

**ČESKÉ VYSOKÉ  
UČENÍ TECHNICKÉ  
V PRAZE**

**FAKULTA STAVEBNÍ**



**HABILITAČNÍ PRÁCE**

**PŘÍLOHA 5: METODY KALIBRACE  
MODELŮ**

**2026**

**ING. VJAČESLAV USMANOV, PH.D.**

## **OBSAH**

**Kalibrace modelu 01: Bayesovská kalibrace**

**Kalibrace modelu 02: Generátor syntetických dat**

**Kalibrace modelu 03: Validace modelu**

# 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]: SEED = 314159

```
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)

    # Log normal
    accel = pm.LogNormal("accel",
                          mu=np.log(0.6),
                          sigma=0.3)

    # truncated normal
    # 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 bodu
    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,
                       random_seed=SEED)
```

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 10 seconds.

## Diagnostika

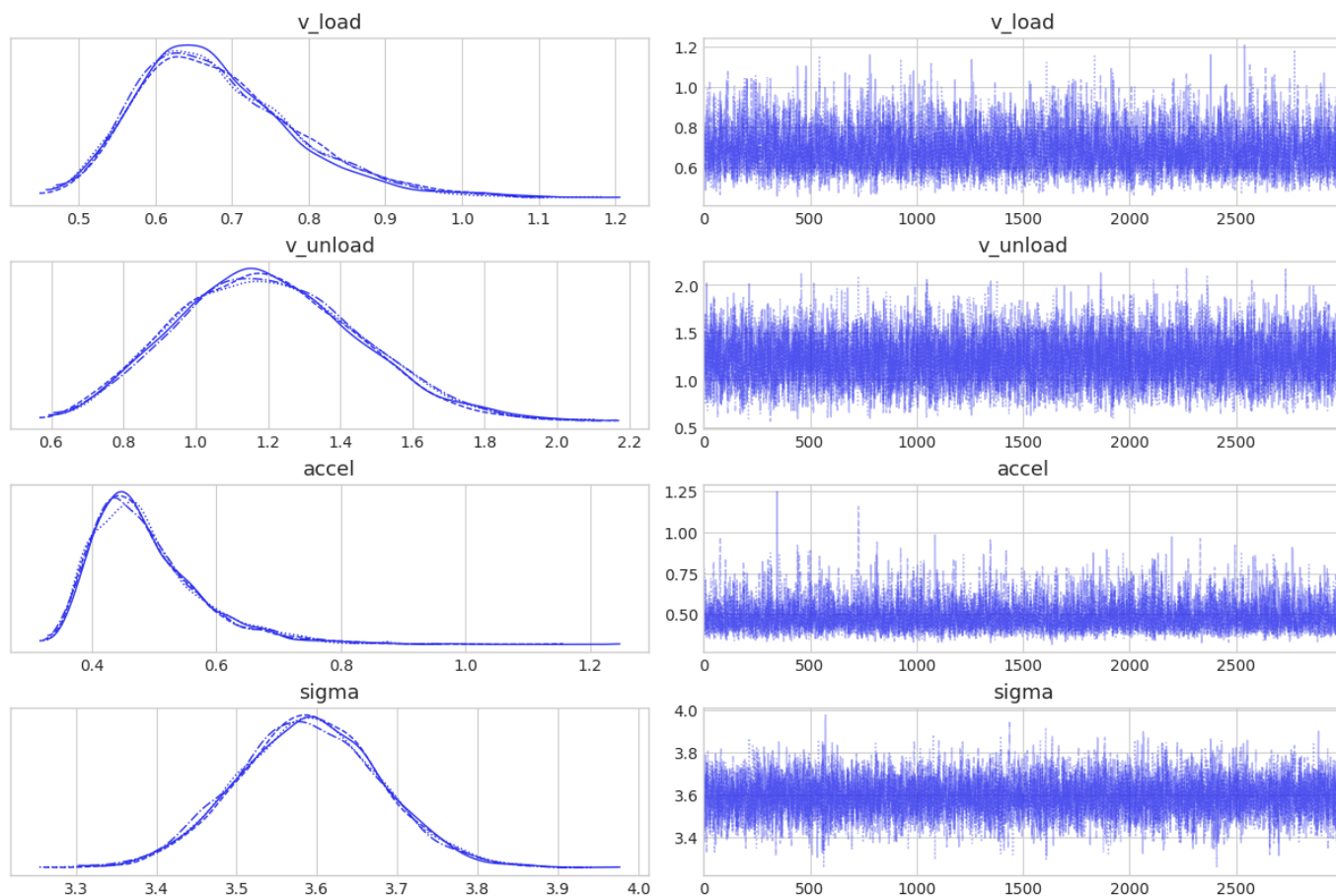
```
In [7]: az.summary(trace)

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)
```

