



دانشگاه صنعتی خواجه نصیرالدین طوسی
دانشکده مهندسی برق - گروه مهندسی کنترل

تمرین درس محاسبات نرم
در رشته مهندسی برق گرایش مهندسی کنترل

عنوان

تمرین کلاسی دوم: بازسازی اعداد از روی اعداد
مخدوش یا نویزی شده (مثال هگان)

نگارش

محمدحسین احمدی

استاد درس

دکتر مهدی علیاری شوره‌دلی

آذرماه ۱۴۰۱

فهرست مطالب

ب

فهرست شکل‌ها

۱

پاسخ‌سوال‌ات

۷

پیوست

فهرست شکل‌ها

۵ (Pseudoinverse) بازسازی اعداد از روی قسمتی از آن‌ها	۱
۵ (Hebb) بازسازی اعداد از روی قسمتی از آن‌ها	۲
۵ (Pseudoinverse) بازسازی اعداد از روی نویزی شده آن‌ها	۳
۶ (Hebb) بازسازی اعداد از روی نویزی شده آن‌ها	۴

پاسخ سوالات

نکات قابل ذکر

کدهای متلب مربوط به این سوال از داخل پوشه Codes قابل اجراست. همچنین می‌توان از دستور nnd7sh در متلب و کدمنبع آورده‌شده در فهرست شکل‌ها برای شبیه‌سازی این سوال استفاده کرد.

برای حل این سوال از دستورات زیر که در پوشه مربوط به کدها نیز آمده است استفاده می‌کنیم:

```
1 % Autoassociative network for digit recognition using Hebbian learning
2 clc;
3 clear all;
4 close all;
5
6 %% load training images(5x7 px)
7 M=7; N=5;
8 P=zeros(M*N,10);
9 for n=0:9
10     s=num2str(n);
11     RGB=imread(s,'png');
12     m=n+1;
13     P(:,m)=reshape(rgb2gray(RGB),[M*N,1]); %column prototype vector
14 end
15 P = P/255*2-1; %normalizes data to be either -1 or 1
16
17 %% compute weight matrix using Hebb rule
18 W=zeros(M*N,M*N);
19 for n=1:10
20     W=W+P(:,n)*P(:,n)';
```

```
21 end
22
23 % simple Hebb rule did not make it recognize images well
24 % try again using pseudoinverse
25 T=P;
26 W1=T*pinv(P);
27
28 %% test out network with training images
29 for n = 10:-1:1
30     a=hardlims(W*P(:,n));
31     outputImg=reshape(a,[M,N]);
32     figure;
33     imshow(outputImg,'InitialMagnification','Fit')
34 end
35
36 %% test out network with noisy inputs
37 P_noisy = P + randi([-1,1],M*N,10);
38
39 for n = 10:-1:1
40     outputImg1=reshape(P_noisy(:,n),[M,N]);
41     figure;
42     subplot(1,2,1)
43     imshow(outputImg1,'InitialMagnification','Fit')
44     title('noisy image (Hebb)')
45
46     subplot(1,2,2)
47     a=hardlims(W*P_noisy(:,n));
48     outputImg2=reshape(a,[M,N]);
49     imshow(outputImg2,'InitialMagnification','Fit')
50     title('reconstructed image (Hebb)')
51 end
52
53 for n = 10:-1:1
54     outputImg1=reshape(P_noisy(:,n),[M,N]);
55     figure;
```

```

56     subplot(1,2,1)
57     imshow(outputImg1,'InitialMagnification','Fit')
58     title('noisy image (Pseudoinverse)')
59
60     subplot(1,2,2)
61     a=hardlims(W1*P_noisy(:,n));
62     outputImg2=reshape(a,[M,N]);
63     imshow(outputImg2,'InitialMagnification','Fit')
64     title('reconstructed image (Pseudoinverse)')
65 end
66
67 %% test out network with parts of image missing
68 P_partial=P;
69 P_partial(8:17,:)=1;
70
71 for n = 10:-1:1
72     outputImg1=reshape(P_partial(:,n),[M,N]);
73     figure;
74     subplot(1,2,1)
75     imshow(outputImg1,'InitialMagnification','Fit')
76     title('partial image (Hebb)')
77
78     subplot(1,2,2)
79     a=hardlims(W*P_partial(:,n));
80     outputImg2=reshape(a,[M,N]);
81     imshow(outputImg2,'InitialMagnification','Fit')
82     title('reconstructed image (Hebb)')
83 end
84
85 P_partial=P;
86 P_partial(8:17,:)=1;
87
88 for n = 10:-1:1
89     outputImg1=reshape(P_partial(:,n),[M,N]);
90     figure;

