



國立台灣大學理學院數學所
碩士論文

Department of Mathematics
College of Science

National Taiwan University
Master Thesis

圖形處理器上的內迭代不精確和混合精度特徵值解法

Inexact and Mixed Precision Eigenvalue Solvers on GPU

黃志銘

Jhih-Ming Huang

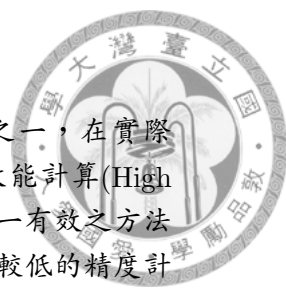
指導教授：王偉仲 博士

Advisor: Wei-Chung Wang Ph.D.

中華民國 103 年 8 月

August 2014


Abstract



特徵值問題是現今工程及科學計算領域中最重要的議題之一，在實際應用中，解決特徵值問題所需的計算量相當龐大，因此高性能計算(High Performance Computing, HPC)在此扮演著重要的角色，其中一有效之方法是運用混合精度以達到更高的效能，也就是在適當的時機使用較低的精度計算，並且不影響最後計算的精度。單精度所需的記憶體容量較小，因此有較大的可能性可造成高速緩存命中(cache hit)，其往往影響最終效能顯著。除此之外，許多運算以單精度皆可獲得較高的效能。因此如果原本的演算法就有較高的精度容忍程度，若重新設該演算法將有機會運用混合精度的方法達到更高的效能。而我們專注的演算法既屬於此種類型。Shift-Invert Residual Arnoldi (SIRA)演算法為計算特徵值的重要方法，該演算法由內外迭代迴圈所組成，其中內迴圈為求解線性系統的過程，其目的在於計算修正方向，以幫助外迴圈計算出所要之特徵值與特徵向量。SIAR 的效率決定於內迴圈迭代中線性系統的解。然而，此線性系統可以在不影響最終精度的狀況下，以較低的精度求解。本研究主要利用此特性及混和精度和圖形處理器針對對稱正定 (Symmetric Positive Definite, SPD) 的大型稀疏矩陣，設計一有效率的特徵值解法。我們提出一名為混合精度口袋演算法，該方法可以自動選擇何時使用單精度或雙精度計算內圈迭代，並且在每一回的內圈計算中可以自適性地調整內圈的容忍度和離開回圈之時機。此方法在我們絕大多數的測試中都有最好的表現。

關鍵字： 特徵值問題、圖形處理器、混合精度

Abstract



Eigenvalue problem is one of the most crucial topics in engineering and science fields nowadays. In practice applications, the target matrix is usually large and sparse, hence solving the eigenvalue problems need huge computation amount. The high efficiency is a strong demand in practice, therefore High Performance Computing, HPC, plays an important role in this topic. One important approach for getting higher performance is mixed precision design, which means it will change the operation precision during the computation without dropping the final accuracy. Since single precision requires less memory storage and it may cause higher cache hit ratio, which may affect performance a lot. In addition, in some numerical operation, single precision is faster than double precision. Hence, if the original algorithm is accuracy insensitive, which means that it could lose some accuracy during the computation and keep the same final accuracy, then it is suitable to be redesigned as a mixed precision type algorithm to enhance the performance. The eigensolver we focus on exactly belongs to this type. Shift-Invert Residual Arnoldi, SIRA, algorithm is a well-known eigenvalue solver, which consists of an inner loop and an outer loop. The inner loop is solving a linear system, which is for searching the correction direction to help outer loop find the desired eigen-pair. The efficiency of SIRA relies on the solutions of the inner-loop linear systems. These systems can be solved in lower accuracy without downgrading the final accuracy of the target eigenvalues. By taking advantage of this algorithmic feature and the computational power of GPU, we develop a mixed precision eigensolver in this research. We develop a method called pocket method, it adaptively chooses the double or single precision to solve the linear system. Moreover, in solving the linear system, it automatically adjusts the inner tolerance and timing of exiting inner loop. Pocket method has the best performance in most of our experiments.

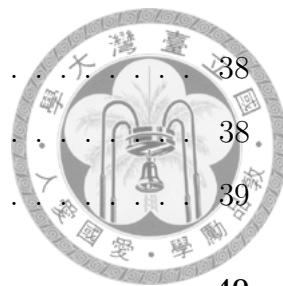
Keyword. Eigenvalue, Jacobi-Davidson, Mixed precision, Shift-Invert Residual Arnoldi, GPU, HPC

Contents



Abstract (in Chinese)	i
Abstract (in English)	ii
Contents	iii
List of Algorithms	v
List of Figures	vi
List of Tables	x
1 Introduction	1
1.1 Research Motivation	1
1.2 Research Target	1
1.3 Literature Review	3
2 Methods	5
2.1 JD and SIRA	5
2.2 Inexact Stopping Criteria in Inner Loops	10
2.3 Mixed Precision Algorithms	14
2.4 Arithmetic Intensity Effects	14
2.5 Search Subspace Effects	15
3 Numerical Results and Discussion	16
3.1 Implementation and Testing Problems	17
3.2 SPD Problem Collection	18
3.3 Photonic Crystal	20
3.4 Precision Performance Analysis	21
4 Conclusion	38
4.1 Summary of Methods and Results	38

4.2 Advantages of the Pocket Methods	38
4.3 Limitations	38
4.4 Future Directions	39
5 Acknowledgement	40
References	41
Appendix A Other graphs.	43



List of Algorithms

2.1	Jacobi-Davidson Method for Single Eigen Pair	6
2.2	Jacobi-Davidson Method for Multiple Eigen Pairs	7
2.3	Shift-Invert Residual Arnoldi for Single Eigen Pair	8
2.4	Shift-Invert Residual Arnoldi for Multiple Eigen Pairs	9
2.5	Fixed Stopping Criterion	11
2.6	H.N. Stopping Criterion	12
2.7	Constant Scaling	12
2.8	Pocket Stopping Criterion	13



List of Figures



3.1 Outer/Inner time ratio of fixed stopping criteria. The inner ratio become greater as the dimension increasing. 28

3.2 Outer/Inner time ratio of Pocket Mixed stopping criteria. The inner ratio become greater as the dimension increasing, but different to fixed in large size the inner proportion is smaller. And it's one of the reason of why mixed could save time, because it need fewer time in inner loop. 30

3.3 Speed Up of H.N. and Pocket w.r.t. fixed, Pocket Mixed has the best speed up in most case. 31

3.4 Residual behavior, H.N. and Pocket has similar behavior. Both of them capture the trend of r_{out} 32

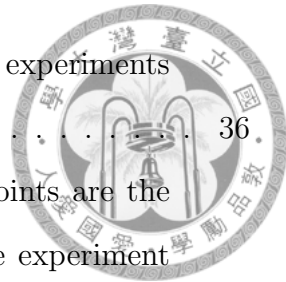
3.5 Matrix: thermomech_TC's counts of linear solver and total computation time. H.N. double and Pocket double have similar counts, but Pocket's performance is better. Since the judge cost in Pocket is cheaper than H.N. 33

3.6 Compare if Using Updating Method. Using updating method may save many time in outer loop iteration, and in fixed case the saving is more. Because fixed case has higher proportion of outer loop than the other two. 33

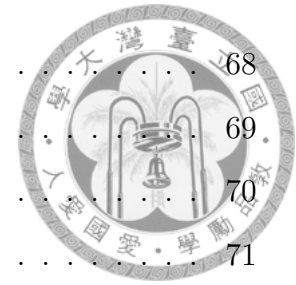
3.7 Inner Outer Comparison. H.N. double and H.N. Mixed have the same iteration number, and We could tell the mixed precision effect by H.N. Mixed bar and H.N. double; The larger size , the more performance gain in mixed precision method. The reason is that the speed up of fft is more significant in large size. 34

3.8 Parameter Effects. The proportion is more balance in small τ , and it's one of the reason to tell why it saving time. Since inner iteration is relative cheaper in this application, more proportion in inner may have better performance. 35

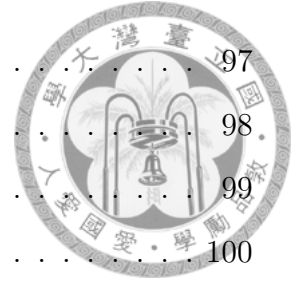
3.9	Performance model results. Our model prediction fits the experiments well.	36
3.10	Double single model result. The numbers above the points are the relative error of predictions. The model almost fits the experiment results.	37
A.1	Computing time of different stopping criteria	44
A.2	Computing time of different stopping criteria	45
A.3	Computing time of different stopping criteria	46
A.4	Total inner counts of different stopping criteria	47
A.5	Total inner counts of different stopping criteria	48
A.6	Total inner counts of different stopping criteria	49
A.7	Total outer counts of different stopping criteria	50
A.8	Total outer counts of different stopping criteria	51
A.9	Total outer counts of different stopping criteria	52
A.10	Using/not using updating method with Fixed-Double	53
A.11	Using/not using updating method with Fixed-Double	54
A.12	Using/not using updating method with HN-Double	55
A.13	Using/not using updating method with HN-Double	56
A.14	Using/not using updating method with HN-Mixed	57
A.15	Using/not using updating method with HN-Mixed	58
A.16	Using/not using updating method with Pocket-Double	59
A.17	Using/not using updating method with Pocket-Double	60
A.18	Using/not using updating method with Pocket-Mixed	61
A.19	Using/not using updating method with Pocket-Mixed	62
A.20	Different precondition with Fixed-Double	63
A.21	Different precondition with Fixed-Double	64
A.22	Different precondition with HN-Double	65
A.23	Different precondition with HN-Double	66
A.24	Different precondition with HN-Mixed	67



A.25 Different precondition with HN-Mixed	68
A.26 Different precondition with Pocket-Double	69
A.27 Different precondition with Pocket-Double	70
A.28 Different precondition with Pocket-Mixed	71
A.29 Different precondition with Pocket-Mixed	72
A.30 Inner iteration times in different preconditioner with Pocket-Mixed	73
A.31 Different initial with Fixed-Double	74
A.32 Different initial with Fixed-Double	75
A.33 Different initial with HN-Double	76
A.34 Different initial with HN-Double	77
A.35 Different initial with HN-Mixed	78
A.36 Different initial with HN-Mixed	79
A.37 Different initial with Pocket-Double	80
A.38 Different initial with Pocket-Double	81
A.39 Different initial with Pocket-Mixed	82
A.40 Different initial with Pocket-Mixed	83
A.41 Outer/Inner time ratio of different stopping criteria	84
A.42 Outer/Inner time ratio of different stopping criteria	85
A.43 Outer/Inner time ratio of different stopping criteria	86
A.44 Matrix inner and outer informations	87
A.45 Matrix inner and outer informations	88
A.46 Matrix inner and outer informations	89
A.47 Matrix inner and outer informations	90
A.48 Matrix inner and outer informations	91
A.49 Matrix inner and outer informations	92
A.50 Matrix inner and outer informations	93
A.51 Matrix inner and outer informations	94
A.52 Matrix inner and outer informations	95
A.53 Matrix inner and outer informations	96



A.54 Matrix inner and outer informations	97
A.55 Matrix inner and outer informations	98
A.56 Matrix inner and outer informations	99
A.57 Matrix inner and outer informations	100
A.58 Matrix inner and outer informations	101
A.59 Matrix inner and outer informations	102
A.60 Matrix inner and outer informations	103
A.61 Matrix inner and outer informations	104
A.62 Matrix inner and outer informations	105



List of Tables

3.1	Test Matrix Size between 10K and 100K	24
3.2	Test Matrix Size between 100K and 500K	25
3.3	Test Matrix Size between 500K and 1M	25
3.4	Test Matrix Size Greater than 1M	25
3.5	CG Inner Time. The higher cache hit ratio of cache hit ratio of double and cache hit ratio of single, the higher cost ratio between double and single precision inner loop cost.	25
3.6	FFT Time. Even the cache hit ratio are similar between double and single precision, the computation time is increasing with the dimension. The reason maybe most of cache hit of the kernels in fft operation are zero. See Table 3.7	26
3.7	FFT Cache. Most of cache hit of the kernels in fft operation are zero, especially for the Radix kernel, which is also the large part in fft operation. Hence we could not tell the single double ratio by cache hit in this case.	26
3.8	Convergence Count in each Stopping Criteria. All of them have the similar convergence.	26
3.9	Sum of Total Time in Each Stopping Criteria. Pocket mixed has the best performance in each size set.	26
3.10	1st Place Accumulation of Total Time in Each Stopping Criteria (sec.). Pocket mixed always gets the first place in our test cases. . . .	27
3.11	Sum of Total Inner Iteration Number in Each Stopping Criteria. Pocket method have the smallest inner iteration number, and it may one of the reason to tell why Pocket is the best stopping criteria. . . .	27
3.12	Sum of Total Time of Pocket Mixed with Updating and without Updating (sec.). Using updating method has better performance than non using, and this results are the same as we expected.	27



3.13 1st Place Accumulation of Pocket Mixed with Updating and without Updating. Using updating method almost gets the first place in our test cases. 27

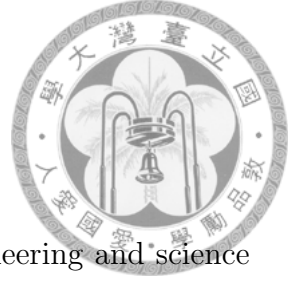
3.14 Sum of Total Time of Pocket Mixed with Different Preconditioners (sec.). FSAI is the best preconditioner in our experiment. 27

3.15 Sum of Total Inner Iteration Number of Pocket Mixed with Different Preconditioners. FSAI has the smallest iteration number, and it may be the reason of why it had the best performance than Jacobi and no using preconditioner. 29

3.16 Sum of Total Time of all K Vectors of Photonic Crystal (sec.). The H.N. mixed and Pocket mixed have the best performance. 29



1 Introduction



1.1 Research Motivation

Eigenvalue problem is one of the most crucial topics in engineering and science fields nowadays. In practice applications, the target matrix is usually large and sparse, hence solving the eigenvalue problems need huge computation amount. The high efficiency is a strong demand in practice, therefore High Performance Computing, HPC, plays an important role in this topic, and one important approach for getting higher performance is mixed precision design.

Jacobi-Davidson, (JD) algorithm is an typical eigenvalue solver[1], which consists of an inner loop and an outer loop, for solving large and sparse matrix's eigen pairs. The inner loop is solving a linear system, called correction equation, which is for searching the correction direction to help outer loop find the desired eigen pair. The efficiency of JD rely on the solutions of the inner-loop linear systems. These systems can be solved in lower accuracy without downgrading the final accuracy of the target eigenvalues. A similar algorithm, called Shift-Invert Residual Arnoldi, SIRA [2] [3], consists of an inner loop and an outer loop. The only difference with JD is that SIAR does not need to do the shift in the inner loop, and JD has to suitably shift the linear system during the iterations. Due to the property of insensitive tolerance in the inner loop, we redesign the SIRA algorithm by using the mixed precision approach. In this paper, our all experiment are focusing on SIRA method.

1.2 Research Target

In this paper, we focus on the standard type eigenvalue problems.

$$Ax = \lambda x$$

which A is a sparse, symmetric positive definite (SPD) matrix.

In practice, the target matrix is usually large and sparse, and there are few specific

desired eigen pairs to be solved. For example, the desired eigen pairs are the five eigen pairs, whose eigenvalue's absolute values are smallest. Since the computation amount for finding those desired eigen pairs are huge, the efficiency is an important issue in eigenvalue problems.



An important property of JD and SIRA is that we have an higher tolerance of the linear system in the inner loop, which means we only need to solve the linear system approximately, than other eigensolver such like Lanczos. A straight way to set the tolerance of inner loop is fixing the tolerance, and we called it the fixed type, and the tolerance may be set the same as outer loop or higher. There are some researchers proving the relation between the residuals of inner loop and outer loop[4], and giving a practice controlling strategy between them. In the theory[4], they gave an stopping criteria for the correction equation. Since the authors are Hochstenbach and Notay, we denote the stopping criteria as H.N. criteria in this paper. The stopping criteria use the information of residual of the outer loop and the behavior of the inner loop to determine when to exist the inner loop. For the early stage of the outer iterations, the tolerance will be higher than the later stage.

Inspired by the H.N. stopping criteria, I develop other two stopping approaches. One is the Constant Scaling stopping criteria, which means inner loop's tolerance is the current outer's residual scaled by a constant. The other one is Pocket stopping criteria, which means during the inner iteration it keeps the best x_i in pocket and keeps iterating until getting better x_i or reaching the iteration number constraining. It will state more detail in the Chapter 3.

Based on those stopping criteria, we apply the mixed precision approach on them. The single precision not only requires lesser memory storage, which may cause higher cache hit ratio, but also could be computed faster in some operation. Hence we expect that redesigning the SIRA algorithm to a mixed precision SIRA algorithm may get higher performance. Since we already know the accuracy requirement of inner loop before entering it at each outer iteration, we may determine using the single or the double precision to solve the linear system. However, in our experiments,

single precision solver is not necessary faster than double precision. Even though it's actually faster, the speedup is not a constant. For example, in some case, the ratio of double time divided by single time is about 1.5, but in some case the ratio is about 1.0, which means the performance of single precision operation is similar to double precision in that case. To clear the reason, we do some cache hit ratio test in different scenarios, and build up a predicting model. We also built another model to predict the performance of mixed precision algorithm compared with full double one.

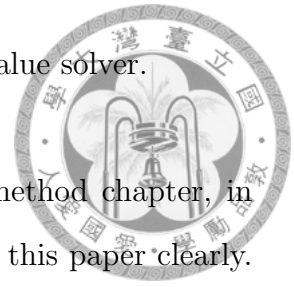
Our main test cases are from the Florida Matrix Collection [5], and we only select those matrix, whose size is greater than 100,000. Based on those matrix, we compare their performance between the four stopping criteria, fixed type, H.N., constant scaling, and Pocket in full double and mixed precision. The mixed precision Pocket method is the best method in most case, and H.N. also has not bad performance. In addition to those matrix from Florida Matrix Collection, we also do the experiments on a special application, finding the band gap of 3D photonic crystal. [6] In our experiments, Pocket and H.N. criteria are good strategies in determining when to exist the inner loop in photonic crystal experiments. Using those two stopping criteria all may save many inner iteration time without lost the final accuracy.

1.3 Literature Review

Here we review some results by previous researchers in mixed precision method. The mixed precision method has been applied to solve linear system well. The work of Baboulin et al.[7, 8, 9] was using mixed precision method on solving dense and sparse matrices via direct and iterative method. Besides they also focused on mixed precision on refinement on dense matrices in direct method. Hogg et al. worked on using mixed precision on symmetric sparse matrices and discuss some performance issue. The mixed precision approach on linear system is developed well, but there is no discussion of mixed precision on eigenvalue solver. In this pa-

per we provide some idea about using mixed precision on eigenvalue solver.

The later arrangement of this paper is: there will be the method chapter, in which it will state the algorithm and stopping criteria we use in this paper clearly. Then it's the numerical results and discussion of our experiments. The last chapter is the conclusion.



2 Methods



2.1 JD and SIRA

In this paper we are focusing on the SPD matrixes only. The Algorithm 2.1 is the JD algorithm for finding single eigenvalue. Given a target matrix A , the tolerance ϵ_{out} and the target value σ , the algorithm will find the eigenvalue, whose absolute value is closest to σ . At the beginning, it will update A to $A - \sigma$ and start with a initial matrix, V_0 , which may consist of the random orthonormal vectors. Then it will start the outer loop:

Step 1: Compute the dense subspace $V_k^t A V_k$, and find out all the eigen pairs of $V_k^t A V_k$, and select the (θ_k, s_k) , such that θ_k is closest to zero, and s_k is a unit vector.

Step 2: Compute $u_k = V_k s_k$ and residual $r_k = (A - \theta_k I) u_k$.

Step 3: Check the residual is small enough or not. if it is small enough return $\lambda = \theta_k + \sigma$ and $x = u_k$ as results. If not, go to next step.

Step 4: Solve the correction equation approximately such that $t_k \perp u_k$

Step 5: Orthogonalize $t_k \perp V_k \rightarrow v_{k+1}, V_{k+1} = [V_k, v_{k+1}]$, and go to step 1.

The Algorithm 2.2 is the JD algorithm for finding multiple eigenvalues. The only differences with JD algorithm for finding single eigenvalue are that

- There is a *SuccessFlag* to tell if it just found a convergent eigenvalue to determine whether it's needed to solve the correction equation in the current outer iteration.
- There is a process to 'lock' subspace, V_k , with the solved eigenspace, *EigSpace* to avoid solving duplicated eigenvectors.

The Algorithm 2.3 is the SIRA algorithm for finding single eigenvalue. It's almost similar to the Algorithm 2.1 [2] [3], and the only difference is in Algorithm 2.3 line 9

SIRA algorithm does not need to shift during solve the linear system. So is between Algorithm 2.4 and Algorithm 2.2.



Algorithm 2.1 Jacobi-Davidson Method for Single Eigen Pair

Input: A : The target matrix; ϵ_{out} : The outer tolerance; σ : The target

Output: (θ, x) : Eigen Pair Solved;

- 1: $A - \sigma \rightarrow A$
 - 2: Choose an n -by- m orthonormal matrix V_0
 - 3: **for** $k = 0, 1, 2, \dots$ **do**
 - 4: Compute all the eigenpairs of $V_k^T A V_k s = \lambda s$.
 - 5: Select the eigenpair (θ_k, s_k) with $\|\theta_k\|$ is smallest and $\|s_k\|_2 = 1$.
 - 6: Compute $u_k = V_k s_k$ and $r_k = (A - \theta_k I) u_k$.
 - 7: If $(\|r_k\| < \epsilon)$, $\lambda = \theta_k + \sigma$, $x = u_k$, Stop
 - 8: Solve (approximately) a $t_k \perp u_k$ from
 - 9: $(I - u_k u_k^T)(A - \theta_k I)(I - u_k u_k^T)t = -r_k$.
 - 10: Orthogonalize $t_k \perp V_k \rightarrow v_{k+1}$, $V_{k+1} = [V_k, v_{k+1}]$
-



Algorithm 2.2 Jacobi-Davidson Method for Multiple Eigen Pairs

Input: A : The target matrix; N : How many eigen pairs needed to be solved; ϵ_{out} : The outer tolerance; σ : The target

Output: $(\theta_1, x_1) \dots (\theta_N, x_N)$: Eigen Pair Solved;

```

1:  $A - \sigma \rightarrow A$ 
2: Choose an  $n$ -by- $m$  orthonormal matrix  $V_0$ 
3:  $EigSpace = []$ 
4:  $j = 1$ 
5: for  $k = 0, 1, 2, \dots$  do
6:    $SuccessFlag = False$ 
7:   Compute all the eigenpairs of  $V_k^T A V_k s = \lambda s$ .
8:   Select the eigenpair  $(\theta_k, s_k)$  with  $\|\theta_k\|$  is smallest and  $\|s_k\|_2 = 1$ .
9:   Compute
10:  Compute  $u_k = V_k s_k$  and  $r_k = (A - \theta_k I) u_k$ .
11:  if ( $\|r_k\| < \epsilon$ ) then
12:     $SuccessFlag = True$ 
13:     $\lambda_j = \theta_k + \sigma$ ,  $x_j = u_k$ ,  $j = j + 1$ ,  $EigSpace = [EigSpace, u_k]$ 
14:    if  $j > N$  then
15:      Stop
16:    else
17:       $V_k \perp EigSpace$ 
18:    end if
19:  end if
20:  if  $SuccessFlag == False$  then
21:    Solve (approximately) a  $t_k \perp u_k$  from
22:     $(I - u_k u_k^T)(A - \theta_k I)(I - u_k u_k^T)t = -r_k$ .
23:    Orthogonalize  $t_k \perp V_k$  and  $t_k \perp EigSpace \rightarrow v_{k+1}$ ,  $V_{k+1} = [V_k, v_{k+1}]$ 
24:  end if

```

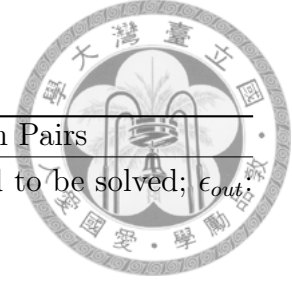


Algorithm 2.3 Shift-Invert Residual Arnoldi for Single Eigen Pair

Input: A : The target matrix; ϵ_{out} : The outer tolerance; σ : The target

Output: (θ, x) : Eigen Pair Solved

- 1: $A - \sigma \rightarrow A$
 - 2: Choose an n -by- m orthonormal matrix V_0
 - 3: **for** $k = 0, 1, 2, \dots$ **do**
 - 4: Compute all the eigenpairs of $V_k^T A V_k s = \lambda s$.
 - 5: Select the eigenpair (θ_k, s_k) with $\|\theta_k\|$ is smallest and $\|s_k\|_2 = 1$.
 - 6: Compute $u_k = V_k s_k$ and $r_k = (A - \theta_k I)u_k$.
 - 7: If $(\|r_k\| < \epsilon)$, $\lambda = \theta_k + \sigma$, $x = u_k$, Stop
 - 8: Solve (approximately) a $t_k \perp u_k$ from
 - 9: $(A - \theta_k I)t = -r_k$.
 - 10: Orthogonalize $t_k \perp V_k \rightarrow v_{k+1}$, $V_{k+1} = [V_k, v_{k+1}]$
-



Algorithm 2.4 Shift-Invert Residual Arnoldi for Multiple Eigen Pairs

Input: A : The target matrix; N : How many eigen pairs needed to be solved; ϵ_{out} : The outer tolerance; σ : The target

Output: $(\theta_1, x_1) \dots (\theta_N, x_N)$: Eigen Pair Solved

```

1:  $A - \sigma \rightarrow A$ 
2: Choose an  $n$ -by- $m$  orthonormal matrix  $V_0$ 
3:  $EigSpace = []$ 
4:  $j = 1$ 
5: for  $k = 0, 1, 2, \dots$  do
6:    $SuccessFlag = False$ 
7:   Compute all the eigenpairs of  $V_k^T A V_k s = \lambda s$ .
8:   Select the eigenpair  $(\theta_k, s_k)$  with  $\|\theta_k\|$  is smallest and  $\|s_k\|_2 = 1$ .
9:   Compute  $u_k = V_k s_k$  and  $r_k = (A - \theta_k I) u_k$ .
10:  if ( $\|r_k\| < \epsilon$ ) then
11:     $SuccessFlag = True$ 
12:     $\lambda_j = \theta_k + \sigma$ ,  $x_j = u_k$ ,  $j = j + 1$ ,  $EigSpace = [EigSpace, u_k]$ 
13:    if  $j > N$  then
14:      Stop
15:    else
16:       $V_k \perp EigSpace$ 
17:    end if
18:  end if
19:  if  $SuccessFlag == False$  then
20:    Solve (approximately) a  $t_k \perp u_k$  from
21:     $(A - \theta_k I)t = -r_k$ .
22:    Orthogonalize  $t_k \perp V_k$  and  $t_k \perp EigSpace \rightarrow v_{k+1}$ ,  $V_{k+1} = [V_k, v_{k+1}]$ 
23:  end if

```

2.2 Inexact Stopping Criteria in Inner Loops



In this section, we will introduce the four stopping criteria in this paper.

