RT-MAP-8401-B

PROCEDIMENTOS PARA CÁLCULOS COM SPLINES

Parte B - Descrição dos Procedimentos

Dirceu Douglas Salvetti

JANEIRO 1984

PARTE B -- CAPITULO 0

RESULTADOS BÁSICOS - ALGORITMOS PARA AVALIAÇÃO, INTEGRAÇÃO E DIFERENCIAÇÃO DE SPLINES POLINOMIAIS

Um breve resúmo de teoria que justifica os algoritmos utilizados nos procedimentos deste manual encontra-se na parte A.

Os procedimentos codificados em ALGOL estão gravados em fita magnética e estão disponíveis aos usuários do CCE da USP.

Para sua utilização devem ser empregados os seguintes comandos de controle:

```
?BEGIN JØB SPL; USER = <código do usuário>/<password>; CLASS 2;
?CØPY SPLØTØ/ = , DECLARA FRØM FITA (SERIALNØ="E30050");
?CØMPILE <nome> ALGOL; ALGOL DATA;
 $SET AUTØBIND
 $BIND=FRØM SPLØTØ/=
       BEGIN
 $INCLUDE "DECLARA"
       FILE CR(KIND=REMOTE), IMP(KIND=PRINTER);
       <programa principal>
       END.
?DATA
 <dados do programa principal>
?END JØB
```

Note que o comando de controle $INCLUDE "DECLARA" é intercalado, entre os comandos do programa principal, imediatamente após o comando BEGIN.

Os procedimentos foram introduzidos na parte A na seguinte ordem:

LBISEC, INTERV, PTE, INVPTE, BSMQ,
BSX, BSX2, BSMCD, BSCDJ, BSDJX,
BSDX, BSDX2, BSMCI, BSID, BSMGRM,
BSRPP, RPPSX, RPPSDX, BSX2H, BSMQH,
BSXH, BSMCDH, BSDXH, BSMCIH, BSIDH, RSRPPH.

A relação,com os nomes e definições dos procedimentos que apresentaremos a seguir, fará uso das seguintes notações:

Todos nomes que começam por BS identificam procedimentos que envolvem a expansão em B-splines de um spline $s \in \mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$. Todos que começam por BS e terminam por H subentendem B-splines associados com nós de uma partição uniforme de passo h.

O espaço $\mathcal{S}_m(\Delta h)$ indica o espaço dos splines com nós simples (multiplicidade 1) em pontos de uma partição uniforme de passo h.

RELAÇÃO DOS PROCEDIMENTOS EM ORDEM ALFABÉTICA DE SEUS NOMES

NOME	FINALIDADE
BSCDJ	Determina os coeficientes da expansão em B-splines de $D_+^{j-1}s(x)$, com as funções B-splines definida sobre nós da partição estendida associada com $\mathcal{S}(\mathcal{P}_{m-j+1}; \mathcal{M}'; \Delta)$
BSDJX	Avalia $D^{j-1}s(x)$, $s \in \mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$
BSDX	Avalia $s(x), Ds(x), \ldots, D^{m-1}s(x)$, $s \in \mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$
BSDXH	A mesma de BSDX, porém com $s \in \mathcal{S}_m(\Delta h)$
BSDX2	A mesma BSDX
BSID	Avalia a integral definida de $s \in \mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$
BSIDH	A mesma de BSID, porém com $s \in \mathcal{S}_m(\Delta h)$
BSMCD	Determina os coeficientes das expansões em B-splines das derivadas sucessivas de s com as funções B-splines definidas sobre nós da partição estendida associada a $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$
BSMCDH	A mesma de BSMCD, porém com $s \in \mathcal{S}_m(\Delta h)$
BSMCI	Determina os coeficientes da expansão em B-splines da integral indefinida $\int_{y_1}^{x} s(t)dt$, $s \in \mathcal{S}(\mathcal{P}_{m+1}; \mathcal{M}; \Delta)$

NOME	FINALIDADE
BSMCIH	A mesma da BSMCI, porém com $s \in \mathcal{S}_m(\Delta h)$
BSMGRM	Determina a matriz de Gram
BSMQ	Determina $\{N_i^m(x)\}_{\ell+1-m}^{\ell}$ em $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$
BSMQH	A mesma de BSMQ, porém em $\mathcal{S}_m(\Delta h)$
BSRPP	Converte a expansão em B-splines de um spline $s \in \mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$ para sua representação polinomial por partes.
BSRPPH	A mesma de BSRPP, porém com $s \in \mathcal{S}_m(\Delta h)$
BSX	Avalia $s \in \mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$ num ponto $x \in I = [a,b]$
BSXH	A mesma de BSX2H
BSX2	A mesma de BSX
BSX2H	A mesma de BSX, porém com $s \in \mathcal{S}_m(\Delta h)$
INTERV	Determina ℓ tal que $y_\ell \leq x \leq y_{\ell+1}$
INVPTE	Determina $\tilde{\Delta}$ a partir de Δ associada com $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$
LBISEC	A mesma de INTERV
PTE	Determina Δ a partir de $\tilde{\Delta}$ associada com $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$
RPPSDX	Derivadas sucessivas de $s \in \mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$ ou $s \in \mathcal{S}_m(\Delta h)$ através de sua representação polinomial por partes.
RPPSX	Avaliação de $s \in \mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$ ou $s \in \mathcal{S}_m(\Delta h)$ através de sua representação polinomial por partes.

RELAÇÃO DOS PROCEDIMENTOS EM ORDEM ALFABÉTICA DE SUAS FINALIDADES

NOME	FINALIDADE
BSX	Avaliação de $s \in \mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$
BSX2	Avaliação de $s \in \mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$
BSX2H	Avaliação de $s \in \mathcal{S}_m(\Delta h)$
BSXH	Avaliação de $s \in \mathcal{S}_m(\Delta h)$
RPPSX	Avaliação de $s \in \mathcal{S}_m(\Delta h)$ ou $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$ através de sua representação polinomial por partes
BSMQ	B-splines normalizados não nulos num ponto x de $[a,b]$, associados com nós de uma partição estendida definida em $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$
BSMQH	B-splines normalizados não nulos num ponto x de $[a,b]$, associados com nós de uma partição estendida definida em $\mathcal{S}_m(\Delta h)$
BSRPP	Conversão da expansão em B-splines de $s \in \mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$ para sua representação polinomial por partes

NOME	FINALIDADE
BSRPPH	Conversão da expansão em B-splines de $s \in \mathcal{S}_m(\Delta h)$ para sua representação polinomial por partes
PTE	Conversão da partição Δ para a partição estendida $\tilde{\Delta}$ definida em $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$
INVPTE	Conversão da partição estendida $\tilde{\Delta}$ para a partição Δ definida em $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$
BSDJX	Derivada--Avaliação de $D_+^{j-1}s(x)$, $s \in \mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$
BSCDJ	Derivada - Coeficientes da expansão em B-splines de $D_+^{j-1}s(x)$ com as funções B-splines definidas sobre nós da partição uniforme associada com $\mathcal{S}(\mathcal{P}_{m-j+1}; \mathcal{M}'; \Delta)$
BSMCD	Derivadas - Coeficientes das expansões em B-splines das derivadas sucessivas de s com as associada com $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$
BSMCDH	Derivadas - Coeficientes das expansões em B-splines das derivadas sucessivas de $s \in \mathcal{S}_m(\Delta h)$
BSDX	Derivadas sucessivas de $s \in \mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$
BSDX2	Derivadas sucessivas de $s \in \mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$
BSDXH	Derivadas sucessivas de $s \in \mathcal{S}_m(\Delta h)$
RPPSDX	Derivadas sucessivas de $s \in \mathcal{S}_m(\Delta h)$ ou $s \in \mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$ através de sua representação polinomial por partes
BSID	Integral - Avaliação de integral definida de $s \in \mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$
BSIDH	Integral - Avaliação da integral definida de $s \in \mathcal{S}_m(\Delta h)$
BSMCI	Integral - Coeficientes da expansão em B-splines da integral indefinida $\int_{y_1}^{x} s(t)dt$, $s \in \mathcal{S}(\mathcal{P}_{m+1}; \mathcal{M}; \Delta)$
BSMCIH	Integral - Coeficientes da expansão em B-splines da integral indefinida $\int_{y_1}^{x} s(t)dt$, $s \in \mathcal{S}_{m+1}(\Delta h)$
BSMGRM	Integral - Matriz de Gram
LBISEC	Intervalo $[y_\ell , y_{\ell+1})$. Determinação de ℓ tal que $y_\ell \leq x < y_{\ell+1}$
INTERV	Intervalo $[y_\ell , y_{\ell+1})$. Determinação de ℓ tal que $y_\ell \leq x < y_{\ell+1}$
BSMGRM	Matriz de Gram

INDICE DO CAPITULO 0

Os procedimentos encontram-se descritos por ordem alfabética de seus nomes.

Cap.0-B - Determina os coeficientes da expansão em B-splines de

$D_+^{j-1}s(x) \quad \epsilon \quad \mathcal{S}(\mathcal{P}_{m-j+1}; \mathcal{M}'; \Delta)$

Dados

m = ordem dos polinômios por partes.

n = dimensão de $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$.

$\{y_i\}_1^{m+n}$ nós da partição estendida Δ associada com $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$

$\{c_i\}_1^n$ coeficientes da expansão em B-splines de $s \in \mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$

j ordem da (j-1)-ésima derivada a direita de s ($1 \leq j \leq n$)

determina

$\tilde{n}$ = dimensão de $\mathcal{S}(\mathcal{P}_{m-j+1}; \mathcal{M}'; \Delta)$

$\tilde{m} = m-j+1$ = ordem dos polinômios por partes.

$\{\tilde{y}_i\}_1^{\tilde{m}+\tilde{n}}$ nós da partição estendida associada a $\mathcal{S}(\mathcal{P}_{m-j+1}; \mathcal{M}'; \Delta)$.

$\{cc_i\}_1^{\tilde{n}}$ coeficientes da expansão em B-splines de $D_+^{j-1}s(x)$, considerando as funções B-splines definidas em pontos da partição estendida associada a $\mathcal{S}(\mathcal{P}_{m-j+1}; \mathcal{M}'; \Delta)$.

Procedimento

BSCDJ (m, n, y, c, j, mtil, ntil, ytil, cc)

Entrada

m, n, y[1],..., y[m+n], c[1],..., c[n], j

As dimensões das matrizes são propiciadas pelo programa principal.

Saída

mtil, ntil, ytil[1],..., ytil[mtil+ntil], cc[1],..., cc[ntil].

As dimensões das matrizes são fornecidas pelo programa principal e dimensionadas iguais as matrizes y e c, respectivamente.

Método

Inicialmente determina a matriz $\{cd(j,i)\}_{j=1,\ i=j}^{m\quad n}$ que contém em cada linha j coeficientes da expansão em B-splines de $D_+^{j-1}s(x)$, com as funções B-splines definidas em nós da partição estendida associada com $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$.

Em seguida elimina os coeficientes cd(j,i) teoricamente nulos e determina

Cap.0-B - Determina os coeficientes da expansão em B-splines de $D_+^{j-1}s(x) \in \mathcal{S}(\mathcal{P}_{m-j+1}; \mathcal{M}'; \Delta)$.

---

j+ntil-1 de tal modo que os primeiros ntil coeficientes $\{cd(j,i)\}_{i=j}^{n}$ não teoricamente nulos fiquem alocados em $\{cd(j,i)\}_{i=j}^{j+ntil-1}$, sendo ntil a dimensão de $\mathcal{S}(\mathcal{P}_{m-j+1}; \mathcal{M}'; \Delta)$.

Finalmente armazena $\{cd(j,i)\}_{i=j}^{j+ntil-1}$ em $\{cc(i)\}_{i=1}^{ntil}$, determina mtil = = m-j+1, ntil e ytil.

Procedimentos Auxiliares

BSMCD, INVPTE e PTE

Multiplicações ou Divisões

Nenhuma, excluindo-se as operações em BSMCD, INVPTE e PTE.

Algoritmo

```
procedimento BSCDJ(m,n,y,c,j,mtil,ntil,ytil,cc);
inteiro m,n,j,mtil,ntil; real y[*], ytil[*], c[*], cc[*];
inteiro ℓm[1:n-m], nℓm[1:n-m]; real x[1:n-m], cd[1:m,1:n];
 inteiro  k,i,mi,kd,ik; real a,b;
 BSMCD(m,n,y,c,cd);  INVPTE(m,n,y,k,a,b,ℓm,x);
 para i de 1 passo 1 até k faça
 nℓm[i] ← ℓm[i];
  se ℓm[i] >m-j+1 então nℓm[i] ← m-j+1 ;
 mi ← m; kd ← 0;
 para i de 1 passo 1 até k faça
  kd ← kd + ℓm[i] - nℓm[i];
    ntil ← mi + nℓm[i]; se kd > 0 então
    para ik de mi+1 passo 1 até ntil faça
    cd[j,ik] ← cd[j,ik+kd];
    mi ← ntil ;
para i de j passo 1 até ntil faça  cc[i-j+1] ← cd[j,i];
mtil ← m-j+1;
PTE(mtil,ntil,ytil,k,a,b,nℓm,x)
```

Cap.0-B - Determina os coeficientes da expansão em B-splines de

$D_+^{j-1}s(x) \quad \in \quad \mathcal{S}(\mathcal{P}_{m-j+1}; \mathcal{M}'; \Delta)$

Codificação em ALGOL

```
%=============================================================================
 PROCEDURE BSCDJ(M,N,Y,C,J,NTIL,NTIL,YTIL,CC);
 INTEGER M,N,J,MTIL,NTIL; REAL ARRAY Y[*],YTILI*],CC[*],C[*];
%-----------------------------------------------------------------------------
%     UTILIZA BSMCD-PTE-INVPTE
%-----------------------------------------------------------------------------
 BEGIN
 INTEGER ARRAY LM[1:N-M], NLM[1:N-M] ; REAL ARRAY X[1:N-M],CD[1:M,1:N];

 INTEGER K, I, MI, KD,        IK ; REAL  A,B ;
 BSMCD(M,N,Y,C,CD);  INVPTE(M,N,Y,K,A,B,LM,X) ;
%
 FOR I:=1  STEP 1  UNTIL  K  DO
  BEGIN NLM[I]:= LM[I];
        IF LM[I] > M-J+1 THEN NLM[I]:= M-J+1
  END;
 MI:= M ;  KD:= 0 ;
 FOR I:= 1  STEP 1  UNTIL  K  DO
  BEGIN  KD:= KD + LM[I] - NLM[I];
         NTIL:= MI + NLM[I] ;
         IF KD  > 0 THEN
         FOR IK:= MI+1 STEP 1 UNTIL NTIL DO CC[J,IK]:= CC[J,IK+KD] ;
         MI:=NTIL
  END;
 FOR I:=  J STEP 1 UNTIL NTIL DO  CC[I-J+1]:=CD[J,I];
 WRITE(IMP,</,"CC[I]=",*(F4.1)>,NTIL-J+1,FOR I:=1 STEP 1 UNTIL N-J+1 DO
CC[I]);
 MTIL := M-J+1;  PTE(MTIL,NTIL,YTIL,K,A,B,NLM,X)
 END;
```

Cap.0-B - Avalia $D_+^{j-1}s(x)$ através da expansão de s em B-splines

---

Dados

m = ordem dos polinômios por partes

n = dimensão de $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$

$\{y_i\}_1^{m+n}$ nós da partição estendida $\bar{\Delta}$ associada com $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$

$\{c_i\}_1^n$ coeficientes da expansão em B-splines de s

j ordem da (j-1)-ésima derivada a direita de s ($1 \leq j \leq m$)

x ponto de [a,b]

determina

$D_+^{j-1}s(x)$

Função Real

BSDJX(m,n,y,c,j,x)

Entrada

m,n,y[1],...,y[m+n], c[1],...,c[n], j e x

Os dimensionamentos das matrizes são propiciados pelo programa principal.

