Fine soccer

Morio Kikuchi

Abstract：
We modify the present rule of soccer in order to be able to get points of plural kinds referring to the point system of rugby.

1. Rugby

Because mean score is about 2 or 3 points in soccer, we must make the point system fine for a player to show mobility fully keeping its motivation. We try inserting a point system into a game in which if a player approaches opponents' goal or shoots frequently, light points are given suitably even if a goal is not got.

In rugby, try, conversion goal and penalty goal are 5, 2 and 3 points respectively. Conversion goal is goal by kick after try and penalty goal is goal by penalty kick. Because we make a point system in which small points of plural kinds like points of conversion goal and penalty goal are got in the form like free throw in basketball, usual goal is made 7 points being made to correspond to try.

2. Corner Kick

If a ball cuts across goal line touching a defensive player, Corner Kick(CK) is given. In this case, we insert a system so that free kick(LK, MK) like free throw in basketball can be done selectively. An offensive player shows referee which of CK, LK(MK) to choose. The following are the differences between LK and MK.

・LK：If a ball cuts across goal line touching goal keeper(GK)
・MK：If a ball cuts across goal line touching field player(FP)

The range of goal line which gives LK, MK is set like the following:

・LK：full-line(Figure 1, left)
・MK：full-line(Figure 1, left), semiline(Figure 1, right) ("alternatives")

In MK, we have not got the most suitable range yet. So, full-line and semiline are shown in Figure 1 as experimental alternatives "alternatives". We must decide which to adopt before beginning of a game previously. After this, "alternatives" is used in the meaning like this.

The range in which the ball to kick must not be placed is set like the following:

・LK：penalty area(Figure 2)
・MK：penalty area(Figure 2),
　　penalty area+penalty arc(Figure 3)
　　("alternatives")

The ball must not be placed on the white line in order to protect it.

The range(excluded zone) in which players except the player to kick the ball and the defensive GK must not be located is set like the following:

・LK：penalty area+circle(centre:location of the ball, radius:9.15m)(Figure 4)
・MK：penalty area+circle(centre:location of the ball, radius:9.15m)(Figure 4),
penalty area+penalty arc+circle(centre:location of the ball, radius:9.15m)(Figure 5)
(”alternatives”)

On MK, the two ranges are not independent but Figure 2 and 3 correspond to Figure 4 and 5 respectively.

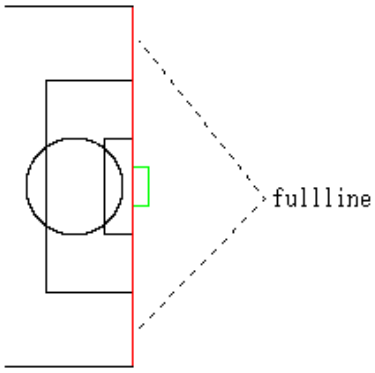
The player to kick the ball can be chosen freely out of 11 players(GK+FP). However, it will be chosen out of FP in fact.

The position of the defensive GK when the kick is done is set like the following:

・on goal line, in goal area (”alternatives”)

On the violations in LK, MK, we adopt judgements on penalty kick. However, if there is a marked delay action, there is a possibility that a judgement that LK, MK are upgraded to penalty kick or recognized to be failure is added.

After the kick, only the defensive GK can touch the ball and the game is continued from goal kick by the defense regardless of success or failure of goal.



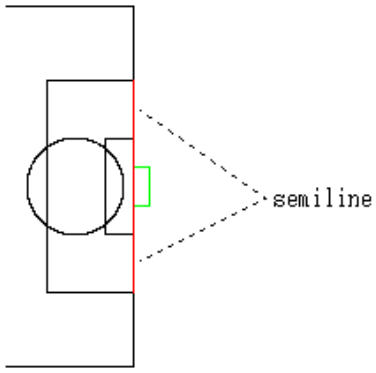


Figure 1

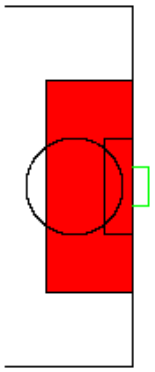


Figure 2

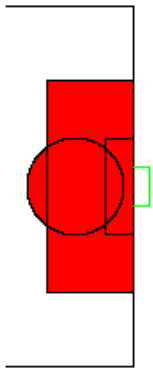


Figure 3

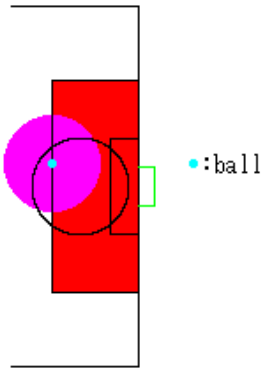


Figure 4

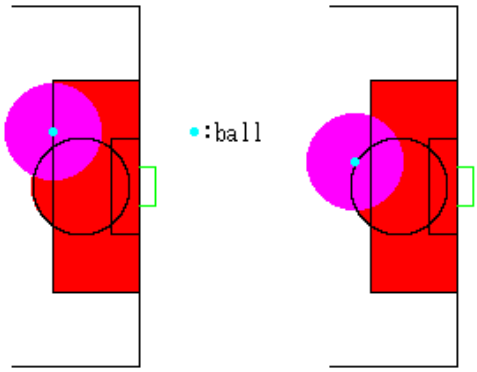


Figure 5

3. Hand signal on LK, MK

If CK occurs, referee specifies it by hand signal. If LK, MK are introduced, referee needs to specify LK, MK making a distinction between them.

Figure 6 is hand signal if a ball cuts across goal line(full-line) touching GK. Figure 7 is hand signal if a ball cuts across goal line(full-line or semiline) touching FP. In either case, referee points at corner area with a horizontal arm. Semiline being adopted and if a ball cuts across goal line which is not semiline, because there is no choice of CK or MK and there is only CK, referee uses usual hand signal.

An offensive player points at corner area if CK and expresses 'L' with arms like Figure 8 if LK(MK) to referee.

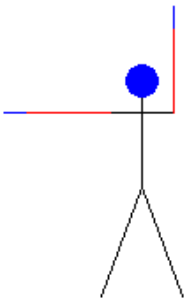


Figure 6

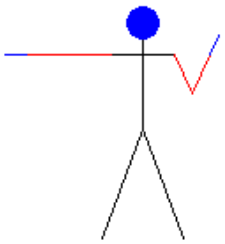


Figure 7

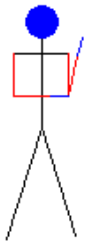


Figure 8

4. Point size of a goal by LK, MK

For example, point size of a goal by LK, MK is set like the following:

・LK：point size=3, point size=2 ("alternatives")
・MK：point size=1, point size=0.5 ("alternatives")

In the above examples,there can be 3 combinations, LK:point size=3, MK:point size=1;LK:point size=3, MK:point size=0.5;LKpoint size=2, MK:point size=0.5. If the probability of goal by CK is 0.04, the expectation per CK becomes 0.04×7=0.28. Assuming that MK's point size is 1 and the probability of goal by MK is 0.5, the expectation per MK becomes 0.5×1=0.5. If we want to make the two expectations about the same degree, we need to try changing "alternatives" which we adopt on MK.

Because LK has the positioning that if a ball cuts across goal line touching GK, we give the chance in which points which is next to points of usual goal can be got, its points is high relatively(1/4〜1/2 of points of usual goal). On the other, MK has the positioning that if a ball cuts across goal line touching FP, we permit early doing of a kind of penalty shoot-out. Therefore, we set MK as a score by for example, 95% point of a statistical distribution of MK goals becomes less than 7 points of usual goal.

