

**ČESKÉ VYSOKÉ
UČENÍ TECHNICKÉ
V PRAZE**

FAKULTA STAVEBNÍ



HABILITAČNÍ PRÁCE

**PŘÍLOHA 4: METODY HYBRIDNÍHO
MODELOVÁNÍ**

2026

ING. VJAČESLAV USMANOV, PH.D.

OBSAH

Hybridní model 01: Sjednocení modelů

Hybridní model 02: Validace modelu

Hybrid Model 01: Sjednocení modelů (Models integration)

```
In [1]: # Instalace potřebných knihoven
#%pip install pandas
#%pip install numpy
#%pip install seaborn matplotlib
```

```
In [2]: # Import potřebných knihoven
import pandas as pd
import numpy as np

import seaborn as sns
import matplotlib.pyplot as plt
```

Vstupní data

```
In [3]: ### Deterministická data

# Soubor je načten a přiřazen do proměnné ,df_cycl‘
other_path = '../..../data/02_DetermModel/model_data.csv'
df_cycl = pd.read_csv(other_path, header=0)
df_cycl
```

Out[3]:

	x	y	z	dist	time
0	74	459	1284	4840	39.019
1	-90	4908	2217	4571	38.212
2	522	3074	26	2718	32.653
3	425	2247	2739	4318	37.453
4	516	3425	1590	3308	34.423
...
19995	3122	589	704	7604	47.311
19996	1693	319	1561	6726	44.677
19997	2078	-4	46	7198	46.093
19998	3151	317	2282	8632	50.395
19999	1015	49	673	6034	42.601

20000 rows × 5 columns

Generování časové trajektorie deterministické doby cyklu

```
In [4]: # Zaokrouhlení hodnot, převod na int a vytvoření čísla cyklu z indexu
df_cycl['time'] = df_cycl['time'].round().astype(int)
df_cycl['dist'] = df_cycl['dist'].round().astype(int)
df_cycl['cycle'] = df_cycl.index + 1
df_cycl
```

Out[4]:

	x	y	z	dist	time	cycle
0	74	459	1284	4840	39	1
1	-90	4908	2217	4571	38	2
2	522	3074	26	2718	33	3
3	425	2247	2739	4318	37	4
4	516	3425	1590	3308	34	5
...
19995	3122	589	704	7604	47	19996
19996	1693	319	1561	6726	45	19997
19997	2078	-4	46	7198	46	19998
19998	3151	317	2282	8632	50	19999
19999	1015	49	673	6034	43	20000

20000 rows × 6 columns

In [5]:

```
# Rozbalení řádků s vytvořením průběžného času
df_cycl_time = (
    df_cycl.loc[df_cycl.index.repeat(df_cycl['time'])]
        .assign(time=lambda x: range(1, len(x) + 1))
        .reset_index(drop=True)
    [[ 'time', 'cycle' ]])
)

df_cycl_time
```

Out[5]:

	time	cycle
0	1	1
1	2	1
2	3	1
3	4	1
4	5	1
...
841483	841484	20000
841484	841485	20000
841485	841486	20000
841486	841487	20000
841487	841488	20000

841488 rows × 2 columns

Generování časové trajektorie stochastických vlivu

In [6]:

```
### Stochastická data
```

```
# Soubor je načten a přiřazen do proměnné ,df_state'
other_path = '../..../data/03_StochModel/simulation_MCMC_samples.csv'
df_state = pd.read_csv(other_path, header=0)
df_state
```

Out[6]:

	time	state_index
0	0	7
1	1	7
2	2	7
3	3	8
4	4	8
...
999995	999995	6
999996	999996	6
999997	999997	6
999998	999998	7
999999	999999	7

1000000 rows × 2 columns

In [7]:

```
# nalezení indexu prvního výskytu stavu S1
idx_start = df_state[df_state['state_index'] == 1].index[0]

# odstranění všech řádků před tímto indexem
df_state = df_state.loc[idx_start: ].reset_index(drop=True)

df_state
```

Out[7]:

	time	state_index
0	9	1
1	10	1
2	11	2
3	12	2
4	13	3
...
999986	999995	6
999987	999996	6
999988	999997	6
999989	999998	7
999990	999999	7

999991 rows × 2 columns

Definice neprovozních stavů

```
In [8]: # definice neprovozních stavů systému
downtime_states = [9, 10, 11, 12]
```

Mapa zpoždění pro jednotlivé stavы

```
In [ ]: # doba trvání neprovozních stavů [s]
state_delay_map = {
    9: 60, # manuální kontrola a servis
    10: 70, # výměna defektního materiálu
    11: 160, # oprava systému
    12: 20, # otáčení zdícího prvků
}
```

