A. Codice dei principali programmi usati

Nella presente appendice è riportato il codice sorgente dei principali programmi utilizzati nel presente lavoro.

I programmi sono stati scritti in linguaggio “C++” e compilati su PC con compilatore “Borland C++ 5.01”. In alcuni casi si è fatto ricorso a routine in “C” di provenienza esterna (W. H. Press et al., 1992).

A.1 Mutua Informazione Media

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#define NR_END 1
#define FREE_ARG char*
void nrerror(char error_text[])
{
  fprintf(stderr,"Numerical Recipes run-time error...\n");
  fprintf(stderr,"%s\n",error_text);
  fprintf(stderr,"...now exiting to system...\n\n");
  exit(1);
}
float *vector(long nl, long nh)
{
  float *v;
  v=(float *)malloc((size_t) ((nh-nl+1+NR_END)*sizeof(float)));
  if (!v) nrerror("allocation failure in vector()");
  return v-nl+NR_END;
}
void free_vector(float *v, long nl)
```

```
{
  free((FREE_ARG) (v+nl-NR_END));
}
float **matrix(long nrl, long nrh, long ncl, long nch)
{
  long i, nrow=nrh-nrl+1,ncol=nch-ncl+1;
  float **m;
  m=(float **) malloc((size_t)((nrow+1)*sizeof(float*)));
  if (!m) nrerror("allocation failure 1 in matrix()");
  m += 1;
  m -= nrl;
  m[nrl]=(float *) malloc((size_t)((nrow*ncol+1)*sizeof(float)));
  if (!m[nrl]) nrerror("allocation failure 2 in matrix()");
  m[nrl] += 1;
  m[nrl] -= ncl;
  for(i=nrl+1;i<=nrh;i++) m[i]=m[i-1]+ncol;
  return m;
}
void free_matrix(float **m, long nrl, long ncl)
{
  free((char*) (m[nrl]+ncl-1));
  free((char*) (m+nrl-1));
}
void main()
{
  float *x, *y, *Px, *Py, **Pxy, *IT;
  float xmin=100000.0, xmax=-100000.0, ymin, ymax;
  float c, cc, d=0.1, r=0.025;//, Pxtot=0.0;
  int i, m, n, N, Nx, Ny, T, Tmax;
  bool found;
  FILE *f_in, *f_out;
  char file_in[128], file_out[128];
  printf("\n Dare nome file input -> ");
  scanf("%s", file_in);
  printf("\n Dare N -> ");
  scanf("%d", &N);
  printf("\n Dare Tmax -> ");
  scanf("%d", &Tmax);
  printf("\n Dare nome file output -> ");
  scanf("%s", file_out);
  f_out = fopen(file_out, "w");
  IT = vector(0,Tmax);
  for(i=0; i<=Tmax; i++) IT[i]=0.0;
  x = vector(1,N+Tmax);
  if((f_in = fopen(file_in, "r"))==NULL) {
    printf("\nError opening file \"%s\".\nExiting program...",
file_in);
    exit(0);
  }
  for(i=1; i<=N+Tmax; i++) fscanf(f_in, "%f", &x[i]);
  fclose(f_in);
  for(i=1; i<=N; i++) {
    if(x[i]<=xmin) xmin=x[i];
    if(x[i]>=xmax) xmax=x[i];
  }
  Nx=(int)ceil((xmax-xmin)/d);
  Px = vector(0,Nx);
  for(i=0; i<=Nx; i++) Px[i]=0.0;
  for(i=1; i<=N; i++) {
    found=false;
    for(m=0; m<=Nx; m++) {
      c = xmin+m*d;
```

```
    if( (x[i]>c-r) && (x[i]<=c+r) ) {
      Px[m]++;
      found=true;
    }
    if(found) break;
  }
}
for(i=0; i<=Nx; i++) Px[i]/=N;
for(T=0; T<=Tmax; T++) {
  y = vector(1,N);
  for(i=1; i<=N; i++) y[i]=x[i+T];
  ymin=100000.0;
  ymax=-100000.0;
  for(i=1; i<=N; i++) {
    if(y[i]<=ymin) ymin=y[i];
    if(y[i]>=ymax) ymax=y[i];
  }
  Ny=(int)ceil((ymax-ymin)/d);
  Py=vector(0,Ny);
  for(i=0; i<=Ny; i++) Py[i]=0.0;
  for(i=1; i<=N; i++) {
    found=false;
    for(m=0; m<=Ny; m++) {
      c = ymin+m*d;
      if( (y[i]>c-r) && (y[i]<c+r) ) {
        Py[m]++;
        found=true;
      }
      if(found) break;
    }
  }
  for(i=0; i<=Ny; i++) Py[i]/=N;
  Pxy=matrix(0,Nx,0,Ny);
  for(m=0; m<=Nx; m++) for(n=0; n<=Ny; n++) Pxy[m][n]=0.0;
  for(i=1; i<=N; i++) {
    found=false;
    for(m=0; m<=Nx; m++) {
      c = xmin+m*d;
      if( (x[i]>c-r) && (x[i]<c+r) ) {
        for(n=0; n<=Ny; n++) {
          cc = ymin+n*d;
          if( (y[i]>cc-r) && (y[i]<cc+r) ) {
            Pxy[m][n]++;
            found=true;
          }
          if(found) break;
        }
      }
      if(found) break;
    }
  }
  for(m=0; m<=Nx; m++) for(n=0; n<=Ny; n++) Pxy[m][n]/=N;
  for(m=0; m<=Nx; m++)
    for(n=0; n<=Ny; n++)
      if(Px[m]!=0.0 && Py[n]!=0.0 && Pxy[m][n]!=0.0)
        IT[T]+=Pxy[m][n]*(log(Pxy[m][n]/(Px[m]*Py[n]))/log(2.0));
  free_vector(y,1);
  free_vector(Py,0);
  free_matrix(Pxy,0,0);
  printf(" IT[%2d] = %f\n", T, IT[T]);
}
free_vector(x,1);
```

