```
%    The Directional Mobility Index, computed by "DMMtool.m", has been designed
%    by C. Ferreti and P. Ganugi; it is thoroughly explored in the paper:
%       "A new mobility index for transition matrices", C.Ferretti and P.Ganugi,
%        Statistical Methods & Applications 22 (3), 403-425.
%    "DMMtool.m" is freely downloadable from the Mathworks's page
%    dedicated to file exchange:
%    www.mathworks.com/matlabcentral/fileexchange/
%
clear License response ans
License=char({'','    Copyright © Danilo Aringhieri, Camilla Ferretti, 2017. ',...
'    This software is released under the license ','    Creative Commons Unported, ',...
'    Attribution – Non Commercial – Share Alike 4.0 ','    (CC  BY – NC – SA  4.0) ',...
'    whose readable summary is displayed at: ',...
'             http://creativecommons.org/licenses/by-nc-sa/4.0/ ',...
'    and whose legal text is available at the following webpage: ',...
'             http://creativecommons.org/licenses/by-nc-sa/4.0/legalcode ',''});
disp(License)
disp('    Press the "Enter" button on the keyboard to START...')
response=input(' ','s');
while isempty(response)==0
disp('    Please, press just the "Enter" button to START!')
response=input(' ','s');
end
Intro1=char({'','',' This is a MATLAB command line interface (R2006a at least) ',...
' for the estimation of the Directional Mobility Index based on raw data  ',...
' for a sample of statistical units respect to a quantitative variable, ',...
' like any size-specific variable (as income in case of families; ',...
' revenues or the number of employees, in case of firms, etc.). ',...
' The not-normalized Directional Mobility Index is defined by: ','',...
' I_w_v(P) = Sum(i=1..n) w_i * [Sum(j=1..n) P_i_j * sign(j-i) * v(|j-i|)] .',...
'',' If we name its minimum as  m1 = Sum(i=1..n) w_i * v(1-i) ',...
' and its maximum as  m2 = Sum(i=1..n) w_i * v(n-i), ',...
' then the normalized version, respect to [-1,+1], of the Index is: ','',...
'         Inorm_w_v(P) =  ',...
'     = { [-I_w_v(P)/m1 if I_w_v(P)<0] and [I_w_v(P)/m2 if I_w_v(P)>=0] } ',...
'',' The summations are computed over a discrete set  {1,2,..,n} ',...
' of n states accessible to every individual of the sample  ',...
' which evolves randomly from the initial state "i" to the final state "j" ',...
' with conditional probability P_i_j , according to the transition matrix "P". ',...
' Each state may be a class grouping several different values of the variable. ',...
'             ( press the "Enter" button to continue about this Index ) '});
disp(Intro1)
response=input(' ','s');
while isempty(response)==0
disp('(please, press [Enter] !)')
response=input(' ','s');
end
Intro2=char({'',' Parameters {w_i} are to be chosen properly to weight ',...
' the contribution of the starting state "i" to the mobility index.',...
' Similarly, the role of the function  v(|j-i|)  is to measure, ',...
' for the mobility, the importance of the phase-jump intensity |j-i| ',...
' while the factor  sign(j-i) accounts for the jump orientation. ',...
'',    This tool does not work  ',...
'    if data consist of qualitative categories ',...
'    of a non-quantitative statistical variable !! ',...
'    It is able to process any panel data for a single variable ',...
'    known at least in two different periods of time (decades, years, months..., ',...
'    according to the frequency of observations). ',...
'',' During the execution of this tool,  ',...
' YOU WILL BE ASKED TO INSERT YOUR FAVOURITE VALUES OR SHAPES  ',...
' for all the parameters and functions appearing in the Index expression, '...
'',        ARE YOU INTERESTED in USING IT ? ',...
' (Possible answers: "Y" for ''Yes'' or "N" for ''Not'', ',...
'  then press the "Enter" button) ',...
```

```
'      Beware that, if you answer "Yes", all the objects ',...
'      from the previous run of this tool,',...
'      WILL BE CANCELLED !! '});
disp(Intro2)
response=input('  ','s');
while strcmp(response,{'Y','y','N','n'})==[0,0,0,0]
            disp('   (You must answer exclusively "Y" or "N" !) ')
        response=input('  ','s');
end
if sum(strcmp(response,{'N','n'}))==1
disp(char({' ',' OK: you are going to return directly ',...
' to the prompt state in the Command Window! ',' '}))
clear License Intro1 Intro2 response
return
elseif sum(strcmp(response,{'Y','y'}))==1
disp(char({' ',' ',' BE CAREFUL !!! '}))
end
Intro3a=char({'',' The following PRESCRIPTIONS MUST BE SATISFIED. ',...
'',' I°] ',...
' A spreadsheet, able to read and save xls-type or xlsx-type files, ',...
' has to be installed on your computer (Windows Excel would be better ',...
' but a free opensource saving into xlsx format goes quite well anyway, ',...
' like Calc from LibreOffice). ',...
' Here, for every statistical unit, the variable values, ',...
' to be imported into MATLAB, must be tabled by columns and rows. ',...
'',' II°] ',...
' Verify that the Current Folder, on the top of the MATLAB Home GUI,',...
' is the one containing both your Excel data file and the script of this interface;  ',...
' otherwise leave this tool, put the data and the script into the same folder ',...
' then run again. ',...
'         (press [Enter] to go on...) '});
disp(Intro3a)
response=input(' ','s');
while isempty(response)==0
disp('(please, press [Enter]...)')
response=input(' ','s');
end
Intro3b=char({' III°] ',...
' From some point onwards in your datasheet, to each row should correspond ',...
' the sequence of numerical values of a particular individual ',...
' relative to the variable for all the dates of survey; ',...
' in every column, all units'' values must be listed referring to a specific period. ',...
' Actually, opening columns on the left may be filled with some lists of strings ',...
' (firms'' names in alphabetic characters, alphanumeric identity codes or addresses ) ',...
' and the opening rows on the top may be filled with the dates of observations ',...
' or any other details written in string characters. ','',...
'                        DO YOU WANT TO CONTINUE ? ',...
' (Answer "Y" for ''Yes'' or "N" for ''Not'', then press [Enter]) '});
disp(Intro3b)
response=input('  ','s');
while strcmp(response,{'Y','y','N','n'})==[0,0,0,0]
            disp('   (You must answer exclusively "Y" or "N" !) ')
        response=input('  ','s');
end
if sum(strcmp(response,{'N','n'}))==1
disp(char({' ',' OK: you are going to return directly ',...
' to the prompt state in the Command Window! ',' '}))
clear Intro1 Intro2 Intro3a Intro3b License response
return
elseif sum(strcmp(response,{'Y','y'}))==1
clear Intro1 Intro2 Intro3a Intro3b
end
clear k h filename AuxData1 Titles dimensions notAvail unavail_id Sz AuxData2 MatData resp dim Min_Sizes
clear Max_Sizes sup units lowest_datum highest_datum classes_number num_k_Lstart_Lend w_k_Lstart_Lend
```

```
clear v V_k_h Number_k_h_Lstart_Lend P_k_h_Lstart_Lend P_k_h_Tstart_Tend num_tot_Lstart_Lend
clear Pup_k_h_Lstart_Lend Pup_k_h_Tstart_Tend Pdn_k_h_Lstart_Lend Pdn_k_h_Tstart_Tend w_k sum_w_k
clear Tstart Tend m1 m2 Lstart Lend Y_V_k_Lstart_Lend Y_V_k_Tstart_Tend ImmobilShares_k_Tstart_Tend
clear Yup_V_k_Lstart_Lend Yup_V_k_Tstart_Tend Ydn_V_k_Lstart_Lend Ydn_V_k_Tstart_Tend ImmobilShares_k_Lstart_Lend
clear I_w_V_Lstart_Lend I_w_V_Tstart_Tend Iup_w_V_Lstart_Lend Iup_w_V_Tstart_Tend Idn_w_V_Lstart_Lend
Idn_w_V_Tstart_Tend
clear Inorm_w_V_Lstart_Lend Inorm_w_V_Tstart_Tend Inorm_up_w_V_Lstart_Lend Inorm_up_w_V_Tstart_Tend
clear Inorm_dn_w_V_Lstart_Lend Inorm_dn_w_V_Tstart_Tend  resp_w last_ans V_infinite V_real V_nan
clear UPvsDN_w_V_Tstart_Tend UPvsDN_w_V_Lstart_Lend
clear Itrace_Tstart_Tend Idet_Tstart_Tend I2nd_eigval_Tstart_Tend Ieigvals_Tstart_Tend Ipredict_Tstart_Tend
clear abs_eigenvals_Tstart_Tend sorted_abs_eigenvals_Tstart_Tend eigenvects_Tstart_Tend eigenvals_Tstart_Tend
clear Intensity_k_h I_2ndBarthol_Tstart_Tend BMorrDN_Tstart_Tend BMorrUP_Tstart_Tend w_k_Tstart_Tend
Input0=char({'','',' [Consider that you CAN STOP AT ANY TIME the run of this program ',...
'                    TYPING "Ctrl" + "C"  ON THE KEYBOARD !!!] ',''});
Input0bis=char({'',' (You CAN STOP AT ANY TIME the program ',...
'  TYPING "Ctrl" + "C"  ON THE KEYBOARD !!!) '});
disp(Input0)
Input1=char({'','  In order to import automatically your data, ',...
'       WRITE JUST THE NAME OF YOUR FILE, ',...
'       including its extension (".xls" or ".xlsx"), ',...
'       then press [Enter]. ',...
'    (any other command will provide an error message as a result; ',...
'     if it should appear a warning message, please wait few seconds. '});
disp(Input1)
filename=input('    ','s');
[AuxData1,Titles]=xlsread(filename);
dimensions=size(AuxData1);
for k=1:dimensions(2)
    notAvail(k)=(sum(isnan(AuxData1(:,k))))./dimensions(1);
end
unavail_id=find(notAvail-1);
Sz=size(unavail_id);
AuxData2=AuxData1(:,[unavail_id]);
Names1=char({'',' This tool tolerates the existence in the original datasheet of some columns ',...
