Investigations on Improving the SEE-GRID Optimization Algorithm

Johannes Watzl

July 2008

The SEE-KID/SEE-GRID Software

SEE-KID

- Software to help doctors treating eye motility disorders
  - Strabism
  - Simulation of certain tests
  - Simulation of surgery results
- Works on a biomechanical model of the human eye
- Developed at the Upper Austrian Research GmbH. (UAR)

SEE-GRID

- Grid Version of the software
- Developed at the Research Institute for Symbolic Computation (RISC)

Problem description

Torque Function

- Function to compute the eye postion with eye data as input
- Function has to be minimized
- $f_{torque} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$

Minimization of the Torque Function

- Done by the Levenberg Marquardt Optimization Algorithm (fast and stable optimizer)
- Time consuming part of the simulations performed by the SEE-KID/SEE-GRID software

Strategies to Accelerate the Computation Process

Broyden Update Method Avoids recomputing the Jacobian or Hessian matrices in every optimization step by using an update formula

Delaunay Algorithm Triangulates a given mesh of torque function result values inside the function's domain. Requested values are interpolated.

Interpolation using the Regular Mesh Structure The Delaunay algorithm is designed for general meshes and therefore uses a triangle search algorithm. This search can be replaced by simple lookups in regular meshes.

Strategies to Accelerate the Computation Process

Description

Updating the Jacobian matrix utilizing

- The input values of the torque function x^{k-1} and x^k
- The torque function result values $F(x^{k-1})$ and $F(x^k)$
- The Jacobian matrix of the last step J^{k-1} (k denotes the number of the step)

Problem

Decreasing computation time by loss of precision

Strategies to Accelerate the Computation Process

Description

- Create a three dimensional mesh in the torque function's domain
- Triangulate the mesh (the given set of node points)
- Store a list of triangles
- Search the triangle surrounding a requested point and interpolate the value with the three triangle vertex points

Problem

- Delaunay Algorithm is designed to deal with general meshes (causes overhead working with a regular mesh).
- A Search Algorithm is necessary.

Strategies to Accelerate the Computation Process

Description

- Decompose the domain into *subcubes*
- Check if the subcube holding the requested point is already computed (if no: fill the subcube with the function values and store it)
- Perform a lookup in the subcube to get the three surrounding node points of a requested function value
- Interpolate the requested function value

Problem

- Increasing computation time
- Precision problems

Parallelization

Parallel Interpolation

Perform the computation of the three elements of the result vector in parallel

Parallel Subcube Computation

Perform the filling of the subcube with the torque function result vectors in parallel

