ĐẠI HỌC BÁCH KHOA HÀ NỘI

BÁO CÁO ĐỒ ÁN

Hướng dẫn sử dụng phần mềm SageMath và ứng dụng trong các học phần Toán đại cương

NGUYỄN TRUNG KIÊN
kien.nt227180@sis.hust.edu.vn

Ngành Hệ thống thông tin quản lý

Giảng viên hướng dẫn: TS. Lê Văn Tứ

(Chữ ký GVHD)

Bộ môn: Toán cơ bản

Khoa: Toán - Tin

HÀ NỘI, 06/2025

Nhận xét của GVHD

Mục tiêu đề tài

Xây dựng một tài liệu hướng dẫn sử dụng phần mềm SageMath để hỗ trợ giải các bài toán trong chương trình Toán đại cương bậc đại học.

Nội dung chính

Tài liệu trình bày tổng quan về phần mềm SageMath, hướng dẫn cài đặt, sử dụng, giới thiệu các tính năng cơ bản và nâng cao. Đồng thời, sinh viên đã xây dựng hệ thống ví dụ ứng dụng cụ thể theo từng học phần Giải tích, Đại số tuyến tính, Xác suất thống kê phù hợp với chương trình đào tạo tại Đại học Bách Khoa Hà Nội.

Kết quả đạt được

Sinh viên đã hoàn thành một tài liệu có cấu trúc rõ ràng, nội dung đầy đủ, dễ tiếp cận và có tính thực tiễn cao.

Ngoài tài liệu chính, sinh viên đã xây dựng thêm một tệp Notebook trực quan kèm toàn bộ mã lệnh đã biên soạn và đính kèm tại phần Phụ lục. Tài liệu này có thể được sử dụng như một tài nguyên tham khảo hữu ích cho sinh viên và giảng viên trong quá trình giảng dạy và học tập các học phần Toán.

Ý thức của sinh viên

Sinh viên thể hiện tinh thần làm việc nghiêm túc, cầu thị. Khi được giảng viên hướng dẫn giao đề tài, sinh viên có ý thức chủ động tìm hiểu các tài liệu bổ sung, sử dụng một cách có chọn lọc. Trong quá trình làm việc, sinh viên thể hiện sự tiến bộ về mặt phương pháp nghiên cứu và kiến thức trong lĩnh vực này.

Giảng viên hướng dẫn

Lê Văn Tứ

Lời nói đầu

Trong xu hướng chuyển đổi số giáo dục và tăng cường ứng dụng công nghệ trong giảng dạy – học tập, việc sử dụng các phần mềm hỗ trợ tính toán và trực quan hóa Toán học trở nên quan trọng hơn bao giờ hết. SageMath là một công cụ mã nguồn mở mạnh mẽ, không chỉ giúp người học giải quyết các bài toán phức tạp mà còn giúp hình thành tư duy toán học hiện đại, kết hợp lập trình và tư duy phân tích.

Tài liệu này được biên soạn nhằm mục tiêu giúp sinh viên nói riêng và người dùng nói chung tiếp cận SageMath từ cơ bản đến nâng cao, đồng thời cung cấp các ví dụ ứng dụng sát với nội dung của các học phần Toán đại cương tại Đại học Bách Khoa Hà Nội, bao gồm: Giải tích 1, Giải tích 2, Giải tích 3, Đại số tuyến tính và Xác suất thống kê.

Cấu trúc tài liệu được chia thành các chương rõ ràng, từ giới thiệu, hướng dẫn sử dụng phần mềm, các tính năng chính, các tính năng mở rộng đến phần ứng dụng thực tiễn trong từng học phần. Bên cạnh đó, tài liệu cũng cung cấp các phụ lục hữu ích như hướng dẫn khắc phục lỗi cài đặt và danh mục tài liệu tham khảo.

Trong quá trình thực hiện tài liệu này, em đã nhận được sự giúp đỡ và đóng góp ý kiến quý báu từ nhiều thầy cô, bạn bè và đồng nghiệp. Đặc biệt, em xin gửi lời cảm ơn sâu sắc đến thầy Lê Văn Tứ, giảng viên hướng dẫn, người đã tận tình chỉ bảo, định hướng và hỗ trợ em trong suốt quá trình nghiên cứu và biên soạn tài liệu.

Em cũng xin chân thành cảm ơn các bạn sinh viên trong lớp đã chia sẻ kinh nghiệm sử dụng SageMath, góp phần giúp em hoàn thiện nội dung. Sự động viên và góp ý của mọi người là nguồn động lực to lớn giúp em hoàn thành đồ án này.

Mặc dù đã rất cố gắng nhưng do thời gian hạn chế và kiến thức còn nhiều thiếu sót, tài liệu khó tránh khỏi những hạn chế và sai sót. Em rất mong nhận được sự góp ý từ thầy cô và bạn đọc để tài liệu được hoàn thiện hơn.

Nguyễn Trung Kiên
Email: kien.nt227180@sis.hust.edu.vn
Hà Nội, tháng 6 năm 2025

Mục lục

Danh sách hình vẽ

Danh sách bảng

Chương 1

Giới thiệu

1.1 Bối cảnh và lý do nghiên cứu

Trong bối cảnh cuộc cách mạng công nghiệp 4.0, việc tích hợp công nghệ vào giảng dạy và nghiên cứu Toán học ngày càng trở nên cần thiết. Các phần mềm tính toán như SageMath đã mở ra những hướng đi mới, giúp sinh viên và giảng viên tiếp cận với Toán học một cách trực quan, chính xác và hiệu quả hơn.

SageMath là một phần mềm mã nguồn mở mạnh mẽ, được phát triển nhằm mục tiêu cung cấp một nền tảng tính toán toàn diện, kết hợp nhiều hệ thống toán học khác nhau như Maxima, NumPy, SciPy, matplotlib, SymPy, GAP, FLINT, R,... Với triết lý *"Free Open-Source Mathematics Software"* (Phần mềm Toán học Mã nguồn mở Miễn phí), SageMath hướng đến việc thay thế các phần mềm thương mại như MATLAB, Mathematica và Maple trong nhiều ứng dụng.

1.2 Mục tiêu của đề tài

Mục tiêu của đề tài này là:

- Giới thiệu tổng quan về SageMath và các tính năng chính của phần mềm.
- Hướng dẫn cài đặt, làm quen với giao diện và cách sử dụng cơ bản.
- Trình bày các ứng dụng cụ thể của SageMath trong các học phần Toán học cương tại Đại học Bách Khoa Hà Nội.
- Cung cấp tài nguyên học tập, ví dụ minh hoạ và hướng dẫn xử lý các lỗi thường gặp khi sử dụng phần mềm.

1.3 Đối tượng sử dụng

Tài liệu này được thiết kế dành cho:

- Sinh viên các ngành kỹ thuật, công nghệ thông tin, toán ứng dụng, kinh tế... đang học các học phần Toán học cơ bản.
- Giảng viên, nhà nghiên cứu muốn tích hợp công cụ tính toán vào bài giảng và nghiên cứu khoa học.
- Người tự học, đam mê khám phá và ứng dụng Toán học với công cụ tính toán hiện đại.

1.4 Phạm vi và cấu trúc tài liệu

Tài liệu được chia thành các chương nhằm tiếp cận từ cơ bản đến nâng cao, với trọng tâm là các ví dụ thực hành có thể chạy trực tiếp trên SageMath. Nội dung trải dài từ hướng dẫn sử dụng, tính năng chính, ứng dụng trong từng học phần đến các tiện ích mở rộng và giải pháp khắc phục lỗi trong quá trình học tập và nghiên cứu.

Tài liệu không đi sâu vào thuật toán, cách xây dựng các hàm toán học đã có của phần mềm mà tập trung vào tính ứng dụng thực tiễn trong giảng dạy và học tập các môn Toán đại cương trên cấp bậc Đại học.

Chương 2

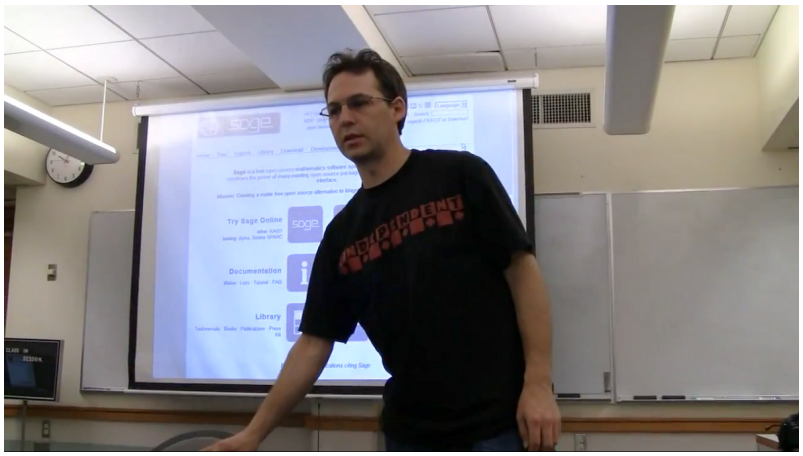
Tổng quan về SageMath

2.1 Sagemath là gì?

SageMath (viết tắt là Sage) là một phần mềm mã nguồn mở được thiết kế nhằm hỗ trợ các phép toán toán học từ cơ bản đến nâng cao. Sage được phát triển bởi giáo sư William Stein tại Đại học Washington vào năm 2005, với mục tiêu tạo ra một công cụ toán học mạnh mẽ nhưng hoàn toàn miễn phí, tương đương với các phần mềm thương mại như Mathematica, Maple, và MATLAB.

Hình 2.1: Logo phần mềm SageMath phiên bản 10.5

2.1.1 Ý tưởng và động lực phát triển



Hình 2.2: William A. Stein đang nói chuyện về Sage tại REU Toán học năm 2011 tại Đại học Washington

William Stein, một nhà toán học chuyên về lý thuyết số, nhận thấy rằng các công cụ toán học mạnh mẽ hiện có đều yêu cầu chi phí cao và không thân thiện với người dùng trong cộng đồng nghiên cứu toán học. Với tư duy mã nguồn mở, ông đã quyết định xây dựng một nền tảng kết hợp từ nhiều công cụ toán học hiện có như:

- **Maxima:** Hỗ trợ tính toán đại số trừu tượng.
- **NumPy và SciPy:** Tính toán số học và khoa học.
- **SymPy:** Tính toán trừu tượng.
- **R:** Phân tích thống kê.
- **Matplotlib:** Vẽ đồ thị và trực quan hóa.

2.1.2 Quá trình phát triển

Phiên bản đầu tiên của Sage được phát hành vào tháng 2 năm 2005 với tên gọi là "SAGE"(System for Algebra and Geometry Experimentation). Qua nhiều năm phát triển, Sage đã trở thành một hệ thống toán học hoàn chỉnh với khả năng tích hợp nhiều thư viện mạnh mẽ từ cộng đồng Python và các dự án mã nguồn mở khác.

Một cột mốc đáng chú ý trong lịch sử Sage là khi nó chuyển sang tên gọi chính thức là "SageMath"để nhấn mạnh vai trò là một hệ thống tính toán toán học. SageMath được sử dụng rộng rãi trong giảng dạy, nghiên cứu và công nghiệp nhờ tính linh hoạt và miễn phí của nó.

2.1.3 Tính mã nguồn mở

SageMath được phát triển dưới giấy phép GPL (GNU General Public License), nghĩa là bất kỳ ai cũng có thể sử dụng, sửa đổi và phân phối lại phần mềm. Điều này tạo ra một cộng đồng người dùng và phát triển lớn mạnh, với sự tham gia của nhiều nhà nghiên cứu và lập trình viên từ khắp nơi trên thế giới.

2.1.4 Những cột mốc quan trọng

- **2005:** Phát hành phiên bản đầu tiên của SageMath.
- **2007:** SageMath giành giải thưởng FSF (Free Software Foundation) cho dự án phần mềm tự do.
- **2013:** Ra mắt CoCalc (trước đây là SageMathCloud), cho phép sử dụng SageMath trực tuyến.
- **2020:** SageMath được tích hợp hoàn chỉnh với Jupyter Notebook, tạo thuận lợi cho việc sử dụng trong môi trường tính toán khoa học.

2.1.5 Tầm nhìn và tương lai

Mục tiêu dài hạn của SageMath là trở thành một hệ thống tính toán toán học mạnh mẽ, thân thiện với người dùng và hoàn toàn miễn phí. SageMath đang tiếp tục được phát triển và hoàn thiện bởi cộng đồng mã nguồn mở, nhằm mở rộng hơn nữa khả năng tính toán và ứng dụng trong nhiều lĩnh vực khoa học, giáo dục và công nghiệp.

2.2 So sánh SageMath với các phần mềm Toán học khác

Tiêu chí	**SageMath**	**Mathematica / Maple**	**MATLAB**
Mã nguồn	Mở hoàn toàn	Đóng	Đóng
Ngôn ngữ lập trình	Python	Ngôn ngữ riêng biệt	MATLAB
Tính toán trừu tượng (Symbolic)	Có (qua SymPy, Maxima)	Rất mạnh	Yếu
Tính toán số	Mạnh (qua NumPy, SciPy)	Có	Rất mạnh
Tính linh hoạt	Cao nhờ tích hợp nhiều thư viện	Trung bình	Trung bình
Khả năng mở rộng	Dễ dàng thông qua thư viện Python	Hạn chế	Có nhưng không đa dạng như Python
Chi phí sử dụng	Miễn phí	Trả phí cao	Trả phí cao

Bảng 2.1: So sánh SageMath với một số phần mềm toán học khác

SageMath nổi bật nhờ khả năng tích hợp các thư viện hiện đại và tận dụng cộng đồng mã nguồn mở để phát triển liên tục.

2.3 Hướng dẫn cài đặt SageMath

Cài đặt trên Windows

- Truy cập trang chính thức: `https://www.sagemath.org/download.html`
- Tải bản cài đặt phù hợp với hệ điều hành.
- Cài đặt như một phần mềm thông thường.
- Đối với người dùng nâng cao, có thể sử dụng Windows Subsystem for Linux (WSL) để cài Sage trên Ubuntu.

Cài đặt trên Linux (Ubuntu)

```
sudo apt update
sudo apt install sagemath
```

Cài đặt trên macOS

Có thể sử dụng Homebrew:

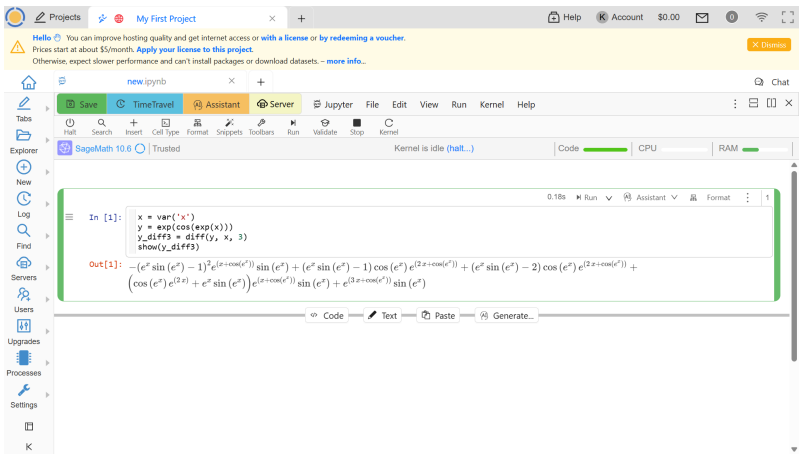
```
brew install --cask sagemath
```

2.4 Sử dụng SageMath trực tuyến không cần cài đặt

Trong trường hợp người dùng không muốn cài đặt SageMath trực tiếp trên máy tính, có thể sử dụng các nền tảng trực tuyến để chạy các mã SageMath mà không cần cấu hình phức tạp.

2.4.1 CoCalc - Collaborative Calculation

CoCalc (trước đây là SageMathCloud) là một dịch vụ trực tuyến giúp người dùng sử dụng SageMath ngay trên trình duyệt web mà không cần cài đặt. CoCalc hỗ trợ lập trình SageMath, Python, Jupyter Notebook và nhiều công cụ khác.



Hình 2.3: Giao diện SageMath Notebook trên nền tảng trực tuyến CoCalc

Hướng dẫn sử dụng CoCalc:

- Truy cập trang web: `https://cocalc.com`
- Đăng ký tài khoản hoặc sử dụng tài khoản Google để đăng nhập.
- Tạo một dự án mới (Project) và chọn môi trường SageMath.

- Tạo một file mới có phần mở rộng `.sagews` hoặc `.ipynb` để bắt đầu viết mã SageMath.

Ưu điểm:

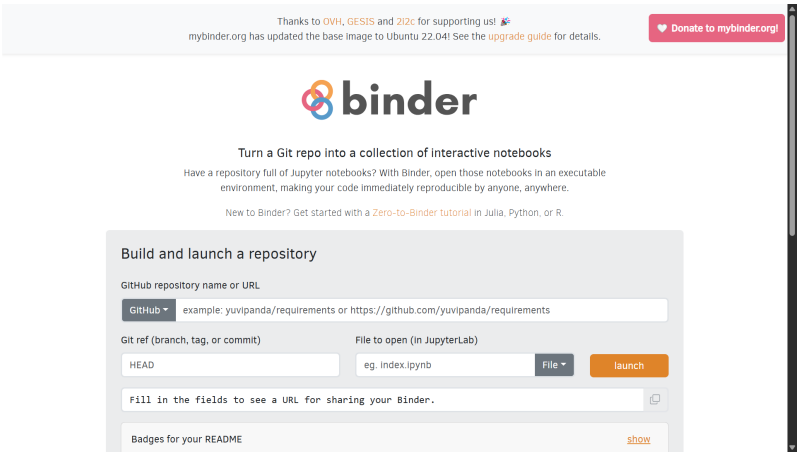
- Trực tiếp trên trình duyệt, không cần cài đặt.
- Hỗ trợ cộng tác và chia sẻ với người khác.
- Chạy được các notebook Jupyter có tích hợp SageMath.

Nhược điểm:

- Phiên bản miễn phí có giới hạn tài nguyên.
- Cần kết nối internet ổn định.

2.4.2 Binder - Chạy Notebook SageMath từ GitHub

Binder là một dịch vụ trực tuyến cho phép chạy các notebook từ GitHub mà không cần cài đặt. Đây là giải pháp hiệu quả cho việc trình bày các đồ án và chia sẻ với giảng viên.



Hình 2.4: Giao diện nền tảng Binder

Hướng dẫn sử dụng Binder:

- Truy cập trang web: `https://mybinder.org`
- Nhập đường dẫn đến kho GitHub chứa file notebook.
- Chọn kernel SageMath nếu có, nhấn `Launch` để bắt đầu.
- Notebook sẽ được tải lên máy chủ và có thể chạy trực tuyến.

Ưu điểm:

- Không cần đăng ký tài khoản.

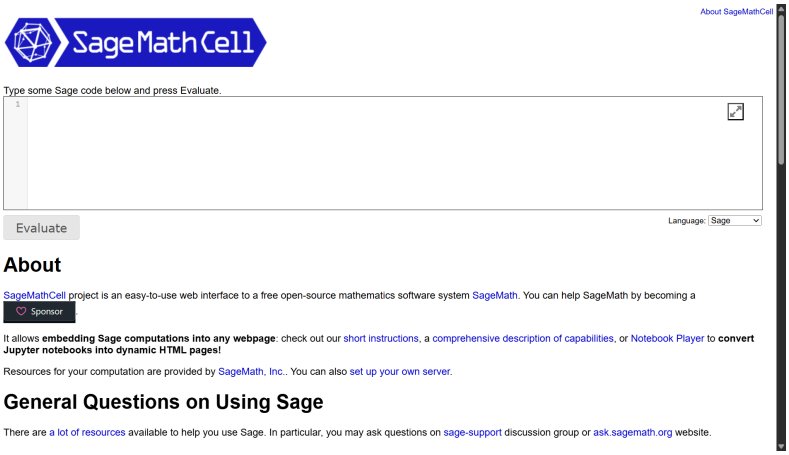
- Dễ dàng chia sẻ với đường dẫn *(link)* công khai.

