AutoMATiC: Code Generation of Model Predictive Control Algorithms for Microcontrollers – Tutorial for the User

June 14, 2019

Contents

1 Introduction

This tutorial describes how the AutoMATiC software system should be used in order to generate the C code of software implementation of Model Predictive Control (MPC) algorithms [1, 5, 8] for a chosen target microcontroller. Multiple-Input Multiple-Output (MIMO) processes are considered with n_{u} manipulated variables (process inputs) $u_1, \ldots, u_{n_{\mathrm{u}}}$ and n_{y} controlled variables (process outputs) $y_1, \ldots, y_{n_{\mathrm{y}}}$. In general, in MPC, at each discrete sampling instant, $k = 0, 1, \ldots$, the increments of the future manipulated variables (a vector of length $n_{\mathrm{u}} N_{\mathrm{u}}$) are repeatedly calculated on-line

$$\triangle \boldsymbol{u}(k) = \begin{bmatrix} \triangle u(k|k) \\ \vdots \\ \triangle u(k + N_{\mathrm{u}} - 1|k) \end{bmatrix} \tag{1}$$

where $\triangle u(k+p|k)$ are backward differences of the manipulated variables for the future sampling instant $k + p$ calculated at the current instant k, N_{u} is the control horizon. When calculation is completed, only the control increments for the current sampling instant are applied to the process. In the consecutive sampling instants process output variables are measured and the optimization procedure is repeated.

Two versions of MPC algorithms are available: analytical and numerical. In the first case the objective of MPC is to minimise the cost-function

$$J(k) = \sum_{p=1}^{N} \| y^{\mathrm{sp}}(k + p|k) - \hat{y}(k + p|k) \|^2_{\boldsymbol{\Psi}_0} + \sum_{p=0}^{N_{\mathrm{u}}-1} \| \triangle u(k + p|k) \|^2_{\boldsymbol{\Lambda}_0} \tag{2}$$

Its first part measures future control errors over the prediction horizon N, the second one is a penalty function used to minimize increments of the manipulated variables. $y^{\mathrm{sp}}(k + p|k)$ and $\hat{y}(k + p|k)$ denote the vectors (of length n_{y}) of set-points and predicted values for the future sampling instant $k+p$ know at the instant k, $\boldsymbol{\Psi}_0 = \mathrm{diag}(\psi_1, \ldots, \psi_{n_{\mathrm{y}}})$ and $\boldsymbol{\Lambda}_0 = \mathrm{diag}(\lambda_1, \ldots, \lambda_{n_{\mathrm{u}}})$ are weighting matrices of dimensionality $n_{\mathrm{y}} \times n_{\mathrm{y}}$ and $n_{\mathrm{u}} \times n_{\mathrm{u}}$, respectively. The increments of the manipulated variables are calculated analytically, using computationally simple formulas [2]. The calculated signals are nex‘t projected onto the admissible set determined by the constraints imposed on the values of the manipulated variables for the current sampling instant k and the corresponding increments

$$u^{\min} \leq u(k) \leq u^{\max} \tag{3}$$

$$-\triangle u^{\max} \leq \triangle u(k) \leq \triangle u^{\max} \tag{4}$$

where $u^{\min}$, $u^{\max}$ and $\triangle u^{\max}$ (the vectors of length n_{u}) denote minimal and maximal values as well as the maximal change of the manipulated variables, respectively.

In the case of the numerical MPC algorithms, at each sampling instant k, the full vector of future control increments (1) is calculated as the solution of the following rudimentary MPC optimisation problem

$$\begin{aligned} &\min_{\triangle \boldsymbol{u}(k)} \{J(k)\} \\ &\text{subject to} \\ &u^{\min} \leq u(k + p|k) \leq u^{\max}, \ p = 0, \ldots, N_{\mathrm{u}} - 1 \\ &-\triangle u^{\max} \leq \triangle u(k + p|k) \leq \triangle u^{\max}, \ p = 0, \ldots, N_{\mathrm{u}} - 1 \end{aligned} \tag{5}$$

Provided that a linear model is used for prediction, i.e. to calculated the vectors $\hat{y}(k + p|k)$ for $p = 1, \ldots, N$, numerical MPC algorithms result in quadratic optimisation. The OSQP [7] library is used for quadratic optimisation in the AutoMATiC system.

Two MPC algorithms are available: Dynamic Matrix Control (DMC) [4] and Generalized Predictive Control (GPC) [3], each of them in analytical and numerical versions. The DMC algorithm uses for prediction a simple to obtain step-response model of the controlled process. Provided that the process is stable with no integration, the model is comprised by D (which is called the horizon of process dynamics) MIMO step-response matrices

$$\boldsymbol{S}_1, \boldsymbol{S}_2, \boldsymbol{S}_3, \ldots, \boldsymbol{S}_{\mathrm{D}} \tag{6}$$

where the matrices of dimensionality $n_{\mathrm{y}} \times n_{\mathrm{u}}$ are

$$\boldsymbol{S}_p = \begin{bmatrix} s_p^{1,1} & \ldots & s_p^{1,n_{\mathrm{u}}} \\ \vdots & \ddots & \vdots \\ s_p^{n_{\mathrm{y}},1} & \ldots & s_p^{n_{\mathrm{y}},n_{\mathrm{u}}} \end{bmatrix} \tag{7}$$

The scalar step-response coefficient for the sampling instant p, the input u_n and the output y_m is denoted by $s_p^{m,n}$. The GPC algorithms uses the following difference equation model [2]

$$\boldsymbol{A}(q^{-1})y(k) = \boldsymbol{B}(q^{-1})u(k) \tag{8}$$

where the entries of the matrices

$$\boldsymbol{A}(q^{-1}) = \begin{bmatrix} A_{1,1}(q^{-1}) & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & A_{n_{\mathrm{y}},n_{\mathrm{y}}}(q^{-1}) \end{bmatrix} \tag{9}$$

$$\boldsymbol{B}(q^{-1}) = \begin{bmatrix} B_{1,1}(q^{-1}) & \ldots & B_{1,n_{\mathrm{u}}}(q^{-1}) \\ \vdots & \ddots & \vdots \\ B_{n_{\mathrm{y}},1}(q^{-1}) & \ldots & B_{n_{\mathrm{y}},n_{\mathrm{u}}}(q^{-1}) \end{bmatrix} \tag{10}$$

are the polynomials in the backward shift operator q^{-1}

$$A_{m,m}(q^{-1}) = 1 + a_1^m q^{-1} + \ldots + a_{n_{\mathrm{A}}}^m q^{-n_{\mathrm{A}}} \tag{11}$$

for $m = 1, \ldots, n_{\mathrm{y}}$ and

$$B_{m,n}(q^{-1}) = b_1^{m,n} q^{-1} + \ldots + b_{n_{\mathrm{B}}}^{m,n} q^{-n_{\mathrm{B}}} \tag{12}$$

for $m = 1, \ldots, n_{\mathrm{y}}$, $n = 1, \ldots, n_{\mathrm{u}}$.

2 Conditions of Using the AutoMATiC System

```
AutoMATiC -- Automatic code generation based on the MATLAB and
C languages.

Copytight (C) 2018-2019 by Patryk Chaber. Developed within the
Warsaw University of Technology, Institute of Control and
Computation Engineering under supervision of Maciej Lawrynczuk.
All rights reserved.

AutoMATiC is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 3 of the License, or
(at your option) any later version.

AutoMATiC is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with AutoMATiC.  If not, see <https://www.gnu.org/licenses/>.
```

AutoMATiC is open-source software, so you can use it free of charge under the terms of the GNU GPL licence. If you are using the software in your research work, you are supposed to cite one or more of the following references.

```
@MISC{AutoMATiCTutorial,
  author = {P. Chaber},
  title = {{AutoMATiC} {T}utorial},
  howpublished = {https://github.com/pjchaber/automatic-mpc},
  year = {2019}
}

@ARTICLE{Chaber2019,
  author = {P. Chaber and M. \L{}awry\'{n}czuk},
  title = {{AutoMATiC}: Code Generation of Model Predictive Control Algorithms
     for Microcontrollers},
  note = {in review}
}
```

3 Requirements

This software was tested using following setup:

- Microsoft Windows 10 Pro
- MATLAB 2019a
- MinGW64 Compiler (C++)

and for the programming of the example microcontroller:

- STM32F746IGT6 microcontroller (part of the Core7XXI evaluation board with Open7XXI-C board – both produced by WaveShare)
- J-Link v9.3
- Keil uVision v5.17.0.0
- STM32CubeMX

Despite aforementioned configuration, this software was also tested with MATLAB 2017b, thus should work with any other MATLAB version in between of those two. This software should work also on Linux based operating systems, but it may require minor changes in paths notation.

4 Installation

To build AutoMATiC system compile its sources using `mex` compiler in MATLAB with the following steps:

1. download or clone the repository of AutoMATiC:
   `https://github.com/pjchaber/automatic-mpc.git`
2. make sure that MATLAB is callable from the shell by executing:
   ```
   matlab -batch "disp(version);"
   ```
   which should print output similar to the following:
   ```
   9.6.0.1072779 (R2019a)
   ```
   If the output differs, make sure that the following MATLAB directory:
   ```
   <MATLAB root directory>\bin\win64
   ```
   is listed in `PATH` variable of your system. For example:
   ```
   E:\Program Files\MATLAB\R2019a\bin\win64
   ```
3. register MATLAB for further use of its engine executing of the following instruction in shell:
   ```
   matlab -regserver
   ```
4. change current working directory to a directory with the source of the project
5. execute in MATLAB following instruction
   ```
   mex -v -client engine main.cpp XGetOpt.cpp -output automatic_transcompiler.
       exe;
   ```
   If the compilation fails, make sure that you have set a C++ compiler by executing in MATLAB:
   ```
   mex -setup C++
   ```
   Also make sure to use the same MATLAB version which was used in the shell in steps 2 and 3.

5 Setup

All further examples focus on the task of controlling output voltages of emulated process. All signals used to connect the controller and the process are voltage signals of range from 0 to 3.3V. From the implementation side, those signals are represented as values from -1 to 1, where the lowest value represents 0V and the highest one represents 3.3V. The process that is emulated is a matrix of single inertia transfer functions (13). Although the theoretical introduction considers any number of input and output signals, further on a process with only two inputs and two outputs is used. The process is configured exactly the same as the controller and utilises the same microcontroller to perform its tasks. Two ADC inputs of the controller are connected directly to the two DAC outputs of the process and two ADC inputs of the process are connected to the two DAC outputs of the controller, as shown in Fig. 1.

6 Generation of a Single MPC

Firstly, let us create the STM32CubeMX project which will be used as a base for further definition of a target platform's configuration (it is expected from the reader to know how to use STM32CubeMX and how to create projects for STM32F746IGT6 or similar and how to, based on such a project, generate a Keil uVision 5 project). Details of this project are as follows:

- Target Platform: STM32F746IGTx

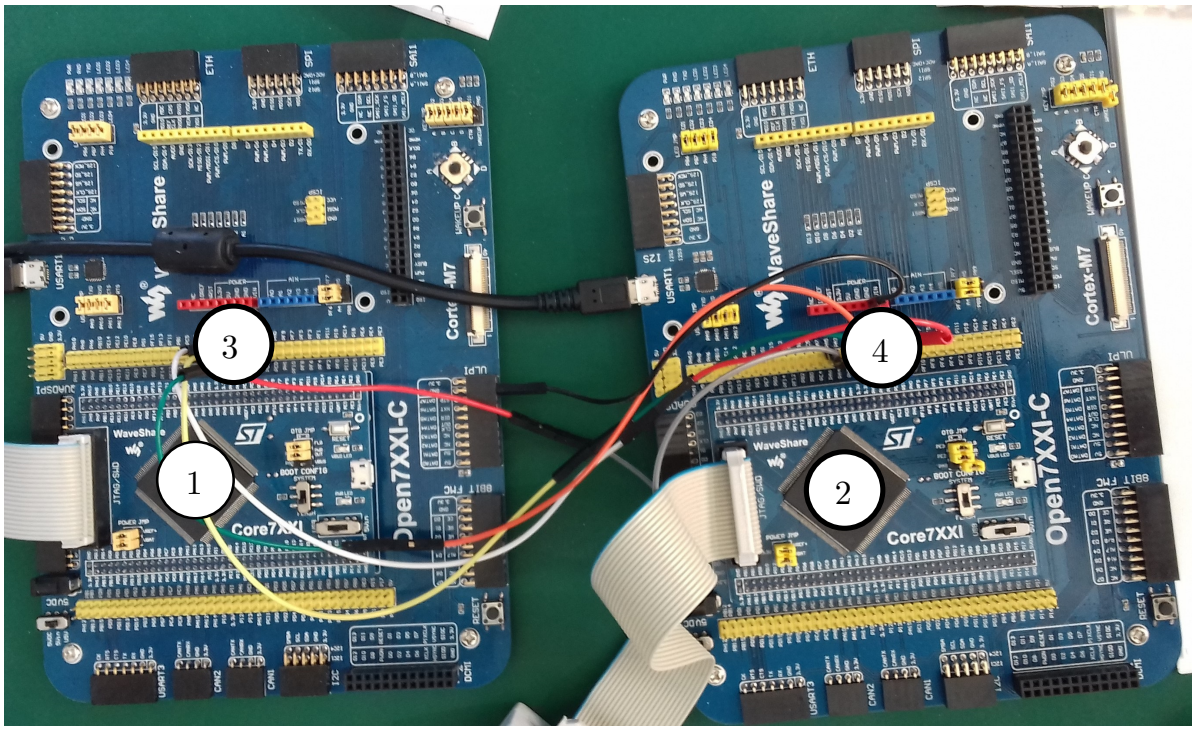

Figure 1: Microcontrollers representing the MPC controller (1) and the emulated process (2) with their corresponding ADC and DAC connection pins (3) and (4), respectively

- Pinout and Configuration:
  - RCC (HSE as Crystal/Ceramic Resonator)
  - ADC3 (IN0 and IN1 enabled):
    - Clock Prescaler: PLCK divided by 4
    - Scan Conversion: Enabled
    - Continuous Conversion: Enabled
    - DMA Continuous Requests: Enabled
    - Number of Regular Conversions: 2
    - Rank 1:
      - Channel: Channel 0
      - Sampling Time: 144 Cycles
    - Rank 2:
      - Channel: Channel 1
      - Sampling Time: 144 Cycles
  - DAC (OUT1 and OUT2 enabled), for both outputs:
    - Output Buffer: Enable
    - Trigger: None
  - TIM2 (Clock Source: Internal Clock):
    - Prescaler: 1000-1
    - Counter Period: 10800
  - TIM5 (Clock Source: Internal Clock):
    - Prescaler: 0
    - Counter Period: 0xffffffff

  - USART1 (Asynchronous):
    - Data Direction: Transmit Only
  - DMA2
    - ADC3:
      - Stream: DMA2 Stream 0
      - Direction: Peripheral to Memory
      - Priority: Low
      - Mode: Circular
      - Peripheral Data Width: Word
      - Memory Data Width: Word
    - ADC3:
      - Stream: DMA2 Stream 0
      - Direction: Peripheral to Memory
      - Priority: Low
      - Mode: Circular
      - Peripheral Data Width: Word
      - Memory Data Width: Word
  - NVIC:
    - Time base of System tick Timer (Preemption Priority: 5)
    - ADC1, ADC2 and ADC3 global interrupts (Enable, Preemption Priority: 3)
    - TIM2 global interrupts (Enabled, Preemption Priority: 1)
    - USART1 global interrupts (Enabled)
    - DMA2 stream0 global interrupts (Preemption Priority: 4)
  - RNG (Activated)

- Clock Configuration:
  - Input Frequency for HSE: 8Hz
  - PLL Source Mux: HSE
  - HCLK (MHz): 216 (confirm by pressing ENTER)

- Project Manager:
  - Toolchain / IDE: MDK-ARM V5
  - Minimum Heap Size: 0x8000
  - Minimum Stack Size: 0x40000

After generating code to a Keil uVision 5 project communication interface with the PC and interface to connect process of control has to be implemented. It is worth making sure that all the code added to the project after its generation is placed between comments `/* USER CODE BEGIN <name> */` and `/* USER CODE END <name> */` which will further on be denoted as "<name>" block. It will further ease the process of regeneration of the microcontroller's configuration.

6.1 Communication with PC using USART

Firstly, in the `main.c` file, `string.h` library header has to be included in the "Includes" block. This will allow to send text messages to a PC in a simple manner. Next a simple wrapper for transmitting of a string should be defined in block "0":

```
void write_string(char * txt){
  while(HAL_UART_GetState(&huart1) == HAL_UART_STATE_BUSY_TX);
  if(HAL_UART_Transmit_IT(&huart1, (uint8_t*)txt, strlen(txt))!= HAL_OK)
    Error_Handler();
  while(HAL_UART_GetState(&huart1) == HAL_UART_STATE_BUSY_TX);
}
```

At this point, it is possible to send text messages to a PC, it is worth noting that the new line character and carriage return is not included by default in the sent string.

All further controllers will send to a PC a message, at each sampling instant, in the following format `x = [<y1>,<y2>,<z1>,<z2>,<u1>,<u2>,<alg>,];`, where the consecutive placeholders are defined as following:

- `<y1>` – the measurement of the first output signal,
- `<y2>` – the measurement of the second output signal,
- `<z1>` – the set point for the first output signal,
- `<z2>` – the set point for the second output signal,
- `<u1>` – the first control signal applied to a controlled process,
- `<u2>` – the second control signal applied to a controlled process,
- `<alg>` – the number of the algorithm used in the controller (used only when more than one control algorithm is used).

This format is convenient because messages that follows this format can be used in the MATLAB script as an argument of the `eval` function and further easily plotted and saved as a CSV file.

6.2 Measurement of Voltage Signals

Voltage signals after conversion to a digital values will be averaged over 100 consecutive samples. Therefore a table for those raw measurements have to be defined in block "PV" as:

```
uint32_t adc_val_raw[ADC_SIZE*2] = {0};
```

where `ADC_SIZE` is defined in block "PD" as:

```
#define ADC_SIZE 100
```

Due to having two signals, the size of table with raw measurements is `ADC_SIZE*2`. To increase the resolution of averaged measurements, those will be stored as `float` variables (block "PV"):

```
float adc_val_f[2] = {0,0};
```

Having those defined, it is important to start a ADC unit. It can be implemented by inserting following code into a block "2":

```
HAL_ADC_Start(&hadc3);
if(HAL_ADC_Start_DMA(&hadc3, (uint32_t*)adc_val_raw, ADC_SIZE*2) != HAL_OK)
    Error_Handler();
```

Thanks to a DMA mechanism, after obtaining 100 consecutive samples of each signal a specific callback function will be executed. It can be redefined to perform measurements' averaging as soon as they are available, by inserting following code in block "0":

```
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* AdcHandle)
{
  static int i=0;
  static uint32_t tmpval[2] = {0,0};
  for(i=0,tmpval[0]=0,tmpval[1]=0;i<ADC_SIZE; ++i){
    tmpval[0] += adc_val_raw[2*i];
    tmpval[1] += adc_val_raw[2*i+1];
  }
  adc_val_f[0] = ((float)tmpval[0]/ADC_SIZE-2047.5f)/2047.5f;
  adc_val_f[1] = ((float)tmpval[1]/ADC_SIZE-2047.5f)/2047.5f;
}
```

This will allow to have constant supply of averaged results from ADC without consuming much of a computational power of a microcontroller core (thanks to DMA).

6.3 Creation of Voltage Signals

To generate voltage signal DAC are used. Those are already configured thanks to the generation of the code from STM32CubeMX, although, they have to be started before they can be used. To start them, in the block "2" there has to be invoked:

```
HAL_DAC_Start(&hdac, DAC_CHANNEL_1);
HAL_DAC_Start(&hdac, DAC_CHANNEL_2);
```

The following calls allow to generate 0V on first output and the 3.3V on the second one, since this is a 12 bit DAC:

```
HAL_DAC_SetValue(&hdac, DAC1_CHANNEL_1,DAC_ALIGN_12B_R,0);
HAL_DAC_SetValue(&hdac, DAC1_CHANNEL_2,DAC_ALIGN_12B_R,4095);
```

and thus, to use this with a variable that ranges from -1 to 1, the following function is defined (in block "WHILE"):

```
void __setControlValue(float* value){
  HAL_DAC_SetValue(&hdac, DAC1_CHANNEL_1,DAC_ALIGN_12B_R, (uint32_t)(value
      [0]*2047.5f+2047.5f));
  HAL_DAC_SetValue(&hdac, DAC1_CHANNEL_2,DAC_ALIGN_12B_R, (uint32_t)(value
      [1]*2047.5f+2047.5f));
}
```

where an array of two floats is an argument and it contains values for consecutive outputs.

6.4 Induction of Interrupts with a Constant Frequency

To generate an interrupt with a constant frequency which will further be used as an indicator of the beginning of new discrete sampling time, a timer TIM2 is used. It is already configured to generate events with a constant period of 100ms, but it is not started yet. To do so, the following code must be implemented in a block "2":

```
HAL_TIM_Base_Init(&htim2);
HAL_TIM_Base_Start_IT(&htim2);
```

This will cause an execution of a callback function which can be redefined:

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){
    if (htim->Instance == TIM2) {
    // interrupt each 100ms
    }
}
```

Although its implementation will be further extended.

6.5 Interface for Further Ease of Use

At this point it is worth to create additional functions that will wrap already existing functionality, so that anyone who will be working with this project in terms of control algorithms, will have easy interface to work with. Firstly, to give access to measurements signals the following will be used (block "0"):

```
float* __measureOutput(){
  return adc_val_f;
}
```

and it will return a pointer to a two element array of measurement values. It is clearly visible that it is analogous to the previously defined `void __setControlValue(float* value)` function. Those two functions have to have their declarations in the header file corresponding to the source file in which they are defined (block "EFP" in `main.h`). In this header, it could be convenient to declare also a function for communication with a PC `void write_string(char * txt)`.

Because the main function of the program (namely `int main(void)`) is defined in the AUTOMATiC system software framework, the existing one have to be redefined using another name,

e.g. `void low_lvl_main(void)` which of course also has to be accessible from other source files. Therefore its declaration has to be given in a proper header file. It is also important to remove the `while` loop from the `low_lvl_main` function, so that it will not hang the controller.

At the end it will be convenient to set up the Keil uVision 5 project in such a way that the transcompilation process is executed always right before the build process starts. For this, "Options for Target" window has to be opened and in the "User" tab, in "Before Build/Rebuild" a "Run #1" has to be filled with an execution of the transcompiler. The following will be used:

```
<absolute path to AutoMATiC>/automatic_transcompiler.exe -I -p -c -i ../Src/
    main.mpc -o ../Src/main_mpc.c -l <absolute path to AutoMATiC>/Libs/MATLAB/ -
    L <absolute path to AutoMATiC>/Libs/C/
```

It is worth noting that the transcompiler is executed while being in the directory with a project file (i.e. `MDK-ARM`). Consecutive arguments of the `automatic_transcompiler.exe` execution denotes:

- `-I` – usage of the interrupt based software framework variant,
- `-p` – usage of the profiler,
- `-c` – usage of the forced delay of the control signal application,
- `-i <path>` – relative path to the input file used for transcompilation,
- `-o <path>` – relative path to the output file storing the result of transcompilation,
- `-l <path>` – relative path to the directory with MATLAB libraries of the transcompiler,
- `-L <path>` – relative path to the directory with C libraries of the transcompiler.

To complete the setup of the project one has to add source files and headers of AutoMATiC system to a project for further compilation. All these files can be found in a directory `<absolute path to AutoMATiC>/Libs/C/`. Also a file named `main.mpc` has to be created and placed in the `../Src/` directory. This file will contain all the logic of the controller with as few low level implementation parts as possible.

6.6 Logic of the Control Algorithm

The file that contains logic of the controller, namely `main.mpc`, contains implementations which can be divided in a few parts:

- utilisation of the low level interface functions,
- invoke of the MATLAB script, used for automatic code generation of the MPC algorithm,
- implementation of controllers behaviour.

The execution of the MATLAB script is not necessary – content of this script could be as well placed right here. This notation nevertheless allows for clearer separation of the aforementioned parts of implementation of this file:

```
#include "stm32f7xx_hal.h"
#include <string.h>
#include "main.h"
#include "mat_lib.h"

ArchiveData ad;
CurrentControl cc;

long get_time(){ return HAL_GetTick(); }

extern void timer_loop(void);
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){
```

```
    if (htim->Instance == TIM2) {
        timer_loop();
    }
}
void measurements(){
    #MPC_PROFILER_BEGIN 4 measurements
    new_output(&ad,__measureOutput());
    #MPC_PROFILER_END 4
}
void controls(){
    #MPC_PROFILER_BEGIN 3 controls' application
    __setControlValue(last_control(&ad));
    #MPC_PROFILER_END 3
}
void hardware_setup(){
    #MPC_PROFILER_BEGIN 1 hardware setup
    low_lvl_main();
    #MPC_PROFILER_END 1
}

#MPC_BEGIN
% Here is the MATLAB code
algorithms_parameters_definitions
#MPC_END

void controller_setup(){
    #MPC_PROFILER_BEGIN 2 software setup
    init_archive_data(&ad, 200, 200, 2, 2, 0, 0, 0.01);
    init_current_control(&cc,&ad);
    controller(NULL,NULL);
    #MPC_PROFILER_END 2
}

void idle(){
    #MPC_PROFILER_BEGIN 13 other procedures
    const int k = 0;
  static char str[1000] = {0};

    sprintf(str, "x = [%f,%f,",ad.y[k][0], ad.y[k][1]);    write_string(str);
    sprintf(str,      "%f,%f,",ad.z[0], ad.z[1]);          write_string(str);
    sprintf(str,      "%f,%f,",ad.u[k-1][0],ad.u[k-1][1]); write_string(str);
    write_string("];\n\r");
    #MPC_PROFILER_END 13
}

void loop(){
    #MPC_PROFILER_BEGIN 10 controls' calculation
    static int i = 0;
  if(i<  50){ ad.z[0] = -0.1; ad.z[1] =  0.2; }
  else      { ad.z[0] =  0.1; ad.z[1] = -0.2; }
    if(++i > 100) i = 0;

    #MPC_PROFILER_BEGIN 50 control algorithm
  controller(&ad,&cc);
  #MPC_PROFILER_END 50

    push_current_controls_to_archive_data(&cc,&ad);
    #MPC_PROFILER_END 10
}

void timeout(){
    while(1);
}
```

On the listing, there is a reference to a function called `controller` which declaration is as follows:

```
void controller(ArchiveData * ad, CurrentControl * c);
```

This function will be generated as soon as the MATLAB script is finished.

The AutoMATiC system software framework assumes that there is an implementation of functions:

- `void write_string(char * str);` – a function used by the profiler to write out the results of its measurements,
- `long get_time(void);` – a function used by the profiler to measure time of execution,
- `void hardware_setup(void);` – a function used to setup hardware configuration of the controller,
- `void controller_setup(void);` – a function used to setup software configuration of the controller,
- `void measurements(void);` – a function used to obtain measurements for the current controller iteration,
- `void loop(void);` – a function used to define the behaviour of the controller – here control algorithms are executed,
- `void controls(void);` – a function used to determine what to do with results obtained from the consecutive control algorithms,
- `void idle(void);` – a function used for other, lower prioritized procedures,
- `void timeout(void);` – a function executed when the calculation lasts longer than the sampling period.

These functions are used in the framework, in the predefined order. If some of these are not implemented, a default (often empty) implementation is assumed.

To implement interrupt based variant of the software framework, a function

```
extern void timer_loop(void);
```

has to be declared and called each time a new discrete time instant starts. In these examples it occurs each time, the timer TIM2 generates an interrupt and thus causes execution of the following function:

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim);
```

Each new measurement has to be stored in the `ArchiveData` structure using

```
void new_output(ArchiveDataPtr ad, float* y);
```

function. Also the final values of new control signals have to be stored in the same structure using

```
void push_current_controls_to_archive_data(CurrentControlPtr c,ArchiveDataPtr
    ad);
```

where `CurrentControl` structure is filled by each control algorithm.

Lastly, there are a few functions in the `void controller_setup()` function that are necessary to use those two structures. A function

```
void init_archive_data(ArchiveDataPtr ad,
  long number_of_u, long number_of_y,
  long size_of_u, long size_of_y,
  float default_u, float default_y,
  float current_yzad);
```

is used to initialize the `ArchiveData` instance. The first argument is a pointer to this structure instance, next there are two parameters used to define number of control and output signals (in this order) of the process of control. Further there are two values that determine how many

previous values of control and output signals have to be stored in the memory of the controller. Finally, there are default values of control, output and set-point signal, used for initialization of the microcontroller memory – those can be changed as soon as the initialization function is finished.

Next function is

```
void init_current_control(CurrentControlPtr cc, ArchiveDataPtr ad);
```

which is used to allocate memory for an instance of `CurrentControl` based on the data stored in the alreaty setup `ArchiveData` instance.

At the end of the software setup, there is an execution of the controller function, with both parameters set to `NULL` which is used to initialize some values of the controller that are constant through its whole existence – this will be clear as soon as the generated code of the algorithm appears.

6.7 Dynamic Matrix Control Algorithm: Analytical version

From this point on, only the `algorithms_parameters_definitions.m` will be modified in order to generate software implementations of various control algorithms. It is important to make sure that this script is placed in the `Src` directory of the Keil uVision 5 project. All further algorithms will be used to control output signals of the process described with the following example transfer function matrix:

$$G(s) = \begin{bmatrix} \frac{5}{0.14s+1} & \frac{1}{0.10s+1} \\ \frac{1}{0.06s+1} & \frac{2}{0.08s+1} \end{bmatrix} \tag{13}$$

Based on this transfer function one can derive various discrete time models, specifically model in a form of step response coefficients (which is required by the DMC algorithm) or a set of difference equations (used in the GPC algorithm). First, one must create a continuous time transfer function in MATLAB:

```
clear all
%% Process definition
Gs  = [...
    tf(5,[.14,1]),tf(1,[.10,1]);...
    tf(1,[.06,1]),tf(2,[.08,1]);...
];              % continuous transfer function
```

It is worth to clear the workspace at the beginning of this script in order to avoid problems with accidental use of old variables. Next, this transfer function have to be transformed into a discrete time one and thus following have to be executed:

```
Gz = c2d(Gs,0.1); % discrete transfer function
ny = 2; % number of output signals
nu = 2; % number of control signals
```

Additional definition of the number of output and control signals is just for the cleaner code in further part of this script. For the discretization, a sampling time of 0.1 s was used.

To determine step response coefficients a `step` function of MATLAB could be used, but results would have to be modified in the convenient way. For the sake of clarity, this script does not utilize this function, it calculates this coefficients based on the GPC model instead. Therefore the ARX model used in GPC will be determined firstly:

```
for m=1:2 % ARX model determination
    tmpa = conv(Gz.Den{m,1},Gz.Den{m,2}); GPC_a(m,:)   = tmpa(2:end);
    tmpb = conv(Gz.Num{m,1},Gz.Den{m,2}); GPC_b(m,1,:) = [tmpb(1:end) 0];
    tmpb = conv(Gz.Num{m,2},Gz.Den{m,1}); GPC_b(m,2,:) = [tmpb(1:end) 0];
end
```

This script cannot be used for a process with a different dimensionality – a more general approach would be necessary. Next, the dynamics horizon is defined and the step response is obtained:

```
D = 5;  % dynamics horizon
S = step_response_generation(GPC_a,GPC_b,D);
```

and the step_response_generation is defined at the end of this script as follows:

```
function S = step_response_generation(GPC_a,GPC_b,D)
    ny = size(GPC_a,1);
    nu = size(GPC_b,2);
    S = zeros(ny,nu,D);
    for k = 1:size(S,3)
        for m=1:ny
            for n=1:nu
                for i=1:min(k,size(GPC_b,3))
                    S(m,n,k) = S(m,n,k) + GPC_b(m,n,i)*1;
                end
                for i=1:min(k-1,size(GPC_a,2))
                    S(m,n,k) = S(m,n,k) - GPC_a(m,i)*S(m,n,k-i);
                end
            end
        end
    end
    % adding a single iteration of delay
    S(:,:,2:end+1) = S(:,:,1:end);
    S(:,:,1) = S(:,:,1)*0;
end
```

Here, a set of general parameters of MPC algorithms is defined:

```
% General control algorithms' parameters
N  = 5; % prediction horizon
Nu = 5; % control horizon
lambda= [1.0 1.0]; % Penalty--control signals' increments
psi   = [1.0 1.0]; % Penalty--control errors
dumin =-[.01 .01]; % Lower bound--control increments
dumax = [.01 .01]; % Upper bound--control increments
umin  =-[1.0 1.0]; % Lower bound--control signal
umax  = [1.0 1.0]; % Upper bound--control signal
```

and lastly, the automatic code generation procedure is called:

```
AutoMATiC_Generate('AutoMATiC_DMC_Analytic_Algorithm','controller');
```

This function is defined in the script in <path to AutoMATiC>/Libs/MATLAB/. Three arguments at most can be passed to this function, in the following order: the name of the algorithm which definition will be used to generate C code (it corresponds to the name of the script with an algorithm definition – these can be found in the <path to AutoMATiC>/Libs/MATLAB/ directory), the name of the generated C function and a binary logic value which defines if there will be more algorithms generated in the same MATLAB script. The meaning of the last parameter will be clear in the example where two algorithms are generated for one controller project.