- Fixed

The first one is Algorithm 2.5, which is the simplest case, called fixed stopping criterion. In fixed stopping criterion, given a inner tolerance ϵ_{in} , if the residual is smaller than the tolerance, then the loop will stop.

- H.N.

The second one is H.N. stopping criterion, Algorithm 2.6 [4], which is developed by Hochstenbach and Notay. In H.N. stopping criterion, the inputs are matrix A , current inner iteration result x_i , right hand side b , outer tolerance ϵ_{out} , current outer iteration result u_k , the shift η in the linear system and three threshold τ_1, τ_2, τ_3 . To note that, in SIRA algorithm the η is always zero and the thresholds are tunable. τ_1 and τ_2 are the reducing coefficients of to tell when to calculate required variable β . Due to the performance issue, the variable β will only be calculate at most twice, and the two occasions are when $\|r_{in}\| < \tau_1\|b\|$ and $\|r_{in}\| < \tau_2\|b\|$. As any one of the conditions is satisfied, the variable β will be calculated, and s, g_k and r_{eig}^k will also be calculated. Then those conditions can be checked. The more details could be found in Hochstenbach's paper [4].

- Constant Scaling

Inspired by the H.N. stopping criterion, we develop other two stopping approaches. One is the Algorithm 2.7, called constant scaling stopping criterion, which means inner loop's tolerance is the current outer's residual scaled by a constant. The constant is set as ten in default. We observe the residual of behavior in the H.N. stopping criterion, and we found that the behaviors

of the last inner residual in each inner iteration followed the pattern of outer residual. Hence we set the inner tolerance to be of higher a order than the current outer residual.



- Pocket

After doing more experiments, we observe there are some convergence patterns of inner iterations in our test cases. One is the monotone convergent type, which means the inner residual decay exponentially and it's the best type. The second type is harmonic type, in which case the residuals will oscillation decay. The last one is the unstable type, in which the residual decays like random walk, sometimes oscillation, sometimes increase unexpected.

From our observation, H.N. stopping criterion and constant scaling stopping criterion, we come out the Pocket stopping criterion, in which we set the ϵ_{in} as $\sqrt{10}r_{out}$. Moreover we kept the best iteration result $x_{best} = x_i$ and continued the iteration until the following conditions all happens:

Condition 1: The residual dose not improve significantly, and the default set is that it must reduce 1/10 from x_{best} .

Condition 2: The iteration times from last getting x_{best} exceeding the *ImproveStep*, and the default set is $\frac{3Dim}{1000}$.

Algorithm 2.5 Fixed Stopping Criterion

if $\|Ax_i - b\|_2 < \epsilon_{inner}$ **then**

 Return True

else

 Return False

end if



Algorithm 2.6 H.N. Stopping Criterion

$$g_k = \|r_{in}\|, s = \|x_i\|, \alpha = 1$$

$$r_{eig}^k = \sqrt{\frac{g_k^2}{1+s^2} + \left(\frac{\beta s}{1+s^2}\right)^2}$$

$$\theta = u_k^* A u_k$$

$$\beta = \|\theta - \eta + u_k^*(A - \eta I)x_i\|$$

Above variables are calculated only twice:

i.e. when $\|r_{in}\| < \tau_1 \|b\|$ and $\|r_{in}\| < \tau_2 \|b\|$

$$Flag = (k > 1) \text{ and } \begin{cases} \left(\frac{g_{k-1}}{g_{k-2}}\right)^2 > \left(2 - \left(\frac{g_{k-1}}{g_{k-2}}\right)^2\right)^{-1}, & \text{if a norm-minimizing method} \\ & \text{(GMRES, QMR) is used} \\ g_k > g_{k-1}, & \text{otherwise} \end{cases}$$

if $g_k < \tau_1 \|r\|_2$ **then**

if $r_{eig}^{(k)} < \epsilon^{(out)}$ **then** Return True

else if $\frac{\beta s}{\alpha(1+s^2)} > \frac{\epsilon}{2}$ **then**

if $g_k < \tau_3 \frac{\beta s}{\sqrt{1+s^2}}$ **then**

 Return True

else if *Flag* is True **then**

 Return True

end if

end if

else

 Return False

end if

Algorithm 2.7 Constant Scaling

if $\|Ax_i - b\| < \frac{\epsilon_{out}}{C}$ **then**

 Return True

else

 Return False

end if



Algorithm 2.8 Pocket Stopping Criterion

```
Step = Step + 1
if  $\|Ax_i - b\| < \frac{r_{best}}{10}$  then
    Step = 0
end if
if  $\|Ax_i - b\| < r_{best}$  then
     $x_{best} = x_i$ 
     $r_{best} = \|Ax_i - b\|$ 
end if
if  $\|Ax_i - b\| < \frac{\epsilon_{out}}{\sqrt{10}}$  then
    Return True
else if Step > ImproveStep then
    Return True
else
    Return False
end if
```

2.3 Mixed Precision Algorithms

After constructing the stopping criteria, we consider how to apply mixed precision approach on those stopping criteria, that is we need to decide when to use single precision to solve the linear system when to use double precision.



In fixed stopping criterion, it's hard to have a strategy to switch precision. Hence we all use double precision in fixed stopping criterion. In H.N. stopping criterion, we found that all the exiting conditions must satisfy $\|r_{in}\| < \tau_1 \|r_{out}\|$, so we set the condition to use single precision is : $\tau_1 \|r_{out}\| > 10^7$. However under this condition, it does not guarantee that the single precision could solve the linear system successfully. Hence in our approach it is an experiment concept. And in constant scaling and Pocket stopping criteria, it's relative easy to decide when to use single precision. That is when ϵ_{in} is greater than 10^7 , We use the single precision to solve the linear system. The main idea is similar to H.N. stopping criterion. Note that the upper bound being smaller than 10^7 does not guarantee the success of using single precision to solve the linear system.

2.4 Arithmetic Intensity Effects

- V^*AV

Besides using mixed precision, we also enhance the algorithm by using the updating method of computation of V^*AV in the Algorithm 2.3 and Algorithm 2.4. Since the $V_{k-1}^*AV_{k-1}$ is exactly the on left and upper corner of $V_k^*AV_k$, i.e.

$$V_k^*AV_k[1 : k - 1, 1 : k - 1] = V_{k-1}^*AV_{k-1}$$

We only need to update the vector $V_k^*AV_k[:, k]$, so that we reduce the *BLAS2* operation to *BLAS1* operation. However because the matrixes size and number of nonzero are also affect the preformance, using updating method may not be the best choice. Which method being the best depends on the matrixes'

condition. In the Section 3, there are some performance results and discussion between updating method and non-updating method.

Note that, we use the Modified Gram Schmidt (MGS) as orthogonalization approach. Due to the numerical stable issue, we may need to reorthogonalize V_k sometimes.[10]



- Preconditioners

In our experiments, we use three preconditioners. In general cases testing, we apply Jacobi [11, 12, 13] and FSAI [14] as preconditioner. In our special application photonic crystal [6], we use fast fourier transform (FFT) as preconditioner. Jacobi is extracting the diagonal of matrix as preconditioner, and FSAI is to minimize the Frobenius norm of $\|I - GL\|_f$ where L is the exact lower triangular part of A and $G = M^{-1}$ is the preconditioner. FSAI and FFT are arithmetic intensive operation with respect to Jacobi. Hence those two preconditioners may have better performance on GPU.

2.5 Search Subspace Effects

- Search Subspace Restarting

We also did the experiments of remaining different many sub dimension of V_k after restarting. That is one of the case is we always remain only one vector of V_k after restarting, and the other case is we remain five vectors of V_k after restarting. We try to tell is there any different performance between those two conditions.

All of the method we mentioned above could be implemented on the environment of all CPU system. However we implement them on a machine with GPU. We regard GPU as an accelerator, and some of the operations prefer GPU more than CPU such like FFT, FSAI preconditioners and BLAS3 operation.

In next chapter, we will show you the numerical result of our experiment, and there is some discussion later.

3 Numerical Results and Discussion

In this chapter, we will discuss the numerical results of our experiment. First of all, we will introduce how we implement the algorithm and what kind of test problems we did. Note that we use the absolute error in our experiment.

Some remarks:

- Relative error and absolute error

The definition of the absolute error of inner loop is :

$$\|Ax_i - b\|_2$$

and the definition of the relative error of inner loop is :

$$\|Ax_i - b\|_2 / \|b\|$$

Since the right hand side b is the outer residual r_{out} , the relative error could be rewritten as :

$$\|Ax_i - b\|_2 / \|r_{out}\|$$

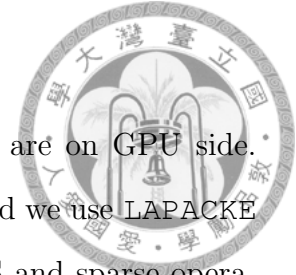
For example, in fixed stopping criterion, assume that $\|Ax_i - b\|_2 = 10^{-5}$ and $\|r_{out}\| = 10^{-6}$, then the absolute error is 10^{-5} , but the relative error is $\|Ax_i - b\|_2 / \|r_{out}\| = 10^{-5} / 10^{-6} = 10$. We could tell that in fixed stopping criterion, using absolute error will have loosing tolerance than relative error in most iterations. In this paper, we used absolute error to measure the error in inner loop.

- Iteration number of relative error and absolute error

In our experiments results, the iteration number of each inner loop is almost constant in using relative error, and using absolute error had the similar results.

More details of experiments results are in Appendix A.

3.1 Implementation and Testing Problems



Our code mainly follows Algorithm 2.4, and most of them are on GPU side. Only solving the small subspace's eigenvalues is on CPU side, and we use LAPACKE `dgeev` and LAPACKE `zgeev` [15] to do that. All of the BALS and sparse operation are using cuBLAS and cuSparse [16]. We use the PARALUTION library [17] to solve the linear system part, and in the library we can select the linear solver and preconditioner we want. In our implementation, we mainly use conjugate gradient (CG) as our linear solver. We also tried use BiCGStab and GMRES to solve the linear system, but the CG had the best performance. Note that, since someone may need to defined they own matrix-vector multiplication of matrix A , we defined the function pointer to handle this. That is for general case, we use functions in PARALUTION to do the matrix-vector multiplication, and if users want to define their own, then they just need to set the function pointer. In our photonic crystal application, we defined our own matrix-vector multiplication of matrix A , which involving FFT operations, and we use `cuffft` [18] to do that.

Table 3.1 are the list of all of the matrixes we run the test, and we separate them in four sets by size. Our general case are downloaded from the University of Florida sparse matrix collection [5]. We only select the SPD and whose size is greater than ten thousand. Note that in Table 3.1, there is a matrix, named `nos3`, whose size is smaller than one thousand, and it's just used for debugging. In each test matrixes, we solved three eigenvalues in each stopping criteria. After running that, because the constant scaling stopping criterion's convergence is not good, we collected the matrix set in which there are matrix convergent with all of the stopping criteria except constant scaling.

In photonic crystal application, we run the simple cubic cases, and there are 47 k vectors in each run. For each k vector, we solve one eigenvalue.

3.2 SPD Problem Collection



- Timing Comparisons of Stopping Criteria

Pocket mixed has the best performance in overall. Table 3.9 shows the sum of computation time in each data set and overall, and Pocket mixed has is the best in each set. Table 3.10 shows the 1st place counts for each stopping criteria, and Pocket mixed has the most 1st place counts. Table 3.11 shows the sum of inner iteration number, we can tell that Pocket mixed and H.N. double have the similar iteration number, but there are some iteration using single precision in Pocket mixed. Hence Pocket mixed has the best performance. Figure 3.3 give the overall results of speedup for H.N. and Pocket w.r.t fixed stopping criterion.

- r_{in} versus r_{out}

Figure 3.4 show the r_{in} and r_{out} record of matrix *boneS01*, in each stopping criteria. In those figures, we could tell that H.N. and Pocket has similar behavior. Both of them capture the trend of r_{out} . Moreover, fixed stopping criterion's iteration number is smallest, and H.N and Pocket are similar. So we could tell fixed solved the linear system strictly, such that it only need few outer iteration number. But the total cost may higher than the other two.

- Pocket's Advantage and Mixed Precision Effects

Figure 3.5 are matrix *thermomech_TC*'s counts of calling linear solver and the total inner computation time. H.N. double and Pocket double have similar counts, but Pocket's performance is better. Since the judge cost in Pocket is cheaper than H.N.. From Figure 3.5, we could tell that four of those stopping criteria has similar counts of calling number, but the performance are different. There are two reasons to explain: first, Pocket has cheaper cost in judging whether existing the loop. Second, there are mixed precision effect that is there are many iterations using single precision to save a lot of time.

- Arithmetic Intensity Effects

- V^*AV

Table 3.12 shows the sum of computation time comparison of using updating method and without using it. Table 3.13 shows the 1st place counts between two of them. We could tell that the results is as our expect. Using updating method's performance is better.

- Preconditioners Effects Table 3.14 shows the sum of the total computation time in each preconditioners. Table 3.15 shows the sum of iteration number in each preconditioners. FSAI is the best preconditioner candidate in our experiments. There are two reason to explain it first, FSAI has the smallest iteration number. Second, though FSAI need higher computation amount, but its high computation intensity favoring GPU such that the higher computation amount won't decrease the performance.

- Search Subspace Effects

- Restarting

We original expect the remaining vectors after restarting will affect the convergence behavior. But after our experiment, the convergence is not relative to remaining vectors. That is remaining vectors are five or only one do not affect the matrix is convergent or not.

- Inner and Outer Loop Costs

Figure 3.1 to Figure 3.2 show the percentage of the inner and outer computation time in fixed and Pocket mixed stopping criteria. The inner ratio become greater as the dimension increasing, but different to fixed, Pocked mixed the inner proportion is smaller in large size. And it's one of the reason of why mixed could save time, because it need fewer time in inner loop. The inner ratio become greater as the dimension increasing, because the dimension is larger the linear system may be harder to solve.



3.3 Photonic Crystal

In this section, we show the results of the photonic crystal application [6].



- Comparison Plan

In this section, we will show the performance results on photonic crystal application. The first is total computation time, and then we show mixed precision effect. We will show the updating method is workable in this application, and compare the inner and outer loop's cost, parameters effects and matrix size effects.

- Timing Compare with HN and Fixed

Table 3.16 shows sum of total time in solving 46 k vectors in each stopping criteria. We could tell the H.N. mixed and Pocket mixed have the best results.

- Mixed Precision Effects

To tell the mixed precision effects, we take the H.N. as example. Figure 3.7a and Figure 3.7b show how mixed precision save the time. H.N. double and H.N mixed have similar inner iteration times, but H.N. mixed has some iteration in single precision hence the total time is fewer than H.N. double. Moreover we can tell that the mixed precision effect is not so significant in photonic crystal application, because in mixed method there are still many double iteration in the loop, which let the effect be not so significant.

- Arithmetic Intensity Effects

Figure 3.6 shows the updating method is useful in photonic crystal case. Using updating method may save many time in outer loop iteration, and in fixed case the saving is more. Because fixed case has higher proportion of outer loop than the others.

- Inner and Outer Loop Costs

Figure 3.8a and Figure 3.8b show the percentage of the inner and outer loop's computation time in different parameter of H.N.. In both parameters, the

outer time is in major and small τ has larger percentage of inner time. Since the inner loop is relative cheap in this application, more proportion in inner more time saving.



- Parameters Effects

Figure 3.8c and Figure 3.8d show that the outer iteration times in different parameter. As we said in previous, the small τ gets the larger percentage of inner time, and from Figure 3.8c and Figure 3.8d we can tell it's because of reducing the outer iteration time. Moreover we can tell that fixed stopping criterion not only has more inner iteration time, but has more outer iteration time. And this tell us that the low accuracy of inner loop's solution does not imply higher outer iteration time.

- Matrix Size Effects

Figure 3.7c and Figure 3.7d show that the total computation time in different matrix size. The larger size, the more performance gain in mixed precision method. The reason is that the speed up of fft is more significant in large size

3.4 Precision Performance Analysis

In this section, we show the results of single time and double time experiments. And we only discuss the subset of our general case and photonic crystal application.

- Relations of Cache Hit Ratio and Mixed Precision Computations

In this section, we try to tell cause of the double and single precision cost's ratio. Our guess is the cache hit, in Table 3.5 show that the higher cache hit ratio of cache hit ratio of double and cache hit ratio of single, the higher cost ratio between double and single precision inner loop cost.

However in photonic crystal it is not the case. In Table 3.6 and Table 3.7 it shows the fft's computation time and cache hit, where the $fft = 1$ and $fft = -1$ means the fft and inverse fft. We could tell that even the cache hit ratio are similar between double and single precision, the computation time is

increasing with the dimension (dimension= $2n_1^3$). We think the reason is most of cache hit of the kernels in fft operation are zero, especially for the Radix kernel, which is also the large part in fft operation. Hence we also consider the flops per unit time as a cause, and built up a model.

$$\begin{aligned} \frac{D_{cost}}{S_{cost}} = & CacheHitFlag(\alpha_{cache} \frac{S_{CacheHitRatio}}{D_{CacheHitRatio}} + \beta_{cache}) \\ & + (1 - CacheHitFlag)(\alpha_{flops} \frac{S_{FlopsRatio}}{D_{FlopsRatio}} + \beta_{flops}) \end{aligned} \quad (1)$$

where *CacheHitFlag* is the flag of whether the cache hit ratio is non zero, and the $S_{CacheHitRatio}$ and $D_{CacheHitRatio}$ are the cache hit ratio of single and double precision operations. The $S_{FlopsRatio}$ and $D_{FlopsRatio}$ are the flops in unit time of single and double precision operations. The parameters α and β are calculated by regression.

We may regard the first term as the communication effect and the second term as computation effect. Figure 3.10 is the results of our model prediction and experiments. We can tell the our model almost fits the experiment results.

- Mixed Precision Effects on Overall Performance

Assume that the mixed precision and double precision stopping criteria have the same outer computation cost. We may build up a model to tell how the mixed precision affect the over performance in SIRA. The mixed precision performance gain is in the following formula:

$$Outer\% + Inner\%(Double\% + Single\% * \frac{D_{cost}}{S_{cost}})$$

where *Outer%* is the percentage of the outer loop computation time in overall computation time, and *Inner%* is the percentage of the inner loop computation time in overall computation time. That is $Outer\% + Inner\% = 1$ in double precision method. And *Single%* means the single precision computation time percentage of all inner computation time. So $Single\% + Double\% = 1$. The $\frac{D_{cost}}{S_{cost}}$ means the ratio of double precision computation time and single precision

computation time. In this model we can tell that

- less *Outer%*
- higher *Single%*
- higher $\frac{D_{cost}}{S_{cost}}$



may have more significant mixed precision performance gain. And in photonic crystal application, the *Outer%* is too large, so the performance gain is not significant. Figure 3.9 is the results of our model predictions and experiments. We can tell the our model almost fits the experiment results.



Matrix Name	Dim	nnz	Density
nos3	960	15844	1.719E-02
ted B unscaled	10605	144579	1.286E-03
msc10848	10848	1229778	1.045E-02
bcsstk17	10974	428650	3.559E-03
bcsstk18	11948	149090	1.044E-03
cbuckle	13681	676515	3.614E-03
crystm02	13965	322905	1.656E-03
Pres Poisson	14822	715804	3.258E-03
bcsstm25	15439	15439	6.477E-05
Dubcova1	16129	253009	9.726E-04
bodyy4	17546	121938	3.961E-04
nd6k	18000	6897316	2.129E-02
Trefethen 20000	20000	554466	1.386E-03
crystm03	24696	583770	9.572E-04
smt	25710	3753184	5.678E-03
wathen100	30401	471601	5.103E-04
nd12k	36000	14220946	1.097E-02
pdb1HYS	36417	4344765	3.276E-03
wathen120	36441	565761	4.260E-04
bcsstm39	46772	46772	2.138E-05
crankseg 1	52804	10614210	3.807E-03
nasasrb	54870	2677324	8.893E-04
Andrews	60000	760154	2.112E-04
crankseg 2	63838	14148858	3.472E-03
Dubcova2	65025	1030225	2.437E-04
qa8fm	66127	1660579	3.798E-04
cfdl	70656	1828364	3.662E-04
nd24k	72000	28715634	5.539E-03
oilpan	73752	3597188	6.613E-04
finan512	74752	596992	1.068E-04
apache1	80800	542184	8.305E-05
thermall	82654	574458	8.409E-05
consph	83334	6010480	8.655E-04

Table 3.1: Test Matrix Size between 10K and 100K

Matrix Name	Dim	nnz	Density
2cubes sphere	101492	1647264	1.599E-04
thermomech TK	102158	711558	6.818E-05
thermomech TC	102158	711558	6.818E-05
cfd2	123440	3087898	2.027E-04
boneS01	127224	6715152	4.149E-04
shipsec1	140874	7813404	3.937E-04
Dubcova3	146689	3636649	1.690E-04
bmwcra 1	148770	10644002	4.809E-04
G2 circuit	150102	726674	3.225E-05
shipsec5	179860	10113096	3.126E-04
thermomech dM	204316	1423116	3.409E-05
hood	220542	10768436	2.214E-04
BenElechi1	245874	13150496	2.175E-04
offshore	259789	4242673	6.286E-05
msdoor	415863	20240935	1.170E-04

Table 3.2: Test Matrix Size between 100K and 500K

Matrix Name	Dim	nnz	Density
parabolic fem	525825	3674625	1.329E-05
apache2	715176	4817870	9.420E-06
tmt sym	726713	5080961	9.621E-06
boneS10	914898	55468422	6.627E-05
ldoor	952203	46522475	5.131E-05
bone010	986703	71666325	7.361E-05
ecology2	999999	4995991	4.996E-06

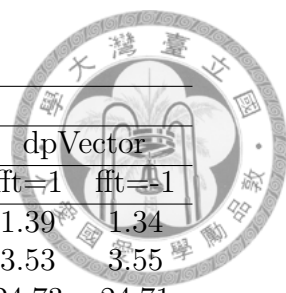
Table 3.3: Test Matrix Size between 500K and 1M

Matrix Name	Dim	nnz	Density
thermal2	1228045	8580313	5.690E-06
StocF-1465	1465137	21005389	9.785E-06
G3 circuit	1585478	7660826	3.048E-06

Table 3.4: Test Matrix Size Greater than 1M

Matrix Name	CG Time			csrMv L1 Cache Hit Rate		
	Single(ms)	Double(ms)	D/S	Single(%)	Double(%)	S/D
nos3	0.183	0.244	1.33	40.60	43.19	0.94
Pres Poisson	0.396	0.574	1.45	44.27	33.78	1.31
Dubcova1	0.249	0.385	1.55	43.13	31.59	1.35
Dubcova2	0.524	0.943	1.80	43.87	30.14	1.46

Table 3.5: CG Inner Time. The higher cache hit ratio of double and cache hit ratio of single, the higher cost ratio between double and single precision inner loop cost.



n_1	time(ms)									
	FFT	spRadix		spVector		dpRadix		dpVector		
	D/S	fft=1	fft=-1	fft=1	fft=-1	fft=1	fft=-1	fft=1	fft=-1	
16	1.10	1.00	0.97	0.54	0.56	2.43	2.37	1.39	1.34	
32	1.60	1.62	1.54	0.77	0.79	7.61	6.39	3.53	3.55	
64	2.05	9.82	9.85	4.53	4.48	53.36	51.58	24.73	24.71	

Table 3.6: FFT Time. Even the cache hit ratio are similar between double and single precision, the computation time is increasing with the dimension. The reason maybe most of cache hit of the kernels in fft operation are zero. See Table 3.7

n_1	L1 Cache Global Hit Rate(%)									
	FFT	spRadix		spVector		dpRadix		dpVector		
	D/S	fft=1	fft=-1	fft=1	fft=-1	fft=1	fft=-1	fft=1	fft=-1	
16	1.10	0.00	0.00	0.00	0.00	0.00	0.00	41.96	41.96	
32	1.60	0.00	0.00	42.14	42.14	0.00	0.00	42.32	42.32	
64	2.05	0.00	0.00	59.31	59.30	0.00	0.00	58.62	58.63	


Table 3.7: FFT Cache. Most of cache hit of the kernels in fft operation are zero, especially for the Radix kernel, which is also the large part in fft operation. Hence we could not tell the single double ratio by cache hit in this case.

