

# Exploring the limits of mixed precision FEM based computations on the Tegra-K1 micro-architecture

Christoph Höppke, Daniel Tomaschewski

TU Dortmund

Date: 2016/06/01

# Content

---

## 1 Mixed precision definition and history overview

- History overview
- Definition

## 2 Floating point operations

- Precision
- Computational precision vs accuracy of result
- Floating point operations. A deeper analysis

## 3 Example calculation

- Algorithm
- Problem definition
- Testresults

# Content

---

## 1 Mixed precision definition and history overview

- History overview
- Definition

## 2 Floating point operations

- Precision
- Computational precision vs accuracy of result
- Floating point operations. A deeper analysis

## 3 Example calculation

- Algorithm
- Problem definition
- Testresults

# A brief history overview

---

## Trends:

- Memory clock speeds are increasing
  - GTX 980 Ti Memory clock speed: 2x 1753 MHz
  - GTX 1080 Memory clock speeds: 4x 2500 MHz (+185%)
- Alternative computing architectures
  - APU's
  - SoC's such as the NVIDIA Tegra K1 micro-architecture
- NVIDIA pushing the use of half precision.
  - half and half2 were announced as important new features in the CUDA Toolkit version 7.5 [1]

→ **Sharing memory between CPU and GPU is becoming easier and more common**

# A brief history overview

---

## Trends:

- Memory clock speeds are increasing
  - GTX 980 Ti Memory clock speed: 2x 1753 MHz
  - GTX 1080 Memory clock speeds: 4x 2500 MHz (+185%)
- Alternative computing architectures
  - APU's
  - SoC's such as the NVIDIA Tegra K1 micro-architecture
- NVIDIA pushing the use of half precision.
  - half and half2 were announced as important new features in the CUDA Toolkit version 7.5 [1]

→ Sharing memory between CPU and GPU is becoming easier and more common

# A brief history overview

---

## Trends:

- Memory clock speeds are increasing
  - GTX 980 Ti Memory clock speed: 2x 1753 MHz
  - GTX 1080 Memory clock speeds: 4x 2500 MHz (+185%)
- Alternative computing architectures
  - APU's
  - SoC's such as the NVIDIA Tegra K1 micro-architecture
- NVIDIA pushing the use of half precision.
  - half and half2 were announced as important new features in the CUDA Toolkit version 7.5 [1]

→ **Sharing memory between CPU and GPU is becoming easier and more common**

# Definition

---

## Definition: Mixed precision algorithm

An algorithm that uses different precisions in its computation

### Goal:

Obtain the **same accuracy** by using high precision  
but **better performance** by utilizing low precision computations

### Performance gains for bandwidth bound algorithms

- 64 bit = 1 double = 2 floats = 4 halves
- More variables per **bandwidth** and variables per **storage**
- Applies to all memory levels: network, disc, cache, main, device, local, register

### Performance gains for computation bound algorithms

- 1 double addition  $\approx$  2 float additions  $\approx$  4 half additions (linear)
- 1 double multip.  $\approx$  4 float multip.  $\approx$  16 half multip. (quadratic)
- $\rightarrow$  up to 16 times better computational efficiency

# Definition

---

## Definition: Mixed precision algorithm

An algorithm that uses different precisions in its computation

### Goal:

Obtain the **same accuracy** by using high precision  
but **better performance** by utilizing low precision computations

### Performance gains for bandwidth bound algorithms

- 64 bit = 1 double = 2 floats = 4 halves
- More variables per **bandwidth** and variables per **storage**
- Applies to all memory levels: network, disc, cache, main, device, local, register

### Performance gains for computation bound algorithms

- 1 double addition  $\approx$  2 float additions  $\approx$  4 half additions (linear)
- 1 double multip.  $\approx$  4 float multip.  $\approx$  16 half multip. (quadratic)
- $\rightarrow$  up to 16 times better computational efficiency



# Definition

---

## Definition: Mixed precision algorithm

An algorithm that uses different precisions in its computation

### Goal:

Obtain the **same accuracy** by using high precision  
but **better performance** by utilizing low precision computations

### Performance gains for bandwidth bound algorithms

- 64 bit = 1 double = 2 floats = 4 halves
- More variables per **bandwidth** and variables per **storage**
- Applies to all memory levels: network, disc, cache, main, device, local, register

### Performance gains for computation bound algorithms

- 1 double addition  $\approx$  2 float additions  $\approx$  4 half additions (linear)
- 1 double multip.  $\approx$  4 float multip.  $\approx$  16 half multip. (quadratic)
- $\rightarrow$  up to 16 times better computational efficiency