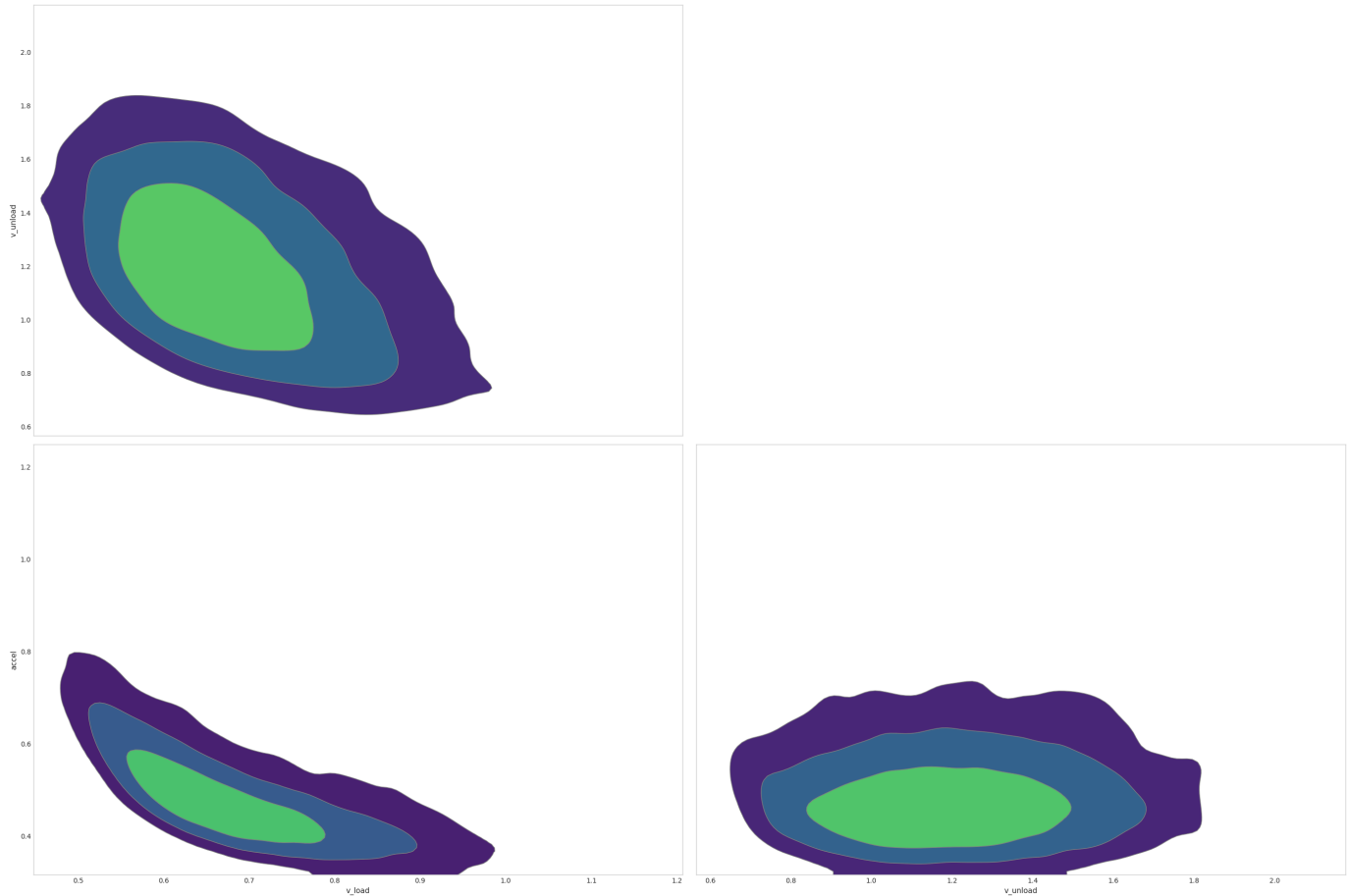
```
Out[7]: array([[<Axes: title={'center': 'v_load'}>,
  <Axes: title={'center': 'v_load'}>],
  [<Axes: title={'center': 'v_unload'}>,
  <Axes: title={'center': 'v_unload'}>],
  [<Axes: title={'center': 'accel'}>,
  <Axes: title={'center': 'accel'}>],
  [<Axes: title={'center': 'sigma'}>,
  <Axes: title={'center':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



Párový graf parametrů

