Parallel Auto-tuned GMRES Method to Solve Complex Non-Hermitian Linear Systems

Pierre-Yves Aquilanti[1][2], Serge Petiton[1], and Henri Calandra[2]

[1] LIFL - UMR Lille 1/CNRS 8022
Cité Scientifique - Bâtiment M3
59655 Villeneuve d'Ascq Cédex - France
[2] TOTAL Exploitation Production
Avenue Larribau
64000 Pau

Abstract. Solving a linear system in a minimum time is a key factor in many scientific fields. It is possible to reduce time of computation for a given linear system by using auto-tuning and adaptive parameters. We propose auto-tuning within the GMRES solver by changing the value of the restart parameter dynamically in order to reduce the time of computation. We outline the limits of our heuristic through sequential an parallel experiments on complex-valued non-hermitian non-symmetric matrices including Helmholtz matrices.

1 Introduction

Linear algebra has been a very active field over the past years. This can be explained by many reasons, one of them being the increasing need of numerical tools for solving new types of problems in a wide range of scientific fields from mechanical engineering to geophysics. An important part of those problems are modelized by linear systems.

A system of linear equations is generally described by

$$Ax = b \tag{1}$$

where $A_{n\times n}$ is the coefficient matrix , x_n the solution, b_n the right hand side of the linear system of equations, n the size of the vectors x and b, $n \times n$ the size of the involved matrix. Two main types of solvers exist : direct and iteratives solvers. Direct solvers (LU, MUMPS[1] for example) will compute the exact solution of the equation 1. The problem with direct solvers is that they face a difficulty to solve huge linear systems because they require a lot of memory storage and computing power. This cost is important when $A_{n\times n}$ is sparse and the number of unknowns very important (it can be counted in millions or billions for some cases).

Because the size of linear systems tends to increase over the years and new types of linear systems emerge (Partial Differential Equations for example), an other kind of linear solver emerged, iterative methods. There is a broad number of iterative methods, starting from the Conjugate Gradient[16] (CG), a classical solver for symmetric matrices, to the Generalized Minimum Residual (GMRES)[15]. GMRES is a common choice for solving large sparse linear systems where $A_{n\times n}$ is a non-symmetric matrix. It minimizes the residual norm over the Krylov subspace :

$$\mathcal{K}_m(A, r_0) = \text{span}\{r_0, Ar_0, A^2 r_0, \ldots, A^{(m-1)} r_0\}, m = 1, 2, \ldots \tag{2}$$

at every step, r_0 is the initial residual vector and x_0 the initial guess ($r_0 \equiv b - Ax_0$). Because the amount of computation and storage increase with the number of iterations, GMRES is restarted with the last approximate solution as the new initial guess every m steps to limit this cost. This Restarted GMRES is called GMRES(m)[15]. This process will repeat until a satisfying criteria is met, ie. the residual norm is small enough. The restarted GMRES parameter m is called the restart parameter or GMRES subspace size because the partial solution will lie into the subspace spanned by GMRES (equation 2).

We focus our work on the auto-tuning of GMRES via the selection of the restart parameter over the restart process of GMRES to reduce the time of computation during the solving process, we studied the behavior of our GMRES auto-tuning for different matrices. Complex domain non-symmetric linear systems have not been as heavily studied as the real case and have a lot of implications in various fields. Our approach is mainly experimental.

An other implication of auto-tuning for linear solvers is to assist the solver user to select the different parameters and handle automatically this selection if necessary. Optimal parameters for a particular problem are hard to guess without heavy knowledge of iteratives solvers and the problem to solve. Auto-tuning is an assistant to help the user to obtain optimal parameters needed to solve a given problem.

Auto-tuning is also very important when parallelism is taken into account. This is generally the case when computation occur on a multi-core workstation, a cluster or a computing grid. Computation involves the processor cores but also the different memory levels from cache memory to Random Access Memory (RAM). Auto-tuning helps to reduce the time of computation by studying the execution environment with multiple phases [7].

This paper is organized as follows. In Section 2 we will present related works to the adaptive restart parameter for GMRES(m). This will lead us to make a proposal of an adaptive based algorithm whose goal is to reduce the time of computation for various complex-valued non-hermitian non-symmetric matrices in 3. We will then present in Section 4 experiments based on our adaptive including a study of the efficiency of our method in a parallel environment. Thus will we'll conclude on the efficiency of our contribution.

