```
   Help
#if defined(PremiaCurrentVersion) && PremiaCurrentVersion <
     (2007+2) //The "#else" part of the code will be freely av
    ailable after the (year of creation of this file + 2)
#else

/*regression basis*/
#include <string.h>
#include <math.h>
#include <stdio.h>

/*maximal dimension of the basis*/
#define DimBasisDefaultHD1 6
#define DimBasisDefaultHD2 21
#define DimBasisDefaultHD3 20
#define DimBasisDefaultHD4 21
#define DimBasisDefaultHD5 20
#define DimBasisDefaultHD6 19
#define DimBasisDefaultHD7 20
#define DimBasisDefaultHD8 23
#define DimBasisDefaultHD9 26
#define DimBasisDefaultHD10 29
#define DimBasisDefaultD1 20
#define DimBasisDefaultD2 21
#define DimBasisDefaultD3 20
#define DimBasisDefaultD4 21
#define DimBasisDefaultD5 20
#define DimBasisDefaultD6 19
#define DimBasisDefaultD7 20
#define DimBasisDefaultD8 23
#define DimBasisDefaultD9 26
#define DimBasisDefaultD10 29

static double HermiteD1(double *x, int ind);
static double HermiteD2(double *x, int ind);
static double HermiteD3(double *x, int ind);
static double HermiteD4(double *x, int ind);
static double HermiteD5(double *x, int ind);
static double HermiteD6(double *x, int ind);
static double HermiteD7(double *x, int ind);
static double HermiteD8(double *x, int ind);
static double HermiteD9(double *x, int ind);
static double HermiteD10(double *x, int ind);
static double CanoniqueD1(double *x, int ind);
static double CanoniqueD2(double *x, int ind);
static double CanoniqueD3(double *x, int ind);
static double CanoniqueD4(double *x, int ind);
static double CanoniqueD5(double *x, int ind);
static double CanoniqueD6(double *x, int ind);
static double CanoniqueD7(double *x, int ind);
static double CanoniqueD8(double *x, int ind);
static double CanoniqueD9(double *x, int ind);
static double CanoniqueD10(double *x, int ind);

static double aux,auxd1;
static int i,id1;

/*multidimensional basis are obtained as tensor product of
    unidimensional ones*/
/*the numbers inside the braces are the indexes of the unde
    rlying one-dimensional basis polynomials*/
/*example : {2,1,4} = p2(x1)*p1(x2)*p4(x3) where p2,p1 and
    p4 are the 2nd, 1st, 4th elements of the underlying one-dim
    ensional basis*/
static int TensorBasisD2[DimBasisDefaultD2][2]=
  {{0,0},

   {1,0},{0,1},

   {1,1},{2,0},{0,2},

   {2,1},{1,2},{3,0},{0,3},

   {2,2},{1,3},{3,1},{4,0},{0,4},

   {1,4},{4,1},{3,2},{2,3},{5,0},{0,5}};

static int TensorBasisD3[DimBasisDefaultD3][3]=
  {{0,0,0},

   {1,0,0},{0,1,0},{0,0,1},
   {2,0,0},{0,2,0},{0,0,2},{1,1,0},{1,0,1},{0,1,1},

   {1,1,1},{2,1,0},{1,2,0},{0,1,2},{0,2,1},{1,0,2},{2,0,1},
    {3,0,0},{0,3,0},{0,0,3}};

static int TensorBasisD4[DimBasisDefaultD4][4]=
  {{0,0,0,0},

   {1,0,0,0},{0,1,0,0},{0,0,1,0},{0,0,0,1},

   {2,0,0,0},{0,2,0,0},{0,0,2,0},{0,0,0,2},{1,1,0,0},{0,1,1
