DOI:10.28974/idojaras.2021.2.3

IDŐJÁRÁS

Quarterly Journal of the Hungarian Meteorological Service
Vol. 125, No. 2, April – June, 2021, pp. 211–227

Predictive control of a solar thermal system via on-line communication with a meteorological database server

János Tóth and **István Farkas***

Institute of Technology
Hungarian University of Agriculture and Life Sciences,
Páter K. u. 1, 2100 Gödöllő, Hungary

**Corresponding author E-mail: farkas.istvan@uni-mate.hu*

(Manuscript received in final form July 8, 2020)

Abstract— In this paper, the mathematical models of a solar thermal system which governs the solar thermal collector, the heat storage system, and the pump are presented. It has been shown that it is possible to connect a Simulink-based model to a meteorological database server using standard communication protocols by a C language-based component in order to import real-life weather information into the simulation. The setup of the model predictive control of this solar thermal system and the results of the simulation are also presented. This computationally heavy control method is possible to use on today's personal computers, and it can be expanded.

Key-words: SimSolar, Simulink, TCP/IP, JSON

1. Introduction

The need for renewable energy solutions has grown over the years, and the growth has continuously been rising. The technologies that utilize solar energy are solar thermal and photovoltaic systems. The process of design and optimization of these systems involves a huge number of calculations based on various types of mathematical models describing the system.

The block-oriented modeling technique is a technique where a complex mathematical model can be abstracted into a block-relation model. The “block” can be interpreted as a function with inputs and outputs, and it can be visualized as a rectangle. In this way, the mathematical model transforms into a “wiring diagram”, where the underlying principles of the model can be seen easier, and

the linking of the blocks represent the relationships among the parts of the physical model (*Parrino et al.*, 2005).

The block-oriented modeling technique can be combined with the model-based design (*Qu et al.*, 2010). This design method focuses on the mathematical models of a system and uses the data of its simulations to reduce the cost of the prototyping phase of the product to optimize the already existing systems.

The hardware-in-the-loop (HIL) simulation is the extension of the block-oriented modeling. This method can connect the real-time world to the simulation by creating a block in the simulation that can read measurement data using dedicated measurement hardware (*Tóth et al.*, 2019).

Another way to connect real-world data to the simulation is by using a database, especially a meteorological source database. These databases can be accessed directly (*Tóth* and *Farkas*, 2018), but there is a security risk for public databases, so this method is common for local networks. A more acceptable way to access data from a public online database is to go through a TCP/IP (Transmission Control Protocol/Internet Protocol) API (Application Programming Interface). The API provides a way to the developer to interface with the application in a controlled manner. The most common format of these server-client messages (APIs) are JOSN (JavaScript Object Notation) and XML (Extensible Markup Language) (*Wang*, 2011). The advantage of this method is that the database is not directly accessible, and the messages can be viewed in a regular web browser in a human-readable format. This approach can greatly improve the productivity of the simulation, because it skips the step of the manual exchange of the data, and the request of a different data set is just a change of a few parameters of the model.

The operation of a dynamical system usually requires a kind of control unit; in this case, the unit is a predictive control unit, which is a sophisticated solution in time-delayed solar systems. The predictive controlling algorithms provide a way to handle the systems with a slow response time. Such algorithms usually are neural network-based (*Shuzhi et al.*, 2008) or mathematical model-based (*Qin* and *Badgwell*, 2003). The model predictive control (MPC) can be used if the mathematical model of the system and its inverse are known. This approach is a natural choice for solar thermal systems, because these types of systems are well known in the mathematical modeling perspective, and they have slow response time, especially if a solar thermal farm is considered.

The aim of this paper is to present the block-oriented mathematical model of a solar thermal system and to show its model predictive control based on the data stored server and accessed through a TCP/IP API. The simulations were done using the Simulink framework. The mathematical models of the operational units of the solar thermal system are parts of a Simulink library, called SimSolar (*Tóth* and *Farkas*, 2017).

2. Materials and methods

In this session the applied mathematical models for the main components of the solar thermal system are presented. The components are composed as blocks in the Simulink simulation.

2.1. Model of the solar collector

The solar thermal collector converts the radiant energy from the Sun into heat. The mathematical model of the solar collector, the Hottel-Whillier model (*Farkas*, 1999), is interpreted as shown in *Fig. 1*.

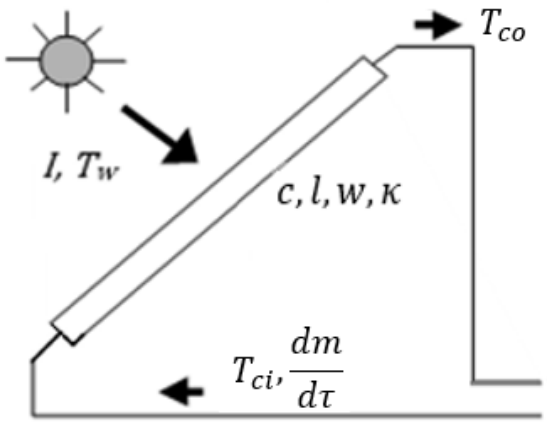

Fig. 1. Schematic diagram of the solar collector.

