ohm

Simulationstools für B-SDC

Übung 1 **(Mangold)** **Wintersemester 2025/26**

Kapitel 1: Erste Monte Carlo Experimente

Aufgabe 1 *Reproduzierbare Python-Umgebung und sauberes Projektgerüst erstellen*

Führen Sie folgende (mittels `venv + pip`) oder ähnliche Schritte aus, um ein Python Environment zu erstellen. Erzeugen Sie zusätzlich eine `requirements.txt` mit fixierten Versionen:

```
python3 -m venv .venv
source .venv/bin/activate # Windows: .venv\Scripts\activate
pip install --upgrade pip
pip install numpy scipy pandas matplotlib
pip freeze > requirements.txt
```

Alternativ erzeugen Sie eine `environment.yml` mit fixierten Versionen:

```
conda create --name my-env -y # name 'my-env', -y ohne 'yes'
conda activate my-env
conda install numpy scipy pandas matplotlib -y
conda env export > environment.yml
```

Legen Sie folgende Struktur an und begründen Sie etwaige Abweichungen in der `README.md`:

```
project/
    configs/          # yaml/json für Seeds, Parameter
    data/
        raw/          # unverändert, read-only
        interim/      # bereinigt/transformiert
        processed/    # final für Analysen
    figures/          # png/pdf-Plots
    notebooks/        # Exploration, Demos
    reports/          # Tabellen + Kurztexte
    src/              # Funktionen/Modelle
    tests/            # Z.B. Unit-Tests
```

```
    requirements.txt # or environment.yml
    README.md
```

Fügen Sie in `reports/` eine kurze `ABOUT.txt` mit Datum, Autoren, Pythonversion und Paketversionen ein.

Aufgabe 2 *Zufallszahlen prüfen (Histogramm, QQ-Plot)*

a) Gleichverteilung

(i) Ziehen Sie $N \in \{10^4, 10^5\}$ Stichproben $U \sim \mathcal{U}(0,1)$ mit

`numpy.random.default_rng` (PCG64)

und festem Seed.

(ii) Erstellen Sie ein Histogramm (z.B. 50 Bins) und vergleichen Sie mit der theoretischen Dichte (Linie).

(iii) Berechnen Sie Mittelwert und Varianz (sollten nahe 0.5 bzw. 1/12 liegen). Berichten Sie Abweichungen relativ in %.

(iv) QQ-Plot gegen die theoretische Quantilfunktion von $\mathcal{U}(0,1)$ (Identitätslinie). Kommentieren Sie Ausreißer.

b) Standardnormalverteilung

(i) Wiederholen Sie die Schritte für $Z \sim \mathcal{N}(0,1)$ (Histogramm + theoretische Dichte).

(ii) QQ-Plot gegen Standardnormal. Interpretieren Sie die Form der Abweichungen (Tails, Schiefe?).

Hinweise: Verwenden Sie `rng = np.random.default_rng(seed)` und übergeben Sie den Generator explizit an Funktionen. Speichern Sie Plots in `figures/`.

Aufgabe 3 *π-Schätzung mit Konfidenzintervallen*

a) Implementieren Sie in `src/pi_mc.py` eine Funktion

```
def estimate_pi(n_samples, rng):
    """
    Schätzt pi mittels Viertelkreis-MC und liefert (pi_hat,
    se) zurück.

    Parameter
    ---------
```

```
    n_samples : int
        Anzahl der Zufallspunkte.
    rng : np.random.Generator
        Zufallszahlengenerator (z. B.
        np.random.default_rng(seed)).

    Returns
    -------
    pi_hat : float
        Schätzer für pi (4 * Trefferquote).
    se : float
        Standardfehler von pi_hat aus der Binomialnäherung.
    """
```

Hinweis: Versuchen Sie eine vektorisierte Umsetzung zu implementieren (keine Python-Schleife), und achten Sie auf Rückgabe des Standardfehlers

$$\mathrm{SE}(\hat{\pi}) = 4\sqrt{\hat{p}(1-\hat{p})/N}$$

b) (i) Führen Sie für $N \in \{10^3, 10^4, 10^5\}$ je $R = 20$ Replikationen durch (unabhängige Seeds; dokumentieren Sie die Seed-Strategie).

(ii) Berichten Sie pro N

- Mittelwert der $\hat{\pi}$
- empirischer Standardfehler (SE) über die Replikate
- sowie das 95%-Konfidenzintervall für den Mittelwert der Replikate.

(iii) Visualisieren Sie die Verteilung der $\hat{\pi}$ (Histogramm) pro N. Kommentieren Sie die $1/\sqrt{N}$ Skalierung

Aufgabe 4 *Monte-Carlo-Integration einfacher Funktionen*

Sie möchten $\mathbb{E}[g(X)]$ mit $X \sim \mathcal{U}(0,1)$ für verschiedenste Funktionen g schätzen.

a) Implementieren Sie zunächst folgende Funktionen in `python`:

(i) $g_1(x) = \sqrt{x}$

(ii) $g_2(x) = \exp -x^2$

(iii) $g_3(x) = \frac{\sin \pi x}{\pi x}$ mit Konvention $g_3(0) = 1$

b) Implementieren Sie eine generische Funktion

`mc_expectation(g, sampler, N, rng)`

in `src/integration.py`, die für verschiedene Funktionen `g` und beliebige Verteilungen (modelliert durch `sampler`) den Schätzer $\hat{\mu}$ und den SE zurückgibt.

c) Verwenden Sie $N \in \{10^3, 10^4, 10^5\}$, berichten Sie $\hat{\mu} \pm 1.96\text{SE}$ und die Laufzeit pro N, für jede der drei Funktionen aus Teilaufgabe a).

Aufgabe 5 *Abgabeformat & Code-Stil (PEP 8, Docstrings)*

Gehen Sie nach Durchführung der Aufgaben 1-4 auf folgende Punkte ein (wenn möglich)

a) `README.md`: Setup-Schritte, Umgebung (Python/Packages), Reproduktionsanleitung.

b) `requirements.txt` *oder* `environment.yml` (Versionen fixiert).

c) `src/`: (saubere Funktionssignaturen, Typannotationen), Docstrings (Google- oder NumPy-Stil).

d) `notebooks/`: kurze, lineare Ausführung (Restart → Run All), keine versteckte Logik, bzw. nur wenn explizit in der Aufgabenstellung gefordert.

e) `figures/`, `reports/`: Plots und kurze Zusammenfassung (1–2 Seiten) mit SE/CI und Kernaussagen.

f) Code-Stil:

- PEP 8 (Zeilenlänge $\leq$ 88 Zeichen), konsistentes Naming, keine `print`-Orgie; Logging optional.
- Docstrings mit `Parameters`/`Returns`; Beispielaufrufe erwünscht.
- Optionale Tools: `black` (Formatierung), `ruff`/`flake8` (Linting), (`pytest` (Smoke-Tests)).

Aufgabe 6 *Erstellen eines Repositories auf github.com*

Veröffentlichen Sie Ihre Ergebnisse auf github (auf diese Art wird die Studienarbeit am Ende des Semesters abgegeben), in dem Sie folgende Schritte durchführen:

a) Erstellen Sie einen kostenlosen Account

b) Erstellen Sie ein leeres Repository auf *github.com*

c) Pushen Sie den Inhalt des Folders der Veranstaltung ins leere Repository

d) Teilen Sie den Link zu Ihrem Repository mit dem github-User benedikt-mangold um einen Stern zu erhalten (alternativ schicken Sie den Link per Email an benedikt.mangold@th-nuernberg.de).