Creative Education Studies 创新教育研究, 2026, 14(3), 558-572
Published Online March 2026 in Hans. https://www.hanspub.org/journal/ces
https://doi.org/10.12677/ces.2026.143233

Hans汉斯

Python可视化工具在高等数学教学中的应用与成效分析

李　钊

成都大学计算机学院，四川　成都

收稿日期：2026年2月9日；录用日期：2026年3月17日；发布日期：2026年3月26日

摘　要

文章旨在探讨Python可视化工具在高等数学教学中的应用与成效分析。凭借强大的科学计算库和丰富的数据可视化功能，Python能够将高等数学中抽象的核心概念与复杂计算过程转化为直观图形和可交互代码，有效降低了学生的认知难度。这不仅有助于激发学生的学习兴趣、增强课堂互动，更有助于培养学生利用Python解决实际工程问题的能力。将Python融入高等数学教学中，能显著提升学生运用编程进行数学建模和计算的技能，对实现应用型和创新型人才的培养目标具有重要的实践意义。

关键词

高等数学，编程语言，极限，定积分，空间解析几何

Application and Effectiveness Analysis of Python Visualization Tools in Higher Mathematics Teaching

Zhao Li

College of Computer Science, Chengdu University, Chengdu Sichuan

Received: February 9, 2026; accepted: March 17, 2026; published: March 26, 2026

Abstract

The main purpose of this article is to explore the application and effectiveness analysis of Python visualization tools in higher mathematics teaching. With its powerful scientific computing library and rich data visualization capabilities, Python can transform abstract core concepts and complex

文章引用: 李钊. Python 可视化工具在高等数学教学中的应用与成效分析[J]. 创新教育研究, 2026, 14(3): 558-572.
DOI: 10.12677/ces.2026.143233

computational processes in advanced mathematics into intuitive graphics and interactive code, effectively reducing students' cognitive difficulty. This not only helps to stimulate students' interest in learning and enhance classroom interaction, but also helps cultivate their ability to use Python to solve practical engineering problems. Integrating Python into higher mathematics teaching can significantly enhance students' skills in using programming for mathematical modeling and computation, and has important practical significance for achieving the training goals of applied and innovative talents.

Keywords

Higher Mathematics, Programming Language, Limit, Definite Integral, Space Analytic Geometry

Copyright © 2026 by author(s) and Hans Publishers Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/
Open Access

1. 引言

《高等数学》[1]-[3]是理工科及经济管理类专业一门至关重要的公共基础课程，其主要涵盖了一元函数与多元函数的微分学、积分学、常微分方程、空间解析几何以及无穷级数等核心内容。该课程通常在大学一年级开设，分上下两册，通常根据不同的专业分为几种不同的学时。例如，面向理工科学生两学期合计开设课时一般为 160 学时，面向经济管理类两学期合计一般为 128 学时，面向其他偏文科专业两学期合计一般为 96 学时。高等数学是一门强大的数学“语言”和“工具”，为后续专业课程的学习提供了不可或缺的理论模型与分析方法。更为深刻的是，高等数学的学习并非仅仅是公式与技巧的机械记忆，其精髓在于对学生理性思维方式的培养[4]-[6]。它致力于培养学生严谨的逻辑推理能力、抽象的数学建模能力，以及将复杂现实问题转化为数学模型并求解的实践能力。因此，熟练掌握高等数学不仅是为专业学习铺路，更是培养一名现代工程师、科研工作者或管理人才科学素养与创新潜力的关键环节。在过去的《高等数学》教学与实践中，长期存在若干突出问题。首先，课程内容本身具有较强的抽象性，常出现教学内容与学生所学专业实际需求相脱节的现象。例如，在极限的学习过程中，学生很难理解数列极限和函数极限的定义；在空间解析几何部分，学生普遍难以凭空想象复杂的空间三维图形，甚至即使掌握了多元函数图像的绘制方法，也很难将多个立体图像合并呈现在同一幅图中，这正体现了抽象内容与直观理解之间的认知差距。此类知识薄弱点将直接影响后续专业课程的学习效果。其次，教学方法较为单一，教师多侧重于基本概念的传授和定理公式的推导，导致学生容易陷入对公式的机械记忆，而缺乏对数学思想与实际意义的深入理解。最后，课程普遍面临课时紧张的问题，部分专业甚至因中途安排专业实训，进一步挤占教学时间，影响知识体系的连贯性与完整性。而且，传统的静态 PPT 或板书教学方式在此情境下也显得力不从心。