Výpočet stochastického zpoždění

```
In [10]: # přiřazení zpoždění podle aktuálního stavu
df_state['stochastic_delay'] = df_state['state_index'].map(state_delay_map)

# provozní stavы mají nulové zpoždění
df_state['stochastic_delay'] = df_state['stochastic_delay'].fillna(0)
df_state['stochastic_delay'] = df_state['stochastic_delay'].round().astype(int)
df_state = df_state[df_state['stochastic_delay'] != 0]
df_state
```

Out[10]:

	time	state_index	stochastic_delay
575	584	10	70
576	585	10	70
577	586	10	70
578	587	10	70
579	588	10	70
...
999652	999661	12	20
999653	999662	12	20
999654	999663	12	20
999655	999664	12	20
999656	999665	12	20

135343 rows × 3 columns

```
In [11]: # začátek výskytu zpoždění
df_state = df_state[df_state['state_index'].ne(df_state['state_index'].shift())]
df_state
```

Out[11]:

	time	state_index	stochastic_delay
575	584	10	70
958	967	12	20
3180	3189	10	70
3492	3501	12	20
5587	5596	9	60
...
996325	996334	10	70
997773	997782	12	20
999026	999035	10	70
999156	999165	9	60
999638	999647	12	20

1702 rows × 3 columns

Sjednocení fyzikálního a stochastického modelu

```
In [12]: # Přiřazení stochastic_delay ke každému cycle podle shodného času
df_delay_cycle = (
    df_cycl_time
    .merge(df_state[['time', 'stochastic_delay']], on='time', how='left')
    .dropna(subset=['stochastic_delay'])
    .drop_duplicates('cycle')
    [['cycle', 'stochastic_delay']])
)

df_cycl = df_cycl.merge(df_delay_cycle, on='cycle', how='left')

df_cycl
```

```
Out[12]:
```

	x	y	z	dist	time	cycle	stochastic_delay
0	74	459	1284	4840	39	1	NaN
1	-90	4908	2217	4571	38	2	NaN
2	522	3074	26	2718	33	3	NaN
3	425	2247	2739	4318	37	4	NaN
4	516	3425	1590	3308	34	5	NaN
...
19995	3122	589	704	7604	47	19996	NaN
19996	1693	319	1561	6726	45	19997	NaN
19997	2078	-4	46	7198	46	19998	NaN
19998	3151	317	2282	8632	50	19999	NaN
19999	1015	49	673	6034	43	20000	NaN

20000 rows × 7 columns

```
In [13]: # Nahrazení NaN ve stochastic_delay nulou a převod na int
df_cycl['stochastic_delay'] = df_cycl['stochastic_delay'].fillna(0).astype(int)
df_cycl
```

Out[13]:

	x	y	z	dist	time	cycle	stochastic_delay
0	74	459	1284	4840	39	1	0
1	-90	4908	2217	4571	38	2	0
2	522	3074	26	2718	33	3	0
3	425	2247	2739	4318	37	4	0
4	516	3425	1590	3308	34	5	0
...
19995	3122	589	704	7604	47	19996	0
19996	1693	319	1561	6726	45	19997	0
19997	2078	-4	46	7198	46	19998	0
19998	3151	317	2282	8632	50	19999	0
19999	1015	49	673	6034	43	20000	0

20000 rows × 7 columns

In [14]:

```
# Výpočet nového total_time jako součet původního času a stochastic_delay
df_cycl['total_time'] = df_cycl['time'] + df_cycl['stochastic_delay']

# Výběr relevantních sloupců
df_cycl = df_cycl[['x','y','z','dist','time','stochastic_delay', 'total_time']]
df_cycl
```

Out[14]:

	x	y	z	dist	time	stochastic_delay	total_time
0	74	459	1284	4840	39	0	39
1	-90	4908	2217	4571	38	0	38
2	522	3074	26	2718	33	0	33
3	425	2247	2739	4318	37	0	37
4	516	3425	1590	3308	34	0	34
...
19995	3122	589	704	7604	47	0	47
19996	1693	319	1561	6726	45	0	45
19997	2078	-4	46	7198	46	0	46
19998	3151	317	2282	8632	50	0	50
19999	1015	49	673	6034	43	0	43

20000 rows × 7 columns

In [15]:

```
# Výběr řádků, kde je stochastic_delay větší než 0
df_cycl[df_cycl['stochastic_delay'] > 0]
```

