INTRODUÇÃO À FÍSICA COMPUTACIONAL

Projeto 1

Luiz Fernando S E Santos, 10892680

1. Introdução

Na presente prática, fomos apresentados à conceitos e funções básicas do Fortran, bem como diversas situações e problemas nos quais elas foram úteis para sua resolução.

Os 5 exercícios a seguir foram apresentados como motivação para a elaboração de algoritmos, que por sua vez foram traduzidos em código de Fortran e testados. Seus resultados foram então analisados e discutidos.

2. Metodologia

2.1 Fatoriais e a aproximação de Stirling

(a)

```
    PROGRAM fatorial
    IMPLICIT NONE

    INTEGER*8 n, fat, aux    ! variáveis ques serão usadas

    DO n = 1,20         ! n irá variar de 1 à 20. Ele será o número o qual queremos o fatorial
       fat = 1    ! fat (fatorial) é o valor do fatorial. Para cada passo de n, ele será iniciado como 1
       DO aux = 1,n     ! o aux (auxiliar) será o passo de 1 até n
         fat = fat*aux    ! como aux será igual à a cada passo 1; 2; ...; n, multiplicar fat por ele garente que ele será multiplicado por cada número inteiro de 1 até n
       END DO
       WRITE(1,*)n,'! =', fat    ! por fim, escrevo num arquivo fort.1 o respetivo n que estou calculando o fatorial e o valor de “n!”
    END DO

    END PROGRAM
```

Conforme mostra o programa, varia-se ‘n’ de 1 até 20, calculando seu fatorial e escrevendo em um arquivo de texto a tabela que mostra cada ‘n’ e em seguida seu respectivo fatorial. O tipo INTEGER*8 foi usado para permitir números maiores, assim melhorando a precisão.

(b)

```
PROGRAM logaritmo
IMPLICIT NONE

INTEGER*8 n, aux
REAL*8 fat                    ! declaração das variáveis

DO n = 2,30          ! n é o número o qual queremos calcular o fatorial. Ele varia de 2 à 30
  fat = 1     ! variável fat (fatorial) inicia como 1 para cada n
  DO aux = 1,n
    fat = fat*FLOAT(aux)! cálculo do fatorial de n como descrito em (a)
  END DO
  WRITE(1,*)'LN(FAT(',n,'))','=', LOG(fat)        ! desta vez, escrevo no documento fort.1 uma tabela com cada n e o respectivo valor do logaritmo natural de seu fatorial
END DO

END PROGRAM
```

É usado o mesmo método descrito em (a) para o cálculo do fatorial, porém desta vez a tabela é feita com o valor do logaritmo natural do fatorial de ‘n’ obtido anteriormente.

(c)

```
PROGRAM stirling
IMPLICIT NONE

REAL*8,PARAMETER :: pi = 3.14159265359d0          ! defino o valor parâmetro pi com precisão de 10^-11
INTEGER*8 n      ! defino a variável n que será usada de contador

DO n = 2,30
  WRITE(1,*)S,'=', FLOAT(n)*LOG(FLOAT(n)) - FLOAT(n) + 0.5*LOG(2*pi*FLOAT(n))    ! para cada n, calculo a fórmula descrita com seu valor
END DO

END PROGRAM
```

Vario ‘n’ de 2 até 30, escrevendo em um documento de texto “fort.1” o resultado de $\ln(n!) \approx S = n*\ln(n) - n + \frac{1}{2}\ln(2\pi n)$, sendo esta a aproximação de Stirling.

Agora, utilizando os três códigos anteriores, a fim de analisar a precisão da aproximação, fazemos:

```
                PROGRAM final
                IMPLICIT NONE

                INTEGER*8 n, aux
                REAL*8,PARAMETER :: pi = 3.14159265359d0
                REAL*8 fat, S

                DO n = 2,30
          fat = 1
                  DO aux = 1,n
                     fat = fat*FLOAT(aux)
                  END DO
                  S = FLOAT(n)*LOG(FLOAT(n)) - FLOAT(n)
+0.5*LOG(2*pi*FLOAT(n))
                  WRITE(2,*)n, LOG(fat), S, (LOG(fat)-S)/LOG(fat)
                END DO

                END PROGRAM
```

O código acima é muito similar é feito juntando os anteriores, com a diferença sendo que escrevemos, em um arquivo de texto “fort.2”, o valor de n, o logaritmo de seu fatorial e “S”, que representa a aproximação de Stirling para o mesmo valor. A quarta coluna da tabela será a precisão do resultado obtido, calculado por (fat(n!)-S)/fat(n!).

2.2 Série de Taylor para o cosseno

```
    PROGRAM taylor
    IMPLICIT NONE

    REAL*8   :: fat
    INTEGER*8   :: n
    REAL*8   :: T, termo
    REAL*8,PARAMETER  :: pi = 3.14159265359d0
    LOGICAL  :: cont
    INTEGER*8   :: passo
    REAL*8   :: aux
    INTEGER*8        ::         ordem
    !DEFINIÇÃO DAS VARIÁVEIS QUE SERÃO USADAS

    DO passo = 1,50
      aux = (pi/50)*FLOAT(passo)    !defino 50 pontos igualmente espaçados entre 0 e pi
      ordem = 0
      T= 0     !valor do polinômio de Taylor = 0 cada vez que é feita a aproximação com um
novo valor
      n= 0
      fat= 1.0d0
      cont= .TRUE.      !idem para as demais variáveis que serão incrementadas no próximo
loop
      DO WHILE (cont)
        IF(MOD(n,2).EQ.0) THEN     !como os termos ímpares do polinômio em torno de zero
são multiplicados por SIN(0), somo apenas os pares
          termo= (aux**n)/fat     !da fórmula do pol de Taylor
          fat= -fat      !como no caso do cosceno há a variação de soma e subtração do termo,
defino o sinal na variável já no fatorial, que multiplica as demais
          T= T + termo      !soma do termo ao pol em sí
        ENDIF

        ordem = ordem + 1
        n= n+1      !o próximo loop será para o termo n+1 do polinômio
        fat= fat*FLOAT(n)    !multiplico aqui o antigo fatorial pelo próximo termo, assim
obtendo o próximo fatorial

        IF (termo.LE.0.000001 .AND. termo.GE.-0.000001) THEN  !caso o último termo
gerado do polinômio esteja em [-10^6, 10^6], posso garantir que esta será a precisão do meu
polinômio
          cont= .FALSE.     !quebro o loop e começo para o próximo X0
        ENDIF
      END DO

      WRITE(2,*)aux,',',T     !escrevo os valores de X e T(X) para serem plotados pelo
programa 'graphing.py'
      WRITE(1,*) 'COS(X)= ', COS(aux), '// T(X)= ', T, '// ordem= ', ordem     !em outro arquivo
coloco os valores originais do COS(X) e o obtido por Taylor

    END DO
    END PROGRAM
```

Os exercícios (a) e (b) são feitos juntos no programa acima. Ao término, teremos duas tabelas: uma com o valor “real” cosceno de x (x representado pelo contador “aux” no programa), calculado diretamente pela função COS(), o valor obtido com o polinômio de Taylor T(x), e por fim a ordem de expansão do polinômio que foi necessária para atinjir a precisão pedida de 10^-6.