随着计算机技术的快速发展，在高等数学的教学中引入 Python 绘图和计算[7]-[9]已经成为提升高等数学教学效果的一个关键切入点。Python 能够将高度抽象的数学概念与逻辑(例如复杂的三维空间图形、逼近无限的极限过程)转化为生动直观的可视图像，对于复杂的计算问题，可以快速获得结果，从而在学生与抽象知识之间架起一座直观的桥梁。具体而言，在讲解空间解析几何时，教师可利用 Python 绘制旋转曲面图形，使学生能从任意视角观察其形态，显著改善从二维平面想象三维空间的壁垒；在阐释微积分核心思想时，通过代码快速求导或者求解定积分或者不定积分等，并且能够绘制定积分的矩阵图，让学生“看见”积分的思想本质，而非仅仅记忆公式。这不仅极大地化解了课程的抽象性，更将教学从定

理公式的机械推导转向数学思想与应用的深入理解，引导学生从被动接受转为主动探索。通过设计与专业相关的数学建模案例，Python 更进一步将高等数学的工具属性凸显出来，强化了与后续课程的衔接。因此，Python 融入《高等数学》的教学不仅仅是教学手段的技术升级，更是一种教学范式的深刻变革，它旨在将《高等数学》从一门侧重于演算技巧的孤立学科，转变为一门培养学生利用计算思维进行建模、仿真与解决实际问题的核心能力课程。

2. Python 在高等数学教学中的典型案例分析与教学设计深化

2.1. 案例 1：Python 在极限教学中的应用

2.1.1. 案例回顾

为了生动呈现如何利用 Python 软件将抽象的数学极限概念转化为直观的视觉认知。本节将围绕高等数学中至关重要的极限公式II：$\lim\limits_{x\to\infty}\left(1+\frac{1}{x}\right)^{x}=\mathrm{e}$ 展开分析。

2.1.2. 课堂教学设计(教案节选)

1) 时间点与目的(0~5 分钟)：回顾极限 $\lim\limits_{x\to\infty}\left(1+\frac{1}{x}\right)^{x}=\mathrm{e}$ 的形式，提出问题：“当 x 趋于无穷大”时，“1^{∞}”型未定式如何收敛于常数 e 这一难点，本节将提出了利用可视化方法进行探索。

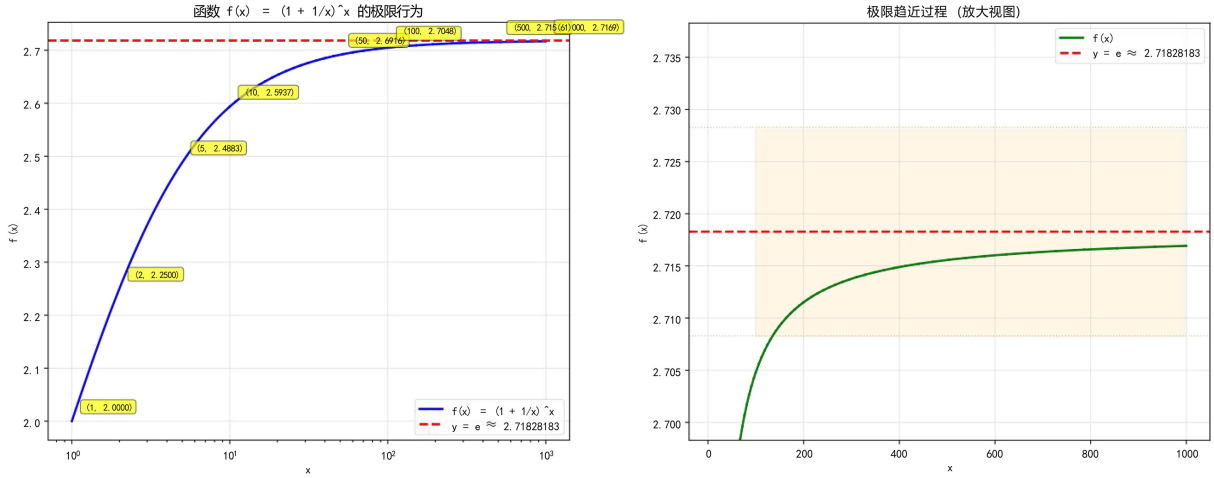

Figure 1. The graph of function $f(x)=\left(1+\frac{1}{x}\right)^{x}$ on the interval $[0,10^{3}]$

图 1. 函数 $f(x)=\left(1+\frac{1}{x}\right)^{x}$ 在区间$[0,10^{3}]$的图形

2) 代码演示与互动(5~20 分钟)：运行附录代码，生成图 1，并引导学生观察图像。利用 Python 软件绘制函数 $f(x)=\left(1+\frac{1}{x}\right)^{x}$ 在区间$[0,10^{3}]$的图像，清晰地揭示了函数的动态行为：图 1 左图中的蓝色曲线在起始段(如$[0,10^{1}]$区间)呈现出显著的上升趋势，直观地反映了函数值从初始变化到逐渐增大的过程；右图通过叠加一条醒目的红色水平渐近线 $y=\mathrm{e}$，精准地刻画了当变量 x 持续增大至$[10^{2},10^{3}]$区间乃至更大范围时，函数值没有增长及剧烈波动，而是无限逼近常数 e 的稳定状态。通过 Python 软件的绘图，可以很容易理解函数的重要极限公式II“无限趋近”的数学思想，也充分证明了 Python 软件在高等数学中