5. Statistics of draw(soccer)

We consider to what extent for rate of draw to become using statistical knowledge and numerical calculation. It is assumed that the mean number of usual goals and the mean number of CKs per game are $\lambda_a = 2.6$, n=10 respectively. If CKs are replaced with LK, MK all(it is assumed that LK is "on goal line", MK is Figure 1 left, Figure 3 and "on goal line"), the mean number of usual goals per game becomes $\lambda_a = 2.6 - 10 \times 0.04 = 2.6 - 0.4$. The distribution of the goals is represented by Poisson distribution. It is assumed that the LK's share is 3 and the MK's share is 7 in the breakdown of n=10. Assuming that the goal probability of LK and the goal probability of MK are 0.6 and 0.5 respectively, the mean number of LK goals and the mean number of MK goals per game are

・LK goals：$\lambda_b = 3 \times 0.6 = 1.8$
・MK goals：$\lambda_c = 7 \times 0.5 = 3.5$

These distributions of the goals is represented by Poisson distribution too. Figure 9, 10, 11 showing them, the 99.9999% points are $h = 12,\ i = 11,\ j = 15$ respectively. If these events are independent, the sum event of these events is represented by Poisson distribution too. Figure 12 showing the distribution of this sum event $\lambda_a + \lambda_b + \lambda_c = 7.5$, the 99.99% point is total $t = 20$.

In Figure 12, the horizontal axis and vertical axis represent the number of events and its rate respectively. t is the number of usual goals+LK goals+MK goals. Because rate of draw is the sum of products of the rate on each score which is the sum of products of each number of goals and each points, the distribution in Figure 12 does not give rate of draw.

We try superimposing the distributions of LK goals and MK goals on the distribution of usual goals. If the distribution of LK goals in Figure 10 is superimposed on for example, $h = 2$ of the distribution of usual goals in Figure 9, $h = 2$ devides into 12 like Figure 13. It means that the number of events exists in the superimposed rate(the product of two rates) from $c = 2 + 0$ to $c = 2 + 11$ on $h = 2$ in Figure 9. Next, if the distribution of MK goals in Figure 11 is superimposed on for example, $c = 2 + 3$ in Figure 13, $c = 2 + 3$ devides into 16 like Figure 14. It means that the number of events exists in the superimposed rate(the product of three rates) from $c = 2 + 3 + 0$ to $c = 2 + 3 + 15$ on $c = 2 + 3$ in Figure 13.

If the above steps are done on all the cases, the sum of superimposed rates is calculated on the same number of events and they are illustrated, they become like Figure 15. We compare the figure and Figure 12 with numerical value.

Fig.15　Fig.12

```
t= 0  0.000553  0.000553
t= 1  0.004148  0.004148
t= 2  0.015555  0.015555
t= 3  0.038889  0.038889
t= 4  0.072916  0.072916
t= 5  0.109375  0.109375
t= 6  0.136718  0.136718
t= 7  0.146484  0.146484
t= 8  0.137329  0.137329
t= 9  0.114440  0.114440
t=10  0.085830  0.085830
t=11  0.058521  0.058521
t=12  0.036575  0.036575
t=13  0.021101  0.021101
t=14  0.011304  0.011304
t=15  0.005652  0.005652
t=16  0.002649  0.002649
t=17  0.001169  0.001169
t=18  0.000487  0.000487
t=19  0.000192  0.000192
t=20  0.000072  0.000072
```

They agree almost. Therefore, we will be able to say that there is no problem especially.

The score in the case that the points of LK goals and the points of MK goals are 3 and 1 respectively and the number of events is $h,\ i,\ j (c = h + i + j)$ is $s = 7 \times h + 3 \times i + 1 \times j$. Generally, a score is represented by plural sets of $h,\ i,\ j$. In the case, the rate at which the score is realized is the sum of superimposed rates of each set(for example, $c = 2 + 3 + 4 : 0.268144 \times 0.160671 \times 0.188812$ in the case $h = 2,\ i = 3,\ j = 4$ in Figure 14). The following shows by how many sets of $h,\ i,\ j$ for a score to be represented. The left, the centre and the right are a score, the number of sets of $h,\ i,\ j$ and the rate at which the score is realized respectively.

```
 0   1  0.000553
 1   1  0.001936
 2   1  0.003388
 3   2  0.004948
 4   2  0.006943
 5   2  0.008519
 6   3  0.009422
 7   4  0.011284
 8   4  0.014413
 9   5  0.017055
10   6  0.019682
11   6  0.023058
12   7  0.025332
13   8  0.026157
14   9  0.027845
15  10  0.030414
```

16	11	0.031897
17	12	0.033241
18	12	0.035315
19	13	0.036103
20	14	0.035434
21	15	0.035406
22	16	0.035784
23	17	0.035122
24	18	0.034418
25	18	0.034302
26	19	0.033296
27	20	0.031411
28	21	0.030015
29	22	0.028825
30	23	0.027046
31	24	0.025380
32	24	0.024148
33	25	0.022525
34	26	0.020577
35	27	0.018990
36	28	0.017556
37	29	0.015921
38	30	0.014451
39	30	0.013263
40	31	0.011982
41	32	0.010655
42	32	0.009556
43	33	0.008570
44	34	0.007562
45	34	0.006680
46	34	0.005955
47	35	0.005238
48	35	0.004552
49	35	0.003985
50	36	0.003484
51	36	0.003004
52	36	0.002594
53	36	0.002257
54	36	0.001941
55	36	0.001653
56	36	0.001417
57	36	0.001212
58	36	0.001025
59	36	0.000868
60	36	0.000739
61	36	0.000623

```
62  36  0.000521
63  36  0.000439
64  36  0.000368
65  36  0.000306
66  36  0.000255
67  36  0.000213
68  36  0.000176
69  36  0.000145
70  36  0.000120
71  36  0.000099
72  36  0.000081
73  36  0.000066
74  36  0.000055
75  36  0.000045
76  36  0.000036
77  36  0.000030
78  36  0.000024
```

99.99% point is score $s = 78$. For example, on score $s = 7$ in the list, the number of sets of $h,\ i,\ j$ is 4 and the rate at which the score is realized is 0.011284. We get the rate like the following:

$$h = 0,\ \ i = 0,\ \ j = 7$$
$$s = 7 \times h + 3 \times i + 1 \times j = 7$$
$$c = 0 + 0 + 7 : 0.110803 \times 0.165299 \times 0.038549 = 0.000706$$

$$h = 0,\ \ i = 1,\ \ j = 4$$
$$s = 7 \times h + 3 \times i + 1 \times j = 7$$
$$c = 0 + 1 + 4 : 0.110803 \times 0.297538 \times 0.188812 = 0.006225$$

$$h = 0,\ \ i = 2,\ \ j = 1$$
$$s = 7 \times h + 3 \times i + 1 \times j = 7$$
$$c = 0 + 2 + 1 : 0.110803 \times 0.267784 \times 0.105691 = 0.003136$$

$$h = 1,\ \ i = 0,\ \ j = 0$$
$$s = 7 \times h + 3 \times i + 1 \times j = 7$$
$$c = 1 + 0 + 0 : 0.243767 \times 0.165299 \times 0.030197 = 0.001217$$

$$0.000706 + 0.006225 + 0.003136 + 0.001217 = 0.011284$$

Because the rate on the score axis was got in this way, we only get the sum of products of the rate after. Actually, we get the sum of products between the rate on the score axis of a team and the rate on the score axis of another team. However, here, assuming that the two teams is the same in the rate on the score axis, the sum of products is got. The result is

・rate of draw：0.025572

Mean score, modal score and variance are

- mean score : 24.299902
- modal score : 19 (rate=0.036103)
- variance : 127.494322

However, the calculating upper limits of superimposing in Figure 13, 14 are made a little bigger than 99.9999% point in Figure 10, 11.

