

Regionální inovační centrum elektrotechniky
Fakulta elektrotechnická
Západočeská univerzita v Plzni

Základy automatického generování kódu z Matlabu

Mikroprocesorové řízení pohonů 2

Jakub Talla

1) Úvod do generování kódu z Matlabu

- ▶ Základy generování kódu z Matlabu

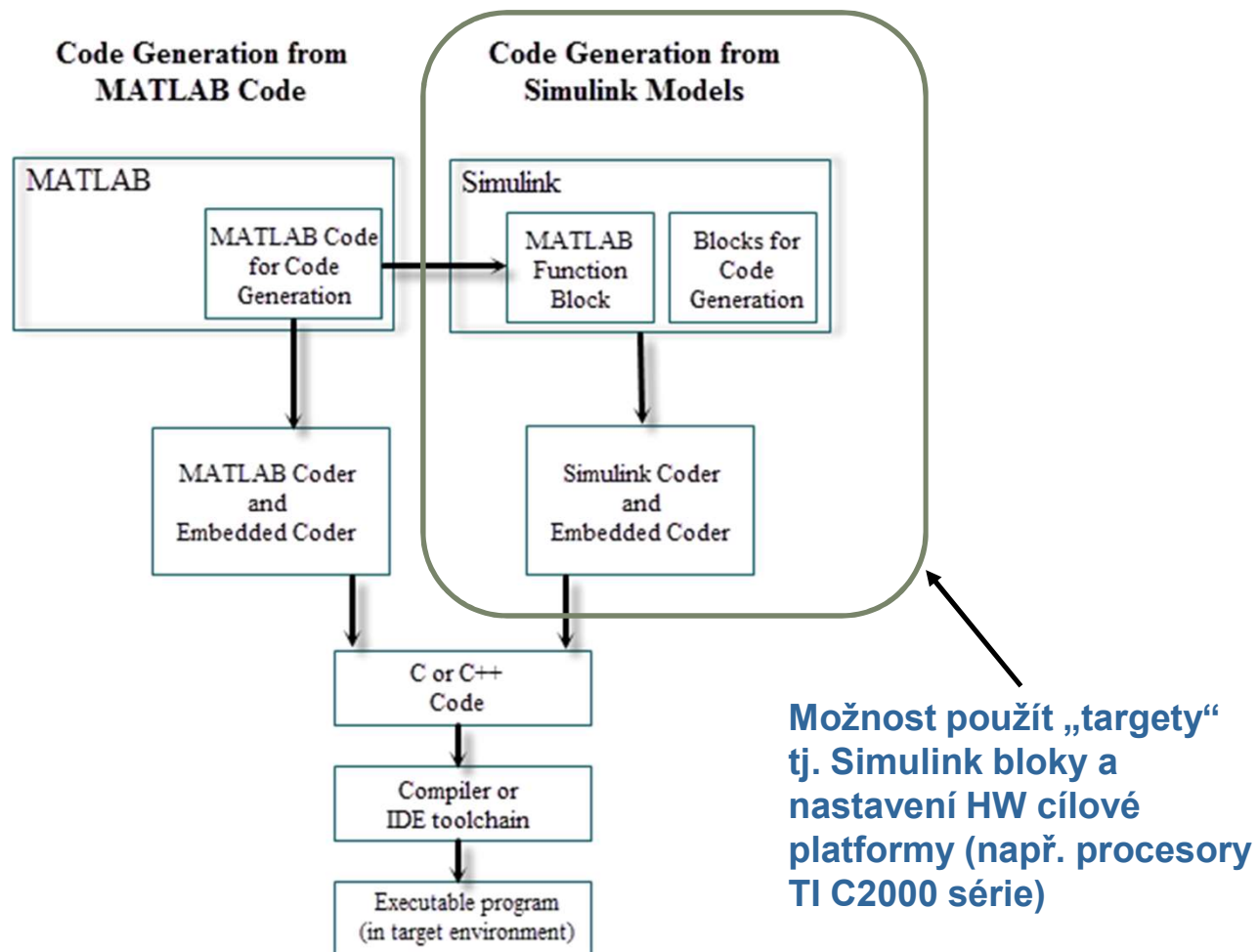
2) Automatické generování kódu z Matlabu

- ▶ Použitím příkazu codegen
- ▶ Použitím aplikace Matlab coder

3) Možnosti importu vlastního C kódu do Matlabu

- ▶ Použitím příkazu mex
- ▶ Použitím aplikace Matlab coder

Základní generování C/C++ kódu z Matlab/Simulink



Generování C kódu z Matlabu

Automatické generování C kódu z Matlabu

Využívá Matlab coder toolbox

Slouží převážně k:

- Akceleraci výpočtů na PC (vytvoření MEX souboru)
- Generování částí aplikačního SW pro PC (vytvoření .C nebo .lib)
- Generování zdrojového kódu aplikačního SW pro embedded zařízení (mikrokontroléry....)

Nutné části:

- Matlab
- Matlab coder
- Podporovaný C/C++ compiler (Visual studio, MinGW,....)

Možnosti testování a generování C kódu v Matlabu

codegen – slouží ke generování kódu, umí vygenerovat zdrojové kódy, knihovny, ale i spustitelné mex (z Matlab) a .exe

Možnost kompilace:

- 1) Z příkazové řádky: `codegen FceGen2 -args {x1in, x2in} -nargout 2`
- 2) Pomocí Matlab coder aplikace

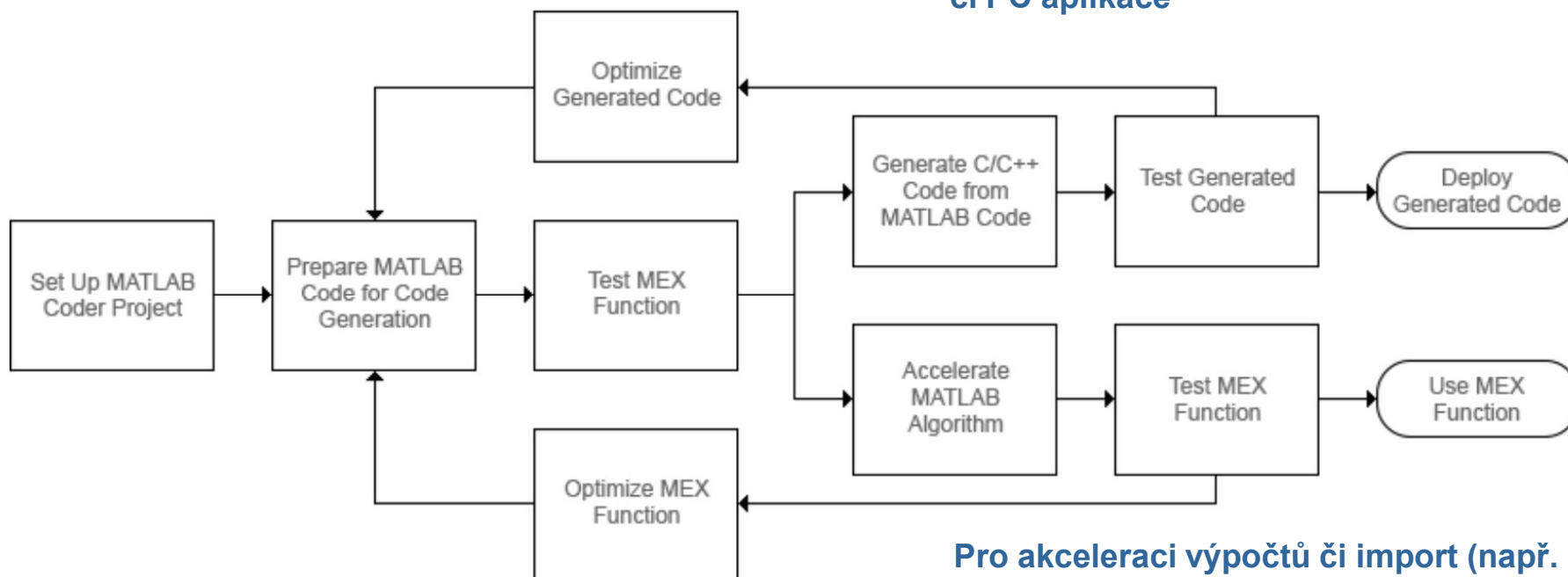
Mex – slouží k vytvoření a zkompilování do mex souboru vlastní projekt ze zdrojových souborů C/C++

Možnost kompilace:

- 1) Z příkazové řádky: `mex parametry` (nutné nadefinovat datové rozhraní s Matlabem ve zdrojovém kódu)
- 2) Pomocí Matlab coder aplikace