对于深化概念理解的强大辅助作用。

3) 突发情况处理：若现场出现代码报错，则转为讲解代码逻辑，附录中的代码都有相应的解释。鼓励学生课后自主搭建代码，并且更改参数的取值范围再运行代码。

2.2. 案例 2：Python 在定积分中的应用

2.2.1. 设计意图

在高等数学的定积分学习中，常常会遇到一类难以积分的问题，这类积分往往形式会很复杂，无法通过换元法、分部积分法等方法直接进行积分求解。面对这类问题，可以考虑利用 Python 软件进行积分。

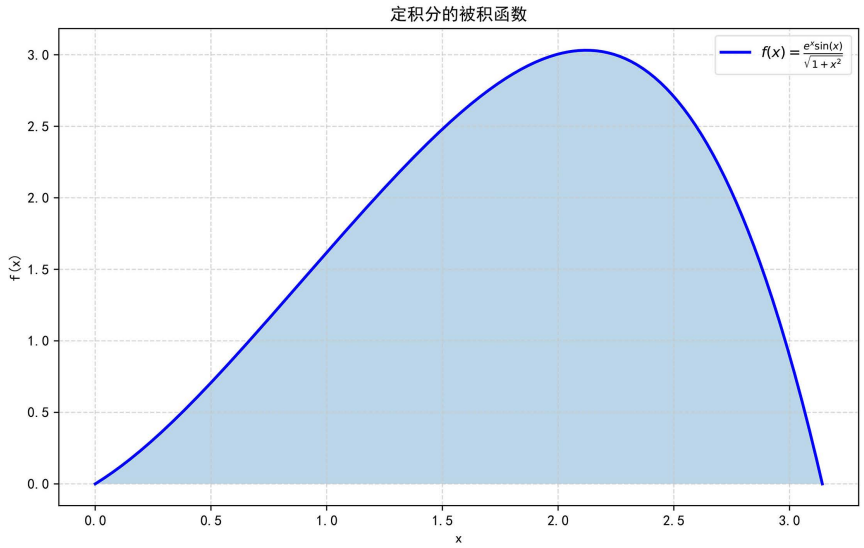

Figure 2. The definite integral of the function $f(x)=\dfrac{e^x \sin x}{\sqrt{1+x^2}}$ on the interval $[0,\pi]$

图 2. 函数 $f(x)=\dfrac{e^x \sin x}{\sqrt{1+x^2}}$ 在 $[0,\pi]$ 上的定积分

2.2.2. 教学内容

考虑函数 $f(x)=\dfrac{e^x \sin x}{\sqrt{1+x^2}}$ 在 $[0,\pi]$ 上的定积分问题。显然通过利用 Python 软件可以获得该积分的值为：5.686501309557569。并且，利用 Python 软件绘制该图形的二维图，如图 2 所示，其 Python 代码见附录。

2.2.3. 课堂活动设计

1) 概念回顾：板书定积分定义的数学表达式。

2) 代码解读(10 分钟)：逐行解释附录中的代码，重点说明如何实现“分割”，以及求和是如何实现的。

3) 动态观察(10 分钟)：观察图 2 的定积分结果以及它的几何意义。

2.3. 案例 3：Python 在空间解析几何绘图中的应用

2.3.1. 设计意图

《高等数学》中的空间解析几何章节往往令人望而生畏，其核心难点在于将抽象的代数方程在脑海中转化为具体的三维图形。这一转化过程在教学中很难通过黑板绘制实现。随着计算机的快速发展，可

以利用 Python 软件将空间直角坐标系中的空间曲面和空间曲线的方程等内容通过计算机软件进行绘制，从而便于观察。

2.3.2. 教学内容

维维安尼曲线是经典的曲线，本文基于 Python 软件绘制半球面 $z=\sqrt{4-x^2-y^2}$ 和圆柱面 $(x-1)^2+y^2=1$ 的维维安尼曲线，如图 3 所示，其 Python 代码见附录。

