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