

# **GPU Optimization of Base Form 820 Collective Artificial Spouse Retirement Plan**

Niklas Schalck Johansson, Nikolaj Aaes and Hildur Flemberg  
`{nsjo, niaa, hufl}@itu.dk`

IT University of Copenhagen

## Table of Contents

1	Introduction.....	3
2	Background .....	3
	2.1 Problem definition.....	3
	2.2 Scope .....	3
	2.3 Assumptions .....	3
3	The Math.....	4
	3.1 Outer model .....	4
	3.2 Middle model .....	5
	3.3 Inner model .....	5
	3.4 Runge-Kutta 4th order .....	5
4	Implementation .....	6
5	CUDA.....	6
	5.1 CUDA Architecture .....	6
	5.2 Parralellisation .....	6
6	Testing .....	8
7	Benchmarks and Comparison .....	8
8	Reflection / Discussion .....	8
9	Future work .....	9
10	Conclusion .....	9
11	Glossary .....	9

**Abstract.** Abstract is the executive summary of the entire paper. As a summary it should encompass not only the main result, but also the problem, and the evaluation. Shaw [4] reports that good ICSE papers discussed evaluation already in their abstract. This is important in software engineering, because of the emphasis put on the evaluation of results. A good abstract is readable for non-experts, as it increases the chances that someone will build on your work (often application opportunities appear in other areas). I try to have four sentences in my abstract. The first states the problem. The second states why the problem is a problem. The third is my startling sentence. The fourth states the implication of my startling sentence.

Simon Peyton Jones summarizes this idea in the following points  
Statetheproblem Saywhyit’saninterestingproblem Saywhatyoursolution-  
achieves Saywhatfollowsfromyoursolution

**Keywords:** CUDA, parallelism, threads, blocks, insurance, capital, software, programming languages, Runge Kutta 4th order

## 1 Introduction

- Why is the project relevant, what is the problem at hand  
Scope? [2]

Hvad regner vi med at læseren har af kompetencer

This report is written at the IT University of Copenhagen in the spring term of 2013 in connection with a CUDA project supervised by Peter Sestoft from ITU and Hans Henrik Brandenborg Sørensen from DTU. The report is addressed to people interested in exploiting the benefits of using graphical processing units (GPUs) to optimise feasible computations in general, but specifically to people interested in CUDA.

In this project, we take a problem from the insurance industry that requires a significant amount of time to solve and show how to speed it up by moving the computations from the CPU to the GPU.

## 2 Background

- Related work, projects that precedes this project, why do we use the technology described, what is this new technology - Nævn at vi har fået udleveret matematik  
beforehand

### 2.1 Problem definition

How can CUDA C be utilized to optimize the calculation of the baseform 820 collective artificial spouse retirement plan using the Runge Kutta 4th order method?

### 2.2 Scope

The scope of this project is to handle a single customer at a time. We do not explore the possibilities for optimizing the process of using the algorithm on several customers at a time. Neither do we explore alternative methods for approximating differential equations.

### 2.3 Assumptions

This project was developed under several assumptions to limit the size of it. We assumed that the spouse of an insurance holder was always of the opposite gender. It would now be possible to take same-gender couples into account but it would require some modification to the code. We assumed that the function to determine the interest rate always returns 5 %.

### 3 The Math

The algorithm in this project is used to determine the lump sum of money the insurance company needs to possess to be able to pay the insurance holder's spouse in the case of his or her death. The payment to the spouse will be in the form of a life interest which are disbursed from the time of death of the insurance holder unless the death occurs before the pension age. In that case the money will be disbursed after a grace period determined by the insurance company at the time the insurance was taken out.

We assume that the spouse of the insurance holder is of the opposite gender. If the insurance holder does not have a spouse at the time of his or her death the insurance is forfeited.