At this point, if the build procedure will be called on the Keil uVision5 project, the automatic code generation will be called first which will generate a file called main_mpc.c which can be found in Appendix A.

As it is visible, the <path to AutoMATiC> in this example is C:\Users\Admin\Documents\GitHub\AutoMATiC. It is worth noting that there is no part where the MATLAB code could be placed, instead there is a definition of the void controller(ArchiveData * ad, CurrentControl * c) function. Also each pair of #MPC_PROFILER_BEGIN<id> <description> and #MPC_PROFILER_END<id> is substituted with a pair of corresponding C functions profiler_start(<id>,<description>) and profiler_end(<id>), where <id> is the identifier of the block, and the <description> is the additional description of this block. If this generated C language source file is not included in the project and the build fails – it is necessary to add this file to the project and repeat the building process. At this point everything should be ready to program the microcontroller and test the MPC controller itself.

A plot of signals measurements obtained from this controller is shown in Fig. 2. Also the result generated with a profiler is shown in Table 1. The consecutive columns in this and further shown tables are as follows:

- id – the identifier of the block of code which was to be inspected (additional description in the parenthesis),

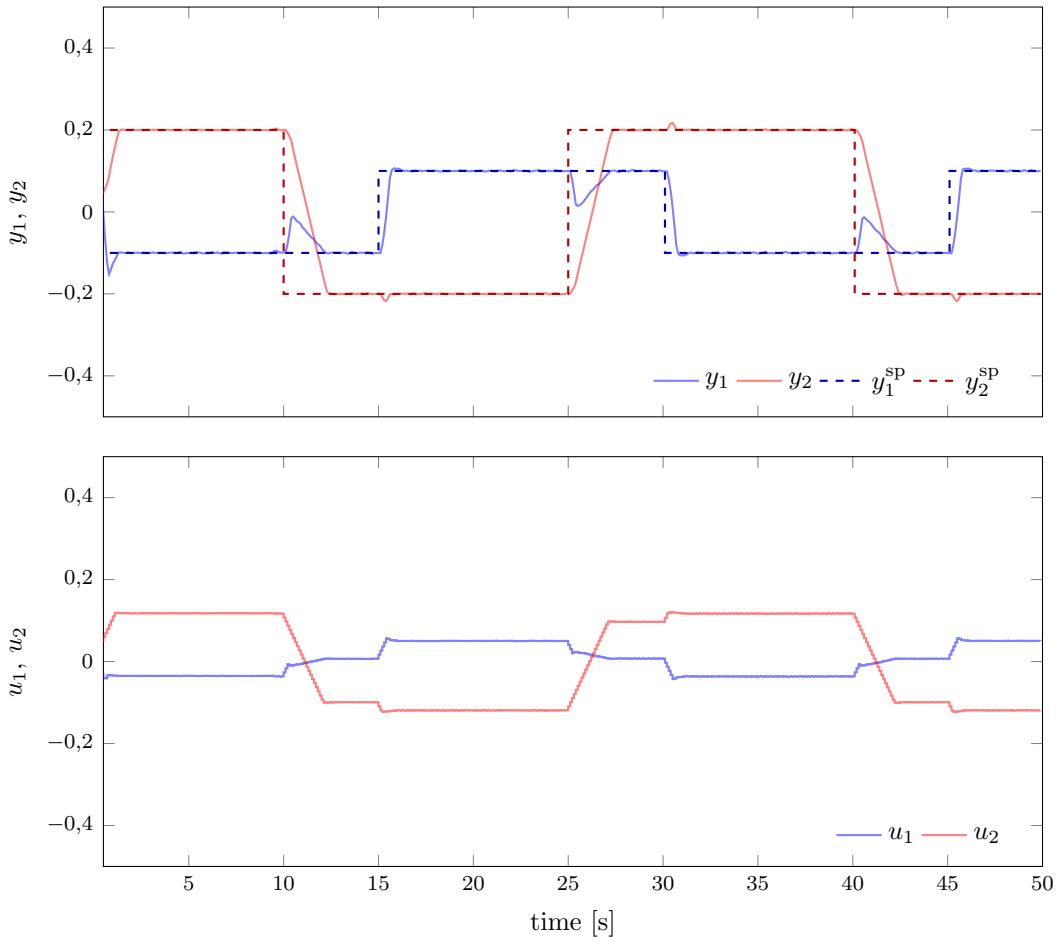

Figure 2: Trajectories of controlled and manipulated variables obtained when the analytical DMC controller is used

Table 1: Profiler results obtained when the analytical DMC controller is used

id	time total	entries	min	mean	max	running
1 (hardware setup)	1	1	1	1.0000	1	0
2 (software setup)	1	1	1	1.0000	1	0
3 (controls' application)	8	1001	0	0.0079	1	0
4 (measurements)	59	1001	0	0.0589	1	0
10 (controls' calculation)	190	1001	0	0.1898	1	0
50 (control algorithm)	88	1001	0	0.0879	1	0
13 (other procedures)	7015	1001	6	7.0079	8	1

Table 2: Profiler results obtained when the numerical DMC controller is used

id	time total	entries	min	mean	max	running
1 (hardware setup)	1	1	1	1.0000	1	0
2 (software setup)	13	1	13	13.0000	13	0
3 (controls' application)	0	1001	0	0.0000	0	0
4 (measurements)	66	1001	0	0.0659	1	0
10 (controls' calculation)	7415	1001	7	7.4075	43	0
50 (control algorithm)	7307	1001	7	7.2997	43	0
13 (other procedures)	6988	1001	6	6.9810	8	1

- time total – the total time that the program spent executing this block,
- entries – the number of times the program entered this block,
- min – the minimum amount of time spent executing this block,
- mean – the mean amount of time spent executing this block,
- max – the maximum amount of time spent executing this block,
- running – the flag denoting if this data was obtained during execution of this block (1 – yes, 0 – no).

All time measurements are obtained using the `get_time` function, thus in this and following examples, their unit is a millisecond.

6.8 Dynamic Matrix Control Algorithm: Numerical Version

Based on the already created project for analytical version of DMC algorithm, a change to its numerical version is straightforward. It only requires the change od the name of used algorithm, i.e. the following line from the `algorithms_parameters_definitions.m` script:

```
AutoMATiC_Generate('AutoMATiC_DMC_Analytic_Algorithm','controller');
```

has to be changed to:

```
AutoMATiC_Generate('AutoMATiC_DMC_Numeric_Algorithm','controller');
```

which, after performing same tasks as before to generate the final pure C code, will result in the code listed in Appendix B.

A plot of signals measurements obtained from this controller is shown in Fig. 3. Also the result generated with a profiler is shown in Table 2.

6.9 General Predictive Control Algorithm: Analytical Version

Due to the already performed determination of the GPC model in considered MATLAB script, only a simple modification has to be performed to change which algorithm is to be generated to a GPC in its analytical version. Therefore the line:

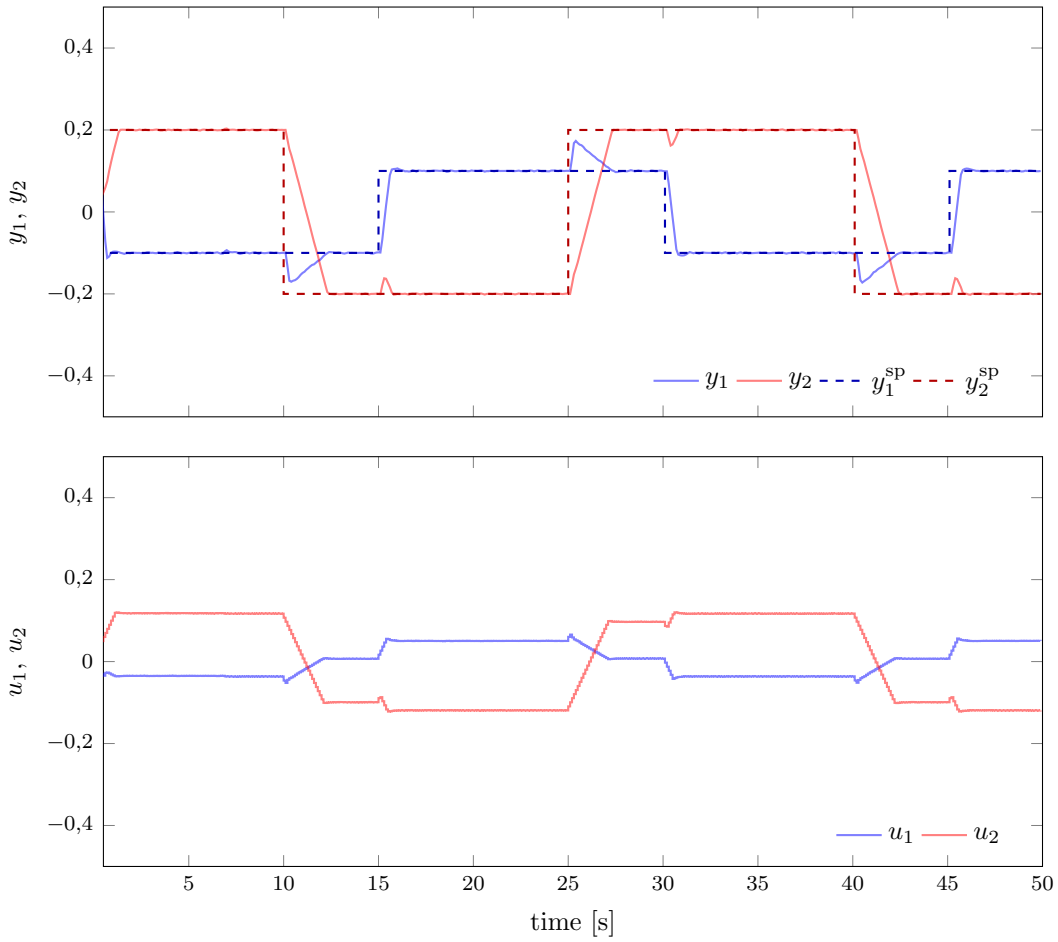

Figure 3: Trajectories of controlled and manipulated variables obtained when the numerical DMC controller is used

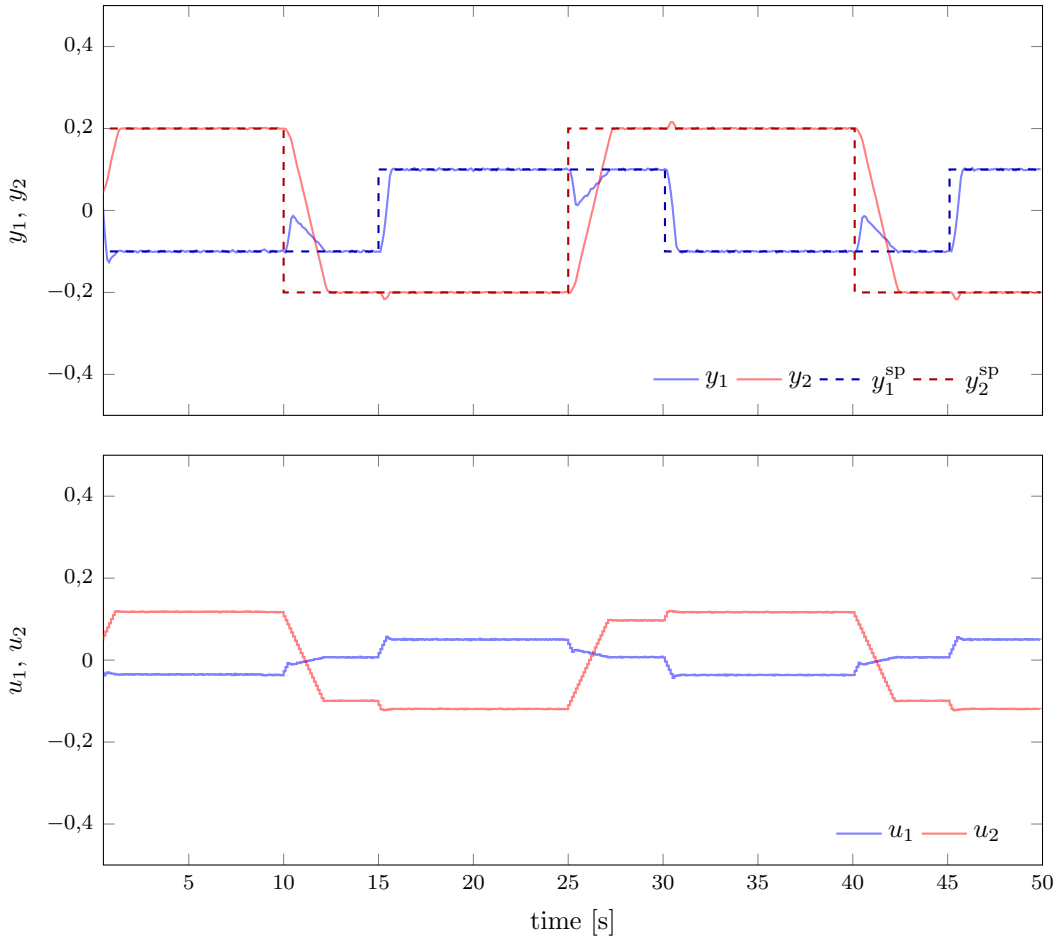

Figure 4: Trajectories of controlled and manipulated variables obtained when the analytical GPC controller is used

```
AutoMATiC_Generate('AutoMATiC_DMC_Numeric_Algorithm','controller');
```
should be changed to:
```
AutoMATiC_Generate('AutoMATiC_GPC_Analytic_Algorithm','controller');
```
which, after performing same tasks as before to generate the final pure C code, will result in the code listed in Appendix C.

A plot of signals measurements obtained from this controller is shown in Fig. 4. Also the result generated with a profiler is shown in Table 3.

6.10 General Predictive Control Algorithm: Numerical Version

Again, to change the analytical version of GPC algorithm to its numerical version, a change from:
```
AutoMATiC_Generate('AutoMATiC_GPC_Analytic_Algorithm','controller');
```
to:
```
AutoMATiC_Generate('AutoMATiC_GPC_Numeric_Algorithm','controller');
```
has to be performed. After performing same tasks as before to generate the final pure C code, will result in the code listed in Appendix D.

A plot of signals measurements obtained from this controller is shown in Fig. 5. Also the result generated with a profiler is shown in Table 4.

Table 3: Profiler results obtained when the analytical GPC controller is used

id	time total	entries	min	mean	max	running
1 (hardware setup)	1	1	1	1.0000	1	0
2 (software setup)	1	1	1	1.0000	1	0
3 (controls' application)	1	1001	0	0.0009	1	0
4 (measurements)	59	1001	0	0.0589	1	0
10 (controls' calculation)	164	1001	0	0.1638	1	0
50 (control algorithm)	58	1001	0	0.0579	1	0
13 (other procedures)	7026	1001	6	7.0189	8	1

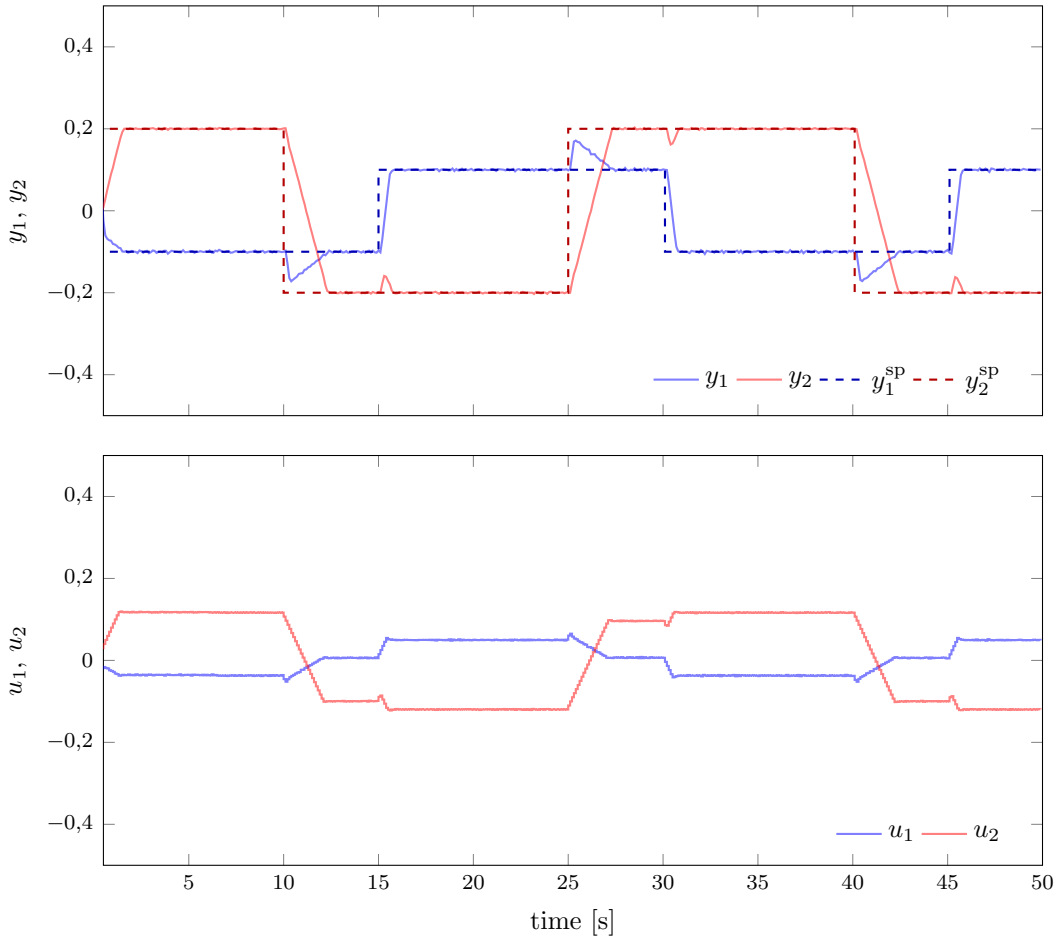

Figure 5: Trajectories of controlled and manipulated variables obtained when the numerical GPC controller is used

Table 4: Profiler results obtained when the numerical GPC controller is used

id	time total	entries	min	mean	max	running
1 (hardware setup)	1	1	1	1.0000	1	0
2 (software setup)	13	1	13	13.0000	13	0
3 (controls' application)	0	1001	0	0.0000	0	0
4 (measurements)	64	1001	0	0.0639	1	0
10 (controls' calculation)	8691	1001	7	8.6823	44	0
50 (control algorithm)	8596	1001	7	8.5874	44	0
13 (other procedures)	6990	1001	6	6.9830	8	1

7 Multiple MPC Algorithms Exchanged On-line in a Seamless Way Using One Microcontroller

It is possible to implement the MPC controller which contains a number of different algorithms which are exchanged on-line in a seamless way. Let us consider an example in which two versions of the DMC algorithm are implemented, i.e. analytical and numerical. In such a case it is required to perform some minor changes both in the controllers' logic (`main.mpc`) and the MATLAB script that defines parameters of used algorithms (`algorithms_parameters_definitions.m`). In the logic part, a new global variable, e.g. `int algorithm_used = 0;` should be created – it will define which algorithm is currently used. Next, the switching logic should be implemented – in the `void loop()` function a simple condition will be added:

```
if(algorithm_used == 0) controllerDMCA(&ad,&cc);
else                    controllerDMCN(&ad,&cc);
```

which will define that if the variable `algorithm_used` equals 0, the analytical version of DMC algorithm will be used, otherwise a numerical version will be executed. It is also clearly visible that those algorithms have new, distinct names. These should be used consequently in both C and MATLAB code. All will be properly set further in this document. First, the software initialization function (named `controller_setup`) has to be modified, by substituting a line:

```
controller(NULL,NULL);
```

with two new lines:

```
controllerDMCA(NULL,NULL);
controllerDMCN(NULL,NULL);
```

Lastly, the value of variable `algorithm_used` has to be defined in time. Therefore, in the function `loop()`, right before the newly added conditional execution of control algorithms, the following code should be added:

```
if((i%10==0) && ((random() % 100) < 10))
  algorithm_used = algorithm_used==0?1:0;
```

which will cause the algorithm to change with a constant probability every 10 sampling instants. The `random()` is defined as follows in the `main.c` file in the block "0":

```
uint32_t random(void){
  static uint32_t rnd = 0;
  HAL_RNG_GenerateRandomNumber(&hrng, &rnd);
  return rnd;
}
```

Having the C part upgraded, next, the MATLAB script has to be modified. Firstly, the line:

```
AutoMATiC_Generate('AutoMATiC_DMC_Numeric_Algorithm','controller');
```

has to be changed to:

```
AutoMATiC_Generate('AutoMATiC_DMC_Numeric_Algorithm','controllerDMCN',1);
```

and the following code has to be appended to this script:

```
lambda= [0.1 0.1]; % Penalty--control signals' increments
dumin =-[.1  .1 ]; % Lower bound--control increments
dumax = [.1  .1 ]; % Upper bound--control increments
AutoMATiC_Generate('AutoMATiC_DMC_Analytic_Algorithm','controllerDMCA',0);
```

which will generate the second definition of an algorithm, with lower penalty for control signals' increments and also more relaxed constraints on control increments rendering this particular implementation a bit more aggressive. By adding the third argument to the execution of the `AutoMATiC_Generate` function, the transcompiler is informed that the generated algorithm is not yet the last one in the project (value of 1), or that it is indeed the last one that will be generated (value of 0). This script finally completes the generation of multiple algorithms in the same project, where a simple condition is used to switch between those two. The resulting code is presented in Appendix E.

A plot of signals measurements obtained from this controller is shown in Fig. 6. Also the result generated with a profiler is shown in Table 5.

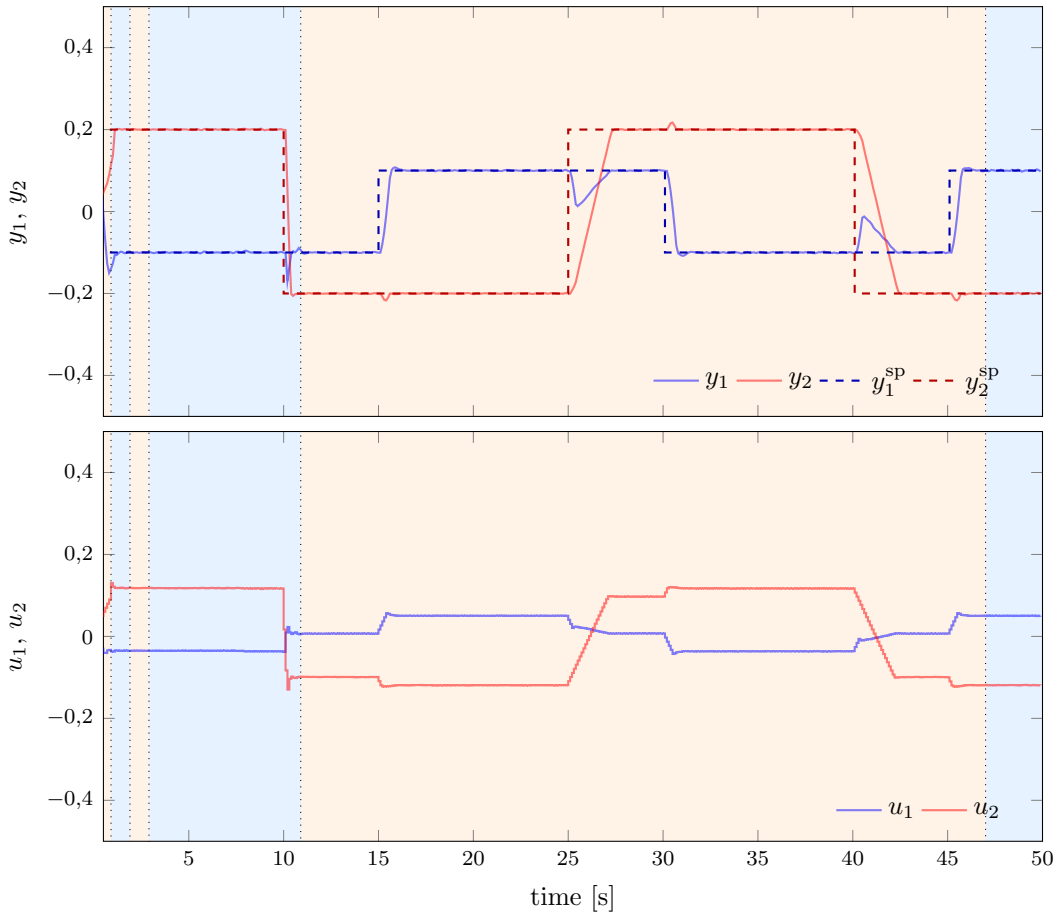

Figure 6: Trajectories of controlled and manipulated variables obtained when the analytical and numerical DMC controllers are exchanged on-line in a seamless way

Table 5: Profiler results obtained when the analytical and numerical DMC controllers are exchanged on-line in a seamless way

id	time total	entries	min	mean	max	running
1 (hardware setup)	1	1	1	1.0000	1	0
2 (software setup)	13	1	13	13.0000	13	0
3 (controls' application)	8	1001	0	0.0079	1	0
4 (measurements)	60	1001	0	0.0599	1	0
10 (controls' calculation)	3015	1001	0	3.0119	14	0
50 (control algorithm)	2916	1001	0	2.9130	14	0
13 (other procedures)	7177	1001	6	7.1698	8	1

8 Extension of the Library of Control Algorithms

To extend the list of algorithms that are available to be used for automatic code generation one has to add a new MATLAB script to a library directory `<path to AutoMATiC>/Libs/MATLAB/`. As an example a script containing two simple single-loop Proportional-Integral-Derivative (PID) controllers will be created.

8.1 Proportional-Integral-Derivative Controller

PID is a control algorithm used for processes that have one input and one output signals. Therefore for a process with many input and many output signals, respective number of PID controllers (loops) has to be defined. It is assumed that PID control loops are created for pairs of consecutive input and output signals, i.e. there will be loops for 1st input and 1st output signal, 2nd input and 2nd output signals, etc. Each loop will be characterised by classic set of PID parameters:

- K (`K`)– proportional gain,
- T_i (`Ti`)– integration time constant,
- T_d (`Td`)– derivative time constant,

and one common parameter for all those loops, namely T (`T`) – sampling time. Also, like in the DMC and GPC algorithms, there are constraints defined for each control (input) signal $u_{\min}$ (`umin`) $\leq u(k) \leq u_{\max}$ (`umax`) and its increments $\triangle u_{\min}$ (`dumin`) $\leq \triangle u(k) \leq \triangle u_{\max}$ (`dumax`), where k is a current time instant. Therefore, the following script will be used as a parameters definition of this algorithm (assuming the same process of control as before):

```
T = 0.1; % sampling time [s]
dumin =-[ .1   .1]; % Lower bound--control increments
dumax = [ .1   .1]; % Upper bound--control increments
umin  =-[1.0 1.0]; % Lower bound--control signal
umax  = [1.0 1.0]; % Upper bound--control signal

K =  [ .1   .05]; % proportional gains
Ti = [1     .1 ]; % integration time constants [s]
Td = [0.0 0.0 ]; % derivative time constants [s]

AutoMATiC_Generate('AutoMATiC_PID_Algorithm','controller');
```

The PID controller that is here implemented is defined as follow:

$$\triangle u_n(k) = e_n(k) r_n^{(0)} + e_n(k-1) r_n^{(1)} + e_n(k-2) r_n^{(0)} \tag{14}$$

where $e_n(k-p) = y_n^{\text{sp}} - y_n(k-p)$. The index n in this case denotes both input and output signal number. The implementation of this controller starts with a definition of constant values. Notice that in the parameters definition constants K, T_i and T_d were used and in the PID definition values of $r_n^{(1)}$, $r_n^{(1)}$ and $r_n^{(2)}$ are needed. There is a simple relation between those two sets of parameters which will be implemented in this very part. The constants have to be defined in a conditional block of:

```
if( INIT == 1 )
  ... here define constants
  INIT = 0; return;
end
```

where the `INIT` is a special variable that is utilised by the transcompiler to determine which part of the code has to be executed and which one should be translated. Constant definition for PID will be therefore defined as follows:

```
if( INIT == 1 )
    nu = length(K);
    r = [K.*Td./T;
         K.*(T./(2*Ti)-2*Td./T-1);
         K.*(1+T./(2*Ti)+Td./T)];
    INIT = 0; return;
end
```

Apart from the aforementioned variables, there is defined an additional one, namely `nu` which denotes a number of PID controllers and thus number of input or output signals. It is worth underlining that the variables used in the algorithms definition are accessed using the base workspace. That means that every variable defined in the base scope, can be used in the definition of the algorithm. This induces that names used in the control algorithms definition should be as unique as possible – for the purpose of clarity of code, this example will ignore this requirement.

After the constants, a code to be translated has to be defined. Therefore a comment line starting from `%% TRANSLATE` has to be inserted. From this point on the code will only be translated, and thus a limited set of MATLAB functionality is available. The PID controller will calculate its results in following steps:

1. create variables for control errors and increments of control signals,
2. create variables for intermediate results,
3. for each control loop:
   (a) calculate errors,
   (b) calculate product of errors time PID constants,
   (c) calculate control signal increment,
   (d) incorporate constraints for control signal increment,
   (e) calculate temporary value of control signal,
   (f) incorporate constraints for control signal
4. store calculated control signal increments in a special variable

Following the steps above, the following implementation can be created:

```
e      = zeros(nu, 3); du = zeros(nu,1);
er     = zeros(nu, 3); tmpu = zeros(nu,1);
control_value = zeros(1,nu);
for n=1:nu
    e( n,1) = AD_Z(n) - AD_Y(AD_K-0,n);
    e( n,2) = AD_Z(n) - AD_Y(AD_K-1,n);
    e( n,3) = AD_Z(n) - AD_Y(AD_K-2,n);
    er(n,1) = e(n,1)*r(1,n);
    er(n,2) = e(n,2)*r(2,n);
    er(n,3) = e(n,3)*r(3,n);

    du(n,1) = du(n,1) + er(n,1);
    du(n,1) = du(n,1) + er(n,2);
    du(n,1) = du(n,1) + er(n,3);

    if(du(n,1)>dumax(1,n)); du(n,1) = dumax(1,n); end
    if(du(n,1)<dumin(1,n)); du(n,1) = dumin(1,n); end
    tmpu(n,1) = AD_U(AD_K-1,n) + du(n,1);
    if(tmpu(n,1)>umax(1,n)); tmpu(n,1) = umax(1,n); end
    if(tmpu(n,1)<umin(1,n)); tmpu(n,1) = umin(1,n); end
    du(n,1) = tmpu(n,1) - AD_U(AD_K-1,n);
end

for n=1:nu
    control_value(1,n) = du(n,1);
end
```

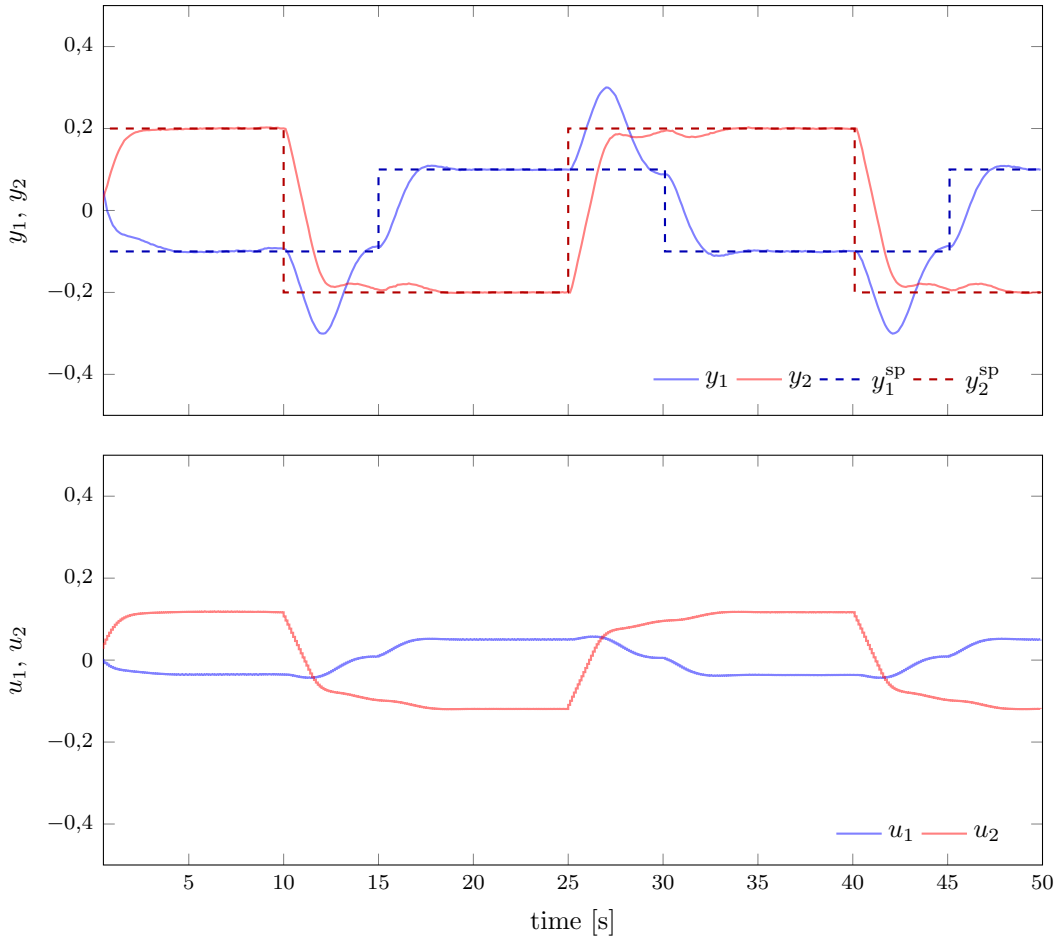

Figure 7: Trajectories of controlled and manipulated variables obtained when the PID controller is used

There are a few aspects worth noting. Firstly, a set point value can be used in the algorithms definition by using an `AD_Z` vector which contains set point values for consecutive output signals. Also, the output signal measurement can be found in the matrix `AD_Y`, where column index denotes the number of output signal and the row index denotes a time instant of the measurement. The variable `AD_K` denotes the current time instant. Lastly, the matrix `AD_U` can be used to access past control signals, again the column index denotes the number of input signal and the row index denotes a time instant of the control value. Analogously, the past control increments can be accessed with the use of variable `AD_DU`. It is important to know that the result of the algorithm, i.e. the final control increments vector, has to be stored in the special variable (horizontal vector) `control_value`. As the variables which name starts with `AD_` corresponds to the `ArchiveData` structure, the variable `control_value` corresponds to the `CurrentControl` structure.

