Benchmarking of linear solvers : link between simulation and high performance computing

Anatole Vercelloni and Méline Trochon

Supervisors : Pierre Jolivet and Théo Mary

Contents

1 Introduction

2 The 2D Poisson equation

3 MUMPS

4 Benchmarks
- Symmetric permutation
- Block Low-Rank (BLR)
- Iterative Refinement

5 Conclusion

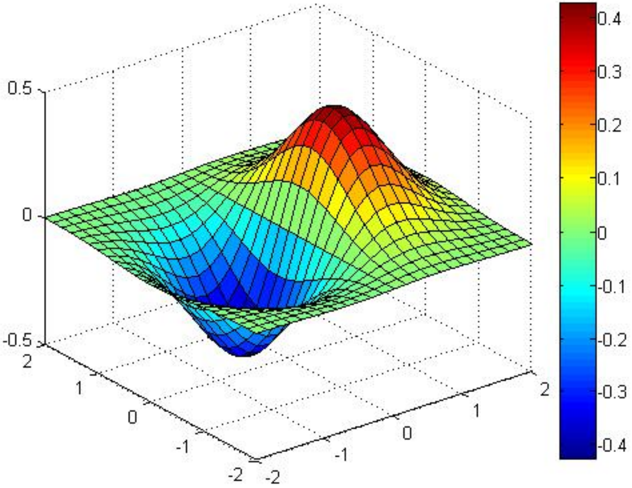


Figure: Solution of a 2D Poisson equation

The 2D Poisson equation

Let us consider the following problem:

$$\left\{ \begin{array}{l} find\, \varphi \in H^1(\Omega)\, s.t. \\ -\partial_x^2\varphi - \partial_y^2\varphi = f \text{ on } \Omega \\ \varphi = g \text{ on } \partial\Omega \end{array} \right. \tag{1}$$

Thanks to Taylor's expansion, we have :

$$\partial_x^2\varphi = \frac{\varphi_{i-1,j} - \varphi_{i+1,j} + 2\varphi_{i,j}}{h^2} \qquad with\ i, j = 1, .., n$$

Into a linear system

$$Ax = h^2 f \tag{2}$$

with A = $\begin{pmatrix} 4 & -1 & 0 & \cdots \\ -1 & 4 & -1 & \cdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & -1 & 4 \end{pmatrix}$ and f = $\begin{pmatrix} f_{0,0} & f_{1,0} & f_{2,0} & \cdots \\ f_{0,1} & f_{1,1} & f_{2,1} & \cdots \\ \vdots & \ddots & \ddots & \vdots \\ f_{0,n} & \cdots & f_{n-1,n} & f_{n,n} \end{pmatrix}$

Sparse linear system

We can remark that A is:

- symmetric (in some case)
- sparse
- low rank properties

The Matrices

- The 2D Poisson equation matrix:
  (1002001×1002001) with 7006001 nnz

- The 3D Poisson equation matrix:
  (1560896×1560896) with 23091886 nnz

Prerequisites

- Single and double precision (Smumps, Dmumps)
- ε-machine
- Scaled residual :

$$r = \frac{\|Ax - b\|_\infty}{\|A\|_\infty \|x\|_\infty} \tag{3}$$

MUltifrontal and Massively Parallel sparse and direct Solver (MUMPS)

- A direct solver for sparse matrices

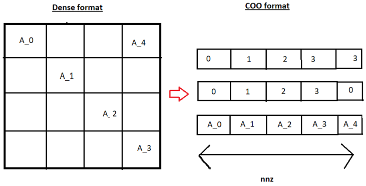


Figure: The format used in MUMPS

MUltifrontal and Massively Parallel sparse and direct Solver (MUMPS)

- A direct solver for sparse matrices
- Three different phases :
- Analysis phase

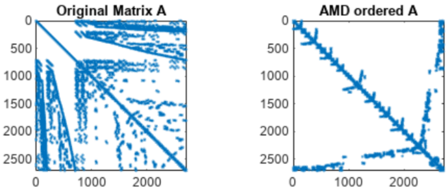


Figure: Illustration of a reordering method (AMD) to reduce the fill-in

MUltifrontal and Massively Parallel sparse and direct Solver (MUMPS)

- A direct solver for sparse matrices
- Three different phases :
- Analysis phase
- Factorizaton phase

MUltifrontal and Massively Parallel sparse and direct Solver (MUMPS)

- A direct solver for sparse matrices
- Three different phases :
- Analysis phase
- Factorizaton phase
- Solve and check phase

Symmetric permutation

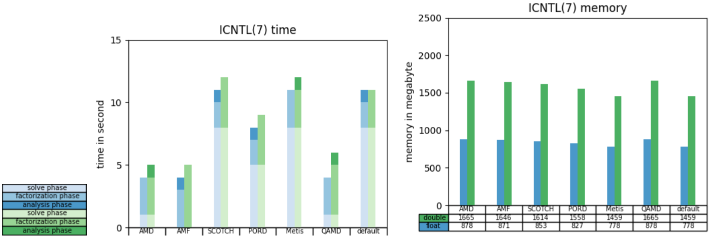
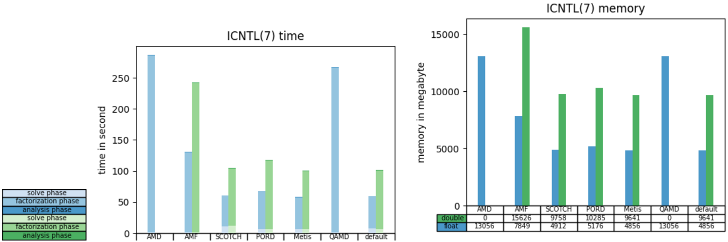