Stopping Criteria	10K to 100K	100K to 500K	500K to 1M	Above 1M	Overall
Fixed	25	11	6	2	44
H.N. Double	24	10	7	2	43
H.N. Mixed	23	11	7	2	43
Pocket Double	25	10	7	2	44
Pocket Mixed	25	10	7	2	44

Table 3.8: Convergence Count in each Stopping Criteria. All of them have the similar convergence.

Stopping Criteria	10K to 100K	100K to 500K	500K to 1M	Above 1M	Overall
Fixed	148.54	734.08	5498.59	495.26	6876.47
H.N. Double	60.08	307.73	4943.69	174.28	5485.79
H.N. Mixed	52.13	242.08	4811.95	163.34	5269.51
Pocket Double	53.99	302.49	4300.49	170.9	4827.87
Pocket Mixed	48.78	214.07	3347.94	155.39	3766.17

Table 3.9: Sum of Total Time in Each Stopping Criteria. Pocket mixed has the best performance in each size set.



Stopping Criteria	10K to 100K	100K to 500K	500K to 1M	Above 1M	Overall
Fixed	2	0	0	0	2
H.N. Double	2	0	0	0	2
H.N. Mixed	1	3	1	0	5
Pocket Double	2	2	0	0	4
Pocket Mixed	11	5	5	2	23

Table 3.10: 1st Place Accumulation of Total Time in Each Stopping Criteria (sec.). Pocket mixed always gets the first place in our test cases.

Stopping Criteria	10K to 100K	100K to 500K	500K to 1M	Above 1M	Overall
Fixed	51842	173953	293201	64109	583105
H.N. Double	17571	61773	212444	19554	311342
H.N. Mixed	20536	63731	230526	20277	335070
Pocket Double	14949	61084	185313	19556	280902
Pocket Mixed	16776	58577	168556	19773	263682

Table 3.11: Sum of Total Inner Iteration Number in Each Stopping Criteria. Pocket method have the smallest inner iteration number, and it may one of the reason to tell why Pocket is the best stopping criteria.

VAV	10K to 100K	100K to 500K	500K to 1M	Above 1M	Overall
w/ Updating Method	62.4	214.07	3544.92	155.39	3976.78
w/o Updating Method	86.77	229.46	3670.66	160.53	4147.42

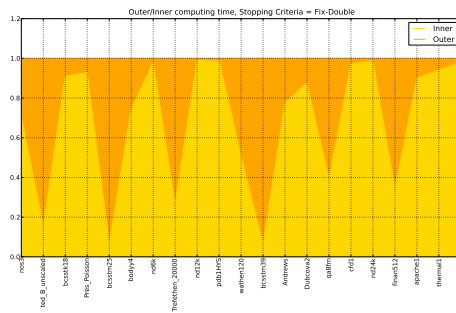
Table 3.12: Sum of Total Time of Pocket Mixed with Updating and without Updating (sec.). Using updating method has better performance than non using, and this results are the same as we expected.

VAV	10K to 100K	100K to 500K	500K to 1M	Above 1M	Overall
w/ Updating Method	18	6	5	1	30
w/o Updating Method	2	4	2	1	9

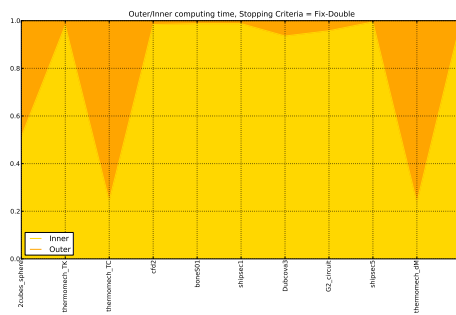
Table 3.13: 1st Place Accumulation of Pocket Mixed with Updating and without Updating. Using updating method almost gets the first place in our test cases.

Preconditioner	10K to 100K	100K to 500K	500K to 1M	Above 1M	Overall
FSAI	45.2	185.36	3544.92	155.39	3930.87
No Preconditioner	417.92	400.87	7232.59	527.11	8578.48
Jacobi	307.72	353.19	3887.28	181.69	4729.89

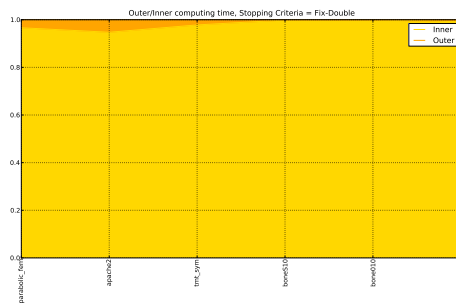
Table 3.14: Sum of Total Time of Pocket Mixed with Different Preconditioners (sec.). FSAI is the best preconditioner in our experiment.



(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k



(c) Matrix of Dimension 500k to 1000k



(d) Matrix of Dimension 1000k up

Figure 3.1: Outer/Inner time ratio of fixed stopping criteria. The inner ratio become greater as the dimension increasing.

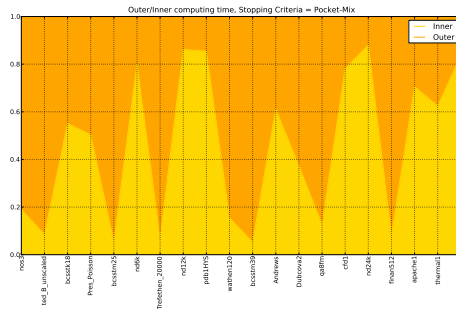


Preconditioner	10K to 100K	100K to 500K	500K to 1M	Above 1M	Overall
FSAI	15664	51436	178221	19773	265094
No Preconditioner	220779	213762	816370	141444	1392355
Jacobi	95187	166846	424019	40184	726236

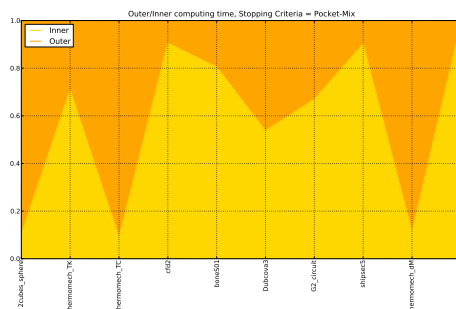
Table 3.15: Sum of Total Inner Iteration Number of Pocket Mixed with Different Preconditioners. FSAI has the smallest iteration number, and it may be the reason of why it had the best performance than Jacobi and no using preconditioner.

Stopping Criteria	Sum of Total Time
Fixed	135.08
H.N. Double	78.29
H.N. Mixed	72.98
Pocket Double	83.93
Pocket Mixed	76.29

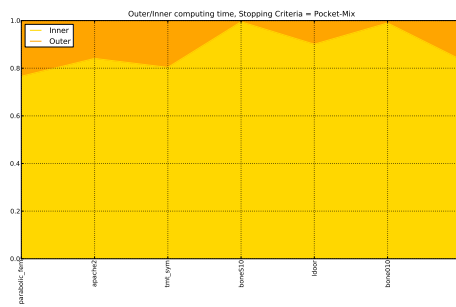
Table 3.16: Sum of Total Time of all K Vectors of Photonic Crystal (sec.). The H.N. mixed and Pocket mixed have the best performance.



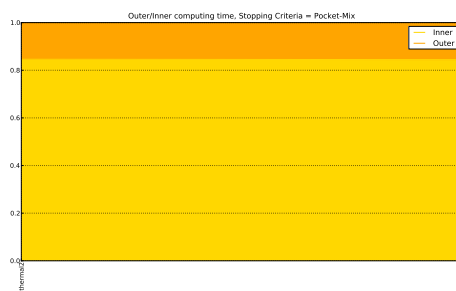
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k



(c) Matrix of Dimension 500k to 1000k



(d) Matrix of Dimension 1000k up

Figure 3.2: Outer/Inner time ratio of Pocket Mixed stopping criteria. The inner ratio become greater as the dimension increasing, but different to fixed in large size the inner proportion is smaller. And it's one of the reason of why mixed could save time, because it need fewer time in inner loop.

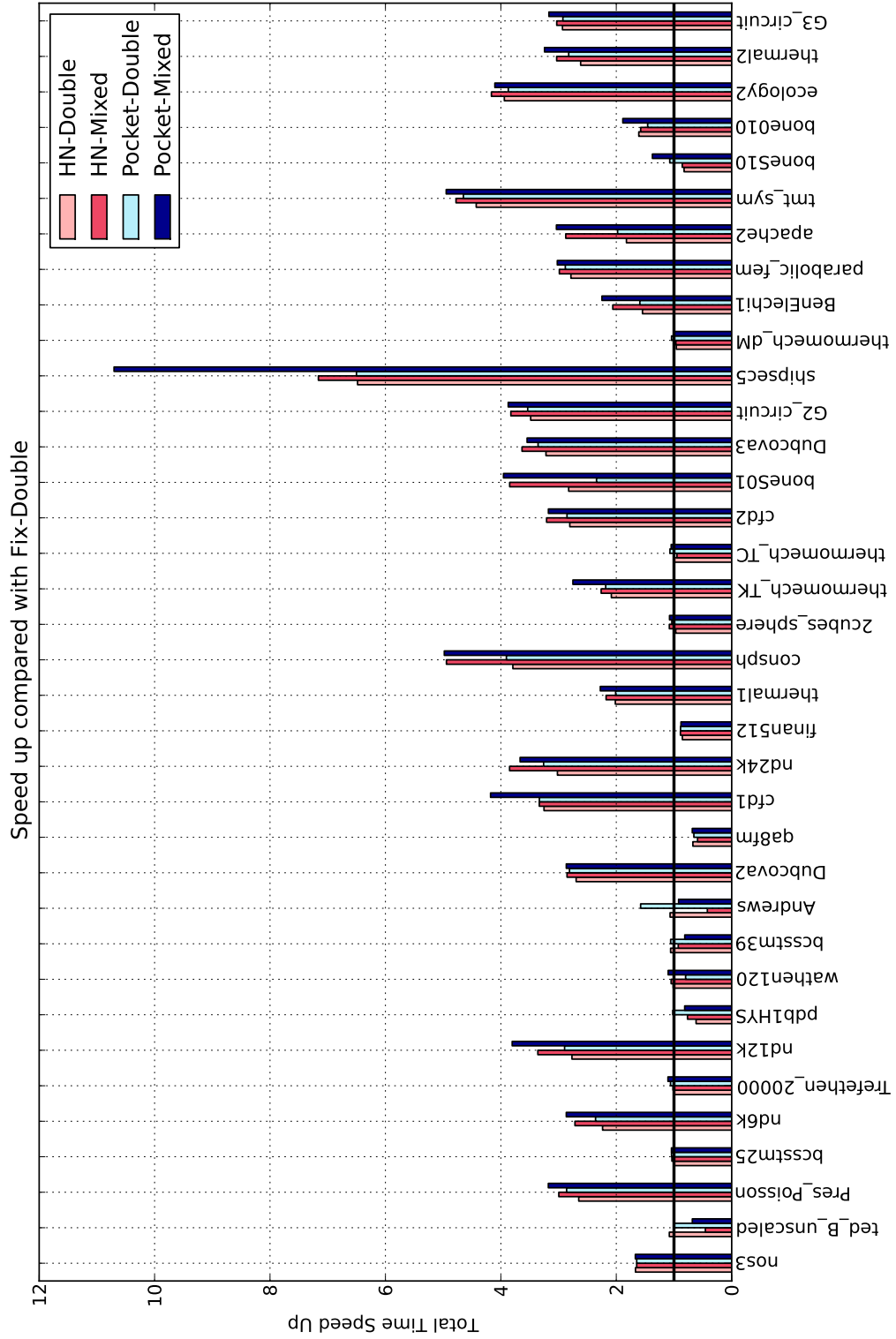
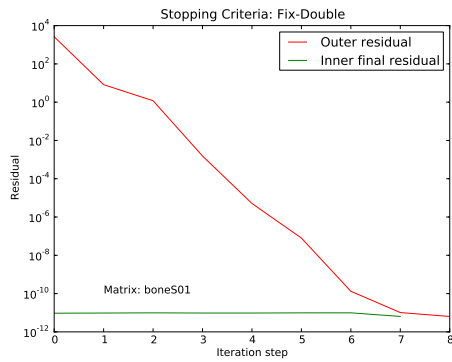
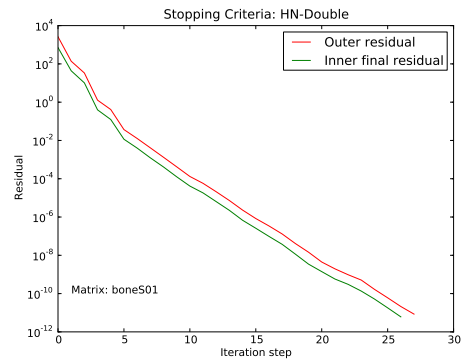


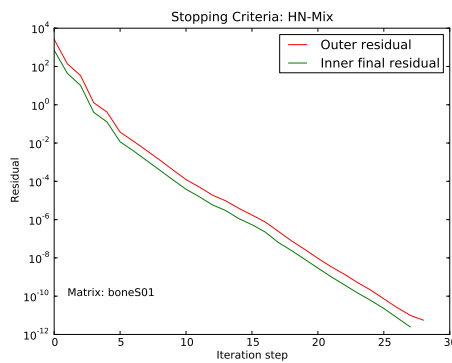
Figure 3.3: Speed Up of H.N. and Pocket w.r.t. fixed, Pocket Mixed has the best speed up in most case.



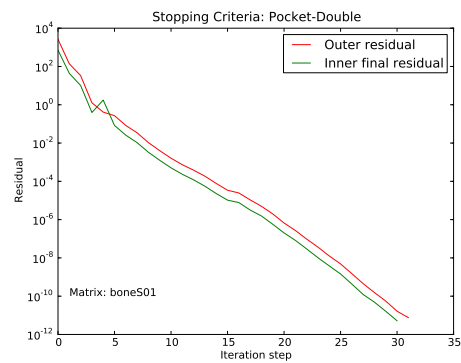
(a) Fixed-Double



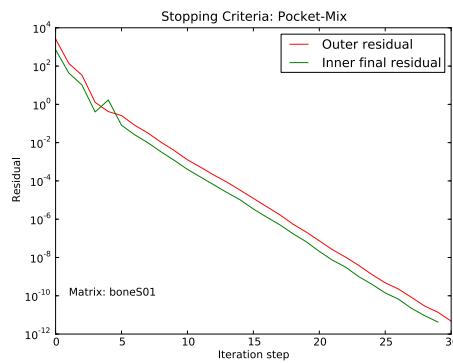
(b) HN-Double



(c) HN-Mixed



(d) Pocket-Double



(e) Pocket-Mixed

Figure 3.4: Residual behavior, H.N. and Pocket has similar behavior. Both of them capture the trend of r_{out}

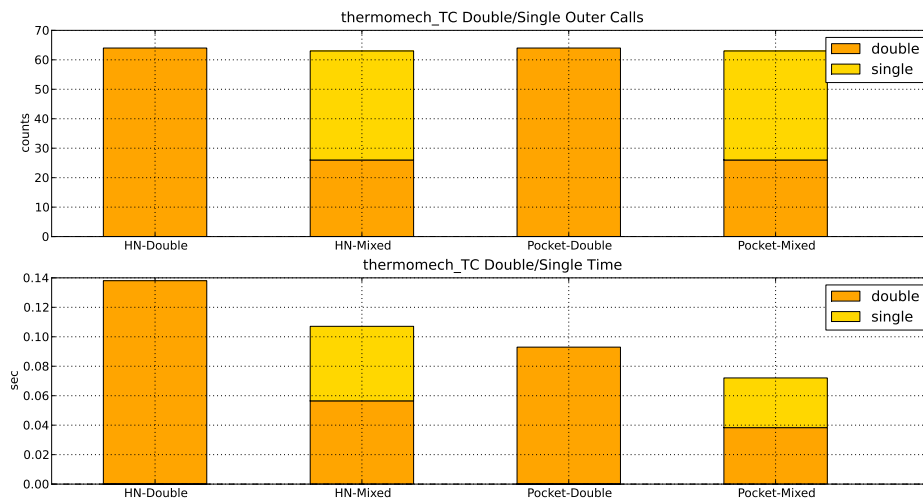
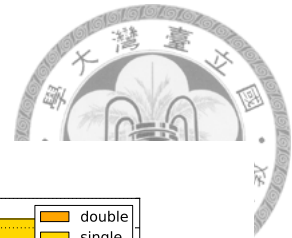


Figure 3.5: Matrix: thermomech_TC’s counts of linear solver and total computation time. H.N. double and Pocket double have similar counts, but Pocket’s performance is better. Since the judge cost in Pocket is cheaper than H.N.

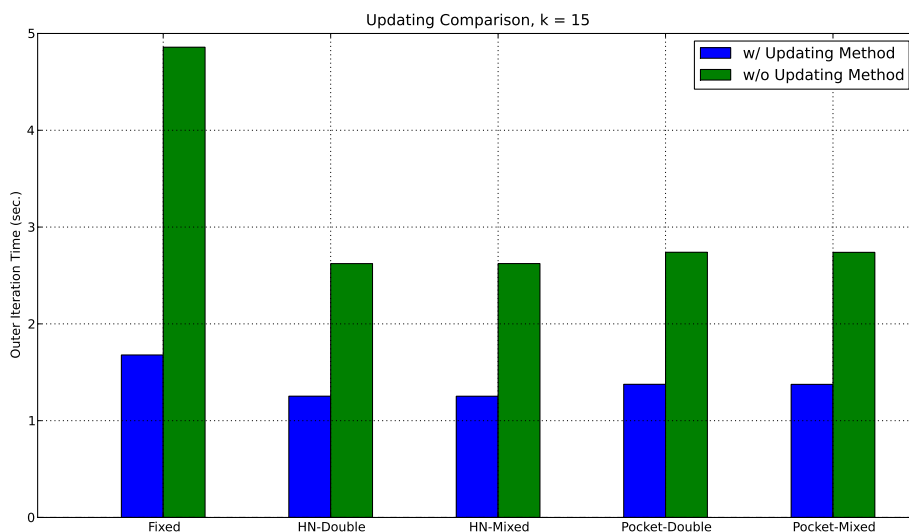
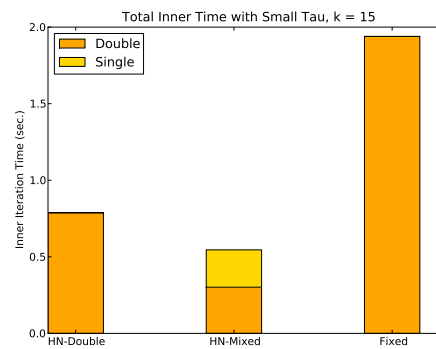
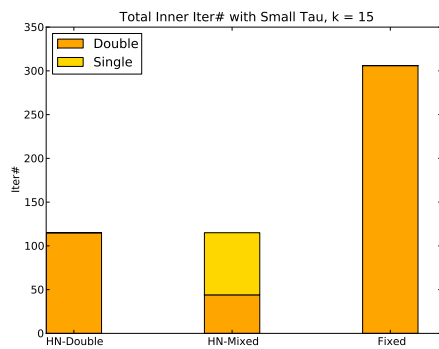
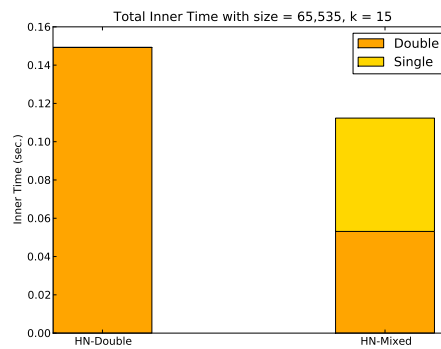
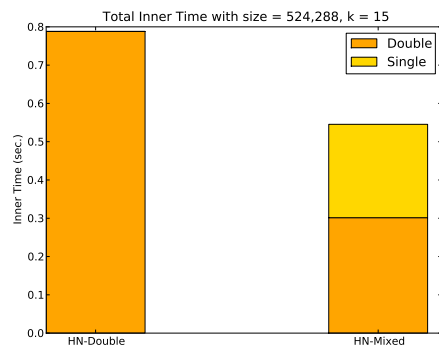


Figure 3.6: Compare if Using Updating Method. Using updating method may save many time in outer loop iteration, and in fixed case the saving is more. Because fixed case has higher proportion of outer loop than the other two.

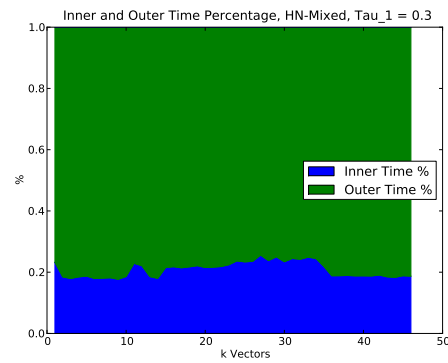
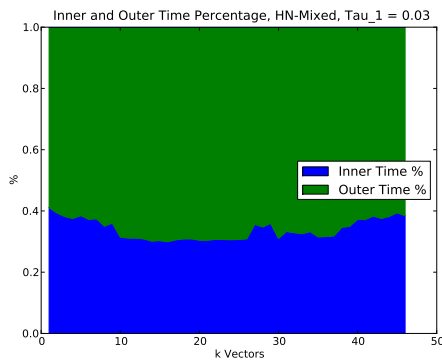


(a) Total Inner Iteration Number with Small τ $k = 15$. (b) Total Inner Iteration Time with Small τ $k = 15$.

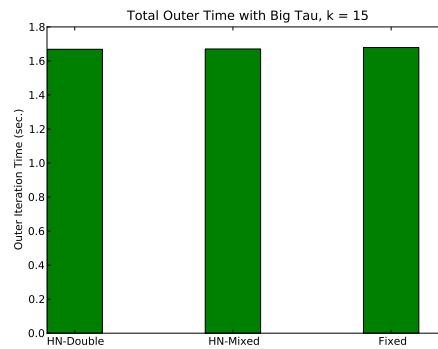
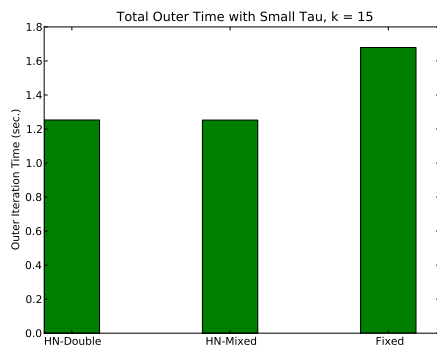


(c) Total Inner Time with size = 524288 $k = 15$. (d) Total Inner Time with size = 65526 $k = 15$.

Figure 3.7: Inner Outer Comparison. H.N. double and H.N. Mixed have the same iteration number, and We could tell the mixed precision effect by H.N. Mixed bar and H.N. double; The larger size , the more performance gain in mixed precision method. The reason is that the speed up of fft is more significant in large size.



(a) Inner and Outer Time Percentage with H.N. Mixed Small τ . (b) Inner and Outer Time Percentage with H.N. Mixed Large τ



(c) Total Outer Time with Small τ k = 15 (d) Total Outer Time with Large τ k = 15

Figure 3.8: Parameter Effects. The proportion is more balance in small τ , and it's one of the reason to tell why it saving time. Since inner iteration is relative cheaper in this application, more proportion in inner may have better performance.

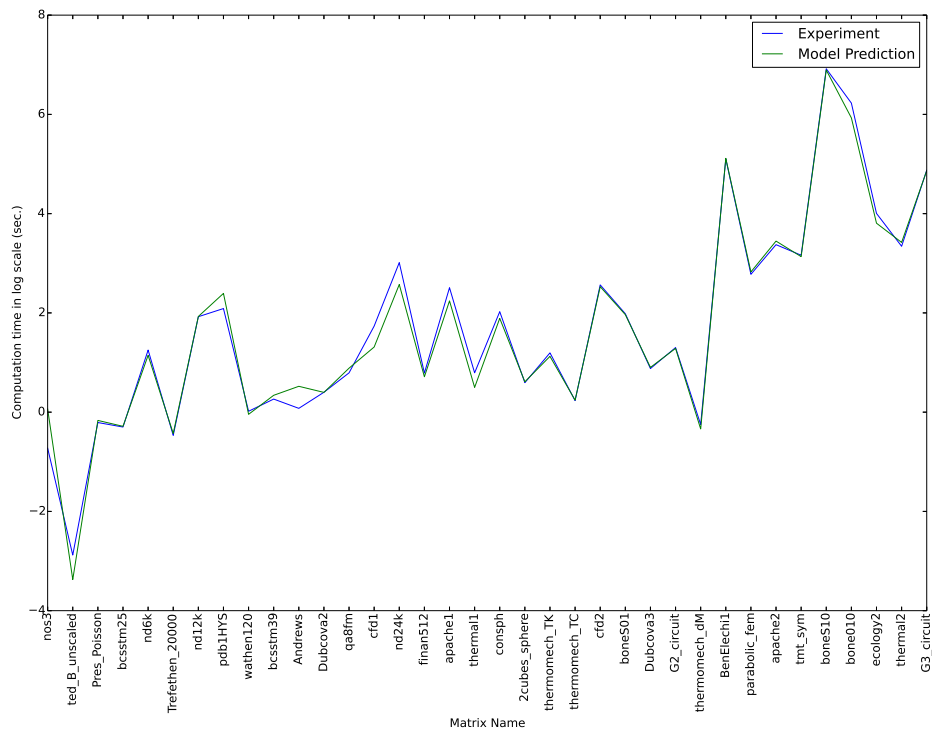


Figure 3.9: Performance model results. Our model prediction fits the experiments well.