The governing equation of the Hottel-Whillier model can be written as

$$T_{co}(\tau) = T_w(\tau) + \frac{I(\tau)}{\kappa_{aw}} + \left(T_{ci}(\tau) - T_w(\tau) - \frac{I(\tau)}{k_{aw}}\right) \exp\left(-\frac{\kappa_{mw} w l}{c_c \frac{dm_c}{d\tau}}\right), \quad (1)$$

where

τ - time (s),
T_{co} - outlet temperature of the solar collector (°C),
T_w - ambient temperature (°C),
I - solar radiation (W m^{-2}),
κ_{aw} - heat transfer coefficient (absorber – environment) (W m^{-2} K^{-1}),
T_{ci} - inlet temperature of the solar collector (°C),
κ_{mw} - heat transfer coefficient (working fluid – environment) (W m^{-2} K^{-1}),
w - width of the solar collector (m),
l - length of the solar collector (m),

c_c - specific heat of the working fluid (J kg^{-1} K^{-1}),
$dm_c/d\tau$- mass flow-rate of the working fluid (kg s^{-1}).

The implementation and the parameters of the applied model can be seen in *Figs. 2* and *3*.

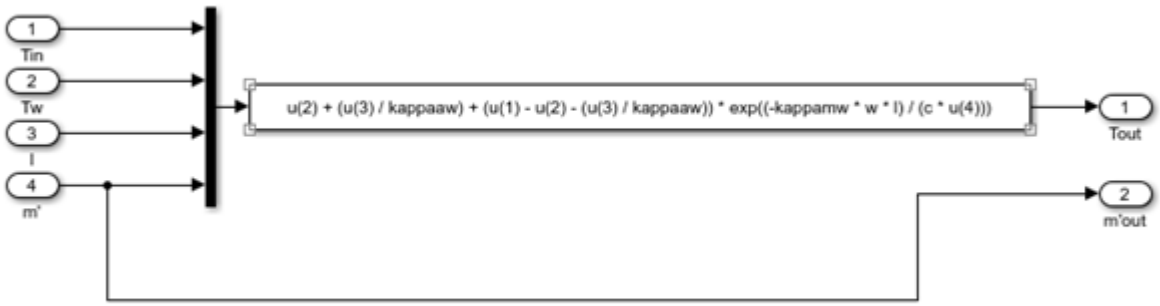

Fig. 2. Simulink implementation of the solar collector model.

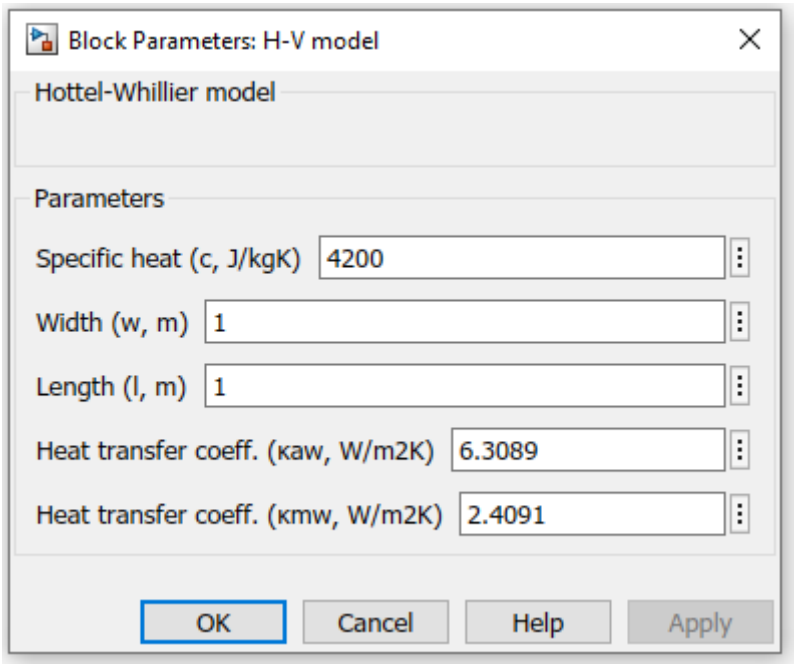

Fig. 3. Parameter window of the solar collector model.

In order to use the block in the simulation, the following parameters have to be set in the parameter window:

- specific heat of the containing fluid (J kg^{-1} K^{-1}),

- width of the solar collector (m),
- length of the solar collector (m),
- heat transfer coefficient (absorber - working fluid) (W m^{-2} K^{-1}),
- heat transfer coefficient (environment - working fluid) (W m^{-2} K^{-1}).

2.2. Model of the heat storage

In most cases, the period of the consumption of the collected solar energy is not necessarily the same as the time of the collection, so this energy must be stored for later use. The storage of the heat is taken care of by this unit. The interpretation of the heat storage model (*Farkas*, 1999) can be seen in *Fig. 4*. The heat storage could provide hot water for additional technological purposes.

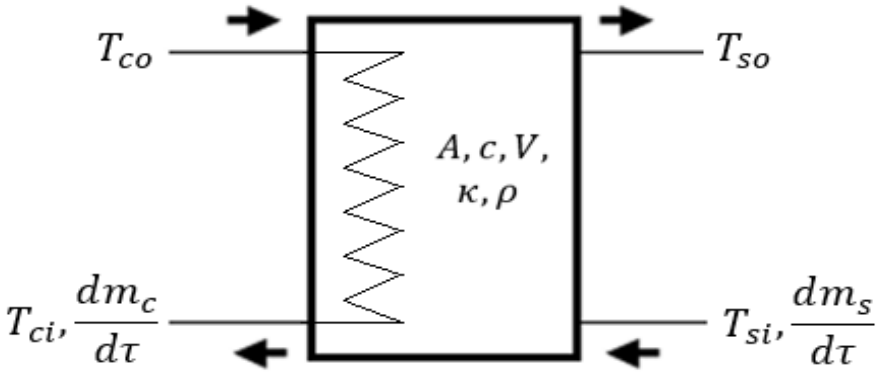

Fig. 4. Schematic diagram of the heat storage.

