DEMO OF THE CSTHM PACKAGE (FANCY)

AGNI DATTA

Contents

1. Introduction

Welcome to the comprehensive demonstration of the `csthm` package! This document serves as both a showcase and a practical guide for mathematical writing in LaTeX. The `csthm` package provides over 40 different environments organized into four main categories, each serving specific purposes in mathematical exposition.

The package offers three distinct visual styles: `fancy` (used in this demo), `normal`, and `oldschool`. Each style is designed to enhance readability while maintaining mathematical rigor. Throughout this document, you'll see examples of every environment with detailed explanations of their intended use cases.

Mathematical writing requires careful organization of ideas, and the right environment can make the difference between a confusing exposition and a clear, engaging presentation. Let's explore each category systematically.

2. Theorem-like Environments

Theorem-like environments are the backbone of mathematical exposition. They present formal mathematical statements that have been proven or are accepted as true. These environments typically appear in a distinctive style to emphasize their importance and are usually numbered for easy reference.

2.1. Core Theorem Environments.

The `theorem` environment is perhaps the most important in mathematical writing. Use it for major results, fundamental principles, or any statement that represents a significant mathematical truth. Theorems are typically the culmination of mathematical reasoning and often serve as stepping stones to more complex results.

Theorem 2.1 (Fundamental Theorem of Arithmetic). Every integer greater than 1 is either prime or can be uniquely factored into prime numbers.

Notice how the theorem environment creates a visually distinct box with automatic numbering. The optional argument in square brackets provides a name or description, which is particularly useful for well-known results. This theorem, for instance, is fundamental to number theory and deserves special recognition.

Sometimes you may want to present a theorem without numbering, perhaps when restating a result for emphasis or when the statement doesn't warrant a formal number. The starred version accomplishes this:

> Theorem (Unnumbered Version). This is an unnumbered theorem for comparison. Use starred versions when you want the visual emphasis of a theorem environment without the formal numbering system.

The `lemma` environment is perfect for auxiliary results that support main theorems. Lemmas are often technical stepping stones that, while important, are not the primary focus of your exposition. They're typically proven before the main theorem and then used in that proof.

> Lemma 2.1. If p is prime and $p|ab$, then $p|a$ or $p|b$.

This lemma about prime divisibility is a classic example. While it's a significant result in its own right, it's often used as a tool to prove larger theorems about prime factorization. The lemma environment signals to readers that this is an important building block.

A `corollary` represents a direct consequence of a theorem or lemma. These are results that follow easily from what has already been established. Corollaries often reveal interesting implications of major theorems that might not be immediately obvious.

> Corollary 2.1. There are infinitely many prime numbers.

This famous corollary follows from Euclid's proof technique and demonstrates how major results often have elegant consequences. The corollary environment helps readers understand the hierarchical structure of mathematical results.

The `proposition` environment is used for statements that are significant but perhaps not as central as theorems. Propositions often present interesting properties or intermediate results that are worth highlighting but don't quite reach the level of a theorem.

> Proposition 2.1. The square root of 2 is irrational.

This classic proposition about $\sqrt{2}$ demonstrates a fundamental property of real numbers. While extremely important, it's often presented as a proposition rather than a theorem, depending on the context and emphasis of your exposition.

Of course, formal mathematical statements require rigorous justification. The `proof` environment provides the perfect framework for presenting logical arguments. Proofs should be clear, complete, and well-structured.

Proof. Assume $\sqrt{2} = \frac{p}{q}$ where $\gcd(p, q) = 1$. Then $2q^2 = p^2$, so p^2 is even, hence p is even. Let $p = 2k$, then $2q^2 = 4k^2$, so $q^2 = 2k^2$. This means q is also even, contradicting $\gcd(p, q) = 1$. ■

Notice how the proof environment automatically includes a tombstone symbol (■) at the end, providing clear visual closure to the argument. The proof demonstrates the classic technique of proof by contradiction.

2.2. **Extended Theorem Environments.** Beyond the core theorem environments, the `csthm` package provides specialized environments for different types of mathematical statements. These environments help you communicate the nature and certainty level of your mathematical claims.

An `assertion` is used when you want to state something confidently but perhaps without a complete formal proof. Assertions are particularly useful in applied mathematics or when dealing with computational results.