It is important to finish the code that is to be used for transcompilation with a special comment `%% STOP` which defines the end of the algorithms definition.

Having the definition of the PID controller prepared, the same Keil uVision 5 project as the one used for the DMC Analytical algorithm, can be utilised. Therefore, a recompilation (which will also execute the transcompilation) results in the code presented in Appendix F. The obtained process trajectories are shown in Fig. 7, and profiler results are shown in table 6.

References

[1] E.F. Camacho and C. Bordons. *Model Predictive Control*. Springer, London, 1999.

Table 6: Profiler results obtained when the PID controller is used

id	time total	entries	min	mean	max	running
1 (hardware setup)	1	1	1	1.0000	1	0
2 (software setup)	1	1	1	1.0000	1	0
3 (controls' application)	8	1001	0	0.0079	1	0
4 (measurements)	65	1001	0	0.0649	1	0
10 (controls' calculation)	150	1001	0	0.1498	1	0
50 (control algorithm)	40	1001	0	0.0399	1	0
13 (other procedures)	7027	1001	6	7.0199	8	1

[2] P. Chaber and M. Ławryńczuk. Fast analytical model predictive controllers and their implementation for stm32 arm microcontroller. *IEEE Transactions on Industrial Informatics*. to appear.

[3] D.W. Clarke, C. Mohtadi, and P.S. Tuffs. Generalized predictive control. *Automatica*, 23:137–160, 1987.

[4] C.R. Cutler and B.L. Ramaker. Dynamic matrix control – a computer control algorithm. *Joint Automatic Control Conference*, 17:72, 1980.

[5] J.M. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, Harlow, 2001.

[6] A. Stachurski. *Wprowadzenie do optymalizacji*. Oficyna Wydawnicza Politechniki Warszawskiej, 2009.

[7] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. OSQP: An operator splitting solver for quadratic programs. *ArXiv e-prints*, November 2017.

[8] P. Tatjewski. *Advanced Control of Industrial Processes: Structures and Algorithms*. Springer, London, 2007.

A Analytical DMC Algorithm: Generated Source of `main_mpc.c`

```c
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\defines.h"
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\profiler.h"
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\mpctools.h"
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\simulated_signals.h"
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\matrix_cal.h"
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\allocation_nr.h"
#include "osqp.h"
#include "util.h"
#include "stm32f7xx_hal.h"
#include <string.h>
#include "main.h"

ArchiveData ad;
CurrentControl cc;

long get_time(){ return HAL_GetTick(); }

extern void timer_loop(void);
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){
    if (htim->Instance == TIM2) {
        timer_loop();
    }
}
void measurements(){
profiler_start(4);
    new_output(&ad,__measureOutput());
profiler_end(4);
}
void controls(){
profiler_start(3);
    __setControlValue(last_control(&ad));
profiler_end(3);
}
void hardware_setup(){
profiler_start(1);
    low_lvl_main();
profiler_end(1);
}

void controller(ArchiveData * ad, CurrentControl * c){
  static float** AutoMATiC_DMC_Y;
  static float** AutoMATiC_DMC_Yzad;
  static float** AutoMATiC_DMC_dUp;
  static float** AutoMATiC_DMC_du;
  static float** AutoMATiC_DMC_dutmp1;
  static float** AutoMATiC_DMC_dutmp2;
  static float** AutoMATiC_DMC_e;
  static long AutoMATiC_DMC_i;
  static long AutoMATiC_DMC_itmp;
  static long AutoMATiC_DMC_j;
  static long AutoMATiC_DMC_n;
  static float** AutoMATiC_DMC_tmpu;
  static float** control_value;
  static long j;
  static long k;
  static float** AutoMATiC_DMC_Ke;
  static float** AutoMATiC_DMC_Ku;
  static float** dumax;
  static float** dumin;
  static float** umax;
  static float** umin;
```

```
if(ad == NULL){
  AutoMATiC_DMC_dUp = darray(1,8,1,1);
  AutoMATiC_DMC_Yzad = darray(1,10,1,1);
  AutoMATiC_DMC_Y = darray(1,10,1,1);
  AutoMATiC_DMC_tmpu = darray(1,2,1,1);
  AutoMATiC_DMC_e = darray(1,2,1,1);
  control_value = darray(1,1,1,2);
  AutoMATiC_DMC_dutmp1 = darray(1,2,1,1);
  AutoMATiC_DMC_Ke = darray(1,2,1,2);
  AutoMATiC_DMC_Ke[1][1] = 3.373006e-01f;
  AutoMATiC_DMC_Ke[1][2] = -8.539666e-02f;
  AutoMATiC_DMC_Ke[2][1] = -1.016548e-01f;
  AutoMATiC_DMC_Ke[2][2] = 5.115693e-01f;
  AutoMATiC_DMC_dutmp2 = darray(1,2,1,1);
  AutoMATiC_DMC_Ku = darray(1,2,1,8);
  AutoMATiC_DMC_Ku[1][1] = 1.246690e+00f;
  AutoMATiC_DMC_Ku[1][2] = 1.325999e-01f;
  AutoMATiC_DMC_Ku[1][3] = 6.350099e-01f;
  AutoMATiC_DMC_Ku[1][4] = 6.239113e-02f;
  AutoMATiC_DMC_Ku[1][5] = 3.043741e-01f;
  AutoMATiC_DMC_Ku[1][6] = 2.660202e-02f;
  AutoMATiC_DMC_Ku[1][7] = 1.005321e-01f;
  AutoMATiC_DMC_Ku[1][8] = 7.749453e-03f;
  AutoMATiC_DMC_Ku[2][1] = 6.013931e-02f;
  AutoMATiC_DMC_Ku[2][2] = 8.674500e-01f;
  AutoMATiC_DMC_Ku[2][3] = -1.172849e-01f;
  AutoMATiC_DMC_Ku[2][4] = 2.406519e-01f;
  AutoMATiC_DMC_Ku[2][5] = -7.501822e-02f;
  AutoMATiC_DMC_Ku[2][6] = 6.519486e-02f;
  AutoMATiC_DMC_Ku[2][7] = -2.764289e-02f;
  AutoMATiC_DMC_Ku[2][8] = 1.396884e-02f;
  AutoMATiC_DMC_du = darray(1,2,1,1);
  dumax = darray(1,1,1,2);
  dumax[1][1] = 1.000000e-02f;
  dumax[1][2] = 1.000000e-02f;
  dumin = darray(1,1,1,2);
  dumin[1][1] = -1.000000e-02f;
  dumin[1][2] = -1.000000e-02f;
  umax = darray(1,1,1,2);
  umax[1][1] = 1.000000e+00f;
  umax[1][2] = 1.000000e+00f;
  umin = darray(1,1,1,2);
  umin[1][1] = -1.000000e+00f;
  umin[1][2] = -1.000000e+00f;
  return;
}
for(j=1;j<=8;++j) for(k=1;k<=1;++k) AutoMATiC_DMC_dUp[j][k] = 0;
for(j=1;j<=10;++j) for(k=1;k<=1;++k) AutoMATiC_DMC_Yzad[j][k] = 0;
for(j=1;j<=10;++j) for(k=1;k<=1;++k) AutoMATiC_DMC_Y[j][k] = 0;
for(j=1;j<=2;++j) for(k=1;k<=1;++k) AutoMATiC_DMC_tmpu[j][k] = 0;
AutoMATiC_DMC_itmp=1;
for(AutoMATiC_DMC_i=1;AutoMATiC_DMC_i<=4;++AutoMATiC_DMC_i){
  for(AutoMATiC_DMC_j=1;AutoMATiC_DMC_j<=2;++AutoMATiC_DMC_j){
    AutoMATiC_DMC_dUp[AutoMATiC_DMC_itmp][1]=ad->du[ad->k-AutoMATiC_DMC_i][
        AutoMATiC_DMC_j-1];
    AutoMATiC_DMC_itmp=AutoMATiC_DMC_itmp+1;
  }
}
AutoMATiC_DMC_itmp=1;
for(AutoMATiC_DMC_i=1;AutoMATiC_DMC_i<=5;++AutoMATiC_DMC_i){
  for(AutoMATiC_DMC_j=1;AutoMATiC_DMC_j<=2;++AutoMATiC_DMC_j){
    AutoMATiC_DMC_Yzad[AutoMATiC_DMC_itmp][1]=ad->z[AutoMATiC_DMC_j-1];
    AutoMATiC_DMC_itmp=AutoMATiC_DMC_itmp+1;
```

```
    }
  }
  AutoMATiC_DMC_itmp=1;
  for(AutoMATiC_DMC_i=1;AutoMATiC_DMC_i<=5;++AutoMATiC_DMC_i){
    for(AutoMATiC_DMC_j=1;AutoMATiC_DMC_j<=2;++AutoMATiC_DMC_j){
      AutoMATiC_DMC_Y[AutoMATiC_DMC_itmp][1]=ad->y[ad->k][AutoMATiC_DMC_j-1];
      AutoMATiC_DMC_itmp=AutoMATiC_DMC_itmp+1;
    }
  }
  for(j=1;j<=2;++j) for(k=1;k<=1;++k) AutoMATiC_DMC_e[j][k] = 0;
  for(AutoMATiC_DMC_i=1;AutoMATiC_DMC_i<=2;++AutoMATiC_DMC_i){
    AutoMATiC_DMC_e[AutoMATiC_DMC_i][1]=ad->z[AutoMATiC_DMC_i-1]-ad->y[ad->k][
        AutoMATiC_DMC_i-1];
  }
  productab(AutoMATiC_DMC_Ke,AutoMATiC_DMC_e,AutoMATiC_DMC_dutmp1,2,2,2,1);
  productab(AutoMATiC_DMC_Ku,AutoMATiC_DMC_dUp,AutoMATiC_DMC_dutmp2,2,8,8,1);
  sumaa(AutoMATiC_DMC_dutmp1,AutoMATiC_DMC_dutmp2,AutoMATiC_DMC_du,2,1,-1);
  for(AutoMATiC_DMC_n=1;AutoMATiC_DMC_n<=2;++AutoMATiC_DMC_n){
    if(AutoMATiC_DMC_du[AutoMATiC_DMC_n][1]>dumax[1][AutoMATiC_DMC_n]){
      AutoMATiC_DMC_du[AutoMATiC_DMC_n][1]=dumax[1][AutoMATiC_DMC_n];
    }
    if(AutoMATiC_DMC_du[AutoMATiC_DMC_n][1]<dumin[1][AutoMATiC_DMC_n]){
      AutoMATiC_DMC_du[AutoMATiC_DMC_n][1]=dumin[1][AutoMATiC_DMC_n];
    }
    AutoMATiC_DMC_tmpu[AutoMATiC_DMC_n][1]=ad->u[ad->k-1][AutoMATiC_DMC_n-1]+
        AutoMATiC_DMC_du[AutoMATiC_DMC_n][1];
    if(AutoMATiC_DMC_tmpu[AutoMATiC_DMC_n][1]>umax[1][AutoMATiC_DMC_n]){
      AutoMATiC_DMC_tmpu[AutoMATiC_DMC_n][1]=umax[1][AutoMATiC_DMC_n];
    }
    if(AutoMATiC_DMC_tmpu[AutoMATiC_DMC_n][1]<umin[1][AutoMATiC_DMC_n]){
      AutoMATiC_DMC_tmpu[AutoMATiC_DMC_n][1]=umin[1][AutoMATiC_DMC_n];
    }
    AutoMATiC_DMC_du[AutoMATiC_DMC_n][1]=AutoMATiC_DMC_tmpu[AutoMATiC_DMC_n
        ][1]-ad->u[ad->k-1][AutoMATiC_DMC_n-1];
  }
  for(j=1;j<=1;++j) for(k=1;k<=2;++k) control_value[j][k] = 0;
  for(AutoMATiC_DMC_n=1;AutoMATiC_DMC_n<=2;++AutoMATiC_DMC_n){
    control_value[1][AutoMATiC_DMC_n]=AutoMATiC_DMC_du[AutoMATiC_DMC_n][1];
  }

  set_current_control_increment(c,&(control_value[1][1])); // du is indexed
      starting with 1, therefore to maintain compatibility it is required to
      refer to first element of an actual array
}

void controller_setup(){
profiler_start(2);
    init_archive_data(&ad, 200, 200, 2, 2, 0, 0, 0.01);
    init_current_control(&cc,&ad);
    controller(NULL,NULL);
profiler_end(2);
}

void idle(){
profiler_start(13);
    const int k = 0;
  static int i = 0;
  static char str[1000] = {0};

    sprintf(str, "x = [%f,%f,",ad.y[k][0], ad.y[k][1]);     write_string(str);
    sprintf(str,       "%f,%f,",ad.z[0], ad.z[1]);          write_string(str);
    sprintf(str,       "%f,%f,",ad.u[k-1][0],ad.u[k-1][1]); write_string(str);
    write_string("];\n\r");
```

```
  if(++i > 1000) profiler_print();
profiler_end(13);
}

void loop(){
profiler_start(10);
    static int i = 0;
       if(i< 100){ ad.z[0] = -0.1f; ad.z[1] = +0.2f; }
  else if(i< 150){ ad.z[0] = -0.1f; ad.z[1] = -0.2f; }
  else if(i< 250){ ad.z[0] = +0.1f; ad.z[1] = -0.2f; }
  else           { ad.z[0] = +0.1f; ad.z[1] = +0.2f; }
  if(++i > 300) i = 0;

profiler_start(50);
  controller(&ad,&cc);
profiler_end(50);

  push_current_controls_to_archive_data(&cc,&ad);
profiler_end(10);
}

void timeout(){
    while(1);
}
```

