BM 593 Numerical Methods & C Programming

3rd week Basic Syntax of C Language

Rules for efficient C Coding

```
/* Bad Code */                  /* Good Code */
sum[i]=0.;
for (j=1; j<100; j++)           for (j=1,temp=0.; j<100; j++)
  sum[i]=a[i][j];                 temp+=a[i][j];
                                 sum[i]=temp;

for (i=1; i<100; i++)           for (i=1; i<100; i++)
  a[i][7]=c;                        a[i][7]=c;

/* rightmost index varies most rapidly  */

for (i=1; i<100; i++){            for (i=1; i<100; i++){
   a=b*x*y*z;                       a=b*(x*y*z);
   c=d*y*z*x;                       c=d*(x*y*z);
   e=x*y+f;                         e=(x*y)+f;
   g=h*y*x+z;                       g=h*(x*y)+z;
}                                   }

sum=0.;                           sum=0.;
for (i=1; i<100; i++)             for (i=1; i<100; i++)
  sum+=fact*a[i];                    sum+=a[i];
                                  sum*=fact;

double x[100],y[100];             double x[100],y[100],c;
for (i=0; i<100; i++){            for (i=0; i<100; i++){
  x[i]*=i;                          c+=1.;
  y[i]=x[i]/2.;                     x[i]*=c;
}                                   y[i]=0.5*x[i];
                                  }
```

Calling C Routines from MATLAB-mex Compilation

```
void shell(double **r, int q, double **s, int N, double **H)
/* THis is a routine to be called from the MATLAB environment */
/* shell.c calculates the electric potential for the 4 shell spherical model */
/* r is a 2 d array with q by 3 */
/* q is the number of dipoles */
/* s is the 2 d array with N by 3 */
/* N is the number of electrodes */
/* H is the potential field matrix with */
/* N by 3*q  */

/* mex file for shell.c   */
/* Usage: H=matshell[r,s]; */
/* Will be stored in matshell.c */

#include "\MATLAB6p5\extern\include\mex.h"
#include "\MATLAB6p5\extern\include\matrix.h"
#include "shell.c"

void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[]){
  int i,j,k,q,N;
  double **r,**s,**H,*data1,*data2;

  if (nrhs!=2){
     mexErrMsgTxt("2 input arguments needed");
  }
  if (nlhs>1){
     mexErrMsgTxt("1 output argument needed");
  }

  q=mxGetNumberOfElements(prh[0]/3);
  /* get the number of dipoles from r which is Number of Dipoles X 3 */
  N=mxGetNumberOfElements(prh[1]/3);
  /* get the number of electrodes from s which is Number of electrodes X 3 */

  r=(double **) mxMalloc(q*sizeof(double));
  for (i=0;i<q;i++)
```

```
    r[i]=(double *) mxMalloc(3*sizeof(double));
  data1=mxGetPr(prh[0]);
  for (k=0,j=0;j<3;j++)
    for (i=0;i<q;i++,k++)
      r[i][j]=data1[k];
  s=(double **) mxMalloc(N*sizeof(double));
  for (i=0;i<q;i++)
    s[i]=(double *) mxMalloc(3*sizeof(double));
  data1=mxGetPr(prhs[1]);
  for (k=0,j=0;j<3;j++)
    for (i=0;i<N;i++,k++)
      s[i][j]=data1[k];
  H=(double **) mxMalloc(N*sizeof(double));
  for (i=0;i<q;i++)
    H[i]=(double *) mxMalloc(q*3*sizeof(double));

  shell(r,q,s,N,H);

  plhs[0]=mxCreateDoubleMatrix(N,3*q,mxREAL);
  data2=(double *) mxMalloc(N*3*q*sizeof(double));
  for (k=0,j=0;j<(3*q);j++)
    for (i=0;i<N;i++,k++)
      data2[k]=H[i][j];
  mxSetPr(plhs[0],data2);
}
```

More Efficient C Coding for MATLAB-mex Compilation

```
void shell(double *r, int q, double *s, int N, double *H)
/* THis is a routine to be called from the MATLAB environment */
/* shell.c calculates the electric potential for the 4 shell spherical model */
/* r is a 1 d array with length q*3 */
/* q is the number of dipoles */
/* s is the 1 d array with length N*3 */
/* N is the number of electrodes */
/* H is the potential field vector with length N*3*q   */


/* mex file for shell.c   */
/* Usage: H=matshell[r,s]; */
```

```
/* Will be stored in matshell.c */

#include "\MATLAB6p5\extern\include\mex.h"
#include "\MATLAB6p5\extern\include\matrix.h"
#include "shell.c"

void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[]){
  int i,j,k,q,N;
  double *r,*s,*H;

  if (nrhs!=2){
     mexErrMsgTxt("2 input arguments needed");
  }
  if (nlhs>1){
     mexErrMsgTxt("1 output argument needed");
  }

  q=mxGetNumberOfElements(prh[0]/3);
  /* get the number of dipoles from r which is Number of Dipoles X 3 */
  N=mxGetNumberOfElements(prh[1]/3);
  /* get the number of electrodes from s which is Number of electrodes X 3 */

  r=mxGetPr(prh[0]);
  s=mxGetPr(prhs[1]);

  H=(double *) mxMalloc(N*3*q*sizeof(double));

  shell(r,q,s,N,H);

  plhs[0]=mxCreateDoubleMatrix(N,3*q,mxREAL);
  mxSetPr(plhs[0],H);
}
```

Passing Arrays to Numerical Recipes Routines

/* Numerical Recipes Routine for LU BACK SUBSTITUTION */

```
void lubksb(a,n,indx,b)
 double **a,b[];
 int n,*indx;
```

Array addressing in NR is from 1 to N.

```
/* For arrays passed to NR routines allocation is done as follows */
   void main(){

       double **Matrix;
       double *Vector;
       int *Index;

       Matrix = (double **) malloc(10*sizeof(double*));
       Matrix--;
       for (i=1;i<=10;i++)
         Matrix[i]-= (double *) malloc(10*sizeof(double));

       Vector = (double *) malloc(10*sizeof(double));
       Vector--;
       Index = (int *) malloc(10,sizeof(int));
       Index--;
   }
```