Algebraic Multigrid Methods on Graphics Processing Units

Manfred Liebmann
Institute for Mathematics and Scientific Computing
University of Graz
manfred.liebmann@uni-graz.at

November 7, 2008

Overview

- General-Purpose Computing on Graphics Processing Units
- Parallelization Concepts on GPUs
- Algebraic Multigrid Solver on GPUs

(1) General-Purpose Computing on Graphics Processing Units

Todays graphics processors are multi-core processors with hundreds of processor cores!

- **GPU Hardware Architecture: Multiprocessor**
  - 8 thread processors + 16384 registers + 64KB shared memory
  - IEEE 754 single and double precision floating point support
  - 3 x 8 FP32 (MAD + MUL) + 2 x FP64 (MAD) per clock
  - Up to 1024 threads per multiprocessor
  - Up to 30 multiprocessors per graphics chip
  - Up to 4GB RAM per graphics board

- **NVIDIA GTX 280**
  - 1.4 billion transistors
  - 240 thread processor @ 1.4 GHz + 1 GB RAM
  - Memory bandwidth $\approx$ 120 GB/s
  - Peak performance: $\approx$ 1 TFLOPS (single) + $\approx$ 100 GFLOPS (double)
  - PCI Express Card: **350 EUR**

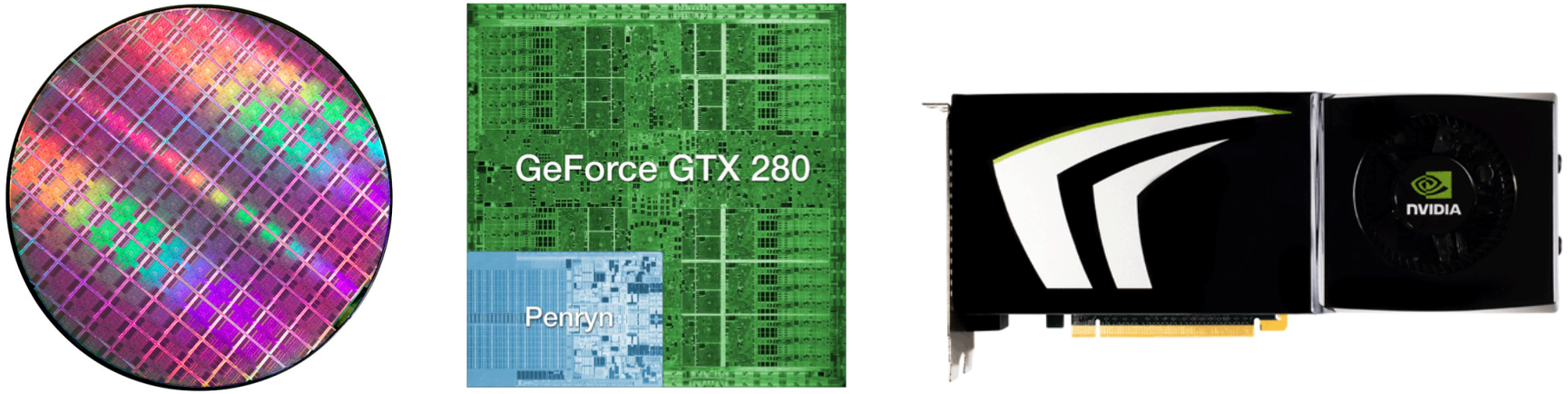


Figure 1: NVIDIA GTX 280

Floating-Point Performance

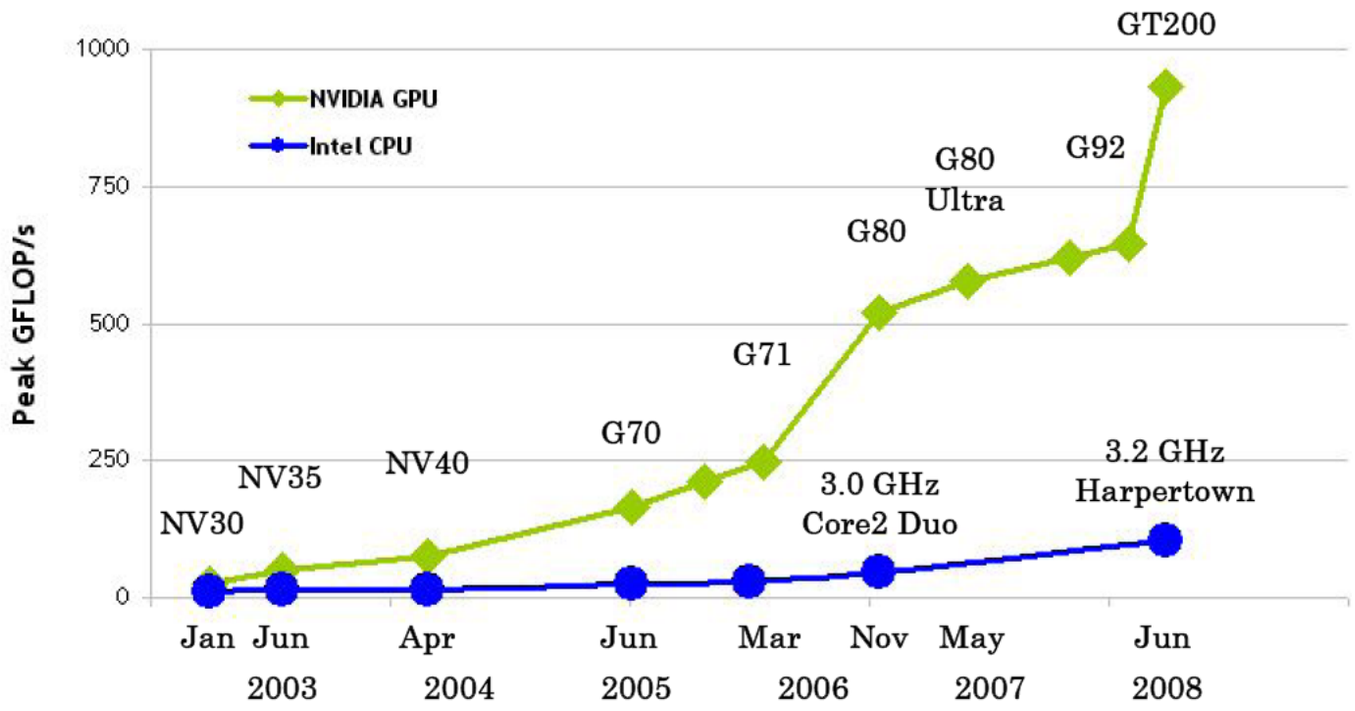


Figure 2: Floating-Point Operations per Second for the CPU and GPU

Memory Bandwidth

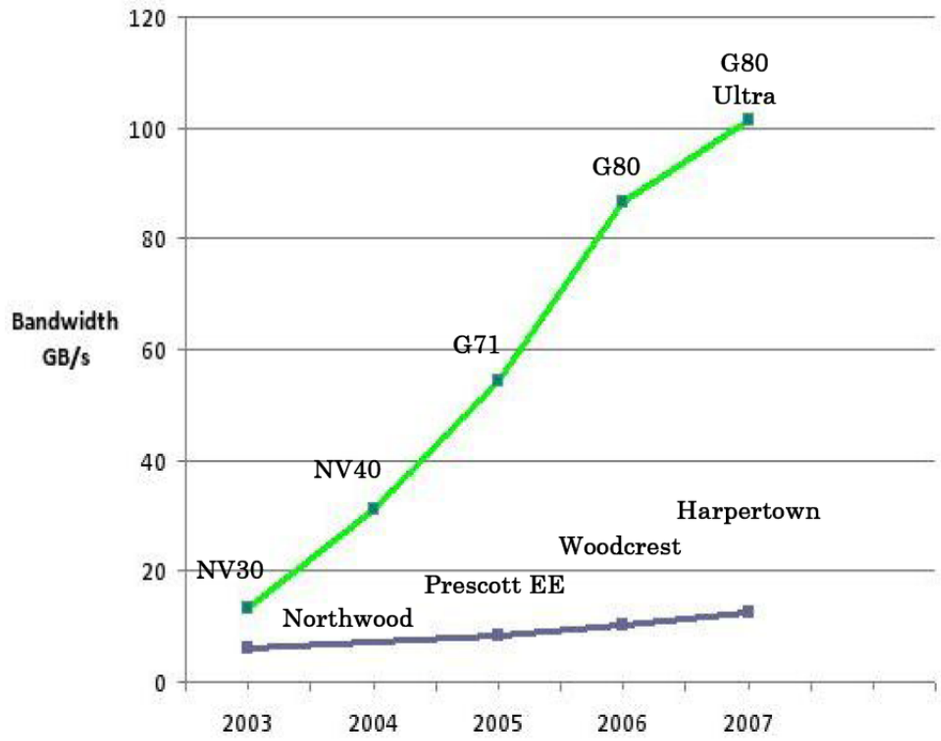


Figure 3: Memory Bandwidth for the CPU and GPU

Architecture Comparison

- Difference between GPU and CPU architecture
  - **GPU** has *no* coherent processor cache only small on-chip shared memory
  - Simple processor cores supporting many threads for latency hiding
  - **CPU** has big L1 + L2 processor caches
  - Complicated deeply pipelined processor to hide memory latency

