under tests. In the following the process controlled by the analysed control algorithm (DMC) is described, instrumentation (facilitating further analysis) of the tested application is introduced, then the fault insertion policy, and finally, applied result qualification is given.

MIMO Process Description. The process is a rectification column with two manipulated (inputs) and two controlled (outputs) variables shown in Fig. 1. It is described by the continuous-time transfer function model [14, 21] (time constants in minutes):

$$\begin{bmatrix} Y^1(s) \\ Y^2(s) \end{bmatrix} = \begin{bmatrix} \frac{12,8}{16,7s+1} & \frac{-18,9e^{-4s}}{21,0s+1} \\ \frac{6,6e^{-8s}}{10,9s+1} & \frac{-19,4e^{-4s}}{14,4s+1} \end{bmatrix} \cdot \begin{bmatrix} U^1(s) \\ U^2(s) \end{bmatrix} + \begin{bmatrix} \frac{3,8e^{-4s}}{14,9s+1} \\ \frac{4,9}{13,2s+1} \end{bmatrix} \cdot U^3(s), \quad (8)$$

where the controlled variables are: y^1 – methanol concentration in the distillate (the top product), y^2 – methanol concentration in the effluent (the bottom product), the manipulated variables are: u^1 – flow rate of the reflux, u^2 – flow rate of the steam into a boiler, u^3 is feed flow rate (a disturbance). All process variables are scaled.

For the considered rectification process the explicit DMC algorithm is designed: the sampling period $T_p=1$ min is assumed, the dynamics horizon is equal to the prediction horizon $D=N=100$, the control horizon $N_c=50$, the values of weighting coefficients are: $\psi^1=\psi^2=1$, $\lambda^1=\lambda^2=10$. The simulation horizon is 300 discrete time-steps. Structure of the control system with the explicit DMC algorithm is shown in Fig. 2.

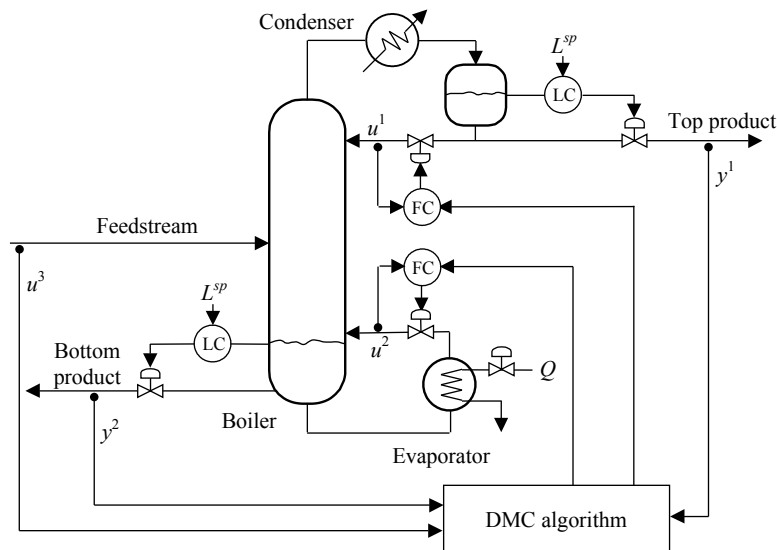


Fig. 1. Rectification column control system structure

Code instrumentation. FITS disturbs directly the tested application only within so-called testing areas [3]. Testing areas limit the scope of disturbances only to the selected parts of the application. Here, the code of the controlled process model is added. The parts of the tested application disturbed during the experiments (dashed box) as well as process models (not disturbed) are marked in Fig. 2. The tested

application is also instrumented to send some measures (e.g. related to internal variables values, output signal deviations) to the fault injector using user-defined messages (collected by FITS) [3].

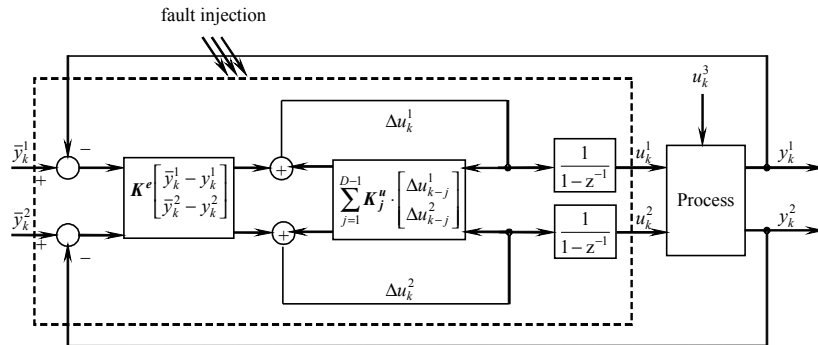


Fig. 2. Structure of the control system with the explicit DMC controller

Fault injection policy. FITS simulates faults by disturbing the running application. In this study the single bit-flip faults within CPU and FPU registers, application's data and machine instruction code are considered. Faults are injected pseudorandomly in time (program execution) and space (bit position within disturbed resource, distribution over application's memory). Such a fault model well mimics Single Event Upset (SEU) effects [7, 8, 16].

