**دانشگاه صنعتي اميركبير**
**(پلی تکنیک تهران)**

**دانشکده ریاضی و علوم کامپیوتر**

استاد درس: مهدی دهقان
بهار 1403

# الگوریتم باریس جهت محاسبهٔ دترمینان ماتریس
## درس جبر خطی عددی

آرین رادجو
شماره دانشجویی : 9913013

پروژه نهایی

# چکیده

این گزارش به بررسی و پیاده‌سازی الگوریتم باریس برای محاسبه دترمینان ماتریس‌های مربعی می‌پردازد. الگوریتم باریس با استفاده از محاسبات صحیح، دترمینان را به طور دقیق و کارآمد محاسبه می‌کند.

## کلید واژه ها:

دترمینان ها، حساب دقیق، حساب اعداد صحیح، جبر خطی

# فهرست مطالب

**مقدمه ..... 4**
شرح مسأله ..... 4
توضیحات پیش‌نیاز ..... 5
مزایا و چالش ها ..... 6
مزایا ..... 6
چالش‌ها ..... 7
**راهکار پیشنهادی ..... 8**
ورودی ..... 8
حلقه‌ها ..... 8
نحوه برخورد با عناصر صفر در قطر اصلی ..... 9
خروجی و جواب نهایی ..... 9
**نتایج عددی ..... 10**
مقایسه پیچیدگی زمانی الگوریتم باریس با بقیه الگوریتم‌ها ..... 10
1. پیچیدگی زمانی الگوریتم باریس ..... 10
2. پیچیدگی زمانی روش حذف گاوسی ..... 10
3. روش‌های LU و QR ..... 10
4. الگوریتم‌های بهینه‌سازی شده و موازی ..... 11
مقایسه عملکرد الگوریتم ها در کد ..... 12
1. شرح ماتریس‌های آزمایشی ..... 12
2. محاسبه دترمینان ..... 12
3. مقایسه نتایج ..... 12
4. زمان اجرا ..... 13
5. تحلیل پایداری ..... 13
**پیوست ..... 14**
**مراجع ..... 17**
**منابع ..... 18**

# مقدمه

## شرح مسأله

محاسبه دترمینان یک ماتریس مربعی یکی از مسائل اساسی در ریاضیات کاربردی و جبر خطی است. دترمینان یک عدد واحد است که می‌تواند اطلاعات زیادی درباره خصوصیات یک ماتریس فراهم کند. برای مثال، دترمینان مشخص می‌کند که آیا یک ماتریس معکوس‌پذیر است یا خیر. همچنین، دترمینان در تحلیل پایداری سیستم‌های دینامیکی، حل سیستم‌های معادلات خطی، محاسبات هندسی، و بسیاری از کاربردهای مهندسی اهمیت زیادی دارد.

با این حال، محاسبه دترمینان به ویژه برای ماتریس‌های بزرگ و پیچیده می‌تواند چالش‌برانگیز باشد. روش‌های سنتی مانند حذف گاوسی ممکن است در برابر خطاهای عددی آسیب‌پذیر باشند و دقت کافی را فراهم نکنند. این مشکل به خصوص زمانی بیشتر نمود پیدا می‌کند که عناصر ماتریس اعداد صحیح بزرگی باشند. به همین دلیل، نیاز به الگوریتم‌های دقیق‌تر و کارآمدتر احساس می‌شود که بتوانند دترمینان ماتریس‌ها را با دقت و سرعت بیشتری محاسبه کنند.

یکی از این الگوریتم‌های پیشرفته، الگوریتم باریس (Bareiss Algorithm) است که برای محاسبه دترمینان ماتریس‌ها با استفاده از محاسبات صحیح طراحی شده است. این الگوریتم با بهره‌گیری از روش‌های پیشرفته و جلوگیری از خطاهای عددی، می‌تواند دترمینان را با دقت بالاتری نسبت به روش‌های سنتی محاسبه کند. در ادامه ، به معرفی و بررسی این الگوریتم می پردازیم.

