Map0201 - Métodos Matriciais Computacionais
Primeira Tarefa Computacional, entregar no 2o encontro da 5a semana de aula

Assunto: normas vetoriais, BLAS, e linguagens de programação Fortran e C
Objetivo: Completar um programa em Fortran 90, ou C, para cálculo da norma p (inteiro) de um vetor x dado.
Tarefa: entregar ao seu Prof., por e-mail, código em Fortran 90, ou C, que implementa a versão completa da subrotina *normap(n,x,p,norma)* que implementa o cálculo da norma vetorial p de um vetor $x \in \mathbb{R}^n$ dado:

- $p = 0 : |x|_\infty$ já feito em aula;
- $p = 1 : |x|_1$ via função dasum da biblioteca BLAS (ou CBLAS), feito em aula;
- $p = 2 : |x|_2$ via função dnrm2 da biblioteca BLAS (ou CBLAS);
- $p > 2 : |x|_p$ via cálculo próprio;

Recursos: desenvolvidos em aula
arquivo trabalho1.f90 (para seus testes somente. Não envie este)

```
program map0201_tc1
!  MAP0201 - Trabalho Computacional 1
   integer,parameter :: nmax=2000
   real (kind=8) :: x(nmax),norma,dnrm2,dasum
   integer :: i,n,p
   external dnrm2,dasum
! em computadores do dominio LICC, exceto hunter::
! gfortran trabalho1.f90 rotinas.f90 -lblas -o trabalho1
! ifort trabalho1.f90 rotinas.f90 -lblas -o trabalho1
!
   write(*,'(a)') 'MAP0201 - Calculo da norma p de um vetor.'
   write(*,'(a)',advance='no') 'Use p=0 para norma inf. Qual o valor de p ?'
   read(*,*) p
   write(*,'(a)',advance='no') 'Qual o valor de n, o numero de componentes ?'
   read(*,*) n
   write(*,'(a)') 'Qual o valor das componentes ? Tecle ENTER apos cada uma!'
   do i=1,n
      read(*,*) x(i)
   end do
   call normap(n,x,p,norma);
   write(*,'(a,f16.6)') 'O valor da norma calculada eh:', norma
end program
```

arquivo rotinas.f90 (é o que deve ser completado e enviado)

```
subroutine normap(n,x,p,norma)
! calcular a norma p de um vetor x(*) dado
  real (kind=8) x(*),norma
  integer :: i,n,p
  real (kind=8) dnrm2,dasum
  external dnrm2,dasum
  norma=.0
```

```
  if (p==0) then     ! norma do maximo
    do i=1,n
       norma=max( norma, abs(x(i)) )
    end do
  elseif (p==1) then ! chamaremos a funcao dasum da BLAS
     norma=dasum(n,x,1)
  elseif (p==2) then
  !   chamaremos a funcao dnrm2 da BLAS
  !   este caso ficarah como tarefa computacional
  else       ! p eh inteiro e maior do que 2
  ! este caso ficarah como tarefa computacional
  end if
end
```

Seu trabalho será avaliado também em relação a documentação (comentários do código), eficiência e eficácia numérica (tratamento contra overflow/underflow evitável). Veja o hipertexto na página `http://www.mat.ufrgs.br/~carvalho/mat01099/benfeitorias`. Caso você queira implementar usando a linguagem ANSI C e a biblioteca CBLAS, USE o modelo abaixo para editar um arquivo que aqui chamaremos de *trabalho1.c*:

```
#include <stdio.h>
#include <cblas.h>
#include <math.h>
#define nmax 2000
//                    nao modifique o main()
void normap(int n,double x[],int p, double *pnorma);
main() {
  //   MAP0201 - primeira atividade computacional
  double x[nmax],norma;
  int i,n,p;
  //
  // compilar este programa nas maquinas do LICC
  // gcc trabalho1.c -lcblas -lm -o trabalho1
  //
  printf("MAP0201 - Calculo da norma p de um vetor.\n");
  printf("Use p=0 para norma inf. Qual o valor de p ? ");
  scanf("%i",&p);
  printf("Qual o valor de n, o nr de componentes ? ");
  scanf("%i",&n);
  printf("Qual o valor das componentes ? Tecle ENTER apos cada uma !!!\n");
  for (i=0;i<n;i++){ scanf("%lf",&x[i]); }
  normap(n,x,p,&norma);
  printf("O valor da norma eh: %f\n",norma);
}

#define max(a,b)({typeof(a) _a=(a);typeof (b) _b=(b);_a>_b ? _a : _b; })
#define min(a,b)({typeof(a) _a=(a);typeof (b) _b=(b);_a<_b ? _a : _b; })
//
//    modifique somente daqui para baixo, com usas melhorias
//
```

```
void normap(int n,double x[],int p, double *pnorma){
// Autor : JB Carvalho  , Ago 2014
// Map0201 - calcular a norma p de um vetor x(*) dado
  int i;
  double norma;
  norma=.0;
  // gcc trabalho1.c -lcblas -o trabalho1
  if (p==0) {    // norma do maximo
    for (i=0;i<n;i++){ norma=max( norma, fabs(x[i]) ); }
  }
  else if (p==1){ // chamaremos a funcao dasum da cblas
    norma = cblas_dasum(n,x,1);
  }
  else if (p==2){ // chamaremos dnrm2 da cblas
    // este caso ficarah como tarefa computacional
  }
  else {    // p eh inteiro e maior do que 2
    // este caso ficarah como tarefa computacional
  }
  *pnorma=norma;
}
```