A outra tabela tem o propósito exclusivamente de ser usada para facilitar a representação gráfica dos resultados obtidos através de um outro programa em Python, utilizando as bibliotecas Matplotlib e Numpy.

2.3 Valores médios e desvio padrão

(a)

```
PROGRAM VMeDP
IMPLICIT NONE

        INTEGER*8, DIMENSION (10**6) :: votes
        INTEGER*8 :: i
        INTEGER*8 :: aux = 0
        REAL*8 :: result

        OPEN(UNIT=1,FILE="votes.dat",ACTION="READ")
        READ(UNIT=1,FMT=*) votes        ! Leio o arquivo votes.dat e armazeno os valores contidos nele em no vetor "votes"
        CLOSE(UNIT=1)

        DO i = 1, 10**6
                aux = aux + votes(i)    ! percorro todo o vetor, somando seus valores em aux
        END DO

        result = FLOAT(aux)/(10**6)! O resultado da média dos velores contidos no vetor, "result", é calculada dividindo a soma dos termos pelo tamanho do vetor
        WRITE(*,*) result

        IF (result.LE.0.5) THEN
                WRITE(*,*) 'O vencedor é 0' ! se a média é menor ou igual à 0.5, o vencedor é 0
        ELSE
                WRITE(*,*) 'O vencedor é 1' ! se for maior que 0.5, o vencedor é 1
        ENDIF

END PROGRAM
```

O programa descrito acima simplesmente calcula a média aritmética dos números contidos em votes.dat, somando todos e dividindo pelos 10^6 votos totais. Caso o resultado seja maior que 0.5, o candidato 1 ganha. Caso contrário 0 ganha.

(b) e (c)

```
PROGRAM prog
IMPLICIT NONE

INTEGER*8   ::      k, i
REAL*8      ::      rkiss05
INTEGER     :: M
REAL*8, DIMENSION(100):: medias
INTEGER*8   ::      index
REAL*8              ::      xmed
INTEGER, DIMENSION(10**6)   ::      votes
REAL*8              :: X = 0
REAL*8              ::      sm = 0

!!!DEFINIÇÃO DAS VARIÁVEIS!!!

CALL kissinit(42)
OPEN(UNIT=1, FILE="votes.dat", ACTION='READ')
READ(UNIT=1, FMT=*) votes       !       leio o arquivo 'votes.dat' e
armazeno em um array
CLOSE(UNIT=1)

DO k = 1, 100 ! k variando de 1 à 100 (100 amostras diferentes)
        xmed = 0
        M = int(rkiss05()*10**4)    ! M será o tamanho da minha amostra
sorteda, portanto ela terá um tamanho <= 10^4
        DO i = 1, M    ! crio uma estrutura de repetição de 1 à M, o que me
dará M elementos para a amostra
                index = int(rkiss05()*10**6)
                xmed = xmed + votes(index)          !Sorteio também
aleatoriamente um indice para votes, o qual o elemento do índice será somado e usado para
calcular a média xmed
        ENDDO
        medias(k) = xmed/M  ! adiciono meus valores de xmed à um outro
vetor para ser usado posteriormente
        X = X + medias(k)           ! X será o valor médio da amostragem,
conforme pedido em (c)
ENDDO
X = X/100       ! X é dividido pelo valor de k (100)

DO k = 1, 100
        sm = sm + (medias(k)-X)**2 ! faço novamente k de 1 à 100, desta
vez para calcular o desvio padrão não enviesado Sm
END DO
WRITE(*,*)'Xm = ', X
WRITE(*,*)'sm = ',(sm/99)**0.5      ! por fim, escrevo o valor médio
encontrado de X e seu respectivo desvio padrão

END PROGRAM
```

Tendo todo o espaço amostral, representado por votes.dat, sorteio 100 amostras aleatórias, de tamanho M <= 10^4 também escolhido aleatóriamente. Seleciono índices aleatórios que representam cada “pessoa” do vetor votes.

Calculo então a média dos votos das pessoas selecionadas e salvo-a em um outro vetor “medias”. Também realizo o cálculo da média dos elementos desse último vetor e a utilizo para calcular o desvio padrão não enviesado Sm.

(d)

```
PROGRAM prog
 IMPLICIT NONE

INTEGER*8   ::      k, i
REAL*8      ::      rkiss05
INTEGER     :: M
REAL*8, DIMENSION(100):: medias
INTEGER*8   ::      index
REAL*8              ::      xmed
INTEGER, DIMENSION(10**6)    ::      votes
REAL*8              :: X = 0
REAL*8              ::      sm = 0
REAL*8              ::      var = 0

CALL kissinit(42)
OPEN(UNIT=1, FILE="votes.dat", ACTION='READ')
READ(UNIT=1, FMT=*) votes       ! leio o arquivo votes.dat
CLOSE(UNIT=1)


DO i = 1, 10**6
        xmed = xmed + votes(i)
END DO
xmed = xmed/(10**6)
DO i = 1, 10**6
        var = var + (votes(i)-xmed)**2
END DO
var = ((var/(10**6))**0.5)     ! calculo a variância do meu conjunto
```

Nessa primeira parte, simplesmente defino as variáveis que serão usadas e calculo a variância da minha amostra. Ela é simplesmente tomando a raiz quadrada do desvio padrão de votes.

```
        DO M = 10, 10**4        ! vario M de 10 até 10^4, representando o tamanho do meu subconjunto
            sm = 0
            X = 0
            DO k = 1, 100   ! por 100 vezes, pego uma amostra aleatória e calculo seu desvio padrão não enviesado como descrito no exercício anterior
                xmed = 0
                DO i = 1, M
                    index = int(rkiss05()*10**6)
                    xmed = xmed + votes(index)
                ENDDO
                medias(k) = xmed/M
                X = X + medias(k)
            ENDDO
            X = X/100

            DO k = 1, 100
                sm = sm + (medias(k)-X)**2
            END DO
            sm = (sm/99)**0.5       ! por fim, para cada subconjunto de tamanho M, tendo pego 100 amostras para cada, terei um desvio padrão enviesado para cada M
            WRITE(1,*) M, ',', sm, ',', var/SQRT(FLOAT(M))          ! Salvo os valores do tamanho M, de Sm e de sigma/(M^0.5), sendo sigma a variância de todo conjunto, para posterior análise
        ENDDO
        WRITE(*,*) sm

    END PROGRAM
```

Após isso, tomo amostras de tamanho M, com M variando de 10 até 10^4, e como em (c), calculo seu desvio padrão não enviesado. Dessa vez, salvando o valor da amostra M, do desvio não enviesado Sm e da variância dividida pela raiz quadrada de M, para fazer uma análise de como as duas últimas se comportam em relação uma a outra ao variar M.