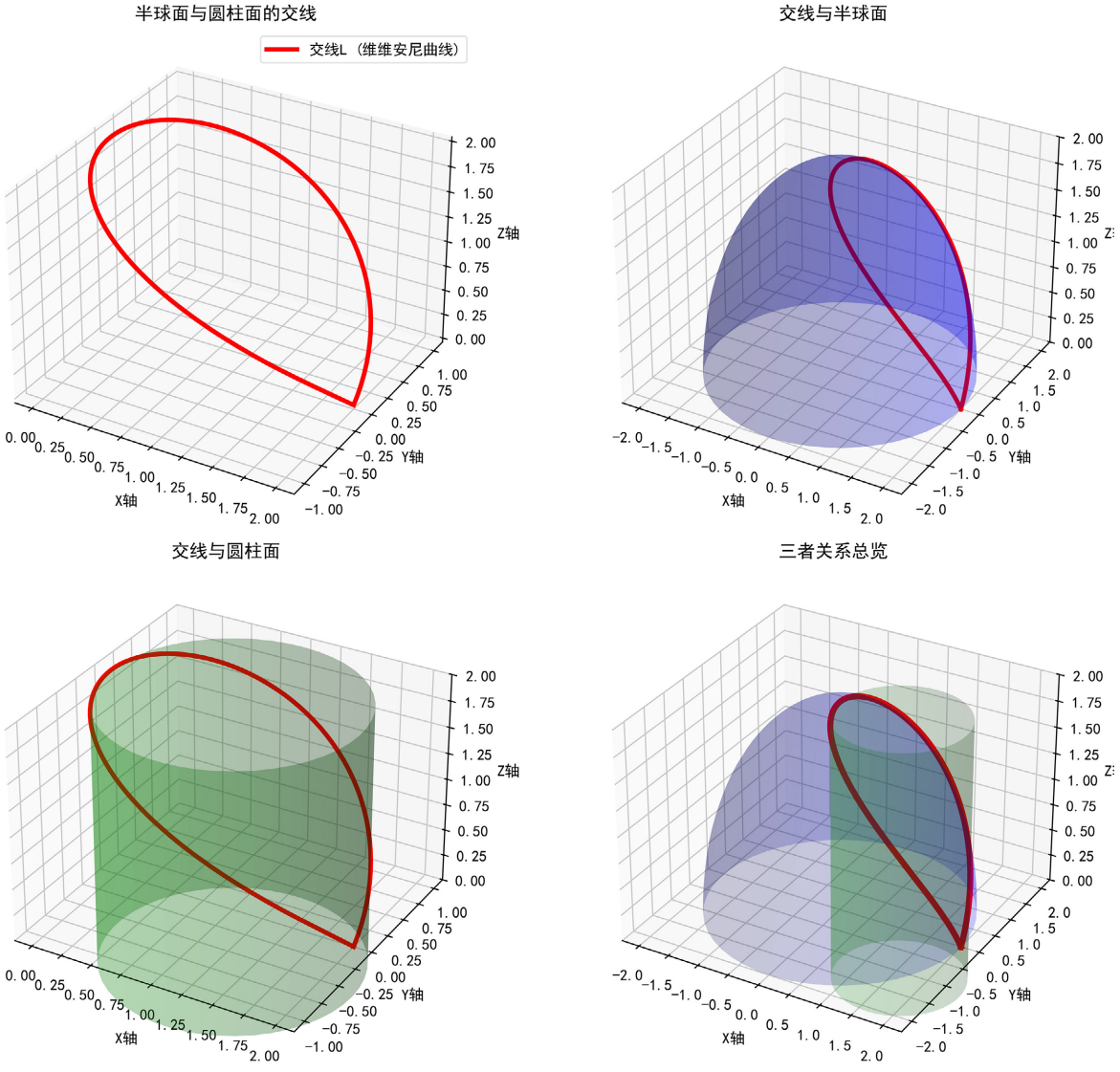

Figure 3. Curve of the intersection line between hemispherical surface $z=\sqrt{4-x^2-y^2}$ and cylindrical surface $(x-1)^2+y^2=1$

图 3. 半球面 $z=\sqrt{4-x^2-y^2}$ 和圆柱面 $(x-1)^2+y^2=1$ 交线的曲线

2.3.3. 实证

在“空间解析几何”章节，A 班(实验组)采用“传统讲解 + Python 动态演示与交互旋转”；B 班(对照组)采用“传统讲解 + 静态 PPT 图示”。教学后，使用同一套试卷进行测试。学生成绩如表 1 所示。从表格中可以看到，显然实验组在使用 Python 软件后的优秀率比传统教学组的人数要多，并且不及格率

也相应降低。

Table 1. Student performance evaluation
表 1. 学生成绩评价

班级	学生人数	等级				
		优秀(90 分以上)	良好(80~89 分)	中等(70~79 分)	及格(60~69 分)	不及格
A 班(实验组)	90 人	9 人(10%)	17 人(18.9%)	24 人(26.7%)	27 人(30%)	13 人(14.4%)
B 组(对照组)	90 人	3 人(3.3%)	16 人(17.8%)	26 人(28.9%)	25 人(27.8%)	20 人(22.2%)

2.4. 案例 4：Python 在三重积分计算中的应用

2.4.1. 教学内容

计算三重积分 $\iiint_{\Omega}(x+y)\mathrm{d}x\mathrm{d}y\mathrm{d}z$ 的问题时，通常计算量非常的大。这里积分区域 Ω 由三个坐标面及平面 $x+y+z=1$ 围成的闭区域。