> Assertion 2.1. Machine learning algorithms can approximate any continuous function with arbitrary precision given sufficient data and computational resources.

This assertion about machine learning capabilities illustrates how this environment works well for statements that are generally accepted but might be based on empirical evidence or informal arguments rather than rigorous mathematical proof.

The `assumption` environment is crucial for setting up mathematical contexts. Many mathematical results depend on specific conditions or hypotheses, and the assumption environment makes these dependencies explicit.

Assumption 2.1. The input data is independently and identically distributed, and the underlying distribution has finite variance.

Statistical and probabilistic results often depend heavily on assumptions about data distribution. Making these assumptions explicit helps readers understand the scope and limitations of subsequent results.

An `axiom` represents a fundamental principle that is accepted without proof. Axioms form the foundation of mathematical systems and are typically few in number but profound in their implications.

Axiom 2.1 (Axiom of Choice). For any collection of non-empty sets, there exists a choice function that selects exactly one element from each set.

The Axiom of Choice is one of the most famous and controversial axioms in mathematics. Its inclusion as an axiom rather than a theorem emphasizes its foundational role in set theory and its acceptance as a basic principle.

A `claim` is useful for intermediate statements in proofs or for assertions that you plan to justify later. Claims help break down complex arguments into manageable pieces.

Claim 2.1. The proposed sorting algorithm runs in $\mathcal{O}(n \log n)$ time in the worst case.

Claims are particularly useful in algorithm analysis, where you might want to state performance characteristics before providing the detailed analysis. The claim environment signals that justification will follow.

The `conclusion` environment is perfect for summarizing results or drawing final inferences from a series of arguments. Conclusions help readers understand the ultimate significance of your mathematical exposition.

Conclusion 2.1. Therefore, the proposed method is both computationally efficient and theoretically sound, making it suitable for large-scale applications.

Conclusions often synthesize multiple results and highlight their practical implications. This environment provides appropriate emphasis for such synthesizing statements.

A `conjecture` represents an educated guess or hypothesis that hasn't been proven but is believed to be true. Conjectures are among the most exciting elements in mathematics, often driving research for decades or centuries.

Conjecture 2.1 (Goldbach's Conjecture). Every even integer greater than 2 can be expressed as the sum of two primes.

Goldbach's conjecture, despite extensive computational verification, remains unproven. The conjecture environment appropriately emphasizes the speculative nature of such statements while acknowledging their mathematical importance.

A `fact` is used for well-established results that are generally known and don't require detailed proof in your context. Facts provide necessary background information without interrupting the flow of your main argument.

Fact 2.1. The complexity class P is contained in NP, and determining whether $\mathrm{P} = \mathrm{NP}$ is one of the most important open problems in computer science.

This fact about computational complexity is fundamental to theoretical computer science. The fact environment is perfect for such widely-accepted background information.

The `folklore` environment is uniquely useful for stating results or principles that are widely known in the mathematical community but might not have formal published proofs or definitive attributions.

> Folklore 2.1. "Premature optimization is the root of all evil" - a principle widely attributed to Donald Knuth, emphasizing that optimization should be guided by measurement rather than intuition.

Folklore results often represent accumulated wisdom in a field. While they might not be formally proven, they represent important heuristics or principles that practitioners have found valuable.

A `hypothesis` is essential for scientific mathematical work. Hypotheses represent testable predictions or proposed explanations that can be investigated through mathematical analysis or experimentation.

> Hypothesis 2.1. Increasing the training data size will improve model accuracy according to a logarithmic scaling law, with diminishing returns beyond a certain threshold.

Hypotheses in mathematical contexts often involve predictions about the behavior of systems or algorithms. The hypothesis environment helps distinguish these predictive statements from established results.

The `postulate` environment is used for fundamental assumptions that form the basis of a mathematical theory. Postulates are similar to axioms but are often more specific to particular mathematical systems.

> Postulate 2.1 (Relativity Postulate). All observers in inertial reference frames observe the same laws of physics, and the speed of light in a vacuum is constant for all observers.

Postulates often appear in mathematical physics or geometry, where they establish the fundamental rules of a particular mathematical universe. Einstein's postulates of special relativity exemplify how postulates can revolutionize entire fields.

