

DOE/ER/54130--T1

FINAL TECHNICAL REPORT

"Chaotic Dynamics in Plasma: Method of Symbolic Kinetic Equation"

Principal Investigators:

A.B. Rechester  
R.B. White

RECEIVED

DOE award number:

DE-FG02-91ER54130

AUG 28 1996

Project Period:

9/30/91 - 04/30/95

OSTI

We have developed a new method of analysis of turbulent plasma fluctuations presented in the enclosed paper. Also, find enclosed Source Codes developed for this project.

RECEIVED

AUG 28 1996

OSTI

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

MASTER

## **DISCLAIMER**

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

## **DISCLAIMER**

**Portions of this document may be illegible  
in electronic image products. Images are  
produced from the best available original  
document.**

The purpose of this paper is to introduce the method for identification of fluctuations governed by the same dynamics. The method of correlation functions or Fourier transform that are often used for the analysis of fluctuations in fluids, plasmas etc., do not allow one to do that. In order to demonstrate how our method works we give an explicit example presented in Fig. 1. These are the time records of  $X(t)$  and  $Z(t)$  generated by the Lorenz model<sup>1</sup> and are related to fluid velocity and temperature fluctuations in a simplified model of Bernard thermal convection. These two signals look quite different but as we know they are representations of the one attractor, governed by the same dynamics. Very little could be learned from the study of cross-correlation function  $C_{xz}(\tau) = \langle X(t + \tau) Z(t) \rangle$  (here the averaging is done over the time  $t$ ) or Fourier transforms of  $X(t)$  and  $Z(t)$ . Using the symbolic analysis presented in this paper we will be able to demonstrate that signals  $X(t)$  and  $Z(t)$  correspond to the same dynamical process. There also exists a geometrical method which allows to reconstruct the phase space dynamics from a single variable.<sup>2</sup> The advantages of our method are that it does not require the determination of dynamical system dimension and it works well in the presence of noise. Different methods of symbolic analysis of noisy chaotic signals is presented in reference<sup>3</sup>.

We begin with discretization of our signals:

$$X_n = X(t_0 + n\Delta t) \quad (1)$$

$$Z_n = Z(t_0 + n\Delta t)$$

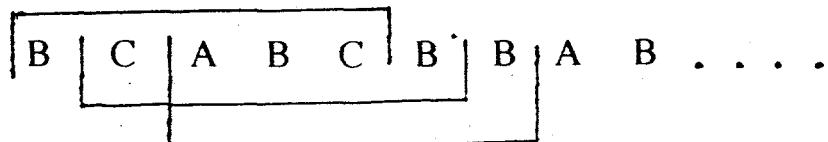
Here  $n = 0, 1, 2, \dots$  In the computations below we choose  $\Delta t=1$  because around this value our main result presented at Fig. 2 appears to be the most pronounced.

In order to recognize time patterns in complex dynamical processes we need to have a language in which to express these patterns. That is, one has to substitute actual signals  $X_n, Z_n$  with their symbolic representation.<sup>4</sup> We will be using a simple example of symbolic dynamics, defined by the following symbols:

$$S_n = \begin{cases} A (0) X_{min} < X_n < X_{C_1} \\ B (1) X_{C_1} < X_n < X_{C_2} \\ C (2) X_{C_2} < X_n < X_{C_3} < X_{max} \\ \dots \end{cases} \quad (2)$$

The range of the variable X has simply been divided into domains, separated by the critical points  $X_{C_i}$ . This symbolic representation of the signal thus consists of a series of the symbols. In the computations below we will often use integer numbers instead of symbols. The coarseness of this representation makes it clear that a very small choice for the time step  $\Delta t$  would not make this number series contain more information, it would simply make each number repeat several times before changing. The number of critical points necessary to obtain a faithful symbolic representation of the dynamics depends on the system studied. We describe below how this is determined. The symbolic dynamics for Z variable is defined in a similar manner, with critical points  $Z_{C_1}, Z_{C_2}, \dots$

The resulting long symbolic series we partition into short sequences of a given length L ( $L=5$  in the example below):



For ease of reference and identification it is convenient to identify every short sequence uniquely by just one integer.<sup>6,7</sup>

$$\ell = \sum_{i=1}^L 4^{L-i} S_i \quad (3)$$

Here we consider an example of four symbols, which we defined as integers according to Eq. (2).

The information content in the symbolic series can be quantified through the introduction of the metric entropy<sup>5</sup>.

$$E = - \lim_{L \rightarrow \infty} \left[ \frac{1}{L} \sum_{\ell} P_{\ell} \ln P_{\ell} \right] \quad (4)$$

Here  $P_{\ell}$  is the probability of finding a particular sequence  $\ell$  that is number of times this sequence can be found in the long symbolic time series divided by the number of all short sequences. The coding ability of the trial language defined by Eq. (2) depends strongly on the values of  $X_{C_i}$ . We performed a computer search to find out the optimum language which maximizes the metric entropy.

To maximize the entropy we introduce a given number of critical points (one, two, etc.) and study the metric entropy as a function of the placement of these critical points, by calculating the entropy for a given random placement of the  $X_{C_i}$ . A placement of these points is readily found which maximizes the entropy for a given number of critical points. If the number of critical points is too small, the symbolic language will however not contain the full information content of the signal. This is apparent by the

fact that the entropy will increase substantially when a new critical point is added. When sufficient critical points are present, the symbolic signal will contain essentially all the information contained in the original signal, and this is reflected by the fact that the addition of another critical point does not increase the entropy. At this point the set  $X_{C_i}$  can be said to give a faithful symbolic representation of the original signal, and the symbolic language can be called optimum. As we will see later, only a rough approximation to the optimum language is required.

The sequences  $\ell$  can be used for symbolic coarse graining of the phase space of the dynamical system.<sup>6,7</sup> Namely to every sequence  $\ell$  correspond a cell volume of the phase space  $\Delta\ell$ . In the case of degeneracy there can be several cells corresponding to one sequence  $\ell$ . We found numerically that the better approximations to the optimum language correspond to the less degenerate coarse grainings.

Now we are ready to do comparative analysis of the  $X_n$  and  $Z_n$  data, presenting them as symbolic states  $\ell_x(n)$ ,  $\ell_z(n)$ . If  $X(t)$  and  $Z(t)$  fluctuations are governed by different dynamics, then evolution of  $\ell_x(n)$  states and  $\ell_z(n)$  is not correlated. Namely every time the variable  $X$  occupies the state  $\ell_O$ , the variable  $Z$  could occupy any of the states available to it. On the other hand if  $X(t)$  and  $Z(t)$  are governed by the same dynamics then we will observe the following relationship between  $\ell_x(n)$  and  $\ell_z(n)$ : everytime the variable  $X$  occupies the state  $\ell_O$ , the variable  $Z$  can occupy only neighboring states. This is due to the fact that these states are just different symbolic coarse grainings of the same orbit<sup>6,7</sup>. We can easily destroy such a correlation by time shifting:  $\ell_x(n)$ ,  $\ell_z(n + n_O)$ . In order to check this effect we have computed the conditional entropy defined as:

$$E(Z|X) = -\frac{1}{N_\ell} \sum_{\ell_x} \frac{1}{L} \sum_{\ell_z|\ell_x} P(\ell_z|\ell_x) \ln P(\ell_z|\ell_x) \quad (5)$$

Here  $P(\ell_z|\ell_x)$  is a conditional probability for the variable  $Z$  to occupy state  $\ell_z$  while the variable  $X$  occupies state  $\ell_x$ ,  $N_\ell$  is the total number of different  $\ell_x$  sequences; the first summation in Eq. (5) is done over all dynamically accessible  $\ell_z$  states and fixed  $\ell_x$  states. The result of these computations is presented at Fig. 2. The sharp minimum of the conditional entropy as a function of a shift parameter  $n_0$  is a clear demonstration of the effect described above. For these computations we maximize entropy (4) with only one critical point ( $X_c = 0.07$ ,  $Z_c = 21.24$ ) while the full optimization requires approximately three critical points. On the other hand without optimization, if we choose some symbolic language with a relatively low entropy, then the minimum in Fig. 2 becomes much less pronounced or disappears.

We also applied the same symbolic analysis to a more complex signal generated from Lorenz model in the presence of external Gaussian noise. We have found that effect presented on Fig. 2 persists until a relatively high level of noise is reached. Thus symbolic analysis presented in this paper appears to be quite robust in the presence of external noise.

## ACKNOWLEDGMENTS

A. B. Rechester is thankful to Professor Edward N. Lorenz for useful discussions and encouragement. He would also like to acknowledge useful discussions and suggestions of Dr. Robert S. Granetz and Dr. Todd Evans. This work was partially supported by the U.S. Department of Energy under the contract DE-FG0291ER54130.

## REFERENCES

1. E.N. Lorenz, J. Atmos. Sci. 20, 130 (1963)
2. N.H. Packard, J.P. Crutchfield, J.D. Farmer and R.S. Shaw, Phys. Rev. Lett. 45, 712 (1980)
3. X.Z. Tang, E.R. Tracy, A.D. Boozer, A. deBrauw and R. Brown, Phys. Rev. E 51, 3871 (1995)
4. These are some mathematical papers on symbolic dynamics.  
Ya. G. Sinai, Russ. Math. Surv. 27, 21 (1972) R. Bowen, Equilibrium States and Ergodic Theory of Anosov Diffeomorphisms (Springer Lecture Notes in Mathematics) 470 (1975); D. Ruelle, Thermodynamic Formalism (Addison-Wesley, Reading, Ma 1978) V.M. Alekseev and M.V. Yakobson, Phys. Reports 75, 287-325 (1981)
5. P. Grassberger and H. Kants, Phys. Lett. A 113, 235 (1985)
6. A.B. Rechester and R.B. White, Phys. Lett. A156, 419 (1991)
7. A.B. Rechester and R.B. White, Phys. Lett. A 158, 51 (1991)

## FIGURE CAPTIONS

Fig. 1 Time records of  $X(t)$  and  $Z(t)$  generated by Lorenz model.

Fig. 2 Conditional entropy as a function of shift parameter  $n_O$ . Here we used 2 symbols,  $X_c = 0.07$ ,  $Z_c = 21.24$ ,  $L=5$ , the total number of points in time series is 4000 and  $\Delta t=1$ .