Saída

BSDJX $\leftarrow D_+^{j-1}s(x)$

Método

Inicialmente determina a matriz $\{cd(j,i)\}_{j=1,\ i=j}^{m\quad n}$ que contém em cada linha j coeficientes da expansão em B-splines de $D_+^{j-1}s(x)$, com as funções B-splines definidas em nós da partição estendida associada a $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$. Em seguida determina ℓ tal que $y_\ell \leq x < y_\ell + 1$. Finalmente avalia $D_+^{j-1}s(x)$ aplicando a fórmula

$$D_+^{j-1}s(x) = \sum_{i=j}^{n} cd(j,i)N_i^{m-j+1}(x)$$

Procedimentos Auxiliares

BSMCD - BSX2 - INTERV(LBISEC)

Cap.0-B - Avalia $D_+^{j-1}s(x)$ através da expansão de s em B-splines

---

Multiplicações ou Divisões

Avaliando-se $D_+^{j-1}s(x)$ por meio da função real BSX2 e excluindo-se as operações para determinação da matriz CD e de ℓ, teremos:

$\frac{1}{2}(3\tilde{m}(\tilde{m}-1)$ multiplicações ou divisões, sendo $\tilde{m} = m-j+1$.

Algoritmo

```
função real BSDJX(m,n,y,c,j,x);
inteiro m,n,j; real y[*], c[*];
┌  inteiro i, mtil, ℓ ; real cd[1:m,1:m], cc[1:n]; real s;
   BSMCD(m,n,y,c,cd);
   se x = y[n+1]então ℓ ← n senão ℓ ← INTERV(1,m+n,(m+n)/2,x,y);
   se j = m então s ← cd[m,ℓ]
           senão ┌┌ para i de j passo 1 até n faça cc[i] ← cd[j,i];
                 mtil ← m-j+1;
                 s ← BSX2(mtil,n,y,cc,x,ℓ)└┘ ;
   BSDJX ← s┘
```

Listagem em ALGOL

```
%=============================================================================
 REAL PROCEDURE BSDJX(M,N,Y,C,J,X);
 INTEGER M,N,J; REAL X; REAL ARRAY Y[*],C[*];
%-----------------------------------------------------------------------------
%     UTILIZA BSMCD-BSX2-INTERV(LEISEC)
%-----------------------------------------------------------------------------
 BEGIN

 INTEGER I,MTIL,L; REAL ARRAY CD[1:M,1:N], CC[1:N] ; REAL S;
 BSMCD(M,N,Y,C,CD);
 IF  X =Y[N+1] THEN L:=N ELSE L:=INTERV(1,M+N,(M+N)/2,X,Y);
 IF  J =M THEN S:=CD[M,L]
         ELSE BEGIN FOR I:=J STEP 1 UNTIL N DO CC[I]:=CD[J,I];
                    MTIL:= M-J+1;
                    S:=BSX2(MTIL,N,Y,CC,X,L)
              END;
 BSDJX:=S
 END;
```

Cap.0-B - Avalia $\{D_+^{j-1}s(x)\}_{j=1}^{m}$ através da expansão de s em B-splines.

Dados

m = ordem dos polinômios por partes

n = dimensão do espaço $\mathcal{S}(\mathcal{P}m;\mathcal{M};\Delta)$

$\{y_i\}_1^{m+n}$ nós da partição estendida $\bar{\Delta}$ associada com $\mathcal{S}(\mathcal{P}m;\mathcal{M};\Delta)$

$\{c_i\}_1^n$ coeficientes da expansão em B-splines do spline s

x ponto de [a,b]

determina

$D_+^{m-1}s(x)$, ... , $D_+s(x)$, s(x) armazenando-os, respectivamente, em dx(m),..., , dx(2),dx(1).

Procedimento

BSDX (m,n,y,c,x,dx)

Entrada

m,n,y[1],..., y[m+n], c[1],..., c[n] e x.

Os dimensionamentos das matrizes são fornecidos pelo programa principal.

Saída

dx[1], dx[2],..., dx[m] cujo dimensionamento é previsto no programa principal.

Método

$$D_+^{m-j}s(x) = \sum_{i=\ell+1-j}^{\ell} cd(m-j+1,i)(y_{i+j}-y_i)Q_i^j(x),\quad j = 1, 2, \ldots, m$$

Inicialmente determina-se a matriz $\{cd(j,i)\}_{j=1,\ i=j}^{m\quad n}$ onde a j-ésima linha contém os coeficientes da expansão em B-splines de $D_+^{j-1}s(x)$, considerando as funções B-splines definidas em nós da partição estendida associada a $\mathcal{S}(\mathcal{P}m;\mathcal{M};\Delta)$.

Em seguida determina-se ℓ tal que $y(\ell) \leq x < y(\ell+1)$ ou $\ell = n$ se $x=y(n+1)$.

Finalmente, por meio do mesmo algoritmo utilizado em BSMQ, constrói-se para cada j o conjunto $\{Q_i^j\}_{i=\ell+1-j}^{\ell}$ e se avalia $D_+^{m-j}s(x)$ por meio da fórmula acima.

Cap.0-B - Avalia $\{D_+^{j-1}s(x)\}_{j=1}^{m}$ através da expansão de s em B-splines.

Procedimentos Auxiliares

BSMCD, INTERV(LBISEC)

Multiplicações ou Divisões

Excluindo-se as operações utilizadas na determinação de ℓ e da matriz cd:

$$(5m^2 + m - 8)/2$$

Comparação com procedimentos equivalentes

m	BSDX2	BSDX
	$(m-1)m(m+1)/2$	$(5m^2+m-8)/2$
2	3	7
3	12	20
4	30	38
5	60	61
6	105	89

O procedimento BSDX torna-se mais eficiente que BSDX2 a partir de m=6.

Algoritmo

```
procedimento BSDX(m,n,y,c,x,dx);
inteiro m,n; real x; real c[*], dx[*];
| inteiro i,j,k; real denom, a1, a2; real cd[1:m,1:n], Q[1:m+1];
  se x = y[n+1] então k ← n senão k ← INTERV(1,m+n,(m+n)/2,x,y);
  BSMCD(m,n,y,c,cd); dx[m] ← cd[m,k];
  se m > 1 então
  || para j de 1 passo 1 até m-1 faça Q[j] ← 0;
     Q[m] ← 1./(y[k+1] - y[k]); Q[m+1] ← 0;
     para j de 2 passo 1 até m-1 faça
     ||| para i de m-j+1 passo 1 até m faça
         |||| denom ← y[i+k-m+j] - y[i+k-m];
              a1 ← (x-y[i+k-m])/denom;
              a2 ← 1-a1;
              Q[i] ← a1*Q[i] + a2*Q[i+1] |||| ;
         dx[m-j+1] ← 0;
         para i de k+1-j passo 1 até k faça
         dx[m-j+1] ←dx[m-j+1] + cd[m-j+1,i]* (y[i+j] - y[i])*Q[m-k+i] ||| ;
     dx[1] ← 0;
     para i de k+1-m passo 1 até k faça
     dx[1] ← dx[1] + c[i]* Q[m-k+i] ||  |
```

Cap.0-B - Avalia $\{D_+^{j-1}s(x)\}_{j=1}^{m}$ através da expansão de s em B-splines.

Codificação em ALGOL

```
%==============================================================================
  PROCEDURE BSDX(M,N,Y,C,X,DX);
  INTEGER M,N; REAL X; REAL ARRAY YI*],CI*],DXI*];
%------------------------------------------------------------------------------
%     UTILIZA LEISEG, INTERV  E  BSMCD
%------------------------------------------------------------------------------
  BEGIN

  INTEGER I,J,K; REAL DENOM,A1,A2; REAL ARRAY CCI1:M,1:N],GI1:M+1];
  IF X=YIN+1] THEN K:=N ELSE K:=INTERVI1,M+N,(M+N)/2,X,Y);
  BSMCD(M,N,Y,C,CC);
  DXIM]:=CCIM,K];
  IF M > 1 THEN BEGIN
  FOR J:=1 STEP 1 UNTIL M-1 DO GIJ]:=0;
  GIM]:=1./(YIK+1]-YIK]); GIM+1]:=0;
  FOR J:=2 STEP 1 UNTIL M-1 DO
    BEGIN FOR I:=M-J+1 STEP 1 UNTIL M DO
            BEGIN DENOM:=YII+K-M+J]-YII+K-M];
                  A1:=(X-YII+K-M])/DENOM; A2:=1-A1;
                  GII]:=A1*GII]+A2*GII+1]
            END;
          DXIM-J+1]:=0;
          FOR I:=K+1-J STEP 1 UNTIL K DO
            DXIM-J+1]:=DXIM-J+1]+CCIM-J+1,I]*(YII+J]-YII])*GIM-K+I] END;
  DXI1]:=0;
  FOR I:=1 STEP 1 UNTIL M DO GII]:=(X-YII+K-M])*GII]+(YII+K]-X)*GII+1];
  FOR I:=K+1-M STEP 1 UNTIL K DO DXI1]:=DXI1]+CII]*GIM-K+I]
  END; END;
```

Cap.0-B - Determina $\{D_+^{j-1}s(x)\}_1^m$ através de sua expansão em B-splines, $s \in \mathcal{S}_m(\Delta h)$.

Dados

Os extremos do intervalo $[a,b]$

k = número de nós da partição uniforme associada a $\mathcal{S}_m(\Delta h)$

m = ordem dos polinômios por partes

$\{c_i\}_1^{m+k}$ coeficientes da expansão em B-splines de $s \in \mathcal{S}_m(\Delta h)$

x = ponto de $[a,b]$

determina

$s(x)$, $D_+s(x)$, ... , $D_+^{m-1}s(x)$, armazenando-os, respectivamente, em $\{dx_j\}_1^m$

Procedimento

BSDXH(a,b,k,m,c,dx)

Entrada

a,b,k,m,c[1],...,c[m+k]

O dimensionamento da matriz deve ser propiciado pelo programa principal

Saída

dx[1],...,dx[m], com dimensionamento previsto no programa principal

Método

Versão modificada do algoritmo utilizado em BSDX2.

Em BSMCDH vimos que $D_+^{j-1}s(x) = \sum_{i=j}^{n} cd(j,i)N^{m-j+1}\left(\frac{x - y_i}{h}\right)$, $j=1,\ldots,m$.

Utilizando BSMCDH calculamos $\{cd(j,i)\}_{j=1,\ i=1}^{m \quad n}$.

Determinando-se ℓ tal que $y_\ell \leq x < y_{\ell+1}$, obtem-se $D_+^{m-1}s(x) = cd(m,\ell)$.

Em seguida para cada j e $\tilde{m} = m-j+1$, $1 \leq j \leq m-1$ calculamos

$D_+^{j-1}s(x) = \sum_{i=\ell-\tilde{m}+1}^{\ell} cd(j,i)N^{\tilde{m}}\left(\frac{x - y_i}{h}\right)$ através de BSX2H.

Procedimentos Auxiliares

BSMCDH e BSX2H

Cap.0-B - Determina $\{D_+^{j-1}s(x)\}_1^m$ através de sua expansão em B-splines, $s \in \mathcal{S}_m(\Delta h)$.

Multiplicações ou Divisões

Não computando as operações para cálculos de h, ℓ e matriz CD:

$$\sum_{i=2}^{m} (i^2 - i + 2) = (m^3 + 5m - 6)/3$$

Comparação com procedimentos equivalentes

m	BSDX	BSDX2	BSDXH
	$(5m^2+m-8)/2$	$m(m-1)(m+1)/2$	$(m^3+5m-6)/3$
2	7	3	4
3	20	12	12
4	38	30	26
5	61	60	48
6	89	105	80

Algoritmo

```
procedimento BSDXH(a,b,k,m,c,dx);
inteiro n,ℓ,ℓx,mtil,i,j; real h; real Q[1:m+1];
 n ← m + k;
  real cd[1:m ; 1:n], cc[1:n];
    h ← (b-a)/(k+1);
    BSMCDH(a,b,k,m,c,cd);
    se x=b então ℓ ← n  senão ℓ ← (x-a)/h + m - 0.5;
    dx[m] ← cd[m,ℓ];
    para j de 1 passo 1 até m-1 faça
      mtil ← m-j+1;
        para i de j passo 1 até n faça
        cc[i-j+1] ← cd[j,i];
        dx[j] ← BSX2H(a,b,k,mtil,cc,x) ;  ;
```

Cap.0-B - Determina $\{D_+^{j-1}s(x)\}_1^m$ através de sua expansão em B-splines, $s \in \mathcal{S}_{m(\Delta h)}$

Codificação em ALGOL

```
%=============================================================================
 PROCEDURE BSDXH(A,B,K,M,C,X,DX);
 INTEGER K,M;  REAL A,B,X;  REAL ARRAY C[*],DX[*];
%-----------------------------------------------------------------------------
%       UTILIZA BSX2H  E  BSMCDH
%-----------------------------------------------------------------------------
 BEGIN
 INTEGER N,L,LX,MTIL,I,J; REAL H; REAL ARRAY C[1:M+1];

 N:=M+K;
  BEGIN
  REAL ARRAY CD[1:M,1:N], CC[1:N];

  H:=(B-A)/(K+1);    BSMCDH(A,B,K,M,C,CD);
  IF X=B THEN L:=N ELSE L:= (X-A)/H + M - 0.5 ;
  DX[M]:= CD[M,L];
  FOR J:=1 STEP 1 UNTIL M-1 DO
    BEGIN
    MTIL:=M-J+1; FOR I:=J STEP 1 UNTIL N DO CC[I-J+1]:=CD[J,I];
        DX[J]:= BSX2H(A,B,K,MTIL,CC,X)
    END;  END;  END;
```

Cap.0-B - Avalia $\{D_+^{j-1}s(x)\}_{j=1}^{m}$ através de sua expansão em B-splines

---

Dados

m = ordem dos polinômios por partes

n = dimensão do espaço $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$

$\{y_i\}_1^{m+n}$ nós da partição estendida Δ associada com $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$

$\{c_i\}_1^n$ coeficientes da expansão em B-splines do spline s.

x ponto de [a,b]

determina

$s(x)$, $D_+s(x)$, $D_+^{m-1}s(x)$, armazenando-os, respectivamente, em dx[1], dx[2],..., dx[m].

Procedimento

BSDX2(m,n,y,c,x,dx)

Entrada

m,n,y[1],..., y[m+n],c[1],..., c[n], x

Os dimensionamentos das matrizes devem ser previstos no programa principal.

Saída

dx[1], dx[2],..., dx[m], com dimensionamento propiciado pelo programa principal.

