An Accurate and Efficient Frontal Solver for Fully-Coupled Hygro-Thermo-Mechanical Problems*

Mauro Bianco[1], Gianfranco Bilardi[1], Francesco Pesavento[2], Geppino Pucci[1], and Bernhard A. Schrefler[2]

[1] Dipartimento di Elettronica e Informatica, Università di Padova, Padova, Italy.
{bianco1,bilardi,geppo}@dei.unipd.it
[2] Dipartimento di Costruzioni e Trasporti, Università di Padova, Padova, Italy.
{pesa,bas}@caronte.dic.unipd.it

Abstract. Solving fully-coupled non-linear hygro-thermo-mechanical problems relative to the behavior of concrete at high temperatures is nowadays a very interesting and challenging computational problem. These models require an extensive use of computational resources, such as main memory and computational time, due to the great number of variables and the numerical characteristics of the coefficients of the linear systems involved.
In this paper a number of different variants of a frontal solver used within HITECOSP, an application developed within the BRITE Euram III "HITECO" EU project, to solve multiphase porous media problems, are presented, evaluated and compared with respect to their numerical accuracy and performance.
The final result of this activity is a new solver which is both much faster and more accurate than the original one. Specifically, the code runs over 5 times faster and numerical errors are reduced of up to three order of magnitude.

1 Introduction

Many successful methods exist for the solution of algebraic equations arising from the discretization of uncoupled problems. For coupled problems, especially when several fields are involved, the problem is still open. We concentrate here on a particular coupled multi-physics problem which deals with concrete under high temperature conditions [1, 2]. Such a model allows for instance to make residual lifetime predictions in concrete vessels of nuclear reactors or to predict the behavior of concrete walls in tunnel fires etc., [1–3]. The model has been implemented in the computer code HITECOSP in the framework of the BRITE Euram III "HITECO" [4] research project. This software uses a frontal technique to solve the final system resulting from the finite element (FE) implementation of the model. The aim of our work has been to improve the efficiency of HITECOSP's frontal solver in terms of performance as well as numerical accuracy, by to exploiting the various characteristics imposed by the model.

Improvements in term of performance have been obtained implementing a number of code optimizations, discussed in Section 2.2, with respect to original HITECOSP program.

In order to improve numerical accuracy in solving the linear systems produced by the Newton-Raphson like procedure used to solve the nonlinear systems arising from the FE implementation of the model, several pivoting strategies have been implemented

* This work was supported, in part, by MURST of Italy within the framework of the Centre for Science and Application of Advanced Computation Paradigms of the University of Padova.

P.M.A. Sloot et al. (Eds.): ICCS 2002, LNCS 2329, pp. 733–742, 2002.

and evaluated based on a modified component-wise backward error analysis (see Section 3). From our analysis it follows that the best strategy in terms of accuracy is also the best in terms of performance. In particular we have noticed errors of the same order of magnitude of the roundoff unit error and a further speed-up with respect to the optimized version of the original solver.

The rest of the paper is organized as follows. Section 2 deals with the frontal methods and the various code optimizations we have introduced and the pivoting strategies implemented. The metrics adopted to evaluate accuracy and performance are described in Section 3. Finally, the test cases used to evaluate our solvers are described in Section 4 while our results are shown in Section 5.

2 Frontal Method: Overview and Implementation

The large linear systems produced by the Newton-Raphson used in HITECOSP, are solved through the *frontal method* (see [5, 6] for a full description). The frontal method solves a linear system by working, at each step, only on a portion of the matrix (called *frontal matrix*), hence it is useful in those situations where core memory becomes the critical resource. The method works by sequentially executing two phases on each element of the finite element grid: an *assembly* phase and an *elimination* phase. During the assembly phase, the frontal matrix is augmented by the appropriate number of columns and rows relative, respectively, to the variables associated to the element and the equations containing those variables, and the matrix entries are updated to account for the new element. An entry becomes *fully-summed* if it will not receive further updates in any subsequent assembly phase. A column (resp., row) becomes fully-summed when all its entries become fully summed. A variable corresponding to a fully-summed column is also said fully-summed.