Lorenz Model, X & Z Time Records

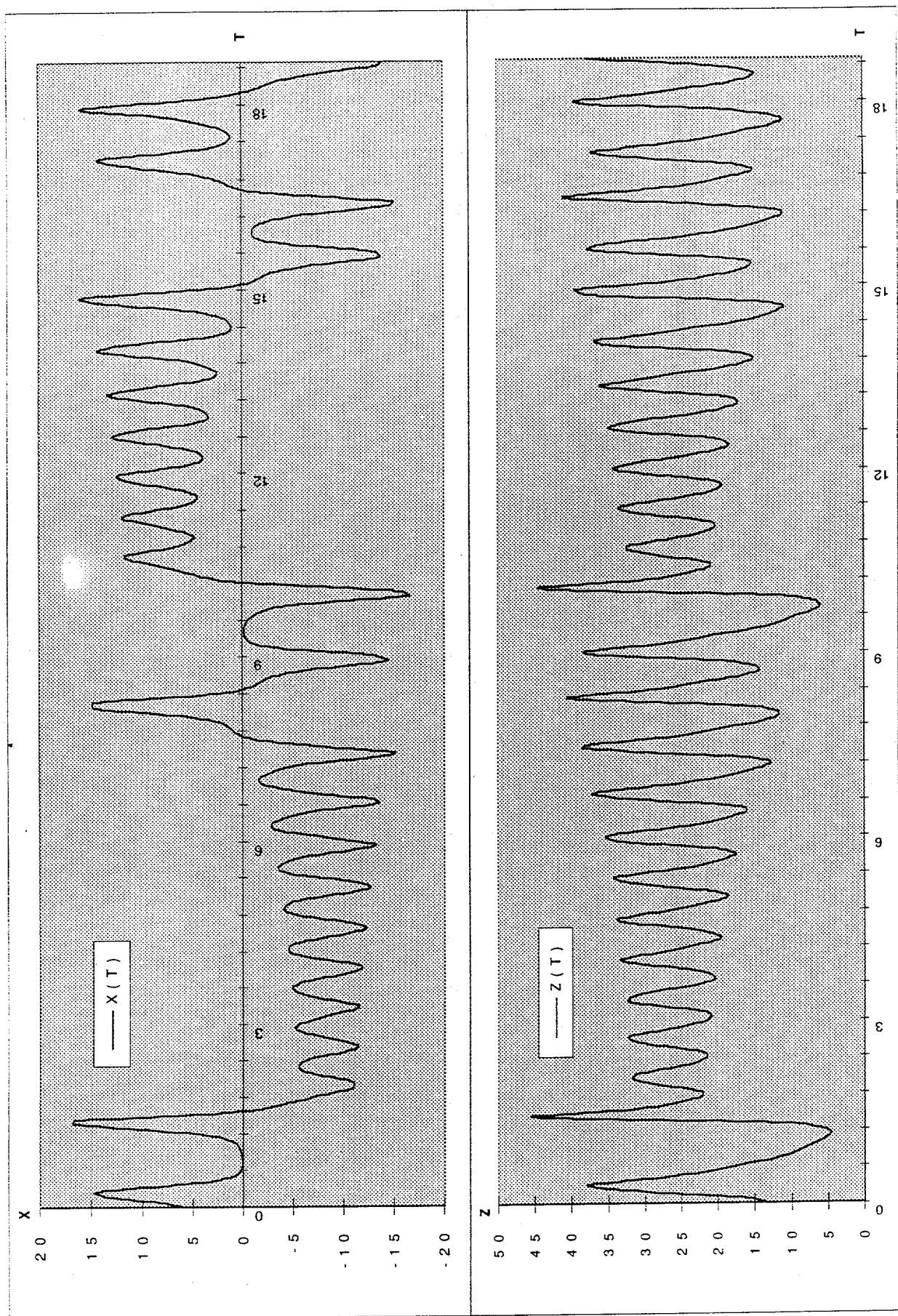
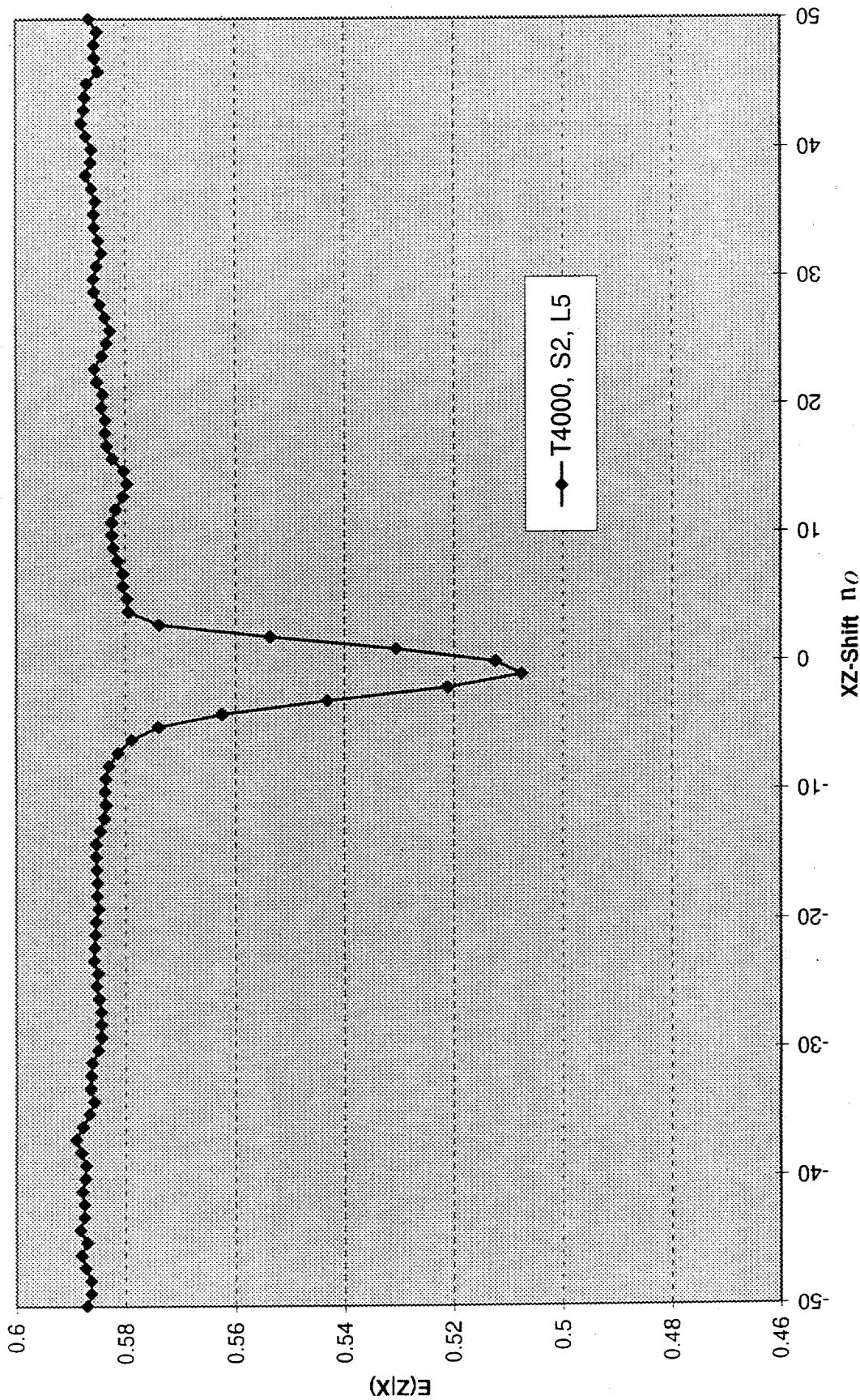


Fig. 1

### Lorenz Model, XZ-Correlation



### III. APPENDIX: SOURCE CODES

```

1      program main
2      implicit real (a-h,o-z)
3
4      common /lc/ aa,bb,rr,rr,rr,xl,xr,y1,yr,pi,p2i,nmult
5      common /vec/ xf0,yf0,xf1,xf2,y(99),x(99),n
6      common /d3/ ylo(10,10),yhi(10,10),nstep,ncount,model
7      common /stx/ lxi(9999),lx2(9999),imax1,imax2,iold1,iold2,ntoss
8
9      character*80 s100
10     istat=lsihell('rm -f mabout')
11     open(6,file='mabout','status=new',form='formatted')
12     call nearcgm(1,'gmetra')
13
14     R. B. White, May 1992 - code to evaluate metric entropy for various models
15     entvst runs until nstep = nmult*states, choose length n small to begin.
16     Grassberger used nmult=500 in Physics Letters 113, 235 (1985)
17     ccccc opt optimizes among random partitions, using much more time than entvst.
18     Number of states, and therefore computing time, varies strongly with the model being investigated. Always begin with n < 6
19     ccccc Do not run opt unless really desired! ntoss is the number of Monte Carlo placements of the partition used by opt.
20
21     acc pi = 4*atan(1.)
22     p2i = 1./p2i*pi
23
24     ccc call datlook
25     ccc go to 110
26     cccc choose model
27     ccc call convg
28     call quadrat
29     ccc call henon
30     ccc call cubic
31     ccc call chirikov
32     ccc call japan
33     ccc call hidim
34     ccc call datread
35     ccc call random
36     ccc call pparr
37     ccc call limlim
38     nmult = 5000
39     do 100 kcd = 1,1
40     aa = kcd*.0
41     n = 4
42     ccc call max
43     ccc call power
44     ccc call pwssym
45     ccc call statlist(1)
46     ccc call entvsk
47     ccc call entvst
48     ccc call deltas
49     ccc call frame(0)
50     ccc call contour
51     write(6,'(a*)') char(12)
52     100 continue
53     ccc opt uses nstep obtained by entvst
54     ntnoss = 100
55     ccc call opt
56     ccc call optdat
57     110 continue
58     call plot()
59     1234 call exit
60
61     subroutine quadrat
62     implicit real (a-h,o-z)
63     character*80 s100
64     common /lc/ aa,bb,rr,rr,xl,xr,y1,yr,pi,p2i,nmult
65     common /d3/ ylo(10,10),yhi(10,10),nstep,ncount,model
66     ccccc quadratic map parameters
67     model = 0
68     aa = .6
69     aa = 4.
70     call grid
71     return
72
73     subroutine henon
74     implicit real (a-h,o-z)
75     character*80 s100
76     common /vec/ xf0,yf0,xf1,xf2,y(99),x(99),n
77     common /d3/ ylo(10,10),yhi(10,10),nstep,ncount,model
78     ccccc henon parameters
79     model = 1
80
81     aa = 1.
82     bb = .54
83     rr = .5*(1+abs(bb)) + .5*sqrt((1+abs(bb))*2+4*aa)
84     xf1 = -.5*(1-bb) - .5*sqrt(4*aa+(1-bb)**2)
85     xf2 = -.5*(1-bb) + .5*sqrt(4*aa+(1-bb)**2)
86     call grid
87
88     return
89
90     subroutine cubic
91     implicit real (a-h,o-z)
92     character*80 s100
93     common /d3/ ylo(10,10),yhi(10,10),nstep,ncount,model
94     ccccc cubic map parameters
95     model = 2
96     aa = 14
97     call grid
98     return
99
100    subroutine chirikov
101    implicit real (a-h,o-z)
102    character*80 s100
103    common /d3/ aa,bb,rr,rr,xl,xr,y1,yr,pi,p2i,nmult
104    common /lc/ aa,bb,rr,rr,xl,xr,y1,yr,pi,p2i,nmult
105    ccccc chirikov map parameters
106    note if aa > 3.14 need nnum>7 in setiz, see mvar
107    model = 3
108    aa = 1.
109    call grid
110    return
111
112    subroutine japan
113    implicit real (a-h,o-z)
114    character*80 s100
115    common /lc/ aa,bb,rr,rr,xl,xr,y1,yr,pi,p2i,nmult
116    common /d3/ ylo(10,10),yhi(10,10),nstep,ncount,model
117    model = 4
118    aa = .1
119    bb = 12
120    rr = .0
121    call grid
122    return
123
124    subroutine hidim
125    implicit real (a-h,o-z)
126    character*80 s100
127    common /lc/ aa,bb,rr,rr,xl,xr,y1,yr,pi,p2i,nmult
128    common /d3/ ylo(10,10),yhi(10,10),nstep,ncount,model