2.4.2. 课堂活动设计

1) 概念回顾：板书三重积分定义的数学表达式。

2) 代码解读(10 分钟)：逐行解释附录中的代码。

3) 动态观察(10 分钟)：可以通过编写 Python 程序，进而运行 Python 程序，即可得到结果为 $\frac{1}{12}$。具体 Python 代码见附录。

3. 反思技术的双面：挑战与风险

3.1. 挑战与风险分析

Python 在《高等数学》教学中的应用，虽然显著提升了学生的直观理解能力与数学建模素养，但其潜在的双刃剑效应仍需我们保持清醒认识。具体而言，主要面临以下两个维度的挑战：第一，过度依赖计算机软件可能导致学生对基础概念的理解浮于表面，削弱其必要的手算和“心算”能力。当学生习惯于直接调用库函数求导、积分时，往往会忽视极限思想的本质内涵和基本运算技巧的训练，将使学生在缺乏软件支持的情况下失去独立解决数学问题的能力，更难以培养严谨的逻辑推理习惯。第二，在教学实施层面存在“师资与内容”的双重压力。作为一门大规模开设的公共基础课，高等数学涉及的教学团队庞大，教师对 Python 技能的掌握程度可能存在显著差异，这为统一、规范地开展教学带来了挑战。同时，在原本就紧张的课时安排中，既要保证核心知识体系的完整讲授，又要融入 Python 实践环节，极易造成两方面都浅尝辄止的困境，反而影响教学质量的提升。

3.2. 教学优化策略建议

为最大限度发挥 Python 的教学价值，我们提出以下针对性策略：在教学模式上，必须确立“数学本体，技术为辅”的基本原则。具体实施中要严格遵循以下三个步骤：首先，通过传统教学确保学生对基本概念、定理证明和典型算例达到深刻理解；其次，引导其进行必要的“笔头”推导，夯实基础；最后，运用 Python 进行数值验证、可视化呈现或拓展探究，以此彰显技术的赋能作用，而非替代价值。在课程设计上，建议构建“分层混合”的教学体系，一方面保证足够量的传统练习，使学生掌握核心运算能力；另一方面精心设计具有代表性的 Python 实践项目。这既巩固了基本功，又培养了学生运用计算机工具解决复杂工程问题的能力。

4. 总结

本文通过系统性地将 Python 引入《高等数学》课程教学。在空间解析几何、函数极限、微积分运算等教学难点环节，借助 Python 强大的计算和图形可视化功能，将抽象的数学概念转化为直观的图像，有效弥合了理论认知与几何直观之间的鸿沟。特别是通过函数极限绘图、曲面绘制、多重积分计算等典型案例，使学生能够直观观察数学对象的形态特征和变化规律，显著提升了对抽象数学语言的理解深度。基于 Python 的数值实验与理论推导形成了良好互补，软件计算结果与传统演算结果的高度一致性，验证了数学理论的严谨性，其强大的计算能力使学生从繁琐的手工运算中解放出来，得以将更多精力投入到数学思想的领悟和实际问题的建模中。这种教学方式不仅培养了学生的计算思维，更强化了其运用数学工具解决工程问题的实践能力。将计算机代数系统有机融入高等数学教学，能够有效促进抽象思维与直观理解的有机结合，推动数学教学从“知识传授”向“能力培养”转型。Python 在高等数学教学中的应用创新，不仅提升了课程本身的教学效果，更从能力培养、考核评价、课程体系等多个维度支持了工程教育专业认证的要求，为专业建设提供了实质性支撑。未来还将进一步探索如何与 Python 工具和其他数学软件(例如 MATLAB、Maple 等)进行优势互补，构建更加完善的数学实验教学体系。

基金项目

四川省哲学社会科学重点研究基地四川省教师教育研究中心高校师德师风建设长效机制创新研究探索课题研究成果，项目编号：TER2024-025。成都大学 2024~2026 年本科教育教学改革研究项目(项目编号：cdjgb2024219)。

参考文献

[1] 姜翀. 基于研究型教学模式的高等数学教改探析[J]. 沈阳师范大学学报(自然科学版), 2012, 30(4): 562-565.

[2] 钟太勇. 基于数学建模的高等数学教学改革与实践研究[J]. 科技风, 2025(32): 74-76.

[3] 林丽英, 江南. 高等数学“四层三面、理实融合”项目化教学模式实践探索[J]. 福建教育学院学报, 2025, 26(10): 88-92.

[4] 韦碧鹏. 混合式教学模式下高等数学课程教学模型的构建与实践研究[J]. 湖北开放职业学院学报, 2025, 17(38): 190-193.

[5] 丁尚文. 混合式下的高等数学课堂教学探索与实践[J]. 大学数学, 2023, 39(4): 51-59.