Automatické generování C kódu z Matlabu

Pro generování kódu pro mikroprocesory
či PC aplikace

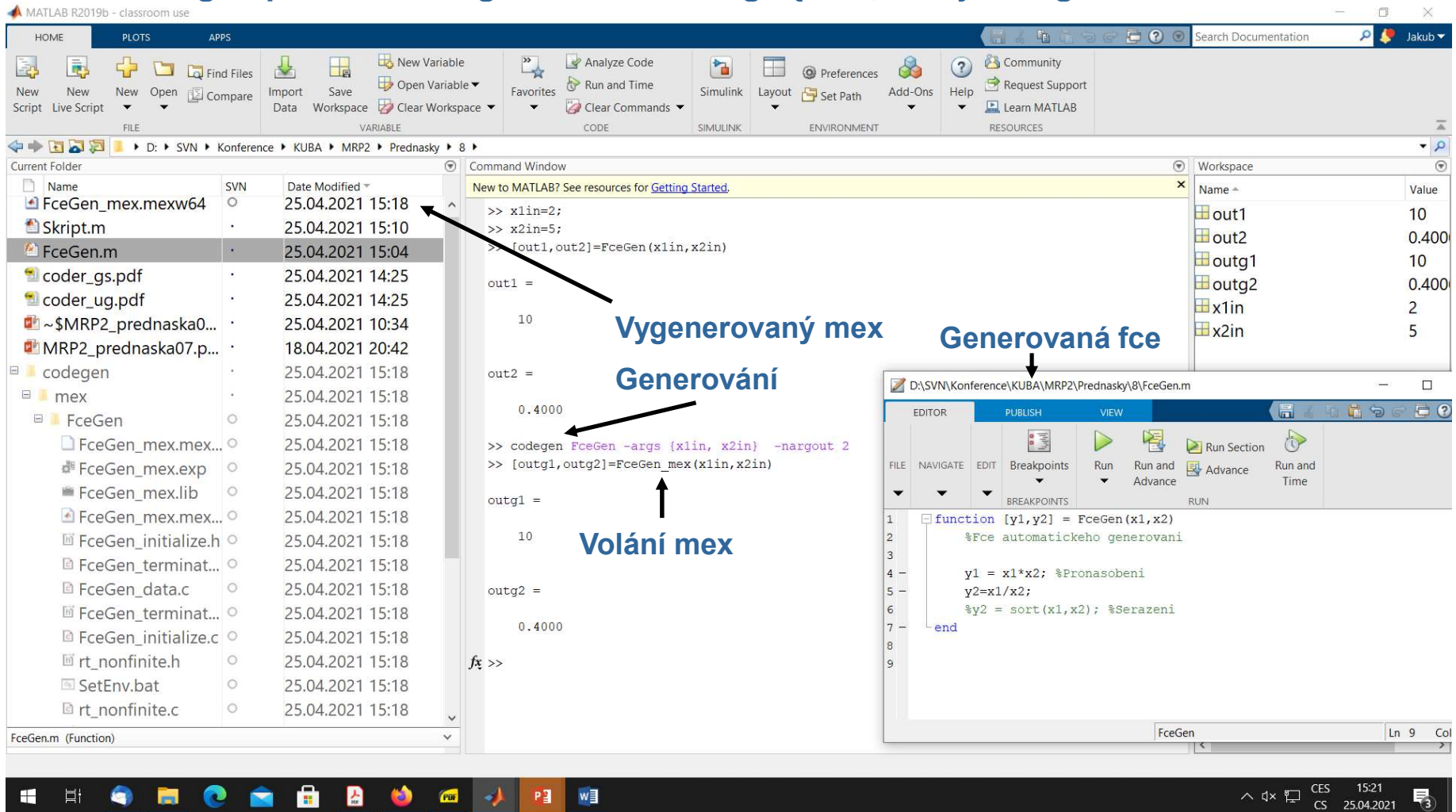


Pro akceleraci výpočtů či import (např.
pro otestování) uživatelských C funkcí

Codegen - generování C kódu z Matlab scriptu/funkce

Generování C kódu z příkazové řádky

Codegen příklad: `codegen FceGen2 -args {x1in, x2in} -nargout 2`



The screenshot illustrates the MATLAB environment during code generation. The Command Window shows the following commands and output:

```

>> x1in=2;
>> x2in=5;
>> [out1,out2]=FceGen(x1in,x2in)

out1 =
    10

out2 =
    0.4000

>> codegen FceGen -args {x1in, x2in} -nargout 2
>> [outg1,outg2]=FceGen_mex(x1in,x2in)

outg1 =
    10

outg2 =
    0.4000

fx >>
  
```

Annotations in the image identify key parts of the process:

- Generování** (Generation): Points to the `codegen` command in the Command Window.
- Vygenerovaný mex** (Generated mex): Points to the `FceGen_mex.mexw64` file in the Current Folder.
- Generovaná fce** (Generated fce): Points to the `FceGen.m` function file in the Current Folder.
- Volání mex** (Calling mex): Points to the `[outg1,outg2]=FceGen_mex(x1in,x2in)` command in the Command Window.

The Workspace window shows the following variables:

Name	Value
out1	10
out2	0.4000
outg1	10
outg2	0.4000
x1in	2
x2in	5

The Editor window shows the source code of `FceGen.m`:

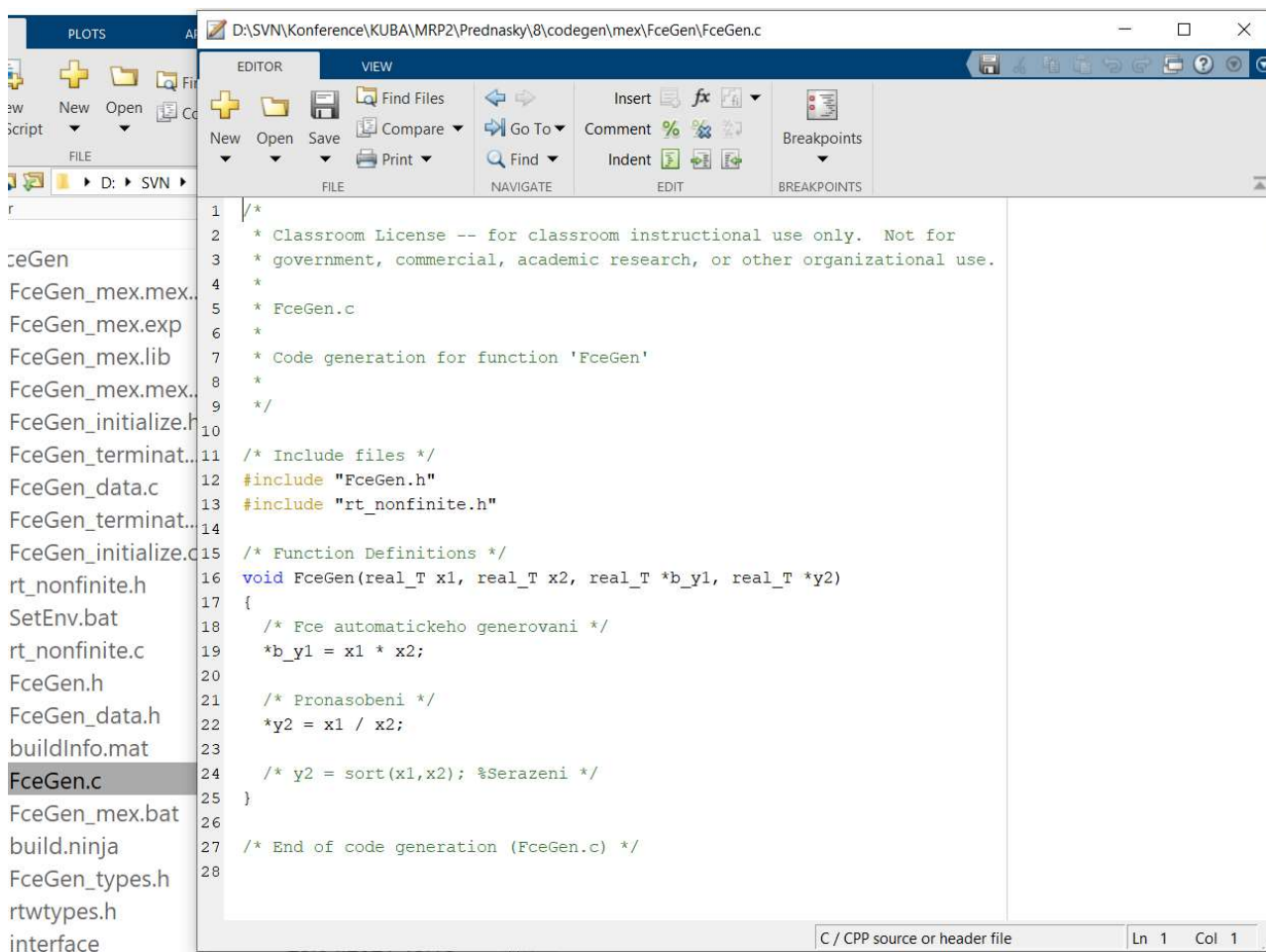
```

function [y1,y2] = FceGen(x1,x2)
    %Fce automatickeho generovani

    y1 = x1*x2; %Pronasobeni
    y2=x1/x2;
    %y2 = sort(x1,x2); %Serazeni
end
  
