ČESKÉ VYSOKÉ
UČENÍ TECHNICKÉ
V PRAZE

FAKULTA STAVEBNÍ

HABILITAČNÍ PRÁCE

PŘÍLOHA 5: METODY KALIBRACE MODELŮ

2026

ING. VJAČESLAV USMANOV, PH.D.

OBSAH

Calibration Model 01: Bayesovská kalibrace (Bayes Calibration)

Počet kalibrovaných parametrů: 3

Kalibrační framework

- Zarovnat časové osy
- Definovat dynamický model
- Zavést prior na parametry
- Zavést prior na šum
- Přidat modelovou diskrepanci
- MCMC sampling
- Posterior predictive kontrola

In [1]:
```
# Instalace potřebných knihoven
#%pip install pandas
#%pip install numpy
#%pip install seaborn matplotlib
#%pip install pymc
#%pip install arviz
#%pip install ipywidgets
#%pip install jupyterlab_widgets
#%pip install pytensor
#%pip install ipywidgets jupyterlab_widgets
```

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

import pymc as pm
import arviz as az
import pytensor.tensor as pt

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

Vstupní data

In [3]:
```
### Načtení časosběru

# Soubor je načten a přiřazen do proměnné ‚df‘
other_path = '../../data/02_DetermModel/model_data_real.csv'
df = pd.read_csv(other_path, header=0)
df = df[['id', 'x', 'y', 'z', 'dist', 'time', 'total_time']]
df
```

Out[3]:

	id	x	y	z	dist	time	total_time
0	150	1315	220	1000	3443	29	29
1	75	220	1190	500	3590	33	33
2	239	220	940	2000	4387	35	41
3	199	1315	220	1500	3636	36	36
4	51	3690	220	250	5767	50	50
...	...	...	...	...	...	...	...
156	83	220	3190	500	1970	29	29
157	26	1815	220	0	3943	45	45
158	190	220	690	1500	4351	33	33
159	234	220	3815	1750	2990	27	27
160	195	220	3940	1500	2981	35	35

161 rows × 7 columns

Vhodné parametry pro kalibraci

Parametr	Kalibrovat	Důvod
speed_max_load	Ano	reálná rychlost ≠ nominální
speed_max_unload	Ano	často vyšší variabilita
accel	Ano	výrazně ovlivňuje krátké cykly
pevné časy	Ne (slabý prior)	většinou měřené přesně

In [4]:
```
dist = df["dist"].values
time_real = df["time"].values
```

Fyzikální model (vektorový)

In [5]:
```
def move_time(dist, v, a):

    t_acc = v / a
    d_acc = 0.5 * a * t_acc**2
    d_crit = 2 * d_acc

    # trojúhelníkový profil
    triangular = 2 * pt.sqrt(dist / a)

    # trapezoidální profil
    trapezoidal = 2 * t_acc + (dist - d_crit) / v

    return pt.switch(dist < d_crit,
                     triangular,
                     trapezoidal)
```

Kompletní model

In [6]:
```
with pm.Model() as model:

    # ------------------------
    # Priory (fyzikálně omezené)
    # ------------------------

    v_load = pm.TruncatedNormal("v_load",
                                mu=0.5,
                                sigma=0.2,
                                lower=0.1)

    v_unload = pm.TruncatedNormal("v_unload",
                                  mu=1.0,
                                  sigma=0.3,
                                  lower=0.2)

    accel = pm.TruncatedNormal("accel",
                               mu=1.0,
                               sigma=0.3,
                               lower=0.2)

    sigma = pm.HalfNormal("sigma", 0.2)

    # ------------------------
    # Fixní část cyklu
    # ------------------------

    time_refer_2_refer = 20   # s, průměrná doba pohybu z referenčního bodu k referenčnímu boc
    time_mounting = 3         # s, doba manipulaci v cílové poloze (umístění prvku)

    T_fix = time_refer_2_refer + time_mounting
    # ------------------------
    # Pohyb
    # ------------------------

    T_load = move_time(dist/1000, v_load, accel)
    T_unload = move_time(dist/1000, v_unload, accel)

    mu = T_fix + T_load + T_unload

    # ------------------------
    # Likelihood
    # ------------------------

    y = pm.Normal("y",
                  mu=mu,
                  sigma=sigma,
                  observed=time_real)

    trace = pm.sample(3000,
                      tune=2000,
                      target_accept=0.95)
```

```
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [v_load, v_unload, accel, sigma]
```

