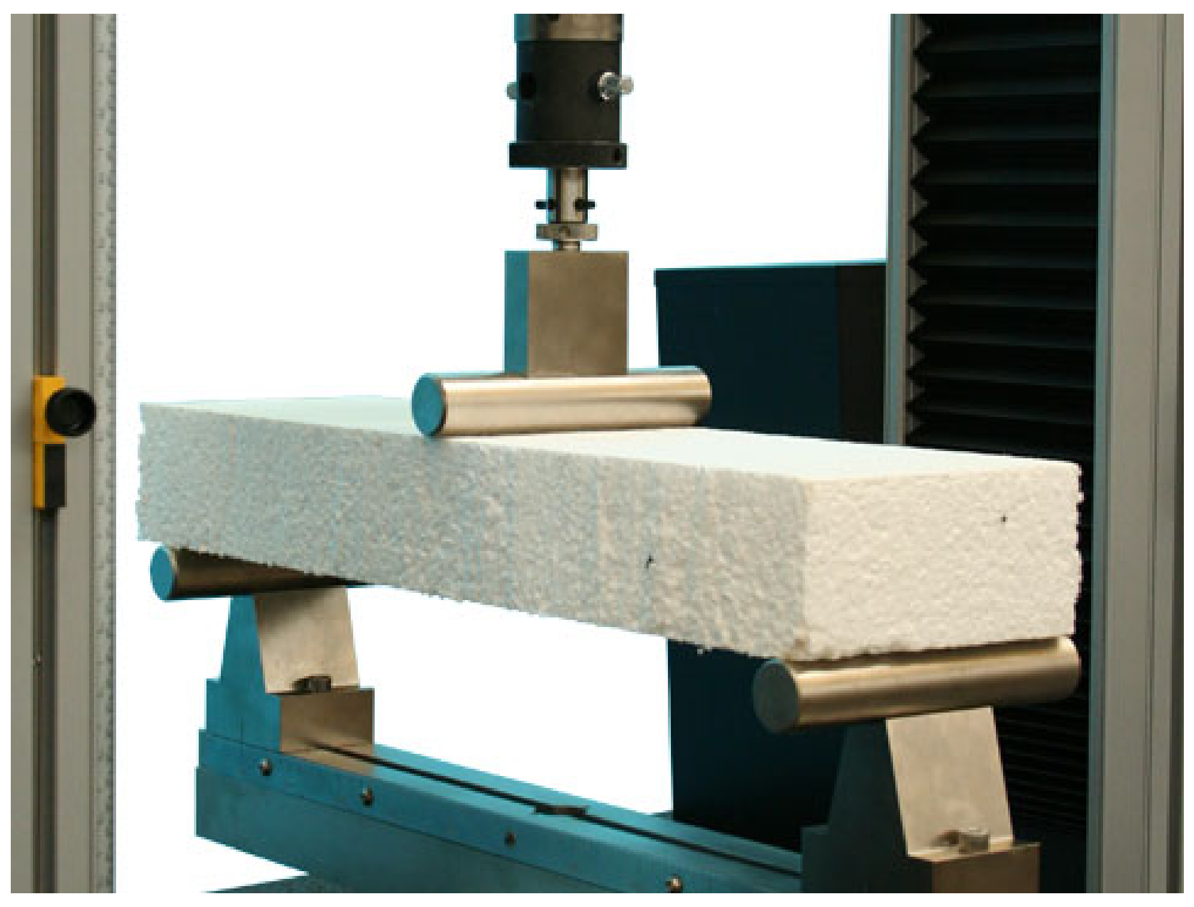
Optimization of Fracture Tests Simulation in Civil Engineering

Gabriel Bordovsky and Jiri Jaros

Faculty of Information Technology, Brno University of Technology, Centre of Excellence IT4Innovations, CZ