Nhược điểm:

- Khởi động lâu nếu dự án có nhiều thư viện.
- Không lưu lại trạng thái làm việc sau khi ngắt kết nối.

2.4.3 Sage Cell - Tính toán nhanh với SageMath

Sage Cell là công cụ trực tuyến cho phép thực thi các đoạn mã SageMath nhỏ mà không cần cài đặt.



Hình 2.5: Giao diện trang web Sage Cell

Hướng dẫn sử dụng Sage Cell:

- Truy cập trang web: `https://sagecell.sagemath.org`
- Nhập mã SageMath vào ô nhập liệu.
- Nhấn `Evaluate` để xem kết quả.

Ưu điểm:

- Nhanh chóng, không cần đăng nhập.
- Thích hợp cho các tính toán nhỏ.

Nhược điểm:

- Không phù hợp với các dự án lớn hoặc tính toán phức tạp.
- Không có khả năng lưu trữ và cộng tác.

2.4.4 So sánh các nền tảng trực tuyến

Nền tảng	Tính năng	Ưu điểm	Nhược điểm
CoCalc	Tương tác mạnh mẽ	Hỗ trợ nhiều công cụ	Cần đăng ký tài khoản
Binder	Chạy từ GitHub	Dễ chia sẻ	Khởi động chậm
Sage Cell	Thực thi mã nhanh	Không cần đăng nhập	Chỉ tính toán nhỏ

Bảng 2.2: So sánh các nền tảng trực tuyến sử dụng SageMath

2.5 Giao diện và cách sử dụng cơ bản

Sau khi cài đặt, người dùng có thể sử dụng SageMath theo các cách sau:

- Giao diện dòng lệnh (CLI): khởi động bằng lệnh `sage`.
- Giao diện notebook (Jupyter): chạy lệnh `sage -n jupyter`.
- Giao diện đồ họa qua CoCalc nếu dùng trực tuyến.

Ví dụ cơ bản

```
# Tinh tong 1 + 2 + ... + 100
sum(i for i in (1..100))

# Giai thua cua 10
factorial(10)

# Dao ham cua ham so
f(x) = x^3 + x^2
diff(f(x), x)

# Ve do thi
plot(sin(x), (x, -2*pi, 2*pi))
```

Sage hỗ trợ gần như toàn bộ các phép toán cơ bản đến nâng cao, cho phép nhập công thức trực tiếp theo cú pháp giống toán học thông thường.

Chương 3

Các tính năng cốt lõi của SageMath

3.1 Ngôn ngữ và biểu thức cơ bản trong SageMath

3.1.1 Tổng quan cú pháp và quy tắc trong SageMath

SageMath sử dụng cú pháp rất giống với Python, nhưng có một số điểm đặc biệt để hỗ trợ tính toán toán học. Dưới đây là một số quy tắc cơ bản mà người dùng cần nắm khi làm việc với SageMath.

3.1.1.1 Quy tắc chung khi viết lệnh trong SageMath

- **Biến và tên hàm**: SageMath phân biệt chữ hoa và chữ thường, vì vậy `x` và `X` được coi là hai biến khác nhau. Cũng giống như Python, các biến và hàm có thể được sử dụng mà không cần khai báo kiểu dữ liệu.

- **Dấu cách**: Dấu cách không ảnh hưởng đến cú pháp trong SageMath, nhưng việc sử dụng dấu cách hợp lý sẽ giúp mã dễ đọc hơn và giảm khả năng xảy ra lỗi.

- **Ký tự toán học đặc biệt**: SageMath sử dụng các ký hiệu toán học quen thuộc như `+` (cộng), `-` (trừ), `*` (nhân), `/` (chia), cùng với nhiều toán tử khác để thực hiện các phép toán.

- **Lệnh kết thúc**: Không giống như nhiều ngôn ngữ lập trình khác, SageMath không yêu cầu dấu chấm phẩy (`;`) để kết thúc lệnh, tuy nhiên bạn vẫn có thể sử dụng dấu đó nếu muốn viết nhiều lệnh trên cùng một dòng.

3.1.1.2 Cú pháp và quy tắc đặc biệt

- **Hàm số và đại số**: SageMath cung cấp nhiều hàm tích hợp sẵn để làm việc với biểu thức đại số, bao gồm các hàm lượng giác (`sin`, `cos`, `tan`), hàm mũ (`exp`), logarit (`log`), và nhiều hàm toán học khác.

- **Đặt tên cho biến**: Việc đặt tên cho biến trong SageMath cần tuân theo các quy tắc sau:

  – Tên biến phải bắt đầu bằng một chữ cái (a–z, A–Z) hoặc dấu gạch dưới (_).

  – Các ký tự tiếp theo có thể là chữ cái, chữ số (0–9) hoặc dấu gạch dưới.

  – Không được sử dụng các tên đã được định nghĩa trước trong SageMath, như các từ khóa hoặc tên hàm sẵn có (`sin`, `log`, `var`, `for`, `if`, ...).

3.1.1.3 Chạy mã trong SageMath

Để thực thi các lệnh trong SageMath, bạn có thể nhập trực tiếp vào terminal của SageMath hoặc sử dụng môi trường lập trình như Jupyter Notebook. Mã nguồn trong SageMath có thể được viết và lưu vào các file `.sage` hoặc `.py`.

Ví dụ về cách chạy một file `.sage`:

```
$ sage myscript.sage
```

Trong trường hợp bạn sử dụng Jupyter Notebook, bạn có thể chạy từng ô mã (`cell`) một cách độc lập.

3.1.1.4 Quản lý và sử dụng thư viện trong SageMath

SageMath cung cấp một hệ thống quản lý thư viện mạnh mẽ. Bạn có thể nhập thư viện vào mã của mình để mở rộng chức năng tính toán, ví dụ như nhập các thư viện đại số, giải tích, ...

Ví dụ nhập thư viện để làm việc với các số nguyên lớn:

```
sage: from sage.modules.free_module import FreeModule
```

3.1.2 Biến và kiểu dữ liệu cơ bản

Trong SageMath, các biến và kiểu dữ liệu cơ bản được sử dụng để thực hiện các phép toán số học, đại số và nhiều phép toán khác. Việc hiểu rõ về các kiểu dữ liệu này là rất quan trọng khi làm việc với SageMath. Dưới đây là các kiểu dữ liệu cơ bản mà bạn sẽ gặp phải khi sử dụng SageMath.

3.1.2.1 Khai báo và sử dụng biến

Biến trong SageMath có thể được khai báo trực tiếp bằng cách gán giá trị cho chúng. SageMath hỗ trợ nhiều loại dữ liệu khác nhau, và bạn không cần phải chỉ định kiểu dữ liệu khi khai báo biến.

Ví dụ:

```
sage: x = 10  # Bien x la mot so nguyen
sage: y = 3.14  # Bien y la mot so thuc
sage: z = "SageMath"  # Bien z la mot chuoi
```

Ở trên, chúng ta đã khai báo ba biến với ba kiểu dữ liệu khác nhau:

- `x` là số nguyên (`integer`),
- `y` là số thực (`float`),
- `z` là chuỗi ký tự (`string`).

3.1.2.2 Kiểu dữ liệu cơ bản trong SageMath

SageMath hỗ trợ một số kiểu dữ liệu cơ bản, bao gồm:

- Số nguyên: Các số nguyên trong SageMath có thể có giá trị âm, dương hoặc bằng không.
- Số thực: Là các số có phần thập phân, được sử dụng để làm việc với các phép toán liên quan đến số học.
- Chuỗi: Là một chuỗi các ký tự, thường được sử dụng để lưu trữ văn bản.
- Danh sách: Là một kiểu dữ liệu để lưu trữ các giá trị trong một dãy. Các giá trị có thể là bất kỳ kiểu dữ liệu nào.
- Boolean: Kiểu dữ liệu Boolean có hai giá trị là `True` và `False`.
- Ma trận và vector: Được sử dụng để làm việc với đại số tuyến tính.
- Số nguyên lớn: SageMath hỗ trợ số nguyên có kích thước lớn hơn nhiều so với kiểu dữ liệu chuẩn của Python.

3.1.2.3 Ví dụ về các kiểu dữ liệu trong SageMath

Dưới đây là một số ví dụ về cách sử dụng các kiểu dữ liệu cơ bản trong SageMath.

```
sage: a = 15  # So nguyen
sage: b = 3.5  # So thuc
sage: c = "Hello, Sage!"  # Chuoi
sage: lst = [1, 2, 3, 4, 5]  # Danh sach
sage: flag = True  # Boolean
```

3.1.2.4 Chuyển đổi giữa các kiểu dữ liệu

SageMath cho phép bạn chuyển đổi giữa các kiểu dữ liệu khác nhau một cách dễ dàng.

Ví dụ, bạn có thể chuyển một số thực thành số nguyên:

```
sage: a = 5.7
sage: int(a)  # Chuyen doi thanh so nguyen
5
```

Hoặc chuyển đổi một chuỗi thành danh sách các ký tự:

```
sage: s = "Sage"
sage: list(s)  # Chuyen doi thanh danh sach ki tu
['S', 'a', 'g', 'e']
```

3.1.2.5 Cách làm việc với các tập hợp

SageMath cũng hỗ trợ kiểu dữ liệu tập hợp, cho phép bạn thực hiện các phép toán trên các tập hợp. Dưới đây là ví dụ về cách tạo và thao tác với tập hợp trong SageMath.

```
sage: A = Set([1, 2, 3, 4])
sage: B = Set([3, 4, 5, 6])
sage: A.union(B)  # Phep hop
Set([1, 2, 3, 4, 5, 6])
sage: A.intersubsection(B)  # Phep giao
Set([3, 4])
```

3.1.2.6 Thao tác với các số nguyên lớn

Một trong những tính năng mạnh mẽ của SageMath là khả năng làm việc với các số nguyên lớn. Bạn có thể khai báo và thao tác với các số nguyên có giá trị cực kỳ lớn mà không gặp phải giới hạn như trong các ngôn ngữ lập trình khác.

Ví dụ:

```
sage: big_number = 12345678912345678912345678 9
sage: big_number * 2  # Phep nhan voi mot so lon
246913578246913578246913578
```

3.1.3 Lệnh `show()` trong SageMath

Lệnh `show()` trong SageMath dùng để hiển thị biểu thức, kết quả tính toán và trực quan hóa đồ thị dưới dạng ký hiệu toán học đẹp mắt.

3.1.3.1 Hiển thị biểu thức toán học

Lệnh `show()` giúp biểu diễn kết quả dưới dạng ký hiệu thay vì dạng văn bản thông thường.

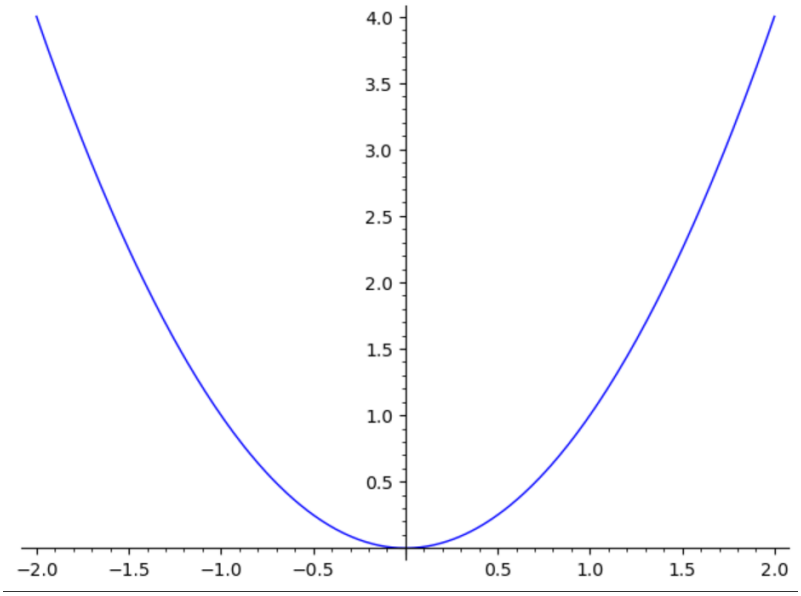
```
# Hien thi bieu thuc
sage: var('x')
sage: f = x^2 + 2*x + 1
sage: show(f)
```

$$x^2 + 2x + 1$$

3.1.3.2 Hiển thị đồ thị

Lệnh `show()` được sử dụng để vẽ và hiển thị các đồ thị trong không gian 2D và 3D.

```
# Ve do thi ham so
sage: plot(x^2, (x, -2, 2)).show()
```



Hình 3.1: Đồ thị hàm số $y = x^2$ trên đoạn $[-2, 2]$

3.1.3.3 Hiển thị ma trận

```
# Hien thi ma tran
sage: M = matrix([[1, 2], [3, 4]])
sage: show(M)
```

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

Lệnh `show()` rất hữu ích khi cần trực quan hóa kết quả và trình bày các phép toán.

3.1.4 Các phép toán số học và đại số cơ bản

SageMath hỗ trợ một loạt các phép toán số học và đại số cơ bản, giúp người dùng thực hiện các phép toán cơ bản trong toán học như cộng, trừ, nhân, chia, và các phép toán đại số với các đối tượng phức tạp hơn như đa thức, số nguyên lớn, và các phép toán với các trường số học khác. Dưới đây là các phép toán cơ bản mà bạn sẽ sử dụng thường xuyên trong SageMath.

3.1.4.1 Các phép toán số học cơ bản

SageMath hỗ trợ các phép toán số học thông thường như cộng, trừ, nhân, chia, lũy thừa. Các phép toán này có thể được thực hiện trực tiếp trên các số nguyên, số thực hoặc các đối tượng số học khác.

Ví dụ về phép toán số học cơ bản:

```
sage: a = 10  # Phep gan
sage: b = 5  # Phep gan
sage: a + b  # Cong
15
sage: a - b  # Tru
5
sage: a * b  # Nhan
50
sage: a / b  # Chia
2.0
sage: a^b  # Luy thua
100000
```

3.1.4.2 Các phép toán với số nguyên lớn

SageMath hỗ trợ làm việc với các số nguyên có độ dài lớn, vượt quá giới hạn của các kiểu dữ liệu số nguyên trong các ngôn ngữ lập trình khác. Điều này đặc biệt hữu ích khi làm việc với các bài toán lý thuyết số, mã hóa, và mật mã học.

Ví dụ:

```
sage: big_num = 12345678912345678912345678 9
sage: big_num + 1000  # Phep cong voi so lon
123456789123456789123457789
sage: big_num * 1000000  # Phep nhan voi so lon
123456789123456789123456789000000
```

3.1.4.3 Các phép toán với đa thức

SageMath hỗ trợ đầy đủ các phép toán đại số với đa thức, bao gồm cộng, trừ, nhân, chia, và các phép toán khác. Để làm việc với đa thức, bạn có thể sử dụng các hàm có sẵn trong SageMath như `PolynomialRing`.

Ví dụ về các phép toán với đa thức:

```
sage: R.<x> = PolynomialRing(QQ)  # Khai bao da thuc voi
    bien x
sage: f = x^2 + 2*x + 1  # Da thuc f(x) = x^2 + 2x + 1
sage: g = x^2 - 1  # Da thuc g(x) = x^2 - 1
sage: f + g  # Cong hai da thuc
x^2 + 2*x + 1 + x^2 - 1
sage: f * g  # Nhan hai da thuc
x^4 + 2*x^3 - x^2 - 2*x - 1
sage: f / g  # Chia hai da thuc
(x^2 + 2*x + 1)/(x^2 - 1)
```

3.1.4.4 Phép chia lấy dư

SageMath hỗ trợ phép chia lấy dư, cho phép bạn chia hai số nguyên và trả về thương và số dư. Điều này rất hữu ích trong lý thuyết số, đặc biệt là khi làm việc với các thuật toán như thuật toán Euclid.

Ví dụ về phép chia dư:

```
sage: a = 17
sage: b = 5
sage: divmod(a, b)  # Chia du giua 17 va 5
(3, 2)  # (Thuong, So du)
```

3.1.4.5 Các phép toán đại số cơ bản

SageMath không chỉ hỗ trợ các phép toán số học cơ bản mà còn hỗ trợ các phép toán đại số với các đối tượng trừu tượng như nhóm, vành và trường.

Ví dụ về các phép toán nhóm:

```
sage: G = SymmetricGroup(3)  # Nhom doi xung S3
sage: G[1] * G[2]  # Nhan hai phan tu trong nhom
(1, 2)(3)
```

3.1.4.6 Phép toán lũy thừa

SageMath hỗ trợ các phép toán lũy thừa với các số nguyên, số thực và các đối tượng đại số. Phép toán lũy thừa có thể được thực hiện bằng ký hiệu mũ ^ hoặc dùng hàm `pow()` (Hàm có sẵn trong ngôn ngữ Python).

Ví dụ về lũy thừa:

```
# Luy thua so nguyen
sage: 2^5
32

# Cach khac su dung hai dau sao
```

```
sage: 2**5
32

# Luy thua so thuc
sage: 3.5^2
12.25

# Cach khac voi so thuc
sage: pow(3.5, 2)
12.25
```

Lưu ý: Ký hiệu ** tương tự như ■ để tính lũy thừa, nhưng trong một số trường hợp, ** được ưu tiên hơn để tránh lỗi cú pháp.

3.2 Đại số, giải tích và hình học

3.2.1 Làm việc với phương trình và hệ phương trình

Trong SageMath, bạn có thể giải các phương trình đại số đơn giản đến phức tạp, bao gồm phương trình bậc một, bậc hai, và hệ phương trình đại số. SageMath cung cấp các công cụ mạnh mẽ để giải phương trình với một biến hoặc nhiều biến, đồng thời hỗ trợ các phương trình với các hệ số phức tạp.

3.2.1.1 Giải phương trình đơn biến

Để giải một phương trìnhn, bạn có thể sử dụng hàm `solve()`, nơi đầu vào là phương trình cần giải.

Ví dụ về giải phương trình bậc nhất một ẩn:

```
sage: x = var('x')  # Dinh nghia bien x
sage: solve(x + 5 == 0, x)  # Giai phuong trinh x + 5 = 0
[x == -5]
```

SageMath trả về kết quả dưới dạng danh sách các nghiệm. Trong ví dụ trên, phương trình $x + 5 = 0$ có nghiệm $x = -5$.

Ví dụ về giải phương trình bậc hai:

```
sage: solve(x^2 + 5*x + 6 == 0, x)  # Giai phuong trinh bac
    hai x^2 + 5x + 6 = 0
[x == -3, x == -2]
```

Phương trình bậc hai $x^2 + 5x + 6 = 0$ có hai nghiệm là $x = -2$ và $x = -3$.

3.2.1.2 Giải phương trình đa biến

SageMath cũng hỗ trợ giải các phương trình có nhiều biến. Bạn có thể định nghĩa nhiều biến và sử dụng hàm `solve()` để giải các hệ phương trình.

Ví dụ về giải hệ phương trình với hai biến:

```
sage: y = var('y')  # Dinh nghia them bien y
sage: solve([x + y == 3, x - y == 1], x, y)  # Giai he
    phuong trinh x + y = 3 va x - y = 1
[x == 2, y == 1]
```

Kết quả là hệ phương trình có nghiệm $x = 2$ và $y = 1$.

3.2.1.3 Giải phương trình với các hệ số phức tạp

SageMath cũng cho phép bạn giải các phương trình với các hệ số phức tạp, bao gồm số phức hoặc các biểu thức đại số phức tạp.

