



Universidad de Investigación de Tecnología  
Experimental Yachay Tech  
School of Mathematics and Computer sciences

Program : Undergraduate \_\_\_\_\_ Semester : VII  
Subject : HPC Professor : Saravana Prakash T  
Date : 14/02/2019 Time:  
Semester: Jan-May-2019  
Student's Surname, Name: Giovanni Caluña  
Student Reg.No: 1725053910

Duration: 10 minutes

INFORMATICS LAB

Maximum Marks: 100

Neatly show all of your work. If you need to use extra paper, please indicate so and attach to this exam.

**Mode of submission : Upload your answers in D2L or hard-copy at my desk**

**Elaboration Date : 07/03/2019**

**Submission Date : 16 /03/2019**

**1. FFT, RANDOM ACCESS, BENCHMARK, system of linear equations in Linpack, class of 06-03-2019, 07-03-2019**

Benchmarks-Chapters-I HPC

1. What is the HPC challenge benchmark?

Combines several benchmarks to test a number of independent attributes of the performance of high-performance computer (HPC) system.

The performance of complex applications of HPC systems can depend in a variety of independent performance attributes of the hardware.

The benchmark currently consists of 7 tests :

- **HPL (High Performance LINPACK):** it measures performance of a solver for a dense system of linear equations.
- **DGEMM:** It measures performance for matrix-matrix multiplication.
- **STREAM:** It measures sustained memory bandwidth to/from memory .
- **PTRANS:** It measures the rate at which the system can transpose a large array.
- **Random Access:** It measures the rate of 64-bit updates to randomly selected elements of a large table.
- **FFT:** It performs a Fast Fourier Transform on a large one-dimensional vector using the generalized Cooley–Tukey algorithm.

- **Communication Bandwidth and Latency:** MPI-centric performance measurements based on the bandwidth/latency benchmark.
2. Get in to the website, [http : //icl.cs.utk.edu/hpcc/hpcc\\_results\\_all.cgi](http://icl.cs.utk.edu/hpcc/hpcc_results_all.cgi), and explain the following questions.

1. How well does HPL data correlate with theoretical peak performance?

- **HPL:** Is a software package that allow the implementation of linpack benchmark, that calculate how faster a computer can solve a system of nxn equations ( $Ax=b$ ) in terms of floating point operations per second (FLOPS). HPL give us a program to quantify the precision of the solution and the required time.
- **Theoretical peak:** Is the maximal theoretical performance a computer can achieve, calculated as the machine's frequency, in cycles per second, times the number of operations per cycle it can perform.

Cluster	HPL	Theoretical peak
Atipa Conquest cluster AMD Opteron	0.252611 Tflop/s	0.3584 Tflop/s
cgna rna Intel Pentium 4	0.00162251 Tflop/s	0.0117 Tflop/s
Clustervision BV Beastie AMD Opteron	0.103764 Tflop/s	0.1536 Tflop/s
ClusterVision/Dell/QLogic	0.63273 Tflop/s	0.758 Tflop/s
Cray XE6 AMD Opteron	64.0194 Tflop/s	81.92 Tflop/s

**The Linpack benchmark (Sustained performance) will always be lower than the theoretical peak performance.**

2. How well does HPL correlate the STREAM-Triad performance?

#### STREAM:

- Memory speed benchmark
- Measure sustainable Memory Bandwidth (GB/s), it means how quickly data can be written to or read from memory by the processor.
  - If memory bandwidth is low, then the processor could be waiting on memory to retrieve or write data.
  - If memory bandwidth is high, then the data needed by the processor can easily be retrieved or written.
- A balanced System will have comparable Memory bandwidth to peak MFlops.
- Machine Balance (MB) is the unit for STREAM =  $\frac{MFlops}{Mem.Bandwidth}$

$MB = 1$       *well balanced*

$MB > 1$       *need very high cache hit rate to achieve useful performance.*

**Triad:** One of the four operations that a machine perform and is measured by STREAM.

$$MB = \frac{PeakFloatingPoint(MFlop/s)}{SustainedTriadMachineBalance(MWords/s)} \quad (1)$$

Cluster	MFlop/s	MW/s	MB
Atipa Conquest cluster AMD Opteron	358400	821.625	435.20
cgna rna Intel Pentium 4	11700	12.295	22.83
Clustervision BV Beastie AMD Opteron	153600	1669.565	92
ClusterVision/Dell/QLogic	768000	1648.725	465.81
Cray XE6 AMD Opteron	81920000	3675.085	22290.64

3. How well does HPL correlate with G-RandomAccess performance?

Random memory performance often maps directly to application performance and application development time. GUPS (Giga Updates per Second) is a measurement that profiles the memory architecture of a system and is a measure of performance similar to MFLOPS. The RandomAccess test is part of the HPC Challenge benchmark developed for the HPCS program. The test intended to exercise the GUPS capability of a system, much like the LINPACK benchmark is intended to exercise the MFLOPS capability of a computer. In each case, we would expect these benchmarks to achieve close to the "peak capability of the memory system. **The extent of the similarities between RandomAccess and LINPACK are limited to both benchmarks attempting to calculate a peak system capability.** GUPS is calculated by identifying the number of memory locations that can be randomly updated in one second, divided by 1 billion (1e9).