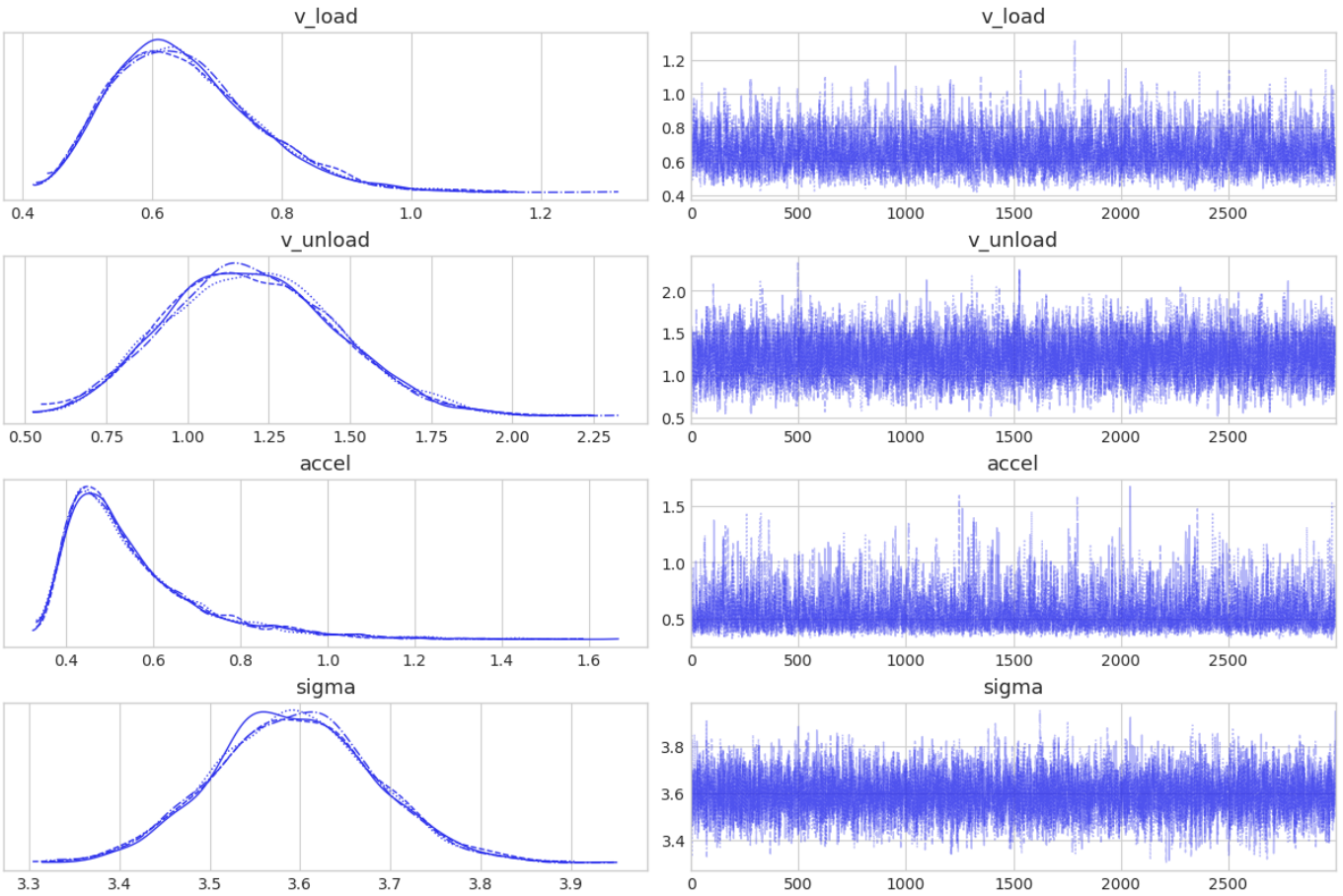
Output()

```
Sampling 4 chains for 2_000 tune and 3_000 draw iterations (8_000 + 12_000 draws total) took 1
1 seconds.
```

Diagnostika

In [7]:
```
az.summary(trace)

az.style.use("arviz-whitegrid")

plt.rcParams.update({
    "figure.figsize": (12, 8),
    "font.size": 12,
    "axes.titlesize": 13,
    "axes.labelsize": 11,
    "legend.fontsize": 10,
    "xtick.labelsize": 10,
    "ytick.labelsize": 10,
    "lines.linewidth": 1.0,
})

az.plot_trace(trace)
```

Out[7]:
```
array([[<Axes: title={'center': 'v_load'}>,
        <Axes: title={'center': 'v_load'}>],
       [<Axes: title={'center': 'v_unload'}>,
        <Axes: title={'center': 'v_unload'}>],
       [<Axes: title={'center': 'accel'}>,
        <Axes: title={'center': 'accel'}>],
       [<Axes: title={'center': 'sigma'}>,
        <Axes: title={'center': 'sigma'}>]], dtype=object)
```



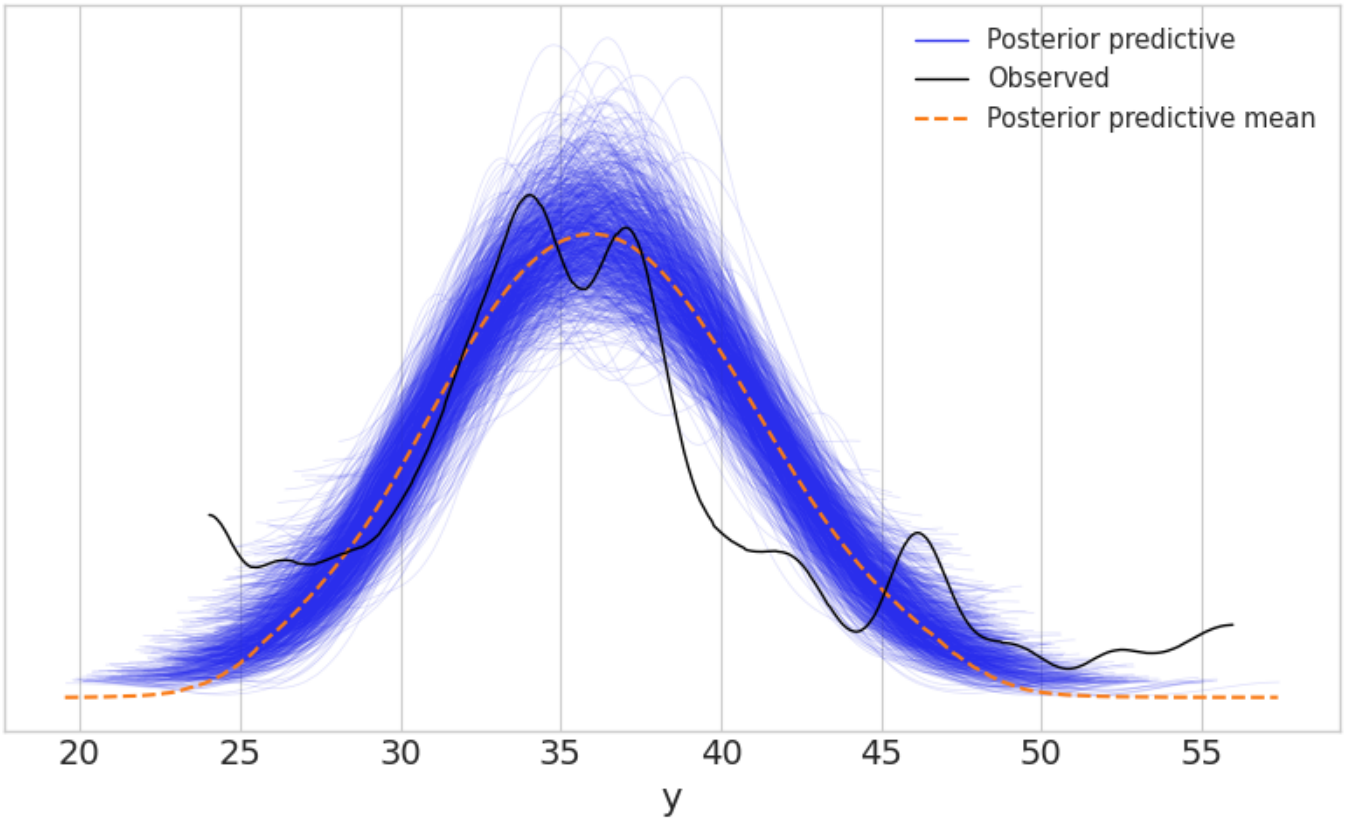
Posterior predictive kontrola

In [8]:
```
with model:
    trace.extend(
        pm.sample_posterior_predictive(trace)
    )
```

```
Sampling: [y]
Output()
```

