Problem ①

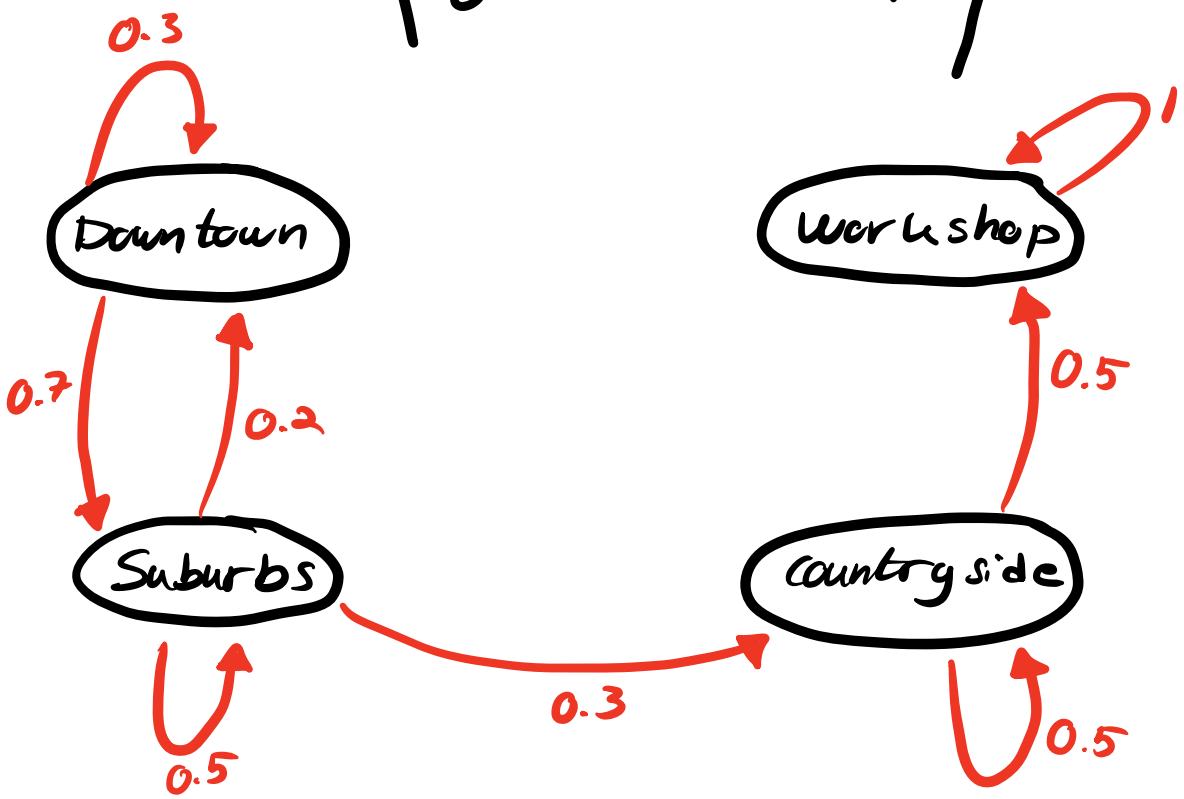
①

The following table shows the probabilities of a truck transitioning between these regions at each time step:

Current region	Probability of transitioning to downtown	Probability of transitioning to the suburbs	Probability of transitioning to the countryside	Probability of transitioning to the Workshop
Downtown	0.3	0.7	0	0
Suburbs	0.2	0.5	0.3	0
Countryside	0	0	0.5	0.5
Workshop	0	0	0	1

1. If a truck is currently in the downtown, what is the probability that it will be in the countryside region after 10 time steps? **[2p]**

So: $P = \begin{pmatrix} 0.3 & 0.7 & 0 & 0 \\ 0.2 & 0.5 & 0.3 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ = Transition matrix



So the **transition matrix** P (rows = current state, columns = next state) is

$$P = \begin{pmatrix} 0.3 & 0.7 & 0 & 0 \\ 0.2 & 0.5 & 0.3 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

we want the probability of being in the countryside after 10 steps.

So: $\mathbb{P}(X_{10} = C \mid X_0 = D) = (P^{10})_{D,C}$

We therefore calculate Transition matrix to the power 10 (P^{10}), and look at the matrix from Downtown to Countryside, and that's the answer.

=> "The probability that a truck that starts in downtown is in the countryside after 10 steps is approximately 0.0849 ≈ 8.5%."

