A. Documentation

The source code for the thesis: `https://github.com/didllicek/bil_sys_id`
Programme was developed in Matlab version 2015 under Windows 2010.

The main folder of the project **bil_sys_id** contains 3 scripts that run experiments commented in the thesis for studied systems:

- *experiments_peptide_elongation.m*
- *experiments_chemotherapy.m*
- *experiments_immune_response.m*
- plus 2 other scripts *experiments_tests.m* and *experiments_enzyme_reaction.m* that are not fully discussed in the thesis

The main folder of the project **bil_sys_id** contains 2 folders. Let's now describe their structure.

The first folder is **studied_systems** that contains folders with the specification of the systems studied in the experiments:

- **immune_response** which contains folders with all experiments for the chosen system

  1. **exp(.*)** – where each identification experiment contains
     (a) **system_specification** – contains specification of the system and thus contains files (one might)
        i. file with state matrix *Ac.txt*
        ii. file with input matrix *Bc.txt*
        iii. file wit output matrix *C.txt*
        iv. file with transmission matrix *D.txt*
        v. files with bilinear matrices *Nc1.txt* ... *Ncr.txt*, where r is number of inputs
        vi. file with initial condition *init_condition.txt*
        One might wonder why all the system specifications are in every experiment, but it is possible to also do the experiments with measurement equation (matrices C,D) and with initial condition x_0 and this way it seemed natural to the author of the thesis.
     (b) **id_meth_specification** – contains specification of parameters for identification method
        i. *id_meth_params.txt* – specifying Δt l_0, l, s, p, $precision$, α, β (the meaning of the parameters is explained in the thesis)
        ii. *W.txt* – specifying applied vectors of pulses

     This folder also contains script *immune_response_symbolic.m* for symbolic computations of observabillity.

2. other folders for experiments **exp(.*)** ... with chosen system

- **model_chemotherapy** – folder with analogical structure as in the previous case. This folder also contains folder **controllability_check** with script for symbolic computations of observability and controllability according to Isidori.
- **peptide_elongation** – folder with analogical structure as in the previous case

If one wants to study unknown system, let's say my_system then script *experiments_my_system.m* should be created in the folder **bil_sys_id** from where the function:
*run_experiment('**studied_systems\my_system\my_exp**',@test_f1,@test_f2)*
will be called as well as folder in **studied_systems\my_system** must be created which will contain at least one experiment **\my_exp**. The folder with experiment will contain information about the system specification and identification specification with structure as explained above. Function *run_experiment.m* is described bellow. The paths to folder with identification and studied system should be added to Matlab paths in the beginning of the script.

The second folder is **identification**. It contains function *run_experiment.m* which is called from above mentioned scripts (as *experiments_my_system.m*) in the folder **bil_sys_id** as:
run_experiment('relative_path_to_experiment',@test_f1,@test_f2).
With the explanation of the experiment also all the other content of identification folder will be clarified. The function *run_experiment.m*:

1. create folders for results from bilinear identification:
   bil_sys_id\studied_systems\my_system\my_exp\results_bilin
   and results from linear identification:
   bil_sys_id\studied_systems\my_system\my_exp\results_lin
2. read the system from the folder with **system_specification** using the function *read_sys.m* from the subfolder **read_write_IO**

   Both, bilinear and linear system are represented as a structure array called sys_spec which groups related data in fields. Our bilinear system representation has following fields:

   - state matrix A_c
   - input matrix B_c
   - output matrix C
   - transmission matrix D
   - bilinear matrices are stored to 3–dimensional array Ncs
   - initial condition x_0

   Since linear system is only a subclass of bilinear system, we use the same representation, where the N_{ci} matrices will have only zero values.

3. read the data from the folder with **id_meth_specification** using the function *read_id_meth_params.m* from the subfolder **read_write_IO**

   The parameters of the identification method are stored in the structure array id_meth_params with the fields: Δt l_0, l, s, p, *precision*, α, β, W.

4. compute some auxiliary calculations as e.g. eigenvalues of state matrix, recommended sampling time etc.

5. from loaded specification id_meth_params we can generate input used in identification using function *generate_input_data.m* from the folder **generate_data**

6. generates output Y_p of the system specified in sys_spec for input generated in previous step using *generate_output_data.m* from the folder **generate_data**

7. runs bilinear identification so that call function *bilin_id.m* which takes as input arguments id_meth_params generated output and all the applied vectors of pulses and returns system specification of identified system in the structure new_sys_spec. The bilinear identification procedure takes several steps:

   - check if some of the parameters from id_meth_params fulfil the conditions naturally arising from the Juang's algorithm described in detail in chapter 3 of the thesis, so the *check_the_id_meth_settings.m* from the folder **read_write_IO** is called with id_meth_params as input argument and returns *true* if the setting of parameters is incorrect.
   - followed by generation of Hankel matrix from the free decay after first applied pulse using the function *generateHankelMatrix.m* from the folder **generate_data**. The function is called with input arguments: Y_p is generated output, parameters α,β,m from id_meth_params determing size of the Hankel matrix as $\alpha m \times \beta$ and k determines at what time output start to form Hankel matrix, $k = 1$ means that we start use output data from $t = \Delta t$, thus after the first pulse was applied. The function returns Hankel matrix H_1 and first column of Hankel matrix shorten by last m rows h1_up used for the estimation of state x_1 in the following step.
   - Then four steps of Juang's identification method follows:
     (a) function *first_step.m* with input arguments Hankel matrix H_1, output vector h1_up as defined in the previous step used for the estimation of the state x_1 and the parameters of identification method id_meth_params. Using the Hankel matrix decomposition the first step returns state matrix A_c, output matrix C, dimension of the state space n, which are all stored in the structure new_sys_spec. Next it returns the estimation of the state x_1 after first applied pulse, later used to identification of initial condition, the estimation of the state x_{l0} which is an initial condition for the next step when second pulse is applied, matrix U1_up is used for the estimation of states in the second step, matrix A what is matrix of

corresponding linear discrete–time system during the free decay and is also used in the second step.