```

```

91 subplot(1,2,1)
92 imshow(outputImg1,'InitialMagnification','Fit')
93 title('partial image (Pseudoinverse)')
94
95 subplot(1,2,2)
96 a=hardlims(W1*P_partial(:,n));
97 outputImg2=reshape(a,[M,N]);
98 imshow(outputImg2,'InitialMagnification','Fit')
99 title('reconstructed image (Pseudoinverse)')
100 end

```

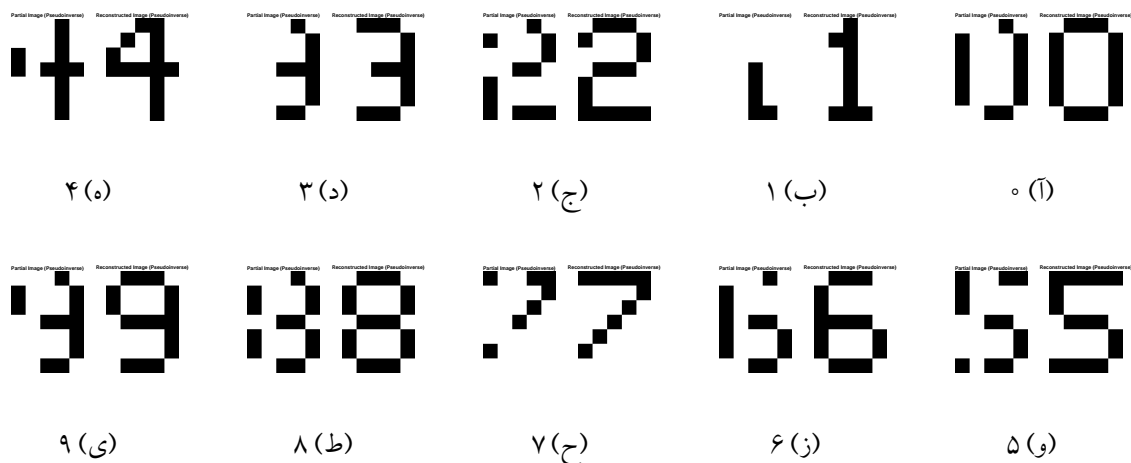
تابع `hardlims` هم که در کد اصلی بالا تعریف شده است به صورت زیر تعریف می گردد:

```

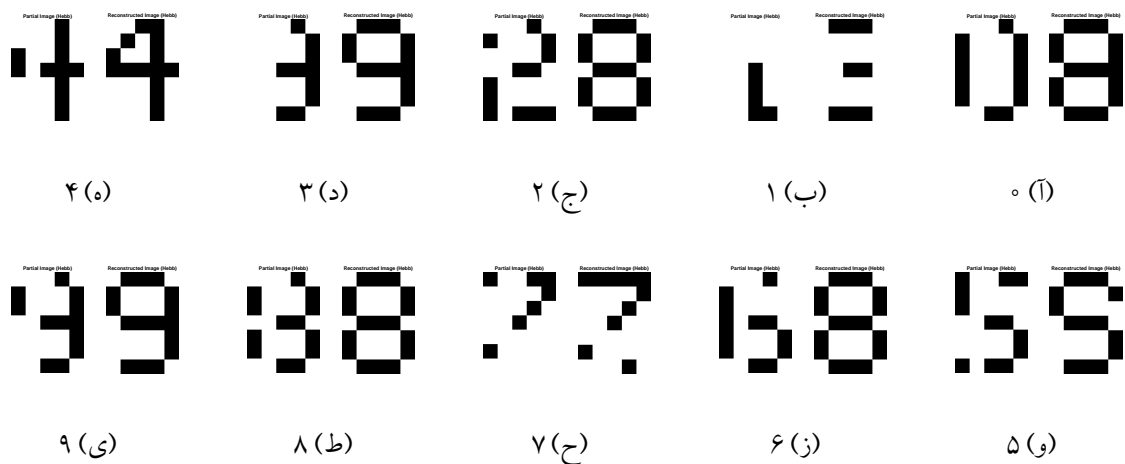
1 function [ a ] = hardlims( n )
2 %   hardlims Symmetric Hard Limit transfer function
3 %   Accepts column vector. Returns column vector.
4 %   behaves like unit step of amplitude 2 and vertical offset of -1
5 %   a = -1 for n < 0
6 %   a =  1 for n >= 0
7   dims = size(n);
8   a = zeros(dims);
9   for i = 1:dims(1)
10      for j = 1:dims(2)
11          if n(i,j) < 0
12              a(i,j) = -1;
13          else
14              a(i,j) = 1;
15          end
16      end
17  end
18 end