    ,0},{0,0,1,1},

   {1,1,1,0},{0,1,1,1},{1,0,1,1},{1,1,0,1},{3,0,0,0},{0,3,0
    ,0},{0,0,3,0},{0,0,0,3},

   {1,1,1,1}};

static int TensorBasisD5[DimBasisDefaultD5][5]=
  {{0,0,0,0,0},

   {1,0,0,0,0},{0,1,0,0,0},{0,0,1,0,0},{0,0,0,1,0},{0,0,0,0
    ,1},

   {2,0,0,0,0},{0,2,0,0,0},{0,0,2,0,0},{0,0,0,2,0},{0,0,0,0
    ,2},

   {1,1,0,0,0},{0,1,1,0,0},{0,0,1,1,0},{0,0,0,1,1},

   {1,1,1,0,0},{0,1,1,1,0},{0,0,1,1,1},

   {1,1,1,1,0},{0,1,1,1,1}};

static int TensorBasisD6[DimBasisDefaultD6][6]=
  {{0,0,0,0,0,0},

   {1,0,0,0,0,0},{0,1,0,0,0,0},{0,0,1,0,0,0},{0,0,0,1,0,0},
    {0,0,0,0,1,0},{0,0,0,0,0,1},

   {2,0,0,0,0,0},{0,2,0,0,0,0},{0,0,2,0,0,0},{0,0,0,2,0,0},
    {0,0,0,0,2,0},{0,0,0,0,0,2},
   {1,1,1,0,0,0},{0,1,1,1,0,0},{0,0,1,1,1,0},{0,0,0,1,1,1},

   {1,1,1,1,1,1}};

static int TensorBasisD7[DimBasisDefaultD7][7]=
  {{0,0,0,0,0,0,0},

   {1,0,0,0,0,0,0},{0,1,0,0,0,0,0},{0,0,1,0,0,0,0},{0,0,0,1
    ,0,0,0},{0,0,0,0,1,0,0},
   {0,0,0,0,0,1,0},{0,0,0,0,0,0,1},

   {2,0,0,0,0,0,0},{0,2,0,0,0,0,0},{0,0,2,0,0,0,0},{0,0,0,2
    ,0,0,0},{0,0,0,0,2,0,0},
   {0,0,0,0,0,2,0},{0,0,0,0,0,0,2},

   {1,1,1,1,0,0,0},{0,1,1,1,1,0,0},{0,0,1,1,1,1,0},{0,0,0,1
    ,1,1,1},

   {1,1,1,1,1,1,1}};


static int TensorBasisD8[DimBasisDefaultD8][8]=
  {{0,0,0,0,0,0,0,0},

   {1,0,0,0,0,0,0,0},{0,1,0,0,0,0,0,0},{0,0,1,0,0,0,0,0},{0
    ,0,0,1,0,0,0,0},
   {0,0,0,0,1,0,0,0},{0,0,0,0,0,1,0,0},{0,0,0,0,0,0,1,0},{0
    ,0,0,0,0,0,0,1},

   {2,0,0,0,0,0,0,0},{0,2,0,0,0,0,0,0},{0,0,2,0,0,0,0,0},{0
    ,0,0,2,0,0,0,0},
   {0,0,0,0,2,0,0,0},{0,0,0,0,0,2,0,0},{0,0,0,0,0,0,2,0},{0
    ,0,0,0,0,0,0,2},

   {1,1,1,1,0,0,0,0},{0,1,1,1,1,0,0,0},{0,0,1,1,1,1,0,0},{0
    ,0,0,1,1,1,1,0},
   {0,0,0,0,1,1,1,1},

   {1,1,1,1,1,1,1,1}};
```

```
static int TensorBasisD9[DimBasisDefaultD9][9]=
  {{0,0,0,0,0,0,0,0,0},

   {1,0,0,0,0,0,0,0,0},{0,1,0,0,0,0,0,0,0},{0,0,1,0,0,0,0,0
    ,0},{0,0,0,1,0,0,0,0,0},
   {0,0,0,0,1,0,0,0,0},{0,0,0,0,0,1,0,0,0},{0,0,0,0,0,0,1,0
    ,0},{0,0,0,0,0,0,0,1,0},
   {0,0,0,0,0,0,0,0,1},

   {2,0,0,0,0,0,0,0,0},{0,2,0,0,0,0,0,0,0},{0,0,2,0,0,0,0,0
    ,0},{0,0,0,2,0,0,0,0,0},
   {0,0,0,0,2,0,0,0,0},{0,0,0,0,0,2,0,0,0},{0,0,0,0,0,0,2,0
    ,0},{0,0,0,0,0,0,0,2,0},
   {0,0,0,0,0,0,0,0,2},

   {1,1,1,1,0,0,0,0,0},{0,1,1,1,1,0,0,0,0},{0,0,1,1,1,1,0,0
    ,0},{0,0,0,1,1,1,1,0,0},
   {0,0,0,0,1,1,1,1,0},{0,0,0,0,0,1,1,1,1},

   {1,1,1,1,1,1,1,1,1}};


static int TensorBasisD10[DimBasisDefaultD10][10]=
  {{0,0,0,0,0,0,0,0,0,0},

   {1,0,0,0,0,0,0,0,0,0},{0,1,0,0,0,0,0,0,0,0},{0,0,1,0,0,0
    ,0,0,0,0},
   {0,0,0,1,0,0,0,0,0,0},{0,0,0,0,1,0,0,0,0,0},{0,0,0,0,0,1
    ,0,0,0,0},
   {0,0,0,0,0,0,1,0,0,0},{0,0,0,0,0,0,0,1,0,0},{0,0,0,0,0,0
    ,0,0,1,0},
   {0,0,0,0,0,0,0,0,0,1},

   {2,0,0,0,0,0,0,0,0,0},{0,2,0,0,0,0,0,0,0,0},{0,0,2,0,0,0
    ,0,0,0,0},