Ví dụ về giải phương trình với số phức:

```
sage: z = var('z', domain=CC)  # Dinh nghia bien Z trong
    tap so phuc
sage: solve(z^2 + 1 == 0, z)  # Giai phuong trinh z^2 + 1 =
    0 trong tap so phuc
[z == -I, z == I]
```

Phương trình $z^2 + 1 = 0$ có hai nghiệm là $z = i$ và $z = -i$, với i là đơn vị ảo.

3.2.1.4 Giải hệ phương trình đại số phi tuyến

SageMath cũng hỗ trợ giải các hệ phương trình đại số phi tuyến, bao gồm các phương trình bậc cao hoặc các phương trình có hàm số phi tuyến.

Ví dụ về hệ phương trình phi tuyến:

```
sage: solve([x^2 + y^2 == 1, x^2 - y^2 == 0], x, y)  # Giai
    he phuong trinh phi tuyen
[x == sqrt(2)/2, y == sqrt(2)/2]
```

Hệ phương trình đại số phi tuyến $x^2 + y^2 = 1$ và $x^2 - y = 0$ có nghiệm $x = \frac{\sqrt{2}}{2}$ và $y = \frac{\sqrt{2}}{2}$.

3.2.1.5 Sử dụng phương trình trong các bài toán ứng dụng

SageMath không chỉ giải các phương trình đơn giản mà còn được sử dụng để giải các phương trình trong các bài toán ứng dụng. Bạn có thể sử dụng phương trình để mô hình hóa các bài toán trong các lĩnh vực như vật lý, kỹ thuật, kinh tế học, và khoa học máy tính.

Ví dụ về bài toán mô phỏng sự rơi tự do của vật thể:

```
sage: t = var('t')  # Dinh nghia bien thoi gian t
sage: h = 100 - 5*t^2  # Phuong trinh mo ta chieu cao cua
    vat the theo thoi gian t
solve(h == 0, t)  # Tim thoi gian vat the cham dat (khi h =
    0)
[t == -2*sqrt(5), t == 2*sqrt(5)]
```

Kết quả là $t = \sqrt{20}$, nghĩa là vật thể chạm đất sau khoảng 4.47 giây.

3.2.2 Tính toán đại số trừu tượng (nhóm, vành, trường, đa thức)

SageMath cung cấp các công cụ mạnh mẽ để làm việc với đại số trừu tượng, bao gồm các cấu trúc đại số cơ bản như vành, nhóm, trường và các phép toán với đa thức. Các cấu trúc này đóng vai trò quan trọng trong các bài toán toán học, vật lý và khoa học máy tính.

3.2.2.1 Làm việc với nhóm

Nhóm là một tập hợp các phần tử với một phép toán nhị phân thỏa mãn các tính chất: đóng, có phần tử trung tính, mỗi phần tử có phần tử nghịch đảo.

Ví dụ về nhóm phép cộng trên các số nguyên modulo 7:

```
sage: G = Group([1, 2, 3, 4, 5, 6], operation='addition␣mod
   ␣7')  # Nhom cac so nguyen modulo voi phep cong
sage: G.is_group()  # Kiem tra G co la 1 nhom khong
True
```

Phép toán 'operation='addition mod 7'' xác định phép cộng trong nhóm modulo 7. SageMath cung cấp phương thức `is_group()` để kiểm tra xem tập hợp có là một nhóm không.

3.2.2.2 Làm việc với vành

Vành là một cấu trúc đại số có hai phép toán: cộng và nhân, và phải thỏa mãn một số tính chất nhất định. Ví dụ về vành các số nguyên modulo 7:

```
sage: R = Ring([0, 1, 2, 3, 4, 5, 6], addition='mod␣7',
   multiplication='mod␣7')  # Vanh cac so nguyen modulo 7
sage: R.is_ring()  # Kiem tra R co la 1 vanh khong
True
```

Trong ví dụ này, ta tạo ra một vành với phép cộng và phép nhân là phép toán modulo 7.

3.2.2.3 Làm việc với trường

Trường là một cấu trúc đại số trong đó các phép toán cộng, trừ, nhân và chia đều được xác định và có tính nghịch đảo. Trường phổ biến nhất là trường số thực và số phức.

Ví dụ về trường số thực:

```
sage: F = RealField()  # Tao truong so thuc
sage: F.is_field()  # Kiem tra F co la 1 truong khong
```

```
True
```

SageMath cũng hỗ trợ các trường hữu hạn. Ví dụ về trường hữu hạn:

```
sage: GF(7)  # Truong huu han F_7
GF(7)
```

Trường này bao gồm các phần tử {0, 1, 2, 3, 4, 5, 6}, và các phép toán cộng, nhân, chia đều được thực hiện modulo 7.

3.2.2.4 Làm việc với đa thức

SageMath cung cấp các công cụ mạnh mẽ để làm việc với đa thức, từ các phép toán cơ bản đến các phép toán phức tạp. Bạn có thể thực hiện cộng, trừ, nhân, chia đa thức, giải phương trình đa thức, và làm việc với các đa thức trong các trường hoặc vành.

Ví dụ về tạo đa thức trong một biến:

```
sage: R.<x> = PolynomialRing(ZZ)  # Tao da thuc trong vanh
   da thuc co he so nguyen
sage: p = x^2 + 2*x + 1  # Dinh nghia da thuc
sage: p.factor()  # Phan tich da thuc thanh nhan tu
(x + 1)^2
```

SageMath phân tích đa thức $x^2 + 2x + 1$ thành $(x + 1)^2$, điều này giúp ích trong việc giải các phương trình đa thức.

Ví dụ về chia đa thức:

```
sage: q, r = (x^3 + 2*x^2 + x + 1).div(x^2 + x)  # Chia da
   thuc
sage: q, r  # Thuong va phan du
(x + 1, 1)
```

Ở đây, $x^3 + 2x^2 + x + 1$ chia cho $x^2 + x$ có thương là $x + 1$ và phần dư là 1.

3.2.2.5 Làm việc với vành đa thức

Trong SageMath, bạn có thể làm việc với các vành đa thức, nơi phép cộng và nhân được xác định trên các đa thức.

Ví dụ về vành đa thức:

```
sage: R.<x> = PolynomialRing(ZZ, 'x')  # Vanh cac da thuc
   voi he so la so nguyen
sage: I = ideal(x^2 + 1, R)  # Dinh nghia mien cua vanh da
   thuc
sage: I.is_ideal()  # Kiem tra I co la mot ideal khong
True
```

3.2.3 Ma trận, vector và đại số tuyến tính

SageMath cung cấp các công cụ mạnh mẽ để làm việc với ma trận, vector và thực hiện các phép toán đại số tuyến tính. Các phép toán này rất quan trọng trong các lĩnh vực như học máy, kỹ thuật, khoa học dữ liệu, và các nghiên cứu toán học.

3.2.3.1 Làm việc với vector

Vector là các đối tượng có thể đại diện cho các điểm trong không gian hoặc các phép toán đại số tuyến tính. Trong SageMath, vector có thể được định nghĩa trong các không gian số học khác nhau như $\mathbb{R}^n$ hoặc $\mathbb{C}^n$.

Ví dụ tạo một vector trong không gian thực:

```
sage: v = vector([1, 2, 3])  # Vector trong khong gian R^3
sage: v
(1, 2, 3)
```

Bạn có thể thực hiện các phép toán cơ bản trên vector như cộng, trừ, nhân với một số vô hướng (scalar), và tính độ dài:

```
sage: v + vector([4, 5, 6])  # Cong vector
(5, 7, 9)
sage: v * 2  # Nhan vector voi mot so vo huong
(2, 4, 6)
sage: v.norm()  # Tinh do dai cua vector
3.7416573867739413
```

Ngoài các phép toán cơ bản, bạn cũng có thể tính tích vô hướng và tích có hướng của các vector.

Tích vô hướng của hai vector là một phép toán trả về một số vô hướng, được tính bằng tổng các tích của các thành phần tương ứng của hai vector. Ví dụ:

```
sage: u = vector([1, 2, 3])
sage: w = vector([4, 5, 6])
sage: u.dot_product(w)  # Tich vo huong
32
```

Tích có hướng (hoặc tích chéo) là một phép toán giữa hai vector trong không gian ba chiều, cho ra một vector vuông góc với cả hai vector ban đầu. Ví dụ:

```
sage: u.cross_product(w)  # Tich co huong
(-3, 6, -3)
```

3.2.3.2 Làm việc với ma trận

Ma trận là các bảng số học dùng để biểu diễn các hệ phương trình tuyến tính, phép biến đổi tuyến tính, hoặc các phép toán đại số tuyến tính khác. SageMath cho phép bạn dễ dàng tạo ma trận và thực hiện các phép toán trên ma trận như cộng, nhân, tính định thức, và nghịch đảo.

Ví dụ tạo một ma trận và thực hiện phép cộng:

```
sage: M = Matrix([[1, 2], [3, 4]])  # Ma tran 2x2
sage: M
[1 2]
[3 4]
sage: M + Matrix([[5, 6], [7, 8]])  # Cong hai ma tran
[6 8]
[10 12]
```

Phép nhân ma trận:

```
sage: M * Matrix([[5], [6]])  # Nhan ma tran voi vector
[17]
[39]
```

Tính định thức của ma trận:

```
sage: M.det()  # Tinh dinh thuc cua ma tran
-2
```

Tính nghịch đảo của ma trận (nếu có):

```
sage: M.inv()  # Tinh ma tran nghich dao
[-2.0  1.0]
[ 1.5 -0.5]
```

3.2.3.3 Làm việc với hệ phương trình tuyến tính

SageMath hỗ trợ giải các hệ phương trình tuyến tính bằng cách sử dụng ma trận. Ví dụ, để giải một hệ phương trình tuyến tính $A \cdot x = b$, bạn có thể sử dụng phương thức `solve()`.

Giải hệ phương trình $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 5 \\ 6 \end{bmatrix}$:

```
sage: A = Matrix([[1, 2], [3, 4]])  # Ma tran he phuong
    trinh
sage: b = vector([5, 6])  # Vector ben phai
sage: A.solve_right(b)  # Giai he phuong trinh
(0, 1)
```

Phương thức `solve_right()` trả về nghiệm của hệ phương trình.

3.2.3.4 Giá trị riêng và vectơ riêng

Một trong những phép toán quan trọng trong đại số tuyến tính là tính giá trị riêng và vectơ riêng của ma trận. SageMath cung cấp phương thức `eigenvalues()` và `eigenvectors()` để tính toán các giá trị này.

Tính giá trị riêng của ma trận:

```
sage: M.eigenvalues()  # Tinh gia tri rieng cua ma tran
[5.0, -0.9999999999999999]
```

Tính vectơ riêng của ma trận:

```
sage: M.eigenvectors()  # Tinh vector rieng cua ma tran
[(5.0, 1, [(1, 1)]), (-1.0, 1, [(1, -2)])]
```

3.2.3.5 Các phép toán đại số tuyến tính khác

SageMath hỗ trợ nhiều phép toán đại số tuyến tính khác như phân rã ma trận, giải các hệ phương trình, và tính toán với không gian vector. Các phương thức khác bao gồm:

- `row_space()`, `column_space()`: Tính không gian hàng và không gian cột. - `rank()`: Tính hạng của ma trận. - `eigenvalues()`: Tính giá trị riêng. - `svd()`: Phân rã giá trị kỳ dị (Singular Value Decomposition).

Ví dụ tính hạng ma trận:

```
sage: M.rank()  # Tinh hang cua ma tran
2
```

3.2.4 Đạo hàm, tích phân và giải tích

SageMath hỗ trợ tính toán đạo hàm, tích phân và các phép toán giải tích khác như giải phương trình vi phân, tính toán số học chính xác cao, và làm việc với các hàm số. Phần này sẽ trình bày cách sử dụng các công cụ giải tích cơ bản trong SageMath để giải quyết các bài toán toán học liên quan đến đạo hàm, tích phân và các phép toán tiếp theo.

3.2.4.1 Đạo hàm

Đạo hàm là một trong những phép toán cơ bản trong giải tích. SageMath cung cấp chức năng để tính đạo hàm của các hàm số, bao gồm cả đạo hàm riêng và đạo hàm toàn phần.

Ví dụ tính đạo hàm của hàm số $f(x) = x^2 + 3x + 5$:

```
sage: x = var('x')  # Dinh nghia bien x
sage: f = x^2 + 3*x + 5  # Dinh nghia ham so
sage: diff(f, x)  # Tinh dao ham cua f theo x
2*x + 3
```

SageMath cũng cho phép tính đạo hàm bậc cao:

```
sage: diff(f, x, 2)  # Dao ham bac 2 cua f
2
```

3.2.4.2 Tích phân

Tích phân là phép toán cơ bản trong giải tích, được sử dụng để tính diện tích dưới đường cong, thể tích và nhiều ứng dụng khác. SageMath hỗ trợ tính tích phân bất định và tích phân xác định.

Tính tích phân bất định của hàm $f(x) = x^2 + 3x + 5$:

```
sage: integrate(f, x)  # Tinh tich phan bat dinh cua f theo x
x^3/3 + 3*x^2/2 + 5*x
```

Tính tích phân xác định của hàm $f(x) = x^2$ từ $x = 0$ đến $x = 1$:

```
sage: integrate(x^2, (x, 0, 1))  # Tinh tich phan xac dinh cua x^2 tu 0 den 1
1/3
```

3.2.4.3 Giải phương trình vi phân

SageMath cũng hỗ trợ giải phương trình vi phân (ODEs), bao gồm các phương trình vi phân thường và các phương trình vi phân riêng. Dưới đây là ví dụ về cách giải một phương trình vi phân thông qua phương thức `desolve()`.

Giải phương trình vi phân $y' = -2y$ với điều kiện ban đầu $y(0) = 1$:

```
sage: y = function('y')(x)  # Dinh nghia ham y(x)
sage: desolve(-2*y, y, ics=[0,1])  # Giai phuong trinh vi phan voi dieu kien ban dau
exp(-2*x)
```

3.2.4.4 Tính toán với chuỗi số học

SageMath cũng có thể tính toán với chuỗi số học (series), bao gồm các chuỗi hội tụ và phân kỳ. Ví dụ tính chuỗi Taylor của hàm $\sin(x)$ tại $x = 0$.

```
sage: taylor(sin(x), x, 0, 5)  # Tinh chuoi Taylor bac 5 cua sin(x) tai x=0
x - x^3/6 + 0(x^5)
```

3.2.5 Chuỗi, giới hạn và phép tính xấp xỉ

SageMath cung cấp các công cụ mạnh mẽ để làm việc với chuỗi số học, tính toán giới hạn và thực hiện các phép tính xấp xỉ. Những công cụ này rất hữu ích khi làm việc với các chuỗi hội tụ hoặc phân kỳ, tính toán giới hạn của các hàm số tại một điểm, và áp dụng các phương pháp xấp xỉ trong các bài toán toán học phức tạp.

3.2.5.1 Chuỗi số học

Chuỗi số học trong SageMath được sử dụng để tính toán và phân tích các chuỗi vô hạn. Các chuỗi này có thể hội tụ hoặc phân kỳ, và việc tính toán các giới hạn của chúng là rất quan trọng trong giải tích.

Ví dụ tính chuỗi của hàm số $f(x) = \frac{1}{n^2}$ từ $n = 1$ đến $n = \infty$:

```
sage: summation(1/n^2, n, 1, infinity)  # Tinh tong cua chuoi 1/n^2 tu n=1 den vo cung
pi^2/6
```

SageMath có thể tính chuỗi vô hạn và xác định tổng của chuỗi hội tụ như trong ví dụ trên.

3.2.5.2 Giới hạn của hàm số

Giới hạn là một trong những công cụ cơ bản trong giải tích, giúp xác định hành vi của một hàm số khi biến số tiến gần đến một giá trị nhất định. SageMath hỗ trợ tính giới hạn của các hàm số tại một điểm.

Ví dụ tính giới hạn của hàm số $f(x) = \frac{1}{x}$ khi x tiến đến 0:

```
# Tinh gioi han cua 1/x khi x tien den 0 tu ben phai
sage: limit(1/x, x=0, dir='plus')
+Infinity

# Tinh gioi han cua 1/x khi x tien den 0 tu ben trai
sage: limit(1/x, x=0, dir='minus')
-Infinity
```

SageMath tự động tính toán giới hạn của các hàm số tại các điểm mà hàm có thể hội tụ hoặc phân kỳ.

3.2.5.3 Phép tính xấp xỉ

SageMath cung cấp nhiều phương pháp để tính xấp xỉ các giá trị của các hàm số, đặc biệt hữu ích khi làm việc với các giá trị không thể tính chính xác hoặc khi cần tốc độ tính toán nhanh.

Ví dụ, tính giá trị xấp xỉ của hàm $\sin(x)$ tại $x = 1$ với 10 chữ số thập phân:

```
sage: sin(1).n(digits=10)  # Tinh gia tri xap xi cua sin(1) voi 10 chu so thap phan
0.8414710000
```

SageMath hỗ trợ tính xấp xỉ cho các phép toán phức tạp, giúp tiết kiệm thời gian tính toán khi không cần độ chính xác tuyệt đối.

3.2.5.4 Chuyển đổi giữa chuỗi và số thực

SageMath cho phép chuyển đổi giữa chuỗi số học và các số thực. Điều này rất hữu ích khi làm việc với các chuỗi số học mà bạn muốn tính toán các giá trị thực gần đúng.

Ví dụ, chuyển đổi chuỗi $\sum_{n=1}^{\infty} \frac{1}{n^2}$ thành giá trị thực:

```
sage: summation(1/n^2, n, 1, infinity).n()  # Tinh gia tri gan dung cua chuoi
1.6449340668
```

SageMath cho phép bạn dễ dàng chuyển đổi giữa các biểu thức chính xác và các giá trị số học gần đúng, giúp tiện lợi trong tính toán.

3.2.6 Làm việc với hàm số và biểu đồ (vẽ đồ thị 2D, 3D)

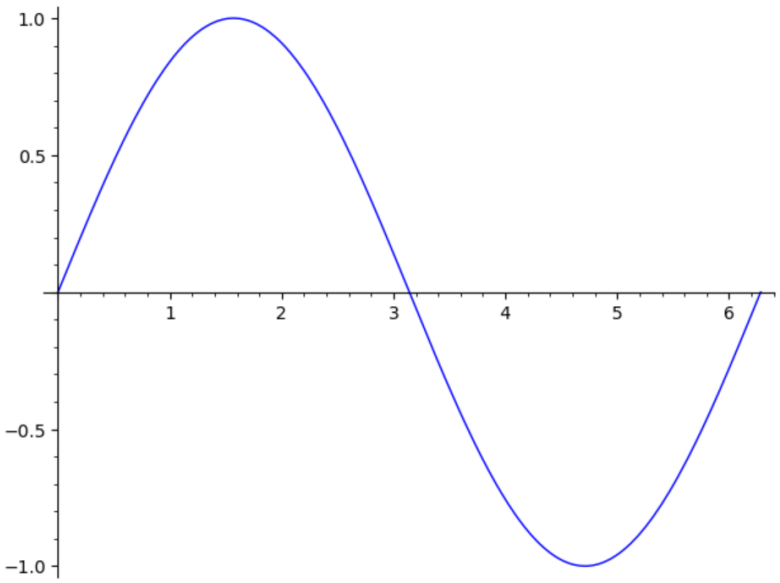
SageMath cung cấp các công cụ mạnh mẽ để làm việc với hàm số và vẽ đồ thị, giúp trực quan hóa các kết quả toán học một cách dễ dàng và hiệu quả. Các đồ thị 2D và 3D có thể được tạo ra cho các hàm số, cung cấp cái nhìn sâu sắc về đặc tính và hành vi của hàm.

3.2.6.1 Vẽ đồ thị 2D của hàm số

Để vẽ đồ thị 2D trong SageMath, bạn có thể sử dụng hàm `plot`, cho phép vẽ đồ thị của một hàm số trên một khoảng xác định.