In [9]:
```
az.style.use("arviz-whitegrid")

fig, ax = plt.subplots(figsize=(8,5))

az.plot_ppc(
    trace,
    num_pp_samples=1000,     # don't plot all draws!
    kind="kde",
    mean=True,
    observed=True,
    alpha=0.15,
    ax=ax
)

plt.tight_layout()
```

```
C:\Users\usman\AppData\Local\Temp\ipykernel_26088\3615469113.py:15: UserWarning: The figure la
yout has changed to tight
  plt.tight_layout()
```



Souhrnná statistika posteriorní prediktivní distribuce modelu

In [10]:
```
summary_df = az.summary(trace, group="posterior_predictive")
summary_df
```

Out[10]:

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
y[0]	35.076	3.648	27.950	41.694	0.034	0.024	11780.0	11754.0	1.0
y[1]	35.395	3.623	28.546	42.153	0.033	0.023	11754.0	11610.0	1.0
y[2]	37.378	3.629	30.475	44.050	0.033	0.024	11829.0	11373.0	1.0
y[3]	35.472	3.642	28.580	42.283	0.033	0.024	11944.0	11685.0	1.0
y[4]	40.741	3.655	34.193	47.982	0.033	0.024	12393.0	11902.0	1.0
...	...	...	...	...	...	...	...	...	...
y[156]	31.350	3.587	24.531	38.050	0.033	0.023	11538.0	11687.0	1.0
y[157]	36.211	3.589	29.611	43.083	0.033	0.023	11917.0	11931.0	1.0
y[158]	37.231	3.594	30.660	44.089	0.033	0.023	11562.0	11187.0	1.0
y[159]	33.884	3.649	27.332	40.911	0.033	0.024	12204.0	11572.0	1.0
y[160]	33.910	3.638	27.065	40.661	0.033	0.023	12163.0	11596.0	1.0

161 rows × 9 columns

Export datové sady do formátu netCDF a CSV

https://www.unidata.ucar.edu/software/netcdf

In [11]:
```
summary_df.to_csv("../../data/05_Calibration/posterior_predictive_summary_three.csv", index=Fa
```

In [12]:
```
az.to_netcdf(trace, "../../data/05_Calibration/posterior_trace_three.nc")
```

Out[12]:
```
'../../data/05_Calibration/posterior_trace_three.nc'
```

In [13]:
```
df_posterior = az.extract(trace, group="posterior").to_dataframe()
df_posterior.to_csv("../../data/05_Calibration/posterior_three.csv", index=False)
```

Autor / Organizace / Datum

Vjačeslav Usmanov, ČVUT v Praze, Fakulta stavební

Přehled změn

Datum (YYYY-MM-DD)	Verze	Autor změny	Popis změny
2026-01-31	1.1	Vjačeslav Usmanov	added CM_01_BayesCalibration.ipynb
2026-02-18	1.2	Vjačeslav Usmanov	changed CM_01_BayesCalibration.ipynb

Calibration Model 02: Generátor syntetických dat (Synthetic Data Generator)

Počet kalibrovaných parametrů: 3

In [1]:
```
# Instalace potřebných knihoven
#%pip install pandas
#%pip install numpy
#%pip install seaborn matplotlib
#%pip install pymc
#%pip install arviz
#%pip install ipywidgets
#%pip install jupyterlab_widgets
#%pip install pytensor
#%pip install ipywidgets jupyterlab_widgets
```

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

import pymc as pm
import arviz as az
import pytensor.tensor as pt

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

Vstupní data

In [3]:
```
### Načtení z formátu netCDF

# Soubor je načten a přiřazen do proměnné ‚trace‘
other_path = '../../data/05_Calibration/posterior_trace_three.nc'
trace = az.from_netcdf(other_path)
```

In [4]:
```
az.summary(trace)
```

Out[4]:

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
v_load	0.651	0.111	0.460	0.854	0.002	0.001	3958.0	5840.0	1.0
v_unload	1.198	0.257	0.705	1.658	0.003	0.002	5690.0	5023.0	1.0
accel	0.550	0.159	0.345	0.854	0.002	0.003	4749.0	5884.0	1.0
sigma	3.593	0.088	3.417	3.749	0.001	0.001	8339.0	7311.0	1.0

Definice a nastavení parametrů robotického systému

In [5]:
```
# SPECIFIKACE TECHNOLOGICKÉHO PROCESU ZDĚNÍ

time_refer_2_refer = 20   # s, průměrná doba pohybu z referenčního bodu k referenčnímu bodu (d
time_mounting = 3         # s, doba manipulaci v cílové poloze (umístění prvku)
brick_thickness = 440     # mm, tloušťka zdicího prvku (Porotherm 440 Profi)
brick_height = 250        # mm, výška zdicího prvku (Porotherm 440 Profi)
brick_width = 250         # mm, šířka zdicího prvku (Porotherm 440 Profi)
```

```
# SOUŘADNICE REFERENČNÍHO BODU (nad verifikačním stolem)

refer_x = 2_000           # mm, souřadnice X referenčního bodu
refer_y = 3_500           # mm, souřadnice Y referenčního bodu
refer_z = 1_000           # mm, souřadnice Z referenčního bodu
```