   {0,0,0,2,0,0,0,0,0,0},{0,0,0,0,2,0,0,0,0,0},{0,0,0,0,0,2
    ,0,0,0,0},
   {0,0,0,0,0,0,2,0,0,0},{0,0,0,0,0,0,0,2,0,0},{0,0,0,0,0,0
    ,0,0,2,0},
   {0,0,0,0,0,0,0,0,0,2},
   {1,1,1,1,0,0,0,0,0,0},{0,1,1,1,1,0,0,0,0,0},{0,0,1,1,1,1
    ,0,0,0,0},
   {0,0,0,1,1,1,1,0,0,0},{0,0,0,0,1,1,1,1,0,0},{0,0,0,0,0,1
    ,1,1,1,0},
   {0,0,0,0,0,0,1,1,1,1},

   {1,1,1,1,1,1,1,1,1}};

static int TestBasisDimension(char *ErrorMessage,char *na
    me, char *nameref ,int Basis_Dimension, int DimMax)
{
  char dim[10];
  *dim='{0';
  if (strcmp(nameref,name)==0){
  if (Basis_Dimension>DimMax){
    strcat(ErrorMessage,"BasisDimensionError");
    strcat(ErrorMessage,nameref);
    sprintf(dim," : %d",DimMax);
    strcat(ErrorMessage,dim);
    strcat(ErrorMessage,")");
  }
  return 1;
  }
  return 0;
}

void CheckBasisDimension(char *ErrorMessage, char *name,
    int Basis_Dimension)
{
  if (Basis_Dimension<1){
  strcat(ErrorMessage,"#Error : basis dimension must be >0
    ");
  }

  if (TestBasisDimension(ErrorMessage,name,"HerD1",Basis_
    Dimension,DimBasisDefaultHD1)) return;
  if (TestBasisDimension(ErrorMessage,name,"HerD2",Basis_
    Dimension,DimBasisDefaultHD2)) return;
  if (TestBasisDimension(ErrorMessage,name,"HerD3",Basis_
    Dimension,DimBasisDefaultHD3)) return;
  if (TestBasisDimension(ErrorMessage,name,"HerD4",Basis_
    Dimension,DimBasisDefaultHD4)) return;
  if (TestBasisDimension(ErrorMessage,name,"HerD5",Basis_
    Dimension,DimBasisDefaultHD5)) return;
  if (TestBasisDimension(ErrorMessage,name,"HerD6",Basis_
    Dimension,DimBasisDefaultHD6)) return;
  if (TestBasisDimension(ErrorMessage,name,"HerD7",Basis_
    Dimension,DimBasisDefaultHD7)) return;
  if (TestBasisDimension(ErrorMessage,name,"HerD8",Basis_
    Dimension,DimBasisDefaultHD8)) return;
  if (TestBasisDimension(ErrorMessage,name,"HerD9",Basis_
    Dimension,DimBasisDefaultHD9)) return;
  if (TestBasisDimension(ErrorMessage,name,"HerD10",Basis_
    Dimension,DimBasisDefaultHD10)) return;
  if (TestBasisDimension(ErrorMessage,name,"CanD1",Basis_
    Dimension,DimBasisDefaultD1)) return;
  if (TestBasisDimension(ErrorMessage,name,"CanD2",Basis_
    Dimension,DimBasisDefaultD2)) return;
  if (TestBasisDimension(ErrorMessage,name,"CanD3",Basis_
    Dimension,DimBasisDefaultD3)) return;
  if (TestBasisDimension(ErrorMessage,name,"CanD4",Basis_
    Dimension,DimBasisDefaultD4)) return;
  if (TestBasisDimension(ErrorMessage,name,"CanD5",Basis_
    Dimension,DimBasisDefaultD5)) return;
  if (TestBasisDimension(ErrorMessage,name,"CanD6",Basis_
    Dimension,DimBasisDefaultD6)) return;
  if (TestBasisDimension(ErrorMessage,name,"CanD7",Basis_
    Dimension,DimBasisDefaultD7)) return;
  if (TestBasisDimension(ErrorMessage,name,"CanD8",Basis_
    Dimension,DimBasisDefaultD8)) return;
  if (TestBasisDimension(ErrorMessage,name,"CanD9",Basis_