B Numerical DMC Algorithm: Generated Source of `main_mpc.c`

```
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\defines.h"
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\profiler.h"
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\mpctools.h"
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\simulated_signals.h"
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\matrix_cal.h"
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\allocation_nr.h"
#include "osqp.h"
#include "util.h"
#include "stm32f7xx_hal.h"
#include <string.h>
#include "main.h"

ArchiveData ad;
CurrentControl cc;

long get_time(){ return HAL_GetTick(); }

extern void timer_loop(void);
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){
    if (htim->Instance == TIM2) {
        timer_loop();
    }
}
void measurements(){
profiler_start(4);
    new_output(&ad,__measureOutput());
profiler_end(4);
}
void controls(){
profiler_start(3);
    __setControlValue(last_control(&ad));
profiler_end(3);
}
void hardware_setup(){
profiler_start(1);
    low_lvl_main();
profiler_end(1);
}

void controller(ArchiveData * ad, CurrentControl * c){
  static float** AutoMATiC_DMC_Y;
  static float** AutoMATiC_DMC_Yzad;
  static float** AutoMATiC_DMC_btmp1;
  static float** AutoMATiC_DMC_btmp2;
  static float** AutoMATiC_DMC_dUp;
  static float** AutoMATiC_DMC_du;
  static float** AutoMATiC_DMC_dutmp1;
  static float** AutoMATiC_DMC_dutmp2;
  static float** AutoMATiC_DMC_e;
  static float** AutoMATiC_DMC_ftmp1;
  static float** AutoMATiC_DMC_ftmp2;
  static float** AutoMATiC_DMC_ftmp3;
  static float** AutoMATiC_DMC_ftmp4;
  static long AutoMATiC_DMC_i;
  static long AutoMATiC_DMC_itmp;
  static long AutoMATiC_DMC_j;
  static long AutoMATiC_DMC_n;
  static float** AutoMATiC_DMC_tmpu;
  static float** AutoMATiC_DMC_uk;
  static float** control_value;
  static long j;
```

```
static long k;
static float** AutoMATiC_DMC_Ke;
static float** AutoMATiC_DMC_Ku;
static float** AutoMATiC_DMC_Mp;
static float** AutoMATiC_DMC_fconst;
static float** AutoMATiC_DMC_bvar;
static float** AutoMATiC_DMC_b;
static float** dumax;
static float** dumin;
static float** umax;
static float** umin;
static c_float AutoMATiC_DMC_A_x[80];
static c_int AutoMATiC_DMC_A_i[80];
static c_int AutoMATiC_DMC_A_p[11];

static c_float AutoMATiC_DMC_H_x[66];
static c_int AutoMATiC_DMC_H_i[66];
static c_int AutoMATiC_DMC_H_p[11];

static OSQPSettings *osqp_settings;
static OSQPWorkspace *osqp_work;
static OSQPData *osqp_data;
static c_float osqp_q[10];
static c_float osqp_lb[40];
static c_float osqp_ub[40];
static c_float AutoMATiC_DMC_ftmp[11];
static c_float AutoMATiC_DMC_btmp[41];
static c_float AutoMATiC_DMC_qpx[11];
if(ad == NULL){
  AutoMATiC_DMC_A_x[0] = -1.000000e+00f;
  AutoMATiC_DMC_A_x[1] = -1.000000e+00f;
  AutoMATiC_DMC_A_x[2] = -1.000000e+00f;
  AutoMATiC_DMC_A_x[3] = -1.000000e+00f;
  AutoMATiC_DMC_A_x[4] = -1.000000e+00f;
  AutoMATiC_DMC_A_x[5] = -1.000000e+00f;
  AutoMATiC_DMC_A_x[6] = 1.000000e+00f;
  AutoMATiC_DMC_A_x[7] = 1.000000e+00f;
  AutoMATiC_DMC_A_x[8] = 1.000000e+00f;
  AutoMATiC_DMC_A_x[9] = 1.000000e+00f;
  AutoMATiC_DMC_A_x[10] = 1.000000e+00f;
  AutoMATiC_DMC_A_x[11] = 1.000000e+00f;
  AutoMATiC_DMC_A_x[12] = -1.000000e+00f;
  AutoMATiC_DMC_A_x[13] = -1.000000e+00f;
  AutoMATiC_DMC_A_x[14] = -1.000000e+00f;
  AutoMATiC_DMC_A_x[15] = -1.000000e+00f;
  AutoMATiC_DMC_A_x[16] = -1.000000e+00f;
  AutoMATiC_DMC_A_x[17] = -1.000000e+00f;
  AutoMATiC_DMC_A_x[18] = 1.000000e+00f;
  AutoMATiC_DMC_A_x[19] = 1.000000e+00f;
  AutoMATiC_DMC_A_x[20] = 1.000000e+00f;
  AutoMATiC_DMC_A_x[21] = 1.000000e+00f;
  AutoMATiC_DMC_A_x[22] = 1.000000e+00f;
  AutoMATiC_DMC_A_x[23] = 1.000000e+00f;
  AutoMATiC_DMC_A_x[24] = -1.000000e+00f;
  AutoMATiC_DMC_A_x[25] = -1.000000e+00f;
  AutoMATiC_DMC_A_x[26] = -1.000000e+00f;
  AutoMATiC_DMC_A_x[27] = -1.000000e+00f;
  AutoMATiC_DMC_A_x[28] = -1.000000e+00f;
  AutoMATiC_DMC_A_x[29] = 1.000000e+00f;
  AutoMATiC_DMC_A_x[30] = 1.000000e+00f;
  AutoMATiC_DMC_A_x[31] = 1.000000e+00f;
  AutoMATiC_DMC_A_x[32] = 1.000000e+00f;
  AutoMATiC_DMC_A_x[33] = 1.000000e+00f;
```

```
AutoMATiC_DMC_A_x[34] = -1.000000e+00f;
AutoMATiC_DMC_A_x[35] = -1.000000e+00f;
AutoMATiC_DMC_A_x[36] = -1.000000e+00f;
AutoMATiC_DMC_A_x[37] = -1.000000e+00f;
AutoMATiC_DMC_A_x[38] = -1.000000e+00f;
AutoMATiC_DMC_A_x[39] = 1.000000e+00f;
AutoMATiC_DMC_A_x[40] = 1.000000e+00f;
AutoMATiC_DMC_A_x[41] = 1.000000e+00f;
AutoMATiC_DMC_A_x[42] = 1.000000e+00f;
AutoMATiC_DMC_A_x[43] = 1.000000e+00f;
AutoMATiC_DMC_A_x[44] = -1.000000e+00f;
AutoMATiC_DMC_A_x[45] = -1.000000e+00f;
AutoMATiC_DMC_A_x[46] = -1.000000e+00f;
AutoMATiC_DMC_A_x[47] = -1.000000e+00f;
AutoMATiC_DMC_A_x[48] = 1.000000e+00f;
AutoMATiC_DMC_A_x[49] = 1.000000e+00f;
AutoMATiC_DMC_A_x[50] = 1.000000e+00f;
AutoMATiC_DMC_A_x[51] = 1.000000e+00f;
AutoMATiC_DMC_A_x[52] = -1.000000e+00f;
AutoMATiC_DMC_A_x[53] = -1.000000e+00f;
AutoMATiC_DMC_A_x[54] = -1.000000e+00f;
AutoMATiC_DMC_A_x[55] = -1.000000e+00f;
AutoMATiC_DMC_A_x[56] = 1.000000e+00f;
AutoMATiC_DMC_A_x[57] = 1.000000e+00f;
AutoMATiC_DMC_A_x[58] = 1.000000e+00f;
AutoMATiC_DMC_A_x[59] = 1.000000e+00f;
AutoMATiC_DMC_A_x[60] = -1.000000e+00f;
AutoMATiC_DMC_A_x[61] = -1.000000e+00f;
AutoMATiC_DMC_A_x[62] = -1.000000e+00f;
AutoMATiC_DMC_A_x[63] = 1.000000e+00f;
AutoMATiC_DMC_A_x[64] = 1.000000e+00f;
AutoMATiC_DMC_A_x[65] = 1.000000e+00f;
AutoMATiC_DMC_A_x[66] = -1.000000e+00f;
AutoMATiC_DMC_A_x[67] = -1.000000e+00f;
AutoMATiC_DMC_A_x[68] = -1.000000e+00f;
AutoMATiC_DMC_A_x[69] = 1.000000e+00f;
AutoMATiC_DMC_A_x[70] = 1.000000e+00f;
AutoMATiC_DMC_A_x[71] = 1.000000e+00f;
AutoMATiC_DMC_A_x[72] = -1.000000e+00f;
AutoMATiC_DMC_A_x[73] = -1.000000e+00f;
AutoMATiC_DMC_A_x[74] = 1.000000e+00f;
AutoMATiC_DMC_A_x[75] = 1.000000e+00f;
AutoMATiC_DMC_A_x[76] = -1.000000e+00f;
AutoMATiC_DMC_A_x[77] = -1.000000e+00f;
AutoMATiC_DMC_A_x[78] = 1.000000e+00f;
AutoMATiC_DMC_A_x[79] = 1.000000e+00f;
AutoMATiC_DMC_A_i[0] = 0;
AutoMATiC_DMC_A_i[1] = 10;
AutoMATiC_DMC_A_i[2] = 12;
AutoMATiC_DMC_A_i[3] = 14;
AutoMATiC_DMC_A_i[4] = 16;
AutoMATiC_DMC_A_i[5] = 18;
AutoMATiC_DMC_A_i[6] = 20;
AutoMATiC_DMC_A_i[7] = 30;
AutoMATiC_DMC_A_i[8] = 32;
AutoMATiC_DMC_A_i[9] = 34;
AutoMATiC_DMC_A_i[10] = 36;
AutoMATiC_DMC_A_i[11] = 38;
AutoMATiC_DMC_A_i[12] = 1;
AutoMATiC_DMC_A_i[13] = 11;
AutoMATiC_DMC_A_i[14] = 13;
AutoMATiC_DMC_A_i[15] = 15;
AutoMATiC_DMC_A_i[16] = 17;
```

```
AutoMATiC_DMC_A_i[17] = 19;
AutoMATiC_DMC_A_i[18] = 21;
AutoMATiC_DMC_A_i[19] = 31;
AutoMATiC_DMC_A_i[20] = 33;
AutoMATiC_DMC_A_i[21] = 35;
AutoMATiC_DMC_A_i[22] = 37;
AutoMATiC_DMC_A_i[23] = 39;
AutoMATiC_DMC_A_i[24] = 2;
AutoMATiC_DMC_A_i[25] = 12;
AutoMATiC_DMC_A_i[26] = 14;
AutoMATiC_DMC_A_i[27] = 16;
AutoMATiC_DMC_A_i[28] = 18;
AutoMATiC_DMC_A_i[29] = 22;
AutoMATiC_DMC_A_i[30] = 32;
AutoMATiC_DMC_A_i[31] = 34;
AutoMATiC_DMC_A_i[32] = 36;
AutoMATiC_DMC_A_i[33] = 38;
AutoMATiC_DMC_A_i[34] = 3;
AutoMATiC_DMC_A_i[35] = 13;
AutoMATiC_DMC_A_i[36] = 15;
AutoMATiC_DMC_A_i[37] = 17;
AutoMATiC_DMC_A_i[38] = 19;
AutoMATiC_DMC_A_i[39] = 23;
AutoMATiC_DMC_A_i[40] = 33;
AutoMATiC_DMC_A_i[41] = 35;
AutoMATiC_DMC_A_i[42] = 37;
AutoMATiC_DMC_A_i[43] = 39;
AutoMATiC_DMC_A_i[44] = 4;
AutoMATiC_DMC_A_i[45] = 14;
AutoMATiC_DMC_A_i[46] = 16;
AutoMATiC_DMC_A_i[47] = 18;
AutoMATiC_DMC_A_i[48] = 24;
AutoMATiC_DMC_A_i[49] = 34;
AutoMATiC_DMC_A_i[50] = 36;
AutoMATiC_DMC_A_i[51] = 38;
AutoMATiC_DMC_A_i[52] = 5;
AutoMATiC_DMC_A_i[53] = 15;
AutoMATiC_DMC_A_i[54] = 17;
AutoMATiC_DMC_A_i[55] = 19;
AutoMATiC_DMC_A_i[56] = 25;
AutoMATiC_DMC_A_i[57] = 35;
AutoMATiC_DMC_A_i[58] = 37;
AutoMATiC_DMC_A_i[59] = 39;
AutoMATiC_DMC_A_i[60] = 6;
AutoMATiC_DMC_A_i[61] = 16;
AutoMATiC_DMC_A_i[62] = 18;
AutoMATiC_DMC_A_i[63] = 26;
AutoMATiC_DMC_A_i[64] = 36;
AutoMATiC_DMC_A_i[65] = 38;
AutoMATiC_DMC_A_i[66] = 7;
AutoMATiC_DMC_A_i[67] = 17;
AutoMATiC_DMC_A_i[68] = 19;
AutoMATiC_DMC_A_i[69] = 27;
AutoMATiC_DMC_A_i[70] = 37;
AutoMATiC_DMC_A_i[71] = 39;
AutoMATiC_DMC_A_i[72] = 8;
AutoMATiC_DMC_A_i[73] = 18;
AutoMATiC_DMC_A_i[74] = 28;
AutoMATiC_DMC_A_i[75] = 38;
AutoMATiC_DMC_A_i[76] = 9;
AutoMATiC_DMC_A_i[77] = 19;
AutoMATiC_DMC_A_i[78] = 29;
AutoMATiC_DMC_A_i[79] = 39;
```

```
AutoMATiC_DMC_A_p[0] = 0;
AutoMATiC_DMC_A_p[1] = 12;
AutoMATiC_DMC_A_p[2] = 24;
AutoMATiC_DMC_A_p[3] = 34;
AutoMATiC_DMC_A_p[4] = 44;
AutoMATiC_DMC_A_p[5] = 52;
AutoMATiC_DMC_A_p[6] = 60;
AutoMATiC_DMC_A_p[7] = 66;
AutoMATiC_DMC_A_p[8] = 72;
AutoMATiC_DMC_A_p[9] = 76;
AutoMATiC_DMC_A_p[10] = 80;

AutoMATiC_DMC_H_x[0] = 1.344566e+02f;
AutoMATiC_DMC_H_x[1] = 4.114478e+01f;
AutoMATiC_DMC_H_x[2] = 1.000269e+02f;
AutoMATiC_DMC_H_x[3] = 3.169500e+01f;
AutoMATiC_DMC_H_x[4] = 6.190011e+01f;
AutoMATiC_DMC_H_x[5] = 2.023139e+01f;
AutoMATiC_DMC_H_x[6] = 2.567725e+01f;
AutoMATiC_DMC_H_x[7] = 8.808511e+00f;
AutoMATiC_DMC_H_x[8] = 4.114478e+01f;
AutoMATiC_DMC_H_x[9] = 3.436150e+01f;
AutoMATiC_DMC_H_x[10] = 3.099487e+01f;
AutoMATiC_DMC_H_x[11] = 2.477126e+01f;
AutoMATiC_DMC_H_x[12] = 1.931418e+01f;
AutoMATiC_DMC_H_x[13] = 1.576652e+01f;
AutoMATiC_DMC_H_x[14] = 8.233726e+00f;
AutoMATiC_DMC_H_x[15] = 6.910587e+00f;
AutoMATiC_DMC_H_x[16] = 1.000269e+02f;
AutoMATiC_DMC_H_x[17] = 3.099487e+01f;
AutoMATiC_DMC_H_x[18] = 8.804002e+01f;
AutoMATiC_DMC_H_x[19] = 2.792375e+01f;
AutoMATiC_DMC_H_x[20] = 5.644363e+01f;
AutoMATiC_DMC_H_x[21] = 1.883764e+01f;
AutoMATiC_DMC_H_x[22] = 2.413991e+01f;
AutoMATiC_DMC_H_x[23] = 8.414358e+00f;
AutoMATiC_DMC_H_x[24] = 3.169500e+01f;
AutoMATiC_DMC_H_x[25] = 2.477126e+01f;
AutoMATiC_DMC_H_x[26] = 2.792375e+01f;
AutoMATiC_DMC_H_x[27] = 2.454154e+01f;
AutoMATiC_DMC_H_x[28] = 1.838345e+01f;
AutoMATiC_DMC_H_x[29] = 1.514642e+01f;
AutoMATiC_DMC_H_x[30] = 8.018635e+00f;
AutoMATiC_DMC_H_x[31] = 6.775021e+00f;
AutoMATiC_DMC_H_x[32] = 6.190011e+01f;
AutoMATiC_DMC_H_x[33] = 1.931418e+01f;
AutoMATiC_DMC_H_x[34] = 5.644363e+01f;
AutoMATiC_DMC_H_x[35] = 1.838345e+01f;
AutoMATiC_DMC_H_x[36] = 4.711061e+01f;
AutoMATiC_DMC_H_x[37] = 1.565679e+01f;
AutoMATiC_DMC_H_x[38] = 2.097070e+01f;
AutoMATiC_DMC_H_x[39] = 7.558490e+00f;
AutoMATiC_DMC_H_x[40] = 2.023139e+01f;
AutoMATiC_DMC_H_x[41] = 1.576652e+01f;
AutoMATiC_DMC_H_x[42] = 1.883764e+01f;
AutoMATiC_DMC_H_x[43] = 1.514642e+01f;
AutoMATiC_DMC_H_x[44] = 1.565679e+01f;
AutoMATiC_DMC_H_x[45] = 1.510759e+01f;
AutoMATiC_DMC_H_x[46] = 7.391926e+00f;
AutoMATiC_DMC_H_x[47] = 6.332568e+00f;
AutoMATiC_DMC_H_x[48] = 2.567725e+01f;
AutoMATiC_DMC_H_x[49] = 8.233726e+00f;
AutoMATiC_DMC_H_x[50] = 2.413991e+01f;
```

```
AutoMATiC_DMC_H_x[51] = 8.018635e+00f;
AutoMATiC_DMC_H_x[52] = 2.097070e+01f;
AutoMATiC_DMC_H_x[53] = 7.391926e+00f;
AutoMATiC_DMC_H_x[54] = 1.634423e+01f;
AutoMATiC_DMC_H_x[55] = 5.541646e+00f;
AutoMATiC_DMC_H_x[56] = 8.808511e+00f;
AutoMATiC_DMC_H_x[57] = 6.910587e+00f;
AutoMATiC_DMC_H_x[58] = 8.414358e+00f;
AutoMATiC_DMC_H_x[59] = 6.775021e+00f;
AutoMATiC_DMC_H_x[60] = 7.558490e+00f;
AutoMATiC_DMC_H_x[61] = 6.332568e+00f;
AutoMATiC_DMC_H_x[62] = 5.541646e+00f;
AutoMATiC_DMC_H_x[63] = 6.871756e+00f;
AutoMATiC_DMC_H_x[64] = 2.000000e+00f;
AutoMATiC_DMC_H_x[65] = 2.000000e+00f;
AutoMATiC_DMC_H_i[0] = 0;
AutoMATiC_DMC_H_i[1] = 1;
AutoMATiC_DMC_H_i[2] = 2;
AutoMATiC_DMC_H_i[3] = 3;
AutoMATiC_DMC_H_i[4] = 4;
AutoMATiC_DMC_H_i[5] = 5;
AutoMATiC_DMC_H_i[6] = 6;
AutoMATiC_DMC_H_i[7] = 7;
AutoMATiC_DMC_H_i[8] = 0;
AutoMATiC_DMC_H_i[9] = 1;
AutoMATiC_DMC_H_i[10] = 2;
AutoMATiC_DMC_H_i[11] = 3;
AutoMATiC_DMC_H_i[12] = 4;
AutoMATiC_DMC_H_i[13] = 5;
AutoMATiC_DMC_H_i[14] = 6;
AutoMATiC_DMC_H_i[15] = 7;
AutoMATiC_DMC_H_i[16] = 0;
AutoMATiC_DMC_H_i[17] = 1;
AutoMATiC_DMC_H_i[18] = 2;
AutoMATiC_DMC_H_i[19] = 3;
AutoMATiC_DMC_H_i[20] = 4;
AutoMATiC_DMC_H_i[21] = 5;
AutoMATiC_DMC_H_i[22] = 6;
AutoMATiC_DMC_H_i[23] = 7;
AutoMATiC_DMC_H_i[24] = 0;
AutoMATiC_DMC_H_i[25] = 1;
AutoMATiC_DMC_H_i[26] = 2;
AutoMATiC_DMC_H_i[27] = 3;
AutoMATiC_DMC_H_i[28] = 4;
AutoMATiC_DMC_H_i[29] = 5;
AutoMATiC_DMC_H_i[30] = 6;
AutoMATiC_DMC_H_i[31] = 7;
AutoMATiC_DMC_H_i[32] = 0;
AutoMATiC_DMC_H_i[33] = 1;
AutoMATiC_DMC_H_i[34] = 2;
AutoMATiC_DMC_H_i[35] = 3;
AutoMATiC_DMC_H_i[36] = 4;
AutoMATiC_DMC_H_i[37] = 5;
AutoMATiC_DMC_H_i[38] = 6;
AutoMATiC_DMC_H_i[39] = 7;
AutoMATiC_DMC_H_i[40] = 0;
AutoMATiC_DMC_H_i[41] = 1;
AutoMATiC_DMC_H_i[42] = 2;
AutoMATiC_DMC_H_i[43] = 3;
AutoMATiC_DMC_H_i[44] = 4;
AutoMATiC_DMC_H_i[45] = 5;
AutoMATiC_DMC_H_i[46] = 6;
AutoMATiC_DMC_H_i[47] = 7;
```

```
AutoMATiC_DMC_H_i[48] = 0;
AutoMATiC_DMC_H_i[49] = 1;
AutoMATiC_DMC_H_i[50] = 2;
AutoMATiC_DMC_H_i[51] = 3;
AutoMATiC_DMC_H_i[52] = 4;
AutoMATiC_DMC_H_i[53] = 5;
AutoMATiC_DMC_H_i[54] = 6;
AutoMATiC_DMC_H_i[55] = 7;
AutoMATiC_DMC_H_i[56] = 0;
AutoMATiC_DMC_H_i[57] = 1;
AutoMATiC_DMC_H_i[58] = 2;
AutoMATiC_DMC_H_i[59] = 3;
AutoMATiC_DMC_H_i[60] = 4;
AutoMATiC_DMC_H_i[61] = 5;
AutoMATiC_DMC_H_i[62] = 6;
AutoMATiC_DMC_H_i[63] = 7;
AutoMATiC_DMC_H_i[64] = 8;
AutoMATiC_DMC_H_i[65] = 9;
AutoMATiC_DMC_H_p[0] = 0;
AutoMATiC_DMC_H_p[1] = 8;
AutoMATiC_DMC_H_p[2] = 16;
AutoMATiC_DMC_H_p[3] = 24;
AutoMATiC_DMC_H_p[4] = 32;
AutoMATiC_DMC_H_p[5] = 40;
AutoMATiC_DMC_H_p[6] = 48;
AutoMATiC_DMC_H_p[7] = 56;
AutoMATiC_DMC_H_p[8] = 64;
AutoMATiC_DMC_H_p[9] = 65;
AutoMATiC_DMC_H_p[10] = 66;

osqp_settings = (OSQPSettings *)c_malloc(sizeof(OSQPSettings));
osqp_data = (OSQPData *)c_malloc(sizeof(OSQPData));
for(int osqp_it=0; osqp_it < 40; ++osqp_it) osqp_lb[osqp_it] = -100000;
osqp_data->n = 10;
osqp_data->m = 40;
osqp_data->q = osqp_q;
osqp_data->l = osqp_lb;
osqp_data->u = osqp_ub;
osqp_data->P = csc_matrix(osqp_data->n, osqp_data->n, 66, AutoMATiC_DMC_H_x
    , AutoMATiC_DMC_H_i, AutoMATiC_DMC_H_p);
osqp_data->A = csc_matrix(osqp_data->m, osqp_data->n, 80, AutoMATiC_DMC_A_x
    , AutoMATiC_DMC_A_i, AutoMATiC_DMC_A_p);
osqp_set_default_settings(osqp_settings);
osqp_work = osqp_setup(osqp_data, osqp_settings);
AutoMATiC_DMC_dUp = darray(1,8,1,1);
AutoMATiC_DMC_Yzad = darray(1,10,1,1);
AutoMATiC_DMC_Y = darray(1,10,1,1);
AutoMATiC_DMC_tmpu = darray(1,2,1,1);
AutoMATiC_DMC_e = darray(1,2,1,1);
AutoMATiC_DMC_du = darray(1,2,1,1);
AutoMATiC_DMC_uk = darray(1,2,1,1);
control_value = darray(1,1,1,2);
AutoMATiC_DMC_dutmp1 = darray(1,2,1,1);
AutoMATiC_DMC_Ke = darray(1,2,1,2);
AutoMATiC_DMC_Ke[1][1] = 3.373006e-01f;
AutoMATiC_DMC_Ke[1][2] = -8.539666e-02f;
AutoMATiC_DMC_Ke[2][1] = -1.016548e-01f;
AutoMATiC_DMC_Ke[2][2] = 5.115693e-01f;
AutoMATiC_DMC_dutmp2 = darray(1,2,1,1);
AutoMATiC_DMC_Ku = darray(1,2,1,8);
AutoMATiC_DMC_Ku[1][1] = 1.246690e+00f;
AutoMATiC_DMC_Ku[1][2] = 1.325999e-01f;
AutoMATiC_DMC_Ku[1][3] = 6.350099e-01f;
```

```
AutoMATiC_DMC_Ku[1][4] = 6.239113e-02f;
AutoMATiC_DMC_Ku[1][5] = 3.043741e-01f;
AutoMATiC_DMC_Ku[1][6] = 2.660202e-02f;
AutoMATiC_DMC_Ku[1][7] = 1.005321e-01f;
AutoMATiC_DMC_Ku[1][8] = 7.749453e-03f;
AutoMATiC_DMC_Ku[2][1] = 6.013931e-02f;
AutoMATiC_DMC_Ku[2][2] = 8.674500e-01f;
AutoMATiC_DMC_Ku[2][3] = -1.172849e-01f;
AutoMATiC_DMC_Ku[2][4] = 2.406519e-01f;
AutoMATiC_DMC_Ku[2][5] = -7.501822e-02f;
AutoMATiC_DMC_Ku[2][6] = 6.519486e-02f;
AutoMATiC_DMC_Ku[2][7] = -2.764289e-02f;
AutoMATiC_DMC_Ku[2][8] = 1.396884e-02f;
AutoMATiC_DMC_ftmp1 = darray(1,10,1,1);
AutoMATiC_DMC_ftmp2 = darray(1,10,1,1);
AutoMATiC_DMC_Mp = darray(1,10,1,8);
AutoMATiC_DMC_Mp[1][1] = 2.552292e+00f;
AutoMATiC_DMC_Mp[1][2] = 6.321206e-01f;
AutoMATiC_DMC_Mp[1][3] = 1.249453e+00f;
AutoMATiC_DMC_Mp[1][4] = 2.325442e-01f;
AutoMATiC_DMC_Mp[1][5] = 6.116594e-01f;
AutoMATiC_DMC_Mp[1][6] = 8.554821e-02f;
AutoMATiC_DMC_Mp[1][7] = 2.994327e-01f;
AutoMATiC_DMC_Mp[1][8] = 3.147143e-02f;
AutoMATiC_DMC_Mp[2][1] = 8.111244e-01f;
AutoMATiC_DMC_Mp[2][2] = 1.426990e+00f;
AutoMATiC_DMC_Mp[2][3] = 1.532016e-01f;
AutoMATiC_DMC_Mp[2][4] = 4.088396e-01f;
AutoMATiC_DMC_Mp[2][5] = 2.893605e-02f;
AutoMATiC_DMC_Mp[2][6] = 1.171345e-01f;
AutoMATiC_DMC_Mp[2][7] = 5.465313e-03f;
AutoMATiC_DMC_Mp[2][8] = 3.355960e-02f;
AutoMATiC_DMC_Mp[3][1] = 3.801745e+00f;
AutoMATiC_DMC_Mp[3][2] = 8.646647e-01f;
AutoMATiC_DMC_Mp[3][3] = 1.861112e+00f;
AutoMATiC_DMC_Mp[3][4] = 3.180924e-01f;
AutoMATiC_DMC_Mp[3][5] = 9.110921e-01f;
AutoMATiC_DMC_Mp[3][6] = 1.170196e-01f;
AutoMATiC_DMC_Mp[3][7] = 2.994327e-01f;
AutoMATiC_DMC_Mp[3][8] = 3.147143e-02f;
AutoMATiC_DMC_Mp[4][1] = 9.643260e-01f;
AutoMATiC_DMC_Mp[4][2] = 1.835830e+00f;
AutoMATiC_DMC_Mp[4][3] = 1.821377e-01f;
AutoMATiC_DMC_Mp[4][4] = 5.259741e-01f;
AutoMATiC_DMC_Mp[4][5] = 3.440136e-02f;
AutoMATiC_DMC_Mp[4][6] = 1.506941e-01f;
AutoMATiC_DMC_Mp[4][7] = 5.465313e-03f;
AutoMATiC_DMC_Mp[4][8] = 3.355960e-02f;
AutoMATiC_DMC_Mp[5][1] = 4.413404e+00f;
AutoMATiC_DMC_Mp[5][2] = 9.502129e-01f;
AutoMATiC_DMC_Mp[5][3] = 2.160545e+00f;
AutoMATiC_DMC_Mp[5][4] = 3.495638e-01f;
AutoMATiC_DMC_Mp[5][5] = 9.110921e-01f;
AutoMATiC_DMC_Mp[5][6] = 1.170196e-01f;
AutoMATiC_DMC_Mp[5][7] = 2.994327e-01f;
AutoMATiC_DMC_Mp[5][8] = 3.147143e-02f;
AutoMATiC_DMC_Mp[6][1] = 9.932621e-01f;
AutoMATiC_DMC_Mp[6][2] = 1.952965e+00f;
AutoMATiC_DMC_Mp[6][3] = 1.876030e-01f;
AutoMATiC_DMC_Mp[6][4] = 5.595337e-01f;
AutoMATiC_DMC_Mp[6][5] = 3.440136e-02f;
AutoMATiC_DMC_Mp[6][6] = 1.506941e-01f;
AutoMATiC_DMC_Mp[6][7] = 5.465313e-03f;
```

```
AutoMATiC_DMC_Mp[6][8] = 3.355960e-02f;
AutoMATiC_DMC_Mp[7][1] = 4.712837e+00f;
AutoMATiC_DMC_Mp[7][2] = 9.816844e-01f;
AutoMATiC_DMC_Mp[7][3] = 2.160545e+00f;
AutoMATiC_DMC_Mp[7][4] = 3.495638e-01f;
AutoMATiC_DMC_Mp[7][5] = 9.110921e-01f;
AutoMATiC_DMC_Mp[7][6] = 1.170196e-01f;
AutoMATiC_DMC_Mp[7][7] = 2.994327e-01f;
AutoMATiC_DMC_Mp[7][8] = 3.147143e-02f;
AutoMATiC_DMC_Mp[8][1] = 9.987274e-01f;
AutoMATiC_DMC_Mp[8][2] = 1.986524e+00f;
AutoMATiC_DMC_Mp[8][3] = 1.876030e-01f;
AutoMATiC_DMC_Mp[8][4] = 5.595337e-01f;
AutoMATiC_DMC_Mp[8][5] = 3.440136e-02f;
AutoMATiC_DMC_Mp[8][6] = 1.506941e-01f;
AutoMATiC_DMC_Mp[8][7] = 5.465313e-03f;
AutoMATiC_DMC_Mp[8][8] = 3.355960e-02f;
AutoMATiC_DMC_Mp[9][1] = 4.712837e+00f;
AutoMATiC_DMC_Mp[9][2] = 9.816844e-01f;
AutoMATiC_DMC_Mp[9][3] = 2.160545e+00f;
AutoMATiC_DMC_Mp[9][4] = 3.495638e-01f;
AutoMATiC_DMC_Mp[9][5] = 9.110921e-01f;
AutoMATiC_DMC_Mp[9][6] = 1.170196e-01f;
AutoMATiC_DMC_Mp[9][7] = 2.994327e-01f;
AutoMATiC_DMC_Mp[9][8] = 3.147143e-02f;
AutoMATiC_DMC_Mp[10][1] = 9.987274e-01f;
AutoMATiC_DMC_Mp[10][2] = 1.986524e+00f;
AutoMATiC_DMC_Mp[10][3] = 1.876030e-01f;
AutoMATiC_DMC_Mp[10][4] = 5.595337e-01f;
AutoMATiC_DMC_Mp[10][5] = 3.440136e-02f;
AutoMATiC_DMC_Mp[10][6] = 1.506941e-01f;
AutoMATiC_DMC_Mp[10][7] = 5.465313e-03f;
AutoMATiC_DMC_Mp[10][8] = 3.355960e-02f;
AutoMATiC_DMC_ftmp3 = darray(1,10,1,1);
AutoMATiC_DMC_ftmp4 = darray(1,10,1,1);
AutoMATiC_DMC_fconst = darray(1,10,1,10);
AutoMATiC_DMC_fconst[1][1] = 0.000000e+00f;
AutoMATiC_DMC_fconst[1][2] = 0.000000e+00f;
AutoMATiC_DMC_fconst[1][3] = -5.104583e+00f;
AutoMATiC_DMC_fconst[1][4] = -1.622249e+00f;
AutoMATiC_DMC_fconst[1][5] = -7.603490e+00f;
AutoMATiC_DMC_fconst[1][6] = -1.928652e+00f;
AutoMATiC_DMC_fconst[1][7] = -8.826808e+00f;
AutoMATiC_DMC_fconst[1][8] = -1.986524e+00f;
AutoMATiC_DMC_fconst[1][9] = -9.425674e+00f;
AutoMATiC_DMC_fconst[1][10] = -1.997455e+00f;
AutoMATiC_DMC_fconst[2][1] = 0.000000e+00f;
AutoMATiC_DMC_fconst[2][2] = 0.000000e+00f;
AutoMATiC_DMC_fconst[2][3] = -1.264241e+00f;
AutoMATiC_DMC_fconst[2][4] = -2.853981e+00f;
AutoMATiC_DMC_fconst[2][5] = -1.729329e+00f;
AutoMATiC_DMC_fconst[2][6] = -3.671660e+00f;
AutoMATiC_DMC_fconst[2][7] = -1.900426e+00f;
AutoMATiC_DMC_fconst[2][8] = -3.905929e+00f;
AutoMATiC_DMC_fconst[2][9] = -1.963369e+00f;
AutoMATiC_DMC_fconst[2][10] = -3.973048e+00f;
AutoMATiC_DMC_fconst[3][1] = 0.000000e+00f;
AutoMATiC_DMC_fconst[3][2] = 0.000000e+00f;
AutoMATiC_DMC_fconst[3][3] = 0.000000e+00f;
AutoMATiC_DMC_fconst[3][4] = 0.000000e+00f;
AutoMATiC_DMC_fconst[3][5] = -5.104583e+00f;
AutoMATiC_DMC_fconst[3][6] = -1.622249e+00f;
AutoMATiC_DMC_fconst[3][7] = -7.603490e+00f;
```

```
AutoMATiC_DMC_fconst[3][8] = -1.928652e+00f;
AutoMATiC_DMC_fconst[3][9] = -8.826808e+00f;
AutoMATiC_DMC_fconst[3][10] = -1.986524e+00f;
AutoMATiC_DMC_fconst[4][1] = 0.000000e+00f;
AutoMATiC_DMC_fconst[4][2] = 0.000000e+00f;
AutoMATiC_DMC_fconst[4][3] = 0.000000e+00f;
AutoMATiC_DMC_fconst[4][4] = 0.000000e+00f;
AutoMATiC_DMC_fconst[4][5] = -1.264241e+00f;
AutoMATiC_DMC_fconst[4][6] = -2.853981e+00f;
AutoMATiC_DMC_fconst[4][7] = -1.729329e+00f;
AutoMATiC_DMC_fconst[4][8] = -3.671660e+00f;
AutoMATiC_DMC_fconst[4][9] = -1.900426e+00f;
AutoMATiC_DMC_fconst[4][10] = -3.905929e+00f;
AutoMATiC_DMC_fconst[5][1] = 0.000000e+00f;
AutoMATiC_DMC_fconst[5][2] = 0.000000e+00f;
AutoMATiC_DMC_fconst[5][3] = 0.000000e+00f;
AutoMATiC_DMC_fconst[5][4] = 0.000000e+00f;
AutoMATiC_DMC_fconst[5][5] = 0.000000e+00f;
AutoMATiC_DMC_fconst[5][6] = 0.000000e+00f;
AutoMATiC_DMC_fconst[5][7] = -5.104583e+00f;
AutoMATiC_DMC_fconst[5][8] = -1.622249e+00f;
AutoMATiC_DMC_fconst[5][9] = -7.603490e+00f;
AutoMATiC_DMC_fconst[5][10] = -1.928652e+00f;
AutoMATiC_DMC_fconst[6][1] = 0.000000e+00f;
AutoMATiC_DMC_fconst[6][2] = 0.000000e+00f;
AutoMATiC_DMC_fconst[6][3] = 0.000000e+00f;
AutoMATiC_DMC_fconst[6][4] = 0.000000e+00f;
AutoMATiC_DMC_fconst[6][5] = 0.000000e+00f;
AutoMATiC_DMC_fconst[6][6] = 0.000000e+00f;
AutoMATiC_DMC_fconst[6][7] = -1.264241e+00f;
AutoMATiC_DMC_fconst[6][8] = -2.853981e+00f;
AutoMATiC_DMC_fconst[6][9] = -1.729329e+00f;
AutoMATiC_DMC_fconst[6][10] = -3.671660e+00f;
AutoMATiC_DMC_fconst[7][1] = 0.000000e+00f;
AutoMATiC_DMC_fconst[7][2] = 0.000000e+00f;
AutoMATiC_DMC_fconst[7][3] = 0.000000e+00f;
AutoMATiC_DMC_fconst[7][4] = 0.000000e+00f;
AutoMATiC_DMC_fconst[7][5] = 0.000000e+00f;
AutoMATiC_DMC_fconst[7][6] = 0.000000e+00f;
AutoMATiC_DMC_fconst[7][7] = 0.000000e+00f;
AutoMATiC_DMC_fconst[7][8] = 0.000000e+00f;
AutoMATiC_DMC_fconst[7][9] = -5.104583e+00f;
AutoMATiC_DMC_fconst[7][10] = -1.622249e+00f;
AutoMATiC_DMC_fconst[8][1] = 0.000000e+00f;
AutoMATiC_DMC_fconst[8][2] = 0.000000e+00f;
AutoMATiC_DMC_fconst[8][3] = 0.000000e+00f;
AutoMATiC_DMC_fconst[8][4] = 0.000000e+00f;
AutoMATiC_DMC_fconst[8][5] = 0.000000e+00f;
AutoMATiC_DMC_fconst[8][6] = 0.000000e+00f;
AutoMATiC_DMC_fconst[8][7] = 0.000000e+00f;
AutoMATiC_DMC_fconst[8][8] = 0.000000e+00f;
AutoMATiC_DMC_fconst[8][9] = -1.264241e+00f;
AutoMATiC_DMC_fconst[8][10] = -2.853981e+00f;
AutoMATiC_DMC_fconst[9][1] = 0.000000e+00f;
AutoMATiC_DMC_fconst[9][2] = 0.000000e+00f;
AutoMATiC_DMC_fconst[9][3] = 0.000000e+00f;
AutoMATiC_DMC_fconst[9][4] = 0.000000e+00f;
AutoMATiC_DMC_fconst[9][5] = 0.000000e+00f;
AutoMATiC_DMC_fconst[9][6] = 0.000000e+00f;
AutoMATiC_DMC_fconst[9][7] = 0.000000e+00f;
AutoMATiC_DMC_fconst[9][8] = 0.000000e+00f;
AutoMATiC_DMC_fconst[9][9] = 0.000000e+00f;
AutoMATiC_DMC_fconst[9][10] = 0.000000e+00f;
```

```
AutoMATiC_DMC_fconst[10][1] = 0.000000e+00f;
AutoMATiC_DMC_fconst[10][2] = 0.000000e+00f;
AutoMATiC_DMC_fconst[10][3] = 0.000000e+00f;
AutoMATiC_DMC_fconst[10][4] = 0.000000e+00f;
AutoMATiC_DMC_fconst[10][5] = 0.000000e+00f;
AutoMATiC_DMC_fconst[10][6] = 0.000000e+00f;
AutoMATiC_DMC_fconst[10][7] = 0.000000e+00f;
AutoMATiC_DMC_fconst[10][8] = 0.000000e+00f;
AutoMATiC_DMC_fconst[10][9] = 0.000000e+00f;
AutoMATiC_DMC_fconst[10][10] = 0.000000e+00f;
AutoMATiC_DMC_btmp1 = darray(1,40,1,1);
AutoMATiC_DMC_bvar = darray(1,40,1,2);
AutoMATiC_DMC_bvar[1][1] = 0.000000e+00f;
AutoMATiC_DMC_bvar[1][2] = 0.000000e+00f;
AutoMATiC_DMC_bvar[2][1] = 0.000000e+00f;
AutoMATiC_DMC_bvar[2][2] = 0.000000e+00f;
AutoMATiC_DMC_bvar[3][1] = 0.000000e+00f;
AutoMATiC_DMC_bvar[3][2] = 0.000000e+00f;
AutoMATiC_DMC_bvar[4][1] = 0.000000e+00f;
AutoMATiC_DMC_bvar[4][2] = 0.000000e+00f;
AutoMATiC_DMC_bvar[5][1] = 0.000000e+00f;
AutoMATiC_DMC_bvar[5][2] = 0.000000e+00f;
AutoMATiC_DMC_bvar[6][1] = 0.000000e+00f;
AutoMATiC_DMC_bvar[6][2] = 0.000000e+00f;
AutoMATiC_DMC_bvar[7][1] = 0.000000e+00f;
AutoMATiC_DMC_bvar[7][2] = 0.000000e+00f;
AutoMATiC_DMC_bvar[8][1] = 0.000000e+00f;
AutoMATiC_DMC_bvar[8][2] = 0.000000e+00f;
AutoMATiC_DMC_bvar[9][1] = 0.000000e+00f;
AutoMATiC_DMC_bvar[9][2] = 0.000000e+00f;
AutoMATiC_DMC_bvar[10][1] = 0.000000e+00f;
AutoMATiC_DMC_bvar[10][2] = 0.000000e+00f;
AutoMATiC_DMC_bvar[11][1] = 1.000000e+00f;
AutoMATiC_DMC_bvar[11][2] = 0.000000e+00f;
AutoMATiC_DMC_bvar[12][1] = 0.000000e+00f;
AutoMATiC_DMC_bvar[12][2] = 1.000000e+00f;
AutoMATiC_DMC_bvar[13][1] = 1.000000e+00f;
AutoMATiC_DMC_bvar[13][2] = 0.000000e+00f;
AutoMATiC_DMC_bvar[14][1] = 0.000000e+00f;
AutoMATiC_DMC_bvar[14][2] = 1.000000e+00f;
AutoMATiC_DMC_bvar[15][1] = 1.000000e+00f;
AutoMATiC_DMC_bvar[15][2] = 0.000000e+00f;
AutoMATiC_DMC_bvar[16][1] = 0.000000e+00f;
AutoMATiC_DMC_bvar[16][2] = 1.000000e+00f;
AutoMATiC_DMC_bvar[17][1] = 1.000000e+00f;
AutoMATiC_DMC_bvar[17][2] = 0.000000e+00f;
AutoMATiC_DMC_bvar[18][1] = 0.000000e+00f;
AutoMATiC_DMC_bvar[18][2] = 1.000000e+00f;
AutoMATiC_DMC_bvar[19][1] = 1.000000e+00f;
AutoMATiC_DMC_bvar[19][2] = 0.000000e+00f;
AutoMATiC_DMC_bvar[20][1] = 0.000000e+00f;
AutoMATiC_DMC_bvar[20][2] = 1.000000e+00f;
AutoMATiC_DMC_bvar[21][1] = -0.000000e+00f;
AutoMATiC_DMC_bvar[21][2] = -0.000000e+00f;
AutoMATiC_DMC_bvar[22][1] = -0.000000e+00f;
AutoMATiC_DMC_bvar[22][2] = -0.000000e+00f;
AutoMATiC_DMC_bvar[23][1] = -0.000000e+00f;
AutoMATiC_DMC_bvar[23][2] = -0.000000e+00f;
AutoMATiC_DMC_bvar[24][1] = -0.000000e+00f;
AutoMATiC_DMC_bvar[24][2] = -0.000000e+00f;
AutoMATiC_DMC_bvar[25][1] = -0.000000e+00f;
AutoMATiC_DMC_bvar[25][2] = -0.000000e+00f;
AutoMATiC_DMC_bvar[26][1] = -0.000000e+00f;
```

```
AutoMATiC_DMC_bvar[26][2] = -0.000000e+00f;
AutoMATiC_DMC_bvar[27][1] = -0.000000e+00f;
AutoMATiC_DMC_bvar[27][2] = -0.000000e+00f;
AutoMATiC_DMC_bvar[28][1] = -0.000000e+00f;
AutoMATiC_DMC_bvar[28][2] = -0.000000e+00f;
AutoMATiC_DMC_bvar[29][1] = -0.000000e+00f;
AutoMATiC_DMC_bvar[29][2] = -0.000000e+00f;
AutoMATiC_DMC_bvar[30][1] = -0.000000e+00f;
AutoMATiC_DMC_bvar[30][2] = -0.000000e+00f;
AutoMATiC_DMC_bvar[31][1] = -1.000000e+00f;
AutoMATiC_DMC_bvar[31][2] = -0.000000e+00f;
AutoMATiC_DMC_bvar[32][1] = -0.000000e+00f;
AutoMATiC_DMC_bvar[32][2] = -1.000000e+00f;
AutoMATiC_DMC_bvar[33][1] = -1.000000e+00f;
AutoMATiC_DMC_bvar[33][2] = -0.000000e+00f;
AutoMATiC_DMC_bvar[34][1] = -0.000000e+00f;
AutoMATiC_DMC_bvar[34][2] = -1.000000e+00f;
AutoMATiC_DMC_bvar[35][1] = -1.000000e+00f;
AutoMATiC_DMC_bvar[35][2] = -0.000000e+00f;
AutoMATiC_DMC_bvar[36][1] = -0.000000e+00f;
AutoMATiC_DMC_bvar[36][2] = -1.000000e+00f;
AutoMATiC_DMC_bvar[37][1] = -1.000000e+00f;
AutoMATiC_DMC_bvar[37][2] = -0.000000e+00f;
AutoMATiC_DMC_bvar[38][1] = -0.000000e+00f;
AutoMATiC_DMC_bvar[38][2] = -1.000000e+00f;
AutoMATiC_DMC_bvar[39][1] = -1.000000e+00f;
AutoMATiC_DMC_bvar[39][2] = -0.000000e+00f;
AutoMATiC_DMC_bvar[40][1] = -0.000000e+00f;
AutoMATiC_DMC_bvar[40][2] = -1.000000e+00f;
AutoMATiC_DMC_btmp2 = darray(1,40,1,1);
AutoMATiC_DMC_b = darray(1,40,1,1);
AutoMATiC_DMC_b[1][1] = 1.000000e-02f;
AutoMATiC_DMC_b[2][1] = 1.000000e-02f;
AutoMATiC_DMC_b[3][1] = 1.000000e-02f;
AutoMATiC_DMC_b[4][1] = 1.000000e-02f;
AutoMATiC_DMC_b[5][1] = 1.000000e-02f;
AutoMATiC_DMC_b[6][1] = 1.000000e-02f;
AutoMATiC_DMC_b[7][1] = 1.000000e-02f;
AutoMATiC_DMC_b[8][1] = 1.000000e-02f;
AutoMATiC_DMC_b[9][1] = 1.000000e-02f;
AutoMATiC_DMC_b[10][1] = 1.000000e-02f;
AutoMATiC_DMC_b[11][1] = 1.000000e+00f;
AutoMATiC_DMC_b[12][1] = 1.000000e+00f;
AutoMATiC_DMC_b[13][1] = 1.000000e+00f;
AutoMATiC_DMC_b[14][1] = 1.000000e+00f;
AutoMATiC_DMC_b[15][1] = 1.000000e+00f;
AutoMATiC_DMC_b[16][1] = 1.000000e+00f;
AutoMATiC_DMC_b[17][1] = 1.000000e+00f;
AutoMATiC_DMC_b[18][1] = 1.000000e+00f;
AutoMATiC_DMC_b[19][1] = 1.000000e+00f;
AutoMATiC_DMC_b[20][1] = 1.000000e+00f;
AutoMATiC_DMC_b[21][1] = 1.000000e-02f;
AutoMATiC_DMC_b[22][1] = 1.000000e-02f;
AutoMATiC_DMC_b[23][1] = 1.000000e-02f;
AutoMATiC_DMC_b[24][1] = 1.000000e-02f;
AutoMATiC_DMC_b[25][1] = 1.000000e-02f;
AutoMATiC_DMC_b[26][1] = 1.000000e-02f;
AutoMATiC_DMC_b[27][1] = 1.000000e-02f;
AutoMATiC_DMC_b[28][1] = 1.000000e-02f;
AutoMATiC_DMC_b[29][1] = 1.000000e-02f;
AutoMATiC_DMC_b[30][1] = 1.000000e-02f;
AutoMATiC_DMC_b[31][1] = 1.000000e+00f;
AutoMATiC_DMC_b[32][1] = 1.000000e+00f;
```

```
  AutoMATiC_DMC_b[33][1] = 1.000000e+00f;
  AutoMATiC_DMC_b[34][1] = 1.000000e+00f;
  AutoMATiC_DMC_b[35][1] = 1.000000e+00f;
  AutoMATiC_DMC_b[36][1] = 1.000000e+00f;
  AutoMATiC_DMC_b[37][1] = 1.000000e+00f;
  AutoMATiC_DMC_b[38][1] = 1.000000e+00f;
  AutoMATiC_DMC_b[39][1] = 1.000000e+00f;
  AutoMATiC_DMC_b[40][1] = 1.000000e+00f;
  dumax = darray(1,1,1,2);
  dumax[1][1] = 1.000000e-02f;
  dumax[1][2] = 1.000000e-02f;
  dumin = darray(1,1,1,2);
  dumin[1][1] = -1.000000e-02f;
  dumin[1][2] = -1.000000e-02f;
  umax = darray(1,1,1,2);
  umax[1][1] = 1.000000e+00f;
  umax[1][2] = 1.000000e+00f;
  umin = darray(1,1,1,2);
  umin[1][1] = -1.000000e+00f;
  umin[1][2] = -1.000000e+00f;
  return;
}
for(j=1;j<=8;++j) for(k=1;k<=1;++k) AutoMATiC_DMC_dUp[j][k] = 0;
for(j=1;j<=10;++j) for(k=1;k<=1;++k) AutoMATiC_DMC_Yzad[j][k] = 0;
for(j=1;j<=10;++j) for(k=1;k<=1;++k) AutoMATiC_DMC_Y[j][k] = 0;
for(j=1;j<=2;++j) for(k=1;k<=1;++k) AutoMATiC_DMC_tmpu[j][k] = 0;
AutoMATiC_DMC_itmp=1;
for(AutoMATiC_DMC_i=1;AutoMATiC_DMC_i<=4;++AutoMATiC_DMC_i){
  for(AutoMATiC_DMC_j=1;AutoMATiC_DMC_j<=2;++AutoMATiC_DMC_j){
    AutoMATiC_DMC_dUp[AutoMATiC_DMC_itmp][1]=ad->du[ad->k-AutoMATiC_DMC_i][
        AutoMATiC_DMC_j-1];
    AutoMATiC_DMC_itmp=AutoMATiC_DMC_itmp+1;
  }
}
AutoMATiC_DMC_itmp=1;
for(AutoMATiC_DMC_i=1;AutoMATiC_DMC_i<=5;++AutoMATiC_DMC_i){
  for(AutoMATiC_DMC_j=1;AutoMATiC_DMC_j<=2;++AutoMATiC_DMC_j){
    AutoMATiC_DMC_Yzad[AutoMATiC_DMC_itmp][1]=ad->z[AutoMATiC_DMC_j-1];
    AutoMATiC_DMC_itmp=AutoMATiC_DMC_itmp+1;
  }
}
AutoMATiC_DMC_itmp=1;
for(AutoMATiC_DMC_i=1;AutoMATiC_DMC_i<=5;++AutoMATiC_DMC_i){
  for(AutoMATiC_DMC_j=1;AutoMATiC_DMC_j<=2;++AutoMATiC_DMC_j){
    AutoMATiC_DMC_Y[AutoMATiC_DMC_itmp][1]=ad->y[ad->k][AutoMATiC_DMC_j-1];
    AutoMATiC_DMC_itmp=AutoMATiC_DMC_itmp+1;
  }
}
for(j=1;j<=2;++j) for(k=1;k<=1;++k) AutoMATiC_DMC_e[j][k] = 0;
for(AutoMATiC_DMC_i=1;AutoMATiC_DMC_i<=2;++AutoMATiC_DMC_i){
  AutoMATiC_DMC_e[AutoMATiC_DMC_i][1]=ad->z[AutoMATiC_DMC_i-1]-ad->y[ad->k][
      AutoMATiC_DMC_i-1];
}
productab(AutoMATiC_DMC_Ke,AutoMATiC_DMC_e,AutoMATiC_DMC_dutmp1,2,2,2,1);
productab(AutoMATiC_DMC_Ku,AutoMATiC_DMC_dUp,AutoMATiC_DMC_dutmp2,2,8,8,1);
sumaa(AutoMATiC_DMC_dutmp1,AutoMATiC_DMC_dutmp2,AutoMATiC_DMC_du,2,1,-1);
for(j=1;j<=2;++j) for(k=1;k<=1;++k) AutoMATiC_DMC_du[j][k] = 0;
for(j=1;j<=2;++j) for(k=1;k<=1;++k) AutoMATiC_DMC_uk[j][k] = 0;
for(AutoMATiC_DMC_i=1;AutoMATiC_DMC_i<=2;++AutoMATiC_DMC_i){
  AutoMATiC_DMC_uk[AutoMATiC_DMC_i][1]=ad->u[ad->k-1][AutoMATiC_DMC_i-1];
}
sumaa(AutoMATiC_DMC_Yzad,AutoMATiC_DMC_Y,AutoMATiC_DMC_ftmp1,10,1,-1);
productab(AutoMATiC_DMC_Mp,AutoMATiC_DMC_dUp,AutoMATiC_DMC_ftmp2,10,8,8,1);
```

```
  sumaa(AutoMATiC_DMC_ftmp1,AutoMATiC_DMC_ftmp2,AutoMATiC_DMC_ftmp3,10,1,-1);
  productab(AutoMATiC_DMC_fconst,AutoMATiC_DMC_ftmp3,AutoMATiC_DMC_ftmp4
      ,10,10,10,1);
  for(AutoMATiC_DMC_i=1;AutoMATiC_DMC_i<=10;++AutoMATiC_DMC_i){
    AutoMATiC_DMC_ftmp[AutoMATiC_DMC_i]=AutoMATiC_DMC_ftmp4[AutoMATiC_DMC_i
        ][1];
  }
  productab(AutoMATiC_DMC_bvar,AutoMATiC_DMC_uk,AutoMATiC_DMC_btmp1,40,2,2,1);
  sumaa(AutoMATiC_DMC_b,AutoMATiC_DMC_btmp1,AutoMATiC_DMC_btmp2,40,1,1);
  for(AutoMATiC_DMC_i=1;AutoMATiC_DMC_i<=40;++AutoMATiC_DMC_i){
    AutoMATiC_DMC_btmp[AutoMATiC_DMC_i]=AutoMATiC_DMC_btmp2[AutoMATiC_DMC_i
        ][1];
  }
  for(AutoMATiC_DMC_i=1;AutoMATiC_DMC_i<=10;++AutoMATiC_DMC_i){
    AutoMATiC_DMC_qpx[AutoMATiC_DMC_i]=0;
  }
  memcpy (osqp_q, &(AutoMATiC_DMC_ftmp[1]), 10*sizeof(c_float));
  osqp_update_lin_cost(osqp_work, osqp_q);
  memcpy (osqp_ub, &(AutoMATiC_DMC_btmp[1]), 40*sizeof(c_float));
  osqp_update_bounds(osqp_work, osqp_lb, osqp_ub);
  osqp_solve(osqp_work);
  memcpy (&(AutoMATiC_DMC_qpx[1]), osqp_work->solution->x, 10*sizeof(c_float));

  for(AutoMATiC_DMC_i=1;AutoMATiC_DMC_i<=2;++AutoMATiC_DMC_i){
    AutoMATiC_DMC_du[AutoMATiC_DMC_i][1]=AutoMATiC_DMC_qpx[AutoMATiC_DMC_i];
  }
  for(AutoMATiC_DMC_n=1;AutoMATiC_DMC_n<=2;++AutoMATiC_DMC_n){
    if(AutoMATiC_DMC_du[AutoMATiC_DMC_n][1]>dumax[1][AutoMATiC_DMC_n]){
      AutoMATiC_DMC_du[AutoMATiC_DMC_n][1]=dumax[1][AutoMATiC_DMC_n];
    }
    if(AutoMATiC_DMC_du[AutoMATiC_DMC_n][1]<dumin[1][AutoMATiC_DMC_n]){
      AutoMATiC_DMC_du[AutoMATiC_DMC_n][1]=dumin[1][AutoMATiC_DMC_n];
    }
    AutoMATiC_DMC_tmpu[AutoMATiC_DMC_n][1]=ad->u[ad->k-1][AutoMATiC_DMC_n-1]+
        AutoMATiC_DMC_du[AutoMATiC_DMC_n][1];
    if(AutoMATiC_DMC_tmpu[AutoMATiC_DMC_n][1]>umax[1][AutoMATiC_DMC_n]){
      AutoMATiC_DMC_tmpu[AutoMATiC_DMC_n][1]=umax[1][AutoMATiC_DMC_n];
    }
    if(AutoMATiC_DMC_tmpu[AutoMATiC_DMC_n][1]<umin[1][AutoMATiC_DMC_n]){
      AutoMATiC_DMC_tmpu[AutoMATiC_DMC_n][1]=umin[1][AutoMATiC_DMC_n];
    }
    AutoMATiC_DMC_du[AutoMATiC_DMC_n][1]=AutoMATiC_DMC_tmpu[AutoMATiC_DMC_n
        ][1]-ad->u[ad->k-1][AutoMATiC_DMC_n-1];
  }
  for(j=1;j<=1;++j) for(k=1;k<=2;++k) control_value[j][k] = 0;
  for(AutoMATiC_DMC_n=1;AutoMATiC_DMC_n<=2;++AutoMATiC_DMC_n){
    control_value[1][AutoMATiC_DMC_n]=AutoMATiC_DMC_du[AutoMATiC_DMC_n][1];
  }

  set_current_control_increment(c,&(control_value[1][1])); // du is indexed
      starting with 1, therefore to maintain compatibility it is required to
      refer to first element of an actual array
}

void controller_setup(){
profiler_start(2);
    init_archive_data(&ad, 200, 200, 2, 2, 0, 0, 0.01);
    init_current_control(&cc,&ad);
    controller(NULL,NULL);
profiler_end(2);
}

void idle(){
```

```
profiler_start(13);
    const int k = 0;
  static int i = 0;
  static char str[1000] = {0};

    sprintf(str, "x = [%f,%f,",ad.y[k][0], ad.y[k][1]);     write_string(str);
    sprintf(str,       "%f,%f,",ad.z[0], ad.z[1]);          write_string(str);
    sprintf(str,       "%f,%f,",ad.u[k-1][0],ad.u[k-1][1]); write_string(str);
    write_string("];\n\r");
  if(++i > 1000) profiler_print();
profiler_end(13);
}

void loop(){
profiler_start(10);
    static int i = 0;
       if(i< 100){ ad.z[0] = -0.1f; ad.z[1] = +0.2f; }
  else if(i< 150){ ad.z[0] = -0.1f; ad.z[1] = -0.2f; }
  else if(i< 250){ ad.z[0] = +0.1f; ad.z[1] = -0.2f; }
  else           { ad.z[0] = +0.1f; ad.z[1] = +0.2f; }
  if(++i > 300) i = 0;

profiler_start(50);
  controller(&ad,&cc);
profiler_end(50);

  push_current_controls_to_archive_data(&cc,&ad);
profiler_end(10);
}

void timeout(){
    while(1);
}
```