# Definition

---

## Definition: Mixed precision algorithm

An algorithm that uses different precisions in its computation

### Goal:

Obtain the **same accuracy** by using high precision  
but **better performance** by utilizing low precision computations

### Performance gains for bandwidth bound algorithms

- 64 bit = 1 double = 2 floats = 4 halves
- More variables per **bandwidth** and variables per **storage**
- Applies to all memory levels: network, disc, cache, main, device, local, register

### Performance gains for computation bound algorithms

- 1 double addition  $\approx$  2 float additions  $\approx$  4 half additions (linear)
- 1 double multip.  $\approx$  4 float multip.  $\approx$  16 half multip. (quadratic)
- $\rightarrow$  up to 16 times better computational efficiency

# Challenges and past achievements

---

## Challenges when it comes to Mixed precision

- Data has to be converted
- When using GPU acceleration data has to be transferred from the host to the device (usually) over the relatively slow PCIe bus
- We are being bottlenecked by memory interfaces  
→ a lot of time is wasted while waiting for data

## Past achievements:

In the year 2013, M. Madsen, S. L. Glimberg and A. P. Engsig-Karup were able to achieve a **38% performance increase** using GPU accelerated mixed precision algorithms for full nonlinear water wave computation.[2]

# Challenges and past achievements

---

## Challenges when it comes to Mixed precision

- Data has to be converted
- When using GPU acceleration data has to be transferred from the host to the device (usually) over the relatively slow PCIe bus
- We are being bottlenecked by memory interfaces  
→ a lot of time is wasted while waiting for data

## Past achievements:

In the year 2013, M. Madsen, S. L. Glimberg and A. P. Engsig-Karup were able to achieve a **38% performance increase** using GPU accelerated mixed precision algorithms for full nonlinear water wave computation.[2]

# Content

---

## 1 Mixed precision definition and history overview

- History overview
- Definition

## 2 Floating point operations

- Precision
- Computational precision vs accuracy of result
- Floating point operations. A deeper analysis

## 3 Example calculation

- Algorithm
- Problem definition
- Testresults

# Roundoff and Cancellation

## Definition Machine Precision

The smallest positive number  $\epsilon$  for which a floating point calculation evaluates the expression  $1 + \epsilon > 1$  to be true.

### Examples:

- $\epsilon_{double} \approx 2.220446049250313 \cdot 10^{-16}$
- $\epsilon_{float} \approx 1.1920929 \cdot 10^{-7}$
- $\epsilon_{half} \approx 9.765625 \cdot 10^{-4}$
- So **more precision** is usually **better**

### Examples for cancellation effects in float

additive roundoff	$a = 1 + 0.00000004 = 1.00000004$	$=_{fl} 1$
multiplicative roundoff	$b = 1.0002 \cdot 0.9998 = 0.99999996$	$=_{fl} 1$
cancellation	$c \in \{a, b\} \quad \pm 4 = (c - 1) \cdot 10^8$	$=_{fl} 0$
order of operations	$1 + 0.00000004 - 1 =_{fl} 0$	
	$1 - 1 + 0.00000004 =_{fl} 0.00000004$	

# Roundoff and Cancellation

## Definition Machine Precision

The smallest positive number  $\epsilon$  for which a floating point calculation evaluates the expression  $1 + \epsilon > 1$  to be true.

### Examples:

- $\epsilon_{double} \approx 2.220446049250313 \cdot 10^{-16}$
- $\epsilon_{float} \approx 1.1920929 \cdot 10^{-7}$
- $\epsilon_{half} \approx 9.765625 \cdot 10^{-4}$
- So **more precision** is usually **better**

### Examples for cancellation effects in float

additive roundoff       $a = 1 + 0.00000004 = 1.00000004 =_{fl} 1$

multiplicative roundoff       $b = 1.0002 \cdot 0.9998 = 0.99999996 =_{fl} 1$

cancellation       $c \in \{a, b\} \quad \pm 4 = (c - 1) \cdot 10^8 =_{fl} 0$

order of operations       $1 + 0.00000004 - 1 =_{fl} 0$   
 $1 - 1 + 0.00000004 =_{fl} 0.00000004$

# Roundoff and Cancellation

## Definition Machine Precision

The smallest positive number  $\epsilon$  for which a floating point calculation evaluates the expression  $1 + \epsilon > 1$  to be true.