    Dimension,DimBasisDefaultD9)) return;
  if (TestBasisDimension(ErrorMessage,name,"CanD10",Basis_
    Dimension,DimBasisDefaultD10)) return;
}

void Tensor_Delta(int j,int k, int BSdim,int *power){
  if (BSdim==1) {*power=j;
  }else  if (BSdim==2) {*power=TensorBasisD2[j][k];
  }else if (BSdim==3) {*power=TensorBasisD3[j][k];
  }else if (BSdim==4) {*power=TensorBasisD4[j][k];
  }else if (BSdim==5) {*power=TensorBasisD5[j][k];
  }else if (BSdim==6) {*power=TensorBasisD6[j][k];
  }else if (BSdim==7) {*power=TensorBasisD7[j][k];
  }else if (BSdim==8) {*power=TensorBasisD8[j][k];
  }else if (BSdim==9) {*power=TensorBasisD9[j][k];
  }else if (BSdim==10) {*power= TensorBasisD10[j][k];}
}
void Name_To_Basis(char *ErrorMessage, char *name,
           double (**UneBase)(double *x, int ind), int
    Basis_Dimension)
{
  /*initialization of the basis.*/
  if (strcmp("HerD1",name)==0){
  *UneBase=HermiteD1;
  } else if (strcmp("HerD2",name)==0){
  *UneBase=HermiteD2;
  } else if (strcmp("HerD3",name)==0){
  *UneBase=HermiteD3;
  } else if (strcmp("HerD4",name)==0){
  *UneBase=HermiteD4;
  } else if (strcmp("HerD5",name)==0){
  *UneBase=HermiteD5;
  } else if (strcmp("HerD6",name)==0){
  *UneBase=HermiteD6;
  } else if (strcmp("HerD7",name)==0){
  *UneBase=HermiteD7;
  } else if (strcmp("HerD8",name)==0){
  *UneBase=HermiteD8;
  } else if (strcmp("HerD9",name)==0){
  *UneBase=HermiteD9;
  } else if (strcmp("HerD10",name)==0){
  *UneBase=HermiteD10;
  } else if (strcmp("CanD1",name)==0){
  *UneBase=CanoniqueD1;
  } else if (strcmp("CanD2",name)==0){
  *UneBase=CanoniqueD2;
  } else if (strcmp("CanD3",name)==0){
  *UneBase=CanoniqueD3;
  } else if (strcmp("CanD4",name)==0){
  *UneBase=CanoniqueD4;
  } else if (strcmp("CanD5",name)==0){
```

```
  *UneBase=CanoniqueD5;
  } else if (strcmp("CanD6",name)==0){
  *UneBase=CanoniqueD6;
  } else if (strcmp("CanD7",name)==0){
  *UneBase=CanoniqueD7;
  } else if (strcmp("CanD8",name)==0){
  *UneBase=CanoniqueD8;
  } else if (strcmp("CanD9",name)==0){
  *UneBase=CanoniqueD9;
  } else if (strcmp("CanD10",name)==0){
  *UneBase=CanoniqueD10;
  } else {
  strcat(ErrorMessage,"Basis_Error");
  strcat(ErrorMessage,"(");
  strcat(ErrorMessage,name);
  strcat(ErrorMessage,")");
  return;
  }
  CheckBasisDimension(ErrorMessage,name,Basis_Dimension);
}

/*canonical basis, dimension=1..10*/
static double CanoniqueD1(double *x, int ind)
{
  auxd1=1;
  for (id1=0;id1<ind;id1++){
  auxd1*=(*x);
  }
  return auxd1;
}

static double CanoniqueD2(double *x, int ind)
{
  aux=1;
  for (i=0;i<2;i++){
  aux*=CanoniqueD1(x+i,TensorBasisD2[ind][i]);
  }
  return aux;
}

static double CanoniqueD3(double *x, int ind)
```

```
{
  aux=1;
  for (i=0;i<3;i++){
  aux*=CanoniqueD1(x+i,TensorBasisD3[ind][i]);
  }
  return aux;
}