C Analytical GPC Algorithm: Generated Source of `main_mpc.c`

```
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\defines.h"
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\profiler.h"
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\mpctools.h"
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\simulated_signals.h"
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\matrix_cal.h"
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\allocation_nr.h"
#include "osqp.h"
#include "util.h"
#include "stm32f7xx_hal.h"
#include <string.h>
#include "main.h"

ArchiveData ad;
CurrentControl cc;

long get_time(){ return HAL_GetTick(); }

extern void timer_loop(void);
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){
    if (htim->Instance == TIM2) {
        timer_loop();
    }
}
void measurements(){
profiler_start(4);
    new_output(&ad,__measureOutput());
profiler_end(4);
}
void controls(){
profiler_start(3);
    __setControlValue(last_control(&ad));
profiler_end(3);
}
void hardware_setup(){
profiler_start(1);
    low_lvl_main();
profiler_end(1);
}

void controller(ArchiveData * ad, CurrentControl * c){
  static float** AutoMATiC_GPC_du;
  static long AutoMATiC_GPC_j;
  static long AutoMATiC_GPC_m;
  static long AutoMATiC_GPC_n;
  static long AutoMATiC_GPC_r;
  static float** AutoMATiC_GPC_tmpu;
  static float** control_value;
  static long j;
  static long k;
  static float** AutoMATiC_GPC_Kyzad;
  static float*** AutoMATiC_GPC_Ku;
  static float*** AutoMATiC_GPC_Ky;
  static float** dumax;
  static float** dumin;
  static float** umax;
  static float** umin;
  if(ad == NULL){
    AutoMATiC_GPC_du = darray(1,2,1,1);
    AutoMATiC_GPC_tmpu = darray(1,2,1,1);
    control_value = darray(1,1,1,2);
    AutoMATiC_GPC_Kyzad = darray(1,2,1,2);
```

```
  AutoMATiC_GPC_Kyzad[1][1] = 3.373006e-01f;
  AutoMATiC_GPC_Kyzad[1][2] = -8.539666e-02f;
  AutoMATiC_GPC_Kyzad[2][1] = -1.016548e-01f;
  AutoMATiC_GPC_Kyzad[2][2] = 5.115693e-01f;
  AutoMATiC_GPC_Ku = darray3(1,2,1,2,1,4);
  AutoMATiC_GPC_Ku[1][1][1] = -1.246174e+00f;
  AutoMATiC_GPC_Ku[1][1][2] = 1.843347e+00f;
  AutoMATiC_GPC_Ku[1][1][3] = -5.971733e-01f;
  AutoMATiC_GPC_Ku[1][1][4] = 0.000000e+00f;
  AutoMATiC_GPC_Ku[1][2][1] = -1.325670e-01f;
  AutoMATiC_GPC_Ku[1][2][2] = 3.024850e-01f;
  AutoMATiC_GPC_Ku[1][2][3] = -1.699180e-01f;
  AutoMATiC_GPC_Ku[1][2][4] = 0.000000e+00f;
  AutoMATiC_GPC_Ku[2][1][1] = -5.999027e-02f;
  AutoMATiC_GPC_Ku[2][1][2] = 2.028332e-02f;
  AutoMATiC_GPC_Ku[2][1][3] = 3.970695e-02f;
  AutoMATiC_GPC_Ku[2][1][4] = 0.000000e+00f;
  AutoMATiC_GPC_Ku[2][2][1] = -8.674656e-01f;
  AutoMATiC_GPC_Ku[2][2][2] = 1.006652e+00f;
  AutoMATiC_GPC_Ku[2][2][3] = -1.391867e-01f;
  AutoMATiC_GPC_Ku[2][2][4] = 0.000000e+00f;
  AutoMATiC_GPC_Ky = darray3(1,2,1,2,1,3);
  AutoMATiC_GPC_Ky[1][1][1] = -8.392055e-01f;
  AutoMATiC_GPC_Ky[1][1][2] = 6.226592e-01f;
  AutoMATiC_GPC_Ky[1][1][3] = -1.207543e-01f;
  AutoMATiC_GPC_Ky[1][2][1] = 1.450209e-01f;
  AutoMATiC_GPC_Ky[1][2][2] = -6.716780e-02f;
  AutoMATiC_GPC_Ky[1][2][3] = 7.543518e-03f;
  AutoMATiC_GPC_Ky[2][1][1] = 2.732700e-01f;
  AutoMATiC_GPC_Ky[2][1][2] = -2.144111e-01f;
  AutoMATiC_GPC_Ky[2][1][3] = 4.279587e-02f;
  AutoMATiC_GPC_Ky[2][2][1] = -8.538780e-01f;
  AutoMATiC_GPC_Ky[2][2][2] = 3.850181e-01f;
  AutoMATiC_GPC_Ky[2][2][3] = -4.270940e-02f;
  dumax = darray(1,1,1,2);
  dumax[1][1] = 1.000000e-02f;
  dumax[1][2] = 1.000000e-02f;
  dumin = darray(1,1,1,2);
  dumin[1][1] = -1.000000e-02f;
  dumin[1][2] = -1.000000e-02f;
  umax = darray(1,1,1,2);
  umax[1][1] = 1.000000e+00f;
  umax[1][2] = 1.000000e+00f;
  umin = darray(1,1,1,2);
  umin[1][1] = -1.000000e+00f;
  umin[1][2] = -1.000000e+00f;
  return;
}
for(j=1;j<=2;++j) for(k=1;k<=1;++k) AutoMATiC_GPC_du[j][k] = 0;
for(j=1;j<=2;++j) for(k=1;k<=1;++k) AutoMATiC_GPC_tmpu[j][k] = 0;
for(AutoMATiC_GPC_r=1;AutoMATiC_GPC_r<=2;++AutoMATiC_GPC_r){
  for(AutoMATiC_GPC_m=1;AutoMATiC_GPC_m<=2;++AutoMATiC_GPC_m){
    AutoMATiC_GPC_du[AutoMATiC_GPC_r][1]=AutoMATiC_GPC_du[AutoMATiC_GPC_r
        ][1]+AutoMATiC_GPC_Kyzad[AutoMATiC_GPC_r][AutoMATiC_GPC_m]*ad->z[
        AutoMATiC_GPC_m-1];
  }
  for(AutoMATiC_GPC_n=1;AutoMATiC_GPC_n<=2;++AutoMATiC_GPC_n){
    for(AutoMATiC_GPC_j=1;AutoMATiC_GPC_j<=4;++AutoMATiC_GPC_j){
      AutoMATiC_GPC_du[AutoMATiC_GPC_r][1]=AutoMATiC_GPC_du[AutoMATiC_GPC_r
          ][1]+AutoMATiC_GPC_Ku[AutoMATiC_GPC_r][AutoMATiC_GPC_n][
          AutoMATiC_GPC_j]*ad->u[ad->k-AutoMATiC_GPC_j][AutoMATiC_GPC_n-1];
    }
  }
```

```
    for(AutoMATiC_GPC_m=1;AutoMATiC_GPC_m<=2;++AutoMATiC_GPC_m){
      for(AutoMATiC_GPC_j=0;AutoMATiC_GPC_j<=2;++AutoMATiC_GPC_j){
        AutoMATiC_GPC_du[AutoMATiC_GPC_r][1]=AutoMATiC_GPC_du[AutoMATiC_GPC_r
            ][1]+AutoMATiC_GPC_Ky[AutoMATiC_GPC_r][AutoMATiC_GPC_m][
            AutoMATiC_GPC_j+1]*ad->y[ad->k-AutoMATiC_GPC_j][AutoMATiC_GPC_m-1];
      }
    }
  }
  for(AutoMATiC_GPC_n=1;AutoMATiC_GPC_n<=2;++AutoMATiC_GPC_n){
    if(AutoMATiC_GPC_du[AutoMATiC_GPC_n][1]>dumax[1][AutoMATiC_GPC_n]){
      AutoMATiC_GPC_du[AutoMATiC_GPC_n][1]=dumax[1][AutoMATiC_GPC_n];
    }
    if(AutoMATiC_GPC_du[AutoMATiC_GPC_n][1]<dumin[1][AutoMATiC_GPC_n]){
      AutoMATiC_GPC_du[AutoMATiC_GPC_n][1]=dumin[1][AutoMATiC_GPC_n];
    }
    AutoMATiC_GPC_tmpu[AutoMATiC_GPC_n][1]=ad->u[ad->k-1][AutoMATiC_GPC_n-1]+
        AutoMATiC_GPC_du[AutoMATiC_GPC_n][1];
    if(AutoMATiC_GPC_tmpu[AutoMATiC_GPC_n][1]>umax[1][AutoMATiC_GPC_n]){
      AutoMATiC_GPC_tmpu[AutoMATiC_GPC_n][1]=umax[1][AutoMATiC_GPC_n];
    }
    if(AutoMATiC_GPC_tmpu[AutoMATiC_GPC_n][1]<umin[1][AutoMATiC_GPC_n]){
      AutoMATiC_GPC_tmpu[AutoMATiC_GPC_n][1]=umin[1][AutoMATiC_GPC_n];
    }
    AutoMATiC_GPC_du[AutoMATiC_GPC_n][1]=AutoMATiC_GPC_tmpu[AutoMATiC_GPC_n
        ][1]-ad->u[ad->k-1][AutoMATiC_GPC_n-1];
  }
  for(j=1;j<=1;++j) for(k=1;k<=2;++k) control_value[j][k] = 0;
  for(AutoMATiC_GPC_n=1;AutoMATiC_GPC_n<=2;++AutoMATiC_GPC_n){
    control_value[1][AutoMATiC_GPC_n]=AutoMATiC_GPC_du[AutoMATiC_GPC_n][1];
  }

  set_current_control_increment(c,&(control_value[1][1])); // du is indexed
      starting with 1, therefore to maintain compatibility it is required to
      refer to first element of an actual array
}

void controller_setup(){
profiler_start(2);
    init_archive_data(&ad, 200, 200, 2, 2, 0, 0, 0.01);
    init_current_control(&cc,&ad);
    controller(NULL,NULL);
profiler_end(2);
}

void idle(){
profiler_start(13);
    const int k = 0;
  static int i = 0;
  static char str[1000] = {0};

    sprintf(str, "x = [%f,%f,",ad.y[k][0], ad.y[k][1]);    write_string(str);
    sprintf(str,       "%f,%f,",ad.z[0], ad.z[1]);         write_string(str);
    sprintf(str,       "%f,%f,",ad.u[k-1][0],ad.u[k-1][1]); write_string(str);
    write_string("];\n\r");
  if(++i > 1000) profiler_print();
profiler_end(13);
}

void loop(){
profiler_start(10);
    static int i = 0;
       if(i< 100){ ad.z[0] = -0.1f; ad.z[1] = +0.2f; }
  else if(i< 150){ ad.z[0] = -0.1f; ad.z[1] = -0.2f; }
```

```
  else if(i< 250){ ad.z[0] = +0.1f; ad.z[1] = -0.2f; }
  else           { ad.z[0] = +0.1f; ad.z[1] = +0.2f; }
  if(++i > 300) i = 0;

profiler_start(50);
  controller(&ad,&cc);
profiler_end(50);

  push_current_controls_to_archive_data(&cc,&ad);
profiler_end(10);
}

void timeout(){
    while(1);
}
```