```
In [8]: az.plot_pair(trace, var_names=["v_load", "v_unload", "accel"], kind="kde")
```

```
Out[8]: array([[<Axes: ylabel='v_unload'>, <Axes: >],
               [<Axes: xlabel='v_load', ylabel='accel'>,
                <Axes: xlabel='v_unload'>]], dtype=object)
```



## Posterior predictive kontrola

```
In [9]: with model:
        trace.extend(
            pm.sample_posterior_predictive(trace, random_seed=SEED)
        )
```

Sampling: [y]

Output()

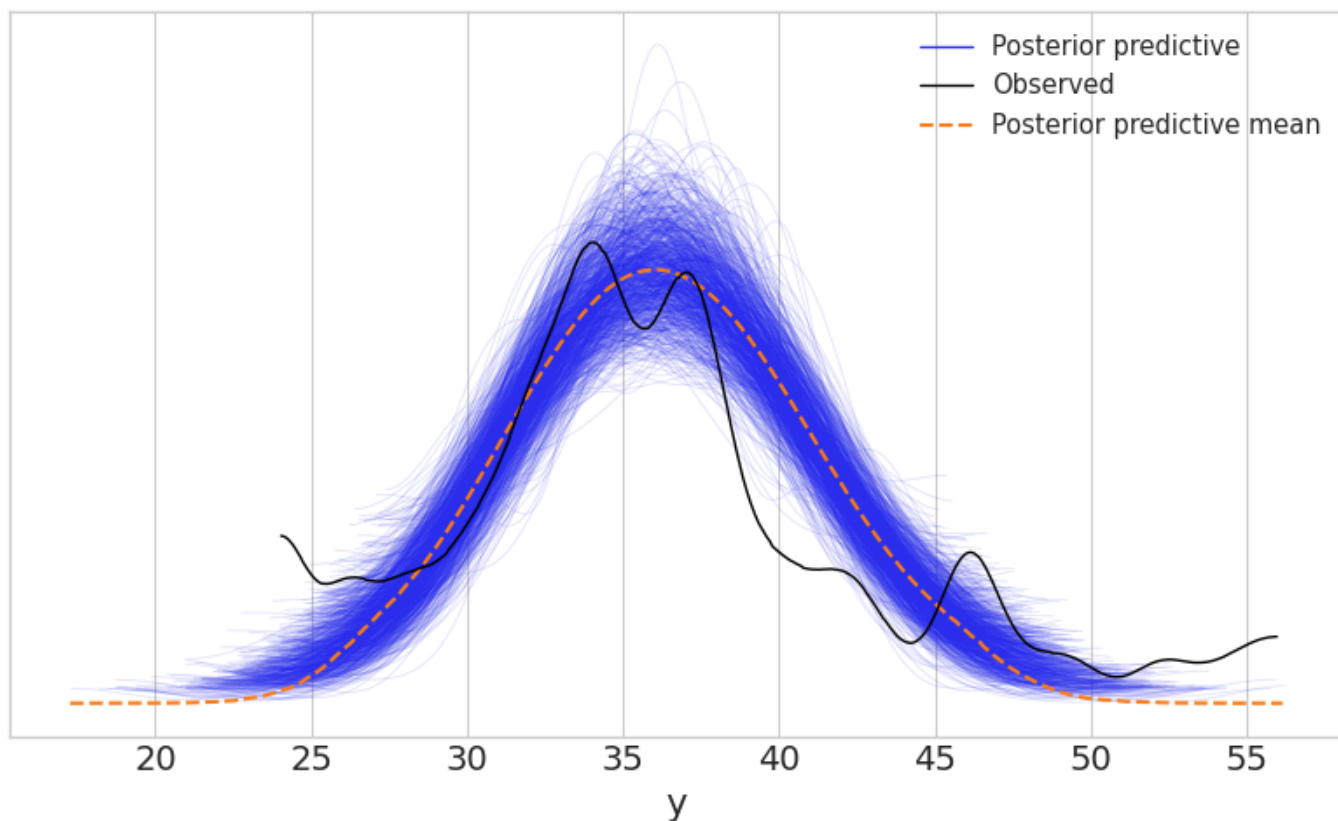
```
In [10]: az.style.use("arviz-whitegrid")

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\_2116\3615469113.py:15: UserWarning: The figure layout has changed to tight  
plt.tight\_layout()