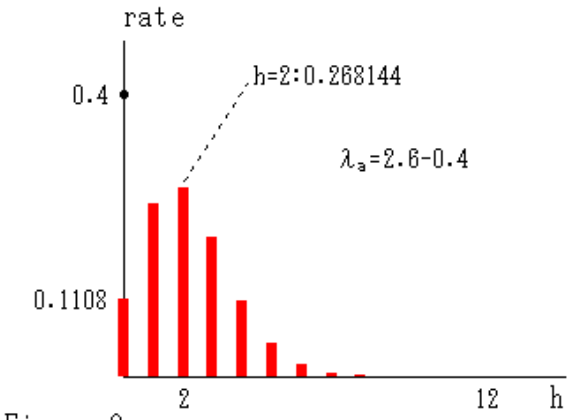


Figure 9

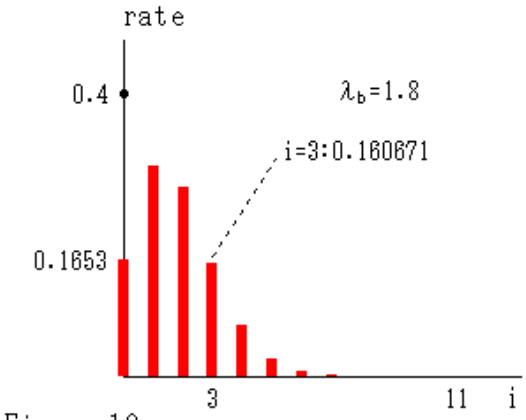


Figure 10

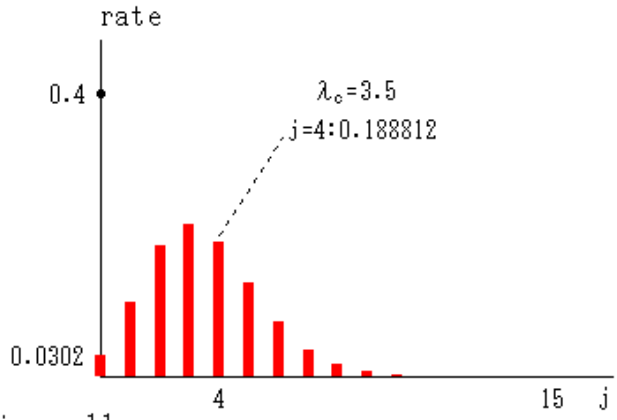


Figure 11

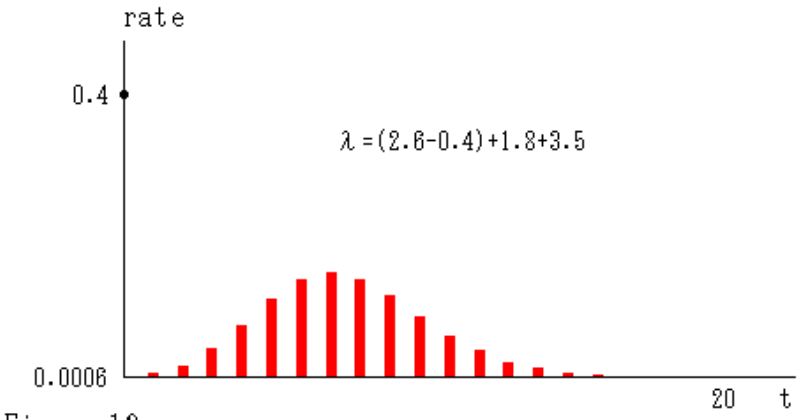


Figure 12

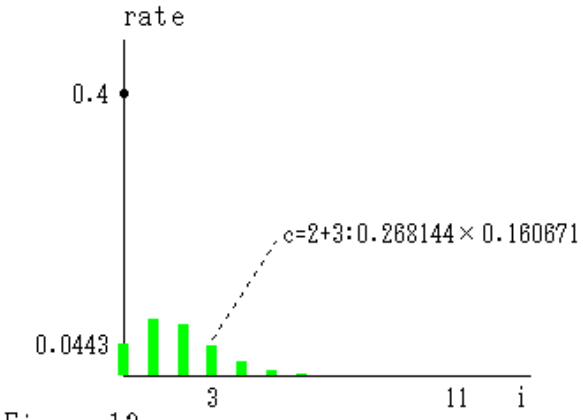


Figure 13

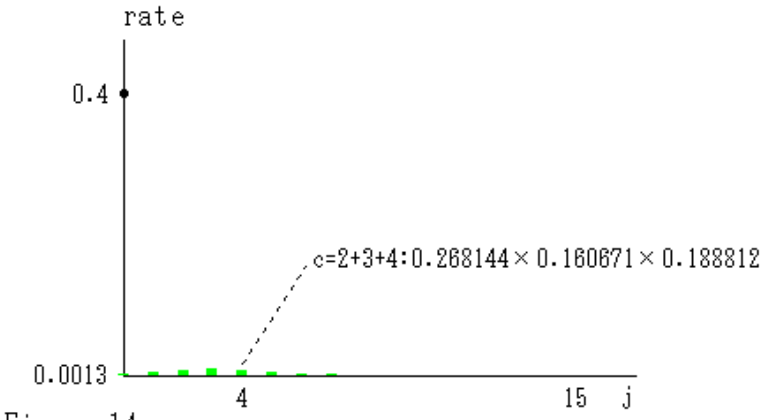


Figure 14

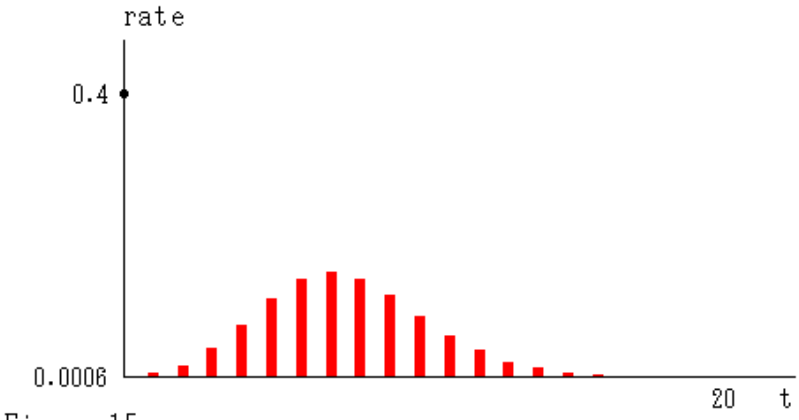


Figure 15

6. Statistics of draw(rugby)

We can get rate of draw in almost the same way as soccer in rugby too. For the sake of ease, points by drop kick is ignored. The following is a comparison of my own style between rugby and soccer.

- try(5 points) $\Longleftrightarrow$ usual goal(7 points)
- conversion goal(2 points) $\Longleftrightarrow$ LK goal(3 points)
- penalty goal(3 points) $\Longleftrightarrow$ MK goal(1 point)

Try and penalty goal are represented by Poisson distribution and it is assumed that their events are independent. Conversion goal is goal by conversion kick and because the number of conversion kicks is the same as the number of tries, Conversion goal is represented by binomial distribution. The following

are mean or probability of an event.

- try : $\lambda_a = 6$ (mean, Poisson dist.)
- conversion goal : $p = 0.7$ (probability, binomial dist.)
- penalty goal : $\lambda_c = 3 \times 0.75$ (mean, Poisson dist.)

Figure 16 shows a distribution of tries. If the distribution of conversion goals is superimposed on for example, $h = 3$ in Figure 16, $h = 3$ devides into 4 like Figure 17 by the binomial distribution of $n = 3,\ p = 0.7$. The value at $i = 2$ in this binomial distribution is

$$P = {}_3\mathrm{C}_2 p^2 (1-p)^{3-2} = 0.441000$$

and the superimposed rate between $h = 3$ and $i = 2$ is $c = 3 + 2 : 0.089235 \times 0.441000$ as it is shown in the figure. If the distribution $\lambda_c = 3 \times 0.75$ of penalty goals is superimposed on this, however the figiure being omitted, for example if $j = 2$, the superimposed rate between $h = 3$, $i = 2$ and $j = 2$ is

$$P = \exp(-\lambda_c){\lambda_c}^2/2! = 0.266792$$
$$c = 3 + 2 + 2 : 0.089235 \times 0.441000 \times 0.266792$$

The difference from soccer's case is that the first superimposing is not Poisson distribution but binomial distribution and nothing more. Figure 18 corresponds to Figure 15 in soccer's case.