    ' filled with numbers which ARE NOT NUMERICAL DATA, ',...
    ' like when the statistical units are indentified (also) by numbers. ',...
    '       ALL these kind of numerical non-data columns ',...
    '       HAVE TO STAY AT THE LEFT of the variable values !! ',...
    ' Do some numerical non-data columns exist in the original datasheet? ',...
    '     ( Answer "Y" if they exist or "N" otherwise, then press [Enter] ) '});
disp(Names1)
response=input('    ','s');
while strcmp(response,{'Y','y','N','n'})==[0,0,0,0]
            disp('    (You must answer exclusively "Y" or "N" !) ')
        response=input(' ','s');
end
if sum(strcmp(response,{'N','n'}))==1
MatData=AuxData2;
elseif sum(strcmp(response,{'Y','y'}))==1
Names2=char({' How many are the numerical non-data columns in the original sheet ? '});
 disp(Names2)
nondata_col=input('    ');
 if (isscalar(nondata_col)==0 | isreal(nondata_col)==0 | isnan(nondata_col)==1 | isinf(nondata_col)==1 | nondata_col<0 |
floor(nondata_col)~=nondata_col)
   while (isscalar(nondata_col)==0 | isreal(nondata_col)==0 | isnan(nondata_col)==1 | isinf(nondata_col)==1 | nondata_col<0
| floor(nondata_col)~=nondata_col)
      if isscalar(nondata_col)==0
      disp(' ( You pressed the [ENTER] button or inserted a vector or an array ! ) ')
      disp('   BUT  you must insert exclusively a positive integer ! ) ')
      nondata_col=input('    ');
      elseif (isscalar(nondata_col)==1 & (isnan(nondata_col)==1 | isinf(nondata_col)==1 | isreal(nondata_col)==0))
      disp(' Your input is a NaN value or infinite or a complex number: WHY? ')
      nondata_col=input('    ');
      elseif nondata_col<0
```

```
190         disp(' ( You must insert exclusively a positive number ! ) ')
191         nondata_col=input('    ');
192       elseif (nondata_col>0 & floor(nondata_col)~=nondata_col)
193         disp(' ( You must insert exclusively a positive integer ! ) ')
194         nondata_col=input('    ');
195       end
196      end
197   elseif (isscalar(nondata_col)==1 & isreal(nondata_col)==1 & isnan(nondata_col)==0 & isinf(nondata_col)==0 &↙
nondata_col>0 & floor(nondata_col)==nondata_col)
198 end
199 MatData=AuxData2(:,nondata_col+1:Sz(2));
200 end
201 clear Names1 Names2 AuxData1 AuxData2 notAvail unavail_id Sz nondata_col
202 dim=size(MatData);
203 Max_Sizes=max(MatData);
204 Min_Sizes=min(MatData);
205 disp(char({' ','  What is the dimensional unit of your data? ',...
206 '  (Use the expression you like and press [Enter]) '}))
207 units=input('  ','s');
208 Input2_1=char({'',' Observation dates will be implicitly marked ',...
209 ' by progressive positive integers ',...
210 ' according to the position of each detection in the sequence. ',...
211 ' These integers will be assigned to two auxiliary variables: ',...
212 ' "Lstart" (when a transition starts) and  "Lend"  (when a transition ends). ',...
213 '',' All the numerical data from your file ',...
214 ' will be recorded into the matrix   "MatData"   in the workspace. ',...
215 ' The strings will be copied in an array named "Titles" ',...
216 ' (it will be cancelled soon after). ',...
217 ' The ''NaN'' elements inside "MatData" correspond, inside the spreadsheet, ',...
218 ' to empty cells or to cells filled with string characters. ',...
219 '         (press [Enter] to go on...) '});
220 disp(Input2_1)
221 response=input(' ','s');
222 while isempty(response)==0
223 disp('(press [Enter] to go on...)')
224 response=input(' ','s');
225 end
226 Input2_2=char({'',' Three vectors will be created, ',...
227 ' named  "dim", "Max_Sizes" and  "Min_Sizes": ',...
228 '',' "dim" consists of the dimensions of MatData:',...
229 '  dim(1) is the number of statistical units enumerated inside the datasheet. ',...
230 '  dim(2) is the number of periods when the variable''s values have been measured. ',...
231 '','  " Max_Sizes "  and  " Min_Sizes " are row vectors: ',...
232 '  " Max_Sizes(m) "  and  " Min_Sizes(m) "  are respectively ',...
233 '  the biggest datum and the smallest one registered in the m-th column of "MatData"  ',...
234 '  referring to the m-th observation period. ',...
235 '         (press [Enter] to see) ','',});
236 disp(Input2_2)
237 clear resp
238 response=input(' ','s');
239 while isempty(response)==0
240 disp('(press [Enter] to see)')
241 response=input(' ','s');
242 end
243 disp('  According to your sample, the vector  " Min_Sizes "  is: ')
244 if dim(2)==2
245 fprintf('     [%-1.3f, ',Min_Sizes(1));...
246 fprintf('%-1.3f] \n',Min_Sizes(2))
247 else
248 fprintf('     [%-1.3f, ',Min_Sizes(1));...
249     fprintf('%-1.3f, ',Min_Sizes(2:dim(2)-1));...
250     fprintf('%-1.3f] \n',Min_Sizes(dim(2)))
251 end
252 disp('  and the vector  " Max_Sizes "  is: ')
```

```
if dim(2)==2
fprintf('     [%-1.3f, ',Max_Sizes(1));...
fprintf('%-1.3f] \n',Max_Sizes(2))
else
fprintf('     [%-1.3f, ',Max_Sizes(1));...
   fprintf('%-1.3f, ',Max_Sizes(2:dim(2)-1));...
   fprintf('%-1.3f] \n',Max_Sizes(dim(2)))
end
clear Titles
lowest_datum=min(Min_Sizes');
highest_datum=max(Max_Sizes');
fprintf('   (all the values are ''%s'') \n',units)
disp(' ')
disp(' In your whole spreadsheet, absolute minimum and maximum values are: ')
fprintf('   Minimal datum = %-1.3f \n',lowest_datum)
fprintf('   Maximal datum = %-1.3f \n',highest_datum)
fprintf('                  (%s) \n',units)
disp('               (Press [Enter] to continue...) ')
response=input(' ','s');
while isempty(response)==0
disp('(press [Enter] to continue...)')
response=input(' ','s');
end
Input3=char({'',' ALL FIRMS are going to be GROUPED ',...
' IN CLASSES OF DIFFERENT WIDTH, ',...
' splitting the whole domain of the variable values ',...
' in the following manner: '...
'','[lowest datum;sup(1)], (sup(1);sup(2)], (sup(2);sup(3)],.., (sup(n-1);highest datum] ',...
'',' Each class represents one of the states among which a transition happens ',...
' and will be identified by a progressive positive integer. ',...
'',' INSERT THE NUMBER OF CLASSES ',...
' and press  [Enter]. '});
disp(Input3)
classes_number=input('   ');
  while (isscalar(classes_number)==0 | isnan(classes_number)==1 | isinf(classes_number)==1 | classes_number<=0 | floor↙
(classes_number)~=classes_number...
       | isreal(classes_number)==0)
    if isscalar(classes_number)==0
     disp(' ( You pressed the [ENTER] button or inserted a vector or an array ! ) ')
     disp('   BUT  you must insert exclusively a positive integer ! ) ')
     classes_number=input('   ');
    elseif (isscalar(classes_number)==1 & (isnan(classes_number)==1 | isinf(classes_number)==1 | isreal(classes_number)↙
==0))
     disp(' Your input is a NaN value or a not finite one or it is not a ral number: WHY? ')
     classes_number=input('   ');
    elseif classes_number<=0
      disp(' ( You must insert exclusively a positive number ! ) ')
      classes_number=input('   ');
    elseif (classes_number>0 & floor(classes_number)~=classes_number)
      disp(' ( You must insert exclusively a positive integer ! ) ')
      classes_number=input('   ');
      end
  end
disp(' ')
fprintf('   Now write the sup for everyone of the first %-1.0d classes, \n',classes_number-1)
disp('   which are the inner points of your partition of the domain [ Minimal datum, Maximal  datum ] ')
disp('   (the last class is excluded because its sup is the Maximal datum just above) ')
disp('   BE CAREFUL NOT TO ROLL OUT the ENTER button if you don''t want to quit !! ')
for k=1:classes_number-1
  fprintf('Insert the top (expressed as number of ''%s'') of the %-1.0d° class:\n',units,k)
  sup(k)=input(' ');
   if k==1 & sup(1)>lowest_datum & sup(1)<highest_datum
     continue
    elseif k==1 & (sup(1)<=lowest_datum | sup(1)>=highest_datum)
```

```
315       while (sup(1)<=lowest_datum | sup(1)>=highest_datum)
316        if sup(1)<=lowest_datum
317         disp(char({' ERROR!! ',...
318         ' Insert a value bigger than Minimal datum (and smaller than Maximal datum): '}))
319         sup(1)=input(' ');
320        elseif sup(1)>=highest_datum
321         disp(char({' ERROR!! ',...
322         ' Insert a value smaller than Maximal datum (but bigger than Minimal datum): '}))
323         sup(1)=input(' ');
324        end
325       end
326     continue
327    end
328    %
329    if (sup(k)>lowest_datum & sup(k)>sup(k-1) & sup(k)<highest_datum)
330      continue
331    elseif (sup(k)<=lowest_datum | sup(k)<=sup(k-1) | sup(k)>=highest_datum)
332      while (sup(k)<=lowest_datum | sup(k)<=sup(k-1) | sup(k)>=highest_datum)
333       if sup(k)<=lowest_datum
334        disp(char({' ERROR!! ',...