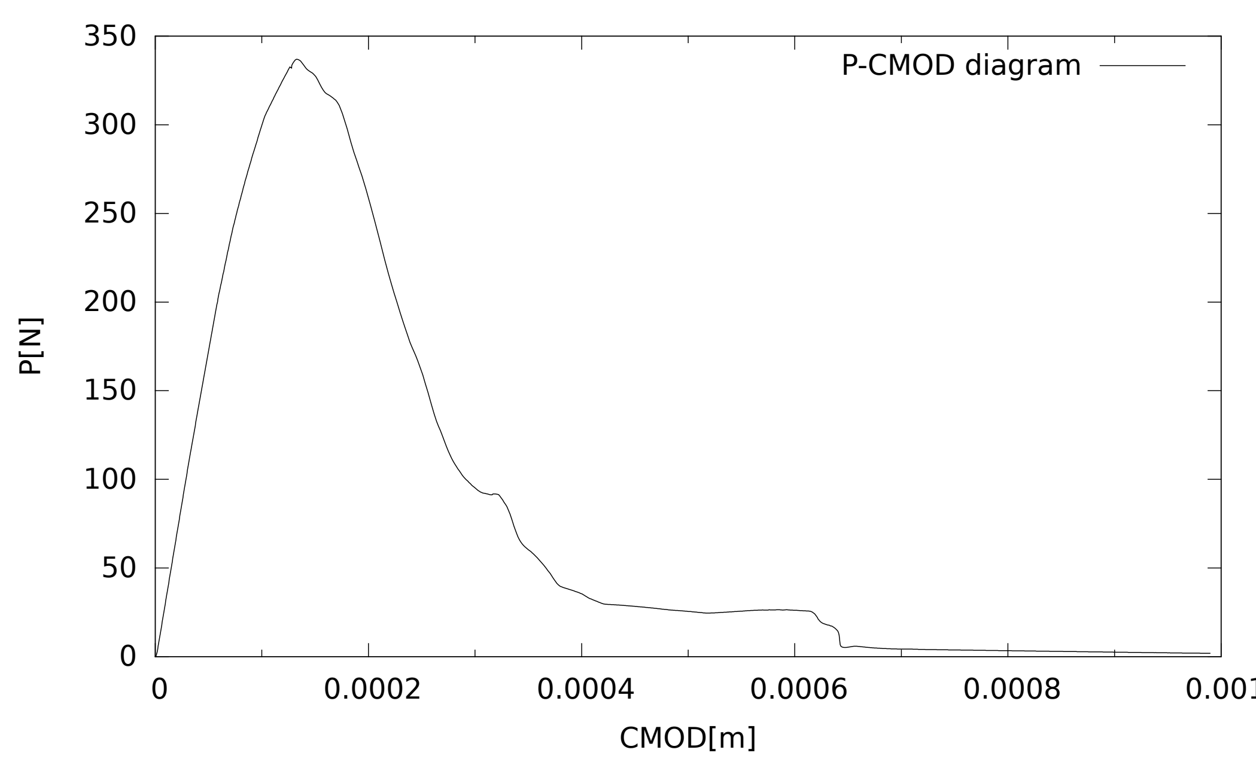
The Three Point Bending Fracture Test

Reconstructions of historic buildings require precise determination of material properties. New blocks of sandstone has to match with others in terms of material fatigue. Sample of the old material has to be destroyed during the bending test to obtain information about material reaction to constant load.