```
  free_vector(Px,0);
  for(i=0; i<=Tmax; i++) fprintf(f_out, "%d\t%f\n", i, IT[i]);
  fclose(f_out);
  free_vector(IT,0);
}
```

A.2 Falsi Vicini

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <conio.h>
#define NR_END 1
#define FREE_ARG char*
static float sqrarg=0.0;
#define SQR(a) (sqrarg=(a), sqrarg*sqrarg)
void nrerror(char error_text[])
{
  fprintf(stderr,"Numerical Recipes run-time error...\n");
  fprintf(stderr,"%s\n",error_text);
  fprintf(stderr,"...now exiting to system...\n\n");
  exit(1);
}
int *ivector(long nl, long nh)
{
  int *v;
  v=(int *)malloc((size_t) ((nh-nl+1+NR_END)*sizeof(int)));
  if (!v) nrerror("allocation failure in ivector()");
  return v-nl+NR_END;
}
void free_ivector(int *v, long nl)
{
  free((FREE_ARG) (v+nl-NR_END));
}
float *vector(long nl, long nh)
{
  float *v;
  v=(float *)malloc((size_t) ((nh-nl+1+NR_END)*sizeof(float)));
  if (!v) nrerror("allocation failure in vector()");
  return v-nl+NR_END;
}
void free_vector(float *v, long nl)
{
  free((FREE_ARG) (v+nl-NR_END));
}
float RMS(float *x, int n)
{
  float mean=0.0, rms=0.0;
  int i;
  for(i=1; i<=n; i++) mean += x[i];
  mean /= n;
  for(i=1; i<=n; i++) rms += fabs(x[i]-mean);
  rms /= n;
  return rms;
}
int Nea_nei(float *x, int np, int m, int T, int k, float* d)
{
  float dist, dist_min = 1000000.0;
  int i, i_min, j;
```

```
  for(i=1; i<=np; i++) {
    if(i!=k) {
      dist = 0.0;
      for(j=0; j<m; j++) dist += SQR(x[k+j*T]-x[i+j*T]);
      dist = sqrt(dist);
      if(dist<=dist_min) {
        dist_min = dist;
        i_min = i;
      }
    }
  }
  if(dist_min == 0.0) dist_min = 0.00000001;
  *d = dist_min;
  return i_min;
}
void main()
{
  int i, m, M_MAX, T;
  int N=0;
  float x, *y, rms;
  FILE *f_in, *f_out;
  char file_in[128], file_out[128];
  printf("\n Dare nome file input -> ");
  scanf("%s", file_in);
  if((f_in = fopen(file_in, "r"))==NULL) {
    printf("\nError opening file \"%s\".\nExiting program...\n\n",
file_in);
    exit(0);
  }
  while(!feof(f_in)) {
    fscanf(f_in, "%f", &x);
    N++;
  }
  printf("\n %d points read.\n", N);
  rewind(f_in);
  y = vector(1,N);
  for(i=1; i<=N; i++) fscanf(f_in, "%f", &y[i]);
  fclose(f_in);
  rms = RMS(y, N);
  printf("\n Dare nome file output -> ");
  scanf("%s", file_out);
  printf("\n Dare m max -> ");
  scanf("%d", &M_MAX);
  printf("\n Dare T -> ");
  scanf("%d", &T);
  printf("\n");
  f_out = fopen(file_out, "w");
  _setcursortype(_NOCURSOR);
  for(m=1; m<=M_MAX; m++) {
    int *nn, NP=N-(m-1)*T, fnn=0, px, py;
    float *dists, num1, num2, d;
    dists = vector(1,NP);
    nn = ivector(1,NP);
    printf(" m = %2d", m);
    px = wherex();
    py = wherey();
    for(i=1; i<=NP; i++) {
      nn[i] = Nea_nei(y, NP, m, T, i, &d);
      dists[i] = d;
      gotoxy(px+3, py);
      printf("%3d%%", i*100/NP);
    }
```

```
    for(i=1; i<=NP; i++) {
      num1 = fabs(y[i+m*T]-y[nn[i]+m*T]);
      num2 = sqrt(SQR(dists[i])+SQR(num1));
      if( ((num1/dists[i])>15.0) || ((num2/rms)>=2.0) ) fnn++;
    }
    free_vector(dists,1);
    free_ivector(nn,1);
    fprintf(f_out,"%d\t%f\t%d\t%d\n", m, (fnn*100.0)/NP, fnn, NP);
    printf("\n");
  }
  _setcursortype(_NORMALCURSOR);
  free_vector(y,1);
  fclose(f_out);
}
```