The related governing equations are as follows:

$$T_{ci}(\tau) = \big(T_{co}(\tau) - T_{so}(\tau)\big)\exp\left(-\frac{\kappa_s A_s}{c_s \frac{dm_c}{d\tau}}\right) + T_{so}(\tau), \tag{2a}$$

$$c_s V_s \rho_s \frac{dT_{so}}{d\tau} = c_s \frac{dm_s}{d\tau}\big(T_{si}(\tau) - T_{so}(\tau)\big) + c_s \frac{dm_c}{d\tau}\big(T_{co}(\tau) - T_{ci}(\tau)\big), \tag{2b}$$

where

τ - time (s),
T_{ci} - inlet temperature of the solar collector (°C),
T_{co} - outlet temperature of the solar collector (°C),
T_{so} - outlet temperature of the storage (°C),
κ_s - heat transfer coefficient in the heat storage (W m^{-2} K^{-1}),

A_s - surface of the heat storage (m^2),
c_s - specific heat of the working fluid in the heat storage (J kg^{-1} K^{-1}),
m_s - mass of the working fluid in the heat storage (kg),
V_s - volume of the heat storage (m^3),
ρ_s - density of the working fluid in the heat storage (kg m^{-3}),
T_{si} - inlet temperature of the heat storage (°C),
c_c - specific heat of the working fluid in the solar collector (J kg^{-1} K^{-1}),
$dm_c/d\tau$ - mass flow-rate of the working fluid in the solar collector (kg s^{-1}).

The implementation and the parameters of the applied model are shown in *Figs. 5* and *6.*

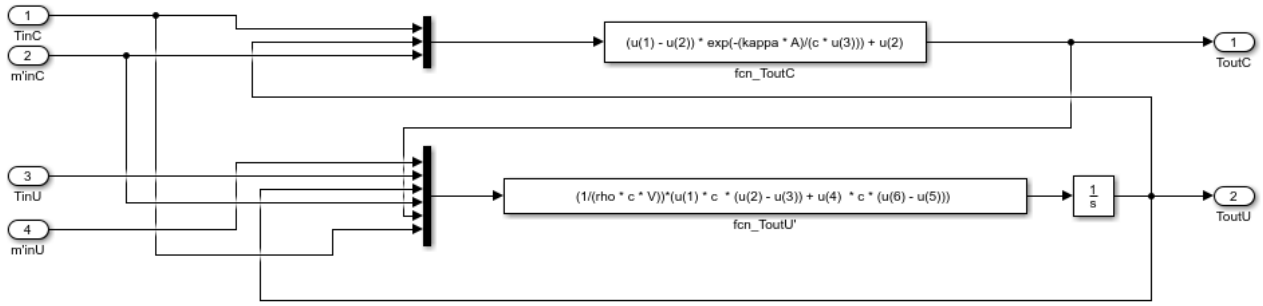

Fig. 5. Simulink implementation of the heat storage model.

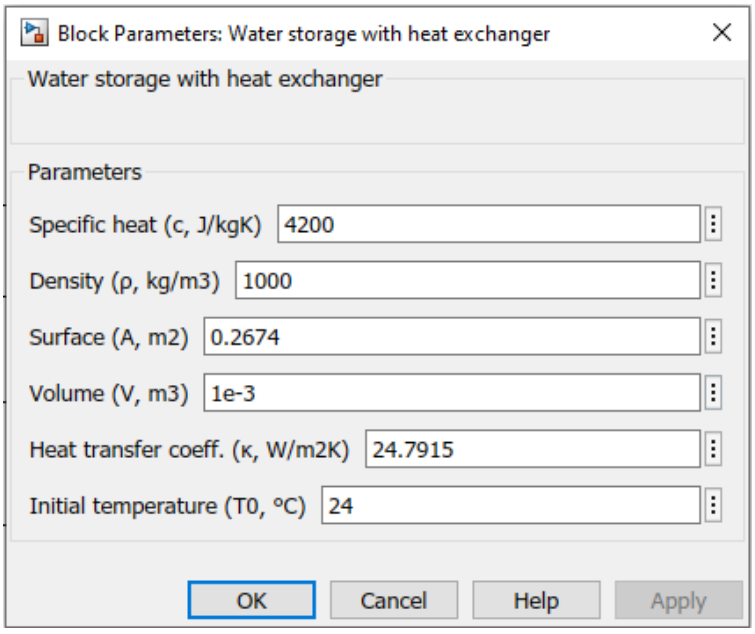

Fig. 6. Parameter window of the heat storage model.

In order to use the block in the simulation, the following parameters have to be set in the parameter window of the block:

- specific heat of the containing fluid (J kg^{-1} K^{-1}),
- density of the containing fluid (kg m^{-3}),
- surface of the heat storage (m^2),
- volume of the heat storage (m^3),
- heat transfer coefficient (storage, environment) (W m^{-2} K^{-1}),
- initial temperature (°C).

