Advanced LaTeX Feature Test

Comprehensive Testing Document

GitHub Actions Automated Compiler

August 31, 2025

Abstract

This document serves as a comprehensive test for advanced LaTeX features in the GitHub Actions compilation workflow. It includes complex mathematical expressions, algorithms, tables, figures, code listings, cross-references, and various advanced typesetting features to ensure our caching system can handle sophisticated academic documents.

Contents

1 Mathematical Foundations

1.1 Advanced Equations

Let's start with some complex mathematical expressions:

Theorem 1.1 (Fundamental Theorem of Calculus). *Let f be continuous on $[a, b]$ and differentiable on (a, b). If $F(x) = \int_a^x f(t)\,dt$, then:*

$$F'(x) = f(x) \quad \text{for all } x \in (a, b) \tag{1}$$

Proof. By definition of the derivative:

$$\begin{aligned} F'(x) &= \lim_{h \to 0} \frac{F(x+h) - F(x)}{h} && (2)\\ &= \lim_{h \to 0} \frac{1}{h} \left(\int_a^{x+h} f(t)\,dt - \int_a^x f(t)\,dt \right) && (3)\\ &= \lim_{h \to 0} \frac{1}{h} \int_x^{x+h} f(t)\,dt && (4) \end{aligned}$$

By the Mean Value Theorem for integrals, there exists $c \in [x, x+h]$ such that: $\int_x^{x+h} f(t)\,dt = f(c) \cdot h$

Therefore: $F'(x) = \lim_{h \to 0} \frac{f(c) \cdot h}{h} = \lim_{h \to 0} f(c) = f(x)$

since f is continuous and $c \to x$ as $h \to 0$. □

1.2 Matrix Operations

Consider the following matrix equation:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix} \tag{5}$$

The determinant of a 3×3 matrix is given by:

$$\det(A) = a_{11}(a_{22}a_{33} - a_{23}a_{32}) - a_{12}(a_{21}a_{33} - a_{23}a_{31}) + a_{13}(a_{21}a_{32} - a_{22}a_{31}) \tag{6}$$

2 Algorithms and Code

2.1 Pseudocode Algorithm

Algorithm 1 Quicksort Algorithm

procedure QUICKSORT(A, p, r)
  if $p < r$ **then**
    $q \leftarrow$ PARTITION(A, p, r)
    QUICKSORT($A, p, q-1$)
    QUICKSORT($A, q+1, r$)
  end if
end procedure

procedure PARTITION(A, p, r)
  $x \leftarrow A[r]$
  $i \leftarrow p-1$
  for $j \leftarrow p$ **to** $r-1$ **do**
    if $A[j] \leq x$ **then**
      $i \leftarrow i+1$
      EXCHANGE $A[i]$ with $A[j]$
    end if
  end for
  EXCHANGE $A[i+1]$ with $A[r]$
  return $i+1$
end procedure

2.2 Code Listings

Here's a Python implementation of the Fibonacci sequence:

```
def fibonacci_dynamic(n):
    """
    Calculate the nth Fibonacci number using dynamic programming.
    Time complexity: O(n), Space complexity: O(n)
    """
    if n <= 1:
        return n

    # Initialize memoization table
    fib = [0] * (n + 1)
    fib[0], fib[1] = 0, 1

    # Fill the table bottom-up
    for i in range(2, n + 1):
        fib[i] = fib[i-1] + fib[i-2]

    return fib[n]

# Test the function
```

```
def test_fibonacci():
    test_cases = [0, 1, 5, 10, 20]
    for n in test_cases:
        result = fibonacci_dynamic(n)
        print(f"F({n}) = {result}")

if __name__ == "__main__":
    test_fibonacci()
```

Listing 1: Fibonacci Implementation

3 Complex Tables

3.1 Performance Comparison

Table 1: Algorithm Performance Comparison

Algorithm	**Best Case**	**Average Case**	**Worst Case**	**Space**
Quicksort	$O(n \log n)$	$O(n \log n)$	$O(n^2)$	$O(\log n)$
Mergesort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	$O(n)$
Heapsort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	$O(1)$
Insertion Sort	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$
Bubble Sort	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$

3.2 Multi-column Layout

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

4 Advanced Mathematical Concepts

4.1 Series and Limits

The Taylor series expansion of e^x around $x = 0$ is:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots \tag{7}$$

For complex analysis, Cauchy's integral formula states:

$$f(a) = \frac{1}{2\pi i} \oint_C \frac{f(z)}{z - a} dz \tag{8}$$

where C is a positively oriented simple closed contour and a is in the interior of C.