```

```

129      model = 5
130      aa = 6.4
131      bb = .2
132      nr = 3
133      call grid
134      return
135
136      subroutine datread
137      implicit real (a-h,o-z)
138      character*80 s100
139      common /lc/ aa,bb,rr,rr,rr,rr,xl,xr,yl,yr,pi,p2i,nmult
140      common /d3/ ylo(10,10),yhi(10,10),nstep,ncount,model
141      ccc
142      model = 6
143      call grid
144      return
145
146      subroutine random
147      implicit real (a-h,o-z)
148      character*80 s100
149      common /lc/ aa,bb,rr,rr,rr,xl,xr,yl,yr,pi,p2i,nmult
150      common /d3/ ylo(10,10),yhi(10,10),nstep,ncount,model
151      ccc
152      model = 7
153      aa = 10
154      bb = 3
155      call grid
156      return
157
158      end
159      subroutine grid
160      implicit real (a-h,o-z)
161      character*80 s100
162      common /lc/ aa,bb,rr,rr,rr,xl,xr,yl,yr,pi,p2i,nmult
163      common /vec/ xf1,xf2,yf1,yf2,yf3,yf4,yf5,yf6,yf7,yf8,yf9,yf10
164      common /param/ xg(10),yg(10,10),beat(10,10),kgrid,kcrit
165      ccc
166      places partitions forming symbol domains according to critical points
167      ccc and lines see Grassberger Physics Letters 113, 235 (1985)
168      and Rechester White Physics Lett 158, 51 (1991)
169      if (model.eq.1) go to 10
170      if (model.eq.2) go to 20
171      if (model.eq.3) go to 30
172      if (model.eq.4) go to 40
173      if (model.eq.5) go to 50
174      if (model.eq.6) go to 60
175      if (model.eq.7) go to 60
176      ccc
177      kgrid = 1
178      kcrit = 1
179      yg(1,1) = .5
180      yg(1,2) = 1.
181      yg(1,1),kcrit+1) = 1.
182      ylo(1,1) = 0
183      yhi(1,1) = 1
184      return
185      10  continue
186      cccc
187      enter the Grassberger partition
188      kgrid = 5
189      kcrit = 1
190      xg(1) = .2
191      xg(2) = .6
192      xg(3) = .85
193      xg(4) = 1.4
194      xg(6) = rr
195      yg(1,1) = -.18
196      yg(2,1) = -.02
197      yg(3,1) = -.2
198      yg(4,1) = -.28
199      yg(5,1) = -.18
200      yg(6,1) = yg(5,1)
201      cccc
202      values for grid variation
203      do 70 kg = 1,kgrid
204      ylo(kg,1) = -.3
205      yhi(kg,1) = .2
206      70  continue
207      ylo(2,1) = -.2
208      return
209      20  continue
210      cubic map
211      kgrid = 1
212      kcrit = 2
213      yg(1,1) = .25
214      yg(1,2) = .75
215      yg(1,kcrit+1) = 1
216      ylo(1,1) = 0
217      ylo(1,2) = 0
218      yhi(1,2) = 1
219      return
220      30  continue
221      cccc
222      chirikov
223      kgrid = 1
224      ccc
225      ccc
226      w1 = acos(-1./aa)
227      w2 = 2*pi - w1
228      yg(1,1) = .7
229      yg(1,2) = 1.5
230      yg(1,kcrit+1) = 2*pi
231      ylo(1,1) = 0
232      ylo(1,2) = 0
233      yhi(1,1) = 2*pi
234      yhi(1,2) = 2*pi
235      40  continue
236      ccc
237      Japanese attractor
238      kgrid = 1
239      kcrit = 1
240      yg(1,1) = -1.
241      yg(1,2) = 4.
242      yg(1,3) = 4.
243      ylo(1,1) = -3.5
244      yhi(1,1) = 3.
245      ylo(1,2) = -3.5
246      yhi(1,2) = 3.
247      return
248      50  continue
249      ccc
250      high dimensional attractor
251      kgrid = 1
252      kcrit = 1
253      yg(1,1) = 0
254      yg(1,2) = 1.
255      yg(1,3) = 1.
256      ylo(1,1) = -3.5
257      yhi(1,1) = 3.
258      ylo(1,2) = -3.5

```

```

257      yhi(1,2) = 3.
258      return
259      60  continue
260      data read
261      kgrid = 1
262      kcrit = 1
263      yg(1,1) = 0.
264      yg(1,2) = 3.
265      yg(1,3) = .15
266      yg(1,kcrit+1) = 30.
267      ylo(1,1) = -.2
268      yhi(1,1) = .2
269      ylo(1,2) = -10
270      yhi(1,2) = 10
271      ylo(1,3) = -.2
272      yhi(1,3) = .2
273      return
274
275      subroutine rangrid(kt)
276      implicit real (a-h,o-z)
277      character*30 s100
278      common /lcr/ aa,bb,rr,rr,ml,x1,xr,y1,yr,p1,p21,nmult
279      common /vec/ xf0,yf0,xf1,xf2,yf99,x(99),n
280      common /d3/ ylo(10,10),yhi(10,10),nstep,ncount,model
281      common /param/ xq(10),yg(10,10),best(10,10),kgrid,kcrit
282      ccc   this is a simulated annealing procedure. The search for a
283      ccc   partition giving a larger entropy is made increasingly
284      ccc   nearer the best point as time goes on, but with always
285      ccc   some decreasing probability of looking far away.
286      rt = kt
287      pow = 1./rc**(.1./kcrit)
288      do 500 kg = 1,kgrid
289      do 500 kc = 1,kcrit
290      yh = yhi(kg,kc)
291      yl = ylo(kg,kc)
292      dum = ranf()
293      if(dum.gt..5) go to 60
294      del = ranf()*pow
295      yg(kg,kc) = yl + del*(best(kg,kc)-yl)
296      go to 500
297      60  continue
298      del = ranf()**pow
299      yg(kg,kc) = yh + del*(best(kg,kc)-yh)
300      500  continue
301      if(kcrit.gt.1) call sort
302      end
303      303  subroutine sort
304      implicit real (a-h,o-z)
305      character*30 s100
306      common /vec/ xf0,yf0,xf1,xf2,y(99),x(99),n
307      common /param/ xq(10),yg(10,10),best(10,10),kgrid,kcrit
308      ccc   put partition points in increasing order
309      do 5 kg = 1,kgrid
310      do 10 kc = 1,kcrit
311      y(kc) = yg(kg,kc)
312      10  continue
313      do 20 k = 1,kcrit
314      ccc   find minimum
315      dmin = 1.e6
316      do 30 kc = 1,kcrit
317      1f(y(kc).gt.dmin) go to 30
318      cmin = kc
319      dmin = y(kc)
320      30  continue

```

3

```

321      yg(kg,k) = y(kmin)
322      y(kmin) = 2.e6
323      20  continue
324      5  continue
325      return
326
327      function mvar(xd,yd)
328      implicit real (a-h,o-z)
329      common /param/ xq(10),yg(10,10),best(10,10),kgrid,kcrit
330      common /d3/ ylo(10,10),yhi(10,10),nstep,ncount,model
331      common /lcr/ aa,bb,rr,rr,x1,xr,y1,yr,p1,p21,nmult
332      ccc   assign symbol to xd,yd according to domain
333      1f(model.eq.1) go to 10
334      1f(model.eq.2) go to 20
335      1f(model.eq.3) go to 30
336      1f(model.eq.4) go to 40
337      1f(model.eq.5) go to 50
338      ccc   model = 6 same as quadratic
339      ccc   model = 7 same as quadratic
340      ccc   quadratic map
341      ccc   find yg(1,k) > xdum
342      xdum = xd
343      k = 0
344      5  continue
345      k = k + 1
346      346  if(k.gt.kcrit+1) go to 998
347      347  if(xdum.gt.yg(1,k)) go to 5
348      ccc   xd < yg(1,k)
349      mvar = k - 1
350      return
351      10  continue
352      cccc   henon
353      dum = yg(1,1)
354      if(xd.lt.xg(1)) go to 80
355      ccc   find xg(k) to right of xd
356      k = 0
357      60  continue
358      k = k + 1
359      d1 = xg(k)
360      360  if(xd.gt.d1) go to 60
361      cccc   km < xg(k)
362      km = k - 1
363      dum = yg(km,1) + (xd-xg(km))* (yg(k,1)-yg(km,1))/ (xg(k)-xg(km))
364      80  continue
365      md = .6*(1. + (yd-dum)/abs(yd-dum))
366      ccc   write(6,11) k,xd,yd,dum,md
367      11  format(' k,x,y,dum mvar ',13,1p3e12.4,14)
368      mvar = md
369      return
370      20  continue
371      ccc   cubic map
372      ccc   find yg(1,k) > xd
373      k = 0
374      7  continue
375      k = k + 1
376      376  if(k.gt.kcrit+1) go to 998
377      377  if(xd.gt.yg(1,k)) go to 7
378      ccc   xd < yg(1,k)
379      ccc   write(6,27) k,xd
380      27  format(' k,xd ',14,1p12.4)
381      mvar = k - 1
382      return
383      30  continue
384      cccc   chirikov

```

95/05/10

```

385      ccc      put yd on torus, count necessary steps
386      ndum = yd*p2i
387      if(yd.lt.0.) ndum = ndum - 1
388      yd = yd - 2*pi*ndum
389      yd now on torus, ndum steps
390      find yg(1,k) > yd
391      k = 0
392      continue
393      k = k + 1
394      if(k.gt.kcrit+1) go to 998
395      yd = xd - 2*pi*ndum
396      ccc      if(yd.gt.yg(1,k)) go to 15
397      ccc      yd < yg(1,k)
398      ccc      now put xd on torus
399      nd = xd*p2i
400      if(xd.lt.0.) nd = nd - 1
401      xd = xd - 2*pi*nd
402      if(xd.gt.pi) xd = xd - 2*pi
403      nd = 2 + (iqrid+1)*ndum
404      if(iabs(nd).gt.3) go to 999
405      mvar = nd
406      return
407      999      continue
408      98      write(6,98) nd
409      format(' not enough domains see function mvar  nd= ',16)
410      stop
411      40      continue
412      41      continue
413      42      Japan
414      43      find yg(1,k) > xd
415      75      continue
416      k = k + 1
417      if(k.gt.kcrit+1) go to 998
418      if(xd.gt.yg(1,k)) go to 75
419      ccc      xd < yg(1,k)
420      mvar = k - 1
421      return
422      continue
423      998      continue
424      97      write(6,97) k,xd,yg(1,k-1)
425      btop
426      50      continue
427      42      hidim
428      ccc      hidim
429      85      continue
430      k = 0
431      if(k.gt.kcrit+1) go to 998
432      if(xd.gt.yg(1,k)) go to 85
433      ccc      xd < yg(1,k)
434      435      mvar = k - 1
436      return
437      end
438      subroutine stepf(xf,yf)
439      implicit real (a-h,o-z)
440      common /d3/ ylo(10,10),yhi(10,10),natep,ncount,model
441      common /bb/ aa,bb,rr,rr,x1,xr,y1,yr,pi,p2i,nmult
442      common /bsb/ xav(20)
443      ccc      forward sequence
444      if(model.eq.1) go to 10
445      if(model.eq.2) go to 20
446      if(model.eq.3) go to 30
447      if(model.eq.4) go to 40
448      if(model.eq.5) go to 50

```

---

```

449      if(model.eq.6) go to 60
450      if(model.eq.7) go to 70
451      ccccc      sin
452      xf = sin(pi*xf)
453      return
454      ccc      hybrid
455      if(xf.lt. .5) xf = 4*xf*(1-xf)*aa + 2*xf*(1-aa)
456      if(xf.ge..5) xf = 4*xf*(1-xf)*aa + 2*(1-xf)*(1-aa)
457      return
458      ccc      quadratic map
459      xf = aa*xf*(1. - xf)
460      return
461      10      continue
462      cccc      henon
463      xn = aa + bb*yf - xf*xf
464      yf = xf
465      xf = xn
466      return
467      20      continue
468      cccc      cubic
469      xf = aa*xf*(xf - .75)**2
470      return
471      30      continue
472      cccc      chirikov
473      xf = xf + aa*sin(yf)
474      yf = yf + xf
475      return
476      40      continue
477      cccc      Japan
478      call stepdf(xf,yf)
479      return
480      50      continue
481      cccc      hidim
482      ccc      store previous up to mr
483      do 51 k = 2,mr
484      km k - 1
485      xav(k) = xav(km)
486      51      continue
487      xav(1) = xf
488      xf = (1-bb)*cos(aa*xav(mr-1)) + bb*xav(mr)
489      return
490      60      continue
491      cccc      data read
492      read(20,66) yf,xf
493      ccccc      edge fluct data 66
494      66      format(6x,2g15.7)
495      return
496      70      continue
497      cccc      gaussian random number source
498      101      continue
499      xd = 2*bb*(ranf()-.5)
500      500      Prob = exp(-(aa*xd)**2)
501      Pr = ranf()
502      if(prob.lt.pr) go to 101
503      xf = xd
504      return
505      end
506      subroutine metric
507      *      implicit real (a-h,o-z)
508      character*80 $100
509      common /stx/ lxl(99999),lxx(99999),imax1,imax2,iold1,iold2,ntoss
510      common /trc/ pri(99999),bri(100,100),ent,emin
511      common /trc/ pr2(99999),br2(999,900),n2(999,900)
512      ccc      Use normalized probabilities to calculate metric entropy = ent