The probability in Poisson distribution is represented like the following:

$$P = \mathrm{e}^{-\lambda}\lambda^N/N!$$

$N!$ gets bigger synergistically if N gets bigger. Not only $N!$ but also λ^N gets bigger. For example if $\lambda = 7.5$, in a language processor, $N!$ falls into output impossibility early at $N = 171$. So, we modify the above expression using Stirling's formula so that we can use it when N is big too.

$$\begin{aligned} N! &= \sqrt{2\pi N} N^N \mathrm{e}^{-N} \\ &= \sqrt{2\pi N}(N\lambda/\lambda)^N \mathrm{e}^{-N} \\ &= \sqrt{2\pi N}(N/\lambda)^N \lambda^N \mathrm{e}^{-N} \\ \lambda^N/N! &= \mathrm{e}^N/\sqrt{2\pi N}(N/\lambda)^N \\ &= (\mathrm{e}\lambda/N)^N/\sqrt{2\pi N} \end{aligned}$$

Therefore

$$P = \mathrm{e}^{-\lambda}\lambda^N/N! = \mathrm{e}^{-\lambda}(\mathrm{e}\lambda/N)^N/\sqrt{2\pi N}$$

The following shows by how many sets of $h,\ i,\ j$ for a score to be represented. The left, the centre and the right are a score, the number of sets of $h,\ i,\ j$ and the rate at which the score is realized respectively.

```
0   1  0.000261
3   1  0.000588
5   1  0.000470
6   1  0.000661
7   1  0.001097
8   1  0.001058
9   1  0.000496
```

10	2	0.002892
11	1	0.001190
12	2	0.002254
13	2	0.003730
14	2	0.003197
15	3	0.004823
16	2	0.003154
17	3	0.007464
18	3	0.005618
19	3	0.006123
20	4	0.010173
21	4	0.007634
22	4	0.011378
23	4	0.009216
24	5	0.013587
25	5	0.013341
26	5	0.011959
27	6	0.018266
28	6	0.014194
29	6	0.018535
30	7	0.017144
31	7	0.019448
32	7	0.021304
33	8	0.018032
34	8	0.024562
35	9	0.020440
36	9	0.023380
37	9	0.023286
38	10	0.022948
39	10	0.025775
40	11	0.021782
41	11	0.026523
42	12	0.023440
43	12	0.024208
44	12	0.024900
45	13	0.022865
46	13	0.025215
47	14	0.021690
48	14	0.024000
49	15	0.022111
50	15	0.021278
51	15	0.021980
52	16	0.019552
53	16	0.020773
54	17	0.018242
55	17	0.018710

56	18	0.017645
57	18	0.016222
58	18	0.016543
59	19	0.014544
60	19	0.014810
61	20	0.013226
62	20	0.012811
63	21	0.012184
64	21	0.010892
65	21	0.010863
66	22	0.009528
67	22	0.009317
68	23	0.008409
69	23	0.007811
70	24	0.007413
71	24	0.006518
72	24	0.006333
73	25	0.005560
74	25	0.005246
75	26	0.004756
76	26	0.004287
77	27	0.004030
78	27	0.003510
79	27	0.003321
80	28	0.002918
81	28	0.002673
82	29	0.002421
83	29	0.002137
84	30	0.001981
85	30	0.001715
86	30	0.001583
87	31	0.001390
88	31	0.001243
89	32	0.001120
90	32	0.000974
91	33	0.000889
92	33	0.000766
93	33	0.000691
94	34	0.000606
95	34	0.000531
96	35	0.000475
97	35	0.000408
98	36	0.000367
99	36	0.000315
100	36	0.000279
101	37	0.000243

102	37	0.000210
103	38	0.000186
104	38	0.000158
105	39	0.000140
106	39	0.000120
107	39	0.000104
108	40	0.000090
109	40	0.000077
110	40	0.000068
111	41	0.000057
112	41	0.000050
113	41	0.000042
114	41	0.000036
115	41	0.000031
116	41	0.000026
117	42	0.000023
118	41	0.000019
119	42	0.000017

99.99% point is score $s = 119$.

The following are rate of draw and so on.

- rate of draw : 0.017407
- mean score : 45.149855
- modal score : 41 (rate=0.026523)
- variance : 271.039845

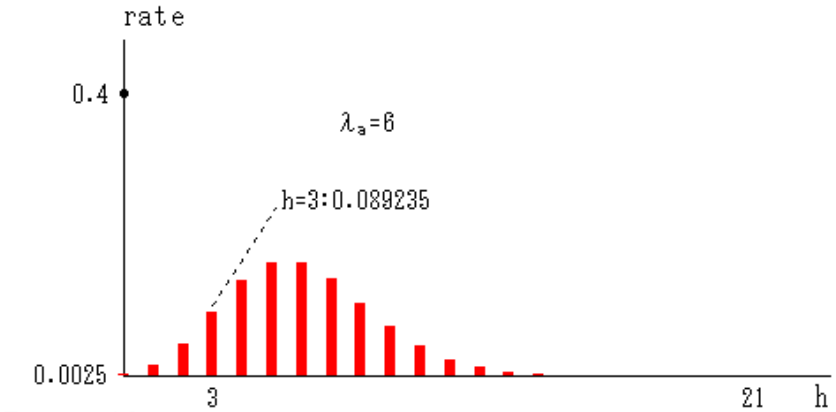


Figure 16

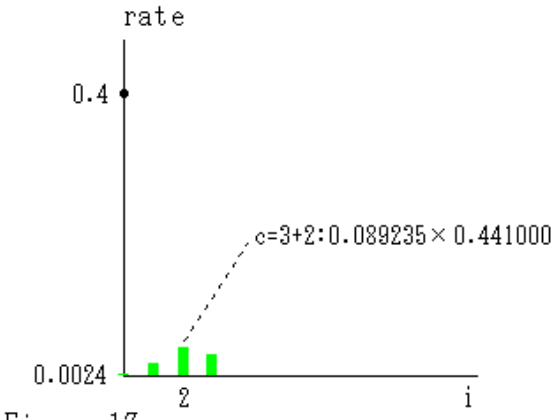


Figure 17

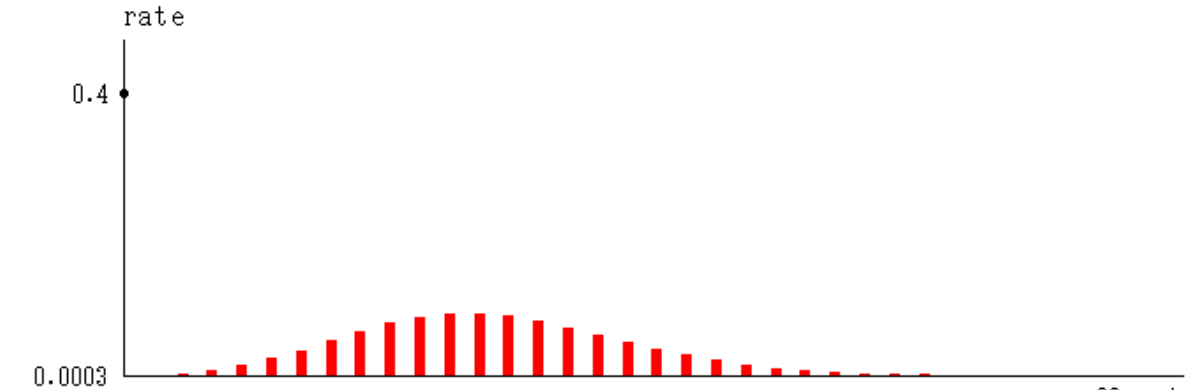


Figure 18

7. Comparison

We compare the previously mentioned data modifying them. Soccer's events are

・usual goal : $\lambda_a = 6$ (mean, Poisson dist.)
・LK goal : not used
・MK goal : $\lambda_c = 3 \times 0.75$ (mean, Poisson dist.)

Rugby's events are

・try : $\lambda_a = 6$ (mean, Poisson dist.)
・conversion goal : not used
・penalty goal : $\lambda_c = 3 \times 0.75$ (mean, Poisson dist.)

Points are

・usual goal, try : 5 points
・MK goal, penalty goal : 3 points

Naturally, rate of draw and so on become the same.