Cluster	HPL (TFlop/s)	Ran. Acc. (Gup/s)	T. Peak
Atipa	0.252611	0.00543244	0.3584
cgna rna Intel Pentium 4	0.00162251	0.00175906	0.0117
Clustervision BV Beastie	0.103764	0.0116474	0.1536
ClusterVision/Dell/QLogic	0.63273	0.0140472	0.758
Cray XE6 AMD Opteron	64.0194	0.0614992	81.92

4. How well does HPL correlate with FFT performance?

**FFT:** Measures the floating point rate of execution of double precision complex one-dimensional Discrete Fourier Transform (DFT).

Cluster	HPL	FFT
Atipa Conquest cluster AMD Opteron	0.252611 Tflop/s	0 Gflop/s
cgna rna Intel Pentium 4	0.00162251 Tflop/s	0.383875 Gflop/s
Clustervision BV Beastie AMD Opteron	0.103764 Tflop/s	0.504795 Gflop/s
ClusterVision/Dell/QLogic	0.63273 Tflop/s	1.27869 Gflop/s
Cray XE6 AMD Opteron	64.0194 Tflop/s	1.19525 Gflop/s

5. How well does STREAM data correlate with PTRANS performance?

**PTRANS:** Measure the rate of transfer for large arrays of data from Multiprocessors Memory in terms of GB/s, similar to STREAM with the memory bandwidth. .

Cluster	STREAM	PTRANS
Atipa Conquest cluster AMD Opteron	1.64325 GB/s	3.24707 GB/s
cgna rna Intel Pentium 4	1.02459 GB/s	0.0437215 GB/s
Clustervision BV Beastie AMD Opteron	3.33913 GB/s	0.815936 GB/s
ClusterVision/Dell/QLogic	3.29745 GB/s	7.30084 GB/s
Cray XE6 AMD Opteron	7.35017 GB/s	173.145 GB/s