During the elimination phase, Gaussian elimination is applied to the frontal matrix, choosing the pivot in the block at the intersection of fully-summed rows and fully-summed columns. At each Gaussian elimination step, the pivot row is *eliminated*, i.e., it is stored somewhere into memory (typically onto a disk, if the problem is too large to fit in main memory). After the last elimination phase, back substitution on the reduced linear system is executed.

2.1 Pivoting Strategies

Recall that in the frontal method only a part of the matrix of the system is available at any given time, hence any pivoting strategy must be adopted to cope with this scenario. In particular, the pivot must always be chosen among those entries of the frontal matrix which reside in fully-summed rows and fully-summed columns.

Many strategies have been developed either to speed up the frontal solver or to improve its numerical stability. In this section we describe those strategies which we have implemented in order to find a solution that achieves the best trade-off between stability and performance for our particular physical problems.

Let A be the frontal matrix in a given elimination step. *Numerical pivoting* [6] entails choosing the pivot among the entries a_{ij} residing in fully-summed columns such that

$$|a_{ij}| \geq \alpha \max_k |a_{ik}|, \tag{1}$$

where $0 < \alpha \leq 1$ is a numerical constant and i is the index of a fully-summed row. Numerical pivoting was adopted to reduce the approximation error introduced by Gaussian elimination for the frontal method.

If an eligible pivot is not found, then the next element is assembled and a new search for a pivot is performed. This strategy is called *postordering for stability* [6]. The algorithm clearly terminates since eventually all the rows and all the columns of the frontal matrix will become fully-summed.

Often it is claimed in the literature that postordering for stability affects performance only slightly, while numerical stability is substantially increased. However, our experiments show that this is not the case for our physical problems. The failure of postordering in our scenario seems to be primarily due to the fact that the matrix entries in our mixed physical problems can be up to 40 orders of magnitude apart. Moreover, when a pivot is chosen, during the elimination step, fill-ins are produced in other rows, thus creating bonds among variables that do not belong to neighboring elements in the finite element mesh. Often these bonds prevent a row to be eliminated since the entries in its fully summed columns do not satisfy condition (1). In the next elimination steps this row continues to be filled with additional nonzero entries, hence the likelihood that it will not be chosen for elimination keeps on increasing, in a sort of "positive feedback" effect. Indeed we have observed extreme cases where rows entered in the frontal matrix at the very beginning of the solver's activity remains in the matrix until the very end. This phenomenon introduces two problems: not only does it cause the frontal matrix to grow inordinately, slowing down the program, but also worsens the numerical stability of the method, since a row that is present for a long time in the frontal matrix will sustain many operations on it, which is likely to amplify accumulation errors.

Another popular pivoting strategy is known as *minimum degree* [7, 6]. This strategy was proposed as a greedy way to reduce fill-ins when performing Gaussian elimination and was proved to be suited for symmetric, positive-definite matrices. Under minimum degree, the pivot is chosen as the diagonal element of a row with the minimum number of entries. Under the frontal method, the minimum degree strategy may be applied to the frontal matrix, choosing the pivot on the diagonal entry of the row with minimum number of entries in the block formed by the intersection between fully summed rows and fully summed columns. Since the full matrix of our systems has a symmetric structure, choosing pivots on the diagonal also preserves this symmetry inside the frontal matrix, allowing the data structures to be simplified.

It has to be remarked that the minimum degree strategy does not make any numerical consideration on the chosen pivot and it was originally developed for matrices that do not need such numerical precautions, e.g., positive definite symmetric matrices. Although our matrices, featuring great differences between numerical values of their entries, appear to be unsuitable for an application of minimum degree pivoting, our experiments have shown that the strategy is an effective way of reducing the accumulation error caused by postordering, perhaps due to the fact that it substantially reduces the amount of floating point operations. Indeed, a careful implementation of the minimum degree strategy has proven to feature both excellent performance and numerical accuracy for our problems.