Método

Inicialmente obtem-se a matriz $\{cd(j,i)\}_{j=1,\ i=j}^{m\qquad n}$ onde a j-ésima linha contém os coeficientes da expansão em B-splines de $D_+^{j+1}s(x)$, considerando-se as funções B-splines definidas em nós de partição estendida associada a $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$.

Em seguida determina-se ℓ tal que $y(\ell) \leq x < y(\ell+1)$ ou $\ell=n$ se $x=y(n+1)$.

Finalmente para cada j=1,2,...,m calcula $D_+^{j-1}s(x) = \sum_{i=j}^{n} cd(j,i)N_i^{m-j+1}(x)$.

Procedimentos Auxiliares

BSMCD, BSX2, INTERV(LBISEC)

Cap.0-B - Avalia $\{D_+^{j-1}s(x)\}_{j=1}^{m}$ através de sua expansão em B-splines

Multiplicações ou Divisões

Não contando as operações para determinação de ℓ e da matriz cd e considerando-se o uso do algoritmo BSX2 para avaliar $D_+^{j-1}s(x)$ teremos:

$$\sum_{i=1}^{m} \frac{3i^2 - 3i}{2} = \frac{1}{2}(m-1)m(m+1)$$

Algoritmo

```
procedimento BSDX2(m,n,y,c,x,dx);
inteiro m,n,; real x ; real y[*], c[*], dx[*];
 inteiro i,j,ℓx,mtil; real cc[1:n], cd[1:m ; 1:n];
  BSMCD(m,n,y,c,cd);
  se x=y[n+1] então ℓx ← n senão ℓx ← INTERV(1,m+n,(m+n)/2,x,y);
  para j de 1 passo 1 até m faça
  | para i de j passo 1 até n faça cc[i] ← cd[j,i];
    mtil ← m-j+1;
    dx[j] ← BSX2(mtil,n,y,cc,x,ℓx) | |
```

Codificação em ALGOL

```
%=============================================================================
  PROCEDURE BSDX2(M,N,Y,C,X,DX);
  INTEGER M,N; REAL X; REAL ARRAY Y[*],C[*],DX[*];
%-----------------------------------------------------------------------------
% UTILIZA BSMCD,INTERV(LBISEC),BSX2
%-----------------------------------------------------------------------------
  BEGIN
  INTEGER I,J,MTIL,LX; REAL ARRAY CD[1:M,1:N], CC[1:N];

  BSMCD(M,N,Y,C,CD);
  IF X=Y[N+1] THEN LX:=N ELSE LX:=INTERV(1,N+M,(N+M)/2,X,Y);
  FOR J:=1 STEP 1 UNTIL M DO
   BEGIN
   FOR I:=J STEP 1 UNTIL N DO CC[I]:=CD[J,I];
   MTIL:=M-J+1;
  DX[J]:=BSX2(MTIL,N,Y,CC,X,LX)
  END; END;
```

Cap.0-B - Avalia integral definida de s(x) a partir de sua expansão em B-splines.

Dados

m = ordem dos polinômios por partes

n = dimensão de $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$

$\{y_i\}_1^{m+n}$ nós da partição estendida $\bar{\Delta}$ associada com $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$

$\{c_i\}_1^n$ coeficientes da expansão em B-splines de $s \in \mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$

ei extremo inferior de integração ($a \leq ei < b$)

es extremo superior de integração ($ei < es \leq b$)

determina

$\int_{ei}^{es} s(t)dt$

Função Real

BSID(m,n,y,c,ei,es)

Entrada

m,n,y[1],..., y[m+n], c[1],..., c[n], ei, es

Os dimensionamentos das matrizes são fornecidos pelo programa principal.

Saída

BSID $\leftarrow \int_{ei}^{es} s(t)dt$

Método

Inicialmente determina os coeficientes $\{ci_j\}_{j=1}^n$ da expansão em B-splines da integral indefinida $\int_{y_1}^{x} s(t)dt$, com as funções B-splines definidas em nós da partição estendida associada a $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$

Em seguida calcula

$Es = \int_{y_1}^{es} s(t)dt = \sum_{j=1}^{n} ci_j N_j^{m+1}(es)$, $EI = \int_{y_1}^{ei} s(t)dt = \sum_{j=1}^{n} ci_j N_j^{m+1}(ei)$

e, finalmente, $\int_{ei}^{es} s(t)dt = (ES-EI)/m$. A divisão por n é uma correção dos valores de ci_j, j=1,...,n+1, determinadas em BSMCI a menos da constante m.

Procedimentos Auxiliares

BSMCI - BSX2(INTERV(LBISEC))

Cap.0-B - Avalia integral definida de s(x) a partir de sua expansão em B-splines.

Multiplicações ou Divisões

Avaliando-se EI e ES através da função real BSX2 e excluindo-se as operações para determinação da matriz ci teremos 3(m+1)m multiplicações ou divisões.

Algoritmo

```
função real BSID(m,n,y,c,ei,es);
inteiro m,n; real y[*], c[*]; real ei, es;
⌈ inteiro ℓx; real iei, ies; real ci[1:n+1];
  BSMCI(m,n,y,c,ci);
  para ℓx de 1 passo 1 até n faça ci[ℓx] ← ci[ℓx+1];
  ℓx ← (m+n)/2;
  iei ← BSX2(m+1,n,y,ci,ei,ℓx);
  ies ← BSX2(m+1,n,y,ci,es,ℓx);
  BSID ← (ies - iei)/m ⌋
```

Codificação em ALGOL

```
%=============================================================================
 REAL PROCEDURE BSID(M,N,Y,C,EI,ES);
 INTEGER M,N; REAL ARRAY Y[*],C[*] ; REAL  EI, ES ;
%-----------------------------------------------------------------------------
%----UTILIZA BSMCI,BSX2(INTERV(LBISEC))
%-----------------------------------------------------------------------------
 BEGIN

 INTEGER LX; REAL IEI, IES; REAL ARRAY CI[1:N+1];
 BSMCI(M,N,Y,C,CI);
 FOR LX:=1 STEP 1 UNTIL N DO CI[LX]:=CI[LX+1]; LX:=(M+N)/2;
 IEI:=BSX2(M+1,N,Y,CI,EI,LX);
 IES:=BSX2(M+1,N,Y,CI,ES,LX);
 BSID :=(IES-IEI)/M
 END;
```

Cap.0-B - Avalia integral definida de s(x) através de sua expansão em B-splines e partição uniforme de passo h.

Dados

Os extremos do intervalo [a,b]

k = número de nós da partição uniforme associada com $\mathcal{S}_{m(\Delta)}$

m = ordem dos polinômios por partes

$\{c_i\}_1^{m+k}$ coeficientes da expansão em B-splines de s $\in \mathcal{S}_{m(\Delta)}$

ei extremo inferior de integração, $a \leq ei < b$

es extremo superior de integração, $a \leq ei < es \leq b$

Determina

O valor de $\int_{ei}^{es} s(t)dt$ e armazena-o em BSIDH

Função Real

BSIDH(a,b,k,m,c,ei,es)

Entrada

a,b,k,m,c[1],...,c[m+k],ei,es, com dimensionamento previsto no programa principal

Saída

BSID = $\int_{ei}^{es} s(t)dt$

Método

Utiliza a mesma abordagem considerada no caso geral, porém substituindo BSXH por BSX2H. A execução de BSX2H com $\bar{m} = m+1$ automaticamente gera a partição estendida associada a $\mathcal{S}_{m+1(\Delta)}$.

Inicialmente calculam-se os coeficientes de $\int_{y_1}^{x} s(t)dt$ através de BSMCIH e, em seguida, avalia-se $\int_{ei}^{es} s(t)dt = \frac{1}{m}\left[\int_{y_1}^{es} s(t)dt - \int_{y_1}^{ei} s(t)dt\right]$. A divisão por m é uma correção aos valores de ci, i=1,...,n+1 determinados em BSHCIH a menos da constante m.

Procedimentos Auxiliares

BSMCIH e BSX2H

Cap.0-B - Avalia integral definida de s(x) através de sua expansão em B-splines e partição uniforme de passo h.

Multiplicações ou Divisões

2m(m-1)+5, não considerando as operações para cálculo de ci

Comparação com procedimento equivalente

m	BSID	BSIDH
	3m(m-1)	2m(m-1)+5
2	6	9
3	18	17
4	36	29
5	60	45
6	90	65

Algoritmo

```
função real  BSIDH(a,b,k,m,c,ei,es);
inteiro k,m; real a,b,ei,es; real c[*];
┌ inteiro  n,i; real iei, ies;
  n ← m+k;
  ┌┌ real ci[1:n+1];
     BSMCIH(a,b,k,m,c,ci);
     iei ← BSX2H(a,b,k,m+1,ci,ei);
     ies ← BSX2H(a,b,k,m+1,ci,es),
     BSIDH ← (ies-iei)/m ┘┘;
```

Codificação em ALGOL

```
%=============================================================================
 REAL PROCEDURE  BSIDH (A,B,K,M,C,EI,ES) ;
 INTEGER K,M;  REAL A,B,EI,ES;  REAL ARRAY C[*];
%-----------------------------------------------------------------------------
%-----------------------------------------------------------------------------
 BEGIN INTEGER N,I; REAL IEI,IES;

       N:= M+K ;
 BEGIN REAL ARRAY CI[1:N+1];

       BSMCIH(A,B,K,M,C,CI);
       IEI:= BSX2H(A,B,K,M+1,CI,EI) ;
       IES:= BSX2H(A,B,K,M+1,CI,ES) ;
       BSIDH:= (IES-IEI)/M
 END;  END;
```

Cap.0-B - Determina coeficientes das expansão em B-splines das derivadas de s(x).

Dados

m = ordem dos polinômios por partes

n = dimensão do espaço $\mathcal{S}(\mathcal{P}^m;\mathcal{M};\Delta)$

$\{y_i\}_1^{m+n}$ nós da partição estendida $\tilde{\Delta}$ associada com $\mathcal{S}(\mathcal{P}^m;\mathcal{M};\Delta)$

$\{c_i\}_1^n$ coeficientes da expansão em B-splines do spline s.

determina

A matriz $\{cd(j,i)\}_{j=1,i=j}^{m\quad n}$ onde a j-ésima linha contém os coeficientes da expansão em B-splines de $D_+^{j-1}s(x)$, considerando as funções B-splines definidas em nós de partição estendida associada a $\mathcal{S}(\mathcal{P}^m;\mathcal{M};\Delta)$.

Procedimento

BSMCD(m,n,y,c,cd)

Entrada

m, n, y[1],..., y[m+n], c[1],..., c[n]

Os dimensionamentos das matrizes são propiciados pelo programa principal.

Saída

cd[1,1],..., cd[1,n], cd[2,2],..., cd[2,n],...,cd[m,m],..., cd[m,n].

O dimensionamento da matriz bidimensional cd deve ser previsto no programa principal.

Método

$$D_+^{j-1}s(x) = \sum_{i=j}^{n} cd(j,i)N_i^{m-j+1}(x),\quad j=1,\ldots,m,\quad i=j,\ldots,n$$

$$cd(1,i) = c(i),\quad i=1,\ldots,n$$

$$cd(1,i) = \begin{cases} 0 \text{ se denom} = y_{i+m-j+1} - y_i = 0 \\ (m-j+1)\,\dfrac{cd(j-1,i)-cd(j-1,i-1)}{y_{i+m-j+1}-y_i}, \text{ em caso contrário} \end{cases}$$

$$j=1,2,\ldots,m,\quad i=1,\ldots,n$$

Cap.0-B - Determina coeficientes das expansão em B-splines das derivadas de s(x).

Procedimentos Auxiliares

Nenhum

Multiplicações ou Divisões

(m-1)(2n-m)

Algoritmo

```
procedimento BSMCD(m,n,y,c,cd);
inteiro m,n; real y[*], c[*], cd[*,*] ;
 inteiro i,j,mj; real denom;
  para i de 1 passo 1 até n faça cd[1,i] ← c[i] ;
  se m ≥ 2 então para j de 2 passo 1 até m faça
              mj ← m-j+1;
                 para i de n passo -1 até j faça
                 denom ← y[i+mj] - y[i] ;
                    se denom = 0 então cd[j,i] ← 0
                                 senão cd[j,i] ← cd[j-1,i] - cd[j-1,i-1] ;
                                       cd[j,i] ← cd[j,i]* mj/denom
```

Codificação em ALGOL

```
%=============================================================================
 PROCEDURE BSMCD(M,N,Y,C,CD);
  INTEGER N,M; REAL ARRAY C[*],CD[*,*],Y[*];
%-----------------------------------------------------------------------------
  BEGIN

  INTEGER I,J,MJ; REAL DENOM;
  FOR I:=1 STEP 1 UNTIL N DO CD[1,I]:=C[I]; IF M >= 2 THEN
  FOR J:=2 STEP 1 UNTIL M DO BEGIN
    MJ:=M-J+1;
    FOR I:=N STEP -1 UNTIL J DO BEGIN
      DENOM:=Y[I+MJ]-Y[I];
      IF DENOM = 0 THEN CD[J,I]:=0
                   ELSE BEGIN CD[J,I]:=CD[J-1,I]-CD[J-1,I-1];
                              CD[J,I]:=MJ*CD[J,I]/DENOM  END;

      END;
    END
  END;
```

Cap.0-B - Determina os coeficientes da expansão em B-splines das derivadas de s(x), considerando partição uniforme de passo h.

Dados

Os extremos do intervalo I = [a,b]

k = número de nós da partição uniforme associada com $\mathcal{S}_{m(\Delta)}$

m = ordem dos polinômios por partes

$\{c_i\}_1^{m+k}$ coeficientes da expansão em B-splines normalizados de $s \in \mathcal{S}_{m(\Delta)}$

determina

Os coeficientes de $D_+^{j-1}s(x) = \sum_{i=j}^{n} cd(j,i)N^{m-j+1}\left(\frac{x - y_i}{h}\right)$ obtendo a matriz $\{cd(j,i)\}_{j=1,i=1}^{m\quad n}$.

Procedimento

BSMCDH(a,b,k,m,c,cd)

Entrada

a,b,k,m,c[1],..., c[m+k]

O dimensionamento da matriz deve ser fornecido pelo programa principal

Saída

cd[1,1],...,cd[1,n],cd[2,2],...,cd[2,n],...,cd[m,m],...,cd[m,n]

O dimensionamento da matriz deve ser previsto no programa principal.

Método

A partir de n=m+k, h=(b-a)/(k+1), $s(x) = \sum_{i=1}^{n} c_i N^m\left(\frac{x - y_i}{h}\right)$ e da relação recursiva $D_+N^m(x) = N^{m-1}(x) - N^{m-1}(x-1)$ obtem-se:

$D_+^j s(x) = h^{-j} \sum_{i=j}^{n} \nabla^j c_i N^{m-j}\left(\frac{x-y_i}{h}\right)$ ou $D_+^{j-1}s(x) = \sum_{i=j}^{n} \frac{cd(j,i)}{h^{j-1}} N^{m-j+1}\left(\frac{x-y_i}{h}\right)$

A matriz $\{cd(j,i)\}_{j=1,i=1}^{m\quad n}$ é calculada:

(i) definindo-se cd(1,i)=c(i), i=1,...,n

(ii) construindo-se a respectiva tabela de diferenças ∇ onde

$\nabla cd(j,i) = cd(j-1,i) - cd(j-1,i-1)$

(iii) multiplicando-se cada cd(j,i) por h^{j-1}.