Out[15]:

	x	y	z	dist	time	stochastic_delay	total_time
16	28	-81	46	5225	40	70	110
26	591	316	254	5351	41	20	61
87	70	4919	886	4090	37	70	107
96	67	1977	360	3155	34	20	54
152	-99	367	143	4629	38	60	98
...
19882	2069	599	1336	6716	45	20	65
19885	4060	587	855	8568	50	70	120
19929	1487	420	1399	6341	44	20	64
19932	934	-16	1057	6095	43	160	203
19951	3238	4	17	8358	50	20	70

1417 rows × 7 columns

In [16]:

```
# celková mimopracovní doba, sek
print(f'Celkový mimopracovní běh systému: {df_cycl["stochastic_delay"].sum()} sek')
print(f'Celkový běh systému: {df_cycl["total_time"].sum()} sek')
```

Celkový mimopracovní běh systému: 88240 sek

Celkový běh systému: 929728 sek

Graf: Podíl zpoždění podle času

In [17]:

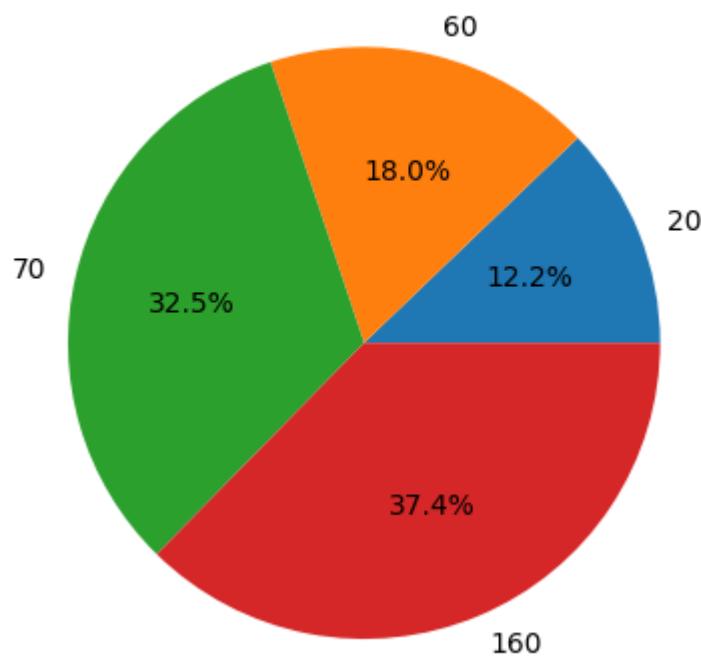
```
# výběr cyklů se zpožděním
df_delay_sum = (
    df_cycl[df_cycl['stochastic_delay'] > 0]
    .groupby('stochastic_delay')['stochastic_delay']
    .sum()
)

plt.figure()

plt.pie(
    df_delay_sum,
    labels=df_delay_sum.index,
    autopct='%1.1f%'
)

plt.title("Podíl zpoždění podle času")
plt.show()
```

