Numerical comparison of solvers for linear systems from the discretization of scalar PDEs

Yvan Notay [*]

Service de Métrologie Nucléaire
Université Libre de Bruxelles (C.P. 165/84)
50, Av. F.D. Roosevelt, B-1050 Brussels, Belgium.
email : ynotay@ulb.ac.be

January 23, 2012

1 General setting

All tests were performed on a computer with two Intel XEON L5420 processors at 2.50 GHz and 16 Gb RAM memory, running 64 bit Linux OS.

All solvers were called from the Matlab environment (version 7.6.0.324, release R2008a), the Matlab process being the only computing process running on the machine. However, all solvers are based on efficient codes written in Fortran or C. When available, we used the gateway (function callable from the Matlab environment) provided with the package; otherwise, we proceeded as indicated in the next section.

The timings are those obtained using the Matlab command *tic* just before calling the solver and the command *toc* just after. Times are reported per million of unknowns; i.e., letting n be the matrix size, what is plotted for each problem and method is $toc{*}1e6/n$ as a function of n.

For iterative solvers, the zero vector was used as initial approximation, and iterations were stopped when the relative residual error was below 10^{-6}. In each case, we checked that the solution returned to the Matlab environment was indeed within the prescribed tolerance. Note that since the stopping criterion in ILUPACK is based on different principles (backward error estimate), we adjusted for this method the tolerance on a trial and error basis so as to reach the same accuracy as with other solvers.

[*]Supported by the Belgian F.R.S.-FNRS ("Directeur de recherches")

2 Methods

- **AGMG**, the aggregation-based algebraic multigrid method described in [7] and further enhanced in [5, 8]. We used the public domain implementation [6], version 3.1.2, using the gateway provided with the package for Linux 64 bit architecture (it was compiled with Matlab mex based on gcc 4.5.0 and g95 0.92, using BLAS and LAPACK provided with Matlab (mwlapack, mwblas)). Default parameters were used, except that *restart* was set to 1 in the symmetric positive definite (SPD) cases to tell the solver that the conjugate gradient method can be used. That is, the calling sequences were *u=agmg(A,b,1)* in the SPD cases and *u=agmg(A,b)* in the other ones.

- **AMG(Hyp)**, the classical AMG method [9] as implemented in the Hypre library [3], version 2.7.0 (in sequential). This method is also known as *Boomer AMG* incorporates several recent enhancements [1]. We used it as a preconditioner for the conjugate gradient method in the SPD cases and for GMRES in the other ones, using again the Hypre implementation. Default parameters were used in all cases. Since no Matlab gateway is provided, we built a simple one similar to the one used for AGMG, compiling it with the same compilers and options.

- **AMG(HSL)**, the HSL implementation [2] of the classical AMG method as described in [10]. We used more particularly the *hsl_mi20* routines, version 1.4.0. No Matlab gateway is provided, and this package appears more as offering a preconditioner than a complete solution toolbox, although there are some iterative solvers provided with the HSL library. Hence we integrated the *hsl_mi20* routines in the software infrastructure of AGMG, just substituting the call for setup and the call for preconditioner application for those provided by respectively *mi20_setup* and *mi20_precondition* (using default options in all cases). Hence the iterative solvers and the gateway to Matlab were the same as for AGMG (i.e., the conjugate gradient method in the SPD cases and GCR(10) in the other ones); we also used the same compilers with same options.

- **ILUPACK**, a package implementing an efficient threshold-based ILU preconditioner and several iterative methods [4]. We used version 2.2 with the gateway provided with the package for Linux 64 bit architecture. We used default parameters; that is, *[PREC, options]=AMGfactor(A)* and *[u,options]=AMGsolver(A,PREC,options,b)*, with just in between the two calls a correction of *options.restol* on a case by case basis, to make the stopping criterion similar to the one used for other iterative solvers.

- **Matlab \**, the sparse direct solver available as built-in function in Matlab, and which is actually based on UMFPACK [12]. As indicated above, we used Matlab version 7.6.0.324, release R2008a.

3 Test problems

Poisson 2D, FD
Five point finite difference discretization with uniform mesh size of

$$-\Delta u \;=\; 1 \quad \text{in} \qquad \Omega = (0,1)\times(0,1)$$

with Dirichlet boundary conditions $u = 0$ everywhere on $\partial\Omega$.