Cap.0-B - Determina os coeficientes da expansão em B-splines das derivadas de s(x), considerando partição uniforme de passo h.

Procedimentos Auxiliares

Nenhum

Multiplicações ou Divisões

$\frac{(m-1)(2n-m)}{2}$, excluindo-se o cálculo de h.

Se a matriz cd for utilizada somente para avaliação das derivadas sucessivas de s(x) pode-se omitir as divisões por h, as quais serão posteriormente introduzidas no algoritmo que avalia as derivadas, efetuando-se para cada j uma única divisão, j=2,...,m. Com esta modificação eliminam-se todas multiplicações ou divisões de BSMCDH.

Entretanto se compararmos com BSMCD, mesmo conservando-se as divisões por h, o procedimento BSMCDH envolve apenas a metade das operações executadas em BSMCD.

Algoritmo

```
procedimento  BSMCDH(a,b,k,m,c,cd);
inteiro k,m; real a,b; real c[*], cd[*,*];
| inteiro i,j,n; real h;
  n ← m+k;
  h ← (b-a)/(k+1);
  para i de 1 passo 1 até n faça cd[1,i] ← c[i];
  para j de 2 passo 1 até m faça
  para i de j passo 1 até n faça
  cd[j,i] ← (cd[j-1,i] - cd[j-1,i-1])/h|
```

Codificação em ALGOL

```
%=============================================================================
 PROCEDURE BSMCDH(A,B,K,M,C,CD);
 INTEGER M,K;  REAL A,B ;  REAL ARRAY C[*],CD[*,*];
%-----------------------------------------------------------------------------
 BEGIN INTEGER I,J,N; REAL H;

       N:=M+K;
       H:=(B-A)/(K+1);
       FOR I:=1 STEP 1 UNTIL N DO  CD[1,I]:= C[I] ;
       FOR J:=2 STEP 1 UNTIL M DO
       FOR I:=J STEP 1 UNTIL N DO  CD[J,I]:= ( CD[J-1,I]-CD[J-1,I-1] )/H
 END;
```

Cap.0-B - Determina os coeficientes da expansão em B-splines da integral indefinida de s(x).

Dados

m = ordem dos polinômios por partes

n = dimensão do espaço $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$

$\{y_i\}_1^{m+n}$ nós da partição estendida $\bar{\Delta}$ associada com $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$

$\{c_i\}_1^n$ coeficientes da expansão em B-splines do spline s

determina

$\{mci_i\}_1^{n+1}$ onde $\{ci_i\}_1^{n+1}$ são os coeficientes da expansão em B-splines de $\int_{y_1}^{x} s(t)dt$ considerando-se as funções B-splines definidas em nós da partição estendida associada com $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$

Procedimento

BSMCI(m,n,y,c,ci)

Entrada

m, n, y[1],..., y[m+n], c[1],..., c[n]

Os dimensionamentos das matrizes são fornecidos pelo programa principal.

Saída

ci[1],..., ci[n+1] contendo, respectivamente, os valores de $mci_1,\ldots,mci_{n+1}$.

O dimensionamento da matriz deve ser previsto no programa principal.

Método

$D_{y_1}^{-1}s(x) = \int_{y_1}^{x} s(t)dt = \sum_{i=0}^{n} ci_i N_i^{m+1}(x)$, $ci_0=0$,

Dado $\{c_i\}_1^n$ calculam-se $ci_i = \sum_{j=1}^{i} \frac{y_{i+j} - y_j}{m} c_j$, $i=1,\ldots,n$

A fim de economizar divisões são calculados $mci_1,\ldots,mci_n$, os quais são armazenados, respectivamente, em ci[2],..., ci[n+1], definindo-se ci[1]= 0.

Procedimentos Auxiliares

Nenhum

Cap.0-B - Determina os coeficientes da expansão em B-splines da integral indefinida de s(x).

Multiplicações ou Divisões

n

Algoritmo

```
procedimento  BSMCI(m,n,y,c,ci);
inteiro m,n; real y[*], c[*], ci[*];
|| inteiro i ; real s;
   s ← 0;
   para i de 1 passo 1 até n faça
   || s ← s + (y[i+m] - y[i]) * c[i];
      ci[i] ← s || ;
   para i de n+1 passo -1 até 2 faça  ci[i] ← ci[i-1] ;
   ci[1] ← 0 |
```

Codificação em ALGOL

```
%=============================================================================
 PROCEDURE BSMCI(M,N,Y,C,CI);
 INTEGER M,N; REAL ARRAY Y[*],C[*],CI[*];
%-----------------------------------------------------------------------------
 BEGIN

 REAL S; INTEGER I;
 S:=0; FOR I:=1 STEP 1 UNTIL N DO BEGIN S:=S+(Y[I+M]-Y[I])*C[I];CI[I]:=S
                                   END;
 FOR I:=N+1 STEP -1 UNTIL 2 DO CI[I]:=CI[I-1]; CI[1]:=0                 END;
```

Cap.0-B - Determina os coeficientes da expasão em B-splines da integral indefinida de s(x) considerando partição uniforme de passo h.

Dados

Os extremos do intervalo $I = [a,b]$

k = número de nós da partição uniforme associada com $\mathcal{S}_{m(\Delta h)}$

m = ordem dos polinômios por partes

$\{c_i\}_1^{m+k}$ coeficientes da expansão em B-splines de $s \in \mathcal{S}_{m(\Delta h)}$

determina

$\{ci_i\}_1^{m+k+1}$, coeficientes da expansão em B-splines de $\int_{y_1}^{x} s(t)dt$

Procedimento

BSMCIH(a,b,k,m,c,ci)

Entrada

a,b,k,m, c[1],..., c[m+k], com dimensionamento dado pelo programa principal.

Saída

ci[1], ci[2],..., ci[n+1] com dimensão prevista no programa principal.

Método

Utiliza a mesma versão considerada no caso geral (BSMCI).

Gera a partição estendida associada com $\mathcal{S}_{m(\Delta h)}$ e calcula os coeficientes $\{ci_i\}_1^{n+1}$ tal que $\int_{y_1}^{x} s(t)dt = \sum_{i=0}^{n} ci_i N^{m+1}\left(\frac{x - y_i}{h}\right)$ por meio de $ci_0=0$

$ci_i = \sum_{j=1}^{i} \frac{(y_{i+j} - y_j)}{m} c_j, \quad i=1,\ldots,n.$

A fim de economizar operações são calculados $\{mci_i\}_0^n$ e armazenados respectivamente em ci[1], ci[2],..., ci[m+1] , definindo-se ci[1] = 0, tendo em vista que o coeficiente $ci_0 = 0$.

Procedimentos Auxiliares

nenhum

Cap.0-B - Determina os coeficientes da expansão em B-splines da integral indefinida de s(x) considerando partição uniforme de passo h.

Multiplicações ou Divisões

n, excluindo-se as operações para cálculo de h e da matriz $\{y_i\}_1^{n+m}$

Algoritmo

```
procedimento  BSMCIH(a,b,k,m,c,ci);
inteiro k,m; real a,b ; real c[*], ci[*]
| inteiro n,i ; real s,h ;
  n ← m+k ;
  || real y[1:n+m] ;
     h ← (b-a)/(k+1) ;
     para i de 1 passo 1 até 'n+m' faça
     y[i] ← a + (i-m) * h ;
     s ← 0 ;
     para i de 1 passo 1 até n faça
     ||| s ← s + (y[i+m] - y[i]) * c[i] ;
         ci[i] ← s ||| ;
     para i de n+1 passo -1 até 2 faça ci[i] ← ci[i-1] ;
     ci[1] ← 0 || ;|
```

Codificação em ALGOL

```
%=============================================================================
 PROCEDURE BSMCIH(A,B,K,M,C,CI);
 INTEGER M,K; REAL  A,B;     REAL ARRAY  C[*], CI[*] ;
%-----------------------------------------------------------------------------
 BEGIN INTEGER N,I; REAL S,H;

       N:=M+K ;  H:=(B-A)/(K+1) ;
 BEGIN REAL ARRAY Y[1:N+M];

       FOR I:=1  STEP  1  UNTIL  N+M  DO  Y[I]:= A + (I-M) * H  ;
       S:=0;
       FOR I:=1  STEP  1  UNTIL   N   DO
 BEGIN      S:=S + (Y[I+M]-Y[I])*C[I]; CI[I]:=S
 END;
       FOR I:=N+1 STEP -1 UNTIL 2 DO  CI[I]:= CI[I-1] ;
       CI[1]:=0;
 END; END;
```

Cap.0-B - Determina a matriz de GRAM

Dados

m = ordem dos polinômios por partes

n = dimensão do espaço $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$

$\{y_i\}_1^{m+n}$ nós da partição estendida $\tilde{\Delta}$ associada com $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$

$\{w_i\}_1^m$ valores positivos retirados da tabela correspondente para a fórmula de integração numérica de Gauss.

$\{z_i\}_1^m$ m valores entre -1 e 1 simétricos em relação a origem, retirados da tabela correspondente para a fórmula de integração numérica de Gauss.

determina

$\{G_{ij}\}_{i=1,j=1}^{n,n}$ onde $G_{ij} := < N_i^m(x), N_j^m(x) > = \int_{y_i}^{y_{i+m}} N_i^m(x) N_j^m(x) dx$

Procedimento

BSMGRM(m,n,y,w,z,G)

Entrada

m, n, y[1],..., y[m+n], w[1],..., w[m], z[1],..., z[m]

Os dimensionamentos das matrizes são fornecidos pelo programa principal.

Saída

G[1,1],..., G[1,n], G[2,1],..., G[2,n], ..., G[n,1],..., G[n,n]

O dimensionamento da matriz deve ser previsto no programa principal.

Método

Como G é uma matriz de banda 2m-1, inicialmente zeramos todos seus elementos e, em seguida, avaliamos as integrais para cada i e $i-m+1 \leq jj \leq i+m-1$ através da fórmula de integração de Gauss:

$$G_{ij} = \int_{y_i}^{y_{i+m}} N_i^m(x) N_j^m(x) dx = \sum_{\nu=i}^{i+m-1} \int_{y_\nu}^{y_{\nu+1}} N_i^m(x) N_j^m(x) dx = \sum_{\nu=i}^{i+m-1} \frac{(y_{\nu+1}-y_\nu)}{2} \sum_{u=1}^{m} w_u N_i^m(t_{u\nu}) N_j^m(t_{u\nu})$$

$$\text{com } t_{u\nu} = \frac{(y_{\nu+1} - y_\nu)}{2} z_u + \frac{(y_{\nu+1} + y_\nu)}{2}$$

Cap.0-B - Determina a matriz de GRAM

A fim de minimizar o número de operações para cada $m \leq v \leq n$ tal que $y_{v+1} - y_v > 0$ calculam-se os B-splines não nulos e, logo após, para todo i e j verificam-se quais $N_i^m(t_{uv})$ e $N_j^m(t_{uv})$ são não nulos, adicionando em G_{ij} e parcela $\frac{(y_{v+1} - y_v)}{2} w_u N_i^m(t_{uv}) N_j^m(t_{uv})$.

Procedimentos Auxiliares

BSMQ

Multiplicações ou Divisões

$(\frac{n-m+1}{2})(9m^3 + m^2 + 2) \sim nm^3$

Algoritmo

```
procedimento BSMGR(m,n,y,w,z,G) ;
inteiro m,n; real y[*], w[*], z[*], G[*,*] ;
| inteiro i,j,ci,u,lx; real d,t; real Q[1:m+1];
  para i de 1 passo 1 até n faça
  para j de 1 passo 1 até n faça G[i,j] ← 0 ;
  para ci de m passo 1 até n faça
  se y[ci+1] - y[ci] > 0 então
  || d ← (y[ci+1] - y[ci])/2;
     para u de 1 passo 1 até m faça
     ||| t ← d * z[u] + (y[ci+1] + y[ci])/2 ;
         BSMQ(m,ci,y,t,Q) ;
         para i de 1 passo 1 até m faça
         para j de 1 passo 1 até m faça
         G[ci-m+i,ci-m+j] ← G[ci-m+i,ci-m+j]+d * w[u] * Q[i] * Q[j] ||| || |
```

Cap.0-B - Determina a matriz de GRAM

Codificação em ALGOL

```
%==========================================================================
 PROCEDURE BSMGRM(M,N,Y,W,Z,G);
 INTEGER M,N; REAL ARRAY Y[*],W[*],Z[*],G[*,*];
%--------------------------------------------------------------------------
%    UTILIZA INTERV-LBISEC  E  BSMQ
%--------------------------------------------------------------------------
 BEGIN
  INTEGER I,J,CI,U,LX; REAL D,T;  REAL ARRAY Q[1:M+1];

  FOR I:=1 STEP 1 UNTIL N DO FOR J:=1 STEP 1 UNTIL N DO G[I,J]:=0;
  FOR CI:=M STEP 1 UNTIL N DO IF Y[CI+1]-Y[CI] > 0 THEN
  BEGIN D:=(Y[CI+1]-Y[CI])/2;
        FOR U:=1 STEP 1 UNTIL M DO
        BEGIN T:=D*Z[U]+(Y[CI+1]+Y[CI])/2;  BSMQ(M,CI,Y,T,Q);
              FOR I:=1 STEP 1 UNTIL M DO FOR J:=1 STEP 1 UNTIL M DO
              G[CI-M+I,CI-M+J]:=G[CI-M+I,CI-M+J]+D*W[U]*Q[I]*Q[J]
        END;
  END;
 END;
```

Cap.0-B - Avalia $\{N_i^m\}_{\ell+1-m}^{\ell}$

Dados

m = ordem dos polinômios por parte

$\{y_i\}_1^{m+n}$ nós da partição estendida $\tilde{\Delta}$ associada com $\mathcal{S}(\mathcal{P}^m; \mathcal{M}; \Delta)$

x ponto de $[a,b]$

ℓ índice tal que $y_\ell \leq x < y_{\ell+1}$

determina

$\{N_i^m(x)\}_{\ell+1-m}^{\ell}$ armazenando-os respectivamente em $\{Q_i\}_1^m$

Procedimento

BSMQ(m,ℓ,y,x,Q)

Entrada

m, ℓ, y[1],..., y[m+n], x, com dimensão da matriz fornecida pelo programa principal.

Saída

Q[1],..., Q[m], com dimensionamento m+1 previsto no programa principal.

Método

Constrói-se a matriz triangular Q a partir de $Q_\ell^1(x) \leftarrow 1/(y_{\ell+1} - y_\ell)$ e das relações recursivas $Q^j(x) = [(x - y_i)Q_i^{j-1}(x) + (y_{i+j} - x)Q_{i+1}^{j-1}(x)]/(y_{i+j}-y_i)$ e $N_i^m(x) = (x - y_i)Q_i^{m-1}(x) + (y_{i+m} - x)Q_{i+1}^{m-1}(x)$.

A matriz triangular Q, cuja última linha contém os B-splines normalizados de ordem m, é gerada linha após linha, conservando-se de cada vez apenas os valores da última linha calculada, motivo pelo qual Q é uma matriz unidimensional.