```

Generování C kódu z příkazové řádky

Příklad vygenerovaného zdrojového C kódu funkce FceGen



```
1 /*
2  * Classroom License -- for classroom instructional use only. Not for
3  * government, commercial, academic research, or other organizational use.
4  *
5  * FceGen.c
6  *
7  * Code generation for function 'FceGen'
8  *
9  */
10
11 /* Include files */
12 #include "FceGen.h"
13 #include "rt_nonfinite.h"
14
15 /* Function Definitions */
16 void FceGen(real_T x1, real_T x2, real_T *b_y1, real_T *y2)
17 {
18     /* Fce automatickeho generovani */
19     *b_y1 = x1 * x2;
20
21     /* Pronasobeni */
22     *y2 = x1 / x2;
23
24     /* y2 = sort(x1,x2); %Serazeni */
25 }
26
27 /* End of code generation (FceGen.c) */
28
```

Generování C kódu z příkazové řádky

Pro vygenerování kódu je často nutná řada nastavení (rozměry a datové typy vstupů, mex/exe/lib apod.), které se zadávají jako parametry funkce `codegen`

COMMON OPTIONS:

- args ARGS Specify the types that the generated code should accept. `ARGS` is a cell array specifying the type of each function argument. (Elements are converted to types using `coder.typeof`.)

- nargout NARGOUT Specify the number of output arguments for the given function. If using `VARARGOUT`, this is required.

- config:mex Generate a MEX function (default)
- config:lib Generate a C/C++ static library
- config:dll Generate a C/C++ dynamic library
- config:exe Generate a C/C++ executable
- config:hdl Generate VHDL/Verilog code

- config CONFIG Specify a custom configuration object (use `coder.config` to create one).

- double2single CONFIG Specify a doubles to singles configuration object to convert double-precision MATLAB code to single-precision MATLAB code (use `coder.config` to create one).

- float2fixed CONFIG Specify a fixpt configuration object (use `coder.config` to create one).

- o OUTPUT Specify the name of the output. **codegen** adds a platform-specific extension to this name depending on the output type.

Generování C kódu z příkazové řádky

Další nastavení se provádí příkazem *coder*. Většinu nastavení lze „naklikat“ v aplikaci Matlab *coder*.

```
cfg = coder.config('lib');
```

Příkaz *coder* najde uplatnění i v konfiguraci Matlab scriptu/funkci

Například definování proměnné *h* s dynamickou velikostí jednorozměrného pole o až 10 prvcích apod. (v Matlabu není nutné definovat velikost proměnných)

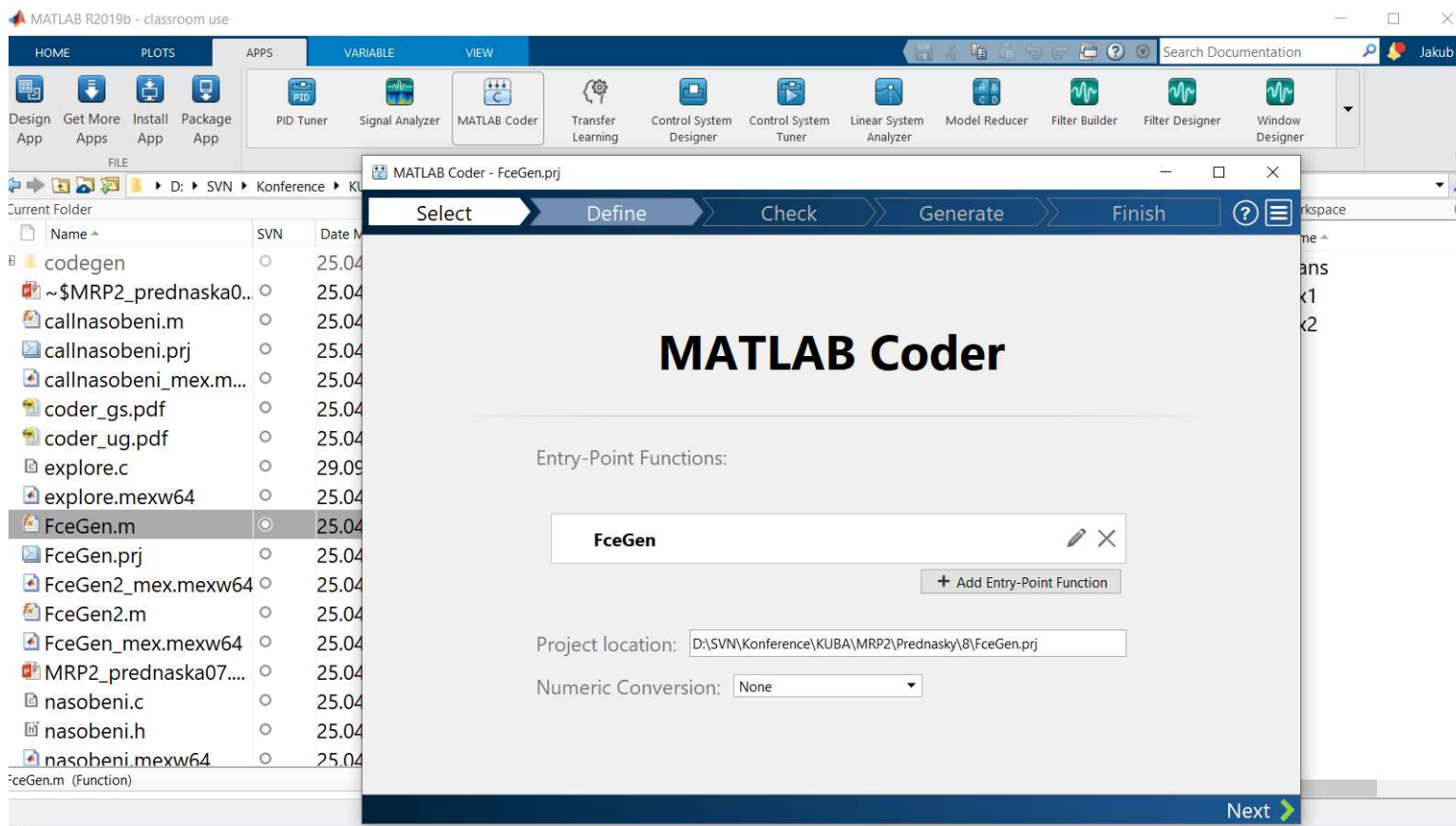
```
cfg = coder.config('lib');  
coder.varsize('h', [1 10]);  
h=...
```

Příkazem *coder* je také možné volat např. uživatelské c funkce v rámci vygenerovaného kódu

```
coder.updateBuildInfo('addSourceFiles','nasobeni.c');  
coder.cinclude('nasobeni.h');
```

```
y = coder.ceval('nasobeni', x1, x2);
```

Generování kódu pomocí Matlab Coder prostředí



MATLAB R2019b - classroom use

HOME PLOTS APPS VARIABLE VIEW Search Documentation Jakub

Design App Get More Apps Install App Package App

PID Tuner Signal Analyzer MATLAB Coder Transfer Learning Control System Designer Control System Tuner Linear System Analyzer Model Reducer Filter Builder Filter Designer Window Designer

FILE D: \SVN \Konference \K... MATLAB Coder - FceGen.prj

Current Folder

Name	SVN	Date Modified
codegen		25.04
~\$MRP2_prednaska0...		25.04
callnasobeni.m		25.04
callnasobeni.prj		25.04
callnasobeni_mex.m...		25.04
coder_gs.pdf		25.04
coder_ug.pdf		25.04
explore.c		29.09
explore.mexw64		25.04
FceGen.m		25.04
FceGen.prj		25.04
FceGen2_mex.mexw64		25.04
FceGen2.m		25.04
FceGen_mex.mexw64		25.04
MRP2_prednaska07....		25.04
nasobeni.c		25.04
nasobeni.h		25.04
nasobeni.mexw64		25.04

Entry-Point Functions:

FceGen

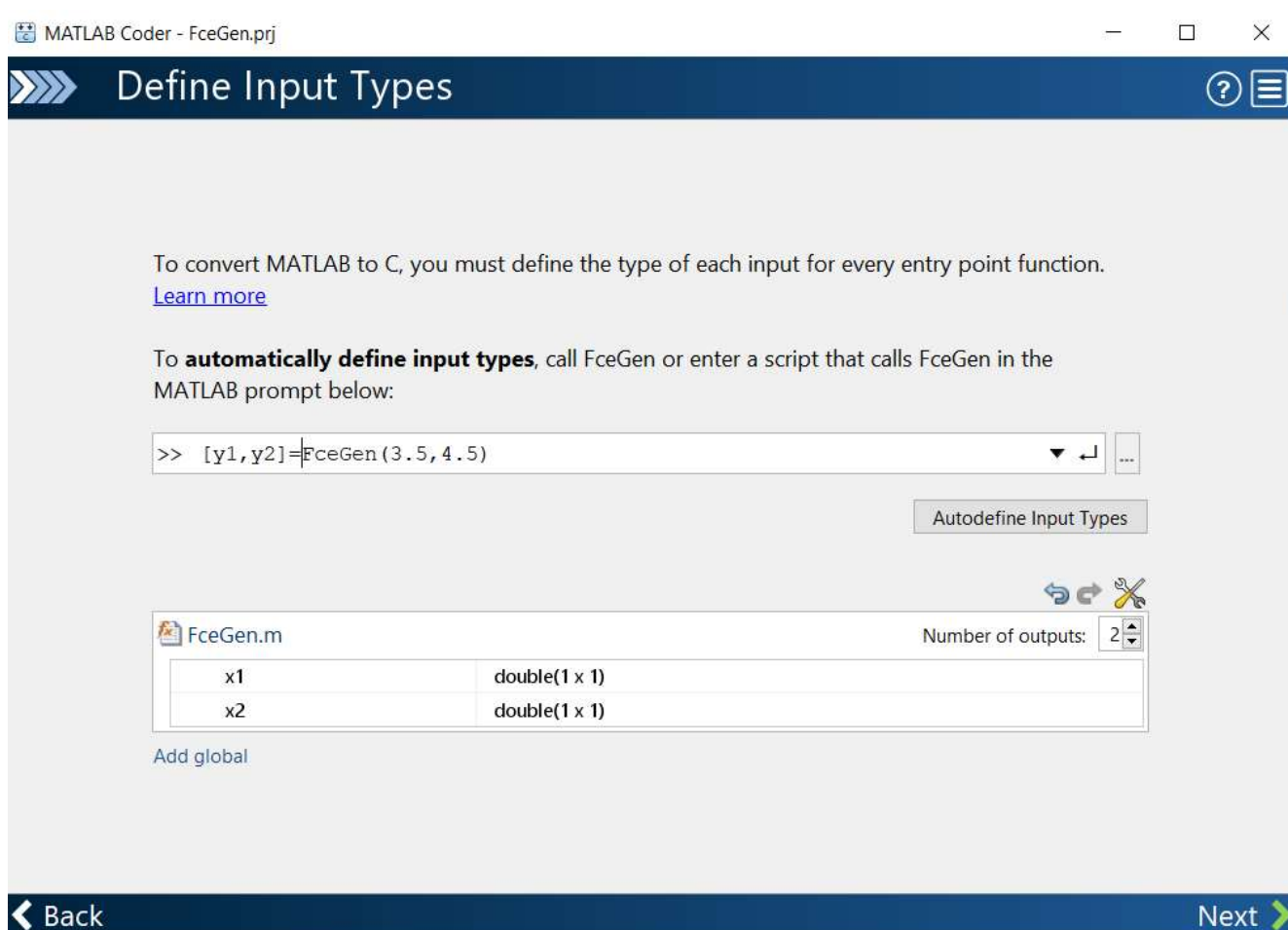
+ Add Entry-Point Function

Project location: D:\SVN\Konference\KUBA\MRP2\Prednasky\8\FceGen.prj

Numeric Conversion: None

Next >

Generování kódu pomocí Matlab Coder prostředí - definování vstupů



MATLAB Coder - FceGen.prj

Define Input Types

To convert MATLAB to C, you must define the type of each input for every entry point function.
[Learn more](#)

To **automatically define input types**, call FceGen or enter a script that calls FceGen in the MATLAB prompt below:

```
>> [y1, y2] = FceGen(3.5, 4.5);
```

Autodefine Input Types

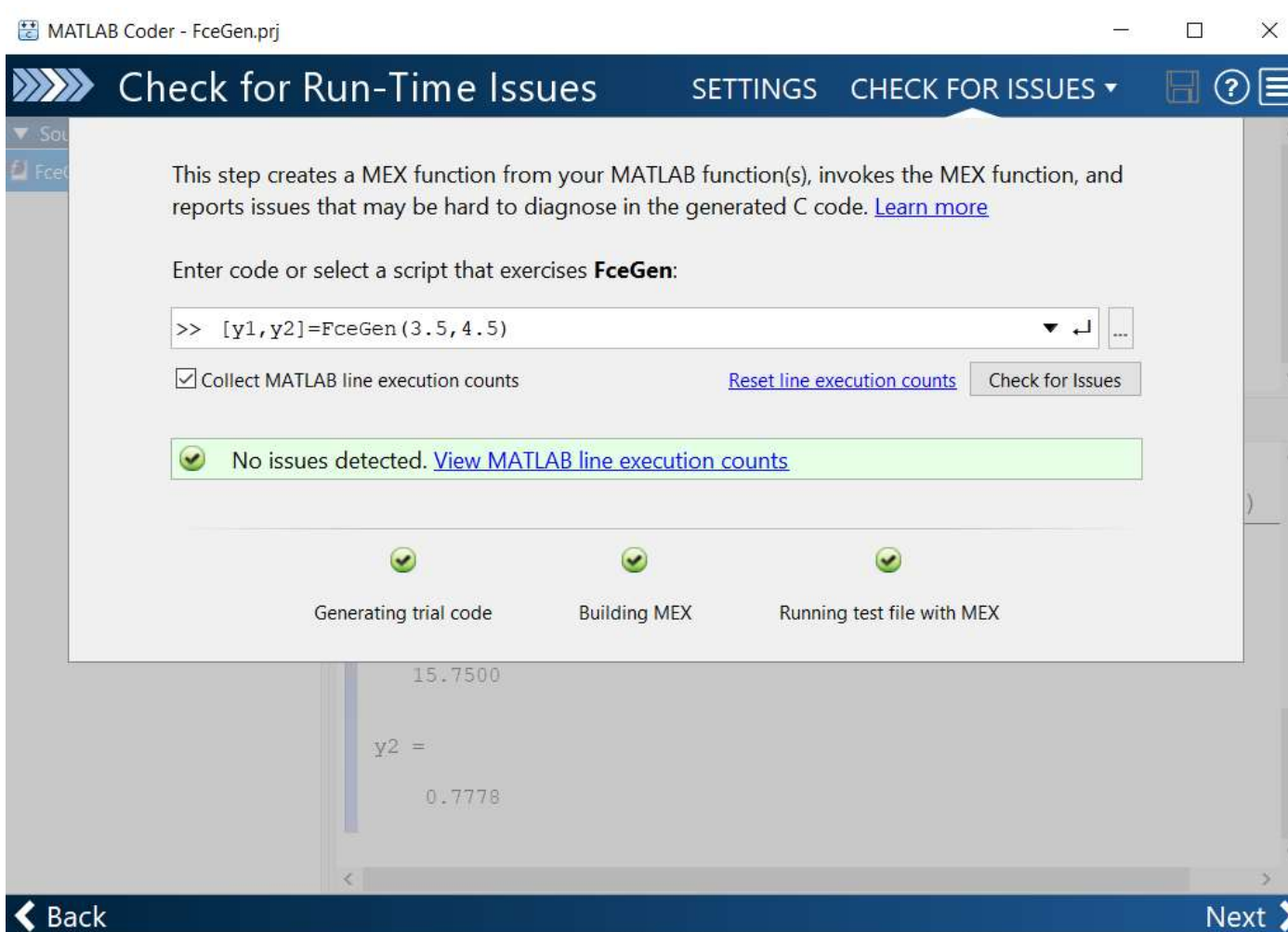
FceGen.m Number of outputs: 2

x1	double(1 x 1)
x2	double(1 x 1)

Add global

Back Next

Vytvoření mex a otestování funkce (rozdíl mezi mex a Matlab kódem)



The screenshot shows the MATLAB Coder interface for the 'Check for Run-Time Issues' step. The dialog box contains the following information:

- Title:** Check for Run-Time Issues
- DESCRIPTION:** This step creates a MEX function from your MATLAB function(s), invokes the MEX function, and reports issues that may be hard to diagnose in the generated C code. [Learn more](#)
- INSTRUCTIONS:** Enter code or select a script that exercises **FceGen**:
- CODE INPUT:** A text field containing the MATLAB command: `>> [y1,y2]=FceGen(3.5,4.5)`
- OPTIONS:** A checkbox labeled 'Collect MATLAB line execution counts' is checked. A 'Reset line execution counts' link and a 'Check for Issues' button are also present.
- RESULTS:** A green message box states: 'No issues detected. [View MATLAB line execution counts](#)'
- PROGRESS:** Three steps are shown with green checkmarks: 'Generating trial code', 'Building MEX', and 'Running test file with MEX'.
- OUTPUT:** Below the dialog, the MATLAB Command Window shows the output: `15.7500` and `y2 = 0.7778`.
- NAVIGATION:** 'Back' and 'Next' buttons are located at the bottom of the dialog.