Qualification of experimental results. The correctness qualification of results produced by the application under test is more complicated in case of applications related to control systems [16] than in case of simple calculation-oriented ones. Control algorithms require complex analysis of the controlled process behaviour. The standard factor ISE (Integrated Sum of Squared Errors) is used as a measure of result (y^1, y^2) correctness. The reference ISE value (obtained during referential execution – non-faulty) is 2.53 (due to delayed response and feed stream disturbances). The whole experiment is conducted by FITS automatically. At the end of the experiment synthetic (aggregated) results for each fault location are given. In general, 4 classes of test results are distinguished:

- C: correct behaviour ($ISE < 10$),
- INC: incorrect (unacceptable) behaviour ($ISE \geq 10$),
- S: test terminated by the system due to un-handled exception,
- T: timed-out test.

Analysis of fault effects requires detailed information upon the faults injected and the application behaviour. FITS provides details about every test (simulated fault injection). Hence, manual replay of the whole test execution can be done. Moreover, all the events and user messages occurring during the test are recorded. The tested application is instrumented to save its outputs (here simulation results, i.e. a set of control signals in subsequent sampling instants) into separate files for each test (file names are managed by FITS). This gives a possibility for post-experiment analysis of fault effects in the correlation with the injected fault and observed behaviour for each test.

4 Simulation Results

Implementation of the control algorithm takes 124 machine instructions (405 bytes of the static code). The algorithm needs execution of 1020000 instructions for the whole simulation horizon (300 discrete sampling instants). As the static and dynamic profile is different, the distribution of mnemonics in the static code as well as in the executed stream (dynamic) is presented in Fig. 3.

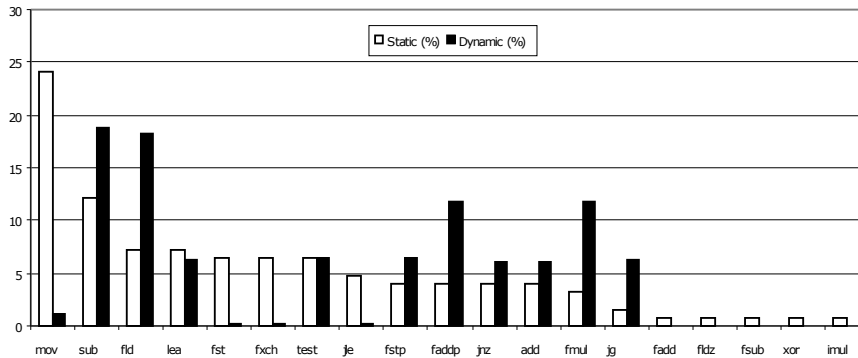


Fig. 3. Distribution of DMC mnemonics in static and dynamic profile

It is worth noting that floating point instructions contains 38% of the code in the static code while dynamically they are executed in 54,8% of time. Moreover, instructions organising computational loops in the DMC implementation (*sub*, *test*, *add*, *jnz*, *jg*) take another 38%. Hence, the application is strongly computational with high degree of FPU utilisation. Nevertheless, the instruction set used is rather limited. On the other hand, the activity ratio for CPU resources is high (98, 94, 80, 98, 97, 81 % for EAX, EBX, ECX, EDX, ESI and EDI, respectively) [16].

Faults in the CPU and FPU registers, data area of the application, executed instruction stream, and static code image are considered. For each fault location approximately 1000 disturbed executions are investigated (single fault injected in each application execution). The summary of results (according to categories described in Section 3) is presented in Fig. 4. As the algorithm uses many parameters (400) it is very robust to fault located in the data area – there are only few data memory locations critical for the algorithm – most of the data correspond to the algorithm's parameters. The high degree of FPU robustness could be astonishing. Past experience shows that the FPU is rarely used hard [8] (e.g. only few FPU stack locations used simultaneously). This results in overall low fault sensitivity of the FPU. Nevertheless, there are some very sensitive locations within the FPU (e.g. control registers).