Laplace 2D, FE(p3)
Cubic (p3) finite element discretization with uniform mesh of

$$-\Delta u \;=\; 0 \quad \text{in} \qquad \Omega = (0,1)\times(0,1)$$

with Neumann boundary conditions $\frac{\partial u}{\partial n} = 0$ everywhere on $\partial\Omega$,
except on $y = 1$, $0 \le x \le 1$, where Dirichlet conditions $u = x$ were prescribed.
In this example, 33% of the nonzero offdiagonal entries are positive.

Poisson 2D, L-shaped, FE, Unstructured
Linear finite element discretization of

$$-\Delta u \;=\; 1 \quad \text{in} \qquad \Omega = (-1,1)\times(-1,-1)\backslash\big((0,1)\times(0,1)\big)$$

with Dirichlet boundary conditions $u = r^{\frac{2}{3}}\sin(\frac{2\theta}{3})$ everywhere on $\partial\Omega$, where (r,θ) is the polar coordinate representation of (x,y).
The mesh is unstructured with simplex size progressively decreased near the reentering corner at $(0,0)$, in such a way that the mesh size in its neighborhood is about 10^4 times smaller.

Convection-Diffusion 2D, FD
Five point finite difference discretization (upwind scheme) of

$$-10^{-6}\,\Delta u \;+\; \overline{v}\,\overline{\nabla} u = 0 \quad \text{in} \qquad \Omega = (0,1)\times(0,1)$$

with convective flow given by

$$\overline{v}(x\,,\,y) = \begin{pmatrix} x(1-x)(2\,y-1) \\ -(2\,x-1)y(1-y) \end{pmatrix} ,$$

and Dirichlet boundary conditions $u = 0$ everywhere on $\partial\Omega$,
except on $y = 1$, $0 \le x \le 1$, where $u = 1$ was prescribed.

Poisson 3D, FD
Seven point finite difference discretization with uniform mesh size of

$$-\Delta u \;=\; 1 \quad \text{in} \qquad \Omega = (0,1)\times(0,1)\times(0,1)$$

with Dirichlet boundary conditions $u = 0$ everywhere on $\partial\Omega$.

Laplace 3D, FE(p3)
Cubic (p3) finite element discretization with uniform mesh of

$$-\Delta u = 0 \quad \text{in} \qquad \Omega = (0,1) \times (0,1) \times (0,1)$$

with Neumann boundary conditions $\frac{\partial u}{\partial n} = 0$ everywhere on $\partial\Omega$,
except on $z = 1$, $0 \leq x$, $y \leq 1$, where Dirichlet conditions $u = xy$ were prescribed.
In this example, 51% of the nonzero offdiagonal entries are positive.

Poisson 3D, FE, Unstructured
Linear finite element discretization of

$$-\Delta u = 1 \quad \text{in} \qquad \Omega = (-2.5, 2.5) \times (-2.5, 2.5) \times (-2.5, 2.5)$$

with Dirichlet boundary conditions $u = 1$ everywhere on $\partial\Omega$,
except on $x = 0,\, 1;\; 0 \leq y,\, z \;\leq 1$, were $u = 0$ was prescribed.
The mesh is unstructured with simplex size progressively decreased near the surface of a small sphere of diameter 0.2 at the center of the domain, in such a way that the mesh size in its neighborhood is about 10 times smaller.

Convection-Diffusion 3D, FD
Seven point finite difference discretization (upwind scheme) of

$$-10^{-6}\,\Delta u + \overline{v}\,\overline{\nabla} u = 0 \quad \text{in} \qquad \Omega = (0,1) \times (0,1) \times (0,1)$$

with convective flow given by

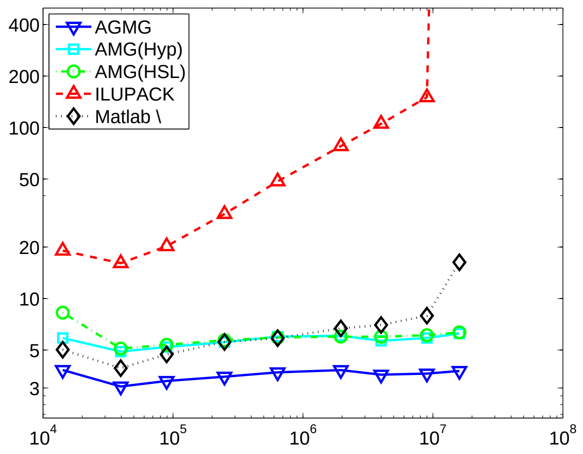
$$\overline{v}(x, y, z) = \begin{pmatrix} 2x(1-x)(2y-1)z \\ -(2x-1)y(1-y) \\ -(2x-1)(2y-1)z(1-z) \end{pmatrix}$$

and Dirichlet boundary conditions $u = 0$ everywhere on $\partial\Omega$,
except on $y = 1$, $0 \leq x$, $y \leq 1$, where $u = 1$ was prescribed.