2 Related Works

It has been accepted that larger was the restart parameter better was the convergence of GMRES because of the information accumulated into the GMRES residual polynomial [12], thus requiring less iterations for a resolution than a small restart. A larger m also avoids stalling, as stated in [15, 18], and reduces the burden of GMRES superlinear convergence [18]. However, a larger m also means higher memory and computational costs because the Krylov base is increasing with the number of iterations, higher storage because we have to store more data, higher computational costs because the Arnoldi orthogonalization process becomes more and more expensive as m increases. Later iterations will require more time to compute than the first ones. This expense does not guarantee a faster convergence or any convergence at all. Besides, Embree higlighted in [9] that a smaller restart parameter can result in fewer iterations for some particular problems. One could understand that it is very difficult to choose a proper restart parameter and it brings back the problem of finding a good balance to attain convergence with a reduced time of computation and/or iterations.

The restart parameter is generally choosed arbitrarily prior to the execution of the solver and will remain fixed over the different restarts. Different works focused on the dynamic change of the restart parameter depending on different heuristics. Joubert [12] proposed a sophisticated heuristic weighting the computational cost against the residual norm reduction which aims to find a good balance between the time of computation and the convergence. Sosonkina and al. presented an adaptive implementation of GMRES(m) to avoid stagnation by detecting it with an heuristic that computes the relative number of iterations before convergence, the algorithm is reacting by increasing m [17]. However, one of the drawback of this adaptive implementation is that the restart parameter m increases until it reachs the maximum boundary and will never decrease, even if convergence conditions are getting better. Later restarts will carry a cost (for memory and computations) that could be avoided. That is what tried to do Habu and Nodera in a work leaning on Sosonkina's implementation to decrease the restart parameter to its initial value [10]. Similar work has been presented by Kuroda, Katagiri and Kanada [13] where the authors increase m at each iteration from a small value till it reachs a maximum value and then set it back to it's inital value as a cycle. The goal of this contribution is to increase convergence and reduce time of computation but it is not based on any analysis of the convergence by heuristics. An other adaptive method provided by Zhang and Nodera [19] compares the harmonic Ritz values (computed by Arnoldi) and the Ritz values (computed by GMRES, they are the approximate eigenvalues) to choose m in order to avoid stagnation. An other approach proposed by Baker and al.

[4] computes an angle, called successive angle, between the residual norms to decrease the restart parameter adaptively over the restarts to decrease the time of computation. This idea has been pursued more deeply in an other work [3] where the same authors also propose an adaptive heuristic to prevent the repetitive (or alternating) convergence behavior of GMRES(m) restarts by computing skip angles between two non-successive restarts for real-valued symmetric and skew-symmetric problems.

3 Proposal

Our goal is to propose an adaptive version of GMRES to reduce the time of computation needed to solve complex-valued non-hermitian non-symmetric linear systems. We were interested in the proposal of Joubert [12] that a decrease of the restart parameter could be a benefit to reduce the time of computation. It conducted us to follow the idea of Baker [3] that we found very seducing for various reasons : a negligible overhead on computation, reducing the restart parameter instead of globally increasing it, a powerful heuristic despite its simplicity. We also found that it was in general difficult to predict the behavior of GMRES by using different restarts and judge all the different approach for the complex-valued case as all previous works were dealing with real-valued matrices. We were also seduced by the fact that this approach was very flexible and could be used in either real or complex domain. We defined an algorithm (algorithm 1) based on [3] that reduces GMRES restart parameter continuously to reduce the general time of computation and increases it if needed like in the case where convergence is really weak, near stagnation, the choice is done thanks to a simple heuristic. Hardware memory specifications helps us to define the boundaries between which will evolve the restart parameter.

3.1 Reducing the Restart Parameter

To reduce the restart parameter we follow an angle based on two successive residual norms (equation 3), this can be seen as a convergence ratio between two successive restarts.

$$cr = \cos\angle(r_i, r_{i-1}) = \frac{\|r_i\|_2}{\|r_{i-1}\|_2} \tag{3}$$

We also define two bounds *max_cr* and *min_cr* for this angle. If cr falls over *max_cr* we consider that the current restart parameter m is satisfying and we will continue to use it for the next restart. If cr is under *max_cr* and over *min_cr* then the next restart will use the current m decreased by a value d. If at any time by decreasing m by d we reach a minimum value m_{min} then m will be settled back to m_{max} which is also the start value for m. If cr brake this vertuous cycle by being under *min_cr* then we will react by a sudden increase of the restart