D Numerical GPC Algorithm: Generated Source of main_mpc.c

```
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\defines.h"
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\profiler.h"
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\mpctools.h"
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\simulated_signals.h"
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\matrix_cal.h"
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\allocation_nr.h"
#include "osqp.h"
#include "util.h"
#include "stm32f7xx_hal.h"
#include <string.h>
#include "main.h"

ArchiveData ad;
CurrentControl cc;

long get_time(){ return HAL_GetTick(); }

extern void timer_loop(void);
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){
    if (htim->Instance == TIM2) {
        timer_loop();
    }
}
void measurements(){
profiler_start(4);
    new_output(&ad,__measureOutput());
profiler_end(4);
}
void controls(){
profiler_start(3);
    __setControlValue(last_control(&ad));
profiler_end(3);
}
void hardware_setup(){
profiler_start(1);
    low_lvl_main();
profiler_end(1);
}

void controller(ArchiveData * ad, CurrentControl * c){
  static long Iuf;
  static long Iuftmp;
  static long Iyf;
  static float** Y0;
  static float** Yzad;
  static float** btmp1;
  static float** btmp2;
  static float** control_value;
  static float** disturbance;
  static float** du;
  static float** ftmp1;
  static float** ftmp4;
  static long i;
  static long itmp;
  static long j;
  static long m;
  static long n;
  static long p;
  static long r;
  static float** tmpu;
  static float** uk;
```

```
static float** y0;
static long k;
static float*** GPC_b;
static float** GPC_a;
static float** fconst;
static float** bvar;
static float** b;
static float** dumax;
static float** dumin;
static float** umax;
static float** umin;
static c_float A_x[80];
static c_int A_i[80];
static c_int A_p[11];

static c_float H_x[66];
static c_int H_i[66];
static c_int H_p[11];

static OSQPSettings *osqp_settings;
static OSQPWorkspace *osqp_work;
static OSQPData *osqp_data;
static c_float osqp_q[10];
static c_float osqp_lb[40];
static c_float osqp_ub[40];
static c_float ftmp[11];
static c_float btmp[41];
static c_float qpx[11];
if(ad == NULL){
  A_x[0] = -1.000000e+00f;
  A_x[1] = 1.000000e+00f;
  A_x[2] = -1.000000e+00f;
  A_x[3] = -1.000000e+00f;
  A_x[4] = -1.000000e+00f;
  A_x[5] = -1.000000e+00f;
  A_x[6] = -1.000000e+00f;
  A_x[7] = 1.000000e+00f;
  A_x[8] = 1.000000e+00f;
  A_x[9] = 1.000000e+00f;
  A_x[10] = 1.000000e+00f;
  A_x[11] = 1.000000e+00f;
  A_x[12] = -1.000000e+00f;
  A_x[13] = 1.000000e+00f;
  A_x[14] = -1.000000e+00f;
  A_x[15] = -1.000000e+00f;
  A_x[16] = -1.000000e+00f;
  A_x[17] = -1.000000e+00f;
  A_x[18] = -1.000000e+00f;
  A_x[19] = 1.000000e+00f;
  A_x[20] = 1.000000e+00f;
  A_x[21] = 1.000000e+00f;
  A_x[22] = 1.000000e+00f;
  A_x[23] = 1.000000e+00f;
  A_x[24] = -1.000000e+00f;
  A_x[25] = 1.000000e+00f;
  A_x[26] = -1.000000e+00f;
  A_x[27] = -1.000000e+00f;
  A_x[28] = -1.000000e+00f;
  A_x[29] = -1.000000e+00f;
  A_x[30] = 1.000000e+00f;
  A_x[31] = 1.000000e+00f;
  A_x[32] = 1.000000e+00f;
  A_x[33] = 1.000000e+00f;
```

```
A_x[34] = -1.000000e+00f;
A_x[35] = 1.000000e+00f;
A_x[36] = -1.000000e+00f;
A_x[37] = -1.000000e+00f;
A_x[38] = -1.000000e+00f;
A_x[39] = -1.000000e+00f;
A_x[40] = 1.000000e+00f;
A_x[41] = 1.000000e+00f;
A_x[42] = 1.000000e+00f;
A_x[43] = 1.000000e+00f;
A_x[44] = -1.000000e+00f;
A_x[45] = 1.000000e+00f;
A_x[46] = -1.000000e+00f;
A_x[47] = -1.000000e+00f;
A_x[48] = -1.000000e+00f;
A_x[49] = 1.000000e+00f;
A_x[50] = 1.000000e+00f;
A_x[51] = 1.000000e+00f;
A_x[52] = -1.000000e+00f;
A_x[53] = 1.000000e+00f;
A_x[54] = -1.000000e+00f;
A_x[55] = -1.000000e+00f;
A_x[56] = -1.000000e+00f;
A_x[57] = 1.000000e+00f;
A_x[58] = 1.000000e+00f;
A_x[59] = 1.000000e+00f;
A_x[60] = -1.000000e+00f;
A_x[61] = 1.000000e+00f;
A_x[62] = -1.000000e+00f;
A_x[63] = -1.000000e+00f;
A_x[64] = 1.000000e+00f;
A_x[65] = 1.000000e+00f;
A_x[66] = -1.000000e+00f;
A_x[67] = 1.000000e+00f;
A_x[68] = -1.000000e+00f;
A_x[69] = -1.000000e+00f;
A_x[70] = 1.000000e+00f;
A_x[71] = 1.000000e+00f;
A_x[72] = -1.000000e+00f;
A_x[73] = 1.000000e+00f;
A_x[74] = -1.000000e+00f;
A_x[75] = 1.000000e+00f;
A_x[76] = -1.000000e+00f;
A_x[77] = 1.000000e+00f;
A_x[78] = -1.000000e+00f;
A_x[79] = 1.000000e+00f;
A_i[0] = 0;
A_i[1] = 10;
A_i[2] = 20;
A_i[3] = 22;
A_i[4] = 24;
A_i[5] = 26;
A_i[6] = 28;
A_i[7] = 30;
A_i[8] = 32;
A_i[9] = 34;
A_i[10] = 36;
A_i[11] = 38;
A_i[12] = 1;
A_i[13] = 11;
A_i[14] = 21;
A_i[15] = 23;
A_i[16] = 25;
```

```
A_i[17] = 27;
A_i[18] = 29;
A_i[19] = 31;
A_i[20] = 33;
A_i[21] = 35;
A_i[22] = 37;
A_i[23] = 39;
A_i[24] = 2;
A_i[25] = 12;
A_i[26] = 22;
A_i[27] = 24;
A_i[28] = 26;
A_i[29] = 28;
A_i[30] = 32;
A_i[31] = 34;
A_i[32] = 36;
A_i[33] = 38;
A_i[34] = 3;
A_i[35] = 13;
A_i[36] = 23;
A_i[37] = 25;
A_i[38] = 27;
A_i[39] = 29;
A_i[40] = 33;
A_i[41] = 35;
A_i[42] = 37;
A_i[43] = 39;
A_i[44] = 4;
A_i[45] = 14;
A_i[46] = 24;
A_i[47] = 26;
A_i[48] = 28;
A_i[49] = 34;
A_i[50] = 36;
A_i[51] = 38;
A_i[52] = 5;
A_i[53] = 15;
A_i[54] = 25;
A_i[55] = 27;
A_i[56] = 29;
A_i[57] = 35;
A_i[58] = 37;
A_i[59] = 39;
A_i[60] = 6;
A_i[61] = 16;
A_i[62] = 26;
A_i[63] = 28;
A_i[64] = 36;
A_i[65] = 38;
A_i[66] = 7;
A_i[67] = 17;
A_i[68] = 27;
A_i[69] = 29;
A_i[70] = 37;
A_i[71] = 39;
A_i[72] = 8;
A_i[73] = 18;
A_i[74] = 28;
A_i[75] = 38;
A_i[76] = 9;
A_i[77] = 19;
A_i[78] = 29;
A_i[79] = 39;
```

```
A_p[0] = 0;
A_p[1] = 12;
A_p[2] = 24;
A_p[3] = 34;
A_p[4] = 44;
A_p[5] = 52;
A_p[6] = 60;
A_p[7] = 66;
A_p[8] = 72;
A_p[9] = 76;
A_p[10] = 80;

H_x[0] = 1.344566e+02f;
H_x[1] = 4.114478e+01f;
H_x[2] = 1.000269e+02f;
H_x[3] = 3.169500e+01f;
H_x[4] = 6.190011e+01f;
H_x[5] = 2.023139e+01f;
H_x[6] = 2.567725e+01f;
H_x[7] = 8.808511e+00f;
H_x[8] = 4.114478e+01f;
H_x[9] = 3.436150e+01f;
H_x[10] = 3.099487e+01f;
H_x[11] = 2.477126e+01f;
H_x[12] = 1.931418e+01f;
H_x[13] = 1.576652e+01f;
H_x[14] = 8.233726e+00f;
H_x[15] = 6.910587e+00f;
H_x[16] = 1.000269e+02f;
H_x[17] = 3.099487e+01f;
H_x[18] = 8.804002e+01f;
H_x[19] = 2.792375e+01f;
H_x[20] = 5.644363e+01f;
H_x[21] = 1.883764e+01f;
H_x[22] = 2.413991e+01f;
H_x[23] = 8.414358e+00f;
H_x[24] = 3.169500e+01f;
H_x[25] = 2.477126e+01f;
H_x[26] = 2.792375e+01f;
H_x[27] = 2.454154e+01f;
H_x[28] = 1.838345e+01f;
H_x[29] = 1.514642e+01f;
H_x[30] = 8.018635e+00f;
H_x[31] = 6.775021e+00f;
H_x[32] = 6.190011e+01f;
H_x[33] = 1.931418e+01f;
H_x[34] = 5.644363e+01f;
H_x[35] = 1.838345e+01f;
H_x[36] = 4.711061e+01f;
H_x[37] = 1.565679e+01f;
H_x[38] = 2.097070e+01f;
H_x[39] = 7.558490e+00f;
H_x[40] = 2.023139e+01f;
H_x[41] = 1.576652e+01f;
H_x[42] = 1.883764e+01f;
H_x[43] = 1.514642e+01f;
H_x[44] = 1.565679e+01f;
H_x[45] = 1.510759e+01f;
H_x[46] = 7.391926e+00f;
H_x[47] = 6.332568e+00f;
H_x[48] = 2.567725e+01f;
H_x[49] = 8.233726e+00f;
H_x[50] = 2.413991e+01f;
```

```
H_x[51] = 8.018635e+00f;
H_x[52] = 2.097070e+01f;
H_x[53] = 7.391926e+00f;
H_x[54] = 1.634423e+01f;
H_x[55] = 5.541646e+00f;
H_x[56] = 8.808511e+00f;
H_x[57] = 6.910587e+00f;
H_x[58] = 8.414358e+00f;
H_x[59] = 6.775021e+00f;
H_x[60] = 7.558490e+00f;
H_x[61] = 6.332568e+00f;
H_x[62] = 5.541646e+00f;
H_x[63] = 6.871756e+00f;
H_x[64] = 2.000000e+00f;
H_x[65] = 2.000000e+00f;
H_i[0] = 0;
H_i[1] = 1;
H_i[2] = 2;
H_i[3] = 3;
H_i[4] = 4;
H_i[5] = 5;
H_i[6] = 6;
H_i[7] = 7;
H_i[8] = 0;
H_i[9] = 1;
H_i[10] = 2;
H_i[11] = 3;
H_i[12] = 4;
H_i[13] = 5;
H_i[14] = 6;
H_i[15] = 7;
H_i[16] = 0;
H_i[17] = 1;
H_i[18] = 2;
H_i[19] = 3;
H_i[20] = 4;
H_i[21] = 5;
H_i[22] = 6;
H_i[23] = 7;
H_i[24] = 0;
H_i[25] = 1;
H_i[26] = 2;
H_i[27] = 3;
H_i[28] = 4;
H_i[29] = 5;
H_i[30] = 6;
H_i[31] = 7;
H_i[32] = 0;
H_i[33] = 1;
H_i[34] = 2;
H_i[35] = 3;
H_i[36] = 4;
H_i[37] = 5;
H_i[38] = 6;
H_i[39] = 7;
H_i[40] = 0;
H_i[41] = 1;
H_i[42] = 2;
H_i[43] = 3;
H_i[44] = 4;
H_i[45] = 5;
H_i[46] = 6;
H_i[47] = 7;
```

```
H_i[48] = 0;
H_i[49] = 1;
H_i[50] = 2;
H_i[51] = 3;
H_i[52] = 4;
H_i[53] = 5;
H_i[54] = 6;
H_i[55] = 7;
H_i[56] = 0;
H_i[57] = 1;
H_i[58] = 2;
H_i[59] = 3;
H_i[60] = 4;
H_i[61] = 5;
H_i[62] = 6;
H_i[63] = 7;
H_i[64] = 8;
H_i[65] = 9;
H_p[0] = 0;
H_p[1] = 8;
H_p[2] = 16;
H_p[3] = 24;
H_p[4] = 32;
H_p[5] = 40;
H_p[6] = 48;
H_p[7] = 56;
H_p[8] = 64;
H_p[9] = 65;
H_p[10] = 66;

osqp_settings = (OSQPSettings *)c_malloc(sizeof(OSQPSettings));
osqp_data = (OSQPData *)c_malloc(sizeof(OSQPData));
for(int osqp_it=0; osqp_it < 40; ++osqp_it) osqp_lb[osqp_it] = -100000;
osqp_data->n = 10;
osqp_data->m = 40;
osqp_data->q = osqp_q;
osqp_data->l = osqp_lb;
osqp_data->u = osqp_ub;
osqp_data->P = csc_matrix(osqp_data->n, osqp_data->n, 66, H_x, H_i, H_p);
osqp_data->A = csc_matrix(osqp_data->m, osqp_data->n, 80, A_x, A_i, A_p);
osqp_set_default_settings(osqp_settings);
osqp_work = osqp_setup(osqp_data, osqp_settings);
tmpu = darray(1,2,1,1);
du = darray(1,2,1,1);
Yzad = darray(1,10,1,1);
y0 = darray(1,2,1,5);
Y0 = darray(1,10,1,1);
uk = darray(1,2,1,1);
disturbance = darray(1,2,1,1);
control_value = darray(1,1,1,2);
ftmp1 = darray(1,10,1,1);
ftmp4 = darray(1,10,1,1);
fconst = darray(1,10,1,10);
fconst[1][1] = 0.000000e+00f;
fconst[1][2] = 0.000000e+00f;
fconst[1][3] = -5.104583e+00f;
fconst[1][4] = -1.622249e+00f;
fconst[1][5] = -7.603490e+00f;
fconst[1][6] = -1.928652e+00f;
fconst[1][7] = -8.826808e+00f;
fconst[1][8] = -1.986524e+00f;
fconst[1][9] = -9.425674e+00f;
fconst[1][10] = -1.997455e+00f;
```

```
fconst[2][1] = 0.000000e+00f;
fconst[2][2] = 0.000000e+00f;
fconst[2][3] = -1.264241e+00f;
fconst[2][4] = -2.853981e+00f;
fconst[2][5] = -1.729329e+00f;
fconst[2][6] = -3.671660e+00f;
fconst[2][7] = -1.900426e+00f;
fconst[2][8] = -3.905929e+00f;
fconst[2][9] = -1.963369e+00f;
fconst[2][10] = -3.973048e+00f;
fconst[3][1] = 0.000000e+00f;
fconst[3][2] = 0.000000e+00f;
fconst[3][3] = 0.000000e+00f;
fconst[3][4] = 0.000000e+00f;
fconst[3][5] = -5.104583e+00f;
fconst[3][6] = -1.622249e+00f;
fconst[3][7] = -7.603490e+00f;
fconst[3][8] = -1.928652e+00f;
fconst[3][9] = -8.826808e+00f;
fconst[3][10] = -1.986524e+00f;
fconst[4][1] = 0.000000e+00f;
fconst[4][2] = 0.000000e+00f;
fconst[4][3] = 0.000000e+00f;
fconst[4][4] = 0.000000e+00f;
fconst[4][5] = -1.264241e+00f;
fconst[4][6] = -2.853981e+00f;
fconst[4][7] = -1.729329e+00f;
fconst[4][8] = -3.671660e+00f;
fconst[4][9] = -1.900426e+00f;
fconst[4][10] = -3.905929e+00f;
fconst[5][1] = 0.000000e+00f;
fconst[5][2] = 0.000000e+00f;
fconst[5][3] = 0.000000e+00f;
fconst[5][4] = 0.000000e+00f;
fconst[5][5] = 0.000000e+00f;
fconst[5][6] = 0.000000e+00f;
fconst[5][7] = -5.104583e+00f;
fconst[5][8] = -1.622249e+00f;
fconst[5][9] = -7.603490e+00f;
fconst[5][10] = -1.928652e+00f;
fconst[6][1] = 0.000000e+00f;
fconst[6][2] = 0.000000e+00f;
fconst[6][3] = 0.000000e+00f;
fconst[6][4] = 0.000000e+00f;
fconst[6][5] = 0.000000e+00f;
fconst[6][6] = 0.000000e+00f;
fconst[6][7] = -1.264241e+00f;
fconst[6][8] = -2.853981e+00f;
fconst[6][9] = -1.729329e+00f;
fconst[6][10] = -3.671660e+00f;
fconst[7][1] = 0.000000e+00f;
fconst[7][2] = 0.000000e+00f;
fconst[7][3] = 0.000000e+00f;
fconst[7][4] = 0.000000e+00f;
fconst[7][5] = 0.000000e+00f;
fconst[7][6] = 0.000000e+00f;
fconst[7][7] = 0.000000e+00f;
fconst[7][8] = 0.000000e+00f;
fconst[7][9] = -5.104583e+00f;
fconst[7][10] = -1.622249e+00f;
fconst[8][1] = 0.000000e+00f;
fconst[8][2] = 0.000000e+00f;
fconst[8][3] = 0.000000e+00f;
```

```
fconst[8][4] = 0.000000e+00f;
fconst[8][5] = 0.000000e+00f;
fconst[8][6] = 0.000000e+00f;
fconst[8][7] = 0.000000e+00f;
fconst[8][8] = 0.000000e+00f;
fconst[8][9] = -1.264241e+00f;
fconst[8][10] = -2.853981e+00f;
fconst[9][1] = 0.000000e+00f;
fconst[9][2] = 0.000000e+00f;
fconst[9][3] = 0.000000e+00f;
fconst[9][4] = 0.000000e+00f;
fconst[9][5] = 0.000000e+00f;
fconst[9][6] = 0.000000e+00f;
fconst[9][7] = 0.000000e+00f;
fconst[9][8] = 0.000000e+00f;
fconst[9][9] = 0.000000e+00f;
fconst[9][10] = 0.000000e+00f;
fconst[10][1] = 0.000000e+00f;
fconst[10][2] = 0.000000e+00f;
fconst[10][3] = 0.000000e+00f;
fconst[10][4] = 0.000000e+00f;
fconst[10][5] = 0.000000e+00f;
fconst[10][6] = 0.000000e+00f;
fconst[10][7] = 0.000000e+00f;
fconst[10][8] = 0.000000e+00f;
fconst[10][9] = 0.000000e+00f;
fconst[10][10] = 0.000000e+00f;
btmp1 = darray(1,40,1,1);
bvar = darray(1,40,1,2);
bvar[1][1] = 0.000000e+00f;
bvar[1][2] = 0.000000e+00f;
bvar[2][1] = 0.000000e+00f;
bvar[2][2] = 0.000000e+00f;
bvar[3][1] = 0.000000e+00f;
bvar[3][2] = 0.000000e+00f;
bvar[4][1] = 0.000000e+00f;
bvar[4][2] = 0.000000e+00f;
bvar[5][1] = 0.000000e+00f;
bvar[5][2] = 0.000000e+00f;
bvar[6][1] = 0.000000e+00f;
bvar[6][2] = 0.000000e+00f;
bvar[7][1] = 0.000000e+00f;
bvar[7][2] = 0.000000e+00f;
bvar[8][1] = 0.000000e+00f;
bvar[8][2] = 0.000000e+00f;
bvar[9][1] = 0.000000e+00f;
bvar[9][2] = 0.000000e+00f;
bvar[10][1] = 0.000000e+00f;
bvar[10][2] = 0.000000e+00f;
bvar[11][1] = 0.000000e+00f;
bvar[11][2] = 0.000000e+00f;
bvar[12][1] = 0.000000e+00f;
bvar[12][2] = 0.000000e+00f;
bvar[13][1] = 0.000000e+00f;
bvar[13][2] = 0.000000e+00f;
bvar[14][1] = 0.000000e+00f;
bvar[14][2] = 0.000000e+00f;
bvar[15][1] = 0.000000e+00f;
bvar[15][2] = 0.000000e+00f;
bvar[16][1] = 0.000000e+00f;
bvar[16][2] = 0.000000e+00f;
bvar[17][1] = 0.000000e+00f;
bvar[17][2] = 0.000000e+00f;
```

```
bvar[18][1] = 0.000000e+00f;
bvar[18][2] = 0.000000e+00f;
bvar[19][1] = 0.000000e+00f;
bvar[19][2] = 0.000000e+00f;
bvar[20][1] = 0.000000e+00f;
bvar[20][2] = 0.000000e+00f;
bvar[21][1] = 1.000000e+00f;
bvar[21][2] = 0.000000e+00f;
bvar[22][1] = 0.000000e+00f;
bvar[22][2] = 1.000000e+00f;
bvar[23][1] = 1.000000e+00f;
bvar[23][2] = 0.000000e+00f;
bvar[24][1] = 0.000000e+00f;
bvar[24][2] = 1.000000e+00f;
bvar[25][1] = 1.000000e+00f;
bvar[25][2] = 0.000000e+00f;
bvar[26][1] = 0.000000e+00f;
bvar[26][2] = 1.000000e+00f;
bvar[27][1] = 1.000000e+00f;
bvar[27][2] = 0.000000e+00f;
bvar[28][1] = 0.000000e+00f;
bvar[28][2] = 1.000000e+00f;
bvar[29][1] = 1.000000e+00f;
bvar[29][2] = 0.000000e+00f;
bvar[30][1] = 0.000000e+00f;
bvar[30][2] = 1.000000e+00f;
bvar[31][1] = -1.000000e+00f;
bvar[31][2] = -0.000000e+00f;
bvar[32][1] = -0.000000e+00f;
bvar[32][2] = -1.000000e+00f;
bvar[33][1] = -1.000000e+00f;
bvar[33][2] = -0.000000e+00f;
bvar[34][1] = -0.000000e+00f;
bvar[34][2] = -1.000000e+00f;
bvar[35][1] = -1.000000e+00f;
bvar[35][2] = -0.000000e+00f;
bvar[36][1] = -0.000000e+00f;
bvar[36][2] = -1.000000e+00f;
bvar[37][1] = -1.000000e+00f;
bvar[37][2] = -0.000000e+00f;
bvar[38][1] = -0.000000e+00f;
bvar[38][2] = -1.000000e+00f;
bvar[39][1] = -1.000000e+00f;
bvar[39][2] = -0.000000e+00f;
bvar[40][1] = -0.000000e+00f;
bvar[40][2] = -1.000000e+00f;
btmp2 = darray(1,40,1,1);
b = darray(1,40,1,1);
b[1][1] = 1.000000e-02f;
b[2][1] = 1.000000e-02f;
b[3][1] = 1.000000e-02f;
b[4][1] = 1.000000e-02f;
b[5][1] = 1.000000e-02f;
b[6][1] = 1.000000e-02f;
b[7][1] = 1.000000e-02f;
b[8][1] = 1.000000e-02f;
b[9][1] = 1.000000e-02f;
b[10][1] = 1.000000e-02f;
b[11][1] = 1.000000e-02f;
b[12][1] = 1.000000e-02f;
b[13][1] = 1.000000e-02f;
b[14][1] = 1.000000e-02f;
b[15][1] = 1.000000e-02f;
```

```
  b[16][1] = 1.000000e-02f;
  b[17][1] = 1.000000e-02f;
  b[18][1] = 1.000000e-02f;
  b[19][1] = 1.000000e-02f;
  b[20][1] = 1.000000e-02f;
  b[21][1] = 1.000000e+00f;
  b[22][1] = 1.000000e+00f;
  b[23][1] = 1.000000e+00f;
  b[24][1] = 1.000000e+00f;
  b[25][1] = 1.000000e+00f;
  b[26][1] = 1.000000e+00f;
  b[27][1] = 1.000000e+00f;
  b[28][1] = 1.000000e+00f;
  b[29][1] = 1.000000e+00f;
  b[30][1] = 1.000000e+00f;
  b[31][1] = 1.000000e+00f;
  b[32][1] = 1.000000e+00f;
  b[33][1] = 1.000000e+00f;
  b[34][1] = 1.000000e+00f;
  b[35][1] = 1.000000e+00f;
  b[36][1] = 1.000000e+00f;
  b[37][1] = 1.000000e+00f;
  b[38][1] = 1.000000e+00f;
  b[39][1] = 1.000000e+00f;
  b[40][1] = 1.000000e+00f;
  GPC_b = darray3(1,2,1,2,1,4);
  GPC_b[1][1][1] = 0.000000e+00f;
  GPC_b[1][1][2] = 2.552292e+00f;
  GPC_b[1][1][3] = -9.389356e-01f;
  GPC_b[1][1][4] = 0.000000e+00f;
  GPC_b[1][2][1] = 0.000000e+00f;
  GPC_b[1][2][2] = 6.321206e-01f;
  GPC_b[1][2][3] = -3.094493e-01f;
  GPC_b[1][2][4] = 0.000000e+00f;
  GPC_b[2][1][1] = 0.000000e+00f;
  GPC_b[2][1][2] = 8.111244e-01f;
  GPC_b[2][1][3] = -2.323910e-01f;
  GPC_b[2][1][4] = 0.000000e+00f;
  GPC_b[2][2][1] = 0.000000e+00f;
  GPC_b[2][2][2] = 1.426990e+00f;
  GPC_b[2][2][3] = -2.695237e-01f;
  GPC_b[2][2][4] = 0.000000e+00f;
  GPC_a = darray(1,2,1,2);
  GPC_a[1][1] = -8.574211e-01f;
  GPC_a[1][2] = 1.800923e-01f;
  GPC_a[2][1] = -4.753804e-01f;
  GPC_a[2][2] = 5.411377e-02f;
  dumax = darray(1,1,1,2);
  dumax[1][1] = 1.000000e-02f;
  dumax[1][2] = 1.000000e-02f;
  dumin = darray(1,1,1,2);
  dumin[1][1] = -1.000000e-02f;
  dumin[1][2] = -1.000000e-02f;
  umax = darray(1,1,1,2);
  umax[1][1] = 1.000000e+00f;
  umax[1][2] = 1.000000e+00f;
  umin = darray(1,1,1,2);
  umin[1][1] = -1.000000e+00f;
  umin[1][2] = -1.000000e+00f;
  return;
}
for(j=1;j<=2;++j) for(k=1;k<=1;++k) tmpu[j][k] = 0;
for(j=1;j<=2;++j) for(k=1;k<=1;++k) du[j][k] = 0;
```

```
for(j=1;j<=10;++j) for(k=1;k<=1;++k) Yzad[j][k] = 0;
for(j=1;j<=2;++j) for(k=1;k<=5;++k) y0[j][k] = 0;
for(j=1;j<=10;++j) for(k=1;k<=1;++k) Y0[j][k] = 0;
itmp=1;
for(i=1;i<=5;++i){
  for(j=1;j<=2;++j){
    Yzad[itmp][1]=ad->z[j-1];
    itmp=itmp+1;
  }
}
for(j=1;j<=2;++j) for(k=1;k<=1;++k) uk[j][k] = 0;
for(i=1;i<=2;++i){
  uk[i][1]=ad->u[ad->k-1][i-1];
}
for(j=1;j<=2;++j) for(k=1;k<=1;++k) disturbance[j][k] = 0;
for(m=1;m<=2;++m){
  disturbance[m][1]=0;
  for(n=1;n<=2;++n){
    for(i=1;i<=4;++i){
      disturbance[m][1]=disturbance[m][1]+GPC_b[m][n][i]*ad->u[ad->k-i][n-1];
    }
  }
  for(i=1;i<=2;++i){
    disturbance[m][1]=disturbance[m][1]-GPC_a[m][i]*ad->y[ad->k-i][m-1];
  }
  disturbance[m][1]=ad->y[ad->k][m-1]-disturbance[m][1];
  for(p=1;p<=5;++p){
    Iyf=p-1;
    if(2<Iyf){
      Iyf=2;
    }
    Iuftmp=4;
    if(p<Iuftmp){
      Iuftmp=p;
    }
    Iuf=0;
    if(Iuftmp>Iuf){
      Iuf=Iuftmp;
    }
    y0[m][p]=disturbance[m][1];
    for(n=1;n<=2;++n){
      for(r=1;r<=Iuf;++r){
        y0[m][p]=y0[m][p]+GPC_b[m][n][r]*ad->u[ad->k-1][n-1];
      }
    }
    for(n=1;n<=2;++n){
      for(r=(Iuf+1);r<=4;++r){
        y0[m][p]=y0[m][p]+GPC_b[m][n][r]*ad->u[ad->k-r+p][n-1];
      }
    }
    for(r=1;r<=Iyf;++r){
      y0[m][p]=y0[m][p]-GPC_a[m][r]*y0[m][p-r];
    }
    for(r=(Iyf+1);r<=2;++r){
      y0[m][p]=y0[m][p]-GPC_a[m][r]*ad->y[ad->k-r+p][m-1];
    }
  }
}
itmp=1;
for(i=1;i<=5;++i){
  for(j=1;j<=2;++j){
    Y0[itmp][1]=y0[j][i];
    itmp=itmp+1;
```

```
    }
  }
  sumaa(Yzad,Y0,ftmp1,10,1,-1);
  productab(fconst,ftmp1,ftmp4,10,10,10,1);
  for(i=1;i<=10;++i){
    ftmp[i]=ftmp4[i][1];
  }
  productab(bvar,uk,btmp1,40,2,2,1);
  sumaa(b,btmp1,btmp2,40,1,1);
  for(i=1;i<=40;++i){
    btmp[i]=btmp2[i][1];
  }
  for(i=1;i<=2;++i){
    qpx[i]=0;
  }
  memcpy (osqp_q, &(ftmp[1]), 10*sizeof(c_float));
  osqp_update_lin_cost(osqp_work, osqp_q);
  memcpy (osqp_ub, &(btmp[1]), 40*sizeof(c_float));
  osqp_update_bounds(osqp_work, osqp_lb, osqp_ub);
  osqp_solve(osqp_work);
  memcpy (&(qpx[1]), osqp_work->solution->x, 10*sizeof(c_float));

  for(i=1;i<=2;++i){
    du[i][1]=qpx[i];
  }
  for(n=1;n<=2;++n){
    if(du[n][1]>dumax[1][n]){
      du[n][1]=dumax[1][n];
    }
    if(du[n][1]<dumin[1][n]){
      du[n][1]=dumin[1][n];
    }
    tmpu[n][1]=ad->u[ad->k-1][n-1]+du[n][1];
    if(tmpu[n][1]>umax[1][n]){
      tmpu[n][1]=umax[1][n];
    }
    if(tmpu[n][1]<umin[1][n]){
      tmpu[n][1]=umin[1][n];
    }
    du[n][1]=tmpu[n][1]-ad->u[ad->k-1][n-1];
  }
  for(j=1;j<=1;++j) for(k=1;k<=2;++k) control_value[j][k] = 0;
  for(n=1;n<=2;++n){
    control_value[1][n]=du[n][1];
  }

  set_current_control_increment(c,&(control_value[1][1])); // du is indexed
      starting with 1, therefore to maintain compatibility it is required to
      refer to first element of an actual array
}

void controller_setup(){
profiler_start(2);
    init_archive_data(&ad, 200, 200, 2, 2, 0, 0, 0.01);
    init_current_control(&cc,&ad);
    controller(NULL,NULL);
profiler_end(2);
}

void idle(){
profiler_start(13);
    const int k = 0;
  static int i = 0;
```

```
  static char str[1000] = {0};

    sprintf(str, "x = [%f,%f,",ad.y[k][0], ad.y[k][1]);     write_string(str);
    sprintf(str,       "%f,%f,",ad.z[0], ad.z[1]);          write_string(str);
    sprintf(str,       "%f,%f,",ad.u[k-1][0],ad.u[k-1][1]); write_string(str);
    write_string("];\n\r");
  if(++i > 1000) profiler_print();
profiler_end(13);
}

void loop(){
profiler_start(10);
    static int i = 0;
       if(i< 100){ ad.z[0] = -0.1f; ad.z[1] = +0.2f; }
  else if(i< 150){ ad.z[0] = -0.1f; ad.z[1] = -0.2f; }
  else if(i< 250){ ad.z[0] = +0.1f; ad.z[1] = -0.2f; }
  else           { ad.z[0] = +0.1f; ad.z[1] = +0.2f; }
  if(++i > 300) i = 0;

profiler_start(50);
  controller(&ad,&cc);
profiler_end(50);

  push_current_controls_to_archive_data(&cc,&ad);
profiler_end(10);
}

void timeout(){
    while(1);
}
```