JBC, Ago 2014

Map0201 - Métodos Matriciais Computacionais
Segunda Tarefa Computacional, entregar no 2o encontro da 7a semana

Assunto: normas matriciais, e linguagens de programação Fortran e C
Objetivo: Completar um programa em Fortran 90, ou C, para cálculo da norma matricial de uma matriz esparsa armazenada em arquivo, no formato de lista coordenada (COO). Leia sobre isso na página de Wikipedia `http://en.wikipedia.org/wiki/Sparse_matrix`, na seção apropriada.
Descrição: Uma estratégia muito comum para armazenar matrizes que têm dimensão muito grande mas são esparsas (têm uma quantidade muito grande de zeros), é usar um arquivo de texto puro (ASCII) que contenha uma lista coordenada (COO). Nesse formato, uma matriz M é representada por meio de uma lista (i, j, m_{ij}) dos s_m elementos que não são nulos. A nível de implementação, podem ser usadas duas estruturas de dados do tipo array de números inteiros, $im(\cdot)$ e $jm(\cdot)$, para armazenar os índices, e também uma estrutura de dados do tipo array de números reais em precisão dupla, $zm(\cdot)$, para armazenar os respectivos elementos m_{ij}. Ao lado, vemos o conteúdo de um arquivo que armazena a matriz

$$M = \begin{bmatrix} -5 & -1 & & & 0.5 & \\ 4 & -4 & & & & \\ & & 4.5 & & 1.5 & \\ & & & 5.5 & & \\ 2 & & & & 2.5 & \\ 1 & & & 6.5 & & 3.5 \end{bmatrix}.$$

Os elementos nulos dessa matriz são todos os que não estão sendo mostrados.

```
%% Matriz M formato coordenado
6 6 13
1  1  -5.0
2  1   4.0
5  1   2.0
6  1   1.0
1  2  -1.0
2  2  -4.0
3  3   4.5
4  4   5.5
6  4   6.5
1  5   0.5
3  5   1.5
5  5   2.5
6  6   3.5
```

Tarefa: entregar ao seu Prof., por e-mail, código em Fortran 90, ou C, que implementa a versão completa da subrotina (F90) ou função (ANSI C) *norma_mat_coo()* que implementa o cálculo da norma matricial p de uma matriz $M \in \mathbb{R}^{m \times n}$ representada por suas três estruturas de dados $im(\cdot)$, $jm(\cdot)$ e $zm(\cdot)$, bem como por seus parâmetros inteiros m, n e sm, e segundo o valor do inteiro p:

- $p = 0 : \|M\|_\infty$;
- $p = 1 : \|M\|_1$;
- $p = 2 : \|M|_F$;
- $p = 3 : \|M|_\triangle$: maior das magnitudes ;

Recursos disponibilizados:

- em `http://www.mat.ufrgs.br/~carvalho/map0201/recursos/trabalho2.f90` você pode encontrar uma implementação inicial em Fortran 90;
- em `http://www.mat.ufrgs.br/~carvalho/map0201/recursos/trabalho2.c` você pode encontrar uma implementação inicial em ANSI C;
- Arquivo que armazena a matriz M do exemplo acima aqui.