Finally, the `property` environment is perfect for highlighting specific characteristics or attributes of mathematical objects. Properties help readers understand the essential features of the objects under study.

> Property 2.1 (Associativity). For all elements a, b, c in the group: $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.

Group theory provides many examples where properties like associativity are fundamental. The property environment helps emphasize these crucial characteristics while maintaining clear organization.

3. Definition-like Environments

Definition-like environments are fundamental to mathematical exposition because they establish the precise meaning of terms and concepts. Clear definitions are essential for rigorous mathematical communication and help ensure that readers understand exactly what is being discussed.

3.1. Core Definition Environments.

The `definition` environment is the cornerstone of mathematical writing. Every mathematical concept must be precisely defined, and the definition environment provides the perfect framework for this essential task.

> Definition 3.1 (Graph). A graph $G = (V, E)$ consists of a finite set of vertices V and a set of edges $E \subseteq V \times V$, where each edge connects two vertices.

This definition of a graph demonstrates the typical structure: we name the object being defined, provide its mathematical representation, and specify the relationships between its components. The definition environment's distinctive formatting helps readers immediately recognize that new terminology is being introduced.

Sometimes you might want to present a definition without formal numbering, perhaps when restating a definition for emphasis or when introducing informal terminology:

Definition (Unnumbered Definition). This shows how unnumbered definitions appear. Use this when you want the visual emphasis of a definition without adding to the formal numbering system.

The `notation` environment is specifically designed for introducing mathematical symbols and notational conventions. Consistent notation is crucial for mathematical clarity, and this environment helps establish and emphasize notational choices.

Notation 3.1. We denote the set of natural numbers by $\mathbb{N}$, the set of integers by $\mathbb{Z}$, the set of rational numbers by $\mathbb{Q}$, and the set of real numbers by $\mathbb{R}$.

Mathematical notation can vary between sources, so explicitly stating your notational conventions prevents confusion. The notation environment makes these conventions prominent and easily referenced.

The `convention` environment establishes standard practices or assumptions that will be used throughout your work. Conventions help streamline exposition by establishing default assumptions.

Convention 3.1. Throughout this paper, all graphs are assumed to be simple (no multiple edges or self-loops) and undirected unless explicitly stated otherwise.

Conventions are particularly important in areas like graph theory, where there are multiple standard definitions. By establishing conventions early, you can avoid repetitive qualifications in later statements.

3.2. **Specialized Definition Environments.** The `construction` environment is perfect for presenting algorithmic or step-by-step procedures for building mathematical objects. Constructions often bridge the gap between abstract definitions and concrete implementations.

Construction 3.1 (Binary Search Tree). To construct a binary search tree from a sequence of elements:

(1) Start with an empty tree (root $=$ null)
(2) For each element in the sequence:
    (a) If the tree is empty, make the element the root
    (b) Otherwise, compare the element with the current node
    (c) If smaller, go to the left subtree; if larger, go to the right subtree
    (d) Repeat until finding an empty position and insert the element
(3) The resulting tree maintains the BST property: left subtree $<$ node $<$ right subtree

This construction demonstrates how to build a fundamental data structure. The construction environment is ideal for such procedural descriptions, making the step-by-step nature clear and easy to follow.

The `problem` environment is essential for presenting mathematical challenges or questions that require solution. Problems often motivate the development of new techniques or illustrate the application of existing theory.

Problem 3.1 (Traveling Salesman Problem). Given a list of cities and the distances between each pair of cities, find the shortest possible route that visits each city exactly once and returns to the starting city.

The TSP is a classic example of a computationally difficult problem that has driven the development of optimization algorithms. The problem environment helps emphasize the challenge and sets up the context for solution approaches.

A `protocol` is particularly useful in computer science and cryptography, where step-by-step procedures involving multiple parties are common. Protocols specify the exact sequence of actions required to achieve a goal.

Protocol 3.1 (Diffie-Hellman Key Exchange). To establish a shared secret key over an insecure channel:

(1) Alice and Bob publicly agree on a large prime p and a generator g of the multiplicative group $\mathbb{Z}_p^*$
(2) Alice chooses a secret integer a and sends $A = g^a \bmod p$ to Bob

(3) Bob chooses a secret integer b and sends $B = g^b \bmod p$ to Alice
(4) Alice computes the shared secret: $s = B^a \bmod p = g^{ab} \bmod p$
(5) Bob computes the shared secret: $s = A^b \bmod p = g^{ab} \bmod p$
(6) Both parties now share the secret s without ever transmitting it

This protocol demonstrates the power of public-key cryptography. The protocol environment clearly delineates the steps and makes the interaction between parties explicit.