## Souhrnná statistika posteriorní prediktivní distribuce modelu

```
In [11]: summary_df = az.summary(trace, group="posterior_predictive")
summary_df
```

```
Out[11]:
```

|               | mean   | sd    | hdi_3% | hdi_97% | mcse_mean | mcse_sd | ess_bulk | ess_tail | r_hat |
|---------------|--------|-------|--------|---------|-----------|---------|----------|----------|-------|
| <b>y[0]</b>   | 35.120 | 3.609 | 28.365 | 41.938  | 0.033     | 0.024   | 11958.0  | 11972.0  | 1.0   |
| <b>y[1]</b>   | 35.527 | 3.616 | 28.882 | 42.478  | 0.034     | 0.023   | 11288.0  | 11993.0  | 1.0   |
| <b>y[2]</b>   | 37.412 | 3.617 | 30.739 | 44.243  | 0.034     | 0.024   | 11375.0  | 11646.0  | 1.0   |
| <b>y[3]</b>   | 35.630 | 3.579 | 29.090 | 42.465  | 0.033     | 0.023   | 12036.0  | 11689.0  | 1.0   |
| <b>y[4]</b>   | 40.712 | 3.607 | 34.331 | 47.755  | 0.033     | 0.023   | 12058.0  | 11886.0  | 1.0   |
| ...           | ...    | ...   | ...    | ...     | ...       | ...     | ...      | ...      | ...   |
| <b>y[156]</b> | 31.485 | 3.628 | 24.308 | 37.948  | 0.033     | 0.024   | 12112.0  | 11634.0  | 1.0   |
| <b>y[157]</b> | 36.370 | 3.628 | 29.225 | 42.894  | 0.033     | 0.023   | 11892.0  | 11757.0  | 1.0   |
| <b>y[158]</b> | 37.365 | 3.650 | 30.420 | 44.187  | 0.034     | 0.024   | 11747.0  | 11748.0  | 1.0   |
| <b>y[159]</b> | 33.991 | 3.661 | 26.767 | 40.667  | 0.034     | 0.024   | 11781.0  | 11884.0  | 1.0   |
| <b>y[160]</b> | 34.050 | 3.595 | 27.208 | 40.752  | 0.034     | 0.023   | 11481.0  | 11581.0  | 1.0   |

161 rows × 9 columns

## Export datové sady do formátu netCDF a CSV

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

```
In [12]: summary_df.to_csv("../data/05_Calibration/posterior_predictive_summary_three.csv", index=False)
```



```
In [13]: az.to_netcdf(trace, "../../../data/05_Calibration/posterior_trace_three.nc")

Out[13]:  '../../../data/05_Calibration/posterior_trace_three.nc'

In [14]: 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 |
|-----------------|-------|-------|--------|---------|-----------|---------|----------|----------|-------|
| <b>v_load</b>   | 0.681 | 0.105 | 0.508  | 0.886   | 0.001     | 0.001   | 5338.0   | 5799.0   | 1.0   |
| <b>v_unload</b> | 1.198 | 0.250 | 0.736  | 1.653   | 0.003     | 0.002   | 6997.0   | 5989.0   | 1.0   |
| <b>accel</b>    | 0.487 | 0.089 | 0.348  | 0.654   | 0.001     | 0.001   | 6799.0   | 6392.0   | 1.0   |
| <b>sigma</b>    | 3.590 | 0.089 | 3.422  | 3.755   | 0.001     | 0.001   | 8415.0   | 7155.0   | 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 (c
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]

        Návrátová 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 deakceleraci

        # 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 = 1_000

        # nastavení limitních hodnot rozsahu
        dist_min = 1_200
        dist_max = 6_600

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

        # Vygeneruje 1000 hodnot z intervalu
        dist_range = np.linspace(dist_min, dist_max, 1000).round().astype(int)
```

## 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]: (1000, 12000)
```

Simulace obsahuje:

- 1 000 hodnot scénářů
- 12 000 Monte Carlo realizací pro každý scénář

## Základní vyhodnocení (pro každý scénář)

```
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 [ ]: # =====  
# 4 VÝSTUPNÍ SHRNUTÍ  
# =====  
  
print("Průměr:", T_mean)  
print("95% interval:", T_lower, "-", T_upper)  
print("Směrodatná odchylka:", T_std)  
print("Medián:", T_median)
```

## 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: 40.67005574843392

Globální 95% interval: 30.01754062247577 - 103.56032593979283

Globální Směrodatná odchylka: 20.426841413134362

Globální medián: 36.74330653215451