335        ' Insert a value bigger than Minimal datum (but smaller than Maximal datum): '}))
336        sup(k)=input(' ');
337       elseif sup(k)>=highest_datum
338        disp(char({' ERROR!! ',...
339        ' Insert a value smaller than Maximal datum (and bigger than Minimal datum): '}))
340        sup(k)=input(' ');
341       elseif sup(k)<=sup(k-1)
342        disp(char({' ERROR!! ',...
343        ' Insert a value greater than the previous one '}))
344        sup(k)=input(' ');
345       end
346      end
347    end
348 end
349    Extrm=[lowest_datum,sup,highest_datum];
350 Input4=char({'',' The extremes of the partition you have chosen ',...
351 ' are arranged orderly as components inside the vector named "Extrm". ','',''});
352 disp(Input4)
353 %
354 %   num_k_Lstart_Lend  is the number of units belonging to the k-th class
355 %                      in the early period (Lstart) and having a numerical datum
356 %                      in the final period (Lend).
357 %   w_k_Lstart_Lend(k) is the new name of the former weight "w_i"
358 %                      for any k-th class as a transition starting state.
359 %                      It is defined as the percentage of units inside the k-th class
360 %                      in the early period with an available datum in the final period.
361 %
362 %   In some spreadsheets it may happen
363 %   some units are lacking of information at some dates.
364 %   ANY UNIT, WITHOUT A DEFINED FINAL VALUE FOR THE VARIABLE,
365 %   WILL BE EXCLUDED FROM THE COMPUTATION !!
366 %   The matrix "isnan_data" serves for this purpose.
367 %
368 clear dimensions Input0 Input1 Input2_1 Input2_2 Input3 Input4
369 isnan_data=isnan(MatData);
370 num_k_Lstart_Lend=zeros(classes_number,dim(2)-1,dim(2)-1);
371 num_tot_Lstart_Lend=zeros(dim(2)-1,dim(2)-1);
372 for Lstart=1:dim(2)-1
373     for Lend=Lstart+1:dim(2)
374 num_k_Lstart_Lend(1,Lstart,Lend)=sum(MatData(:,Lstart)<=sup(1) & isnan_data(:,Lend)==0);
375 for k=2:classes_number-1
376 num_k_Lstart_Lend(k,Lstart,Lend)=sum(MatData(:,Lstart)>sup(k-1) & MatData(:,Lstart)<=sup(k) & isnan_data(:,Lend)==0);
377 end  %(for "k")
378 num_k_Lstart_Lend(classes_number,Lstart,Lend)=sum(MatData(:,Lstart)>sup(classes_number-1) & isnan_data(:,Lend)==0);
```

```
379     end %(for "Lend")
380 end  %(for "Lstart")
381 for Lstart=1:dim(2)-1
382 for Lend=Lstart+1:dim(2)
383 num_tot_Lstart_Lend(Lstart,Lend)=sum(num_k_Lstart_Lend(:,Lstart,Lend));
384 end
385 end
386 default=char({' By default the weights {w_i} are the rates of individuals ',...
387 ' starting a transition from the state "i" at time Tstart ',...
388 ' to reach anyone of the states at time Tend. ',...
389 ' If you want to change, it is possible to assign a number, ',...
390 ' different from the starting probability, ',...
391 ' weighting every single initial state "i" inside the Index; ',...
392 '',' Do you want to modify the default weights? ',...
393 ' ( "Y" for ''Yes'' or "N" for ''Not'', then [Enter]) '});
394 Input_5ter=char({'','','        Lastly write, in the command line, the handle  " @(x) " ',...
395 '        followed by the mathematical shape of your v(|i-j|), ',...
396 ' according to the MATLAB coding for operations and functions: ',...
397 ' for example   @(x)exp(x^(1/3))   or   @(x)log(2*x^3)  . ',...
398 '        You can express it respect to the auxiliary variable "x", ',...
399 '        so that x = |i-j| : anyway, to avoid conflicts, ',...
400 '        you must refer the handle "@" and the function to the same variable. ',''});
401 disp(default)
402 resp_w=input('  ','s');
403 while strcmp(resp_w,{'Y','y','N','n'})==[0,0,0,0]
404       disp('   (You must answer exclusively "Y" or "N" !) ')
405       resp_w=input('  ','s');
406 end
407 if sum(strcmp(resp_w,{'N','n'}))==1
408 for Lstart=1:dim(2)-1
409     for Lend=Lstart+1:dim(2)
410 w_k_Lstart_Lend(1,Lstart,Lend)=num_k_Lstart_Lend(1,Lstart,Lend)./sum(MatData(:,Lstart)>=lowest_datum & isnan_data(:,↙
Lend)==0);
411 for k=2:classes_number-1
412 w_k_Lstart_Lend(k,Lstart,Lend)=num_k_Lstart_Lend(k,Lstart,Lend)./sum(MatData(:,Lstart)>=lowest_datum & isnan_data(:,↙
Lend)==0);
413 end  %(for "k")
414 w_k_Lstart_Lend(classes_number,Lstart,Lend)=num_k_Lstart_Lend(classes_number,Lstart,Lend)./sum(MatData(:,Lstart)↙
>=lowest_datum & isnan_data(:,Lend)==0);
415     end %(for "Lend")
416 end  %(for "Lstart")
417 elseif sum(strcmp(resp_w,{'Y','y'}))==1
418 change=char({'   If you desire to decide which scalar weights each initial state, ',...
419 '   not belonging to the starting distribution, BEWARE that ',...
420 '   THE SUM OVER ALL THE STATES MUST BE EQUAL TO 1. ',...
421 '',' If you dislike the option about {w_i} ',...
422 '  you have to modify directly the part of this script concerning definitions ',...
423 '  of  " w_k_Lstart_Lend(k,Lstart,Lend) ". ',...
424 '','BE also CAREFUL NOT TO PINCH the ENTER button if you don''t want to quit !!'});
425 disp(change)
426   for k=1:classes_number
427   fprintf('Insert the numerical value of the %-1.0d° weight w(%-1.0d):\n',k,k)
428   w_k(k)=input(' ');
429   while w_k(k)<0 || w_k(k)>1 || isnan(w_k(k))==1 || isreal(w_k(k))==0 || isscalar(w_k(k))==0
430     if w_k(k)<0
431       disp(' This input is a probability: you have to insert a positive number !! ')
432       w_k(k)=input(' ');
433     elseif w_k(k)>1
434        disp(char({' This input is a probability of a single event: ',...
435       ' it cannot be a number greater than 1 !! '}))
436       w_k(k)=input(' ');
437     elseif (isnan(w_k(k))==1 || isscalar(w_k(k))==0)
438        disp(' This input is a probability: you have to insert a scalar !! ')
439        w_k(k)=input(' ');
```

```
  elseif isreal(w_k(k))==0
     disp(' This input is a probability: you have to insert a real scalar !! ')
     w_k(k)=input(' ');
  end
end
end
sum_w_k=single(sum(w_k));
if (sum_w_k>1 | sum_w_k<1)
while (sum_w_k>1 | sum_w_k<1)
warn=char({' Pay attention: ',...
' THE SUM OF ALL THE PREVIOUS INPUTS IS NOT EQUAL TO 1 !! ',...
' YOU NEED TO INSERT ALL THE WEIGHTS AGAIN. ','',...
' Be careful not to pinch the enter button !'});
disp(warn)
for k=1:classes_number
fprintf('Insert the numerical value of the %-1.0d° weight w(%-1.0d):\n',k,k)
w_k(k)=input(' ');
while w_k(k)<0 || w_k(k)>1 || isnan(w_k(k))==1 || isreal(w_k(k))==0 || isscalar(w_k(k))==0
  if w_k(k)<0
    disp(' This input is a probability: you have to insert a positive number !! ')
    w_k(k)=input(' ');
  elseif w_k(k)>1
     disp(char({' This input is a probability of a single event: ',...
    ' it cannot be a number greater than 1 !! '}))
    w_k(k)=input(' ');
  elseif (isnan(w_k(k))==1 || isscalar(w_k(k))==0)
     disp(' This input is a probability: you have to insert a scalar !! ')
     w_k(k)=input(' ');
  elseif isreal(w_k(k))==0
     disp(' This input is a probability: you have to insert a real scalar !! ')
     w_k(k)=input(' ');
  end
end
end
sum_w_k=single(sum(w_k));
end
else
end
for Lstart=1:dim(2)-1
  for Lend=Lstart+1:dim(2)
    for k=1:classes_number
w_k_Lstart_Lend(k,Lstart,Lend)=w_k(k);
    end
  end
end
end
disp(Input_5ter)
%
%  It is applied the classical definition of Laplace and Bernoulli
%  (the number of favorable cases divided by the number of all possible cases)
%  to estimate, by the data sample, the following:
%
%  P_k_h_Lstart_Lend(k,h,Lstart,Lend) which is the probability that a statistical unit may make a transition
%                                  from the initial k-th class to the final h-th one, reached in the Lend-th period.
%
%  In " P_k_h_Lstart_Lend " the divider of the ratio (the number of all possible cases)
%  is the number of all units completing a transition between Lstart and Lend,
%  whatever the initial status and the destination.
%  The top of the same ratio is:
%
%  Number_k_h_Lstart_Lend(k,h,Lstart,Lend) that is the number of units moving from the state
%                                     named as "k", occupied at time Lstart, to the state "h",
%                                     where they are observed at time Lend.
%
```

```
%
%  We are also going to introduce
%
%  V_k_h(h,k)  the suitable function of the difference (h-k), weighting for the index
%              the significance of a jump from the k-th class to the h-th one
%              along with its orientation.
%                     V_k_h(h,k) = sign(h-k)*v(|h-k|)
%
%  Below an auxiliary variable, named “x”, will be used.