・rate of draw : 0.024482
・mean score : 36.749886

・modal score：36 (rate=0.043398)
・variance：170.243658

This is a technique with which if two programs on both soccer and rugby were made, we check that there is no miss in them.

This method by superimposing of distribution will be able to be applied to sports except soccer and rugby too. Because the point system of American football resembles rugby's, analysis will be easy. We assumed that the distribution of tries in rugby is Poisson distribution. Sometimes, if defensive strength of the other team falls down, there is a case that the number of tries increases abnormally. It is one of assignments how to process such situation.

8. Program

This program can be used for both soccer and rugby. If a data is interpreted arbitrarily, it is found whether the result of this program has reliability. The following is an example in soccer.

```
/*#define Rugby*/

/* Soccer */
#define tsum (90.)           /* game time */
#define mnt (90)             /* required time <= tsum */

#define n1CK (10)            /* number(CK) */
#define p1CK (0.04)          /* probability(CK) */
#define n1DFK (0)            /* number(DFK) */
#define p1DFK (0.04)         /* probability(DFK) */
#define lmd1a (2.6-(n1CK*p1CK+n1DFK*p1DFK))  /* mean in Poisson dist.(Usual goal) */
#define n1CK_LK (3)          /* number(CK, LK's share, from GK)
#define p1LK (0.6)           /* probability(LK, "on goal line") */
#define lmd1b (n1CK_LK*p1LK)  /* mean in Poisson dist.(LK goal) */
#define n1CK_MK (n1CK-n1CK_LK)  /* number(CK, MK's share, from FP, Figure 1 left)
#define p1MK (0.5)           /* probability(MK, Figure 3, "on goal line") */
#define lmd1c (n1CK_MK*p1MK)  /* mean in Poisson dist.(MK goal) */
#define p1NK (0.4)           /* probability(NK) */
#define lmd1d (n1DFK*p1NK)   /* --- */
#define p1b (0)              /* --- */

#define n2CK (10)            /* number(CK) */
#define p2CK (0.04)          /* probability(CK) */
#define n2DFK (0)            /* number(DFK) */
#define p2DFK (0.04)         /* probability(DFK) */
#define lmd2a (2.6-(n2CK*p2CK+n2DFK*p2DFK))  /* mean in Poisson dist.(Usual goal) */
#define n2CK_LK (3)          /* number(CK, LK's share, from GK)
#define p2LK (0.6)           /* probability(LK, "on goal line") */
#define lmd2b (n2CK_LK*p2LK)  /* mean in Poisson dist.(LK goal) */
#define n2CK_MK (n2CK-n2CK_LK)  /* number(CK, MK's share, from FP, Figure 1 left)
#define p2MK (0.5)           /* probability(MK, Figure 3, "on goal line") */
#define lmd2c (n2CK_MK*p2MK)  /* mean in Poisson dist.(MK goal) */
```

```
#define p2NK (0.4)           /* probability(NK) */
#define lmd2d (n2DFK*p2NK)   /* --- */
#define p2b (0)              /* --- */

#define ra 7                 /* point size(Usual goal) */
#define rb 3                 /* point size(LK goal) */
#define rc 1                 /* point size(MK goal) */
#define rd 0                 /* --- */
#define dgt 0                /* digits after decimal point of r?(ex. 1 <=> 0.5) */

#define dmax_ (100)
#define break_9s (0.999999)  /* standard value for loop break */
#define break9s  (0.9999)    /* standard value for loop break */
```

1 and 2 in the variables express one team and the other team respectively, `lmd??` is mean in Poisson distribution, `r?` is point size of a goal, dgt is the number of digits after decimal point of `r?` and break_9s and break9s are standard values for loop break. The previous result is got with the above.

・rate of draw：0.025572
・mean score：24.299902
・modal score：19 (rate=0.036103)
・variance：127.494322

Rate of draw is a value on one team and the other team and the others are values on one team.
We modify part of the above. LK goal and MK goal are not used and the point size is made 1.

```
#define lmd1a (2.6)
#define lmd1b (0)
#define lmd1c (0)
#define lmd2a (2.6)
#define lmd2b (0)
#define lmd2c (0)
#define ra 1
#define rb 0
#define rc 0
```

In this case

・rate of draw：0.179749
・mean score：2.599991
・modal score：2 (rate=0.251045)
・variance：2.599911

The following is an example in rugby.

```
#define Rugby
```

```
/* Rugby */
#define tsum (80.)           /* game time */
#define mnt (80)             /* required time <= tsum */

#define lmd1a (6)            /* mean in Poisson dist.(try) */
#define lmd1b (0)            /* --- */
#define n1PK (3)             /* number(PK) */
#define p1PK (0.75)          /* probability(PK) */
#define lmd1c (n1PK*p1PK)    /* mean in Poisson dist.(penalty goal) */
#define lmd1d (0)            /* --- */
#define p1b (0.7)            /* probability in binomial dist.(conversion goal) */

#define lmd2a (6)            /* mean in Poisson dist.(try) */
#define lmd2b (0)            /* --- */
#define n2PK (3)             /* number(PK) */
#define p2PK (0.75)          /* probability(PK) */
#define lmd2c (n2PK*p2PK)    /* mean in Poisson dist.(penalty goal) */
#define lmd2d (0)            /* --- */
#define p2b (0.7)            /* probability in binomial dist.(conversion goal) */

#define ra 5                 /* point size(try) */
#define rb 2                 /* point size(conversion goal) */
#define rc 3                 /* point size(penalty goal) */
#define rd 0                 /* --- */
#define dgt 0                /* digits after decimal point of r?(ex. 1 <=> 2.5) */

#define dmax_ (100)
#define break_9s (0.999999)  /* standard value for loop break */
#define break9s  (0.9999)    /* standard value for loop break */
```

p?b is probability of success of conversion goal. The previous result is got with the above.

・rate of draw：0.017407
・mean score：45.149855
・modal score：41 (rate=0.026523)
・variance：271.039845

Converson goal is not used.

```
#define ra 5
#define rb 0
#define rc 3
```

In this case

・rate of draw：0.024482
・mean score：36.749886

・modal score：36 (rate=0.043398)
・variance：170.243658

If converson goal is not used, rate of draw increases from 0.017407 to 0.024482, not that it increases according to this precisely. In fact, it will increase about to 40% extent.
Conversion goal is not used and the point size of try is made 6.

```
#define ra 6
#define rb 0
#define rc 3
```

In this case

・rate of draw：0.055578
・mean score：42.749868
・modal score：42 (rate=0.078142)
・variance：236.241214

Because if try is 5 points and penalty goal is 3 points, L.C.M is 15 and if try is 6 points and penalty goal is 3 points, L.C.M is 6, it is found intuitively that rate of draw of the latter is bigger. The ratio becomes a little more than 2 roughly.
Conversion goal is not used and each point size is made 1.

```
#define ra 1
#define rb 0
#define rc 1
```

In this case

・rate of draw：0.098984
・mean score：8.249977
・modal score：8 (rate=0.139053)
・variance：8.249738

Conversion goal and penalty goal are not used and the point size is made 1.

```
#define ra 1
#define rb 0
#define rc 0
```

In this case

・rate of draw：0.116426
・mean score：5.999991
・modal score：5 (rate=0.160623)
・variance：5.999895

Comparing rugby with soccer, because mean score of rugby is bigger than soccer, rate of draw of rugby is smaller than soccer.
Try and conversion goal are not used and the point size is made 1.

```
#define ra 0
#define rb 0
#define rc 1
```

In this case

・rate of draw：0.194198
・mean score：2.249990
・modal score：2 (rate=0.266792)
・variance：2.249909

Comparing rugby with soccer, because mean score of rugby is smaller than soccer, rate of draw of rugby is bigger than soccer.