Algorithm 1 adaptive version of GMRES by adaptive reduce of the restart parameter

```
1: $m_{memory\_level} \leftarrow 1$
2: $m_{count} \leftarrow 5$
3: $m_{cpt} \leftarrow 1$
4: $m_{memory\_max\_level} \leftarrow$ maximum number of memory levels
5: $m_{memory}[] \leftarrow$ m limits depending on memory size
6: $m_{memory}[m_{memory\_max\_level}] \leftarrow 1000$
7: $cr \leftarrow 1$
8: $d \leftarrow 3$
9: while not converged do
10:   if $m_{cpt} > m_{count}$ then
11:     $m \leftarrow m_{max}$,$m_{cpt} = 1$,$m_{memory\_level} = 1$
12:     if $m_{memory\_level} < m_{memory\_max\_level} - 1$ then
13:       $m_{memory\_level} \leftarrow m_{memory\_level} + 1$
14:     else
15:       $m_{memory\_level} \leftarrow 1$
16:     end if
17:   else if $cr > max\_cr$ then
18:     $m \leftarrow m_{max}$,$m_{cpt} \leftarrow 1$,$m_{memory_level} \leftarrow 1$
19:   else if $cr < max\_cr$ and $cr > min\_cr$ then
20:     if $m - d \geq m_{min}$ and $m_{cpt} = 0$ then
21:       $m \leftarrow m - d$
22:     else
23:       $m \leftarrow m_{max}$
24:     end if
25:     $m_{cpt} \leftarrow 0$,$m_{memory\_level} \leftarrow 1$
26:   else if $cr \leq min\_cr$ then
27:     if $m_{cpt} < m_{count}$ then
28:       if $m \times 2 \leq m_{memory}[m_{memory\_level}]$ then
29:         $m \leftarrow m \times 2$,$m_{cpt} \leftarrow m_{cpt} + 1$
30:       else
31:         $m \leftarrow m_{memory}[m_{memory\_level}]$,$m_{cpt} \leftarrow m_{cpt} + 1$
32:       end if
33:     else if $m_{memory\_level} = m_{memory\_max\_level}$ then
34:       $m \leftarrow m_{memory}[m_{memory\_level}]$,$m_{cpt} \leftarrow m_{count} + 1$
35:     end if
36:     if $m_{memory\_level} > m_{memory\_max\_level} - 1$ then
37:       $m_{memory\_level} \leftarrow 1$
38:     end if
39:   end if
40:   GMRES($m$)
41:   $cr \leftarrow \|r_i\|_2 / \|r_{i-1}\|_2$
42: end while
```

parameter.

3.2 Sudden Increase of Restart Parameter

Prior to the execution of the solver we define the restart parameter increasing levels with respect to the execution hardware platform. Those levels are defined by computing the maximum restart value depending on the different memory levels (cache, RAM) of the host platform.

Our idea is to increase the restart parameter when its decrease (presented previously) is not sufficient to improve the gain in convergence (ie. the time of computation) and possibly overcomes the stagnation process. To sum up, if cr is under min_cr. This follows the statements that were made in the contributions we presented earlier [13, 17, 10]. The difference appears in the fact that we take into account that the time of processing data will depend on its location between the different memory levels. Access times are higher if data is in RAM memory than if it is in cache memory. We compute an m restart parameter for each memory level, those computed m (or m_{memory}) will define access memory bounds that we will have to respect if we want to guarantee a certain optimality in terms of time of computation. Granularity of memory is taken into account if multiple cores share the same processor die. Typically, the L1 cache is specific to a core, the L2 and/or L3 are usually shared between the cores.

We compute each m_{memory} by the above formula :

$$(nnz + 3n + m(m+1) + n(m+1)) \times SizeOfScalar \geq MemoryBytesLevel \quad (4)$$

Which brings us back to solve the following second-order equation. Each variable in the left part is a constant :

$$\frac{MemoryBytesLevel}{SizeOfScalar} - nnz - 4n = m^2 + m(n+1) \quad (5)$$

All computed m_{memory} will be differently processed. The last one, the one that is computed by taking into account the RAM memory size, will be used to as a control value. It means that we use this value to evaluate if m (or m_{max}), given by the user at solver's launch, is under the m_{memory} RAM. This is needed to verify the consistency of user's given parameter.

The other m_{memory} values, the ones that are computed from the different cache memory levels, will be placed into an array $m_{memory}[]$ to which we will add a last value. The reason we added this arbitrarly choosed value is because, as we saw in Section 1, large values of restart parameter can sometimes avoid stalling and reduces the superlinear convergence of GMRES [15, 18]. This last value is used if no other classical increase or decrease of the restart parameter helped the solver to converge. In practice we choosed a value of 1000, which is generally under the m_{memory} defined by the RAM memory.

The sudden increase of the restart parameter process will also use three variables : m_{cpt} a counter of successive increases, m_{count} the maximum number of successive increases, and *m_memory_level* that corresponds to the current m_{memory} used level. To increase the current m, the solver will multiply it by 2 and increment m_{cpt} until this counter reachs m_{count} and no improvement was met (if *cr* is still under *min_cr* for m_{count} times). If no improvement has been encountered, then the current augmented m will be replaced by the current m_{memory} designated by *m_memory_level* ($m \leftarrow m_{memory}[m\_memory\_level]$). If no improvement is met, then the sudden increase process will continue until convergence gets better or the last *m_memory_level* value is reached (the one we added arbitrarly). If after using this last value we still had no convergence improvement, then the method is declared as stalling.