static double CanoniqueD4(double *x, int ind)
{
  aux=1;
  for (i=0;i<4;i++){
  aux*=CanoniqueD1(x+i,TensorBasisD4[ind][i]);
  }
  return aux;
}

static double CanoniqueD5(double *x, int ind)
{
  aux=1;
  for (i=0;i<5;i++){
  aux*=CanoniqueD1(x+i,TensorBasisD5[ind][i]);
  }
  return aux;
}

static double CanoniqueD6(double *x, int ind)
{
  aux=1;
  for (i=0;i<6;i++){
  aux*=CanoniqueD1(x+i,TensorBasisD6[ind][i]);
  }
  return aux;
}

static double CanoniqueD7(double *x, int ind)
{
  aux=1;
  for (i=0;i<7;i++){
  aux*=CanoniqueD1(x+i,TensorBasisD7[ind][i]);
  }
  return aux;
}

static double CanoniqueD8(double *x, int ind)
{
  aux=1;
  for (i=0;i<8;i++){
  aux*=CanoniqueD1(x+i,TensorBasisD8[ind][i]);
  }
  return aux;
}

static double CanoniqueD9(double *x, int ind)
{
  aux=1;
  for (i=0;i<9;i++){
  aux*=CanoniqueD1(x+i,TensorBasisD9[ind][i]);
  }
  return aux;
}

static double CanoniqueD10(double *x, int ind)
{
  aux=1;
  for (i=0;i<10;i++){
  aux*=CanoniqueD1(x+i,TensorBasisD10[ind][i]);
  }
  return aux;
}

/*Hermite basis, dimension=1..10*/
static double HermiteD1(double *x, int ind)
{
  switch (ind){
  case 0 : return 1;
  case 1 : return 1.414213562*(*x);
  case 2 : return 1.414213562*(*x)*(*x)-0.707106781;
  case 3 : return (1.154700538*(*x)*(*x)-1.732050808)*(*x);
  case 4 : return (0.816496581*(*x)*(*x)-2.449489743)*(*x)*
    (*x)+0.612372436;
  case 5 : return ((0.516397779*(*x)*(*x)-2.581988897)*(*x)
    *(*x)+1.936491673)*(*x);

  default : return 1;
  }
}

static double HermiteD2(double *x, int ind)
{
  aux=1;
  for (i=0;i<2;i++){
  aux*=HermiteD1(x+i,TensorBasisD2[ind][i]);
  }
  return aux;
}

static double HermiteD3(double *x, int ind)
{
  aux=1;
  for (i=0;i<3;i++){
  aux*=HermiteD1(x+i,TensorBasisD3[ind][i]);
  }
  return aux;
}

static double HermiteD4(double *x, int ind)
{
  aux=1;
  for (i=0;i<4;i++){
  aux*=HermiteD1(x+i,TensorBasisD4[ind][i]);
  }
  return aux;
}

static double HermiteD5(double *x, int ind)
{
  aux=1;
  for (i=0;i<5;i++){
  aux*=HermiteD1(x+i,TensorBasisD5[ind][i]);
  }
  return aux;
}
```

```
static double HermiteD6(double *x, int ind)
{
  aux=1;
  for (i=0;i<6;i++){
  aux*=HermiteD1(x+i,TensorBasisD6[ind][i]);
  }
  return aux;
}

static double HermiteD7(double *x, int ind)
{
  aux=1;
  for (i=0;i<7;i++){
  aux*=HermiteD1(x+i,TensorBasisD7[ind][i]);
  }
  return aux;
}

static double HermiteD8(double *x, int ind)
{
  aux=1;
  for (i=0;i<8;i++){
  aux*=HermiteD1(x+i,TensorBasisD8[ind][i]);
  }
  return aux;
}

static double HermiteD9(double *x, int ind)
{
  aux=1;
  for (i=0;i<9;i++){
  aux*=HermiteD1(x+i,TensorBasisD9[ind][i]);
  }
  return aux;
}

static double HermiteD10(double *x, int ind)
{
  aux=1;
  for (i=0;i<10;i++){
```

```
  aux*=HermiteD1(x+i,TensorBasisD10[ind][i]);
  }
  return aux;
}
#endif //PremiaCurrentVersion
```

References