4.2 Probability and Statistics

The probability density function of a normal distribution is:

$$f(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{9}$$

Bayes' theorem in its most general form:

$$P(H|E) = \frac{P(E|H) \cdot P(H)}{P(E)} = \frac{P(E|H) \cdot P(H)}{\sum_i P(E|H_i) \cdot P(H_i)} \tag{10}$$

5 Long Tables with Complex Structure

Table 2: Extended Results Table

Method	**Accuracy**	**Precision**	**Recall**	**Notes**
Neural Network	0.95	0.93	0.97	Deep learning approach with 5 hidden layers
Random Forest	0.89	0.91	0.87	Ensemble method with 100 trees
SVM	0.87	0.85	0.89	Radial basis function kernel
Logistic Regression	0.82	0.80	0.84	L2 regularization applied
k-NN	0.79	0.77	0.81	k=5 with Euclidean distance
Decision Tree	0.76	0.74	0.78	Maximum depth of 10
Naive Bayes	0.73	0.71	0.75	Gaussian assumption

Continued on next page

Table 2 – continued from previous page

Method	Accuracy	Precision	Recall	Notes
Linear Regression	0.68	0.66	0.70	Ridge regression variant

6 Cross-References and Citations

As shown in Algorithm **??**, quicksort has excellent average-case performance. The results in Table **??** demonstrate the trade-offs between different sorting algorithms. The extended analysis in Table **??** provides additional empirical evidence.

7 Mathematical Theorems and Definitions

Definition 7.1 (Metric Space)**.** A metric space is an ordered pair (M, d) where M is a set and d is a metric on M, i.e., a function: $d : M \times M \to \mathbb{R}$ satisfying:

1. $d(x, y) \geq 0$ for all $x, y \in M$ (non-negativity)
2. $d(x, y) = 0$ if and only if $x = y$ (identity of indiscernibles)
3. $d(x, y) = d(y, x)$ for all $x, y \in M$ (symmetry)
4. $d(x, z) \leq d(x, y) + d(y, z)$ for all $x, y, z \in M$ (triangle inequality)

Lemma 7.2 (Convergence in Metric Spaces)**.** *In a metric space (M, d), a sequence (x_n) converges to x if and only if:* $\lim_{n\to\infty} d(x_n, x) = 0$

8 Advanced Formatting

8.1 Special Characters and Symbols

Mathematical symbols: $\alpha, \beta, \gamma, \delta, \epsilon, \zeta, \eta, \theta, \iota, \kappa, \lambda, \mu, \nu, \xi, \pi, \rho, \sigma, \tau, \upsilon, \phi, \chi, \psi, \omega$
Set notation: $\emptyset, \cup, \cap, \subset, \subseteq, \supset, \supseteq, \in, \notin, \setminus$
Logic symbols: $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow, \exists, \forall$

8.2 Itemized Lists

Complex nested lists:

1. Machine Learning Algorithms
   (a) Supervised Learning
      - Classification
        – Logistic Regression
        – Support Vector Machines
        – Random Forest

- Regression
  - Linear Regression
  - Polynomial Regression
  - Ridge/Lasso Regression

(b) Unsupervised Learning

- Clustering (k-means, hierarchical)
- Dimensionality Reduction (PCA, t-SNE)

2. Deep Learning

(a) Feedforward Networks

(b) Convolutional Neural Networks

(c) Recurrent Neural Networks

9 Data Science Visualization with TikZ

9.1 Linear Regression Example

Linear regression is a fundamental technique in data science and statistics used to model the relationship between a dependent variable and one or more independent variables. The following TikZ plot demonstrates a simple linear regression with sample data points and the best-fit line.