E Analytical and Numerical DMC Algorithms Exchanged On-Line: Generated Source of `main_mpc.c`

```
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\defines.h"
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\profiler.h"
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\mpctools.h"
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\simulated_signals.h"
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\matrix_cal.h"
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\allocation_nr.h"
#include "osqp.h"
#include "util.h"
#include "stm32f7xx_hal.h"
#include <string.h>
#include "main.h"

ArchiveData ad;
CurrentControl cc;

int algorithm_used = 0;

long get_time(){ return HAL_GetTick(); }

extern void timer_loop(void);
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){
    if (htim->Instance == TIM2) {
        timer_loop();
    }
}
void measurements(){
profiler_start(4);
    new_output(&ad,__measureOutput());
profiler_end(4);
}
void controls(){
profiler_start(3);
    __setControlValue(last_control(&ad));
profiler_end(3);
}
void hardware_setup(){
profiler_start(1);
    low_lvl_main();
profiler_end(1);
}

void controllerDMCA(ArchiveData * ad, CurrentControl * c){
  static float** AutoMATiC_DMC_Y;
  static float** AutoMATiC_DMC_Yzad;
  static float** AutoMATiC_DMC_dUp;
  static float** AutoMATiC_DMC_du;
  static float** AutoMATiC_DMC_dutmp1;
  static float** AutoMATiC_DMC_dutmp2;
  static float** AutoMATiC_DMC_e;
  static long AutoMATiC_DMC_i;
  static long AutoMATiC_DMC_itmp;
  static long AutoMATiC_DMC_j;
  static long AutoMATiC_DMC_n;
  static float** AutoMATiC_DMC_tmpu;
  static float** control_value;
  static long j;
  static long k;
  static float** AutoMATiC_DMC_Ke;
  static float** AutoMATiC_DMC_Ku;
  static float** dumax;
```

```
static float** dumin;
static float** umax;
static float** umin;
if(ad == NULL){
  AutoMATiC_DMC_dUp = darray(1,8,1,1);
  AutoMATiC_DMC_Yzad = darray(1,10,1,1);
  AutoMATiC_DMC_Y = darray(1,10,1,1);
  AutoMATiC_DMC_tmpu = darray(1,2,1,1);
  AutoMATiC_DMC_e = darray(1,2,1,1);
  control_value = darray(1,1,1,2);
  AutoMATiC_DMC_dutmp1 = darray(1,2,1,1);
  AutoMATiC_DMC_Ke = darray(1,2,1,2);
  AutoMATiC_DMC_Ke[1][1] = 3.373006e-01f;
  AutoMATiC_DMC_Ke[1][2] = -8.539666e-02f;
  AutoMATiC_DMC_Ke[2][1] = -1.016548e-01f;
  AutoMATiC_DMC_Ke[2][2] = 5.115693e-01f;
  AutoMATiC_DMC_dutmp2 = darray(1,2,1,1);
  AutoMATiC_DMC_Ku = darray(1,2,1,8);
  AutoMATiC_DMC_Ku[1][1] = 1.246690e+00f;
  AutoMATiC_DMC_Ku[1][2] = 1.325999e-01f;
  AutoMATiC_DMC_Ku[1][3] = 6.350099e-01f;
  AutoMATiC_DMC_Ku[1][4] = 6.239113e-02f;
  AutoMATiC_DMC_Ku[1][5] = 3.043741e-01f;
  AutoMATiC_DMC_Ku[1][6] = 2.660202e-02f;
  AutoMATiC_DMC_Ku[1][7] = 1.005321e-01f;
  AutoMATiC_DMC_Ku[1][8] = 7.749453e-03f;
  AutoMATiC_DMC_Ku[2][1] = 6.013931e-02f;
  AutoMATiC_DMC_Ku[2][2] = 8.674500e-01f;
  AutoMATiC_DMC_Ku[2][3] = -1.172849e-01f;
  AutoMATiC_DMC_Ku[2][4] = 2.406519e-01f;
  AutoMATiC_DMC_Ku[2][5] = -7.501822e-02f;
  AutoMATiC_DMC_Ku[2][6] = 6.519486e-02f;
  AutoMATiC_DMC_Ku[2][7] = -2.764289e-02f;
  AutoMATiC_DMC_Ku[2][8] = 1.396884e-02f;
  AutoMATiC_DMC_du = darray(1,2,1,1);
  dumax = darray(1,1,1,2);
  dumax[1][1] = 1.000000e-02f;
  dumax[1][2] = 1.000000e-02f;
  dumin = darray(1,1,1,2);
  dumin[1][1] = -1.000000e-02f;
  dumin[1][2] = -1.000000e-02f;
  umax = darray(1,1,1,2);
  umax[1][1] = 1.000000e+00f;
  umax[1][2] = 1.000000e+00f;
  umin = darray(1,1,1,2);
  umin[1][1] = -1.000000e+00f;
  umin[1][2] = -1.000000e+00f;
  return;
}
for(j=1;j<=8;++j) for(k=1;k<=1;++k) AutoMATiC_DMC_dUp[j][k] = 0;
for(j=1;j<=10;++j) for(k=1;k<=1;++k) AutoMATiC_DMC_Yzad[j][k] = 0;
for(j=1;j<=10;++j) for(k=1;k<=1;++k) AutoMATiC_DMC_Y[j][k] = 0;
for(j=1;j<=2;++j) for(k=1;k<=1;++k) AutoMATiC_DMC_tmpu[j][k] = 0;
AutoMATiC_DMC_itmp=1;
for(AutoMATiC_DMC_i=1;AutoMATiC_DMC_i<=4;++AutoMATiC_DMC_i){
  for(AutoMATiC_DMC_j=1;AutoMATiC_DMC_j<=2;++AutoMATiC_DMC_j){
    AutoMATiC_DMC_dUp[AutoMATiC_DMC_itmp][1]=ad->du[ad->k-AutoMATiC_DMC_i][
        AutoMATiC_DMC_j-1];
    AutoMATiC_DMC_itmp=AutoMATiC_DMC_itmp+1;
  }
}
AutoMATiC_DMC_itmp=1;
for(AutoMATiC_DMC_i=1;AutoMATiC_DMC_i<=5;++AutoMATiC_DMC_i){
```

```
    for(AutoMATiC_DMC_j=1;AutoMATiC_DMC_j<=2;++AutoMATiC_DMC_j){
      AutoMATiC_DMC_Yzad[AutoMATiC_DMC_itmp][1]=ad->z[AutoMATiC_DMC_j-1];
      AutoMATiC_DMC_itmp=AutoMATiC_DMC_itmp+1;
    }
  }
  AutoMATiC_DMC_itmp=1;
  for(AutoMATiC_DMC_i=1;AutoMATiC_DMC_i<=5;++AutoMATiC_DMC_i){
    for(AutoMATiC_DMC_j=1;AutoMATiC_DMC_j<=2;++AutoMATiC_DMC_j){
      AutoMATiC_DMC_Y[AutoMATiC_DMC_itmp][1]=ad->y[ad->k][AutoMATiC_DMC_j-1];
      AutoMATiC_DMC_itmp=AutoMATiC_DMC_itmp+1;
    }
  }
  for(j=1;j<=2;++j) for(k=1;k<=1;++k) AutoMATiC_DMC_e[j][k] = 0;
  for(AutoMATiC_DMC_i=1;AutoMATiC_DMC_i<=2;++AutoMATiC_DMC_i){
    AutoMATiC_DMC_e[AutoMATiC_DMC_i][1]=ad->z[AutoMATiC_DMC_i-1]-ad->y[ad->k][
        AutoMATiC_DMC_i-1];
  }
  productab(AutoMATiC_DMC_Ke,AutoMATiC_DMC_e,AutoMATiC_DMC_dutmp1,2,2,2,1);
  productab(AutoMATiC_DMC_Ku,AutoMATiC_DMC_dUp,AutoMATiC_DMC_dutmp2,2,8,8,1);
  sumaa(AutoMATiC_DMC_dutmp1,AutoMATiC_DMC_dutmp2,AutoMATiC_DMC_du,2,1,-1);
  for(AutoMATiC_DMC_n=1;AutoMATiC_DMC_n<=2;++AutoMATiC_DMC_n){
    if(AutoMATiC_DMC_du[AutoMATiC_DMC_n][1]>dumax[1][AutoMATiC_DMC_n]){
      AutoMATiC_DMC_du[AutoMATiC_DMC_n][1]=dumax[1][AutoMATiC_DMC_n];
    }
    if(AutoMATiC_DMC_du[AutoMATiC_DMC_n][1]<dumin[1][AutoMATiC_DMC_n]){
      AutoMATiC_DMC_du[AutoMATiC_DMC_n][1]=dumin[1][AutoMATiC_DMC_n];
    }
    AutoMATiC_DMC_tmpu[AutoMATiC_DMC_n][1]=ad->u[ad->k-1][AutoMATiC_DMC_n-1]+
        AutoMATiC_DMC_du[AutoMATiC_DMC_n][1];
    if(AutoMATiC_DMC_tmpu[AutoMATiC_DMC_n][1]>umax[1][AutoMATiC_DMC_n]){
      AutoMATiC_DMC_tmpu[AutoMATiC_DMC_n][1]=umax[1][AutoMATiC_DMC_n];
    }
    if(AutoMATiC_DMC_tmpu[AutoMATiC_DMC_n][1]<umin[1][AutoMATiC_DMC_n]){
      AutoMATiC_DMC_tmpu[AutoMATiC_DMC_n][1]=umin[1][AutoMATiC_DMC_n];
    }
    AutoMATiC_DMC_du[AutoMATiC_DMC_n][1]=AutoMATiC_DMC_tmpu[AutoMATiC_DMC_n
        ][1]-ad->u[ad->k-1][AutoMATiC_DMC_n-1];
  }
  for(j=1;j<=1;++j) for(k=1;k<=2;++k) control_value[j][k] = 0;
  for(AutoMATiC_DMC_n=1;AutoMATiC_DMC_n<=2;++AutoMATiC_DMC_n){
    control_value[1][AutoMATiC_DMC_n]=AutoMATiC_DMC_du[AutoMATiC_DMC_n][1];
  }

  set_current_control_increment(c,&(control_value[1][1])); // du is indexed
      starting with 1, therefore to maintain compatibility it is required to
      refer to first element of an actual array
}
void controllerDMCN(ArchiveData * ad, CurrentControl * c){
  static float** AutoMATiC_DMC_Y;
  static float** AutoMATiC_DMC_Yzad;
  static float** AutoMATiC_DMC_btmp1;
  static float** AutoMATiC_DMC_btmp2;
  static float** AutoMATiC_DMC_dUp;
  static float** AutoMATiC_DMC_du;
  static float** AutoMATiC_DMC_dutmp1;
  static float** AutoMATiC_DMC_dutmp2;
  static float** AutoMATiC_DMC_e;
  static float** AutoMATiC_DMC_ftmp1;
  static float** AutoMATiC_DMC_ftmp2;
  static float** AutoMATiC_DMC_ftmp3;
  static float** AutoMATiC_DMC_ftmp4;
  static long AutoMATiC_DMC_i;
  static long AutoMATiC_DMC_itmp;
```

```
static long AutoMATiC_DMC_j;
static long AutoMATiC_DMC_n;
static float** AutoMATiC_DMC_tmpu;
static float** AutoMATiC_DMC_uk;
static float** control_value;
static long j;
static long k;
static float** AutoMATiC_DMC_Ke;
static float** AutoMATiC_DMC_Ku;
static float** AutoMATiC_DMC_Mp;
static float** AutoMATiC_DMC_fconst;
static float** AutoMATiC_DMC_bvar;
static float** AutoMATiC_DMC_b;
static float** dumax;
static float** dumin;
static float** umax;
static float** umin;
static c_float AutoMATiC_DMC_A_x[80];
static c_int AutoMATiC_DMC_A_i[80];
static c_int AutoMATiC_DMC_A_p[11];

static c_float AutoMATiC_DMC_H_x[66];
static c_int AutoMATiC_DMC_H_i[66];
static c_int AutoMATiC_DMC_H_p[11];

static OSQPSettings *osqp_settings;
static OSQPWorkspace *osqp_work;
static OSQPData *osqp_data;
static c_float osqp_q[10];
static c_float osqp_lb[40];
static c_float osqp_ub[40];
static c_float AutoMATiC_DMC_ftmp[11];
static c_float AutoMATiC_DMC_btmp[41];
static c_float AutoMATiC_DMC_qpx[11];
if(ad == NULL){
  AutoMATiC_DMC_A_x[0] = -1.000000e+00f;
  AutoMATiC_DMC_A_x[1] = -1.000000e+00f;
  AutoMATiC_DMC_A_x[2] = -1.000000e+00f;
  AutoMATiC_DMC_A_x[3] = -1.000000e+00f;
  AutoMATiC_DMC_A_x[4] = -1.000000e+00f;
  AutoMATiC_DMC_A_x[5] = -1.000000e+00f;
  AutoMATiC_DMC_A_x[6] = 1.000000e+00f;
  AutoMATiC_DMC_A_x[7] = 1.000000e+00f;
  AutoMATiC_DMC_A_x[8] = 1.000000e+00f;
  AutoMATiC_DMC_A_x[9] = 1.000000e+00f;
  AutoMATiC_DMC_A_x[10] = 1.000000e+00f;
  AutoMATiC_DMC_A_x[11] = 1.000000e+00f;
  AutoMATiC_DMC_A_x[12] = -1.000000e+00f;
  AutoMATiC_DMC_A_x[13] = -1.000000e+00f;
  AutoMATiC_DMC_A_x[14] = -1.000000e+00f;
  AutoMATiC_DMC_A_x[15] = -1.000000e+00f;
  AutoMATiC_DMC_A_x[16] = -1.000000e+00f;
  AutoMATiC_DMC_A_x[17] = -1.000000e+00f;
  AutoMATiC_DMC_A_x[18] = 1.000000e+00f;
  AutoMATiC_DMC_A_x[19] = 1.000000e+00f;
  AutoMATiC_DMC_A_x[20] = 1.000000e+00f;
  AutoMATiC_DMC_A_x[21] = 1.000000e+00f;
  AutoMATiC_DMC_A_x[22] = 1.000000e+00f;
  AutoMATiC_DMC_A_x[23] = 1.000000e+00f;
  AutoMATiC_DMC_A_x[24] = -1.000000e+00f;
  AutoMATiC_DMC_A_x[25] = -1.000000e+00f;
  AutoMATiC_DMC_A_x[26] = -1.000000e+00f;
  AutoMATiC_DMC_A_x[27] = -1.000000e+00f;
```

```
AutoMATiC_DMC_A_x[28] = -1.000000e+00f;
AutoMATiC_DMC_A_x[29] = 1.000000e+00f;
AutoMATiC_DMC_A_x[30] = 1.000000e+00f;
AutoMATiC_DMC_A_x[31] = 1.000000e+00f;
AutoMATiC_DMC_A_x[32] = 1.000000e+00f;
AutoMATiC_DMC_A_x[33] = 1.000000e+00f;
AutoMATiC_DMC_A_x[34] = -1.000000e+00f;
AutoMATiC_DMC_A_x[35] = -1.000000e+00f;
AutoMATiC_DMC_A_x[36] = -1.000000e+00f;
AutoMATiC_DMC_A_x[37] = -1.000000e+00f;
AutoMATiC_DMC_A_x[38] = -1.000000e+00f;
AutoMATiC_DMC_A_x[39] = 1.000000e+00f;
AutoMATiC_DMC_A_x[40] = 1.000000e+00f;
AutoMATiC_DMC_A_x[41] = 1.000000e+00f;
AutoMATiC_DMC_A_x[42] = 1.000000e+00f;
AutoMATiC_DMC_A_x[43] = 1.000000e+00f;
AutoMATiC_DMC_A_x[44] = -1.000000e+00f;
AutoMATiC_DMC_A_x[45] = -1.000000e+00f;
AutoMATiC_DMC_A_x[46] = -1.000000e+00f;
AutoMATiC_DMC_A_x[47] = -1.000000e+00f;
AutoMATiC_DMC_A_x[48] = 1.000000e+00f;
AutoMATiC_DMC_A_x[49] = 1.000000e+00f;
AutoMATiC_DMC_A_x[50] = 1.000000e+00f;
AutoMATiC_DMC_A_x[51] = 1.000000e+00f;
AutoMATiC_DMC_A_x[52] = -1.000000e+00f;
AutoMATiC_DMC_A_x[53] = -1.000000e+00f;
AutoMATiC_DMC_A_x[54] = -1.000000e+00f;
AutoMATiC_DMC_A_x[55] = -1.000000e+00f;
AutoMATiC_DMC_A_x[56] = 1.000000e+00f;
AutoMATiC_DMC_A_x[57] = 1.000000e+00f;
AutoMATiC_DMC_A_x[58] = 1.000000e+00f;
AutoMATiC_DMC_A_x[59] = 1.000000e+00f;
AutoMATiC_DMC_A_x[60] = -1.000000e+00f;
AutoMATiC_DMC_A_x[61] = -1.000000e+00f;
AutoMATiC_DMC_A_x[62] = -1.000000e+00f;
AutoMATiC_DMC_A_x[63] = 1.000000e+00f;
AutoMATiC_DMC_A_x[64] = 1.000000e+00f;
AutoMATiC_DMC_A_x[65] = 1.000000e+00f;
AutoMATiC_DMC_A_x[66] = -1.000000e+00f;
AutoMATiC_DMC_A_x[67] = -1.000000e+00f;
AutoMATiC_DMC_A_x[68] = -1.000000e+00f;
AutoMATiC_DMC_A_x[69] = 1.000000e+00f;
AutoMATiC_DMC_A_x[70] = 1.000000e+00f;
AutoMATiC_DMC_A_x[71] = 1.000000e+00f;
AutoMATiC_DMC_A_x[72] = -1.000000e+00f;
AutoMATiC_DMC_A_x[73] = -1.000000e+00f;
AutoMATiC_DMC_A_x[74] = 1.000000e+00f;
AutoMATiC_DMC_A_x[75] = 1.000000e+00f;
AutoMATiC_DMC_A_x[76] = -1.000000e+00f;
AutoMATiC_DMC_A_x[77] = -1.000000e+00f;
AutoMATiC_DMC_A_x[78] = 1.000000e+00f;
AutoMATiC_DMC_A_x[79] = 1.000000e+00f;
AutoMATiC_DMC_A_i[0] = 0;
AutoMATiC_DMC_A_i[1] = 10;
AutoMATiC_DMC_A_i[2] = 12;
AutoMATiC_DMC_A_i[3] = 14;
AutoMATiC_DMC_A_i[4] = 16;
AutoMATiC_DMC_A_i[5] = 18;
AutoMATiC_DMC_A_i[6] = 20;
AutoMATiC_DMC_A_i[7] = 30;
AutoMATiC_DMC_A_i[8] = 32;
AutoMATiC_DMC_A_i[9] = 34;
AutoMATiC_DMC_A_i[10] = 36;
```

```
AutoMATiC_DMC_A_i[11] = 38;
AutoMATiC_DMC_A_i[12] = 1;
AutoMATiC_DMC_A_i[13] = 11;
AutoMATiC_DMC_A_i[14] = 13;
AutoMATiC_DMC_A_i[15] = 15;
AutoMATiC_DMC_A_i[16] = 17;
AutoMATiC_DMC_A_i[17] = 19;
AutoMATiC_DMC_A_i[18] = 21;
AutoMATiC_DMC_A_i[19] = 31;
AutoMATiC_DMC_A_i[20] = 33;
AutoMATiC_DMC_A_i[21] = 35;
AutoMATiC_DMC_A_i[22] = 37;
AutoMATiC_DMC_A_i[23] = 39;
AutoMATiC_DMC_A_i[24] = 2;
AutoMATiC_DMC_A_i[25] = 12;
AutoMATiC_DMC_A_i[26] = 14;
AutoMATiC_DMC_A_i[27] = 16;
AutoMATiC_DMC_A_i[28] = 18;
AutoMATiC_DMC_A_i[29] = 22;
AutoMATiC_DMC_A_i[30] = 32;
AutoMATiC_DMC_A_i[31] = 34;
AutoMATiC_DMC_A_i[32] = 36;
AutoMATiC_DMC_A_i[33] = 38;
AutoMATiC_DMC_A_i[34] = 3;
AutoMATiC_DMC_A_i[35] = 13;
AutoMATiC_DMC_A_i[36] = 15;
AutoMATiC_DMC_A_i[37] = 17;
AutoMATiC_DMC_A_i[38] = 19;
AutoMATiC_DMC_A_i[39] = 23;
AutoMATiC_DMC_A_i[40] = 33;
AutoMATiC_DMC_A_i[41] = 35;
AutoMATiC_DMC_A_i[42] = 37;
AutoMATiC_DMC_A_i[43] = 39;
AutoMATiC_DMC_A_i[44] = 4;
AutoMATiC_DMC_A_i[45] = 14;
AutoMATiC_DMC_A_i[46] = 16;
AutoMATiC_DMC_A_i[47] = 18;
AutoMATiC_DMC_A_i[48] = 24;
AutoMATiC_DMC_A_i[49] = 34;
AutoMATiC_DMC_A_i[50] = 36;
AutoMATiC_DMC_A_i[51] = 38;
AutoMATiC_DMC_A_i[52] = 5;
AutoMATiC_DMC_A_i[53] = 15;
AutoMATiC_DMC_A_i[54] = 17;
AutoMATiC_DMC_A_i[55] = 19;
AutoMATiC_DMC_A_i[56] = 25;
AutoMATiC_DMC_A_i[57] = 35;
AutoMATiC_DMC_A_i[58] = 37;
AutoMATiC_DMC_A_i[59] = 39;
AutoMATiC_DMC_A_i[60] = 6;
AutoMATiC_DMC_A_i[61] = 16;
AutoMATiC_DMC_A_i[62] = 18;
AutoMATiC_DMC_A_i[63] = 26;
AutoMATiC_DMC_A_i[64] = 36;
AutoMATiC_DMC_A_i[65] = 38;
AutoMATiC_DMC_A_i[66] = 7;
AutoMATiC_DMC_A_i[67] = 17;
AutoMATiC_DMC_A_i[68] = 19;
AutoMATiC_DMC_A_i[69] = 27;
AutoMATiC_DMC_A_i[70] = 37;
AutoMATiC_DMC_A_i[71] = 39;
AutoMATiC_DMC_A_i[72] = 8;
AutoMATiC_DMC_A_i[73] = 18;
```

```
AutoMATiC_DMC_A_i[74] = 28;
AutoMATiC_DMC_A_i[75] = 38;
AutoMATiC_DMC_A_i[76] = 9;
AutoMATiC_DMC_A_i[77] = 19;
AutoMATiC_DMC_A_i[78] = 29;
AutoMATiC_DMC_A_i[79] = 39;
AutoMATiC_DMC_A_p[0] = 0;
AutoMATiC_DMC_A_p[1] = 12;
AutoMATiC_DMC_A_p[2] = 24;
AutoMATiC_DMC_A_p[3] = 34;
AutoMATiC_DMC_A_p[4] = 44;
AutoMATiC_DMC_A_p[5] = 52;
AutoMATiC_DMC_A_p[6] = 60;
AutoMATiC_DMC_A_p[7] = 66;
AutoMATiC_DMC_A_p[8] = 72;
AutoMATiC_DMC_A_p[9] = 76;
AutoMATiC_DMC_A_p[10] = 80;

AutoMATiC_DMC_H_x[0] = 1.326566e+02f;
AutoMATiC_DMC_H_x[1] = 4.114478e+01f;
AutoMATiC_DMC_H_x[2] = 1.000269e+02f;
AutoMATiC_DMC_H_x[3] = 3.169500e+01f;
AutoMATiC_DMC_H_x[4] = 6.190011e+01f;
AutoMATiC_DMC_H_x[5] = 2.023139e+01f;
AutoMATiC_DMC_H_x[6] = 2.567725e+01f;
AutoMATiC_DMC_H_x[7] = 8.808511e+00f;
AutoMATiC_DMC_H_x[8] = 4.114478e+01f;
AutoMATiC_DMC_H_x[9] = 3.256150e+01f;
AutoMATiC_DMC_H_x[10] = 3.099487e+01f;
AutoMATiC_DMC_H_x[11] = 2.477126e+01f;
AutoMATiC_DMC_H_x[12] = 1.931418e+01f;
AutoMATiC_DMC_H_x[13] = 1.576652e+01f;
AutoMATiC_DMC_H_x[14] = 8.233726e+00f;
AutoMATiC_DMC_H_x[15] = 6.910587e+00f;
AutoMATiC_DMC_H_x[16] = 1.000269e+02f;
AutoMATiC_DMC_H_x[17] = 3.099487e+01f;
AutoMATiC_DMC_H_x[18] = 8.624002e+01f;
AutoMATiC_DMC_H_x[19] = 2.792375e+01f;
AutoMATiC_DMC_H_x[20] = 5.644363e+01f;
AutoMATiC_DMC_H_x[21] = 1.883764e+01f;
AutoMATiC_DMC_H_x[22] = 2.413991e+01f;
AutoMATiC_DMC_H_x[23] = 8.414358e+00f;
AutoMATiC_DMC_H_x[24] = 3.169500e+01f;
AutoMATiC_DMC_H_x[25] = 2.477126e+01f;
AutoMATiC_DMC_H_x[26] = 2.792375e+01f;
AutoMATiC_DMC_H_x[27] = 2.274154e+01f;
AutoMATiC_DMC_H_x[28] = 1.838345e+01f;
AutoMATiC_DMC_H_x[29] = 1.514642e+01f;
AutoMATiC_DMC_H_x[30] = 8.018635e+00f;
AutoMATiC_DMC_H_x[31] = 6.775021e+00f;
AutoMATiC_DMC_H_x[32] = 6.190011e+01f;
AutoMATiC_DMC_H_x[33] = 1.931418e+01f;
AutoMATiC_DMC_H_x[34] = 5.644363e+01f;
AutoMATiC_DMC_H_x[35] = 1.838345e+01f;
AutoMATiC_DMC_H_x[36] = 4.531061e+01f;
AutoMATiC_DMC_H_x[37] = 1.565679e+01f;
AutoMATiC_DMC_H_x[38] = 2.097070e+01f;
AutoMATiC_DMC_H_x[39] = 7.558490e+00f;
AutoMATiC_DMC_H_x[40] = 2.023139e+01f;
AutoMATiC_DMC_H_x[41] = 1.576652e+01f;
AutoMATiC_DMC_H_x[42] = 1.883764e+01f;
AutoMATiC_DMC_H_x[43] = 1.514642e+01f;
AutoMATiC_DMC_H_x[44] = 1.565679e+01f;
```

```
AutoMATiC_DMC_H_x[45] = 1.330759e+01f;
AutoMATiC_DMC_H_x[46] = 7.391926e+00f;
AutoMATiC_DMC_H_x[47] = 6.332568e+00f;
AutoMATiC_DMC_H_x[48] = 2.567725e+01f;
AutoMATiC_DMC_H_x[49] = 8.233726e+00f;
AutoMATiC_DMC_H_x[50] = 2.413991e+01f;
AutoMATiC_DMC_H_x[51] = 8.018635e+00f;
AutoMATiC_DMC_H_x[52] = 2.097070e+01f;
AutoMATiC_DMC_H_x[53] = 7.391926e+00f;
AutoMATiC_DMC_H_x[54] = 1.454423e+01f;
AutoMATiC_DMC_H_x[55] = 5.541646e+00f;
AutoMATiC_DMC_H_x[56] = 8.808511e+00f;
AutoMATiC_DMC_H_x[57] = 6.910587e+00f;
AutoMATiC_DMC_H_x[58] = 8.414358e+00f;
AutoMATiC_DMC_H_x[59] = 6.775021e+00f;
AutoMATiC_DMC_H_x[60] = 7.558490e+00f;
AutoMATiC_DMC_H_x[61] = 6.332568e+00f;
AutoMATiC_DMC_H_x[62] = 5.541646e+00f;
AutoMATiC_DMC_H_x[63] = 5.071756e+00f;
AutoMATiC_DMC_H_x[64] = 2.000000e-01f;
AutoMATiC_DMC_H_x[65] = 2.000000e-01f;
AutoMATiC_DMC_H_i[0] = 0;
AutoMATiC_DMC_H_i[1] = 1;
AutoMATiC_DMC_H_i[2] = 2;
AutoMATiC_DMC_H_i[3] = 3;
AutoMATiC_DMC_H_i[4] = 4;
AutoMATiC_DMC_H_i[5] = 5;
AutoMATiC_DMC_H_i[6] = 6;
AutoMATiC_DMC_H_i[7] = 7;
AutoMATiC_DMC_H_i[8] = 0;
AutoMATiC_DMC_H_i[9] = 1;
AutoMATiC_DMC_H_i[10] = 2;
AutoMATiC_DMC_H_i[11] = 3;
AutoMATiC_DMC_H_i[12] = 4;
AutoMATiC_DMC_H_i[13] = 5;
AutoMATiC_DMC_H_i[14] = 6;
AutoMATiC_DMC_H_i[15] = 7;
AutoMATiC_DMC_H_i[16] = 0;
AutoMATiC_DMC_H_i[17] = 1;
AutoMATiC_DMC_H_i[18] = 2;
AutoMATiC_DMC_H_i[19] = 3;
AutoMATiC_DMC_H_i[20] = 4;
AutoMATiC_DMC_H_i[21] = 5;
AutoMATiC_DMC_H_i[22] = 6;
AutoMATiC_DMC_H_i[23] = 7;
AutoMATiC_DMC_H_i[24] = 0;
AutoMATiC_DMC_H_i[25] = 1;
AutoMATiC_DMC_H_i[26] = 2;
AutoMATiC_DMC_H_i[27] = 3;
AutoMATiC_DMC_H_i[28] = 4;
AutoMATiC_DMC_H_i[29] = 5;
AutoMATiC_DMC_H_i[30] = 6;
AutoMATiC_DMC_H_i[31] = 7;
AutoMATiC_DMC_H_i[32] = 0;
AutoMATiC_DMC_H_i[33] = 1;
AutoMATiC_DMC_H_i[34] = 2;
AutoMATiC_DMC_H_i[35] = 3;
AutoMATiC_DMC_H_i[36] = 4;
AutoMATiC_DMC_H_i[37] = 5;
AutoMATiC_DMC_H_i[38] = 6;
AutoMATiC_DMC_H_i[39] = 7;
AutoMATiC_DMC_H_i[40] = 0;
AutoMATiC_DMC_H_i[41] = 1;
```

```
AutoMATiC_DMC_H_i[42] = 2;
AutoMATiC_DMC_H_i[43] = 3;
AutoMATiC_DMC_H_i[44] = 4;
AutoMATiC_DMC_H_i[45] = 5;
AutoMATiC_DMC_H_i[46] = 6;
AutoMATiC_DMC_H_i[47] = 7;
AutoMATiC_DMC_H_i[48] = 0;
AutoMATiC_DMC_H_i[49] = 1;
AutoMATiC_DMC_H_i[50] = 2;
AutoMATiC_DMC_H_i[51] = 3;
AutoMATiC_DMC_H_i[52] = 4;
AutoMATiC_DMC_H_i[53] = 5;
AutoMATiC_DMC_H_i[54] = 6;
AutoMATiC_DMC_H_i[55] = 7;
AutoMATiC_DMC_H_i[56] = 0;
AutoMATiC_DMC_H_i[57] = 1;
AutoMATiC_DMC_H_i[58] = 2;
AutoMATiC_DMC_H_i[59] = 3;
AutoMATiC_DMC_H_i[60] = 4;
AutoMATiC_DMC_H_i[61] = 5;
AutoMATiC_DMC_H_i[62] = 6;
AutoMATiC_DMC_H_i[63] = 7;
AutoMATiC_DMC_H_i[64] = 8;
AutoMATiC_DMC_H_i[65] = 9;
AutoMATiC_DMC_H_p[0] = 0;
AutoMATiC_DMC_H_p[1] = 8;
AutoMATiC_DMC_H_p[2] = 16;
AutoMATiC_DMC_H_p[3] = 24;
AutoMATiC_DMC_H_p[4] = 32;
AutoMATiC_DMC_H_p[5] = 40;
AutoMATiC_DMC_H_p[6] = 48;
AutoMATiC_DMC_H_p[7] = 56;
AutoMATiC_DMC_H_p[8] = 64;
AutoMATiC_DMC_H_p[9] = 65;
AutoMATiC_DMC_H_p[10] = 66;

osqp_settings = (OSQPSettings *)c_malloc(sizeof(OSQPSettings));
osqp_data = (OSQPData *)c_malloc(sizeof(OSQPData));
for(int osqp_it=0; osqp_it < 40; ++osqp_it) osqp_lb[osqp_it] = -100000;
osqp_data->n = 10;
osqp_data->m = 40;
osqp_data->q = osqp_q;
osqp_data->l = osqp_lb;
osqp_data->u = osqp_ub;
osqp_data->P = csc_matrix(osqp_data->n, osqp_data->n, 66, AutoMATiC_DMC_H_x
    , AutoMATiC_DMC_H_i, AutoMATiC_DMC_H_p);
osqp_data->A = csc_matrix(osqp_data->m, osqp_data->n, 80, AutoMATiC_DMC_A_x
    , AutoMATiC_DMC_A_i, AutoMATiC_DMC_A_p);
osqp_set_default_settings(osqp_settings);
osqp_work = osqp_setup(osqp_data, osqp_settings);
AutoMATiC_DMC_dUp = darray(1,8,1,1);
AutoMATiC_DMC_Yzad = darray(1,10,1,1);
AutoMATiC_DMC_Y = darray(1,10,1,1);
AutoMATiC_DMC_tmpu = darray(1,2,1,1);
AutoMATiC_DMC_e = darray(1,2,1,1);
AutoMATiC_DMC_du = darray(1,2,1,1);
AutoMATiC_DMC_uk = darray(1,2,1,1);
control_value = darray(1,1,1,2);
AutoMATiC_DMC_dutmp1 = darray(1,2,1,1);
AutoMATiC_DMC_Ke = darray(1,2,1,2);
AutoMATiC_DMC_Ke[1][1] = 4.287541e-01f;
AutoMATiC_DMC_Ke[1][2] = -1.697703e-01f;
AutoMATiC_DMC_Ke[2][1] = -2.150208e-01f;
```

```
AutoMATiC_DMC_Ke[2][2] = 7.408936e-01f;
AutoMATiC_DMC_dutmp2 = darray(1,2,1,1);
AutoMATiC_DMC_Ku = darray(1,2,1,8);
AutoMATiC_DMC_Ku[1][1] = 1.478655e+00f;
AutoMATiC_DMC_Ku[1][2] = 5.747707e-02f;
AutoMATiC_DMC_Ku[1][3] = 7.739905e-01f;
AutoMATiC_DMC_Ku[1][4] = 4.678980e-02f;
AutoMATiC_DMC_Ku[1][5] = 3.847941e-01f;
AutoMATiC_DMC_Ku[1][6] = 2.458927e-02f;
AutoMATiC_DMC_Ku[1][7] = 1.274552e-01f;
AutoMATiC_DMC_Ku[1][8] = 7.796081e-03f;
AutoMATiC_DMC_Ku[2][1] = -1.211545e-01f;
AutoMATiC_DMC_Ku[2][2] = 1.179103e+00f;
AutoMATiC_DMC_Ku[2][3] = -2.753100e-01f;
AutoMATiC_DMC_Ku[2][4] = 3.224919e-01f;
AutoMATiC_DMC_Ku[2][5] = -1.704160e-01f;
AutoMATiC_DMC_Ku[2][6] = 8.648664e-02f;
AutoMATiC_DMC_Ku[2][7] = -6.033504e-02f;
AutoMATiC_DMC_Ku[2][8] = 1.809708e-02f;
AutoMATiC_DMC_ftmp1 = darray(1,10,1,1);
AutoMATiC_DMC_ftmp2 = darray(1,10,1,1);
AutoMATiC_DMC_Mp = darray(1,10,1,8);
AutoMATiC_DMC_Mp[1][1] = 2.552292e+00f;
AutoMATiC_DMC_Mp[1][2] = 6.321206e-01f;
AutoMATiC_DMC_Mp[1][3] = 1.249453e+00f;
AutoMATiC_DMC_Mp[1][4] = 2.325442e-01f;
AutoMATiC_DMC_Mp[1][5] = 6.116594e-01f;
AutoMATiC_DMC_Mp[1][6] = 8.554821e-02f;
AutoMATiC_DMC_Mp[1][7] = 2.994327e-01f;
AutoMATiC_DMC_Mp[1][8] = 3.147143e-02f;
AutoMATiC_DMC_Mp[2][1] = 8.111244e-01f;
AutoMATiC_DMC_Mp[2][2] = 1.426990e+00f;
AutoMATiC_DMC_Mp[2][3] = 1.532016e-01f;
AutoMATiC_DMC_Mp[2][4] = 4.088396e-01f;
AutoMATiC_DMC_Mp[2][5] = 2.893605e-02f;
AutoMATiC_DMC_Mp[2][6] = 1.171345e-01f;
AutoMATiC_DMC_Mp[2][7] = 5.465313e-03f;
AutoMATiC_DMC_Mp[2][8] = 3.355960e-02f;
AutoMATiC_DMC_Mp[3][1] = 3.801745e+00f;
AutoMATiC_DMC_Mp[3][2] = 8.646647e-01f;
AutoMATiC_DMC_Mp[3][3] = 1.861112e+00f;
AutoMATiC_DMC_Mp[3][4] = 3.180924e-01f;
AutoMATiC_DMC_Mp[3][5] = 9.110921e-01f;
AutoMATiC_DMC_Mp[3][6] = 1.170196e-01f;
AutoMATiC_DMC_Mp[3][7] = 2.994327e-01f;
AutoMATiC_DMC_Mp[3][8] = 3.147143e-02f;
AutoMATiC_DMC_Mp[4][1] = 9.643260e-01f;
AutoMATiC_DMC_Mp[4][2] = 1.835830e+00f;
AutoMATiC_DMC_Mp[4][3] = 1.821377e-01f;
AutoMATiC_DMC_Mp[4][4] = 5.259741e-01f;
AutoMATiC_DMC_Mp[4][5] = 3.440136e-02f;
AutoMATiC_DMC_Mp[4][6] = 1.506941e-01f;
AutoMATiC_DMC_Mp[4][7] = 5.465313e-03f;
AutoMATiC_DMC_Mp[4][8] = 3.355960e-02f;
AutoMATiC_DMC_Mp[5][1] = 4.413404e+00f;
AutoMATiC_DMC_Mp[5][2] = 9.502129e-01f;
AutoMATiC_DMC_Mp[5][3] = 2.160545e+00f;
AutoMATiC_DMC_Mp[5][4] = 3.495638e-01f;
AutoMATiC_DMC_Mp[5][5] = 9.110921e-01f;
AutoMATiC_DMC_Mp[5][6] = 1.170196e-01f;
AutoMATiC_DMC_Mp[5][7] = 2.994327e-01f;
AutoMATiC_DMC_Mp[5][8] = 3.147143e-02f;
AutoMATiC_DMC_Mp[6][1] = 9.932621e-01f;
```

```
AutoMATiC_DMC_Mp[6][2] = 1.952965e+00f;
AutoMATiC_DMC_Mp[6][3] = 1.876030e-01f;
AutoMATiC_DMC_Mp[6][4] = 5.595337e-01f;
AutoMATiC_DMC_Mp[6][5] = 3.440136e-02f;
AutoMATiC_DMC_Mp[6][6] = 1.506941e-01f;
AutoMATiC_DMC_Mp[6][7] = 5.465313e-03f;
AutoMATiC_DMC_Mp[6][8] = 3.355960e-02f;
AutoMATiC_DMC_Mp[7][1] = 4.712837e+00f;
AutoMATiC_DMC_Mp[7][2] = 9.816844e-01f;
AutoMATiC_DMC_Mp[7][3] = 2.160545e+00f;
AutoMATiC_DMC_Mp[7][4] = 3.495638e-01f;
AutoMATiC_DMC_Mp[7][5] = 9.110921e-01f;
AutoMATiC_DMC_Mp[7][6] = 1.170196e-01f;
AutoMATiC_DMC_Mp[7][7] = 2.994327e-01f;
AutoMATiC_DMC_Mp[7][8] = 3.147143e-02f;
AutoMATiC_DMC_Mp[8][1] = 9.987274e-01f;
AutoMATiC_DMC_Mp[8][2] = 1.986524e+00f;
AutoMATiC_DMC_Mp[8][3] = 1.876030e-01f;
AutoMATiC_DMC_Mp[8][4] = 5.595337e-01f;
AutoMATiC_DMC_Mp[8][5] = 3.440136e-02f;
AutoMATiC_DMC_Mp[8][6] = 1.506941e-01f;
AutoMATiC_DMC_Mp[8][7] = 5.465313e-03f;
AutoMATiC_DMC_Mp[8][8] = 3.355960e-02f;
AutoMATiC_DMC_Mp[9][1] = 4.712837e+00f;
AutoMATiC_DMC_Mp[9][2] = 9.816844e-01f;
AutoMATiC_DMC_Mp[9][3] = 2.160545e+00f;
AutoMATiC_DMC_Mp[9][4] = 3.495638e-01f;
AutoMATiC_DMC_Mp[9][5] = 9.110921e-01f;
AutoMATiC_DMC_Mp[9][6] = 1.170196e-01f;
AutoMATiC_DMC_Mp[9][7] = 2.994327e-01f;
AutoMATiC_DMC_Mp[9][8] = 3.147143e-02f;
AutoMATiC_DMC_Mp[10][1] = 9.987274e-01f;
AutoMATiC_DMC_Mp[10][2] = 1.986524e+00f;
AutoMATiC_DMC_Mp[10][3] = 1.876030e-01f;
AutoMATiC_DMC_Mp[10][4] = 5.595337e-01f;
AutoMATiC_DMC_Mp[10][5] = 3.440136e-02f;
AutoMATiC_DMC_Mp[10][6] = 1.506941e-01f;
AutoMATiC_DMC_Mp[10][7] = 5.465313e-03f;
AutoMATiC_DMC_Mp[10][8] = 3.355960e-02f;
AutoMATiC_DMC_ftmp3 = darray(1,10,1,1);
AutoMATiC_DMC_ftmp4 = darray(1,10,1,1);
AutoMATiC_DMC_fconst = darray(1,10,1,10);
AutoMATiC_DMC_fconst[1][1] = 0.000000e+00f;
AutoMATiC_DMC_fconst[1][2] = 0.000000e+00f;
AutoMATiC_DMC_fconst[1][3] = -5.104583e+00f;
AutoMATiC_DMC_fconst[1][4] = -1.622249e+00f;
AutoMATiC_DMC_fconst[1][5] = -7.603490e+00f;
AutoMATiC_DMC_fconst[1][6] = -1.928652e+00f;
AutoMATiC_DMC_fconst[1][7] = -8.826808e+00f;
AutoMATiC_DMC_fconst[1][8] = -1.986524e+00f;
AutoMATiC_DMC_fconst[1][9] = -9.425674e+00f;
AutoMATiC_DMC_fconst[1][10] = -1.997455e+00f;
AutoMATiC_DMC_fconst[2][1] = 0.000000e+00f;
AutoMATiC_DMC_fconst[2][2] = 0.000000e+00f;
AutoMATiC_DMC_fconst[2][3] = -1.264241e+00f;
AutoMATiC_DMC_fconst[2][4] = -2.853981e+00f;
AutoMATiC_DMC_fconst[2][5] = -1.729329e+00f;
AutoMATiC_DMC_fconst[2][6] = -3.671660e+00f;
AutoMATiC_DMC_fconst[2][7] = -1.900426e+00f;
AutoMATiC_DMC_fconst[2][8] = -3.905929e+00f;
AutoMATiC_DMC_fconst[2][9] = -1.963369e+00f;
AutoMATiC_DMC_fconst[2][10] = -3.973048e+00f;
AutoMATiC_DMC_fconst[3][1] = 0.000000e+00f;
```

```
AutoMATiC_DMC_fconst[3][2] = 0.000000e+00f;
AutoMATiC_DMC_fconst[3][3] = 0.000000e+00f;
AutoMATiC_DMC_fconst[3][4] = 0.000000e+00f;
AutoMATiC_DMC_fconst[3][5] = -5.104583e+00f;
AutoMATiC_DMC_fconst[3][6] = -1.622249e+00f;
AutoMATiC_DMC_fconst[3][7] = -7.603490e+00f;
AutoMATiC_DMC_fconst[3][8] = -1.928652e+00f;
AutoMATiC_DMC_fconst[3][9] = -8.826808e+00f;
AutoMATiC_DMC_fconst[3][10] = -1.986524e+00f;
AutoMATiC_DMC_fconst[4][1] = 0.000000e+00f;
AutoMATiC_DMC_fconst[4][2] = 0.000000e+00f;
AutoMATiC_DMC_fconst[4][3] = 0.000000e+00f;
AutoMATiC_DMC_fconst[4][4] = 0.000000e+00f;
AutoMATiC_DMC_fconst[4][5] = -1.264241e+00f;
AutoMATiC_DMC_fconst[4][6] = -2.853981e+00f;
AutoMATiC_DMC_fconst[4][7] = -1.729329e+00f;
AutoMATiC_DMC_fconst[4][8] = -3.671660e+00f;
AutoMATiC_DMC_fconst[4][9] = -1.900426e+00f;
AutoMATiC_DMC_fconst[4][10] = -3.905929e+00f;
AutoMATiC_DMC_fconst[5][1] = 0.000000e+00f;
AutoMATiC_DMC_fconst[5][2] = 0.000000e+00f;
AutoMATiC_DMC_fconst[5][3] = 0.000000e+00f;
AutoMATiC_DMC_fconst[5][4] = 0.000000e+00f;
AutoMATiC_DMC_fconst[5][5] = 0.000000e+00f;
AutoMATiC_DMC_fconst[5][6] = 0.000000e+00f;
AutoMATiC_DMC_fconst[5][7] = -5.104583e+00f;
AutoMATiC_DMC_fconst[5][8] = -1.622249e+00f;
AutoMATiC_DMC_fconst[5][9] = -7.603490e+00f;
AutoMATiC_DMC_fconst[5][10] = -1.928652e+00f;
AutoMATiC_DMC_fconst[6][1] = 0.000000e+00f;
AutoMATiC_DMC_fconst[6][2] = 0.000000e+00f;
AutoMATiC_DMC_fconst[6][3] = 0.000000e+00f;
AutoMATiC_DMC_fconst[6][4] = 0.000000e+00f;
AutoMATiC_DMC_fconst[6][5] = 0.000000e+00f;
AutoMATiC_DMC_fconst[6][6] = 0.000000e+00f;
AutoMATiC_DMC_fconst[6][7] = -1.264241e+00f;
AutoMATiC_DMC_fconst[6][8] = -2.853981e+00f;
AutoMATiC_DMC_fconst[6][9] = -1.729329e+00f;
AutoMATiC_DMC_fconst[6][10] = -3.671660e+00f;
AutoMATiC_DMC_fconst[7][1] = 0.000000e+00f;
AutoMATiC_DMC_fconst[7][2] = 0.000000e+00f;
AutoMATiC_DMC_fconst[7][3] = 0.000000e+00f;
AutoMATiC_DMC_fconst[7][4] = 0.000000e+00f;
AutoMATiC_DMC_fconst[7][5] = 0.000000e+00f;
AutoMATiC_DMC_fconst[7][6] = 0.000000e+00f;
AutoMATiC_DMC_fconst[7][7] = 0.000000e+00f;
AutoMATiC_DMC_fconst[7][8] = 0.000000e+00f;
AutoMATiC_DMC_fconst[7][9] = -5.104583e+00f;
AutoMATiC_DMC_fconst[7][10] = -1.622249e+00f;
AutoMATiC_DMC_fconst[8][1] = 0.000000e+00f;
AutoMATiC_DMC_fconst[8][2] = 0.000000e+00f;
AutoMATiC_DMC_fconst[8][3] = 0.000000e+00f;
AutoMATiC_DMC_fconst[8][4] = 0.000000e+00f;
AutoMATiC_DMC_fconst[8][5] = 0.000000e+00f;
AutoMATiC_DMC_fconst[8][6] = 0.000000e+00f;
AutoMATiC_DMC_fconst[8][7] = 0.000000e+00f;
AutoMATiC_DMC_fconst[8][8] = 0.000000e+00f;
AutoMATiC_DMC_fconst[8][9] = -1.264241e+00f;
AutoMATiC_DMC_fconst[8][10] = -2.853981e+00f;
AutoMATiC_DMC_fconst[9][1] = 0.000000e+00f;
AutoMATiC_DMC_fconst[9][2] = 0.000000e+00f;
AutoMATiC_DMC_fconst[9][3] = 0.000000e+00f;
AutoMATiC_DMC_fconst[9][4] = 0.000000e+00f;
```

```
AutoMATiC_DMC_fconst[9][5] = 0.000000e+00f;
AutoMATiC_DMC_fconst[9][6] = 0.000000e+00f;
AutoMATiC_DMC_fconst[9][7] = 0.000000e+00f;
AutoMATiC_DMC_fconst[9][8] = 0.000000e+00f;
AutoMATiC_DMC_fconst[9][9] = 0.000000e+00f;
AutoMATiC_DMC_fconst[9][10] = 0.000000e+00f;
AutoMATiC_DMC_fconst[10][1] = 0.000000e+00f;
AutoMATiC_DMC_fconst[10][2] = 0.000000e+00f;
AutoMATiC_DMC_fconst[10][3] = 0.000000e+00f;
AutoMATiC_DMC_fconst[10][4] = 0.000000e+00f;
AutoMATiC_DMC_fconst[10][5] = 0.000000e+00f;
AutoMATiC_DMC_fconst[10][6] = 0.000000e+00f;
AutoMATiC_DMC_fconst[10][7] = 0.000000e+00f;
AutoMATiC_DMC_fconst[10][8] = 0.000000e+00f;
AutoMATiC_DMC_fconst[10][9] = 0.000000e+00f;
AutoMATiC_DMC_fconst[10][10] = 0.000000e+00f;
AutoMATiC_DMC_btmp1 = darray(1,40,1,1);
AutoMATiC_DMC_bvar = darray(1,40,1,2);
AutoMATiC_DMC_bvar[1][1] = 0.000000e+00f;
AutoMATiC_DMC_bvar[1][2] = 0.000000e+00f;
AutoMATiC_DMC_bvar[2][1] = 0.000000e+00f;
AutoMATiC_DMC_bvar[2][2] = 0.000000e+00f;
AutoMATiC_DMC_bvar[3][1] = 0.000000e+00f;
AutoMATiC_DMC_bvar[3][2] = 0.000000e+00f;
AutoMATiC_DMC_bvar[4][1] = 0.000000e+00f;
AutoMATiC_DMC_bvar[4][2] = 0.000000e+00f;
AutoMATiC_DMC_bvar[5][1] = 0.000000e+00f;
AutoMATiC_DMC_bvar[5][2] = 0.000000e+00f;
AutoMATiC_DMC_bvar[6][1] = 0.000000e+00f;
AutoMATiC_DMC_bvar[6][2] = 0.000000e+00f;
AutoMATiC_DMC_bvar[7][1] = 0.000000e+00f;
AutoMATiC_DMC_bvar[7][2] = 0.000000e+00f;
AutoMATiC_DMC_bvar[8][1] = 0.000000e+00f;
AutoMATiC_DMC_bvar[8][2] = 0.000000e+00f;
AutoMATiC_DMC_bvar[9][1] = 0.000000e+00f;
AutoMATiC_DMC_bvar[9][2] = 0.000000e+00f;
AutoMATiC_DMC_bvar[10][1] = 0.000000e+00f;
AutoMATiC_DMC_bvar[10][2] = 0.000000e+00f;
AutoMATiC_DMC_bvar[11][1] = 1.000000e+00f;
AutoMATiC_DMC_bvar[11][2] = 0.000000e+00f;
AutoMATiC_DMC_bvar[12][1] = 0.000000e+00f;
AutoMATiC_DMC_bvar[12][2] = 1.000000e+00f;
AutoMATiC_DMC_bvar[13][1] = 1.000000e+00f;
AutoMATiC_DMC_bvar[13][2] = 0.000000e+00f;
AutoMATiC_DMC_bvar[14][1] = 0.000000e+00f;
AutoMATiC_DMC_bvar[14][2] = 1.000000e+00f;
AutoMATiC_DMC_bvar[15][1] = 1.000000e+00f;
AutoMATiC_DMC_bvar[15][2] = 0.000000e+00f;
AutoMATiC_DMC_bvar[16][1] = 0.000000e+00f;
AutoMATiC_DMC_bvar[16][2] = 1.000000e+00f;
AutoMATiC_DMC_bvar[17][1] = 1.000000e+00f;
AutoMATiC_DMC_bvar[17][2] = 0.000000e+00f;
AutoMATiC_DMC_bvar[18][1] = 0.000000e+00f;
AutoMATiC_DMC_bvar[18][2] = 1.000000e+00f;
AutoMATiC_DMC_bvar[19][1] = 1.000000e+00f;
AutoMATiC_DMC_bvar[19][2] = 0.000000e+00f;
AutoMATiC_DMC_bvar[20][1] = 0.000000e+00f;
AutoMATiC_DMC_bvar[20][2] = 1.000000e+00f;
AutoMATiC_DMC_bvar[21][1] = -0.000000e+00f;
AutoMATiC_DMC_bvar[21][2] = -0.000000e+00f;
AutoMATiC_DMC_bvar[22][1] = -0.000000e+00f;
AutoMATiC_DMC_bvar[22][2] = -0.000000e+00f;
AutoMATiC_DMC_bvar[23][1] = -0.000000e+00f;
```

```
AutoMATiC_DMC_bvar[23][2] = -0.000000e+00f;
AutoMATiC_DMC_bvar[24][1] = -0.000000e+00f;
AutoMATiC_DMC_bvar[24][2] = -0.000000e+00f;
AutoMATiC_DMC_bvar[25][1] = -0.000000e+00f;
AutoMATiC_DMC_bvar[25][2] = -0.000000e+00f;
AutoMATiC_DMC_bvar[26][1] = -0.000000e+00f;
AutoMATiC_DMC_bvar[26][2] = -0.000000e+00f;
AutoMATiC_DMC_bvar[27][1] = -0.000000e+00f;
AutoMATiC_DMC_bvar[27][2] = -0.000000e+00f;
AutoMATiC_DMC_bvar[28][1] = -0.000000e+00f;
AutoMATiC_DMC_bvar[28][2] = -0.000000e+00f;
AutoMATiC_DMC_bvar[29][1] = -0.000000e+00f;
AutoMATiC_DMC_bvar[29][2] = -0.000000e+00f;
AutoMATiC_DMC_bvar[30][1] = -0.000000e+00f;
AutoMATiC_DMC_bvar[30][2] = -0.000000e+00f;
AutoMATiC_DMC_bvar[31][1] = -1.000000e+00f;
AutoMATiC_DMC_bvar[31][2] = -0.000000e+00f;
AutoMATiC_DMC_bvar[32][1] = -0.000000e+00f;
AutoMATiC_DMC_bvar[32][2] = -1.000000e+00f;
AutoMATiC_DMC_bvar[33][1] = -1.000000e+00f;
AutoMATiC_DMC_bvar[33][2] = -0.000000e+00f;
AutoMATiC_DMC_bvar[34][1] = -0.000000e+00f;
AutoMATiC_DMC_bvar[34][2] = -1.000000e+00f;
AutoMATiC_DMC_bvar[35][1] = -1.000000e+00f;
AutoMATiC_DMC_bvar[35][2] = -0.000000e+00f;
AutoMATiC_DMC_bvar[36][1] = -0.000000e+00f;
AutoMATiC_DMC_bvar[36][2] = -1.000000e+00f;
AutoMATiC_DMC_bvar[37][1] = -1.000000e+00f;
AutoMATiC_DMC_bvar[37][2] = -0.000000e+00f;
AutoMATiC_DMC_bvar[38][1] = -0.000000e+00f;
AutoMATiC_DMC_bvar[38][2] = -1.000000e+00f;
AutoMATiC_DMC_bvar[39][1] = -1.000000e+00f;
AutoMATiC_DMC_bvar[39][2] = -0.000000e+00f;
AutoMATiC_DMC_bvar[40][1] = -0.000000e+00f;
AutoMATiC_DMC_bvar[40][2] = -1.000000e+00f;
AutoMATiC_DMC_btmp2 = darray(1,40,1,1);
AutoMATiC_DMC_b = darray(1,40,1,1);
AutoMATiC_DMC_b[1][1] = 1.000000e-01f;
AutoMATiC_DMC_b[2][1] = 1.000000e-01f;
AutoMATiC_DMC_b[3][1] = 1.000000e-01f;
AutoMATiC_DMC_b[4][1] = 1.000000e-01f;
AutoMATiC_DMC_b[5][1] = 1.000000e-01f;
AutoMATiC_DMC_b[6][1] = 1.000000e-01f;
AutoMATiC_DMC_b[7][1] = 1.000000e-01f;
AutoMATiC_DMC_b[8][1] = 1.000000e-01f;
AutoMATiC_DMC_b[9][1] = 1.000000e-01f;
AutoMATiC_DMC_b[10][1] = 1.000000e-01f;
AutoMATiC_DMC_b[11][1] = 1.000000e+00f;
AutoMATiC_DMC_b[12][1] = 1.000000e+00f;
AutoMATiC_DMC_b[13][1] = 1.000000e+00f;
AutoMATiC_DMC_b[14][1] = 1.000000e+00f;
AutoMATiC_DMC_b[15][1] = 1.000000e+00f;
AutoMATiC_DMC_b[16][1] = 1.000000e+00f;
AutoMATiC_DMC_b[17][1] = 1.000000e+00f;
AutoMATiC_DMC_b[18][1] = 1.000000e+00f;
AutoMATiC_DMC_b[19][1] = 1.000000e+00f;
AutoMATiC_DMC_b[20][1] = 1.000000e+00f;
AutoMATiC_DMC_b[21][1] = 1.000000e-01f;
AutoMATiC_DMC_b[22][1] = 1.000000e-01f;
AutoMATiC_DMC_b[23][1] = 1.000000e-01f;
AutoMATiC_DMC_b[24][1] = 1.000000e-01f;
AutoMATiC_DMC_b[25][1] = 1.000000e-01f;
AutoMATiC_DMC_b[26][1] = 1.000000e-01f;
```

```
  AutoMATiC_DMC_b[27][1] = 1.000000e-01f;
  AutoMATiC_DMC_b[28][1] = 1.000000e-01f;
  AutoMATiC_DMC_b[29][1] = 1.000000e-01f;
  AutoMATiC_DMC_b[30][1] = 1.000000e-01f;
  AutoMATiC_DMC_b[31][1] = 1.000000e+00f;
  AutoMATiC_DMC_b[32][1] = 1.000000e+00f;
  AutoMATiC_DMC_b[33][1] = 1.000000e+00f;
  AutoMATiC_DMC_b[34][1] = 1.000000e+00f;
  AutoMATiC_DMC_b[35][1] = 1.000000e+00f;
  AutoMATiC_DMC_b[36][1] = 1.000000e+00f;
  AutoMATiC_DMC_b[37][1] = 1.000000e+00f;
  AutoMATiC_DMC_b[38][1] = 1.000000e+00f;
  AutoMATiC_DMC_b[39][1] = 1.000000e+00f;
  AutoMATiC_DMC_b[40][1] = 1.000000e+00f;
  dumax = darray(1,1,1,2);
  dumax[1][1] = 1.000000e-01f;
  dumax[1][2] = 1.000000e-01f;
  dumin = darray(1,1,1,2);
  dumin[1][1] = -1.000000e-01f;
  dumin[1][2] = -1.000000e-01f;
  umax = darray(1,1,1,2);
  umax[1][1] = 1.000000e+00f;
  umax[1][2] = 1.000000e+00f;
  umin = darray(1,1,1,2);
  umin[1][1] = -1.000000e+00f;
  umin[1][2] = -1.000000e+00f;
  return;
}
for(j=1;j<=8;++j) for(k=1;k<=1;++k) AutoMATiC_DMC_dUp[j][k] = 0;
for(j=1;j<=10;++j) for(k=1;k<=1;++k) AutoMATiC_DMC_Yzad[j][k] = 0;
for(j=1;j<=10;++j) for(k=1;k<=1;++k) AutoMATiC_DMC_Y[j][k] = 0;
for(j=1;j<=2;++j) for(k=1;k<=1;++k) AutoMATiC_DMC_tmpu[j][k] = 0;
AutoMATiC_DMC_itmp=1;
for(AutoMATiC_DMC_i=1;AutoMATiC_DMC_i<=4;++AutoMATiC_DMC_i){
  for(AutoMATiC_DMC_j=1;AutoMATiC_DMC_j<=2;++AutoMATiC_DMC_j){
    AutoMATiC_DMC_dUp[AutoMATiC_DMC_itmp][1]=ad->du[ad->k-AutoMATiC_DMC_i][
        AutoMATiC_DMC_j-1];
    AutoMATiC_DMC_itmp=AutoMATiC_DMC_itmp+1;
  }
}
AutoMATiC_DMC_itmp=1;
for(AutoMATiC_DMC_i=1;AutoMATiC_DMC_i<=5;++AutoMATiC_DMC_i){
  for(AutoMATiC_DMC_j=1;AutoMATiC_DMC_j<=2;++AutoMATiC_DMC_j){
    AutoMATiC_DMC_Yzad[AutoMATiC_DMC_itmp][1]=ad->z[AutoMATiC_DMC_j-1];
    AutoMATiC_DMC_itmp=AutoMATiC_DMC_itmp+1;
  }
}
AutoMATiC_DMC_itmp=1;
for(AutoMATiC_DMC_i=1;AutoMATiC_DMC_i<=5;++AutoMATiC_DMC_i){
  for(AutoMATiC_DMC_j=1;AutoMATiC_DMC_j<=2;++AutoMATiC_DMC_j){
    AutoMATiC_DMC_Y[AutoMATiC_DMC_itmp][1]=ad->y[ad->k][AutoMATiC_DMC_j-1];
    AutoMATiC_DMC_itmp=AutoMATiC_DMC_itmp+1;
  }
}
for(j=1;j<=2;++j) for(k=1;k<=1;++k) AutoMATiC_DMC_e[j][k] = 0;
for(AutoMATiC_DMC_i=1;AutoMATiC_DMC_i<=2;++AutoMATiC_DMC_i){
  AutoMATiC_DMC_e[AutoMATiC_DMC_i][1]=ad->z[AutoMATiC_DMC_i-1]-ad->y[ad->k][
      AutoMATiC_DMC_i-1];
}
productab(AutoMATiC_DMC_Ke,AutoMATiC_DMC_e,AutoMATiC_DMC_dutmp1,2,2,2,1);
productab(AutoMATiC_DMC_Ku,AutoMATiC_DMC_dUp,AutoMATiC_DMC_dutmp2,2,8,8,1);
sumaa(AutoMATiC_DMC_dutmp1,AutoMATiC_DMC_dutmp2,AutoMATiC_DMC_du,2,1,-1);
for(j=1;j<=2;++j) for(k=1;k<=1;++k) AutoMATiC_DMC_du[j][k] = 0;
```

```
  for(j=1;j<=2;++j) for(k=1;k<=1;++k) AutoMATiC_DMC_uk[j][k] = 0;
  for(AutoMATiC_DMC_i=1;AutoMATiC_DMC_i<=2;++AutoMATiC_DMC_i){
    AutoMATiC_DMC_uk[AutoMATiC_DMC_i][1]=ad->u[ad->k-1][AutoMATiC_DMC_i-1];
  }
  sumaa(AutoMATiC_DMC_Yzad,AutoMATiC_DMC_Y,AutoMATiC_DMC_ftmp1,10,1,-1);
  productab(AutoMATiC_DMC_Mp,AutoMATiC_DMC_dUp,AutoMATiC_DMC_ftmp2,10,8,8,1);
  sumaa(AutoMATiC_DMC_ftmp1,AutoMATiC_DMC_ftmp2,AutoMATiC_DMC_ftmp3,10,1,-1);
  productab(AutoMATiC_DMC_fconst,AutoMATiC_DMC_ftmp3,AutoMATiC_DMC_ftmp4
      ,10,10,10,1);
  for(AutoMATiC_DMC_i=1;AutoMATiC_DMC_i<=10;++AutoMATiC_DMC_i){
    AutoMATiC_DMC_ftmp[AutoMATiC_DMC_i]=AutoMATiC_DMC_ftmp4[AutoMATiC_DMC_i
        ][1];
  }
  productab(AutoMATiC_DMC_bvar,AutoMATiC_DMC_uk,AutoMATiC_DMC_btmp1,40,2,2,1);
  sumaa(AutoMATiC_DMC_b,AutoMATiC_DMC_btmp1,AutoMATiC_DMC_btmp2,40,1,1);
  for(AutoMATiC_DMC_i=1;AutoMATiC_DMC_i<=40;++AutoMATiC_DMC_i){
    AutoMATiC_DMC_btmp[AutoMATiC_DMC_i]=AutoMATiC_DMC_btmp2[AutoMATiC_DMC_i
        ][1];
  }
  for(AutoMATiC_DMC_i=1;AutoMATiC_DMC_i<=10;++AutoMATiC_DMC_i){
    AutoMATiC_DMC_qpx[AutoMATiC_DMC_i]=0;
  }
  memcpy (osqp_q, &(AutoMATiC_DMC_ftmp[1]), 10*sizeof(c_float));
  osqp_update_lin_cost(osqp_work, osqp_q);
  memcpy (osqp_ub, &(AutoMATiC_DMC_btmp[1]), 40*sizeof(c_float));
  osqp_update_bounds(osqp_work, osqp_lb, osqp_ub);
  osqp_solve(osqp_work);
  memcpy (&(AutoMATiC_DMC_qpx[1]), osqp_work->solution->x, 10*sizeof(c_float));

  for(AutoMATiC_DMC_i=1;AutoMATiC_DMC_i<=2;++AutoMATiC_DMC_i){
    AutoMATiC_DMC_du[AutoMATiC_DMC_i][1]=AutoMATiC_DMC_qpx[AutoMATiC_DMC_i];
  }
  for(AutoMATiC_DMC_n=1;AutoMATiC_DMC_n<=2;++AutoMATiC_DMC_n){
    if(AutoMATiC_DMC_du[AutoMATiC_DMC_n][1]>dumax[1][AutoMATiC_DMC_n]){
      AutoMATiC_DMC_du[AutoMATiC_DMC_n][1]=dumax[1][AutoMATiC_DMC_n];
    }
    if(AutoMATiC_DMC_du[AutoMATiC_DMC_n][1]<dumin[1][AutoMATiC_DMC_n]){
      AutoMATiC_DMC_du[AutoMATiC_DMC_n][1]=dumin[1][AutoMATiC_DMC_n];
    }
    AutoMATiC_DMC_tmpu[AutoMATiC_DMC_n][1]=ad->u[ad->k-1][AutoMATiC_DMC_n-1]+
        AutoMATiC_DMC_du[AutoMATiC_DMC_n][1];
    if(AutoMATiC_DMC_tmpu[AutoMATiC_DMC_n][1]>umax[1][AutoMATiC_DMC_n]){
      AutoMATiC_DMC_tmpu[AutoMATiC_DMC_n][1]=umax[1][AutoMATiC_DMC_n];
    }
    if(AutoMATiC_DMC_tmpu[AutoMATiC_DMC_n][1]<umin[1][AutoMATiC_DMC_n]){
      AutoMATiC_DMC_tmpu[AutoMATiC_DMC_n][1]=umin[1][AutoMATiC_DMC_n];
    }
    AutoMATiC_DMC_du[AutoMATiC_DMC_n][1]=AutoMATiC_DMC_tmpu[AutoMATiC_DMC_n
        ][1]-ad->u[ad->k-1][AutoMATiC_DMC_n-1];
  }
  for(j=1;j<=1;++j) for(k=1;k<=2;++k) control_value[j][k] = 0;
  for(AutoMATiC_DMC_n=1;AutoMATiC_DMC_n<=2;++AutoMATiC_DMC_n){
    control_value[1][AutoMATiC_DMC_n]=AutoMATiC_DMC_du[AutoMATiC_DMC_n][1];
  }

  set_current_control_increment(c,&(control_value[1][1])); // du is indexed
      starting with 1, therefore to maintain compatibility it is required to
      refer to first element of an actual array
}

void controller_setup(){
profiler_start(2);
    init_archive_data(&ad, 200, 200, 2, 2, 0, 0, 0.01);
```

```
    init_current_control(&cc,&ad);
    controllerDMCA(NULL,NULL);
    controllerDMCN(NULL,NULL);
profiler_end(2);
}

void idle(){
profiler_start(13);
    const int k = 0;
  static int i = 0;
  static char str[1000] = {0};

    sprintf(str, "x = [%f,%f,",ad.y[k][0], ad.y[k][1]);     write_string(str);
    sprintf(str,       "%f,%f,",ad.z[0], ad.z[1]);          write_string(str);
    sprintf(str,       "%f,%f,",ad.u[k-1][0],ad.u[k-1][1]); write_string(str);
    sprintf(str,       "%d,",algorithm_used);               write_string(str);
    write_string("];\n\r");
  if(++i > 1000) profiler_print();
profiler_end(13);
}

void loop(){
profiler_start(10);
    static int i = 0;
       if(i< 100){ ad.z[0] = -0.1f; ad.z[1] = +0.2f; }
  else if(i< 150){ ad.z[0] = -0.1f; ad.z[1] = -0.2f; }
  else if(i< 250){ ad.z[0] = +0.1f; ad.z[1] = -0.2f; }
  else           { ad.z[0] = +0.1f; ad.z[1] = +0.2f; }
  if(++i > 300){
    i = 0;
  }
    if((i%10==0) && ((random() % 100) < 10))
    algorithm_used = algorithm_used==0?1:0;


profiler_start(50);
  if(algorithm_used == 0) controllerDMCA(&ad,&cc);
  else                    controllerDMCN(&ad,&cc);
profiler_end(50);

  push_current_controls_to_archive_data(&cc,&ad);
profiler_end(10);
}

void timeout(){
    while(1);
}
```