Podíl zpoždění podle času

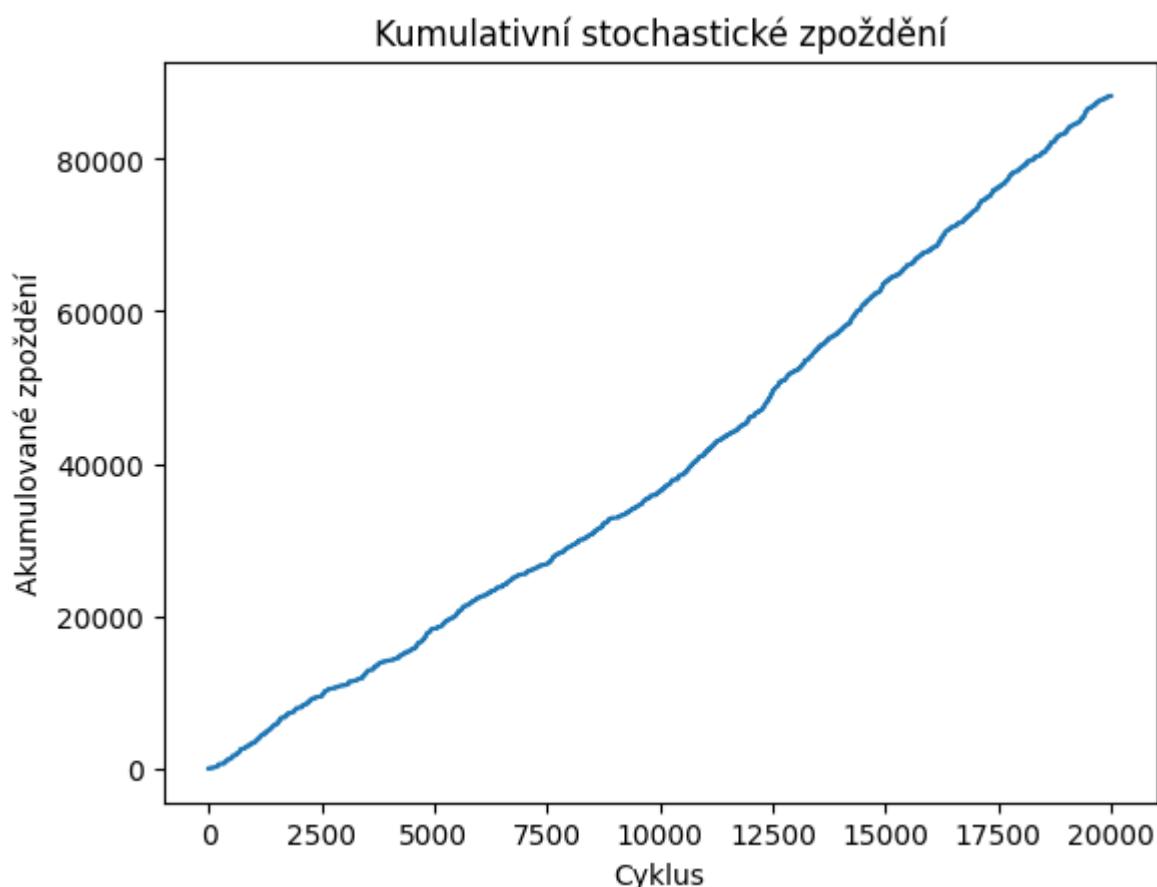


Graf: Kumulativní stochastické zpoždění

```
In [18]: df_cycl = df_cycl.copy()
df_cycl.loc[:, 'cum_delay'] = df_cycl['stochastic_delay'].cumsum()

plt.figure()
plt.plot(df_cycl['cum_delay'])

plt.title("Kumulativní stochastické zpoždění")
plt.xlabel("Cyklus")
plt.ylabel("Akumulované zpoždění")
plt.show()
```



Export datové sady do formátu CSV

```
In [19]: df_cycl.to_csv('..../data/04_HybridModel/hybrid_model.csv', index=False)
```

Autor / Organizace / Datum

Vjačeslav Usmanov, ČVUT v Praze, Fakulta stavební

Přehled změn

Datum (YYYY-MM-DD)	Verze	Autor změny	Popis změny
2026-01-27	1.1	Vjačeslav Usmanov	added HM_01_Models_Integration.ipynb
2026-02-16	1.2	Vjačeslav Usmanov	changed HM_01_Models_Integration.ipynb

Hybrid Model 02: Validace modelu (Model Validation)

```
In [1]: # Instalace potřebných knihoven  
#%pip install pandas  
#%pip install numpy
```

```
In [2]: # Import potřebných knihoven  
import pandas as pd  
import numpy as np  
  
from scipy import stats  
from scipy.stats import ks_2samp, ttest_ind, mannwhitneyu  
  
import pandas as pd  
import matplotlib.pyplot as plt
```

Načtení reálných a simulačních dat

```
In [3]: # Soubor je načten a přiřazen do proměnné ,df_sim"  
other_path = '../data/04_HybridModel/hybrid_model.csv'  
df_sim = pd.read_csv(other_path, header=0)
```

```
In [4]: # Zobrazení prvních 5 řádků datasetu  
print('Prvních 5 řádků datového rámce')  
df_sim.head(5)
```

Prvních 5 řádků datového rámce

```
Out[4]:   x      y      z    dist  time  stochastic_delay  total_time  cum_delay  
0   74    459   1284    4840     39           0         39          0  
1   -90   4908   2217    4571     38           0         38          0  
2    522   3074     26    2718     33           0         33          0  
3    425   2247   2739    4318     37           0         37          0  
4    516   3425   1590    3308     34           0         34          0
```

In [5]: # Základní deskriptivní statistika simulovaného datasetu
df_sim.describe()