Definice funkce pro výpočet celkové doby pracovního cyklu

In [6]:
```
def simulate_time(dist, v_load, v_unload, accel):
    """
    Funkce pro výpočet celkové doby pracovního cyklu robotického zdění.

    Parametry:
    dist (int): dráha trajektorie od referenčního bodu k cílové poloze prvku [mm]

    Návratová hodnota:
    total_time (int): celková doba pracovního cyklu [s]
    """
    # výpočet času potřebného k dosažení maximální rychlosti
    time_to_load_speed = v_load / accel
    dist_to_load_speed = (1/2) * accel * time_to_load_speed  # uražená dráha při akceleraci

    # výpočet času potřebného k dosažení 0 rychlosti
    time_to_unload_speed = v_unload / accel
    dist_to_unload_speed = (1/2) * accel * time_to_unload_speed  # uražená dráha při deakceler

    # pevné technologické časy (manipulace a přesuny mezi pevnými body)
    total_time = time_refer_2_refer

    # manipulace v cílové poloze
    total_time += time_mounting

    # pohyb s naloženým prvkem (převod mm → m)
    total_time += (dist - dist_to_load_speed) / 1_000 / v_load

    # pohyb bez zátěže (zpětný pohyb)
    total_time += (dist - dist_to_unload_speed) / 1_000 / v_unload

    # započtení akceleračních časů
    total_time += time_to_load_speed + time_to_unload_speed
    return total_time
```

Vymezení pracovního rozsahu pro generování dat

In [7]:
```
# počet scénářů simulace
number_simulation = 10_000

# nastavení limitních hodnot rozsahu
dist_min = 2_000
dist_max = 10_000

# Vygeneruje 1000 náhodných hodnot z rovnoměrného rozdělení
dist_range = np.random.uniform(dist_min, dist_max, number_simulation)
```

Empirická bootstrap distribuce zdržení (stochastické vlivy)

In [8]:
```
### Načtení hybridního modelu

# Soubor je načten a přiřazen do proměnné ‚stochastic_delay‘
other_path = '../../data/04_HybridModel/hybrid_model.csv'
```

```
stochastic_delay = pd.read_csv(other_path, header=0)
```

In [9]:
```
delay_samples = stochastic_delay["stochastic_delay"].values
delay_samples
```

Out[9]:
```
array([0, 0, 0, ..., 0, 0, 0], shape=(20000,))
```

Generátor syntetických dat

In [10]:
```
# ======================================================
# 1 EXTRAKCE POSTERIOR VZORKŮ
# ======================================================

# Sloučení chain + draw do jedné dimenze
posterior = trace.posterior.stack(sample=("chain", "draw"))

# Extrakce parametrů
v_load_samples = posterior["v_load"].values
v_unload_samples = posterior["v_unload"].values
accel_samples = posterior["accel"].values
sigma_samples = posterior["sigma"].values

n_samples = len(v_load_samples)
```

In [11]:
```
# ======================================================
# 2 MONTE CARLO SIMULACE
# ======================================================

simulated = []

rng = np.random.default_rng(122)

for d in dist_range:

    T_samples_for_dist = []

    for i in range(n_samples):

        T_det = simulate_time(
            dist=d,
            v_load=v_load_samples[i],
            v_unload=v_unload_samples[i],
            accel=accel_samples[i]
        )
        # bootstrap náhodná realizace
        delay = rng.choice(delay_samples)

        # šum (noise) s normálním rozdělením
        #delay = np.random.normal(0, sigma_samples[i], size=np.shape(T_det))

        T_sim = T_det + delay

        T_samples_for_dist.append(T_sim)

    simulated.append(T_samples_for_dist)

simulated_data = np.array(simulated)
```

In [12]:
```
simulated_data.shape
```

Out[12]:
```
(10000, 12000)
```

Simulace obsahuje:

- 10 000 hodnot scénářů
- 12 000 Monte Carlo realizací pro každý scénář

Základní vyhodnocení (pro každý scenář)

In [13]:
```
# ====================================================
# 3 STATISTICKÉ VYHODNOCENÍ
# ====================================================

# Střední hodnota
T_mean = simulated_data.mean(axis=1)

# Směrodatná odchylka
T_std = simulated_data.std(axis=1)

# 95% interval spolehlivosti
T_lower = np.percentile(simulated_data, 2.5, axis=1)
T_upper = np.percentile(simulated_data, 97.5, axis=1)

# Medián
T_median = np.median(simulated_data, axis=1)
```

In [14]:
```
# ====================================================
# 4 VÝSTUPNÍ SHRNUTÍ
# ====================================================

print("Průměr:", T_mean)
print("95% interval:", T_lower, "-", T_upper)
print("Směrodatná odchylka:", T_std)
print("Medián:", T_median)
```

```
Průměr: [42.92518796 53.95936201 54.82303924 ... 37.44775095 37.30331261
 47.87585148]
95% interval: [37.76592548 47.04481144 47.73994339 ... 32.33923617 32.0517479
 42.06722063] - [108.24822245 119.01031363 119.54546478 ... 102.76242383 102.48601093
 113.00674861]
Směrodatná odchylka: [20.31263712 20.30124266 20.2458304  ... 20.00979547 19.63225683
 19.82160762]
Medián: [38.45984569 49.68085014 50.55506495 ... 33.18958129 32.9398544
 43.53660312]
```

Globální vyhodnocení (přes všechny scénáře)

In [15]:
```
global_distribution = simulated_data.flatten()

global_mean = global_distribution.mean()
global_std = global_distribution.std()

global_lower = np.percentile(global_distribution, 2.5)
global_upper = np.percentile(global_distribution, 97.5)

global_median = np.median(global_distribution)
```

In [16]:
```
print("Globální průměr:", global_mean)
print("Globální 95% interval:", global_lower, "-", global_upper)
print("Globální Směrodatná odchylka:", global_std)
print("Globální medián:", global_median)
```

```
Globální průměr: 45.78089707663459
Globální 95% interval: 32.02139839545803 - 108.69217282407138
Globální Směrodatná odchylka: 20.879563589788212
Globální medián: 42.00183691917726
```

In [17]:
```
results_df = pd.DataFrame({
    "dist": dist_range,
    "global_mean": global_mean,
    "global_CI_lower": global_lower,
    "global_CI_upper": global_upper,
    "global_std": global_std,
    "global_median": global_median,
    "T_mean": T_mean,
    "T_CI_lower": T_lower,
    "T_CI_upper": T_upper,
    "T_std": T_std,
    "T_median": T_median,

})
```