The original application, HITECOSP, from which this work started, uses the following hybrid strategy. Before the pivot is chosen, if the absolute value of the previous pivot is less than a fixed numerical threshold value (10^{-4} in our cases), then the fully-summed rows are normalized so they will contain only values included between -1 and 1. After that, the pivot is chosen as the entry with the maximum absolute value among those in the intersection between fully-summed rows and columns. No postordering is performed. This strategy seems to work well for our physical problems. Namely, it exhibits good numerical accuracy and lends itself to an efficient implementation, which however requires a complete redesign of the relevant data structures.

2.2 Our Solutions

In this section we describe the frontal solvers which we have implemented. Each variant is characterized by a short name (in parentheses) which suggests the specific pivoting strategy adopted by the variant. Our first intervention has aimed at improving performance of the HITECOSP software (`HIT`) by providing an optimized implementation of its solver. In particular, the greatest improvement in performance has been obtained by the redesign of the main data structures in order to reduce at the minimum the number of linear searches inside arrays.

Another important issue that has been considered in redesigning HITECOSP's solver is the enhancement of the temporal and spatial locality of the memory accesses performed. To this purpose, extensive cache optimization has been applied, such as performing operations (e.g., row elimination, pivot search, etc.) within the solver by column rather that by row, in order to exploit the column major allocation of matrices featured by the FORTRAN compiler.

Another important source of performance enhancement has come from conditional branch optimization. As an example, consider that when computing on a sparse matrix, many operations are performed to no effect on zero entries (e.g., divisions and multiplications). However in most modern microprocessor architectures (and, in particular, on the ALPHA platform where our experiments run), keeping these operations improves performance since they take less cycles than those necessary by the processor to recover from a mispredicted branch. Indeed, conditional branch elimination in HITECOSP has improved the performance of the resulting code of up to 20%. This version of the frontal solver, implementing the same pivoting strategy as HITECOSP, named `BASE`, exhibits rather good performance.

The above code optimizations have also been employed to speed up the execution of the other solvers developed within our study. However, the main justification for designing new solvers mainly stems from our desire to compare the efficiency and numerical stability of the pivoting approach of HITECOSP with the other more established strategies described in the previous section. As a first step, basic numerical pivoting (as illustrated before) was implemented. Specifically we have developed a version without postordering that chooses as pivot the element that maximizes the value of α in (1) (`NUMPIV`), and another one implementing postordering (`NUMPPO`) which set $\alpha = 10^{-6}$ in (1).

Next, we have implemented the minimum degree strategy (`MINDEG`). This latter solver chooses the pivot on the diagonal of the frontal matrix and is endowed with recovery features when an entry in the diagonal is zero (however, this has never occurred in our experiments). A further optimization stems from the fact that, since the structure of the frontal matrix depends only on that of the finite element mesh, and, under the minimum degree strategy, the pivots depend only on the structure of the matrix, the pivotal sequence remains the same for all the executions of the frontal solver over the different iterations of the Newton-Raphson method (unless the chosen pivot is zero, which requires special care). Hence, it is possible to store the pivotal sequence during the first execution and to use it for the next ones, choosing the pivot, at each stage, with a single memory access. Our version of the minimum degree algorithm stores the pivotal sequence after the first execution of the frontal solver and uses it in the subsequent calls.

Finally, for the purpose of comparison, we have produced a further implementation (`HSL`) which uses the free version of the HSL (Harvell Subroutine Library) library [8]. Specifically, we have used the MA32 routine [9] that implements a frontal solver featuring a sophisticated numerical pivoting strategy. Version `HSL` gives us insight to compare our strategies against standard solutions available using third-party software.

3 Comparison Metrics

All the solver versions introduced in the previous section are evaluated in terms of their numerical stability (limited to the solution of the linear system) and performance. The next two sections discuss the metrics used to measure these two characteristics.