### Examples:

- $\epsilon_{double} \approx 2.220446049250313 \cdot 10^{-16}$
- $\epsilon_{float} \approx 1.1920929 \cdot 10^{-7}$
- $\epsilon_{half} \approx 9.765625 \cdot 10^{-4}$
- So **more precision** is usually **better**

### Examples for cancellation effects in float

additive roundoff	$a = 1 + 0.00000004 = 1.00000004$	$=_{fl} 1$
multiplicative roundoff	$b = 1.0002 \cdot 0.9998 = 0.99999996$	$=_{fl} 1$
cancellation	$c \in \{a, b\} \quad \pm 4 = (c - 1) \cdot 10^8$	$=_{fl} 0$
order of operations	$1 + 0.00000004 - 1 =_{fl} 0$	
	$1 - 1 + 0.00000004 =_{fl} 0.00000004$	



# Computational precision vs accuracy of result

---

Instructive Example [S.M. Rump, 1988]

$$f(x, y) = (333.75 - x^2)y^6 + x^2(11x^2y^2 - 121y^4 - 2) + 5.5y^8 + 0.5x/y$$
$$x_0 = 77617, y_0 = 33096$$

float s23e8

1.1726

double s52e11

1.17260394005318

The correct result is:

-0.82739605994682136814116509547981629

# Computational precision vs accuracy of result

---

Instructive Example [S.M. Rump, 1988]

$$f(x, y) = (333.75 - x^2)y^6 + x^2(11x^2y^2 - 121y^4 - 2) + 5.5y^8 + 0.5x/y$$
$$x_0 = 77617, y_0 = 33096$$

float s23e8

1.1726

double s52e11

1.17260394005318

**The correct result is:**

-0.82739605994682136814116509547981629

# Floting point operations. A deeper analysis

**Number representation** → almost all numbers have to be truncated

half s10e5 a	1 bit sign $s_a$	10 bit mantissa $m_a$	5 bit exp. $e_a$
float s23e8 b	1 bit sign $s_b$	23 bit mantissa $m_b$	8 bit exp. $e_b$
double s52e11 c	1 bit sign $s_c$	52 bit mantissa $m_c$	11 bit exp. $e_c$

<b>multiplication</b>		
precision	exact format	mantissa truncation:
half(a) · half(b)	s20e6	from 20 to 10 bit
float(a) · float(b)	s46e9	from 46 to 23 bit
double(a) · double(b)	s104e12	from 104 to 52 bit
<b>addition</b>		
precision	exact format	mantissa truncation:
half(a) + half(b)	s41e5	from 41 to 10 bit
float(a) + float(b)	s278e8	from 278 to 23 bit
double(a) + double(b)	s2099e11	from 2099 to 52 bit

# Content

---

## 1 Mixed precision definition and history overview

- History overview
- Definition

## 2 Floating point operations

- Precision
- Computational precision vs accuracy of result
- Floating point operations. A deeper analysis

## 3 Example calculation

- Algorithm
- Problem definition
- Testresults

# Algorithm

mixed precision iterative refinement

---

---

## Algorithm 1 mixed precision iterative refinement

---

```
1: while  $\|r_{m-1}\| \cdot \|r_0\|^{-1} > TOL$  do  
2:    $r_m^h = b - Ax_m$   
3:    $r_m^l = r_m^h$            //convert from high to low precision  
4:    $Ad_m^l = r_m^l$          //solve using a fixed number of iterations  
5:    $d_m^h = d_m^l$          //convert from low to high precision  
6:    $x_{m+1} = x_m + d_m$   
7: end while
```

---

- Mixed precision iterative refinement deals with the truncation errors of lower precision data formats
- The optimal number of inner iterations was empirically determined

# Problem definition and Hardware configuration

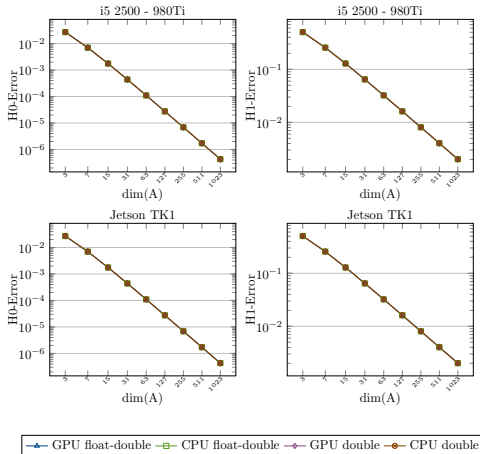
---

## Problem

Solve the Poisson problem  $-\Delta u = 1$ ,  $x \in \Omega$  with dirichlet boundary conditions  $u \equiv 0$ ,  $x \in \partial\Omega$  using conforming quadrilateral elements for the finite-element discretization of the unit square  $\Omega = (-1, 1)^2$

