Informatik
Hauptcampus

HOCHSCHULE TRIER

Implementierung eines Verfahrens zu effizienten Berechnung oszillierender Integrale

Implementation of a method for the efficient calculation of oscillatory integrals

Tobias Arens

Master-Abschlussarbeit

Betreuer: Titel Vorname Nachname

Mehren, 08.10.2022

Kurzfassung

In der Kurzfassung soll in kurzer und prägnanter Weise der wesentliche Inhalt der Arbeit beschrieben werden. Dazu zählen vor allem eine kurze Aufgabenbeschreibung, der Lösungsansatz sowie die wesentlichen Ergebnisse der Arbeit. Ein häufiger Fehler für die Kurzfassung ist, dass lediglich die Aufgabenbeschreibung (d.h. das Problem) in Kurzform vorgelegt wird. Die Kurzfassung soll aber die gesamte Arbeit widerspiegeln. Deshalb sind vor allem die erzielten Ergebnisse darzustellen. Die Kurzfassung soll etwa eine halbe bis ganze DIN-A4-Seite umfassen.

Hinweis: Schreiben Sie die Kurzfassung am Ende der Arbeit, denn eventuell ist Ihnen beim Schreiben erst vollends klar geworden, was das Wesentliche der Arbeit ist bzw. welche Schwerpunkte Sie bei der Arbeit gesetzt haben. Andernfalls laufen Sie Gefahr, dass die Kurzfassung nicht zum Rest der Arbeit passt.

Abstract

The same in English.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1

Einleitung und Problemstellung

Begonnen werden soll mit einer Einleitung zum Thema, also Hintergrund und Ziel erläutert werden.

Weiterhin wird das vorliegende Problem diskutiert: Was ist zu lösen, warum ist es wichtig, dass man dieses Problem löst und welche Lösungsansätze gibt es bereits. Der Bezug auf vorhandene oder eben bisher fehlende Lösungen begründet auch die Intention und Bedeutung dieser Arbeit. Dies können allgemeine Gesichtspunkte sein: Man liefert einen Beitrag für ein generelles Problem oder man hat eine spezielle Systemumgebung oder ein spezielles Produkt (z.B. in einem Unternehmen), woraus sich dieses noch zu lösende Problem ergibt.

Im weiteren Verlauf wird die Problemstellung konkret dargestellt: Was ist spezifisch zu lösen? Welche Randbedingungen sind gegeben und was ist die Zielsetzung? Letztere soll das beschreiben, was man mit dieser Arbeit (mindestens) erreichen möchte.

1.1 Zielsetzung

1.2 Aufbau der Arbeit

2

Das Verfahren

In diesem Kapitel wird das zu implemnetierende VErfahren vorgestellt.

2.1 Accoustic single-layer integral operator

Welches Problem löst das Verfahren:

$$\Delta I_{r,\Delta}(k) = \int_{\Delta} \frac{e^{ik(\|r-r'\|+\theta\cdot r')}}{\|r-r'\|}\, dr'$$

Wie wird es im Paper angegangen:

Und dann auf die relevanten Mathematischen "Grundlagenëingehen die verwendet werden um das Verfahren implementieren:

2.2 Mathematische Grundlagen

2.2.1 Cauchy-Integral-Theorem

2.2.2 Gauss-Laguerre Quadratur

2.2.3 Steepest Descent Method

3

Vorgehensweise

- Einfache implementierung in simplen FUnktionen
- Hotpath-Änalyse"
- Parallelisierung mit OneTBB
- Evaluierung von Parallen Architekturen

  Vorgehensweise zum Optimieren darlegen.
  Historie der Implementierung ggf darstellen?

3.1 Profiling

Hotpath darstellen?

3.2 Benchmarks

Automatische Benchmarks mit Googlebenchmark =¿ genereller Überblick
Implementiertes Matlab Plugin kann im direkten Verlgeich mit der origniallen Matlab Implementierung ausgeführt werden

3.3 Manuelle Tests

Manuelle TEsts + PRofiling haben gezeig,t dass in den TEsts der Hotpath wie erwartet der Fall ohne Singularität ist. D.h. die Optimierung lohnt sich am meisten auf dem Hotpath! (Ambehls law oder wie hießt das?)

3.4 Nicht berücksichtigte Optimierungen

- Verschiedene Compiler
- GPU beschleunigung: Numerische integrationsverfahren nicht auf GPU implementiert

3.5 Architekturmodelle

4

Entwurf

4.1 Nichtfunktionale Anforderungen

- Geschwindigkeit
- Genauigkeit
- Bedienbarkeit

4.2 Anforderungen

Trivial: Implementierung des Algorithmus

4.3 Architektur

- Iteratives Design
- In Tests hat sich gezeigt, dass zu viel Design zu schlechtere Performance führt (z.B. nutzen von Parameterobjekten oder zu viel indirektion für besser lesbaren code)
- Functor Objekte erläutern
- Numerische Integraiton mit GSL muss noch gelöst werden

4.4 Multithreading betrachtung des Algorithmus

Was kann man wo Parallelisieren?