(e)

```
        PROGRAM prog
        IMPLICIT NONE

        INTEGER*8   ::      k, i
        REAL*8      ::      rkiss05
        INTEGER     :: M
        REAL*8, DIMENSION(100)    :: medias
        INTEGER*8   ::      index
        REAL*8              ::      xmed
        INTEGER, DIMENSION(10**6)       ::      votes
        REAL*8              :: X = 0
        REAL*8              ::      sm = 0
        LOGICAL             :: cont = .TRUE.
        REAL*8              ::      var = 0, erro
```

```
                CALL kissinit(42)
                OPEN(UNIT=1, FILE="votes.dat", ACTION='READ')
                READ(UNIT=1, FMT=*) votes
                CLOSE(UNIT=1)

                DO i = 1, 10**6
                        xmed = xmed + votes(i)
                END DO
                xmed = xmed/(10**6)
                DO i = 1, 10**6
                        var = var + (votes(i)-xmed)**2
                END DO
                var = ((var/(10**6))**0.5)
!!! EXATAMENTE IGUAL O ÚLTIMO PROGRAMA ATÉ AQUI !!!

                DO M = 10, 10**4
                        erro = var/SQRT(FLOAT(M)) ! O erro agora é definido pela variância de
todos os votos dividida pela raíz quadrada do tamanho M da amostra
                        IF (erro .LE. 0.1) THEN! Caso o erro seja menor ou igual a 10%, isto é,
caso a chance de acertar seja maior ou igual à 90% o loop é interrompido
                                WRITE(*,*) M ! Retorna o tamanho M da amostra necessária
para se ter 90% de chance de acertar o resultado
                                EXIT
                        ENDIF
                ENDDO

    END PROGRAM
```

Conhecendo agora a relação entre a variância do meu conjunto e o desvio padrão não enviesado de cada amostra dele (que pôde ser observada no último caso), agora determino, tendo uma única amostra, qual será o seu tamanho mínimo M necessário para que se tenha 90% de chance de acertar o resultado.

Uma vez que tenho o uma aproximação do desvio padrão não enviesado para M, representado pela variância do conjunto todo dividida pela raíz quadrada de M, este será o erro da minha média. Caso este erro seja igual ou menor que 10% (ou 0.1), a probabilidade de se saber o resultado final para todo o conjunto será de 90% ou mais.

2.4 Organize uma lista

```
PROGRAM list
IMPLICIT NONE

LOGICAL      :: troca = .True.
INTEGER      :: i, n, m
REAL*8       :: aux
REAL, DIMENSION(10**4):: lista  ! a variável lista representa todo o arquivo dos Rnumbers
REAL, DIMENSION(:), ALLOCATABLE   :: vetor

OPEN(UNIT=2,FILE="Rnumbers.dat",ACTION="READ")
READ(UNIT = 2, FMT = *) lista
CLOSE(UNIT = 2)
WRITE(*,*) 'DIGITE O VALOR DE N: '
READ(*,*) n
ALLOCATE(vetor(n))          ! armazeno até o termo n da lista dentro do meu vetor, tendo sido n entrado no terminal anteriormente
vetor(:) = lista(:n)
!======================BUBBLE=SORT==================
DO WHILE(troca)
        troca = .False.
        DO i = 1, n -1
                IF(vetor(i).GT.vetor(i+1)) THEN
                        aux = vetor(i)
                        vetor(i) = vetor(i+1)
                        vetor(i+1) = aux
                        troca = .True.
                ENDIF
        ENDDO
ENDDO
!======FONTE = https://pt.wikipedia.org/wiki/Bubble_sort ==========
WRITE(*,*)'DIGITE M: '
READ(*,*) m ! leio o valor de m desejado
OPEN(UNIT=1,FILE="menores.dat",ACTION="WRITE")  ! crio o arquivo em que vou salvar os resultados
DO i =1,m
        WRITE(UNIT=1,FMT = *) vetor(i)   ! percorro os m primeiros termos do vetor já ordenado de ordem crescente e os salvo no arquivo
ENDDO
CLOSE(UNIT=1)

END PROGRAM
```

Para organizarmos a lista, primeiarmente armazeno-a em um array “lista”. Este array contém todos os elementos da lista. Logo após, é pedido o inteiro “N”, que será usado para definir o tamanho de um outro array “vetor”, que conterá os elementos de “lista” de 1 até N.

Para que possamos retornar os “M menores elementos de N”, organizo o array utilizando um algoritmo de Bubble Sort, que consiste em percorrer o vetor diversas vezes, levando os maiores termos para o final e por consequência deixando-a em ordem crescente. O algoritmo foi traduzido para o Fortran do descrito na página https://pt.wikipedia.org/wiki/Bubble_sort .

Agora, com “vetor” organizado em ordem crescente, pede-se o valor de M e então escrevemos os M primeiros termos de “vetor” num arquivo chamado “menores.dat”.