A.3 Box Counting Dimension

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#define LN2 0.69314718
#define MIN(A,B) ((A)<(B))?(A):(B)
#define NR_END 1
#define FREE_ARG char*
void nrerror(char error_text[])
{
  fprintf(stderr,"Numerical Recipes run-time error...\n");
  fprintf(stderr,"%s\n",error_text);
  fprintf(stderr,"...now exiting to system...\n\n");
  exit(1);
}
float *vector(long nl, long nh)
{
  float *v;
  v=(float *)malloc((size_t) ((nh-nl+1+NR_END)*sizeof(float)));
  if (!v) nrerror("allocation failure in vector()");
  return v-nl+NR_END;
}
void free_vector(float *v, long nl)
{
  free((FREE_ARG) (v+nl-NR_END));
}
int *ivector(long nl, long nh)
{
  int *w;
  w=(int *)malloc((size_t) ((nh-nl+1+NR_END)*sizeof(int)));
  if (!w) nrerror("allocation failure in vector()");
  return w-nl+NR_END;
}
void free_ivector(int *w, long nl)
{
  free((FREE_ARG) (w+nl-NR_END));
}
void main()
{
  float *x, *y, xmin, ymin, a, b, d, x0, y0, w, h;
  int *box, i, j, k, N=0, Db, ND, ndx, ndy;
  bool found;
  FILE *f_in;
```

```
  if((f_in = fopen("henon.txt", "r"))==NULL) {
    printf("\nError opening file \"%s\".\nExiting program...",
"henon.txt");
    exit(0);
  }
  while(!feof(f_in)) {
    fscanf(f_in, "%f\t%f", &a, &b);
    N++;
  }
  rewind(f_in);
  x=vector(1,N);
  y=vector(1,N);
  for(i=1; i<=N; i++) fscanf(f_in, "%f\t%f", &x[i], &y[i]);
  fclose(f_in);
  printf("\n %d points read.\n", N);
  printf("\n");
  x0=-1.5;
  y0=-0.4;
  w=3.0;
  h=0.8;
  for(i=9; i<=10; i++) {
    Db=0;
    d=MIN(w,h)/pow(2,i);
    ndx=(int)ceil(w/d);
    ndy=(int)ceil(h/d);
    ND=ndx*ndy;
    box=ivector(0,ND-1);
    for(j=0; j<ND; j++) box[j]=0;
    printf("\n ND=%8d\t", ND);
    for(k=0; k<ND;k++) {
      found=false;
      xmin=x0+(k%ndx)*d;
      ymin=y0+(k/ndx)*d;
      for(j=1; j<=N; j++) {
        if( (x[j]>xmin) && (x[j]<=xmin+d) &&
           (y[j]>ymin) && (y[j]<=ymin+d) ) found=true;
        if(found) break;
      }
      if(found) box[k]=1;
    }
    for(j=0; j<ND; j++) Db+=box[j];
    printf("log2(1/d)=%f\tlog2(N(d))=%f", log(1.0/d)/LN2,
log(Db)/LN2);
    free_ivector(box,0);
  }
  free_vector(x,1);
  free_vector(y,1);
}
```

A.4 Dimensione di Informazione

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#define LN2 0.69314718
#define MIN(A,B) ((A)<(B))?(A):(B)
#define NR_END 1
#define FREE_ARG char*
void nrerror(char error_text[])
```

```
{
  fprintf(stderr,"Numerical Recipes run-time error...\n");
  fprintf(stderr,"%s\n",error_text);
  fprintf(stderr,"...now exiting to system...\n\n");
  exit(1);
}
float *vector(long nl, long nh)
{
  float *v;
  v=(float *)malloc((size_t) ((nh-nl+1+NR_END)*sizeof(float)));
  if (!v) nrerror("allocation failure in vector()");
  return v-nl+NR_END;
}
void free_vector(float *v, long nl)
{
  free((FREE_ARG) (v+nl-NR_END));
}
int *ivector(long nl, long nh)
{
  int *w;
  w=(int *)malloc((size_t) ((nh-nl+1+NR_END)*sizeof(int)));
  if (!w) nrerror("allocation failure in vector()");
  return w-nl+NR_END;
}
void free_ivector(int *w, long nl)
{
  free((FREE_ARG) (w+nl-NR_END));
}
void main()
{
  float *x, *y, *box, xmin, ymin, a, b, d, x0, y0, w, h, Is;
  int i, j, k, N=0, ND, ndx, ndy;
  FILE *f_in;
  if((f_in = fopen("henon.txt", "r"))==NULL) {
    printf("\nError opening file \"%s\".\nExiting program...",
"henon.txt");
    exit(0);
  }
  while(!feof(f_in)) {
    fscanf(f_in, "%f\t%f", &a, &b);
    N++;
  }
  rewind(f_in);
  x=vector(1,N);
  y=vector(1,N);
  for(i=1; i<=N; i++) fscanf(f_in, "%f\t%f", &x[i], &y[i]);
  fclose(f_in);
  printf("\n %d points read.\n", N);
  x0=-1.5;
  y0=-0.4;
  w=3.0;
  h=0.8;
  for(i=2; i<=10; i++) {
    Is=0.0;
    d=MIN(w,h)/pow(2,i);
    ndx=(int)ceil(w/d);
    ndy=(int)ceil(h/d);
    ND=ndx*ndy;
    box=vector(0,ND-1);
    for(j=0; j<ND; j++) box[j]=0;
    printf("\n ND=%8d\t", ND);
    for(k=0; k<ND;k++) {
```

```
      xmin=x0+(k%ndx)*d;
      ymin=y0+(k/ndx)*d;
      for(j=1; j<=N; j++) {
        if( (x[j]>xmin) && (x[j]<=xmin+d) &&
            (y[j]>ymin) && (y[j]<=ymin+d) ) box[k]++;
      }
    }
    for(j=0; j<ND; j++)  box[j]/=N;
    for(j=0; j<ND; j++)
      if(box[j]>0.0)
        Is+=box[j]*log(1.0/box[j])/LN2;
    printf("log2(1/d)=%f\tI(d)=%f", log(1.0/d)/LN2, Is);
    free_vector(box,0);
  }
  free_vector(x,1);
  free_vector(y,1);
}
```