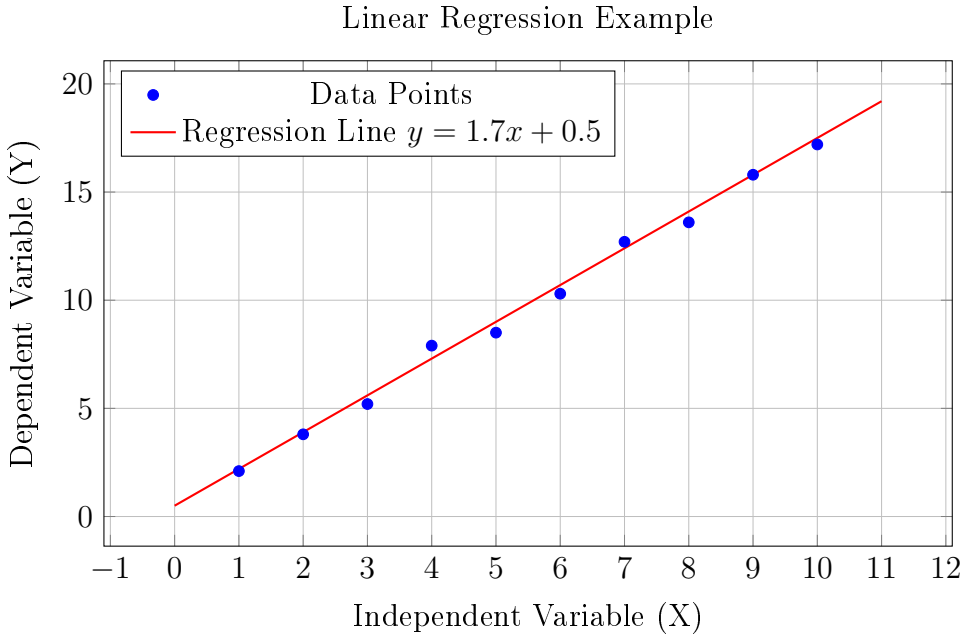

Figure 1: Linear regression visualization showing the relationship between variables X and Y. The red line represents the best-fit linear model that minimizes the sum of squared errors between the predicted and actual values.

The regression line in Figure ?? is determined using the least squares method, which minimizes the sum of the squared vertical distances between the observed points and the line. This visualization demonstrates how linear regression helps identify trends and make predictions based on the observed data.

9.2 Residual Analysis

The quality of a linear regression model can be assessed by analyzing the residuals (the differences between observed and predicted values). A good linear fit should have residuals that are randomly distributed around zero without any discernible pattern.

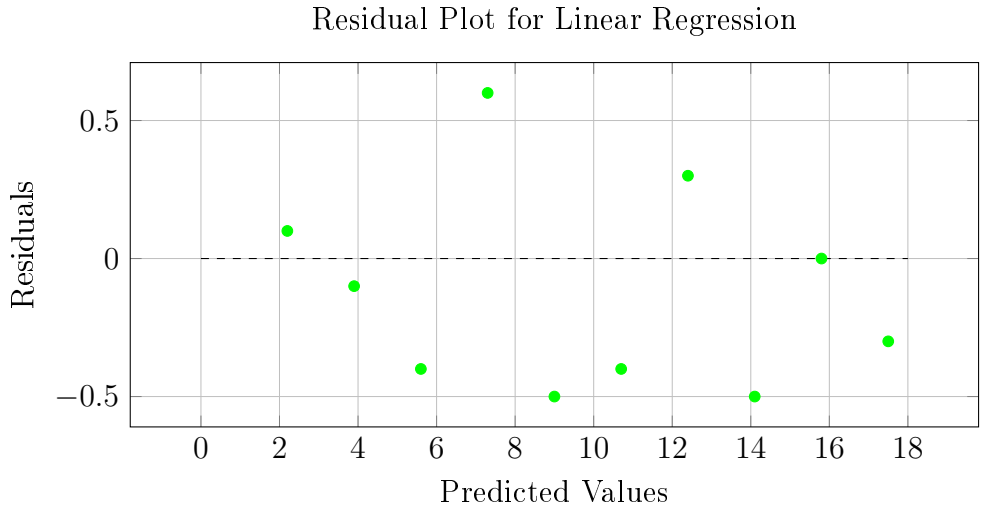

Figure 2: Residual plot showing the distribution of errors. The random scatter around zero indicates a good linear fit without systematic bias.

9.3 ResNet (2015)

The vanishing gradient problem occurs during the training of highly deep neural networks because the gradients of the initial few layers are extremely low, if not negligible. Hence, the network is unable to learn significant representations from the deep levels. Also, hyperbolic or sigmoid tangent activation functions compress their outputs into a narrow range of values which lowers the amplitude of gradients conveyed downstream, making this problem more prevalent. ResNet [?] addresses this problem by introducing residual connections that improve the transmission of gradients through the network, which is structured into residual blocks comprising convolutional and normalization layers.