□

Map0201 - Métodos Matriciais Computacionais
Terceira Tarefa Computacional, entregar no 2o encontro da 10a semana (=>11a)

Assunto: fatoração LU atualizada, solução numérica de sistemas não-lineares, ambientes *Scilab* e *Matlab*.
Objetivo: solução computacional eficiente de um sistema quasi-linear de equações algébricas, via fatoração LU, em ambiente *Scilab* ou *Matlab*.
Descrição do problema: em um intervalo $[0, L]$ dado, onde $L > 0$, e para um inteiro n dado, queremos encontrar valores para $a_0, a_1, a_2, \ldots, a_n$ tais que a representação ímpar:

$$y = \frac{x}{1 + a_0 x^2} + \sum_{j=1}^{n} a_j \text{sen}\left(\frac{\pi j x}{L}\right), x \in [0, L] \qquad (*)$$

interpola uma tabela $\{(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)\}$ dada, onde $y_i > 0$. Aplicando (*) em cada um desses pares, produzimos um sistema não-linear (mas quasi-linear) com $n + 1$ equações e $n + 1$ incógnitas:

$$F(a_1, a_2, \ldots, a_n, a_0) =$$

$$\begin{bmatrix} \text{sen}\left(\frac{\pi x_0}{L}\right) & \text{sen}\left(\frac{2\pi x_0}{L}\right) & \text{sen}\left(\frac{3\pi x_0}{L}\right) & \ldots & \text{sen}\left(\frac{n\pi x_0}{L}\right) \\ \text{sen}\left(\frac{\pi x_1}{L}\right) & \text{sen}\left(\frac{2\pi x_1}{L}\right) & \text{sen}\left(\frac{3\pi x_1}{L}\right) & \ldots & \text{sen}\left(\frac{n\pi x_1}{L}\right) \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \text{sen}\left(\frac{\pi x_2}{L}\right) & \text{sen}\left(\frac{2\pi x_2}{L}\right) & \text{sen}\left(\frac{3\pi x_2}{L}\right) & \ldots & \text{sen}\left(\frac{n\pi x_2}{L}\right) \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \text{sen}\left(\frac{\pi x_n}{L}\right) & \text{sen}\left(\frac{2\pi x_n}{L}\right) & \text{sen}\left(\frac{3\pi x_n}{L}\right) & \ldots & \text{sen}\left(\frac{n\pi x_n}{L}\right) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \ldots \\ a_n \end{bmatrix} + \begin{bmatrix} \frac{x_0}{1+a_0 x_0^2} \\ \frac{x_1}{1+a_0 x_1^2} \\ \frac{x_2}{1+a_0 x_2^2} \\ \ldots \\ \frac{x_n}{1+a_0 x_n^2} \end{bmatrix} - \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \ldots \\ y_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \ldots \\ 0 \end{bmatrix}$$

Metodologia: a solução $\mathbf{u}$ de um sistema não-linear $F(\mathbf{u}) = \vec{0}$ será encontrada pelo Método de Newton para sistemas, descrito abaixo, onde $\mathbf{u}_0$ é uma aproximação para a solução:
Para $k = 0, 1, 2, \ldots$, até convergência
Resolve $JF(\mathbf{u}_k)\,\mathbf{z}_k = F(\mathbf{u}_k)$
Atualiza $\mathbf{u}_{k+1} = \mathbf{u}_k - \mathbf{z}_k$
Fim-Para

Aqui $JF(\cdot)$ denota a matriz Jacobiana de $F(\cdot)$. Pela estrutura da $F(\cdot)$ acima, observe que somente a última coluna da jacobiana muda de uma iteração para outra, uma vez que F é linear em todas as variáveis exceto a_0, que consideramos ser sua última variável.
Seu trabalho: apresente código em *Scilab* ou *Matlab* para encontrar a solução do sistema quasi-linear acima e plotar gráfico com os pontos dados e com a curva calculada, seguindo as especificações:

1. fatoração PA=LU, usando o comando *lu*, deve ser feita somente uma vez;
2. encontre uma maneira econômica de atualizar essa fatoração de uma iteração para outra;
3. na solução dos sistemas lineares, deve ser usada a fatoração da matriz de coeficientes, via sistemas lineares triangulares que podem ser resolvidos usando o operador (comando) $\backslash$
4. use e plote (também) o gráfico de $y_0(x)$ que corresponde a aproximação inicial $\mathbf{u}_0$ dada por
$$a_0 = \frac{\sum x_i^2 \left(\frac{x_i}{y_i} - 1\right)}{\sum x_i^4}; a_1 = a_2 = \ldots = a_n = 0; \text{(estratégia de mínimos quadrados)}$$
5. use $\frac{norm(\mathbf{u}_{k+1} - \mathbf{u}_k)}{norm(\mathbf{u}_k)} \leq 10^{-5}$ como critério de parada da iteração;
6. use o modelo em `http://www.mat.ufrgs.br/~carvalho/map0201/recursos/trabalho3.sci`; ou adapte para *Matlab*;
7. Método de Newton em Scilab: veja script em `http://www.mat.ufrgs.br/~carvalho/mat01169/dist_curvas`.