Problem ①:

② 2. If a truck is currently in the downtown, what is the probability that it will be in the countryside region the first time after three time steps or more? [2p]

Förklaring: Markdown text: koden?

Problem ①:

③ 3. Is this Markov chain irreducible? Explain your answer. [3p]

- A Markov chain is irreducible if:

- You can get from every state to every other state
- As long as there is some path with positive probability.

* If matrix has no zeros at all, it is trivially irreducible, because all transitions can occur in 1 step

x This does not need to necessarily happen in one step

So: No zeros in matrix → irreducible ?

Answer:

— In this case, it's not possible since you can't go to any other state from the countryside state.

Problem ④:

④ 4. What is the stationary distribution? Furthermore is it reversible? (Explain your answer) **[3p]**

We label the stationary distribution as

$\pi = (\pi_1, \pi_2, \pi_3, \pi_4) = (\text{Downtown}, \text{suburbs}, \text{country side}, \text{workshop})$

Transition matrix is still the same: $\begin{pmatrix} 0.3 & 0.7 & 0 & 0 \\ 0.2 & 0.5 & 0.3 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

Stationary distribution satisfies: $\pi \cdot P = \pi$

$$\Rightarrow \begin{cases} \pi_1 = 0.3\pi_1 + 0.2\pi_2 \\ \pi_2 = 0.7\pi_1 + 0.5\pi_2 \\ \pi_3 = 0.3\pi_2 + 0.5\pi_3 \\ \pi_4 = 0.5\pi_3 + 1\pi_4 \end{cases}$$

and we also know that: $\boxed{\pi_1 + \pi_2 + \pi_3 + \pi_4 = 1}$

Put these 5 equations into matrixes and solve:

$$\begin{cases} \pi_1 = 0 \\ \pi_2 = 0 \\ \pi_3 = 0 \\ \pi_4 = 1 \end{cases}$$

$$\Rightarrow (0, 0, 0, 1)$$

In python code, we get:

```
stationary = np.array([0, 0, 0, 1]).
```

array since we only have vector

and not matrix.

AND: is it reversible??

Look at exam hand written notes for exam: 2024-06-07 to know reversible, but in our case this is False.

Transition matrix is still the same:

$$\begin{array}{c} \\ D \\ S \\ C \\ W \end{array}\begin{array}{c} \begin{array}{cccc} D & S & C & W \end{array} \\ \begin{pmatrix} 0.3 & 0.7 & 0 & 0 \\ 0.2 & 0.5 & 0.3 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

Check this from D to S.

$$\pi = (\underset{D}{0}, \underset{S}{0}, \underset{C}{0}, \underset{W}{1})$$

So:

1) Check detailed balance on edges where transitions are non-zero.

- **Pair D<->S:**
  - **Forward**: $\pi_D \cdot P_{DS} = 0 \cdot 0.7 = 0$

    ↑ From D to S probability
  - **Backward**: $\pi_S \cdot P_{SD} = 0 \cdot 0.2 = 0$
  - **Equal**: ✓

Just ran the code i have called is_reversible(..) to check reversibility. It works for every matrix?

we do this for all possible paths.

So this you should also check, not only check in order:

Pair: A<=>C

A<=>D

If they were not 0 (zero).

Problem 1

(5)

5. **Advanced question:** What is the expected number of steps it takes starting from the Downtown region to first reach the Workshop?

Hint: to get within 1 decimal point, it is enough to compute the probabilities for hitting times below 50.
Motivate your answer in detail. **[4p]**

You could also solve this question by simulation, but this gives you a maximum of **[2p]**.

✅ PART 1 — Solve by hand (step-by-step)

We want:

$$\mathbb{E}[\tau_{\text{Workshop}} \mid X_0 = D]$$

where

- D = Downtown
- S = Suburbs
- C = Countryside
- W = Workshop (absorbing)

Transition matrix:

$$P = \begin{pmatrix} 0.3 & 0.7 & 0.0 & 0.0 \\ 0.2 & 0.5 & 0.3 & 0.0 \\ 0.0 & 0.0 & 0.5 & 0.5 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix}$$

We define expected hitting times:

- h_D = expected steps to reach W starting from D
- h_S = expected steps to reach W starting from S
- h_C = expected steps to reach W starting from C
- $h_W = 0$ (because we are already at Workshop)