```

```

513      dnorm = 0
514      do 10 id = 1,imax1
515        dnorm = dnorm + pr1(id)
516      10  continue
517      dum = 0
518      do 50 id = 1,imax1
519        dum = dum - pr1(id)*alog(pr1(id)/dnorm)/dnorm
520      50  continue
521        h1 = dum
522        dnorm = 0
523        do 20 id = 1,imax2
524          dnorm = dnorm + pr2(id)
525        20  continue
526          dum = 0
527          do 60 id = 1,imax2
528            dum = dum - pr2(id)*alog(pr2(id)/dnorm)/dnorm
529          60  continue
530            h2 = dum
531            ent = h1 - h2
532          return
533        end
534        subroutine position(xf,yf)
535          implicit real (a-h,o-z)
536          character*80 s100
537          common /vec/ xf0,yf0,xf1,xf2,y(99),x(99),n
538          common /d3/ ylo(10,10),yhi(10,10),nstep,ncount,model
539          common /dc/ aa,bb,rr,rr1,x1,xr,y1,yr,pi,p2l,nmult
540          ccc  set initial phase space point
541          if (model.eq.1) go to 10
542          if (model.eq.2) go to 20
543          if (model.eq.3) go to 30
544          if (model.eq.4) go to 40
545          if (model.eq.5) go to 50
546          ccc  quadratic map
547          xf = .2
548          yf = .2
549          return
550        10  continue
551          cccc  henon
552          xf = xf1 + .0001
553          yf = xf1 + .0001
554          return
555        20  continue
556          cccc  cubic
557          xf = .2
558          return
559        30  continue
560          cccc  chirikov
561          xf = .01
562          yf = .001
563          return
564        40  continue
565          cccc  Japan
566          xf = 0
567          yf = -1
568          return
569        50  continue
570          cccc  hidem
571          xf = 0
572          yf = -1
573          return
574        end
575        subroutine setup(xf,yf)
576          implicit real (a-h,o-z)
577      character*80 s100
578      common /mat/ mv(999),iz(999),mw,1w,ns2
579      common /stx/ l1(99999),l2(99999),imax1,imax2,iold1,iold2,ntoss
580      common /vec/ xt0,yf0,xf1,xf2,y(99),x(99),n
581      common /d3/ ylo(10,10),yhi(10,10),nstep,ncount,model
582      common /tric/ pr1(99999),br1(100,100),ent,enin
583      common /tric2/ pr2(99999),br2(900,900),n2(900,900)
584      ccc  first step of time sequence
585      ncount = 0
586      call setiz
587      imax1 = 0
588      imax2 = 0
589      mvar called each step to put point on torus (chirikov)
590      call position(xf,yf)
591      eliminate transients
592      do 5 kt = 1,10
593        call stepf(xf,yf)
594        mv(1) = mvar(xf,yf)
595      5  continue
596      ccc
597      ccc  forward sequence
598      call stepf(xf,yf)
599      mv(k) = mvar(xf,yf)
600      continue
601      call comprs
602      limit = mw
603      call newsat1(limit)
604      pr1(1) = 1
605      iold1 = 1
606      limit = limit - iz(1)*mv(1)
607      call newsat2(limit)
608      pr2(1) = 1
609      iold2 = 1
610      return
611      end
612      subroutine setupw(xf,yf)
613      implicit real (a-h,o-z)
614      character*80 s100
615      common /mat/ mv(999),iz(999),mw,1w,ns2
616      common /stx/ l1(99999),l2(99999),imax1,imax2,iold1,iold2,ntoss
617      common /vec/ xf0,yf0,xf1,xf2,y(99),x(99),n
618      common /d3/ ylo(10,10),yhi(10,10),nstep,ncount,model
619      common /tric/ pr1(99999),br1(100,100),ent,enin
620      common /tric2/ pr2(99999),br2(900,900),n2(900,900)
621      ccc  first step of time sequence
622      ncount = 0
623      call setiz
624      imax1 = 0
625      imax2 = 0
626      call position(xf,yf)
627      ccc  eliminate transients
628      do 5 kt = 1,10
629        call stepf(xf,yf)
630        mv(1) = mvar(xf,yf)
631      ccc  mvar called each step to put point on torus (chirikov)
632      5  continue
633      do 20 ks = 2,n
634      ccc  forward sequence
635      call stepf(xf,yf)
636      mv(k) = mvar(xf,yf)
637      20  continue
638      call comprs
639      limit = mw
640      call newrite(limit)

```

```

641      pri(1) = 1
642      ioldl = 1
643      return
644  end
645  subroutine mvax
646    implicit real (a-h,o-z)
647    character*80 s100
648    common /mat/ mv(99),iz(999),mw,1w,ns2
649    common /atx/ 1x1(99999),1x2(99999),imax1,imax2,ioldl,iold2,ntoss
650    common /vec/  xfo,yfo,xfl,xf2,yf(99),x(99),n
651    common /1c/  aa,bb,rr,rr,xl,xr,yr,pi,p21,nmult
652    common /q3/  ylo(10,10),yhi(10,10),nstep,ncount,model
653    common /tric/ pri(99999),br1(100,100),ent,emin
654    common /plt/ ww(40000),xx(999),yy(999),zz(40000),ymax,um
655    common /d3/ ylo(10,10),yhi(10,10),nstep,ncount,model
656    plot m value versus initial x
657    ncount = 0
658    call setiz
659    imax1 = 0
660    imax2 = 0
661    do 10 kk = 1,100
662      xf = .01*kk
663      xx(kk) = xf
664      do 5 kt = 1,n
665        mv(kt) = mvav(xf,yf)
666        call stepf(xf,yf)
667        continue
668        call compr8
669        yy(kx) = mw
670        linit = mw
671        do 20 km = 1,imax1
672          if(lx1(km).eq.mw) go to 30
673        continue
674        call newrite(linit)
675        30   continue
676        continue
677        call showa(xx,yy,100)
678        call setch(10.,155.,1.,0,2,0)
679        write(s100,35) n,aa
680        call gtext(s100,80,0)
681        35   format(' metric.f mvax n,a , ',i4,1pe12.4)
682        call frame(0)
683        return
684      end
685      subroutine datlook
686      implicit real (a-h,o-z)
687      character*80 s100
688      common /plt/ ww(40000),xx(999),yy(999),zz(40000),ymax,um
689      ccc  plot data string
690      kp = 0
691      do 100 k = 1,40000
692      cccc  data read
693      read(20,66) tf,xf
694      ccc  write(6,66) tf,xf
695      ccc  edgefluct 66  format(2g18.8)
696      format(6x,2g15.7)
697      ndum = mod(k,1)
698      if(ndum.ne.0) go to 10
699      kp = kp + 1
700      ww(kp) = xf
701      zz(kp) = tf
702      10   continue
703      call showa(zz,ww,kp)
704      call setch(10.,155.,1.,0,2,0)

```

---

```

705      write(s100,35)
706      call gtext(s100,80,0)
707      format(' metric.f data string ')
708      call frame(0)
709      return
710  end
711  subroutine advanc(xf,yf)
712    implicit real (a-h,o-z)
713    character*80 s100
714    common /mat/ mv(99),iz(999),mw,1w,ns2
715    common /vec/  xfo,yfo,xfl,xf2,yf(99),x(99),n
716    common /stx/  1x1(99999),1x2(99999),imax1,max2,ioldl,iold2,ntoss
717    common /d3/ ylo(10,10),yhi(10,10),nstep,ncount,model
718    common /tric/ pri(99999),br1(100,100),ent,emin
719    common /plt/ ww(40000),xx(999),yy(999),zz(40000),ymax,um
720    ccccc  advance xf,yf in time until ncount = nstep
721    10   continue
722    ncount = ncount + 1
723    if(ncount.gt.nstep) return
724    call stepf(xf,yf)
725    do 25 ks = 1,n
726      kp = ks + 1
727      mv(ks) = mv(kp)
728      25   continue
729      mv(n) = mvav(xf,yf)
730      call compr8
731      linit = mw
732      ccc  linit is in the list
733      do 50 ic = 1,imax1
734        inew = ic
735        if(linit.eq.lx1(ic)) go to 60
736      continue
737      call newstat1(linit)
738      inew = imax1
739      pri(inew) = 0
740      60   continue
741      pri(inew) = pri(inew) + 1
742      ccc  bri(inew,iold) = bri(inew,iold) + 1
743      iold = inew
744      ccc now find value for n-1
745      linit = linit - iz(1)*mv(1)
746      acc  linit is in the list
747      do 52 ic = 1,imax2
748        inew = ic
749        if(linit.eq.lx2(ic)) go to 62
750      52   continue
751      call newstat2(linit)
752      inew = imax2
753      pr2(inew) = 0
754      62   continue
755      pr2(inew) = pr2(inew) + 1
756      ccc  br2(inew,iold) = br2(inew,iold) + 1
757      iold2 = inew
758      go to 10
759      return
760    end
761    subroutine advanc1(xf,yf)
762    implicit real (a-h,o-z)
763    character*80 s100
764    common /mat/ mv(99),iz(999),mw,1w,ns2
765    common /vec/  xfo,yfo,xfl,xf2,yf(99),x(99),n
766    common /stx/  1x1(99999),1x2(99999),imax1,max2,iold1,iold2,ntoss
767    common /d3/ ylo(10,10),yhi(10,10),nstep,ncount,model
768    common /tric/ pri(99999),br1(100,100),ent,emin

```

```

769      common /tric2/ pr2(99999),br2(900,900),n2(900,900)
770      ccccccc advance xf,yf in time until ncount = nstep
771      10 continue
772      ncount = ncount + 1
773      if(ncount.gt.nstep) return
774      call stepf(xf,yf)
775      do 25 ks = 1,n
776          kp = ks + 1
777          mv(kp) = mv(kp)
778      25 continue
779      mv(n) = mvar(xf,yf)
780      call comprs
781      limit = mw
782      ccc limit is in the list
783      do 50 ic = 1,imax1
784      inew = ic
785      if(linit.eq.lx1(ic)) go to 60
786      continue
787      call newstat(linit)
788      inew = imax1
789      pri(inew) = 0
790      60 continue
791      pri(inew) = pri(inew) + 1
792      bri(inew,iold) = bri(inew,iold) + 1
793      ioldl = inew
794      go to 10
795      return
796
797      subroutine statlist(nwrit)
798      implicit real (a-h,o-z)
799      character*80 s100
800      common /lcr/ aa,bb,rr,mf,x1,xr,y1,yr,pi,p21,nmult
801      common /mat/ iz(999),iz(999),mw,1w,ns2
802      common /vec/ xf0,yf0,xf1,xf2,y(99),x(99),n
803      common /stx/ lxi(9999),lx2(9999),imax1,imax2,iold1,iold2,ntoss
804      common /d3/ ylo(10,10),yhi(10,10),nrep,ncount,model
805      common /tric/ pri(99999),br(100,100),ent,emin
806      common /tric2/ pr(99999),br2(900,900),n2(900,900)
807      cccccadvancexyf in time until ncount = nstep
808      cccc nwrite1 to write out all states and branching ratios
809      if(model.eq.6) rewind 20
810      if(nwrit.eq.1) write(6,15) aa,bb
811      15 format(' metric,f routine statlist a,b , ',lp2e12.4)
812      if(nwrit.eq.1) write(6,77) model,n,rr,mr
813      77 format('model, parameters n, rr, mr ',214,lpel2.4,14)
814      if(model.ne.6) write(6,97) nmult
815      97 format(' nmult ',16)
816      if(nwrit.eq.1) call writgrid
817      call setup(xf,yf)
818      iold1 = 1
819      ns = 0
820      ysav = 440
821      yfin = 480
822      98 continue
823      ns = ns + 1
824      ccc if(model.eq.6.and.ns.gt.52000) return
825      call stepf(xf,yf)
826      ccc if(yf.lt.ysav) go to 98
827      ccc if(yf.gt.yfin) go to 99
828      do 25 ks = 1,n
829          kp = ks + 1
830          mv(kp) = mv(kp)
831      25 continue
832      mv(n) = mvar(xf,yf)
833
834      call comprs
835      ccc limit = mw
836      do 50 ic = 1,imax1
837      inew = ic
838      if(linit.eq.lx1(ic)) go to 60
839      continue
840      call newstat(linit)
841      inew = imax1
842      if(imax1.lt.900) go to 51
843      write(6,52)
844      format(' more than 900 states ')
845      stop
846      51 continue
847      pri(inew) = 0
848      br2(inew,iold1) = 0
849      60 continue
850      pri(inew) = pri(inew) + 1
851      br2(inew,iold1) = br2(inew,iold1) + 1
852      iold1 = inew
853      ndum = imax1*nmult
854      if(ns.lt.ndum) go to 98
855      ccc
856      99 continue
857      dnorm = 0
858      dbran = 0
859      do 5 i = 1,imax1
860          dnorm = dnorm + pri(i)
861      5 continue
862      emet = 0
863      do 6 i = 1,imax1
864          pri(i) = pri(i)/(dnorm+1.e-10)
865          pri(i) = pri(i)*(dnorm+1.e-10)
866          emet = emet + pri(i)*alog(pri(i))
867      6 continue
868      emet = -emet/n
869      ent = emet
870      do 35 k = 1,imax1
871          dbran = 0
872          do 16 i = 1,imax1
873              n2(i,k) = br2(i,k) + .0001
874          dbran = dbran + br2(i,k)
875          16 continue
876          do 17 i = 1,imax1
877              br2(i,k) = br2(i,k)/(dbran+1.e-10)
878          17 continue
879          35 continue
880      if(nwrit.eq.0) return
881      if(model.eq.6) write(6,81) ysav,yf
882      81 format(' initial, final time ',lp2e12.4)
883      thres = .0
884      write(6,77) thres
885      77 format(' write only states with prob > ',lpel0.2)
886      do 22 i = 1,imax1
887          pr2(i) = lxi(i)
888          nd1 = lxi(i)
889          nd2 = 0
890          mdum = 0
891          do 23 k = 1,n
892              nd1 = nd1 - mdum*nd2
893              nd2 = 1z(k)
894              mdum = nd1/nd2
895              mv(k) = mdum
896      ccc
897      write(6,198) nd1,nd2,mv(k)