□

Map0201 - Métodos Matriciais Computacionais
Quarta Tarefa Computacional, entregar no 2o encontro da 13a semana

Assunto: fatoração LU tridiagonal, equações diferenciais com condições de contorno, ambientes computacionais.
Objetivo: desenvolver estratégia para solução computacional de equações diferenciais lineares de segunda ordem via técnica de diferenças finitas,
Descrição do problema: para $\alpha, \beta, \gamma \in \mathbb{R}$, queremos encontrar solução computacional de

$$\begin{cases} \alpha y'' + \beta y' + \gamma y = & g(t), c < t < d \\ y(c) = y_c, & y(d) = y_d \end{cases} \qquad (\dagger)$$

para valores $\alpha, \beta, \gamma, c, d, y_c, y_d$ e para uma função $g(t)$ dados.
Metodologia: a partir de uma discretização que define uma partição regular $t_i = c + i\triangle t, i = 0, 1, \ldots, n$, onde $\triangle t = (d-c)/n$, usaremos diferenças finitas centrais

$$y''(t_i) \approx \frac{y_{i+1} - 2y_i + y_{i-1}}{(\triangle t)^2} \; ; \; y'(t_i) \approx \frac{y_{i+1} - y_{i-1}}{2\triangle t}.$$

que então implicam em um conjunto de $n-1$ equações lineares algébricas

$$\alpha \frac{y_{i+1} - 2y_i + y_{i-1}}{(\triangle t)^2} + \beta \frac{y_{i+1} - y_{i-1}}{2\triangle t} + \gamma y_i = g(t_i), i = 1, 2, 3, \ldots, n-1$$

sobre incógnitas $y_1, y_2, \ldots, y_{n-1}$, onde ainda $y_0 = y_c$ e $y_n = y_d$ são determinados pela condição de contorno. Multiplicando ambos os lados por $(\triangle t)^2$ e re-agrupando, obtemos um sistema linear triadiagonal na forma $Ax = b$, com $n-1$ equações e $n-1$ incógnitas, onde

$$A = \begin{bmatrix} u_1 & v_1 & & & \\ w_1 & u_2 & v_2 & & \\ & w_2 & u_3 & \ldots & \\ & & \ldots & \ldots & v_{n-2} \\ & & & w_{n-2} & u_{n-1} \end{bmatrix}, \; b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \ldots \\ b_{n-1} \end{bmatrix}$$

e ainda $u_i = -2\alpha + \gamma(\triangle t)^2$, $v_i = \alpha + \beta\triangle t/2$, $w_i = \alpha - \beta\triangle t/2$. Além disso,

$b_1 = (\triangle t)^2 g(t_1) - (\alpha - \beta\triangle t/2) y_0$;
$b_i = (\triangle t)^2 g(t_i), i = 2, \ldots, n-2$;
$b_{n-1} = (\triangle t)^2 g(t_{n-1}) - (\alpha + \beta\triangle t/2) y_n$.

Sua tarefa é implementar, em *Scilab*, *Matlab*, *Fortran 90* ou *Ansi C*:

(i) modificações dos algoritmos 4.3.1, 4.3.2 e 4.3.3 de Golub (Matrix Computations) para o caso $p = q = 1$ em que a matriz A é armazenada nos três vetores (u_i), (v_i) e (w_i) da descrição acima.

(ii) algoritmo para encontrar a solução computacional (y_i), $i = 0, 1, \ldots, n$ da equação diferencial (†) quando $\alpha, \beta, \gamma, c, d, n, y_c, y_d$ e o vetor com as quantidades $g(t_i)$, $i = 1, \ldots, n-1$ são dados.