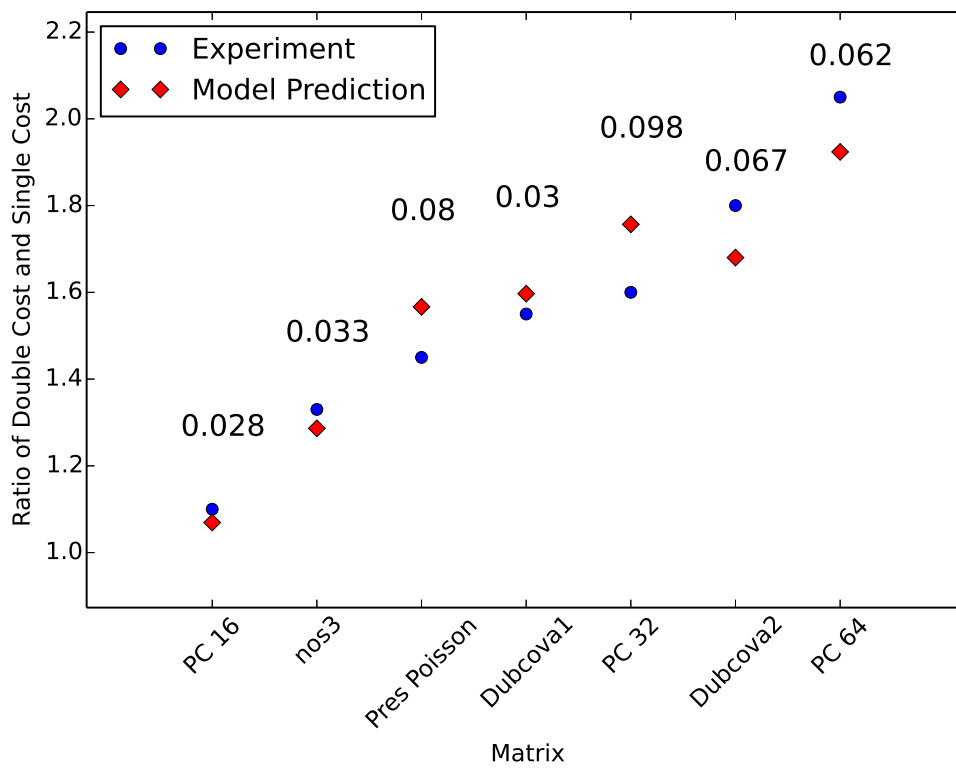


Figure 3.10: Double single model result. The numbers above the points are the relative error of predictions. The model almost fits the experiment results.

4 Conclusion



4.1 Summary of Methods and Results

In this paper we develop some stopping criteria with double precision version and mixed precision version. There are fixed, constant scaling, H.N., and Pocket. The constant scaling is set the inner tolerance as outer residual scaled by a constant. The H.N. stopping criteria is referred by [4], and we develop the mixed precision version. The Pocket method is keeping the best solution and keeping iterating until the residual does not decrease significantly in the steps we set. In the most case Pocket and H.N. have the best performance, and in some cases pocket method will get higher performance. Moreover the mixed precision version has higher performance than double precision version.

4.2 Advantages of the Pocket Methods

For Pocket methods, there are double version and mixed version. In most cases of our experiments, the mixed precision version may get higher performance than the double version, and the convergence is the same. Moreover, mixed precision version's pocket method has the best performance in most cases. So mixed pocket method is a good choice for choosing the stopping criteria in SIAR algorithm.

4.3 Limitations

In this paper, we only focus on the SPD matrixes and standard eigenvalue problem, and our mixed precision method only apply on double and single precisions. Besides, there are many parameters are not tuned in the best, so the parameter tuning may be an other issue in this project. In mixed precision algorithm, it needs more about a half memory than double precision, so if the problem strongly demands the memory resource, it may not use the mixed precision method.

4.4 Future Directions

Since the different hardware may need different compile optimization, one of the future work is using the different hardware to see the performance gain. We will port our code to PARALUTION project in the future[17], and we could compare the performance on different platforms, such like Xen Phi. Moreover we need to consider the ratio of double precision computation time and single precision computation time as a parameter to determine mixed precision approach and stopping criteria optimization. And the other future work is that we hope the mixed precision method may apply on not only double/single precision, but also any arbitrary precision. As we mentioned, parameter auto tuning may also be an important future work. We will extend our project to the non-symmetric matrices, and there are some work needed to be done for locking the searching space to the solved eigenspace. We also want to applied our method on general and polynomial types' eigenvalue problems.



5 Acknowledgement

I acknowledge my advisor, Professor Wang ^a, for advising my research and sharing his experience to me. His encouragement and kind suggestions help me to finish this project and research. Also, Professor Wang, provides many supports and resources including hardware and software. These help and experience are all precious for me. I also thank my oral committees, Professor Huang ^b and Professor Lee ^c. They gave me useful suggestions and comments. I thank Jia-Hung Chen ^d, for collaborating with the photonic crystal project, and I also thank Po-Chuang Wang ^e, Yan-An Chen ^f and Chun-Hua Huang ^g for help me do the testing work, and all the team members in my Lab help me in my master life. We discussed and produced many good ideas.



^aWeichung Wang, National Taiwan University, Taiwan, wwang@ntu.edu.tw

^bTsung-Min Hwang, National Taiwan Normal University, Taiwan, min@ntnu.edu.tw

^cChe-Rung Lee, National Tsing Hua University, Taiwan, cherung@cs.nthu.edu.tw

^dJia-Hung Chen, National Taiwan University, Taiwan, jrpg0618@gmail.com

^ePo-Chuang Wang, National Taiwan University, Taiwan, yukienanaha@gmail.com

^fYan-An Chen, National Taiwan University, Taiwan, keeneking@gmail.com

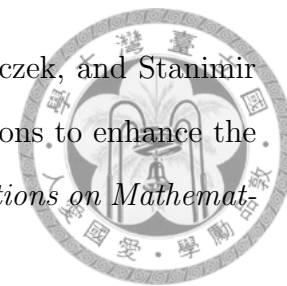
^gChun-Hua Huang, National Taiwan University, Taiwan, ditto.h@gmail.com

References



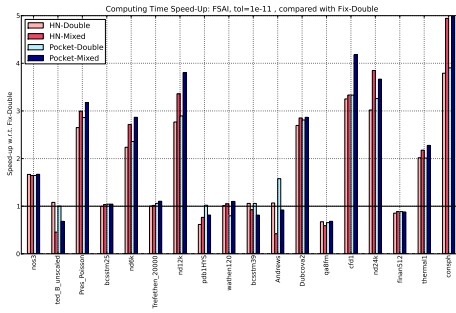
- [1] Zhaojun Bai, James Demmel, Jack Dongarra, Axel Ruhe, and Henk van der Vorst. *Templates for the solution of algebraic eigenvalue problems: a practical guide*, volume 11. Siam, 2000.
- [2] Zhongxiao Jia and Cen Li. On inner iterations in the shift-invert residual arnoldi method and the jacobi–davidson method. *arXiv preprint arXiv:1109.5455*, 2011.
- [3] Che-Rung Lee. Residual arnoldi method, theory, package and experiments. 2007.
- [4] Michiel E Hochstenbach and Yvan Notay. Controlling inner iterations in the jacobi-davidson method. *SIAM journal on matrix analysis and applications*, 31(2):460–477, 2009.
- [5] Timothy A Davis and Yifan Hu. The university of florida sparse matrix collection. *ACM Transactions on Mathematical Software (TOMS)*, 38(1):1, 2011.
- [6] Tsung-Ming Huang, Han-En Hsieh, Wen-Wei Lin, and Weichung Wang. Eigenvalue solvers for three dimensional photonic crystals with face-centered cubic lattice. *Journal of Computational and Applied Mathematics*, 2014.
- [7] Marc Baboulin, Alfredo Buttari, Jack Dongarra, Jakub Kurzak, Julie Langou, Julien Langou, Piotr Luszczek, and Stanimire Tomov. Accelerating scientific computations with mixed precision algorithms. *Computer Physics Communications*, 180(12):2526–2533, 2009.
- [8] Alfredo Buttari, Jack Dongarra, Julie Langou, Julien Langou, Piotr Luszczek, and Jakub Kurzak. Mixed precision iterative refinement techniques for the solution of dense linear systems. *International Journal of High Performance Computing Applications*, 21(4):457–466, 2007.

- [9] Alfredo Buttari, Jack Dongarra, Jakub Kurzak, Piotr Luszczek, and Stanimir Tomov. Using mixed precision for sparse matrix computations to enhance the performance while achieving 64-bit accuracy. *ACM Transactions on Mathematical Software (TOMS)*, 34(4):17, 2008.
- [10] Luc Giraud, Julien Langou, and Miroslav Rozložník. On the round-off error analysis of the gram-schmidt algorithm with reorthogonalization. Technical report, Technical Report TR/PA/02/33, CERFACS, Toulouse, France, 2002.
- [11] Richard Barrett, Michael W Berry, Tony F Chan, James Demmel, June Donato, Jack Dongarra, Victor Eijkhout, Roldan Pozo, Charles Romine, and Henk Van der Vorst. *Templates for the solution of linear systems: building blocks for iterative methods*, volume 43. Siam, 1994.
- [12] James W Demmel. *Applied numerical linear algebra*. Siam, 1997.
- [13] Yousef Saad. *Iterative methods for sparse linear systems*. Siam, 2003.
- [14] L Yu Kolotilina and A Yu Yeremin. Factorized sparse approximate inverse preconditionings i. theory. *SIAM Journal on Matrix Analysis and Applications*, 14(1):45–58, 1993.
- [15] Edward Anderson, Zhaojun Bai, Jack Dongarra, A Greenbaum, A McKenney, Jeremy Du Croz, S Hammerling, James Demmel, C Bischof, and Danny Sorensen. Lapack: A portable linear algebra library for high-performance computers. In *Proceedings of the 1990 ACM/IEEE conference on Supercomputing*, pages 2–11. IEEE Computer Society Press, 1990.
- [16] NVIDIA CUSPARSE. Cublas libraries.
- [17] Dimitar Lukarski. Paralution project, 2012.
- [18] CUDA Nvidia. Cufft library, 2010.

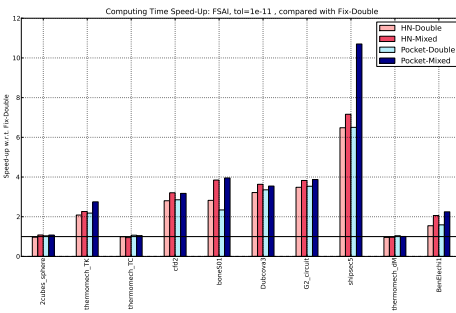


Appendix A Other graphs.

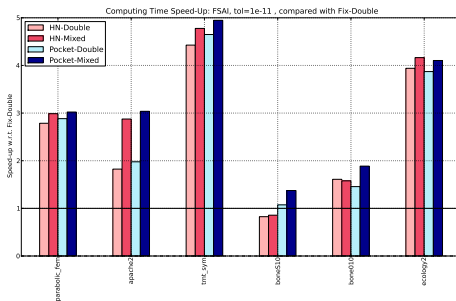




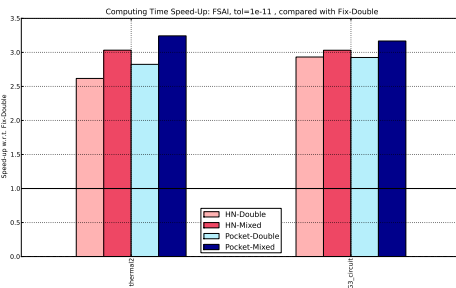
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

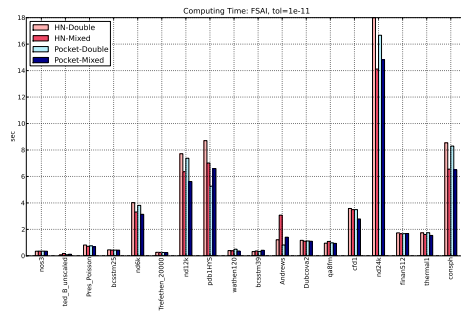


(c) Matrix of Dimension 500k to 1000k

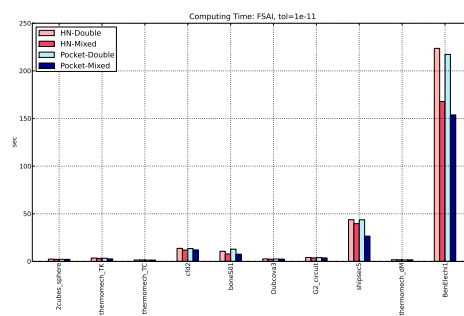


(d) Matrix of Dimension 1000k up

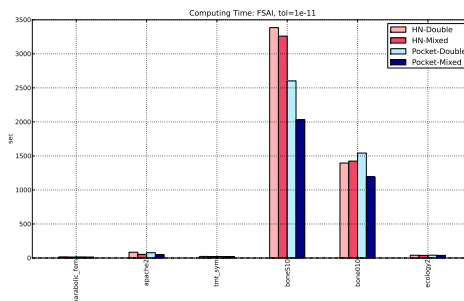
Figure A.1: Computing time of different stopping criteria



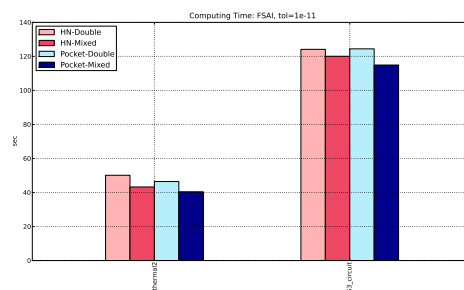
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

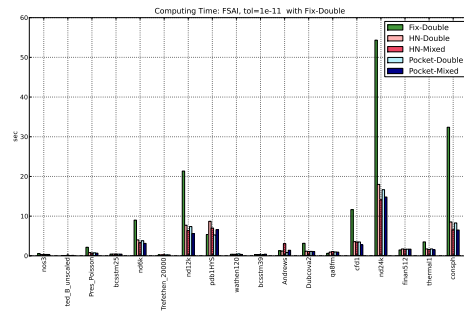


(c) Matrix of Dimension 500k to 1000k

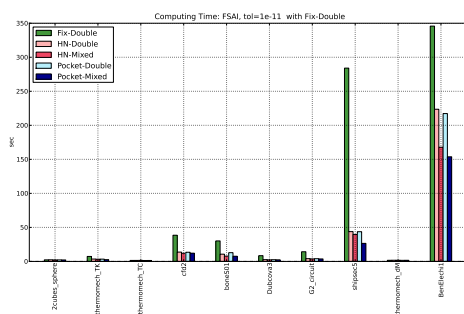


(d) Matrix of Dimension 1000k up

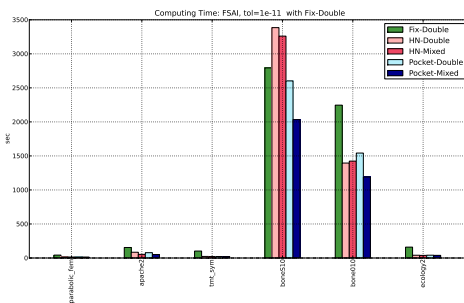
Figure A.2: Computing time of different stopping criteria



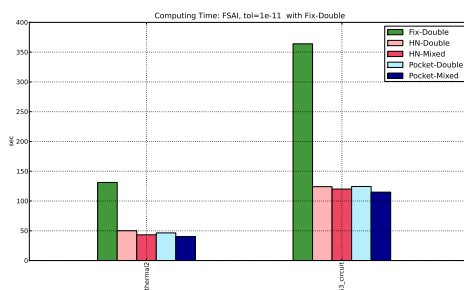
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

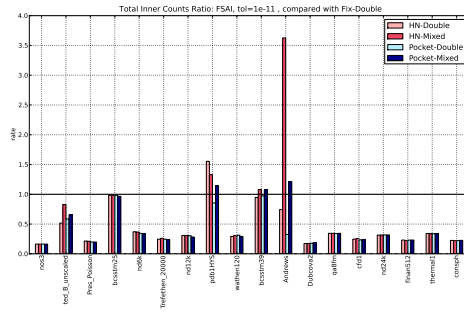


(c) Matrix of Dimension 500k to 1000k

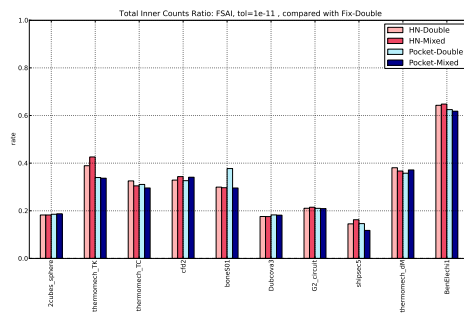


(d) Matrix of Dimension 1000k up

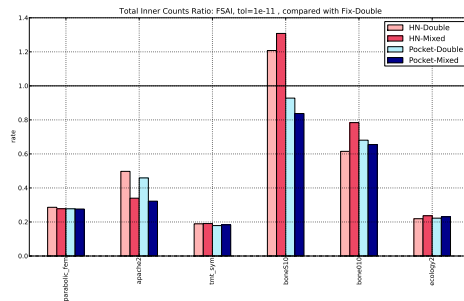
Figure A.3: Computing time of different stopping criteria



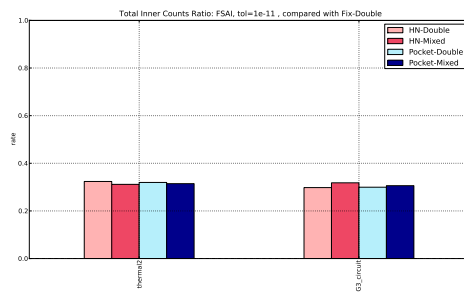
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

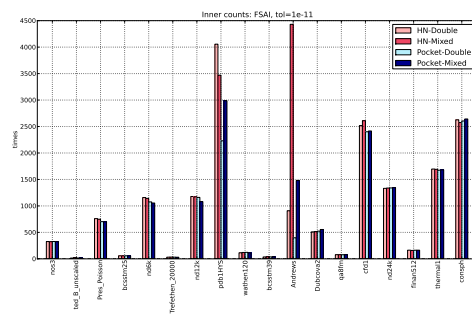


(c) Matrix of Dimension 500k to 1000k

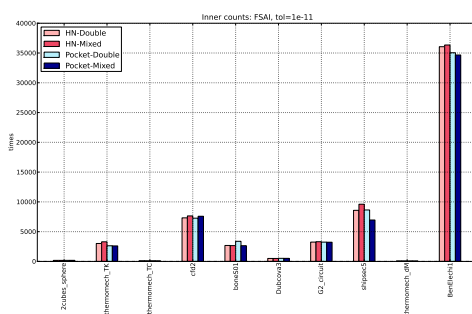


(d) Matrix of Dimension 1000k up

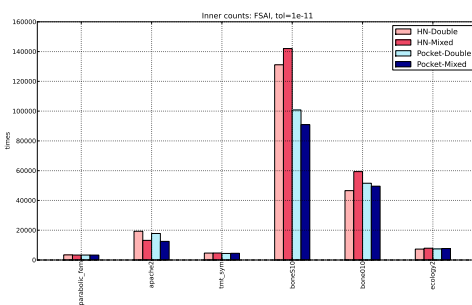
Figure A.4: Total inner counts of different stopping criteria



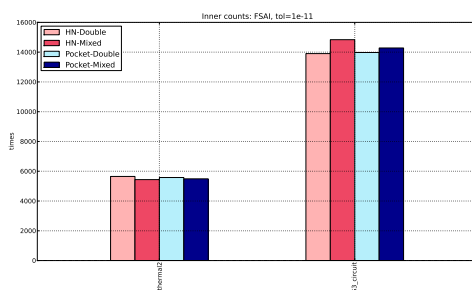
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

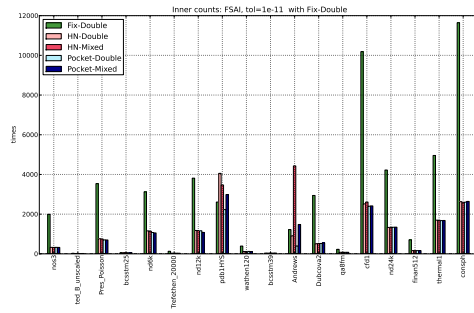


(c) Matrix of Dimension 500k to 1000k

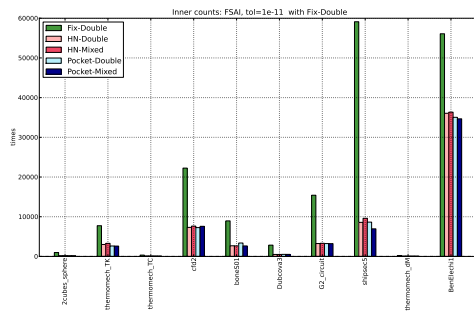


(d) Matrix of Dimension 1000k up

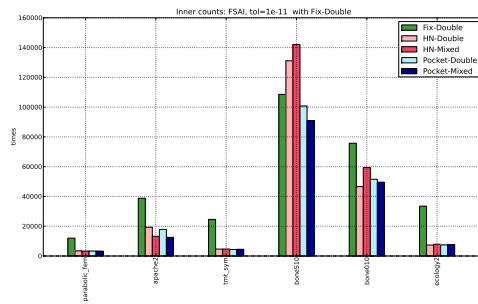
Figure A.5: Total inner counts of different stopping criteria



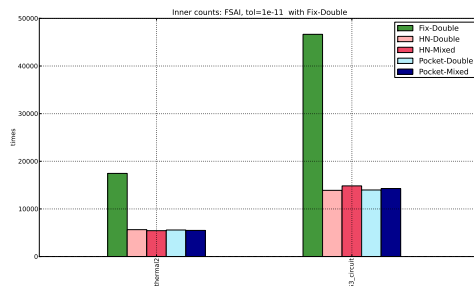
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

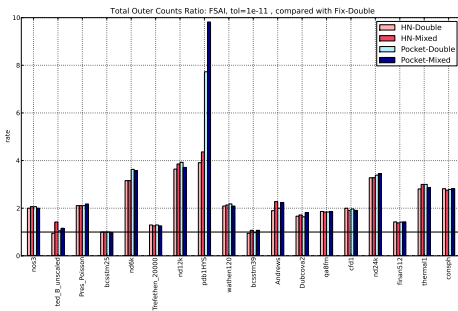


(c) Matrix of Dimension 500k to 1000k

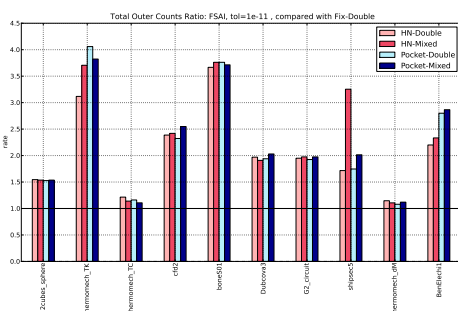


(d) Matrix of Dimension 1000k up

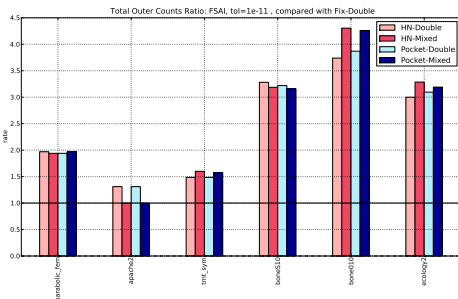
Figure A.6: Total inner counts of different stopping criteria



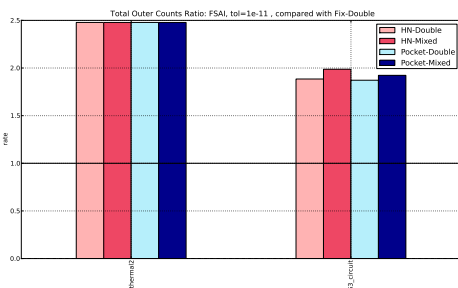
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

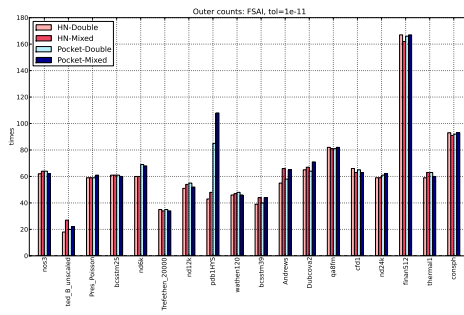


(c) Matrix of Dimension 500k to 1000k

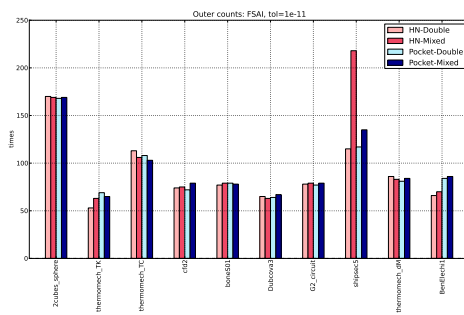


(d) Matrix of Dimension 1000k up

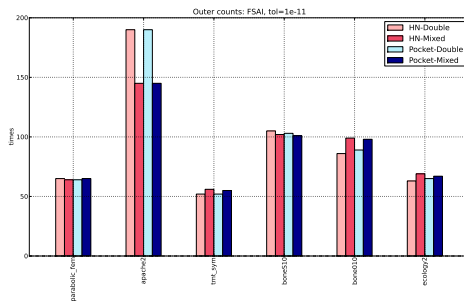
Figure A.7: Total outer counts of different stopping criteria