## توضیحات پیش‌نیاز

برای درک و پیاده‌سازی الگوریتم باریس جهت محاسبه دترمینان ماتریس‌ها، آشنایی با چندین مفهوم پایه‌ای در ریاضیات و جبر خطی ضروری است. در این بخش، به بررسی این پیش‌نیازها می‌پردازیم تا زمینه مناسبی برای فهم الگوریتم فراهم شود.

**ماتریس و عملیات ماتریسی**: درک مفاهیم پایه‌ای ماتریس‌ها، شامل انواع ماتریس‌ها (مربعی، قطری، متقارن)، عملیات ماتریسی (جمع، ضرب، ترانهاده)، و خاصیت‌های آن‌ها.

**دترمینان**: آشنایی با مفهوم دترمینان، روش‌های محاسبه آن برای ماتریس‌های کوچک، و اهمیت دترمینان در جبر خطی و کاربردهای آن.[1]

**ماتریس معکوس**: فهم شرایط معکوس‌پذیری یک ماتریس و نحوه محاسبه ماتریس معکوس.[2]

**دقت و پایداری عددی**: آشنایی با مفاهیم دقت و پایداری عددی در محاسبات، به ویژه در الگوریتم‌های ماتریسی.

**خطاهای عددی**: درک انواع خطاهای عددی مانند خطاهای گرد کردن و سرریز عددی، و تاثیر آن‌ها بر نتایج محاسبات.[3]

**پیاده‌سازی الگوریتم‌ها**: مهارت در برنامه‌نویسی و پیاده‌سازی الگوریتم‌های ریاضی با استفاده از زبان‌های برنامه‌نویسی مانند Python یا C++.

**کتابخانه‌های محاسباتی**: آشنایی با کتابخانه‌های محاسباتی مانند NumPy در Python که ابزارهای لازم برای کار با ماتریس‌ها و محاسبات عددی را فراهم می‌کنند.[4]

با داشتن این پیش‌نیازها، می‌توان به سراغ درک و پیاده‌سازی الگوریتم باریس رفت و از مزایای آن در محاسبات دقیق دترمینان ماتریس‌ها بهره مند شد.

# مزایا و چالش ها

### مزایا

الگوریتم باریس به عنوان یک روش موثر برای محاسبه دترمینان ماتریس‌ها، چندین مزیت مهم را ارائه می‌دهد:

**دقت بالا**:

یکی از مهم‌ترین مزایای الگوریتم باریس استفاده از محاسبات صحیح برای کاهش خطاهای عددی است. این الگوریتم با کاهش نیاز به گردکردن اعداد و انجام محاسبات با اعداد صحیح تا حد امکان، دقت بالاتری نسبت به روش‌های معمولی مانند حذف گاوسی فراهم می‌کند.

**پایداری نسبت به ماتریس‌های منفرد:**

در الگوریتم باریس، اگر در حین اجرای الگوریتم یک عنصر قطری صفر مواجه شود، به دنبال یک سطر دیگر با عنصر غیر صفر در همان ستون می‌گردد. اگر چنین سطری پیدا نشود (یعنی همه عناصر آن ستون صفر باشند)، الگوریتم نتیجه‌گیری می‌کند که ماتریس منفرد است و دترمینان آن صفر است. این ویژگی به الگوریتم اجازه می‌دهد تا به طور صحیح و پایدار با ماتریس‌های منفرد برخورد کند و از بروز خطاهای عددی جلوگیری کند.

**استفاده از عملیات ساده‌تر**:

الگوریتم باریس از عملیات ریاضی ساده‌تری نسبت به روش‌های دیگر استفاده می‌کند. این مسئله منجر به کاهش پیچیدگی و افزایش سرعت اجرای الگوریتم می‌شود.