%
v=input('  v(|i-j|) = v(x)=  ');
%
%  The function handle “ v ”, inserted by you, corresponds to the weight function  v(|i-j|)
%  that is the handle in x=0 is the weight  v(|i-j|)  computed in |i-j|=0,
%  the handle in x=1 is the weight  v(|i-j|)  computed in |i-j|=1, and so on.
%
while isa(v,'function_handle')==0
    disp(' Your input is not a function handle: ')
    disp(' please follow the instructon above and retype your function. ')
    v=input('  v(|i-j|) = v(x)=  ');
end
 for h=1:classes_number
  for k=1:classes_number
  V_k_h(h,k)=sign(h-k)*feval(v,abs(h-k));
%
%  for the matrix V_k_h, 'h' is a row index and 'k' is a column index
%  and they appear, as parameters, also in the scalar ‘abs(h-k)’
%  which is a non negative integer.
%
  end
 end
V_nan=isnan(V_k_h);
V_infinite=isinf(V_k_h);
V_real=isreal(V_k_h);
 while (isequal(V_nan,zeros(classes_number,classes_number))==0 | isequal(V_infinite,zeros(classes_number,classes_number))↙
==0 |...
       V_real==0)
   disp(char({'','  Note that the function  " v(|i-j|) "  of the shape you have chosen ',...
              '  is not defined over the whole domain of the couple (i,j): ',...
              '  it could invalidate the computation of the Index. ',...
              '  You can check it by observing the position of all the possible ',...
              '  NaN values, infinite values or non-real values ',...
              '  in the following matrix  " V_k_h(h,k) " : ',''}))
   disp(V_k_h)
   disp(char({'','  So insert a more convenient functional " v(|i-j|) ", ',...
              '  defined for each possible value of the couple (i,j). ',...
              '  The name ''x'' for the auxiliary variable is only a suggestion: ',''}))
 v=input('  v(|i-j|) = v(x)=  ');
 while isa(v,'function_handle')==0
    disp(' Your input is not a function handle: ')
    disp(' please follow the instructon above and retype your function. ')
    v=input('  v(|i-j|) = v(x)=  ');
 end
 for h=1:classes_number
  for k=1:classes_number
   V_k_h(h,k)=sign(h-k)*feval(v,abs(h-k));
  end
 end
  V_nan=isnan(V_k_h);
  V_infinite=isinf(V_k_h);
  V_real=isreal(V_k_h);
 end
for Lstart=1:dim(2)-1
 for Lend=Lstart+1:dim(2)
```

```
%
Number_k_h_Lstart_Lend(1,1,Lstart,Lend)=sum(MatData(:,Lstart)<=sup(1) & MatData(:,Lend)<=sup(1));
%
  for h=2:classes_number-1
Number_k_h_Lstart_Lend(1,h,Lstart,Lend)=sum(MatData(:,Lstart)<=sup(1) &(MatData(:,Lend)>sup(h-1) & MatData(:,Lend)<=sup(h)));
Number_k_h_Lstart_Lend(classes_number,h,Lstart,Lend)=sum(MatData(:,Lstart)>sup(classes_number-1) &(MatData(:,Lend)>sup(h-1) & MatData(:,Lend)<=sup(h)));
  end % (for "h")
Number_k_h_Lstart_Lend(1,classes_number,Lstart,Lend)=sum(MatData(:,Lstart)<=sup(1) & MatData(:,Lend)>sup(classes_number-1));
%
  for k=2:classes_number-1
Number_k_h_Lstart_Lend(k,1,Lstart,Lend)=sum((MatData(:,Lstart)>sup(k-1) & MatData(:,Lstart)<=sup(k))& MatData(:,Lend)<=sup(1));
Number_k_h_Lstart_Lend(k,classes_number,Lstart,Lend)=sum((MatData(:,Lstart)>sup(k-1) & MatData(:,Lstart)<=sup(k))&MatData(:,Lend)>sup(classes_number-1));
  end % (for "k")
Number_k_h_Lstart_Lend(classes_number,1,Lstart,Lend)=sum(MatData(:,Lstart)>sup(classes_number-1) & MatData(:,Lend)<=sup(1));
%
  for h=2:classes_number-1
   for k=2:classes_number-1
Number_k_h_Lstart_Lend(k,h,Lstart,Lend)=sum((MatData(:,Lstart)>sup(k-1) & MatData(:,Lstart)<=sup(k))&(MatData(:,Lend)>sup(h-1) & MatData(:,Lend)<=sup(h)));
   end  %(for "k")
  end  %(for "h")
%
Number_k_h_Lstart_Lend(classes_number,classes_number,Lstart,Lend)=sum(MatData(:,Lstart)>sup(classes_number-1) &MatData(:,Lend)>sup(classes_number-1));
 end  %(for "Lend")
end  %(for "Lstart")
disp(Input0bis)
Input6=char({'','',' Now this tool is about to begin the computation ',...
' of the normalized Directional Mobility Index. ',...
' Its value will be calculated respect to the starting period " Tstart " ',...
' for one of the subsequent periods " Tend " (Tend > Tstart). ',...
'',' Please, insert your favorite start " Tstart ", ',...
' and indicate it by a positive integer (1,2,..) corresponding ',...
' to its location inside the ordered sequence of the observation dates: '});
disp(Input6)
Tstart=input('      Tstart = ');
 if (isscalar(Tstart)==0 | isreal(Tstart)==0 | isnan(Tstart)==1 | isinf(Tstart)==1 | not(Tstart>0 & floor(Tstart)==ceil(Tstart) &Tstart<dim(2)))
  while (isscalar(Tstart)==0 | isreal(Tstart)==0 | isnan(Tstart)==1 | isinf(Tstart)==1 | not(Tstart>0 & floor(Tstart)==ceil(Tstart)& Tstart<dim(2)))
    if (isscalar(Tstart)==0)
     disp(' ( You pressed the [ENTER] button or inserted a vector or an array ! ) ')
     disp('   BUT  you must insert exclusively a positive integer ! ) ')
     Tstart=input('      Tstart = ');
    elseif (isnan(Tstart)==1 | isinf(Tstart)==1 | isreal(Tstart)==0)
       disp(' Your input is a NaN value or infinite or a complex number: WHY? ')
       Tstart=input('      Tstart = ');
    elseif not(Tstart>0 & floor(Tstart)==ceil(Tstart) & Tstart<dim(2))
    disp(' You must insert a positive integer (1,2,3,...) !! ')
    Tstart=input('      Tstart = ');
    elseif Tstart==dim(2)
    disp(' Your value for Tstart cannot correspond to the last detection !! ')
    Tstart=input('      Tstart = ');
    elseif Tstart>dim(2)
    disp(' Your value for Tstart is too big: there are not enough periods in the data !! ')
    Tstart=input('      Tstart = ');
    end
    end
```

```
 elseif (isscalar(Tstart)==1 & isreal(Tstart)==1 & isnan(Tstart)==0 & isinf(Tstart)==0 & (Tstart>0 & floor(Tstart)==ceil(Tstart) & Tstart<dim(2)))
 end
Input7=char({'',' Insert the finish " Tend " of your interest, ',...
' it also denoted as a progressive positive integer (2,3,..) ',...
' corresponding to its position inside the ordered sequence of dates ',...
' and greater than Tstart: '});
disp(Input7)
Tend=input('      Tend = ');
while (isscalar(Tend)==0 | isreal(Tend)==0 | isnan(Tend)==1 | isinf(Tend)==1 | not(Tend>Tstart & floor(Tend)==ceil(Tend) & Tend<=dim(2)))
    if isscalar(Tend)==0
      disp(' ( You pressed the [ENTER] button or inserted a vector or an array ! ) ')
      disp('   BUT  you must insert exclusively a positive integer ! ) ')
      Tend=input('      Tend = ');
    elseif isnan(Tend)==1 | (isinf(Tend)==1 & Tend>dim(2)) | isreal(Tend)==0
        disp(' Your input is a NaN value or infinite or a complex number: WHY? ')
        Tend=input('      Tend = ');
    elseif (Tend<=0|(Tend>0 & floor(Tend)~=ceil(Tend)))
    disp(' You must insert a positive integer (2,3,4,..) !! ')
    Tend=input('      Tend = ');
    elseif Tend<=Tstart
    disp(' Tend must be greater than Tstart !! ')
    Tend=input('      Tend = ');
    elseif Tend>dim(2) & isinf(Tend)==0
    disp(' Your value for Tend is too big: there are not enough periods !! ')
    Tend=input('      Tend = ');
    end
end
clear Input0bis Input5 default change Input_5bis Input_5ter warn Input6 Input7
Output1=char({'','',' The weights {w_i} will appear as rows ',...
' inside the array  " w_k_Lstart_Lend(k,Lstart,Lend) " ). ','',...
' The greatest and the smallest values of the not-normalized Index ',...
' may depend on the initial period "Lstart" and the final one "Lend" ',...
' and will be collected into the 2-dim arrays ',...
' " m1(Lstart,Lend) "  and  " m2(Lstart,Lend) "  respectively. ',...
'',' All the probabilities are estimated in the classical Laplacian manner !! ',...
'       (press [Enter] on the keyboard to begin) '});
disp(Output1)
response=input(' ','s');
while isempty(response)==0
disp('(press [Enter] to begin)')
response=input(' ','s');
end
for Lstart=1:dim(2)-1
    for Lend=Lstart+1:dim(2)
m1(Lstart,Lend)=(V_k_h(1,:))*(w_k_Lstart_Lend(:,Lstart,Lend));
m2(Lstart,Lend)=(V_k_h(classes_number,:))*(w_k_Lstart_Lend(:,Lstart,Lend));
    end
end
%
%  m1 and m2  are, respectively, the global minimum and the global maximum
%  theoretically possible for the not-normalized index.