In [18]:
```
simulated_data
```

Out[18]:
```
array([[38.07152777, 38.23328284, 38.4415863 , ..., 38.62730757,
        38.67811025, 58.44591751],
       [48.61452465, 48.22831172, 48.62301511, ..., 48.407293  ,
        50.57451908, 49.26972128],
       [49.41340623, 48.98567175, 49.39449935, ..., 49.1483584 ,
        51.47595364, 50.08988062],
       ...,
       [33.09402733, 33.51448598, 33.63478735, ..., 34.0100358 ,
        33.06164458, 33.3358441 ],
       [32.8523095 , 33.28533133, 33.40135914, ..., 33.78581143,
        32.78889726, 33.08768825],
       [42.84534264, 42.75898061, 43.05168501, ..., 43.05563477,
        44.0647431 , 43.3468803 ]], shape=(10000, 12000))
```

Generování datasetu syntetických dat

In [19]:
```
n_dist = simulated_data.shape[0]
n_mc = simulated_data.shape[1]

# zopakujeme každé dist 12000x
dist_long = np.repeat(dist_range, n_mc)

# rozbalíme časy
total_time_long = simulated_data.flatten()

# vytvoříme dataframe
synthetic_df = pd.DataFrame({
    "dist": dist_long,
    "total_time": total_time_long
})
```

In [20]:
```
synthetic_df.shape
```

Out[20]:
```
(120000000, 2)
```

In [21]:
```
# náhodně podvzorkování
synthetic_df = synthetic_df.sample(500_000, random_state=122)
```

In [22]:
```
synthetic_df.head()
```

Out[22]:

	dist	total_time
52121957	5483.830858	40.567093
12437378	9013.410179	49.481206
92811520	5679.449597	40.936673
42394026	9643.054314	49.766864
67187642	8990.433929	49.055589

Export datové sady do formátu CSV

In [23]:
```
synthetic_df.to_csv("../../data/05_Calibration/synthetic_dataset.csv", index=False)
results_df.to_csv("../../data/05_Calibration/results_synthetic_dataset.csv", index=False)
```

Autor / Organizace / Datum

Vjačeslav Usmanov, ČVUT v Praze, Fakulta stavební

Přehled změn

Datum (YYYY-MM-DD)	Verze	Autor změny	Popis změny
2026-01-31	1.1	Vjačeslav Usmanov	added CM_02_SyntGenerator.ipynb
2026-02-18	1.2	Vjačeslav Usmanov	changed CM_02_SyntGenerator.ipynb

Calibration Model 03: 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 syntetických dat

In [3]:
```
# Soubor je načten a přiřazen do proměnné „df_syn"
other_path = '../../data/05_Calibration/synthetic_dataset.csv'
df_syn = pd.read_csv(other_path, header=0)
```

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

```
Prvních 5 řádků datového rámce
```

Out[4]:

	dist	total_time
0	5483.830858	40.567093
1	9013.410179	49.481206
2	5679.449597	40.936673
3	9643.054314	49.766864
4	8990.433929	49.055589

In [5]:
```
# Základní deskriptivní statistika syntetického datasetu
global_distribution = df_syn[['total_time']]
df_syn.describe()
```

Out[5]:

	dist	total_time
count	500000.000000	500000.000000
mean	6002.213262	45.777590
std	2314.176711	20.917843
min	2000.224407	29.722498
25%	4009.035893	36.821901
50%	5983.489163	41.987051
75%	8061.298159	47.360877
max	9999.961862	215.749756

Načtení souboru reálných dat z izolované sady měření

Syntetická data se validují **proti reálným datům, která nebyla použita při kalibraci modelu**.

In [6]:
```
# Soubor je načten a přiřazen do proměnné ,real_validation_df"
other_path = '../../data/06_AI/val/valid_timelaps.csv'
real_validation_df = pd.read_csv(other_path, header=0)
```

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

```
Prvních 5 řádků datového rámce
```

Out[7]:

	id	x	y	z	time	delay	type_delay	total_time
0	13	220	2940	0	32	0	0	32
1	77	220	1690	500	33	23	2	56
2	220	2190	220	1750	35	0	0	35
3	105	252	220	750	53	0	0	53
4	45	2190	220	250	45	0	0	45

In [8]:
```
# SPECIFIKACE TECHNOLOGICKÉHO PROCESU ZDĚNÍ

brick_thickness = 440      # mm, tloušťka zdicího prvku (Porotherm 440 Profi)
brick_height = 250         # mm, výška zdicího prvku (Porotherm 440 Profi)
brick_width = 250          # mm, šířka zdicího prvku (Porotherm 440 Profi)

# SOUŘADNICE REFERENČNÍHO BODU (nad verifikačním stolem)

refer_x = 2_000            # mm, souřadnice X referenčního bodu
refer_y = 3_500            # mm, souřadnice Y referenčního bodu
refer_z = 1_000            # mm, souřadnice Z referenčního bodu
```

In [9]:
```
def calculation_dist(x, y, z):
    """
    Funkce pro výpočet dráhy trajektorie od referenčního bodu k cílové poloze prvku

    Parametry:
    x, y, z (int): souřadnice cílové polohy prvku [mm]

    Návratová hodnota:
```

```
    dist (int): dráha trajektorie od referenčního bodu k cílové poloze prvku [mm]
    """
    dist = 0

    # Fáze 1: dráha od referenčního bodu k cílové stěně.
    dist = ((refer_z - (z + brick_height*2))**2 + (refer_x - brick_thickness//2)**2)**(1/2)

    # Fáze 2: dráha ve směru osy X
    if x != brick_thickness / 2:
        dist = dist + abs(x - brick_thickness//2 + brick_width * 2)

    # Fáze 3: dráha ve směru osy Y
    if y != brick_thickness / 2:
        dist = dist + abs(y - refer_y + brick_width * 2)

    return int(dist)
```

In [10]:
```
# Funkce pro výpočet dráhy trajektorie od referenčního bodu k cílové poloze prvku
# Pro každý řádek datového rámce je aplikována funkce calculation_dist
# na základě souřadnic 'x','y','z'.
real_validation_df['dist'] = real_validation_df.apply(lambda x : calculation_dist(x['x'],x['y'
```

In [11]:
```
# Základní deskriptivní statistika datasetu
real_validation_df.describe()
```