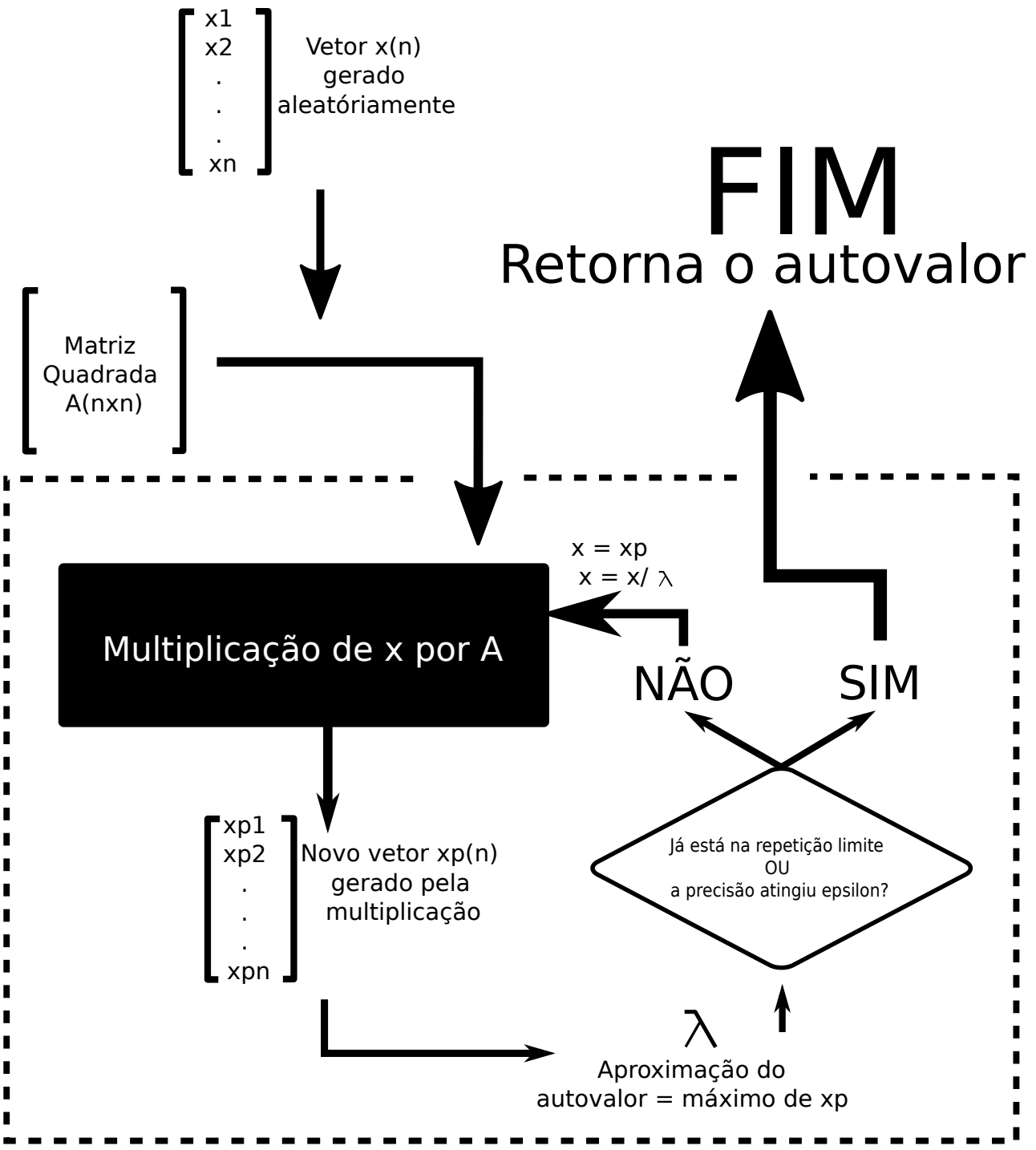
2.5 Método da potência para o cálculo do autovalor/autovetor dominante

```
PROGRAM domeig
IMPLICIT NONE

REAL*8, DIMENSION(:,:), ALLOCATABLE     ::      A
REAL*8, DIMENSION(:), ALLOCATABLE       ::      x, xp
REAL*8      ::      rkiss05
REAL*8      ::      lamb, term
INTEGER*8           ::      n, i, j, k, contador = 0
REAL*8      ::      rand
LOGICAL     ::      cont= .True.
!!! DECLARAÇÃO DAS VARIÁVEIS !!!

CALL kissinit(17)
WRITE(*,*)'DIMENSÃO DA MATRIZ:'
READ(*,*) n
ALLOCATE(A(n,n))
ALLOCATE(x(n))
ALLOCATE(xp(n))    ! Leio a dimensão n da matriz que se deseja calcular o autovalor e aloco a matriz A(n,n), assim como os vetores x(n) e xp(n), que serão usados para multiplicar a matriz

WRITE(*,*)'ENTRE NUMERO MÁXIMO DE ITERAÇÕES'
READ(*,*) k             ! É digitado o número máximo de loops de múltiplicação que serão realizados antes de finalizar o programa
WRITE(*,*)'ENTRE COM A MATRIZ'
DO i = 1, n
        rand  = rkiss05()
        x(i) = rand                     ! Aproveito o loop de 1 até n, usado para digitar a matriz, para gerar o vetor aleatório x
        READ(*,*) A(i,:)        ! Lê a matriz no terminal, linha a linha
END DO
WRITE(*,*) 'CALCULANDO'
```

```
                DO WHILE(cont)
                        contador = contador +1          ! O contador será usado para ser comparado com o número máximo de loops, "K", lido anteriormente. Ele será incrementado em 1 a cada loop
                        DO i= 1, n
                                term = 0.0d0    ! 'term' representa cada linha resultante da multiplicação entre A e x, para cada uma das i linhas
                                DO j= 1, n
                                        term = term + x(j)*A(i,j)       ! Faço a operação descrita acima
                                END DO
                                xp(i) = term    ! Defino a linha "i" do vetor xp auxiliar como o termo 'term'. Ainda não posso alterar o valor do vetor original x pois ele está sendo usado na múltiplicação, então xp está simplesmente sendo usado para armazenar todos os valoes que serão substituídos de uma vez após o término da operação
                                END DO
                                x(:) = xp(:)    ! Após a multiplicação substituo os valores de x pelos armazenados em xp
                        IF (ABS(lamb-MAXVAL(x)).LE.10**(-5) .OR. contador.GE.k) THEN ! Caso a diferença entre a última aproximação do autovalor dominante e a nova seja menor que um valor sigma (no caso sigma = 10**-5) OU caso o número máximo de loops 'k' tenha sido atingido, o programa é interrompido
                                cont = .False.
                        END IF

                        lamb = MAXVAL(x)  ! o autovalor dominante lambda é igual ao maior número do vetor x
                        x(:)  = x(:)/lamb       ! divido x pelo autovalor para ajustá-lo para o próximo ciclo
                        WRITE(1,*)contador,',',lamb ! Salvo em um arquivo para posterior representação gráfica
                END DO

                WRITE(*,*) lamb     ! Retorna no terminal o autovalor dominante

                END PROGRAM
```

Em síntese, o programa recebe a dimensão de uma matriz quadrada qualquer e então a matriz pode ser entrada no terminal. Um vetor aeatório x é então gerado e multiplicações sucessivas do vetor com a matriz são realizadas, gerando assim novos vetores e o processo se repetirá usando eles. O autovalor dominante aproximado, que desejamos calcular, será igual ao maior elemento desse vetor gerado. Como fator de correção, dividimos cada cordenada do vetor pelo seu maior valor. Isso pode ser tomado como um ‘ajuste’ do seu comprimento.