## چالش‌ها

در کنار مزایای بسیاری که الگوریتم باریس ارائه می‌دهد، چندین چالش نیز در پیاده‌سازی و استفاده از آن وجود دارد:

**کاربرد محدود در محیط‌های خاص:**

در مواردی که نیاز به محاسبات اعشاری و تقریب‌های سریع است، الگوریتم باریس ممکن است گزینه مناسبی نباشد زیرا تمرکز آن بر دقت بالا در حساب صحیح است. همچنین الگوریتم باریس به طور خاص برای محاسبه دترمینان ماتریس‌های مربعی طراحی شده است. برای سیستم‌های غیر مربعی یا مسائل دیگری که نیاز به تحلیل ماتریس‌های غیر مربعی دارند، این الگوریتم قابل استفاده نیست و باید از روش‌های دیگر استفاده کرد.

**حساسیت به صفر شدن عناصر قطری:**

در صورتی که عنصری در قطر اصلی ماتریس برابر با صفر شود، الگوریتم نیاز به جابجایی سطرها دارد تا از ناپایداری جلوگیری شود. این امر می‌تواند پیاده‌سازی را پیچیده‌تر کند و نیاز به مدیریت دقیق‌تری داشته باشد.

**محدودیت‌های عددی:**

الگوریتم باریس به دلیل استفاده از محاسبات صحیح، ممکن است در مواجهه با اعداد بسیار بزرگ یا بسیار کوچک با محدودیت‌های عددی روبرو شود. این مسئله می‌تواند باعث کاهش دقت یا افزایش زمان محاسبه شود.

**نیاز به حافظه بیشتر:**

به دلیل نیاز به ذخیره و مدیریت دقیق مقادیر ماتریس در طی محاسبات، الگوریتم باریس ممکن است نسبت به برخی روش‌های ساده‌تر، نیاز به فضای حافظه بیشتری داشته باشد.

# راهکار پیشنهادی

## ورودی

در ابتدا برای ورودی با کمک تابع generate_random_matrix یک ماتریس مربعی n×n می سازیم و در ادامه با کمک تابع bareiss_algorithm دترمینان ماتریس داده شده را محاسبه می کنیم.

## حلقه‌ها

الگوریتم باریس از سه حلقه تودرتو برای به روزرسانی عناصر ماتریس استفاده می‌کند:

- **حلقه بیرونی (k):**
  - این حلقه از k=0 تا k = n-1 اجرا می شود
  - در هر مرحله k,الگوریتم بررسی می‌کند که آیا عنصر قطری M[k,k ] برابر صفر است یا خیر.
  - اگر صفر باشد سطرهای ماتریس را با یکدیگر جابجا می‌کند تا یک عنصر غیر صفر در همان ستون پیدا کند. این عملیات شامل تغییر علامت نهایی دترمینان می‌شود.

- **حلقه داخلی (i و j):**
  - برای هر مقدار k حلقه‌های داخلی برای i = k+1 تا i = n-1 و j = k+1 تا j = n-1 اجرا می شوند. در این حلقه‌ها، ماتریس با استفاده از فرمول زیر به روز رسانی می‌شود:
  - اگر k > 0 باشد M[i,j] بر M[k-1,k-1] تقسیم می شود.

$$M_{i,j} = \frac{M_{i,j}M_{k,k} - M_{i,k}M_{k,j}}{M_{k-1,k-1}}$$

## نحوه برخورد با عناصر صفر در قطر اصلی

اگر در هر مرحله k , عنصر M[k,k ] برابر با صفر باشد، الگوریتم شروع به جستجوی یک سطر با عنصر غیر صفر در همان ستون را می‌کند. سپس، این دو سطر جا به جا می‌شوند و علامت نهایی دترمینان معکوس می‌شود. اگر همه عناصر در همان ستون صفر باشند، دترمینان ماتریس صفر خواهد بود.

## خروجی و جواب نهایی

در نهایت، الگوریتم دترمینان ماتریس اصلی را به دست می‌آورد که در عنصر M[-1,-1 ] ماتریس به دست می‌آید و با علامت آخرین جابه‌جایی‌ها (اگر انجام شده باشد) ضرب می‌شود. این مقدار به عنوان خروجی نهایی ارائه می‌شود.