6. Draw Kiviati chart of any 5 different machines (clusters), and explain your understanding.

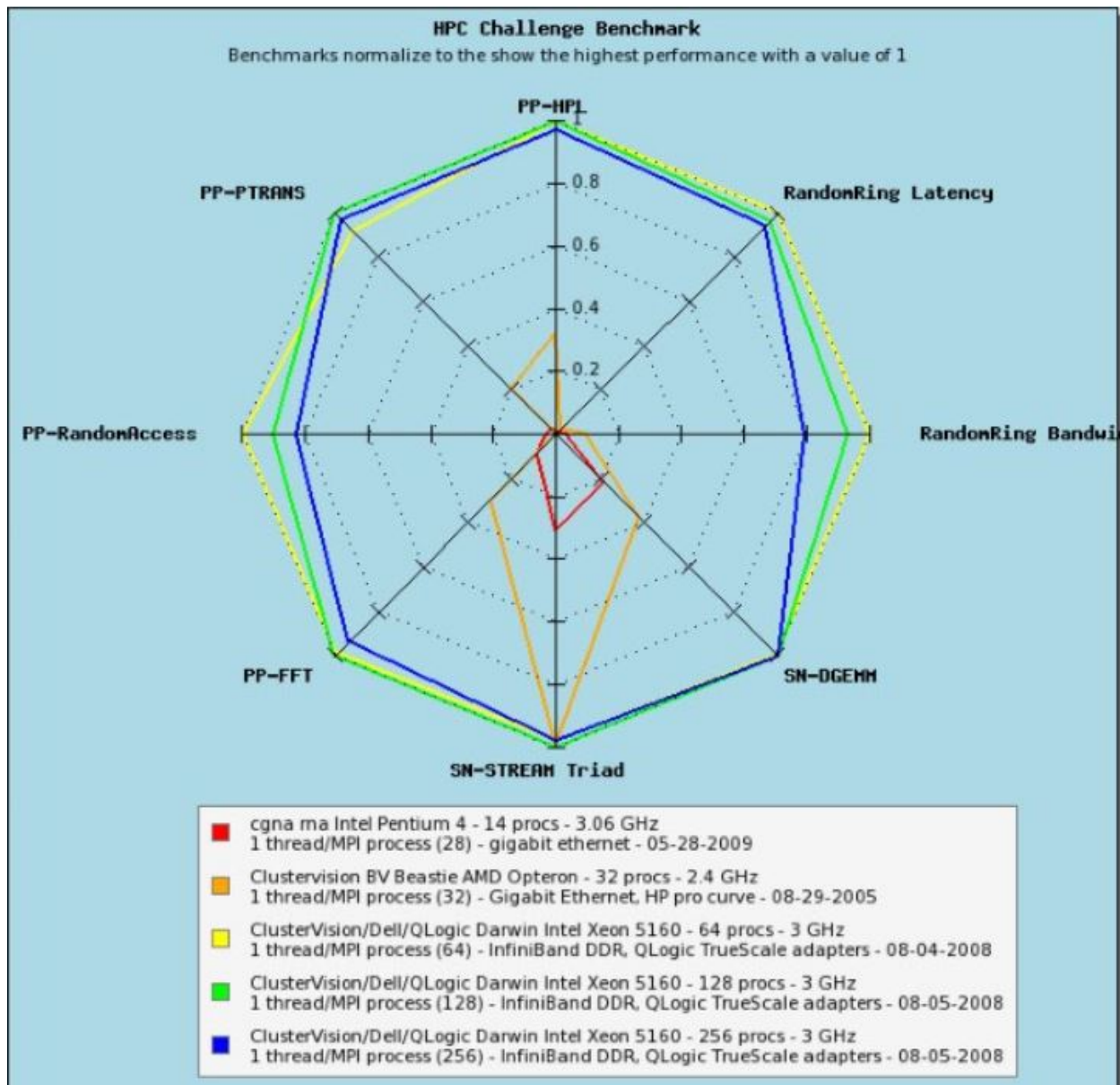


Figura 1: Kiviati chart of 5 different machines

We can observe in the Kiviati Chart 1 that the yellow, blue and green machine has the best

performance in terms of HPC challenge. They present a very well and balanced efficiency with respect to the 7 points of HPC challenge. The similarity between them is due to the same machine with different amount of MPI process. (Yellow 64, Green 128, Blue 256). On another hand, we see the orange and red machine. The orange machine has a great STREAM-Triad performance, but it is the unique advantage of it. Last, we have the red machine. It has a poor performance in the seven aspects measured by the HPC challenge Benchmark. Its performance almost is insignificant in relation with the rest of machines.

7. Explain Geometric mean and weighted geometric mean.

**Geometric mean:** Geometric mean of normalized results for G-HPL, G-RandomAccess, EP-STREAM-Sys, and G-FFT. These are the four performance results which are featured in the HPCC Awards. The normalization is done independently for each column rather than against a single machine's results. Consequently, the value of mean will change over time as faster machines appear in the HPCC database.

**Weighted geometric mean:** It generates results of Geometric mean and weighted geometric mean taking into account the variation of: G-HPL weight, G-RandomAccess weight, G-FFT weight, EP-STREAM Triad System weight.

8. What is Ping-pong latency and ping-pong bandwidth.

The latency and bandwidth benchmark measures two different communication patterns. First, it measures the single-process-pair latency and bandwidth, and second it measures the parallel all-processors-in-a-ring latency and bandwidth. For the first pattern, ping-pong communication is used on a pair of processors. Several different pairs are reported. While the ping-pong benchmark is executed on one process pair, all other processors are waiting in a blocking receive. To limit the total benchmark time used for this first pattern to 30 sec, only a subset of the set of possible pairs is used.