Ví dụ vẽ đồ thị của hàm $f(x) = \sin(x)$ trên đoạn $[0, 2\pi]$:

```
sage: plot(sin(x), (x, 0, 2*pi))  # Ve do thi cua sin(x) tu 0 den 2*pi
```



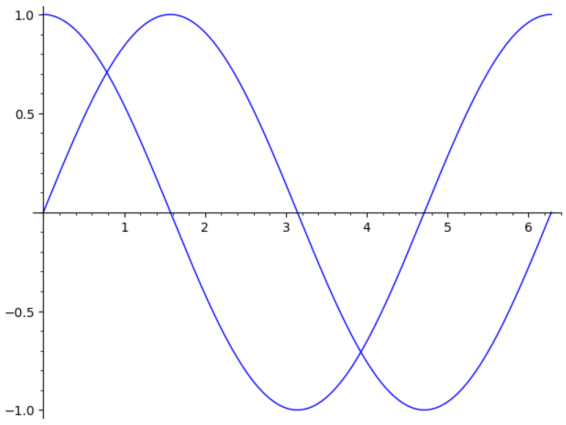
Hình 3.2: Đồ thị 2D của hàm số $\sin(x)$ trên đoạn $[0, 2\pi]$

3.2.6.2 Vẽ đồ thị 2D của nhiều hàm số

SageMath hỗ trợ vẽ đồ thị của nhiều hàm số trên cùng một đồ thị. Bạn có thể vẽ nhiều hàm số trên cùng một biểu đồ bằng cách sử dụng toán tử cộng (+).

Ví dụ, vẽ đồ thị của $f(x) = \sin(x)$ và $g(x) = \cos(x)$ trên cùng một đồ thị:

```
sage: plot(sin(x), (x, 0, 2*pi)) + plot(cos(x), (x, 0, 2*pi
    ))  # Ve dong thoi do thi cua ham sin(x) va cos(x)
```

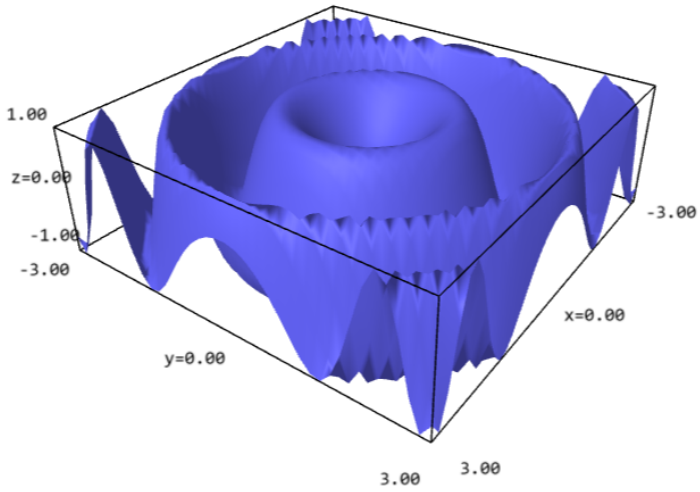


Hình 3.3: Đồ thị với cả hai hàm $\sin(x)$ và $\cos(x)$ trên cùng một biểu đồ

3.2.6.3 Vẽ đồ thị 3D của hàm số

SageMath cũng cho phép vẽ đồ thị 3D của các hàm hai biến. Để vẽ đồ thị 3D, bạn có thể sử dụng hàm `plot3d`. Ví dụ, vẽ đồ thị của hàm $f(x, y) = \sin(x^2 + y^2)$:

```
sage: plot3d(sin(x^2 + y^2), (x, -3, 3), (y, -3, 3))  # Ve
    do thi 3D cua ham sin(x^2 + y^2)
```



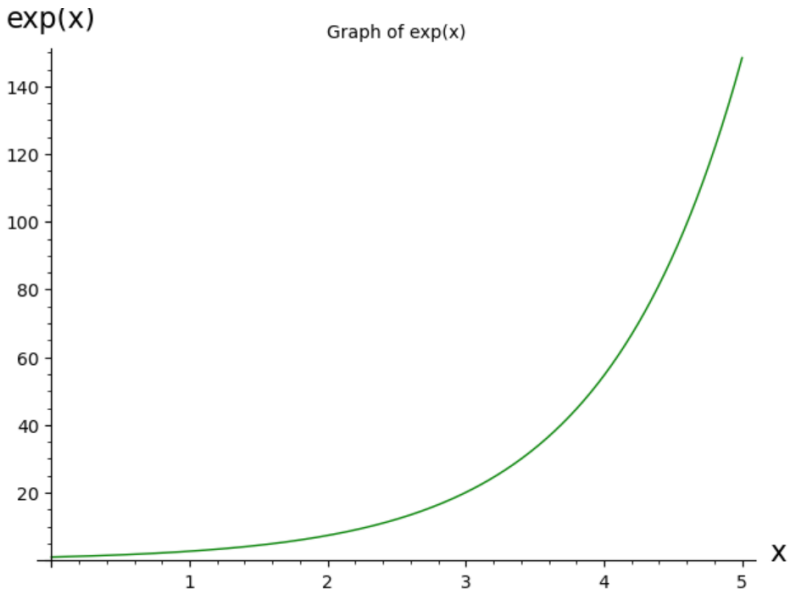
Hình 3.4: Một bề mặt 3D của hàm $\sin(x^2 + y^2)$ trong khoảng $[-3, 3]$ cho cả x và y

3.2.6.4 Tùy chỉnh đồ thị

SageMath cung cấp nhiều tùy chọn để tùy chỉnh đồ thị, chẳng hạn như thay đổi màu sắc, thêm tiêu đề, điều chỉnh trục, hoặc thay đổi độ dày đường nét. Các tùy chọn này có thể giúp bạn tạo ra các biểu đồ trực quan và dễ hiểu hơn.

Ví dụ, vẽ đồ thị của hàm $f(x) = e^x$ và thêm tiêu đề, nhãn trục:

```
# Ve do thi ham so exp(x) voi tieu de va nhan
sage: p = plot(exp(x), (x, 0, 5), color='green')
sage: p.axes_labels(['x', 'exp(x)'])
sage: p.show(title='Graph of exp(x)')
```

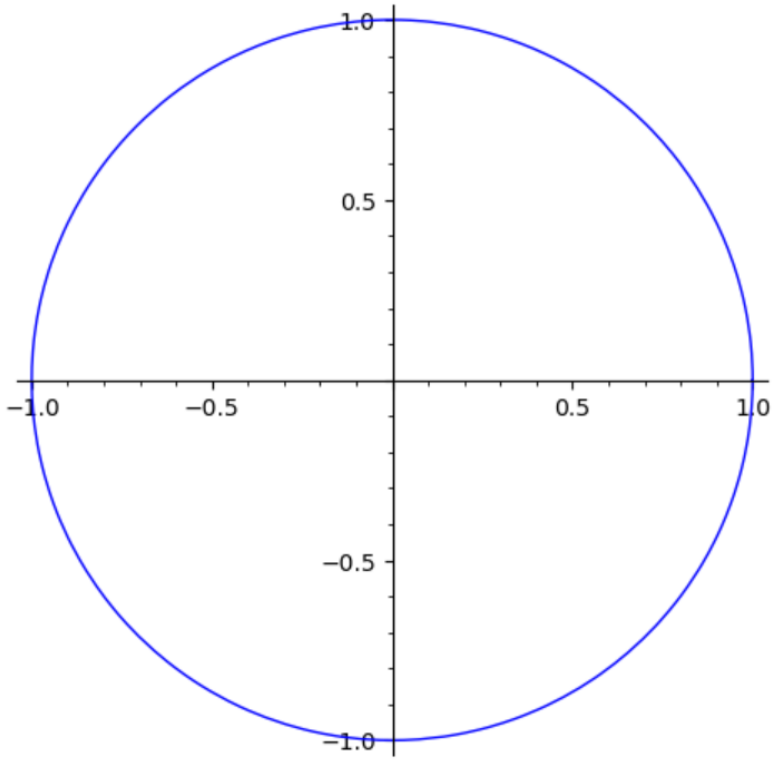


Hình 3.5: Đồ thị của e^x với màu xanh lá, thêm tiêu đề cùng với nhãn cho các trục

3.2.6.5 Vẽ đồ thị parametric

SageMath cũng hỗ trợ vẽ đồ thị các hàm parametric. Các hàm parametric được định nghĩa bởi một cặp hàm số $x(t)$ và $y(t)$, với t là tham số. Ví dụ, vẽ đồ thị của đường tròn với bán kính $r = 1$:

```
sage: var('t')  #Khai bao bien t
sage: parametric_plot([cos(t), sin(t)], (t, 0, 2*pi))  # Ve
    do thi cua duong tron
```



Hình 3.6: Đồ thị của một đường tròn đơn vị trong mặt phẳng Oxy

3.3 Tổ hợp, xác suất và logic

3.3.1 Tính toán tổ hợp, xác suất và thống kê

SageMath cung cấp một bộ công cụ mạnh mẽ để làm việc với các vấn đề tổ hợp, xác suất và thống kê. Phần này sẽ giới thiệu cách sử dụng SageMath để thực hiện các phép toán tổ hợp, tính xác suất và xử lý các bài toán thống kê.

3.3.1.1 Tính toán tổ hợp

Trong tổ hợp, các phép toán như chèn chỗ và kết hợp được sử dụng để tính toán số cách chọn một tập hợp con từ một tập hợp lớn hơn. SageMath cung cấp các hàm như `binomial` và `factorial` để tính các giá trị tổ hợp, chỉnh hợp và giai thừa.

Ví dụ, tính số cách chọn 3 phần tử từ một tập hợp 5 phần tử:

```
1 sage: binomial(5, 3)  # Tinh to hop chap 3 cua 5
2 10
```

Ví dụ, tính số cách chọn và sắp xếp 3 phần tử từ một tập hợp 5 phần tử:

```
1 sage: permutation(5, 3)  # Tinh chinh hop 3 cua 5
2 60
```

Ngoài ra, bạn có thể tính giai thừa của một số bằng cách sử dụng hàm `factorial`:

```
1 sage: factorial(5)  # Tinh 5!
```

```
120
```

3.3.1.2 Tính toán xác suất

SageMath hỗ trợ tính toán xác suất trong các bài toán tổ hợp. Để tính xác suất, bạn có thể sử dụng các hàm như `probability`, `random`, và các hàm xác suất phân phối.

Ví dụ, tính xác suất để một con xúc xắc ra mặt 6:

```
sage: P = 1/6  # Xac suat ra mat 6
P
1/6
```

SageMath cung cấp hàm `probability` để tính toán xác suất trong các bài toán phức tạp hơn, đặc biệt khi làm việc với các phân phối xác suất. Ví dụ, ta có thể tính xác suất của một sự kiện trong một không gian mẫu xác định, chẳng hạn xác suất để lăn xúc xắc và ra mặt 6:

```
sage: P = probability(lambda: random_integer(1, 6) == 6)  # Xac suat ra mat 6
P
1/6
```

SageMath cũng hỗ trợ việc tạo các giá trị ngẫu nhiên từ phân phối xác suất. Ví dụ, để tạo một giá trị ngẫu nhiên từ phân phối đồng đều trên khoảng từ 1 đến 6 (giống như lăn xúc xắc), bạn có thể sử dụng hàm `random_integer`:

```
sage: random_integer(1, 6)  # Sinh so ngau nhien giua 1 va 6
```

Hàm `random_integer(a, b)` sẽ trả về một giá trị ngẫu nhiên trong khoảng từ a đến b, bao gồm cả a và b.

Bạn cũng có thể tính xác suất của các sự kiện phức tạp hơn, chẳng hạn như xác suất ra mặt chẵn khi lăn một con xúc xắc. Sử dụng hàm `probability` với một điều kiện phức tạp:

```
sage: P_even = probability(lambda: random_integer(1, 6) in [2, 4, 6])  # Xac suat ra mat chan
P_even
1/2
```

Ở đây, chúng ta đã tính xác suất ra mặt chẵn của con xúc xắc. Kết quả xác suất là $\frac{3}{6} = \frac{1}{2}$, vì có ba mặt chẵn (2, 4, 6) trong tổng số sáu mặt của xúc xắc.

3.3.1.3 Phân phối xác suất

SageMath hỗ trợ nhiều loại phân phối xác suất, bao gồm phân phối nhị thức, phân phối chuẩn và phân phối Poisson. Bạn có thể sử dụng các hàm như

`binomial_distribution`, `normal_distribution`, và `poisson_distribution` để làm việc với các phân phối này.

Ví dụ, tính xác suất để có ít nhất 3 mặt "được"trong 5 lần gieo xúc xắc với phân phối nhị thức:

```
sage: B = binomial_distribution(5, 1/6)  # Phan phoi nhi thuc cho 5 lan gieo
P = B.cdf(3)  # Cumulative distribution function de tinh xac suat co it nhat 3 lan "duoc"
P
0.868050607
```

3.3.1.4 Tính toán thống kê

SageMath cũng cung cấp các công cụ mạnh mẽ để tính toán thống kê mô tả như trung bình, phương sai, độ lệch chuẩn và các phép toán thống kê khác.

Ví dụ, tính trung bình và độ lệch chuẩn của một dãy số:

```
sage: data = [12, 15, 20, 25, 30, 35, 40]
sage: mean(data)  # Tinh trung binh
25.2857142857143
sage: stdev(data)  # Tinh do lech chuan
9.49460415914107
```

Các hàm `mean` và `stdev` tính toán trung bình và độ lệch chuẩn của dãy số.

3.3.1.5 Kiểm định giả thuyết

SageMath cung cấp các công cụ để thực hiện các kiểm định giả thuyết thống kê. Ví dụ, kiểm định t-Student để kiểm tra sự khác biệt giữa trung bình mẫu và trung bình giả thuyết.

Ví dụ, thực hiện kiểm định t với dãy số và trung bình giả thuyết là 25:

```
sage: t_test(data, 25)  # Kiem dinh t voi trung binh gia thuyet 25
(0.285714285714286, 0.781595168902066)
```

Kết quả trả về là giá trị thống kê t và giá trị p. Với giá trị p này, bạn có thể quyết định có bác bỏ giả thuyết hay không.

3.3.2 Làm việc với số học rời rạc và lý thuyết số

SageMath cung cấp các công cụ rất mạnh mẽ để làm việc với số học rời rạc và lý thuyết số, cho phép thực hiện từ những thao tác đơn giản như tính `mod`, kiểm tra nguyên tố đến những thao tác phức tạp hơn như định lý phần dư Trung Hoa, logarit rời rạc, và hệ thống số học hữu hạn.

3.3.2.1 Số nguyên tố và kiểm tra nguyên tố

Bạn có thể dễ dàng tạo danh sách các số nguyên tố hoặc kiểm tra một số có phải nguyên tố hay không bằng các hàm `is_prime`, `next_prime`, `prime_pi`, và `primes`.

```
sage: is_prime(37)
True
sage: next_prime(100)
101
sage: primes(50, 70)
[53, 59, 61, 67]
```

3.3.2.2 Ước số chung lớn nhất và bội số chung nhỏ nhất

SageMath hỗ trợ tính ước số chung lớn nhất (GCD) và bội số chung nhỏ nhất (LCM) bằng các hàm `gcd` và `lcm`.

```
sage: gcd(48, 180)
12
sage: lcm(12, 18)
36
```

3.3.2.3 Số học modulo

Số học modulo là nền tảng trong nhiều thuật toán lý thuyết số. Bạn có thể sử dụng toán tử `%` hoặc tạo trường số dư với `Zmod(n)`.

```
sage: 17 % 5
2
sage: R = Zmod(7)
sage: a = R(3)
sage: b = R(5)
sage: a * b
R(1)
```

3.3.2.4 Định lý phần dư Trung Hoa

SageMath hỗ trợ định lý phần dư Trung Hoa qua hàm `crt` để tìm số thỏa mãn một hệ phương trình đồng dư.

```
sage: crt([2, 3, 2], [3, 5, 7])  # Chinese Remainder
    Theorem
23
```

3.3.2.5 Hàm Euler và số nguyên tố cùng nhau

Hàm Euler $\phi(n)$ đếm số nguyên nhỏ hơn n nguyên tố cùng nhau với n. SageMath cung cấp hàm `euler_phi` và `is_coprime` để làm việc với tính chất này.

```
sage: euler_phi(10)
4
sage: gcd(7, 10) == 1  # 7 and 10 are coprime
True
```

3.3.2.6 Lũy thừa modulo và logarit rời rạc

Phép toán lũy thừa modulo có thể được thực hiện hiệu quả với hàm `power_mod`, và SageMath cũng hỗ trợ tìm logarit rời rạc trong một số trường hợp.

```
sage: power_mod(3, 4, 5)
1
sage: Z5 = Zmod(5)
sage: Z5(3).log(2)  # Solve 2^x = 3 mod 5
3
```

3.3.2.7 Trường số và phần tử đại số

SageMath cho phép bạn tạo và làm việc với các trường số (number fields), ví dụ trường chứa căn bậc hai hoặc các phần tử đại số khác.

```
sage: K.<a> = NumberField(x^2 - 2)
sage: a^2
2
```

3.3.3 Tính toán với tập hợp, logic và điều kiện

SageMath hỗ trợ nhiều công cụ để thao tác với tập hợp, biểu thức logic và điều kiện. Các cấu trúc này không chỉ hữu ích trong toán học rời rạc mà còn trong việc xây dựng mô hình, tự động hóa kiểm định mệnh đề và viết mã có điều kiện.

3.3.3.1 Tập hợp và các phép toán trên tập hợp

Bạn có thể định nghĩa các tập hợp và thực hiện các phép toán như hợp, giao, hiệu, hiệu đối xứng và kiểm tra phần tử.

```
sage: A = Set([1, 2, 3, 4])
sage: B = Set([3, 4, 5, 6])
sage: A.union(B)
{1, 2, 3, 4, 5, 6}
sage: A.intersubsection(B)
{3, 4}
sage: A.difference(B)
{1, 2}
sage: A.symmetric_difference(B)
{1, 2, 5, 6}
```

```
sage: 3 in A
True
```

Ngoài ra, SageMath còn hỗ trợ tập rỗng, tập vô hạn, tập định nghĩa bằng điều kiện và các tập số học đặc biệt như tập số nguyên, tập số thực, tập hữu tỷ, ...

```
sage: EmptySet()
{}
sage: QQ.is_subset(RR)
True
```

3.3.3.2 Biểu thức logic và phép toán mệnh đề

SageMath cho phép tạo và đánh giá các biểu thức logic sử dụng các phép toán: `and`, `or`, `not`, `xor`, `implies`, `iff`.

```
sage: a, b = var('a␣b')
sage: f = a & ~b
sage: f.simplify_logic()
a & ~b
```

Bạn cũng có thể kiểm tra tính đúng sai hoặc tương đương logic giữa các biểu thức.

```
sage: f1 = (a & b) | (~a & b)
sage: f2 = b
sage: bool(f1.simplify_logic() == f2.simplify_logic())
True
```

3.3.3.3 Câu lệnh điều kiện `if` và các biểu thức điều kiện

Trong khi SageMath chủ yếu dùng cho biểu thức toán học, nó cũng hỗ trợ biểu thức điều kiện trong mã Python:

```
sage: x = 5
sage: if x % 2 == 0:
....:     y = "even"
....: else:
....:     y = "odd"
sage: y
'odd'
```

Bạn có thể dùng biểu thức điều kiện ngắn gọn dạng `a if condition else b`:

```
sage: z = "positive" if x > 0 else "non-positive"
sage: z
'positive'
```

3.3.3.4 Mệnh đề định lượng: forall và exists

Trong SageMath, việc biểu diễn các lượng từ toán học (`forall`, `exists`) không được hỗ trợ trực tiếp thông qua các hàm `ForAll` và `Exists` như trong một số ngôn ngữ lập trình khác. Thay vào đó, chúng ta có thể sử dụng các phương pháp kiểm tra tính đúng của mệnh đề bằng cách sử dụng các hàm kiểm tra giá trị cụ thể hoặc dùng giả thuyết với hàm `assume()`.

