Advanced Numerical Analysis

Lecture 3

Spring 2025

Fabio Matti

Quiz from Exercise Set 2

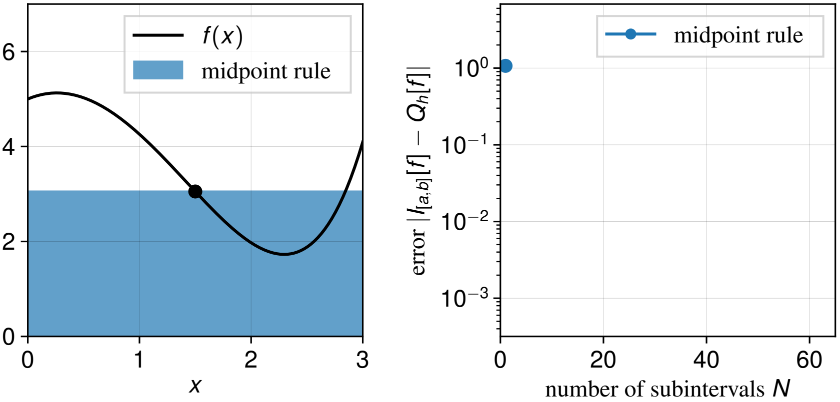
a) Consider the harmonic series $\sum_{k=1}^{\infty} \frac{1}{k}$, which is known to diverge. When attempting to compute the partial sum $1 + 1/2 + 1/3 + \ldots + 1/n$ (from the smallest to the largest) in double precision, what will happen as $n \to \infty$?

- ◯ The computed partial sums will overflow.
- ◯ The computed partial sums will stagnate (“converge”) to ≈ 34.
- ◯ The computed partial sum will stagnate (“converge”) to $\approx 2 \times 10^{16}$.
- ◯ The computed partial sum will stagnate (“converge”) to $\approx 10^{300}$.

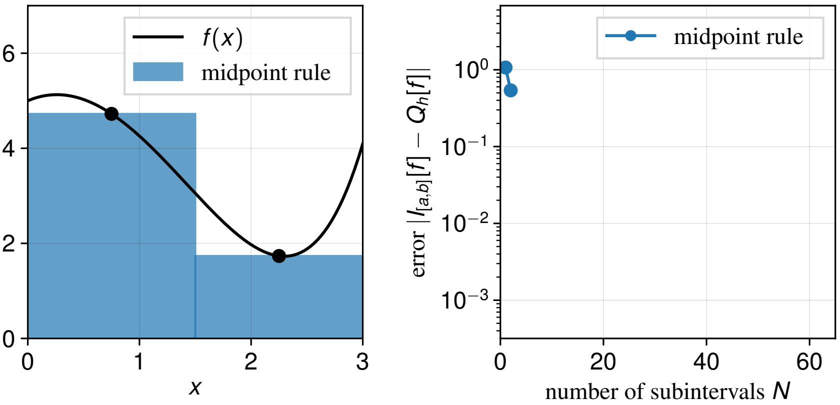
b) Consider the same question for the alternating harmonic series $\sum_{k=1}^{\infty} (-1)^{k-1} \frac{1}{k}$, which is known to converge to $\log(2)$.

- ◯ The computed partial sums will overflow.
- ◯ The computed partial sums will stagnate (“converge”) to $\approx \log(2)$.
- ◯ The computed partial sums will underflow.
- ◯ The computed partial sum will stagnate (“converge”) to ≈ 0.