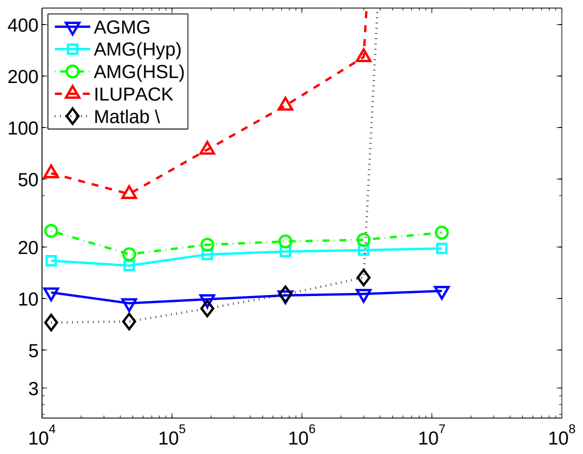
4 results

Pay attention that both scales are logarithmic.

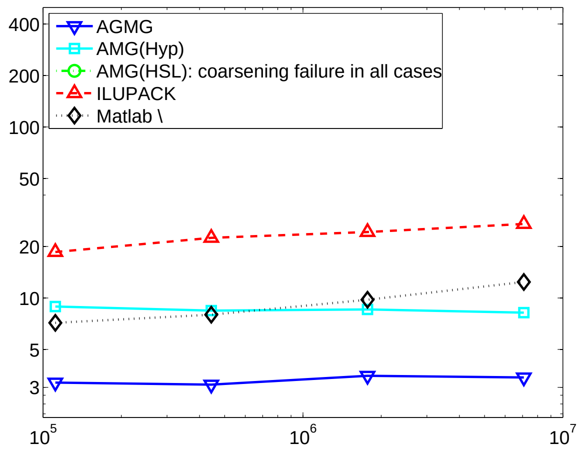
Poisson 2D, FD



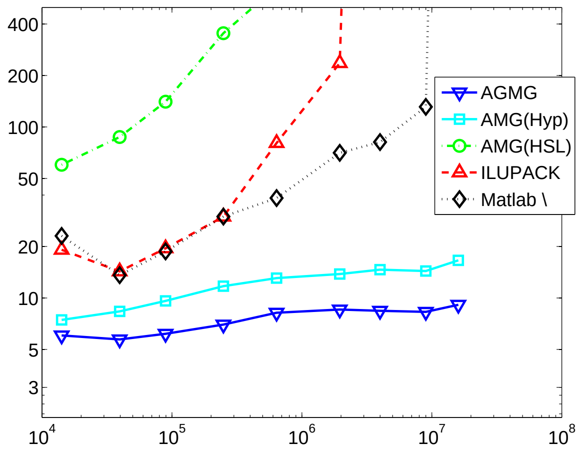
Laplace 2D, FE(p3)