%  They are necessary to carry out the normalization of the Directional Mobility Index
%  to the target [-1;+1] in the following manner:
%       if the not normalized index is negative then
%                   normalized index =  -1*(not normalized index) / m1;
%       else if the not normalized index is non negative then
%                   normalized index =  (not normalized index) / m2;
%
P_k_h_Tstart_Tend=zeros(classes_number,classes_number);
Pup_k_h_Tstart_Tend=zeros(classes_number,classes_number);
Pdn_k_h_Tstart_Tend=zeros(classes_number,classes_number);
  for h=1:classes_number
```

```
  for k=1:classes_number
if num_k_Lstart_Lend(k,Tstart,Tend)==0
 continue
end
  P_k_h_Tstart_Tend(k,h)=(Number_k_h_Lstart_Lend(k,h,Tstart,Tend)./...
     sum(MatData(:,Tstart)>=lowest_datum & MatData(:,Tend)>=lowest_datum)) ...
     ./(num_k_Lstart_Lend(k,Tstart,Tend)./(sum(MatData(:,Tstart)<=highest_datum & isnan_data(:,Tend)==0)));
  end
 end
 for h=1:classes_number
  for k=1:classes_number
     if k>=h
        continue
     end
    Pup_k_h_Tstart_Tend(k,h)=P_k_h_Tstart_Tend(k,h);
  end
 end
 for h=1:classes_number
  for k=1:classes_number
     if k<=h
        continue
     end
    Pdn_k_h_Tstart_Tend(k,h)=P_k_h_Tstart_Tend(k,h);
  end
 end
 for k=1:classes_number
 Y_V_k_Tstart_Tend(k)=P_k_h_Tstart_Tend(k,:)*V_k_h(:,k);
 Yup_V_k_Tstart_Tend(k)=Pup_k_h_Tstart_Tend(k,:)*V_k_h(:,k);
 Ydn_V_k_Tstart_Tend(k)=Pdn_k_h_Tstart_Tend(k,:)*V_k_h(:,k);
 ImmobilShares_k_Tstart_Tend(k)=Number_k_h_Lstart_Lend(k,k,Tstart,Tend)./num_tot_Lstart_Lend(Tstart,Tend);
 end
I_w_V_Tstart_Tend=(w_k_Lstart_Lend(:,Tstart,Tend))'*Y_V_k_Tstart_Tend(:);
Iup_w_V_Tstart_Tend=(w_k_Lstart_Lend(:,Tstart,Tend))'*Yup_V_k_Tstart_Tend(:);
Idn_w_V_Tstart_Tend=(w_k_Lstart_Lend(:,Tstart,Tend))'*Ydn_V_k_Tstart_Tend(:);
%
%       I_w_V_Tstart_Tend = I_w_V_Lstart_Lend(Tstart,Tend)
%   is the not-normalized version of the Directional Mobility Index.
%       Iup_w_V_Tstart_Tend = Iup_w_V_Lstart_Lend(Tstart,Tend)
%   and  Idn_w_V_Tstart_Tend = Idn_w_V_Lstart_Lend(Tstart,Tend)
%   are respectively its upward and downward components.
%   The new value of the Normalized Directional Mobility Index
%       Inorm_w_V_Tstart_Tend = Inorm_w_V_Lstart_Lend(Tstart,Tend)
%   along with its respective upward and downward components
%   are:
%
  if I_w_V_Tstart_Tend>=0
    Inorm_w_V_Tstart_Tend=I_w_V_Tstart_Tend./m2(Tstart,Tend);
    Inorm_up_w_V_Tstart_Tend=Iup_w_V_Tstart_Tend./m2(Tstart,Tend);
    Inorm_dn_w_V_Tstart_Tend=Idn_w_V_Tstart_Tend./m2(Tstart,Tend);
  elseif I_w_V_Tstart_Tend<0
    Inorm_w_V_Tstart_Tend=-I_w_V_Tstart_Tend./m1(Tstart,Tend);
    Inorm_up_w_V_Tstart_Tend=-Iup_w_V_Tstart_Tend./m1(Tstart,Tend);
    Inorm_dn_w_V_Tstart_Tend=-Idn_w_V_Tstart_Tend./m1(Tstart,Tend);
  end
  UPvsDN_w_V_Tstart_Tend=Iup_w_V_Tstart_Tend./abs(Idn_w_V_Tstart_Tend);
disp(' ')
Itrace_Tstart_Tend=(classes_number-trace(P_k_h_Tstart_Tend))/(classes_number-1); % Trace index of mobility
Idet_Tstart_Tend=1-abs(det(P_k_h_Tstart_Tend)); % Determinant index of mobility
[eigenvects_Tstart_Tend,eigenvals_Tstart_Tend]=eig(P_k_h_Tstart_Tend,'vector');
abs_eigenvals_Tstart_Tend=abs(eigenvals_Tstart_Tend);
sorted_abs_eigenvals_Tstart_Tend=sort(abs_eigenvals_Tstart_Tend,'descend');
I2nd_eigval_Tstart_Tend=1-sorted_abs_eigenvals_Tstart_Tend(2); % Mobility index of the 2nd biggest eigenvalue
Ieigvals_Tstart_Tend=(classes_number-sum(abs_eigenvals_Tstart_Tend))/(classes_number-1); % All eigenvalues mobility
index
```

```
Ipredict_Tstart_Tend=classes_number*(sum(sum((P_k_h_Tstart_Tend).^2,2))-1)/(classes_number-1); % Index of predictability
for k=1:classes_number
    w_k_Tstart_Tend(k)=w_k_Lstart_Lend(k,Tstart,Tend);
    for h=1:classes_number
        Intensity_k_h(k,h)=abs(h-k);
    end
end
BMorrUP_Tstart_Tend=w_k_Tstart_Tend*sum(Pup_k_h_Tstart_Tend,2); % Bourguignon and Morrison Upward Index
BMorrDN_Tstart_Tend=w_k_Tstart_Tend*sum(Pdn_k_h_Tstart_Tend,2); % Bourguignon and Morrison Downward Index
I_2ndBarthol_Tstart_Tend=(classes_number/(classes_number-1))*(sum(sum(P_k_h_Tstart_Tend.*Intensity_k_h))); % Second Index of Bartholomew
disp(' The value of the normalized Directional Mobility Index, ')
fprintf(' from Tstart = %-1.0d  to  Tend = %-1.0d , is: \n ',Tstart,Tend)
fprintf('        Inorm_w_v = %-1.7g \n',Inorm_w_V_Tstart_Tend)
disp(char({'',' You can compare its value to the ones, for the same transition matrix, ',...
' between the same dates, of some other not-directional mobility indices ',...
' well known in the literature: ',...
' the trace and the determinant indices, " Itrace_Tstart_Tend " and " Idet_Tstart_Tend ", ',...
' the All Eigenvalues index, " Ieigvals_Tstart_Tend " ',...
' and the Second Eigenvalue index, " I2nd_eigval_Tstart_Tend ", ',...
' the Index of predictability " Ipredict_Tstart_Tend " ',...
' and the Second Index of Bartholomew " I_2ndBarthol_Tstart_Tend " . ','',...
' The downward and upward components of the Normalized Directional Index ',...
' will be compared to the Bourguignon-Morrison Downward and Upward Indices, ',...
' indicated as " BMorrDN_Tstart_Tend " and " BMorrUP_Tstart_Tend " . ','',...
' Moreover the eigenvalues of the transition matrix " P_k_h_Tstart_Tend " are collected ',...
' in the column vector " eigenvals_Tstart_Tend ", whose components correspond orderly, ',...
' one-to-one, to the columns of the matrix " eigenvects_Tstart_Tend ", where are stored ',...
' the relative eigenvectors of " P_k_h_Tstart_Tend ". ',''}))
fprintf('        Inorm_w_v =    %-1.7g \n',Inorm_w_V_Tstart_Tend)
fprintf('        Itrace =       %-1.7g \n',Itrace_Tstart_Tend)
fprintf('        Idet =         %-1.7g \n',Idet_Tstart_Tend)
fprintf('        I2nd_eigval =  %-1.7g \n',I2nd_eigval_Tstart_Tend)
fprintf('        Ieigvals =     %-1.7g \n',Ieigvals_Tstart_Tend)
fprintf('        Ipredict =     %-1.7g \n',Ipredict_Tstart_Tend)
fprintf('        I_2ndBarthol = %-1.7g \n',I_2ndBarthol_Tstart_Tend)
disp(' ')
disp(' The downward and upward components of the normalized Index are respectively: ')
fprintf(' Inorm_dn_w_v = %-1.7g \n',Inorm_dn_w_V_Tstart_Tend)
fprintf(' Inorm_up_w_v = %-1.7g \n',Inorm_up_w_V_Tstart_Tend)
disp(' Compare to the values of the Downward and Upward Morrison-Bourguignon Indices, ')
disp(' which are respectively: ')
fprintf(' BMorrDN = %-1.7g \n',BMorrDN_Tstart_Tend)
fprintf(' BMorrUP = %-1.7g \n',BMorrUP_Tstart_Tend)
disp(' ')
disp(' The ratio between the upward and the downward components of the Index is: ')
fprintf(' UPvsDN_w_V = %-1.7g \n',UPvsDN_w_V_Tstart_Tend)
fprintf(' and the percentages of agents remaining into each starting state at Tend=%-1.0d are: \n ',Tend)
fprintf('   [ImmobilShare_1,.., ImmobilShare_%-1.0d] = [',classes_number);...
            fprintf('%-1.5g, ',(ImmobilShares_k_Tstart_Tend(1:classes_number-1))');...
            fprintf('%-1.5g] \n',(ImmobilShares_k_Tstart_Tend(classes_number))')
disp(' ')
fprintf(' The current values of parameters {w_i, i = 1,..,%-1.0d}, \n',classes_number)
disp(' computed from the imported data, in this step are: ')
fprintf('   [w_1,.., w_%-1.0d] = [',classes_number);...
            fprintf('%-1.5g, ',(w_k_Lstart_Lend(1:classes_number-1,Tstart,Tend))');...
            fprintf('%-1.5g] \n',(w_k_Lstart_Lend(classes_number,Tstart,Tend))')
Output_zero=char({'',' Would you like to see the conditional probability ',...