9. Mean score

In soccer, the distributions of goals of three kinds are represented like Figure 9, 10, 11. The mean of points of each goal is $\lambda_c r_c$ assuming λ_c to be the mean of each goal($c = a,\ b,\ c$). Here, we assume that the mean of points of all the goals is $\Sigma\,\lambda_c r_c$. Using parameters in "8. Program"

$$\text{mean score} = \lambda_{1a} r_a + \lambda_{1b} r_b + \lambda_{1c} r_c = (2.6 - 0.4) \times 7 + 1.8 \times 3 + 3.5 \times 1 = 24.3$$

In rugby, because the first superimposing is binomial distribution(Figure 17)

$$\text{mean score} = \lambda_{1a} r_a + (\lambda_{1a} p_{1b}) r_b + \lambda_{1c} r_c = 6 \times 5 + (6 \times 0.7) \times 2 + (3 \times 0.75) \times 3 = 45.15$$

10. Direct Free Kick

A Corner Kick can be replaced with LK, MK. Direct Free Kick(DFK) can be replaced with a free kick of a different kind selectively too. We call it NK for short. The differences between NK and LK(MK) are point size and excluded zone of players except kicker and GK.

・example of point size：2
・excluded zone of players except kicker and GK：excluded zone of LK+triangle which location of the ball and two intersections which are made by goal line and line which indicates penalty area make(Figure 19)
・others：the same as LK

The points of three kinds are made MK<NK<LK. An offensive player points at goal if DFK and expresses 'L' with arms like Figure 8 if NK to referee. At location on which the ball to kick is placed, referee points at goal with its two arms putting its palms together if the former (like the expression of choice of penalty kick in ragby) and opens its two arms wide facing goal if the latter (like the Shiranui type in sumo)

We assume that the mean number of DFKs per game and the goal probability of DFK are 5 and 0.04 respectively. If DFKs are replaced with NK all, parameters becomes like the following:

```
/*#define Rugby*/

/* Soccer */
#define tsum (90.)           /* game time */
```

```
#define mnt (90)             /* required time <= tsum */

#define n1CK (10)            /* number(CK) */
#define p1CK (0.04)          /* probability(CK) */
#define n1DFK (5)            /* number(DFK) */
#define p1DFK (0.04)         /* probability(DFK) */
#define lmd1a (2.6-(n1CK*p1CK+n1DFK*p1DFK))  /* mean in Poisson dist.(Usual goal) */
#define n1CK_LK (3)          /* number(CK, LK's share, from GK)
#define p1LK (0.6)           /* probability(LK, "on goal line") */
#define lmd1b (n1CK_LK*p1LK)  /* mean in Poisson dist.(LK goal) */
#define n1CK_MK (n1CK-n1CK_LK)  /* number(CK, MK's share, from FP, Figure 1 left)
#define p1MK (0.5)           /* probability(MK, Figure 3, "on goal line") */
#define lmd1c (n1CK_MK*p1MK)  /* mean in Poisson dist.(MK goal) */
#define p1NK (0.4)           /* probability(NK) */
#define lmd1d (n1DFK*p1NK)   /* mean in Poisson dist.(NK goal) */
#define p1b (0)              /* --- */

#define n2CK (10)            /* number(CK) */
#define p2CK (0.04)          /* probability(CK) */
#define n2DFK (5)            /* number(DFK) */
#define p2DFK (0.04)         /* probability(DFK) */
#define lmd2a (2.6-(n2CK*p2CK+n2DFK*p2DFK))  /* mean in Poisson dist.(Usual goal) */
#define n2CK_LK (3)          /* number(CK, LK's share, from GK)
#define p2LK (0.6)           /* probability(LK, "on goal line") */
#define lmd2b (n2CK_LK*p2LK)  /* mean in Poisson dist.(LK goal) */
#define n2CK_MK (n2CK-n2CK_LK)  /* number(CK, MK's share, from FP, Figure 1 left)
#define p2MK (0.5)           /* probability(MK, Figure 3, "on goal line") */
#define lmd2c (n2CK_MK*p2MK)  /* mean in Poisson dist.(MK goal) */
#define p2NK (0.4)           /* probability(NK) */
#define lmd2d (n2DFK*p2NK)   /* mean in Poisson dist.(NK goal) */
#define p2b (0)              /* --- */

#define ra 7                 /* point size(Usual goal) */
#define rb 3                 /* point size(LK goal) */
#define rc 1                 /* point size(MK goal) */
#define rd 2                 /* point size(NK goal) */
#define dgt 0                /* digits after decimal point of r?(ex. 1 <=> 0.5) */

#define dmax_ (100)
#define break_9s (0.999999)  /* standard value for loop break */
#define break9s  (0.9999)    /* standard value for loop break */
```

In this case

- rate of draw : 0.025707
- mean score : 26.899902
- modal score : 24 (rate=0.036131)

・variance：125.694766

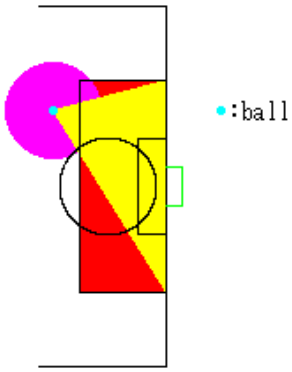


Figure 19

****

ファインサッカー

菊池盛雄

アブストラクト：
ラグビーの得点体系を参考にして複数の得点が入るように現在のサッカーのルールを修正します。

1. ラグビー

サッカーの平均スコアは2、3点ですから、プレーヤーがモチベーションを維持してフルに機動性を発揮するためには得点をきめ細かくする必要があります。プレーヤーがこまめに相手側のゴールに接近したら、シュートしたら、たとえゴールに至らなくても、相応の軽微な得点を与えるシステムを試合に挿入してみましょう。

ラグビーにおいてはトライが5点、コンバージョンゴールが2点、ペナルティゴールが3点となっています。コンバージョンゴールはトライ後のキックによるゴールであり、ペナルティゴールはペナルティキックによるゴールです。バスケットボールにおけるフリースローのような形式でコンバージョンゴール、ペナルティゴールの得点のような複数の小さな得点が得られるようにするので、従来のゴールはトライに対応させて7点とします。

2. コーナーキック

ボールが防御側のプレーヤーに触れてゴールラインを割った場合はコーナーキック (CK) が与えられます。この場合に攻撃側がバスケットボールにおけるフリースローのようなフリーキック (LK、MK) も選択的に行えるようにしましょう。攻撃側はCK、LK(MK) のどちらを選択するかを主審に示します。LK と MK の違いを以下に示します。

・LK：ボールがゴールキーパー (GK) に触れてゴールラインを割った場合
・MK：ボールがフィールドプレーヤー (FP) に触れてゴールラインを割った場合

LK、MK を与えるゴールラインの範囲は以下のように設定します。

・LK：フルライン (図1左)
・MK：フルライン (図1左)、セミライン (図1右) (”alternatives”)

MK では最適なゴールラインの範囲を求めることができていないので、図1にフルラインとセミラインを実験的な選択肢”alternatives”として示しています。試合前にどちらを採用するかをあらかじめ決めておきます。以後、”alternatives”はこのような意味で用います。

キックするボールを置いてはいけない範囲は以下のように設定します。

・LK：ペナルティエリア (図2)
・MK：ペナルティエリア (図2)、
　　ペナルティエリア＋ペナルティアーク (図3)
　　(”alternatives”)

白線を保護するため、ボールは白線にかからないこととします。

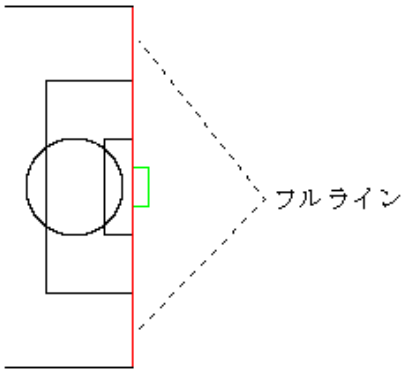
ボールをキックするプレーヤーと守備側の GK 以外のプレーヤーが位置してはいけない範囲 (排除ゾーン) は以下のように設定します。

・LK：ペナルティエリア＋ボールの位置を中心とする半径 9.15m の円 (図 4)
・MK：ペナルティエリア＋ボールの位置を中心とする半径 9.15m の円 (図 4)、
　　ペナルティエリア＋ペナルティアーク＋ボールの位置を中心とする半径 9.15m の円 (図 5)
　　(”alternatives”)