Figure: Time then Memory depending on the choice of ICNTL(7) for the 2D Poisson equation

Symmetric permutation



	AMD	AMF	SCOTCH	PORD	Metis	QAMD	default
double	0	15626	9758	10285	9641	0	9641
float	13056	7849	4912	5176	4856	13056	4856

Figure: Time then Memory depending on the choice of ICNTL(7) for the 3D Poisson equation

Block Low-Rank (BLR)

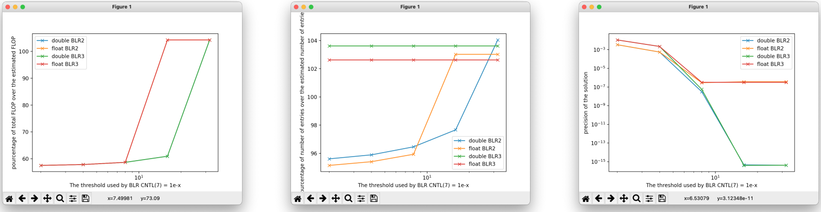


Figure: The number of operation, the memory space then the scaled residual depending on ϵ-BLR for the 2D Poisson problem

Block Low-Rank (BLR)

ϵ	1e-2	1e-4	1e-8
percentage of total over estimated FLOP	3.3	3.8	7.2
percentage of total over estimation entries	17.9	23.3	40.1
scaled residual	5.37e-3	5.53e-4	1.5e-7

Table: The percentage of FLOP and entries and the scaled residual for 3D Poisson equation

Iterative Refinement

repeat

Solve $A\Delta x = r$ using the computed factorization

$x = x + \Delta x$

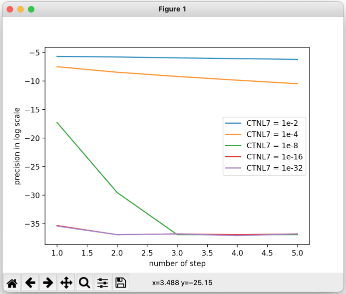
$r = b - Ax$

The computed backward error ω

until $\omega < \alpha$

Figure: Pseudo-code of the iterative refinement (for a positive parameter)

Iterative refinement



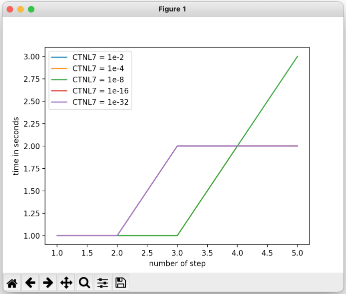


Figure: For 2D Poisson equation, the scaled residual then the time of the solve phase depending on the number of step

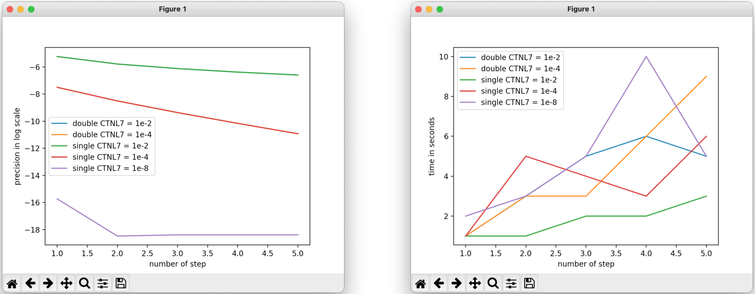


Figure: For 3D Poisson equation, the scaled residual then the time of the solve phase depending on the number of step

Conclusion

Let μ be the value set by the user, and ε the accuracy they want to reach.

- For 2D : AMD (ICNTL(7) = 0) instead of default
- For 3D : default value for ICNTL(7), and depending on the accuracy :

Conclusion

Let μ be the value set by the user, and ε the accuracy they want to reach.

- For 2D : AMD (ICNTL(7) = 0) instead of default
- For 3D : default value for ICNTL(7), and depending on the accuracy :
- $\varepsilon > \mu$: smumps, BLR (set to ε)

Conclusion

Let μ be the value set by the user, and ε the accuracy they want to reach.

- For 2D : AMD (ICNTL(7) = 0) instead of default
- For 3D : default value for ICNTL(7), and depending on the accuracy :
- $\varepsilon > \mu$: smumps, BLR (set to ε)
- 1e-16 < $\varepsilon < \mu$: smumps, BLR (set to μ) + iteration step

Conclusion

Let μ be the value set by the user, and ε the accuracy they want to reach.

- For 2D : AMD (ICNTL(7) = 0) instead of default
- For 3D : default value for ICNTL(7), and depending on the accuracy :
- $\varepsilon > \mu$: smumps, BLR (set to ε)
- 1e-16 $< \varepsilon < \mu$: smumps, BLR (set to μ) + iteration step
- $\varepsilon \leq$ 1e-16 : dmumps, BLR (set to μ) + iteration step

Conclusion

Let μ be the value set by the user, and ε the accuracy they want to reach.

- For 2D : AMD (ICNTL(7) = 0) instead of default
- For 3D : default value for ICNTL(7), and depending on the accuracy :
- $\varepsilon > \mu$: smumps, BLR (set to ε)
- 1e-16 < $\varepsilon < \mu$: smumps, BLR (set to μ) + iteration step
- $\varepsilon \leq$ 1e-16 : dmumps, BLR (set to μ) + iteration step
- Thank you for listening !