Další konfigurace projektu

MATLAB Coder - FceGen.prj

Check for Run-Time Issues SETTINGS CHECK FOR ISSUES

Source Code: function [v1,v2] = FceGen(x1) Change project settings

Type to search for settings

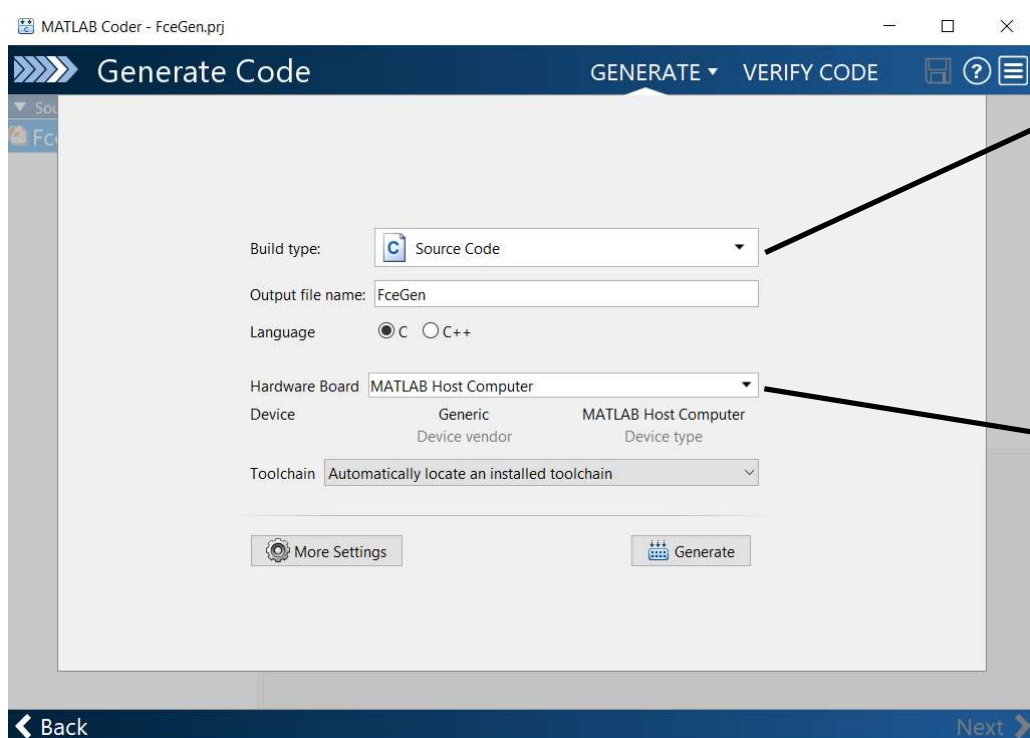
Setting	Value
Paths	
Working folder	Project folder
Build folder	A subfolder of the project folder
Search paths	
Speed	
Enable responsiveness to CTRL+C and graphi...	Yes
Keep extrinsic calls	Yes
Ensure memory integrity	Yes
Saturate on integer overflow	Yes
Memory	
Global data synchronization mode	At MEX-function entry, exit and extrinsic calls
Variable Sizing Support	
Enable variable-sizing	Yes
Dynamic memory allocation	For arrays with max size at or above threshold
Dynamic memory allocation threshold	65536
Array layout	Column-major

Paths
 Speed
 Memory
 Code Appearance
 Debugging
 Custom Code
All Settings
 MISRA Compliance
 Import/Export Settings






Help Close

Back Next

Generování projektu



Varianty implementace




	Source Code	C/C++ source code to integrate with an external project
	MEX	Compiled code to run inside MATLAB
	Static Library (.lib)	Binary library for static linking with an external project
	Dynamic Library (.dll)	Binary library for dynamic linking with an external project
	Executable (.exe)	Standalone program (requires a separate main file written in C)

Varianty cílové platformy (ovlivňují hlavně velikost int apod.)

Hardware Board	None - Select device below	
Device	Texas Instruments	C2000
	Device vendor	Device type
Toolchain	Texas Instruments Code Composer Studio (C2000)	

Vygenerovaný projekt

MATLAB Coder - FceGen.prj

Generate Code GENERATE ▾ VERIFY CODE   

Source Code

FceGen

```

15 /*
16 * Fce automatickeho generovani
17 * Arguments   : real_T x1
18 *              real_T x2
19 *              real_T *b_y1
20 *              real_T *y2
21 * Return Type : void
22 */
23 void FceGen(real_T x1, real_T x2, real_T *b_y1, real_T *y2)
24 {
25     *b_y1 = x1 * x2;
26
27     /* Pronasobeni */
28     *y2 = x1 / x2;
29
30     /* y2 = sort(x1,x2); %Serazeni */
31 }
32

```

Output Files

- FceGen_initialize.c
- FceGen_terminate.c
- FceGen.c**
- main.c
- FceGen_initialize.h
- FceGen_terminate.h
- FceGen_types.h
- FceGen.h
- main.h
- rtwtypes.h
- report.mldatx
- rtw_proj.tmw

Variable	Type	Size
Input		
x1	double	1 x 1
x2	double	1 x 1
Output		
y1	double	1 x 1
y2	double	1 x 1

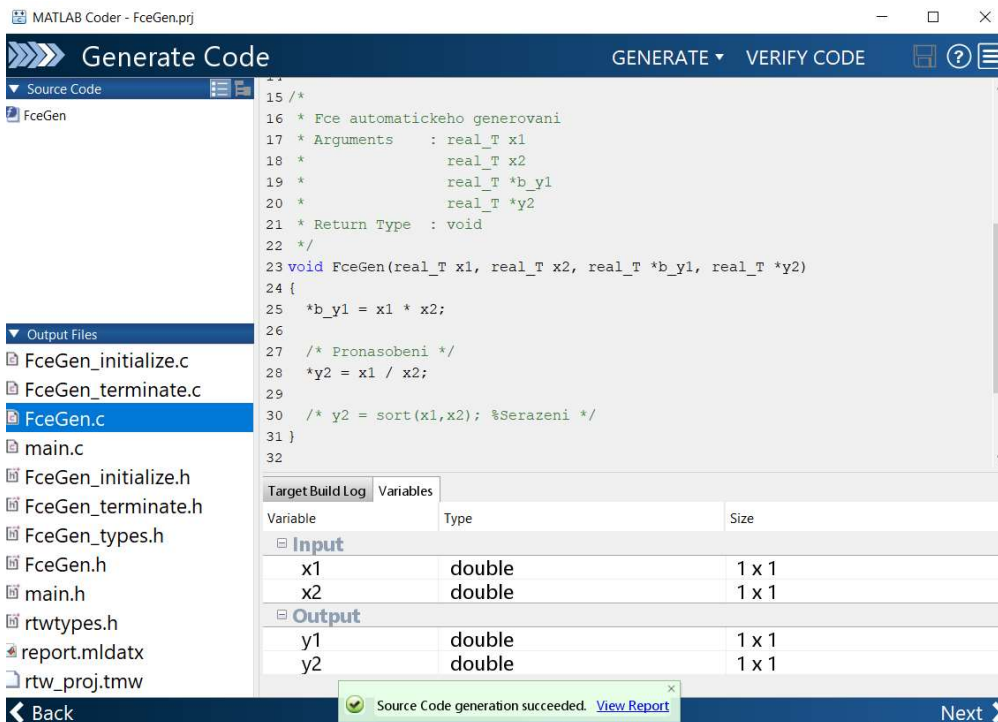
Target Build Log Variables

Source Code generation succeeded. [View Report](#)

Back Next >

Vygenerovaný projekt

FceGen.c vlastní tělo funkce



MATLAB Coder - FceGen.prj

Generate Code GENERATE VERIFY CODE

Source Code

```

15 /*
16 * Fce automatickeho generovani
17 * Arguments   : real_T x1
18 *              real_T x2
19 *              real_T *b_y1
20 *              real_T *y2
21 * Return Type : void
22 */
23 void FceGen(real_T x1, real_T x2, real_T *b_y1, real_T *y2)
24 {
25     *b_y1 = x1 * x2;
26
27     /* Pronasobeni */
28     *y2 = x1 / x2;
29
30     /* y2 = sort(x1,x2); %Serazeni */
31 }
32