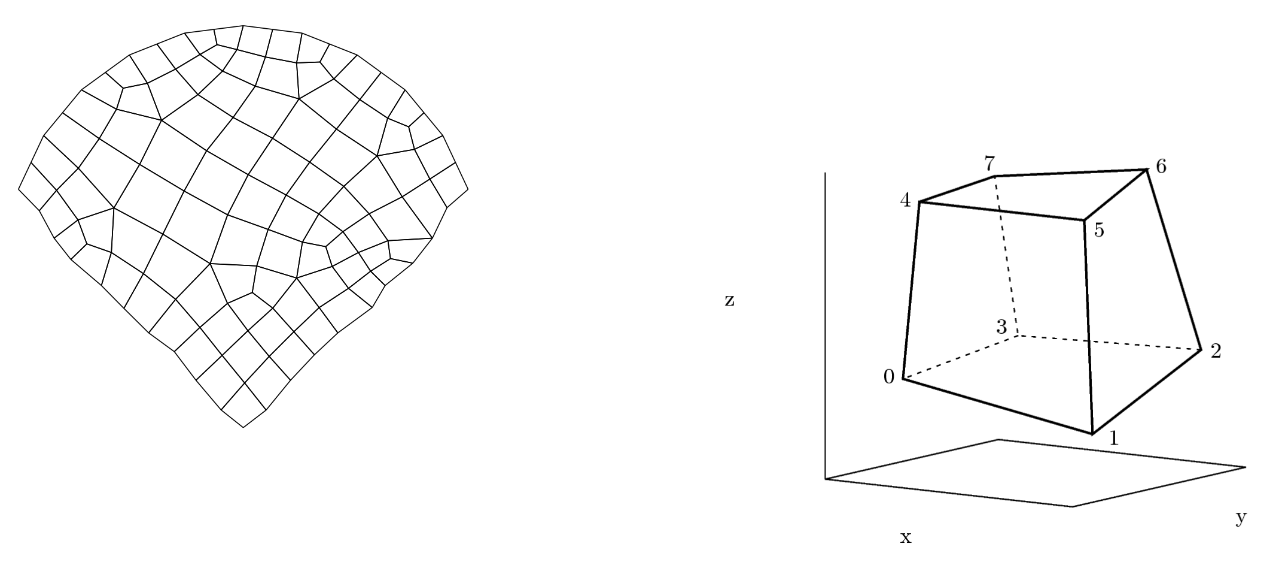


Characteristics Validation Based on Simulation

The output of the test is Load P and deformation (Crack Mouth Opening Displacement) diagram and depends on many material characteristics that may be difficult to determinate analytically. The result of the simulation matches the real test if correct characteristics are provided. During the bending test, the quasi-bristle material such as sandstone change behavior from elastic to quasi-fragile. There are first created a micro fractures and then a main fracture line is formed and widened in the quasi-fragile phase.



Model Description



Used model is composed of randomly placed points bonded together to form blocks, finite elements. Each block has its own matrix of stiffness describing how the movement of one point in the block affects others. The reactions R affecting a point i are computed from all N blocks that the point is a member of, and changes in position of other points j from the beginning of simulation.