For benchmarking latency and bandwidth, 8 byte and 2,000,000 byte long messages are used. The major results reported by this benchmark are:

- maximal ping pong latency
- average latency of parallel communication in randomly ordered rings
- minimal ping pong bandwidth
- bandwidth per process in the naturally ordered ring.
- average bandwidth per process in randomly ordered rings.

9. How Does Cholesky Factorization Work and purpose of it? How this algorithm implemented in Lapack.

In linear algebra, the Cholesky factorization is a decomposition of a Hermitian, positive-definite matrix into the product of a lower triangular matrix and its conjugate transpose, which is useful for efficient numerical solutions.

- **Hermitian:** is a complex square matrix that is equal to its own conjugate transpose, it means that the element in the  $i$ -th row and  $j$ -th column is equal to the complex conjugate of the element in the  $j$ -th row and  $i$ -th column, for all indices  $i$  and  $j$ .

$$\mathbf{A} \text{ Hermitian} \iff A = \overline{A^T}$$

- **Positive-definite matrix:**  $M$  is said to be positive definite if the scalar  $z^* M z$  is strictly positive for every non-zero column vector  $z$  of  $n$  complex numbers. Here  $z^*$  denotes the conjugate transpose of  $z$ . Note that  $z^* M z$  is automatically real since  $M$  is **Hermitian**.
- **Lower triangular matrix:** A square matrix is called lower triangular if all the entries above the main diagonal are zero
- **Conjugate transpose:** or **Hermitian transpose** of an  $m - by - n$  matrix  $A$  with complex entries is the  $n - by - m$  matrix  $A^H$  obtained from  $A$  by taking the transpose and then taking the complex conjugate of each entry.

A defined symmetric and positive matrix  $A$  can be efficiently factored by means of an inferior triangular matrix and an upper triangular matrix. For a non-singular matrix the  $LU$  decomposition leads us to consider a decomposition of such type  $A = LU$ ; given the conditions of  $A$ , **symmetric and positive definite**, it is not necessary to pivot, so this factorization is done efficiently and in a number of operations half of  $LU$  taking the form  $A = LL^T$ , where  $L$  is a **lower triangular matrix** where the elements of the diagonal are positive.