4.5 Thread pool bzw. object pool

5

Implementierung

5.1 Verwendete Technologien

Mit Stern markierte gerne etwas weiter ausführen

- Armadillo
- GNU Scientific Library *
- Pybind (Eigen3) *
- OneTBB *
- Catch2
- Google Benchmarks
- CMake
- Visualstudio Profiler
- Gprof (malsehen ob das noch genutzt wird)
- Valgrind*

  Nicht genutzte Alternativen
  Gründe noch aufführen

- SIMD intrinsics * (mal sehen ob das sinnvoll ist)
- Boost numeric
- OpenCL
- OpenMP
- CUDA

5.1.1 GNU Scientific Library

Gerade bei der Frage nach dem Framework für Linalg und numerischer Integration lässt sich nicht abschätzen was die beste Lösung ist. Es gibt so viele Frameworks und Bibliotheken, dass es nicht möglich ist alle gegeneinander Abzwuwägen.

Die Entshceidung GSL und Armadillo basier auf: Aramdillo ist einfach einzubinden und unkompliziert in der Anwendung. GSL hat eine gute Perfomance. Der Verglecih lief mit Boost wobei einfache Tests zeigten das GSL schneller ist und mit weitaus weniger Aufwand in das Projekt integriert werden kann. (Boost integration ist furchtbar!)

5.1.2 Armadillo

Kurze Einführng (1 Seite max?)

5.1.3 Intel Threading Building Blocks

Darstellung der beschleunigten Stellen, bzw beispiele wie man damit was parallelsiert + kurze einführenng

5.1.4 Valgrind

Erklärung der genutzten Tools Cachegrind und Cachetool?

5.1.5 Pybind

Kurze erklärung wie es klappt und beispiel für einfaches Plugin

5.2 Ausgewählte Codestellen

5.2.1 Gauss laguerre integration und Cauchy Integral Theoerem

```
1  auto calculate_integral_cauchy(const path_utils::path_function
       first_path, const path_utils::path_function second_path, std::
       vector<double> nodes, std::vector<double> weights)->std::
       complex<double>
2  {
3      std::vector<std::complex<double>> eval_points1, eval_points2,
       eval_points_summed;
4      // Evaluierung entlang der Pfade
5      std::transform(nodes.begin(), nodes.end(), std::back_inserter(
       eval_points1), first_path);
6      std::transform(nodes.begin(), nodes.end(), std::back_inserter(
       eval_points2), second_path);
7      // Summierung
8      std::transform(eval_points1.begin(), eval_points1.end(),
       eval_points2.begin(), std::back_inserter(eval_points_summed),
       [](const auto left, const auto right) -> auto {
9          return left - right;
10         });
11     // Gewichtung
12     return std::inner_product(weights.begin(), weights.end(),
       eval_points_summed.begin(), 0. + 0.i);
13 }
```

5.2.2 1D Integration

5.2.3 2D Integration

Warum konnte 1d Integration hier nicht direkt genutzt werden? =¿ Indirektion durch das 1d Integral war relativ hoch, das teste ich aber besser noch mal

6

Bewertung

6.1 Vergleiche der Laufzeiten

7

Zusammenfassung und Ausblick

7.1 Schwierigkeiten

Komplizierte Mathematik dahinter, speziell die Auswahl der Punkte um die Splitting Points konnte ich nicht nachvollziehen. (Keine explizierte Erklärung im Paper, dafür aber im MatlabCode)

7.2 Weitere Möglichkeiten das Verfahren zu beschleunigen

mögliche beschlenigungen die nicht bearbeitet wurden:

- Pipelinebasierte Architektur (siehe TBB dokumente!)
- GPU beschleunigung: steht und fällt mit numerischer Integration auf GPU (Eigene implementierung von Numerischer Integration würde viele Möglichkeiten bieten)
- Templated/Constexpression zeug: Je nach Szenario könnte eine Compiletime optimierte Anwendung bereitgestellt werden. So könnten zum einen Parameter wie die Wellenfrequenz k bereits in das Programm kompliiert werden un so zu einer enomren beschleunigugn beitragen.
- Templated/Constexpression 2: Für fixe Node größen der Laguerre Implementeirung könnt eine gezielte festgelegte Implementierung durch eine fixe FUnktion zur Kompilezeit ausgetauscht werden und so für eine weitere beschleunigung sorgen?

In diesem Kapitel soll die Arbeit noch einmal kurz zusammengefasst werden. Insbesondere sollen die wesentlichen Ergebnisse Ihrer Arbeit herausgehoben werden. Erfahrungen, die z.B. Benutzer mit der Mensch-Maschine-Schnittstelle gemacht haben oder Ergebnisse von Leistungsmessungen sollen an dieser Stelle präsentiert werden. Sie können in diesem Kapitel auch die Ergebnisse oder das Arbeitsumfeld Ihrer Arbeit kritisch bewerten. Wünschenswerte Erweiterungen sollen als Hinweise auf weiterführende Arbeiten erwähnt werden.

Literatur

Gasperini, David and Beise, Hans-Peter and Schroeder, Udo and Antoine, Xavier and Geuzaine, Christophe (2022). „An analysis of the steepest descent method to efficiently compute the 3D acoustic single-layer operator in the high-frequency regime“. In: *IMA Journal of Numerical Analysis*. URL: `https://hal.archives-ouvertes.fr/hal-03209144`.

A

Glossar

B

Selbstständigkeitserklärung

☐ Diese Arbeit habe ich selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

☐ Diese Arbeit wurde als Gruppenarbeit angefertigt. Meinen Anteil habe ich selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Namen der Mitverfasser:

Meine eigene Leistung ist:

Datum

Unterschrift der Kandidatin/des Kandidaten