🔷 Step 1: Write the first-step equations

For any *non-absorbing* state i:

$$h_i = 1 + \sum_j P_{ij} h_j$$

Equation for D

$$h_D = 1 + 0.3h_D + 0.7h_S$$

Equation for S

$$h_S = 1 + 0.2h_D + 0.5h_S + 0.3h_C$$

Equation for C

$$h_C = 1 + 0.5h_C + 0.5h_W = 1 + 0.5h_C$$

Solve C first:

$$h_C - 0.5h_C = 1$$

$$0.5h_C = 1$$

$$h_C = 2$$

🔷 Step 2: Solve the D–S subsystem

Equation (D)

$$h_D = 1 + 0.3h_D + 0.7h_S$$

Bring terms together:

$$0.7h_D = 1 + 0.7h_S$$

$$h_D = \frac{1}{0.7} + h_S$$

$$h_D = 1.428571 + h_S$$

Equation (S)

$$h_S = 1 + 0.2h_D + 0.5h_S + 0.3(2)$$

$$h_S = 1 + 0.2h_D + 0.5h_S + 0.6$$

$$0.5h_S = 1.6 + 0.2h_D$$

$$h_S = 3.2 + 0.4h_D$$

🔷 Step 3: Solve the two equations

Substitute $h_D = h_S + 1.428571$:

$$h_S = 3.2 + 0.4(h_S + 1.428571)$$

Simplify:

$$h_S = 3.2 + 0.4h_S + 0.571428$$

$$0.6h_S = 3.771428$$

$$h_S = 6.285714$$

Now compute h_D:

$$h_D = h_S + 1.428571 = 7.714285$$

Thus the expected number of steps from Downtown is:

$$\boxed{7.7 \text{ steps (approx)}}$$

🎉 Final answer:

$$\boxed{h_D \approx 7.7}$$

Problem ②:

⑥

Step-by-step explanation (plain text)

We have residuals

- $r_i = y_i - \hat{y}_i$, for $i = 1, \ldots, n$,
  where y_i = true MPG (test set), $\hat{y}_i$ = predicted MPG.

Let n be the number of residuals in the **test** set.

6(a) Plot the empirical distribution function (EDF) of the residuals

1. Collect the residuals

From your code you already have

```python
residuals = problem2_y_test - problem2_y_pred
```


These values are your sample $r_1, \ldots, r_n$.

2. Sort the residuals:

To define the empirical distribution function F_n, you conceptually:

- Order the residuals from smallest to largest:

$$r_{(1)} \leq r_{(2)} \leq \cdots \leq r_{(n)}.$$

On paper, if n is small, you can actually list and sort them. For your real dataset n is larger, so you let the computer sort them, but the **mathematics** is always this sorting step.

---

3. Definition of the EDF

The empirical distribution function of the residuals is

$$F_n(x) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\{r_i \leq x\}.$$

Interpretation:

- For a given value x, $F_n(x)$ is the **proportion of residuals** that are $\leq x$.
- It is a **step function** that jumps by $1/n$ at each observed residual.

6(b) Add 95% DKW confidence bands

We use the Dvoretzky–Kiefer–Wolfowitz (DKW) inequality. For a given confidence level $1 - \alpha$ (here $1 - 0.05 = 0.95$) it tells us that

$$\Pr\Big(\sup_x |F_n(x) - F(x)| \leq \varepsilon\Big) \geq 1 - \alpha,$$

where F is the **true** CDF of the residuals, and

$$\varepsilon = \sqrt{\frac{1}{2n} \log\Big(\frac{2}{\alpha}\Big)}.$$

Here $\log$ is the natural logarithm.

---

1. Compute n

- n = number of residuals (length of the test set):
  $n = |\text{problem2\_y\_test}|$.

On paper, you just count how many residuals you have; in code you do `n = len(residuals)`.

2. Choose α and compute ε

You are told to use $\alpha = 0.05$. Then compute

$$\varepsilon = \sqrt{\frac{1}{2n} \log\Big(\frac{2}{0.05}\Big)} = \sqrt{\frac{1}{2n} \log(40)}.$$

Steps if you wanted to do it by hand:

1. Compute $\frac{2}{\alpha} = \frac{2}{0.05} = 40$.
2. Take $\log(40)$ (usually with a calculator).
3. Multiply by $1/(2n)$.
4. Take the square root.