To solve a linear system  $Ax = b$  with  $A$  positive definite symmetric and given its Cholesky factorization, we must:

- First **solve  $Ly = b$  to get  $y$** .
- Then **solve  $L^T x = y$  to get  $x$** .

10. Explain the steps of Cholesky Factorization, and Algorithm.

To find the factorization  $A = LL^T$ , it would be enough to see the shape of  $L$  and observe the equations that the right product leads us to equal elements:

$$\begin{pmatrix} a_{11} & a_{21} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ a_{31} & a_{32} & \cdots & a_{3n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \cdots & 0 \\ l_{31} & l_{32} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{pmatrix} \begin{pmatrix} l_{11} & l_{21} & \cdots & l_{1n} \\ 0 & l_{22} & \cdots & l_{2n} \\ 0 & 0 & \cdots & l_{3n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & l_{nn} \end{pmatrix}$$

we obtain that:

$$\begin{aligned} a_{11} &= l_{11}^2 \longrightarrow l_{11} = \sqrt{a_{11}} \\ a_{21} &= l_{21} l_{11} \longrightarrow l_{21} = a_{21} / l_{11} \\ a_{22} &= l_{21}^2 + l_{22}^2 \longrightarrow l_{22} = \sqrt{a_{22} - l_{21}^2} \\ a_{32} &= l_{31} l_{21} + l_{32} l_{22} \longrightarrow l_{32} = (a_{32} - l_{31} l_{21} / l_{22}) \end{aligned}$$

in a general manner for  $i=1, \dots, n$  and  $j=1, \dots, n$

$$l_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2}$$

$$l_{ji} = (a_{ji} - \sum_{k=1}^{i-1} l_{jk}l_{ik})/l_{ii}$$

Now, since  $A$  is **symmetric** and **positive definite**, we can assure that the elements on the diagonal of  $L$  are positive and the remaining elements are reals.

11. Solve the following systems of linear equations using Cholesky Factorization through fortran or C. Send your code, results and calculate the floating point rate of this problem.

Listing 1: fortran version

```

1      Program dposv_example
2      !      DPOSV Example Program Text
3
4      !      Copyright (c) 2018, Numerical Algorithms Group (NAG Ltd.)
5
6      !      .. Use Statements ..
7      Use lapack_example_aux, Only: nagf_file_print_matrix_real_gen
8      Use lapack_interfaces, Only: dposv
9      Use lapack_precision, Only: dp
10     !      .. Implicit None Statement ..
11     Implicit None
12     !      .. Parameters ..
13     Integer, Parameter :: nin = 5, nout = 6
14     !      .. Local Scalars ..
15     Integer :: i, ifail, info, lda, n
16     !      .. Local Arrays ..
17     Real (Kind=dp), Allocatable :: a(:, :), b(:)
18     !      .. Executable Statements ..
19     Write (nout, *) 'DPOSV Example Program Results'
20     Write (nout, *)
21     !      Skip heading in data file
22     Read (nin, *)
23     Read (nin, *) n
24     lda = n
25     Allocate (a(lda,n), b(n))
26
27     !      Read the upper triangular part of A from data file
28
29     Read (nin, *) (a(i,i:n), i=1, n)
30     !      Read b from data file
31     Read (nin, *) b(1:n)
32
33     !      Solve the equations Ax = b for x
34     Call dposv('Upper', n, 1, a, lda, b, n, info)
35
36     If (info==0) Then
37
```

```

38  !           Print solution
39
40      Write (nout, *) 'Solution '
41      Write (nout, 100) b(1:n)
42
43  !           Print details of factorization
44
45      Write (nout, *)
46      Flush (nout)
47
48  !           ifail: behaviour on error exit
49  !           =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
50      ifail = 0
51  Call nagf_file_print_matrix_real_gen('Upper', 'Non-unit diagonal', n, &
52      n, a, lda, 'Cholesky factor U', ifail)
53
54      Else
55      Write (nout, 110) 'The leading minor of order ', info, &
56      ' is not positive definite '
57      End If
58
59 100  Format ((3X,7F11.4))
60 110  Format (1X, A, I3, A)
61      End Program

```

---

Listing 2: fortran version

---

1	DPOSV Example	Program	Data	
2	4			: Value of N
3	4.16	-3.12	0.56	-0.10
4		5.03	-0.83	1.18
5			0.76	0.34
6				1.18 :End of matrix A
7	8.70	-13.35	1.89	-4.14 :End of vector b

---



```
osv_example.exe < ../data/dposv_example.d
DPOSV Example Program Results

Solution
      1.0000      -1.0000      2.0000      -3.0000

Cholesky factor U
      1          2          3          4
1      2.0396     -1.5297      0.2746     -0.0490
2          1.6401     -0.2500      0.6737
3          0.7887      0.6617
4          0.5347
```

