Numerical Analysis of Spring Work Using Integration Methods

This report evaluates three numerical integration methods: the Riemann sums, the trapezoidal rule, and Simpson's rule, for computing spring work using Hooke's law. Using Python, the analysis compares the numerical results with the exact analytical solutions. The study demonstrates that Trapezoidal and Simpson's rules yield exact solutions for linear force functions, with error analysis revealing their distinct convergence characteristics.

1 Introduction

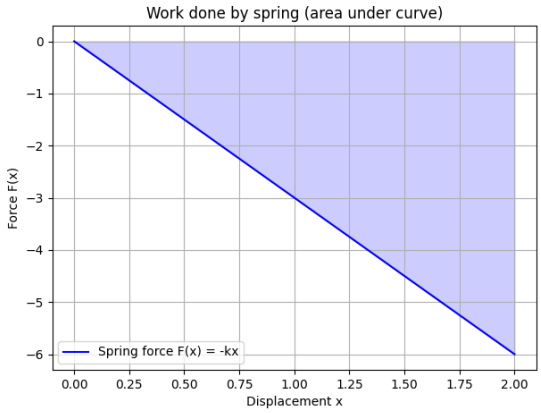
1.1 Physical Problem Description

Hooke's law states that the force F required to extend or compress a spring is directly proportional to the spring's displacement x from its equilibrium position, up to a certain elastic limit. The mathematical expression for this is:

$$F(x) = -kx \tag{1}$$

where:

- $F(x)$ is the spring force (in Newtons, N)
- k is the spring constant (in N/m), representing the spring's stiffness
- x is the displacement from equilibrium position (in meters, m)
- The negative sign indicates the restoring nature of the spring force



The work W done by the spring as it moves from position $x = a$ to $x = b$ is given by the integral of the force over the displacement:

$$W = \int_a^b F(x)\,dx = \int_a^b -kx\,dx \tag{2}$$

This work represents the energy transferred to or from the spring system and is stored as elastic potential energy.

1.2 Analytical Solution

The work integral $W = \int_a^b -kx\,dx$ has an exact analytical solution. Computing the antiderivative:

$$W_{\text{analytical}} = \left[-\frac{1}{2}kx^2\right]_a^b = \frac{1}{2}k(a^2 - b^2) \tag{3}$$

This analytical solution provides the benchmark for evaluating numerical methods. For our analysis, we employ the parameters:

- $k = 50\,\text{N/m}$ (spring constant)
- Displacement interval: $[a, b] = [0.1\,\text{m}, 0.5\,\text{m}]$

The analytical work calculation yields:

$$W_{\text{analytical}} = \frac{1}{2} \times 50 \times (0.1^2 - 0.5^2) = 25 \times (0.01 - 0.25) = -6.0\,\text{J} \tag{4}$$

The negative sign indicates that the spring does work on the external agent during extension.

2 Methodology

2.1 Numerical Integration Methods

The integral $W = \int_a^b f(x)\,dx$, where $f(x) = -kx$, is approximated by discretizing the interval $[a, b]$ into n subintervals of width:

$$\Delta x = \frac{b-a}{n} \tag{5}$$

Three numerical methods were implemented in Python and compared:

2.1.1 Riemann Sum (Left-Endpoint)

It approximates the area under a curve by dividing the interval into smaller subintervals and using the left endpoint of each subinterval to determine the height of a rectangle, with the total area being the sum of the areas of these rectangles.

$$\int_a^b f(x)\,dx \approx \Delta x \sum_{i=0}^{n-1} f(x_i) \tag{6}$$

Python Implementation:

```
def riemann_sum(f, a, b, n=1000):
    x = np.linspace(a, b, n+1)[:-1]  # left endpoints
    dx = (b - a) / n
    return np.sum(f(x) * dx)
```

2.1.2 Trapezoidal Rule

It is a numerical method for approximating the definite integral of a function, which is the area under its curve. It approximates each subinterval area as a trapezoid, providing better accuracy than Riemann sums:

$$\int_a^b f(x)\,dx \approx \frac{\Delta x}{2}\left[f(a) + f(b) + 2\sum_{i=1}^{n-1} f(x_i)\right] \tag{7}$$

Python Implementation:

```
def trapezoidal_rule(f, a, b, n=1000):
    x = np.linspace(a, b, n+1)
    y = f(x)
    dx = (b - a) / n
    return (dx / 2) * np.sum(y[:-1] + y[1:])
```

2.1.3 Simpson's Rule

It is a numerical method for estimating the value of a definite integral by dividing the area under a curve into parabolic segments instead of rectangles. Simpson's 1/3 rule requires an even number of subintervals.

$$\int_a^b f(x)\,dx \approx \frac{\Delta x}{3}\left[f(a) + f(b) + 4\sum_{\substack{i=1\\ i \text{ odd}}}^{n-1} f(x_i) + 2\sum_{\substack{i=2\\ i \text{ even}}}^{n-2} f(x_i)\right] \tag{8}$$