The algorithm used here is a 4th order Runge-Kutta solution with a fixed step-size (indsæt reference) where we use a series of constants determined before any calculation begins:

Constants	
Name	Meaning
$\tau$	The time of death of the insurance holder
$r$	The pension age
$g$	The grace period
$x$	The age of the insurance holder at calculation time ( $t = 0$ )
$t$	The time of calculation
$h$	The Stepsize of the Runge-Kutta solution

$\tau$  is expressed as  $x + t$ . Apart from this there is also a constant  $k$  which are determined by  $\tau, r$  and  $g$  in the following manner:

How $k$ is defined	
If this statement holds	then $k$ equals
$\tau < r$	$g$
$r \leq \tau < r + g$	$r + g - \tau$
$r + g \leq \tau$	0

The algorithm can be described as a combination of three models; an outer model that describes the life/death state of the insurance holder, a middle model that describes the married/unmarried state of the insurance holder and an inner model that describes the life/death state of the potential spouse.

NOTE: The description in the three model sections is loosely based on a description for Edlund[1] and some passages are directly translated from this document.

#### 3.1 Outer model

The outer model is expressed as the following equation:

$$\frac{d}{dt}f(t) = r(t)f(t) - \mu_t(x+t)(S_{x+t}^d(t) - f(t))$$

Where  $S_{\tau}^d$  is the DØDSFALDSSUM that for a  $\tau$  year old at the time  $t$  is needed to cover the payment from the insurance company,  $\mu_t(x+t)$  is the mortality rate for a  $(x+t)$  year old and  $r(t)$  is the interest rate function. In this project the interest rate function returns the constant 0.05.

The differential equation is solved from  $t = 120-x$  to  $t = 0$  with the boundary condition  $f(120-x) = 0$

### 3.2 Middle model

The middle model is used to calculate  $S_{x+t}^d(t)$  and is expressed with the following equation:

$$S_{\tau}^d(t) = \begin{cases} g_{\tau} \int f(\eta|\tau) a_{[\eta]+g}^I(t) d\eta & \tau \leq r \\ g_{\tau} \int f(\eta|\tau) a_{[\eta]+r+g-\tau}^I(t) d\eta & r \leq \tau \leq r+g \\ g_{\tau} \int f(\eta|\tau) a_{[\eta]}^I(t) d\eta & r+g \leq \tau \end{cases}$$

Where  $g_{\tau}$  is the propability that a  $\tau$  year is married and  $f(\eta|\tau)$  is the probability distribution for, that a  $\tau$  year old is married to a  $\eta$  year provided that the  $\tau$  year old is married. This equation can be rewritten to this form:

$$\frac{d}{dn}f(\eta) = -g_{\tau}f(\eta|\tau)a_{[\eta]+k}^I(t)$$

here the DET-DER-TAL-SOM-LIGNER-EN-EKSPONENT-MEN-ER-NEDE-I-STEDET-FOR-OPPE for  $a$  is rewritten to  $[\eta] + k$  where  $k$  can fall into three different categories as shown in THE-TABLE-CONSTANTS.

This differential equation is solved from  $\eta = 120$  to  $\eta = 1$  with the boundary condition  $f(120) = 0$

### 3.3 Inner model

The inner model is used to calculate  $a_{[\eta]+k}^I(t)$  and is expressed with the following equation:

$$\frac{d}{ds}f(s) = r(t+s)f(s) - 1_{s \geq k} - \mu_{t+s}(\eta+s)(0 - f(s))$$

Where  $t$ ,  $\eta$  and  $k$  are constants and  $\mu_{t+s}(\eta+s)$  is the mortality rate for a  $(\eta+s)$  year old.

This differential equation is solved from  $s = 120 - \eta$  to  $s = 0$  with the boundary condition  $f(120 - \eta) = 0$

### 3.4 Runge-Kutta 4th order

The Runge-Kutta method is a method for approximating differential equations that builds on Eulers method (INDSÆT REFERENCE) and the midpoint method

(INDSÆT REFERENCE). When given a start point and a differential equation one can choose a stepsize and approximate the graph. When given a point  $(x_n, y_n)$ , a differential equation  $f$  and a stepsize  $h$  one can use the Runge-Kutta method to approximate the next point in the following way:

$$\begin{aligned} k1 &= hf(x_n, y_n) \\ k2 &= hf(x_n + \frac{h}{2}, y_n + \frac{k1}{h}) \\ k3 &= hf(x_n + \frac{h}{2}, y_n + \frac{k2}{h}) \\ k4 &= hf(x_n + h, y_n + k3) \end{aligned}$$

$$\begin{aligned} x_{n+1} &= x_n + h \\ y_{n+1} &= y_n + \frac{k1}{6} + \frac{k2}{3} + \frac{k3}{3} + \frac{k4}{6} + O(h^5) \text{ HVORFOR KAN VI IGNORERE } O(h^5)?? \end{aligned}$$