```

Output Files

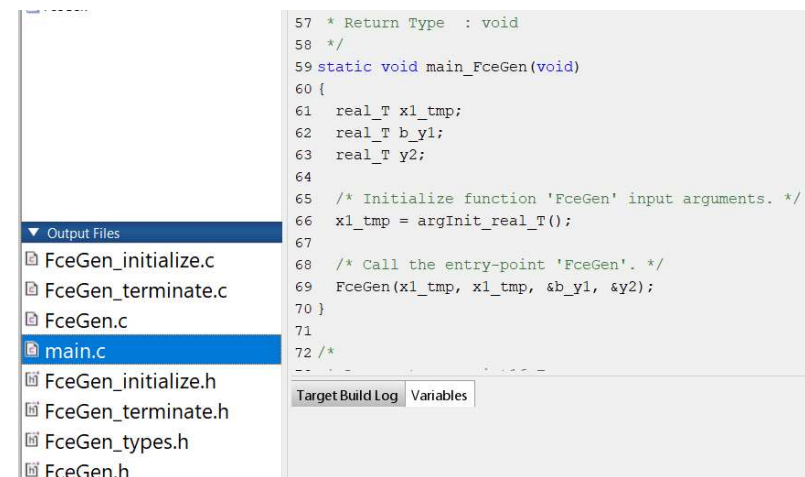
- FceGen_initialize.c
- FceGen_terminate.c
- FceGen.c**
- main.c
- FceGen_initialize.h
- FceGen_terminate.h
- FceGen_types.h
- FceGen.h
- main.h
- rtwtypes.h
- report.mldatx
- rtw_proj.tmw

Target Build Log Variables

Variable	Type	Size
Input		
x1	double	1 x 1
x2	double	1 x 1
Output		
y1	double	1 x 1
y2	double	1 x 1

Source Code generation succeeded. [View Report](#)

main.c volání funkce



Output Files

- FceGen_initialize.c
- FceGen_terminate.c
- FceGen.c
- main.c**
- FceGen_initialize.h
- FceGen_terminate.h
- FceGen_types.h
- FceGen.h

```

57 * Return Type : void
58 */
59 static void main_FceGen(void)
60 {
61     real_T x1_tmp;
62     real_T b_y1;
63     real_T y2;
64
65     /* Initialize function 'FceGen' input arguments. */
66     x1_tmp = argInit_real_T();
67
68     /* Call the entry-point 'FceGen'. */
69     FceGen(x1_tmp, x1_tmp, &b_y1, &y2);
70 }
71
72 /*

```

Target Build Log Variables

Import a testování vlastní C funkce v Matlabu

Tvorba MEX (Matlab EXecutable) z vlastního C kódu

mex příkaz umožňuje vytvořit z prostředí Matlab spustitelný mex/mexw64 soubor

Musí mít nadefinované rozhraní C a Matlab tj. datové typy, jejich velikosti apod. (Podobně jako v Simulink S-funkce)

**Gateway funkce s
definicí vstupů
a voláním uživatelské
funkce**

Uživatelská funkce

```

#include "mex.h"

/* The computational routine */
void arrayProduct(double x, double *y, double *z, mwSize n)
{
    mwSize i;
    /* multiply each element y by x */
    for (i=0; i<n; i++) {
        z[i] = x * y[i];
    }
}

```

```

/* The gateway function */
void mexFunction( int nlhs, mxArray *plhs[],
                  int nrhs, const mxArray *prhs[])

double multiplier;          /* input scalar */
double *inMatrix;          /* 1xN input matrix */
size_t ncols;              /* size of matrix */
double *outMatrix;         /* output matrix */

/* check for proper number of arguments */
if(nrhs!=2) {
    mexErrMsgIdAndTxt("MyToolbox:arrayProduct:nrhs","Two inputs required.");
}
if(nlhs!=1) {
    mexErrMsgIdAndTxt("MyToolbox:arrayProduct:nlhs","One output required.");
}
/* make sure the first input argument is scalar */
if( !mxIsDouble(prhs[0]) ||
    mxIsComplex(prhs[0]) ||
    mxGetNumberOfElements(prhs[0])!=1 ) {
    mexErrMsgIdAndTxt("MyToolbox:arrayProduct:notScalar","Input multiplier must be a scalar.");
}

/* make sure the second input argument is type double */
if( !mxIsDouble(prhs[1]) ||
    ...
    ...
    ...
)
    ...

/* call the computational routine */
arrayProduct(multiplier,inMatrix,outMatrix,(mwSize)ncols);
}

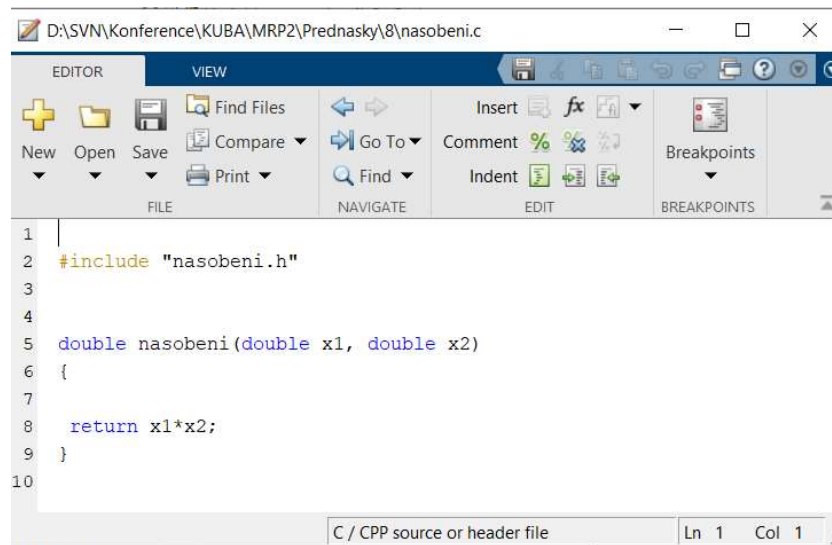
```

Tvorba MEX z vlastního C pomocí Matlab coder

Tvorbu mex z vlastního C kódu lze velmi usnadnit za pomoci Matlab coder. Často navíc chceme porovnat funkci v C s funkcí v Matlabu (cosimulací či Software In the Loop).

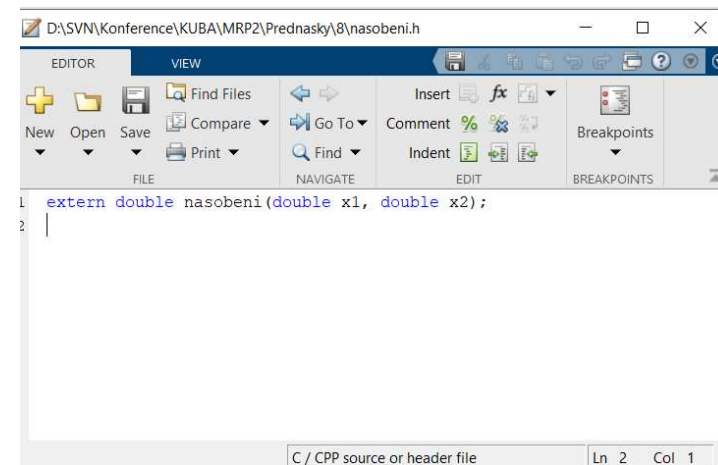
V podstatě vytvoříme v Matlabu funkci, která bude volat naši C funkci a pro tu nakonfigurujeme interface s Matlabem a vygenerujeme kód

Uživatelská funkce nasobeni v modulu nasobeni.c



```
D:\SVN\Konference\KUBA\MRP2\Prednasky\8\nasobeni.c
EDITOR VIEW
+ New Open Save Find Files Find Compare Go To Insert fx Breakpoints
FILE NAVIGATE EDIT BREAKPOINTS
1
2 #include "nasobeni.h"
3
4
5 double nasobeni(double x1, double x2)
6 {
7
8     return x1*x2;
9 }
10
C / CPP source or header file Ln 1 Col 1
```