```

```

897      198    format(' nl,n2,mv ',315)
898      23     continue
899      1     if(nwrit.eq.1.and.pri(1).gt.thres)
900           write(6,75) lx1(i),pri(i),(mv(k),k=1,n)
901           format('state M= ',i6,' prob ',lpe12.4,' m=',2012)
902           continue
903           go to 779
904           nuniq = 0
905           do 24 k = 1,imax1
906               do 24 i = 1,imax1
907                   if(n2(i,k).eq.0.) go to 24
908                   if(br2(i,k).gt..999) nuniq = nuniq + 1
909                   if(nwrit.eq.1.and.pri(k).gt.thres)
910                       write(6,27) lx1(k),lx1(i),br2(i,k),n2(i,k)
911           format('branch ',i6,'-> ',i6,lpe16.4,' occur',110)
912           continue
913           write(6,91) emet
914           format(' sum -p(i)*log(p(i))/n ',lpe12.4)
915           dum = nuniq/(imax1+1.e-6)
916           write(6,87) imax1,nuniq,dum
917           87    format(i6,' states',i6,' unique trans, fraction',lpe12.4)
918           call sortlp
919           call showa(pr1,pri,imax1,0)
920           call setch(10.,2.1.,0.2,0)
921           write(s100,39) n_aa
922           format(' metric.f n,a composite, ',i4,lpe12.4)
923           call gtext(s100,80,0)
924           return
925       end
926   subroutine sortlp
927   implicit real (a-h,o-z)
928   character*80 s100
929   common /tric/ pr1(99999),br1(100,100),ent,emin
930   common /atx/  lx1(99999),lx2(99999),imax1,imax2,old1,old2,nfosa
931   common /tric2/ pr2(99999),br2(900,900),n2(900,900)
932   common /stx/  xx(99999),yy(99999)
933   do 5 k = 1,imax1
934     xx(k) = pr2(k)
935     yy(k) = pri(k)
936   5  continue
937   do 10 k = 1,imax1
938     pmn = 20
939   do 20 j = 1,imax1
940     if(xx(j).gt.pmn) go to 20
941     jmin = j
942   continue
943   20  continue
944   pmn = 20
945   do 20 j = 1,imax1
946     xx(jmin) = xx(j)
947   10  continue
948   continue
949   951  return
950
951
952   end
953   subroutine deltas
954   implicit real (a-h,o-z)
955   character*80 s100
956   common /mat/ mv(99),iz(999),mw,lw,nz2
957   common /lc/ aa,bb,rr,rr,x1,xr,y1,yr,p1,p2i,nmult
958   common /vec/ xf0,yf0,xf2,yf2(99),x(99),n
959   common /stx/ lx1(99999),lx2(99999),imax1,imax2,iold1,iold2,nfoss
960   common /d3/ ylo(10,10),yhi(10,10),nstep,ncount,model
961           common /plt/ ww(40000),xx(999),yy(999),zz(40000),ymax,nm
962           common /param/ xg(10),yg(10,10),best(10,10),kgrid,kcrit
963           common /tric/ pri(99999),br1(100,100),ent,emin
964           common /tric2/ pr2(99999),br2(900,900),n2(900,900)
965           call setup(xf,yf)
966           nstep = 100000
967           call advanci(xf,yf)
968           cc   normalize prob
969           dnorm = 0
970           do 5 i = 1,imax1
971               dnorm = dnorm + pri(i)
972           5  continue
973           do 6 i = 1,imax1
974               pri(i) = pri(i)/dnorm
975           6  continue
976           ccc   calculate entropy
977           dum = 0.
978           do 7 k = 1,imax1
979               dum = dum + pri(k)*alog(pri(k))
980           7  continue
981           call setiz
982           ent = -dum/n
983           983  do 100 k = 1,1000
984               xf = .001*(k-.5)
985           985  xx(k) = xf
986           986  mv(1) = mvar(xx,yf)
987           987  do 20 nd = 2,n
988               call stepf(xf,yf)
989           989  mv(nd) = mvar(xx,yf)
990           990  continue
991           991  call compr_a
992           992  yy(k) = mw
993           993  ccc   find probability
994           994  prob = 0
995           995  do 22 i = 1,imax1
996           996  if(mw.eq.lx1(i)) prob = pri(i)
997           997  continue
998           998  ww(k) = prob
999           100  continue
1000          call showa(xx,yy,1000,0)
1001          call setch(10.,155.,1.,0,2,0)
1002          write(s100,35) n_aa
1003          call gtext(s100,80,0)
1004          35  format(' metric.f routine deltas n,a , ',i4,lpe12.4)
1005          do 82 k = 1,kcrit
1006            write(s100,81) yg(1,k)
1007            call gtext(s100,80,0)
1008            81  format(' partition x ',lpe12.4)
1009            82  continue
1010            write(s100,89) imax1,ent
1011            format(' 1 values states, ent=',i5,lpe14.4)
1012            call gtext(s100,80,0)
1013            call frame(0)
1014            call showa(xx,ww,1000,0)
1015            call setch(10.,155.,1.,0,2,0)
1016            write(s100,35) n_aa
1017            call gtext(s100,80,0)
1018            do 83 k = 1,kcrit
1019              write(s100,81) yg(1,k)
1020              call gtext(s100,80,0)
1021              continue
1022              write(s100,87) imax1,ent,nstep
1023              format(' probab states,ent,steps=',i4,lpe12.4,18)
1024              call gtext(s100,80,0)

```

```

1025      return
1026      end
1027      subroutine limlim
1028      implicit real (a-h,o-z)
1029      character*80 s100
1030      common /mat/ mv(99),iz(999),mw,1w,nz2
1031      common /ic/ aa,bb,rr,rr,x1,xr,y1,yr,pi,p2i,nmult
1032      common /vec/ xf0,yf0,xf1,yf2,y(99),x(99),n
1033      common /astk/ lx(199999),lx(299999),imax1,imax2,iold1,iold2,ntoss
1034      common /d3/ ylo(10,10),yhi(10,10),natep,ncount,model
1035      common /pit/ ww(40000),xx(999),yy(999),zz(40000),ymax,nm
1036      common /param/ xg(10,10),yg(10,10),best(10,10),kgrid,kcrit
1037      common /tric/ pr1(9999),br1(100,100),ent,emin
1038      common /tric2/ pr2(99999),br2(900,900),n2(900,900)
1039      common /lin/ xz(999),yz(999),wz(999)
1040      write(6,15),aa,bb
1041      format(' metric.f routine limlim a,b , ',1p2e12.4)
1042      call writgrid
1043      nmult = 500
1044      itop = 2000
1045      write(6,77) model,nmult,itop
1046      format(' model,nmult,max states ',14,218)
1047      cccc
1048      nptt = 0
1049      n = 2
1050      97  continue
1051      n = n + 2
1052      npts = 0
1053      call setup(xf,yf)
1054      nstep = 1000
1055      99  continue
1056      nstep = 2*nstep
1057      call advanc(xf,yf)
1058      cc
1059      dnorm = 0.
1060      do 5 i = 1,imax1
1061      dnorm = dnorm + pr1(i)
1062      5  continue
1063      cccc
1064      find relative rms deviation of probabilities
1065      pr1(i) = pr1(i)/dnorm
1066      6  continue
1067      ccc
1068      calculate top and metric entropy
1069      r2 = 0
1070      dum = 0
1071      dum = 0
1072      do 7 k = 1,imax1
1073      dq = pr1(k)*alog(pr1(k)) - 1./imax1*i2
1074      r2 = r2 - dq*(pr1(k) - 1./imax1)*i2
1075      dum = dum + dq
1076      7  continue
1077      r2 = sqrt(r2)/d2
1078      npts = npts + 1
1079      xx(npts) = 1./nstep
1080      yy(npts) = -dum/n
1081      stat = imax1
1082      kw(npts) = alog(stat)/n
1083      ndum = nmult*imax1
1084      if (ndum.gt.nstep) go to 99
1085      if (npts.lt.2) go to 99
1086      finished steps, extrapolate
1087      npts = npts - 1
1088      dmst = yy(npts) - xx(npm)*(yy(npm)-yy(npts))/(xx(npm)-xx(npts))

```

---

```

1089      dtop = ww(npts) - xx(npts)*(ww(npm)-ww(npts))/(xx(npm)-xx(npts))
1090      nppt = nppt + 1
1091      xz(nppt) = 1./n
1092      yz(nppt) = dmet
1093      wz(nppt) = dtop
1094      format(' n-states, intercept metric, top ',13,16,1p2e14.6)
1095      91  format(' n-states, intercept metric, top ',13,16,1p2e14.6)
1096      if (nppt.lt.2) go to 97
1097      if (nppt.lt.2) go to 97
1098      ccc
1099      finished advancing n, extrapolate
1100      rpm = nppt - 1
1101      rpm = rpm - 1
1102      dl = yz(nppt) - xz(nppt)*(yz(npmt)-yz(nppt))/(xz(npmm)-xz(npmt))
1103      d2 = xz(npmt)*(yz(npmm)-yz(npmt))/(xz(npmm)-xz(npmt))
1104      err = abs(d2-d1)
1105      write(6,92),dl,err
1106      format(' extrapolate in n, metric= ',1pe14.6,' +-',1pe12.4)
1107      d3 = wz(nppt) - xz(nppt)*(wz(npmm)-wz(nppt))/(xz(npmm)-xz(npmt))
1108      err2 = abs(d4-d3)
1109      write(6,93),d3,err2
1110      format(' extrapolate in n,topological= ',1pe14.6,' +-',1pe12.4)
1111      ccc
1112      return
1113      call show(xz,yz,nppt)
1114      write(s100,16)
1115      format(' metric entropy')
1116      call gtext(s100,80,0)
1117      write(s100,15) aa,bb
1118      call qtext(s100,80,0)
1119      call wgridp
1120      write(s100,77) model,nmult,itop
1121      call gtext(s100,80,0)
1122      write(s100,92) dl,err
1123      call gtext(s100,80,0)
1124      call frame(0)
1125      call show(xz,wz,nppt,0)
1126      call btext(10.,50.,1.,0,2,0)
1127      write(s100,17)
1128      format(' topolog entropy')
1129      write(s100,15) aa,bb
1130      call gtext(s100,80,0)
1131      call wgridp
1132      write(s100,77) model,nmult,itop
1133      call gtext(s100,80,0)
1134      write(s100,92) dl,err
1135      call qtext(s100,80,0)
1136      return
1137
1138      subroutine writgrid
1139      implicit real (a-h,o-z)
1140      character*80 s100
1141      common /d3/ ylo(10,10),yhi(10,10),natep,ncount,model
1142      common /param/ xg(10,10),yg(10,10),best(10,10),kgrid,kcrit
1143      common /astk/ lx(199999),lx(299999),imax1,imax2,iold1,iold2,ntoss
1144      kcp = kcrt
1145      l1(kgrid,gt,1) go to 10
1146      write(6,50)(yg(1,ke),ke=1,kcp)
1147      50  format(' partition ',1p7e12.4)
1148      10  go to 20
1149      10  continue
1150      do 82 k = 1,kgrid
1151      write(6,81) xg(k),yg(k,1)
1152      81  format(' grid x,y ',1p2e12.4)