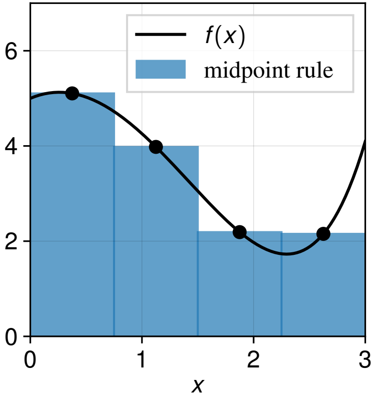
Midpoint rule

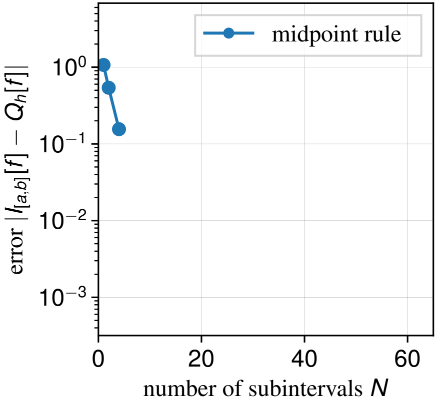


Midpoint rule

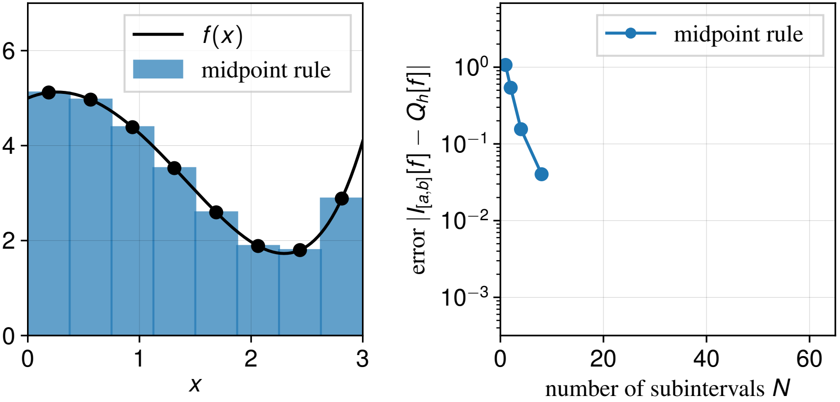


Midpoint rule

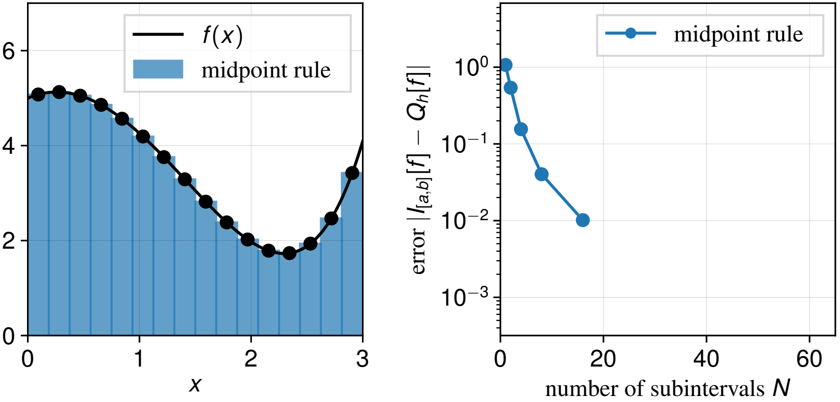




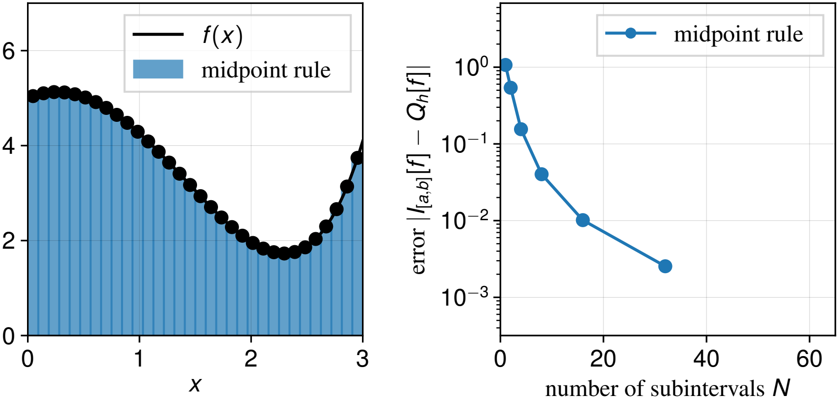
Midpoint rule



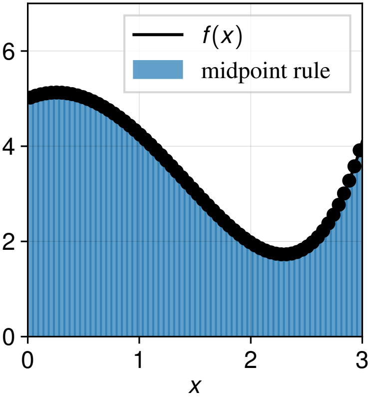
Midpoint rule

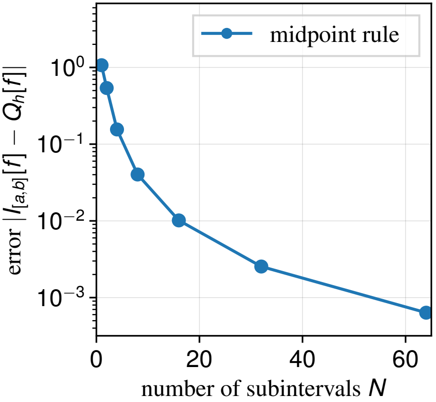


Midpoint rule

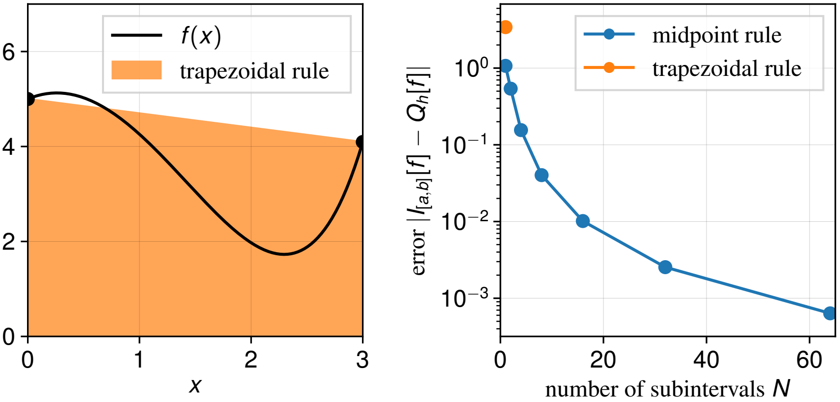


Midpoint rule

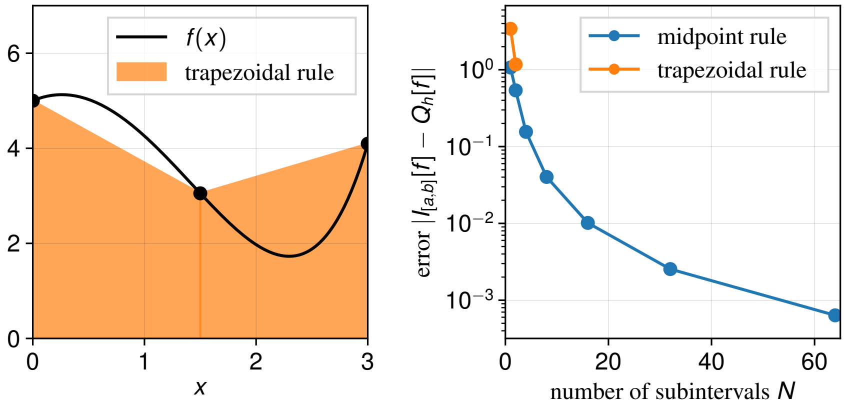




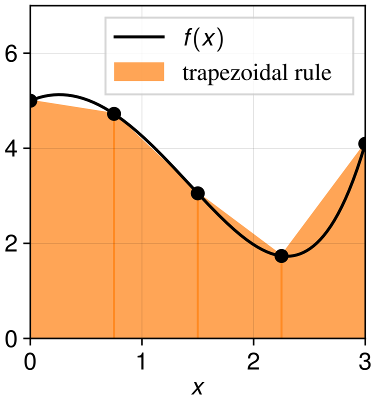