Evaluating numerical stability

Let $\tilde{x}$ be the computed (approximated) solution to one of the linear systems $Ax = b$ arising during the solution of a FEM problem. Approximation errors have been evaluated using a metric based on the *component-wise backward error* [10]

$$w = \min\{\varepsilon : (A + \Delta A)\tilde{x} = b + \Delta b, |\Delta A| \leq \varepsilon |A|, |\Delta b| \leq \varepsilon |b|\} \tag{2}$$

(The absolute values and the comparisons are intended to be component-wise). It can be proved (see [10, 11]) that, setting $0/0 = 0$ and $\xi/0 = \infty$ if $\xi \neq 0$, w can be computed as

$$w = \max_i \frac{|r_i|}{(|A|\ |\tilde{x}| + |b|)_i} \tag{3}$$

Intuitively, w measures the minimum variation that the matrix of the system and the right hand side vector should sustain to obtain the approximated $\tilde{x}$ solution. There is evidence in the literature that the component-wise backward error is more sensitive to instability than other metrics [12].

In this work, in order to have a more detailed description of the approximation errors introduced by our solvers, we have refined the component-wise backward error metric as follows. Let a_i denote the i-th row of A. Define the *ith equation error* to be

$$w_i = \min\{\varepsilon : (a_i + \Delta a_i)\tilde{x} = b_i + \Delta b_i, |\Delta a_i| \leq \varepsilon |a_i|, |\Delta b_i| \leq \varepsilon |b_i|\}. \tag{4}$$

Value w_i gives a measure of "how well" vector $\tilde{x}$ satisfies the ith equation of the system and can be readily observed as follow.

Theorem 1. *Let v be a vector with $v_i = \frac{r_i}{(|A|\ |\tilde{x}|+|b|)_i}$. Then $w_i = v_i$.*

Proof. Let $\Delta\hat{a}_i$ and $\Delta\hat{b}_i$ be a pair of minimal perturbations associated to w_i. Since $|\Delta\hat{a}_i| \leq w_i|a_i|$ and $|\Delta\hat{b}_i| \leq w_i|b_i|$, we have that

$$|r_i| = |b_i - a_i\tilde{x}| = |\Delta\hat{a}_i\tilde{x} - \Delta\hat{b}_i| \leq |\Delta\hat{a}_i||\tilde{x}| + |\Delta\hat{b}_i| \leq w_i(|a_i||\tilde{x}| + |b_i|),$$

whence $w_i \geq |v_i|$. Also, from the definition of v it follows that $r_i = v_i(|a_i||\tilde{x}| + |b_i|)$. Define now $\Delta a'_i = v_i|a_i|\text{diag}(\text{sign}(\tilde{x}))$ and $\Delta b'_i = -v_i|b_i|$. It is easy to see that $(a_i + \Delta a'_i)\tilde{x} - (b_i + \Delta b'_i) = 0$. Therefore, since $|\Delta a'_i| = |v_i||a_i|$ and $|\Delta b'_i| = |v_i||b_i|$, it follows that $w_i \leq |v_i|$, and the theorem follows.

It is easy to see that $\Delta\hat{A} = \text{diag}(v)|A|\text{diag}(\text{sign}(\tilde{x}))$ and $\Delta\hat{b} = -\text{diag}(v)|b|$ are such that $|\Delta\hat{A}| \leq w|A|$, $|\Delta\hat{b}| \leq w|b|$ and $(A + \Delta\hat{A})\tilde{x} - (b + \Delta\hat{b}) = 0$. Moreover, the above theorem proves that all the perturbations are the minimum possible, in the sense indicated by (4). Hence, vector v provides a readily obtainable indication about the minimum perturbation that each equation should sustain to obtain the approximated solution. In particular any element of v can be compared against the roundoff unit error to gain immediate appreciation of the significance of the corresponding perturbations. Finally, observe that the standard component-wise error metric w can be obtained as $w = \| v \|_\infty$.

A plot of vector v (using the equation indices as the abscissae) can be used to ascertain whether numerical errors tend to affect some groups of equations more than others. We feel that this is particularly useful in multi-physics applications as the ones treated in this paper.