4 Numerical Experiments

We will here provide results of our adaptive GMRES(m_{min}, m_{max}) for various problems. We experimented our method in parallel on complex non-symmetric non-hermitian matrices available from the University of Florida Sparse Matrix Collection [6] and different matrices that we generated from discretization of the Helmholtz equation by finite differences simulating a wave propagation in frequency in a heterogeneous medium.

We studied the adaptive restart parameter on complex-valued non-hermitian non-symmetric matrices to reduce the time of computation needed to attain convergence. Each experiment was run till the solver reached a precision of $10e^{-6}$ for the true residual with a maximum number of 10000 iterations (which is the default for PETSc [5]) and was run 10 times. For each, we recorded the number of iterations needed the attain the desired precision and we also recorded the average time required to solve the linear system.

The experiments where conducted using PETSc 3.0.0-p10 [5] modified for our needs. The two first experiment set were done using an HP wx94000 workstation with two AMD Opteron 2200 dual-cores and 16GB of RAM memory. Those experiments were using four processes, we solved each linear system in parallel. The third set of experiments was conducted on a SGI Altix ICE8200EX cluster from Total Exploitation Production, the number of processors cores used is varying from 1 to 512 for each solving process.

4.1 Matrices

We accomplished our study on different complex-valued non-hermitian non-symmetric matrices, previous contributions were made using only real-valued matrices. We also used different Helmholtz matrices, a type of linear system that is described as difficult to solve in the litterature. The complex-valued case has not been as heavily studied as the real-valued case because of its complexity and the fact that properties cannot always be applied equally on both domains. GMRES behavior is also not as much understood for the complex-valued non-symmetric non-hermitian case than compared to the real one. Right-Hand side vectors used for the following matrices are equal to the 2-norm of the vector.

Matrix	n	nnz	kind
aft02	8,184	127,762	acoustics problem
conf5_0-4x4-26	3,072	119,808	theorical/quantum chemistry problem
kim1	38,415	933,195	2D/3D problem
mhd1280a	1,280	47906	electromagnetics problem
young4c	841	4,089	acoustics problem

Table 1. UF complex-valued non-hermitian non-symmetric matrices

We also generated matrices by the discrete formulation of the Helmholtz Equation in the frequency domain with boundaries conditions on a heterogeneous medium which is proposed by Pinel [14]. The boundaries conditions are modelized by the Perfectly Matched Layer Technique (PML), it consists in the absorption of the wave reflections on the boundaries to simulate an infinite medium. The discretization of the Helmholtz formulation is done via a second-order finite differences scheme. The resulting matrix is sparse five-banded (or seven-banded in 3-D), complex-valued, non-hermitian and non-symmetric generated by a classical 5 points Cartesian stencil (figure 1) (7 points for the 3-D case).

Our Helmholtz matrices are generated from the discretization of two heterogeneous velocity models commonly used in the geophysical domain for algorithmic evaluations : IFP Marmousi velocity model [11] and SEG/EAGE Overthrust 3-D model [2]. We dicretized the models with different frequencies producing matrices with different caracteristics. We will restrict our study on the Helmholtz matrices to *hel_369* for the first two sets of experiments. We also generated a right-hand side equal to the norm of each UF matrix. For the Helmoltz matrices we generated a $b[n] = 0$, $n = 1, 2, \ldots, m$ except at the position that is corresponding to the location of the source generating the wave $b[x] = 1$.

4.2 Adaptive Variation of GMRES(m) Restart

In a first set of experiments we experimentally studied the behavior of GMRES(m) with an adaptive restart parameter depending on the successive angle defined in Section 3. Our goal was to study the feasibility of this approach for complex-valued non-symmetric matrices. The results proved that we could reach some

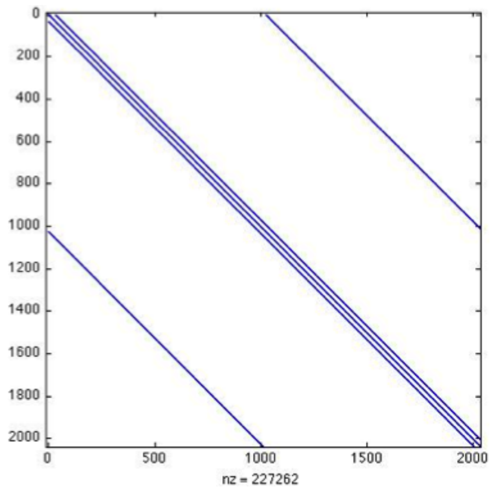

Fig. 1. Five banded *Helmholtz* matrix

Matrix	n	nnz	Model	Discretization Frequency
hel_161	984,998	6,842,820	Overthrust	1Hz
hel_737	177,617	886,609	Marmousi	10Hz
hel_369	44,649	222,505	Marmousi	5Hz