2.3. Model of the pump

The pump circulates the working fluid in the system, which allows the flow of the liquid through the collector circuit to be controlled. The current model of the pump is linear. It describes a linear characteristic, i.e., the output mass flow rate is the product of the maximum mass flow rate and the control signal:

$$\frac{dm}{d\tau} = \frac{dm}{d\tau}_{max} u(\tau). \tag{3}$$

The implementation and the parameters of the applied model are presented in *Figs. 7* and *8*.

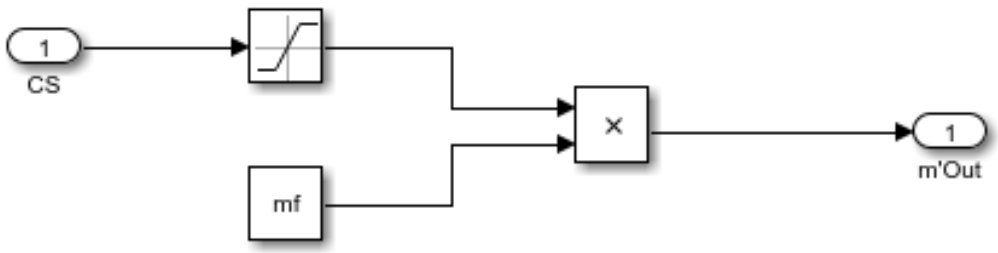

Fig. 7. Simulink implementation of the pump model.

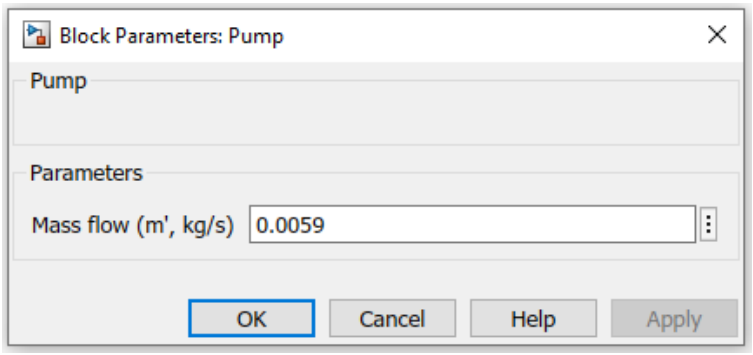

Fig. 8. Parameter window of the pump model.

In order to use the block in the simulation, the maximal mass flow-rate (kg s^{-1}) has to be set in the parameter window of the block.

2.4. Weather API

To access data from a server to run the simulation, a custom Simulink component had to be made. The Weather API block needs a server to communicate with. This server contains a database and a JSON based API. The server-client messages using TCP/IP provides the necessary information about the requested data. The server responds with a JSON formatted message containing the required data or an error message if an error occurred.

The API was designed in a way, that it can also be accessed using a web browser. For example, a request can be sent by typing into the address bar:

```
http://127.0.0.1/api/api.php?startDate=2018-04-
17&startTime=06:00:00&endDate=2018-04-
17&endTime=20:30:00&sensors=TW,I
```

This works only if a local server with the necessary API is up and running.

The response of this request contains the data from the sensors “TW” and “I” from the measurement interval from 06:00, April 17, 2018 to 20:30, April 17, 2018 in JSON format or an error message describing the error.

The Simulink implementation of this block contains a Level-2 MATLAB S-Function (*mathworks.com*, 2020b) written in the C programming language to handle the low-level file input/output duties and to implement the TCP/IP client. The parsing of the JSON formatted response is done using DaveGamble’s cJSON library (*github.com*, 2020).

In order to reduce the workload of the server, the block only requests for data, if updated data available. After the data arrives, the block saves it in a file, and uses the request to name this file uniquely. This mechanism allows the block to keep track of the already requested data.

The working principle of the Weather API block can be seen in *Fig. 9*.

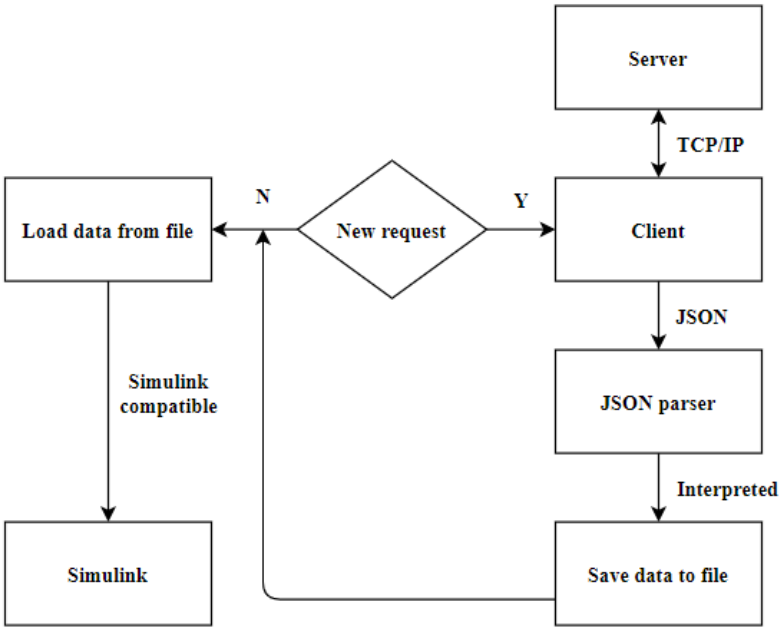

Fig. 9. Working principle of the Weather API block.