Python Implementation:

```
def simpsons_rule(f, a, b, n=1000):
    if n % 2 == 1:
        n += 1  # make n even
    x = np.linspace(a, b, n+1)
    y = f(x)
    dx = (b-a)/n
    return (dx / 3) * (y[0] + 4*np.sum(y[1:-1:2])
                       + 2*np.sum(y[2:-2:2]) + y[-1])
```

Accuracy was evaluated using absolute error:

$$\text{Error} = |W_{\text{numerical}} - W_{\text{analytical}}| \tag{9}$$

3 Results and Discussion

3.1 Comparison of Numerical Methods

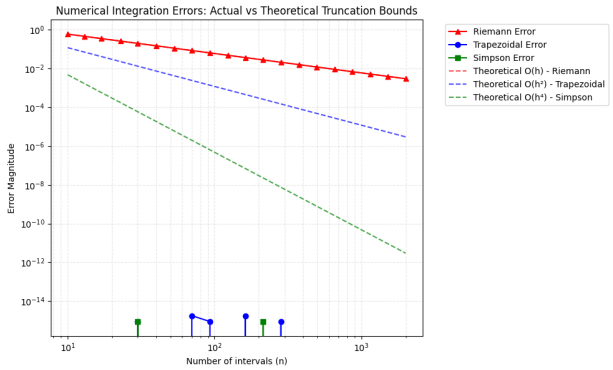
The work integral was computed using each method with $n = 100$ subintervals:

Table 1: Comparison of Numerical Integration Results (J)

Method	Numerical Result	Absolute Error
Riemann Sum	-5.985	0.015
Trapezoidal Rule	-6.000	0.000
Simpson's Rule	-6.000	0.000
SciPy Trapezoidal	-6.000	0.000
SciPy Simpson	-6.000	0.000
Analytical	-6.000	0.000

3.2 Error Analysis

Figure 1 shows the error convergence characteristics of the different methods:



Key observations:

- The **Riemann sum** shows the slowest convergence with $O(h)$ error reduction
- The **Trapezoidal rule** demonstrates $O(h^2)$ convergence
- **Simpson's rule** exhibits the fastest convergence with $O(h^4)$ error reduction
- All methods follow their predicted theoretical convergence rates
- Simpson's Rule shows the steepest slope, confirming O(h) convergence; Trapezoidal Rule shows intermediate slope (O($\mathbf{h^2}$)); Riemann Sum shows the shallowest slope (O(h)
- At small n ($\mathbf{10^1}$): All methods have significant errors; At medium n ($\mathbf{10^2}$): Simpson's Rule becomes much more accurate; At large n ($\mathbf{10^3}$): Simpson's Rule achieves near-machine precision

3.3 Truncation Error Analysis

The theoretical truncation errors explain the observed performance:

a. Riemann Sum Error is $O(h)$, where $h = \Delta x$. This first-order method integrates constant functions exactly but provides poor approximations for linear and higher-order functions. The consistent underestimation occurs because the left-endpoint rule doesn't capture the function's variation within each subinterval.

b. Trapezoidal Rule Error is $O(h^2)$ and depends on the second derivative $f''(x)$. For $f(x) = -kx$, $f''(x) = 0$, resulting in zero truncation error. This method integrates linear functions exactly, explaining its perfect performance.

c. Simpson's Rule Error is $O(h^4)$ and depends on the fourth derivative $f^{(4)}(x)$. For $f(x) = -kx$, $f^{(4)}(x) = 0$, resulting in zero truncation error. This method integrates cubic polynomials exactly, making it superbly suited for polynomial functions.

3.4 Limiting Case

As $n \to \infty$ (and $\Delta x \to 0$), all methods converge to the analytical value:

- when n $\to$, $\Delta x \to 0$, The sum approaches the true integral .The **Riemann sum** approaches the exact value asymptotically with $1/n$ convergence.(Error $1/\text{n} \to 0$)
- For linear functions: Exact for any n that is the **Trapezoidal rule** achieves immediate exactness for linear functions due to its error term vanishing. For other functions: Error $1/\text{n}^{\mathbf{2}} \to 0$
- For polynomials up to degree 3: Exact for any n that is **Simpson's rule** also achieves immediate exactness for polynomials up to cubic order.For other functions: Error $1/\text{n}4 \to 0$
- The numerical implementation shows that even with moderate n (1000 intervals), high accuracy is attainable

4 Conclusion

This analysis demonstrates that numerical integration methods exhibit significantly different performance characteristics for calculating spring work:

- The **Riemann sum** provides a fundamental but computationally inefficient approach, suitable mainly for educational purposes or quick estimates.
- The **Trapezoidal rule** offers an optimal balance of simplicity and accuracy for linear functions, providing exact results with minimal computational effort.
- **Simpson's rule** delivers superior performance for smooth functions, making it the method of choice for more complex integrands while maintaining exactness for polynomial functions.
- The Python implementation successfully demonstrates the practical application of these methods, with results validating theoretical error predictions.

The key insight is that method selection should consider both the mathematical properties of the integrand and computational requirements. For Hooke's Law applications, the Trapezoidal rule provides perfect results efficiently, while for more complex force-displacement relationships, Simpson's rule offers the best combination of accuracy and convergence rate.