O processo descrito acima é ilustrado no seguinte esquema:



Nesse algoritmo, o valor para o primeiro vetor X0 foi selecionado aleatoriamente. Isso pode ser um problema, uma vez que não garante que o termo C1 será diferente de zero.

3. Resultados e discussões

3.1 Fatoriais e a aproximação de Stirling

Calculando os fatoriais dos números de 1 até 20, obteve-se a tabela:

1 ! = 1

2 ! = 2

3 ! = 6

4 ! = 24

5 ! = 120

6 ! = 720

7 ! = 5040

8 ! = 40320

9 ! = 362880

10 ! = 3628800

11 ! = 39916800

12 ! = 479001600

13 ! = 6227020800

14 ! = 87178291200

15 ! = 1307674368000

16 ! = 20922789888000

17 ! = 355687428096000

18 ! = 6402373705728000

19 ! = 121645100408832000

20 ! = 2432902008176640000

Portanto os resultados foram exatos.

Em seguida, foi caculado o logarítmo natural dos fatoriais, obtendo-se então a seguinte tabela:

LN(FAT(2)) = 0.69314718055994529

LN(FAT(3)) = 1.7917594692280550

LN(FAT(4)) = 3.1780538303479458

LN(FAT(5)) = 4.7874917427820458

LN(FAT(6)) = 6.5792512120101012

LN(FAT(7)) = 8.5251613610654147

LN(FAT(8)) = 10.604602902745251

LN(FAT(9)) = 12.801827480081469

LN(FAT(10)) = 15.104412573075516

LN(FAT(11)) = 17.502307845873887

LN(FAT(12)) = 19.987214495661885

LN(FAT(13)) = 22.552163853123425

LN(FAT(14)) = 25.191221182738680

LN(FAT(15)) = 27.899271383840890

LN(FAT(16)) = 30.671860106080672

LN(FAT(17)) = 33.505073450136891

LN(FAT(18)) = 36.395445208033053

LN(FAT(19)) = 39.339884187199495

LN(FAT(20)) = 42.335616460753485

LN(FAT(21)) = 45.380138898476908

LN(FAT(22)) = 48.471181351835227

LN(FAT(23)) = 51.606675567764377

LN(FAT(24)) = 54.784729398112319

LN(FAT(25)) = 58.003605222980518

LN(FAT(26)) = 61.261701761002001

LN(FAT(27)) = 64.557538627006338

LN(FAT(28)) = 67.889743137181540

LN(FAT(29)) = 71.257038967168015

LN(FAT(30)) = 74.658236348830158

O que também expressa os resultados esperados com uma precisão maior que 10^-10.

Após isso, foi feita a aproximação de Stirling para todos os elementos de 2 à 30, calculada a diferença entre o valor obtido diretamente e com a aproximação, e obtida a seguinte tabela:

2	0.69314718055994529	0.65180648460456891	5.9642017041719952E-002
3	1.7917594692280550	1.7640815435430897	1.5447344445674867E-002
4	3.1780538303479458	3.1572631582442137	6.5419508962363026E-003
5	4.7874917427820458	4.7708470515922583	3.4767038950787501E-003
6	6.5792512120101012	6.5653750831870630	2.1090741751444032E-003
7	8.5251613610654147	8.5132646511195560	1.3954820843850328E-003
8	10.604602902745251	10.594191637483311	9.8176851669232114E-004
9	12.801827480081469	12.792572017898790	7.2297976184107059E-004
10	15.104412573075516	15.096082009642185	5.5153177212466874E-004
11	17.502307845873887	17.494734170385968	4.3272439009827633E-004
12	19.987214495661885	19.980271655554709	3.4736406659786217E-004
13	22.552163853123425	22.545754858935453	2.8418533271175209E-004
14	25.191221182738680	25.185269812625954	2.3624778130269678E-004
15	27.899271383840890	27.893716650288962	1.9909959208279159E-004
16	30.671860106080672	30.666652450161095	1.6978611344622366E-004
17	33.505073450136891	33.500172054188489	1.4628817201949899E-004
18	36.395445208033053	36.390816054283754	1.2719046910513693E-004
19	39.339884187199495	39.335498626950297	1.1147872800865320E-004
20	42.335616460753485	42.331450141061524	9.8411693043935015E-005
21	45.380138898476908	45.376170944258298	8.7438124142517079E-005
22	48.471181351835227	48.467393733766812	7.8141649590067740E-005
23	51.606675567764377	51.603052607539723	7.0203325147209076E-005
24	54.784729398112319	54.781257376729371	6.3375714749216853E-005
25	58.003605222980518	58.000272067343822	5.7464628687898789E-005
26	61.261701761002001	61.258496790773982	5.2316049601806656E-005
27	64.557538627006338	64.554452348323750	4.7806634952624593E-005
28	67.889743137181540	67.886767073198016	4.3836724753926522E-005
29	71.257038967168015	71.254165517805689	4.0325130035845425E-005
30	74.658236348830158	74.655458673900441	3.7205204215450987E-005

Onde a primeira coluna é N, a segunda e Ln(n!), a terceira a aproximação de Stirling do valor (S(n)) e a quarta, a diferença a diferença [Ln(n!) - S(n)]/Ln(n!).

3.2 Série de Taylor para o cosseno

O valor original para o cosceno de x, o valor obtido através do polinômio e a ordem de expansão necessária para se obter a precisão desejada estão representados na tabela:

COS(X)= 0.99802672842827134 T(X)= 0.99802672851372209 ordem= 5

COS(X)= 0.99211470131447677 T(X)= 0.99211470131293478 ordem= 7

COS(X)= 0.98228725072868639 T(X)= 0.98228725068917544 ordem= 7

COS(X)= 0.96858316112862697 T(X)= 0.96858316073408468 ordem= 7

COS(X)= 0.95105651629514720 T(X)= 0.95105651629772592 ordem= 9

COS(X)= 0.92977648588824224 T(X)= 0.92977648590420403 ordem= 9

COS(X)= 0.90482705246600714 T(X)= 0.90482705254054552 ordem= 9

COS(X)= 0.87630668004384760 T(X)= 0.87630668032705505 ordem= 9

COS(X)= 0.84432792550199520 T(X)= 0.84432792642119348 ordem= 9

COS(X)= 0.80901699437492314 T(X)= 0.80901699700966234 ordem= 9

COS(X)= 0.77051324277576017 T(X)= 0.77051324275102029 ordem= 11

COS(X)= 0.72896862742137758 T(X)= 0.72896862735112833 ordem= 11

COS(X)= 0.68454710592864954 T(X)= 0.68454710574518529 ordem= 11

COS(X)= 0.63742398974864511 T(X)= 0.63742398930246158 ordem= 11

COS(X)= 0.58778525229242284 T(X)= 0.58778525127195425 ordem= 11

COS(X)= 0.53582679497894070 T(X)= 0.53582679276659462 ordem= 11

COS(X)= 0.48175367410165365 T(X)= 0.48175366952560755 ordem= 11

COS(X)= 0.42577929156500532 T(X)= 0.42577928248586311 ordem= 11

COS(X)= 0.36812455268460481 T(X)= 0.36812455282077616 ordem= 13

COS(X)= 0.30901699437486868 T(X)= 0.30901699465391580 ordem= 13

COS(X)= 0.24868988716477067 T(X)= 0.24868988771689388 ordem= 13

COS(X)= 0.18738131458563509 T(X)= 0.18738131564385888 ordem= 13

COS(X)= 0.12533323356420975 T(X)= 0.12533323553449779 ordem= 13

COS(X)= 6.2790519529214245E-002 T(X)= 6.2790523101641404E-002 ordem= 13

COS(X)= -1.0341155355510722E-013 T(X)= 6.3213660393019565E-009 ordem= 13

COS(X)= -6.2790519529420663E-002 T(X)= -6.2790508591804781E-002 ordem= 13

COS(X)= -0.12533323356441517 T(X)= -0.12533323378726166 ordem= 15

COS(X)= -0.18738131458583845 T(X)= -0.18738131498431937 ordem= 15

COS(X)= -0.24868988716497098 T(X)= -0.24868988786309029 ordem= 15

COS(X)= -0.30901699437506563 T(X)= -0.30901699557504431 ordem= 15

COS(X)= -0.36812455268479732 T(X)= -0.36812455471099237 ordem= 15

COS(X)= -0.42577929156519251 T(X)= -0.42577929492987049 ordem= 15

COS(X)= -0.48175367410183489 T(X)= -0.48175367960238019 ordem= 15

COS(X)= -0.53582679497911534 T(X)= -0.53582680383992221 ordem= 15

COS(X)= -0.58778525229259015 T(X)= -0.58778526636979112 ordem= 15

COS(X)= -0.63742398974880432 T(X)= -0.63742398937853351 ordem= 17

COS(X)= -0.68454710592880041 T(X)= -0.68454710532293117 ordem= 17

COS(X)= -0.72896862742151924 T(X)= -0.72896862644311666 ordem= 17

COS(X)= -0.77051324277589217 T(X)= -0.77051324121550724 ordem= 17

COS(X)= -0.80901699437504471 T(X)= -0.80901699191583287 ordem= 17

COS(X)= -0.84432792550210600 T(X)= -0.84432792166976567 ordem= 17

COS(X)= -0.87630668004394729 T(X)= -0.87630667413552321 ordem= 17

COS(X)= -0.90482705246609518 T(X)= -0.90482704344960041 ordem= 17

COS(X)= -0.92977648588831852 T(X)= -0.92977647226238946 ordem= 17

COS(X)= -0.95105651629521115 T(X)= -0.95105649589442431 ordem= 17

COS(X)= -0.96858316112867848 T(X)= -0.96858316179675252 ordem= 19

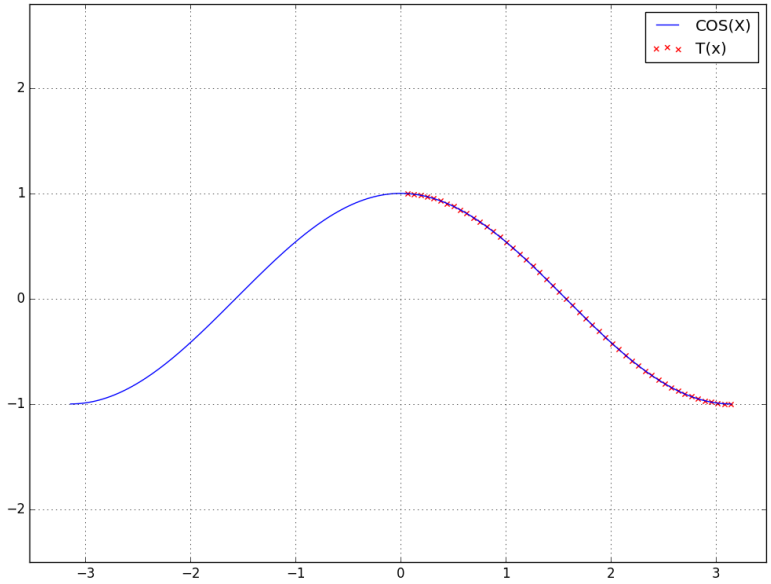
COS(X)= -0.98228725072872514 T(X)= -0.98228725175504816 ordem= 19

COS(X)= -0.99211470131450274 T(X)= -0.99211470287694381 ordem= 19

COS(X)= -0.99802672842828433 T(X)= -0.99802673078629234 ordem= 19

COS(X)= -1.0000000000000000 T(X)= -1.0000000035290801 ordem= 19

Pode-se ver que a ordem de expansão necessária para obter a mesma precisão aumenta conforme x de distancia do entorno em que o polinômio está sendo calculado, porém por conta do algoritmo usado, não é desperdiçada memória expandindo o polinômio até nenhuma ordem a mais que a necessária. Os resultados podem ser visualizados gráficamente:



3.3 Valores médios e desvio padrão

Calculando a média dos valores em votes.dat, e utilizando-a para determinar se 1 ou 0 ganha, obtemos:

```
 0.52387600000000001
O vencedor é 1
```