Trapezoidal rule

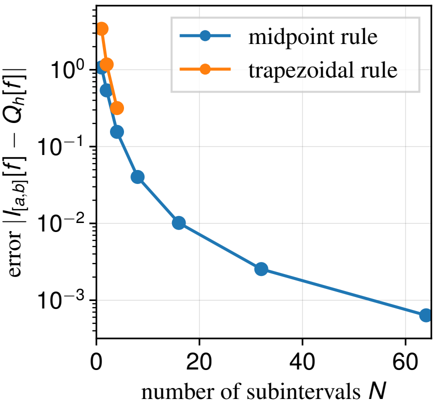


Trapezoidal rule

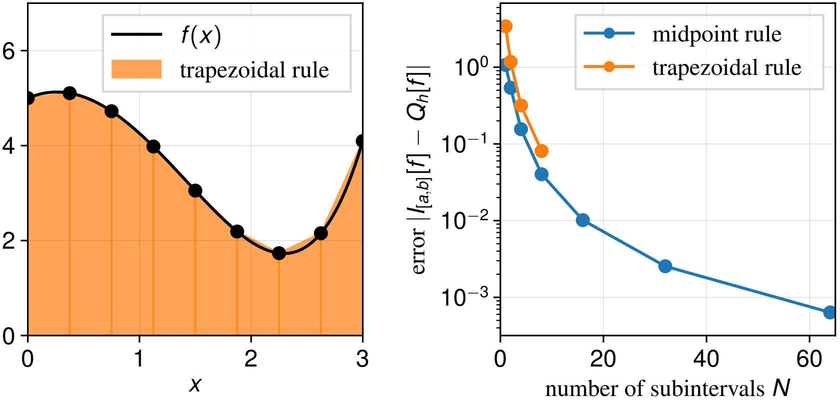


Trapezoidal rule

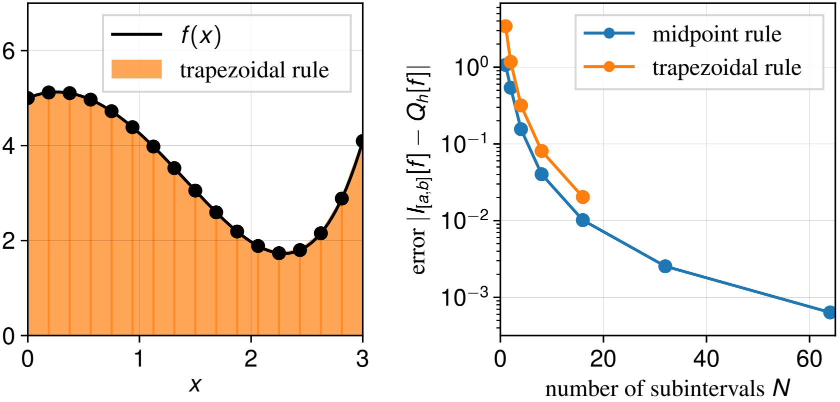




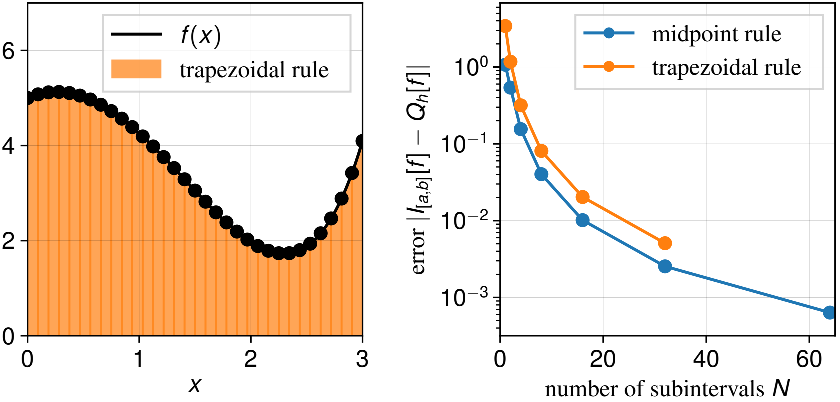
Trapezoidal rule



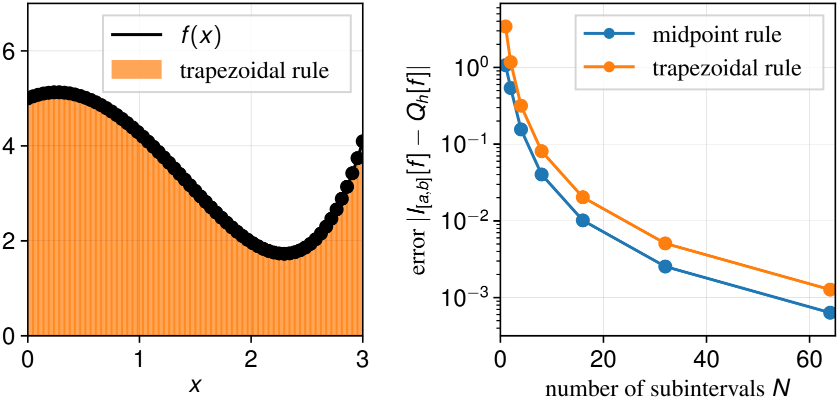
Trapezoidal rule



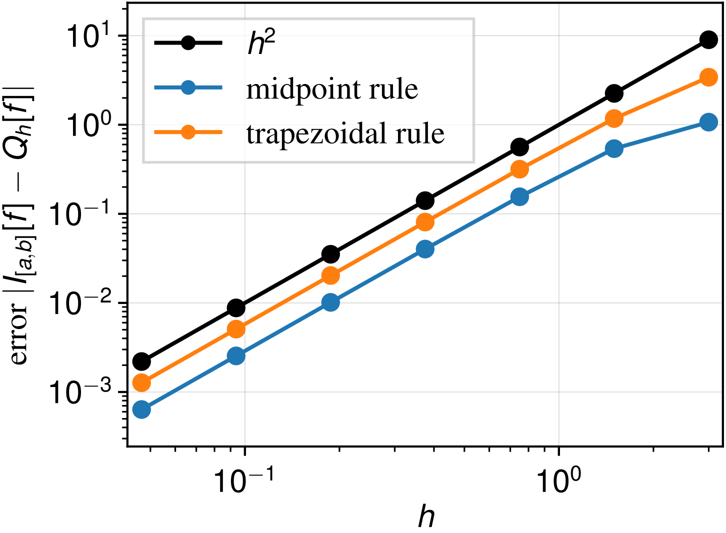
Trapezoidal rule



Trapezoidal rule



Convergence



Vandermonde matrix

Coefficients can be obtained by solving linear system involving the Vandermonde matrix

$$V = \begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{pmatrix}$$

- Is numerically problematic
- Still the standard way in Python (NumPy)

```
def polyfit(x, y, deg, rcond=None, full=False, w=None, cov=False):
    # set up least squares equation for powers of x
    lhs = vander(x, order)
    rhs = y

    # apply weighting
    if w is not None:
        w = NX.asarray(w) + 0.0
        if w.ndim != 1:
            raise TypeError("expected a 1-d array for weights")
        if w.shape[0] != y.shape[0]:
            raise TypeError("expected w and y to have the same length")
        lhs *= w[:, NX.newaxis]
        if rhs.ndim == 2:
            rhs *= w[:, NX.newaxis]
        else:
            rhs *= w

    # scale lhs to improve condition number and solve
    scale = NX.sqrt((lhs*lhs).sum(axis=0))
    lhs /= scale
    c, resids, rank, s = lstsq(lhs, rhs, rcond)
    c = (c.T/scale).T  # broadcast scale coefficients
```

$\longrightarrow$ Vandermonde matrix (for `lhs = vander(x, order)`)

$\longrightarrow V^{-1}x$ (for `lstsq(lhs, rhs, rcond)`)

Lagrange basis polynomials