A.5 Funzione di Correlazione

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <conio.h>
#define SWAP2(a,b) temp=(a);(a)=(b);(b)=temp;
#define M2 3                /* serve a sort */
#define NSTACK 50           /* serve a sort */
#define MAX(A,B) ((A)>(B))?(A):(B)
#define N_R  1000
#define M_MIN  1
#define M_MAX  10
#define T 1
#define N_ROW   (M_MAX - M_MIN + 1)*3
#define BUF 150
#define NR_END 1
#define FREE_ARG char*
void nrerror(char error_text[])
{
  fprintf(stderr,"Numerical Recipes run-time error...\n");
  fprintf(stderr,"%s\n",error_text);
  fprintf(stderr,"...now exiting to system...\n\n");
  exit(1);
}
float *vector(long nl, long nh)
{
  float *v;
  v=(float *)malloc((size_t) ((nh-nl+1+NR_END)*sizeof(float)));
  if (!v) nrerror("allocation failure in vector()");
  return v-nl+NR_END;
}
int *ivector(long nl, long nh)
{
  int *w;
  w=(int *)malloc((size_t) ((nh-nl+1+NR_END)*sizeof(int)));
  if (!w) nrerror("allocation failure in vector()");
  return w-nl+NR_END;
}
void free_vector(float *v, long nl)
{
```

```
  free((FREE_ARG) (v+nl-NR_END));
}
void free_ivector(int *w, long nl)
{
  free((FREE_ARG) (w+nl-NR_END));
}
float Alpha(float *x, int n);
void Derive(float y[], float h, int n);
void sort(unsigned long n, float arr[]);
void main()
{
  int Np=0, i, j, k, m, W;
  float out[N_ROW][N_R], c[N_R], x, *y, r_max=1.0;
  FILE *f_in, *f_out;
  char file_in[BUF];//, file_out[BUF];
  printf("\n Dare nome file input -> ");
  scanf("%s", file_in);
  if((f_in = fopen(file_in, "r"))==NULL) {
    printf("\nError opening file \"%s\".\nExiting program...",
file_in);
    exit(0);
  }
  while(!feof(f_in)) {
    fscanf(f_in, "%f", &x);
    Np++;
  }
  printf("\n %d points read.\n", Np);
  rewind(f_in);
  y=vector(1,Np);
  for(i=1; i<=Np; i++) fscanf(f_in, "%f", &y[i]);
  fclose(f_in);
  W = (int)(1.0/log(1.0/Alpha(y, Np)));
  printf("\n\n Calculating distances... ");
  for(m=M_MIN; m<=M_MAX; m++)
  {
    int N=Np-(m-1)*T;
    long ND0=(N*(N-W-2))/2, ND;
    long i_d=1, j_d=1;
    float *d, a, b, h, r_tmp;
    printf("\n ND = %d", ND0);
    printf("\tm = %2d... ", m);
    d=vector(1,ND0);
    for(j=1; j<N-W; j++) {
      for(i=j+W+1; i<=N; i++) {
        float d_elem = 0.0;
        for(k=0; k<m; k++)
          d_elem += (y[j+k*T]-y[i+k*T])*(y[j+k*T]-y[i+k*T]);
        d_elem = sqrt(d_elem);
        if((d_elem>0) && (d_elem<r_max)) d[i_d++] = d_elem;
      }
    }
    printf("done.");
    ND=i_d;
    printf("  Sorting %d numbers... ", ND);
    sort(ND, d);
    printf("done.");
    a=log10(d[10]);
    b=log10(r_max);
    h=(b-a)/(N_R-1);
    for(i=0; i<N_R; i++)
    {
      r_tmp = pow(10.0, a+i*h);
```

```
      out[(m-M_MIN)*3][i]=log10(r_tmp);
      while((d[j_d]<r_tmp) && (j_d<=ND)) j_d++;
      c[i] = log10((float)j_d/(float)ND);
      out[(m-M_MIN)*3+1][i]=c[i];
    }
    free_vector(d,1);
    Derive(c, h, N_R);
    for(i=0; i<N_R; i++) out[(m-M_MIN)*3+2][i]=c[i];
  }
  free_vector(y,1);
  printf("\n done.");
  f_out = fopen("Udine_gpa.txt"/*file_out*/, "w");
  printf("\n\n Writing output... ");
  for(i=0; i<N_ROW/3; i++) fprintf(f_out, "r%d\tc%d\tdc%d\t",
                        i+M_MIN, i+M_MIN, i+M_MIN);
  fprintf(f_out, "\n");
  for(i=0; i<N_R; i++)
  {
    for(j=0; j<N_ROW; j++) fprintf(f_out, "%f\t", out[j][i]);
    fprintf(f_out, "\n");
  }
  fclose(f_out);
  printf("done.\n\n\tProgram Terminated.\n");
}
float Alpha(float *x, int n)
{
  float mean=0.0, sigma2=0.0, alpha=0.0;
  int i;
  for(i=1; i<=n; i++) mean += x[i];
  mean /= n;
  for(i=1; i<=n; i++) sigma2 += (x[i]-mean)*(x[i]-mean);
  for(i=1; i<=n; i++) alpha += (x[i+1]-mean)*(x[i]-mean);
  alpha /= sigma2;
  return alpha;
}
void Derive(float y[], float h, int n)
{
  float *t;
  float a0=1.0/12.0, a2=25.0/12.0, a4=4.0/3.0, a6=5.0/6.0,
a8=2.0/3.0;
  float a1=-1.0*a0, a3=-1.0*a2, a5=-1.0*a4, a7=-1.0*a6, a9=-1.0*a8;
  int i;
  if((t = calloc(n, sizeof(float))) == NULL)
  {
    printf("\nNot enough memory to allocate array\nExiting
program...");
    exit(0);
  }
  for(i=0; i<n; i++) t[i] = y[i];
  y[0] = (a3*t[0]+4*t[1]-3*t[2]+a4*t[3]-0.25*t[4])/h;
  y[1] = (0.25*t[0]+a7*t[1]+1.5*t[2]-0.5*t[3]+a0*t[4])/h;
  for(i=2; i<n-2; i++) y[i] = (a0*t[i-2]+a9*t[i-
1]+a8*t[i+1]+a1*t[i+2])/h;
  y[n-2] = (a1*t[n-5]+0.5*t[n-4]-1.5*t[n-3]+a6*t[n-2]+0.25*t[n-
1])/h;
  y[n-1] = (0.25*t[n-5]+a5*t[n-4]+3*t[n-3]-4*t[n-2]+a2*t[n-1])/h;
  free(t);
}
void sort(unsigned long n, float arr[])
{
  unsigned long i, ir=n, j, k, l=1;
  int jstack=0, *istack;
```