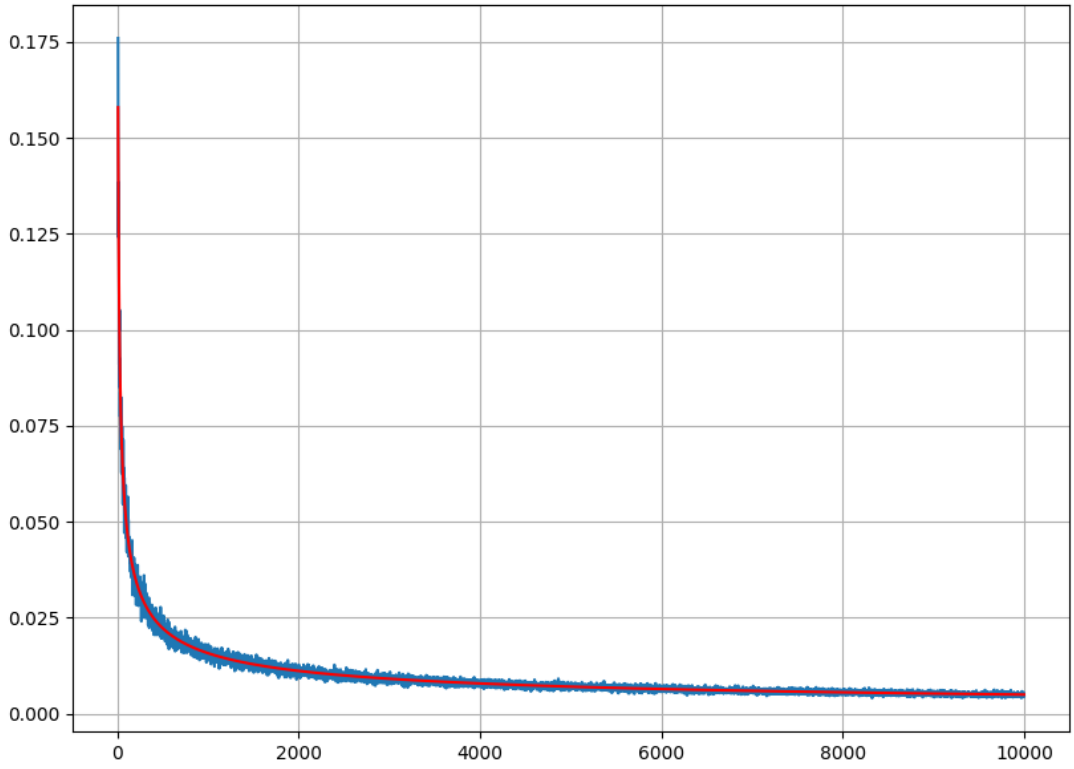
Como 0.52 > 0.5, isto é, existem mais votos 1 do que 0, o vencedor é 1.

Após isso, tomamos k = 100 amostras com M <= N elementos e calculamos através delas a média Xm e o desvio padrão não enviesado Sm:

```
Xm =   0.52736647118908464
sm =    1.6016874382156283E-002
```

Podemos ver que, considerando o desvio padrão, o valor da média calculada dessa forma está de acordo com a calculada anteriormente.

Agora, usando esse método e variando M de 10 à 10^4, para comparar a relação entre o desvio padrão não enviesado e a variância do conjunto todo, conseguimos o seguinte gráfico, onde o eixo x representa M, em vermelho está a variância dividida pela raiz quadrada de M e em azul o valor do desvio padrão não enviesado, com seus respectivos valores estando no eixo y:



Vemos que eles estão relacionados tendem a se aproximar ainda mais ao aumentarmos o tamanho da amostra M.

Agora calculamos, usando a variância e a sua relação com o desvio padrão não enviesado, qual deve ser o menor valor de M para o qual temos 90% de chance de acertar o resultado, assim obtendo:

Isto é, conhecendo a variância de meu espaço amostral, basta uma amostra de tamanho M = 25 para ter 90% de chance de acertar qual será o resultado.

3.4 Organize uma lista

Na execução do programa, selecionando N = 1000 e M = 20, obteu-se o seguinte resultado no arquivo menores.dat:

4.31183307E-03

4.36148653E-03

5.01992926E-03

5.11389412E-03

5.46672521E-03

5.59070660E-03

6.33674162E-03

8.26599821E-03

8.57812073E-03

8.79637431E-03

9.03421827E-03

1.01734949E-02

1.09684533E-02

1.10858670E-02

1.13666849E-02

1.16068590E-02

1.18134888E-02

1.32239070E-02

1.49526531E-02

1.67539772E-02

Pode-se ver que os números estão em ordem crescente, de acordo com o esperado.

3.5 Método da potência para o cálculo do autovalor/autovetor dominante

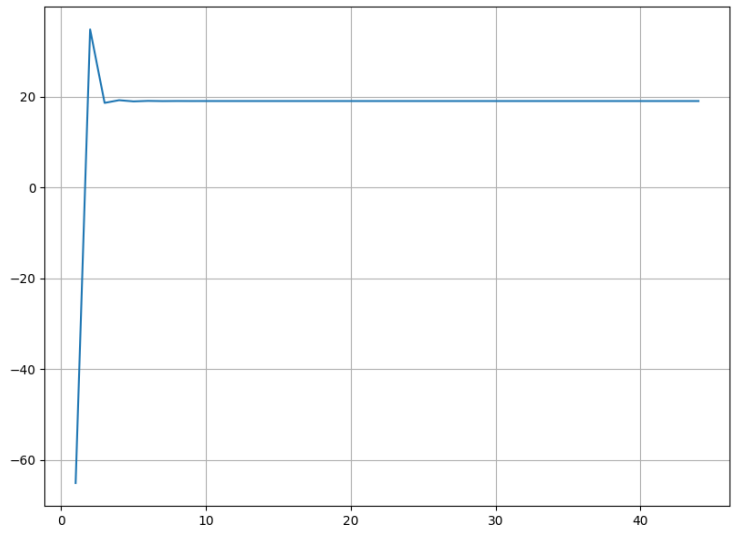
É utilizado o método da potência, através do algoritmo descrito anteriormente, para as seguintes três matrizes:

$$\mathbb{A}_1 = \begin{pmatrix} 2 & 8 & 10 \\ 8 & 0 & 5 \\ 10 & 5 & 7 \end{pmatrix}, \ \mathbb{A}_2 = \begin{pmatrix} 10 & -2 & 3 & 2 \\ -2 & 0 & -3 & 4 \\ 3 & -3 & 0 & 3 \\ 2 & 4 & 3 & 6 \end{pmatrix} \text{ e } \mathbb{A}_3 = \begin{pmatrix} -10 & 2 & -3 & -2 & -1 \\ 2 & -10 & 3 & -4 & -2 \\ -3 & 3 & -6 & -3 & -3 \\ -2 & -4 & -3 & -6 & -4 \\ -1 & -2 & -3 & -4 & 0 \end{pmatrix}.$$

Usa-se um número máximo de iterações k = 1000 e uma precisão epsilon de 10^-5 para as três matrizes. Para a primeira, obteve-se:

```
 DIMENSÃO DA MATRIZ:
3
 ENTRE NUMERO MÁXIMO DE ITERAÇÕES
1000
 ENTRE COM A MATRIZ
2 8 10
8 0 5
10 5 7
 CALCULANDO
   19.029959170703965
```

Onde 19.02995 é o autovalor dominante calculado. A mudança do autovalor aproximado (eixo y) ao longo das iterações (eixo x) podem ser vistos em:

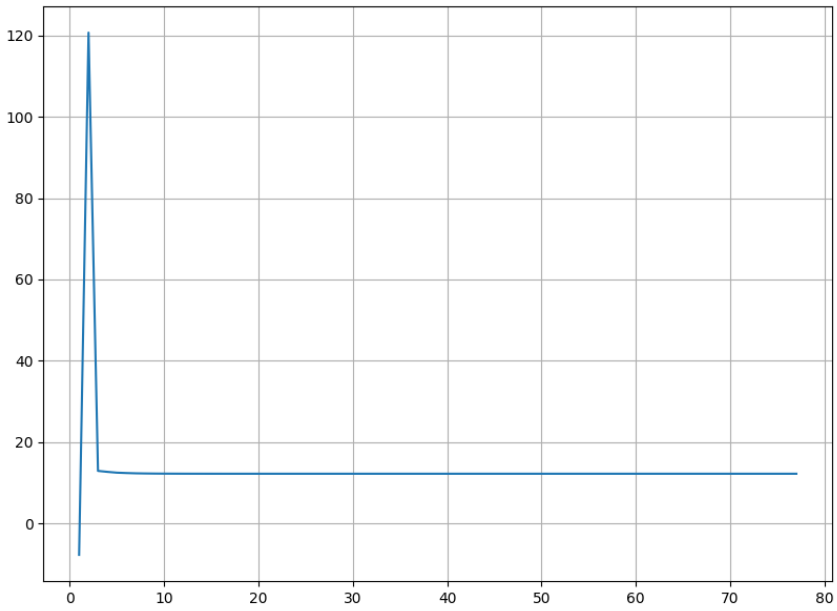


Também pode-se ver claramente que o gráfico é interrompido em x antes de k = 1000, portanto a precisão desejada foi atingida.

Para a segunda matriz, obteve-se:

```
 DIMENSÃO DA MATRIZ:
4
 ENTRE NUMERO MÁXIMO DE ITERAÇÕES
1000
 ENTRE COM A MATRIZ
10 -2 3 2
-2 0 -3 4
3 -3 0 3
2 4 3 6
 CALCULANDO
   12.228021818599778
```

E seu respectivo gráfico:

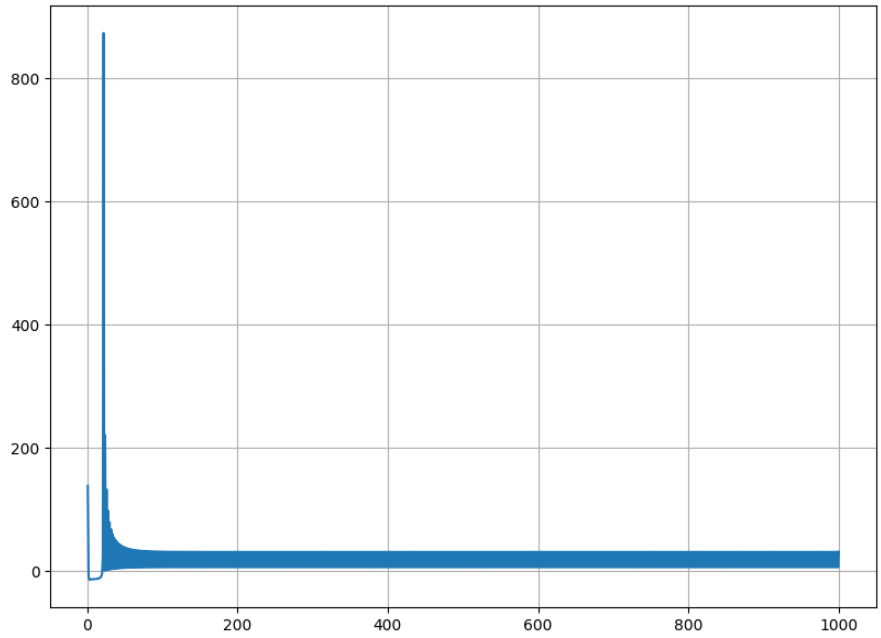


Novamente, ele indica que a precisão foi atingida.

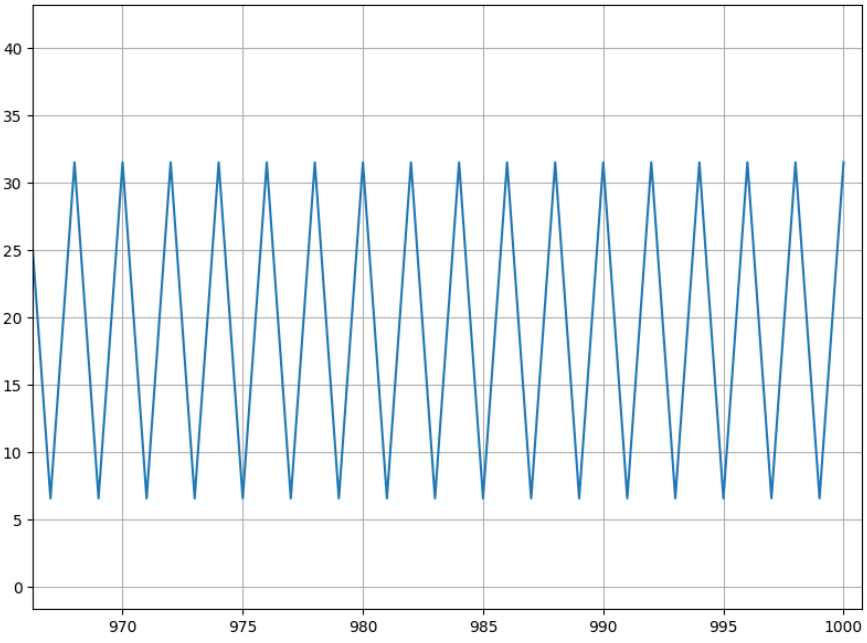
Já para o terceiro e último caso, obteve-se:

```
 DIMENSÃO DA MATRIZ:
5
 ENTRE NUMERO MÁXIMO DE ITERAÇÕES
1000
 ENTRE COM A MATRIZ
-10 2 -3 -2 -1
2 -10 3 -4 -2
-3 3 -6 -3 -3
-2 -4 -3 -6 -4
-1 -2 -3 -4 0
 CALCULANDO
   31.533691248108113
```

Porém, desta vez o gráfico obtido foi:



Ampliando:



Isso indica, além de que o número de iterações k = 1000 foi alcançado, que o autovalor dominante estava em um loop entre dois valores. Isso provavelmente indica algum erro no código ou no algorítmo em sí, o qual não pude identificar.

Embora possa haver algum problema com o vetor inícial X sorteado, o erro persiste também com outras seeds.