```
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([[ 29.28111717,  29.36527694,  28.90403591, ...,  30.28711427,  
                  31.18226487,  50.03836442],  
                [ 29.2939492 ,  29.37840905,  28.9164497 , ...,  30.2986678 ,  
                  31.19199615,  30.0504439 ],  
                [ 29.30934764,  29.39416758,  28.93134625, ...,  30.31253203,  
                  31.20367368,  30.06493928],  
                ...,  
                [ 43.11148511,  43.51906465,  42.28362193, ...,  42.7395023 ,  
                  41.67063595,  43.05762762],  
                [ 43.12688355,  43.53482318,  42.29851848, ...,  42.75336653,  
                  101.68231348,  43.07212299],  
                [ 43.13971559,  43.54795529,  42.31093227, ...,  42.76492005,  
                  41.69204476,  43.08420247]], shape=(1000, 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)  
  
# rozbálíme 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]: (12000000, 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 |
|-----------------|------|------------|
| <b>8875038</b>  | 5195 | 38.933762  |
| <b>9799933</b>  | 5611 | 40.150066  |
| <b>3368641</b>  | 2714 | 33.359114  |
| <b>10520965</b> | 5935 | 41.106167  |
| <b>4406391</b>  | 3184 | 34.030619  |

## 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 [105... # Instalace potřebných knihoven
#%pip install pandas
#%pip install numpy
```

```
In [106... # 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 [107... # 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 [108... # Zobrazení prvních 5 řádků datasetu
print('Prvních 5 řádků datového rámce')
df_syn.head(5)
```

Prvních 5 řádků datového rámce

```
Out[108...      dist  total_time
0    5195    38.933762
1    5611    40.150066
2    2714    33.359114
3    5935    41.106167
4    3184    34.030619
```

```
In [109... # Základní deskriptivní statistika syntetického datasetu
global_distribution = df_syn[['total_time']]
df_syn.describe()
```



Out[109...

|       | dist          | total_time    |
|-------|---------------|---------------|
| count | 500000.000000 | 500000.000000 |
| mean  | 3901.498332   | 40.691468     |
| std   | 1560.017891   | 20.473294     |
| min   | 1200.000000   | 27.948727     |
| 25%   | 2551.000000   | 33.322842     |
| 50%   | 3897.000000   | 36.741907     |
| 75%   | 5249.000000   | 40.176527     |
| max   | 6600.000000   | 204.146807    |

## Načtení souboru reálných dat z izolované sady měření

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

In [110...

```
# 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 [111...

```
# Zobrazení prvních 5 řádků datasetu
print('Prvních 5 řádků datového rámce')
real_validation_df.head(5)
```

Prvních 5 řádků datového rámce

Out[111...

|   | 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 [112...

```
# 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 [113...

```
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]

    Návrátová 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 [114... # 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'],x['z']),axis=1)

```

```

In [115... # Základní deskriptivní statistika datasetu
real_validation_df.describe()

```

```

Out[115...

```

|       | id         | x           | y           | z           | time       | delay      | type_delay | total_time |
|-------|------------|-------------|-------------|-------------|------------|------------|------------|------------|
| count | 108.000000 | 108.000000  | 108.000000  | 108.000000  | 108.000000 | 108.000000 | 108.000000 | 108.000000 |
| mean  | 141.120370 | 1273.351852 | 992.740741  | 1013.888889 | 36.472222  | 5.324074   | 0.231481   | 41.796696  |
| std   | 79.777578  | 1240.172134 | 1274.175272 | 718.443064  | 6.443802   | 27.813743  | 0.804270   | 28.458494  |
| min   | 1.000000   | 95.000000   | 95.000000   | 0.000000    | 22.000000  | 0.000000   | 0.000000   | 22.000000  |
| 25%   | 77.750000  | 220.000000  | 220.000000  | 500.000000  | 32.000000  | 0.000000   | 0.000000   | 32.750000  |
| 50%   | 136.500000 | 690.000000  | 220.000000  | 875.000000  | 36.000000  | 0.000000   | 0.000000   | 36.500000  |
| 75%   | 212.250000 | 2221.250000 | 1690.000000 | 1750.000000 | 41.000000  | 0.000000   | 0.000000   | 42.000000  |
| max   | 276.000000 | 4002.000000 | 4565.000000 | 2250.000000 | 58.000000  | 260.000000 | 4.000000   | 296.000000 |

## Omezení syntetických dat

```

In [116... df_syn = df_syn[
    (df_syn["dist"] >= real_validation_df['dist'].min()) &
    (df_syn["dist"] <= real_validation_df['dist'].max())
]

```

```

In [117... df_syn.describe()

```

Out[117...

|       | dist          | total_time    |
|-------|---------------|---------------|
| count | 422449.000000 | 422449.000000 |
| mean  | 4186.467465   | 41.382240     |
| std   | 1318.165573   | 20.421828     |
| min   | 1908.000000   | 30.052507     |
| 25%   | 3043.000000   | 34.449459     |
| 50%   | 4184.000000   | 37.339604     |
| 75%   | 5324.000000   | 40.241792     |
| max   | 6465.000000   | 203.606824    |

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

In [118...