All the metrics described above have been collected over several iterations of each solver. No significant variation of each metric has been observed over the different iterations. However, in what follows, we report the maximum errors encountered on each test case.

Measuring performances
Performance is measured both in terms of computational time and rate of floating point operations (Mflops) relatively to the frontal solver only.

4 Test Cases

The various solver versions have been executed on a number of test cases arising in several practical scenarios and characterized by an increasing complexity of the underlying physical system. As for the solver version, each test case is indicated by a short name (in parentheses):

1. *small column* (*smcol*): a regular 10×10 mesh of 100 elements in which all the degrees of freedom, except for the ones related to displacements, are set to zero.
2. *wall* (*wall*): 69 elements lined up in a row where the fifth degree of freedom (y-displacement) is fixed to zero;
3. *container* (*cont*): 288 elements outlining a container;
4. *column* (*col*): a square section of a column made of a 20×20 mesh of 400 elements;
5. *big column* (*bigcol*): like *column* but made of a 25×25 mesh of 625 elements;

Such a variety of test cases allows us to evaluate the behavior of the solver variants when the complexity of the physics behind the problem to be solved varies, from simpler (*smcol*) to harder (*bigcol*).

5 Results

The solver versions shown above have been tested on an Alpha workstation which uses a 21264 processor clocked at 666Mhz with two floating point units that make it capable of a 1354Mflops peak performance

The next two subsections report the results obtained by evaluating the various versions of the solvers described above. In section 5.1 we analyze the numerical stability properties exhibited by the solvers with respect to the metrics discussed in section 3. In section 5.2 we examine the performance achieved by the solvers.

5.1 Numerical Quality

Table 1 reports the component-wise backward error analysis (2) for all the solver versions and test cases described before. We note that `MINDEG` exhibits the least errors (in order of magnitude), scaling extremely well as the physical problems become more complex. Also, the table shows that `HIT` and `BASE` do not exhibit exactly the same errors, with `BASE` featuring slightly larger errors. This is explained by the fact that `HIT` uses some extra heuristic precautions to reduce fill-ins. We have chosen not to implement these expedients in `BASE`, since they complicate the code while not providing significant improvements in term of either accuracy or performance.

Table 1. Component-wise backward errors exhibited by the various solvers for each test cases.

	HIT	BASE	NEWPIV	NEWPPO	MINDEG	HSL
smcol	$3 \cdot 10^{-16}$	$3 \cdot 10^{-16}$	$3 \cdot 10^{-16}$	$3 \cdot 10^{-16}$	$3 \cdot 10^{-16}$	$4 \cdot 10^{-16}$
wall	$4 \cdot 10^{-14}$	$1 \cdot 10^{-13}$	$1 \cdot 10^{-11}$	$3 \cdot 10^{-6}$	$4 \cdot 10^{-16}$	$1 \cdot 10^{-6}$
cont	$6 \cdot 10^{-13}$	$5 \cdot 10^{-13}$	$6 \cdot 10^{-11}$	$2 \cdot 10^{-4}$	$6 \cdot 10^{-16}$	$5 \cdot 10^{-3}$
col	$4 \cdot 10^{-12}$	$2 \cdot 10^{-12}$	$2 \cdot 10^{-5}$	$2 \cdot 10^{-3}$	$9 \cdot 10^{-16}$	$4 \cdot 10^{-4}$
bigcol	$2 \cdot 10^{-12}$	$6 \cdot 10^{-13}$	$7 \cdot 10^{-5}$	$5 \cdot 10^{-3}$	$1 \cdot 10^{-15}$	$5 \cdot 10^{-5}$