Out[11]:

	id	x	y	z	time	delay	type_delay	total_ti
count	108.000000	108.000000	108.000000	108.000000	108.000000	108.000000	108.000000	108.000
mean	141.120370	1273.351852	992.740741	1013.888889	36.472222	5.324074	0.231481	41.796
std	79.777578	1240.172134	1274.175272	718.443064	6.443802	27.813743	0.804270	28.458
min	1.000000	95.000000	95.000000	0.000000	22.000000	0.000000	0.000000	22.000
25%	77.750000	220.000000	220.000000	500.000000	32.000000	0.000000	0.000000	32.750
50%	136.500000	690.000000	220.000000	875.000000	36.000000	0.000000	0.000000	36.500
75%	212.250000	2221.250000	1690.000000	1750.000000	41.000000	0.000000	0.000000	42.000
max	276.000000	4002.000000	4565.000000	2250.000000	58.000000	260.000000	4.000000	296.000

Omezení syntetických dat

In [12]:
```
df_syn = df_syn[
    (df_syn["dist"] >= real_validation_df['dist'].min()) &
    (df_syn["dist"] <= real_validation_df['dist'].max())
]
```

In [13]:
```
df_syn.describe()
```

Out[13]:

	dist	total_time
count	281025.000000	281025.000000
mean	4247.312817	41.470066
std	1296.278484	20.363578
min	2000.224407	29.722498
25%	3138.655480	34.522064
50%	4247.387445	37.481842
75%	5387.460387	40.406839
max	6465.716208	204.340501

Coverage test (Kolik reálných bodů leží v 95% CI)

In [14]:
```
T_lower_interp = np.percentile(global_distribution, 2.5)
T_upper_interp = np.percentile(global_distribution, 97.5)

real_time = real_validation_df["total_time"].values

within_ci = (
    (real_time >= T_lower_interp) &
    (real_time <= T_upper_interp)
)

coverage = within_ci.mean()
coverage
```

Out[14]:
```
np.float64(0.7314814814814815)
```

Monte Carlo Validation: Opakované podvzorkování na velikost reality

In [15]:
```
# Bootstrap vzorkování z kalibrace na 1/3 velikosti reálného datasetu
n_real = len(real_validation_df)//3

syn_samples = []

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

KS test pro každé podvzorkování

In [16]:
```
# Výpočet KS p-hodnot pro porovnání reálných a kalibrovaných dat
p_vals = []

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

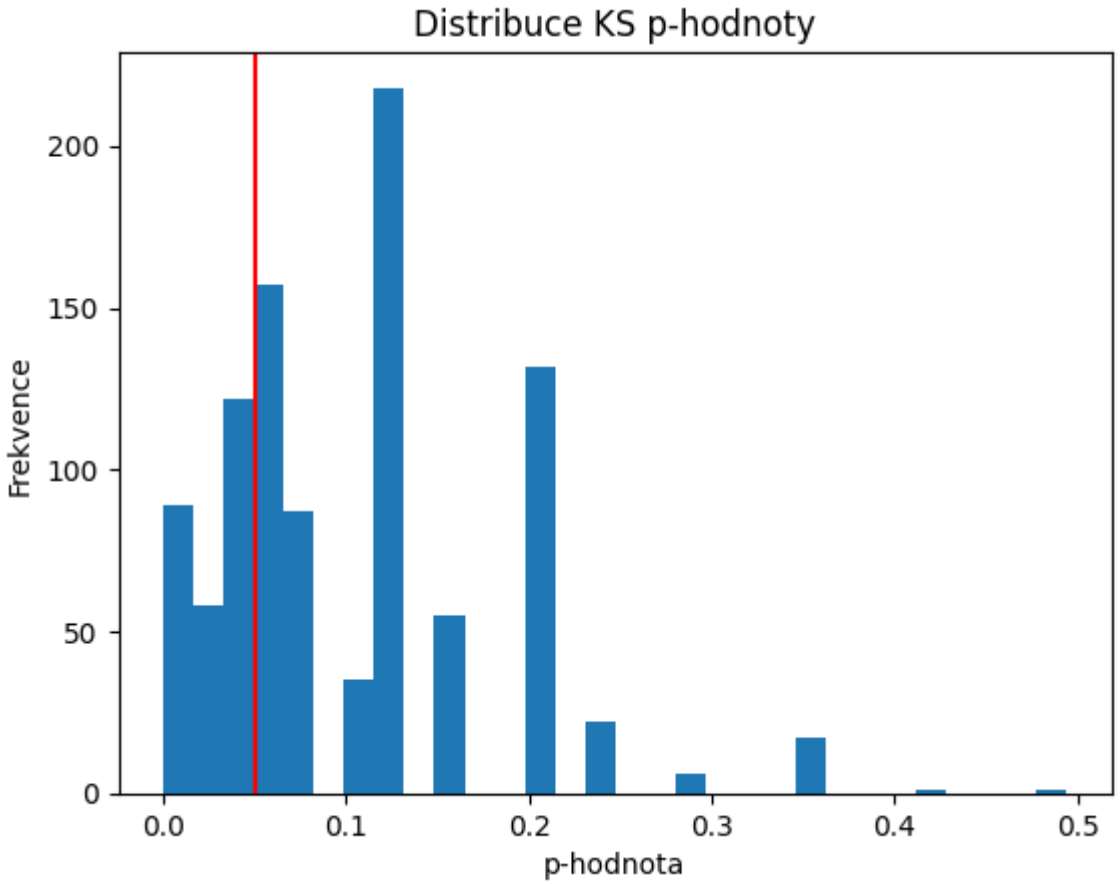
Pravděpodobnost shody modelu

In [17]:
```
valid_ratio = np.mean(np.array(p_vals) > 0.05)

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

```
Podíl validních simulací: 0.731
```

Distribuce KS p-hodnoty

In [18]:
```
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 |

Ve 73.1 % bootstrap vzorků z kalibrace nelze statisticky rozlišit kalibraci od reality (na hladině významnosti α = 0.05 pomocí KS testu).