```

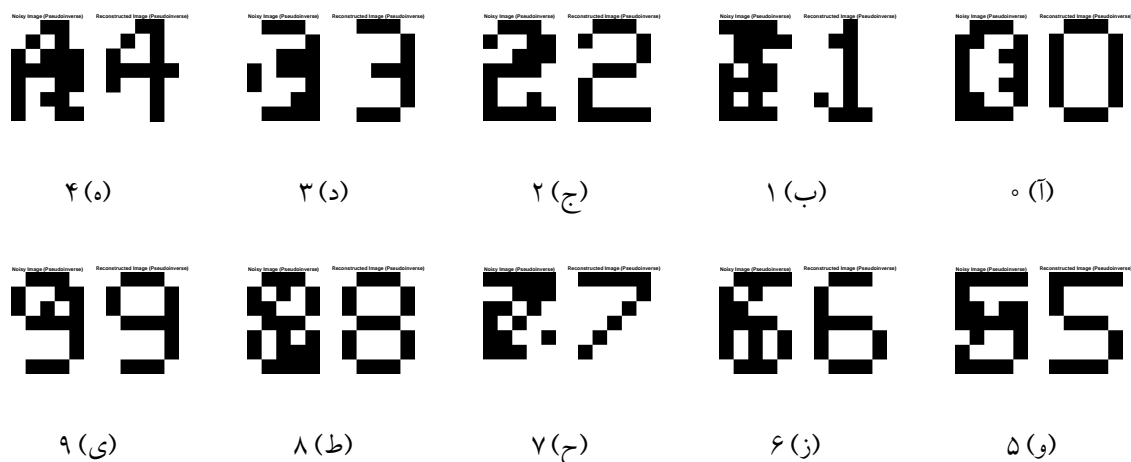
با اجرای این دستورات نتایج به صورتی که در ادامه آورده شده است خواهد بود:



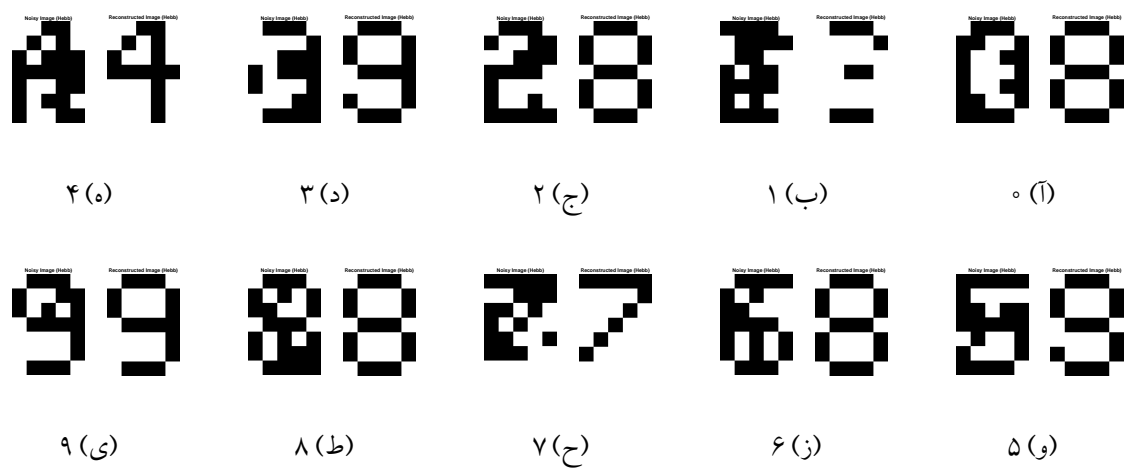
شکل ۱: بازسازی اعداد از روی قسمتی از آن‌ها (Pseudoinverse).