The block contains an option for linear interpolation; this is useful when the measurement frequency of the data is low compared to the accuracy of the simulation. The parameters of the related Simulink block are shown in *Fig. 10*.

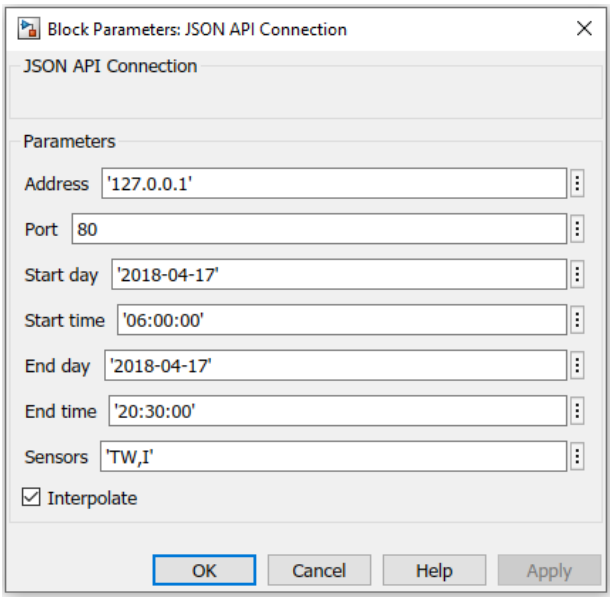

Fig. 10. Parameter window of the Weather API block.

In order to use the block in a simulation, the following parameters have to be set in the parameter window of the block:

- address of the server (IP address or URI),
- port of the server (integer value),
- the start day of the simulation (in yyyy-MM-dd format),
- the start time of the interval (in HH-mm-ss format),
- end day of the interval (in yyyy-MM-dd format),
- the end time of the interval (in HH-mm-ss format),
- name of the data fields (comma separated list),
- request for interpolation (boolean value).

2.5. Model predictive controller

In a solar thermal system, the controllers manipulate the pump operation ensuring the expected temperature of the outlet medium. In this paper, the MPC algorithm was chosen to study the behavior of the control process. This control would be an ideal choice for solar thermal systems, even for solar thermal farms, due to its ability to handle slow response times.

The controller is part of the Model Predictive Control Toolbox of Simulink (*mathworks.com*, 2020a). *Fig. 11.* shows the working principle of the model predictive controller.

The setup and tuning of the controller can be done using its graphical interface. The main parameters of the controller are:

- *Prediction horizon:* the quantity of future control intervals the controller must assess by prediction while optimizing its values at the control interval (*mathworks.com*, 2020c).
- *Control horizon:* the quantity of adjustments steps to be optimized at the control interval (*mathworks.com*, 2020c).
- *Constraints:* the minimum and maximum values that the output of the controller must obey.

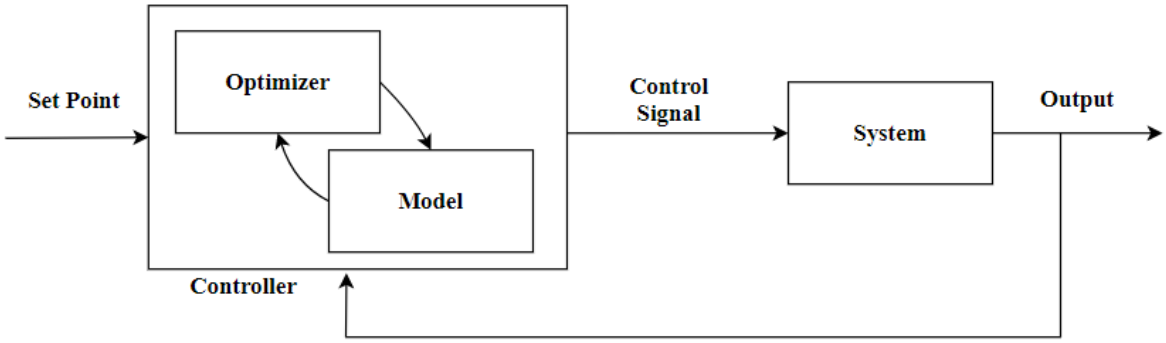

Fig. 11. Working principle of the model predictive controller.

3. Results

The examined system and the results of the simulation are presented in this section.

3.1. Examined system

The examined system consists of the models of the solar thermal collector, the heat storage, the two pumps (in the primary (collector-side) loop and in the secondary (consumer-side) loop), and the controller. The input data of the simulation comes from the database through TCP/IP protocol using the JSON format. The technical parameters were identified based on previous measurements using a physically built solar thermal system (*Tóth et al.*, 2019).

Technical parameters of the solar collector:

- width of the solar collector: 1 m,
- height of the solar collector: 1 m,
- heat transfer coefficient (absorber – working fluid): 6.31 W m^{-2} K^{-1},
- heat transfer coefficient (environment – working fluid): 4.21 W m^{-2} K^{-1},
- constant mass flow rate of the collector-side Pump1: 0.0059 kg s^{-1},
- working fluid: water.

Technical parameters of the water storage with heat exchanger:

- volume of the storage: 10^{-3} m^3,
- area of the storage: 0.2674 m^2,

- heat transfer coefficient in the heat storage: 24.79 W m^{-2} K^{-1},
- initial temperature: 22.9 °C,
- maximum mass flow rate of the consumer-side Pump2: 0.0059 kg s^{-1},
- working fluid: water.