Out[5]:

	x	y	z	dist	time	stochastic_delay	total_
count	20000.000000	20000.000000	20000.000000	20000.000000	20000.000000	20000.000000	20000.00
mean	1422.944700	1336.856500	1374.817600	5858.04950	42.074400	4.412000	46.48
std	1440.916105	1512.725045	805.317921	2172.83209	6.525181	20.073475	21.40
min	-99.000000	-99.000000	0.000000	1984.00000	30.000000	0.000000	30.00
25%	248.000000	202.000000	676.000000	3918.00000	36.000000	0.000000	36.00
50%	599.500000	521.000000	1374.500000	5705.00000	42.000000	0.000000	42.00
75%	2591.250000	2435.000000	2076.000000	7737.25000	48.000000	0.000000	49.00
max	4599.000000	4949.000000	2749.000000	10789.00000	57.000000	160.000000	215.00

In [6]: # Soubor je načten a přiřazen do proměnné ,df_real"
other_path = '../data/02_DetermModel/model_data_real.csv'
df_real = pd.read_csv(other_path, header=0)

In [7]: # Zobrazení prvních 5 řádků datasetu
print('Prvních 5 řádků datového rámce')
df_real.head(5)

Prvních 5 řádků datového rámce

Out[7]:

	id	x	y	z	time	delay	type_delay	total_time	dist	time_calc
0	150	1315	220	1000	29	0	0	29	3443	34.828
1	75	220	1190	500	33	0	0	33	3590	35.269
2	239	220	940	2000	35	6	3	41	4387	37.660
3	199	1315	220	1500	36	0	0	36	3636	35.407
4	51	3690	220	250	50	0	0	50	5767	41.800

In [8]: # Základní deskriptivní statistika simulovaného datasetu
df_real.describe()

Out[8]:

	id	x	y	z	time	delay	type_delay	total_
count	161.000000	161.000000	161.000000	161.000000	161.000000	161.000000	161.000000	161.000000
mean	139.049689	1314.428571	1091.807453	993.788820	36.285714	5.708075	0.254658	41.993
std	79.374177	1288.402896	1361.839636	702.645843	6.771658	25.355384	0.800621	27.037
min	2.000000	95.000000	95.000000	0.000000	24.000000	0.000000	0.000000	24.000000
25%	71.000000	220.000000	220.000000	500.000000	32.000000	0.000000	0.000000	32.000000
50%	141.000000	690.000000	220.000000	1000.000000	35.000000	0.000000	0.000000	36.000000
75%	203.000000	2440.000000	1565.000000	1500.000000	39.000000	0.000000	0.000000	42.000000
max	277.000000	4002.000000	4690.000000	2250.000000	56.000000	200.000000	4.000000	250.000000

Monte Carlo Validation: Opakování podvzorkování na velikost reality

```
In [9]: # Bootstrap vzorkování ze simulace na velikost reálného datasetu
n_real = len(df_real)

sim_samples = []

for _ in range(1000):
    sample = df_sim.sample(n=n_real, replace=True, random_state=122 + _)
    sim_samples.append(sample['total_time'])
```

KS test pro každé podvzorkování

```
In [10]: # Výpočet KS p-hodnot pro porovnání reálných a simulovaných dat
p_vals = []

for s in sim_samples:
    _, p = ks_2samp(df_real['total_time'], s)
    p_vals.append(p)
```