```
  float a, temp;
  istack = ivector(0,NSTACK);
  for(;;) {
    if(ir-l < M2) {
      for(j=l+1; j<=ir; j++) {
        a=arr[j];
        for(i=j-1; i>=1; i--) {
          if(arr[i] <= a ) break;
          arr[i+1]=arr[i];
        }
        arr[i+1] = a;
      }
      if(jstack == 0) break;
      ir=istack[jstack];
      l =istack[jstack-1];
      jstack-=2;
    }
    else {
      k = (l+ir) >> 1;
      SWAP2(arr[k], arr[l+1]);
      if(arr[l+1] > arr[ir])
      {
        SWAP2(arr[l+1], arr[ir]);
      }
      if(arr[l] > arr[ir])
      {
        SWAP2(arr[l], arr[ir]);
      }
      if(arr[l+1] > arr[l])
      {
        SWAP2(arr[l+1], arr[l]);
      }
      i=l+1;
      j=ir;
      a=arr[l];
      for(;;) {
        do i++; while(arr[i] < a);
        do j--; while(arr[j] > a);
        if(j<i) break;
        SWAP2(arr[i], arr[j]);
      }
      arr[l]=arr[j];
      arr[j]=a;
      jstack += 2;
      if(jstack > NSTACK) {
        printf("jstack > NSTACK");
        exit(0);
      }
      if(ir-i+1 >= j-l) {
        istack[jstack]=ir;
        istack[jstack-1]=i;
        ir=j-1;
      }
      else {
        istack[jstack]=j-1;
        istack[jstack-1]=l;
        l=i;
      }
    }
  }
  free_ivector(istack,0);
}
```

A.6 Esponente di Lyapunov

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <conio.h>
#define NR_END 1
#define FREE_ARG char*
static float sqrarg=0.0;
#define SQR(a) (sqrarg=(a), sqrarg*sqrarg)
void nrerror(char error_text[])
{
  fprintf(stderr,"Numerical Recipes run-time error...\n");
  fprintf(stderr,"%s\n",error_text);
  fprintf(stderr,"...now exiting to system...\n\n");
  exit(1);
}
int *ivector(long nl, long nh)
{
  int *v;
  v=(int *)malloc((size_t) ((nh-nl+1+NR_END)*sizeof(int)));
  if (!v) nrerror("allocation failure in ivector()");
  return v-nl+NR_END;
}
void free_ivector(int *v, long nl)
{
  free((FREE_ARG) (v+nl-NR_END));
}
float *vector(long nl, long nh)
{
  float *v;
  v=(float *)malloc((size_t) ((nh-nl+1+NR_END)*sizeof(float)));
  if (!v) nrerror("allocation failure in vector()");
  return v-nl+NR_END;
}
void free_vector(float *v, long nl)
{
  free((FREE_ARG) (v+nl-NR_END));
}
int Nea_nei(float *x, int np, int m, int T, int k)
{
  float dist, dist_min = 1000000.0;
  int i, i_min, j;
  for(i=1; i<=np; i++) {
    if(abs(i-k)>0) {      //periodo medio
      dist=0.0;
      for(j=0; j<m; j++) dist+=SQR(x[k+j*T]-x[i+j*T]);
      dist = sqrt(dist);
      if(dist<=dist_min) {
        dist_min=dist;
        i_min = i;
      }
    }
  }
  return i_min;
}
void main()
{
  int ND=0, NP, M, T, P, Ndist;
  int *nn;
  float dist, e, x, dt;
```

```
  float *y;
  FILE *f_in, *f_out;
  char file_in[128], file_out[128];
  printf("\n Nome file input -> ");
  scanf("%s", file_in);
  if((f_in = fopen(file_in, "r"))==NULL) {
    printf("\nError opening file \"%s\".\nExiting program...\n\n",
file_in);
    exit(0);
  }
  while(!feof(f_in)) {
    fscanf(f_in, "%f", &x);    //conta i dati
    ND++;
  }
  printf("\n Letti %d dati.\n", ND);
  rewind(f_in);
  y = vector(1,ND);
  for(int i=1; i<=ND; i++) fscanf(f_in, "%f", &y[i]); //legge i dati
  fclose(f_in);
  printf("\n Dimensione di embedding -> ");
  scanf("%d", &M);
  printf("\n Time delay -> ");
  scanf("%d", &T);
  printf("\n Tempo di campionamento -> ");
  scanf("%f", &dt);
  NP=ND-(M-1)*T;      //calcola il numero di punti dell'orbita
  printf("\n Step max -> ");
  scanf("%d", &P);
  nn=ivector(1,NP);
  printf("\n Nome file output -> ");
  scanf("%s", file_out);
  printf("\n");
  f_out = fopen(file_out, "w");
  for(int i=1; i<=NP; i++) nn[i]=Nea_nei(y, NP, M, T, i);
  for(int i=0; i<=P; i++) {
    Ndist=0;
    e=0;
    for(int j=1; j<=NP; j++) {
      dist=0.0;
      if(((j+i)<=NP) && ((nn[j]+i)<=NP)) {
        for(int k=0; k<M; k++)
          dist+=SQR(y[j+k*T+i]-y[nn[j]+k*T+i]);
      }
      if(dist>0) {
        Ndist++;
        e+=log(sqrt(dist));
      }
    }
    e/=Ndist;
    fprintf(f_out,"%f\t%f\n", i*dt, e);
  }
  free_vector(y,1);
  free_ivector(nn,1);
  fclose(f_out);
}
```