12. Use Lapack routine to solve this problem. For  $f(x, y) = 4x + 2y - x^2 - 3y^2$  a) Find the gradient. Use that to find a critical point (x,y) that makes the gradient 0. b) Use the eigenvalues of the Hessian at that point to determine whether the critical point in a) is a maximum, minimum, or neither.

a)

We have:

$$f(x, y) = 4x + 2y - x^2 - 3y^2$$

So, calculate the gradients

$$\nabla f(x, y) = \left[ \frac{\partial f(x, y)}{\partial x}, \frac{\partial f(x, y)}{\partial y} \right]$$

$$\frac{\partial f(x, y)}{\partial x} = \frac{\partial}{\partial x}(4x + 2y - x^2 - 3y^2) = 4 - 2x$$

$$\frac{\partial f(x, y)}{\partial y} = \frac{\partial}{\partial y}(4x + 2y - x^2 - 3y^2) = 2 - 6y$$

$$\nabla f(x, y) = [4 - 2x, 2 - 6y]$$

Then, calculate the critical points that makes the gradient 0

$$\begin{aligned} 4 - 2x &= 0 & \text{so, } x &= 2 \\ 2 - 6y &= 0 & \text{so, } y &= \frac{1}{3} = 0,33 \end{aligned}$$

The critical point of the equations are x=2 and y =0.33.

b)

$$H(x, y) = \begin{pmatrix} \frac{\partial^2 f(x, y)}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix}$$

$$H(x, y) = \begin{pmatrix} \frac{\partial}{\partial x}(4 - 2x) & \frac{\partial}{\partial y}(4 - 2x) \\ \frac{\partial}{\partial x}(2 - 6y) & \frac{\partial}{\partial y}(2 - 6y) \end{pmatrix}$$

$$H(x, y) = \begin{pmatrix} -2 & 0 \\ 0 & -6 \end{pmatrix}$$

Then, using the Hessian matrix we evaluate that, if the critical points in a) are the maximum and minimum

$$\det(A - I\lambda) = \begin{vmatrix} -2 - \lambda & 0 \\ 0 & -6 - \lambda \end{vmatrix}$$

$$= (-2 - \lambda)(-6 - \lambda) = (2 + \lambda)(6 + \lambda)$$

$$\Rightarrow \lambda_1 = -2, \quad \lambda_2 = -6$$

Finally, using a DGEEV subroutine we calculate the eigenvalues and eigenvectors of the H matrix.

```
DGEEV Example Program Results

Eigenvalue( 1) = -2.0000E+00

Eigenvector( 1)
  1.0000E+00
  0.0000E+00

Eigenvalue( 2) = -6.0000E+00

Eigenvector( 2)
  0.0000E+00
  1.0000E+00
```

Figura 2: Eigenvalues and Eigenvectors

We have to calculate the determinant of the Hessian matrix. In this case the value of the determinant is 12.

If the value of the determinant is greater than 0, then we have to analyze two possible solutions to determine if the function has a minimum or maximum value.

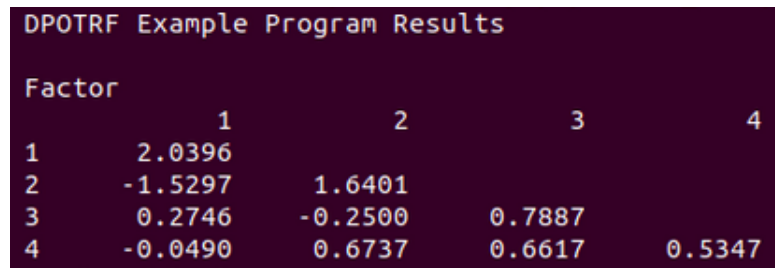
if the element of the matrix in the position  $H_{1x1}$  is less than 0 we have a maximum value. On the contrary, if the element is greater than 0 we have a minimum value.

In this case, the element in that position is -2 therefore in this exercise we have a **maximum value**.

13. Explain what is positive-definite matrix and negative-definite matrix. Using Lapack solve the above said two matrix types and demonstrate your results. What type of Hessian (Under what condition of Hessian matrix) you will have Saddle point Eigen values (Eigen value mapping as a saddle surface).

**Positive-definite matrix:** A positive definite matrix is a symmetric matrix whose eigenvalues are all positive. The scalar product  $z^T M z$  is strictly positive for every non-zero column vector  $z$  of  $n \times n$  real numbers.

**Negative-definite matrix:** A negative definite matrix is a symmetric matrix whose eigenvalues are all negative. The scalar product  $z^T M z$  is strictly negative for every non-zero column vector  $z$  of  $n \times n$  real numbers.