شکل ۲: بازسازی اعداد از روی قسمتی از آن‌ها (Hebb).



شکل ۳: بازسازی اعداد از روی نویزی شده آن‌ها (Pseudoinverse).



شکل ۴: بازسازی اعداد از روی نویزی شده آن‌ها (Hebb).

پیوست

کد منبع nnd7sh هم به صورت زیر تعریف می گردد:

```
1 function nnd7sh(cmd,arg1,arg2,arg3)
2 %NND7SH Supervised Hebb demonstration.
3
4 % Copyright 1994-2002 PWS Publishing Company and The MathWorks, Inc.
5 % $Revision: 1.6 $
6
7 %=====
8
9 % CONSTANTS
10 me = 'nnd7sh';
11 p_x = 5; % pattern horizontal size
12 p_y = 6; % pattern vertical size
13
14 % DEFAULTS
15 if nargin == 0, cmd = ''; else cmd = lower(cmd); end
16
17 % FIND WINDOW IF IT EXISTS
18 fig = nndfgflg(me);
19 if length(get(fig,'children')) == 0, fig = 0; end
20
21 % GET WINDOW DATA IF IT EXISTS
22 if fig
23     H = get(fig,'userdata');
24     fig_axis = H(1);           % window axis
25     desc_text = H(2);          % handle to first line of text sequence
```

```

26 pattern = H(3:5);           % pattern axes 1-3
27 tp_axis = H(6);            % test pattern
28 rp_axis = H(7);            % response pattern
29 P_ptr = H(8);               % handle to pattern matrix P
30 p_ptr = H(9);               % handle to test pattern
31 w_ptr = H(10);              % handle to weight matrix W
32 rule1 = H(11);              % handle to first radio button
33 rule2 = H(12);              % handle to second radio button
34 end
35
36 %=====
37 % Activate the window.
38 %
39 % ME() or ME('')
40 %=====
41
42 if strcmp(cmd, '')
43     if fig
44         figure(fig)
45         set(fig, 'visible', 'on')
46     else
47         feval(me, 'init')
48     end
49
50 %=====
51 % Close the window.
52 %
53 % ME() or ME('')
54 %=====
55
56 elseif strcmp(cmd, 'close') & (fig)
57     delete(fig)
58
59 %=====
60 % Initialize the window.

```

```

61 %
62 % ME('init')
63 %=====
64
65 elseif strcmp(cmd,'init') & (~fig)
66
67 % STANDARD DEMO FIGURE
68 fig = nndemof2(me,'DESIGN','Supervised Hebb','', 'Chapter 7');
69 set(fig, ...
70     'windowbuttondownfcn',nncallbk(me,'down'), ...
71     'BackingStore','off',...
72     'nextplot','add');
73 H = get(fig,'userdata');
74 fig_axis = H(1);
75 desc_text = H(2);
76
77 % ICON
78 nndicon(7,458,363,'shadow')
79
80 % ORIGINAL PATTERNS
81 p1 = [0 1 1 1 1 0 ...
82       1 0 0 0 0 1 ...
83       1 0 0 0 0 1 ...
84       1 0 0 0 0 1 ...
85       0 1 1 1 1 0]';
86 p2 = [0 0 0 0 0 0 ...
87       1 0 0 0 0 0 ...
88       1 1 1 1 1 1 ...
89       0 0 0 0 0 0 ...
90       0 0 0 0 0 0]';
91 p3 = [1 0 0 0 0 0 ...
92       1 0 0 1 1 1 ...
93       1 0 0 1 0 1 ...
94       1 0 0 1 0 1 ...
95       0 1 1 0 0 1]';

```

```

96  P = [p1 p2 p3]*2-1;
97  p  = p1*2-1;
98
99  % WEIGHTS & OUTPUTS
100 w = P*P';
101 a = w*p;
102
103 % PATTERN AXES
104 pattern = zeros(1,5);
105 ltyell = nnltyell;
106 for k=1:5
107     if k < 4
108         title = sprintf('Pattern %g',k);
109         pos = [25+115*(k-1) 230 100 100];
110         pp = reshape(P(:,k),p_y,p_x);
111         color = nngreen;
112     elseif k == 4
113         title = 'Test Pattern';
114         pos = [25 20 160 160];
115         pp = reshape(p,p_y,p_x);
116         color = nndkgray;
117     else
118         title = 'Response Pattern';
119         pos = [195 20 160 160];
120         pp = reshape(a,p_y,p_x);
121         color = nnred;
122     end
123     pattern(k) = nnsfo('a2',title,'','');
124     set(pattern(k), ...
125         'units','points',...
126         'position',pos,...
127         'color',nnltyell,...
128         'xlim',[0 p_x], ...
129         'ylim',[0 p_y],...
130         'ydir','reverse')