Table 2. Helmholtz matrices characteristics

improvements with our method over the use of a fixed restart like for the classical GMRES(m). The adaptive parameter values used where $d = 3$, $min\_cr = \cos(80)$, $max\_cr = \cos(8)$. Our study of the sequential angle shows that those values would be the most appropriate. For each matrix we defined the m_{min} value to 3, we have varied this parameter but noticed that there was no real change in the behavior of the adaptive restart. Concerning m_{max} we choosed values from 3 to 1000 which defined good limits to understand the behavior of our study. In figure 2 and 3, doted lines are the best and worst convergences for classical GMRES.

As stated by Joubert [12], small restart parameter can make the solver converge where high values cannot. However, it is also taken for granted that high restart parameter values can make problems converge when this would not be the case with small values[15]. As Baker [3] we observed that varying the restart parameter was not the panacea for problems that stagnate, this was the case for *conf5_0-4x4-26*. Nevertheless, we noticed interesting behavior for the stagnating $mhd1280a$ (figure 2) as for the two diverging linear problems $aft02$ and $kim1$ even if the adaptive restart would not make the problem converge it had a certain influence over the classical GMRES(m) on the convergence.

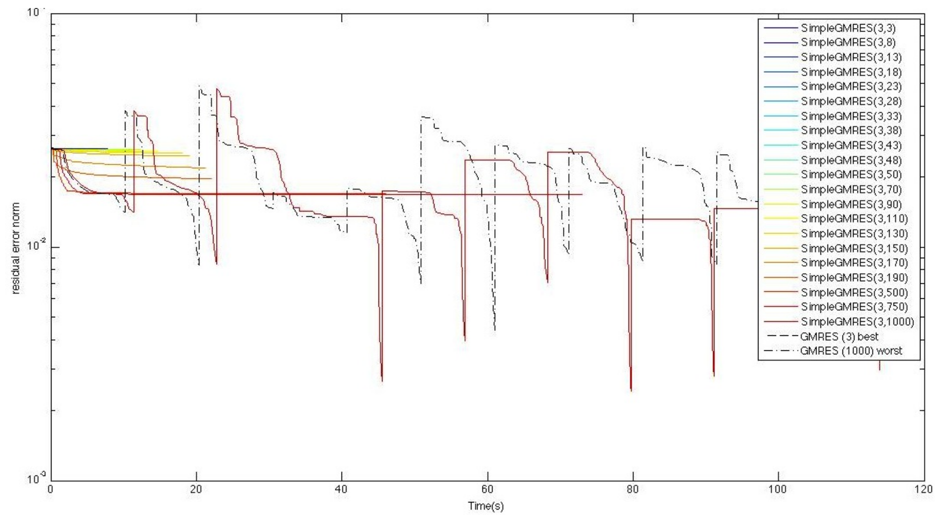

Fig. 2. $mhd1280$ convergence for various values of m_{max}

On the other hand, for our two converging matrices (*fig:young4c* and *hel_369*) we had two different behavior. For *young*4*c* we did not noticed any improvment at first sight over GMRES(m). Varying the restart parameter adaptively can become to costly in time of computation if the problem is already converging to well. However, we observed that for the *hel_369* problem the improvement of the adaptive restart was up to 20% in terms of time of computation. One could notice on figure 3 that the adaptive restart version of GMRES is faster for a range of m_{max} (or m) from 8 to 38 than the best time of computation for the classical GMRES(m) with $m = 18$.

Our analysis of the results will follow the one made by Baker in [3] on the real-valued symmetric and skew-symmetric linear problems. The adaptive GMRES(m_{min},m_{max}) cannot overcome stagnation or divergence of problems even by the sudden increase. However we were not totally surprised for this last point, for high values of m the classical GMRES(m) would also not converge.

We pursued our study by focusing our attention on the two converging matrices behavior (*young4c* and *Helmholtz*) by making an evaluation of our methods for a wide range of m_{max} (m for the classical GMRES) in a second set of experiments. This highlighted that high restart parameters values would not mean faster convergence. Moreover, we were allowed to observe a gain of 20% in time in general for our method over the classical GMRES(m) for the *hel_369* matrix. It also appears that there is an optimal restart parameter value that minimize more the time to solution than other ones, this highlight the bearing of choosing a proper restart parameter for GMRES.