[6] 曹亚群. “腾讯课堂 + 智慧职教”混合教学在高等数学中的应用[J]. 湖北开放职业学院学报, 2024, 37(8): 147-149.

[7] 宋婷婷, 王琳琳. Python 语言在高等数学积分教学中的应用探析[J]. 电脑知识与技术, 2023, 19(25): 118-121.

[8] 赵禹琦. Python 软件在求解线性非齐次微分方程中的应用[J]. 集成电路应用, 2021, 38(5): 53-55.

[9] 李冰峰. 计算机技术在高等数学教学中的应用[J]. 科技信息, 2010(35): 96.

附　录

1. Python 在极限教学中的应用

具体的代码如下：

```
import numpy as np
import matplotlib.pyplot as plt
import math

# 设置中文字体和图形参数
plt.rcParams['font.sans-serif'] = ['SimHei', 'Arial Unicode MS', 'DejaVu Sans']
plt.rcParams['axes.unicode_minus'] = False

def visualize_limit():
    """
    可视化函数 f(x) = (1 + 1/x)^x 在 x 趋近于无穷大时的极限行为
    展示其趋近于自然常数 e 的过程
    """
    # 生成 x 值：从较小值到较大值，更好地展示极限过程
    x_small = np.linspace(1, 10, 100)          # 小范围，观察快速变化
    x_medium = np.linspace(10, 100, 200)     # 中范围，观察趋近过程
    x_large = np.linspace(100, 1000, 300)   # 大范围，观察稳定趋近

    # 计算对应的 y 值
    y_small = (1 + 1/x_small)**x_small
    y_medium = (1 + 1/x_medium)**x_medium
    y_large = (1 + 1/x_large)**x_large

    # 创建图形和子图
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))

    # 第一个子图：完整范围的视图
    ax1.plot(x_small, y_small, 'b-', linewidth=2, label='f(x) = (1 + 1/x)^x')
    ax1.plot(x_medium, y_medium, 'b-', linewidth=2)
    ax1.plot(x_large, y_large, 'b-', linewidth=2)

    # 添加理论极限线
    e_value = math.e
    ax1.axhline(y=e_value, color='r', linestyle='--', linewidth=2,
                label=f'y = e ≈ {e_value:.8f}')
```

```
# 添加一些关键点的标注
key_points = [1, 2, 5, 10, 50, 100, 500, 1000]
for x_point in key_points:
    if x_point <= 1000:
        y_point = (1 + 1/x_point)**x_point
        ax1.annotate(f'({x_point}, {y_point:.4f})',
                     xy=(x_point, y_point), xytext=(10, 10),
                     textcoords='offset points', fontsize=8,
                     bbox=dict(boxstyle='round,pad=0.3', facecolor='yellow', alpha=0.7))

ax1.set_xlabel('x')
ax1.set_ylabel('f(x)')
ax1.set_title('函数 f(x) = (1 + 1/x)^x 的极限行为')
ax1.legend()
ax1.grid(True, alpha=0.3)
ax1.set_xscale('log')  # 使用对数坐标更好地展示大范围数据

# 第二个子图：放大视图，关注趋近过程
ax2.plot(x_medium, y_medium, 'g-', linewidth=2, label='f(x)')
ax2.plot(x_large, y_large, 'g-', linewidth=2)

# 添加极限线和误差带
ax2.axhline(y=e_value, color='r', linestyle='--', linewidth=2,
            label=f'y = e ≈ {e_value:.8f}')
# 添加误差带(±0.01)
ax2.axhline(y=e_value + 0.01, color='orange', linestyle=':', linewidth=1,
            alpha=0.5, label='e ± 0.01')
ax2.axhline(y=e_value - 0.01, color='orange', linestyle=':', linewidth=1, alpha=0.5)
ax2.fill_between(x_large, e_value - 0.01, e_value + 0.01, alpha=0.1, color='orange')

ax2.set_xlabel('x')
ax2.set_ylabel('f(x)')
ax2.set_title('极限趋近过程(放大视图)')
ax2.legend()
ax2.grid(True, alpha=0.3)
ax2.set_ylim(e_value - 0.02, e_value + 0.02)  # 放大 y 轴范围

# 调整布局
plt.tight_layout()
```

```
    plt.show()

    # 打印数值分析结果
    print("数值分析结果：")
    print("=" * 50)
    print(f"理论极限值: e = {e_value:.10f}")
    print("-" * 50)
    test_points = [1, 10, 100, 1000, 10000, 100000]
    for x in test_points:
        y_calculated = (1 + 1/x)**x
        error = abs(y_calculated - e_value)
        error_percent = (error / e_value) * 100
        print(f"x = {x:6d}: f(x) = {y_calculated:.10f}, "
              f"误差 = {error:.2e} ({error_percent:.6f}%)")

def analyze_convergence_rate():
    """
    分析函数收敛到 e 的速度
    """
    print("\n 收敛速度分析：")
    print("=" * 50)
    x_values = [10**i for i in range(1, 7)]  # 10, 100, 1000, ..., 1000000
    e_value = math.e
    prev_error = float('inf')

    for i, x in enumerate(x_values):
        y = (1 + 1/x)**x
        error = abs(y - e_value)
        if i > 0:
            convergence_ratio = prev_error / error
            print(f"x = {x:8d}: 误差 = {error:.2e}, 收敛比 = {convergence_ratio:.2f}")
        else:
            print(f"x = {x:8d}: 误差 = {error:.2e}")
        prev_error = error

# 主程序
def main():
    """
    主函数：执行可视化分析和收敛速度分析
    """
```

```
    visualize_limit()
    analyze_convergence_rate()

# 测试用例
def test_functions():
    """
    测试函数，验证基本功能
    """
    print("测试函数计算结果：")
    print("=" * 50)

    # 测试几个关键点的函数值
    test_points = [1, 2, 5, 10, 100]
    for x in test_points:
        y = (1 + 1/x)**x
        print(f"f({x}) = {y:.8f}")

    # 验证极限值
    e_value = math.e
    large_x = 1000000
    y_large = (1 + 1/large_x)**large_x
    error = abs(y_large - e_value)
    print(f"\n 验证极限：")
    print(f"数学常数 e = {e_value:.10f}")
    print(f"当 x = {large_x} 时，f(x) = {y_large:.10f}")
    print(f"误差 = {error:.10f} (相对误差 = {(error/e_value)*100:.8f}%)")

if __name__ == "__main__":
    # 运行主程序
    main()

    # 运行测试用例
    print("\n" + "=" * 50)
    print("功能测试：")
    print("=" * 50)
    test_functions()
```