O algoritmo para geração de Q baseado nas relações recursivas é extremamente estável porque envolve somente combinações convexas de quantidades não negativas.

Se $x = y_{\ell+1}$ são geradas $N_{\ell+1-m}^m(y_{\ell+1}^-),\ldots,N_\ell^m(y_{\ell+1}^-)$

Cap.0-B - Avalia $\{N_i^m\}_{\ell+1-m}^{\ell}$

Procedimentos Auxiliares

Nenhum

Multiplicações ou Divisões

$(3m^2 + m-4)/2$

Algoritmo

```
procedimento BSMQ(m,ℓ,y,x,Q) ;
inteiro m,ℓ; real x ; real y[*], Q[*] ;
┌ inteiro  i,j ; real denom, a1, a2;
  para j de 1 passo 1 até m-1 faça Q[j] ← 0 ;
  Q[m] ← 1/(y[ℓ+1] - y[ℓ]); Q[m+1] ← 0 ;
  se m-1 ≥ 2 então para j de 2 passo 1 até m-1 faça
                ┌┌ para i de m-j+1 passo 1 até m faça
                  ┌┌┌ denom ← y[i+ℓ-m+j] - y[i+ℓ-m] ;
                     a1 ← (x - y[i+ℓ-m])/denom ;
                     a2 ← 1-a1 ;
                     Q[i] ← a1 * Q[i] + a2 * Q[i+1] └┘└┘;
  para i de 1 passo 1 até m faça
     Q[i] ← (x-y[i+ℓ-m]) * Q[i] + (y[i+ℓ] - x) * Q[i+1] ┘
```

Codificação em ALGOL

```
%=============================================================================:
 PROCEDURE BSMQ(M,L,Y,X,Q);
   REAL X; INTEGER L,M; REAL ARRAY Y[*], Q[*];
%-----------------------------------------------------------------------------
   BEGIN
   INTEGER I,J; REAL DENOM,A1,A2;

   FOR J:=1  STEP  1  UNTIL  M-1  DO Q[J]:=0;
   Q[M]:=1/(Y[L+1]-Y[L]); Q[M+1]:=0; IF M-1 >= 2 THEN
   FOR J:=2  STEP  1  UNTIL  M-1  DO
     BEGIN
     FOR I:= M-J+1  STEP  1  UNTIL  M  DO
       BEGIN
       DENOM:=Y[I+L-M+J]-Y[I+L-M]; A1:=(X-Y[I+L-M])/DENOM;
       A2:=1-A1;  Q[I]:= A1*Q[I] + A2*Q[I+1]
       END;
     END;
   FOR I:=1  STEP  1  UNTIL  M  DO
     Q[I]:=(X-Y[I+L-M])*Q[I] + (Y[I+L]-X)*Q[I+1]
   END;
```

Cap.0-B - Avalia $\{N_i^m(x)\}_{\ell+1-m}^{\ell}$ considerando partição uniforme de passo h.

Dados

Os extremos do intervalo $I = [a,b]$

k = número de nós da partição uniforme associada com $\mathcal{S}_{m(\Delta h)}$

m = ordem dos polinômios por partes

x ponto de $[a,b]$

ℓ índice do nó da partição estendida associada com $\mathcal{S}_{m(\Delta h)}$ tal que

$y_\ell \leq x < y_{\ell+1}$

calcula

$\{N^m_{\ell+i-m}\}_{i=1}^{m}$ armazenando-os, respectivamente, em $\{Q_i\}_1^m$

Procedimento

BSMQH(a,b,k,m,ℓ,x,Q)

Entrada

a,b,k,m,ℓ,x

Saída

Q[1], Q[2],..., Q[m], sendo o dimensionamento de m+1 previsto no programa principal.

Método

Versão simplificada do algoritmo utilizado em BSMQ.

São utilizadas as relações recursivas:

$Q^m(x) = (xQ^{m-1}(x) + (m-x)Q^{m-1}(x))/m$

$N^m(x) = xQ^{m-1}(x) + (m-x)Q^{m-1}(x)$

Procedimentos Auxiliares

Nenhum

Multiplicações ou Divisões

$m^2 + 2m - 1$, excluidos os cálculos de h e fat.

Cap.0-B - Avalia $\{N_i^m(x)\}_{\ell+1-m}^{\ell}$ considerando partição uniforme de passo h.

Se BSMQH for utilizado apenas para avaliação de s(x), o algoritmo pode ser modificado excluindo-se Q[i]/fat e multiplicando-se em BSXH a soma que gera s(x) por fat = (m-1)!

Comparação com procedimento equivalente

m	BSMQ	BSMQH	BSMQH excluidos Q[i]/fat
	$(3m^2+m-4)/2$	m^2+2m-1	m^2+m-1
2	5	7	5
3	13	14	11
4	24	23	19
5	38	34	29
6	55	47	41

Algoritmo

```
procedimento BSMQH(a,b,k,m,ℓ,x,Q) ;
inteiro k,m,ℓ; real a,b,x; real Q[*] ;
┌ inteiro i,j; real xdh, fat, h ;
  para j de 1 passo 1 até m-1 faça Q[j] ← 0 ;
  Q[m] ← 1 ; Q[m+1] ← 0 ;
  fat ← 1 ;
  h ← (b-a)/(k+1) ;
  xdh ← (x-a)/h-ℓ+m ;
  para j de 2 passo 1 até m faça
  ┌┌ fat ← fat * (j-1) ;
    para i de m-j+1 passo 1 até m faça
    Q[i] ← (xdh-i+m) * Q[i] + (i+j-m-xdh) * Q[i+1] ┘┘;
    para i de 1 passo 1 até m faça
    Q[i] ←Q[i]/fat ┘
```

Codificação em ALGOL

```
%=============================================================================
 PROCEDURE BSMQH(A,B,K,M,L,X,Q);
 INTEGER M,L,K;REAL A,B,X;     REAL ARRAY Q[*];
%-----------------------------------------------------------------------------
 BEGIN INTEGER I,J; REAL XDH, FAT,H;

       FOR J:=1 STEP 1 UNTIL M-1 DO  Q[J]:=0;    Q[M]:=1;  Q[M+1]:=0;
     FAT:=1; H:=(B-A)/(K+1);  XDH:=(X-A)/H-L+M;
       FOR J:=2 STEP 1 UNTIL  M  DO
 BEGIN      FAT:= FAT*(J-1);
            FOR I:=M-J+1 STEP 1 UNTIL M DO
                Q[I]:=(XDH-I+M)*Q[I]+(I+J-M-XDH)*Q[I+1];
 END;
       FOR I:=1 STEP 1 UNTIL M DO  Q[I]:= Q[I]/FAT
 END;
```

Cap.0-B - Conversão da expansão em B-splines para a Representação Polinomial por Partes.

Dados

m = ordem dos polinômios por partes

n = dimensão do espaço $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$

$\{y_i\}_{i=1}^{m+n}$ nós da partição estendida $\tilde{\Delta}$ associada com $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$

$\{c_i\}_{i=1}^{n}$ coeficientes da expansão em B-splines do spline s

calcula

k número de nós distintos da partição Δ,

$\{x_i\}_1^k$ nós da partição Δ

e os coeficientes $\{cw_{ij}\}_{i=0,j=1}^{k\quad m}$ dos polinômios $P_0(x), P_1(x), \ldots, P_k(x)$

que representam o spline s em cada subintervalo $I_0=[a,x_1), \ldots, I_{k-1}=[x_{k-1},x_k)$

e $I_k = [x_k, b]$: $P_0(x) = \sum_{j=1}^{m} cw_{0j}(x-x_1)^{j-1}$, $x \in I_0$

$$P_i(x) = \sum_{j=1}^{m} cw_{ij}(x-x_i)^{j-1},\quad x \in I_i,\quad i=1,\ldots,k$$

armazenando-os, respectivamente, em $\{cp(i,j)\}_{i=0,j=1}^{k\quad m}$

---

procedimento

BSRPP(m,n,y,c,k,x,cp)

Entrada

m,n,y[1],..., y[m+n], c[1],..., c[n]

Os dimensionamentos das matrizes são fornecidos pelo programa principal.

Saída

k, x[1], x[2],..., x[k],cp[0,1],..., cp[0,m], cp[1,1],..., cp[1,m],...,
, cp[k,1],..., cp[k,m]

Os dimensionamentos das matrizes devem ser previstos no programa principal.

Método

A partir de $s(x) = \sum_{j=1}^{m} cw_{ij}(x-x_i)^{j-1}$ para $x \in I_i$ obtem-se

Cap.0-B - Conversão da expansão em B-splines para a Representação Polinomial por Partes.

$cw_{ij} = \dfrac{D_+^{j-1}s(x_i)}{(j-1)!}$, $i=1,2,\ldots,k$, $j=1,2,\ldots,m$ e $cw_{0j} = \dfrac{D_-^{j-1}s(x_1)}{(j-1)!}$, $j=1,2,\ldots,m$ ...

Os valores de $\{D_+^{j-1}s(x_i)\}_{i=1,j=1}^{k\quad m}$ são obtidos através do procedimento BSDX2, e os de $\{D_-^{j-1}s(x_1)\}_{j=1}^{m}$ por meio do mesmo algoritmo utilizado em BSDX2, porém aplicado no ponto $x = x_1$.

Procedimentos Auxiliares

INVPTE, BSMQ e BSDX2(BSMCD, INTERV(LBISEC))

Multiplicações ou Divisões

$\frac{k}{2}(m-1)m(m+1) + \frac{m}{2}(m^2+ 3m - 2)$,

não contando as operações envolvidas na determinação de fat, CD e divisões por fat.

Algoritmo

```
procedimento BSRPP(m,n,y,c,k,x,cp) ;
inteiro m,n,k; real y[*], c[*], x[*], cp[*,*];
| inteiro i,j,mtil; inteiro &m[1:n]; real a,b,BSDJSX ;
  real cd[1:m,1:n], fat[1:m], dx[1:m], Q[1:m+1] ;
  fat[1] ← 1 ;
  para i de 2 passo 1 até m faça fat[i] ← fat[i-1] * (i-1) ;
  INVPTE(m,n,y,k,a,b,&m,x) ;
  para i de 1 passo 1 até k faça
  || BSDX2(m,n,y,c,x[i], dx) ;
     para j de 1 passo 1 até m faça
     cp[i,j] ← dx[j]/fat[j] || ;
     BSMCD(m,n,y,c,cd) ;
     para j de 1 passo 1 até m faça
     || mtil ← m-j+1 ;
        BSMQ(mtil,m,y,x[1],Q) ;
        para i de mtil passo -1 até 1 faça Q[i+j-1] ← Q[i] ;
        BSDJSX ← 0 ;
        para i de j passo 1 até m faça BSDJSX ← BSDJSX + cd[j,i] * Q[i] ;
        cp[0,j] ← BSDJSX/fat[j] || ;|
```

Cap.0-B - Conversão da expansão em B-splines para a Representação Polinomial por Partes.

Codificação em ALGOL

```
%=============================================================================
 PROCEDURE BSRPP(M,N,Y,C,K,X,CP);
 INTEGER M,N,K; REAL ARRAY Y[*],C[*],X[*],CP[*,*];
%-----------------------------------------------------------------------------
% UTILIZA INVPTE, EXCX2, BSMCO, INTERV, LBISEC  E  BSMG
%-----------------------------------------------------------------------------
 BEGIN

 INTEGER I,J,MTIL; INTEGER ARRAY LM[1:M]; REAL A,E,BSCJSX;
 REAL ARRAY CO[1:M,1:N],FAT[1:M],DX[1:M], G[1:M+1];
 FAT[1]:=1; FOR J:=2 STEP 1 UNTIL M DO FAT[I]:=FAT[I-1]*(I-1);
 INVPTE(M,N,Y,K,A,E,LM,X);
 FOR I:=1 STEP 1 UNTIL K DO
  BEGIN BSDX2(M,N,Y,C,X[I],DX);
        FOR J:=1 STEP 1 UNTIL M DO CP[I,J]:=DX[J]/FAT[J] END;
 BSMCO(M,N,Y,C,CO);
 FOR J:=1 STEP 1 UNTIL M DO
  BEGIN MTIL:=M-J+1; BSMG(MTIL,M,Y,X[1],0);
        FOR I:=MTIL STEP -1 UNTIL 1 DO G[I+J-1]:=G[I];  BSCJSX:=0;
        FOR I:=J STEP 1 UNTIL M DO BSCJSX:=BSCJSX+CO[J,I]*G[I];
        CP[0,J]:=BSCJSX/FAT[J] END;
 END;
```

Cap.0-B - Conversão da Expansão em B-splines para Representação Polinomial por Partes, considerando partição uniforme de passo h.

Dados

Os extremos do intervalo $I = [a,b]$

k = número de nós da partição uniforme associada com $\mathcal{S}_m(\Delta h)$

m = ordem dos polinômios por partes

$\{c_i\}_1^{m+k}$ coeficientes da expansão em B-splines de $s \in \mathcal{S}_m(\Delta h)$

determina

$\{x_i\}_1^k$ nós da partição uniforme e $\{cp_{ij}\}_{i=0,j=1}^{k\quad m}$ coeficientes dos polinômios que representam s em cada subintervalo I_i, $i=0,\ldots,k$

Procedimento

BSRPPH(a,b,k,m,c,x,cp)

Entrada

a,b,k,m,c[1],..., c[m+k], com dimensionamento previsto no programa principal.

Saída

x[1],..., x[k], cp[0,1],..., cp[0,m],..., cp[k,1],..., cp[k,m]

As dimensões das matrizes devem ser fornecidas pelo programa principal.

Método

Inicialmente determina $\{x_i\}_1^k$ e para cada $i=1,\ldots,k$ calcula $s(x_i), D_+s(x_i), \ldots, D_+^{m-1}s(x_i)$ através de BSDXH; em seguida obtem

$$cp_{ij} = \frac{D_+^{j-1}s(x_i)}{(j-1)!}, \quad j=1,\ldots,m.$$

Finalmente, para determinar $\{D_-^{j-1}s(x_1)\}_1^m$ utiliza o mesmo algoritmo empregado em BSDXH, calculando a matriz cp através de BSMCDH e executando BSMQH (a,b,k,m,ℓx,Q) com $\ell = m-j+1$ e $x=x_1$ para $j=1,\ldots,m$.

Procedimentos Auxiliares

BSDXH(BSX2H), BSMCDH, BSMQH

Cap.0-B - Conversão da Expansão em B-splines para Representação Polinomial por Partes, considerando partição uniforme de passo h.

Multiplicações ou Divisões

$\frac{k}{3}(m^3+ 5m - 3) + \frac{m}{3}(m^2+ 6m + 2)$, não computando as operações para cálculo de h, fat, cp e divisões por fat[j].