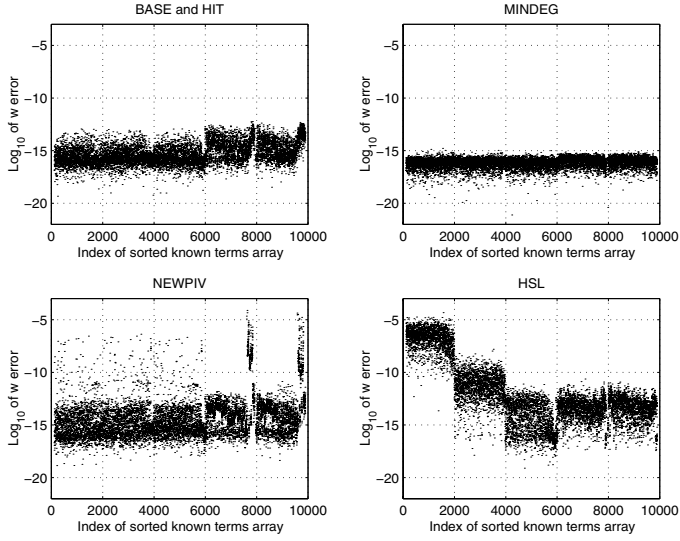

Fig. 1. Plot of the component-wise errors for the *bigcol* test case, exhibited by the various solvers. The abscissae are the equations indices and the ordinates are the base 10 logarithms of the errors.

If we compare NEWPIV with NEWPPO, implementing, respectively, numerical pivoting without and with postordering, we see that the errors exhibited by the latter (in all cases except *smcol*) are orders of magnitude worse than those exhibited by the former. Indeed, for the two largest test cases, the errors exhibited by NEWPPO in the solution of the linear systems became so large to prevent the Newton-Raphson method from converge. This provides numerical evidence that postordering does not achieve its intended purpose in our physical scenario.

To achieve a more profound understanding on the numerical behavior of the implemented variants, in Figure 1, we show all the components of the v vector (see sec. 3) for different solvers running the *bigcol* test case, plotted (in logarithmic scale) against their respective indices. For our specific physical problems, we have that indices between 1 and 2000 are relative gas pressure equations, indices between 2001 and 4000 are relative to capillary pressure, indices between 4001 and 6000 are relative to temperatures, and, finally, the remaining indices are relative to displacement equations. It is interesting to observe how errors tend to cluster according to the type of equation.

It is clear from the figure that MINDEG exhibits extremely homogeneous errors that are all close to the roundoff unit error (which is about 10^{-16} for our machine), while BASE, although still behaving quite well, tends to show a more varied range of errors, which implies that different equations (corresponding to different physical constraints) are solved with different degrees of accuracy. As for NEWPIV, we can see a rather good behavior on average, but the plot highlights a non-negligible set of outliers with high

Table 2. Times and floating point operations rates (frontal solver only). Note the $+\infty$ entries in the NEWPPO columns are related to that test cases for which the method does not converge as the errors become too large.

		HIT	BASE	NEWPIV	NEWPPO	MINDEG	HSL
wall	Time (s)	4.6	2.23	2.48	1.96	1.07	4.48
	Mflops	59.54	109.18	134.15	163.08	161.34	231.73
smcol	Time (s)	6.05	1.98	2.12	1.83	1.64	6.97
	Mflops	50.28	164.3	168.94	179.05	180.45	201.91
cont	Time (s)	77.5	27.5	30.4	790.0	22.9	207.1
	Mflops	79.63	237.63	232.06	98.4	262.02	206.47
col	Time (s)	314.1	116.8	124.5	3212	96.5	1230.5
	Mflops	54.08	224.09	222.62	7.89	256.93	154.76
bigcol	Time (s)	1276.4	260.1	274.0	$+\infty$	225.9	1609.5
	Mflops	30.58	232.53	226.6	–	261.77	137.47

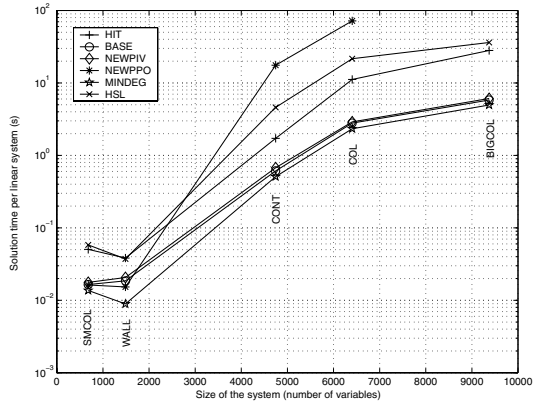

Fig. 2. Time taken by the various solvers on a single linear system as a function of the size of the system itself.

values of the error w. Finally, looking at the plot for HSL, we may note how it behaves very poorly for two entire groups of equations (especially those related to pressures) while it is only slightly worse that BASE for the other two groups.