Factor	1	2	3	4
1	2.0396			
2	-1.5297	1.6401		
3	0.2746	-0.2500	0.7887	
4	-0.0490	0.6737	0.6617	0.5347

Figura 3: Positive-definite matrix

In this case, we use cholesky factorization to obtain eigenvalues of the matrix

Listing 3: fortran version

---

```

1 DPOTRF Example Program Data
2   4                               : Value of N
3   'L'                             : Value of UPLO
4   4.16
5   -3.12    5.03
6   0.56    -0.83    0.76
7   -0.10    1.18    0.34    1.18    : End of matrix A

```

---

The result is that all eigenvalues of the matrix are positive. In conclusion, we may say that this matrix is positive definite.

## 2. Results of HPCC

### 2.1. MPI Random Access - Words

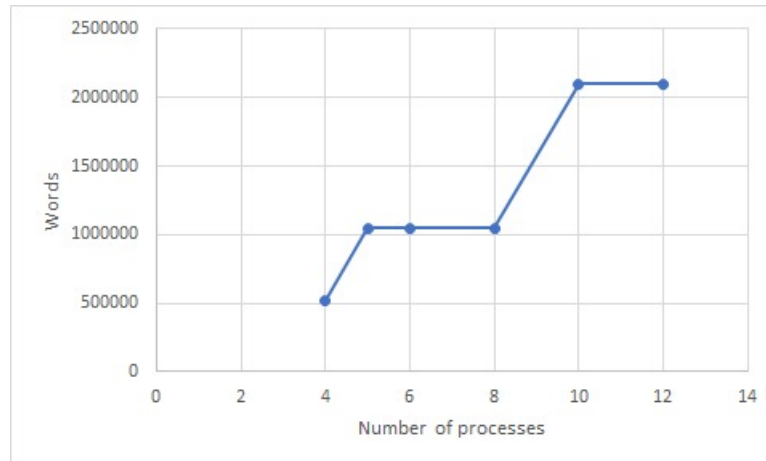


Figura 4: N words vs N processors

### 2.2. MPI Random Access - Time

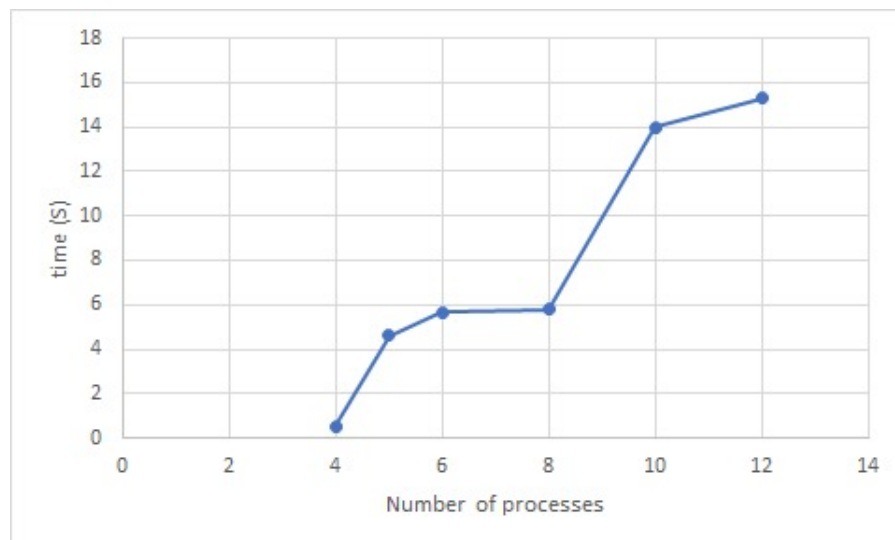


Figura 5: Time vs N processors

### 2.3. MPI Random Access - Total updates

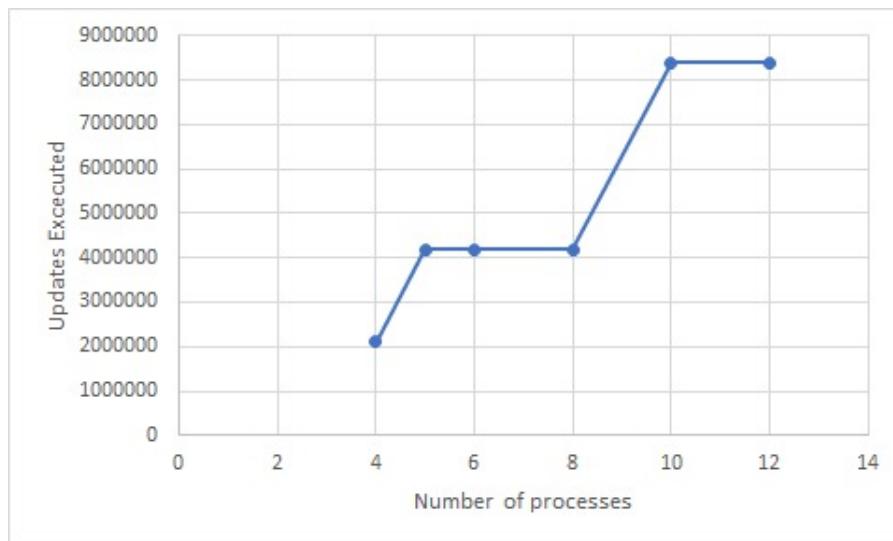


Figura 6: Updates/s vs N processors

### 2.4. MPI Random Access - Updates/sec

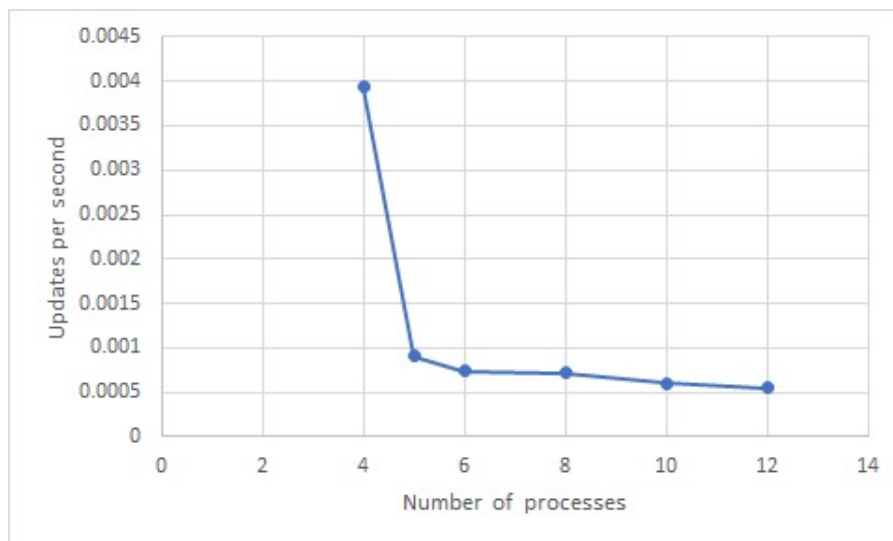


Figura 7: Total Updates vs N processors

Analyzing the graphs, we can see: First in the 4, obviously the number of words increase when the number of processors increase. However, they are constant until more than 4 processors are added. This fact is reflected in the graph of time 5, when we have more words to process we spent more time in the CPU. Also, in the graph 6, where all the updates are made by each execution of the program, we can see that the graph looks very similar to the graph of number

of words because each of them must be processed, so when there are more words, the machine must execute more updates. For another hand, with the graph of updates per second 7, we can see something interesting. The best rate is reached when we use 4 processors. It happens because of the machine can manage just four processors at a time. So, when we add processors the updates get slower because they have to wait for a processor.