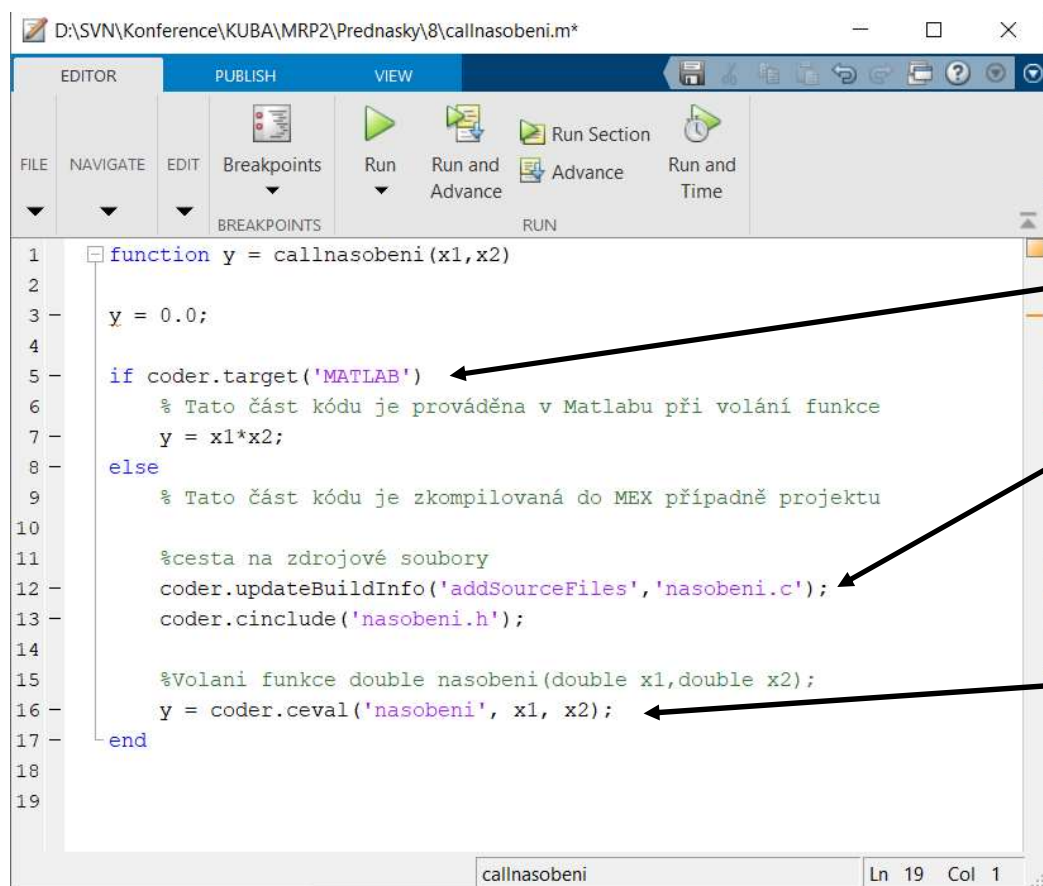
nasobeni.h



```
D:\SVN\Konference\KUBA\MRP2\Prednasky\8\nasobeni.h
EDITOR VIEW
+ New Open Save Find Files Find Compare Go To Insert fx Breakpoints
FILE NAVIGATE EDIT BREAKPOINTS
1 extern double nasobeni(double x1, double x2);
2
C / CPP source or header file Ln 2 Col 1
```

Tvorba MEX z vlastního C pomocí Matlab coder

1) Vytvoříme funkci v Matlabu, která pomocí příkazu `coder.ceval` volá naší C funkci



```

1 function y = callnasobeni(x1,x2)
2
3     y = 0.0;
4
5     if coder.target('MATLAB')
6         % Tato část kódu je prováděna v Matlabu při volání funkce
7         y = x1*x2;
8     else
9         % Tato část kódu je zkompileovaná do MEX případně projektu
10
11        %cesta na zdrojové soubory
12        coder.updateBuildInfo('addSourceFiles','nasobeni.c');
13        coder.cinclude('nasobeni.h');
14
15        %Volani funkce double nasobeni(double x1,double x2);
16        y = coder.ceval('nasobeni', x1, x2);
17    end
18
19
    
```

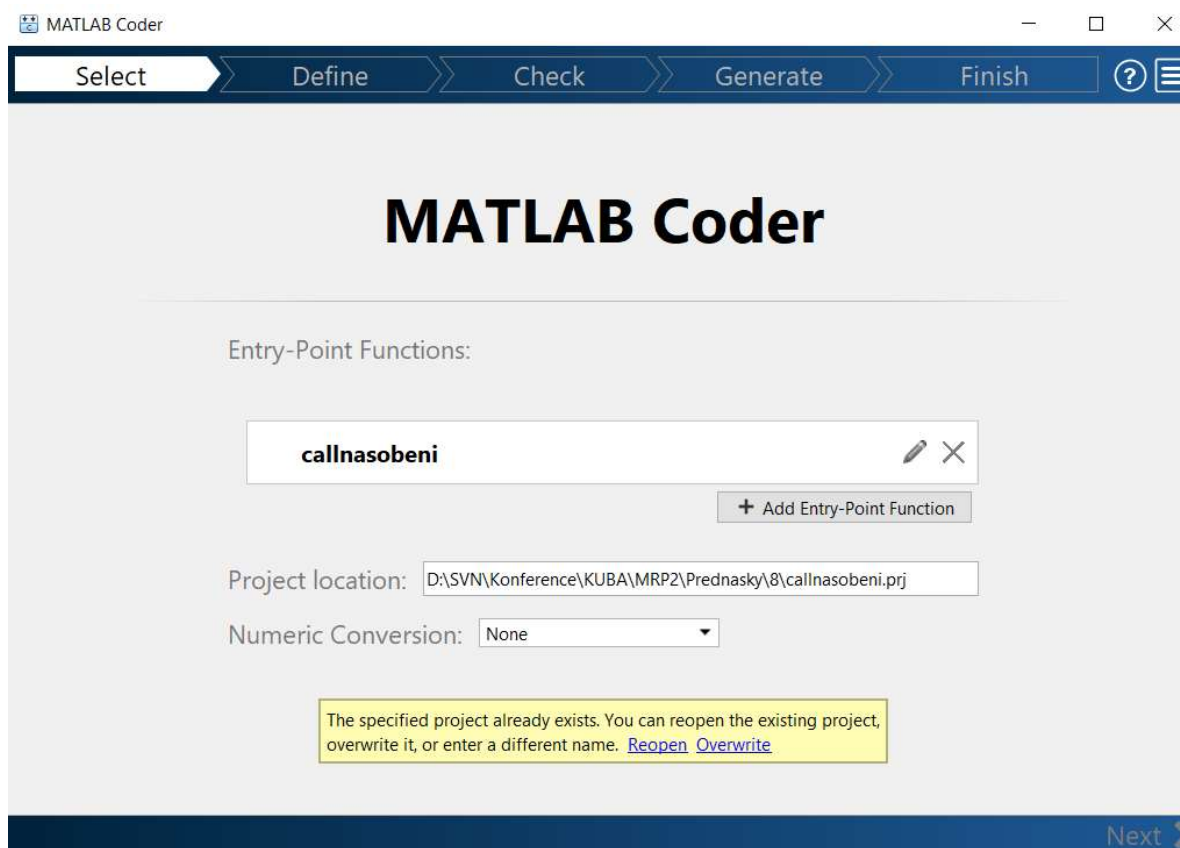
Použito pro otestování naší uživatelské funkce

Import uživatelských souborů

Volání uživatelské funkce o dvou proměnných (po zkompileování do mex)

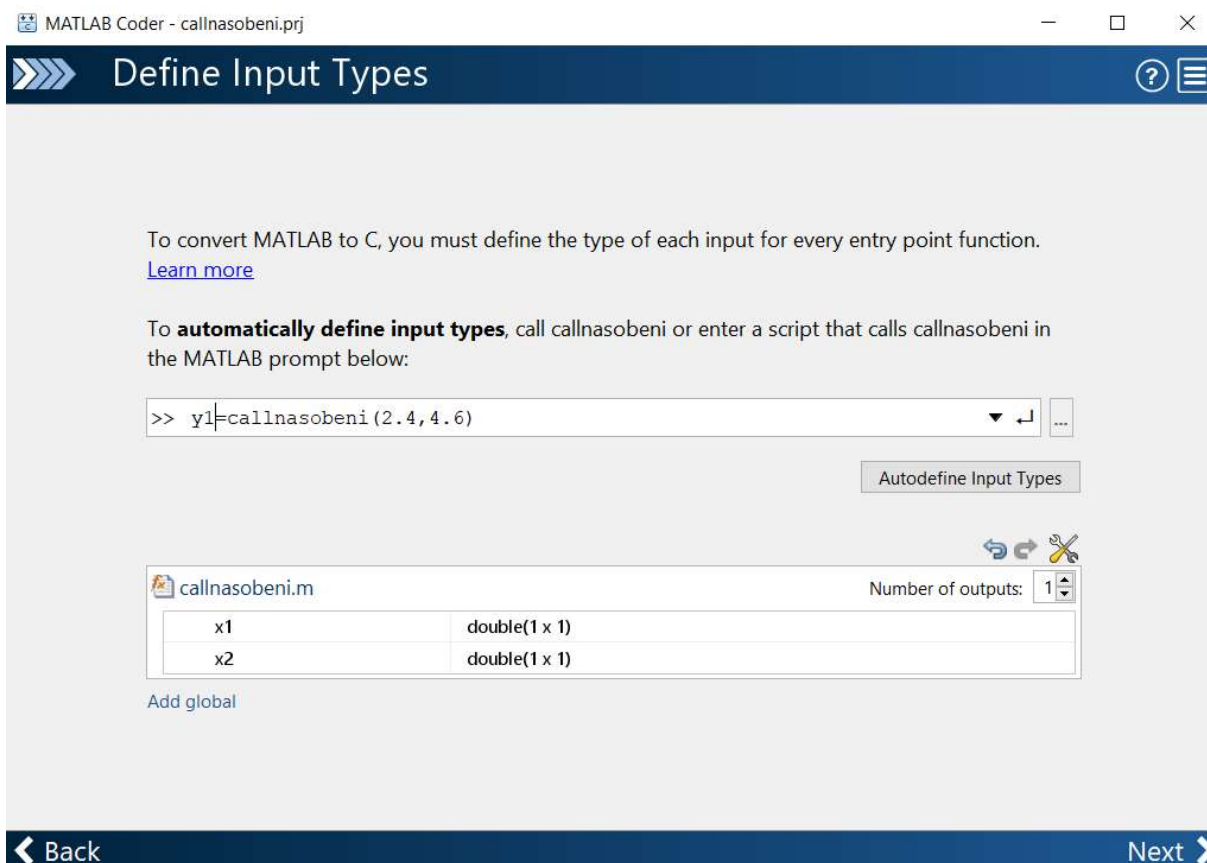
Tvorba MEX z vlastního C pomocí Matlab coder