The most fault sensitive resource of the DMC controller is its code. Fortunately, there are software techniques (e.g. exception handling, duplication of critical data and code) that can be applied at the source code level to provide fault robustness [5-7,9]. They have already proved their advantages; nevertheless, their effectiveness in the considered algorithm is going to be investigated in the further research. One can expect, that in the DMC case some simplifications in the fault hardening implementa-

tions can be applied without noticeable aggravation of dependability and performance (i.e. rare errors on the controlled process inputs are not critical for its behaviour).

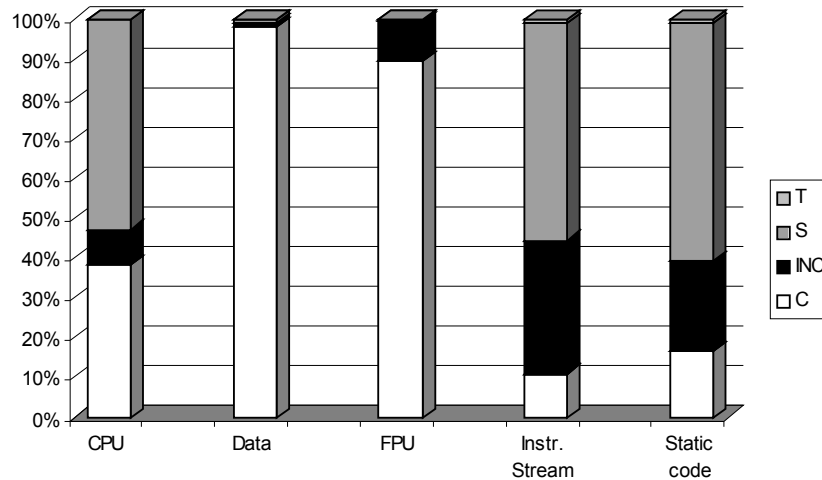


Fig. 4. Experimental results

Analysing fault susceptibility it is worth correlating the observed effects (simulated process behaviour) with the injected fault details in accordance to the source and machine code of the disturbed DMC. Fig. 5 and 6 present plots of the application outputs (y^1, y^2 – left plots and u^1, u^2 – right plots) over the DMC iteration number in case of sample fault disturbed executions. For reference the undisturbed simulation results are shown, i.e. the golden run (solid lines). The simulation scenario is as follows (solid lines). At the beginning, the process is driven to a given set-point. Then, at sampling instant 30 the change in the feed stream flow rate (u^3) is introduced (from 0 to 0.1). Another change in u^3 is made at the instant 140 (from 0.1 to -0.05).

In the case considered in Fig 5, the fault is injected at the sampling instant 28. It results in the change of the *faddp* instruction into the *fmlp* (operands remained unchanged). The instruction disturbed is used to calculate the *dul* variable of the DMC algorithm (corresponding to the Δu_k^1 in Fig. 2). As a result the source code statement `dul+=r1[i]*vektup[i]` is changed to `dul*=r1[i]*vektup[i]`. The result of this disturbance varies on the control signal and process states. For instance, the considered fault injected at the 28th sampling instant results in ISE=3.39 (Fig. 5a), at the 101st sampling instant in ISE=4.25 (Fig. 5b), at the 224th – ISE=2.53 (Fig. 5c) (the same as the reference ISE value). Hence, it disturbs the top product composition.

More critical situation is illustrated in Fig. 6. Single bit inversion (at 61st sampling instant) within the same instruction as described above destabilises the process. In this case the instruction mnemonic remain unchanged while its first operand changed from *st(2)* to *st(0)*. Observed ISE is 4.40.