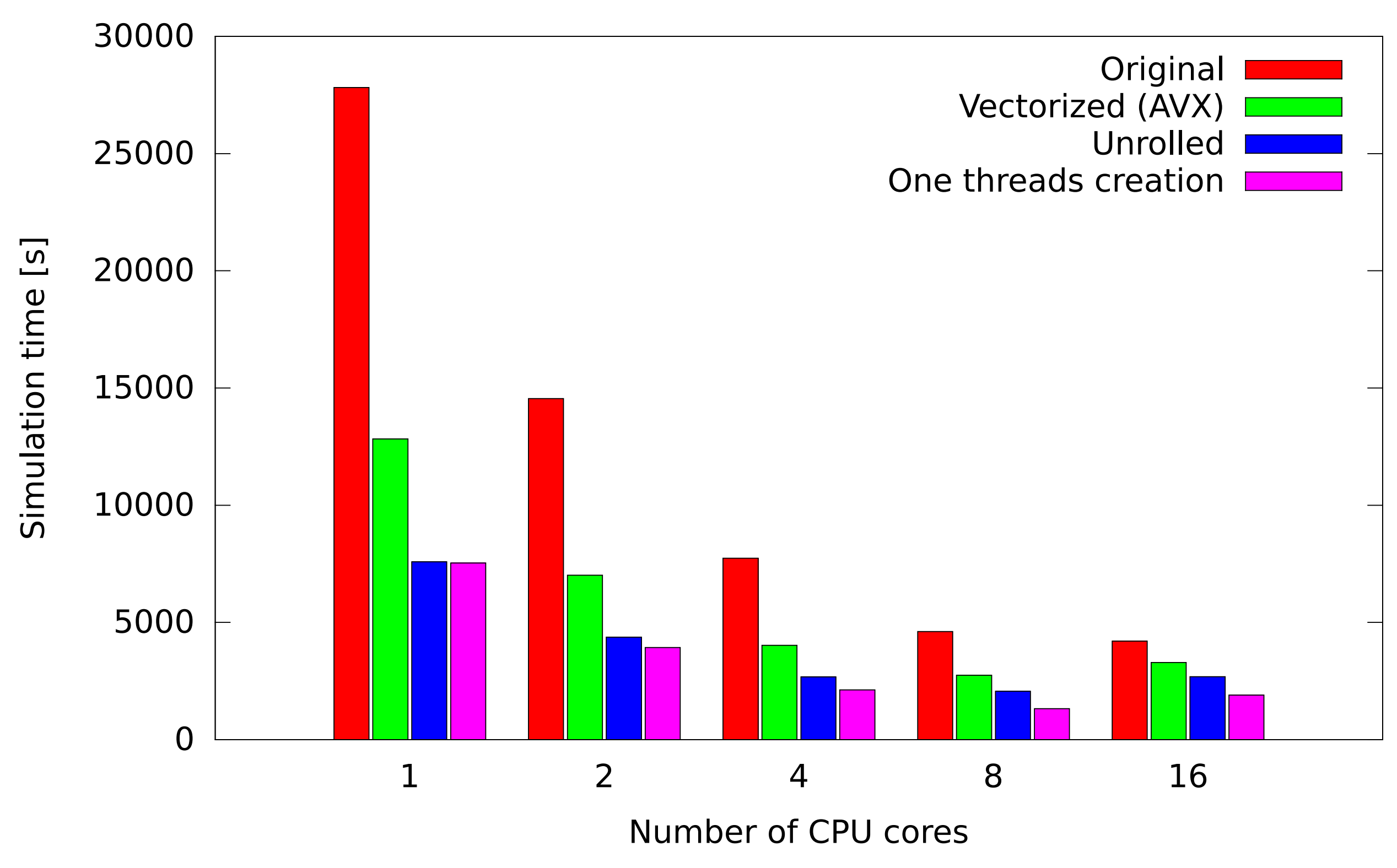
$$R_i = \sum_{n=1}^{N}\sum_{j=0}^{7}\big(p_j(t) - p_j(0)\big)\cdot S_n[i][j]$$

The new position $p(t+h)$ of the points is then computed from the current position $p(t)$ and speed $v(t+h)$ using the Euler method with step h. For the speed $v(t+h)$ of the point, the Euler method is also used. The c is used as attenuation coefficient, m as a weight of the point and R as total reaction that affected given point in step t.

$$p(t+h) = p(t) + h\cdot v(t+h)$$
$$v(t+h) = v(t) + h\cdot\left(\frac{R - c\cdot v(t)}{m}\right)$$

Used Optimization and Scalability

The original prototype has taken 27 815 $seconds$ to simulate the fracture test with a model composed from 2 310 $elements$ / 2 946 $points$ on a single thread (on Anselm). The critical section of simulation's main loop was parallelized using physical threads, vectorized using AVX, reordered and unrolled. In last phase the parallel processing was modified to run for whole simulation, and not just for the critical section, therefore, the threads creation overhead was lowered.