5.2 Performances

Table 2 shows the execution times and the floating point operation rates (Mflops) exhibited by the frontal solvers. Each test case involves several time steps, with each time step in turn requiring the solution of a number of linear systems, one per Newton-Raphson iteration. More specifically, the number of linear systems solved in each test case is about 120 for *wall* and *smcol*, 45 for *cont* and *col*, and 30 for *bigcol*. For the last three test cases, the most computationally intensive ones, we have that the solver accounts for about 75% of the total time for MINDEG, and goes up to 95% for HIT. This fact justifies our focusing on the solver only, rather than other parts of the program, such as the numerical integration routines.

In Figure 2 we plot the time taken by the various solvers on a single linear system as a function the size of the system itself. We want to remark that the size of the system is not, however, the only parameter that affects performance, since, depending on the

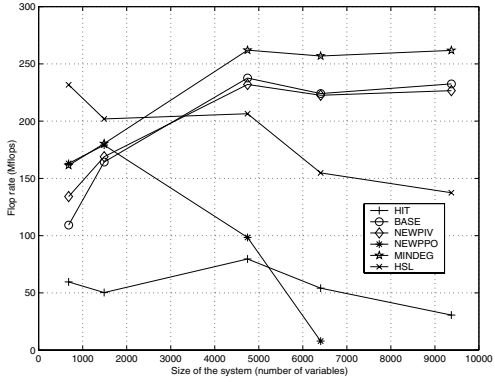

Fig. 3. Floating point operations rates exhibited by the solvers varying the size of the systems resolved.

solver used, other peculiarities may influence execution times, such as, for example, the shape of the mesh.

Looking at Table 2, we note that MINDEG exhibits the by far best performance. Together with the numerical stability data presented in the previous section, this implies that MINDEG achieves both performance and accuracy at the same time, which is somewhat surprising since an increase in accuracy often comes at the expense of a deterioration of performance.

Unlike HIT and HSL, MINDEG, BASE, and NEWPIV seem to be able to sustain high flops rate when the size of the problem increases (see fig. 3). Such scalability rewards our redesign of the data structures, which affords access times to data which are independent of the amount of the data itself. Comparing BASE against HIT, we see that redesigning the data structures, optimizing data accesses and carefully eliminating conditional branches alone made the solver more than 4 times faster. Changing the pivoting strategy yielded an extra time saving: indeed, MINDEG is more than 5.6 times faster than HIT for the *bigcol* test case.

Looking at the numerical pivoting strategies, we note that HSL exhibits a rather unsatisfactory performance. This seems to be mainly due to postordering. Comparing NEWPPO to NEWPIV, for the test cases for which both strategies converge, we note that when postordering is extensively used (as for *cont* and *col* test cases), the execution time explodes. This can be explained by the feedback effect described in section 2.1 and due to postordering. It has to be remarked that HSL behaves quite well for the *wall* and *smcol* test cases, for which it exhibits the highest flops rate. In fact, when the physical problem is simple, HSL becomes competitive. However, as the problem becomes more complex, the time performance of HSL degrades, even though its floating point operations rate remains quite high. As a bottomline we can say that HSL features a very good implementation (high flops rate), but its pivoting strategy, however, turns out to be a poor choice for our physical problems.