In Python, that's something like:

```python
alpha = 0.05
n = len(residuals)
eps = np.sqrt( (1 / (2*n)) * np.log(2/alpha) )
```

3. Construct the confidence bands

For each x, we already have the EDF $F_n(x)$.
The DKW bands are:

- **Lower band**: $L(x) = F_n(x) - \varepsilon$
- **Upper band**: $U(x) = F_n(x) + \varepsilon$

But probability values must be between 0 and 1, so we **clip**:

$$L(x) = \max\big(F_n(x) - \varepsilon, 0\big),$$
$$U(x) = \min\big(F_n(x) + \varepsilon, 1\big).$$

On paper:

- For each jump level $F_n(x) = k/n$, compute:
  - $L = \max(k/n - \varepsilon, 0)$,
  - $U = \min(k/n + \varepsilon, 1)$.
- Draw these as two step functions around the EDF.

(c) Interpretation

The DKW confidence band is a **uniform** confidence band around the EDF $F_n(x)$. With probability at least $1 - \alpha = 0.95$, the entire true CDF $F(x)$ of the residuals lies within the band:

$$L(x) \le F(x) \le U(x) \quad \text{for all } x.$$

This band quantifies the **sampling uncertainty** due to having only n residuals. For large n, the value of ε is small and the bands are narrow (less uncertainty). For small n, ε is larger and the bands are wider (more uncertainty).

In model checking, these bands can be used as follows:

1. **Checking distributional assumptions (e.g. normality of residuals)**
   We can compare a theoretical CDF $F_{\text{theory}}(x)$ (for example a normal CDF with mean 0 and estimated variance) to the EDF and its bands. If $F_{\text{theory}}(x)$ lies outside the band for some x, this is evidence that the residual distribution is not consistent with the assumed distribution at the 95% level.
2. **Comparing models or reference distributions**
   Different models imply different residual distributions. By comparing the EDF of residuals from a model to a reference CDF and checking whether the reference CDF lies within the DKW band, we can assess whether that model is compatible with the observed residuals.
3. **Detecting outliers and heavy tails**
   Large jumps in the EDF in the tails correspond to extreme residuals. If these parts of the EDF are close to or outside what would be expected under a reference distribution (and outside the DKW band when compared to that reference), this may indicate heavy tails or outliers and suggest that the model does not adequately capture the error structure.

Overall, if the EDF of the residuals and a reasonable reference CDF are compatible within the DKW bands, the residual distribution is consistent with the model assumptions. Systematic deviations beyond the bands indicate potential model misspecification (non-normality, incorrect variance structure, missing nonlinear effects, etc.).

Problem 3:

① 1. **[4p]** Using inversion sampling, construct 1000 samples from the below distribution

$$F[x] = \begin{cases} 0, & x \leq 0 \\ e^x - 1, & 0 < x < \ln(2) \\ 1, & x \geq \ln(2) \end{cases}$$

We want to generate 1000 samples from a distribution whose CDF is:

$$F[x] = \begin{cases} 0, & x \leq 0 \\ e^x - 1, & 0 < x < \ln(2) \\ 1, & x \geq \ln(2) \end{cases}$$

This is a truncated exponential-like distribution on the interval $[0, \ln(2)]$.

The inversion sampling recipe is:

1) Generate $U \sim \text{Uniform}(0,1)$
2) Compute $X = F^{-1}(U)$

so we need to compute the inverse CDF.

Step 1: Find $F^{-1}(u)$:

we only invert the part where $0 < x < \ln(2)$:

$u = F(x) = e^x - 1$

Solve for x:

$e^x = u + 1$

$\Rightarrow x = \ln(u+1)$

Domain check:

- when $u=0$, $x = \ln(1) = 0$
- when $u=1$, $x = \ln(2)$

So: The inverse CDF is simply:

$F^{-1}(u) = \ln(u+1)$, $u \in (0,1)$

Step 2: Generate samples:

1) Generate 1000 uniform samples $U_1, \dots, U_{1000}$

2) Transform using $X_i = \ln(1+U_i)$.

Problem ③

② 2. **[2p]** Use the above 1000 samples to estimate the mean and variance.

Given the simulated values:

$X_1, X_2, \dots, X_{1000}$

the sample mean is:

$$\hat{\mu} = \frac{1}{1000} \cdot \sum_{i=1}^{1000} x_i$$

The sample variance (population-style estimator, matching NumPy's np.var) is:

$$\hat{\sigma}^2 = \frac{1}{1000} \cdot \sum_{i=1}^{1000} (x_i - \hat{\mu})^2$$

Problem ③

③

3. **[4p]** Using the Accept–Reject sampler (Algorithm 1 in TFDS notes) construct 1000 samples from the same distribution.
What proposal distribution did you choose and why?
What proportion of samples were accepted?