Ví dụ: Kiểm tra mệnh đề với tập giá trị hữu hạn

```
# Kiem tra dieu kien x^2 >= 0 voi cac gia tri cu the
sage: x = var('x')
sage: all((x^2 >= 0) for x in [-10, -1, 0, 1, 10])
# Ket qua: True
```

Ví dụ: Kiểm tra với giả thuyết

```
# Kiem tra dieu kien voi gia thuyet x la so thuc
sage: assume(x, 'real')
sage: bool(x^2 >= 0)
# Ket qua: True
```

Chương 4

Các tính năng nâng cao và tùy chỉnh

4.1 Lập trình trong SageMath bằng Python

SageMath được xây dựng dựa trên Python, cho phép người dùng sử dụng đầy đủ các cấu trúc điều khiển, hàm, lớp và thư viện của ngôn ngữ này. Điều này giúp mở rộng khả năng xử lý và tạo ra các chương trình toán học linh hoạt.

4.1.1 Câu lệnh điều kiện và vòng lặp

Người dùng có thể sử dụng các cấu trúc điều kiện như `if-else` và vòng lặp như `for`, `while` trong SageMath.

```
        # Kiem tra so nguyen to bang cau lenh if
        sage: def is_prime_check(n):
        ....:     if is_prime(n):
        ....:         return "La so nguyen to"
        ....:     else:
        ....:         return "Khong la so nguyen to"

        sage: is_prime_check(17)
        'La so nguyen to'

        # Duyet cac so nguyen to nho hon 20
        sage: for i in range(1, 20):
        ....:     if is_prime(i):
        ....:         print(i)
        2
        3
        5
        7
```

```
11
13
17
19
```

4.1.2 Định nghĩa và sử dụng hàm

Người dùng có thể định nghĩa hàm tùy chỉnh trong SageMath tương tự như trong Python.

```
# Ham tinh giai thua bang de quy
sage: def giaithua(n):
....:     if n == 0:
....:         return 1
....:     else:
....:         return n * giaithua(n - 1)

sage: giaithua(5)
120
```

4.1.3 Làm việc với danh sách và tập hợp

Các cấu trúc dữ liệu như danh sách, tập hợp, từ điển... đều có thể sử dụng trong SageMath để xử lý dữ liệu toán học linh hoạt.

```
# Tao danh sach cac binh phuong
sage: squares = [n^2 for n in range(1, 11)]
sage: squares
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

# Tap hop cac uoc so cua 24
sage: divisors(24)
[1, 2, 3, 4, 6, 8, 12, 24]
```

4.1.4 Sử dụng lambda và biểu thức hàm

Biểu thức lambda giúp định nghĩa hàm ngắn gọn và kết hợp với các hàm bậc cao như `map`, `filter`, `reduce`.

```
# Su dung lambda de tinh lap phuong
sage: lap_phuong = lambda x: x^3
sage: lap_phuong(4)
64

```

```
        # Loc cac so chia het cho 3
        sage: list(filter(lambda x: x % 3 == 0, range(1, 20))
           )
        [3, 6, 9, 12, 15, 18]
```

Như vậy, SageMath không chỉ là công cụ toán học mạnh mẽ mà còn là môi trường lập trình tiện lợi, linh hoạt với đầy đủ sức mạnh của Python.

4.2 SageMath trong tính toán khoa học

SageMath không chỉ là công cụ tính toán toán học thuần túy mà còn được sử dụng rộng rãi trong tính toán khoa học (scientific computing), nhờ tích hợp với các thư viện nổi tiếng như NumPy, SciPy và matplotlib.

4.2.1 Sử dụng NumPy để xử lý mảng số liệu

SageMath hỗ trợ thư viện NumPy để thao tác với mảng và thực hiện các phép toán số học hiệu quả.

```
        # Tao mang numpy va tinh tong
        sage: import numpy as np
        sage: a = np.array([1, 2, 3, 4])
        sage: b = np.array([10, 20, 30, 40])
        sage: a + b
        array([11, 22, 33, 44])
```

4.2.2 Sử dụng SciPy để giải bài toán khoa học

SciPy cung cấp nhiều công cụ để giải phương trình, tích phân, tối ưu hóa và các bài toán vật lý kỹ thuật.

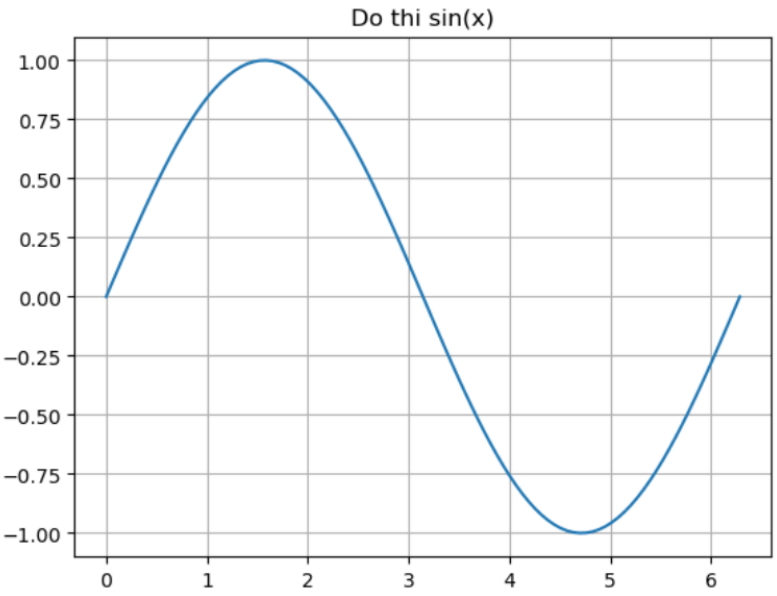
```
        # Tim nghiem gan dung cua phuong trinh
        sage: from scipy.optimize import fsolve
        sage: f = lambda x: x**3 - 1
        sage: fsolve(f, 0.5)
        array([1.])
```

4.2.3 Vẽ đồ thị khoa học với matplotlib

SageMath có thể dùng matplotlib để tạo các biểu đồ trực quan hóa dữ liệu.

```
        # Ve do thi ham sin(x)
        sage: import matplotlib.pyplot as plt
        sage: import numpy as np
```

```
sage: x = np.linspace(0, 2*np.pi, 100)
sage: y = np.sin(x)
sage: plt.plot(x, y)
sage: plt.title('Do thi sin(x)')
sage: plt.grid()
sage: plt.show()
```



Hình 4.1: Vẽ đồ thị $y = sin(x)$ trên đoạn $[0, 2\pi]$ với matplotlib

Nhờ tích hợp tốt với các công cụ khoa học hiện đại, SageMath cho phép người dùng thực hiện các bài toán từ lý thuyết đến ứng dụng kỹ thuật một cách hiệu quả.

4.3 Mở rộng SageMath với thư viện bên ngoài

Một trong những điểm mạnh của SageMath là khả năng mở rộng thông qua các thư viện bên ngoài, đặc biệt là các thư viện của Python. Người dùng có thể dễ dàng cài đặt và sử dụng các gói bổ sung để tăng cường chức năng của SageMath.

4.3.1 Cài đặt thư viện bổ sung

Người dùng có thể cài đặt các thư viện Python bằng lệnh `pip` ngay trong SageMath:

```
# Cai thu vien sympy
sage: !pip install sympy
```

Sau khi cài đặt, ta có thể nhập và sử dụng ngay:

```
# Su dung sympy de rut gon bieu thuc
sage: import sympy as sp
```

```
sage: x = sp.Symbol('x')
sage: expr = (x**2 + 2*x + 1)/(x + 1)
sage: sp.simplify(expr)
x + 1
```

4.3.2 Kết hợp SageMath với pandas để xử lý dữ liệu

Pandas là một thư viện mạnh để thao tác với dữ liệu dạng bảng.

```
# Doc file CSV va thong ke
sage: import pandas as pd
sage: data = pd.read_csv('du_lieu.csv')
sage: data.head()
sage: data.describe()
```

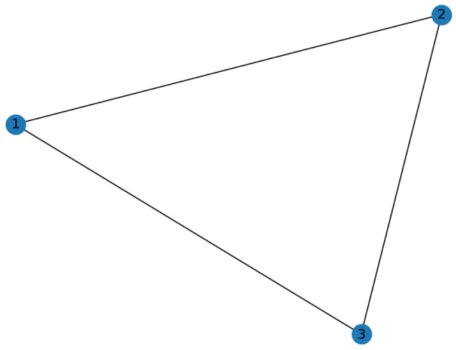
4.3.3 Sử dụng thư viện tính toán nâng cao

Một số thư viện khác có thể dùng để xử lý các bài toán đặc biệt như:

- **networkx** – phân tích đồ thị và mạng.
- **scikit-learn** – học máy cơ bản.
- **cvxpy** – tối ưu hóa lồi.

Ví dụ dùng `networkx` để tạo đồ thị:

```
# Tao do thi bang networkx
sage: import networkx as nx
sage: G = nx.Graph()
sage: G.add_edges_from([(1, 2), (2, 3), (3, 1)])
sage: nx.draw(G, with_labels=True)
```



Hình 4.2: Đồ thị được tạo bằng networkx

Khả năng mở rộng giúp SageMath trở thành nền tảng mở, thân thiện với nghiên cứu và giảng dạy hiện đại.

4.4 Kết nối SageMath với các phần mềm khác

SageMath có thể tích hợp với nhiều phần mềm và hệ thống khác để phục vụ các mục đích đa dạng như trình bày tài liệu, xử lý dữ liệu, và phát triển ứng dụng.

4.4.1 Sử dụng SageMath với LaTeX

SageMath hỗ trợ tích hợp trực tiếp với LaTeX thông qua gói `sagetex`. Gói này cho phép người dùng chèn mã SageMath vào file LaTeX và tự động sinh kết quả.

Ví dụ: file `main.tex` chứa đoạn sau:

```
\documentclass{article}
\usepackage{sagetex}
\begin{document}
        Gia tri cua $f(x) = x^2 + 1$ tai $x = 3$ la \
           sage{3^2 + 1}.
\end{document}
```

Sau khi biên dịch bằng `pdflatex` và chạy mã Sage, kết quả sẽ tự động được chèn vào văn bản.

4.4.2 Kết nối với Jupyter Notebook

SageMath có thể chạy trong môi trường Jupyter Notebook – nơi cho phép người dùng viết mã, hiển thị công thức, hình ảnh và chú thích trong cùng một giao diện tương tác.

```
# Chay Sage trong Jupyter Notebook
$ sage -n jupyter
```

Người dùng có thể dễ dàng thao tác các biểu thức toán học, vẽ đồ thị và phân tích dữ liệu trực tiếp trong các ô (cell).

4.4.3 Tích hợp với Python IDE

Vì SageMath là một lớp bao ngoài của Python, ta có thể dùng SageMath trong các môi trường lập trình như VSCode hoặc PyCharm với một chút thiết lập. Điều này giúp phát triển các ứng dụng lớn và chuyên sâu hơn.

4.4.4 Giao tiếp với phần mềm toán học khác

SageMath còn có thể giao tiếp với các phần mềm khác như:

- **R** – phân tích thống kê.
- **Mathematica, Maple** – chuyển đổi định dạng.

- **Maxima, GAP, Singular** – tích hợp trong nội bộ SageMath.

Ví dụ gọi Maxima trong Sage:

```
# Goi Maxima de rut gon bieu thuc
sage: maxima('(x^2 + 2*x + 1)/(x + 1)').ratsimp()
x + 1
```

Tính linh hoạt trong tích hợp giúp SageMath dễ dàng được đưa vào hệ sinh thái học thuật, nghiên cứu và công nghiệp.

Chương 5

Ứng dụng trong các học phần Toán đại cương

5.1 Ứng dụng trong Giải tích 1

5.1.1 Phép tính vi phân hàm một biến số

5.1.1.1 Tìm tập xác định của hàm số

Xét hàm số $y = \sqrt{2\ln x - 3}$. Điều kiện xác định là biểu thức trong hàm ln phải dương và biểu thức trong căn phải không âm:

$$\begin{cases} x > 0 \\ 2\ln x - 3 \geq 0 \Rightarrow \ln x \geq \dfrac{3}{2} \Rightarrow x \geq e^{\frac{3}{2}} \end{cases}$$

Giải điều kiện trong SageMath:

```
sage: var('x')
sage: solve(log(x) >= 3/2, x)
[x >= e^(3/2)]
```

Vậy tập xác định của hàm là $D = \left[e^{\frac{3}{2}}; +\infty\right)$.

Lưu ý: Hàm `solve()` trong SageMath có thể trả về kết quả dưới dạng các biểu thức logic hoặc bất phương trình, và bạn có thể áp dụng các hàm như `find_root()` hoặc `plot()` để kiểm tra trực quan kết quả.

5.1.1.2 Tìm miền giá trị của hàm số

Xét hàm số $y = \sqrt{1 - \left(\frac{x-1}{x+1}\right)^2}$. Tìm miền giá trị của hàm số trên tập (0;2).

```
sage: var('x')
sage: f(x) = sqrt(1 - ((x - 1)/(x + 1))^2)
sage: f.find_local_maximum(0, 2)
(1.00000000000000, 0.999999967094019)
```

```
sage: f.find_local_minimum(0, 2)
(0.0001869787205816002, 8.740260678362169e-09)
```

Kết quả cho biết trên tập (0;2), tập giá trị của $f(x)$ là $(8.740260678362169e-09, 0.999999967094019)$.

5.1.1.3 Làm việc với hàm hyperbolic

Ví dụ hàm $f(x) = \sinh(x) + \cosh(x)$:

```
sage: f(x) = sinh(x) + cosh(x)
sage: f(1)
cosh(1) + sinh(1)
sage: float(f(1))
2.718281828459045
```

5.1.1.4 Tìm hàm $f(x)$

Tìm hàm $f(x)$, biết $f(x + \frac{1}{x}) = x^2 + \frac{1}{x^2}$.

Gợi ý biến đổi biểu thức: Đặt $t = x + \frac{1}{x} \Rightarrow f(t) = t^2 - 2$.

```
sage: var('x t')
sage: t = x + 1/x
sage: expr = x^2 + 1/x^2
sage: f_t = simplify((t)^2 - 2)
sage: f_t
(x + 1/x)^2 - 2
```

5.1.1.5 Tìm hàm ngược

Cho hàm $y = \frac{1}{2}(e^x - e^{-x}) = \sinh(x)$. Hàm ngược của hàm số này là $x = \text{arcsinh}(y)$, hay còn gọi là hàm sinh ngược.

Giải phương trình để tìm hàm ngược trong SageMath:

```
sage: var('x y')
sage: solve(y == (1/2)*(exp(x) - exp(-x)), x)
[x == log(y - sqrt(y^2 + 1)), x == log(y + sqrt(y^2 + 1))]
```

Kết quả SageMath trả về là:

$$x = \ln\left(y + \sqrt{y^2 + 1}\right)$$

Do đó, hàm ngược là:

$$\boxed{f^{-1}(y) = \ln\left(y + \sqrt{y^2 + 1}\right)}$$

5.1.1.6 Tính giới hạn, kiểm tra tính liên tục

Xét hàm $f(x) = \frac{x}{4-x^2}$ trên khoảng $[-1, 1]$.

Giới hạn tại biên:

```
sage: limit(x/(4 - x^2), x=1)
1/3
sage: limit(x/(4 - x^2), x=-1)
-1/3
```

5.1.1.7 Tính đạo hàm, cực trị, tiệm cận

Cho hàm $y = x - \ln(1 + x)$:

```
   sage: f(x) = x - log(1 + x)
sage: diff(f, x)               # Dao ham cap 1
x |--> -1/(x + 1) + 1
sage: diff(f, x, 2)            # Dao ham cap 2
x |--> (x + 1)^(-2)
sage: solve(diff(f, x) == 0, x)  # Tim cuc tri
[x == 0]
sage: limit(f, x=-oo)   # Tiem can ngang (Gioi han trai)
x |--> -Infinity
sage: limit(f, x=+oo)   # Tiem can ngang (Gioi han phai)
x |--> +Infinity
sage: limit(f, x=-1)           # Tiem can dung
x |--> Infinity
```

5.1.2 Phép tính tích phân hàm một biến số

5.1.2.1 Tính tích phân bất định

```
sage: integrate(tan(x)^3, x)
-1/2/(sin(x)^2 - 1) + 1/2*log(sin(x)^2 - 1)
```

5.1.2.2 Tính tích phân xác định

```
sage: integrate(exp(-x^2), (x, -1, 1))
sqrt(pi)*erf(1)
```

5.1.2.3 Tính tích phân suy rộng

```
sage: integrate(1/x^2, (x, 1, oo))
1
```

5.1.2.4 Tính diện tích hình phẳng

Diện tích giữa đồ thị $y = \sin x$ và trục hoành từ 0 đến π:

```
sage: integrate(abs(sin(x)), (x, 0, pi))
2
```

5.1.2.5 Thể tích vật thể quay quanh trục hoành

Cho $y = \sqrt{x}$, thể tích khi quay quanh trục Ox từ 0 đến 1:

```
sage: integrate(pi * (sqrt(x))^2, (x, 0, 1))
1/2*pi
```

5.1.3 Hàm số nhiều biến số

5.1.3.1 Tính giới hạn hàm nhiều biến

SageMath không tự động tính giới hạn nhiều biến tại điểm như Mathematica hay Maple.

5.1.3.2 Đạo hàm riêng và vi phân toàn phần

Xét hàm $f(x, y) = \frac{x^2 y}{x^2+y^2}$:

```
sage: var('x y')
sage: f(x, y) = (x^2 * y)/(x^2 + y^2)
sage: diff(f, x)          # Dao ham theo bien x
(x, y) |--> -2*x^3*y/(x^2 + y^2)^2 + 2*x*y/(x^2 + y^2)
sage: diff(f, y)          # Dao ham theo bien y
(x, y) |--> -2*x^2*y^2/(x^2 + y^2)^2 + x^2/(x^2 + y^2)
```

Kết quả là:

$$f'_x(x, y) = \frac{-2x^3 y}{x^2 + y^2} + \frac{2xy}{x^2 + y^2}$$

$$f'_y(x, y) = \frac{-2x^2 y^2}{x^2 + y^2} + \frac{x^2}{x^2 + y^2}$$

5.1.3.3 Khai triển chuỗi Taylor-Maclaurin hàm nhiều biến

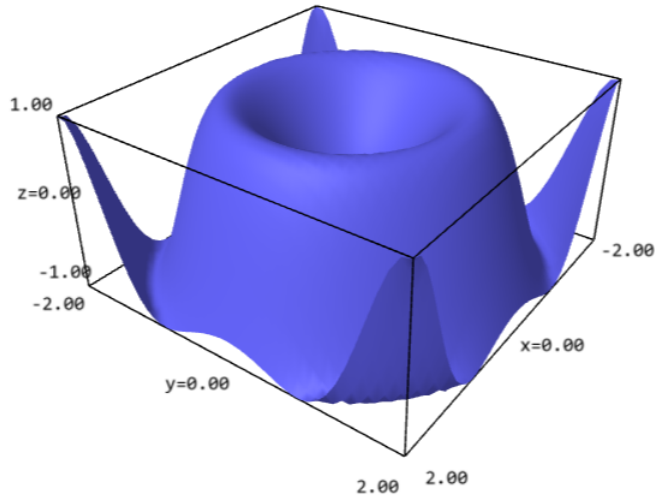
Xét hàm $f(x, y) = e^{x+y}$:

```
sage: var('x y')
sage: f(x, y) = exp(x + y)
sage: taylor(f, (x, 0), 3)
(x, y) |--> 1/6*x^3*e^y + 1/2*x^2*e^y + x*e^y + e^y
```

Kết quả là: $\frac{1}{6}x^3 e^y + \frac{1}{2}x^2 e^y + xe^y + e^y$.