(b) function *second_step.m* is used when the i–th pulse vector where i can be $1 \dots s$ is applied p times and bilinear system then actually corresponds to the discrete–time linear system in the form $A_{di}x + B_{di}$. The second step allows us to identify and return all matrices $A_{d1} \dots A_{ds}$ in 3D array all_Ai_d and also $B_{d1} \dots B_{ds}$ as 3D array all_bi_d. Also all the estimated states at which the pulses are applied are returned as the matrix X. The input arguments of the second step are measured output Y_p, matrix A what is matrix of corresponding linear discrete–time system during the free decay identified during the first step, initial condition for second pulse x_{l0}, structure with identification parameters id_meth_params and dimension of the system identified in the previous step n.

(c) The function *third_step.m* returns the initial condition x_0 and transmission matrix D which are stored in the new_sys_spec. The input arguments are matrices identified in previous step all_Ai_d and all_bi_d, although we just need the first A_{d1} and B_{d1} to identify the initial condition. To identify initial condition also the input argument x_1 is needed. To identify matrix D the following input arguments are needed: measured outputs Y_p, matrix with all applied pulses U, earlier identified output matrix C, all the estimated states at which the pulses are applied, which were returned from previous step as X and parameters of the identification method id_meth_params.

(d) The function *fourth_step.m* transform the matrices identified in the second step and returns bilinear matrices Ncs and input matrix B_c of identified system stored in the structure new_sys_spec. The needed input arguments are all_Ai_d, all_bi_d returned from the second step, state matrix A_c and dimension of the system n identified during the first step as well as the structure with parameters of identification method id_meth_params.

All the functions for identification steps are saved in the folder **bilin_procedure**.

8. evaluation of the identification has following steps:

(a) The 6. step of the experiment is repeated but now with the identified system: generates output Y_{id} of the system specified in new_sys_spec for input generated in the 5.step of the experiment – which means that function *generate_output_data.m* is used to generate output of the identified system to the same input as was originally used for the identification

(b) outputs of original system Y_p from the 6.step and response to the pulses of the identified system Y_{id} is compared in measure points and in all points of the integration using the function *compare_pulses.m* from the folder **evaluation**, which plots the outputs and returns also computed norm c_{pulses} computed as normed difference in measured output points

of original and identified system divided by number of measured points n_{points}:

$$c_{pulses} = \frac{||Y_p - Y_{id}||}{n_{points}} \tag{A.1}$$

(c) function *run_experiment.m* is called with 2 testing functions *test_f1* and *test_f2* for which the original and identified system is compared using the function *compare.m* from the folder **evaluation**. Testing functions used in the thesis are in the folder **evaluation\test_input_functions**. For given testing function e.g.*test_f1* the output for both original y_o and identified y_i system is generated using *generate_output_continuous.m* and then the norm c_{test_f1} is computed in the all points used for the integration $n_{intPoints}$

$$c_{test\_f1} = \frac{||y_o - y_i||}{n_{intPoints}} \tag{A.2}$$

Both outputs for original and identified systems are plotted to show possible differences and the graphs are saved as *comparison_u1.png* for function *test_f1* and *comparison_u2.png* for function *test_f2* .
All the norms c_{pulses}, c_{test_f1}, and c_{test_f2} are written to the file: *c_output_norms_bilinear.txt* with the function *write_norms.m* from the folder **read_write_IO**.

(d) if the original and the identified system have the same dimension n of the state space then system can be transformed to original structure with the function *transformation.m* from the folder **evaluation**. Transformed M_t and original system matrices M_o are compared and the norm is computed:

$$c_M = ||M_t - M_o|| \tag{A.3}$$

The norms for all the system matrices are written to the file *c_matrix_norms.txt*.

(e) Also the eigenvalues for the state matrix A_c and bilinear matrices Ncs are computed for original and identified system and saved to the files *eig_or.txt* and *eig_new.txt*.

All the computed norms, eigenvalues and plots are saved in the folder **results_bilin** in the folder corresponding to the running experiment (from where the data about original system were obtained).

9. The linear identification is of the system is proceeded.

(a) input for the linear identification is generated using the same function as for bilinear case *generate_input_data.m* but we have to specify in the input argument that the type of the system is linear.

(b) At first the free response of the system Y_{free} is computed with function *generate_output_data.m*.

(c) Then response to the pulses (always only one input is turned on and the other inputs are turned off, thus we compute the response to r different inputs vector now using the function *generate_output_data.m*) is saved to matrix Y_p.

(d) The identification is in the file *lin_id.m* and is called with input arguments id_meth_params, Y_{free}, Y_p, and $forces$ which carry the information about the magnitude of individual pulses. The identification procedure is decribed in detil in the chapter 3 of the thesis. The identified system is saved to the structure lin_sys_spec.

(e) The same process of evaluation as explained in the 8. step of the experiment is done also for identified linear system. All the computed norms, eigenvalues and plots are saved in the folder **results_lin** in the folder corresponding to the running experiment (from where the data about original system were obtained).