# نتایج عددی

## مقایسه پیچیدگی زمانی الگوریتم باریس با بقیه الگوریتم‌ها

برای درک بهتر کارایی الگوریتم باریس، لازم است آن را با دیگر روش‌های محاسبه دترمینان ماتریس و حل سیستم‌های خطی مقایسه کنیم.

### 1. پیچیدگی زمانی الگوریتم باریس

پیچیدگی زمانی الگوریتم باریس( O(n^3 است. این به این معنی است که زمان اجرای الگوریتم به صورت چندجمله‌ای از مرتبه سه با تعداد عناصر ماتریس (تعداد سطرها یا ستون‌ها) تغییر می‌کند. این پیچیدگی مشابه پیچیدگی زمانی روش حذف گاوسی استاندارد است، اما الگوریتم باریس از نظر دقت عددی و پایداری در برخی موارد بهتر عمل می‌کند.

### 2. پیچیدگی زمانی روش حذف گاوسی

روش حذف گاوسی نیز پیچیدگی زمانی ( O(n^3 دارد. با این حال، حذف گاوسی تعداد بیشتری عملیات تقسیم را شامل می‌شود که می‌تواند به خطای عددی بیشتری منجر شود. در حذف گاوسی، تقریباً نیمی از عملیات‌ها شامل تقسیم می‌شوند، در حالی که در الگوریتم باریس، تعداد تقسیمات به طور قابل توجهی کاهش می‌یابد.

### 3. روش‌های LU و QR

روش‌های فاکتورگیری LU و QR نیز هر دو پیچیدگی زمانی ( O(n^3 دارند. این روش‌ها معمولاً برای حل سیستم‌های خطی و محاسبه دترمینان ماتریس‌ها استفاده می‌شوند. در این روش‌ها، ماتریس به دو ماتریس مثلثی (LU) یا یک ماتریس مثلثی و یک ماتریس اورتوگونال (QR) تجزیه می‌شود. این روش‌ها

نیز در مواجهه با ماتریس‌های دارای اعداد بزرگ یا کوچک بسیار، ممکن است با مشکلات پایداری عددی مواجه شوند.

### 4. الگوریتم‌های بهینه‌سازی شده و موازی

برای ماتریس‌های بسیار بزرگ، الگوریتم‌های بهینه‌سازی شده و موازی وجود دارند که می‌توانند کارایی بهتری داشته باشند. این الگوریتم‌ها با استفاده از تکنیک‌های موازی‌سازی و بهینه‌سازی سخت‌افزاری، زمان اجرا را بهبود می‌بخشند. برخی از این روش‌ها حتی پیچیدگی زمانی کمتری از ( O(n^3 دارند، اما نیازمند سخت‌افزار خاص و پیچیدگی‌های پیاده‌سازی بیشتر هستند.

## مقایسه عملکرد الگوریتم ها در کد

### 1. شرح ماتریس‌های آزمایشی

برای آزمایش الگوریتم باریس، چندین ماتریس با ابعاد و ویژگی‌های مختلف تولید شده‌اند. این ماتریس‌ها شامل ماتریس‌های همانی، ماتریس‌های تصادفی با مقادیر کوچک و بزرگ، و ماتریس‌های پایین‌مثلثی و بالامثلثی می‌باشند.

### 2. محاسبه دترمینان

| ماتریس | باریس | حدفی گاوس | LU |
|---|---|---|---|
| B(3*3) | 14.0 | 14.0000000000002 | 14.0000000000002 |
| I(4*4) | 1.0 | 1.0 | 1.0 |
| D(3*3) | 24.0 | 24.0 | 24.0 |
| Random matrix(10*10) | 186492.0 | 186491.999999995 | |

### 3. مقایسه نتایج

همان طور که می بینید هر سه الگوریتم دترمینان را به خوبی محاسبه می کنند اما در دو روش گاوس و LU در بعضی موارد خطا های کوچکی یافت می شود اما در باریس مقادیر به صورت دقیق محاسبه می شوند.