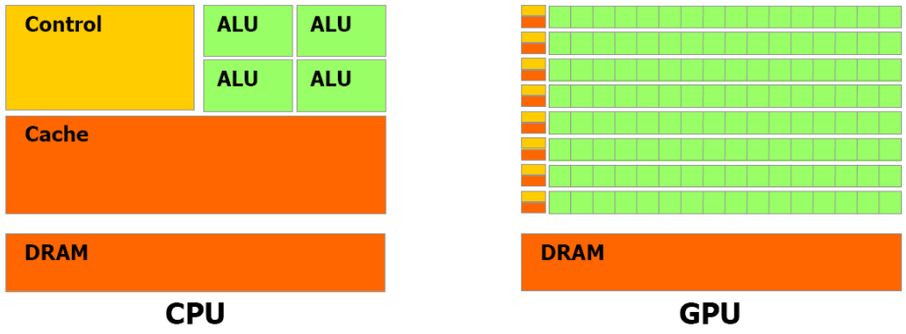


Figure 4: The GPU Devotes More Transistors to Data Processing

(2) Parallelization Concepts on GPUs

Problem: Reading data from main memory has 400-600 cycles latency.

Solution: *Schedule hundreds of threads to hide the latency!*

- **Grids and Thread Blocks**
  - Threads are organized in 1D/2D/3D **thread blocks** of up to 512 threads
  - Thread blocks are organized in a 1D/2D **grids**
  - Thread blocks are scheduled for execution on the multiprocessors
- **No global thread synchronization**
  - Synchronization only per multiprocessor i.e. thread block

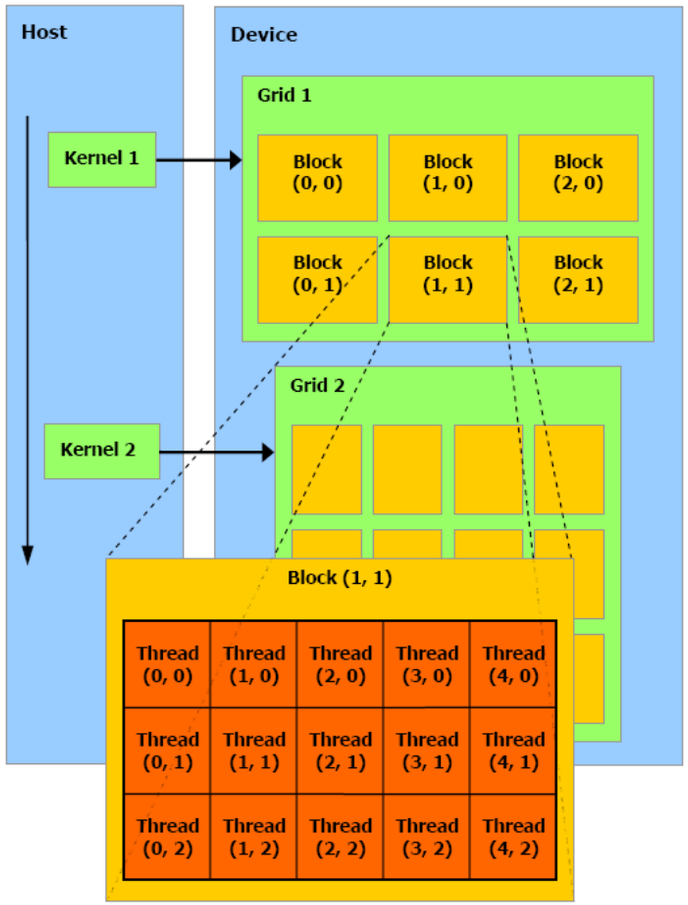


Figure 5: Thread Batching on the GPU

Data Caching

- **There is no coherent data cache on the GPU!**
  - Processor cores directly access the DRAM memory
  - Only a small 16KB read-only texture cache is available