```

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[118...

```
np.float64(0.8240740740740741)
```

## Monte Carlo Validation: Opakované podvzorkování na velikost reality

In [119...

```

# 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 [120...

```

# 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 [121...

```

valid_ratio = np.mean(np.array(p_vals) > 0.05)

print("Podíl validních simulací:", valid_ratio)

```

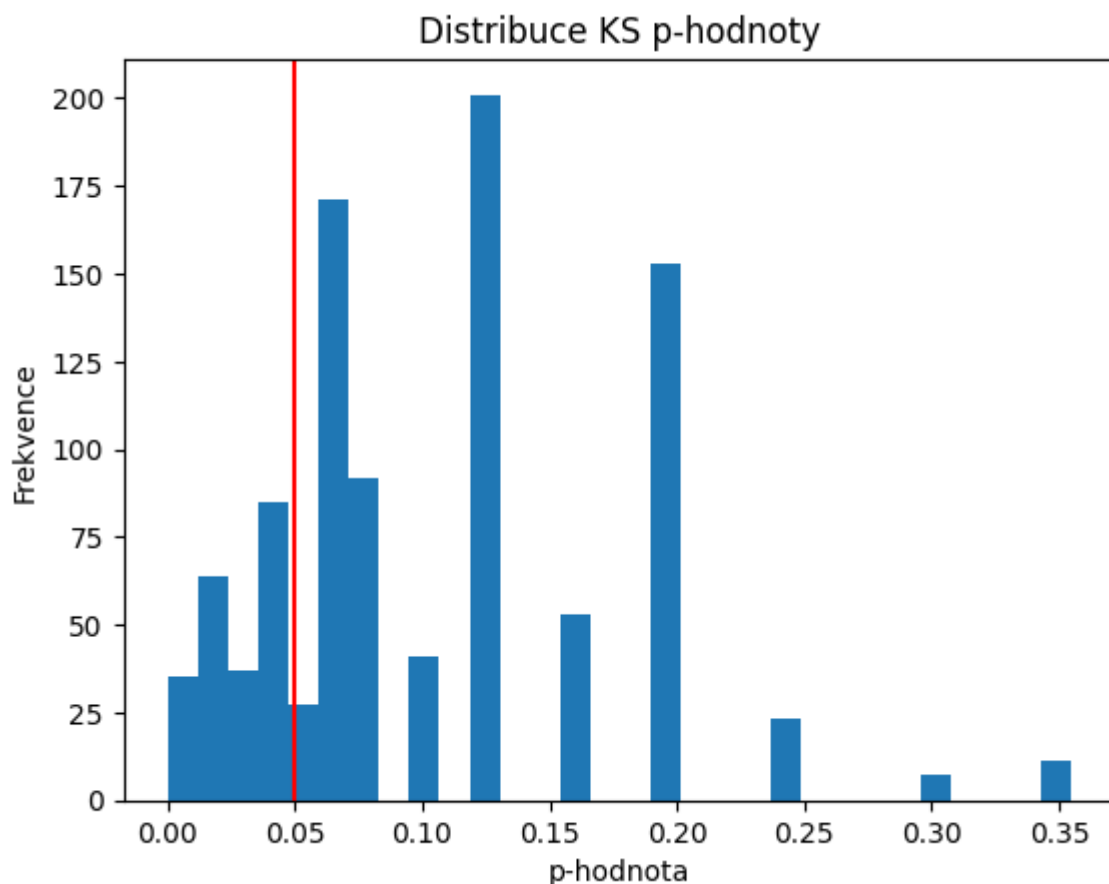
Podíl validních simulací: 0.752

## Distribuce KS p-hodnoty

In [122...

```
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 75.2 % bootstrap vzorků z kalibrace nelze statisticky rozlišit kalibraci od reality (na hladině významnosti  $\alpha = 0.05$  pomocí KS testu).

## Porovnání průměru a směrodatné odchylky

Reálná data – referenční hodnoty:

In [123...