Comparação com procedimento equivalente

m	BSRPP	BSRPPH
	$\frac{k}{2}(m-1)m(m+1) + \frac{m}{2}(m^2+3m-2)$	$\frac{k}{3}(m^3+5m-3) + \frac{m}{3}(m^2+6m+2)$
2	3k + 8	4k + 12
3	12k + 24	12k + 29
4	30k + 52	26k + 56
5	60k + 95	48k + 95
6	105k + 156	80k + 148

Algoritmo

```
procedimento BSRPPH(a,b,k,m,c,x,cp);
inteiro k,m; real a,b; real c[*], x[*], cp[*,*];
| inteiro i,j,mtil,n; real h,BSDJSX; real fat[1:m], dx[1:m], Q[1:m+1] ;
  n ← m+k ;
  || real cd[1:m,1:n] ;
     fat[1] ← 1;
     para i de 2 passo 1 até m faça fat[i] ← fat[i-1] * [i-1] ;
     h ← (b-a)/(k+1)
     para i de 1 passo 1 até k faça
     ||| x[i] ← a + i * h ;
         BSDXH(a,b,k,m,c,x[i], dx) ;
         para j de 1 passo 1 até m faça cp[i,j] ← dx[j]/fat[j] |||;
     BSMCDH(a,b,k,m,c,cd) ;
     para j de 1 passo 1 até m faça
     ||| mtil ← m-j+1 ;
         BSMQH(a,b,k,mtil,m,x[1],Q);
         BSDJSX ← 0 ;
         para i de m-mtil+1 passo 1 até m faça
         BSDJSX ← BSDJSX + cd[j,i] * Q[i+mtil-m] ;
         cp[0,j] ← BSDJSX/fat[j] ||| || |
```

Cap.0-B - Conversão da Expansão em B-splines para Representação Polinomial por Partes, considerando partição uniforme de passo h.

Codificação em ALGOL

```
%=================================================================
 PROCEDURE BSRPPH(A,B,K,M,C,X,CP);
 INTEGER M,K; REAL A,B; REAL ARRAY X[*], CP[*,*], C[*];
%-----------------------------------------------------------------
%     UTILIZA  BSDXH,  BSMCDH   E   BSMQH
%-----------------------------------------------------------------
 BEGIN INTEGER I,J,MTIL,N; REAL H,BSDJSX;

       REAL ARRAY FAT[1:M],DX[1:M],Q[1:M+1];
       N:=M+K;
 BEGIN REAL ARRAY CD[1:M,1:N];

       FAT[1]:=1;
       FOR I :=2  STEP  1  UNTIL M DO  FAT[I]:=FAT[I-1] * (I-1) ;
       H:=(B-A)/(K+1);
       FOR I :=1  STEP  1  UNTIL K DO
 BEGIN     X[I]:= A + I*H ;
           BSDXH(A,B,K,M,C,X[I],DX);
           FOR J:= 1  STEP  1  UNTIL  M  DO  CP[I,J]:=DX[J]/FAT[J]
 END;
       BSMCDH(A,B,K,M,C,CD);
       FOR J:=1  STEP  1  UNTIL  M  DO
 BEGIN     MTIL:= M-J+1 ;
           BSMQH(A,B,K,MTIL,MTIL,X[1],0) ;
           BSDJSX:= 0 ;
           FOR I := M-MTIL+1  STEP  1  UNTIL  M  DO
               BSDJSX:= BSDJSX + CD[J,I] * Q[I+MTIL-M];
           CP[0,J]:= BSDJSX/FAT[J]
 END; END; END;
```

Cap.0-B - Avalia s(x) a partir de sua expansão em B-splines.

Dados

m = ordem dos polinômios por partes.

n = dimensão do espaço $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$

$\{y_i\}_1^{m+n}$ nós da partição estendida $\tilde{\Delta}$ associada com $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$

$\{c_i\}_1^n$ coeficientes da expansão em B-splines do spline s.

x em [a,b]

ℓx estimativa de ℓ tal que $y_\ell \leq x < y_{\ell+1}$

determina

$$s(x) = \sum_{i=\ell+1-m}^{\ell} c_i N_i^m(x)$$

Função Real

BSX(m,n,y,c,x,ℓx) ;

Entrada

m, n, y[1],..., y[m+n], c[1],..., c[n], x e ℓx; os dimensionamentos das matrizes devem ser previstos no programa principal.

Saída

BSX = s(x)

Método

Inicialmente determina-se ℓ; se x=y[n+1] então ℓ=n senão ℓ=INTERV(1,m+n,ℓx,x,y) . Em seguida calcula $N_{\ell+1-m}^m(x),\ldots,N_\ell^m(x)$ através do procedimento BSMQ(m,ℓ,y,x,Q) e finalmente computa $s(x) = \sum_{i=\ell+1-m}^{\ell} c_i N_i^m(x) = \sum_{i=1}^{m} c[\ell+i-m]\ Q[i]$.

Se $x = y_{\ell+1}$ obtem-se $s(y_{\ell+1}-)$.

Procedimentos Auxiliares

BSMQ, INTERV(LBISEC).

Cap.0-B - Avalia s(x) a partir de sua expansão em B-splines

Multiplicações ou Divisões

$(3m^2 + 3m - 4)/2$, excluidas as operações para determinação de ℓ.

Algoritmo

```
função real BSX(m,n,y,c,x,ℓx) ;
inteiro  m,n,ℓx; real x ; real y[*], c[*] ;
| inteiro i,j; real sx; real Q[1:m+1] ;
  se x = y[n+1] então  ℓx = n
                senão| j ← INTERV(1,m+n,ℓx,x,y);ℓx ← j |;
  BSMQ(m,ℓx,y,x,Q) ;
  sx ← 0;
  para i de 1 passo 1 até m faça sx ← sx + c[ℓx+i-m] * Q[i] ;
  BSX ← sx |
```

Codificação em ALGOL

```
%=============================================================================
 REAL PROCEDURE BSX(M,N,Y,C,X,LX);
  INTEGER M,N,LX; REAL X; REAL ARRAY  Y[*], C[*];
%-----------------------------------------------------------------------------
% UTILIZA  LBSEC,INTERV  E  BSMQ
%-----------------------------------------------------------------------------
   BEGIN

   INTEGER I,J; REAL SX; REAL ARRAY Q[1:M+1];
   IF X=Y[N+1] THEN LX:=N ELSE BEGIN J:=INTERV(1,M+N,LX,X,Y); LX:=J END;
   BSMQ(M,LX,Y,X,Q);  SX:=0;
   FOR I:=1 STEP 1 UNTIL M DO  SX:=SX+C[LX+I-M]*Q[I];
   BSX:=SX
   END;
```

Cap.0-B - Avalia s(x) através de sua expansão em B-splines e partição uniforme de passo h.

Dados

Os extremos do intervalo $I = [a,b]$

k = número de nós da partição uniforme associada com $\mathcal{S}_{m(\Delta)}$

m = ordem dos polinômios por partes

$\{c_i\}_1^{m+k}$ coeficientes da expansão em B-splines de $s \in \mathcal{S}_{m(\Delta)}$

x ponto de $[a,b]$

determina

O valor de s(x)

Função Real

BSXH(a,b,k,m,c,x)

Entrada

a,b,k,m,c[1],...,c[m+k] e x

O dimensionamento da matriz c deve ser previsto no programa principal.

Saída

BSXH = s(x)

Método

Versão simplificada do algoritmo utilizado em BSX.

Incialmente determina ℓ tal que $y_\ell \leq x < y_{\ell+1}$ e, em seguida avalia

$$s(x) = \sum_{i=\ell-m+1}^{\ell} c_i N^m\left(\frac{x-y_i}{h}\right) = \sum_{i=1}^{m} c_{\ell+i-m} Q_i$$

Os valores de $\{Q_i\}_1^m$ são obtidos

Procedimentos Auxiliares

BSMQH

Multiplicações ou Divisões

$m^2 + 3m - 1$, não contando o cálculo de ℓ e h.

Cap.0-B - Avalia s(x) através de sua expansão em B-splines e partição uniforme de passo h.

Comparação com procedimentos equivalentes

m	BSX2	BSXH	BSX2H
	$\frac{3}{2}$ m(m-1)	m^2+3m-1	m(m-1)+2
2	3	9	4
3	9	17	8
4	18	27	14
5	30	39	22
6	45	53	32

A comparação acima revela a vantagem no uso de BSX2H. O procedimento BSXH foi desenvolvido apenas para conferência dos resultados fornecidos por algoritmos equivalentes porém com eficiências diferentes.

Algoritmo

```
função real BSXH(a,b,k,m,c,x) ;
inteiro k,m ; real a,b,x ; real c[*] ;
| inteiro ℓ,i ; real h,s ;real Q[1:m+1] ;
  h ← (b-a)/(k+1) ;
  se x=b então ℓ ←m+k senão ℓ ← (x-a)/h+m-0.5 ;
  BSMQH(a,b,k,m,ℓ,x,Q) ;
  s ← 0;;
  para i de 1 passo 1 até m faça  s ← s + c[ℓ+i-m] * Q[i] ;
  BSXH ← s |
```

Codificação em ALGOL

```
%=============================================================================
 REAL PROCEDURE BSXH(A,B,K,M,C,X);
 INTEGER K,M; REAL A,B,X;   REAL ARRAY C[*];
%-----------------------------------------------------------------------------
%     UTILIZA  BSMQH
%-----------------------------------------------------------------------------
 BEGIN INTEGER L,I; REAL H,S; REAL ARRAY Q[1:M+1];

       H:=(B-A)/(K+1);
       IF X=B THEN L:=M+K  ELSE  L:=(X-A)/H + M - 0.5 ;
       BSMQH(A,B,K,M,L,X,Q);
       S:=0;
       FOR I:=1 STEP 1 UNTIL M DO  S:=S + C[L+I-M]*Q[I] ;
       BSXH :=S
 END;
```

Cap.0-B - Avalia s(x) a partir de sua expansão em B-splines

Dados

m = ordem dos polinômios por partes

n = dimensão do espaço $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$

$\{y_i\}_1^{m+n}$ nós da partição estendida Δ associada com $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$

$\{c_i\}_1^n$ coeficientes da expansão em B-splines do spline s

x em [a,b]

ℓx estimativa de ℓ tal que $y_\ell \leq x < y_{\ell+1}$

determina

s(x) através de $s(x) = c_\ell^{[m]}(x)$

Função Real

BSX2(m,n,y,c,x,ℓx)

Entrada

m, n, y[1],..., y[m+n], c[1],..., c[n], x e ℓx; os dimensionamentos das matrizes devem ser previstas no programa principal.

Saída

BSX2 = s(x)

Método

$s(x) = \sum_{i=1}^{n} c_i N_i^m = \sum_{i=1}^{n+j-1} c_i^{[j]} N^{m-j+1}(x)$. Inicialmente determina-se ℓ tal que $y_\ell \leq x < y_{\ell+1}$. Em seguida consideramos:

$$s(x) = \sum_{i=\ell+1-m}^{\ell} c_i^{[1]} N_i^m = \sum_{i=\ell-m}^{\ell} c_i^{[2]} N_i^{m-1}(x) = \ldots = \sum_{i=\ell}^{\ell} c_i^{[m]} N_i^1 = c_\ell^{[m]}$$

O valor de $c_\ell^{[m]}(x)$ é computado através da geração da matriz $\{c_{\ell+i-m}^{[j]}\}_{j=1,i=j}^{m\ \ m}$, considerando-se:

Cap.0-B - Avalia s(x) a partir de sua expansão em B-splines

$$c_i^{[1]} = c_i, \quad i=1,2,\ldots,n$$

$$c_0^{[j]} = c_{n+j}^{[j]} = 0, \quad j=1,2,\ldots,m$$

$$c_i^{[j+1]} = \begin{cases} 0 \text{ se } y_{i+m-j} - y_i = 0 \\ \dfrac{(x-y_i)\, c_i^{[j]} + (y_{i+m-j} - x)\, c_{i-1}^{[j]}}{y_{i+m-j} - y_i} \end{cases}, \quad \begin{matrix} i=1,2,\ldots,n+j \\ j=1,2,\ldots,m-1 \end{matrix}$$

Este algoritmo é extremamente estável por envolver somente combinação convexas de quantidades não negativas.

Se $x = y_{\ell+1}$ obtem-se $s(y_{\ell+1}^-)$.

Procedimentos Auxiliares

INTERV(LBISEC)

Multiplicações ou Divisões

$(3m^2 - 3m)/2$, excluídas as operações para cálculo de ℓ.

Comparação com procedimento equivalente

m	BSX	BSX2
	$\frac{3m^2+3m-4}{2}$	$\frac{3m^2-3m}{2}$
2	7	3
3	16	9
4	28	18
5	43	30
6	61	45

Cap.0-B - Avalia s(x) a partir de sua expansão em B-splines

Algoritmo

```
função real BSX2(m,n,y,c,x,ℓx) ;
inteiro  m,n,ℓx ; real x ; real y[*], c[*] ;
 inteiro  i,j,k; real denom, a1, a2 ; real cx[1:m] ;
 se x=y[n+1]então k ← n senão k ← INTERV(m,n,ℓx,x,y) ;
 para j de 1 passo 1 até m faça cx[j] ← c[j+k-m] ;
 se m > 1 então
 para j de 2 passo 1 até m faça
    para i de m passo -1 até j faça
       denom ← y[i+k-j+1] - y[i+k-m] ;
             a1 ← (x-y[i+k-m]/denom ;
             a2 ← 1 - a1 ;
          cx[i] ← a1 * cx[i] + a2 * cx[i-1] ;
BSX2 ← cx[m]
```

Codificação em ALGOL

```
%=============================================================
 REAL PROCEDURE BSX2(M,N,Y,C,X,LX);
   INTEGER N,M; REAL X; ARRAY Y[*],C[*]; INTEGER LX;
%-------------------------------------------------------------
%     UTILIZA  LBISEQ, INTERV
%-------------------------------------------------------------
   BEGIN
    INTEGER I,J,K; REAL DENOM,A1,A2; REAL ARRAY CX[1:M];

  IF X=Y[N+1] THEN K:=N ELSE K:=INTERV(M,N,LX,X,Y);
 WRITE(IMP,</,"M=",I2,X2,"N=",I2,X2,"K=",I2,X2,"LX=",I2>,M,N,K,LX);
 WRITE(IMP,</,"Y",*(F4.1)>,M+N,FOR I:= 1 STEP 1 UNTIL N+M DO Y[I]);
   FOR J:= 1 STEP 1 UNTIL M DO CX[J]:=C[J+K-M];
   IF M > 1 THEN
   FOR J:= 2 STEP 1 UNTIL M DO
     BEGIN FOR I:=M STEP -1 UNTIL J DO
             BEGIN DENOM:=Y[I+K-J+1]-Y[I+K-M];
                     A1:=(X-Y[I+K-M])/DENOM; A2:=1-A1;
                CX[I]:=A1*CX[I] + A2*CX[I-1]
             END;
     END;
   BSX2:=CX[M]
   END;
```

Cap.0-B - Avalia s(x) através de sua expansão em B-splines e partição uniforme de passo h.

Dados

Os extremos do intervalo $I = [a,b]$

k = número de nós distintos da partição uniforme associada com $\mathcal{S}_{m(\Delta h)}$

m = ordem dos polinômios por partes

$\{c_i\}_1^{m+k}$ coeficientes da expansão em B-splines de um spline $s \in \mathcal{S}_{m(\Delta h)}$

x ponto de [a,b]

determina

s(x) e armazena-o em BSX2H

Função Real

BSX2H(a,b,k,m,c,x)

Entrada

a, b, k, m, c[1],..., c[m+k] , x

O dimensionamento da matriz deve ser propiciado pelo programa principal.

Saída

BSX2H = s(x)

Método

Versão simplificada do algoritmo utilizado em BSX2.