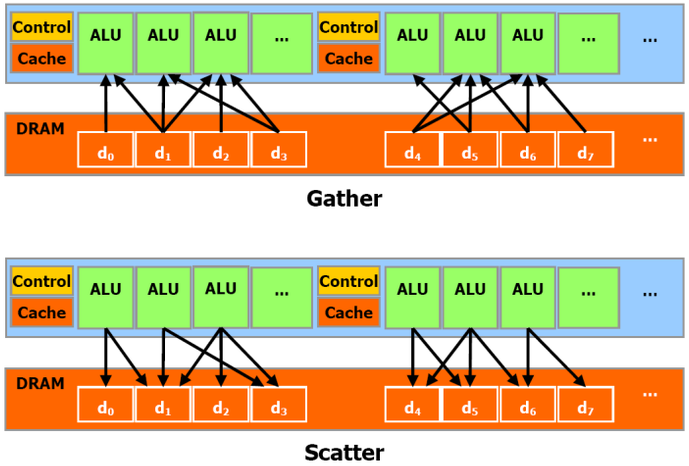


Figure 6: Memory Access Patterns

Shared Memory

- **Multiprocessors have a small on-chip memory region**

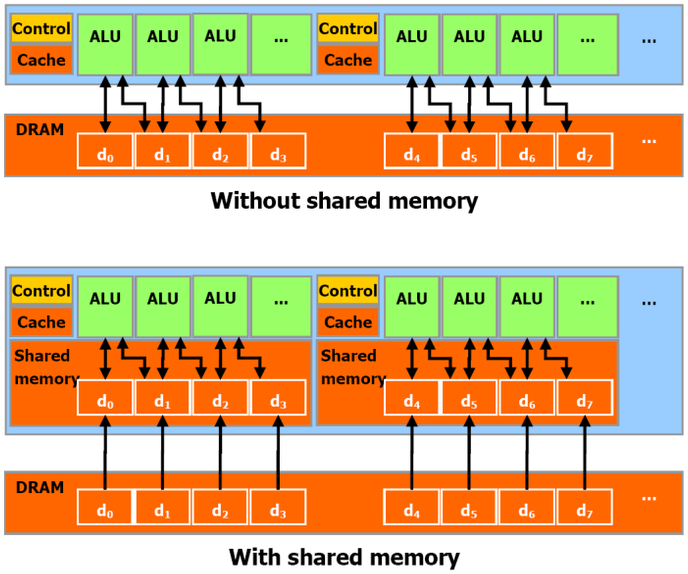


Figure 7: Shared Memory Brings Data Closer to the ALUs

Coalesced Memory Access

- **Coalesced memory access is mandatory!**
  - Every thread must access the correct memory bank!
  - Coalescing over a range of **16 threads**
  - Extreme performance penalty for non-coalesced memory access
  - Relaxed coalescing rules on the latest hardware

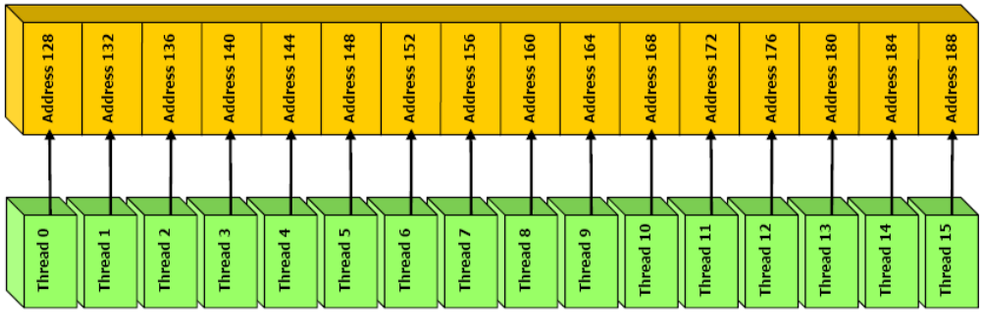


Figure 8: Coalesced Memory Access

What are the Implications?

- **Algorithms must be designed to fit within all these restrictions!**
  - Failure to do so results in very bad performance
  - Challenging algorithm design
  - Design for 1000s of threads per computational kernel
  - Excellent performance if done right!