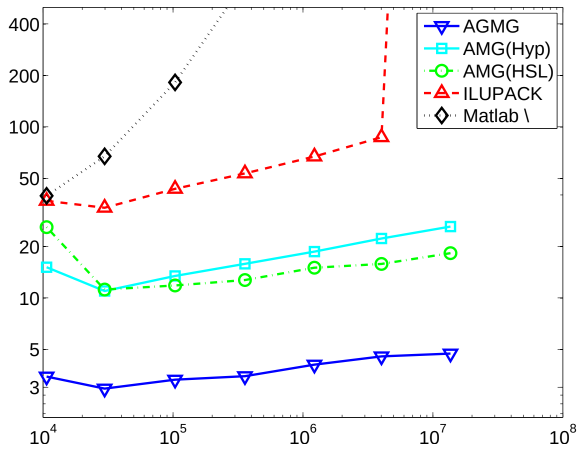
Poisson 2D, FE, Unstructured



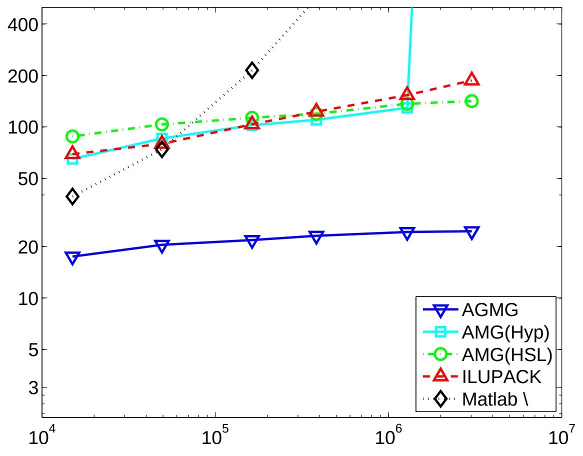
Convection-Diffusion 2D, FD

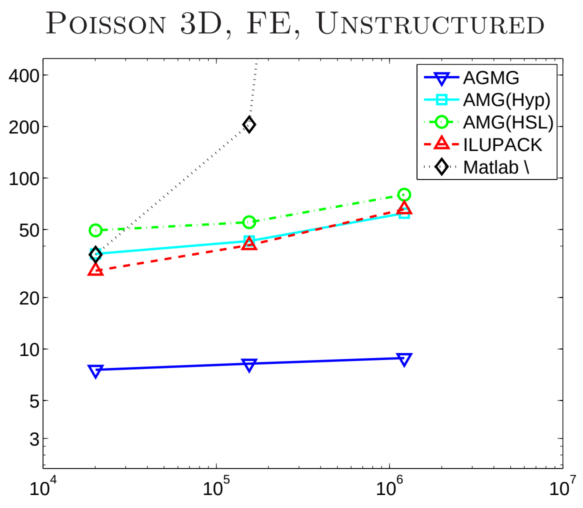


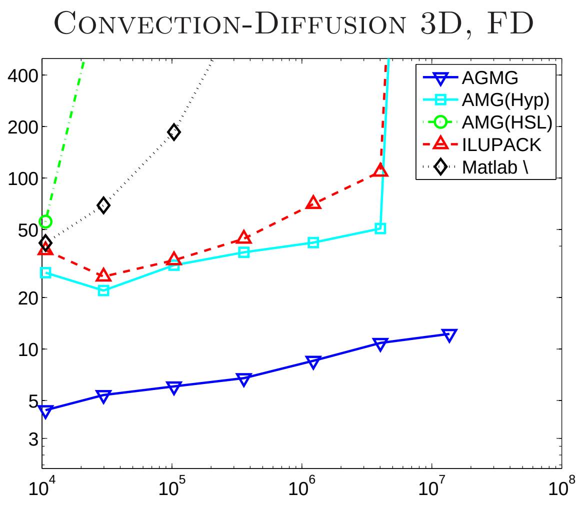
Poisson 3D, FD



Laplace 3D, FE(p3)







Acknowledgments

I thank Artem Napov for the installation of the ILUPACK and Hypre libraries.

References

[1] V. E. Henson and U. M. Yang, *BoomerAMG: A parallel algebraic multigrid solver and preconditioner*, Appl. Numer. Math., 41 (2002), pp. 155 – 177.

[2] *HSL (2011), a collection of fortran codes for large-scale scientific computation.* `http://www.hsl.rl.ac.uk`.

[3] *Hypre software and documentation.* Available online at `http://acts.nersc.gov/hypre/`.

[4] *ILUPACK software and documentation.* Available online at `http://ilupack.tu-bs.de`.

[5] A. Napov and Y. Notay, *An algebraic multigrid method with guaranteed convergence rate*, SIAM J. Sci. Comput., (2012). To appear; `http://homepages.ulb.ac.be/~ynotay`.

[6] Y. Notay, *AGMG software and documentation.* Available online at `http://homepages.ulb.ac.be/~ynotay/AGMG`.

[7] Y. Notay, *An aggregation-based algebraic multigrid method*, Electronic Trans. Numer. Anal., 37 (2010), pp. 123–146.

[8] Y. Notay, *Aggregation-based algebraic multigrid for convection-diffusion equations*, Tech. Rep. GANMN 11–01, Université Libre de Bruxelles, Brussels, Belgium, 2011. `http://homepages.ulb.ac.be/~ynotay`.

[9] J. W. Ruge and K. Stüben, *Algebraic multigrid (AMG)*, in Multigrid Methods, S. F. McCormick, ed., vol. 3 of Frontiers in Applied Mathematics, SIAM, Philadelphia, PA, 1987, pp. 73–130.

[10] K. Stüben, *An introduction to algebraic multigrid*, in Trottenberg et al. [11], 2001, pp. 413–532. Appendix A.

[11] U. Trottenberg, C. W. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, London, 2001.

[12] *UMFPACK software and documentation*. Available online at `http://www.cise.ufl.edu/research/sparse/umfpack/`.