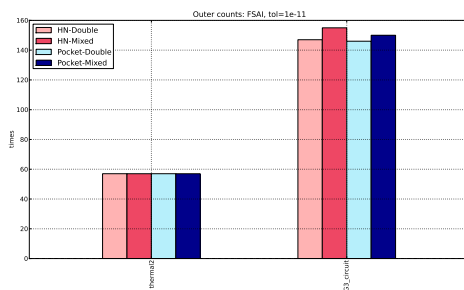
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

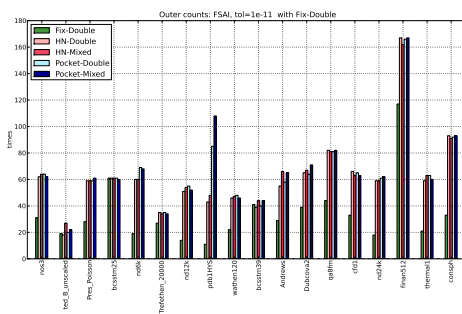


(c) Matrix of Dimension 500k to 1000k

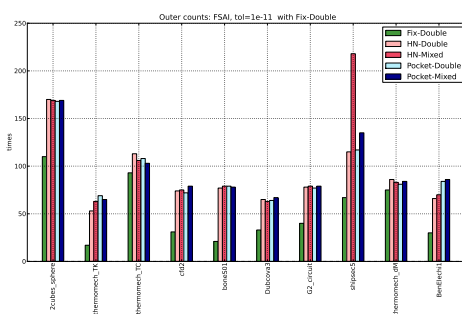


(d) Matrix of Dimension 1000k up

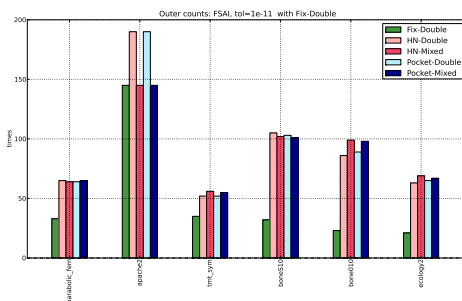
Figure A.8: Total outer counts of different stopping criteria



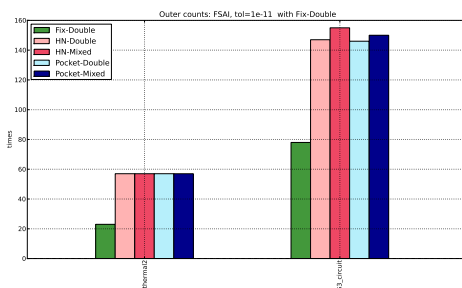
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

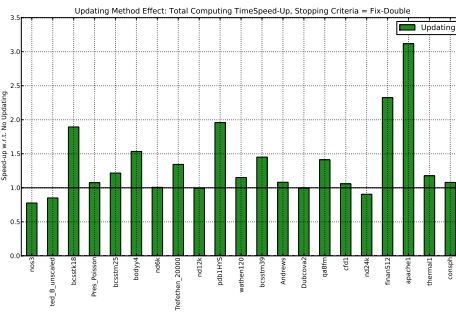


(c) Matrix of Dimension 500k to 1000k

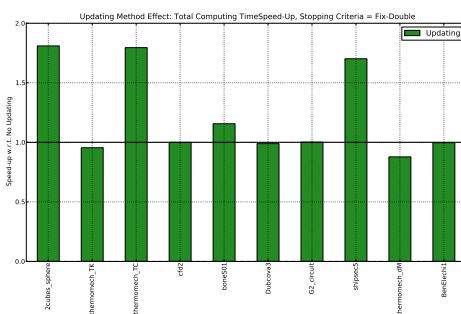


(d) Matrix of Dimension 1000k up

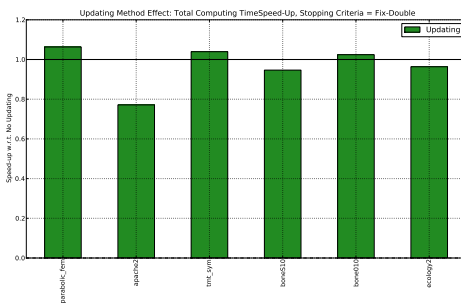
Figure A.9: Total outer counts of different stopping criteria



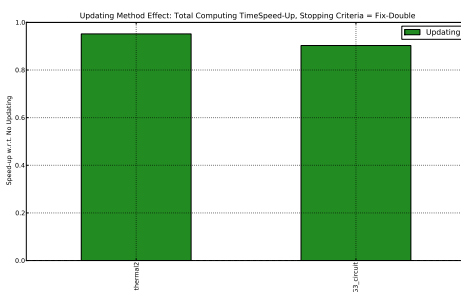
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

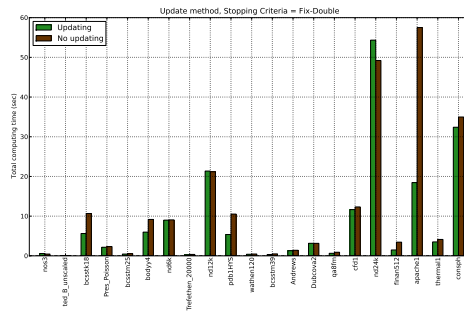


(c) Matrix of Dimension 500k to 1000k

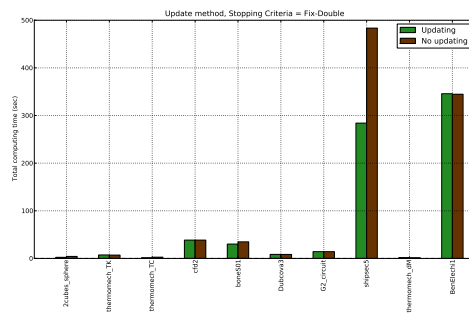


(d) Matrix of Dimension 1000k up

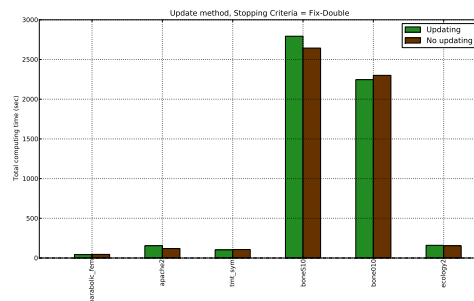
Figure A.10: Using/not using updating method with Fixed-Double



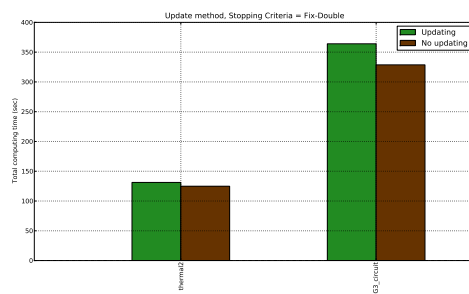
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

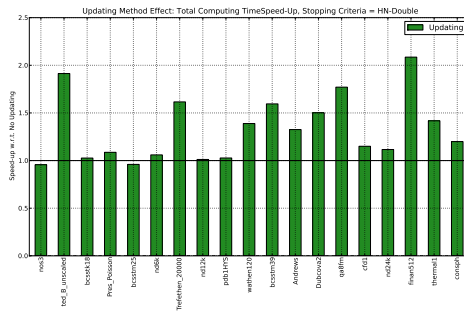


(c) Matrix of Dimension 500k to 1000k

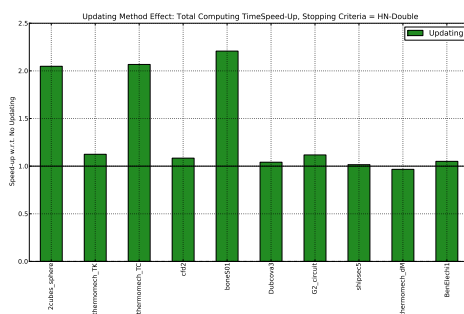


(d) Matrix of Dimension 1000k up

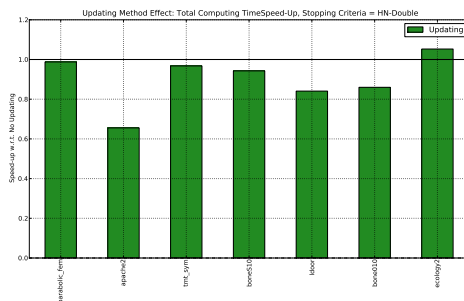
Figure A.11: Using/not using updating method with Fixed-Double



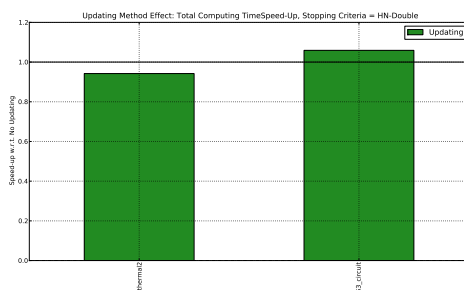
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

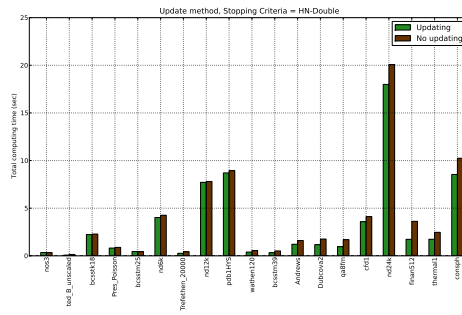


(c) Matrix of Dimension 500k to 1000k

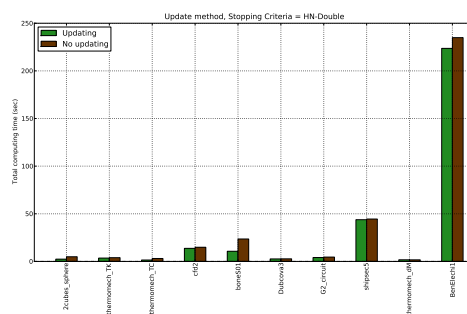


(d) Matrix of Dimension 1000k up

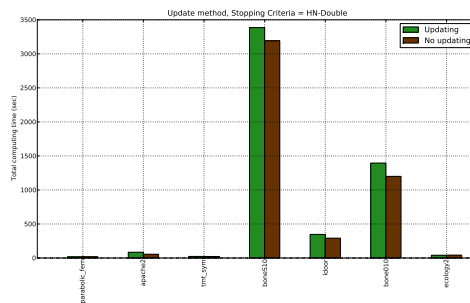
Figure A.12: Using/not using updating method with HN-Double



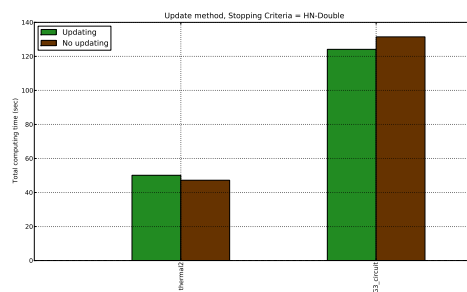
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

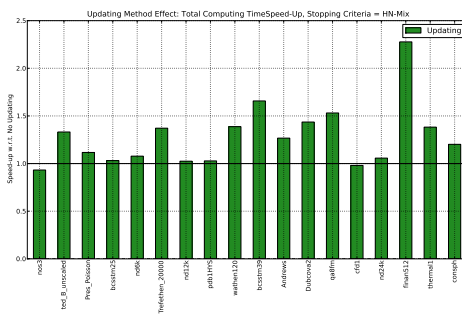


(c) Matrix of Dimension 500k to 1000k

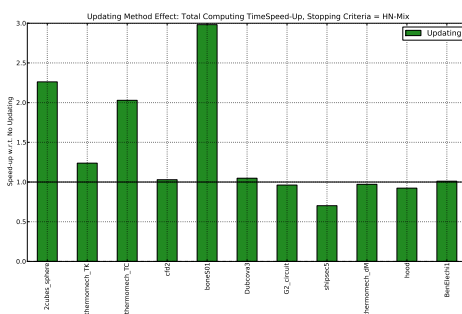


(d) Matrix of Dimension 1000k up

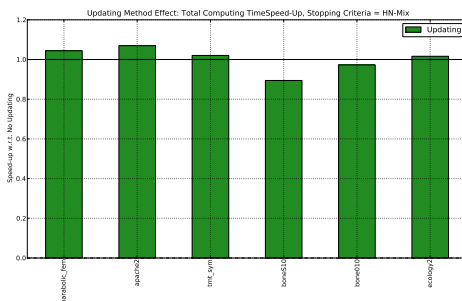
Figure A.13: Using/not using updating method with HN-Double



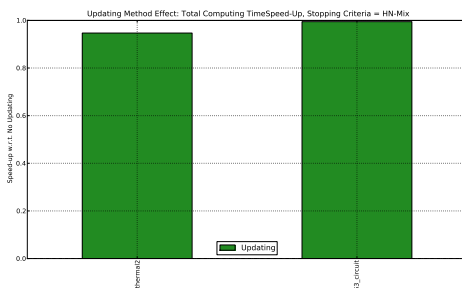
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

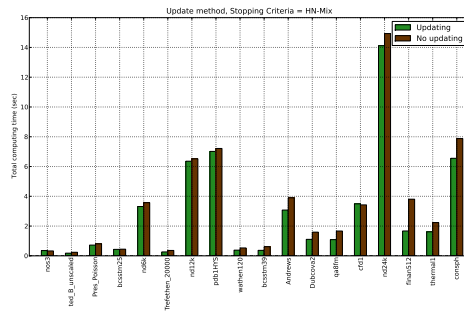


(c) Matrix of Dimension 500k to 1000k

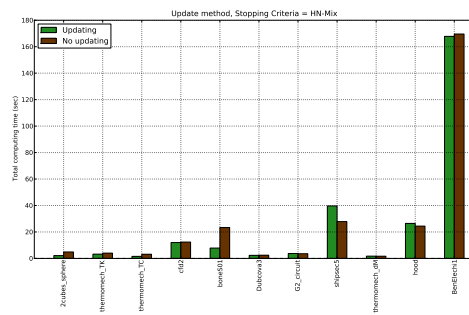


(d) Matrix of Dimension 1000k up

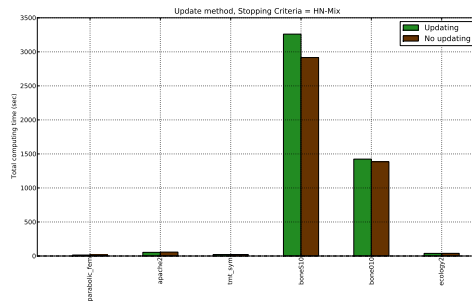
Figure A.14: Using/not using updating method with HN-Mixed



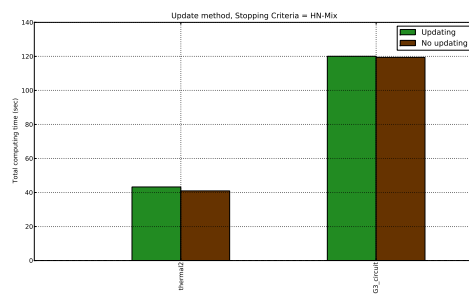
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

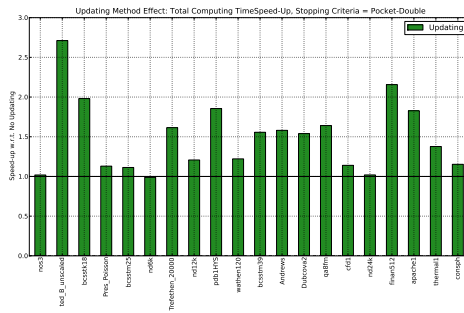


(c) Matrix of Dimension 500k to 1000k

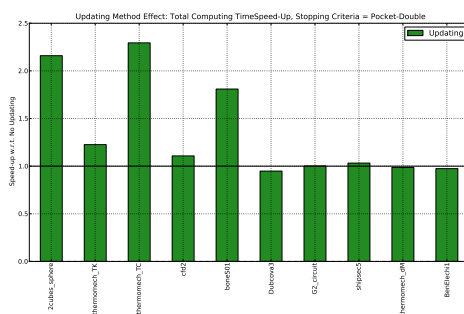


(d) Matrix of Dimension 1000k up

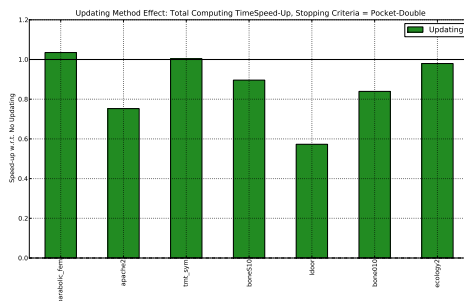
Figure A.15: Using/not using updating method with HN-Mixed



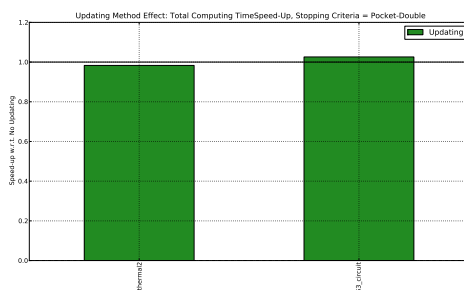
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

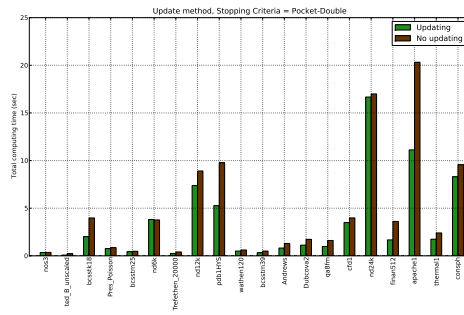


(c) Matrix of Dimension 500k to 1000k

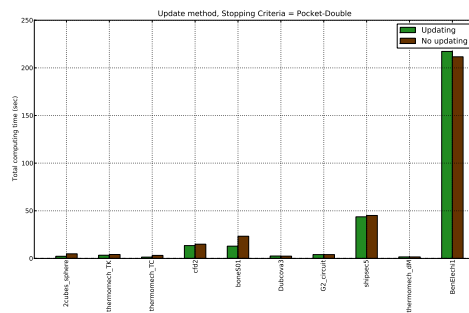


(d) Matrix of Dimension 1000k up

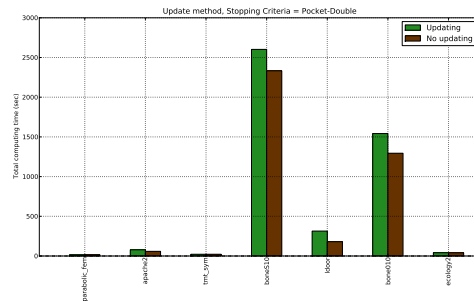
Figure A.16: Using/not using updating method with Pocket-Double



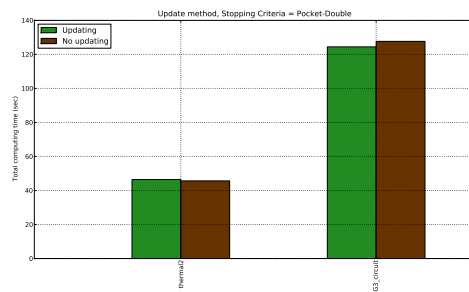
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

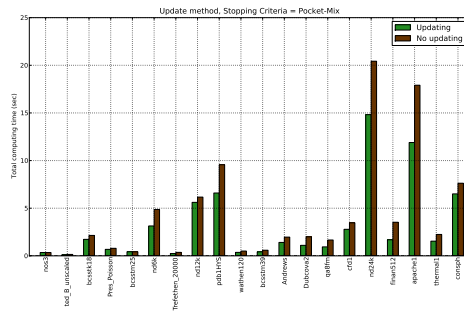


(c) Matrix of Dimension 500k to 1000k

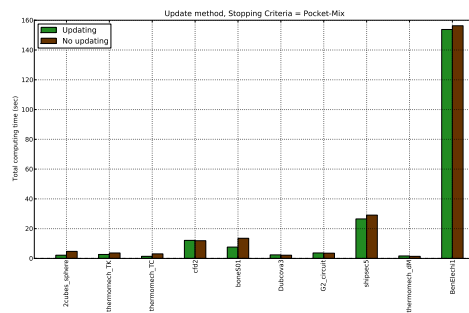


(d) Matrix of Dimension 1000k up

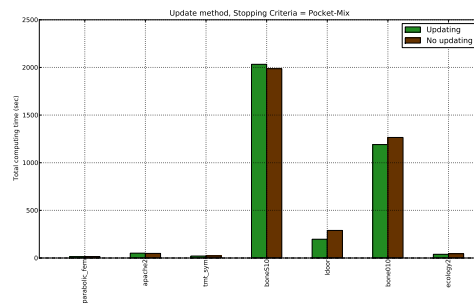
Figure A.17: Using/not using updating method with Pocket-Double



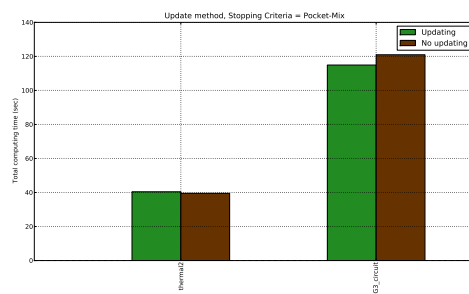
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

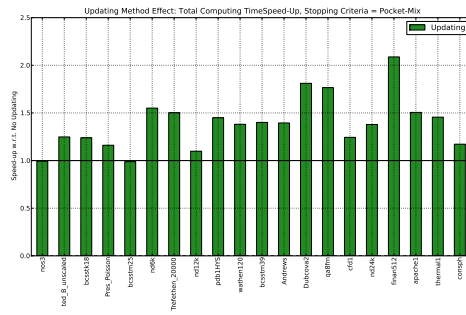


(c) Matrix of Dimension 500k to 1000k

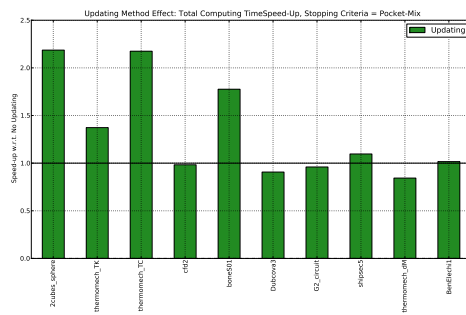


(d) Matrix of Dimension 1000k up

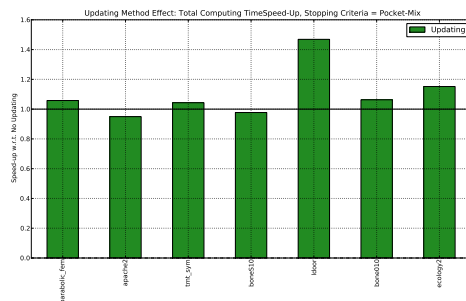
Figure A.18: Using/not using updating method with Pocket-Mixed



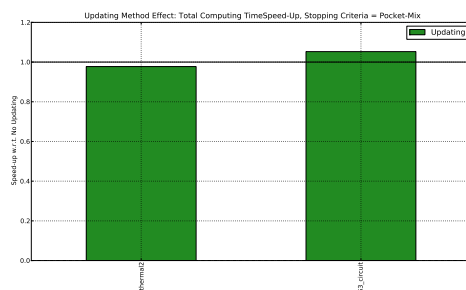
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

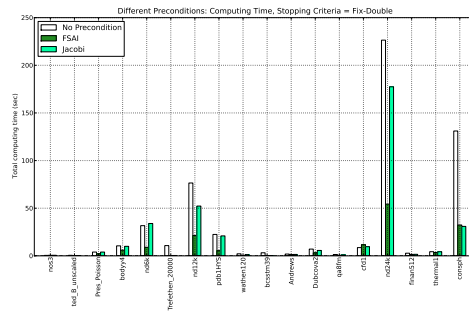


(c) Matrix of Dimension 500k to 1000k

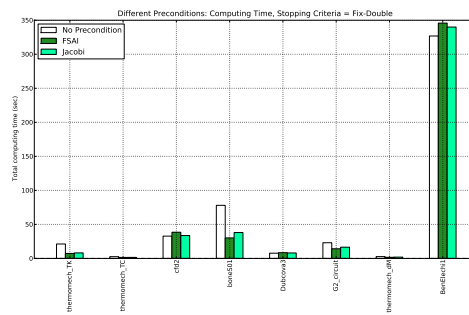


(d) Matrix of Dimension 1000k up

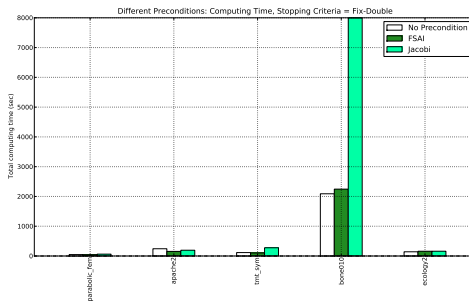
Figure A.19: Using/not using updating method with Pocket-Mixed