(3) Algebraic Multigrid Solver on GPUs

- Current Work: Solver components on GPUs
  - CPU/GPU parallel algebraic multigrid solver component
  - Integration with the Parallel Toolbox framework
  - Support for multi-GPU computing node configurations
- Key performance indicator: Sparse matrix-vector multiplication
  - Multigrid prolongation and restriction operators and Jacobi smoother
  - Vector operations work just fine
  - Naive Gauss-Seidel does *not* work!
  - High algorithmic complexity in the AMG setup

Quad-GPU Compute Server

- **Custom Double Precision GPU Server with 960 Thread Processors**
  - MSI K9A2 Platinum Motherboard
  - AMD Phenom 9950 Quadcore Processor + 8 GB DDR2 RAM
  - **Quad** Nvidia GTX 280 @ 1.4 GHz + 1 GB Graphics RAM

Figure 9: Quad-GPU Compute Server

PCG-AMG Parallel Benchmark (Double Precision)

Domain decomposition based parallelization for CPU/GPU cluster computers!

NP	Solver	Setup
1	31.37	5.42
2	14.83	2.57
4	7.59	1.64
8	3.95	1.11
16	2.13	1.05
32	**1.35**	1.31

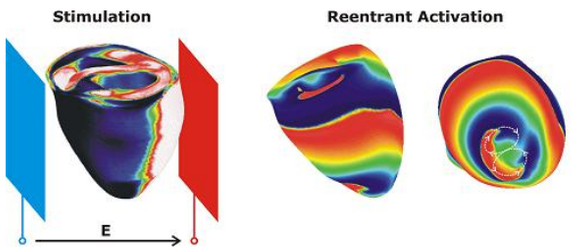
NP	Solver	Setup
1	30.59	5.93
2	14.84	3.11
4	7.48	1.95
8	4.10	1.51
16	**3.44**	1.91

NP	Solver	Setup
1	1.419	4.82
2	0.821	2.71
4	**0.530**	1.94

Table 1: Solver and setup times in seconds for Kepler (IB), Liebmann (SHM), and GTX (GPU)

Cooperations

- SFB: Mathematical Optimization and Applications in Biomedical Sciences (K. Kunisch, G. Plank, et al.)
  - Integration of CPU/GPU Solver Components with Cardiac Bidomain Model Simulation (CARP Software): **60x**



GPU Accelerated Algebraic Multigrid Methods for the Virtual Heart Model

Manfred Liebmann[†], Gernot Plank[‡], Anton Prassl[‡] and Gundolf Haase[†]

Overview of the Virtual Heart Model

The Bidomain Equations:

The bidomain equations are a set of coupled partial differential equations which describe the current flow in the myocardium (region I). Optionally, the current flow in a surrounding medium (torso, fluid bath, blood-filled cavities of the heart) are included in the formulation (regions II). The bidomain equations are written as follows:

$$\begin{aligned} -\nabla\cdot(\bar{\sigma}_i\nabla\phi_i) &= -\beta I_m \\ -\nabla\cdot(\bar{\sigma}_e\nabla\phi_e) &= \beta I_m \\ -\nabla\cdot(\bar{\sigma}_b\nabla\phi_e) &= I_e \end{aligned}$$

where

$$\begin{aligned} I_m &= C_m\frac{\partial V_m}{\partial t} + I_{ion}(V_m,\vec{\eta}) - I_{tr} \\ \frac{d\vec{\eta}}{dt} &= g(V_m,\vec{\eta}) \\ V_m &= \phi_i - \phi_e \end{aligned}$$

The bidomain solver of the Cardiac Arrhythmia Research Package CARP, one of the most efficient solvers for this type of problem, implements several solution methods. Most frequently, however, based on an operator splitting technique the following scheme is employed [7, 8, 1]. The bidomain equations are decoupled into

an Elliptic PDE

$$(A_i + A_e)\Phi_e^{k+1} = A_i V^{k+1} + I_e$$

a Parabolic PDE

$$\begin{cases} V^{k*} = (1-\Delta t A_i)\,V^k - \Delta t A_e\phi_e^k & \Delta x > 100\mu m \\ \left[1+\frac{1}{2}\Delta t A_i\right]V^{k*} = \left[1-\frac{1}{2}\Delta t A_i\right]V^k - \Delta t A_e\phi_e^k & \Delta x < 100\mu m \end{cases}$$

and a set of ODE's

$$\begin{aligned} V^{k+1} &= V^{k*} + \frac{\Delta t}{C_m} i_{ion}\left(V^{k*},\vec{\eta}^k\right) \\ \vec{\eta}^{k+1} &= \vec{\eta}^k + \Delta t\, g(V^{k+1},\vec{\eta}^k) \end{aligned}$$

where

$$A_i = -\frac{\nabla\cdot(\bar{\sigma}_i\nabla)}{\beta C_m};\; A_e = -\frac{\nabla\cdot(\bar{\sigma}_i\nabla)}{\beta C_m};\; t = k\Delta t$$