### 4. زمان اجرا

| ماتریس | باریس | حذفی گاوس | LU |
|---|---|---|---|
| B(3*3) | 0.001374 | 0.000430 | 0.000948 |
| I(4*4) | 0.00202 | 0.000856 | 0.000885 |
| D(3*3) | 0.00254 | 0.000716 | 0.000915 |
| Random matrix(10*10) | 0.065903 | 0.006340 | 0.002264 |

### 5. تحلیل پایداری

الگوریتم باریس در مواجهه با ماتریس‌هایی که مقادیر بسیار بزرگ یا کوچک دارند، دقت و پایداری بهتری نسبت به حذف گاوسی نشان داد. در برخی موارد که مقادیر عناصر ماتریس بسیار بزرگ یا کوچک بودند، الگوریتم حذف گاوسی دچار خطای عددی شد، در حالی که الگوریتم باریس پایداری خود را حفظ کرد.

نتایج عددی نشان می‌دهند که الگوریتم باریس علاوه بر پیچیدگی زمانی مشابه با دیگر الگوریتم‌های استاندارد، دقت و پایداری بهتری در مواجهه با شرایط خاص دارد(اگر چه از لحاظ زمانی مقداری کندتر است). این ویژگی‌ها الگوریتم باریس را به یک گزینه مناسب برای محاسبه دترمینان ماتریس‌ها و حل سیستم‌های خطی در کاربردهای عملی تبدیل می‌کنند.

# پیوست

```python
import numpy as np


def generate_random_matrix(n):
    matrix = np.random.randint(0, 5, size=(n, n))
    return matrix


def bareiss_algorithm(A):
    n = A.shape[0]
    M = A.astype(np.float64).copy()
    sign = 1



    for k in range(n):
        # If the diagonal element is zero, swap rows
        if M[k, k] == 0:
            # Find a row with non-zero element in the same column
            swap_row = k + 1
            while swap_row < n and M[swap_row, k] == 0:
                swap_row += 1
            # If all elements in the column are zero, determinant is zero
            if swap_row == n:
                return 0
            # Swap rows
            M[[k, swap_row]] = M[[swap_row, k]]
            sign = - sign
```

```
        for i in range(k + 1, n):
            for j in range(k + 1, n):
                M[i, j] = (M[i, j] * M[k, k] - M[i, k] * M[k, j])
                if k > 0:
                    M[i, j] /= M[k-1, k-1]



    return sign * M[-1, -1]


# Example usage

A = np.array([
    [0, 2, 2],
    [2, 4, 2],
    [2, 2, 4]
], dtype=float)

B = np.array([
    [3, 1, 6],
    [2, 4, 3],
    [1, 5, 2]
], dtype=float)

C = generate_random_matrix(4)

print("Original Matrix:")
print(C)
```

```
determinant = bareiss_algorithm(C)


print("\nDeterminant of the original matrix:")
print(determinant)
```

# مراجع

[1] https://www.cuemath.com/algebra/matrices-and-determinants/

[2] https://www.cuemath.com/algebra/inverse-of-a-matrix/

[3] https://en.wikipedia.org/wiki/Numerical_error

[4] https://www.w3schools.com/python/numpy/default.asp

# منابع

https://en.wikipedia.org/wiki/Bareiss_algorithm#:~:text=In%20mathematics%2C%20the%20Bareiss%20algorithm,(there%20is%20no%20remainder).

https://cs.stackexchange.com/questions/124759/determinant-calculation-bareiss-vs-gauss-algorithm

https://www.ams.org/journals/mcom/1968-22-103/S0025-5718-1968-0226829-0/

https://chatgpt.com/

لینک کولب:
https://colab.research.google.com/drive/1_XY8LFSuqJNdLwdPsV5tsBnNEPVJfjt5#scrollTo=dS1HdInSiKGq