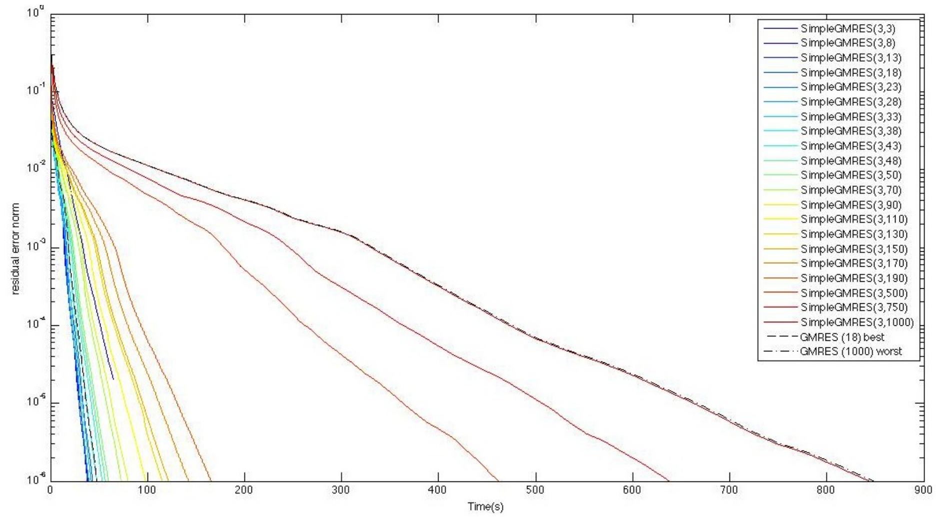

Fig. 3. The *Helmholtz* problem convergence

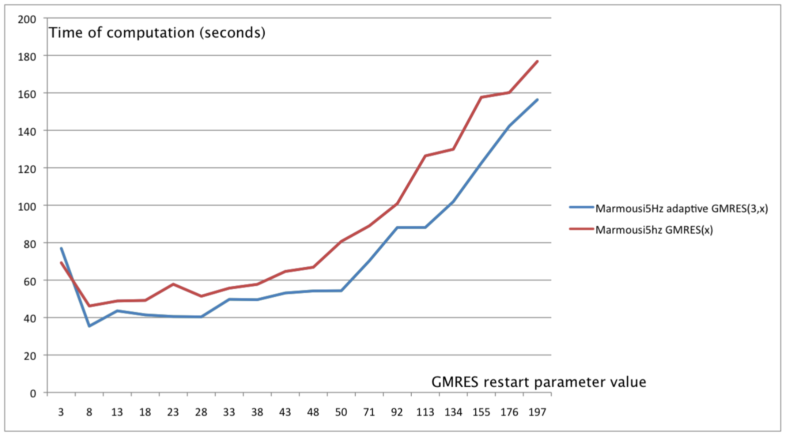

Fig. 4. *Helmholtz* wide range convergence in seconds for many restart parameter values.

One will notice that the restart parameter value as an influence on the number of iterations. This influence becomes obscured as the restart value grows. About the number of iterations two GMRES variants, there no clear discrepancy of behavior for the number of iterations on the exception of the *hel_369* matrix on small restart values (figure 5). Results from the *young4c* case, where corroborating the ones made on *hel_369*.

Parallelism is a critical key for today solvers, it will become even more critical when fully entering the petaflopic era and the exaflopic one. It is already not rare

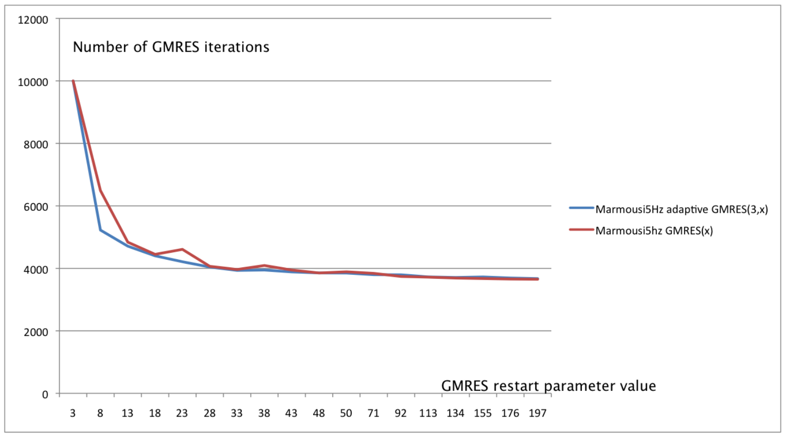

Fig. 5. *Helmholtz* wide range convergence in number of iterations for many restart parameter values.