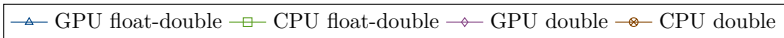
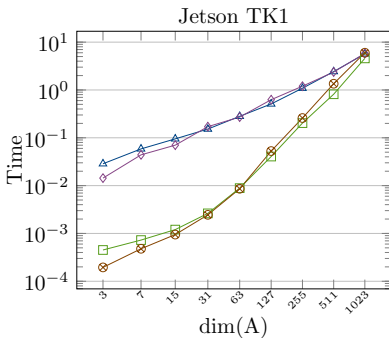
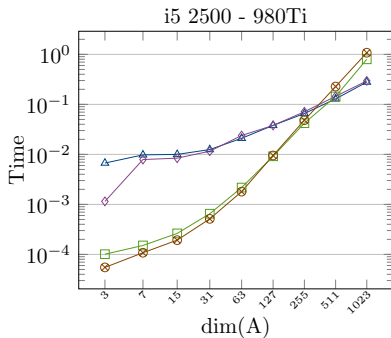
- Method 1: Using a V-cycle MG Solver
- Method 2 : Using a V-cycle MG Solver inside of an iterative refinement loop
- Hardware configuration 1: i5 2500 + 980Ti  
→ representing a high preformance convetional setup
- Hardware configuration 2: Tegra K1  
→ representing alternative computing hardware

# Testresults - Accuracy



- (virtually) same accuracy as double calculation
- H0-Error is  $O(m^2)$ , H1-Error is  $O(m^1)$ . [ $V_h = Q_1$ ]

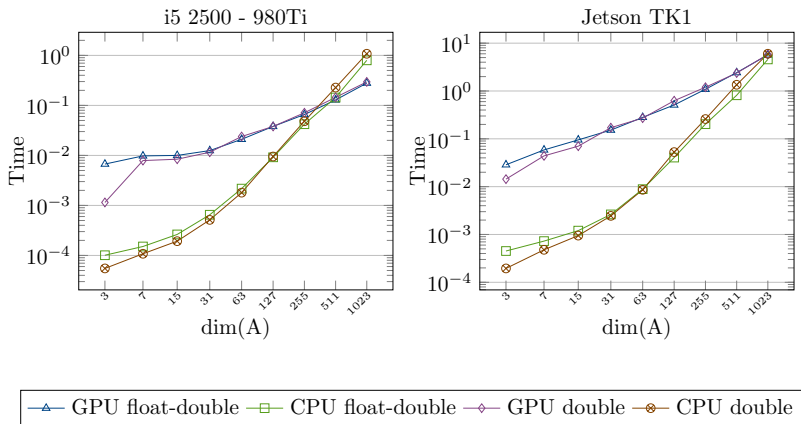
# Testresults - Performance on commodity Hardware



■ up to 57% increase in performance on x86

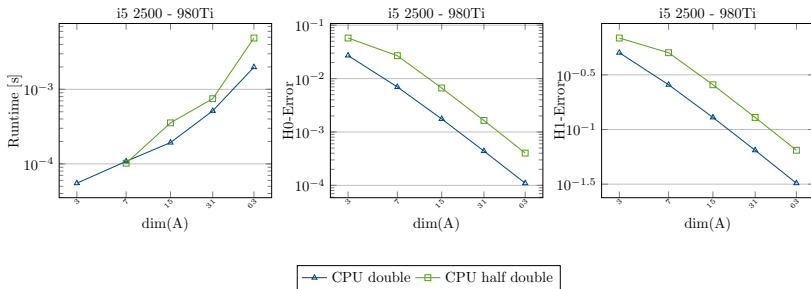


# Testresults - Performance on the Tegra K1



- **up to 65%** increase in performance on a Jetson TK1
- → **Speedup of about 1.6**
- → performance model show that a speedup of 2 is optimal
- → for longer calculations a speedup of 1.6 can save multiple days

# Testresults - Half



- Accuracy loss. Inner solver diverges at more than  $63^2$  DOF's.

# Discussion and Future work

---

## **Future work**

- Try combining half, float and double precision in order to achieve an even greater performance increase

Thank you for your attention!

# Bibliography

---



New features in cuda 7.5.

<https://devblogs.nvidia.com/parallelforall/new-features-cuda-7-5/>.

Accessed: 2016-05-27.



Madsen M. Glimberg S. L., Engsig-Karup A. P.

A fast gpu-accelecrated mixed-precision strategy for full nonlinear water wave computation.

Technical report, ResearchGate, January 2013.