A.7 Wavelet Transform

```
#include <stdlib.h>
```

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <fstream.h>
#include <classlib\arrays.h>
static float sqrarg;
#define SQR(a) ((sqrarg=(a)) == 0.0 ? 0.0 : sqrarg*sqrarg)
inline float Arg(int x, int b, int a)
{
  return SQR(((float)x-b)/a);
}
inline float Psi(int x, int b, int a)
{
  return (1-Arg(x,b,a))*exp(-Arg(x,b,a)/2.0);
}
typedef TArray<float> Vector;
typedef TArrayIterator<float> VectorIter;
class Line : public Vector
{
  public :
    Line() : Vector(1000, 1, 100) {}
    bool operator ==(const Line& other) const
      { return &other == this; }
};
typedef TArray<Line> Matrix;
typedef TArrayIterator<Line> MatrixIter;
char Question(char* qst);
void WriteFile(char* filename, Vector x);
class TimeSerie
{
  public :
    Vector* data;
    int ndata;
    TimeSerie(char* filein);
    ~TimeSerie() { delete data; }
    Matrix Analysis(int amax);
    Vector GetData();
    void WriteOutput(char* fileout, Matrix coeff, int amax);
};
class WaveCoeff
{
  public:
    Matrix* coeff;
    int na, nb;
    WaveCoeff(char* filein);
    WaveCoeff(Matrix data_in, int na_in, int nb_in);
    ~WaveCoeff() { delete coeff; }
    Vector Synthesis(int rec_amin, int rec_amax);
};
TimeSerie::TimeSerie(char* filein)
{
  data = new Vector(1000, 1, 100);
  ifstream is(filein);
  if (!is)
  {
    printf("Can't open file %s", filein);
    exit(0);
  }
  data->Flush();
  while( !is.eof() )
  {
    float in_elem;
```

```
    is >> in_elem;
    data->Add(in_elem);
  }
  int n = data->GetItemsInContainer();
  data->Detach(n-1);
  ndata = data->GetItemsInContainer();
  printf("\n  ndata = %d", ndata);
}
Matrix
TimeSerie::Analysis(int amax)
{
  int yp = wherey()+1;
  gotoxy(3, yp);
  printf("Calculating transform coefficients [");
  int xp = wherex();
  Matrix coeff(amax, 1, 0);
  Line line;
  float out_elem;
  int xmin, xmax;
  for(int a=1; a<=amax; a++)
  {
    line.Flush();
    for(int b=1; b<=ndata; b++)
    {
      xmin = b - 4*a;
      xmax = b + 4*a;
      out_elem = 0.0;
      for(int x=xmin; x<=xmax; x++)
      {
        int i = x;
        if(    x<1) i = 1-x;
        if(x>ndata) i = 2*ndata-x+1;
        out_elem += Psi(x,b,a)*(*data)[i];
      }
      out_elem /= sqrt(a);
      line.Add(out_elem);
      gotoxy(xp, yp);
      printf("%3d,%4d]", a, b);
    }
    coeff.Add(line);
  }
  return coeff;
}
Vector
TimeSerie::GetData()
{
  Vector out(1000, 1, 100);
  VectorIter vi(*data);
  while(vi) out.Add(vi++);
  return out;
}
void
TimeSerie::WriteOutput(char* fileout, Matrix coeff, int amax)
{
  ofstream os(fileout);
  if(!os)
  {
    printf("Can't open file.");
    exit(0);
  }
  os <<'#'<<"\t"<<amax<<"\t"<<ndata<<"\n";
  MatrixIter ma(coeff);
```

```
  while(ma)
  {
    VectorIter ve(ma++);
    while( ve ) os << ve++ << "\t";
    os << "\n";
  }
  printf("\n\n  Coefficients written on file: %s", fileout);
}
WaveCoeff::WaveCoeff(char* filein)
{
  coeff = new Matrix(30, 1, 10);
  Line line;
  char dummy;
  ifstream is(filein);
  if (!is)
  {
    printf("Can't open file.");
    exit(0);
  }
  printf("\n  Reading data... ");
  is >> dummy >> na >> nb;
  coeff->Flush();
  float elem;
  for(int i=1; i<=na; i++)
  {
    line.Flush();
    for(int j=1; j<=nb; j++)
    {
      is >> elem;
      line.Add(elem);
    }
    coeff->Add(line);
  }
  printf("done.");
}
WaveCoeff::WaveCoeff(Matrix data_in, int na_in, int nb_in)
{
  coeff = new Matrix(30, 1, 10);
  Line line;
  MatrixIter mi(data_in);
  while(mi)
  {
    line.Flush();
    VectorIter vi(mi++);
    while( vi ) line.Add(vi++);
    coeff->Add(line);
  }
  na = na_in;
  nb = nb_in;
}
Vector
WaveCoeff::Synthesis(int rec_amin, int rec_amax)
{
  int yp = wherey()+1;
  gotoxy(3, yp);
  printf("Calculating reconstructed vector [");
  int xp = wherex();
  Vector out_vec(1000, 1, 100);
  Line line;
  float tmp_elem, out_elem;
  int bmin, bmax;
  out_vec.Flush();
```