```



```

1281      do 105 k = 1,4
1282        kp = k + nd
1283        xx(k) = xx(kp)
1284        yy(k) = yy(kp)
1285      continue
1286      npts = 4
1287      100 continue
1288      call show(xx,yy,npts,0).
1289      call setch(10.,60.,1.,0,2,0)
1290      write(s100,79) aa,bb,nstep,ntoss
1291      call gtext(s100,80,0)
1292      79 format('metric.f aa,bb',lp2e10.2,' nstep,ntoss ',2110)
1293      write(s100,80) model,n,imaxl,ent,emin
1294      80 format('model,n,imax,ent,emin',3i6,1p3e12.4)
1295      acc  i(ntos,ne,0) call wdatp
1296      rpm = npts - 1
1297      dinter = yy(npts)*(yy(npm)-yy(npts))/ (xx(npm)-xx(npts))
1298      write(s100,91) dinter
1299      call gtext(s100,80,0)
1300      91 format(' intercept ',1pe14.6)
1301      call wgridp
1302      call frame(0)
1303      call ppart
1304      return
1305
1306
1307      subroutine entvax
1308      implicit real (a-h,o-z)
1309      character*80 s100
1310      common /mat/ mv(99),iz(999),mw,lw,ns2
1311      common /vec/ xf0,yf0,xfl,xfl2,yr(99),x(99),n
1312      common /1c/ aa,bb,rr,rr,xi,xr,yr,pi,p21,nmult
1313      common /plt/,ww(40000),xx(999),yy(999),zz(40000),ymax,im
1314      common /stck/ xl(99999),lx(99999),imax1,imax2,iold1,iold2,ntoss
1315      common /d3/ yl(10,10),yh(10,10),nstep,ncount,model
1316      common /param/ xq(10),yr(10,10),best(10,10),kgrid,kcrit
1317      common /tric/ pri(99999),bri(100,100),ent,emin
1318      common /tric2/ pr2(99999),br2(900,900),n2(900,900)
1319      acc  plot metric entropy vs partition
1320      npts = 60
1321      do 20 jd = 1,npts
1322        yg(1,1) = -.2 + (.jd-.5)*.4/npts
1323      if(model.eq.6) rewind 20
1324
1325      78 format(' initial time ',1pe12.4)
1326      nstep = 10000
1327      if(model.eq.6.and.nstep.gt.52000) return
1328      do 10 k = 1,nstep
1329        call advanc(xf,yf)
1330
1331      10 continue
1332      call metric
1333
1334      yy(jd) = ent
1335      if(model.eq.6) write(6,81) ysav,yf
1336      format(' initial, final time ',1p2e12.4)
1337      81 continue
1338      xx(jd) = ent
1339      97 format(' nstep,states, ent ',2i8,1pe16.6)
1340      call show(xx,yy,npts,0)
1341      call setch(10.,10.,120.,1.,0,2,0)
1342      write(s100,79) aa,bb,nstep
1343      call gtext(s100,80,0)
1344      format('metric.f, entvsk aa,bb',lp2e10.2,' nstep',i10)
1345      write(s100,'80) model,n,imaxl,ent,emin
1346      call gtext(s100,80,0)
1347      80 format('model,n,imax,ent,emin',3i6,1p3e12.4)
1348      if(model.eq.6) write(s100,81) ysav,yf
1349      if(model.eq.6) call gtext(s100,80,0)
1350      call wgridp
1351      return
1352
1353      subroutine ppart
1354      implicit real (a-h,o-z)
1355      character*80 s100
1356      common /mat/ mv(99),iz(999),mw,lw,ns2
1357      common /vec/ xf0,yf0,xfl,xfl2,yr(99),x(99),n
1358      common /1c/ aa,bb,rr,rr,xi,xr,yr,pi,p21,nmult
1359      common /plt/,ww(40000),xx(999),yy(999),zz(40000),ymax,im
1360      common /stck/ xl(99999),lx(99999),imax1,imax2,iold1,iold2,ntoss
1361      common /d3/ yl(10,10),yh(10,10),nstep,ncount,model
1362      common /param/ xq(10),yr(10,10),best(10,10),kgrid,kcrit
1363      common /tric/ pri(99999),bri(100,100),ent,emin
1364      cccc  plot the partition (not useful for all models)
1365      if(model.eq.1) go to 10
1366      if(model.eq.2) go to 20
1367      if(model.eq.3) go to 30
1368      if(model.eq.4) go to 40
1369      if(model.eq.5) return
1370
1371      cccc  quadratic map
1372      10 continue
1373      10 continue
1374      cccc  henon
1375      xmin = -rr
1376      xmax = rr
1377      ymin = -rr
1378      ymax = rr
1379      xf = xfl + .01
1380      yf = xf1 + .02
1381      npts = 1000
1382      do 15 k = 1,npts
1383        call step(xf,yf)
1384        ww(k) = xf
1385        zz(k) = yf
1386
1387      15 continue
1388      call maps(xmin,xmax,ymin,ymax,-1.0,-1.0,-1.0,-1.0)
1389      call points(ww,zz,npts,-1,-1,0,0,0,0)
1390      kgp = kgrid + 2
1391      do 17 kc = 1,kcrit
1392        do 18 kq = 1,kgrid
1393          kp = kg + 1
1394          ww(kp) = xg(kg)
1395          zz(kp) = yg(kg,kc)
1396          continue
1397          ww1 = -rr
1398          zz1 = yg(1,kc)
1399          ww(kgp) = rr
1400          call trace(ww,zz,kgp,-1,-1,0,0,0,0)
1401          17 continue
1402          call setch(10.,5.,1,0,2,0)
1403          177 format('metric.f,aa,bb',lp2e10.2,' nstep,ntoss ',2110)
1404          write(s100,79) aa,bb,nstep
1405          write(s100,80) n,imaxl,ent
1406          call gtext(s100,80,0)
1407          format('n,imax,ent,',2i6,1p3e14.4)
1408          80

```

```

1409 call frame(0)
1410 return
1411 20 continue
1412 cccc
1413 cubic
1414 30 return
1415 cccc chirikov
1416 40 continue
1417 40 continue
1418 cccc japan
1419 xmin = -3
1420 xmax = 3
1421 ymin = -5
1422 ymax = 3
1423 xf = 3
1424 yf = 1
1425 npts = 10000
1426 do 75 k = 1,npts
1427 call stepf(xf,yf)
1428 ww(k) = xf
1429 zz(k) = yf
1430 continue
1431 call maps(xmin,xmax,ymin,ymax,-1.0,-1.0,-1.0,-1.0)
1432 call points(ww,zz,npts,-1,-1,0,0,0,0)
1433 call setch(10, 5., 1., 0, 2, 0)
1434 write(s100,79) aa,bb,nstep,ntoss
1435 call gtext(s100,80,0)
1436 write(s100,80) n,imax1,ent
1437 call gtext(s100,80,0)
1438 call frame(0)
1439 return
1440 end
1441 subroutine ckpart
1442 implicit real (a-h,o-z)
1443 character*80 s100
1444 common /matr/ mv(999),iz(999),mw,lw,ns2
1445 common /vec/ xf0,yf0,xfl,xf2,y(99),x(99),n
1446 common /lc/ aa,bb,rr,mr,xi,xr,y1,yr,p1,p21,nmult
1447 common /plt/ ww(40000),xx(999),yy(999),zz(40000),ymax,nn
1448 common /atx/ x1(99999),x2(99999),y1(99999),y2(99999),imx1,imax2,iold1,iold2,ntoss
1449 common /d3/ ylo(10,10),yhi(10,10),nstep,ncount,model
1450 common /param/ xg(10),yg(10,10),best(10,10),kgrid,kcrit
1451 routine to check partition
1452 xmin = -rr
1453 xmax = rr
1454 ymin = -rr
1455 ymax = rr
1456 npts = 0
1457 do 10 ky = 1,6
1458 do 10 kx = 1,20
1459 xf = 1.+ .04*kx*rr
1460 yf = -.25 + .1*ky
1461 md = mvav(xf,yf)
1462 if (md.eq.0) go to 10
1463 npts = npts + 1
1464 ww(npts) = xf
1465 zz(npts) = yf
1466 continue
1467 call maps(xmin,xmax,ymin,ymax,-1.0,-1.0,-1.0,-1.0)
1468 call points(ww,zz,npts,-1,-1,0,0,0,0)
1469 kgp = kgrid + 2
1470 do 17 kc = 1,kcrit
1471 do 18 kg = 1,kgrid
1472 kp = kg + 1

1473 ww(kp) = xg(kg)
1474 zz(kp) = yg(kg,kc)
1475 18 continue
1476 ww(1) = -rr
1477 zz(1) = yg(1,kc)
1478 ww(kgp) = rr
1479 zz(kgp) = yg(kg,kc)
1480 call trace(ww,zz,kgp,-1,-1,0.0,0.0)
1481 17 continue
1482 call frame(0)
1483 return
1484 end
1485 subroutine opt
1486 implicit real (a-h,o-z)
1487 character*80 s100
1488 common /matr/ mv(999),iz(999),mw,lw,ns2
1489 common /vec/ xf0,yf0,xfl,xf2,y(99),x(99),n
1490 common /lc/ aa,bb,rr,mr,xi,xr,y1,yr,p1,p21,nmult
1491 common /plt/ ww(40000),xx(999),yy(999),zz(40000),ymax,nn
1492 common /atx/ x1(99999),x2(99999),y1(99999),y2(99999),imx1,imax2,iold1,iold2,ntoss
1493 common /d3/ ylo(10,10),yhi(10,10),nstep,ncount,model
1494 common /param/ xg(10),yg(10,10),best(10,10),kgrid,kcrit
1495 common /tric/ pri(99999),br1(900,100),ent,emin
1496 common /tric2/ pr2(99999),br2(900,900),n2(900,900)
1497 ccc find partition with maximum entropy for fixed n, kcrit, nstep
1498 ccc nstep assumed defined by entvst, otherwise it must be chosen
1499 sup = 0
1500 sup = 5 kg = 1,kgrid
1501 do 5 kg = 1,kgrid
1502 best(kg,kc) = .5*(ylo(kg,kc)+yhi(kg,kc))
1503 5 continue
1504 emin = 1.e6
1505 do 95 kt = 1,ntoss
1506 call randid(kt)
1507 call setup(xf,yf)
1508 call advanc(xf,yf)
1509 call metric
1510 write(6,35) yg(1,1),ent
1511 format('yg,ent,' ,1p2e12.4)
1512 if (ent.lt.emin) emin = ent
1513 if (ent.lt.sup) go to 95
1514 sup = ent
1515 do 51 kg = 1,kgrid
1516 do 50 kc = 1,kcrit
1517 best(kg,kc) = yg(kg,kc)
1518 50 continue
1519 51 continue
1520 95 continue
1521 do 61 kg = 1,kgrid
1522 do 60 kc = 1,kcrit
1523 yg(kg,kc) = best(kg,kc)
1524 60 continue
1525 61 continue
1526 write(6,79) aa,bb,nstep,ntoss
1527 format('metric.f opt aa,bb,1p2e10.2,' ,nstep,ntoss ,2,10)
1528 write(6,80) n,imax1,sup,emin
1529 format(' ,model, kcrit ,215)
1530 81 format('n,imax,ent,emin ,216,1p3e12.4)
1531 80 format(' ,model, kcrit ,215)
1532 call wrgrid
1533 call statist(1)
1534 return
1535 end
1536 subroutine optdat

