AMPL format .mod, to Modelica for later treatment by JModelica.org. These problem series offer similar systems of different size. In Figures 2 and 3 the speedups of both the Broyden and Moreaux problems are plotted against the number of variables of the systems. In Figure 2 the speedup of the total time is plotted while Figure 3 plots the speedup of the total time except the time spent evaluating system functions and Jacobians. Here speedup means the time measured with dense solver divided by time measured with sparse solver, ei.e. how many times faster the sparse solver is than the dense solver. It should be noticed that computation of Jacobians in JModelica.org used for the benchmarks is slow, due to limitations in the CppAD package, [3], with regards to sparse Jacobians. Therefore, we focus on comparison of the time spent in KINSOL in the cases of sparse versus dense linear solvers.

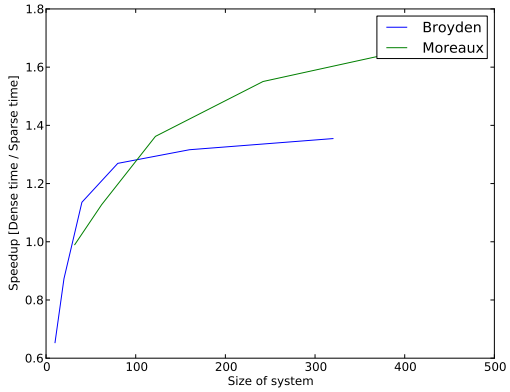

Figure 2: Speedup of the total times for the Broyden and Moreaux problems.

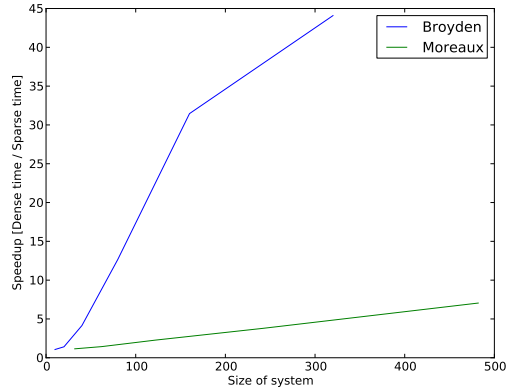

Figure 3: Speedup of the Broyden and Moreaux problems not counting time in fevals and jevals.

In Tables 1 and 2 the data from the test runs on the Broyden and Moreaux preoblems are presented. The data consists of the time spent in total is presented (tot) along the time spent in KINSOL and linear solver without evaluations of Jacobians and system functions (KIN), the time spent on evaluating the residual and Jacobian (Evals) and the number of non-linear iterations required (iters). The data is scaled by the total time of the dense solver to simplify comparison.

Table 1: Times measured for the Broyden problems.

	Broyden	10	40	80	320
	size	10	40	80	320
tot	Dense	1.0	1.0	1.0	1.0
	Sparse	1.530	0.880	0.787	0.738
KIN	Dense	0.452	0.249	0.244	0.243
	Sparse	0.425	0.060	0.019	0.006
Evals	Dense	0.548	0.751	0.756	0.757
	Sparse	1.105	0.820	0.768	0.732
iters	Dense	17	60	112	137
	Sparse	17	60	112	137

Table 2: Times measured for the Moreaux problems.

	Moreaux	10	40	80	160
	size	32	122	242	482
tot	Dense	1.0	1.0	1.0	1.0
	Sparse	1.010	0.734	0.645	0.584
KIN	Dense	0.666	0.495	0.457	0.435
	Sparse	0.580	0.218	0.121	0.062
Evals	Dense	0.334	0.505	0.543	0.565
	Sparse	0.430	0.516	0.524	0.522
iters	Dense	38	112	123	137
	Sparse	38	112	123	137

The models tested in Tables 1 and 2 are not models originally written in Modelica but rather optimization benchmarks. To test how the initialization algorithm using a sparse solver behaves when used on 'real' Modelica models, the same test performed in Tables 1 and 2 are performed on some models with different sizes in Table 3.

- CSTR: an example from the JModelica.org package describing two continously stirred tank reactors in series.
- DIST: an example from the JModelica.org package already mentioned in Section 6.1.

- CoCy: a Modelica model describing a combined cycle power plant initialized at full load.

Table 3: Times measured for the Modelica models.

	Model	CSTR	Dist	CoCy
	size	15	99	150
tot	Dense	1.0	1.0	1.0
	Sparse	1.112	0.599	0.635
KIN	Dense	0.645	0.552	0.426
	Sparse	0.610	0.115	0.112
Evals	Dense	0.355	0.448	0.574
	Sparse	0.502	0.484	0.523
iters	Dense	10	24	34
	Sparse	10	24	34

6.3 Sparsity of Jacobians

It is also interesting to take a look at the sparsity of the systems to put the results obtained in Section 6.2. In Table 4 the sparsity of the systems, that is the sparsity of the system Jacobian, timed in Table 1 and 2, are presented.

Table 4: Sparsity measured in the percentage of elements differnet from zero in the Jacobian.

Model	10	20	40	80	160	320
Broyden	54.0	31.0	16.5	8.5	4.31	2.17
Moreaux	8.98	4.73	2.43	1.23	0.62	-

In Table 5 the sparsity of the Modelica models timed in Table 3 are presented supporting the hypothesis of bigger models being more sparse..

Table 5: Sparsity measured in the percentage of elements differnet from zero in the Jacobian.