```
  for(int x=1; x<=nb; x++)
  {
    out_elem = 0;
    MatrixIter mi(*coeff);
    mi.Restart(rec_amin, rec_amax);
    int a = rec_amin;
    while(mi)
    {
      tmp_elem = 0;
      bmin = x - 4*a;
      bmax = x + 4*a;
      for(int b=bmin; b<=bmax; b++)
      {
        int i = b;
        if( b<1) i = 1-b;
        if(b>nb) i = 2*nb-b+1;
        tmp_elem += Psi(x,b,a)*(mi.Current())[i];
      }
      out_elem += tmp_elem/pow(a,2.5);
      a++;
      mi++;
    }
    out_vec.Add(out_elem/3.14);
    gotoxy(xp, yp);
    printf("%d]", x);
  }
  return out_vec;
}
void main()
{
  int rec_amin_in, rec_amax_in;
  char answer = Question("Do you want to perform data analysis
(y/n)?");
  TimeSerie tmserie("input01.txt");
  Vector signal = tmserie.GetData();
  if( answer == 'y')
  {
    int amax_in;
    printf("\n  Input a max -> ");
    scanf("%d", &amax_in);
    Matrix coeff_in = tmserie.Analysis(amax_in);
    tmserie.WriteOutput("coeff.txt", coeff_in, amax_in);
    WaveCoeff wcoeff(coeff_in, amax_in, tmserie.ndata);
    printf("\n\n  Input a rec min ( >= 1 ) -> ");
    scanf("%d", &rec_amin_in);
    printf("\n  Input a rec max ( <= %d ) -> ", amax_in);
    scanf("%d", &rec_amax_in);
    Vector recostr = wcoeff.Synthesis(rec_amin_in, rec_amax_in);
    WriteFile("nn_rec.txt", recostr);
  }
  else
  {
    WaveCoeff wcoeff("coeff.txt");
    printf("\n\n  Input a rec min ( >= 1 ) -> ");
    scanf("%d", &rec_amin_in);
    printf("\n  Input a rec max ( <= %d ) -> ", wcoeff.na);
    scanf("%d", &rec_amax_in);
    Vector recostr = wcoeff.Synthesis(rec_amin_in, rec_amax_in);
    WriteFile("nn_rec.txt", recostr);
  }
  printf("\n\n  End of Program.");
}
```