Further, besides the bidomain solver, CARP consists of the ionic model library IMP, the visualization tool Meshalyzer, and codes for the computation of ECG (ϕ_e recovery) and MCG (cooperation with Dr. Weber dos Santos and the Physikalisch Technische Bundesanstalt in Berlin).

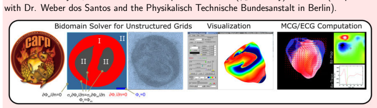
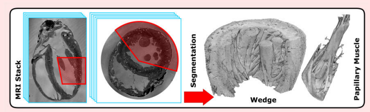


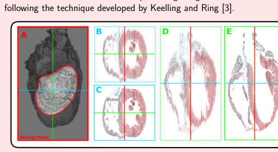
Image Processing and Mesh Generation:

A first attempt to segment high-resolution MRI image stacks and histological image stacks in a semi-automatic fashion has been undertaken [2]. In trichrome-stained histological sections, interesting tissue types and the extracellular space occupy areas of the color histogram with minimal overlapping and thus can be segmented directly using color thresholding. Segmentation of the 3D MRI datasets constitutes a more complicated challenge. This process required several steps including the estimation and removal of the bias field and a set of ad-hoc morphological operations.



Registration

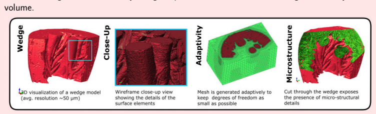
In the process of slicing the specimen, preparing the section and acquiring histological images, considerable rigid and non-rigid distortion is introduced. We correct for this distortion by using slice-to-slice registration, in two steps. In the first step, an initial coarse rigid registration alignment is realized. In the second step, non-rigid registration between adjacent slices is performed following the technique developed by Keelling and Ring [3].



A) 3D Reconstruction of a contracted rabbit heart. Shown Serial sectioning was performed approximately along the medial-lateral axis of the heart, i.e., the slicing plane was parallel to the ventricular septum. Slices were registered using an elastic registration with maximum rigidity. Registration results are visualized by comparing images before (B and D) and after (C and E) registration in observation planes which are perpendicular to the slicing plane.

Mesh Generation

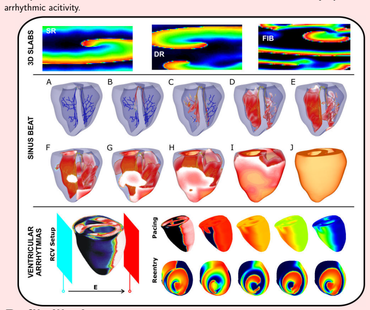
Within the framework of a national cooperation, the Octree-based meshing software TARANTULA has been developed which produces boundary-fitted, locally-refined and conformal multi-element meshes on unstructured grids to allow 1) smooth representation of organ boundaries, and 2) to reduce the degrees of freedom by using adaptive methods when discretizing the non-myocardial volume.



3D visualization of a wedge model (avg. resolution ~50 μm) — Wireframe close-up view showing the details of the surface elements — Mesh is generated adaptively to keep degrees of freedom as small as possible — Cut through the wedge exposes the presence of micro-structural details

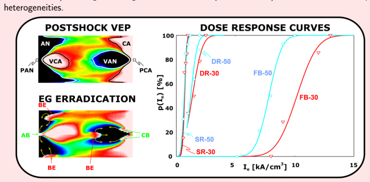
Arrhythmogenesis:

The group of the applicant has gathered experience in carrying out computer simulations dealing with mechanisms underlying the formation of arrhythmias using both simple slab-like geometries and more realistic setups using anatomically realistic representations of the ventricular geometry. Further, a novel method to simulate to incorporate the Purkinje system has been developed recently to allow the simulation of sinus beats and the interaction of the Purkinje system with arrhythmic acitivity.