### 2.4.1. FFT-Vector Size

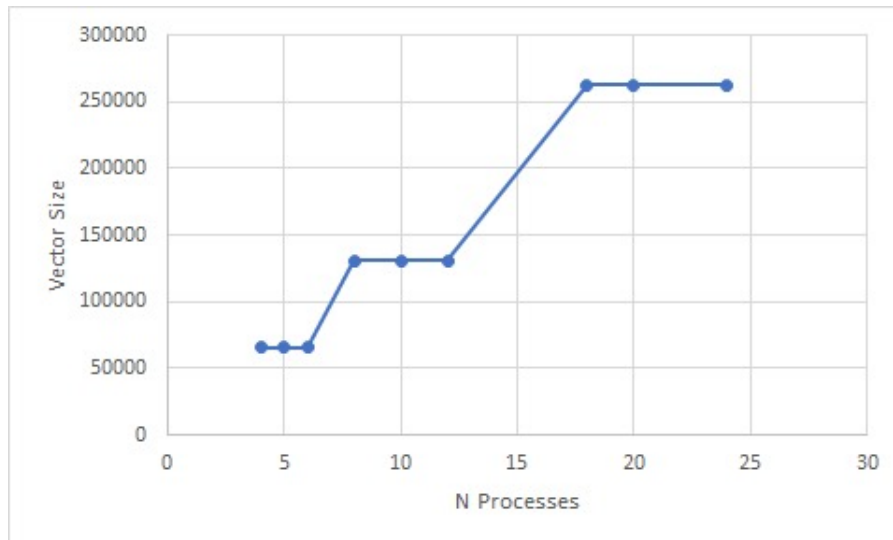


Figure 8: N of processors vs Vector Size

### 2.5. FFT-Computing Time

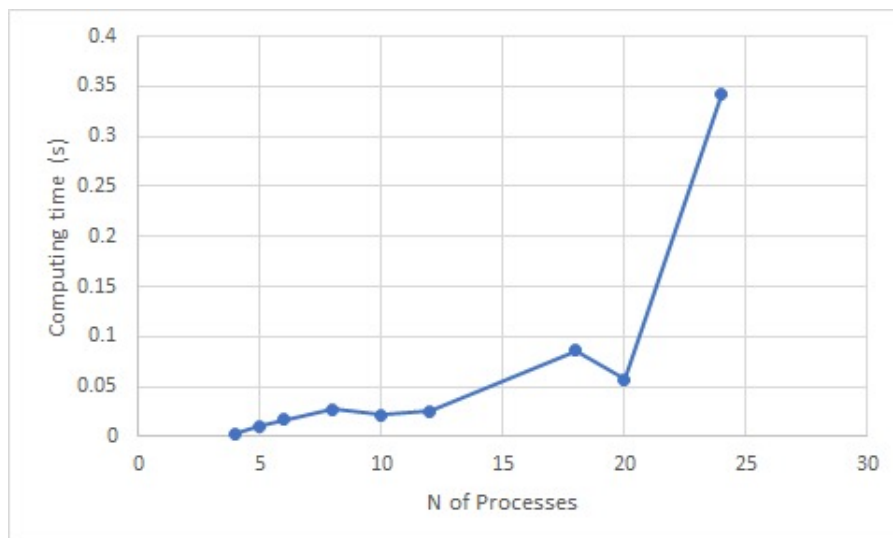


Figure 9: N of processors vs Computing time

## 2.6. FFT-Gflops/s

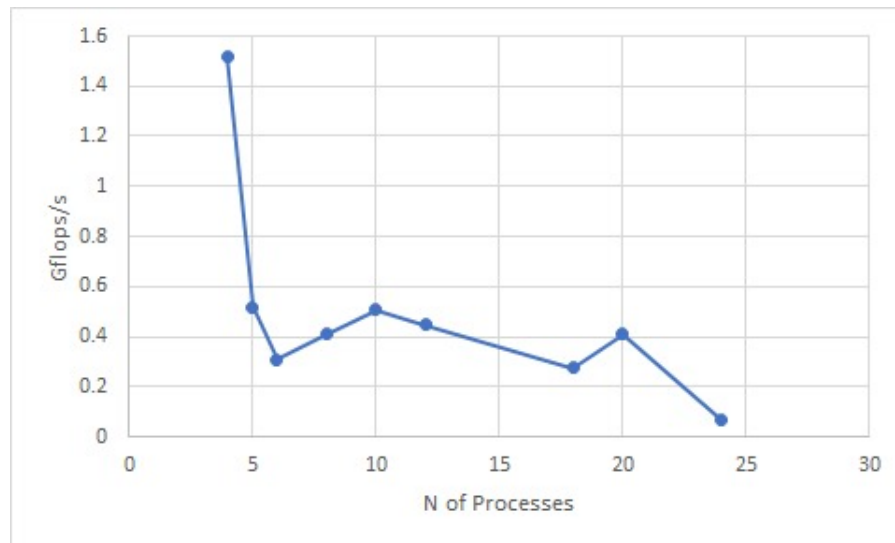


Figure 10: N of processors vs Gflops/s

## 2.7. FFT-Inverse Time

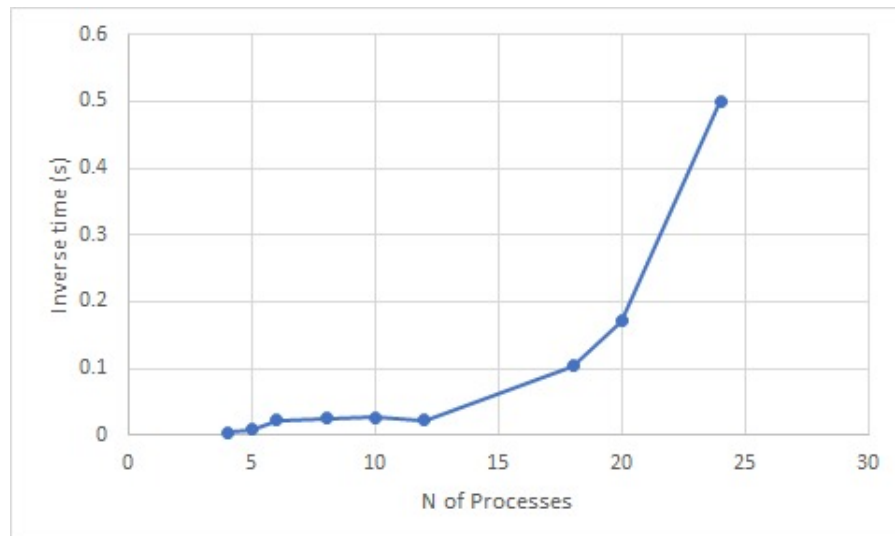


Figure 11: N of processors vs Inverse Time

In the graphics of FFT, we can see that: First, in the figure 8, the vector size increases only after 4 processors. Also, the size of the vector is the double of the previous. Related with this fact, we see in the graph 9, that the computing time increase very fast when the size of the vector is big. Also, the computing time is almost the same although the number of processors increase. It means that it only depends on the size of the vector. Analyzing the graph 10 of speed, we see that it starts in the best point, with 4 processors. After that, when the number



of processors and the vector size increase the speed gets so much slower. In the graphic 11 of the inverse time , we can see that the perform the inverse just depends on the size vector. We can assume this because the time with 4 processors is almost the same that with 8 processors. Also, we can see that the inverse time increase exponentially with the size of the vector.