```
char Question(char* qst)
{
  char answ = 'x';
  while( (answ != 'y') && (answ != 'n') )
  {
    printf("  %s ", qst);
    scanf("%c", &answ);
    if( answ == 'Y' ) answ = 'y';
    if( answ == 'N' ) answ = 'n';
  }
  return answ;
}
void WriteFile(char* filename, Vector x)
{
  ofstream os(filename);
  if(!os)
  {
    printf("Can't open file %s aborting.", filename);
    exit(0);
  }
  VectorIter xi(x);
  while( xi ) os << xi++  << "\n";
  printf("\n\n  Not normalized vector written on file: %s",
filename);
}
```

A.8 Modello Previsionale

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#define NR_END 1
#define FREE_ARG char*
static float sqrarg;
#define SQR(a) ((sqrarg=(a)) == 0.0 ? 0.0 : sqrarg*sqrarg)
void nrerror(char error_text[])
{
  fprintf(stderr,"Numerical Recipes run-time error...\n");
  fprintf(stderr,"%s\n",error_text);
  fprintf(stderr,"...now exiting to system...\n\n");
  exit(1);
}
int *ivector(long nl, long nh)
{
  int *v;
  v=(int *)malloc((size_t) ((nh-nl+1+NR_END)*sizeof(int)));
  if (!v) nrerror("allocation failure in ivector()");
  return v-nl+NR_END;
}
void free_ivector(int *v, long nl)
{
  free((FREE_ARG) (v+nl-NR_END));
}
float *vector(long nl, long nh)
{
  float *v;
  v=(float *)malloc((size_t) ((nh-nl+1+NR_END)*sizeof(float)));
  if (!v) nrerror("allocation failure in vector()");
  return v-nl+NR_END;
```

```
}
void free_vector(float *v, long nl)
{
  free((FREE_ARG) (v+nl-NR_END));
}
float **matrix(long nrl, long nrh, long ncl, long nch)
{
  long i, nrow=nrh-nrl+1,ncol=nch-ncl+1;
  float **m;
  m=(float **) malloc((size_t)((nrow+1)*sizeof(float*)));
  if (!m) nrerror("allocation failure 1 in matrix()");
  m += 1;
  m -= nrl;
  m[nrl]=(float *) malloc((size_t)((nrow*ncol+1)*sizeof(float)));
  if (!m[nrl]) nrerror("allocation failure 2 in matrix()");
  m[nrl] += 1;
  m[nrl] -= ncl;
  for(i=nrl+1;i<=nrh;i++) m[i]=m[i-1]+ncol;
  return m;
}

void free_matrix(float **m, long nrl, long ncl)
{
  free((char*) (m[nrl]+ncl-1));
  free((char*) (m+nrl-1));
}
bool Check(int i, int *y, int n)
{
  for(int j=1; j<=n; j++) if(y[j]==i) return false;
  return true;
}
int PrimoVicino(float **yo, float **yl, int i, int nr, int nc)
{
  int j, k, index=0;
  float d, d_min=10000000.;
  for(j=1; j<=nr; j++) {
    d=0.0;
    for(k=1; k<=nc; k++) d+=SQR(yo[i][k]-yl[j][k]);
    d=sqrt(d);
    if(d<d_min) {
      d_min=d;
      index=j;
    }
  }
  return index;
}
void PrimiVicini(float **yl, int *y_idx, int nn, int nr, int nc, int
Nx)
{
  int i, j, k, index=0;
  float d, d_min=10000000.;
  for(i=1; i<=Nx; i++) y_idx[i]=nn;
  for(i=2; i<=Nx; i++) {
    for(j=1; j<=nr; j++) {
      if( Check(j,y_idx,Nx) ) {
        d=0.0;
        for(k=1; k<=nc; k++) d+=SQR(yl[nn][k]-yl[j][k]);
        d=sqrt(d);
        if(d<d_min) {
          d_min=d;
          index=j;
        }
```

```
      }
    }
    y_idx[i]=index;
  }
}
float Dist(float **a, float **b, int ia, int ib, int m)
{
  float d = 0.0;
  for(int j=1; j<=m; j++) d+=SQR(a[ia][j]-b[ib][j]);
  if(d>0) return sqrt(d);
  else return 0.000001;
}
void Weight(float *dst, int n)
{
  float d_tot=0.0;
  for(int i=1; i<=n; i++)
  {
    dst[i] = 1.0/dst[i];
    d_tot += dst[i];
  }
  for(int i=1; i<=n; i++) dst[i] /= d_tot;
}
void main()
{
  int Ntot=0, N, NL, NP, NO, M, T, Nx, Ndays;
  int *y_idx;
  float x, *y, *dists, **yl, **yo, **yp, eqm=0.0;
  FILE *f_in, *f_out;
  char file_in[128], file_out[128];
  printf("\n Nome file input -> ");
  scanf("%s", file_in);
  if((f_in = fopen(file_in, "r"))==NULL) {
    printf("\n Error opening file \"%s\".\n Exiting program...",
file_in);
    exit(0);
  }
  while(!feof(f_in)) {
    fscanf(f_in, "%f", &x);
    Ntot++;
  }
  rewind(f_in);
  printf("\n Letti %d dati dal file \"%s\".\n", Ntot, file_in);
  printf("\n Dimensione di embedding -> ");
  scanf("%d", &M);
  printf("\n Delay time -> ");
  scanf("%d", &T);
  N=Ntot-(M-1)*T;
  printf("\n Numero di punti sull'attrattore: %d.\n", N);
  printf("\n Learning points (NL) -> ");
  scanf("%d", &NL);
  printf("\n Ampiezza previsione (giorni) -> ");
  scanf("%d", &Ndays);
  NO=N-NL;
  NP=NO-Ndays;
  printf("\n Punti previsti: %d\n", NP);
  printf("\n Vicini usati  -> ");
  scanf("%d", &Nx);
  printf("\n Dare nome file output -> ");
  scanf("%s", file_out);
  printf("\n");
  y  = vector(1,Ntot);
  yl = matrix(1,NL,1,M);
```

```
  yo = matrix(1,NO,1,M);
  yp = matrix(1,NP,1,M);
  dists = vector(1,Nx);
  y_idx = ivector(1,Nx);  //indici degli Nx primi vicini
  for(int i=1; i<=Ntot; i++) fscanf(f_in, "%f", &y[i]);
  fclose(f_in);
  for(int i=1; i<=NL; i++) for(int j=1; j<=M; j++) yl[i][j]=y[i+(j-
1)*T];
  for(int i=1; i<=NO; i++) for(int j=1; j<=M; j++)
yo[i][j]=y[NL+i+(j-1)*T];
  for(int i=1; i<=NP; i++) for(int j=1; j<=M; j++) yp[i][j]=0.0;
  for(int i=1; i<=NP; i++) {
    int nn = PrimoVicino(yo, yl, i, NL, M);  //calcola il primo
vicino di yo[i]
    PrimiVicini(yl, y_idx, nn, NL, M, Nx);  //calcola i primi vicini
di yl[nn]
    for(int ii=1; ii<=Nx; ii++) dists[ii]=Dist(yo,yl,i,y_idx[ii],M);
    Weight(dists, Nx);
    for(int j=1; j<=M; j++) {
      for(int k=1; k<=Nx; k++) {
        yp[i][j]+=dists[k]*yl[y_idx[k]+Ndays][j];
      }
    }
  }
  for(int i=1; i<=NP; i++)
    for(int j=1; j<=M; j++)
      eqm+=fabs(yo[i][j]-yp[i][j]);
  eqm=eqm/NP;
  f_out = fopen(file_out, "w");
   fprintf(f_out, "%d\t%f\n", Ndays, eqm);
  for(int i=1; i<=NP; i++)
    fprintf(f_out, "%f\t%f\n", yo[i][M], yp[i][M]);
  fclose(f_out);
  free_vector(y,1);
  free_vector(dists,1);
  free_matrix(yl,1,1);
  free_matrix(yo,1,1);
  free_matrix(yp,1,1);
  free_ivector(y_idx,1);
}
```