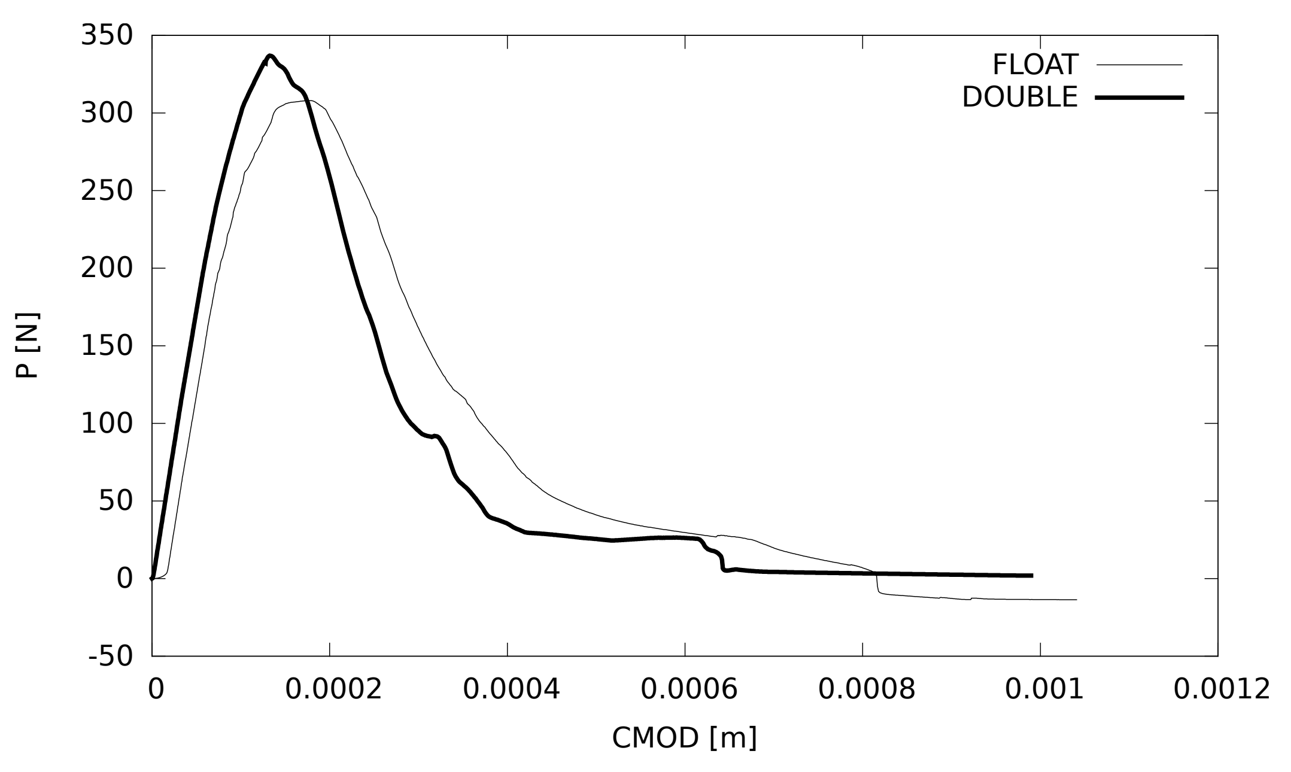


The single thread version was **3.69 times faster** (7 531 s) after the optimization. When using all eight processors cores on one chip, the solution was even **5.7 times faster** (1 320 s). In total the solution on 8 cores was **more then 21 times faster** then prototype.

Optimized Simulation Analyze

Since for the accurate simulation, it is sufficient to compose the model from circa 3 000 points, great part of required data may be stored in L3 cache. Increasing granularity of the model does not increase the output quality.

The simulation requires the usage of *double precision* number representation. The interactions between points in the beginning of the simulation are very small but significant. Neglecting them by using *single precision* cause the load P to go into negative numbers, which is physically impossible.



The main problem with the scalability of the simulation is a short iteration runtime. On one chip the computation of a single iteration takes 132μs in average, from which 44μs in the critical section. This does not provide enough time to exchange required data between cores placed on different chips. Also random distribution of points and undefined maximal number of finite elements per point makes it difficult to split the simulation domain into balanced chunks.

Conclusions

The optimized solution provides 3.69 times faster solution thanks to identification of compute demanding sections, theirs vectorization and rearranging of the code, mainly loops. With effective thread creation the code provides another $0.71 * number\ of\ cores$ speedup on single processor.

	One core	All cores		Speedup	Effectivity
Anselm (8 cores)	7 531s	1 320s	22,00min	5,705	71,316%
Salomon (12 cores)	7 060s	826s	13,76min	8,547	71.226%

This work was supported by The Ministry of Education, Youth and Sports of the Czech Republic from the Large Infrastructures for Research, Experimental Development and Innovations project "IT4Innovations National Supercomputing Center – LM2015070", and the National Programme of Sustainability (NPU II) project "IT4Innovations Excellence in Science – LQ1602".