Going back to NEWPIV and NEWPPO, we note that for the *wall* and *smcol* test cases, the latter exhibits better execution times than the former. This is due to the fact that, while NEWPIV maximizes α in equation (1), NEWPPO simply picks the first element which satisfies (1) for a fixed $\alpha = 10^{-6}$. This strategy proves to be beneficial for performance since fewer entries of the frontal matrix need to be scanned. The gain in performance is however limited to those simple cases where postordering is rarely applied. We have chosen not to pick the best possible pivot when implementing postordering since we have observed that otherwise some rows would remain longer in the frontal matrix, which has detrimental effects on both time performance and accuracy.

6 Conclusion

When solving very non linear and strictly coupled physical problems where there may be many orders of magnitude among the numerical values involved, our experiments suggest that the best strategy is to strive for simplicity. Indeed, the `MINDEG` version of the solver does not do any numerical consideration when choosing the pivot in Gaussian elimination, but only structural ones. Yet, this suffices to get excellent performance and good accuracy.

Redesigning the data structures and performing code optimizations has proved to be the most effective way to speed-up the program considering that `BASE` achieves an improvement of a factor 4 with respect to `HIT`. A further improvement is then obtained by simplifying the pivoting strategy.

A possible further improvement in performance, that we mean to investigate, is to find the right tradeoff between avoiding linear searches inside the arrays and limiting indirect addressing. Specifically, observe that data structures designed to avoid linear searches make large use of indirect addressing, which, however, may disrupt temporal locality and slow down the algorithm by forcing the processor to wait for the data to become available from main memory.

References

1. D. Gawin, C. E. Majorana, F. Pesavento and B. A. Schrefler. A fully coupled multiphase FE model of hygro- thermo- mechanical behaviour of concrete at high temperature. In Computational Mechanics., Onate, E. & Idelsohn, S.R. (eds.), New Trends and Applications:, Proc. of the 4th World Congress on Computational Mechanics, Buenos Aires 1998; 1–19. Barcelona: CIMNE, 1998.
2. D. Gawin, C. E. Majorana and B. A. Schrefler. Numerical analysis of hygro-thermic behaviour and damage of concrete at high temperature. In Mech. Cohes.-Frict. Mater. 1999; 4:37–74.
3. D. Gawin, F. Pesavento and B. A. Schrefler. Modelling of hygro-thermal behaviour and damage of concrete at temperature above the critical point of water. Submitted for publication.
4. BRITE Euram III BRPR-CT95-0065 1999 "HITECO". Understanding and industrial applications of High Performance Concrete in High Temperature Environment, Final Report, 1999.
5. B. M. Irons. A frontal solution program for finite element analysis Int. J. Numer. Meth. Engng, 1970; 2:5–32.
6. I. S. Duff, A. M. Erisman and J. K. Reid. *Direct Methods for Sparse Matrices*, Clarendon Press, 1986.
7. W. F. Tinney and J. W. Walker. Direct solutions of sparse network equations by optimally ordered triangular factorization, Proc. IEEE 55, 1967; 1801–1809,
8. HSL (Formerly the Harwell Subroutine Library). http://www.cse.clrc.ac.uk/Activity/HSL
9. I. S. Duff and J. K. Reid. MA32 - a package for solving sparse unsymmetric systems using the frontal method, *Report R.10079*, HMSO, London, 1981.
10. W. Oettli and W. Prager. Compatibility of Approximate Solution of Linear Equations with Given Error Bounds for Coefficients and Right-Hand Sides, *Numerische Mathematik*, 1964; 6:405–409.
11. N. J. Higham. How accurate is Gaussian elimination? In D. F. Griffiths and G. A. Watson, editors, *Numerical Analysis 1989, Proceedings of the 13th Dundee Conference*, volume 228, pages 137–154, Essex, UK, 1990. Longman Scientific and Technical.
12. N. J. Higham. Testing Linear Algebra Software, in R. F. Boisvert, editor, *Quality of Numerical Software: Assessment and Enhancement*, pages 109-122. Chapman and Hall, London, 1997
13. I. S. Duff and J. A. Scott. The use of multiple fronts in Gaussian Elimination, *Technical Report RAL-TR-94-040*, Department for Computation and Information, Rutherford Appleton Laboratory, September 1994.