MK に関しては前の設定とこの設定は対応関係があり、図 2 なら図 4、図 3 なら図 5 とします。
ボールをキックするプレーヤーは 11 人 (GK+FP) の中から自由に選択できます。ただし、実際は FP の中から選択することになります。
キック時の守備側の GK のポジションは以下のように設定します。

・ゴールライン上、ゴールエリア内 (”alternatives”)

LK,MK における違反に関してはペナルティキックに関する判定を採用します。ただし、著しい遅延行為があれば、ペナルティキックに格上げされるか不成功と認定される判定が付加される可能性があります。
キック後、ボールに触れることができるのは守備側の GK だけであり、ゴールの成否に関わらず守備側によるゴールキックで試合を再開します。



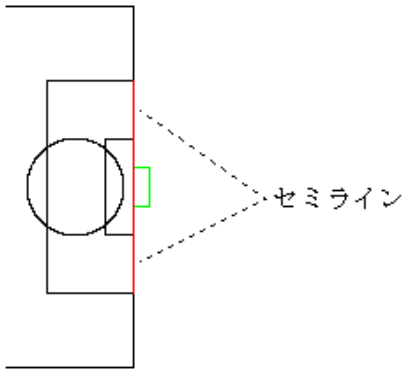


図 1

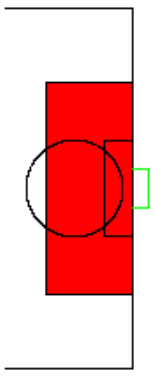


図 2

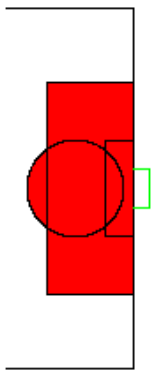


図 3

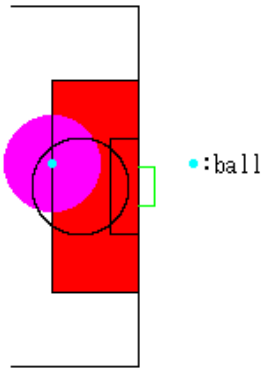


図 4

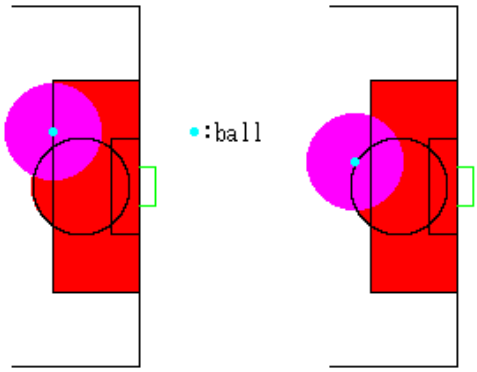


図 5

3. LK、MK に関するハンドシグナル

CK が発生すれば審判はハンドシグナルによってこれを明示します。LK、MK を導入すれば審判はこれらを区別して明示する必要があります。

図 6 はボールが GK に触れてゴールライン (フルライン) を割った場合のハンドシグナルです。図 7 はボールが FP に触れてゴールライン (フルラインまたはセミライン) を割った場合のハンドシグナルです。いずれにおいても、水平にした腕でコーナーエリアを指し示します。もしセミラインを採用していて、ボールがセミラインでないゴールラインを割った場合は CK、MK の選択はなく、CK だけとなるので従来のハンドシグナルを用います。

攻撃側は審判に対して、CK ならばコーナーエリアを指し示し、LK(MK) ならば図 8 のように両腕で'L' を表現します。

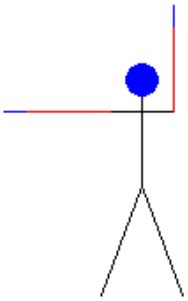


図 6

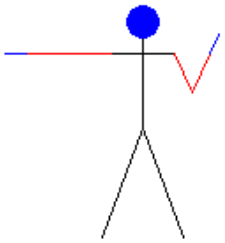


図 7

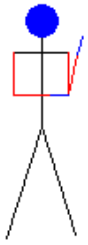


図 8

4. LK、MK によるゴールの得点

LK、MK によるゴールの得点はたとえば以下のように設定します。

・LK：3 点、2 点 (”alternatives”)

・MK：1 点、0.5 点 (”alternatives”)

上の例では、LK=3 点、MK=1 点;LK=3 点、MK=0.5 点;LK=2 点、MK=0.5 点の三通りの組合せとします。CK によるゴールの確率が 0.04 であるとすると、CK1 回当たりの期待値は 0.04 × 7=0.28 となります。MK が 1 点に設定され、MK によるゴールの確率が 0.5 であるとすると、MK1 回当たりの期待値は 0.5 × 1=0.5 となります。MK の期待値を CK と同程度にしたい場合は MK に関する採用する”alternatives”を変えてみる必要があります。

LK は、ボールが GK に触れてゴールラインを割ったら従来のゴールの得点に準じる得点を得る機会を与えます、という位置づけになっているので、比較的高得点 (従来のゴールの得点の 1/4〜1/2) になっています。一方、MK は、ボールが FP に触れてゴールラインを割ったら一種の PK 戦の先行を許可します、という位置づけになっています。したがって、MK は、MK によるゴールの統計学的分布のたとえば 95%点による得点が通常のゴールの 7 点未満になる、というように設定します。

5. 引き分けの統計学 (サッカー)

LK、MK を導入した場合に引き分けの割合がどの程度になるかを、統計学的な知識と数値計算を用いて考察してみます。一試合平均の従来のゴールの数と CK の数を各々$\lambda_a = 2.6$、n=10 とします。CK をすべて LK、MK に置き換えるとすれば (LK は”ゴールライン上”、MK は図 1 左、図 3、”ゴールライン上”とします)、一試合平均の従来のゴールの数は $\lambda_a = 2.6 - 10 \times 0.04 = 2.6 - 0.4$ となります。このゴールの分布をポアソン分布で表します。一試合平均 10 の CK の内訳を LK 分が 3、MK 分が 7 とします。LK、MK によるゴールの確率を各々0.6、0.5 とすれば、一試合平均の LK、MK によるゴールの数は

・LK によるゴールの数：$\lambda_b = 3 \times 0.6 = 1.8$
・MK によるゴールの数：$\lambda_c = 7 \times 0.5 = 3.5$

これらによるゴールの分布もポアソン分布で表します。図 9、図 10、図 11 はこれらの分布を表していて、99.9999%点は各々$h = 12,\ i = 11,\ j = 15$ となっています。これらの事象が独立であると仮定すれば、これらの事象の和事象もポアソン分布で表されます。図 12 はこの和事象 $\lambda_a + \lambda_b + \lambda_c = 7.5$ の分布を表していて、99.99%点はトータル $t = 20$ となっています。

図 12 において、横軸は事象数 t、縦軸はその割合を表しています。t は、従来のゴールの数+LK によるゴールの数+MK によるゴールの数、です。引き分けの割合は、各ゴール数に各得点を乗じたものの和であるスコアに関する割合の積和ですから、図 12 の分布からは引き分けの割合を求めることができません。

従来のゴールの分布に LK によるゴールの分布と MK によるゴールの分布を重ね合わせてみましょう。図 9 の従来のゴールの分布のたとえば $h = 2$ に図 10 の LK によるゴールの分布を重ね合わせると、$h = 2$ は図 13 のように 12 に分裂します。これは図 9 の $h = 2$ に関して事象数が $c = 2 + 0$ から $c = 2 + 11$ まで合成された割合 (二つの割合の積) で存在するということを意味します。次に、図 13 のたとえば $c = 2 + 3$ に図 11 の MK によるゴールの分布を重ね合わせると、$c = 2 + 3$ は図 14 のように 16 に分裂します。これは図 13 の $c = 2 + 3$ に関して事象数が $c = 2 + 3 + 0$ から $c = 2 + 3 + 15$ まで合成された割合 (三つの割合の積) で存在するということを意味します。