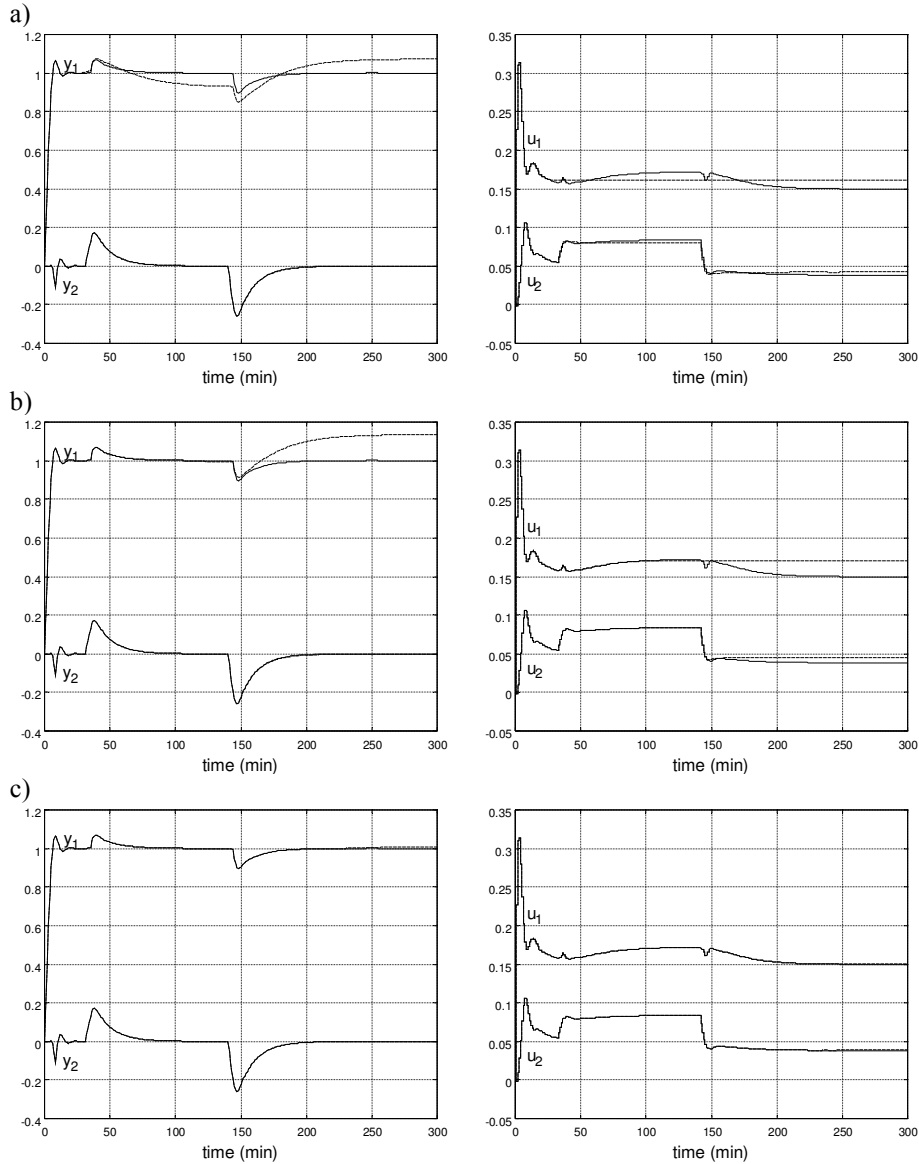


Fig. 5. Single bit inversion within the *faddp* instruction disturbs the top product composition. Faults injected at sampling instants: a) 28th, ISE=3.39, b) 101st, ISE=4.25, c) 224th ISE=2.53; golden run responses – solid line, fault injected responses – dashed line.

It is worth noting that overwhelming majority of incorrect behaviour relates to very high ISE values (higher than 1000). It means that the control values errors cumulates. At the other hand it gives the possibility to easily detect such big deviations using additional diagnostic subroutines.

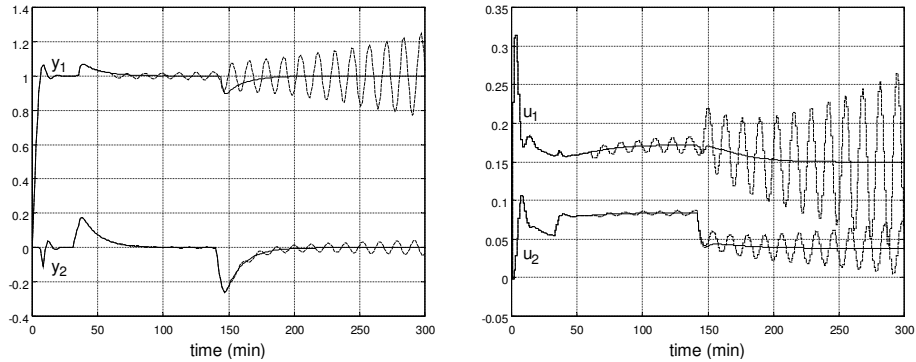


Fig. 6. Single bit inversion within the *faddp* instruction destabilises the process. Fault injected at the 44th sampling instant, ISE=4.40; golden run responses – solid line, fault injected responses – dashed line.

5 Conclusion

The paper studies dependability of software implementation of the explicit DMC algorithm applied to a rectification process. The process has two manipulated variables, two controlled variables and significant time-delays. Dependability is examined using software implemented fault injector.