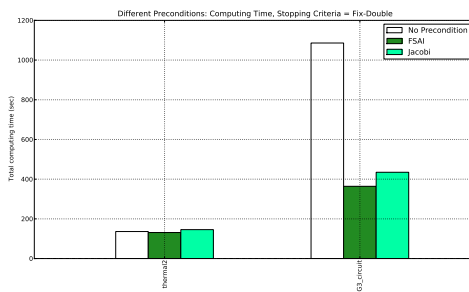
(a) Matrix of Dimension 10k to 100k

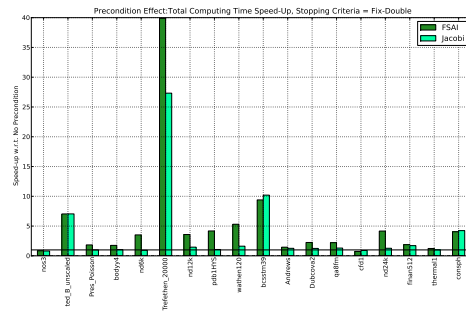


(b) Matrix of Dimension 100k to 500k

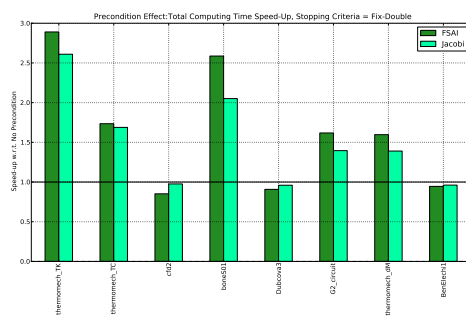


(c) Matrix of Dimension 500k to 1000k

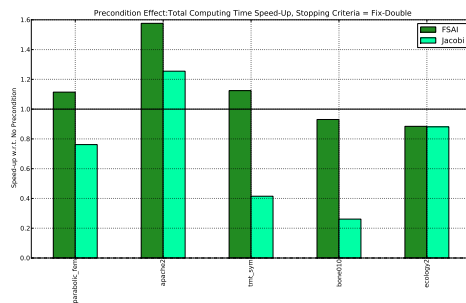




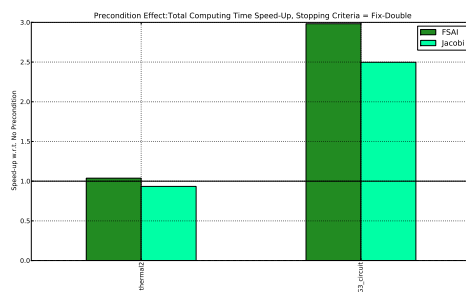
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

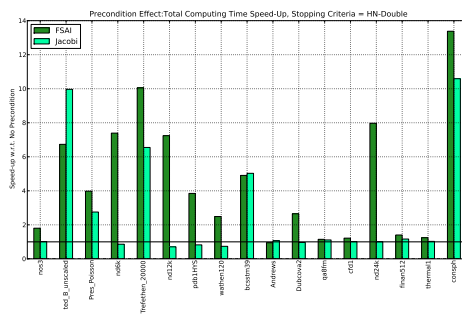


(c) Matrix of Dimension 500k to 1000k

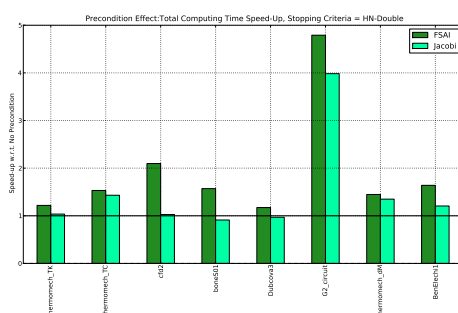


(d) Matrix of Dimension 1000k up

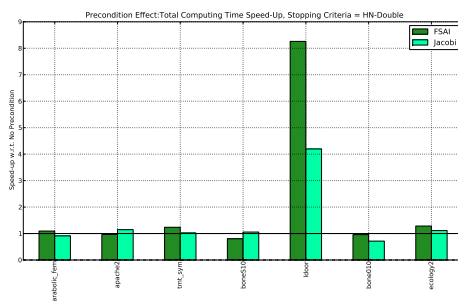
Figure A.21: Different precondition with Fixed-Double



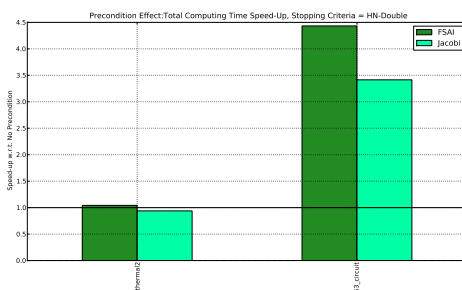
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

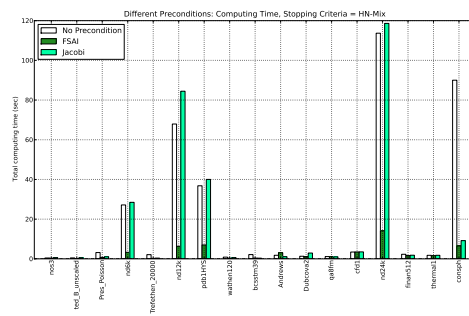


(c) Matrix of Dimension 500k to 1000k

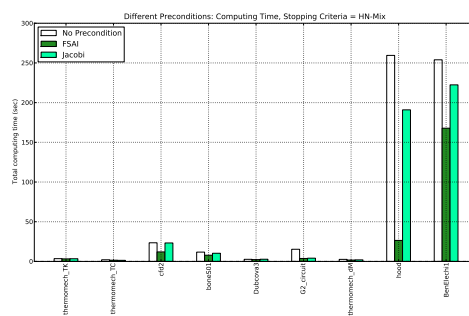


(d) Matrix of Dimension 1000k up

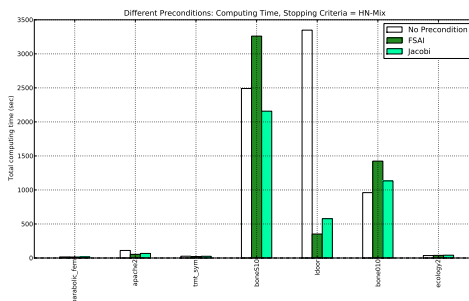
Figure A.23: Different precondition with HN-Double



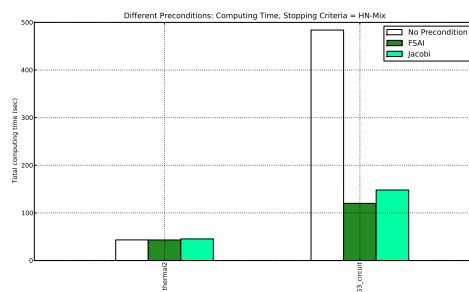
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

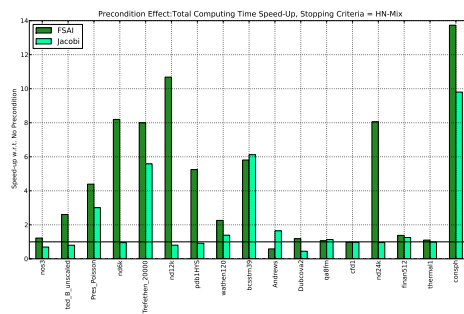


(c) Matrix of Dimension 500k to 1000k

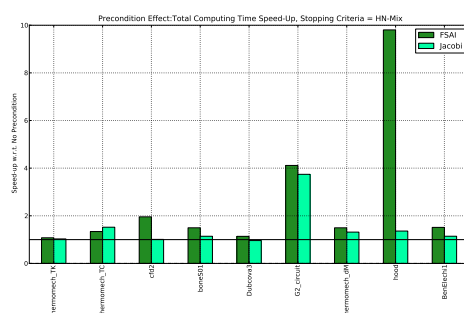


(d) Matrix of Dimension 1000k up

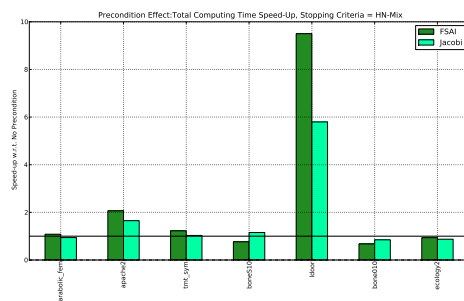
Figure A.24: Different precondition with HN-Mixed



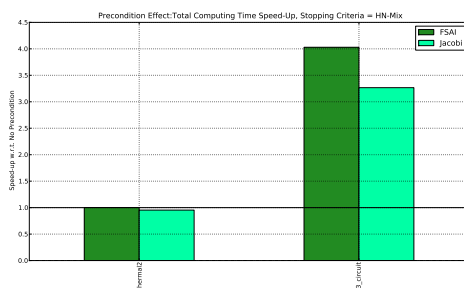
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

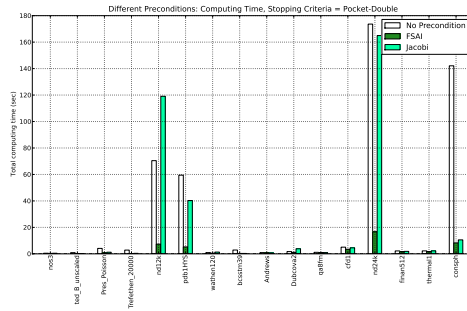


(c) Matrix of Dimension 500k to 1000k

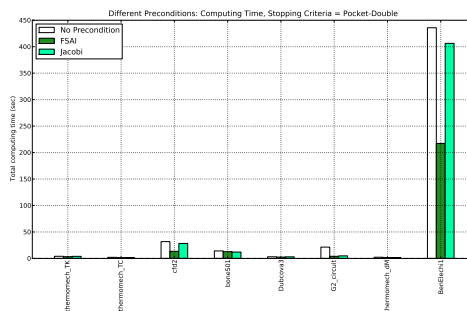


(d) Matrix of Dimension 1000k up

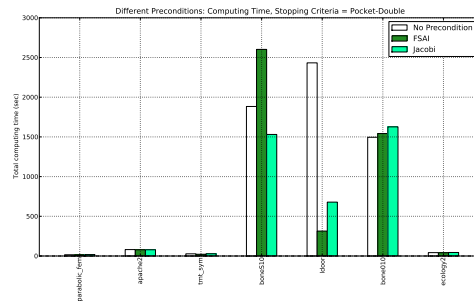
Figure A.25: Different precondition with HN-Mixed



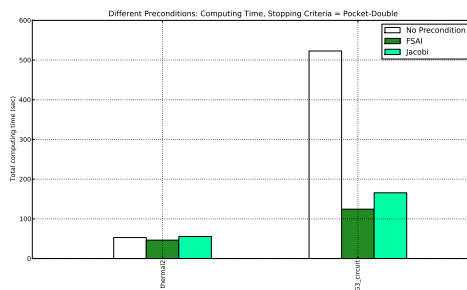
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

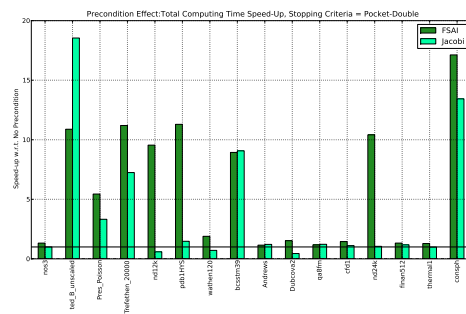


(c) Matrix of Dimension 500k to 1000k

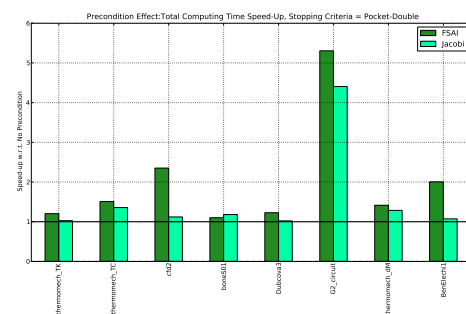


(d) Matrix of Dimension 1000k up

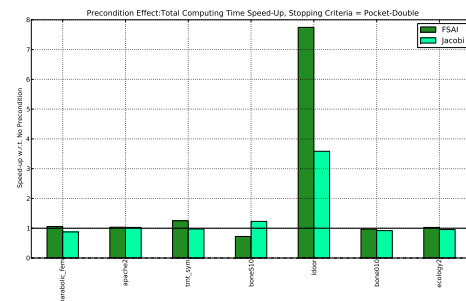
Figure A.26: Different precondition with Pocket-Double



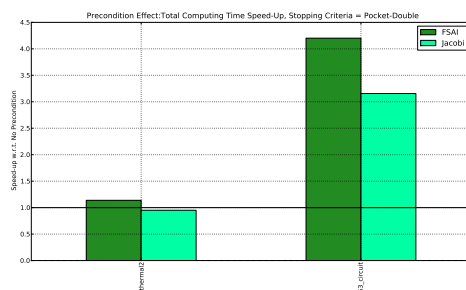
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

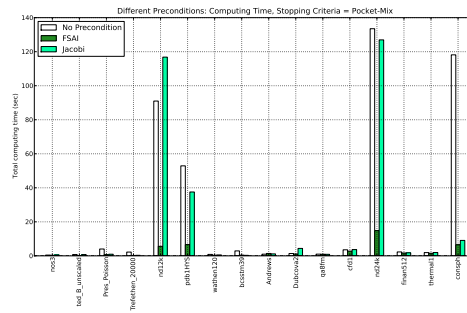


(c) Matrix of Dimension 500k to 1000k

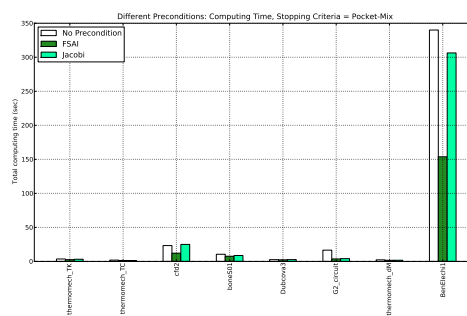


(d) Matrix of Dimension 1000k up

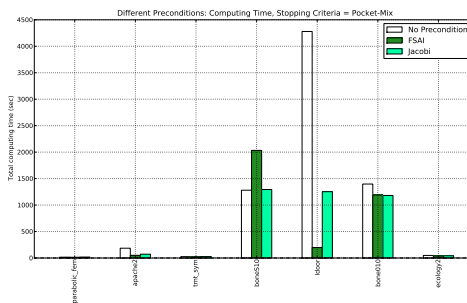
Figure A.27: Different precondition with Pocket-Double



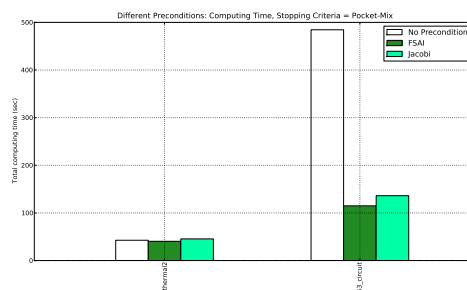
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

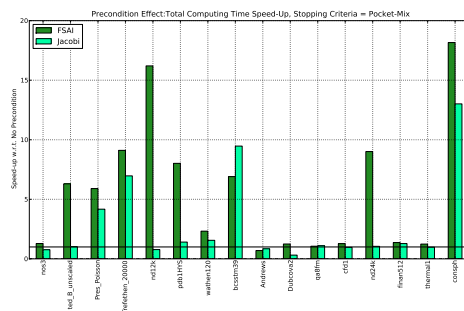


(c) Matrix of Dimension 500k to 1000k

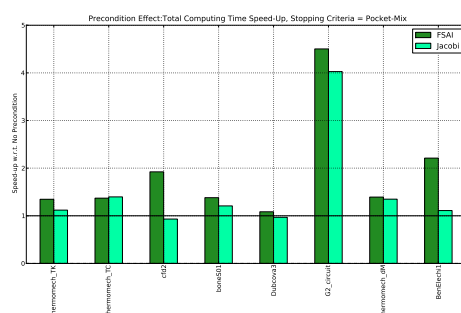


(d) Matrix of Dimension 1000k up

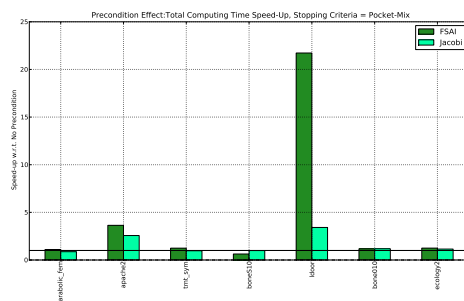
Figure A.28: Different precondition with Pocket-Mixed



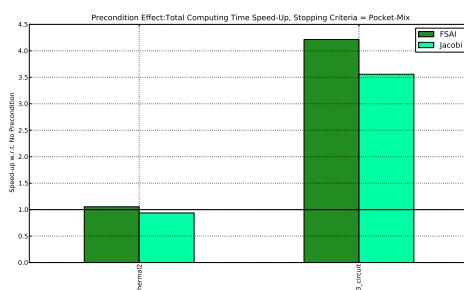
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

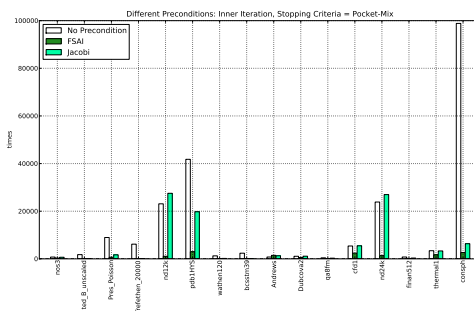


(c) Matrix of Dimension 500k to 1000k

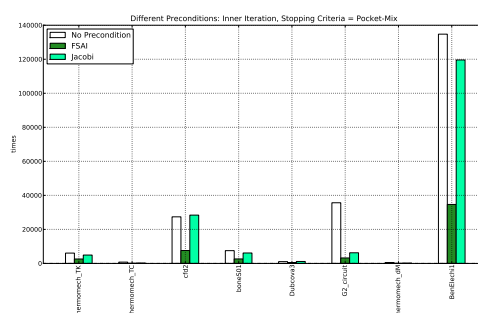


(d) Matrix of Dimension 1000k up

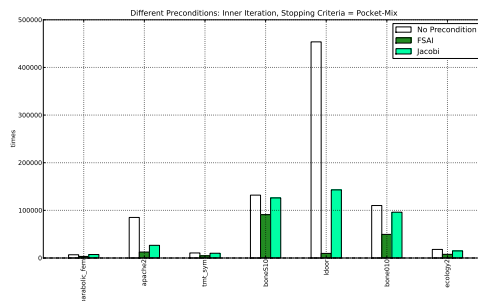
Figure A.29: Different precondition with Pocket-Mixed



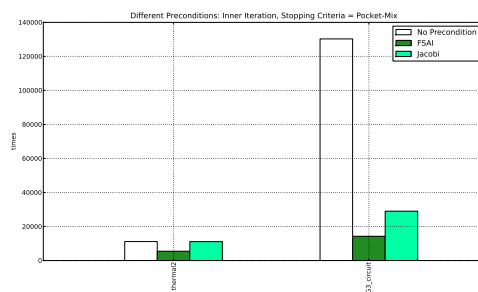
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

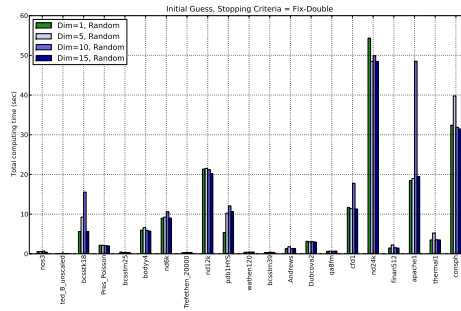


(c) Matrix of Dimension 500k to 1000k

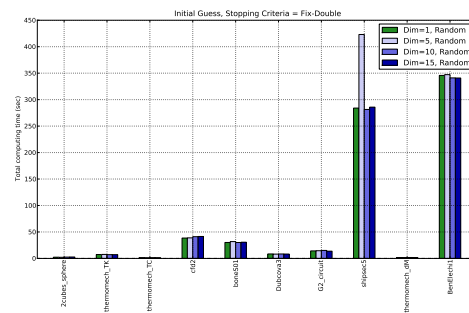


(d) Matrix of Dimension 1000k up

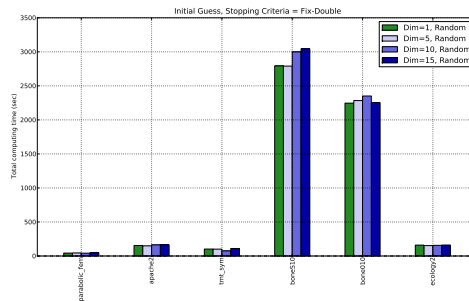
Figure A.30: Inner iteration times in different preconditioner with Pocket-Mixed



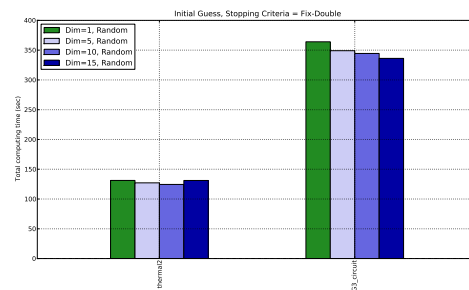
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

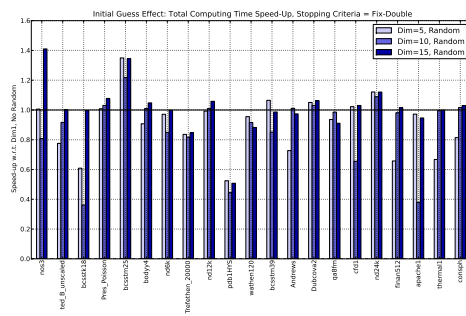


(c) Matrix of Dimension 500k to 1000k

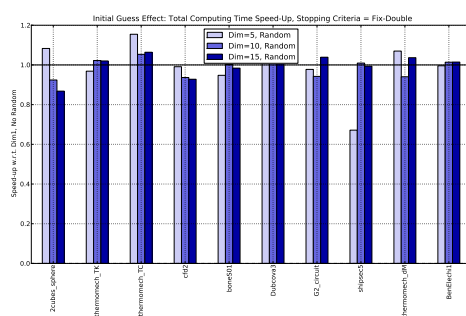


(d) Matrix of Dimension 1000k up

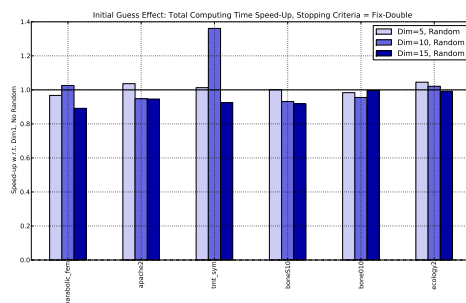
Figure A.31: Different initial with Fixed-Double



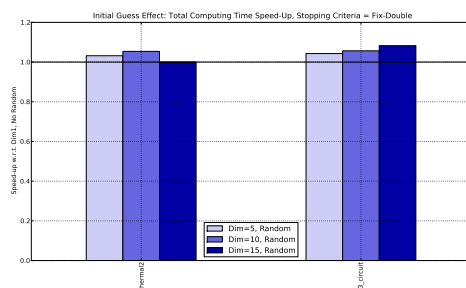
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

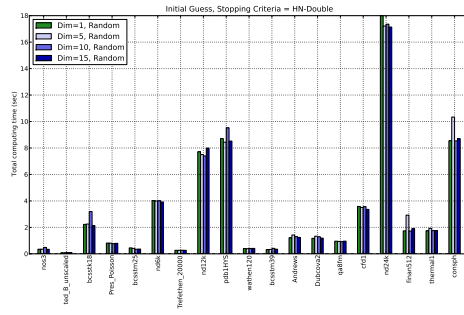


(c) Matrix of Dimension 500k to 1000k

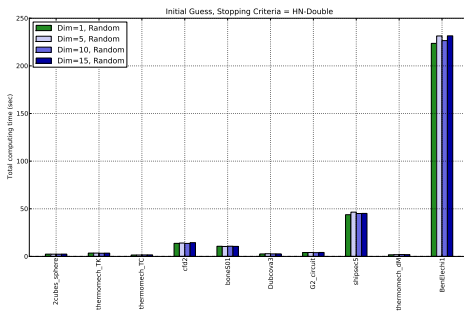


(d) Matrix of Dimension 1000k up

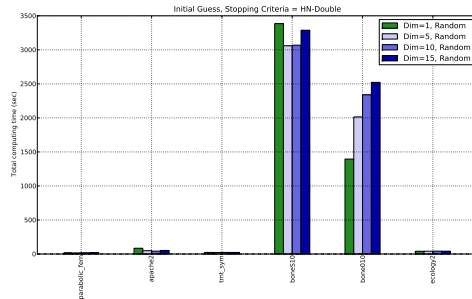
Figure A.32: Different initial with Fixed-Double



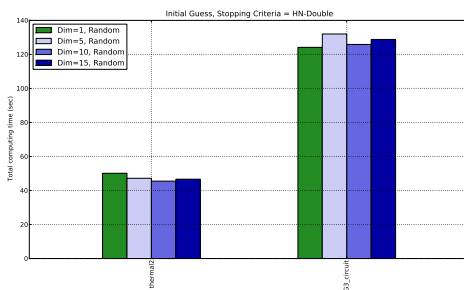
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