5.1.3.4 Vẽ đồ thị hàm số nhiều biến

```
sage: var('x y')
sage: plot3d(sin(x^2 + y^2), (x, -2, 2), (y, -2, 2))
```



Hình 5.1: Đồ thị hàm $f(x, y) = sin(x^2 + y^2)$ trên $[-2, 2] * [-2, 2]$

5.2 Ứng dụng trong Giải tích 2

5.2.1 Tích phân bội

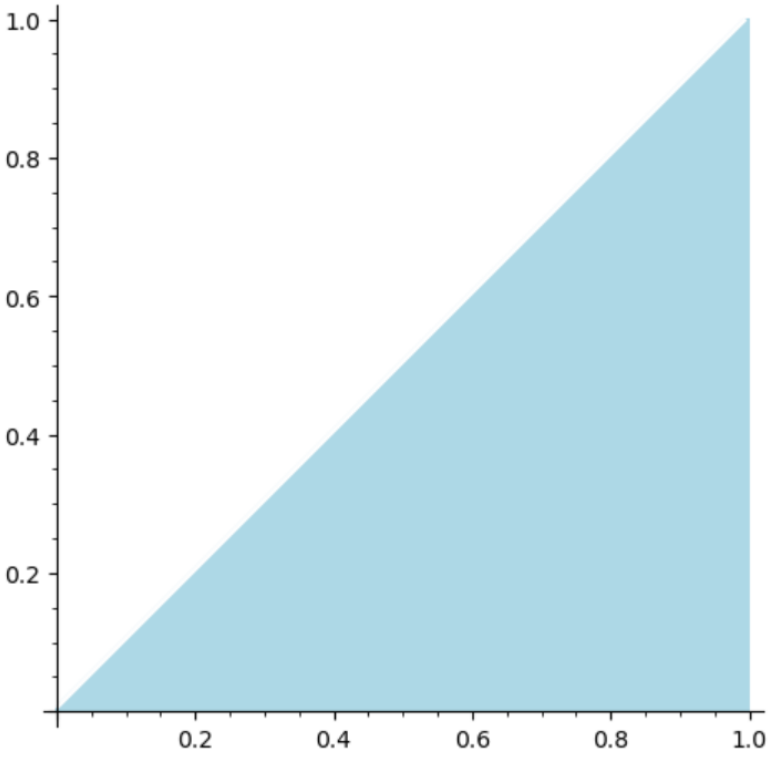
5.2.1.1 Tính tích phân kép

Xét tích phân kép:

$$I = \iint_D (x + y)\, dx\, dy \quad \text{với } D : 0 \le x \le 1,\ 0 \le y \le x$$

Trong SageMath:

```
sage: var('x y')
sage: integrate(integrate(x + y, y, 0, x), x, 0, 1)
1/2              #Ket qua
sage: # Ve mien D
sage: region = region_plot(lambda x, y: y <= x and y >= 0
    and x >= 0 and x <= 1, (x, 0, 1), (y, 0, 1), incol='
    lightblue')
sage: show(region)
```



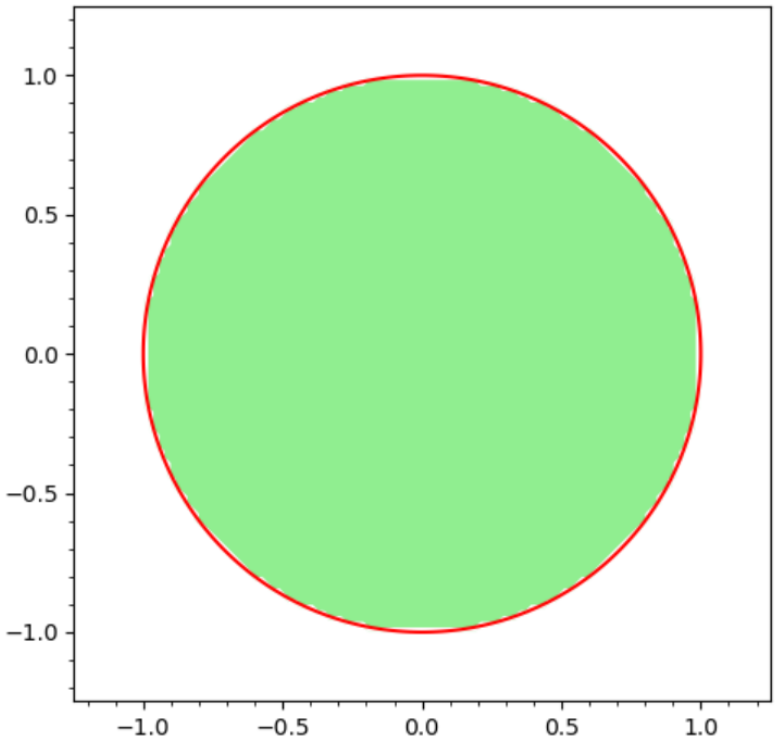
Hình 5.2: Miền $D : 0 \leq x \leq 1,\ 0 \leq y \leq x$

5.2.1.2 Toạ độ cực và tích phân trong toạ độ cực

Xét tích phân:

$I = \iint\limits_D (x^2+y^2)\,dx\,dy$ với D là hình tròn tâm O bán kính 1 $\Rightarrow x = r\cos\theta,\ y = r\sin\theta$

```
sage: var('r theta')
sage: assume(r >= 0)
sage: integrand = (r^2) * r  # x^2 + y^2 = r^2 va Jacobian la r
sage: integrate(integrate(integrand, r, 0, 1), theta, 0, 2*pi)
1/2*pi          # Ket qua
sage: # Ve mien D
sage: circle = implicit_plot(x^2 + y^2 == 1, (x, -1.2, 1.2), (y, -1.2, 1.2), color='red')
sage: region = region_plot(lambda x, y: x^2 + y^2 <= 1, (x, -1, 1), (y, -1, 1), incol='lightgreen')
show(region + circle)
```

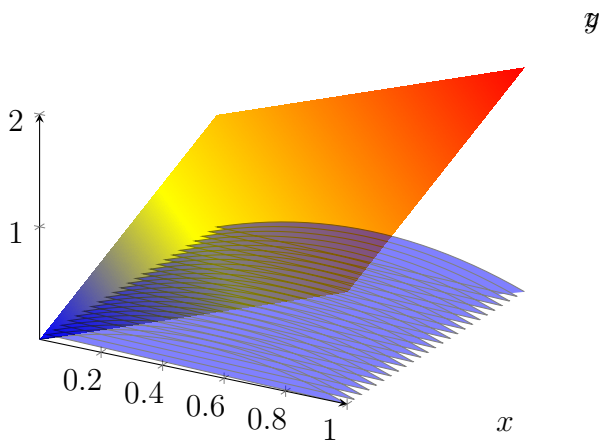


Hình 5.3: Miền D được vẽ bằng phương pháp tọa độ cực

5.2.1.3 Tích phân bội ba

$$I = \iiint_E z\,dV \quad \text{với } E = \{(x,y,z) \mid 0 \le x \le 1, 0 \le y \le 1-x, 0 \le z \le x+y\}$$

```
sage: var('x y z')
sage: integrate(integrate(integrate(z, z, 0, x + y), y, 0,
    1 - x), x, 0, 1)
1/8                 # Ket qua
```



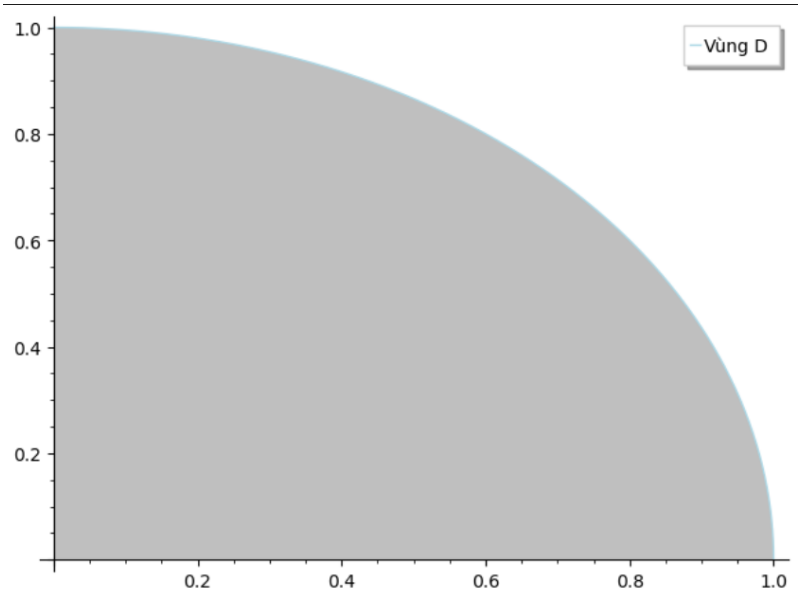
Hình 5.4: Mô phỏng miền tính tích phân bội ba

5.2.2 Ứng dụng tích phân bội: diện tích, thể tích

5.2.2.1 Tính diện tích mặt phẳng

$$A = \iint_D 1\,dx\,dy \quad \text{với } D: 0 \le x \le 1,\ 0 \le y \le \sqrt{1-x^2}$$

```
sage: integrate(integrate(1, y, 0, sqrt(1 - x^2)), x, 0, 1)
1/4*pi          # Ket qua
sage: # Ve mien D
sage: f(x) = sqrt(1 - x^2)
sage: p1 = plot(f(x), (x, 0, 1), fill=True, color='
    lightblue', legend_label="Vung D")
sage: p1.show()
```

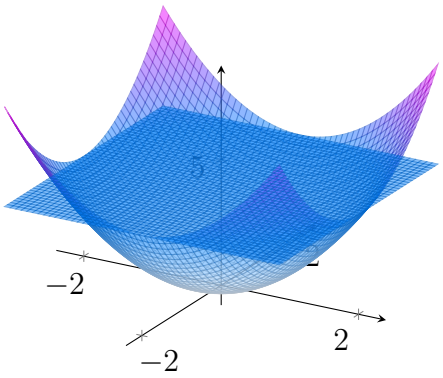


Hình 5.5: Miền cần tính diện tích $D: 0 \le x \le 1,\ 0 \le y \le \sqrt{1-x^2}$

5.2.2.2 Thể tích vật thể

Thể tích vật thể giới hạn bởi mặt phẳng $z = x^2 + y^2$ phía dưới và mặt phẳng $z = 4$ phía trên:

```
sage: var('r theta')
sage: assume(r >= 0)
sage: integrand = (4 - r^2) * r  # The tich = \int (z tren
    - z duoi)*Jacobian
sage: integrate(integrate(integrand, r, 0, 2), theta, 0, 2*
    pi)
8*pi            # Ket qua
```



Hình 5.6: Mô phỏng miền vật thể giới hạn bởi mặt phẳng $z = x^2 + y^2$ phía dưới và mặt phẳng $z = 4$ phía trên.

5.2.2.3 Tích phân phụ thuộc tham số

$$I(a) = \int_0^1 \frac{x^a - 1}{\ln x}\, dx \quad (a > 0)$$

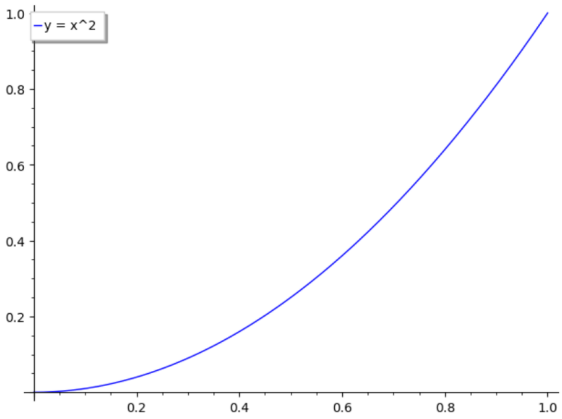
```
sage: var('a x')
sage: assume(a > 0)
sage: integral = integrate((x^a - 1)/log(x), x, 0, 1)
sage: integral
I*pi + log(a + 1)
```

5.2.3 Tích phân đường

5.2.3.1 Tích phân đường loại 1

Tính $\int_C f(x,y)ds$, với $C : y = x^2, 0 \le x \le 1$

```
sage: f(x, y) = x + y
sage: dx = 1
sage: dy = diff(x^2, x)
sage: ds = sqrt(dx^2 + dy^2)
sage: I = integrate(f(x, x^2)*ds, x, 0, 1)
sage: I
67/96*sqrt(5) - 1/64*arcsinh(2) - 1/12          #Ket qua
sage: # Ve mien D
sage: var('x y')
sage: f(x) = x^2
sage: plot1 = plot(f(x), (x, 0, 1), color='blue', sage:
    legend_label='y = x^2')
sage: f_xy(x, y) = x + y
sage: plot1.show()
```

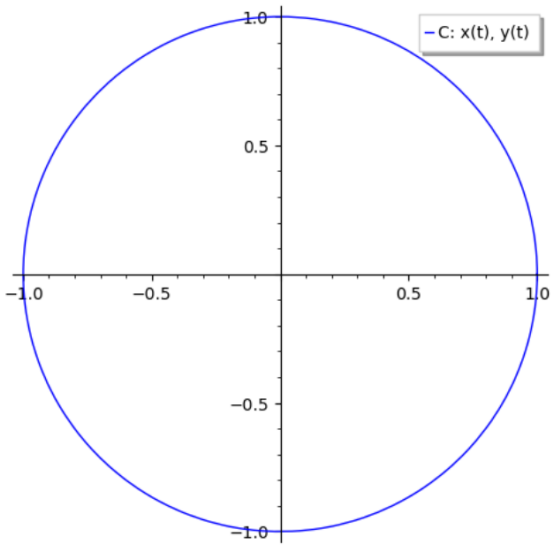


Hình 5.7: Miền $C : y = x^2, 0 \leq x \leq 1$

5.2.3.2 Tích phân đường loại 2

Tính $\int_C (x\,dx + y\,dy)$, với C là đường tròn đơn vị.

```
sage: x(t) = cos(t)
sage: y(t) = sin(t)
sage: dx = diff(x(t), t)
sage: dy = diff(y(t), t)
sage: integrand = x(t)*dx + y(t)*dy
sage: integrate(integrand, t, 0, 2*pi)
0                # Ket qua
sage: # Ve mien D
sage: var('t')
sage: x(t) = cos(t)
sage: y(t) = sin(t)
sage: parametric_plot((x(t), y(t)), (t, 0, 2*pi), color='
   blue', legend_label='C: x(t), y(t)')
```

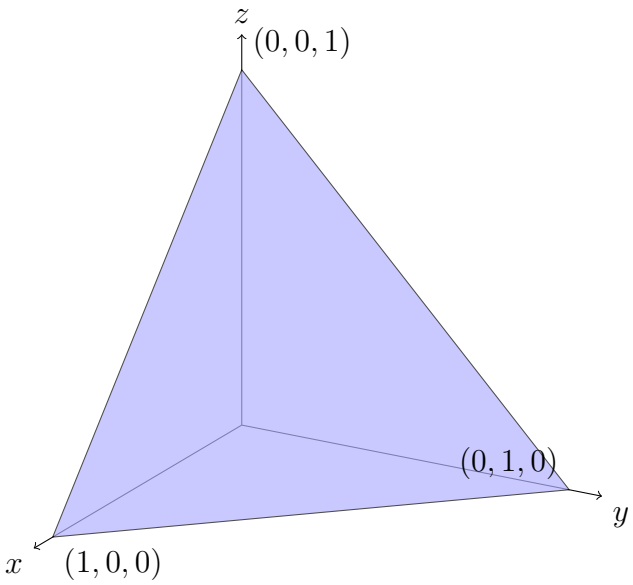


Hình 5.8: Miền C là đường tròn đơn vị

5.2.4 Tích phân mặt

Tính tích phân mặt của hàm $f(x, y, z) = z$ trên mặt phẳng $z = 1 - x - y, 0 \leq x, y \leq 1, x + y \leq 1$

```
sage: z(x, y) = 1 - x - y
sage: f(x, y) = z(x, y)
sage: I = integrate(integrate(f(x, y), y, 0, 1 - x), x, 0,
    1)
sage: I
1/6              # Ket qua
```



Hình 5.9: Mặt phẳng $z = 1 - x - y$ trên miền tam giác $x + y \leq 1$

5.2.5 Lý thuyết trường

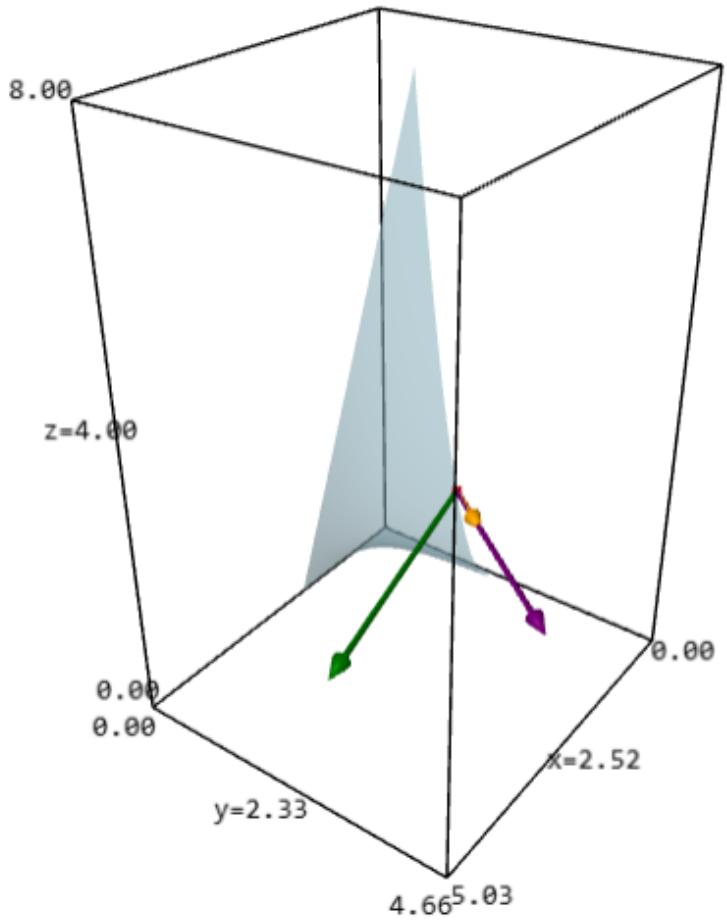
5.2.5.1 Đạo hàm theo hướng

Tính đạo hàm theo hướng của $f(x, y) = x^2 y$ tại $A(1, 2)$ theo hướng $\vec{v} = (3, 4)$

```
sage: f(x, y) = x^2*y
sage: u = vector([3, 4]).normalized()
sage: grad_f = vector([diff(f(x, y), x), diff(f(x, y), y)])
sage: point = vector([1, 2])
sage: grad_at_A = grad_f.subs(x=point[0], y=point[1])
sage: directional_derivative = grad_at_A.dot_product(u)
sage: directional_derivative
16/5    # Ket qua

sage: # Ve mien D
sage: var('x y')
sage: # Ham so
sage: f(x, y) = x^2 * y
sage: # Diem A
```

```
sage: A = vector([1, 2])
sage: # Vector huong
sage: v = vector([3, 4])
sage: u = v.normalized()
sage: # Gradient
sage: grad_f = vector([diff(f(x, y), x), diff(f(x, y), y)])
sage: grad_A = grad_f.subs(x=A[0], y=A[1])
sage: # Vector dao ham theo huong
sage: D_u_f = grad_A.dot_product(u)
sage: vec_D = D_u_f * u
sage: # Ve mat ham
sage: surface = plot3d(f(x, y), (x, 0, 2), (y, 0, sage: 2),
    opacity=0.6, color='lightblue')
sage: # Ve diem A tren mat
sage: pointA = point3d((A[0], A[1], f(*A)), color='red',
   size=30)
sage: # Ve gradient tai A
sage: grad_arrow = arrow3d((A[0], A[1], f(*A)), (A[0] +
   grad_A[0], A[1] + grad_A[1], f(*A)), color='green',
   width=2)
sage: # Ve vector huong tai A
sage: u_arrow = arrow3d((A[0], A[1], f(*A)), (A[0] + u[0],
   A[1] + u[1], f(*A)), color='orange', width=2)
sage: # Ve vector dao ham theo huong (chi phan vector tren
   mat, khong tang do cao)
sage: vecD_arrow = arrow3d((A[0], A[1], f(*A)), (A[0] +
   vec_D[0], A[1] + vec_D[1], f(*A)), color='purple', width
   =2)
sage: # Gop cac thanh phan
sage: show(surface + pointA + grad_arrow + u_arrow +
   vecD_arrow, frame=True)
```



Hình 5.10: Đồ thị mô phỏng đạo hàm theo hướng của $f(x, y) = x^2y$ tại $A(1, 2)$ theo hướng $\vec{v} = (3, 4)$

5.2.5.2 Tính gradient

```
sage: var('x y')
sage: f(x, y) = x^2 * y
sage: # Tinh gradient thu cong
sage: grad_f = vector([diff(f(x, y), x), diff(f(x, y), y)])
sage: grad_f
(2*x*y, x^2)
```

5.3 Ứng dụng trong Giải tích 3

5.3.1 Chuỗi số, xét sự hội tụ của chuỗi số

Xét sự hội tụ của chuỗi: $\sum_{n=1}^{\infty} \frac{1}{n^p}$ với $p > 1$.

```
sage: var('n')
sage: f = 1/n^100
sage: sum(f, n, 1, oo)
sage: var('n')
sage: f = 1/n^100
sage: sum(f, n, 1, oo)
189196075638244.../981520542075751...*pi^100    # Ket qua +
   oo
```

5.3.2 Chuỗi hàm, tính tổng chuỗi, chuỗi hội tụ đều

Tính tổng chuỗi hàm $\sum_{n=0}^{\infty} \frac{x^n}{n!}$

```
sage: var('x n')
sage: f = x^n/factorial(n)
sage: S = sum(f, n, 0, oo)
sage: S
e^x                          # Ket qua
```

5.3.3 Chuỗi lũy thừa, khai triển Taylor, Maclaurin

Khai triển chuỗi Maclaurin của $\sin x$.

