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	0.681	0.105	0.508	0.886	0.001	0.001	5338.0	5799.0	1.0
v_unload	1.198	0.250	0.736	1.653	0.003	0.002	6997.0	5989.0	1.0
accel	0.487	0.089	0.348	0.654	0.001	0.001	6799.0	6392.0	1.0
sigma	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 (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 = 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 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 [ ]:
```
# ======================================================
# 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)

# 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]:
```
(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
8875038	5195	38.933762
9799933	5611	40.150066
3368641	2714	33.359114
10520965	5935	41.106167
4406391	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