'          " P_k_h_Tstart_Tend " ',...
' that any unit moved from the early k-th class to the h-th one',...
' between Tstart and Tend chosen by you ? ',...
' (Answer "Y" if ''yes'' or "N" if ''not'', then press [Enter]) '});
disp(Output_zero)
answer=input('  ','s');
```

```
809 while strcmp(answer,{'Y','y','N','n'})==[0,0,0,0]
810            disp('   (You must answer exclusively "Y" or "N" !) ')
811        answer=input('  ','s');
812 end
813 if sum(strcmp(answer,{'N','n'}))==1
814 disp(char({'  Ok: anyway, you can stop this run and quit ','  typing "Ctrl" + "C" on the keyboard. '}))
815 elseif sum(strcmp(answer,{'Y','y'}))==1
816 disp('    The transition probability is: ')
817 fprintf('    P_k_h_Tstart_Tend(k,h, Tstart = %-1.0d, Tend = %-1.0d)= \n ',Tstart,Tend)
818 disp(' ')
819 disp(P_k_h_Tstart_Tend)
820 end
821 Output_0bis=char({'',' Do you want to change the temporal extremes Tstart and Tend ? ',...
822 ' ("Y" for ''Yes'' or "N" for ''Not'' then press [Enter]) '});
823 disp(Output_0bis)
824 response=input('  ','s');
825 while strcmp(response,{'Y','y','N','n'})==[0,0,0,0]
826            disp('   (You must answer exclusively "Y" or "N" !) ')
827        response=input('  ','s');
828 end
829 if sum(strcmp(response,{'N','n'}))==1
830 clear Output_zero Output_0bis Output1 Output_1bis answer response resp
831 elseif sum(strcmp(response,{'Y','y'}))==1
832  while sum(strcmp(response,{'Y','y'}))==1
833     clear Tstart Tend
834  Input_6bis=char({'',' Insert the new start " Tstart ", ',...
835  ' indicating it by a positive integer corresponding to its location ',...
836  ' inside the ordered sequence of the observations: '});
837 disp(Input_6bis)
838 Tstart=input('      Tstart = ');
839  if (isscalar(Tstart)==0 | isreal(Tstart)==0 | isnan(Tstart)==1 | isinf(Tstart)==1 | not(Tstart>0 & floor(Tstart)==ceil(Tstart) &↙
Tstart<dim(2)))
840   while (isscalar(Tstart)==0 | isreal(Tstart)==0 | isnan(Tstart)==1 | isinf(Tstart)==1 | not(Tstart>0 & floor(Tstart)==ceil(Tstart)↙
& Tstart<dim(2)))
841      if (isscalar(Tstart)==0)
842       disp(' ( You pressed the [ENTER] button or inserted a vector or an array ! ) ')
843       disp('   BUT  you must insert exclusively a positive integer ! ) ')
844       Tstart=input('      Tstart = ');
845      elseif (isnan(Tstart)==1 | isinf(Tstart)==1 | isreal(Tstart)==0)
846         disp(' Your input is a NaN value or infinite or a complex number: WHY? ')
847         Tstart=input('      Tstart = ');
848      elseif not(Tstart>0 & floor(Tstart)==ceil(Tstart) & Tstart<dim(2))
849      disp(' You must insert a positive integer (1,2,3,...) !! ')
850      Tstart=input('      Tstart = ');
851      elseif Tstart==dim(2)
852      disp(' Your value for Tstart cannot correspond to the last detection !! ')
853      Tstart=input('      Tstart = ');
854      elseif Tstart>dim(2)
855      disp(' Your value for Tstart is too big: there are not enough periods in the data !! ')
856      Tstart=input('      Tstart = ');
857      end
858      end
859  elseif (isscalar(Tstart)==1 & isreal(Tstart)==1 & isnan(Tstart)==0 & isinf(Tstart)==0 & (Tstart>0 & floor(Tstart)==ceil↙
(Tstart) & Tstart<dim(2)))
860  end
861 Input_7bis=char({' Then, insert the new finish " Tend " ',...
862 ' it also denoted as a progressive positive integer ',...
863 ' according to its position inside the ordered sequence of dates ',...
864 ' and greater than Tstart: '});
865 disp(Input_7bis)
866  Tend=input('      Tend = ');
867  while (isscalar(Tend)==0 | isreal(Tend)==0 | isnan(Tend)==1 | isinf(Tend)==1 | not(Tend>Tstart & floor(Tend)==ceil(Tend)↙
& Tend<=dim(2)))
868     if isscalar(Tend)==0
```

```
    disp(' ( You pressed the [ENTER] button or inserted a vector or an array ! ) ')
    disp('   BUT  you must insert exclusively a positive integer ! ) ')
    Tend=input('      Tend = ');
  elseif isnan(Tend)==1 | (isinf(Tend)==1 & Tend>dim(2)) | isreal(Tend)==0
      disp(' Your input is a NaN value or infinite or a complex number: WHY? ')
      Tend=input('      Tend = ');
  elseif (Tend<=0|(Tend>0 & floor(Tend)~=ceil(Tend)))
  disp(' You must insert a positive integer (2,3,4,..) !! ')
  Tend=input('      Tend = ');
  elseif Tend<=Tstart
  disp(' Tend must be greater than Tstart !! ')
  Tend=input('      Tend = ');
  elseif Tend>dim(2) & isinf(Tend)==0
  disp(' Your value for Tend is too big: there are not enough periods !! ')
  Tend=input('      Tend = ');
  end
end
Output_1bis=char({'',' Press [Enter] on the keyboard ',' to recompute the Index between the new extremes ',...
' (it would be better to take a note of its previous value, ',...
' displayed inside the Command Window, ',...
' and not to command "clc" after exiting). '});
disp(Output_1bis)
response=input(' ','s');
while isempty(response)==0
disp('(press [Enter] to begin)')
response=input(' ','s');
end
P_k_h_Tstart_Tend=zeros(classes_number,classes_number);
Pup_k_h_Tstart_Tend=zeros(classes_number,classes_number);
Pdn_k_h_Tstart_Tend=zeros(classes_number,classes_number);
 for h=1:classes_number
  for k=1:classes_number
if num_k_Lstart_Lend(k,Tstart,Tend)==0
 continue
end
  P_k_h_Tstart_Tend(k,h)=(Number_k_h_Lstart_Lend(k,h,Tstart,Tend)./...
     sum(MatData(:,Tstart)>=lowest_datum & MatData(:,Tend)>=lowest_datum)) ...
     ./(num_k_Lstart_Lend(k,Tstart,Tend)./(sum(MatData(:,Tstart)<=highest_datum & isnan_data(:,Tend)==0)));
  end
 end
  for h=1:classes_number
  for k=1:classes_number
     if k>=h
        continue
     end
    Pup_k_h_Tstart_Tend(k,h)=P_k_h_Tstart_Tend(k,h);
  end
 end
 for h=1:classes_number
  for k=1:classes_number
     if k<=h
        continue
     end
    Pdn_k_h_Tstart_Tend(k,h)=P_k_h_Tstart_Tend(k,h);
  end
 end
 for k=1:classes_number
 Y_V_k_Tstart_Tend(k)=P_k_h_Tstart_Tend(k,:)*V_k_h(:,k);
 Yup_V_k_Tstart_Tend(k)=Pup_k_h_Tstart_Tend(k,:)*V_k_h(:,k);
 Ydn_V_k_Tstart_Tend(k)=Pdn_k_h_Tstart_Tend(k,:)*V_k_h(:,k);
 ImmobilShares_k_Tstart_Tend(k)=Number_k_h_Lstart_Lend(k,k,Tstart,Tend)./num_tot_Lstart_Lend(Tstart,Tend);
 end
I_w_V_Tstart_Tend=(w_k_Lstart_Lend(:,Tstart,Tend))'*Y_V_k_Tstart_Tend(:);
Iup_w_V_Tstart_Tend=(w_k_Lstart_Lend(:,Tstart,Tend))'*Yup_V_k_Tstart_Tend(:);
```

```
Idn_w_V_Tstart_Tend=(w_k_Lstart_Lend(:,Tstart,Tend))'*Ydn_V_k_Tstart_Tend(:);
%
%    The new value of the Normalized Directional Mobility Index
%         Inorm_w_V_Tstart_Tend = Inorm_w_V_Lstart_Lend(Tstart,Tend)
%    along with its respective upward and downward components
%    are:
%
  if I_w_V_Tstart_Tend>=0
     Inorm_w_V_Tstart_Tend=I_w_V_Tstart_Tend./m2(Tstart,Tend);
     Inorm_up_w_V_Tstart_Tend=Iup_w_V_Tstart_Tend./m2(Tstart,Tend);
     Inorm_dn_w_V_Tstart_Tend=Idn_w_V_Tstart_Tend./m2(Tstart,Tend);
  elseif I_w_V_Tstart_Tend<0
     Inorm_w_V_Tstart_Tend=-I_w_V_Tstart_Tend./m1(Tstart,Tend);
     Inorm_up_w_V_Tstart_Tend=-Iup_w_V_Tstart_Tend./m1(Tstart,Tend);
     Inorm_dn_w_V_Tstart_Tend=-Idn_w_V_Tstart_Tend./m1(Tstart,Tend);
  end
  UPvsDN_w_V_Tstart_Tend=Iup_w_V_Tstart_Tend./abs(Idn_w_V_Tstart_Tend);
disp(' ')
Itrace_Tstart_Tend=(classes_number-trace(P_k_h_Tstart_Tend))/(classes_number-1);
Idet_Tstart_Tend=1-abs(det(P_k_h_Tstart_Tend));
[eigenvects_Tstart_Tend,eigenvals_Tstart_Tend]=eig(P_k_h_Tstart_Tend,'vector');
abs_eigenvals_Tstart_Tend=abs(eigenvals_Tstart_Tend);
sorted_abs_eigenvals_Tstart_Tend=sort(abs_eigenvals_Tstart_Tend,'descend');
I2nd_eigval_Tstart_Tend=1-sorted_abs_eigenvals_Tstart_Tend(2);
Ieigvals_Tstart_Tend=(classes_number-sum(abs_eigenvals_Tstart_Tend))/(classes_number-1);
Ipredict_Tstart_Tend=classes_number*(sum(sum((P_k_h_Tstart_Tend).^2,2))-1)/(classes_number-1);
for k=1:classes_number
    w_k_Tstart_Tend(k)=w_k_Lstart_Lend(k,Tstart,Tend);
end
BMorrUP_Tstart_Tend=w_k_Tstart_Tend*sum(Pup_k_h_Tstart_Tend,2);
BMorrDN_Tstart_Tend=w_k_Tstart_Tend*sum(Pdn_k_h_Tstart_Tend,2);
I_2ndBarthol_Tstart_Tend=(classes_number/(classes_number-1))*(sum(sum(P_k_h_Tstart_Tend.*Intensity_k_h)));
disp(' The value of the normalized Directional Mobility Index, ')
fprintf(' from Tstart = %-1.0d  to  Tend = %-1.0d , is: \n ',Tstart,Tend)
fprintf('        Inorm_w_v = %-1.7g \n',Inorm_w_V_Tstart_Tend)
disp(char({'',' Compare it, for the same transition matrix and between the same dates, ',...