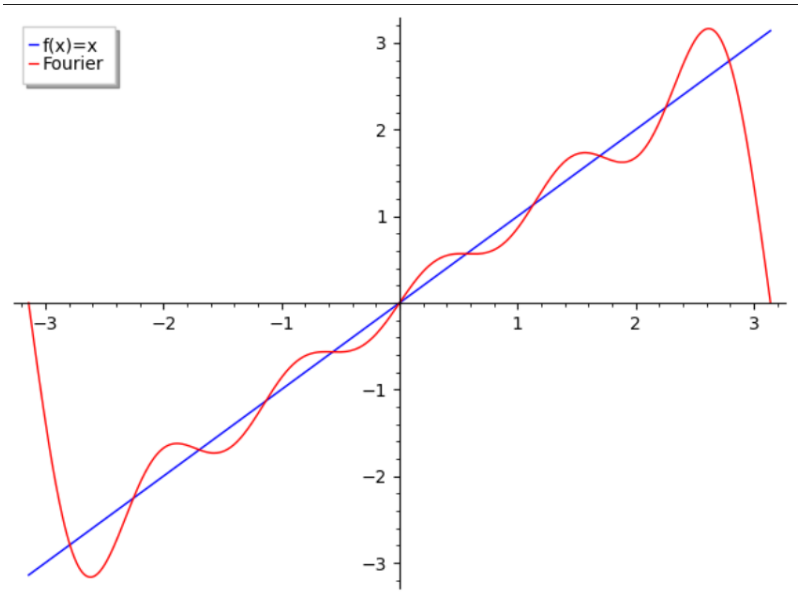
```
sage: taylor(sin(x), x, 0, 7)
-1/5040*x^7 + 1/120*x^5 - 1/6*x^3 + x          # Ket qua
```

5.3.4 Chuỗi Fourier, khai triển Fourier

Khai triển chuỗi Fourier của $f(x) = x$ trên $[-\pi, \pi]$.

```
sage: var('x n')
sage: f(x) = x
sage: a0 = (1/pi)*integrate(f(x), x, -pi, pi)
sage: a0
0
sage: an = (1/pi)*integrate(f(x)*cos(n*x), x, -pi, pi)
sage: an
0
sage: bn = (1/pi)*integrate(f(x)*sin(n*x), x, -pi, pi)
sage: bn
-2*(pi*n*cos(pi*n) - sin(pi*n))/(pi*n^2)
```

```
12
13          # Ve do thi khai trien Fourier bac 5
14          sage: fourier = sum(2*(-1)^(k+1)/k*sin(k*x) for k in (1..5)
               )
15          sage: p1 = plot(f(x), (x, -pi, pi), color='blue',
               legend_label='f(x)=x')
16          sage: p2 = plot(fourier, (x, -pi, pi), color='red',
               legend_label='Fourier')
17          sage: show(p1 + p2)
```



Hình 5.11: Đồ thị minh họa khai triển chuỗi Fourier của $f(x) = x$ trên $[-\pi, \pi]$

5.3.5 Giải phương trình vi phân tuyến tính cấp 1, cấp 2

Giải phương trình:

$$y' + y = e^x$$

```
1           sage: var('x y')
2           sage: y = function('y')(x)
3           sage: de = diff(y, x) + y == e^x
4           sage: desolve(de, y)
5           1/2*(2*_C + e^(2*x))*e^(-x)              # Ket qua
```

Giải phương trình:

$$y'' - y = 0$$

```
1           sage: y = function('y')(x)
2           sage: de2 = diff(y, x, 2) - y == 0
3           sage: desolve(de2, y)
4           _K2*e^(-x) + _K1*e^x              # Ket qua
```

5.3.6 Giải phương trình Euler

Giải phương trình vi phân Euler:

$$x^2y'' - xy' + y = 0$$

```
sage: y = function('y')(x)
sage: de = x^2*diff(y, x, 2) - x*diff(y, x) + y == 0
sage: desolve(de, y)
(_K2*log(x) + _K1)*x              # Ket qua
```

5.3.7 Giải hệ phương trình vi phân cấp 1

Giải hệ:

$$\begin{cases} x' = x + y \\ y' = x - y \end{cases}$$

```
sage: t = var('t')
sage: x = function('x')(t)
sage: y = function('y')(t)
sage: de1 = diff(x, t) == x + y
sage: de2 = diff(y, t) == x - y
sage: desolve_system([de1, de2], [x, y])
[x(t) == 1/2*sqrt(2)*(x(0) + y(0))*sinh(sqrt(2)*t) + cosh(
   sqrt(2)*t)*x(0),
y(t) == 1/2*sqrt(2)*(x(0) - y(0))*sinh(sqrt(2)*t) + cosh(
   sqrt(2)*t)*y(0)]
```

5.3.8 Biến đổi Laplace, ứng dụng giải phương trình vi phân

Giải phương trình bằng biến đổi Laplace:

$$y'' + y = \sin(t), \quad y(0) = 0,\ y'(0) = 1$$

```
sage: var('t␣s')
sage: y = function('y')(t)
sage: de = diff(y, t, 2) + y == sin(t)
sage: desolve_laplace(de, y, ivar=t)
-1/2*t*cos(t) + 1/2*(2*D[0](y)(0) + 1)*sin(t) + cos(t)*y(0)
                  # Ket qua
```

5.4 Ứng dụng trong Đại số tuyến tính

5.4.1 Các phép toán Logic, tập hợp

Các phép toán logic và phép toán trên tập hợp có thể được thực hiện bằng các công cụ trong SageMath. Ví dụ, để tính toán hợp, giao, hiệu của các tập hợp, hoặc thực hiện các phép toán logic như AND, OR, NOT, ta có thể sử dụng cú pháp sau.

```
sage: # Hop va giao cua tap hop
sage: A = Set([1, 2, 3])
sage: B = Set([3, 4, 5])
sage: A.union(B)  # Hop
{1, 2, 3, 4, 5}
sage: A.intersection(B)  # Giao
{3}
```

5.4.2 Ánh xạ, tìm ảnh và nghịch ảnh

Để tính ảnh và nghịch ảnh trong ánh xạ, ta có thể áp dụng trực tiếp công thức của ánh xạ. Ví dụ dưới đây minh họa cách tìm ảnh và nghịch ảnh của một số trong ánh xạ $f(x) = x^2$.

```
sage: var('x')
sage: f = x^2
sage: f(x=3)  # Tinh anh cua 3
9
sage: f.inverse_function()(9)  # Tim nghich anh cua 9
```

5.4.3 Cấu trúc đại số nhóm, vành, trường, nhóm Abel

Các phép toán trong các cấu trúc đại số như nhóm, vành và trường có thể thực hiện thông qua các phép toán trong SageMath. Các phép toán nhóm Abel có thể sử dụng phép toán nhóm trong Sage.

```
# Phep toan nhom Abel
G = Group([(1, 1), (2, 2)])
G * G  # Nhan nhom
```

5.4.4 Số phức, tính module, phép toán số phức, thu gọn về dạng chính tắc

Các phép toán với số phức như cộng, trừ, nhân, chia, tính module và thu gọn về dạng chính tắc có thể thực hiện trong SageMath. Dưới đây là cách tính module và thu gọn số phức về dạng chính tắc.

```
sage: z = 3 + 4*I  # So phuc
sage: abs(z)  # Tinh module
5
sage: conjugate(z)  # Tinh phan ao
11/3
```

5.4.5 Tìm căn bậc n của số phức, giải phương trình nghiệm phức

Để tìm căn bậc n của số phức hoặc giải phương trình nghiệm phức, ta sử dụng các hàm `nth_root` và `solve`.

```
sage: z = 1 + I
sage: nth_root(z, 3)  # Tim can bac 3 cua z
sage: solve(x^2 + 1 == 0, x)  # Giai phuong trinh x^2 + 1 =
    0
```

5.4.6 Ma trận và các phép toán ma trận, định thức, ma trận nghịch đảo, ma trận mũ, hạng của ma trận, hệ phương trình tuyến tính (Phương pháp Gauss)

Các phép toán ma trận như tính định thức, nghịch đảo, ma trận mũ và hạng có thể thực hiện dễ dàng trong SageMath. Hệ phương trình tuyến tính có thể giải bằng phương pháp Gauss.

```
sage: A = Matrix([[1, 2], [3, 4]])
sage: det(A)  # Tinh dinh thuc
sage: A.inv()  # Tinh ma tran nghich dao
sage: A^2  # Ma tran mu
sage: A.rank()  # Hang cua ma tran
sage: solve(A*x == [1, 2], x)  # Giai he phuong trinh tuyen
    tinh
```

5.4.7 Không gian vector, kiểm tra hệ độc lập tuyến tính

Để kiểm tra tính độc lập tuyến tính của các vector, ta có thể tính hạng của ma trận tạo thành từ các vector đó.

```
sage: v1 = vector([1, 2])
sage: v2 = vector([2, 4])
sage: matrix([v1, v2]).rank()  # Kiem tra tinh doc lap
   tuyen tinh
```

5.4.8 Tìm cơ sở và số chiều sinh bởi hệ vector

Cơ sở của không gian vector có thể tìm được thông qua phép biến đổi ma trận, ví dụ dưới đây minh họa cách tìm cơ sở và số chiều sinh bởi hệ vector.

```
sage: v1 = vector([1, 0])
sage: v2 = vector([0, 1])
sage: matrix([v1, v2]).column_space()  # Tim co so
```

5.4.9 Ánh xạ tuyến tính, giá trị riêng và vector riêng

Các phép toán như tìm giá trị riêng và vector riêng của một ma trận có thể được thực hiện qua hàm ‘eigenvalues‘ và ‘eigenvectors‘.

```
sage: A = Matrix([[1, 2], [3, 4]])
sage: A.eigenvalues()  # Tinh gia tri rieng
sage: A.eigenvectors()  # Tinh vector rieng
```

5.4.10 Chéo hóa ma trận

Chéo hóa ma trận có thể thực hiện bằng cách sử dụng hàm ‘diagonalize‘, nếu ma trận có đủ số lượng vector riêng độc lập.

```
sage: A = Matrix([[4, 1], [2, 3]])
sage: A.diagonalize()  # Cheo hoa ma tran
```

5.4.11 Trực chuẩn hóa cơ sở, xác định ma trận chuyển cơ sở, tìm hình chiếu trực giao

Các phép toán như chuẩn hóa cơ sở, ma trận chuyển cơ sở và hình chiếu trực giao có thể thực hiện bằng các phép toán ma trận và vector trong SageMath.

```
sage: v1 = vector([1, 0])
sage: v2 = vector([0, 1])
sage: v1.normalized()  # Chuan hoa vector
sage: projection = v1.project_onto(v2)  # Tinh hinh chieu
   truc giao
```

5.4.12 Vẽ các đường cong phẳng, mặt bậc hai

Để vẽ các đường cong phẳng và mặt bậc hai, ta có thể sử dụng các hàm vẽ 2D và 3D trong SageMath. Dưới đây là cách vẽ đường cong và mặt bậc hai.

```
sage: plot(x^2 + y^2, (x, -2, 2), (y, -2, 2))  # Ve duong
   cong
sage: implicit_plot3d(x^2 + y^2 - z, (x, -2, 2), (y, -2, 2)
   , (z, 0, 10))  # Ve mat bac hai
```

5.5 Ứng dụng trong Xác suất thống kê

5.5.1 Tổ hợp, chỉnh hợp, giai thừa

SageMath hỗ trợ các hàm tính tổ hợp và giai thừa để giải các bài toán xác suất. Tuy nhiên, SageMath không có sẵn hàm tính chỉnh hợp, nên ta sẽ sử dụng cách tính theo đúng công thức $P(n,k) = \dfrac{n!}{(n-k)!}$.

```
# Tinh giai thua
sage: factorial(5)
120

# Tinh to hop chap 3 cua 5
sage: binomial(5, 3)
10

# Tinh chinh hop chap 2 cua 4
sage: factorial(4) / factorial(4 - 2)
12
```

5.5.2 Biến ngẫu nhiên và phân phối xác suất

SageMath có các hàm tích hợp để tính kỳ vọng, phương sai và độ lệch chuẩn của các biến ngẫu nhiên.

```
# Tinh ky vong va phuong sai cua bien ngau nhien
sage: var('x p')
sage: X = [1, 2, 3]
sage: P = [0.2, 0.5, 0.3]
sage: mean = sum(x * p for x, p in zip(X, P))
sage: variance = sum((x - mean)^2 * p for x, p in zip(X, P)
    )
sage: mean, variance
(2.10000000000000, 0.490000000000000)
```

Ta cũng có thể sử dụng từ thư viện `NumPy` để tính giá trị trung bình (mean), phương sai (variance) và độ lệch chuẩn như sau:

```
# Tinh ky vong va phuong sai cua bien ngau nhien voi NumPy
sage: import numpy as np
sage: data = [10, 20, 30, 40, 50]
sage: mean = np.mean(data)
sage: variance = np.var(data)
sage: std_dev = np.std(data)
sage: mean, variance, std_dev
(30.0, 200.0, 14.142135623730951)
```

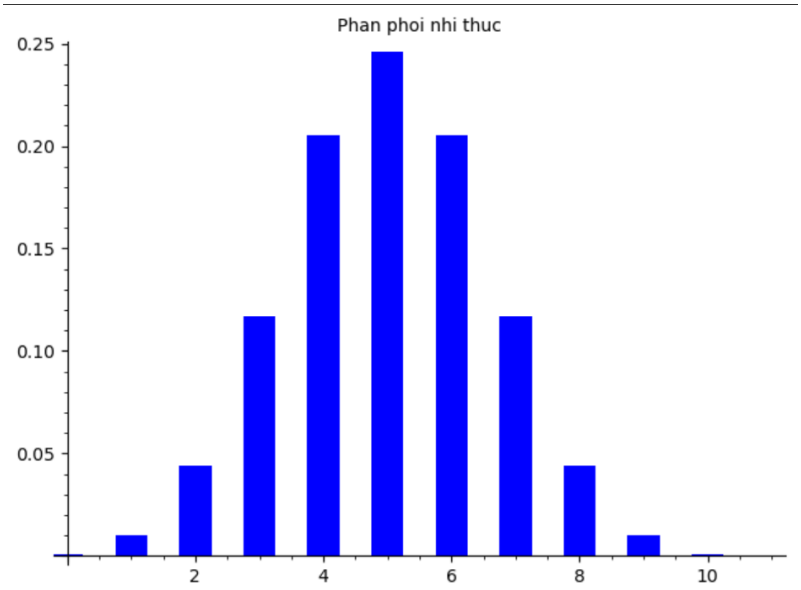
Hoặc sử dụng hàm từ `SciPy`:

```
# Tinh ky vong va phuong sai cua bien ngau nhien voi SciPy
sage: from scipy.stats import describe
sage: data = [10, 20, 30, 40, 50]
sage: stats = describe(data)
sage: stats.mean, stats.variance
(30.0, 200.0)
```

5.5.2.1 Vẽ đồ thị phân phối xác suất

Dưới đây là ví dụ vẽ đồ thị phân phối nhị thức với SageMath.

```
# Ve do thi phan phoi nhi thuc voi p = 0.5, n = 10
sage: var('k')
sage: p = 0.5
sage: n = 10
sage: P = [binomial(n, k) * p^k * (1 - p)^(n - k) for k in
    range(n + 1)]
sage: bar_chart(P, width=0.5, title='Phan phoi nhi thuc',
    axes=True)
```



Hình 5.12: Biểu đồ phân phối nhị thức với $(n, p) = (10, 0.5)$

5.5.3 Biến ngẫu nhiên nhiều chiều

Tính hàm mật độ xác suất biên và xác suất điều kiện.

```
# Phan phoi xac suat bien
sage: var('x y')
sage: f = x * y
sage: integrate(f, (x, 0, 1))  # Ham mat do xac suat bien
    theo y
1/2 * y^2
```

5.5.4 Ước lượng tham số

Tính khoảng tin cậy cho kỳ vọng và tỷ lệ.

```
# Uoc luong ky vong voi do tin cay 95%
sage: var('x n s')
sage: mu = x + 1.96 * s / sqrt(n)
sage: mu.subs({x: 50, s: 10, n: 100})
51.9600000000000
```

5.5.5 Kiểm định giả thuyết

Thực hiện kiểm định giả thuyết với mẫu và tính p-value.

```
# Kiem dinh Z
sage: var('xbar mu sigma n')
sage: Z = (xbar - mu) / (sigma / sqrt(n))
sage: Z.subs({xbar: 110, mu: 100, sigma: 15, n: 30})
3.651483716701107
```

5.5.6 Thống kê mô tả và suy diễn

Tính các tham số thống kê như trung bình, phương sai, độ lệch chuẩn.

```
# Thong ke mo ta voi scipy
sage: from scipy.stats import describe
sage: data = [10, 20, 30, 40, 50]
sage: describe(data)
# Ket qua: N, min, max, mean, variance
DescribeResult(nobs=5, minmax=(10, 50), mean=30.0, variance
    =250.0, skewness=0.0, kurtosis=-1.3)
```

5.5.7 Phân tích hồi quy

Hồi quy tuyến tính và kiểm định độ phù hợp của mô hình.

```
# Hoi quy tuyen tinh
sage: var('x')
sage: data = [(1, 2), (2, 4), (3, 6)]
sage: m, b = var('m b')
sage: model = m * x + b
sage: eqs = [model.subs(x=xv) == yv for xv, yv in data]
sage: solve(eqs, m, b)
[[m == 2, b == 0]]
```

5.5.8 Phân tích phương sai (ANOVA)

Phân tích phương sai để kiểm định giả thuyết về nhiều trung bình.

```
        # Phan tich ANOVA
        sage: from scipy.stats import f_oneway
        sage: group1 = [23, 20, 25, 22]
        sage: group2 = [27, 29, 30, 28]
        sage: f_oneway(group1, group2)
        # Ket qua: p-value va thong ke F
        F_onewayResult(statistic=24.0, pvalue=0.002713682035093796)
```

Chương 6

Kết luận và hướng phát triển

6.1 Tổng kết nội dung nghiên cứu

Trong quá trình thực hiện đồ án, em đã nghiên cứu và triển khai việc sử dụng SageMath vào giải quyết các bài toán toán học thuộc nhiều môn học khác nhau trong các học phần Toán đại cương tại Đại học Bách Khoa Hà Nội.