```

```

1537 implicit real (a-h,o-z)
1538 character*80 s100
1539 common /mat/ mv(99), iz(999), mw, 1w, ns2
1540 common /vec/ xf0,yf0,xfl,xf2,y(99),x(99),n
1541 common /lc/ aa,bb,rr,rr,xl,xr,y1,yr,pi,p21,nmult
1542 common /plt/ ww(40000),xx(999),yy(999),zz(40000),ymax,nm
1543 common /stx/ lxi(9999),lx2(9999),imax1,imax2,iold1,iold2,ntoss
1544 common /d3/ ylo(10,10),yhi(10,10),ntsep,ncount,model
1545 common /param/ xg(10),yg(10,10),best(10,10),kgrid,kcrit
1546 common /tric2/ pr2(9999),br2(900,900),n2(900,900)
1547 common /tric2/ pr1(9999),br1(100,100),ent,emin
1548 ccc find partition with maximum entropy for fixed n, kcrit, ntsep
1549 ccc nstep assumed defined by entvat, otherwise it must be chosen
1550 sup = 0
1551 do 5 kg = 1,kgrid
1552 do 5 kc = 1,kcrit
1553      5 best(kg,kc) = .5*(ylo(kg,kc)+yhi(kg,kc))
1554      continue
1555 emin = 1.e6
1556      do 95 kt = 1,ntoss
1557      call rangrid(kt)
1558      call statlist(0)
1559      ccc write(6,35) yg(1,1),ent
1560      format(' yg ent ',1p2e12.4)
1561      if(ent.lt.emin) emin = ent
1562      if(ent.lt.sup) go to 95
1563      aup = ent
1564      do 51 kg = 1,kgrid
1565      best(kg,kc) = yg(kg,kc)
1566      continue
1567      50 continue
1568      51 continue
1569      95 continue
1570      do 61 kg = 1,kgrid
1571      do 60 kc = 1,kcrit
1572      yg(kg,kc) = best(kg,kc)
1573      60 continue
1574      61 continue
1575      79 write(6,79) aa,bb,ntsep,ntoss
1576      write(6,80) n,imax1,sup
1577      write(6,81) model,kcrit
1578      81 format(' model, kcrit ',2i5)
1579      80 format(' n,imax,ent ',2i5,1p3e12.4)
1580      call wrgrid
1581      call wrgrid
1582      call statlist(1)
1583      return
1584 end
1585 subroutine contour
1586 implicit real (a-h,o-z)
1587 character*80 s100
1588 common /mat/ mv(999),iz(999),mw,1w,ns2
1589 common /vec/ xf0,yf0,xfl,xf2,y(99),x(99),n
1590 common /lc/ aa,bb,rr,rr,xl,xr,y1,yr,pi,p21,nmult
1591 common /plt/ ww(40000),xx(999),yy(999),zz(40000),ymax,nm
1592 common /stx/ lxi(9999),lx2(9999),imax1,imax2,iold1,iold2,ntoss
1593 common /d3/ ylo(10,10),yhi(10,10),ntsep,ncount,model
1594 common /param/ xg(10),yg(10,10),best(10,10),kgrid,kcrit
1595 common /tric2/ pr1(9999),br1(100,100),ent,emin
1596 common /tric2/ pr2(9999),br2(900,900),n2(900,900)
1597 ccc find contour of entropy with two partition points
1598 kgrid = 1
1599 kcrit = 2
1600 nstep = 2000

```

---

```

1601      sup = 0
1602      smin = 1.e6
1603      do 100 k1 = 1,200
1604      do 100 k2 = k1,200
1605      yg(1,1) = k1*.0314
1606      yg(1,2) = k2*.0314
1607      call setup(xf,yf)
1608      call advanc(xf,yf)
1609      call metric
1610      br2(k1,k2) = ent
1611      if(ent.lt.smin) smin = ent
1612      if(ent.lt.sup) go to 100
1613      sup = ent
1614      ib = imax1
1615      xb = yg(1,1)
1616      yb = yg(1,2)
1617      100 continue
1618      xmin = 0
1619      xmax = 6.28
1620      ymin = 0
1621      ymax = 6.28
1622      call mape(xmin,xmax,ymin,ymax,-1.0,-1.0,-1.0)
1623      npts = 0
1624      do 200 k1 = 1,200
1625      do 200 k2 = k1,200
1626      ndum = 5*br2(k1,k2)/(sup - smin)
1627      if(ndum.eq.nsav) go to 200
1628      npts = npts + 1
1629      ww(npts) = k1*.0314
1630      zz(npts) = (k2-.5)*.0314
1631      nsav = ndum
1632      if(k1.eq.k2) npts = npts - 1
1633      200 continue
1634      call points(ww,zz,npts,-1,-1,0.0,0.0)
1635      call setch(20,.30,.1,.0,.2,.0)
1636      write(s100,79) aa,bb,ntsep
1637      79 write(s100,79) aa,bb,ntsep
1638      format(' metric.f aa,bb ',1p2e10.2,' ntsep ',18)
1639      write(s100,80) model,n,ib,sup
1640      call gtext(s100,80,0)
1641      80 format('model,n,ntsep,ent ',3i6,1p3e12.4)
1642      write(s100,81) xb,yb
1643      call gtext(s100,80,0)
1644      81 format(' x,y ',1p2e12.4)
1645      return
1646 end
1647 subroutine newstat(lidum)
1648      1 implicit real (a-h,o-z)
1649      1 common /mat/ mv(99),iz(999),mw,1w,ns2
1650      1 common /vec/ xf0,yf0,xfl,xf2,y(99),x(99),n
1651      1 common /lc/ aa,bb,rr,rr,xl,xr,y1,yr,pi,p21,nmult
1652      1 common /plt/ ww(40000),xx(999),yy(999),zz(40000),ymax,nm
1653      1 common /stx/ lxi(9999),lx2(9999),imax1,imax2,iold1,iold2,ntoss
1654      1 common /d3/ ylo(10,10),yhi(10,10),ntsep,ncount,model
1655      1 lidum is new
1656      1 max1 = max1 + 1
1657      1 f(max1,1t,99999) go to 134
1658      write(6,133)
1659      133 format(' increase dimension of lx' )
1660      stop
1661      134 continue
1662      134 x1(imax1) = lidum
1663      134 common /vec/ mv(6,55) imax1, lidum, (mv(k1),k1 = 1,n)
1664      134 format('il=,i3, ' L=,i8, mv(k) ',30i2)

```

```

1665      return
1666      end
1667      subroutine newrite(lidum)
1668      implicit real (a-h,o-z)
1669      common /mat/ mv(99),iz(999),mw,1w,nz2
1670      common /vec/ xf0,yf0,xfl,xf2,y(99),x(99),n
1671      common /1c/ aa,bb,rr,rr,rr,xl,xr,yl,yr,p1,p21,nmult
1672      common /plt/ mw(40000),xx(999),yy(999),zz(999),mw,1w,nz
1673      common /stx/ lx(99999),lx2(99999),mx1,imx1,iold1,iold2,ntoss
1674      common /d3/ ylo(10,10),yhi(10,10),nstep,ncount,model
1675      cccc
1676      lidum is new
1677      imax1 = imax1 + 1
1678      if(imax1.lt.99999) go to 134
1679      write(6,133)
1680      format(' increase dimension of lx')
1681      stop
1682      continue
1683      imx1(imax1) = lidum
1684      write(6,55) imax1,lidum,(mv(k1),k1 = 1,n)
1685      format('il=,13,' ,18,' ,mv(k)',3012)
1686      return
1687
1688      subroutine newstat2(lidum)
1689      implicit real (a-h,o-z)
1690      common /mat/ mv(99),iz(999),mw,1w,nz2
1691      common /vec/ xf0,yf0,xfl,xf2,y(99),x(99),n
1692      common /1c/ aa,bb,rr,rr,xl,xr,yl,yr,p1,p21,nmult
1693      common /plt/ mw(40000),xx(999),yy(999),zz(999),mw,1w,nz
1694      common /stx/ lx(99999),lx2(99999),imax1,imx2,iold1,iold2,ntoss
1695      lidum is new
1696      imax2 = imaxx2 + 1
1697      if(imax2.lt.99999) go to 134
1698      write(6,133)
1699      format(' increase dimension of lx')
1700      stop
1701      continue
1702      imx2(imax2) = lidum
1703      ccc
1704      write(6,55) imax2,xf0,yf0,lidum,(mv(k1),k1 = 1,n)
1705      format('il=,13,' ,18,' ,mv(k)',3012)
1706      return
1707
1708      subroutine expnd
1709      implicit real (a-h,o-z)
1710      common /mat/ mv(99),iz(999),mw,1w,nz2
1711      common /vec/ xf0,yf0,xfl,xf2,y(99),x(99),n
1712      integer is value,word
1713      cccc
1714      expand mw into mv(k)
1715      do 10 k = 1,n
1716      kp = n - k + 1
1717      mv(kp) = and(l.shiftfr(mw,k-1))
1718      continue
1719      return
1720
1721      subroutine comprs
1722      implicit real (a-h,o-z)
1723      common /param/ xq(10),yg(10,10),best(10,10),kgrid,kcrit
1724      common /mat/ mv(99),iz(999),mw,1w,nz2
1725      common /vec/ xf0,yf0,xfl,xf2,y(99),x(99),n
1726      mw = 0
1727      nd = 0
1728      nmod = kcrit+1
1729      do 10 k = 1,n
1730
1731      ccc
1732      ccc
1733      mw = mw + nd*iz(k)
1734      10   continue
1735
1736      return
1737
1738      subroutine setiz
1739      implicit real (a-h,o-z)
1740      common /mat/ mv(99),iz(999),mw,1w,nz2
1741      common /param/ xg(10),yg(10,10),best(10,10),kgrid,kcrit
1742      common /d3/ ylo(10,10),yhi(10,10),nstep,ncount,model
1743      lidum = kcrit + 1
1744      if(model.eq.3) lidum = 7
1745      do 10 k = 1,n
1746      iz(k) = lidum**(n-k)
1747      continue
1748      10
1749
1750      subroutine show(xa,ya,ns,nz)
1751      implicit real (a-h,o-z)
1752      character*80 s100
1753      dimension xa(999),ya(999)
1754      cccc
1755      kmnl = 0
1756      xmax = xa(1)
1757      ymin = 1.ee+nz
1758      ymax = -1.ee
1759      do 10 k = 1,ns
1760      if(ya(k).lt.ymin) ymin = ya(k)
1761      if(ya(k).gt.ymax) ymax = ya(k)
1762      10
1763      continue
1764      ymax = ymax + 1.*(ymax-ymin)
1765      call maps(kmnl,xmax,ymin,ymax,-1.0,-1.0,-1.0,-1.0)
1766      call trace(xa,ya,ns,-1,-1,0.0,0.0)
1767      call sech(10.,35.,1.,0,2,0)
1768      write(6,100,35)
1769      call gtext(8100,80,0)
1770      format(' metric.f routine show ',i20)
1771      return
1772
1773      subroutine showa(xa,ya,ns,nz)
1774      implicit real (a-h,o-z)
1775      dimension xa(40000),ya(40000)
1776      1777      cccc
1777      general plotting routine
1778      xmin = xa(1)
1779      ymin = xa(ns)
1780      ymax = 1.ee+nz
1781      do 10 k = 1,ns
1782      if(ya(k).lt.ymin) ymin = ya(k)
1783      if(ya(k).gt.ymax) ymax = ya(k)
1784      if(xa(k).lt.xmin) xmin = xa(k)
1785      if(xa(k).gt.xmax) xmax = xa(k)
1786      10
1787      continue
1788      ymax = ymax + 2.*(ymax-ymin)
1789      call maps(xmin,xmax,ymin,ymax,-1.0,-1.0,-1.0,-1.0)
1790      call trace(xa,ya,ns,-1,-1,0.0,0.0)
1791
1792      subroutine stepdf(xf,yf)