Parameters of the controller:

- set point temperature (SP): 50 °C,
- prediction horizon: 3600 s,
- control horizon: 1800 s,
- constraint interval of control signal (CS): 0-1.

The block-oriented implementation of the solar thermal system can be seen in *Fig. 12*. The figure shows that the control signal of the primary loop (CS1) is set to a constant value of 1, as it is indicated in the rectangular block, which means that the mass flow rate of this loop is 0.0059 kg s^{-1}.

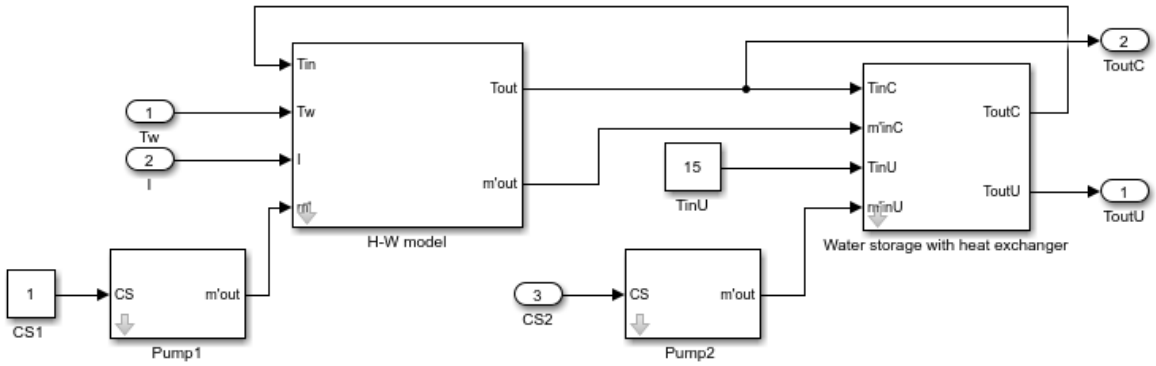

Fig. 12. Simulink implementation for the model of the solar thermal system.

The block-oriented implementation of the controlled system can be seen in *Fig 13*. During this experiment due to technical reasons, the solar thermal system was controlled at the consumer-side (CS2).

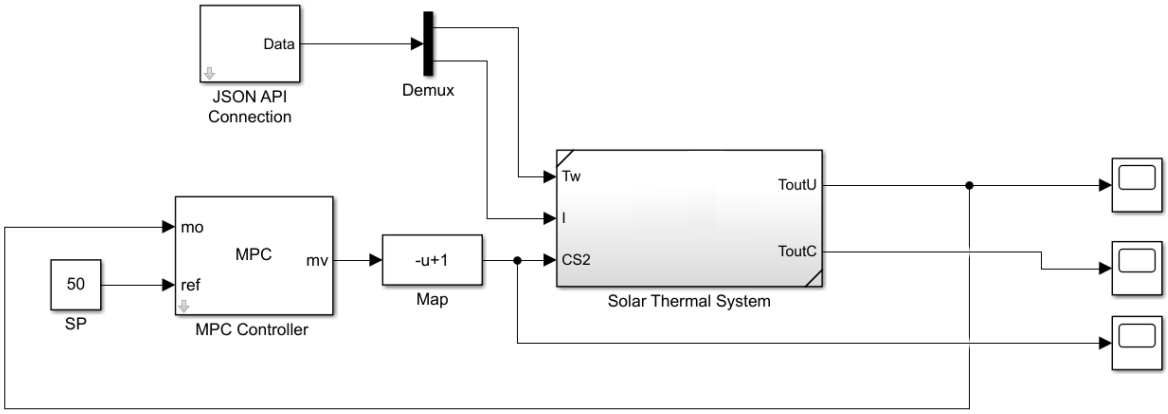

Fig. 13. Simulink implementation for the controlled solar thermal system.

The input of the simulation, the environmental temperature, and the solar irradiance comes from a verified mathematical model. The main purpose of this simulation was to study the model predictive controller and the Weather API block/server, which was easily achievable by the generation of the input data. If the simulation works with this data set, obviously it will equally work with a different data set that comes from measurements. The inputs of the simulation can be seen in *Fig. 14* and *15*.

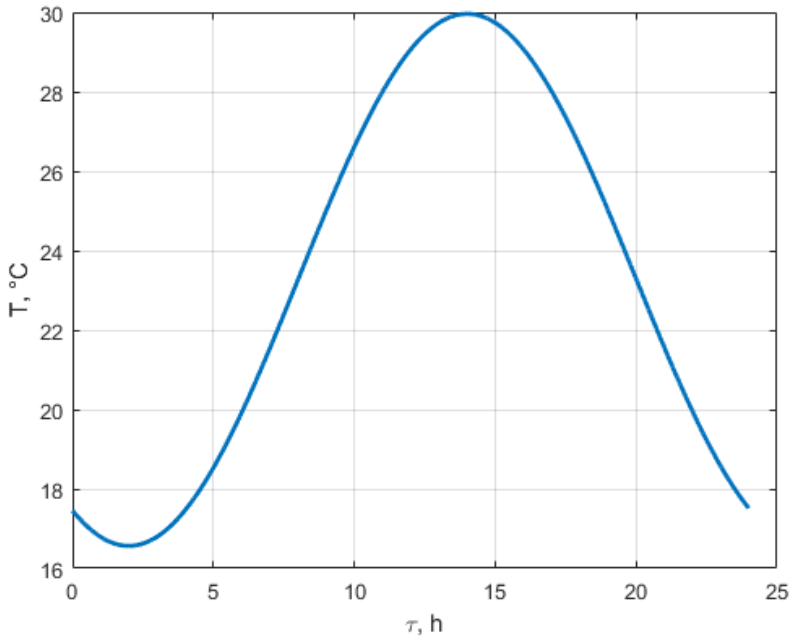

Fig. 14. Input data: ambient temperature.