we want to sample from the same distribution as in part ①, i.e., with CDF:

$$F[x] = \begin{cases} 0 & , \text{ for } x \leq 0 \\ e^x - 1 & , 0 < x < \ln(2) \\ 1 & , x \geq \ln(2) \end{cases}$$

The density is the derivative of the CDF on the interval $0 < x < \ln(2)$:

$f(x) = F[x] = e^x, \quad 0 < x < \ln(2)$

Check normalization:

$$\int_0^{\ln(2)} e^x dx = [e^x]_0^{\ln(2)} = 2 - 1 = 1$$

So: $f(x) = e^x$ on $(0, \ln(2))$ is already a valid density (no extra normalizing constant)

We now want to use the Accept-Reject sampler:

- Choose a proposal density $g(x)$ that is easy to sample from.
- Find a constant $M \geq 1$ such that
  $f(x) \leq M \cdot g(x)$ for all x in the support.
- Use the accept-reject algorithm.

Step 1: Choose a proposal distribution $g(x)$

A natural choice is a uniform proposal on exactly the same support:

$$Y \sim \text{Uniform}(0, \ln(2))$$

Then the density of Y is:

$$g(x) = \frac{1}{\ln(2)}, \quad 0 < x < \ln(2)$$

Why this choice?

- It has same support as f
- It is very easy to sample from
- The bound M is easy to compute.

Step 2: Find the envelope constant M:

We need M such that:

$$f(x) \leq M \cdot g(x) \text{ for all } x \in (0, \ln(2))$$

Compute:

$$\frac{f(x)}{g(x)} = \frac{e^x}{\frac{1}{\ln(2)}} = e^x \ln(2)$$

On $0 < x < \ln(2)$, that is maximized at $x = \ln(2)$

$$\max_{x \in (0, \ln(2))} e^x \ln(2) = e^{\ln(2)} \cdot \ln(2) = 2\ln(2)$$

So: A tight valid choise for M is:

$$M = 2\ln(2)$$

This ensures $f(x) \leq M \cdot g(x)$ $\forall x$

Step 3: Compute the acceptance probability

The accept-reject algorithm says:

- Propose $Y \sim g$
- Draw $U \sim \text{Uniform}(0,1)$
- Accept Y as a sample from f if:

$$U \leq \frac{f(Y)}{M \cdot g(Y)}$$

Here:

- $f(Y) = e^Y$
- $g(Y) = \frac{1}{\ln(2)}$
- $M = 2\ln(2)$

Thus:

$$\frac{f(Y)}{M \cdot g(Y)} = \frac{e^Y}{2\ln(2) \cdot \frac{1}{\ln(2)}} = \frac{e^Y}{2}$$

So: The acceptance test is:

$$U \leq \frac{e^Y}{2}$$

Note:

- On $[0, \ln(2)]$, e^Y ranges from 1 to 2,

So $\frac{e^y}{2}$ ranges from 0.5 to 1.

- So every proposed point is accepted with probability between 50% and 100%.

The theoretical acceptance rate from the simulation will be close to this.

Step 4 : Implement algorithm to get 1000 samples

Algorithm in words:

1. Set `n_accepted = 0`, `n_proposed = 0`, and an empty list `samples = []`.
2. While `n_accepted < 1000`:
   1. Draw `Y` from `Uniform(0, ln(2))`.
   2. Draw `U` from `Uniform(0,1)`.
   3. If `U <= exp(Y)/2`, **accept**:
      - Append `Y` to `samples`.
      - Increase `n_accepted` by 1.
   4. Always increase `n_proposed` by 1.
3. After the loop, set:
   - `problem3_samples_accept_reject = np.array(samples)`
   - `problem3_acceptance_rate = n_accepted / n_proposed`

For the written part of the question:

- **Proposal distribution**: $g(x) = \text{Uniform}(0, \ln(2))$.
- **Reason**: same support, easy to sample, simple bound $M = 2\ln(2)$.
- **Proportion accepted**: empirical value from the code, plus you can mention that it is theoretically $1/(2\ln(2)) \approx 0.72$.

Problem 3

(4)

4. **[3p]** Explain if it is possible to sample from the density

$$f(x) = Ce^{-(x^2-2)^2}$$

using the Accept–Reject sampler (Algorithm 1 in TFDS notes) with sampling density given by the Gaussian.
Here (C) is a constant to make sure that (f) is a density, and it is between roughly 1.34 and 1.35.

We have the target density :

$$f(x) = C \cdot e^{-(x-2)}$$

where C is a normalization constant between 1.34 and 1.35.