2) Zapneme Matlab coder a vybereme funkci callnasobeni



Tvorba MEX z vlastního C pomocí Matlab coder

3) Nadefinujeme vstupy funkce callnasobeni



MATLAB Coder - callnasobeni.prj

Define Input Types

To convert MATLAB to C, you must define the type of each input for every entry point function.
[Learn more](#)

To **automatically define input types**, call callnasobeni or enter a script that calls callnasobeni in the MATLAB prompt below:

```
>> y1=callnasobeni(2.4,4.6)
```

Autodefine Input Types

callnasobeni.m Number of outputs: 1

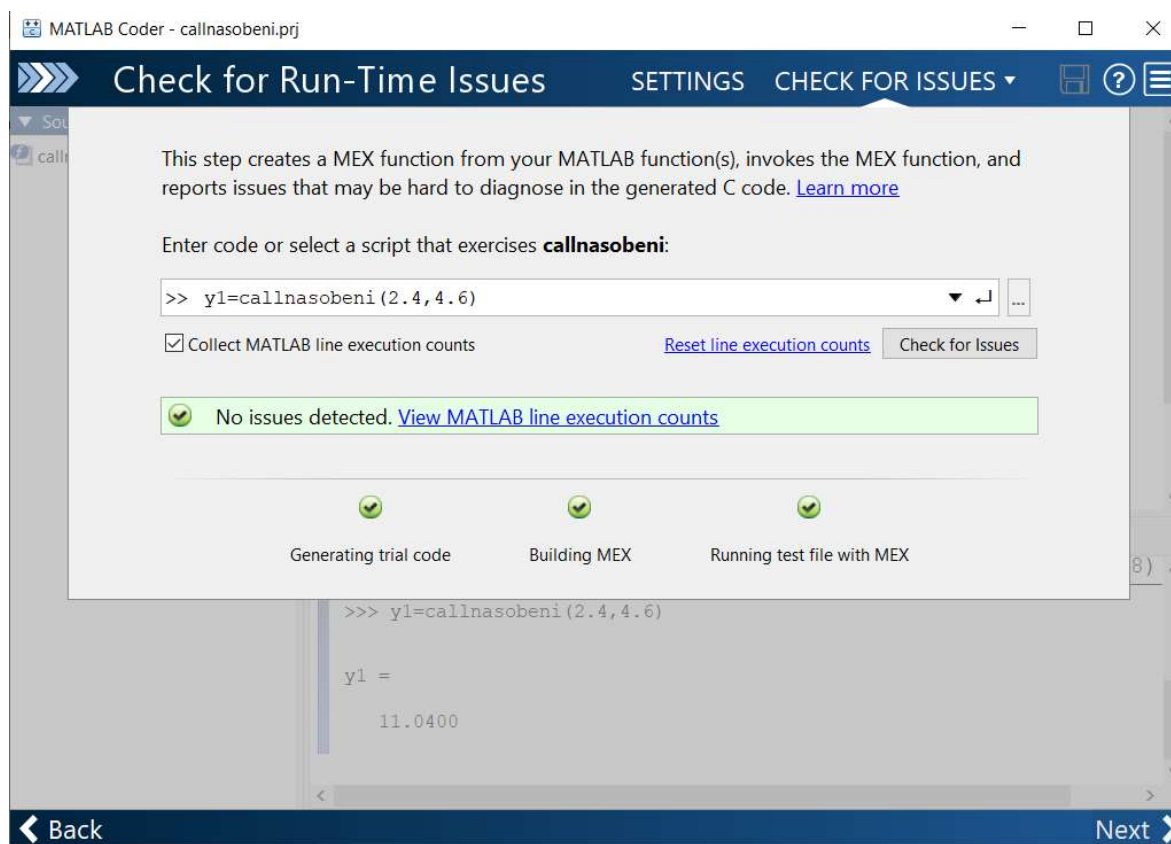
x1	double(1 x 1)
x2	double(1 x 1)

Add global

Back Next

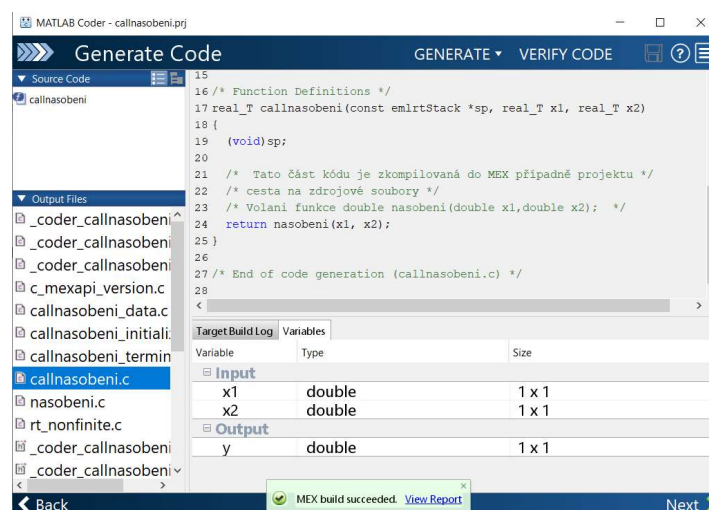
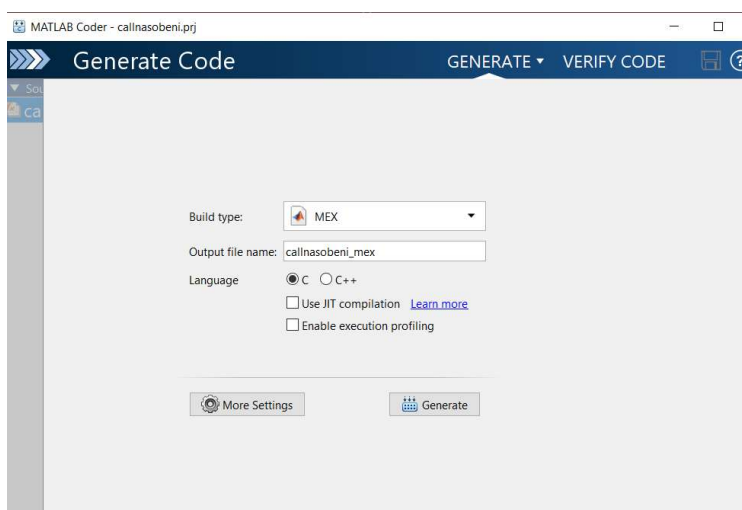
Tvorba MEX z vlastního C pomocí Matlab coder

4) Provedeme kontrolu kódu – dojde k porovnání výsledku volané funkce z matlabu a vygenerovaného a zkompilevaného mex souboru



Tvorba MEX z vlastního C pomocí Matlab coder

5) Vygenerujeme mex soubor (znova) a můžeme funkci volat



Vygenerovaný mex

Name	Size	Date modified
codegen		25.04.2021 22:10
~\$MRP2_prednaska07.p...		25.04.2021 10:34
callnasobeni.m		25.04.2021 22:35
callnasobeni.prj		25.04.2021 22:52
callnasobeni_mex.mexw64		25.04.2021 22:52
coder_gs.pdf		25.04.2021 14:25
coder_ug.pdf		25.04.2021 14:25

```

NEW TO MATLAB? SEE RESOURCES FOR Getting Started.

>> y1=callnasobeni_mex(6,10)

y1 =

    60

fx >> |
    
```

Volání mex (vstupní) funkce

Regionální inovační centrum elektrotechniky
Fakulta elektrotechnická
Západočeská univerzita v Plzni

Příští přednáška

Automatické generování kódu ze Simulinku

Jakub Talla

Regionální inovační centrum elektrotechniky
Fakulta elektrotechnická
Západočeská univerzita v Plzni

Děkuji za pozornost!

Adresa: Univerzitní 26
306 14 Plzeň
Česká republika

Tel: +420 377 634 443
Fax: +420 377 634 402

Email: talic@kev.zcu.cz

www.rice.zcu.cz