Pravděpodobnost shody modelu

```
In [11]: valid_ratio = np.mean(np.array(p_vals) > 0.05)

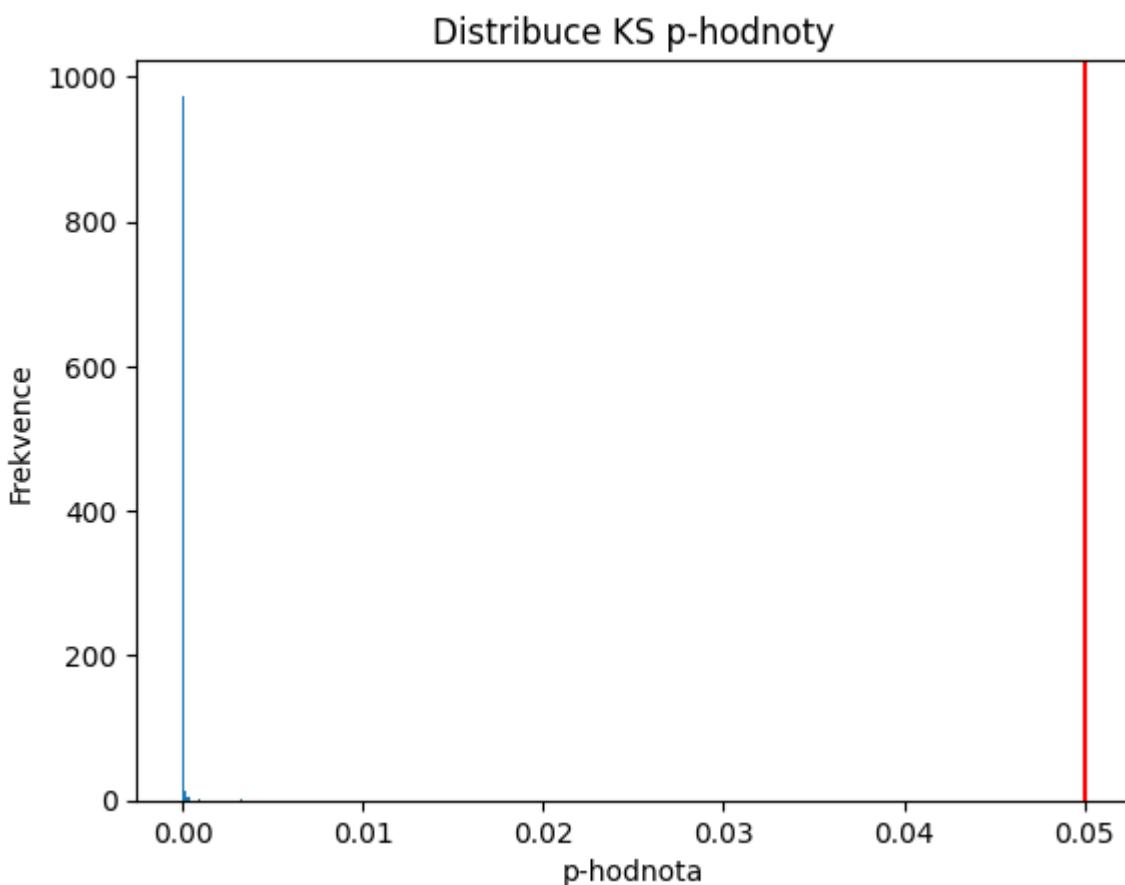
print("Podíl validních simulací:", valid_ratio)
```

Podíl validních simulací: 0.0

Distribuce KS p-hodnoty

```
In [12]: plt.figure()
plt.hist(p_vals, bins=30)
plt.axvline(0.05, color='red')

plt.title("Distribuce KS p-hodnoty")
plt.xlabel("p-hodnota")
plt.ylabel("Frekvence")
plt.show()
```



Interpretace (DES validace)

| Podíl | Interpretace | | ----- | ----- | 0.8 | model VALIDNÍ | | 0.5 – 0.8 | model přijatelný | | < 0.5 | model nevalidní |

Parametry nejsou kalibrované.

Porovnání průměru a směrodatné odchylky

Reálná data – referenční hodnoty:

```
In [13]: # Výpočet průměru a směrodatné odchylky z reálných dat
mean_real = df_real['total_time'].mean()
std_real = df_real['total_time'].std()

print("Real Mean:", mean_real)
print("Real STD:", std_real)
```