We want to know if we can use Accept-Reject sampling with a Gaussians proposal density $g(x)$

Recall the Accept-Reject condition:

- We need a proposal density $g(x)$ and a constant M, such that

$$f(x) \leq M \cdot g(x) \text{ for all } x.$$

Then:

- Propose $Y \sim g$
- Draw $U \sim \text{Uniform}(0, 1)$
- Accept Y if:

$$U \leq \frac{f(Y)}{M \cdot g(Y)}$$

If such finite M exist, then it's possible to sample from f using the accept-reject algorithm.

Step 1: Drop the constant C:

For accept-reject, it's enough to know the

density up to a constant factor.

So: Instead of working with

$$f(x) = C \cdot e^{-(x^2-2)^2}$$

we can work with the unnormalized "kernel":

$$\hat{f}(x) = e^{-(x^2-2)^2}$$

The normalizing constant C is irrelevant for the algorithm, because:

- If $f(x) = \frac{\hat{f}(x)}{Z}$ and we know that $\hat{f}(x) \leq M \cdot g(x)$, then the normalized version is still covered.

So: we can focus on whether we can bound

$$\hat{f}(x) \leq M \cdot g(x)$$

for some finite M.

Step 2: Write down the Gaussian proposal:

Take for example the standard normal proposal:

$$g(x) = \phi(x) = \frac{1}{\sqrt{2\pi}} \cdot e^{-x^2/2}$$

This is a perfectly standard choise when the target is symmetric and bell-shaped.

Step 3: Study the tail behaviour (key idea)!

We want to check whether the ratio

$$R(x) = \frac{\hat{f}(x)}{g(x)}$$

stays bounded as x ranges over all real numbers

If $\sup_x R(x) < \infty$, then we can take

$$M = \sup_x \frac{\hat{f}(x)}{g(x)}$$

and use that as our envelope constant.

Compute the ratio:

$$R(x) = \frac{e^{-(x^2-2)^2}}{\frac{1}{\sqrt{2\pi}} \cdot e^{-x^2/2}} = \sqrt{2\pi} \cdot e^{\left(-(x^2-2)^2 + \frac{x^2}{2}\right)}$$

Define the exponent $\psi(x) = -(x^2-2)^2 + \frac{x^2}{2}$

Expand $(x^2-2)^2 \Rightarrow x^4 - 4x^2 + 4$

So: $\psi(x) = -(x^4 - 4x^2 + 4) + \frac{x^2}{2} = -x^4 + \frac{9}{2}x^2 - 4$

as $|x| \to \infty$, the dominant term is $-x^4$, so:

$$\psi(x) \to -\infty$$

and therefore

$$R(x) = \sqrt{2\pi} \cdot e^{\psi(x)} \to 0 \quad \text{as } |x| \to \infty$$

So:

- $R(x)$ is a continuous function on $\mathbb{R}$.
- As $|x| \to \infty$, $R(x) \to 0$.

Therefore $R(x)$ attains a finite maximum at some finite x. Call this maximum value M_0.

$$M_0 = \sup_{x \in \mathbb{R}} R(x) < \infty.$$

This means there exists a finite constant M_0 such that:

$$\hat{f}(x) \leq M_0 \cdot g(x) \quad \text{for all } x.$$

Because $f(x) = C^{-1} \cdot \hat{f}(x)$ is just a normalized version of $\hat{f}(x)$, we can define a (possibly different) finite constant M for f, but the important part is: a finite M exists.

This shows that yes, it is possible to use accept-reject with a Gaussian proposal

Intuition:

- The target density decays like e^{-x^4} in the tails.

- A gaussian decays like e^{-x^2} in the tails.
- Since e^{-x^4} decays faster than e^{-x^2}, we can always cover f by a scaled Gaussian.

Step 4: Accept-Reject algorithm for this f:

Using the unnormalized target $\hat{f}(x) = e^{-(x^2-2)^2}$ and Gaussian proposal $g(x) = \phi(x) = \frac{1}{\sqrt{2\pi}} \cdot e^{-x^2/2}$:

1) Precompute (numerically) approximate bound M for $\frac{\hat{f}(x)}{g(x)}$

2) To generate one sample:

   1) Draw $Y \sim N(0,1)$

   2) Draw $U \sim \text{Uniform}(0,1)$

   3) Accept Y if:

$$U \leq \frac{\hat{f}(Y)}{M \cdot g(Y)}$$

   4) otherwise, reject and repeat.