' to the other not-directional mobility indices ',''}))
fprintf('        Inorm_w_v =    %-1.7g \n',Inorm_w_V_Tstart_Tend)
fprintf('        Itrace =       %-1.7g \n',Itrace_Tstart_Tend)
fprintf('        Idet =         %-1.7g \n',Idet_Tstart_Tend)
fprintf('        I2nd_eigval =  %-1.7g \n',I2nd_eigval_Tstart_Tend)
fprintf('        Ieigvals =     %-1.7g \n',Ieigvals_Tstart_Tend)
fprintf('        Ipredict =     %-1.7g \n',Ipredict_Tstart_Tend)
fprintf('        I_2ndBarthol = %-1.7g \n',I_2ndBarthol_Tstart_Tend)
disp(' ')
disp(' The downward and upward components of the normalized Index are respectively: ')
fprintf(' Inorm_dn_w_v = %-1.7g \n',Inorm_dn_w_V_Tstart_Tend)
fprintf(' Inorm_up_w_v = %-1.7g \n',Inorm_up_w_V_Tstart_Tend)
disp(' Compare to the values of the Downward and Upward Morrison-Bourguignon Indices, ')
disp(' which are respectively: ')
fprintf(' BMorrDN = %-1.7g \n',BMorrDN_Tstart_Tend)
fprintf(' BMorrUP = %-1.7g \n',BMorrUP_Tstart_Tend)
disp(' ')
disp(' The ratio between the upward and the downward components of the Index is: ')
fprintf(' UPvsDN_w_V = %-1.7g \n',UPvsDN_w_V_Tstart_Tend)
fprintf(' and the percentages of agents remaining into each starting state at Tend=%-1.0d are: \n ',Tend)
fprintf('   [ImmobilShare_1,.., ImmobilShare_%-1.0d] = [',classes_number);...
             fprintf('%-1.5g, ',(ImmobilShares_k_Tstart_Tend(1:classes_number-1))');...
             fprintf('%-1.5g] \n',(ImmobilShares_k_Tstart_Tend(classes_number))')
disp(' ')
fprintf(' The current values of parameters {w_i, i = 1,..,%-1.0d}, \n',classes_number)
disp(' computed from the imported data, in this step are: ')
fprintf('   [w_1,.., w_%-1.0d] = [',classes_number);...
             fprintf('%-1.5g, ',(w_k_Lstart_Lend(1:classes_number-1,Tstart,Tend))');...
```

```
997              fprintf('%-1.5g] \n',(w_k_Lstart_Lend(classes_number,Tstart,Tend))')
998 clear Input_6bis Input_7bis Output1 Output_1bis
999 Output_1ter=char({'',' Would you like to see the new transition probability ',...
1000 '            " P_k_h_Tstart_Tend " ',...
1001 ' that a unit moved from the k-th class to the h-th one',...
1002 ' between the new pair (Tstart, Tend) ? ',...
1003 ' (Answer "Y" if ''yes'' or "N" if ''not'', then press [Enter]) '});
1004 disp(Output_1ter)
1005 answer=input('  ','s');
1006 while strcmp(answer,{'Y','y','N','n'})==[0,0,0,0]
1007             disp('   (You must answer exclusively "Y" or "N" !) ')
1008         answer=input('  ','s');
1009 end
1010 if sum(strcmp(answer,{'N','n'}))==1
1011 disp(char({'  Ok: anyway, you can stop this run and quit ','  typing "Ctrl" + "C" on the keyboard. '}))
1012 elseif sum(strcmp(answer,{'Y','y'}))==1
1013 disp('    The new transition probability is: ')
1014 fprintf('    P_k_h_Lstart_Lend(k,h, Tstart = %-1.0d, Tend = %-1.0d)= \n ',Tstart,Tend)
1015 disp(' ')
1016 disp(P_k_h_Tstart_Tend)
1017 end
1018 disp(Output_0bis)
1019 response=input('  ','s');
1020 while strcmp(response,{'Y','y','N','n'})==[0,0,0,0]
1021             disp('   (You must answer exclusively "Y" or "N" !) ')
1022         response=input('  ','s');
1023 end
1024  end
1025 clear abs_eigenvals_Tstart_Tend sorted_abs_eigenvals_Tstart_Tend
1026 Output_1quater=char({'',' Are you interested in knowing, ',...
1027 ' for every possible pair (Lstart,Lend), ',...
1028 ' all the transition matrices ',...
1029 ' and all the values of the Index ? ',...
1030 ' (Answer "Y" if ''yes'' or "N" if ''not'', then press [Enter]) '});
1031 disp(Output_1quater)
1032 resp=input('  ','s');
1033 while strcmp(resp,{'Y','y','N','n'})==[0,0,0,0]
1034             disp('   (You must answer exclusively "Y" or "N" !) ')
1035         resp=input('  ','s');
1036 end
1037 clear Output_zero Output_0bis Output1 Output_1bis Output_1ter Output_1quater
1038 Output2=char({'','',' In the workspace, you could find an array, ',...
1039 ' depending only on (Lstart,Lend) and named: ',...
1040 '        " Inorm_w_V_Lstart_Lend "  ',...
1041 ' where  1 <= Lstart <= dim(2)-1 ',...
1042 ' and    1 <= Lend <= dim(2) . ',' as well as other related outputs. ',...
1043 ' Remember: dim(2) is the number of all observation dates in your data. ',...
1044 ' Logically "Lend" should be such that  Lstart+1 <= Lend <= dim(2) ',...
1045 ' but Matlab tends by default to complete inwards the indefinite positions ',...
1046 ' inside the arrays using zeros, so as they are rectangular anyway. ',...
1047 ' As a consequence, the first column in " Inorm_w_V_Lstart_Lend " ',...
1048 ' corresponds to  " Lstart=1 and Lend=1 " and it is made up of zeros. ',...
1049 ' At the positions for 1 <= Lstart <= dim(2)-1 and Lstart+1 <= Lend <= dim(2) ',...
1050 ' the matrices " Inorm_w_V_Lstart_Lend ", " Inorm_up_w_V_Lstart_Lend ", '...
1051 ' " Inorm_dn_w_V_Lstart_Lend " are composed by the normalized index''s values, ',...
1052 ' the values of its upward part and its downward one. ',...
1053 ' Elsewhere the elements of these matrices are zeros. '});
1054 Output2_2=char({' Fixing "Lstart", you have fixed a row; ',' fixing "Lend", you have fixed a column. ',...
1055 ' When  Lstart > Lend,  it is an impossible case, ',...
1056 ' and if  Lstart = Lend there is not any transition: ',...
1057 ' thus the total mobility associated to both of these cases is zero! ',...
1058 '',' After completing the current run, you can still display ',...
1059 ' anyone of the outputs, linked to the Directional Index, ',...
1060 ' just typing its name into the keyboard in the MATLAB Command Window, ',...
```

```
' for instance " Inorm_w_V_Lstart_Lend ". ',...
'',' Such directional mobility output are listed below : ',...
' the not-normalized directional mobility index " I_w_V_Lstart_Lend " ',...
' along with its upward and downward parts ',' " Iup_w_V_Lstart_Lend " and " Idn_w_V_Lstart_Lend " ',...
' and the upward and downward parts of the normalized index ',...
' " Inorm_up_w_V_Lstart_Lend " and " Inorm_up_w_V_Lstart_Lend "; ',...
' the share of unmoving individuals at class "i" ',...
' over the numerosity of the entire sample, ',' " ImmobilShares_k_Lstart_Lend ", ',...
' the upward versus downward partial mobilities ratio " UPvsDN_w_V_Lstart_Lend " . '});
Output3=char({'',' We have named as: ',...
'                " P_k_h_Lstart_Lend(k,h,Lstart,Lend) " ',...
' the probability that an individual in the sample may make a transition ',...
' from the k-th class at the time "Lstart" to the h-th class at the final period "Lend". '});
Output3_2=char({' After getting out of this tool, it is also possible to visualize, ',...
' for the current session, the transition probabilities'' matrix typing ',...
'                " P_k_h_Lstart_Lend(:,:,Lstart,Lend) " ',...