BESKRIV HVORDAN EN NORMAL KØRSEL SERUD = EKSEMPEL kig evt på en simpson integration når vi kigger på middle da det egentlig ikke er en differential ligning men et integrale

## 4 Implementation

- A description of the new implementation and design choices I den kode der er nu er insurance holder kvinde og spouse mand ALTID

## 5 CUDA

en tråd skal lave et vist stykke arbejde før det kan betale sig at oprette den. Der er overhead for at lave context for hver tråd så hvis den arbejde tråden laver er mindre end at sætte overheaden op, giver det ingen mening.

- A more elaborate description of GPGPUs, architecture and CUDA C.

### 5.1 CUDA Architecture

### 5.2 Parallellisation

De forskellige parallelliserings muligheder: - beskriv teorien - beskriv hvad vi får ud af det i køretid? - beskriv hvorfor vi har/ikke har valgt denne løsning - Kan den kombineres med de andre for bedre performance?

Outer Par

Da man på forhånd kan beregne alle x værdier outer skal bruge i sin kørsel og alle Middle(x) kald i outerDiff kun skal bruge en x værdi er det muligt at regne alle disse middle kald i parallel og gemme hvert middle kalds resultat i et array. Når man derefter skal gå fra startpunkt til næste punkt kan man betragte Middle(p.x) i  $returnr(px) * py - GmFemale(x + px) * (Middle(px).y - py)$ ; som en udregning der tager konstant tid.

### Middle Par

Da man på forhånd kan beregne alle  $x$  værdier middle skal bruge i sin kørsel og vi ved at hvert step i Middle har udregninger til  $k_1, k_2/k_3$  og  $k_4$  i hvert skridt der kun afhænger af  $x$ -værdien kan vi regne alle de sæt af  $k_1, k_2/k_3$  og  $k_4$  som middle skal bruge i parallel. Derefter kan de lægges sammen som et vægtet gennemsnit i et array og bruger når man vil gå fra et punkt til et nyt. Kaldet  $doubley = p.y + k_1/6 + k_2/3 + k_3/3 + k_4/6$ ; kan altså betragtes som konstant. Her udregnes hvert sæt af  $k_1, k_2/k_3$  og  $k_4$  dog i en enkelt tråd. Det vil sige at der er en tråd for hvert step.

### Inner Par

Man har også muligheden for at parallelisere middles kald til inner der bruges i udregningen af  $k_1, k_2/k_3$  og  $k_4$ . Ved at sætte 3 tråde til at udregne de tre værdier kan dette gå tre gange så hurtigt.

### Kombination af Outer Par og Middle par

Ved en valgt block size  $b = \text{antal outersteps}$  og en thread per block  $tpb$  kan man sætte hver block til at tage sig af en  $x$ -værdi i outer, altså tage sig af et enkelt middle kald. Hvis man forestiller sig at middle skal tage  $ms$  steps og  $ms > tpb$ , vil man først udregne  $k_1, k_2/k_3$  og  $k_4$  sæt for de først  $tpb$  steps og derefter gå videre til de til næste  $tpb$  steps indtil alle steps er udregnet. Det betyder også at der kan være inaktive tråde i den sidste omgang  $tpb$  steps og det kan give divergence. Det er muligt at bruge reduction (se cuda bogen) når alle  $k$  er blevet regnet ud og man skal lægge  $p.y$  til.

### Kombination af Middle par og Inner par

Ligesom i den ovenstående løsning lader man hver block handle en  $x$ -værdi for middle og giv hver block tre tråde hver af disse tråde udregner hver sin  $k$  værdi og hver tredje tråd lægger dem sammen.