```
# 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 [124...

```
# 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

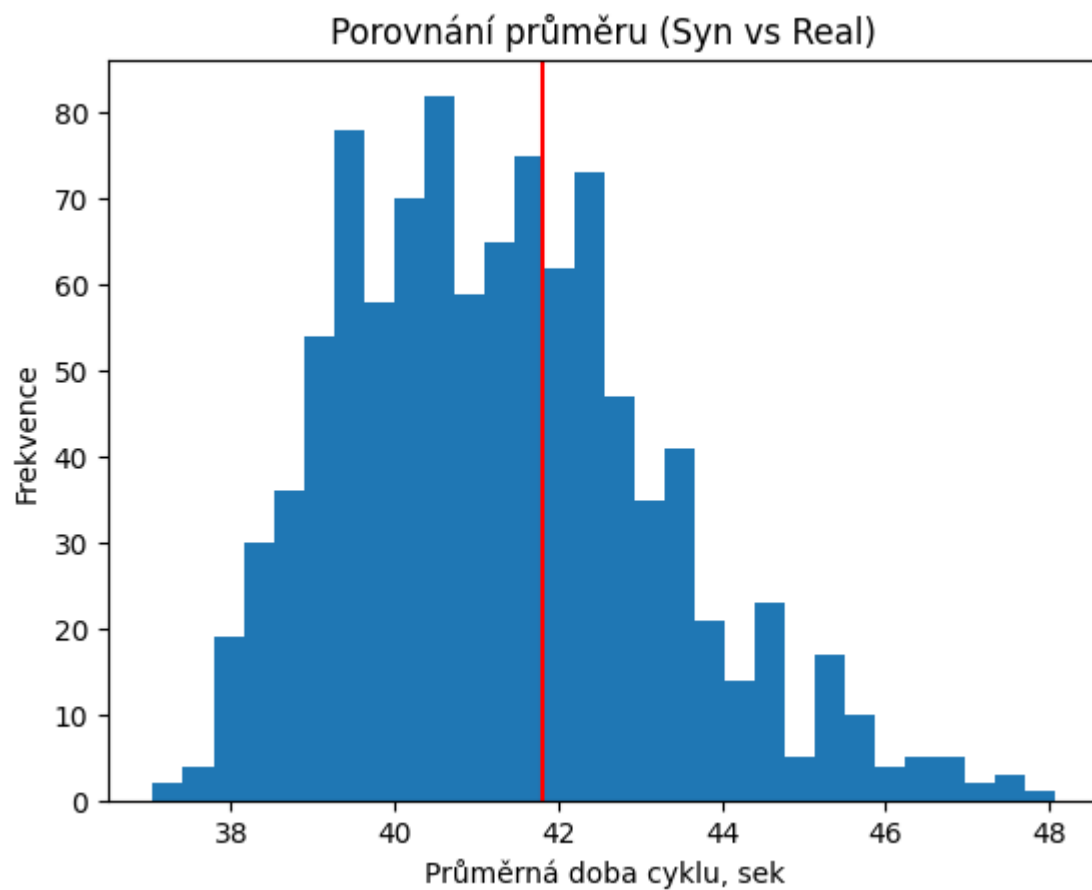
In [125...