```

```
131     axis('off')
132     pattern_h = zeros(p_y,p_x);
133     box_x = [0 1 1 0 0];
134     box_y = [0 0 1 1 0];
135     for i=1:p_x, for j=1:p_y
136         if pp(j,i) >= 0
137             pattern_h(i,j) = fill(box_x+i-1,box_y+j-1,color,...
138                 'edgecolor',nndkblue,...
139                 'erasemode','none');
140         else
141             pattern_h(i,j) = fill(box_x+i-1,box_y+j-1,ltyell,...
142                 'edgecolor',nndkblue,...
143                 'erasemode','none');
144         end
145     end, end
146     set(pattern(k),'userdata',pattern_h);
147 end
148
149 % WEIGHT RULE BUTTONS
150 drawnow % Let everything else appear before buttons
151 rule1 = uicontrol(...
152     'units','points',...
153     'position',[395 190 70 20],...
154     'style','radio',...
155     'string','Hebb',...
156     'backg',nnltgray,...
157     'callback',[me '('rule'',1)'],...
158     'value',1);
159 rule2 = uicontrol(...
160     'units','points',...
161     'position',[395 170 90 20],...
162     'style','radio',...
163     'string','Psuedoinverse',...
164     'backg',nnltgray,...
165     'max',[2],...
```

```
166     'callback',[me '('rule'',2)'];
167
168 % BUTTONS
169 if (exist('hintonw'))
170     uicontrol(...
171         'units','points',...
172         'position',[410 140 60 20],...
173         'string','Weights',...
174         'callback',[me '('weights'')'])
175 end
176 uicontrol(...
177     'units','points',...
178     'position',[410 110 60 20],...
179     'string','Contents',...
180     'callback','nndtoc')
181 uicontrol(...
182     'units','points',...
183     'position',[410 80 60 20],...
184     'string','Close',...
185     'callback','delete(gcf)')
186
187 % DATA POINTERS
188 P_ptr = nnsfo('data'); set(P_ptr,'userdata',P);
189 p_ptr = nnsfo('data'); set(p_ptr,'userdata',p);
190 w_ptr = nnsfo('data'); set(w_ptr,'userdata',w);
191
192 % SAVE WINDOW DATA AND LOCK
193 H = [fig_axis desc_text pattern P_ptr p_ptr w_ptr rule1 rule2];
194 set(fig,'userdata',H,'nextplot','new','color',nnltgray)
195
196 % INSTRUCTION TEXT
197 feval(me,'instr');
198
199 nnchkfs;
200
```

```
201 %=====
202 % Display the instructions.
203 %
204 % ME('instr')
205 %=====
206
207 elseif strcmp(cmd,'instr') & (fig)
208     nnsettxt(desc_text,...
209         'Click on the green',...
210         'grids to define target.',...
211         'patterns. Click on the',...
212         'gray grid to define',...
213         'a test pattern.',...
214         '',...
215         'Select the rule to',...
216         'calculate the network',...
217         'weights below:')
218
219 %=====
220 % Show weights.
221 %
222 % ME('weights')
223 %=====
224
225 elseif strcmp(cmd,'weights') & (fig) & (margin == 1)
226
227     % GET DATA
228     w = get(w_ptr,'userdata');
229
230     f = figure;
231     feval('hintonw',w);
232     axis('equal');
233     set(f,'name','Network Weights')
234     t = get(gca,'title');
235     set(t,'string','Green = Positive, Red = Negative')
```



```
236
237 %=====
238 % Respond to mouse down.
239 %
240 % ME('down')
241 %=====
242
243 elseif strcmp(cmd,'down') & (fig) & (nargin == 1)
244
245     set(fig,'nextplot','add')
246     for i=1:3
247         [in,x,y] = nnaxclick(pattern(i));
248         if in
249             feval(me,'down',x,y,i);
250             break
251         end
252     end
253     [in,x,y] = nnaxclick(tp_axis);
254     if in
255         feval(me,'down',x,y);
256     end
257     set(fig,'nextplot','new')
258
259 %=====
260 % Respond to mouse down in pattern 1-3.
261 %
262 % ME('down',x,y,i)
263 %=====
264
265 elseif strcmp(cmd,'down') & (fig) & (nargin == 4)
266
267     % GET DATA
268     x = floor(arg1)+1;
269     y = floor(arg2)+1;
270     i = arg3;
```

```

271 green = nngreen;
272 ltyell = nnltyell;
273 red = nnred;
274 P = get(P_ptr, 'userdata');
275 p = get(p_ptr, 'userdata');
276 squares = get(pattern(i), 'userdata');
277 rp_squares = get(rp_axis, 'userdata');
278 rule = get(rule2, 'value')+1;
279
280 % TOGGLE SQUARE
281 ind = (x-1)*p_y+y;
282 P(ind,i) = -P(ind,i);
283 if P(ind,i) > 0
284     set(squares(x,y), 'facecolor', green);
285 else
286     set(squares(x,y), 'facecolor', ltyell);
287 end
288 drawnow
289
290 % UPDATE WEIGHTS
291 if rule == 1
292     w = P*P';
293 else
294     w = P*inv(P'*P)*P';
295 end
296
297 % STORE DATA
298 set(P_ptr, 'userdata', P);
299 set(w_ptr, 'userdata', w);
300
301 % UPDATE OUTPUTS
302 feval(me, 'update')
303
304 %=====
305 % Respond to mouse down in test pattern.