### Kombination af alle tre

Som i kombinationen af outer par og middle oar vælger man et antal blokke svarende til antallet af outersteps og en thread per block  $tpb$ . Man lader stadig hver blok tage sig af et enkelt kald til Middle med en bestemt  $x$ -værdi. Forskellen kommer når man i stedet for at lade hvert tråd udregne et sæt af  $k_1, k_2/k_3$  og  $k_4$  værdier sætter man i stedet 3 tråde til at udregne dem i parallel. Dette vil kun have indvirkning hvis block størrelsen er større end det antal steps middle skal udføre. Hvis ikke bruger man tre gange så mange tråde til at så udregnes først  $tpb$   $k$ -værdier, tre gange hurtigere end normal, dernæst de næste  $tpb$   $k$ -værdier, stadig tre gange så hurtigt men i stedet for at stoppe ved middlesteps som normal vil man stoppe ved  $\text{middlesteps} * 3$  da der skal bruges 3 gange så mange  $k$ -værdier. Optimalt set skal der altså være  $\text{middlesteps} * 3$  tråde i hvert block. Ved en vilkårlig stepsize vil det være  $120 * \text{stepsize} * 3$  tråde. Med en stepsize på 1 eller to giver det henholdsvis 360 og 720 tråde hvilket stadig er inden for det 1024-tråde limit hver block har. Men allerede ved stepsize 3 giver det 1080 tråde

per block hvilket ikke er legalt.

Hvad har vi valgt

Vi har valgt Kombination af Outer Par og Middle Par for simpelheden. Kombination af alle tre er ikke smart fordi det kun giver optimering når threads per block er  $\text{middlesteps} \times 3$  eller over hvilket ikke er legalt med en stepsize over 2. SKRIV NOGET OM HVORFOR VI HAR VALGT KOMBI AF OUTER OG MIDDLE I STEDET FOR MIDDLE OG INNER. HVORFOR ER DET MERE EFFEKTIVIT?

## 6 Testing

test på forskellige maskiner hvis vi deler steps op i blocks af 32 tråde kan der i worst case scenario være en sidste block med 1 aktiv og 31 inaktive tråde. Lav en graf over rest trådene

- A test to ensure that the new implementation produces the same output as the current implementation

lige nu er insurance holder altid en kvinde og spouse altid en mand

## 7 Benchmarks and Comparison

skriv noget om memory bound og compute bound i rapporten - Benchmarking and comparison on speed between the current and new implementation

## 8 Reflection / Discussion

improvements: det kan optimeres til at kunne køre flere kunder. Simpson løsning i middle?

KUNNE DET VÆRE INTERESSANT AT BRUGE DYNAMIC PROGRAMMING? KUNNE MAN GEMME HVILKE KALD TIL INNER DER ER LAVET OG HVIS INNER SÅ KALDES IGEN MED SAMME VÆRDIER, KAN MAN SLIPPE FOR AT REGNE DET UD OG BARE HENTE RESULTATET. DOG SKAL MAN VÆRE OPMÆRKSOM PÅ AT MAN KAN KOMME I EN SITUATION HVOR EN BEREGNING SER AT DER IKKE ER NOGET RESULTAT FOR ET SÆT PARAMETRE OG DERFOR GÅR I GANG MED AT REGNE DET UD, MENS DEN FØRST BEREGNING KØRER KAN ANDRE BEREGNINGER MED SAMME PARAMETRE OG DE NYE BEREGNINGER VIL OGSÅ GERNE GEMME RESULTATET NÅR DE ER FÆRDIGE HVILKET KAN GIVE CONFLICTS (MÅSKE). eller Det kan være vi skal dokumentere en del om at man kan gemme alle resultater i en tabel og efter et vist antal kunder kan man ende med nok til bare at have opslag i stedet for beregninger. skal denne tabel et ligge på GPU så den kan slås op i konstant tid.



skriv evt noget om autotuning, hvor programmet først tester med forskellige mængder af tråde og blokke og vælger den hurtigste løsning.

Threats to validity??? kan det betale sig at køre dele af programmet på CPU'en?

- Did we achieve what we wanted? what did we discover during the project?  
What can be changed in future implementations?

## 9 Future work

Future work:

optimize for several customers at a single time

Take the interest curve into account

## 10 Conclusion

Conclusion

## 11 Glossary

- Forsikringstager - insurance holder
- ægtefælle - Spouse
- livrente - life interest
- dødfaldssum
- pause periode - grace period
- randbetingelse - boundary condition

## References

- [1] EDLUND DOKUMENTET
- [2] David B. Kirk, Wen-mei W. Hwu - Programming Massively Parallel Processors, A Hands-on Approach - Elsevier Inc. - 2010