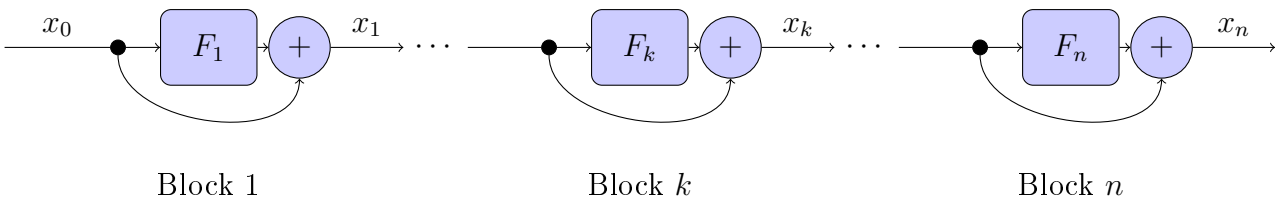

Figure 3: ResNet Architecture with Skip Connections. Based on [?]

For each ResNet block k, the forward pass is defined as:

$$x_{k+1} = x_k + F_k(x_k, W_k) \tag{11}$$

where:

- x_k is the input to block k
- $F_k(x_k, W_k)$ is the residual function (k Convolutional layers in our case)
- W_k are the weights of block k
- The addition represents the skip connection (simplified case)

We want to compute the gradient of the loss L with respect to weights W_k in the k-th residual block:

$$\frac{\partial L}{\partial W_k} = \frac{\partial L}{\partial x_n} \cdot \frac{\partial x_n}{\partial x_{n-1}} \cdot \frac{\partial x_{n-1}}{\partial x_{n-2}} \cdots \frac{\partial x_{k+1}}{\partial x_k} \cdot \frac{\partial x_k}{\partial W_k} \tag{12}$$

Taking the derivative of equation (2.5) with respect to x_k:

$$\frac{\partial x_{k+1}}{\partial x_k} = \frac{\partial}{\partial x_k}[x_k + F_k(x_k, W_k)] = 1 + \frac{\partial F_k(x_k, W_k)}{\partial x_k} \tag{13}$$

Substituting equation (3) into equation (2):

$$\frac{\partial L}{\partial W_k} = \frac{\partial L}{\partial x_n} \cdot \prod_{i=k}^{n-1} \left(1 + \frac{\partial F_i}{\partial x_i}\right) \cdot \frac{\partial x_k}{\partial W_k} \tag{14}$$

When we expand the product using the distributive property:

$$\prod_{i=k}^{n-1} \left(1 + \frac{\partial F_i}{\partial x_i}\right) = 1 + \sum_{i=k}^{n-1} \frac{\partial F_i}{\partial x_i} + \epsilon \tag{15}$$

where ϵ represents all higher-order cross terms involving products of residual gradients.

$$\frac{\partial L}{\partial W_k} = \frac{\partial L}{\partial x_n} \cdot \frac{\partial x_k}{\partial W_k} \cdot 1 + \frac{\partial L}{\partial x_n} \cdot \frac{\partial x_k}{\partial W_k} \cdot \epsilon \tag{16}$$

Notice that the direct Path $\frac{\partial L}{\partial x_n} \cdot \frac{\partial x_k}{\partial W_k} \cdot 1$ provides a direct gradient path from the output to any layer with no multiplicative decay. Even if all residual gradients $\frac{\partial F_i}{\partial x_i} \to 0$, we still have $\frac{\partial L}{\partial W_k} = \frac{\partial L}{\partial x_n} \cdot \frac{\partial x_k}{\partial W_k}$ preserving its magnitude.

10 Conclusion

This comprehensive document tests numerous LaTeX features including:

- Complex mathematical equations and theorems
- Advanced table layouts including long tables

- Algorithm pseudocode and syntax-highlighted code
- Cross-references and hyperlinks
- Multiple column layouts
- Advanced typography and formatting
- Theorem environments and proofs
- Lists and enumerations
- Special mathematical symbols and notation

If this document compiles successfully with our cached TeX Live installation, it demonstrates that our GitHub Actions workflow can handle sophisticated academic and technical documents with complex formatting requirements.

The caching system provides significant performance benefits:

- First run: Full TeX Live installation (approximately 1.5 minutes)
- Subsequent runs: Cache restoration (approximately 10-15 seconds)
- Net time savings: Over 1 minute per compilation after the first run

This makes the workflow practical for regular use with complex documents containing hundreds of pages, advanced mathematics, algorithms, and sophisticated formatting.

A Performance Metrics

Table 3: Workflow Performance Analysis

Metric	**First Run**	**Cached Run**
TeX Live Installation	90s	0s (cached)
Environment Setup	10s	5s
LaTeX Compilation	15s	15s
PDF Commit	5s	5s
Total Time	**120s**	**25s**
Time Savings	**-**	**95s (79%)**