F PID Algorithm: Generated Source of main_mpc.c

```
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\defines.h"
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\profiler.h"
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\mpctools.h"
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\simulated_signals.h"
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\matrix_cal.h"
#include "C:\Users\Admin\Documents\GitHub\AutoMATiC\Libs\C\allocation_nr.h"
#include "osqp.h"
#include "util.h"
#include "stm32f7xx_hal.h"
#include <string.h>
#include "main.h"

ArchiveData ad;
CurrentControl cc;

long get_time(){ return HAL_GetTick(); }

extern void timer_loop(void);
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim){
    if (htim->Instance == TIM2) {
        timer_loop();
    }
}
void measurements(){
profiler_start(4);
    new_output(&ad,__measureOutput());
profiler_end(4);
}
void controls(){
profiler_start(3);
    __setControlValue(last_control(&ad));
profiler_end(3);
}
void hardware_setup(){
profiler_start(1);
    low_lvl_main();
profiler_end(1);
}

void controller(ArchiveData * ad, CurrentControl * c){
  static float** control_value;
  static float** du;
  static float** e;
  static float** er;
  static long n;
  static float** tmpu;
  static long j;
  static long k;
  static float** r;
  static float** dumax;
  static float** dumin;
  static float** umax;
  static float** umin;
  if(ad == NULL){
    e = darray(1,2,1,3);
    tmpu = darray(1,2,1,1);
    er = darray(1,2,1,3);
    du = darray(1,2,1,1);
    control_value = darray(1,1,1,2);
    r = darray(1,3,1,2);
    r[1][1] = 0.000000e+00f;
```

```
    r[1][2] = 0.000000e+00f;
    r[2][1] = -9.500000e-02f;
    r[2][2] = -2.500000e-02f;
    r[3][1] = 1.050000e-01f;
    r[3][2] = 7.500000e-02f;
    dumax = darray(1,1,1,2);
    dumax[1][1] = 1.000000e-02f;
    dumax[1][2] = 1.000000e-02f;
    dumin = darray(1,1,1,2);
    dumin[1][1] = -1.000000e-02f;
    dumin[1][2] = -1.000000e-02f;
    umax = darray(1,1,1,2);
    umax[1][1] = 1.000000e+00f;
    umax[1][2] = 1.000000e+00f;
    umin = darray(1,1,1,2);
    umin[1][1] = -1.000000e+00f;
    umin[1][2] = -1.000000e+00f;
    return;
  }
  for(j=1;j<=2;++j) for(k=1;k<=3;++k) e[j][k] = 0;
  for(j=1;j<=2;++j) for(k=1;k<=1;++k) tmpu[j][k] = 0;
  for(j=1;j<=2;++j) for(k=1;k<=3;++k) er[j][k] = 0;
  for(j=1;j<=2;++j) for(k=1;k<=1;++k) du[j][k] = 0;
  for(j=1;j<=1;++j) for(k=1;k<=2;++k) control_value[j][k] = 0;
  for(n=1;n<=2;++n){
    e[n][1]=ad->z[n-1]-ad->y[ad->k-0][n-1];
    e[n][2]=ad->z[n-1]-ad->y[ad->k-1][n-1];
    e[n][3]=ad->z[n-1]-ad->y[ad->k-2][n-1];
    er[n][1]=e[n][1]*r[1][n];
    er[n][2]=e[n][2]*r[2][n];
    er[n][3]=e[n][3]*r[3][n];
    du[n][1]=du[n][1]+er[n][1];
    du[n][1]=du[n][1]+er[n][2];
    du[n][1]=du[n][1]+er[n][3];
    if(du[n][1]>dumax[1][n]){
      du[n][1]=dumax[1][n];
    }
    if(du[n][1]<dumin[1][n]){
      du[n][1]=dumin[1][n];
    }
    tmpu[n][1]=ad->u[ad->k-1][n-1]+du[n][1];
    if(tmpu[n][1]>umax[1][n]){
      tmpu[n][1]=umax[1][n];
    }
    if(tmpu[n][1]<umin[1][n]){
      tmpu[n][1]=umin[1][n];
    }
    du[n][1]=tmpu[n][1]-ad->u[ad->k-1][n-1];
  }
  for(n=1;n<=2;++n){
    control_value[1][n]=du[n][1];
  }

  set_current_control_increment(c,&(control_value[1][1])); // du is indexed
      starting with 1, therefore to maintain compatibility it is required to
      refer to first element of an actual array
}

void controller_setup(){
profiler_start(2);
    init_archive_data(&ad, 200, 200, 2, 2, 0, 0, 0.01);
    init_current_control(&cc,&ad);
    controller(NULL,NULL);
```

```
profiler_end(2);
}

void idle(){
profiler_start(13);
    const int k = 0;
  static int i = 0;
  static char str[1000] = {0};

    sprintf(str, "x = [%f,%f,",ad.y[k][0], ad.y[k][1]);     write_string(str);
    sprintf(str,       "%f,%f,",ad.z[0], ad.z[1]);          write_string(str);
    sprintf(str,       "%f,%f,",ad.u[k-1][0],ad.u[k-1][1]); write_string(str);
    write_string("];\n\r");
  if(++i > 1000) profiler_print();
profiler_end(13);
}

void loop(){
profiler_start(10);
    static int i = 0;
       if(i< 100){ ad.z[0] = -0.1f; ad.z[1] = +0.2f; }
  else if(i< 150){ ad.z[0] = -0.1f; ad.z[1] = -0.2f; }
  else if(i< 250){ ad.z[0] = +0.1f; ad.z[1] = -0.2f; }
  else           { ad.z[0] = +0.1f; ad.z[1] = +0.2f; }
  if(++i > 300) i = 0;

profiler_start(50);
  controller(&ad,&cc);
profiler_end(50);

  push_current_controls_to_archive_data(&cc,&ad);
profiler_end(10);
}

void timeout(){
    while(1);
}
```