Porovnání průměru a směrodatné odchylky

Reálná data – referenční hodnoty:

In [19]:
```
# Výpočet průměru a směrodatné odchylky z reálných dat
mean_real = real_validation_df['total_time'].mean()
std_real = real_validation_df['total_time'].std()

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

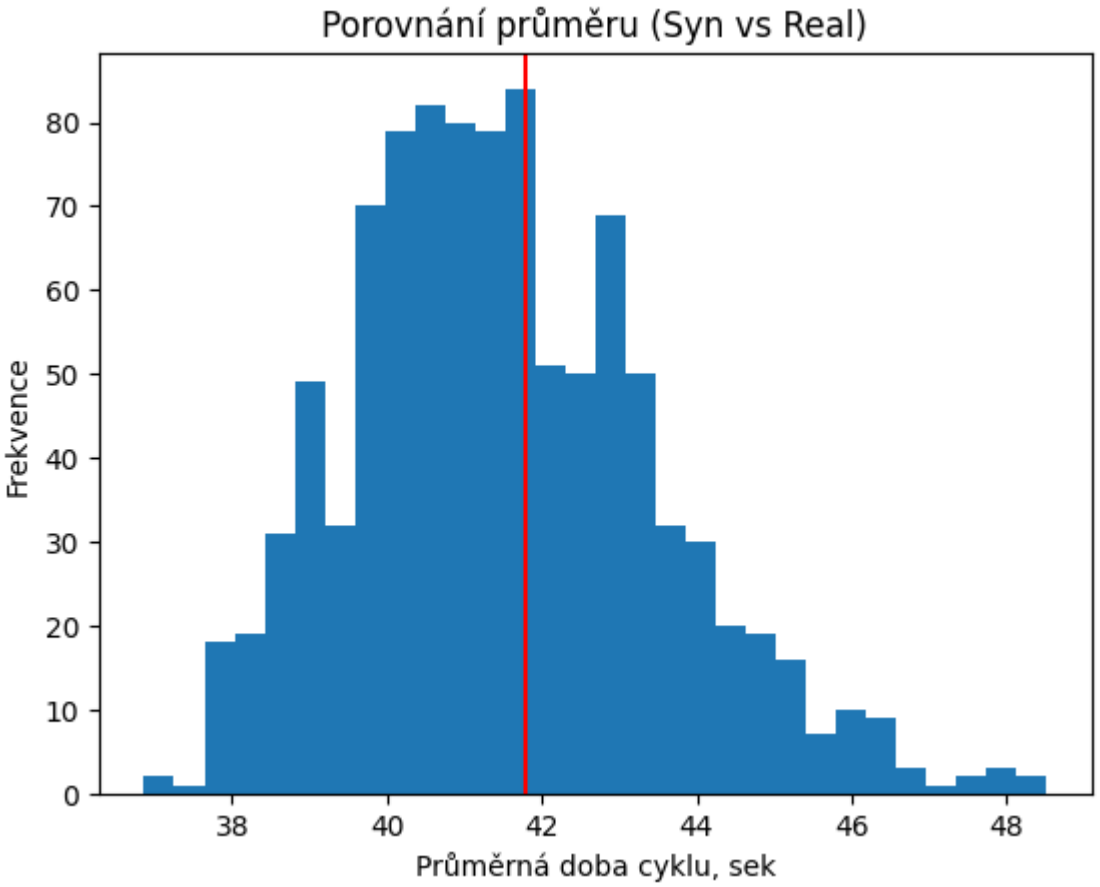
```
Real Mean: 41.7962962962963
Real STD: 28.458449547372744
```

Bootstrap z kalibrace (na velikost reality):

In [20]:
```python
# velikost reálného datasetu
n_real = len(real_validation_df)

syn_means = []
syn_stds = []

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

    sample = df_syn.sample(
        n=n_real,
        replace=True,
        random_state=122 + i
    )

    syn_means.append(sample['total_time'].mean())
    syn_stds.append(sample['total_time'].std())
```

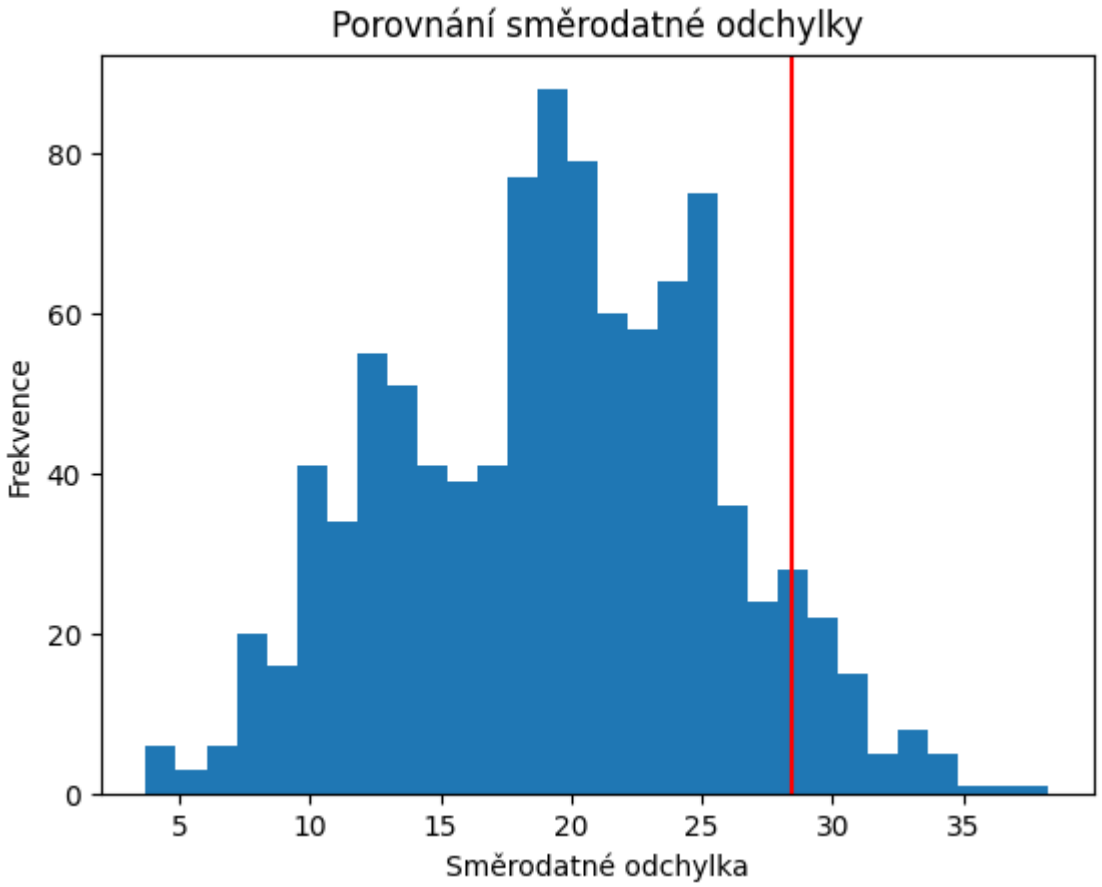
Porovnání průměru

In [21]:
```python
# Histogram průměrů z kalibrace s vyznačením průměru reálných dat
plt.figure()