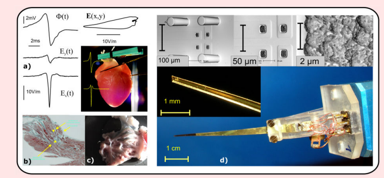
Defibrillation:

Electrical defibrillation is the application of a large electrical shock to the heart to terminate otherwise life-threatening arrhythmias. The applicant has been carrying out simulation studies [6] which investigated the relationship between shock energy requirements for successfull defibrillation and 1) the degree of organization of an arrhythmia, and 2) the effect of microscopic heterogeneities.



Experimental Validation:

A new flexible sensor for in-vitro experiments has been developed [11, 10, 9, 5] to measure the surface potential Φ, and its gradient, $\mathbf{E}$ (electric near field), at given sites of the heart. During depolarisation, $\mathbf{E}$ describes a vector loop from which direction and magnitude of local conduction velocity ϑ can be computed. Four recording silver electrodes separated by 50 μm, conducting leads, and solderable pads were patterned on a 50 μm thick polyimide film. Combined with high-resolution data acquisition (sampling rate 100-800 kHz at 16- 24-bit) we are able to discriminate signal latencies of just a few microseconds and to monitor the computed time-course of $\mathbf{E}$ as well as the magnitude and direction of local conduction velocity ϑ on-line from beat-to-beat. Thus, the capability to measure accurately activation time as well as direction and velocity of propagation at the microscopic size scale makes the method ideally suited for the validation of computer simulations using micro-anatomical grids, as suggested in this SFB proposal. A map of $\mathbf{E}$ combined with the corresponding histographs forms the fundament for a micro-structure related computer mode. (a) Guinea-Pig heart and signals Φ and $\mathbf{E}$. (b) histograph with myocytes (red) and barriers from connective tissue (blue). (c) right atrial endocard. (d) sensor.



References

[1] G. Plank, M. Liebmann, R. Weber dos Santos, E. Vigmond and G. Haase. **Algebraic Multigrid Preconditioner for the Cardiac Bidomain Model**, IEEE Trans. Biomed. Eng., accepted pending minor revision.

[2] Burton* A.B.R., G. Plank*, J. Schneider*, V. Grau Colomer, H. Ahammer, S.L. Keeling, J.L. Lee, N. Smith, N.A. Trayanova and P. Kohl. (* shared first authors) **3-Dimensional Models of Individual Cardiac Histo-Anatomy: Tools and Challenges.** Ann. N.Y. Acad. Sci., in press.

[3] S. Keeling, W. Ring. **Medical image registration and interpolation by optical flow with maximal rigidity.** J. Math. Imag. Vision. 23:4765, 2005.

[4] N. Trayanova, G. Plank, and B. Rodriguez. **What have we learned from Mathematical Models of Defibrillation and Postshock Arrhythmogenesis?, Application of Bidomain Simulations**,E Heart Rhythm, in press.

[5] Hofer, E., F. Kepplinger, T. Thurner, T. Wiener, and G. Plank. **A novel floating sensor array to detect electric nearfields of beating heart preparations.**, Biosensors and Bioelectronics, 21(12), 2232-2239, 2006.

[6] Plank G., L.J. Leon, S. Kimber, E.J. Vigmond, **Defibrillation depends on conductivity fluctuations and the degree of disorganization in reentry patterns.**, J. Cardiovasc. Electrophysiol., 16(2):205-216, 2005.

[7] Weber dos Santos, R., G. Plank, S. Bauer, E.J. Vigmond, **Parallel multigrid preconditioner for the cardiac bidomain model.**, IEEE Trans. Biomed. Eng., 51(11):1960-8, 2004.

[8] Weber dos Santos, R., G. Plank, S. Bauer, E.J. Vigmond, **Preconditioning techniques for the bidomain equations.** Lecture Notes in Computational Science and Engineering (LNCSE), ISBN/ISSN 14397358, 40:571-580, 2004.

[9] Plank, G. E.J. Vigmond, L.J. Leon, E. Hofer. **Cardiac near-field morphology during conduction around a microscopic obstacle - a computer simulation study.** Ann. Biomed. Eng., 31:1-7, 2003.

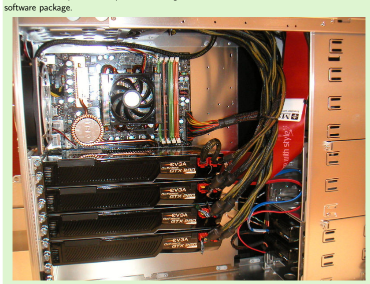
[10] Plank, G. E. Hofer. **The use of cardiac near-field measurements to determine activation times.** Ann. Biomed. Eng., 31:1066-1076, 2003.