O cálculo de ℓ é o maior inteiro contido em (x-a)/h+m; a expressão (x-a)/h+m-0,5 foi utilizada porque em ALGOL, linguagem utilizada para codificação do algoritmo, o valor da expressão é arredondado.

Procedimentos Auxiliares

Nenhum

Multiplicações ou Divisões

m(m-1)+2, não contando as operações para cálculo de h, ℓ e fatoriais.

Cap.0-B - Avalia s(x) através de sua expansão em B-splines e partição uniforme de passo h.

Comparação com procedimento equivalente

	BSX2	BSX2H
m	3m(m-1)/2	m(m-1)+2
2	3	4
3	9	8
4	18	14
5	30	22
6	45	32

Algoritmo

```
função real BSX2H(a,b,k,m,c,x) ;
inteiro k,m ; real a,b,x ; real c[*] ;
┌ inteiro n,j,i,ℓ ; real h, xdh, fat ;
  n ← m+k ;
  ┌┌ real cx[1:n] ;
    h ← (b-a)/(k+1) ;
    se x=b então ℓ ← n senão ℓ ← (x-a)/h+m-0.5 ;
    xdh ← (x-a)/h-ℓ+m ;
    para j de 1 passo 1 até m faça cx[j] ← c[j+ℓ-m] ;
    fat ← 1 ;
    para j de 2 passo 1 até m faça
    ┌┌┌ fat ← fat * (j-1) ;
        para i de m passo -1 até j faça
        cx[i] ← (xdh+m-i) * cx[i] + (i-j+1-xdh) * cx[i-1] └└└;
    BSX2H ← cx[m]/fat └└;
```

Cap.0-B - Avalia s(x) através de sua expansão em B-splines e partição uniforme de passo h.

Codificação em ALGOL

```
%=============================================================================
 REAL PROCEDURE BSX2H(A,B,K,M,C,X);
 INTEGER M,K; REAL A,B,X;  REAL ARRAY C[*];
%-----------------------------------------------------------------------------
 BEGIN INTEGER N,J,I,L; REAL H, XDH, FAT;

       N:=M+K; H:=(B-A)/(M+1);
 BEGIN REAL ARRAY CX[1:N];

       IF X=B THEN L:=N  ELSE  L:=(X-A)/H + M - 0.5 ;
       XDH:=(X-A)/H-L+M ;
       FOR J:=1 STEP 1 UNTIL M DO  CX[J]:=C[J+L-M];
       FAT:=1;
       FOR J:=2 STEP 1 UNTIL M DO
 BEGIN      FAT:=FAT*(J-1);
            FOR I:=M STEP -1 UNTIL J  DO
                CX[I]:=(XDH+M-I)*CX[I]+(I-J+1-XDH)*CX[I-1]
 END;
       BSX2H:=CX[M]/FAT
 END; END;
```

Cap.0-B - Determina ℓ tal que $y_\ell \leq x < y_{\ell+1}$

Dados

$\{y_i\}_1^{m+n}$ nós da partição estendida $\bar{\Delta}$ associada com $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$

os índices p e q dos extremos do intervalo $[y_p , y_q)$

$x \in [y_p , y_q) \subset [a,b]$

ℓx estimativa de ℓ tal que $y_p \leq y_\ell \leq x < y_{\ell+1} \leq y_q$

determina

ℓ tal que $y_\ell \leq x < y_{\ell+1}$

Função Inteira

INTERV(p,q,ℓx,x,y)

Entrada

p, q, ℓx, x, y[1],..., y[m+n], com dimensionamento previsto no programa principal.

Saída

INTERV = ℓ

Método

Utiliza a bissecção do intervalo $[y_p , y_q)$ combinado com uma estimativa ℓx de ℓ .

A vantagem no uso de INTERV em lugar de LBISEC surge quando se deseja tabelar $s \in \mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$ em vários pontos de I = [a,b] pois a proximidade dos x's entre si implica na proximidade dos respectivos ℓ's.

Procedimento Auxiliar

LBISEC

Multiplicações ou Divisões

Se a estimativa ℓx de ℓ for boa ocorre uma considerável redução das operações em relação a LBISEC.

Cap:0-B - Determina ℓ tal que $y_\ell \leq x < y_{\ell+1}$

Algoritmo

```
função inteira INTERV(p,q,ℓx,x,y) ;
inteiro p,q,ℓx; real x; real y[*] ;
┌ inteiro ℓ,t;
  se x ≥ y[ℓx] então
          se x < y[ℓx+1] então ℓ ← ℓx
                         senão
                         se x < y[ℓx+2] então ℓ ← ℓx+1
                                        senão
                                        ┌┐ t ← ℓx+2;
                                           ℓ ← LBISEC(t,q,x,y)└┘
          senão
          se x ≥ y[ℓx-1] então ℓ ←ℓx-1
                         senão ┌┐ t ← ℓx-1;
                                  ℓ ← LBISEC(p,t,x,y)└┘ ;
  INTERV ← ℓ └
```

Codificação em ALGOL

```
%=============================================================================
 INTEGER PROCEDURE INTERV(P,Q,LX,X,Y);
 INTEGER P,Q,LX; REAL X; REAL ARRAY Y[*];
%-----------------------------------------------------------------------------
%     UTILIZA LBISEC
%-----------------------------------------------------------------------------
 BEGIN
 INTEGER L,T;

 IF X >= Y[LX] THEN IF X < Y[LX+1] THEN L:=LX
                                   ELSE IF X < Y[LX+2]
                                        THEN L:=LX+1
                                        ELSE BEGIN T:=LX+2;
                                                   L:=LBISEC(T,Q,X,Y)
                                             END
               ELSE IF X>= Y[LX-1] THEN L:=LX-1
                                   ELSE BEGIN T:=LX-1;
                                              L:=LBISEC(P,T,X,Y)
                                        END;
 INTERV:=L
 END;
```

Cap.0-B - Determina Δ a partir da partição estendida $\tilde{\Delta}$

Dados

m = ordem dos polinômios por parte

n = dimensão do espaço $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$

$\{y_i\}_1^{m+n}$ nós da partição estendida $\tilde{\Delta}$ associada com $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$

determina

Os extremos do intervalo [a,b]

d = número de nós da partição Δ referida em $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$

$\{z_i\}_1^d$ nós da partição Δ

$\{\ell m\}_1^d$ componentes do vetor de multiplicidade $\mathcal{M}$

Procedimento

INVPTE(m,n,y,d,a,b,ℓm,z)

Entrada

m,n,y[1],..., y[m+n], com dimensão fornecida pelo programa principal

Saída

d, a, b, ℓm[1],..., ℓm[d], z[1],..., z[d]

Os dimensionamentos das matrizes devem ser previstos no programa principal

Método

Inicialmente determina $a=y_m$ e $b=y_{n+1}$. Em seguida por um processo de comparação e contagem define d, as componentes do vetor de multiplicidade $\mathcal{M}$ e os nós da partição Δ .

Procedimentos Auxiliares

Nenhum

Multiplicações ou Divisões

0

Cap.0-B - Determina Δ a partir da partição estendida $\tilde{\Delta}$

Algoritmo

```
procedimento INVPTE(m,n,y,d,a,b,ℓm,z);
inteiro m,n,d; real a,b; real z[*]; inteiro ℓm[*];
| inteiro i;
  a ← y[m];  b ← y[n+1];
  i ← m+1;   d ← 0;
  repita || d ← d+1;
            ℓm[d] ← 1;
            enquanto y[i] = y[i+1] faça ||| i ← i+1;
                                            ℓm[d] ← ℓm[d] + 1 |||
            z[d] ← y[i];
            i ← i+1 ||
            até que i > n |
```

Codificação em ALGOL

```
%=============================================================================
  PROCEDURE INVPTE(M,N,Y,D,A,B,LM,Z);
  INTEGER M,N,D; REAL A,B; REAL ARRAY Y[*],Z[*]; INTEGER ARRAY LM[*];
%-----------------------------------------------------------------------------
  BEGIN
  INTEGER I;

  A:=Y[M];  B:=Y[N+1]; I:=M+1; D:=0;
  DO BEGIN D:=D+1; LM[D]:=1;
           WHILE Y[I]=Y[I+1] DO BEGIN I:=I+1; LM[D]:=LM[D]+1  END;
           Z[D]:=Y[I]; I:=I+1;END
     UNTIL I>N
  END;
```

Cap.0-B - Determina ℓ tal que $y_\ell \leq x < y_{\ell+1}$

Dados

$\{y_i\}_1^{m+n}$ nós da partição estendida $\tilde{\Delta}$ associada com $\mathcal{S}(p\,m; m; \Delta)$

Os índices p e q dos extremos do intervalo $[y_p , y_q)$

$x \in [y_p , y_q) \subset [a,b]$

determina

ℓ tal que $y_\ell \leq x < y_{\ell+1}$ armazenando-o em LBISEC

Função Inteira

LBISEC(p,q,x,y);

Entrada

p, q, x, y[1], y[2],..., y[m+n]

O dimensionamento da matriz deve ser previsto no programa principal

Saída

LBISEC = ℓ

Método

Bissecção do intervalo $[y_p , y_q)$

Procedimentos Auxiliares

Nenhum

Multiplicações ou Divisões

Depende da localização de x em [a,b].

Não excede ν tal que $2^{\nu-1} \leq q-p < 2^\nu$

Cap.0-B - Determina ℓ tal que $y_\ell \leq x < y_{\ell+1}$

Algoritmo

```
função inteira LBISEC(p,q,x,y);
inteiro p,q; real x; real y[*];
| inteiro ℓ,u,mid;
  ℓ ← p;   u ← q;
  enquanto u-ℓ > 1 faça || mid ← (ℓ+u)/2;
                           se x < y[mid] então u ← mid
                                         senão ℓ ← mid || ;
  enquanto y[ℓ] = y[ℓ+1]  faça ℓ ← ℓ+1;
  LBISEC ← ℓ |
```

Codificação em ALGOL

```
%=============================================================================
  INTEGER PROCEDURE LBISEC(P,Q,X,Y);
   INTEGER P,Q; REAL X; REAL ARRAY Y[*];
%-----------------------------------------------------------------------------
   BEGIN
   INTEGER L,U,MID;

   L:=P; U:=Q;
   WHILE U-L > 1 DO
      BEGIN MID:=(L+U)/2;
            IF X < Y[MID] THEN U:=MID
                          ELSE L:=MID END;
   WHILE Y[L]=Y[L+1] DO L:=L+1;
   LBISEC:=L
   END;
```

Cap.0-B - Determina partição estendida $\tilde{\Delta}$ a partir de Δ

Dados

Os extremos do intervalo $I = [a,b]$

m = ordem dos polinômios por partes

d = número de nós da partição Δ associada com $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$

$\{z_i\}_1^d$ nós da partição Δ

$\{\ell m\}_1^d$ componentes do vetor de multiplicidade $\mathcal{M}$

determina

n = dimensão do espaço $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$

$\{y_i\}_1^{m+n}$ nós da partição estendida $\tilde{\Delta}$ associada com $\mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$

Procedimento

PTE(m,n,y,d,a,b,ℓm,z)

Entrada

m, d, a, b, ℓm[1],...,ℓm[d], z[1],..., z[d]

Os dimensionamentos das matrizes ℓm e z devem ser fornecidos pelo programa principal.

Saída

n,y[1],..., y[n+m], com dimensionamento previsto no programa principal.

Método

Define os m valores iniciais da matriz y iguais ao extremo inferior de $I = [a,b]$. Em seguida, utilizando um processo de contagem constrói sucessivamente $\{y_{m+i} = z_1\}_{i=1}^{\ell m_1}$, $\{y_{m+\ell m_1+i} = z_2\}_{i=1}^{\ell m_2}, \ldots, \{y_{m+\ell m_1+\ldots+\ell m_{d-1}+i=z_d}\}_{i=1}^{\ell m_d}$ e $n = m+\ell m_1+\ldots+\ell m_d$. Finalmente define os últimos m valores da matriz y iguais ao extremo superior de $I = [a,b]$.

Procedimentos Auxiliares

Nenhum

Cap.0-B - Determina partição estendida $\bar{\Delta}$ a partir de Δ

Multiplicações ou Divisões

0

Algoritmo

```
procedimento PTE(m,n,y,d,a,b,ℓm,z);
inteiro m,n,d; real a,b; real z[*]; inteiro ℓm[*];
| inteiro i,j,iℓ1, iℓ2;
  para i de 1 passo 1 até m faça  y[i] ← a;
  iℓ2 ← m;
  para j de 1 passo 1 até d faça
      || iℓ2 ← iℓ2 + ℓm[j];
         iℓ1 ← iℓ2 - ℓm[j]+ 1;
         para i de iℓ1 passo 1 até iℓ2 faça y[i] ← z[j] ||
  n ← iℓ2 ;
  para i de 1 passo 1 até m faça  y[n+i] ← b |
```

Codificação em ALGOL

```
%=============================================================================
  PROCEDURE PTE(M,N,Y,D,A,B,LM,Z);
  INTEGER M,N,D;REAL A,B;REAL ARRAY Y[*],Z[*];INTEGER ARRAY LM[*];
%-----------------------------------------------------------------------------
  BEGIN

  INTEGER I,J,IL1,IL2;
  FOR I:=1 STEP 1 UNTIL M DO Y[I]:=A;
  IL2:=M;
  FOR J:=1 STEP 1 UNTIL D DO BEGIN IL2:=IL2+LM[J]; IL1:=IL2-LM[J]+1;
                                   FOR I:=IL1 STEP 1 UNTIL IL2 DO
                                       Y[I]:=Z[J]
                              END;
  N:=IL2;
  FOR I:=1 STEP 1 UNTIL M DO Y[N+I]:=B
  END;
```

Cap.0-B - Avalia $\{D_+^{j-1}s(x)\}_1^m$ através de sua expansão polinomial por partes

---

Dados

k = número de nós distintos da partição Δ.

m = ordem dos polinômios por partes.

$\{z_i\}_{i=1}^k$ nós distintos da partição Δ em ordem crescente.

$\{cp_{ij}\}_{i=0,j=1}^{k\quad m}$ coeficientes dos polinômios $p_i(x) = \sum_{j=1}^{m} cp_{ij}(x-z_i)^{j-1}$ que representam $s \in \mathcal{S}(\mathcal{P}_m;\mathcal{M};\Delta)$ em cada subintervalo I_i, i=0,1,...,k definidos pela partição Δ de $[a,b]$.

x ponto de $[a,b]$.

determina

$\{D_+^{j-1}s(x)\}_{j=1}^m$ armazenando-os, respectivamente, em $\{dx_i\}_{i=1}^m$.

Procedimento

RPPSDX(k,m,z,cp,x,dx)

Entrada

k, m, z[1],..., z[k], cp[0,1],..., cp[0,m],..., cp[k,1],..., cp[k,m], x

Saída

dx[1], dx[2],..., dx[m]

Método

Inicialmente determina o intervalo I_i , i=0,1,...,k ao qual x pertence.