' for specific couples of values of (Lstart,Lend). '});
P_k_h_Lstart_Lend=zeros(classes_number,classes_number,dim(2)-1,dim(2)-1);
Pup_k_h_Lstart_Lend=zeros(classes_number,classes_number,dim(2)-1,dim(2)-1);
Pdn_k_h_Lstart_Lend=zeros(classes_number,classes_number,dim(2)-1,dim(2)-1);
for Lstart=1:dim(2)-1
for Lend=Lstart+1:dim(2)
for h=1:classes_number
for k=1:classes_number
if num_k_Lstart_Lend(k,Lstart,Lend)==0
continue
end
P_k_h_Lstart_Lend(k,h,Lstart,Lend)=(Number_k_h_Lstart_Lend(k,h,Lstart,Lend)./...
sum(MatData(:,Lstart)>=lowest_datum & MatData(:,Lend)>=lowest_datum)) ...
./(num_k_Lstart_Lend(k,Lstart,Lend)./(sum(MatData(:,Lstart)<=highest_datum & isnan_data(:,Lend)==0)));
%   After fixing “Lstart” and “Lend”, P_k_h_Lstart_Lend is a matrix,
%   'k' is a row index, 'h' is a column index.
end
end
end
end
for Lstart=1:dim(2)-1
for Lend=Lstart+1:dim(2)
  for h=1:classes_number
   for k=1:classes_number
      if k>=h
         continue
      end
     Pup_k_h_Lstart_Lend(k,h,Lstart,Lend)=P_k_h_Lstart_Lend(k,h,Lstart,Lend);
   end
  end
  for h=1:classes_number
   for k=1:classes_number
      if k<=h
         continue
      end
     Pdn_k_h_Lstart_Lend(k,h,Lstart,Lend)=P_k_h_Lstart_Lend(k,h,Lstart,Lend);
   end
  end
end
end
for Lstart=1:dim(2)-1
 for Lend=Lstart+1:dim(2)
  for k=1:classes_number
  Y_V_k_Lstart_Lend(k,Lstart,Lend)=P_k_h_Lstart_Lend(k,:,Lstart,Lend)*V_k_h(:,k);
  Yup_V_k_Lstart_Lend(k,Lstart,Lend)=Pup_k_h_Lstart_Lend(k,:,Lstart,Lend)*V_k_h(:,k);
  Ydn_V_k_Lstart_Lend(k,Lstart,Lend)=Pdn_k_h_Lstart_Lend(k,:,Lstart,Lend)*V_k_h(:,k);
  ImmobilShares_k_Lstart_Lend(k,Lstart,Lend)=Number_k_h_Lstart_Lend(k,k,Lstart,Lend)./num_tot_Lstart_Lend(Lstart,Lend);
  end
```

```
I_w_V_Lstart_Lend(Lstart,Lend)=(w_k_Lstart_Lend(:,Lstart,Lend))'*Y_V_k_Lstart_Lend(:,Lstart,Lend);
Iup_w_V_Lstart_Lend(Lstart,Lend)=(w_k_Lstart_Lend(:,Lstart,Lend))'*Yup_V_k_Lstart_Lend(:,Lstart,Lend);
Idn_w_V_Lstart_Lend(Lstart,Lend)=(w_k_Lstart_Lend(:,Lstart,Lend))'*Ydn_V_k_Lstart_Lend(:,Lstart,Lend);
%
%    The array of the Normalized Directional Mobility Index
%              Inorm_w_V_Lstart_Lend(Lstart,Lend),
%    along with the respective upward and downward components
%              Inorm_up_w_V_Lstart_Lend(Lstart,Lend)
%         and  Inorm_dn_w_V_Lstart_Lend(Lstart,Lend),
%    (where, in all cases, Lstart is a row index and Lend is a column index)
%    are:
%
    if I_w_V_Lstart_Lend(Lstart,Lend)>=0
      Inorm_w_V_Lstart_Lend(Lstart,Lend)=I_w_V_Lstart_Lend(Lstart,Lend)./m2(Lstart,Lend);
      Inorm_up_w_V_Lstart_Lend(Lstart,Lend)=Iup_w_V_Lstart_Lend(Lstart,Lend)./m2(Lstart,Lend);
      Inorm_dn_w_V_Lstart_Lend(Lstart,Lend)=Idn_w_V_Lstart_Lend(Lstart,Lend)./m2(Lstart,Lend);
    elseif I_w_V_Lstart_Lend(Lstart,Lend)<0
      Inorm_w_V_Lstart_Lend(Lstart,Lend)=-I_w_V_Lstart_Lend(Lstart,Lend)./m1(Lstart,Lend);
      Inorm_up_w_V_Lstart_Lend(Lstart,Lend)=-Iup_w_V_Lstart_Lend(Lstart,Lend)./m1(Lstart,Lend);
      Inorm_dn_w_V_Lstart_Lend(Lstart,Lend)=-Idn_w_V_Lstart_Lend(Lstart,Lend)./m1(Lstart,Lend);
    end
    UPvsDN_w_V_Lstart_Lend(Lstart,Lend)=Iup_w_V_Lstart_Lend(Lstart,Lend)./abs(Idn_w_V_Lstart_Lend(Lstart,Lend));
  end
end
if sum(strcmp(resp,{'Y','y'}))==1
disp(Output2)
disp(char({' It is: ','         Inorm_w_V_Lstart_Lend = ',' '}))
disp(Inorm_w_V_Lstart_Lend)
disp('         (Press "Enter" to go on) ')
response=input(' ','s');
while isempty(response)==0
disp('    Please, press just "Enter" to go on.')
response=input(' ','s');
end
disp(Output2_2)
disp('(Press [Enter] to move towards a conclusion...)');
response=input(' ','s');
while isempty(response)==0
disp('(Press [Enter] to move towards a conclusion...)')
response=input(' ','s');
end
disp(Output3)
disp(Output3_2)
clear Output2 Output2_2 Output3 Output3_2 response answer resp
elseif sum(strcmp(resp,{'N','n'}))==1
    disp(Output2)
disp(char({'     DO YOU WANT TO CANCEL IT ? ',...
' (Press "Y" for ''yes'' or "N" for ''not'', then [Enter]) '}))
last_ans=input('  ','s');
while strcmp(last_ans,{'Y','y','N','n'})==[0,0,0,0]
           disp('   (You must answer exclusively "Y" or "N" !) ')
       last_ans=input('  ','s');
end
if sum(strcmp(last_ans,{'Y','y'}))==1
clear Y_V_k_Lstart_Lend Yup_V_k_Lstart_Lend Ydn_V_k_Lstart_Lend
clear I_w_V_Lstart_Lend Iup_w_V_Lstart_Lend Idn_w_V_Lstart_Lend
clear Inorm_w_V_Lstart_Lend Inorm_up_w_V_Lstart_Lend Inorm_dn_w_V_Lstart_Lend
clear ImmobilShares_k_Lstart_Lend UPvsDN_w_V_Lstart_Lend
elseif sum(strcmp(last_ans,{'N','n'}))==1
    disp(Output2_2)
end
    disp(Output3)
disp(char({'     DO YOU WANT TO CANCEL ALL THESE MATRICES ? ',...
' (Press "Y" for ''yes'' or "N" for ''not'', then [Enter]) '}))
```

```
1189 last_ans=input('   ','s');
1190 while strcmp(last_ans,{'Y','y','N','n'})==[0,0,0,0]
1191             disp('    (You must answer exclusively "Y" or "N" !) ')
1192         last_ans=input('   ','s');
1193 end
1194 if sum(strcmp(last_ans,{'Y','y'}))==1
1195     clear P_k_h_Lstart_Lend Pup_k_h_Lstart_Lend Pdn_k_h_Lstart_Lend
1196 elseif sum(strcmp(last_ans,{'N','n'}))==1
1197     disp(' ')
1198     disp(Output3_2)
1199 Output3_3=char({' If  Lstart > Lend,  it is an impossible case, ',...
1200 ' and if  Lstart = Lend there is not any transition, ',...
1201 ' so all the transition probability associated to those cases ',...
1202 ' are zero for each couple of state (k,h)! '});
1203     disp(Output3_3)
1204 end
1205 clear Output2 Output2_2 Output3 Output3_2 Output3_3 answer response resp last_ans
1206 end
1207 clear Output2 Output2_2 Output3 Output3_2 Output3_3 answer response resp last_ans
1208 end
1209 clear Output2 Output2_2 Output3 Output3_2 Output3_3 answer response resp last_ans
1210 Ending1=char({'','' In the MATLAB workspace there are listed all your input parameters  ',...
1211 ' and all the arrays served for the estimation of the Directional Index''s values ',...
1212 ' with the relative transition matrices. ',...
1213 ' To know the meaning of the other auxiliary arrays, like "m1" and "m2", ',...
1214 ' " Number_k_h_Lstart_Lend ", " num_k_Lstart_Lend " or " Y_V_k_Tstart_Tend " ',...
1215 ' you can refer to the text notes included in the script ''DMMtool.m''  ',...
1216 ' or to the related paper by Aringhieri, Ferretti, Ganugi. ','',...
1217 ' If you want to recompute the Directional Mobility Index ',...
1218 ' in case of a different shape of the function " v(|j-i|) ", ',...
1219 ' a different definition of the set of states or of the weights {w_i}, ',...
1220 ' it is necessary to restart ''DMMtool.m'' for a new session in MATLAB. '});
1221 disp(Ending1)
1222 Ending2=char({'','    So this session is going to terminate: ',...
1223     '    you will return directly to the prompt state ',...
1224     '    after pressing [Enter] on your keyboard ! '});
1225 disp(Ending2)
1226 response=input(' ','s');
1227 while isempty(response)==0
1228 disp('(press [Enter] to exit from ''DMMtool.m'')')
1229 response=input(' ','s');
1230 end
1231 clear Ending1 Ending2 k h Tstart Tend Lstart Lend response resp_w
1232 return
```