```
# 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

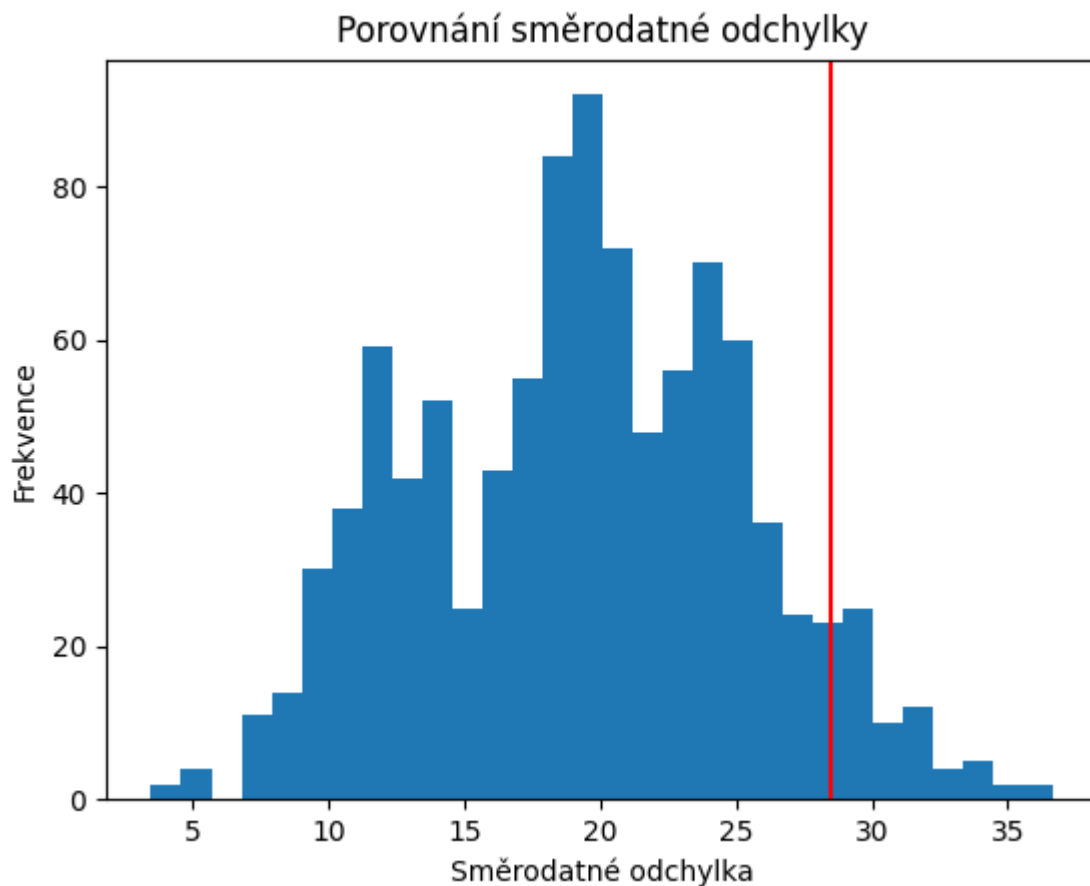
In [126...

```
# 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 [127...

```
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: [38.17522988 45.7577712 ]
STD 95% CI: [ 8.41923038 31.02962672]
Real mean: 41.7962962962963
Real STD: 28.458449547372744
```

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

## Parametrické porovnání dat

### Welchův t-test

In [128...

```
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: 0.8801080338064252
```

$p > 0.05 \rightarrow$  nelze zamítnout nulovou hypotézu o shodě středních hodnot porovnávaných souborů

## Cohen's d (velikost efektu)

In [129...

```
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: 0.016717222048998708

Cohen's d &lt; 0.2 → zanedbatelný význam

## Neparametrické porovnání (Distribuce)

### Mann–Whitney U test

In [130...

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

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

Mann-Whitney p-value: 0.25247430548963856

## Vyhodnocení shody simulovaných a reálných dat

---

### Bootstrap KS test

0,752

*Interpretace:*

Ve 75.2 % případů nelze statisticky rozlišit simulaci od reality .

### STD

Real STD: 28.458449547372744

STD 95% CI: [ 8.41923038 31.02962672]

*Interpretace:*

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

### Průměr

Real mean: 41.7962962962963

Mean 95% CI: [38.17522988 45.7577712 ]

*Interpretace:*



Reálný průměr se nachází v intervalu simulace

---

### Welch t-test

$p = 0.8801080338064252$

*Interpretace:*

Hodnota  $p$ -value = 0.880 je výrazně vyšší než běžně používaná hladina významnosti ( $\alpha = 0.05$ ). 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

0.016717222048998708

*Interpretace:*

Znamená prakticky nulový efekt. Hodnota Cohenova koeficientu  $d = 0.016$  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)

$p = 0.25247430548963856$

*Interpretace:*

$p > 0,05 \rightarrow$  nelze zamítnout nulovou hypotézu.

Výsledky Mann-Whitneyova testu ( $p = 0.252$ ) 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 75,2 % 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 [8,43; 31,03], přičemž model vykazuje mírně zvýšenou variabilitu, která však zůstává statisticky přijatelná. Reálný průměr (41,80) leží uvnitř 95% intervalu simulace [38,12; 45,76]. Welchův t-test neprokázal statisticky významný rozdíl mezi průměry ( $p = 0,880$ ), Cohenovo  $d = 0,0167$  indikuje prakticky nulový efekt a neparametrický Mann–Whitneyův test rovněž nepotvrdil rozdíl mezi rozděleními ( $p = 0,252$ ). 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                          |
|--------------------|-------|-------------------|--------------------------------------|
| 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 |