The `application` environment highlights practical uses of mathematical concepts. Applications help readers understand the relevance and impact of abstract mathematical ideas.

Application 3.1. Hash tables are widely used in database indexing, caching systems, and implementing associative arrays. Their average-case $\mathcal{O}(1)$ lookup time makes them indispensable for applications requiring fast data retrieval.

Applications bridge the gap between theory and practice, showing how mathematical concepts solve real-world problems. This environment helps emphasize the practical value of mathematical results.

The `experiment` environment is valuable for presenting empirical investigations or computational studies. In an era of computational mathematics, experiments often provide crucial insights.

Experiment 3.1. We implemented the algorithm in Python and tested it on randomly generated datasets of varying sizes: 1K, 10K, 100K, and 1M records. Each test was repeated 100 times to ensure statistical reliability, and we measured both execution time and memory usage.

Computational experiments are increasingly important in mathematics and computer science. The experiment environment helps distinguish empirical investigations from theoretical analysis.

Finally, the `result` environment presents the outcomes of experiments or investigations. Results often provide the empirical evidence needed to support theoretical claims.

Result 3.1. The proposed algorithm achieved 95.7% accuracy on the test dataset, with a standard deviation of 1.2%. Execution time scaled linearly with input size, confirming the theoretical $\mathcal{O}(n)$ complexity analysis.

Results provide concrete evidence for the performance or behavior of algorithms and methods. The result environment helps distinguish these empirical findings from theoretical predictions.

4. Remark-like Environments

Remark-like environments provide commentary, clarification, and additional insights that enhance understanding without being part of the main logical flow. These environments help create a more conversational and educational tone in mathematical writing.

4.1. **Commentary and Clarification.**

The `remark` environment is one of the most versatile tools in mathematical exposition. Use it to provide additional insight, point out special cases, or offer alternative perspectives on results.

Remark 4.1. This result generalizes to higher dimensions with appropriate modifications. In n-dimensional space, the constant factor changes from π to $\frac{\pi^{n/2}}{\Gamma(n/2+1)}$, but the fundamental structure of the proof remains unchanged.

Remarks like this help readers understand the broader context and potential extensions of results. They're perfect for insights that are valuable but not essential to the main argument.

For informal comments that don't need numbering, use the starred version:

Remark (Unnumbered Remark). Unnumbered remarks are useful for casual observations or side comments that enhance understanding but don't require formal reference. They maintain the visual emphasis while keeping the numbering system clean.

The `example` environment is crucial for illustrating abstract concepts with concrete instances. Examples help readers understand definitions and theorems by showing them in action.

Example 4.1. Consider the function $f(x) = x^2$ on the interval $[0, 2]$. Its derivative is $f'(x) = 2x$, which is positive for all $x > 0$, confirming that f is strictly increasing on $(0, 2]$. At $x = 1$, we have $f(1) = 1$ and $f'(1) = 2$, so the tangent line has equation $y = 2x - 1$.

This example demonstrates calculus concepts with specific calculations. Examples should be carefully chosen to illuminate the most important aspects of the concept being illustrated.

A `note` provides additional information or draws attention to important details that might otherwise be overlooked. Notes are particularly useful for pointing out connections between results.

Note 4.1. The proof technique used here is essentially the same as the one employed in Theorem 2.3, but applied to a different metric space. This suggests that the result might generalize to a broader class of spaces.

Notes help readers recognize patterns and connections in mathematical arguments. They're invaluable for building mathematical intuition and understanding.

The `observation` environment is perfect for stating facts that are noticed during analysis but aren't formally proven results. Observations often lead to deeper insights or future research directions.

Observation 4.1. The algorithm's performance appears to degrade linearly with input noise levels, suggesting that preprocessing to reduce noise could significantly improve results.

Observations like this often come from experimental work or careful analysis of examples. They help readers understand the practical behavior of mathematical objects or algorithms.