[11] Plank, G. and E. Hofer. **Model study of vector-loop morphology during electrical mapping of microscopic conduction in cardiac tissue.** Ann. Biomed. Eng., 28(10):1244-1252, 2000.

Multi-GPU Accelerated Algebraic Multigrid Methods

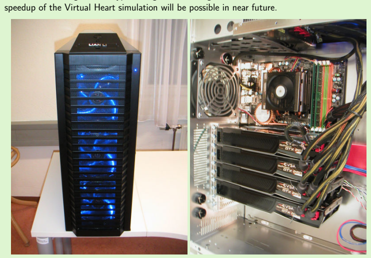
CARP Simulator:

The capabilities of the CARP simulator are extended to include a multi-GPU algebraic multigrid solver for the elliptic PDE implemented using the Nvidia CUDA Toolkit and the Parallel Toolbox software package.



Quad-GPU Compute Server:

A test platform for the multi-GPU PCG-AMG solver has been set up using a MSI K9A2 Platinum mainboard, a quadcore Phenom 9950 processor, and four Nvidia GTX 280 boards. Early tests show good scaling of the parallel multigrid solver on the test platform. Giving a 50-fold performance advantage over a typical single CPU configuration. With these results a tremendous speedup of the Virtual Heart simulation will be possible in near future.



PCG-AMG Benchmark

The algebraic multigrid method (AMG) provides an efficient preconditioner for the preconditioned conjugate gradient (PCG) algorithm. Currently the complete multigrid iteration within the PCG solver is executed on the GPU while the setup of the algebraic multigrid method is handled by the CPU due to high complexity of the algorithm. Table 1 shows the comparison of a single PCG-AMG iteration on different cluster computers: Kepler (32 Opteron processors, Infiniband interconnect), Boltzmann (16 Opteron processors, Gigabit Ethernet network) and Liebmann (4 Quadcore Barcelona processors, Shared memory architecture). The finite element matrix of the Virtual Heart simulation has dimension 862.515 with $12,795.209$ nonzero elements. All computations use double precision arithmetics.

N	$\#A$	Kepler 32P (IB)	Boltzmann 16P (GBE)	Liebmann 16P (SHM)	Single GTX 280	Dual GTX 280	Quad GTX 280
862.515	12,795.209	0.021094	0.058125	0.053750	0.022178	0.012828	0.008277

Table 1: Timing of PCG-AMG iteration in seconds (double precision)

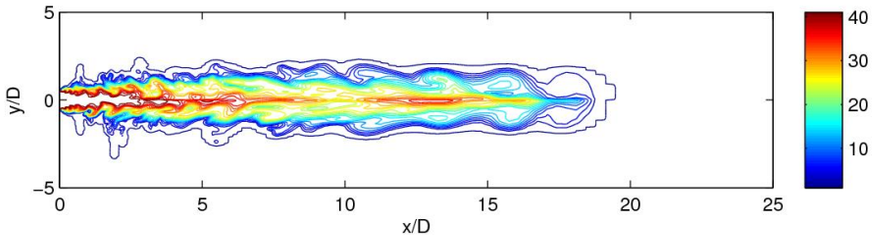
Considering the cost of a typical cluster computer compared to a single Nvidia GTX 280 board we see two to three orders of magnitude in price/performance advantage.

FWF MOBIS AUSTRIAN GRID UNI GRAZ

[†] Institute for Mathematics and Scientific Computing, Univerisity of Graz, Austria [‡] Institute of Biophysics, Center for Physiological Medicine, Medical University of Graz, Austria

More Cooperations

- TU Graz: ISW (H. Steiner, D. Heidorn)
  - Algebraic Multigrid on GPUs for Fluid Dynamics Simulations: **26x**



- NAWI Graz: Time Optimal Control (K. Kunisch, J. Rubesa)
  - Time Optimal Control Problem on GPU: **13x** (50x)

Conclusions

- **GPU computing looks very promising!**
  - Attractive price / performance ratio
  - Tesla computing solutions for HPC applications
  - Incremental software development supporting C/C++
  - Reasonable porting effort using the free CUDA toolkit
  - Website: **www.nvidia.com/cuda/**

Figure 10: Tesla Computing Solutions

Thank you!