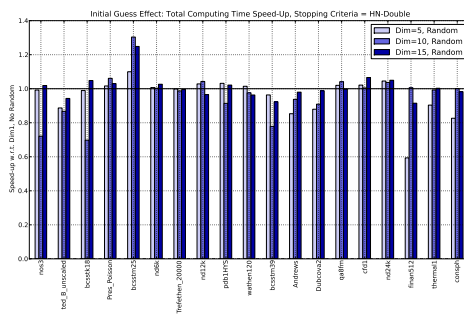


(c) Matrix of Dimension 500k to 1000k

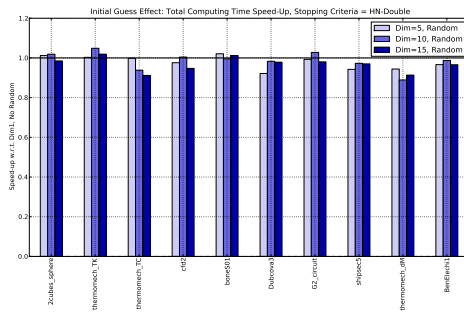


(d) Matrix of Dimension 1000k up

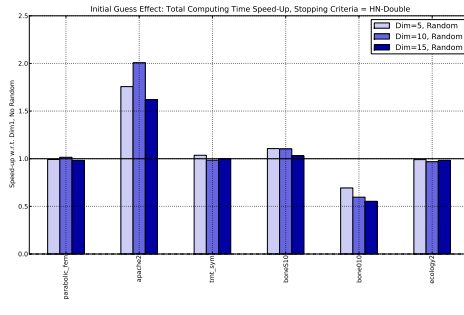
Figure A.33: Different initial with HN-Double



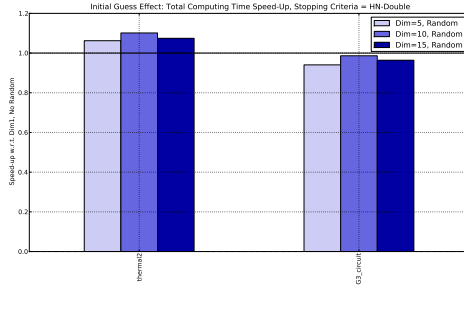
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

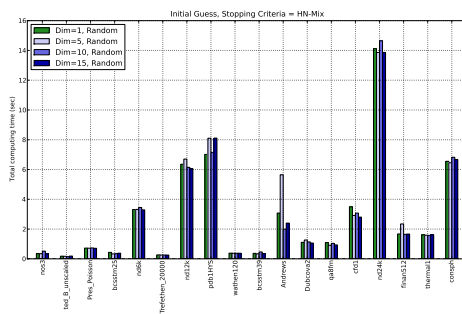


(c) Matrix of Dimension 500k to 1000k

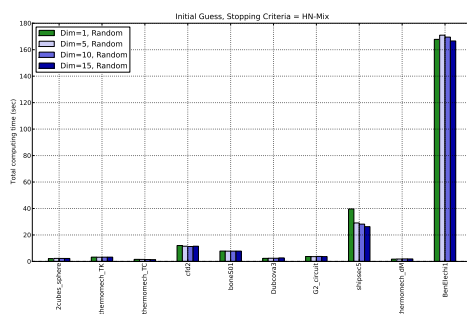


(d) Matrix of Dimension 1000k up

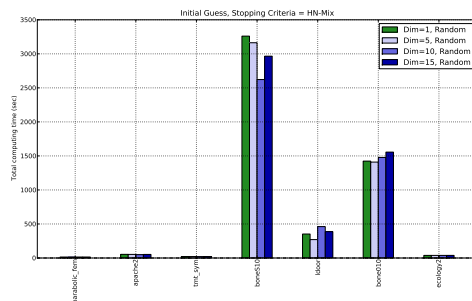
Figure A.34: Different initial with HN-Double



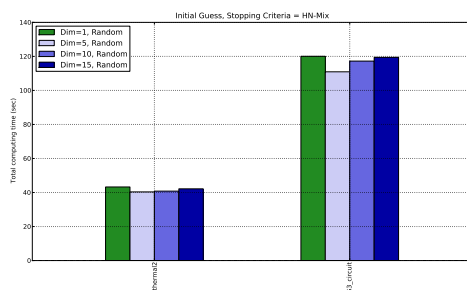
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

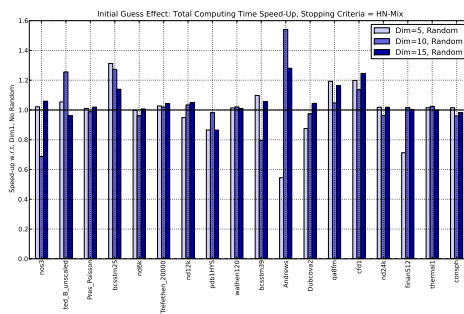


(c) Matrix of Dimension 500k to 1000k

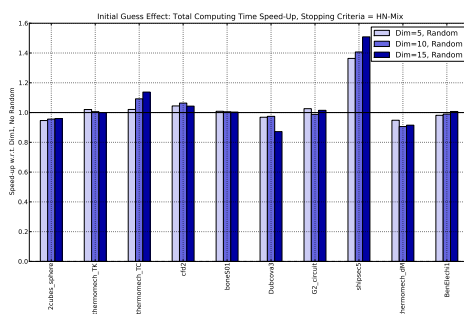


(d) Matrix of Dimension 1000k up

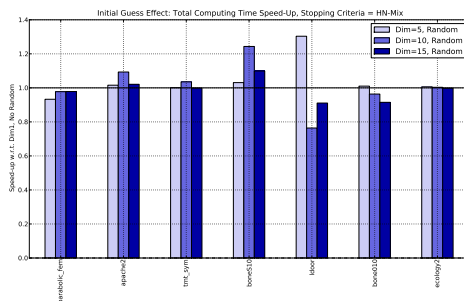
Figure A.35: Different initial with HN-Mixed



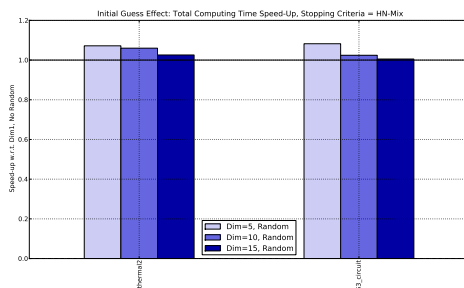
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

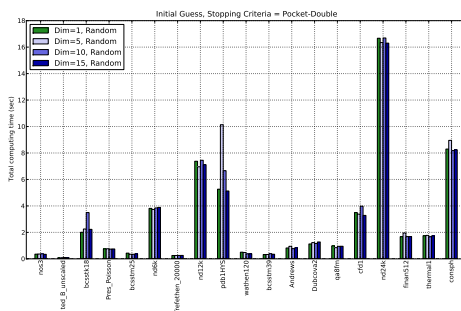


(c) Matrix of Dimension 500k to 1000k

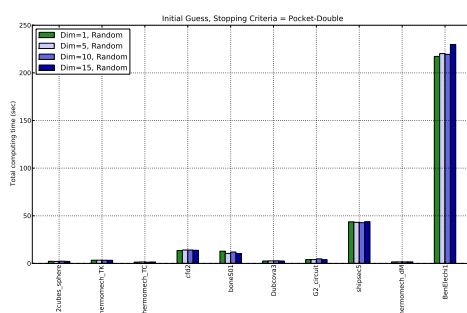


(d) Matrix of Dimension 1000k up

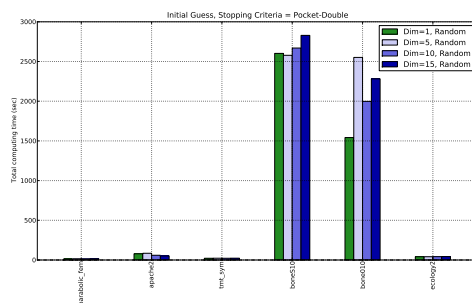
Figure A.36: Different initial with HN-Mixed



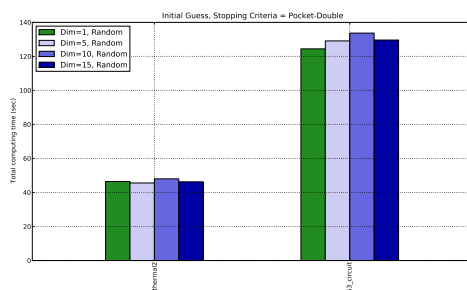
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

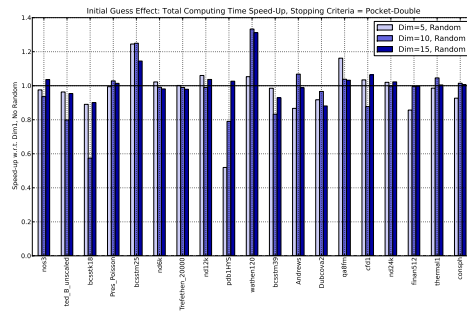


(c) Matrix of Dimension 500k to 1000k

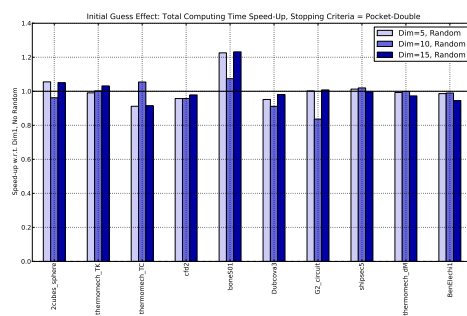


(d) Matrix of Dimension 1000k up

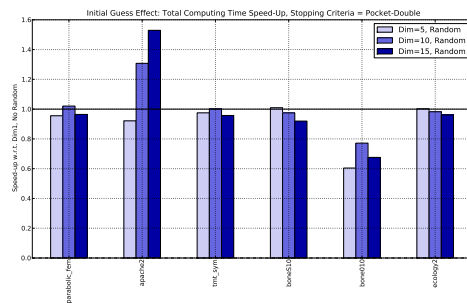
Figure A.37: Different initial with Pocket-Double



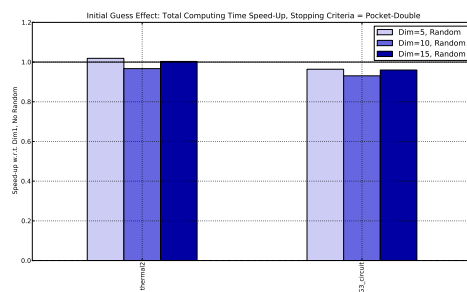
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

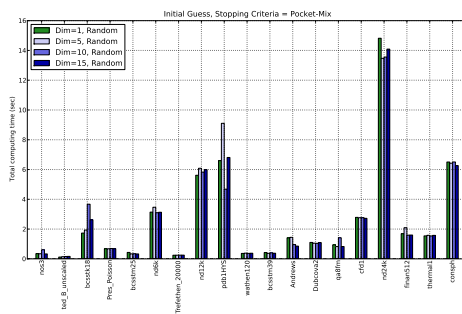


(c) Matrix of Dimension 500k to 1000k

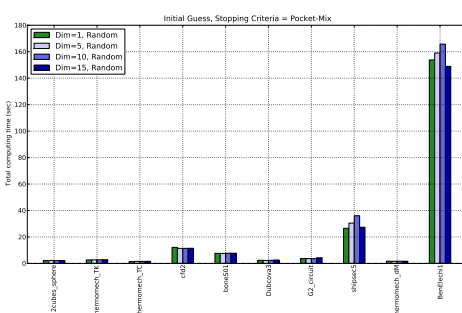


(d) Matrix of Dimension 1000k up

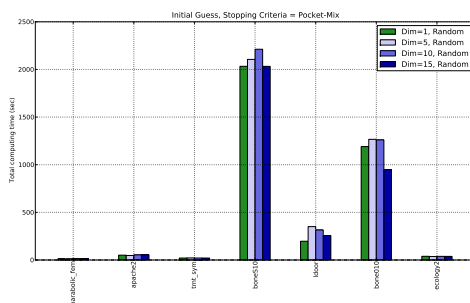
Figure A.38: Different initial with Pocket-Double



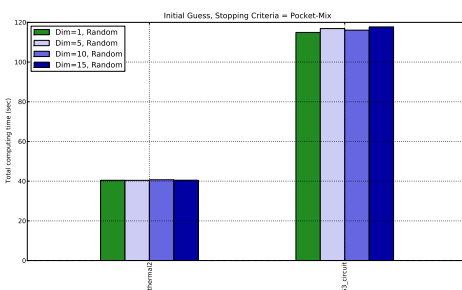
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

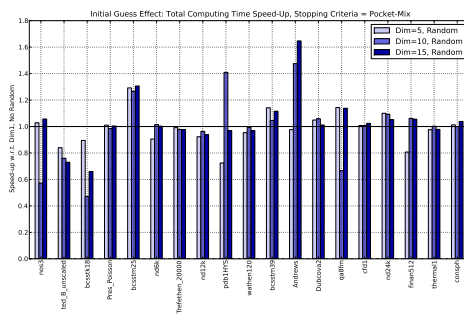


(c) Matrix of Dimension 500k to 1000k

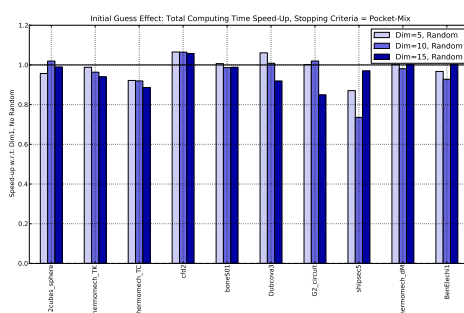


(d) Matrix of Dimension 1000k up

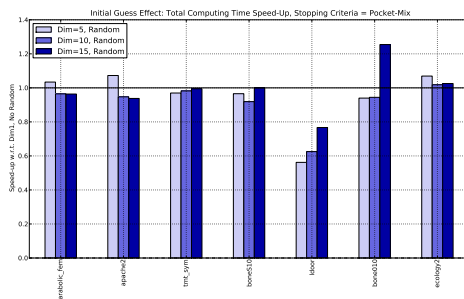
Figure A.39: Different initial with Pocket-Mixed



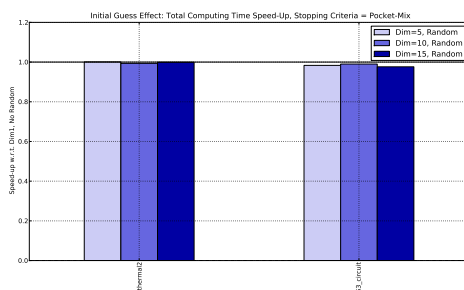
(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

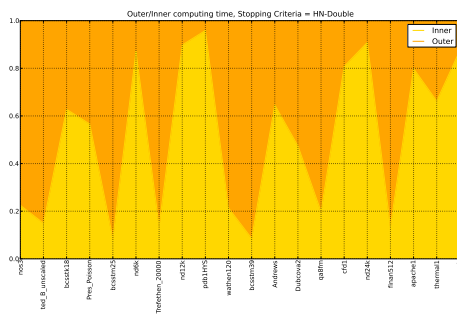


(c) Matrix of Dimension 500k to 1000k

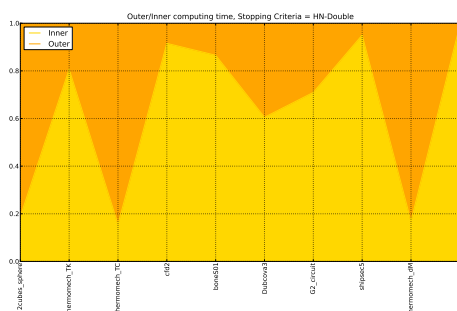


(d) Matrix of Dimension 1000k up

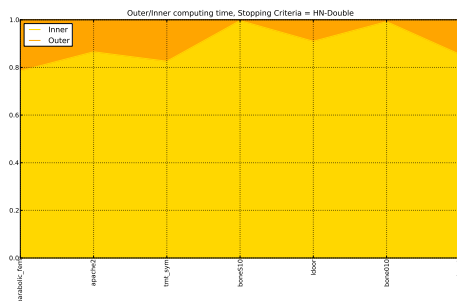
Figure A.40: Different initial with Pocket-Mixed



(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

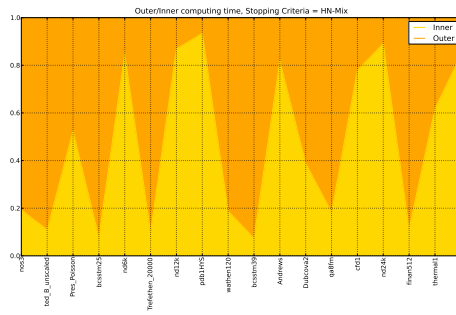


(c) Matrix of Dimension 500k to 1000k

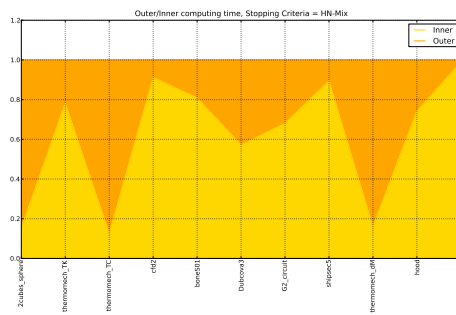


(d) Matrix of Dimension 1000k up

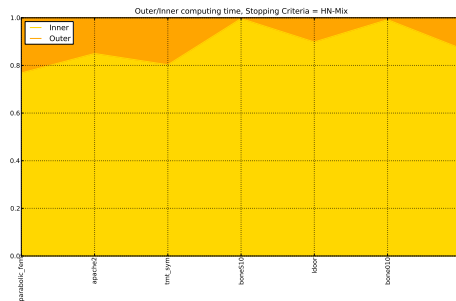
Figure A.41: Outer/Inner time ratio of different stopping criteria



(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

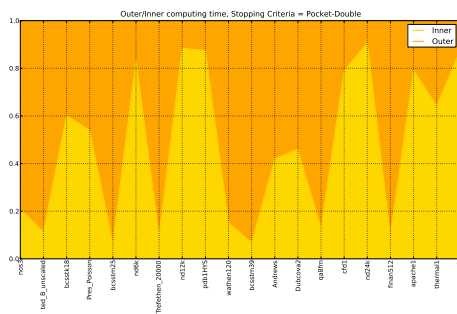


(c) Matrix of Dimension 500k to 1000k

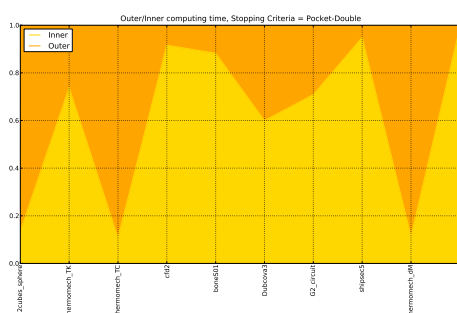


(d) Matrix of Dimension 1000k up

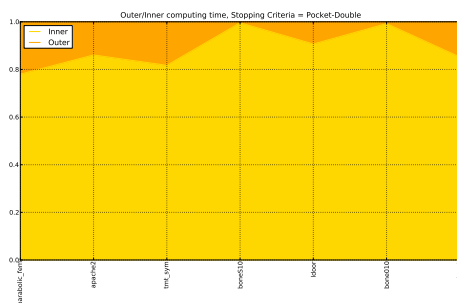
Figure A.42: Outer/Inner time ratio of different stopping criteria



(a) Matrix of Dimension 10k to 100k



(b) Matrix of Dimension 100k to 500k

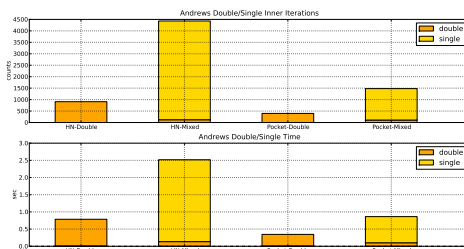


(c) Matrix of Dimension 500k to 1000k

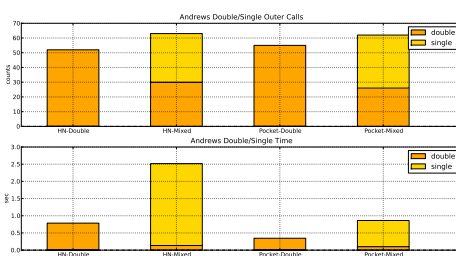


(d) Matrix of Dimension 1000k up

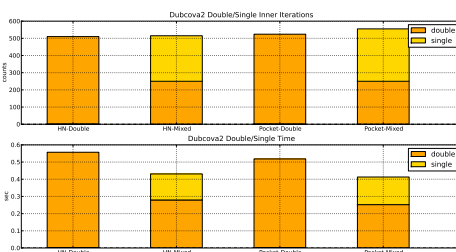
Figure A.43: Outer/Inner time ratio of different stopping criteria



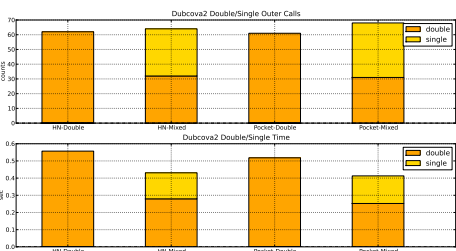
(a) Matrix: Andrews's inner information



(b) Matrix: Andrews's outer information

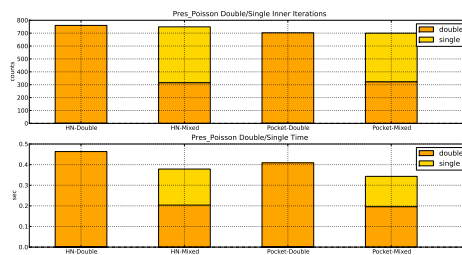


(c) Matrix: Dubcova2's inner information

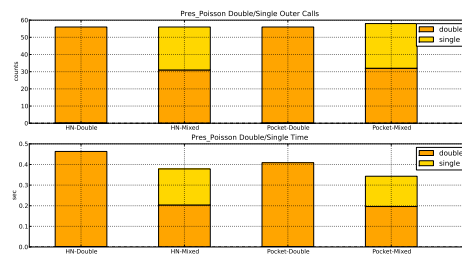


(d) Matrix: Dubcova2's outer information

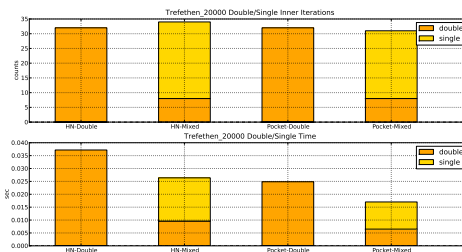
Figure A.44: Matrix inner and outer informations



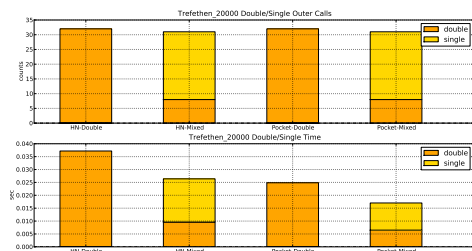
(a) Matrix: Pres_Poisson's inner information



(b) Matrix: Pres_Poisson's outer information

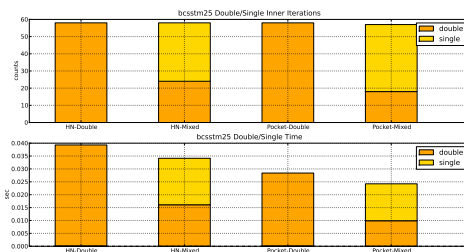


(c) Matrix: Trefethen_20000's inner information

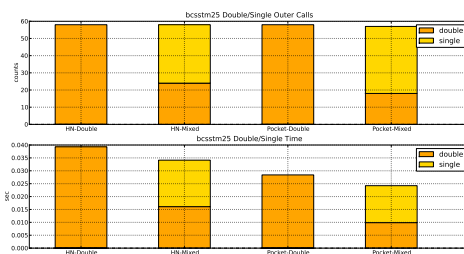


(d) Matrix: Trefethen_20000's outer information

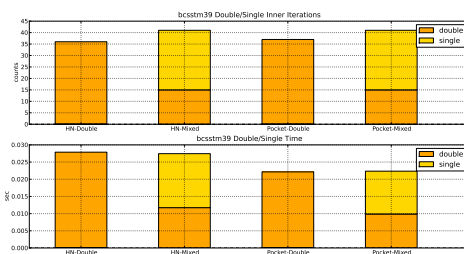
Figure A.45: Matrix inner and outer informations



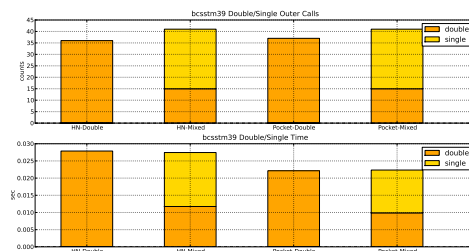
(a) Matrix: bcsstm25's inner information