Computation in Advance

Overlapping strategy between the optimization and the grid point computation

subcubes are computed in advance during the optimization process

Parallelization

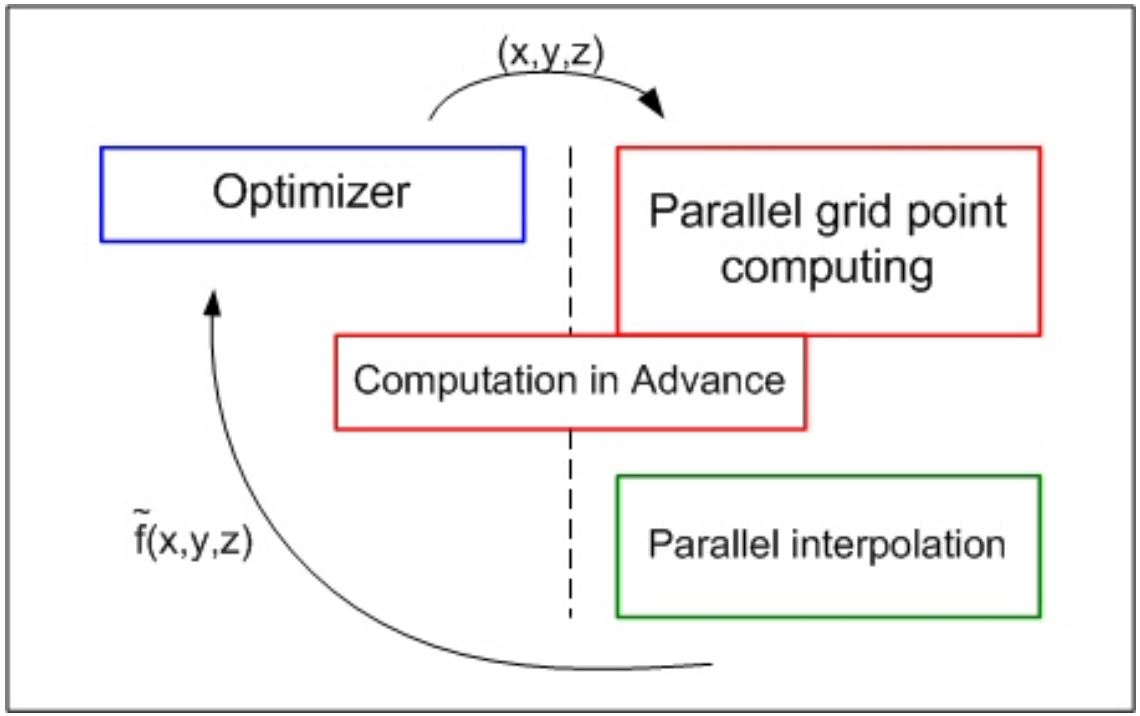

Figure: Schematical view of the parallel interpolation process

Implementation

Parallel Interpolation

Performed making use of POSIX threads (one thread for each element of the result vector)

Parallel Subcube Computation

Performed with the help of the OpenMP automatic parallelizer's parallel for (parallelizing nested loops)
$\longrightarrow$ **Problem**

Computation in Advance

Performed featuring the OpenMP automatic parallelizer with parallel sections $\longrightarrow$ **Problem**

Implementation Problem

Newmat library

Newmat Matrix library used in SEE-KID/SEE-GRID is not thread safe

Counter Strategies

- Replace the Newmat library by a thread safe matrix library
- Use a Model to override the parallel usage of Newmat (Client Server Model, process forking)

Results of the Sequential Approaches

Broyden Update Method

- Higher computation time (optimizer negatively influenced)
- Incorrect results due to loss of precision

Delaunay Algorithm

- Correct interpolation results in most of the cases
- Higher computation time caused by the triangle search

Interpolation using the Regular Mesh Structure

- Provided exact results in most of the cases
- Higher computation time because of the large number of torque function evaluations

Results of the Parallel Approach

Parallel Interpolation using the Regular Mesh Structure

- Speedup gained with respect to the sequential interpolation
- No Speedup gained as to the original computation

Model 1	$time_{orig}$ (s)	$time_{seq}$ (s)	$time_2$ (s)	$time_4$ (s)	$time_8$ (s)	$time_{16}$ (s)
Parameter 1	5.735	83.428	41.251	22.21	12.108	8.11
			43.222	24.174	12.129	9.096
			44.245	23.131	13.063	9.161
			42.276	22.091	12.11	9.202
			48.104	23.141	12.114	10.146
Average			43.248	22.827	12.118	9.153
Parameter 2	5.735	139.131	83.478	45.349	23.155	17.234
			82.219	44.283	24.204	15.17
			82.44	44.16	23.239	16.116
			479.231	46.593	23.145	14.227
			82.345	44.279	22.224	14.199
Average			82.335	44.637	23.18	15.171

Figure: Timings of the parallel subcube computation

Conclusions

- Both sequential and parallel strategies for acceleration were investigated.
- The strategies were implemented on the Altix 4700 multiprocessor system.
  - OpenMP
  - Posix Threads
  - Unix processes with shared memory communication
- The results were systematically experimentally evaluated compared to the original solution
  - Precision of results
  - Speed of computation