运行，即可得到图 1。

2. Python 在定积分中的应用

```
import numpy as np
```

```
from scipy import integrate
import matplotlib.pyplot as plt
# 1. 定义被积函数
def integrand(x):
    «  » »定义被积函数：(e^x * sin(x)) / sqrt(1 + x^2) «  » »
    return (np.exp(x) * np.sin(x)) / np.sqrt(1 + x**2)
# 2. 设置积分下限和上限
a = 0  # 下限
b = np.pi  # 上限
# 3. 调用 quad 函数进行数值积分
# quad 返回两个值：积分结果和一个对误差的估计
result, error_estimate = integrate.quad(integrand, a, b)
# 4. 打印结果
print("数值积分结果:")
print(f"积分值 I ≈ {result}")
print(f"误差估计: ± {error_estimate}")
# (可选)可视化被积函数，理解我们正在积什么
x_vals = np.linspace(a, b, 500)
y_vals = integrand(x_vals)
plt.figure(figsize=(10, 6))
plt.plot(x_vals, y_vals, 'b-', linewidth=2, label=r'$f(x) = \frac{e^x \sin(x)}{\sqrt{1 + x^2}}$')
plt.fill_between(x_vals, y_vals, alpha=0.3) # 填充积分区域
plt.xlabel('x')
plt.ylabel('f(x)')
plt.title('定积分的被积函数')
plt.grid(True, linestyle='—', alpha=0.7)
plt.legend()
plt.show()
```

运行，即可得到图 2。

3. Python 在空间解析几何绘图中的应用

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
# 设置参数
a = 2.0  # 半径参数
t = np.linspace(0, 2*np.pi, 200)  # 参数 t
# 计算交线的参数方程(维维安尼曲线)
x = a/2 * (1 + np.cos(t))
y = a/2 * np.sin(t)
```

```
z = a * np.sin(t/2)
# 创建 3D 图形
fig = plt.figure(figsize=(12, 10))
# 1. 绘制交线曲线
ax1 = fig.add_subplot(221, projection='3d')
ax1.plot(x, y, z, 'r-', linewidth=3, label='交线 L (维维安尼曲线)')
ax1.set_xlabel('X')
ax1.set_ylabel('Y')
ax1.set_zlabel('Z')
ax1.set_title('半球面与圆柱面的交线')
ax1.legend()
ax1.grid(True)
# 2. 绘制交线 + 半球面
ax2 = fig.add_subplot(222, projection='3d')
# 生成半球面数据
theta = np.linspace(0, 2*np.pi, 50)
phi = np.linspace(0, np.pi/2, 25)  # 只取上半部分
theta, phi = np.meshgrid(theta, phi)
X_hemi = a * np.sin(phi) * np.cos(theta)
Y_hemi = a * np.sin(phi) * np.sin(theta)
Z_hemi = a * np.cos(phi)
# 绘制半球面(半透明)
ax2.plot_surface(X_hemi, Y_hemi, Z_hemi, alpha=0.3, color='blue', label='半球面')
ax2.plot(x, y, z, 'r-', linewidth=3, label='交线')
ax2.set_xlabel('X')
ax2.set_ylabel('Y')
ax2.set_zlabel('Z')
ax2.set_title('交线与半球面')
ax2.set_zlim(0, a)
# 3. 绘制交线 + 圆柱面
ax3 = fig.add_subplot(223, projection='3d')
# 生成圆柱面数据
theta_cyl = np.linspace(0, 2*np.pi, 50)
z_cyl = np.linspace(-0.5, a, 50)  # 圆柱的高度范围
theta_cyl, z_cyl = np.meshgrid(theta_cyl, z_cyl)
X_cyl = a/2 * (1 + np.cos(theta_cyl))
Y_cyl = a/2 * np.sin(theta_cyl)
Z_cyl = z_cyl
# 绘制圆柱面(半透明)
ax3.plot_surface(X_cyl, Y_cyl, Z_cyl, alpha=0.3, color='green', label='圆柱面')
```