Real Mean: 41.993788819875775

Real STD: 27.037357695975476

Bootstrap ze simulace (na velikost reality):

```
In [14]: # velikost reálného datasetu
n_real = len(df_real)

sim_means = []
sim_stds = []

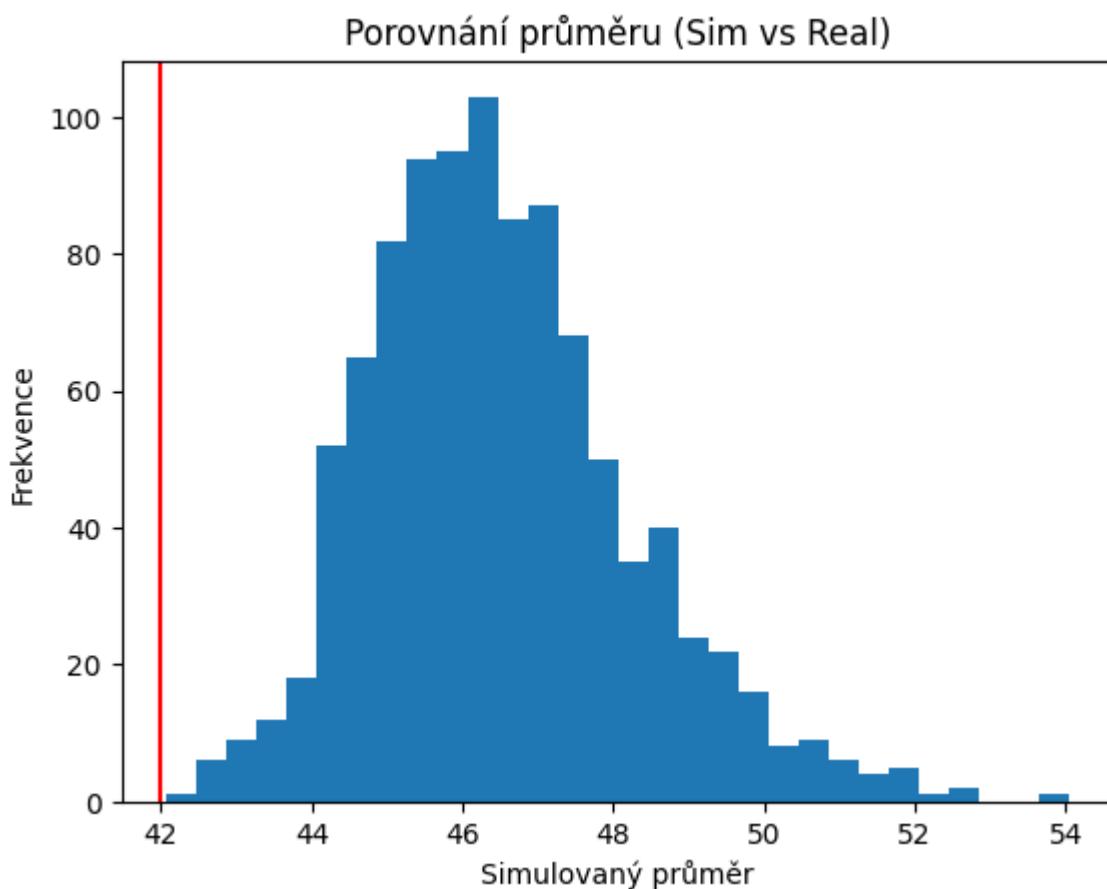
# opakování podvzorkování simulace
for i in range(1000):

    sample = df_sim.sample(
        n=n_real,
```

```
        replace=True,  
        random_state=122 + i  
  
    )  
  
    sim_means.append(sample['total_time'].mean())  
    sim_stds.append(sample['total_time'].std())
```

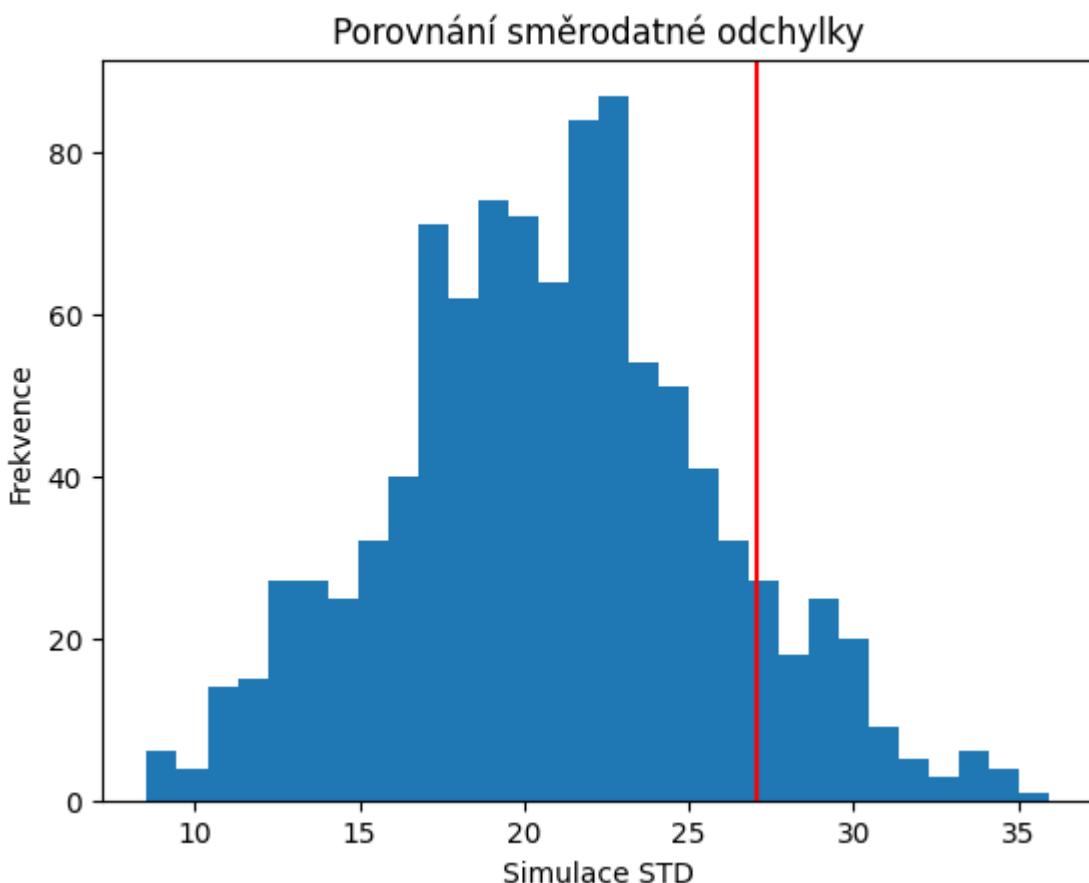
Porovnání průměru

```
In [15]: # Histogram průměrů ze simulace s vyznačením průměru reálných dat  
plt.figure()  
  
plt.hist(sim_means, bins=30)  
plt.axvline(mean_real, color='red')  
  
plt.title("Porovnání průměru (Sim vs Real)")  
plt.xlabel("Simulovaný průměr")  
plt.ylabel("Frekvence")  
  
plt.show()
```