```

```
306 %
307 % ME('down',x,y)
308 %=====
309
310 elseif strcmp(cmd,'down') & (fig) & (nargin == 3)
311
312     % GET DATA
313     x = floor(arg1)+1;
314     y = floor(arg2)+1;
315     dkgray = nndkgray;
316     ltyell = nnltyell;
317     tp_squares = get(tp_axis,'userdata');
318     rp_squares = get(rp_axis,'userdata');
319     p = get(p_ptr,'userdata');
320     w = get(w_ptr,'userdata');
321
322     % TOGGLE SQUARE
323     ind = (x-1)*p_y+y;
324     p(ind) = -p(ind);
325     if p(ind) > 0
326         set(tp_squares(x,y),'facecolor',dkgray);
327     else
328         set(tp_squares(x,y),'facecolor',ltyell);
329     end
330     drawnow
331
332     % STORE DATA
333     set(p_ptr,'userdata',p);
334
335     % UPDATE OUTPUTS
336     feval(me,'update')
337
338 %=====
339 % Set weight rule.
340 %
```

```
341 % ME('rule',i)
342 %=====
343
344 elseif strcmp(cmd,'rule') & (fig) & (nargin == 2)
345
346 % SET RADIO BUTTONS
347 if arg1 == 1
348     set(rule1,'value',1)
349     set(rule2,'value',0)
350 else
351     set(rule1,'value',0)
352     set(rule2,'value',2)
353 end
354
355 % GET DATA
356 P = get(P_ptr,'userdata');
357
358 % UPDATE WEIGHTS
359 if arg1 == 1
360     w = P*P';
361 else
362     w = P*inv(P'*P)*P';
363 end
364
365 % STORE DATA
366 set(w_ptr,'userdata',w)
367
368 % UPDATE OUTPUTS
369 feval(me,'update')
370
371 %=====
372 % Update response pattern.
373 %
374 % ME('update')
375 %=====
```

```

376
377 elseif strcmp(cmd,'update') & (fig)
378
379     % GET DATA
380     red = nnred;
381     ltyell = nnltyell;
382     p = get(p_ptr,'userdata');
383     w = get(w_ptr,'userdata');
384     rp_squares = get(rp_axis,'userdata');
385
386     % UPDATE OUTPUTS
387     a = w*p;
388     a = reshape(a,p_y,p_x);
389     for i=1:p_x, for j=1:p_y
390         if a(j,i) > 0
391             set(rp_squares(i,j),'facecolor',red)
392         else
393             set(rp_squares(i,j),'facecolor',ltyell)
394         end
395     end, end
396
397     %=====
398 end

```