4.2. **Educational and Explanatory Environments.** The `commentary` environment provides space for editorial remarks about mathematical content. Commentary can discuss the historical development of ideas, their significance, or their relationship to other work.

Commentary 4.1. This approach represents a significant departure from traditional methods in numerical analysis. While classical techniques focus on polynomial approximation, this method leverages the geometric structure of the problem to achieve superior convergence rates.

Commentary helps place mathematical work in context and can provide valuable perspective on the significance and novelty of results.

An `exercise` presents problems for readers to solve, promoting active engagement with the material. Exercises should be carefully designed to reinforce key concepts and develop problem-solving skills.

Exercise 4.1. Prove that the sum of the first n positive integers is $\frac{n(n+1)}{2}$ using mathematical induction. Then, use this result to find a formula for the sum of the first n positive odd integers.

Exercises like this build mathematical skills progressively. The first part reviews a fundamental technique, while the second part applies the result to a related problem.

The `motivation` environment explains why certain concepts or approaches are important. Motivation helps readers understand the purpose behind mathematical development and maintains interest in the material.

Motivation 4.1. Understanding the time complexity of recursive algorithms is crucial for several reasons: it helps predict performance on large inputs, guides optimization efforts, and provides insight into the inherent difficulty of computational problems.

Motivation is particularly important when introducing abstract concepts or technical machinery. It helps readers understand why they should invest effort in understanding the material.

The `notationabuse` environment acknowledges places where notation is used loosely or where common shortcuts are employed. This promotes honesty in mathematical communication.

Notation Abuse 4.1. We sometimes write $\mathcal{O}(f(n))$ when we technically mean $\Theta(f(n))$, following the common computer science convention where big-O notation is used to describe tight bounds rather than just upper bounds.

Acknowledging notation abuse helps prevent confusion and demonstrates awareness of mathematical precision. It's particularly important in applied fields where notation conventions may differ from pure mathematics.

Finally, the `question` environment poses inquiries that arise naturally from the material. Questions can highlight open problems, suggest extensions, or prompt readers to think more deeply about concepts.

Question 4.1. Can this convergence result be extended to infinite-dimensional Banach spaces? What additional conditions would be necessary to ensure convergence in such spaces?

Questions like this encourage mathematical thinking and can point toward future research directions. They help readers engage more actively with the material and develop their own mathematical intuition.

5. Highlight-like Environments

Highlight-like environments are designed to draw special attention to crucial information, warnings, insights, and key takeaways. These environments help create visual hierarchy and ensure that important points aren't overlooked by readers.

5.1. **Critical Information and Warnings.**

The `important` environment is used to emphasize information that is absolutely crucial for understanding or applying the material. Use this environment sparingly to maintain its impact.

Important 5.1. Always validate input data before processing to prevent security vulnerabilities, buffer overflows, and incorrect results. Failure to validate inputs is one of the most common sources of software bugs and security exploits.

Security considerations like this are genuinely important and deserve special emphasis. The important environment ensures that such critical information stands out from the surrounding text.

For critical notes that don't need numbering, use the starred version:

Important (Critical Security Note). This is an unnumbered important note for emphasis. Use this when you want maximum visual impact without adding to the formal numbering system.

The `warning` environment alerts readers to potential problems, common mistakes, or dangerous practices. Warnings can prevent readers from making costly errors or falling into common traps.

Warning 5.1. Modifying the convergence parameter below 0.001 may cause numerical instability and lead to completely incorrect results. Always test parameter changes thoroughly before using them in production code.

Warnings like this can save readers significant time and frustration. They're particularly valuable in computational mathematics where parameter choices can have dramatic effects on results.

A `tip` provides helpful advice or suggestions that can improve understanding or performance. Tips often represent practical wisdom gained through experience.

Tip 5.1. Use version control for all your code, even small projects. Git repositories help track changes, enable collaboration, and provide backup. Initialize a repository with `git init` before writing any code.

Tips like this provide practical guidance that goes beyond the immediate mathematical content but supports effective mathematical work. They help readers develop good practices.

5.2. **Insights and Key Points.** The `highlight` environment draws attention to particularly important or interesting aspects of the material. Use highlights to emphasize key insights or crucial connections.