Suas rotinas deverão atender a padronização:

```
// Padronizacao em Scilab ou Matlab
function [u,v,w]=alg431trid(u,v,w)  % atualiza vetores u,v,w
function b=alg432trid(w,b)          % atualiza vetor b, mantem w
function b=alg433trid(u,v,b)        % atualiza vetor b, mantem u e v
function y=tarefa4(al,be,ga,c,d,n,y0,yn,g) % g eh vetor com os g(ti)
// Padronizacao em Fortran 90 ou similar em ANSI C, usando void
```

```
subroutine alg431trid(n,u,v,w)
  integer n
  real (kind=8) u(*),v(*),w(*)
subroutine alg432trid(n,w,b)
  integer n
  real (kind=8) w(*),b(*)
subroutine alg433trid(n,u,v,b)
  integer n
  real (kind=8) u(*),v(*),b(*)
subroutine tarefa4(al,be,ga,c,d,n,y0,yn,g,y)
  integer n
  double precision al,be,ga,c,d,y0,yn,g(*),y(*)
```

Para fins de teste, apresente a plotagem da solução numérica de

$$\begin{cases} 2y'' + 5y' + 7y = & \cos(t), 0 < t < 2\pi \\ y(0) = -2, & y(2\pi) = 3 \end{cases}$$

para $n = 200$.

□

Map0201 - Métodos Matriciais Computacionais
Quinta Tarefa Computacional, entregar até a data da segunda prova

Assunto: sistemas lineares especiais, fatorações matriciais, equações diferenciais com condições de contorno, ambientes computacionais.
Objetivo: desenvolver estratégia para solução computacional do problema com condições de contorno periódicas

$$\begin{cases} -\kappa \frac{d^2T}{dx^2} = g(x), & 0 < x < L \\ T(0) = T(L) = Q \\ T'(0) = T'(L) \end{cases}$$

onde $\kappa > 0$, $L > 0$, $Q \geq 0$ são dados.
Metodologia: para todo inteiro $n > 2$, definimos uma discretização (particionamento) $\{x_0, x_1, \ldots, x_n\}$ via $x_i = i\triangle x$, $i = 0, 1, \ldots, n$, onde $\triangle x = L/n$. Sendo $T_0, T_1, \ldots, T_n$ as respectivas discretizações de T, ou seja $T_i = T(x_i)$, sabemos que

$$T''(x_i) \approx \frac{T_{i-1} - 2T_i + T_{i+1}}{(\triangle x)^2} \Rightarrow \frac{-T_{i-1} + 2T_i - T_{i-1}}{(\triangle x)^2} = \frac{g(x_i)}{\kappa}, i = 1, 2, \ldots, n-1$$

com as condições de contorno periódicas

$$\begin{cases} T_0 = T_n, & \text{que elimina } T_0 \text{ do sistema de equações} \\ \frac{T_n - T_{n-1}}{\triangle x} = \frac{T_1 - T_n}{\triangle x} & \text{onde já substituimos } T_0 = T_n, \text{ implica } -T_{n-1} + 2T_n - T_1 = 0 \end{cases}$$

que então implicam

$$\begin{bmatrix} 2 & -1 & 0 & \ldots & 0 & -1 \\ -1 & 2 & -1 & 0 & \ldots & 0 \\ 0 & -1 & 2 & -1 & 0 & \ldots \\ \ldots & 0 & -1 & 2 & -1 & 0 \\ 0 & \ldots & 0 & -1 & 2 & -1 \\ -1 & 0 & \ldots & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ \ldots \\ T_{n-1} \\ T_n \end{bmatrix} = \frac{(\triangle x)^2}{\kappa} \begin{bmatrix} g(x_1) \\ g(x_2) \\ g(x_3) \\ \ldots \\ g(x_{n-1}) \\ g(x_n) \end{bmatrix}$$

Sua tarefa: à luz do problema P4.3.11 de Golub and Van Loan:

1. mostre que o sistema é singular e encontre restrições que garantam a existência de soluções;
2. sob a garantia de existência de soluções, eliminando a última linha e a última coluna da matriz de coeficientes, encontre o conjunto de equações que permitam a solução computacional a partir de um valor $T_n = Q$ prescrito pelo problema; a matriz de coeficientes é simétrica e positiva definida;
3. implemente o Algoritmo 4.3.6 (fatoração LDL^T tridiagonal) de Golub and Van Loan (Scilab, Matlab ou Fortran 90);
4. apresente códigos (Scilab, Matlab ou Fortran 90) para a solução computacional desse problema;
5. apresente a solução computacional dos problemas-teste.

Problemas Teste:

1. $L = 10, g(x) = 0.1\text{sen}\,(2\pi x/L), Q = 3, \kappa = 10, n = 200$;
2. $L = 4, g(x) = \dfrac{x(x - L/2)(x - L)}{80}, Q = 10, \kappa = 20, n = 100$;

□