to use iterative methods on clusters, as parallelism grows it is important to understand the behavior of algorithms in order to propose the best possible optimization. The same code will not have the same behavior on a sequential or execution platforms. Parallelism has many levels (GPU, cores, processors, nodes, racks, clusters, grids,...), for each of them we need to define the limits of it's processing possibilites, parallel or not. This is exactly the same for solvers where boundaries needs to be drawn for each sofware element. Following this statement we studied the scalability of our adaptive method to see its effectiveness in parallel environment. We were already executing our in parallel but we extended our study to a range from 1 to 512 nodes. We runned our solver on the Helmholtz matrices described in Table 4.1 and *young4c*. For each experiment we proceeded as previously. We recorded the average time of computation for ten runs for each matrix, for three solvers : a classical GMRES(m), our adaptive GMRES(m_{min},m_{max}) with sudden increase and the adaptive GMRES(m_{min},m_{max}) without sudden increase corresponding to the algorithm defined in [3].

Our experiments showed that gain in time was in general higher when allowing sudden increase of the restart parameter (figure 6) compared to the simple decrease of this parameter (figure 7). Comparing the two figures highlight the fact that the improvement is not equal for all matrices and might face a certain overhead for particular ones as it is the case for *young4c*. Gain in time of computation for convergence does not obey to a linear law. As we increase the number of nodes we can see that the gain induced by our method is decreasing but not equally for all matrices. It is hard to make a general conclusion based on our two figures but it stands out that varying the restart parameter for GMRES has a certain influence on convergence and that the sudden increase is in general more efficient compared to a method that just decrease the restart parameter. This

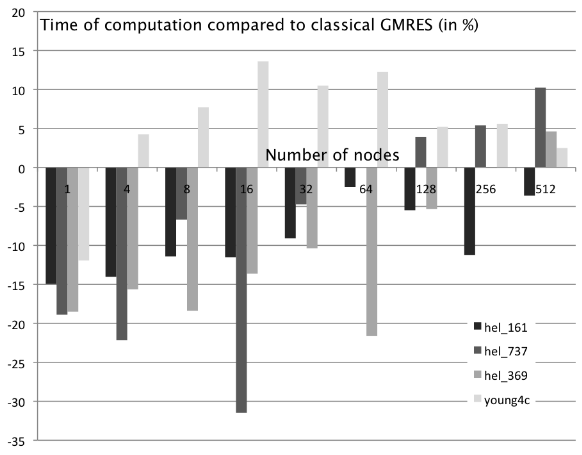

Fig. 6. Decrease in time for adaptive GMRES with sudden increase of the restart parameter in percents from 1 to 512 nodes.

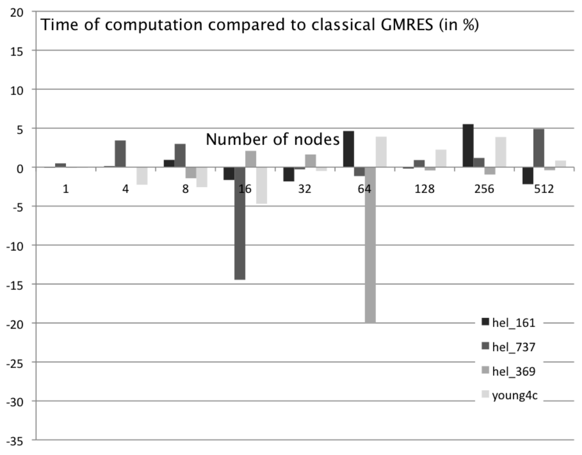

Fig. 7. Decrease in time for adaptive GMRES without sudden increase of the restart parameter in percents ffrom 1 to 512 nodes.

also highlight the fact that there exists a scalability boundary that need to be respected in order to keep a certain gain. However, if there is no clear gain when using a lot of nodes we can see that the influence of our method on convergence is noticeable even if it produces an overhead.

Varying the restart parameter by an adaptive method is an easy way to improve GMRES convergence. Computation overhead is light over classical GMRES and it allows to improve convergence in time from a few percents to 30%. Nevertheless, when considering stagnating problems, varying the restart parameter adaptively, even for high values, does not allow the problem to converge. Our experiments showed that there is is a clear gain to both increase and decrease the restart parameter during the solving process. Our work focused on complex-valued non-hermitian non-symmetric matrices showed that simple techniques coming from the real domain can be fully applied to the complex. We also presented experiments on the scalability of adaptive restart parameter method that demonstrated that the gain over classical GMRES for our method is valuable when using multiple nodes.