(b) Matrix: bcsstm25's outer information

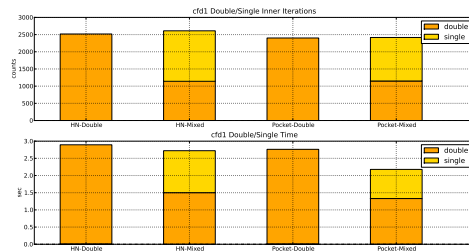


(c) Matrix: bcsstm39's inner information

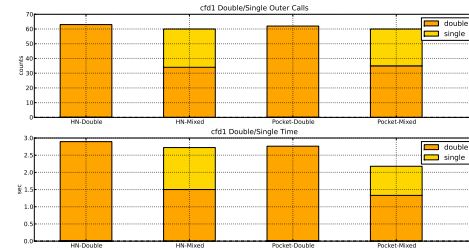


(d) Matrix: bcsstm39's outer information

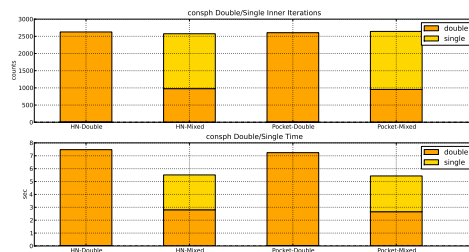
Figure A.46: Matrix inner and outer informations



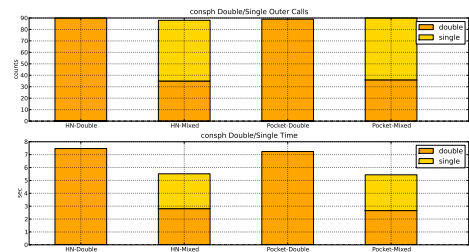
(a) Matrix: cfd1's inner information



(b) Matrix: cfd1's outer information

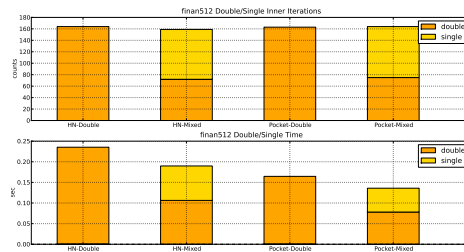


(c) Matrix: consph's inner information

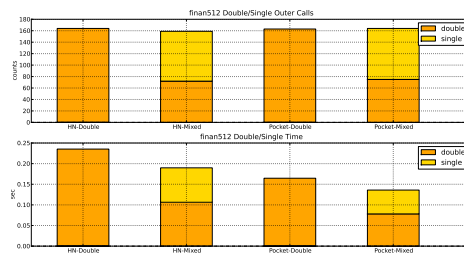


(d) Matrix: consph's outer information

Figure A.47: Matrix inner and outer informations



(a) Matrix: finan512's inner information



(b) Matrix: finan512's outer information

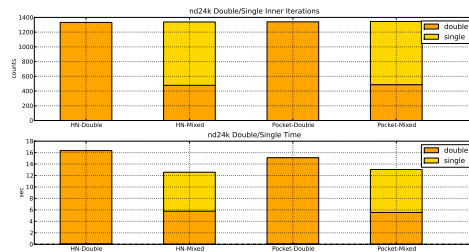


(c) Matrix: nd12k's inner information

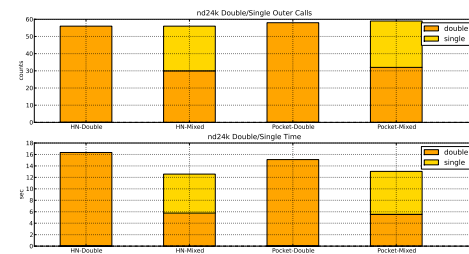


(d) Matrix: nd12k's outer information

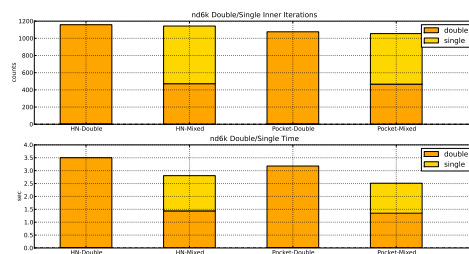
Figure A.48: Matrix inner and outer informations



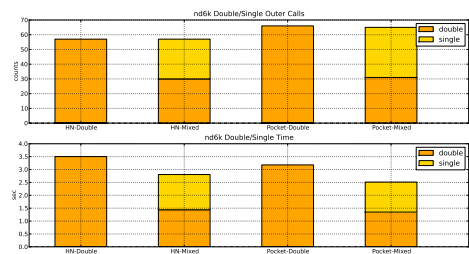
(a) Matrix: nd24k's inner information



(b) Matrix: nd24k's outer information

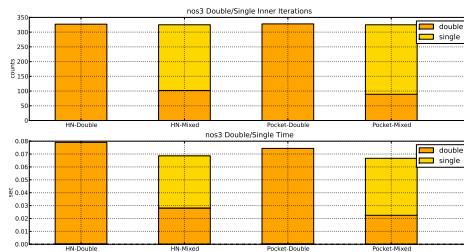


(c) Matrix: nd6k's inner information

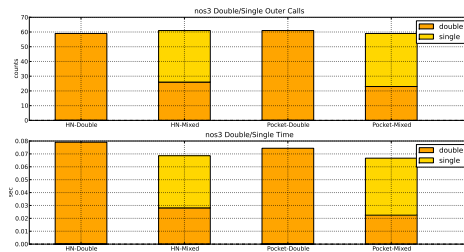


(d) Matrix: nd6k's outer information

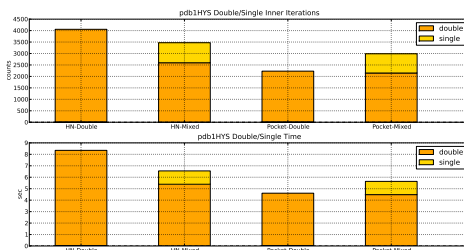
Figure A.49: Matrix inner and outer informations



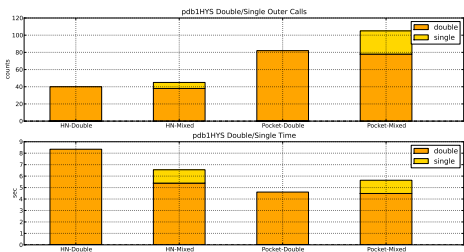
(a) Matrix: nos3's inner information



(b) Matrix: nos3's outer information

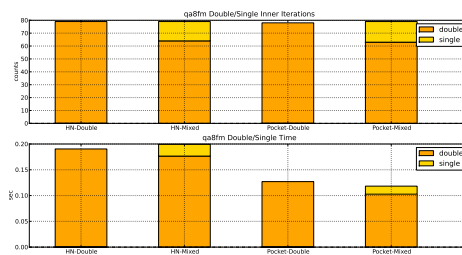


(c) Matrix: pdb1HYS's inner information



(d) Matrix: pdb1HYS's outer information

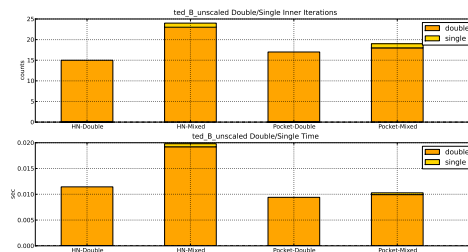
Figure A.50: Matrix inner and outer informations



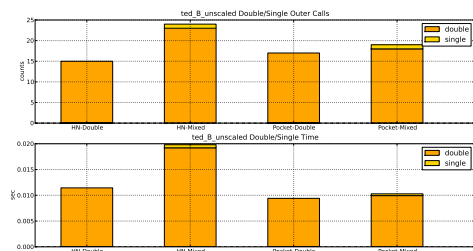
(a) Matrix: qa8fm's inner information



(b) Matrix: qa8fm's outer information

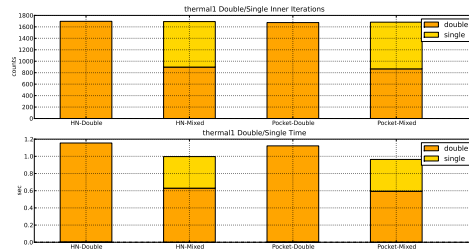


(c) Matrix: ted_B_unscaled's inner information

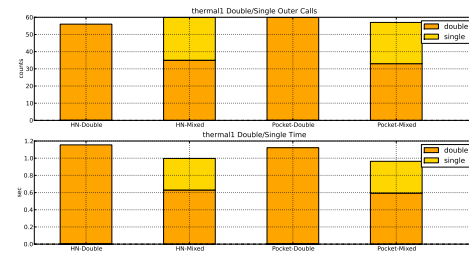


(d) Matrix: ted_B_unscaled's outer information

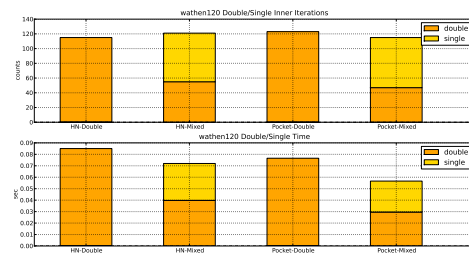
Figure A.51: Matrix inner and outer informations



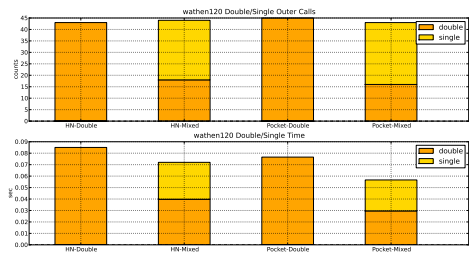
(a) Matrix: thermal1's inner information



(b) Matrix: thermal1's outer information

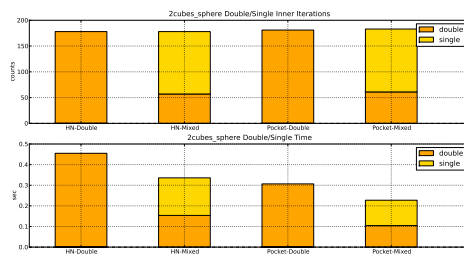


(c) Matrix: wathen120's inner information

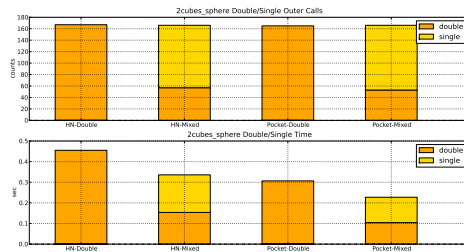


(d) Matrix: wathen120's outer information

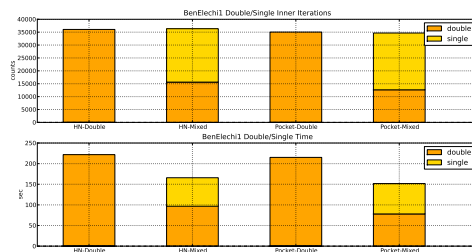
Figure A.52: Matrix inner and outer informations



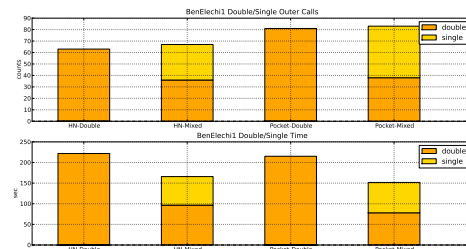
(a) Matrix: 2cubes_sphere's inner information



(b) Matrix: 2cubes_sphere's outer information



(c) Matrix: BenElechi1's inner information

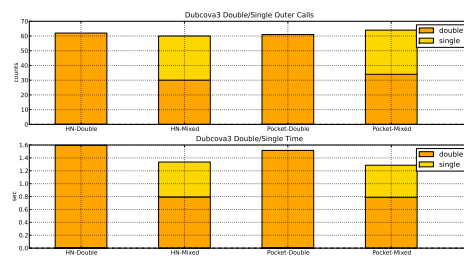


(d) Matrix: BenElechi1's outer information

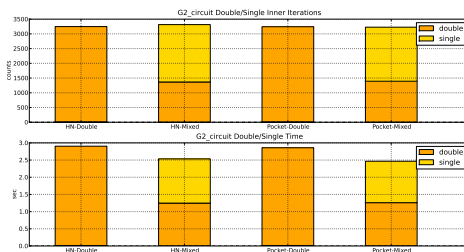
Figure A.53: Matrix inner and outer informations



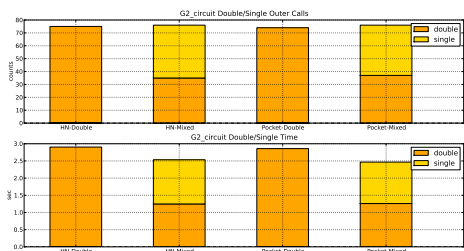
(a) Matrix: Dubcova3's inner information



(b) Matrix: Dubcova3's outer information

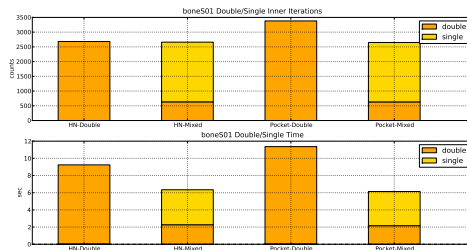


(c) Matrix: G2_circuit's inner information



(d) Matrix: G2_circuit's outer information

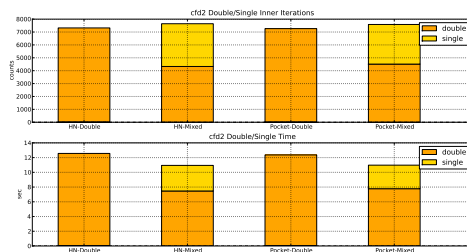
Figure A.54: Matrix inner and outer informations



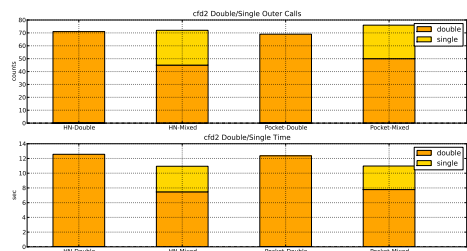
(a) Matrix: boneS01's inner information



(b) Matrix: boneS01's outer information

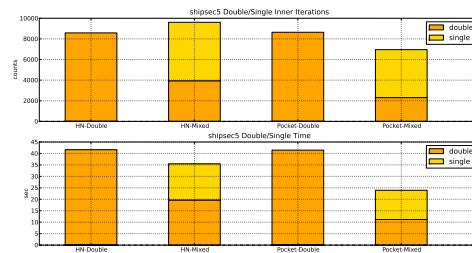


(c) Matrix: cfd2's inner information

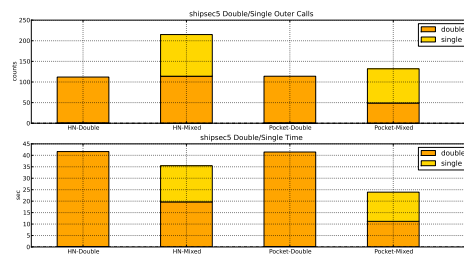


(d) Matrix: cfd2's outer information

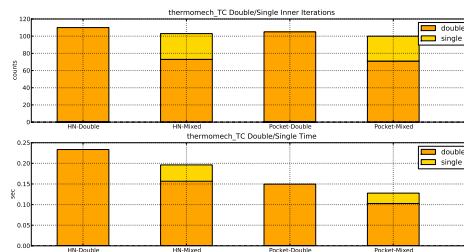
Figure A.55: Matrix inner and outer informations



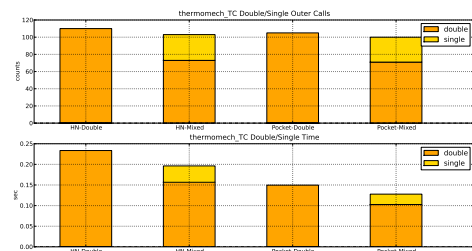
(a) Matrix: shipsec5's inner information



(b) Matrix: shipsec5's outer information

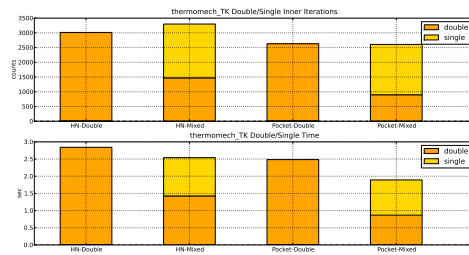


(c) Matrix: thermomech_TC's inner information

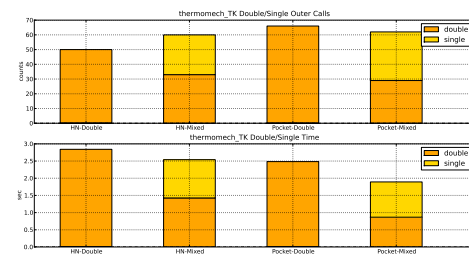


(d) Matrix: thermomech_TC's outer information

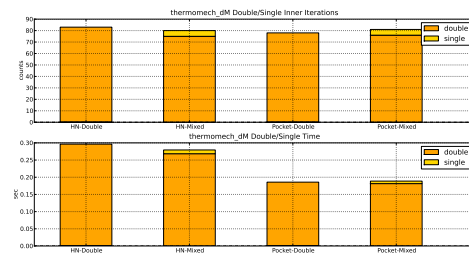
Figure A.56: Matrix inner and outer informations



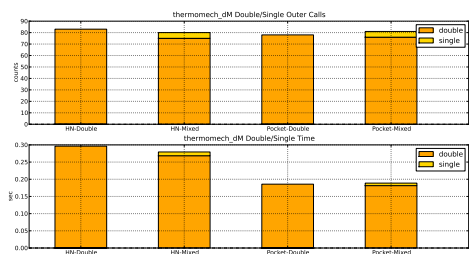
(a) Matrix: thermomech_TK's inner information



(b) Matrix: thermomech_TK's outer information

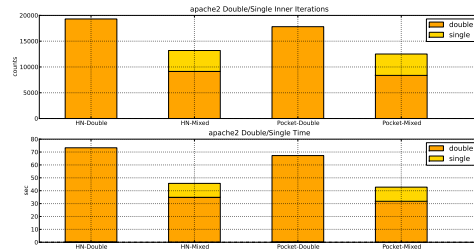


(c) Matrix: thermomech_dM's inner information

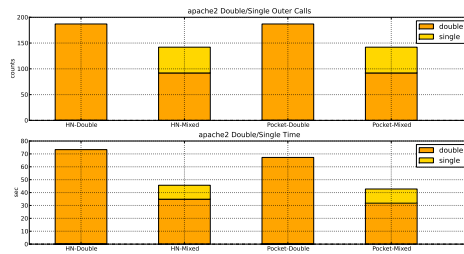


(d) Matrix: thermomech_dM's outer information

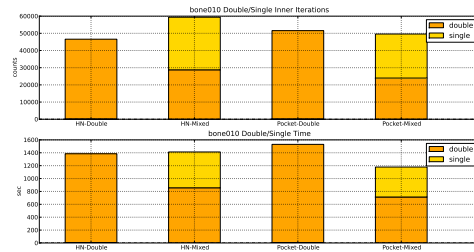
Figure A.57: Matrix inner and outer informations



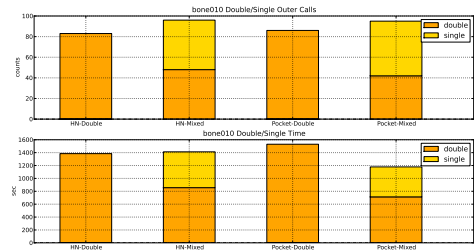
(a) Matrix: apache2's inner information



(b) Matrix: apache2's outer information

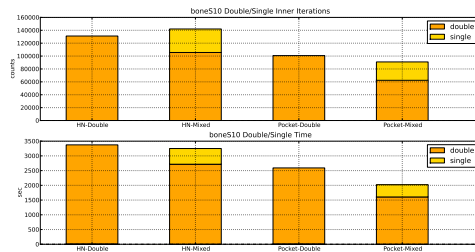


(c) Matrix: bone010's inner information

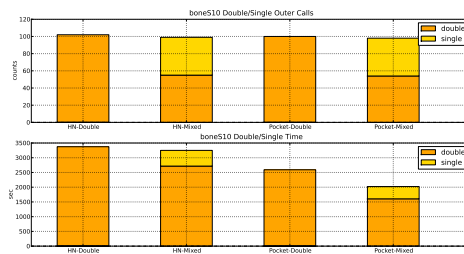


(d) Matrix: bone010's outer information

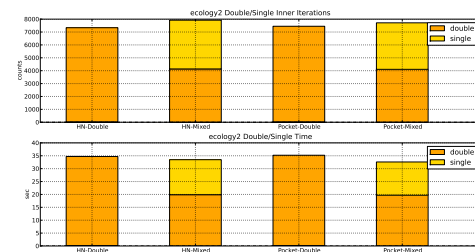
Figure A.58: Matrix inner and outer informations



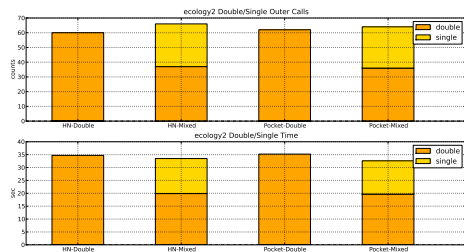
(a) Matrix: boneS10's inner information



(b) Matrix: boneS10's outer information

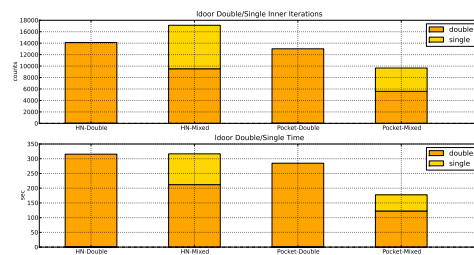


(c) Matrix: ecology2's inner information

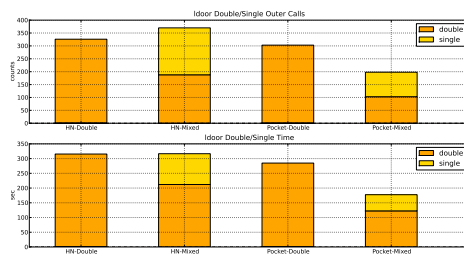


(d) Matrix: ecology2's outer information

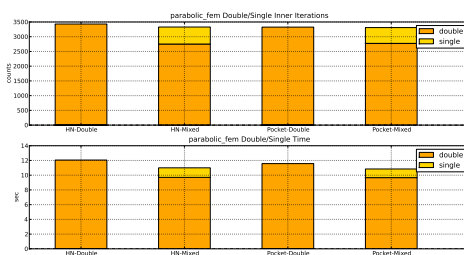
Figure A.59: Matrix inner and outer informations



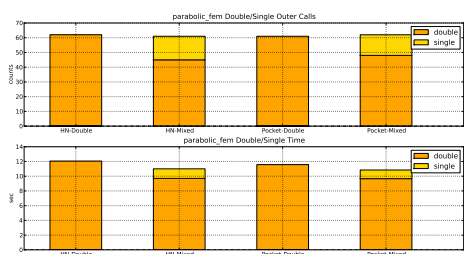
(a) Matrix: ldoor's inner information



(b) Matrix: ldoor's outer information

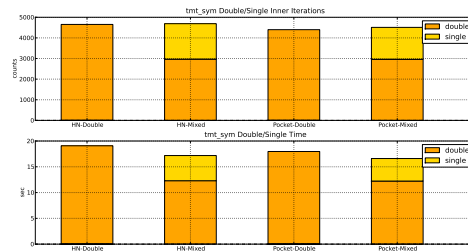


(c) Matrix: parabolic_fem's inner information

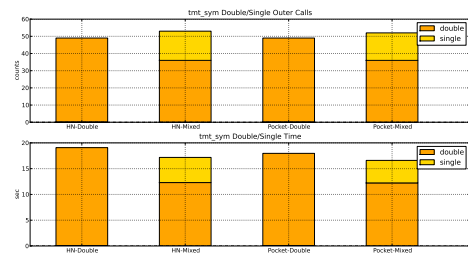


(d) Matrix: parabolic_fem's outer information

Figure A.60: Matrix inner and outer informations

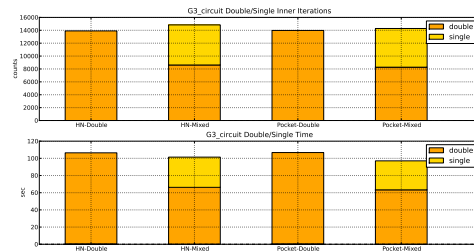


(a) Matrix: tmt_sym's inner information

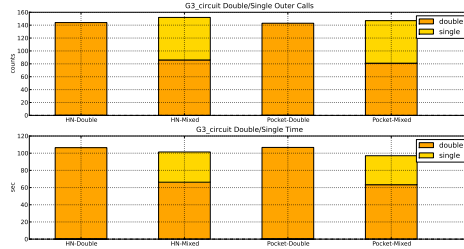


(b) Matrix: tmt_sym's outer information

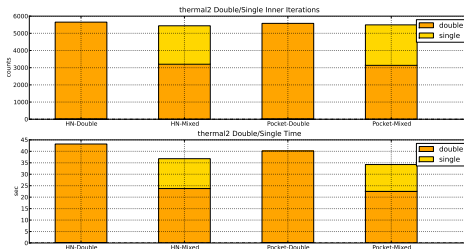
Figure A.61: Matrix inner and outer informations



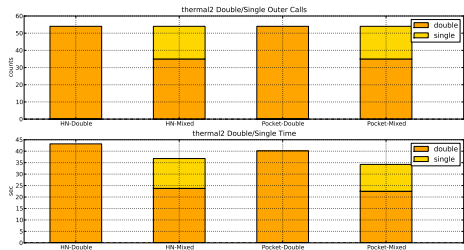
(a) Matrix: G3_circuit's inner information



(b) Matrix: G3_circuit's outer information



(c) Matrix: thermal2's inner information



(d) Matrix: thermal2's outer information

Figure A.62: Matrix inner and outer informations