Porovnání STD

```
In [16]: # Histogram STD ze simulace s vyznačením průměru reálných dat  
plt.figure()  
  
plt.hist(sim_stds, bins=30)  
plt.axvline(std_real, color='red')  
  
plt.title("Porovnání směrodatné odchylky")  
plt.xlabel("Simulace STD")  
plt.ylabel("Frekvence")  
  
plt.show()
```



Kvantilová validace

```
In [17]: mean_CI = np.percentile(sim_means, [2.5, 97.5])
std_CI = np.percentile(sim_stds, [2.5, 97.5])

print("Mean 95% CI:", mean_CI)
print("STD 95% CI:", std_CI)

print("Real mean:", mean_real)
print("Real STD:", std_real)
```

Mean 95% CI: [43.56490683 50.54161491]
 STD 95% CI: [11.39980265 30.74196232]
 Real mean: 41.993788819875775
 Real STD: 27.037357695975476

Model je validní, ale rozsah hodnot je moc široký -> **simulace není kalibrována**.

Parametrické porovnání dat

Welchův t-test

```
In [18]: stat, p = ttest_ind(
    df_real['total_time'],
    df_sim['total_time'],
    equal_var=False
)

print("Welch t-test p-value:", p)
```

Welch t-test p-value: 0.0370080094281292

p < 0.05 → simulace má jiný průměr.

Cohen's d (velikost efektu)

```
In [19]: mean_diff = abs(df_real['total_time'].mean() - df_sim['total_time'].mean())
pooled_std = np.sqrt(
    (df_real['total_time'].std()**2 + df_sim['total_time'].std()**2) / 2
)
d = mean_diff / pooled_std
print("Cohen's d:", d)
```

Cohen's d: 0.18425210268660908

Cohen's d < 0.2 → zanedbatelný význam

Neparametrické porovnání (Distribuce)

Mann–Whitney U test

```
In [20]: stat, p = mannwhitneyu(
    df_real['total_time'],
    df_sim['total_time']
)
print("Mann-Whitney p-value:", p)
```

Mann-Whitney p-value: 3.3892693033431795e-16

Vyhodnocení shody simulovaných a reálných dat

Bootstrap KS test

0,00

Interpretace:

Ve 0.00 % případů nelze statisticky rozlišit simulaci od reality

STD

Real STD = 27.037357695975476 STD 95% CI (Sim) [11.39980265 30.74196232]

Interpretace:

Reálná směrodatná odchylka je v intervalu, avšak simulace vykazuje nadměrnou variabilitu

Průměr

Real Mean = 41.99 Mean 95% CI (Sim) [43.56490683 50.54161491]

Interpretace:

Reálný průměr se nachází v intervalu simulace

Welch t-test

p = 0.0370080094281292

Interpretace:

Simulace má jiný průměr.

Cohen's d

0.18425210268660908

Interpretace:

Malý rozdíl mezi průměry reálných a simulovaných dat

Mann–Whitney U test (neparametrický)

p = 0.000

Interpretace:

Zamítnutí nulové hypotézy.

Simulační model vykazuje systematickou odchylku od reálných dat a jeho kalibrace je nutná.

Autor / Organizace / Datum

Vjačeslav Usmanov, ČVUT v Praze, Fakulta stavební

Přehled změn

Datum (YYYY-MM-DD)	Verze	Autor změny	Popis změny
2026-01-27	1.1	Vjačeslav Usmanov	added HM_02_Model_Validation.ipynb
2026-02-16	1.2	Vjačeslav Usmanov	changed HM_02_Model_Validation.ipynb