Thông qua việc khai thác các tính năng cốt lõi của SageMath, em đã đạt được những kết quả đáng khích lệ như sau:

- Tìm hiểu và ứng dụng các phép toán cơ bản và nâng cao trong SageMath, bao gồm tính toán đại số, giải tích, tổ hợp, thống kê và các phép toán số học.
- Sử dụng các hàm hỗ trợ mạnh mẽ của SageMath để giải quyết các bài toán thực tiễn như tính giới hạn, tích phân, đạo hàm, giải hệ phương trình tuyến tính, chéo hóa ma trận và phân tích Fourier.
- Xây dựng các ứng dụng tính toán thực tiễn trong các học phần Toán đại cương, từ các bài toán đại số trừu tượng đến các phép tính trong giải tích số và toán rời rạc.
- Tạo các đồ thị trực quan, minh họa các khái niệm toán học thông qua các hàm vẽ đồ thị 2D, 3D, biểu đồ phân phối xác suất, biểu đồ hàm số và các hình học không gian.
- Thiết kế SageMath Notebook giúp thầy cô và các bạn sinh viên tương tác và kiểm tra trực tiếp các ví dụ, bài tập, đồng thời tích hợp các thư viện Python để mở rộng khả năng tính toán.

Kết quả của đồ án đã chứng minh rằng SageMath là một công cụ mạnh mẽ trong việc hỗ trợ giảng dạy và nghiên cứu toán học, đặc biệt là trong các học phần Toán đại cương tại các trường đại học. Việc tích hợp SageMath vào quá trình học tập giúp sinh viên rèn luyện kỹ năng tính toán, tư duy lập trình, cũng như khả năng phân tích và trực quan hóa dữ liệu toán học.

Tuy nhiên, trong quá trình thực hiện, em cũng nhận thấy một số hạn chế cần khắc phục, sẽ được trình bày trong các phần tiếp theo. Bên cạnh đó, đồ án cũng mở ra những hướng nghiên cứu mới nhằm cải thiện việc ứng dụng SageMath trong giáo dục và nghiên cứu khoa học.

6.2 Những khó khăn và hạn chế của SageMath

Trong quá trình tìm hiểu và sử dụng SageMath, em đã gặp phải một số khó khăn và hạn chế, cả về mặt kỹ thuật lẫn trong việc ứng dụng vào các bài toán thực tiễn. Dưới đây là một số vấn đề nổi bật mà em nhận thấy:

6.2.1 Khó khăn trong cài đặt và cấu hình

Việc cài đặt SageMath không phải lúc nào cũng đơn giản, đặc biệt trên các hệ điều hành như Windows. Một số khó khăn em gặp phải bao gồm:

- SageMath yêu cầu dung lượng lớn và tài nguyên hệ thống tương đối cao.
- Việc tích hợp với các công cụ như Jupyter Notebook đôi khi gặp lỗi phiên bản và xung đột thư viện.
- Cài đặt SageMath trên Windows thường đòi hỏi sử dụng WSL (Windows Subsystem for Linux), gây khó khăn cho những người không quen với môi trường Linux.

6.2.2 Hạn chế về khả năng tính toán và thư viện

Mặc dù SageMath tích hợp nhiều thư viện mạnh mẽ, nhưng trong một số trường hợp đặc biệt, em nhận thấy có những hạn chế như sau:

- Một số phép toán phức tạp như tính giới hạn, tích phân đa biến hoặc giải phương trình vi phân đôi khi không cho kết quả chính xác hoặc không hỗ trợ trực tiếp.
- Các hàm thống kê và phân phối xác suất chưa phong phú so với các phần mềm chuyên dụng như R hoặc MATLAB.
- Việc xử lý dữ liệu lớn đôi khi không hiệu quả do thiếu các hàm tối ưu hóa chuyên dụng.

6.2.3 Hạn chế trong việc trực quan hóa

Mặc dù SageMath hỗ trợ vẽ đồ thị 2D và 3D, nhưng khi so sánh với các phần mềm như Mathematica hay MATLAB, khả năng trực quan hóa còn hạn chế:

- Việc tùy chỉnh đồ thị phức tạp đòi hỏi nhiều mã lệnh và không thân thiện với người mới bắt đầu.
- Các đồ thị 3D khi vẽ trong môi trường notebook có thể bị chậm và khó thao tác.
- Việc kết hợp nhiều đồ thị trên cùng một hệ trục không dễ dàng như trong các phần mềm chuyên nghiệp.

6.2.4 Khó khăn trong việc học và sử dụng

SageMath là một hệ thống mạnh mẽ nhưng không dễ tiếp cận đối với sinh viên mới làm quen với lập trình:

- Cú pháp của SageMath đôi khi gây nhầm lẫn vì nó kết hợp nhiều thư viện từ các ngôn ngữ khác nhau.
- Tài liệu hướng dẫn chính thức chưa thật sự đầy đủ, đặc biệt đối với các chức năng nâng cao.
- Việc tìm kiếm giải pháp cho các lỗi gặp phải trong quá trình sử dụng cũng là một thách thức, do cộng đồng người dùng không lớn bằng các phần mềm thương mại.

6.3 Định hướng phát triển và nghiên cứu trong tương lai

Trong quá trình thực hiện đồ án, em nhận thấy SageMath là một công cụ mạnh mẽ nhưng vẫn còn nhiều tiềm năng chưa được khai thác hết. Để nâng cao hiệu quả sử dụng, em xin đề xuất một số định hướng phát triển trong tương lai như sau:

6.3.1 Tích hợp AI để chuyển ngôn ngữ tự nhiên thành mã SageMath

Một trong những khó khăn lớn nhất đối với người dùng SageMath là phải thành thạo cú pháp và các hàm toán học trong phần mềm. Điều này đòi hỏi thời gian học tập và nắm vững ngôn ngữ lập trình của SageMath.

Để khắc phục vấn đề này, em đề xuất sử dụng trí tuệ nhân tạo (AI) để tự động chuyển đổi ngôn ngữ tự nhiên thành mã SageMath. Mô hình AI sẽ nhận đầu vào là câu lệnh toán học từ người dùng bằng ngôn ngữ tự nhiên, sau đó chuyển đổi thành mã SageMath, thực thi lệnh và trả về kết quả. Điều này sẽ giúp người dùng giải toán mà không cần học SageMath, nâng cao trải nghiệm sử dụng phần mềm, đặc biệt đối với sinh viên và người dùng không chuyên.

6.3.2 Phát triển giao diện thân thiện người dùng

Một trong những yếu tố quan trọng giúp SageMath tiếp cận rộng rãi hơn là việc cải thiện giao diện sử dụng. Hiện tại, giao diện của SageMath chủ yếu dựa trên dòng lệnh hoặc Jupyter Notebook, gây khó khăn cho người mới bắt đầu.

Trong tương lai, cần phát triển một ứng dụng trực quan hơn, có giao diện đồ họa giống như các phần mềm thương mại khác như Mathematica hay MATLAB. Giao diện này sẽ hỗ trợ:

- Nhập công thức toán học trực tiếp bằng cách kéo thả các biểu thức.
- Hiển thị kết quả ngay lập tức mà không cần chạy mã lệnh.
- Tích hợp các tính năng vẽ đồ thị, giải phương trình và biểu diễn ma trận với các nút bấm trực quan.

6.3.3 Xây dựng thư viện bài tập và tài liệu hỗ trợ học tập

Để hỗ trợ sinh viên trong quá trình học tập và nghiên cứu, em đề xuất xây dựng một thư viện bài tập và hướng dẫn giải chi tiết bằng SageMath. Thư viện này sẽ bao gồm:

- Các bài tập từ cơ bản đến nâng cao thuộc các học phần Toán đại cương.
- Ví dụ minh họa và mã SageMath tương ứng để sinh viên dễ dàng tiếp cận.
- Tài liệu giải thích các tính năng quan trọng của SageMath kèm theo ví dụ thực hành.

6.3.4 Tăng cường khả năng tính toán thống kê và xử lý dữ liệu lớn

Mặc dù SageMath đã tích hợp các thư viện thống kê và tính toán số học như NumPy, SciPy, nhưng vẫn chưa đáp ứng được nhu cầu xử lý dữ liệu lớn và phân tích thống kê phức tạp. Trong tương lai, cần:

- Tích hợp chặt chẽ hơn với các thư viện chuyên dụng như Pandas và R.
- Tối ưu hóa hiệu năng để tính toán nhanh hơn khi xử lý dữ liệu lớn.
- Cải thiện các hàm thống kê và phân tích dữ liệu để cạnh tranh với các phần mềm chuyên dụng.

6.3.5 Tăng cường cộng đồng người dùng và tài liệu tham khảo

Một trong những hạn chế hiện tại của SageMath là cộng đồng người dùng chưa thực sự mạnh như các phần mềm thương mại khác. Để phát triển bền vững, cần:

- Khuyến khích người dùng tham gia đóng góp vào mã nguồn và phát triển các gói mở rộng.
- Tạo diễn đàn trao đổi kinh nghiệm, giải đáp thắc mắc và chia sẻ tài liệu hướng dẫn.
- Xây dựng khóa học và video hướng dẫn sử dụng SageMath từ cơ bản đến nâng cao.

6.3.6 Kết luận

Với những định hướng trên, em hy vọng rằng SageMath sẽ ngày càng hoàn thiện và trở thành công cụ toán học mạnh mẽ, tiện dụng hơn cho sinh viên và các nhà nghiên cứu. Việc kết hợp giữa trí tuệ nhân tạo, giao diện thân thiện và cộng đồng người dùng mạnh mẽ sẽ giúp SageMath đạt được mục tiêu phổ cập trong giảng dạy và nghiên cứu toán học.

Lời tổng kết

Sau quá trình tìm hiểu và thực hiện đồ án, em đã hoàn thành việc nghiên cứu và ứng dụng SageMath vào giải quyết các bài toán thuộc các học phần Toán đại cương tại Đại học Bách Khoa Hà Nội. Qua đồ án này, em đã học được cách sử dụng SageMath để hỗ trợ giảng dạy và học tập, đồng thời phát triển khả năng tư duy toán học kết hợp với lập trình.

Việc sử dụng SageMath trong giải quyết các bài toán Giải tích, Đại số tuyến tính, Giải tích số, Xác suất thống kê và các môn học khác đã chứng minh rằng phần mềm này có khả năng hỗ trợ mạnh mẽ trong quá trình học tập và nghiên cứu. Đặc biệt, SageMath giúp tự động hóa nhiều phép toán phức tạp, giảm thiểu sai sót và tiết kiệm thời gian.

Bên cạnh đó, thông qua việc triển khai đồ án, em nhận ra rằng SageMath vẫn còn một số hạn chế nhất định như khó tiếp cận đối với người mới bắt đầu, khả năng trực quan hóa chưa cao và cần cài đặt phức tạp. Tuy nhiên, với tiềm năng của một phần mềm mã nguồn mở, SageMath hoàn toàn có thể được phát triển và cải thiện trong tương lai.

Trong quá trình thực hiện đồ án, em đã nhận được nhiều sự giúp đỡ, hướng dẫn và động viên từ thầy cô, bạn bè và gia đình. Em xin gửi lời cảm ơn sâu sắc đến thầy Lê Văn Tứ, giảng viên hướng dẫn, người đã luôn tận tình chỉ bảo, định hướng và hỗ trợ em trong suốt thời gian thực hiện đồ án.

Em cũng xin chân thành cảm ơn các thầy cô trong Khoa Toán - Tin, Đại học Bách Khoa Hà Nội đã cung cấp kiến thức nền tảng và tạo điều kiện thuận lợi trong quá trình học tập. Đặc biệt, em cảm ơn các bạn sinh viên trong lớp đã cùng nhau chia sẻ kinh nghiệm và đóng góp ý kiến quý báu.

Cuối cùng, em xin dành lời cảm ơn sâu sắc nhất đến gia đình, những người luôn ở bên cạnh, động viên và tạo điều kiện tốt nhất để em có thể tập trung học tập và nghiên cứu.

Em rất mong nhận được những ý kiến đóng góp từ thầy cô và các bạn để tài liệu được hoàn thiện hơn.

— HẾT —

Phụ lục A

Tài liệu đính kèm

A.1 Tài liệu đính kèm

Để hỗ trợ việc học và sử dụng SageMath một cách trực quan và thuận tiện hơn, em đã xây dựng một tài liệu Jupyter Notebook kèm theo toàn bộ các đoạn mã và hướng dẫn đi kèm trong đồ án.

Toàn bộ nội dung này được lưu trữ công khai tại địa chỉ GitHub:

`https://github.com/Ethan-HUST/SageMath-Tutorial`

Dự án bao gồm:

- File `SageMath_Notebook.ipynb` chứa toàn bộ các câu lệnh và ví dụ minh họa trong đồ án, phân chia theo từng học phần Toán đại cương.
- Các hướng dẫn sử dụng SageMath trực tuyến, không cần cài đặt.
- Các liên kết truy cập nhanh vào `SageMathCell` và `CoCalc`, giúp người dùng chạy trực tiếp mã mà không cần môi trường cục bộ.
- Nội dung tài liệu này ở định dạng PDF để tham khảo nhanh cùng với mã nguồn tương ứng.

Người dùng có thể:

- Tải về hoặc chạy trực tiếp Notebook từ GitHub thông qua nền tảng `CoCalc` hoặc `Google Colab` (có cấu hình SageMath).
- Sao chép từng đoạn mã từ Notebook để thử nghiệm với các nền tảng trực tuyến như `https://sagecell.sagemath.org`.

Tài liệu này nhằm hỗ trợ người dùng tiếp cận SageMath một cách trực quan, linh hoạt và hiệu quả hơn trong giảng dạy và học tập các học phần Toán học tại đại học.

Phụ lục B

Câu hỏi thường gặp

B.1 Hướng dẫn khắc phục lỗi khi cài đặt SageMath

Việc cài đặt SageMath có thể gặp khó khăn do phụ thuộc hệ điều hành hoặc môi trường. Một số gợi ý khắc phục:

- **Trên Windows:**
  - Dùng phiên bản SageMath cho Windows (bản '.exe' hoặc WSL).
  - Nếu cài qua WSL, cần đảm bảo đã bật tính năng WSL2 và cài Ubuntu đầy đủ.
  - Nên sử dụng môi trường 'conda' để dễ quản lý gói và tránh xung đột.
- **Trên Linux/macOS:**
  - Dùng bản '.tar.bz2' và biên dịch lại nếu cần tối ưu cho máy cá nhân.
  - Cài thiếu thư viện hệ thống? Kiểm tra lỗi và cài thêm các gói 'libreadline-dev', 'build-essential', 'python3-dev',...
- **Lỗi "Permission Denied":** Chạy với quyền 'sudo' hoặc chỉnh lại quyền thư mục.
- **Lỗi thiếu Python / sai phiên bản:** SageMath đi kèm Python riêng – không nên can thiệp nếu không có kinh nghiệm.

B.2 Các lỗi thường gặp và cách xử lý

- **Không khởi động được SageMath:**
  - Kiểm tra biến môi trường 'PATH', đảm bảo đường dẫn SageMath chính xác.

  - Trên Windows, thử khởi động lại máy hoặc dùng dòng lệnh ‘sage -n jupyter‘.

- **Không vẽ được đồ thị / đồ thị không hiển thị:**
  - Thiếu thư viện đồ họa (như ‘matplotlib‘)? Cài lại bằng: `sage -pip install matplotlib`.
  - Nếu dùng WSL, cần cài thêm ‘x11‘, ‘VcXsrv‘, hoặc dùng WSLg (trên Windows 11).
- **Cài gói Python bị lỗi trong SageMath:**
  - Dùng đúng lệnh `sage -pip install <ten_goi>` thay vì pip ngoài.
- **Lỗi tương thích phiên bản:**
  - Kiểm tra trang chủ SageMath xem phiên bản bạn cài đã tương thích với hệ điều hành chưa.

B.3 Tài nguyên học tập và tham khảo về SageMath

- **Trang chính thức:** `https://www.sagemath.org`
- **Tài liệu hướng dẫn:** `https://doc.sagemath.org`
- **Notebook thử online:** `https://sagecell.sagemath.org`
- **Jupyter Notebook:** SageMath tương thích tốt với Jupyter, hỗ trợ viết tài liệu, bài giảng tương tác.
- **Diễn đàn thảo luận:** `https://ask.sagemath.org`
- **Tài liệu tiếng Việt (cộng đồng):**
  - GitHub: `https://github.com/sagemath`
  - Một số trường đại học có chia sẻ giáo trình nội bộ: hãy tìm với từ khóa ‘‘`SageMath site:hust.edu.vn`’’.
- **Sách tham khảo:**
  - *Sage for Undergraduates* – Gregory Bard
  - *Computational Mathematics with SageMath* – Paul Zimmermann et al.

Phụ lục C

Tài liệu tham khảo

1. Sách và giáo trình

- Paul Zimmermann, et al., *Computational Mathematics with SageMath*, Springer, 2018.
- William Stein, *Sage for Power Users*, University of Washington, 2011.
- Gregory Bard, *Sage for Undergraduates*, American Mathematical Society, 2015.
- Robert A. Beezer, *A First Course in Linear Algebra (with Sage)*, University of Puget Sound, 2017.

2. Bài báo và công trình khoa học

- William Stein et al., "Sage: Open Source Mathematical Software," *Notices of the American Mathematical Society*, 2008.
- Nicolas M. Thiéry, "Sage: un logiciel pour les mathématiques libres et collaboratives," *Gazette des Mathématiciens*, 2007.
- P. Zimmermann, "The Design of SageMath," *Lecture Notes in Computer Science*, Springer, 2016.
- Jeroen Demeyer, "The Development of SageMath as an Open Source Project," *Journal of Open Source Software*, 2019.
- Clément Pernet, "High-Performance Linear Algebra with SageMath," *Journal of Symbolic Computation*, 2020.

3. Nguồn tham khảo trực tuyến

- Trang chủ SageMath: `https://www.sagemath.org`

- Tài liệu chính thức: `https://doc.sagemath.org`
- Hệ thống thử nghiệm trực tuyến: `https://sagecell.sagemath.org`
- Diễn đàn hỗ trợ cộng đồng: `https://ask.sagemath.org`
- Tài liệu Python: `https://docs.python.org`
- StackOverflow (chủ đề SageMath và Python): `https://stackoverflow.com/questions/tagged/sagemath`
- GitHub SageMath: `https://github.com/sagemath/sage`
- SageMath Wiki: `https://wiki.sagemath.org`
- SageManifolds: `https://sagemanifolds.obspm.fr`
- Hướng dẫn sử dụng SageMath trên Math3ma: `https://www.math3ma.com/sagemath`