Highlight 5.1. The key insight is that the graph coloring problem can be reformulated as a constraint satisfaction problem, which opens up a wide range of algorithmic approaches including backtracking, forward checking, and arc consistency.

Highlights help readers identify the most important ideas in complex material. They're particularly valuable for emphasizing conceptual breakthroughs or novel approaches.

A `keypoint` emphasizes fundamental principles or essential facts that readers should remember. Key points often summarize the most important aspects of a section or concept.

Key Point 5.1. Performance optimization should always be based on actual profiling data rather than assumptions or intuition. The bottlenecks are often in unexpected places, and premature optimization can actually harm performance.

Key points like this distill important principles that apply broadly. They help readers extract the most valuable lessons from detailed technical material.

An `insight` presents a deeper understanding or novel perspective that emerges from analysis. Insights often represent "aha moments" that significantly advance understanding.

Insight 5.1. The real bottleneck in this algorithm is not the computational complexity but the memory access patterns. Cache misses dominate execution time, which explains why the theoretical analysis doesn't match empirical performance.

Insights like this often come from careful analysis or experimentation. They represent genuine advances in understanding that can guide future work.

5.3. **Synthesis and Summary.** The `takeaway` environment summarizes the most important lessons or conclusions from a section. Takeaways help readers extract the essential value from complex material.

Takeaway 5.1. Simplicity often leads to more maintainable and efficient solutions than clever optimizations. Focus on clear, correct implementations first, then optimize based on actual performance measurements.

Takeaways like this distill practical wisdom that applies beyond the immediate context. They help readers develop good judgment and effective approaches to problem-solving.

A `summary` provides a concise overview of results, methods, or conclusions. Summaries are particularly valuable at the end of sections or after presenting multiple related results.

Summary 5.1. We presented three sorting algorithms with different time complexities: bubble sort ((n^2)), merge sort ($(n \log n)$), and counting sort ($(n + k)$). The choice depends on input characteristics, stability requirements, and space constraints.

Summaries help readers consolidate their understanding and see the big picture. They're especially valuable in technical material where details might obscure the main points.

The `recall` environment reminds readers of important facts or principles that are relevant to the current discussion. Recall statements help maintain context and prevent readers from forgetting crucial background.

Recall 5.1. Remember that correlation does not imply causation. Even strong statistical relationships don't establish causal links without additional evidence such as controlled experiments or causal inference methods.

Recall statements like this reinforce fundamental principles that are easy to forget in the midst of technical details. They help maintain good mathematical and scientific thinking.

Finally, the `guideline` environment presents best practices or recommended approaches. Guidelines help readers develop good habits and avoid common problems.

Guideline 5.1. Follow the single responsibility principle: each function should have one clear, well-defined purpose. This makes code easier to understand, test, debug, and maintain.

Guidelines like this promote good programming and mathematical practices. They help readers develop skills that will serve them well across many different contexts.

6. Case Environment Demo

The `case` environment is a specialized tool for organizing proofs and arguments that involve multiple scenarios or conditions. Case analysis is a fundamental proof technique that helps break complex problems into manageable parts.

Case analysis is particularly common in discrete mathematics, where problems often involve considering different possibilities exhaustively. The case environment provides clear visual organization for such arguments.

Theorem 6.1 (Absolute Value Properties). For any real number x, we have $|x| \geq 0$, and $|x| = 0$ if and only if $x = 0$.

This theorem about absolute values naturally leads to a case-by-case analysis. The proof demonstrates how the case environment organizes different scenarios clearly:

Proof. We prove both parts by considering all possible cases for the real number x:

Case 1: If $x \geq 0$, then by definition $|x| = x \geq 0$. Moreover, $|x| = 0$ if and only if $x = 0$.
Case 2: If $x < 0$, then by definition $|x| = -x > 0$ since x is negative. In this case, $|x| \neq 0$ since $x \neq 0$.

In both cases, we have $|x| \geq 0$, and $|x| = 0$ occurs precisely when $x = 0$. ■

Notice how the case environment creates a numbered list that clearly distinguishes between different scenarios. This organization makes the proof much easier to follow than a paragraph that tries to handle all cases simultaneously.

Case analysis is also valuable in algorithm analysis, where different input conditions can lead to different performance characteristics, and the case environment is equally useful in mathematical proofs involving induction, where base cases and inductive steps need clear organization.