LA_PROJECT_4002023027_4013333054

محمدامین فهیمی – ساغر قاسم زاده

کارآیی زمانی روش گاوس بسیار بهتر از روش بسط لاپلاس هستند

فضای مصرفی هر دو روش گاوس و بسط لاپلاس در صورت پیاده‌سازی بهینه آنها، همان فضای لازم برای ذخیره یک ماتریس مربعی از مرتبه nاست

هر سه روش ظاهری بازگشتی - با **تقسیم و غلبه** - دارند که حل مسئله از مرتبه nnرا به حل زیرمسئله یا زیرمسائلی از مرتبه $n-1n-1$تقسیم می‌کنند. اما پیاده‌سازی غیربازگشتی این روش‌ها نیز ممکن است

بر اساس روش بسط لاپلاس و با استفاده از استقرای ریاضی، می‌توان ثابت کرد که اگر تمامی درایه‌های یک ماتریس مربعی اعداد صحیح باشند، دترمینان آن نیز عدد صحیح خواهد بود. روش بسط لاپلاس صحیح بودن عدد دترمینان را در این شرایط تضمین می‌کند. چرا که تنها از اعمال جمع و ضرب تشکیل شده است که صحیح بودن عدد را تغییر نمی‌دهند. اما دو روش دیگر - با توجه به این که شامل عمل تقسیم نیز هستند - در زمان پیاده‌سازی ممکن است خطای محاسباتی ایجاد کنند. چرا که اکثر زبان‌های برنامه‌نویسی اعداد را به دو فرم صحیح یا اعشاری - با دقت مشخص - ذخیره می‌کنند. بنابراین اعداد گویای غیرصحیح به صورت تقریبی ذخیره شده و در محاسبات هم به همان صورت تقریبی به کار می‌روند

منبع :الگوریتمستان

برای مقایسه پیچیدگی زمانی و مکانی دو روش محاسبه دترمینان ماتریس، یعنی روش بسط لاپلاس و روش حذفی گاوس-جردن، باید به تحلیل دقیق هر کدام از این روش‌ها پرداخته و مقایسه کنیم:

روش بسط لاپلاس

1. **پیچیدگی زمانی:**
   - روش بسط لاپلاس به صورت بازگشتی عمل می‌کند. برای محاسبه دترمینان ماتریس n× n، این روش به محاسبه دترمینان‌های ماتریس‌های کوچکتر نیاز دارد.
   - اگر T(n)T(n)T(n)زمان مورد نیاز برای محاسبه دترمینان ماتریس n× nباشد، به صورت بازگشتی داریم : T(n)=n×T(n−1)T(n) = n \times T(n-1)T(n)=n×T(n−1)
   - بنابراین، پیچیدگی زمانی این روش به صورت O(n!)O(n!)O(n!)خواهد بود که بسیار بالا است و برای ماتریس‌های بزرگ به هیچ وجه عملی نیست.
2. **پیچیدگی مکانی:**
   - روش بسط لاپلاس به حافظه اضافی برای نگهداری ماتریس‌های کوچکتر نیاز دارد.
   - با توجه به بازگشتی بودن این روش، حافظه لازم برای ذخیره مراحل بازگشت تقریباً O(n)است.
   - بنابراین، پیچیدگی مکانی این روش به صورت O(n)خواهد بود.

روش حذفی گاوس-جردن

1. **پیچیدگی زمانی:**
   - روش حذفی گاوس-جردن بر مبنای عملیات سطر به سطر روی ماتریس عمل می‌کند تا ماتریس به صورت مثلثی بالایی تبدیل شود.
   - تعداد عملیات اصلی مورد نیاز برای تبدیل ماتریس n×nبه صورت مثلثی بالایی برابر با O(n^3)است.
   - بنابراین، پیچیدگی زمانی این روش به صورت O(n^3)خواهد بود.
2. **پیچیدگی مکانی:**
   - در روش حذفی گاوس-جردن نیاز به حافظه اضافی برای ذخیره ماتریس اولیه و ماتریس‌های واسطه داریم.
   - با این حال، تعداد خانه‌های حافظه مورد نیاز ثابت و برابر با n^2است.
   - بنابراین، پیچیدگی مکانی این روش به صورت O(n^2)خواهد بود.

مقایسه نهایی

- **پیچیدگی زمانی:**
  - روش بسط لاپلاس(O(n! :
  - روش حذفی گاوس-جردن(O(n^3 :
  - نتیجه: روش حذفی گاوس-جردن از نظر پیچیدگی زمانی بسیار بهینه‌تر از روش بسط لاپلاس است و برای ماتریس‌های بزرگ قابل استفاده است.

- **پیچیدگی مکانی:**
  - روش بسط لاپلاس(O(n
  - روش حذفی گاوس-جردن(O(n^2
  - نتیجه: روش بسط لاپلاس از نظر پیچیدگی مکانی بهینه‌تر است، اما به دلیل پیچیدگی زمانی بسیار بالا، معمولاً در عمل برای ماتریس‌های بزرگ قابل استفاده نیست.

در نتیجه، روش حذفی گاوس-جردن به دلیل پیچیدگی زمانی به مراتب کمتر، معمولاً برای محاسبه دترمینان ماتریس‌های بزرگ‌تر به کار می‌رود، هرچند که پیچیدگی مکانی بیشتری دارد.

و در مقایسه با این سه روش بهترین روش , روش مقاله امید رضایی فر است .

Table 7
Comparison of time consumptions among different methods

Order	Old time (s)	New time (s)
3	0.0160	0.0000
4	0.0160	0.0000
5	0.0310	0.0150
6	0.1400	0.0670
7	0.4690	0.1250
8	3.3900	0.5470
9	30.4540	2.2960
10	232.1720	7.5160
11	2433.3334	30.2150

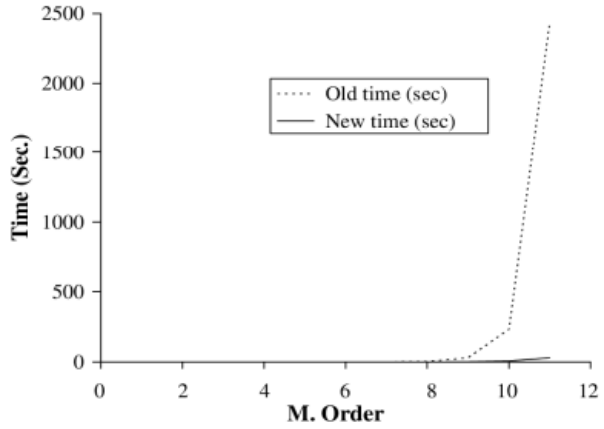

Fig. 1. Speed comparison between the new and expanding method.

دقت :به دلیل کاهش عملیات محاسباتی در روش جدید، احتمال خطاهای عددی کمتر است

توضیحات:

در فولدر "سوال یک" >>>> دترمینان ها هستن با نام های:

1-final-terminal

2-final-terminal

3-final-terminal

و

1-final-GUI

2-final- GUI

3-final- GUI

که گرافیکی پیاده سازی شده

در فولدر "سوال دو و سه" >>> همین فایل هست که توضیحاتی در مورد کدام روش بهتر است و مقایسه انها

در فولدر "سوال اخر">>> مربوط به cipher هست که هر دو صورت گرافیکی و کامندلاین پیاده سازی شده

5-final1-terminal انکریپشن

5-final2-terminal دیکریپشن

5-final1-GUI انکریپشن

5-final2-GUI دیکریپشن

GitHub Repository :
https://github.com/MA82F/linear-algebra
gui is implemented