Another thing that we noticed through our experiments is that in order to reduces time of computation, varying the restart parameter cannot be done by one way ie. by decreasing or increasing. The use of both methods has its advantages and also its drawbacks. By only decreasing the restart parameter GMRES restart cycles will be shorter but the resulting residual vector will contain less information than with higher values. In some cases this is not crucial ie. when convergence is good. In the cases where convergence is weak then one may need to increase the restart parameter value. Even if restart cycles take more time to produce a new GMRES iterate, the gain in information is valuable for convergence optimization. We studied both views on the restart parameter made by Joubert [12] and Saad [15] and our experiments let us think that both are right but we will add some overtone : it will just depend on the case. This depends on the matrix and the convergence during the GMRES restart process. Restart parameter value increase and decrease approachs are distinctly valuable but a mix of both can be powerfull as illustrated in our experiments. It is about a tradeoff between iterate information and time of computation.

5 Conclusion

Our study shows that simple adaptive improvements are valuable for complex-valued non-hermitian non-symmetric cases. Our experiments showed a real improvement for Helmholtz matrices, but this depends on the matrix. There is a real need to define new heuristics and act on proper parameters. Futhermore, adaptive parameters can be introduced at every stage of the linear system solver, including the preconditioner. Adaptive parameter do not only involve the linear system numeric but also on its parallelism when existing to select proper resources for a matching case. It is rather difficult to determine a restart parameter prior to the solve process as it is very dependant on the problem and the hardware that is used but there's a real interest of an adaptive change parameter that will provide auto-tuning and help the user to select optimal parameters. Hybridation of solvers might be a key point to explore, like in the MERAM method [8]. For this last point the parallelism has to be taken into account and its limits sketched as it is a key point and will become more important in the next future with the democratization of petaflopic solutions and emergence of exaflopic ones.

References

1. P. R. Amestoy, I. S. Duff, J. Koster, and J.-Y. L'Excellent. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications*, 23(1):15–41, 2001.
2. F. Aminzadeh, J. Brac, and T. Kunz. 3-D Salt and Overthrust Models. *SEG/EAGE 3-D Modeling Series*, 1, 1997.
3. Alison Baker, Elizabeth R Jessup, and TV Kolev. A Simple Strategy for Varying the Restart Parameter in GMRES(m). *Journal of Computational and Applied Mathematics*, 230(2):751–761, 2009.
4. Alison Baker, Elizabeth R Jessup, and T Manteuffel. A Technique for Accelerating the Convergence of Restarted GMRES. *SIAM Journal on Matrix Analysis and Applications*, 26:962, 2005.
5. Satish Balay, Kris Buschelman, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc Web page, 2009. http://www.mcs.anl.gov/petsc.
6. Timothy A. Davis. University of Florida Sparse Matrix Collection. *NA Diges*, 92, 1994.
7. James Demmel, Jack Dongarra, Victor Eijkhout, Erika Fuentes, Richard Wilson Vuduc, R .Clint Whaley, and Katherine Yelick. Self-adapting linear algebra algorithms and software. *Proceedings of the IEEE*, 93(2):293–312, 2005.
8. Nahid Emad, Serge Petiton, and Guy Edjlali. Multiple explicitly restarted arnoldi method for solving large eigenproblems. *SIAM J. Sci. Comput.*, 27(1):253–277, 2005.
9. Mark Embree. The Tortoise and the Hare Restart Gmres. Dec 2001.
10. Mitsuru Habu and Takashi Nodera. GMRES (m) Algorithm with Changing the Restart Cycle Adaptively. *Proceedings of Algorithmy 2000 Conference on Scientific Computing*, pages 254–263, 2000.
11. IFP. Marmousi Model. Synthetic 2D Acoustic Model, 1988.
12. Wayne Joubert. On the Convergence Behavior of the Restarted Gmres Algorithm for Solving Nonsymmetric Linear Systems. Oct 1997.
13. Hisayasu Kuroda, Takahiro Katagiri, and Yasumasa Kanada. Performance of automatically tuned parallel gmres(m) method on distributed memory machines. Apr 2000.
14. Xavier Pinel. *A Perturbed Two-Level Preconditioner for the Solution of the Three-Dimensional Heterogeneous Helmholtz Problems with Applications to Geophysics*. PhD thesis, CERFACS, 2010.
15. Yousef Saad and Martin H Schultz. GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, Jan 1986.
16. Jonathan R Shewchuk. An Introduction to the Conjugate Gradient Method Without the Agonizing Pain. Technical report, 1994.
17. Maria Sosonkina, Layne T. Watson, and Rakesh K. Kapania. A New Adaptive Gmres Algorithm for Achieving High Accuracy. Mar 1996.
18. Henk A van der Vorst and C Vuik. Superlinear Convergence Behaviour of GMRES. *Journal of Computational and Applied Mathematics*, 48(3):327–341, 1993.
19. Linjie Zhang and Takashi Nodera. A New Adaptive GMRES(m) Algorithm with Correction. *The 12th Biennial Computational Techniques and Applications Conference*, 2004.