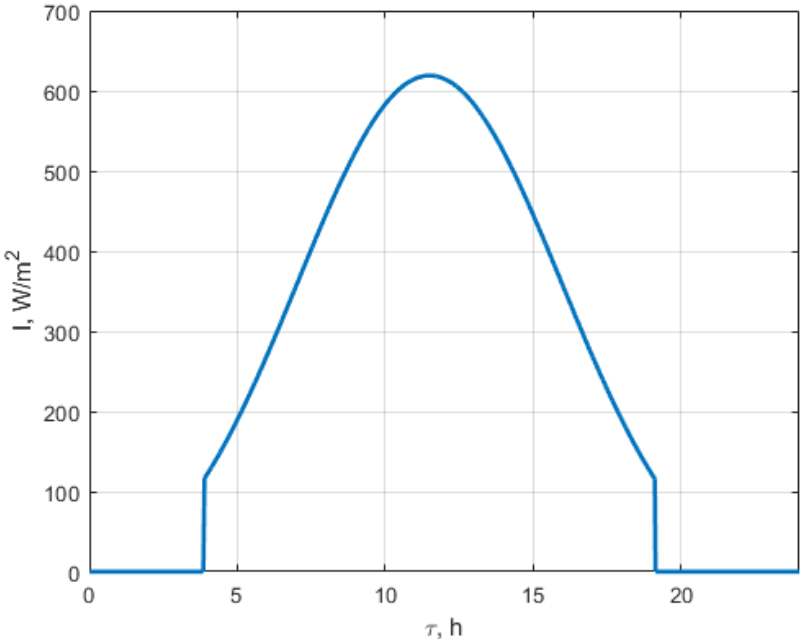

Fig. 15. Input data: solar irradiance.

3.2. Results of the simulation

The outlet temperature of the consumer-side (ToutU) of the heat storage and the outlet temperature of the solar collector (ToutC) can be seen in *Fig. 16*. The figure shows that the controlled heat storage temperature kept the 50 °C set point value with less than 0.2 °C dynamic error in the controlled period. Additionally, it was observed that in the non-controlled period, the solar collector outlet and the heat storage temperatures had almost the same value because of the relatively small volume of the storage tank.

Fig. 17 shows the flow rate of the consumer-side pump which is proportional to the control signal (CS2). It can be seen, that the controller solved a dynamic problem with online optimization, and the control signal reflects the characteristics of both the environmental temperature and the solar irradiance, as it was expected.

The MPC was chosen for its ability to handle dynamic systems with slow response time, and the results show that this control algorithm is capable of solving the controlling task of a solar thermal system. The study confirmed that this control approach works with the current framework, namely the SimSolar block-library and the Model Predictive Control Toolbox of Simulink, and this fact opens up a way for further expansion of this idea, e.g., the HIL implementation.

The simulation verified that the Weather API block works. This result is as important as the working of the MPC, because it proved that a generic C language-based TCP/IP client could be implemented within the Simulink framework. This

means that it can be the base of a bridge between the modern online data storage methods and the Simulink framework, because every major online system has some form of an API, either TCP/IP based or C language library. Despite the fact that the C language is mainly used as a system programming language, it is a generic programming language with nearly 50 years' worth of programmer knowledge along with the proven-to-work function libraries.

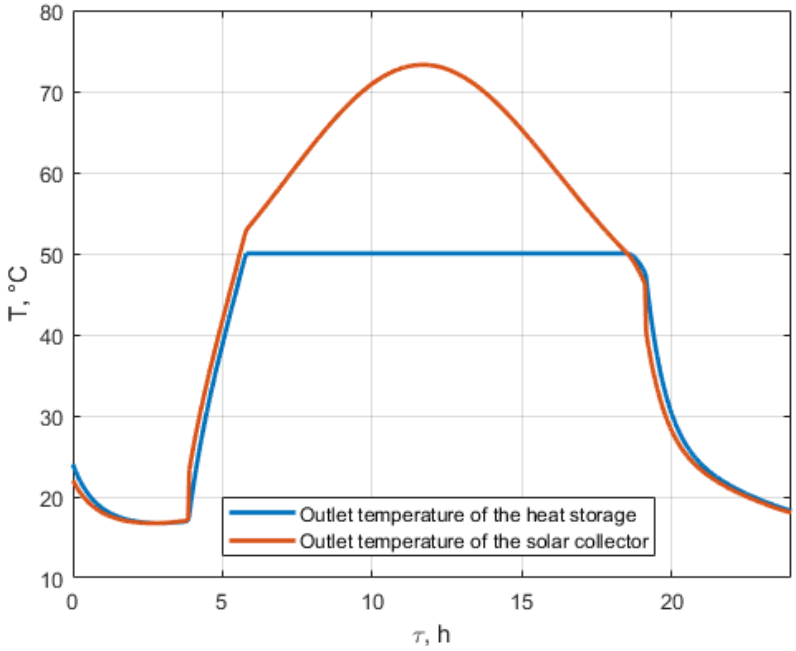

Fig. 16. Outlet temperatures of the system.