Results of the experiments carried out clearly indicate that the well-known and widely used formulation of the DMC algorithm is susceptible to software faults. It is particularly important in case of industrial processes such as the considered rectification column because faults are likely to lead to undesirable behaviour of the process. More specifically, in the least difficult situation a fault can disturb composition of the products whereas in the most dangerous case it can destabilise the system. Technological and financial consequences of faults are of fundamental importance. Energy losses and unacceptable compositions of the products (which means that the product cannot be sold) are just two examples of such situations.

In order to increase fault robustness of software implementation of the DMC algorithm, a few techniques can be applied. In particular, current research embrace DMC algorithm based fault detection. In particular, the most critical situations (technologically unacceptable control values leading to instability) may be detected and handled by an additional diagnostic subroutine.

Acknowledgement. This work was supported by a grant from the dean of the Faculty of Electronics and Information Technology, Warsaw University of Technology.

References

1. Benso A., Prinetto P.: *Fault injection techniques and tools for embedded systems reliability evaluation*, Kluwer Academic Publishers. (2003).
2. Cutler R., Ramaker B.: *Dynamic matrix control – a computer control algorithm*, AIChE National Meeting, Houston. (1979).
3. Gawkowski P., Sosnowski J.: *Experiences with software implemented fault injection*, Int. Conf. on Architecture of Computing Systems. VDE Verlag GMBH. (2007) 73–80.
4. Gawkowski P., Sosnowski J.: *Analysing system susceptibility to faults with simulation tools*, Annales UMCS Informatica AI. **4** (2006) 123–134.
5. Gawkowski P., Sosnowski J.: *Software implemented fault detection and fault tolerance mechanisms – part I: Concepts and algorithms*, Kwartalnik Elektroniki i Telekomunikacji. **51** (2005) 291–303.
6. Gawkowski P., Sosnowski J.: *Software implemented fault detection and fault tolerance mechanisms -- part II: Experimental evaluation of error coverage*, Kwartalnik Elektroniki i Telekomunikacji. **51** (2005) 495–508.
7. Gawkowski P., Sosnowski J., Radko B.: *Analyzing the effectiveness of fault hardening procedures*, 11th IEEE Int. On-Line Testing Symposium. (2005) 14–19.
8. Gawkowski P., Sosnowski J.: *Dependability evaluation with fault injection experiments*, IEICE Transactions on Information & System. **E86-D** (2003) 2642–2649.
9. Gawkowski P., Sosnowski J.: *Experimental Validation Of Fault Detection And Fault Tolerance Mechanisms*, 7th IEEE Int. Workshop on High Level Design Validation And Test. Cannes (2002).
10. Ławryńczuk M., Marusak M., Tatjewski P.: *Multilayer and integrated structures for predictive control and economic optimisation*, IFAC Symposium on Large Scale Systems, Gdańsk (2007), accepted for publication.
11. Maciejowski J. M.: *Predictive control with constraints*, Prentice Hall. Harlow. (2002).
12. Morari M., Lee J. H.: *Model predictive control: past, present and future*, Computers and Chemical Engineering. **23** (1999) 667–682.
13. Qin S. J., Badgwell T. A.: *A survey of industrial model predictive control technology*, Control Engineering Practice. **11** (2003) 733–764.
14. Pułaczewski J.: *Multidimensional DMC algorithm*, Report of ICCE WUT no 98–11, Warsaw (1998), in Polish.
15. Rossiter J. A.: *Model-based predictive control*, CRC Press, Boca Raton. (2003)
16. Sosnowski J., Gawkowski P., Lesiak A.: *Fault injection stress strategies in dependability analysis*, Control and Cybernetics. **33** (2005) 679–699.
17. Sosnowski J., Lesiak A., Gawkowski P., Włodawiec P.: *Software implemented fault inserters*, IFAC Workshop on Programmable Devices and Systems. Ostrava (2003) 293–298.
18. Sosnowski J., Gawkowski P., Lesiak A.: *Fault injection stress strategies*, 4th IEEE LATW 2003 Workshop. (2003) 258–263.
19. Tatjewski P.: *Advanced control of industrial processes, Structures and algorithms*, Springer. London. (2007).
20. Tatjewski P., Ławryńczuk M., Marusak P.: *Linking nonlinear steady-state and target set-point optimisation for model predictive control*, IEE International Control Conference ICC 2006, Glasgow (2006), CD-ROM.
21. Wood R. K., Berry M. W.: *Terminal Composition Control of a Binary Distillation Column*, Chemical Engineering Science **28** (1973) 1707–1717.