plt.hist(syn_means, bins=30)
plt.axvline(mean_real, color='red')

plt.title("Porovnání průměru (Syn vs Real)")
plt.xlabel("Průměrná doba cyklu, sek")
plt.ylabel("Frekvence")

plt.show()
```



Porovnání STD

In [22]:
```
# Histogram STD z kalibrace s vyznačením průměru reálných dat
plt.figure()

plt.hist(syn_stds, bins=30)
plt.axvline(std_real, color='red')

plt.title("Porovnání směrodatné odchylky")
plt.xlabel("Směrodatné odchylka")
plt.ylabel("Frekvence")

plt.show()
```



Kvantilová validace

In [23]:
```
mean_CI = np.percentile(syn_means, [2.5, 97.5])
std_CI  = np.percentile(syn_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: [38.11782809 45.89966754]
STD  95% CI: [ 7.63934824 30.93422506]
Real mean: 41.7962962962963
Real STD: 28.458449547372744
```

Model je validní, ale rozsah hodnot je moc široký -> **velký význam stochastichých vlivů**.

Parametrické porovnání dat

Welchův t-test

In [24]:
```
stat, p = ttest_ind(
    real_validation_df['total_time'],
    df_syn['total_time'],
    equal_var=False
)

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

```
Welch t-test p-value: 0.9054042298576069
```

p > 0.05 → nelze zamítnout nulovou hypotézu o shodě středních hodnot porovnávaných souborů

Cohen’s d (velikost efektu)

In [25]:
```
mean_diff = abs(real_validation_df['total_time'].mean() - df_syn['total_time'].mean())

pooled_std = np.sqrt(
    (real_validation_df['total_time'].std()**2 + df_syn['total_time'].std()**2) / 2
)

d = mean_diff / pooled_std

print("Cohen's d:", d)
```

```
Cohen's d: 0.01318405048140204
```

Cohen's d < 0.2 → zanedbatelný význam

Neparametrické porovnání (Distribuce)

Mann–Whitney U test

In [26]:
```
stat, p = mannwhitneyu(
    real_validation_df['total_time'],
    df_syn['total_time']
)

print("Mann-Whitney p-value:", p)
```

```
Mann-Whitney p-value: 0.18632449055958722
```

Vyhodnocení shody simulovaných a reálných dat

Bootstrap KS test

0,731

Interpretace:

Ve 73.1 % případů nelze statisticky rozlišit simulaci od reality .

STD

Real STD: 28.458449547372744 STD 95% CI: [7.63934824 30.93422506]

Interpretace:

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

Průměr

Real mean: 41.7962962962963 Mean 95% CI: [38.11782809 45.89966754]

Interpretace:

Reálný průměr se nachází v intervalu simulace

Welch t-test

p = 0.9054042298576069

Interpretace:

Hodnota p-value = 0.905 je výrazně vyšší než běžně používaná hladina významnosti (α = 0.05). Na základě výsledku Welchova t-testu tedy nelze zamítnout nulovou hypotézu o shodě středních hodnot porovnávaných souborů. Nebyl prokázán statisticky významný rozdíl mezi průměry sledovaných skupin a případné odlišnosti lze přičíst náhodné variabilitě dat.

Cohen's d

0.0131840504814020

Interpretace:

Znamená prakticky nulový efekt. Hodnota Cohenova koeficientu d = 0.013 indikuje zanedbatelnou velikost efektu mezi reálnými a syntetickými daty. Rozdíl mezi jejich středními hodnotami je tedy prakticky nulový a nemá významný věcný dopad, což je v souladu s výsledkem Welchova t-testu, který neprokázal statisticky významný rozdíl mezi soubory.

Mann–Whitney U test (neparametrický)

p = 0.186

Interpretace:

p > 0,05 → nelze zamítnout nulovou hypotézu.

Výsledky Mann-Whitneyova testu (p = 0.186) neprokázaly statisticky významný rozdíl mezi rozdělením reálných a syntetických dat při hladině významnosti 5 %. Lze tedy konstatovat, že model reprodukuje pozorovaná data bez systematického posunu v mediánu.

Celkové vyhodnocení shody simulovaných a reálných dat

Celková validační analýza prokázala vysokou míru shody mezi simulovanými a reálnými daty ve všech posuzovaných charakteristikách. Bootstrapovaný Kolmogorov–Smirnovův test indikoval, že v 73,1 % případů nelze statisticky rozlišit simulaci od reality, což potvrzuje dobrou distribuční konzistenci modelu. Reálná směrodatná odchylka (28,46) se nachází uvnitř 95% intervalu spolehlivosti simulace [7,64; 30,93], přičemž model vykazuje mírně zvýšenou variabilitu, která však zůstává statisticky přijatelná. Reálný průměr (41,80) leží uvnitř 95% intervalu simulace [38,12; 45,90]. Welchův t-test neprokázal statisticky významný rozdíl mezi průměry (p = 0,905), Cohenovo d = 0,013 indikuje prakticky nulový efekt a neparametrický Mann–Whitneyův test rovněž nepotvrdil rozdíl mezi rozděleními (p = 0,186). Souhrnně lze konstatovat, že model reprodukuje reálná data bez statisticky ani věcně významných odchylek a je v analyzovaném rozsahu empiricky validován.

Autor / Organizace / Datum

Vjačeslav Usmanov, ČVUT v Praze, Fakulta stavební

Přehled změn

Datum (YYYY-MM-DD)	Verze	Autor změny	Popis změny

Datum (YYYY-MM-DD)	Verze	Autor změny	Popis změny
2026-01-27	1.1	Vjačeslav Usmanov	added CM_03_Model_Validation.ipynb
2026-02-16	1.2	Vjačeslav Usmanov	changed CM_03_Model_Validation.ipynb