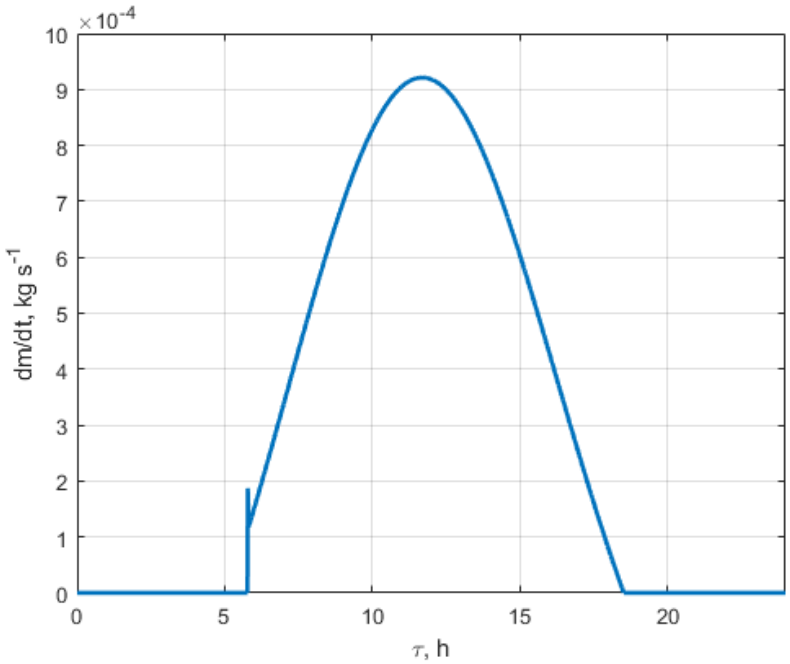

Fig. 17. Mass flow-rate of the controlled consumer-side pump.

4. Summary

A previously verified mathematical model capable of describing a solar thermal system was created in a block-oriented way. This model of the system contains the mathematical models of the solar collector, the heat storage, and the pump. These components are part of the block-oriented library for the solar thermal simulations, called SimSolar.

In order to use real-life measurement data in the simulation, a custom Simulink block was created in the C programming language that can handle the TCP/IP based communication with the server, and it can interpret the JSON formatted response of the server. This block has proven that an external source of data, especially the meteorological ones, can be integrated into the simulation workflow of the Simulink framework, and the usage of an external programming language can be beneficial to solve problems that Simulink is not designed to solve by itself.

The controlling of the model of the solar thermal system was performed by a model predictive controller to study and refine its behavior. Usually, the solar thermal systems have a slow response time, so the predictive control algorithm would be more suitable for them. The development in the IT field has paved the way to exploit such a computationally heavy control method by using an average personal computer.

The studied system had a set point of 50 °C, and the controller kept the outlet temperature within the set limits.

This research has shown that the usage of an online database server containing input data for the simulation is a quality-of-life addition; it eases the process of the import of the real-life data into the simulation. It has also shown that the model predictive control of a solar thermal system can be carried out using computers available at home, and it can be expanded with an HIL simulation in the future.

References

*Farkas, I.,*1999: Computer simulation, Textbook, 45–68.

Parrino, M.., Mannoni, A., Bonisoli, E., and *Sorli, M.,* 2005: Block-oriented models for transient HVAC simulations, *J.Passenger Car: Mech. Syst. J. 114*, No. 6. https://doi.org/10.4271/2005-01-2002

Tóth, J. and *Farkas, I.,* 2017: A Simulink library for solar energy applications, *R&D in Mech. Engin. Lett. 16*.

Tóth, J. and *Farkas, I.,* 2018. Implementing database support for Simulink applied for solar thermal systems. Book of Abstracts, 24th Workshop on Energy and Environment.

Tóth, J., Erdélyi, V., Farkas, I., and *Jánosi, L.,* 2019. Hardware-in-the-loop control of a small-scale thermal system. Book of abstracts, BioPhys Spring 2019.

Shuzhi, S.G., Chenguang, Y.; and *Tong, H.L.,* 2008. Adaptive predictive control using neural network for a class of pure-feedback systems in discrete time. *IEEE Trans. Neural Netw. 19*, 1599–1614. https://doi.org/10.1109/TNN.2008.2000446

Qin, S.J. and *Badgwell, T.A.*, 2003: A survey of industrial model predictive control technology. *Contr. Engin. Practicle 11*. 733–764. https://doi.org/10.1016/S0967-0661(02)00186-7

Qu, M., Yin, H., and *Archer, D.H.,* 2010: A solar thermal cooling and heating system for a building: Experimental and model based performance analysis and design, *Solar Energy 84*. 166–182. https://doi.org/10.1016/j.solener.2009.10.010

Wang G., 2011: Improving data transmission in web applications via the translation between XML and JSON. Proceedings - 2011 3rd International Conference on Communications and Mobile Computing, CMC 2011. https://doi.org/10.1109/CMC.2011.25

https://www.github.com/DaveGamble/cJSON

https://www.mathworks.com/products/mpc.html

https://www.mathworks.com/help/simulink/sfg/writing-level-2-matlab-s-functions.html

https://www.mathworks.com/help/mpc/ug/choosing-sample-time-and-horizons.html