```

```

1793 implicit real (a-h,o-z)
1794 character*80 s100
1795 common /1c/ aa,bb,rr,rr,rr,xl,xr,y1,yf,pi,p2i,nmult
1796 dimension a(4),e(4),bx(4),y(4),d(4)
1797 n1 = 2
1798 tt = 0
1799 ndiv = 200
1800 dt = 2*pi(ndiv
1801 do 99 kdiv = 1,ndiv
1802 tt = 2*pi*(kdiv-1)/ndiv
1803 ccc write(6,110) aa,bb,xf,yf,tt
1804 110 format(' aa,bb, x,y,t ',1p5e12.4)
1805 cc xf dot
1806 e(1) = yf
1807 cc yf dot
1808 e(2) = rr + bb*sin(tt) - xf**3 - aa*yf
1809 cccc half step
1810 xh = xf + e(1)*.5*dt
1811 yh = yf + e(2)*.5*dt
1812 cc xf dot
1813 e(1) = yh
1814 cc yf dot
1815 tt = 2*pi*(kdiv-.5)/ndiv
1816 e(2) = rr + bb*sin(tt) - xh**3 - aa*yh
1817 cccc full step
1818 xf = xf + e(1)*dt
1819 yf = yf + e(2)*dt
1820 if(xf.gt.100.) stop
1821 continue
1822 return
1823 end
1824 subroutine stdif(xf,yf)
1825 implicit real (a-h,o-z)
1826 character*80 s100
1827 common /1c/ aa,bb,rr,rr,xl,xr,y1,yf,pi,p2i,nmult
1828 c provides 1 step of the kutta merson process for a system
1829 c of n first ord ordinary differential equations,
1830 c dy/dt=e(t,y). vectorized 1981, r. b. white
1831 c
1832 c y(1),...,y(n) should contain on entry the n solutions of the
1833 c system at t, and on return will contain the computed values
1834 c at t + dt.
1835 c before each updating of y the comment gives the appropriate
1836 c mathematical formula.
1837 dimension a(4),e(4),bx(4),y(4),d(4)
1838 n1 = 2
1839 tt = 0
1840 ndiv = 100
1841 dt = 2*pi(ndiv
1842 do 99 kdiv = 1,ndiv
1843 tt = 2*pi*(kdiv-1)/ndiv
1844 y(1) = xf
1845 y(2) = yf
1846 write(6,110) xf,yf,tt
1847 110 format(' x,y,t ',1p3e12.4)
1848 cc old y is saved in d
1849 d(1) = y(1)
1850 d(2) = y(2)
1851 h=dt/3.0
1852 200 continue
1853 c do 41 j=1,5
1854 cc xf dot
1855 cc do 41 j=1,5
1856 cc xf dot
1857 e(1) = y(2)
1858 cc yf dot
1859 e(2) = -.045 + bb*sin(tt) - y(1)**3 - aa*y(2)
1860 321 continue
1861 62 continue
1862 npos = 55
1863 go to(11,12,13,14,15),j
1864 c
1865 c
1866 11 do 21 i=1,n1
1867 a(i)=h*e(i)
1868 21 y(1)=d(i)+a(i)
1869 npos = 56
1870 go to 40
1871 c
1872 c y2=y0+1/6*dt*f(t0,y0)+1/6*dt*f(t0+1/3*dt,y1)
1873 12 do 22 i=1,n1
1874 22 y(1)=d(i)+(a(i)+h*e(i))*0.5
1875 npos = 57
1876 go to 40
1877 c
1878 c y3=y0+1/8*dt*f(t0,y0)+3/8*dt*f(t0+1/3*dt,y1)
1879 13 do 23 i=1,n1
1880 bx(1)=h*e(1)
1881 23 y(1)=d(i)+(a(i)+bx(1))*0.375
1882 npos = 58
1883 go to 40
1884 c
1885 c
1886 c y4=y0+1/2*dt*f(t0,y0)-3/2*dt*f(t0+1/3*dt,y2)+2*dt*e(t0+1/2*dt,y3)
1887 14 do 24 i=1,n1
1888 cl=h*e(i)
1889 1889 y(1)=d(i)+(a(i)-bx(i))*3.0+c1*4.0)*1.5
1890 bx(1)=cl
1891 npos = 59
1892 go to 40
1893 c
1894 c
1895 c y5=y0+1/6*dt*f(t0,y0)+2/3*dt*f(t0+1/2*dt,y3)+1/3*dt*e(t0+dt,y4)
1896 15 do 25 i=1,n1
1897 25 y(1)=d(i)+(a(i)+bx(i)*4.0+h*e(i))
1898 40 continue
1899 41 continue
1900 1900
1901 xf = y(1)
1902 yf = y(2)
1903 99 continue
1904 return
1905 c =====
1906 end
1907 subroutine power
1908 character*80 s100
1909 common /1c/ aa,bb,rr,rr,rr,xl,xr,y1,yf,pi,p2i,nmult
1910 common /vec/ xl0,yf0,xfl,xft,yf99,x,(99),n
1911 common /d3/ yf0(10,10),yf1(10,10),ratep,ncount,model
1912 common /mat/ mv(99),iz(999),mw,w,ns2
1913 dimension xx(64),om(64)
1914 dimension av(20),sd(20),u(64),z(64)
1915 dimension xj(64),x0(520)
1916 dimension r(20),s(20),t(20)
1917 complex a(64),ex(64,128),ei
1918 pi2 = 2.*pi
1919 ei = cmplx(0.,1.)
1920 cccocadvance xf,yf in time ncy times

```

```

1921 call setup(xf,yf)
1922 ncy = 100
1923 nt = 100
1924 ntp = nt + 1
1925 rnt = nt
1926 nmid = ntp/2
1927 frequencies, initialize
1928 do 10 j = 1,nmid
1929 om(j) = pi2*(j-1)/rnt
1930 u(j) = 0.
1931 do 10 ns = 1,nt
1932 ex(j,ns) = exp(-ei*om(j)*(ns-1))
1933 continue
1934 choose initial point if desired
1935 xf = -1.71
1936 yf = 3
1937 xsav = xf
1938 ysav = yf
1939 do 200 k = 1,ncy
1940 np = 1
1941 do 201 j = 1,nmid
1942 xj(1) = xf
1943 a(j) = cmplx(0.,0.)
1944 continue
1945 do 20 j = 1,nt
1946 np = j + 1
1947 call stepf(xf,yf)
1948 test xj(np) = xf
1949 continue
1950 do 40 j = 2,np
1951 now fourier trasform
1952 do 40 j = 2,np
1953 a(j) = a(j) + xj(ns)*ex(j,ns)
1954 continue
1955 ccccc construct the sum of a**2
1956 do 33 j = 1,nmid
1957 u(j) = u(j) + cab(a(j))**2
1958 continue
1959 do 200 continue
1960 time step finished
1961 ccccc store data
1962 125 continue
1963 cccc plot data
1964 ccccc normalize
1965 sum = 0
1966 do 32 j = 1,nmid
1967 sum = sum + u(j)
1968 continue
1969 do 34 j = 1,nmid
1970 z(j) = u(j)/sum
1971 continue
1972 call showa(km,z,nmid,0)
1973 call bsetch10.,150.,1.,0,2,0)
1974 write(s100,1620),nt,ncy
1975 1620 format(' av, pow spec t=,,13,' ncyl=,,18)
1976 call gtext(s100,80,0)
1977 write(s100,15) aa,bb
1978 15 format(' metric,f routine power a,b , ,1p2e12.4)
1979 call gtext(s100,80,0)
1980 write(s100,77) model,n,rr,rr
1981 format(' model, parameters n, rr, mr ,2i4,1p2e12.4,14)
1982 call gtext(s100,80,0)
1983 write(s100,78) xsav,ysav
1984 78 format(' initial x,y ',1p2e12.4)

1985 100 call gtext(s100,80,0)
1986 100 continue
1987 call frame(0)
1988 end
1989 subroutine powsym
1990 character*80 s100
1991 common /ic/ aa,bb,rr,rr,xl,xr,yl,yr,pi,p2i,rnult
1992 common /vec/ xf0,yf0,xf1,yf1,xf2,yf2,x(99),x(99),n
1993 common /d3/ ylo(10,10),yhi(10,10),nstep,ncount,model
1994 common /mat/ mv(99),iz(999),mw,1w,ns2
1995 dimension xx(64),om(64)
1996 dimension av(20),sgf(20),u(64),z(64)
1997 dimension xj(64),x0(5120)
1998 dimension r(20),s(20),t(20)
1999 complex a(64),ex(64,128),ei
2000 pi2 = 2.*pi
2001 ei = cmplx(0.,1.)
2002
2003 cccccc advance xf,yf in time ncy times
2004 call setup(xf,yf)
2005 rcy = 1000
2006 nt = 100
2007 rnt = nt + 1
2008 nmid = ntp/2
2009
2010 cc frequencies, initialize
2011 do 10 j = 1,nmid
2012 om(j) = pi2*(j-1)/rnt
2013 u(j) = 0.
2014 do 10 ns = 1,nt
2015 ex(j,ns) = exp(-ei*om(j)*(ns-1))
2016 continue
2017 dnorm = 2.*n
2018 xsav = xf
2019 ysav = yf
2020 do 200 k = 1,ncy
2021 mav = mw
2022 do 201 j = 1,nmid
2023 xj(1) = mw/dnorm
2024 a(j) = cmplx(0.,0.)
2025 continue
2026 do 20 j = 1,nt
2027 np = j+1
2028 call stepf(xf,yf)
2029 do 25 ks = 1,n
2030 kp = ks + 1
2031 mv(kp) = mw/kp
2032 25 continue
2033 mv(n) = mvav(xf,yf)
2034 call compa
2035 2000
2036 20 continue
2037 cccc now fourier transform
2038 do 40 ns = 1,nt
2039 do 40 j = 2,nmid
2040 a(j) = a(j) + xj(ns)*ex(j,ns)
2041 40 continue
2042 ccccc construct the sum of a**2
2043 do 33 j = 1,nmid
2044 u(j) = u(j) + cab(a(j))**2
2045 33 continue
2046 200 continue
2047 ccccc store data
2048

```

```

2049    125  continue
2050      cccc
2051      normalize
2052      sum = 0
2053      do 32 j = 1,nmid
2054          sum = sum + u(j)
2055      32  continue
2056      call showa(om,z,nmid,0)
2057      do 34 j = 1,nmid
2058      34  continue
2059      call gtext(a(100,15),aa,bb)
2060      write(8100,1620) nt,ncy
2061      1620  format(' av pow spec t=',i3,' ncy=',i8)
2062      call gtext(a(100,80,0))
2063      write(8100,80,0)
2064      format(' metric.f routine powsym a,b , ,1p2e12.4)
2065      call gtext(a(100,80,0))
2066      write(8100,77) model,n,rr,rr
2067      77  format('model, Parameters n, rr, rr, mr ',2i4,1pe12.4,14)
2068      call gtext(a(100,80,0))
2069      write(8100,78) xsav,ysav
2070      78  format(' initial x,y ',1p2e12.4)
2071      call gtext(a(100,80,0))
2072      100  continue
2073      call frame(0)
2074      return
2075
2076  end
2077
2078  subroutine powold
2079      character*80 s100
2080      common /lc/ aa,bb,rr,mr,x1,xr,y1,yr,p1,p21,nmult
2081      common /vec/ xfo,yfo,xfi,yfi,(99),x,(99),n
2082      common /d3/ ylo(10,10),yhi(10,10),nsep,ncount,model
2083      dimension xx(64),om(64)
2084      dimension av(20),sg(20),u(64),z(64)
2085      dimension xj(64),x0(520)
2086      dimension r(20),s(20),t(20)
2087      dimension a(64)
2088      pi2 = 2.*pi
2089      ei = cmplx(0.,1.)
2090      cccccccadvance x1,yf in time  ncy times
2091      call setup(xf,yf)
2092      ncy = 10000
2093      nt = 100
2094      ntp = nt + 1
2095      rnt = nt
2096      nmid = ntp/2
2097      cc  initialize
2098      do 10 j = 1,nmid
2099          om(j) = pi2*(j-1)/rnt
2100          u(j) = 0.
2101          a(j) = 0
2102          10  continue
2103          choose initial point if desired
2104          xf = -1.71
2105          yf = 3
2106          xsav = xf
2107          ysav = yf
2108          do 200 k = 1,ncy
2109              msav = mw
2110              do 20 j = 1,nt
2111                  call stepf(xf,yf)
2112                  do 25 ks = 1,n

```