Em seguida define-se $w=x-z_i$ e aplica-se o esquema de Horner para calcular $p_i(w)$, $p_i'(w)$, $\frac{p_i''(w)}{2!}$, $\frac{p_i'''(w)}{3!}$,..., $\frac{p_i^{(m-1)}(w)}{(m-1)!}$. Recordemos brevemente que o esquema de Horner é a repetição do dispositivo prático de Briot-Ruffini, aplicado inicialmente ao polinômio $p_i(x)$ e sucessivamente aos quocientes das divisões por x-w.

Finalmente multiplica-se cada resultado pelo número fatorial correspondente a fim de obter as derivadas sucessivas de $s(x) = p_i(x)$, $x \in I_i$.

Cap.0-B - Avalia $\{D_+^{j-1}s(x)\}_1^m$ através de sua expansão polinomial por partes

Procedimentos Auxiliares

INTERV(LBISEC)

Multiplicações ou Divisões

$\frac{1}{2}[m^2+3m-8]$, não contando as operações na execução de INTERV.

Comparação com procedimento equivalente

m	BSDX2	RPPSDX
	$\frac{1}{2}(m-1)m(m+1)$	$\frac{1}{2}[m^2+3m-8]$
2	3	1
3	12	5
4	30	10
5	60	16
6	105	23

Algoritmo

```
procedimento RPPSDX(k,m,z,cp,x,dx);
inteiro k,m; real x; real z[*], cp[*,*] , dx[*];
┌ inteiro kx, i; real zk,w,d;
  se x < z[1] então kx ← 0
              senão se x ≥ z[k]então kx ← k
                                 senão kx ← INTERV(1,k,k/2,x,z);
  se kx=0 então zk ← z[1] senão zk ← z[kx] ;
  w ← x-zk;
  para i de 1 passo 1 até m faça dx[i] ← cp[kx,i] ;
  para kx de 1 passo 1 até m-1 faça
  para i de m-1 passo -1 até kx faça
  dx[i] ← dx[i] + w * dx[i+1];
  d ← 1;
  para kx de 3 passo 1 até m faça
  ┌ d ← d * (kx-1);
    dx[kx] ← d * dx[kx] ┘ ;
```

Cap.0-B - Avalia $\{D_{+}^{j-1}s(x)\}_{1}^{m}$ através de sua expansão polinomial por partes.

Codificação em ALGOL

```
%=============================================================================
 PROCEDURE RPPSDX(K,M,Z,CP,X,DX);
 INTEGER K,M; REAL X; REAL ARRAY Z[*],CP[*,*],DX[*];
%-----------------------------------------------------------------------------
% UTILIZA INTERV  E  LBISEC
%-----------------------------------------------------------------------------
 BEGIN
 INTEGER KX,I; REAL ZK, W, D;

%
 IF X<Z[1] THEN KX:=0
           ELSE IF X>=Z[K] THEN KX:=K ELSE KX:=INTERV(1,K,K/2,X,Z);
%
 IF KX =0  THEN ZK:=Z[1] ELSE  ZK:=Z[KX];
 W:=X-ZK;
 FOR I:=1 STEP 1 UNTIL M DO DX[I]:=CP[KX,I];
 FOR KX:=1 STEP 1 UNTIL M-1 DO
 FOR  I:=M-1 STEP -1 UNTIL KX DO  DX[I]:= DX[I] + W*DX[I+1];
 D:=1; FOR KX:=3 STEP 1 UNTIL M DO
       BEGIN
       D:=D*(KX-1); DX[KX]:= D*DX[KX]
       END;
 END;
```

Cap.0-B - Avalia s(x) através de sua representação polinomial por partes.

Dados

k = número de nós distintos da partição Δ

m = ordem dos polinômios por parte

$\{z_i\}_{i=1}^{k}$ nós distintos da partição Δ em ordem crescente

$\{cp_{ij}\}_{i=0,\ j=1}^{k\quad m}$ coeficientes dos polinômios $p_i(x) = \sum_{j=1}^{m} cp_{ij}(x-z_i)^{j-1}$ que representam $s \in \mathcal{S}(\mathcal{P}_m; \mathcal{M}; \Delta)$ em cada subintervalo I_i, i=0,...,k determinados pela partição Δ de [a,b]

x = ponto de [a,b]

determina

O valor de s(x) armazenando-o em RPPSX

Função Real

RPPSX(k,m,z,cp,x)

Entrada

k, m, z[1],..., z[k], cp[0,1], ..., cp[0,m], ..., cp[k,1],..., cp[k,m], x

Os dimensionamentos das matrizes são propiciados pelo programa principal

Saída

RPPSX = s(x)

Método

Inicialmente determina o intervalo I_i, i=0,1,...,k ao qual x pertence. Em seguida define $w = x-x_i$ e avalia $p_i(x) = \sum_{j=1}^{m} cp[i,j]w^{j-1}$ através do dispositivo prático de Briot-Ruffini.

Procedimentos Auxiliares

INTERV(LBISEC)

Multiplicações ou Divisões

m-1, não contando as operações na execução de INTERV.

Cap.0-B - Avalia s(x) através de sua representação polinomial por partes.

Comparação com procedimento equivalente

m	BSX2	RPPSX
	$\frac{3m^2 - 3m}{2}$	m-1
2	3	1
3	9	2
4	18	3
5	30	4
6	45	5

Algoritmo

```
função real RPPSX(k,m,z,cp,x);
inteiro k,m; real x; real z[*], cp[*,*];
┌ inteiro  i, kx; real d, zk, w;
  se x < z[1] então  kx ← 0
              senão  se x ≥ z[k] então kx ← k
                                 senão kx ← INTERV(1,k,k/2,x,z);
  se kx = 0 então zk ← z[1]  senão zk ← z[kx] ;
  w ← x-zk;
  d ← cp[kx,m];
  para i de m-1 passo -1 até 1 faça
  d ← d * w + cp[kx,i];
  RPPSX ← d ┘
```

Codificação em ALGOL

```
%==========================================================================
 REAL PROCEDURE RPPSX(K,M,Z,CP,X);
 INTEGER K,M; REAL X; REAL ARRAY CP[*,*],Z[*];
%--------------------------------------------------------------------------
% UTILIZA INTERV  E  LBISEC
%--------------------------------------------------------------------------
 BEGIN
 INTEGER I,KX; REAL D,ZK,W;

%
 IF X<Z[1] THEN KX:=0
           ELSE IF X>=Z[K] THEN KX:=K ELSE KX:=INTERV(1,K,K/2,X,Z);
%
 IF KX=0 THEN ZK:=Z[1] ELSE ZK:=Z[KX];
 W:=X-ZK; D:=CP[KX,M];
 FOR I:=M-1 STEP -1 UNTIL 1 DO D:=D*W + CP[KX,I];
%
 RPPSX:=D
 END;
```

"RELATÓRIO TÉCNICO"
DEPARTAMENTO DE MATEMÁTICA APLICADA
TÍTULOS PUBLICADOS

RT-MAP-7701 - Ivan de Queiroz Barros
On equivalence and reducibility of Generating Matrices of RK-Procedures - Agosto 1977

RT-MAP-7702 - V.W. Setzer
A Note on a Recursive Top-Down Analizer of N.Wirth - Dezembro 1977

RT-MAP-7703 - Ivan de Queiroz Barros
Introdução a Aproximação Ótima - Dezembro 1977

RT-MAP-7704 - V.W. Setzer, M.M. Sanches
A linguagem "LEAL" para Ensino básico de Computação - Dezembro 1977

RT-MAP-7801 - Ivan de Queiroz Barros
Proof of two Lemmas of interest in connection with discretization of Ordinary Differential Equations - Janeiro 1978

RT-MAP-7802 - Silvio Ursic, Cyro Patarra
Exact solution of Systems of Linear Equations with Iterative Methods
Fevereiro 1978

RT-MAP-7803 - Martin Grötschel, Yoshiko Wakabayashi
Hypohamiltonian Digraphs - Março 1978

RT-MAP-7804 - Martin Grötschel, Yoshiko Wakabayashi
Hypotraceable Digraphs - Maio 1978

RT-MAP-7805 - W. Hesse, V.W. Setzer
The Line-Justifier: an example of program development by transformations
Junho 1978

RT-MAP-7806 - Ivan de Queiroz Barros
Discretização
Capítulo I - Tópicos Introdutórios
Capítulo II - Discretização
Julho 1978

RT-MAP-7807 - Ivan de Queiroz Barros
(Γ', Γ) - Estabilidade e Métodos Preditores-Corretores - Setembro 1978

RT-MAP-7808 - Ivan de Queiroz Barros
Discretização
Capítulo III - Métodos de passo progressivo para Eq. Dif. Ord. com condições iniciais - Setembro 1978

RT-MAP-7809 - V.W. Setzer
Program development by transformations applied to relational Data-Base queries - Novembro 1978

RT-MAP-7810 - Nguiffo B. Boyom, Paulo Boulos
Homogeneity of Cartan-Killing spheres and singularities of vector fields - Novembro 1978

TÍTULOS PUBLICADOS

RT-MAP-7811 - D.T. Fernandes e C. Patarra
Sistemas Lineares Esparsos, um Método Exato de Solução - Novembro 1978

RT-MAP-7812 - V.W. Setzer e G. Bressan
Desenvolvimento de Programas por Transformações: uma Comparação entre dois Métodos - Novembro 1978

RT-MAP-7813 - Ivan de Queiroz Barros
Variação do Passo na Discretização de Eq. Dif. Ord. com Condições Iniciais - Novembro 1978

RT-MAP-7814 - Martin Grötschel e Yoshiko Wakabayashi
On the Complexity of the Monotone Asymetric Travelling Salesman Polytope I: HIPOHAMILTONIAN FACETS - Dezembro 1978

RT-MAP-7815 - Ana F. Humes e E.I. Jury
Stability of Multidimensional Discrete Systems: State-Space Representation Approach - Dezembro 1978

RT-MAP-7901 - Martin Grötschel, Yoshiko Wakabayashi
On the complexity of the Monotone Asymetric Travelling Salesman Polytope II: HYPOTRACEABLE FACETS - Fevereiro 1979

RT-MAP-7902 - M.M. Sanches e V.W. Setzer
A portabilidade do Compilador para a Linguagem LEAL - Junho 1979

RT-MAP-7903 - Martin Grötschel, Carsten Thomassen, Yoshiko Wakabayashi
Hypotraceable Digraphs - Julho 1979

RT-MAP-7904 - N'Guiffo B. Boyom
Translations non triviales dans les groupes (transitifs) des transformations affines - Novembro 1979

RT-MAP-8001 - Ângelo Barone Netto
Extremos detectáveis por jatos - Junho 1980

RT-MAP-8002 - Ivan de Queiroz Barros
Medida e Integração
Cap. I - Medida e Integração Abstrata - Julho 1980

RT-MAP-8003 - Routo Terada
Fast Algorithms for NP-Hard Problems which are Optimal or Near-Optimal with Probability one - Setembro 1980

RT-MAP-8004 - V.W. Setzer e R. Lapyda
Uma Metodologia de Projeto de Bancos de Dados para o Sistema ADABAS
Setembro 1980

RT-MAP-8005 - Imre Simon
On Brzozowski's Problem: $(1uA)^m = A^*$ - Outubro 1980

RT-MAP-8006 - Ivan de Queiroz Barros
Medida e Integração
Cap. II - Espaços Lp - Outubro 1980

TÍTULOS PUBLICADOS

RT-MAP-8101 - Luzia Kazuko Yoshida e Gabriel Richard Bitran
Um algoritmo para Problemas de Programação Vetorial com Variáveis Zero-um - Fevereiro 198.

RT-MAP-8102 - Ivan de Queiroz Barros
Medida e Integração
Cap. III - Medidas em Espaços Topológicos - Março 1981

RT-MAP-8103 - V.W. Setzer, R. Lapyda
Design of Data Models for the ADABAS System using the Entity-Relationship Approach - Abril 1981

RT-MAP-8104 - Ivan de Queiroz Barros
Medida e Integração
Cap. IV - Medida e Integração Vetoriais - Abril 1981

RT-MAP-8105 - U.S.R. Murty
Projective Geometries and Their Truncations - Maio 1981

RT-MAP-8106 - V.W. Setzer, R. Lapyda
Projeto de Bancos de Dados, Usando Modelos Conceituais
Este relatório Técnico complementa o RT-MAP-8103. Ambos substituem o RT-MAP-8004 ampliando os conceitos ali expostos. - Junho 1981

RT-MAP-8107 - Maria Angela Gurgel, Yoshiko Wakabayashi
Embedding of Trees - August 1981

RT-MAP-8108 - Ivan de Queiroz Barros
Mecânica Analítica Clássica - Outubro 1981

RT-MAP-8109 - Ivan de Queiroz Barros
Equações Integrais de Fredholm no Espaço das Funções A-Uniformemente Contínuas - Novembro 1981

RT-MAP-8110 - Ivan de Queiroz Barros
Dois Teoremas sobre Equações Integrais de Fredholm - Novembro 1981

RT-MAP-8201 - Siang Wun Song
On a High-Performance VLSI Solution to Database Problems - Janeiro 1982

RT-MAP-8202 - Maria Angela Gurgel, Yoshiko Wakabayashi
A Result on Hamilton-Connected Graphs - Junho 1982

RT-MAP-8203 - Jörg Blatter, Larry Schumaker
The Set of Continuous Selections of a Metric Projection in C(X)
- Outubro 1981

RT-MAP-8204 - Jörg Blatter, Larry Schumaker
Continuous Selections and Maximal Alternators for Spline Approximation
- Dezembro 1981

RT-MAP-8205 - Arnaldo Mandel
Topology of Oriented Matroids - Junho 1982

RT-MAP-8206 - Erich J. Neuhold
Database Management Systems; A General Introduction - Novembro 1982

RT-MAP-8207 - Béla Bollobás
The Evolution of Random Graphs - Novembro 1982

TÍTULOS PUBLICADOS

RT-MAP-8208 - V.W. Setzer
Um Grafo Sintático para a Linguagem PL/M-80 - Novembro 1982

RT-MAP-8209 - Jayme Luiz Szwarcfiter
A Sufficient Condition for Hamilton Cycles - Novembro 1982

RT-MAP-8301 - W.M. Oliva
Stability of Morse-Smale Maps - Janeiro 1983

RT-MAP-8302 - Belá Bollobás, Istvan Simon
Repeated Random Insertion into a Priority Queue - Fevereiro 1983

RT-MAP-8303 - V.W. Setzer, P.C.D. Freitas e B.C.A. Cunha
Um Banco de Dados de Medicamentos - Julho 1983

RT-MAP-8304 - Ivan de Queiroz Barros
O Teorema de Stokes em Variedades Celuláveis - Julho 1983

RT-MAP-8305 - Arnaldo Mandel
The 1-Skeleton of Polytopes, oriented Matroids and some other lattices - - Julho 1983

RT-MAP-8306 - Arnaldo Mandel
Alguns Problemas de Enumeração em Geometria - Agosto 1983

RT-MAP-8307 - Siang Wun Song
Complexidade de E/S e Projetos Optimais de Dispositivos para Ordenação - - Agosto 1983

RT-MAP-8401-A - Dirceu Douglas Salvetti
Procedimentos para Cálculos com Splines
Parte A - Resumos Teóricos - Janeiro 1984
RT-MAP-8401-B
Parte B - Descrição de Procedimentos - Janeiro 1984
RT-MAP-8401-C
Parte C - Listagem de Testes - Janeiro 1984