Model	Size	Sparisty
CSTR	15	24.0
DIST	99	2.67
CoCy	150	1.75

7 Conclusions

The regularization method is handling singular Jacobians at initialization. So far, no models, supported by JModelica.org, have caused the initialization algorithm based on regularization to stop due to a singular Jacobian. A problem with a singular Jacobian at the solution is however solved slower, as shown in the end of Section 4.1. Pan and Fan [10] proposes techniques to handle this problem that may be used in a later implementations.

Table 5 imply that larger Modelica models are more sparse than smaller Modelica models, thus supporting the hypothesis stated in section 5.1 of Modelica models getting more sparse as they grow in size.

Regarding sparsity, Figures 2, 3 and Tables 1, 2 and 3 imply that the problems are initialized faster with the sparse version of the initialization algorithm. Due to CppAD slowing down the evaluations of Jacobian, the times spent in KINSOL are compared instead of the total time.

When applied to the Modelica models in Table 3, the sparse version solves the bigger problems (of size $n \approx 100$ or bigger) around 4-5 times faster than the dense version. The bigger benchmarks from the Broyden and Moreaux series show an even bigger speedup, Broyden320 is for example solved 40 times faster. For smaller model, like the model of the two stirred tank reactors, the organizational effort of SuperLU and the model being less sparse, outweighs the advantages and the two methods are equal.

In the benchmarks presented here, the time for evaluating Jacobians outweights the time spent in KINSOL, especially if SuperLU is employed. This is due to the fact that the package used for generation of Jacobians has weak support for computation of sparse deriviatives. This deficiency will be adressed in future versions of JModelica.org.

In conclusion, the sparse version of the initialization algorithm is advantageous when applied to bigger models. For smaller models however, the two version performs equally. However, the slow evaluation of sparse Jacobians make the dense solver a better choice for smaller models.

References

[1] The assimulo homepage. `http://www.jmodelica.org/page/199`.

[2] The coconut benchmark: Library 3 constraint satisfaction test problems. `http://www.mat.univie.ac.at/~neum/glopt/coconut/Benchmark/Library3_new_v1.html`.

[3] The cppad webpage. `http://www.coin-or.org/CppAD/`.

[4] Stefan Behnel, Robert Bradshaw, Dag Sverre Seljebotn, and other contributors. Cython c-extensions for python - homepage. `http://www.cython.org/`.

[5] Aaron M. Collier, Alan C. Hindmarshand Radu Serban, and Carol S. Woodward. *User Documentation for kinsol v2.6.0.* Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, May 2009.

[6] The SciPy community. *SciPy Reference Guide*, 0.9.0.dev6665 edition, October 2010.

[7] J.E. Dennis and R.B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations.* Prentice Hall, 1983.

[8] Hilding Elmqvist. *A Structured Model Language for Large Continuous Systems.* PhD thesis, Department of Automatic Control, Lund University, Sweden, may 1978. TFRT-1015.

[9] Jan Eriksson. A note on the decomposition of systems of sparse non-linear equations. *BIT Numerical Mathematics*, 16(4):462–465, 1976. 10.1007/BF01932730.

[10] Jinyan Fan and Jianyu Pan. A note on the levenberg-marquardt parameter. *Applied Mathematics and Computation*, 207:351–359, 2009.

[11] John E. Hopcroft and Richard M. Karp. An $^{2/5}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973.

[12] X. S. L. W. Demmel James W. Demmel, John R. Gilbert, Stanley C. Eisenstat, John R. Gilbert, Xiaoye S. Li, and Joseph W. H. Liu. A supernodal approach to sparse partial pivoting. *SIAM J. MATRIX ANAL. APPL*, 20(3):720–755, 1999.

[13] Xiaoye S. Li James W. Demmel, John R. Gilbert. *SuperLU Users Guide.*

[14] Johan Åkesson, Karl-Erik Årzén, Magnus Gäfvert, Tove Bergdahl, and Hubertus Tummescheit. Modeling and optimization with optimica and jmodelica.org - languages and tools for solving large-scale dynamic optimization problem. *Computers and Chemical Engineering*, 34(11):1737–1749, nov 2010.

[15] K. Levenberg. A method for the solution of certain nonlinear problems in least squares. *Quart. Appl. Math.*, 2:164–166, 1944.

[16] D.W. Marquardt. An algorithm for least-squares estimation of nonlinear inequalities. *SIAM J. Appl. Math.*, 11:431–441, 1963.

[17] S.E Mattson, H. Elmqvist, M. Otter, and H. Olsson. Initialization of hybrid differential-algebraic equations in modelica 2.0. In *Second International Modelica Conferencem, Proceedings*, pages 9–15. The Modelica Association, March 2002.

[18] Sven Erik Mattsson and Gustaf Söderlind. Index reduction in differential-algebraic equations using dummy derivatives. *SIAM J. Sci. Comput*, 14(3):677–692, May 1993.

[19] Arnold Neumaier. Solving ill-conditioned and singular linear systems: A tutorial on regularization. *SIAM Review*, 40(3):636–666, September 1998.

[20] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization.* Springer, 2006.

Acknowledgements

The authors would like to acknowledge the kind assistance from Francesco Casella in providing the Combined Cycle benchmark model used in the paper. This work was partially funded by the Swedish funding agency Vinnova under the grant program "Forska and Väx".