```
ax3.plot(x, y, z, 'r-', linewidth=3, label='交线')
ax3.set_xlabel('X')
ax3.set_ylabel('Y')
ax3.set_zlabel('Z')
ax3.set_title('交线与圆柱面')
ax3.set_zlim(0, a)
# 4. 三者在同一坐标系中
ax4 = fig.add_subplot(224, projection='3d')
ax4.plot_surface(X_hemi, Y_hemi, Z_hemi, alpha=0.2, color='blue')
ax4.plot_surface(X_cyl, Y_cyl, Z_cyl, alpha=0.2, color='green')
ax4.plot(x, y, z, 'r-', linewidth=4, label='交线')
ax4.set_xlabel('X')
ax4.set_ylabel('Y')
ax4.set_zlabel('Z')
ax4.set_title('三者关系总览')
ax4.set_zlim(0, a)
plt.tight_layout()
plt.show()
# 单独绘制更清晰的交线图
fig2 = plt.figure(figsize=(10, 8))
ax = fig2.add_subplot(111, projection='3d')
# 绘制交线曲线
ax.plot(x, y, z, 'r-', linewidth=4, label=f'维维安尼曲线 (a={a})')
# 添加坐标轴上的标记
# 曲线起点和终点
ax.scatter([a], [0], [0], color='blue', s=100, label='起点 (a,0,0)')
ax.scatter([0], [0], [a], color='green', s=100, label='最高点 (0,0,a)')
# 设置视角
ax.view_init(elev=30, azim=45)
ax.set_xlabel('X 轴', fontsize=12)
ax.set_ylabel('Y 轴', fontsize=12)
ax.set_zlabel('Z 轴', fontsize=12)
ax.set_title('维维安尼曲线(半球面与圆柱面的交线)', fontsize=14)
ax.legend()
ax.grid(True)
# 设置坐标轴范围
ax.set_xlim([-0.5, a+0.5])
ax.set_ylim([-a/2-0.5, a/2+0.5])
ax.set_zlim([0, a+0.5])
plt.tight_layout()
```

```
plt.show()
# 打印曲线特征点信息
print("=== 维维安尼曲线特征点 ===")
print(f"参数 a = {a}")
print("曲线方程(参数形式)：")
print(f"    x = {a/2}(1 + cos(t))")
print(f"    y = {a/2} sin(t)")
print(f"    z = {a} sin(t/2)")
print("\n 重要点坐标：")
print(f"    t=0:     ({a}, 0, 0)          # 起点")
print(f"    t=π:     (0, 0, {a})          # 最高点")
print(f"    t=2π:    ({a}, 0, 0)          # 回到起点")
print(f"    t=π/2: ({a/2}, {a/2}, {a/np.sqrt(2):.3f})")
```

运行，即可得到图 3。

4. Python 在三重积分计算中的应用

```
import numpy as np
from scipy.integrate import tplquad
# 定义被积函数 f(x, y, z) = x + y
def integrand(z, y, x):
    return x + y
# 积分区域 Ω: 由平面 x + y + z = 1 与坐标面围成的四面体
# x 范围: 0 到 1
# y 范围: 对于每个 x，y 从 0 到 1-x
# z 范围: 对于每个(x, y)，z 从 0 到 1-x-y
result, error = tplquad(
    integrand,                          # 被积函数
    0, 1,                               # x 范围[0, 1]
    lambda x: 0,                        # y 的下限(关于 x 的函数)
    lambda x: 1 - x,                    # y 的上限(关于 x 的函数)
    lambda x, y: 0,                     # z 的下限(关于 x, y 的函数)
    lambda x, y: 1 - x - y          # z 的上限(关于 x, y 的函数)
)
print("数值积分结果:", result)
print("误差估计:", error)
print("解析结果 1/12:", 1/12)
print("两者差值:", abs(result - 1/12))
```

运行，即可得到结果。