上述のステップをすべての場合について行い、同じ事象数に関して合成された割合の和をとり、図示すれば図 15 のようになります。この図と図 12 を数値で比較してみます。

	図 15	図 12
t= 0	0.000553	0.000553
t= 1	0.004148	0.004148
t= 2	0.015555	0.015555
t= 3	0.038889	0.038889

```
t= 4  0.072916  0.072916
t= 5  0.109375  0.109375
t= 6  0.136718  0.136718
t= 7  0.146484  0.146484
t= 8  0.137329  0.137329
t= 9  0.114440  0.114440
t=10  0.085830  0.085830
t=11  0.058521  0.058521
t=12  0.036575  0.036575
t=13  0.021101  0.021101
t=14  0.011304  0.011304
t=15  0.005652  0.005652
t=16  0.002649  0.002649
t=17  0.001169  0.001169
t=18  0.000487  0.000487
t=19  0.000192  0.000192
t=20  0.000072  0.000072
```

ほぼ一致します。したがって、上述の重ね合わせを行っても特に問題はないと言えるでしょう。

LK、MK によるゴールの得点が各々3、1 であり、事象数が $h,\ i,\ j(c = h + i + j)$ である場合のスコアは $s = 7 \times h + 3 \times i + 1 \times j$ となります。一般にスコアは複数の $h,\ i,\ j$ の組によって表されます。その場合は、そのスコアが実現される割合は各々の組の合成された割合 (たとえば図 14 中の $h = 2,\ i = 3,\ j = 4$ の場合の $c = 2 + 3 + 4 : 0.268144 \times 0.160671 \times 0.188812$) の和となります。以下のデータはスコアがいくつの $h,\ i,\ j$ の組によって表されるかを示しています。左がスコア、中が $h,\ i,\ j$ の組の数、右がスコアが実現される割合です。

```
 0   1  0.000553
 1   1  0.001936
 2   1  0.003388
 3   2  0.004948
 4   2  0.006943
 5   2  0.008519
 6   3  0.009422
 7   4  0.011284
 8   4  0.014413
 9   5  0.017055
10   6  0.019682
11   6  0.023058
12   7  0.025332
13   8  0.026157
14   9  0.027845
15  10  0.030414
16  11  0.031897
17  12  0.033241
18  12  0.035315
19  13  0.036103
20  14  0.035434
```

21	15	0.035406
22	16	0.035784
23	17	0.035122
24	18	0.034418
25	18	0.034302
26	19	0.033296
27	20	0.031411
28	21	0.030015
29	22	0.028825
30	23	0.027046
31	24	0.025380
32	24	0.024148
33	25	0.022525
34	26	0.020577
35	27	0.018990
36	28	0.017556
37	29	0.015921
38	30	0.014451
39	30	0.013263
40	31	0.011982
41	32	0.010655
42	32	0.009556
43	33	0.008570
44	34	0.007562
45	34	0.006680
46	34	0.005955
47	35	0.005238
48	35	0.004552
49	35	0.003985
50	36	0.003484
51	36	0.003004
52	36	0.002594
53	36	0.002257
54	36	0.001941
55	36	0.001653
56	36	0.001417
57	36	0.001212
58	36	0.001025
59	36	0.000868
60	36	0.000739
61	36	0.000623
62	36	0.000521
63	36	0.000439
64	36	0.000368
65	36	0.000306
66	36	0.000255

```
67  36  0.000213
68  36  0.000176
69  36  0.000145
70  36  0.000120
71  36  0.000099
72  36  0.000081
73  36  0.000066
74  36  0.000055
75  36  0.000045
76  36  0.000036
77  36  0.000030
78  36  0.000024
```

99.99%点はスコア $s = 78$ となっています。この中のたとえばスコア $s = 7$ について説明します。$h,\ i,\ j$ の組の数は 4 であり、このスコアが実現される割合は 0.011284 です。この割合は以下のようにして求めています。

$$h = 0,\ i = 0,\ j = 7$$
$$s = 7 \times h + 3 \times i + 1 \times j = 7$$
$$c = 0 + 0 + 7 : 0.110803 \times 0.165299 \times 0.038549 = 0.000706$$

$$h = 0,\ i = 1,\ j = 4$$
$$s = 7 \times h + 3 \times i + 1 \times j = 7$$
$$c = 0 + 1 + 4 : 0.110803 \times 0.297538 \times 0.188812 = 0.006225$$

$$h = 0,\ i = 2,\ j = 1$$
$$s = 7 \times h + 3 \times i + 1 \times j = 7$$
$$c = 0 + 2 + 1 : 0.110803 \times 0.267784 \times 0.105691 = 0.003136$$

$$h = 1,\ i = 0,\ j = 0$$
$$s = 7 \times h + 3 \times i + 1 \times j = 7$$
$$c = 1 + 0 + 0 : 0.243767 \times 0.165299 \times 0.030197 = 0.001217$$

$$0.000706 + 0.006225 + 0.003136 + 0.001217 = 0.011284$$

こうしてスコア軸上の割合が求められたので後は割合の積和を求めるだけです。現実なら、あるチームのスコア軸上の割合と他のチームのスコア軸上の割合との積和を求めるのですが、ここでは自チームと相手チームはスコア軸上の割合が同じであるとして積和を求めます。結果は

・引き分けの割合：0.025572

平均スコア、最頻スコア、分散は

・平均スコア：24.299902
・最頻スコア：19 (割合=0.036103)

・分散：127.494322

ただし、図 13、図 14 における重ね合わせの計算上限は図 10、図 11 における 99.9999%点より少し大きくしてあります。

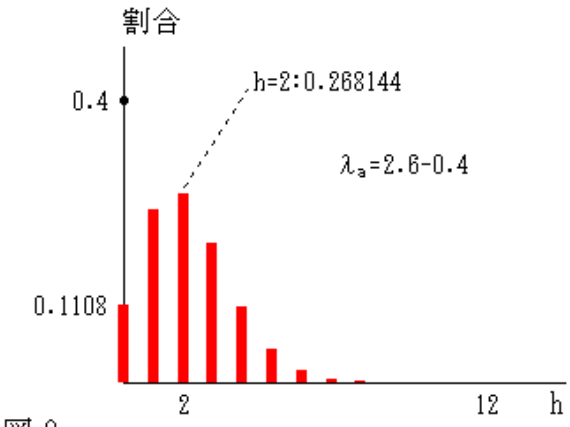


図 9

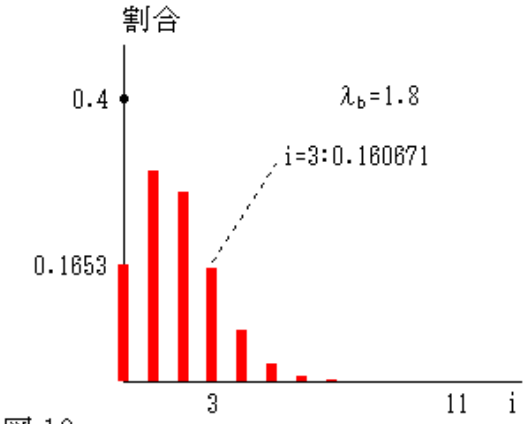


図 10

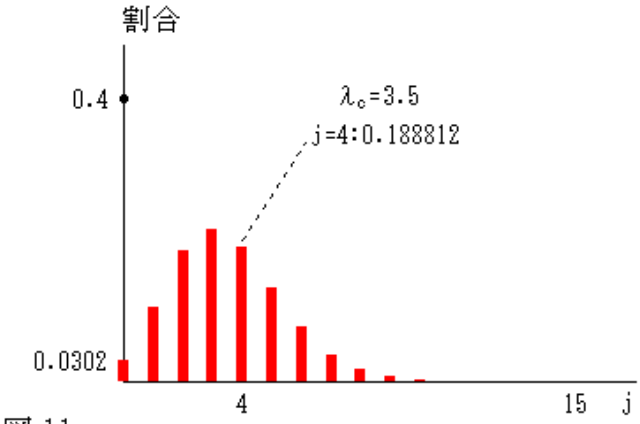


図 11

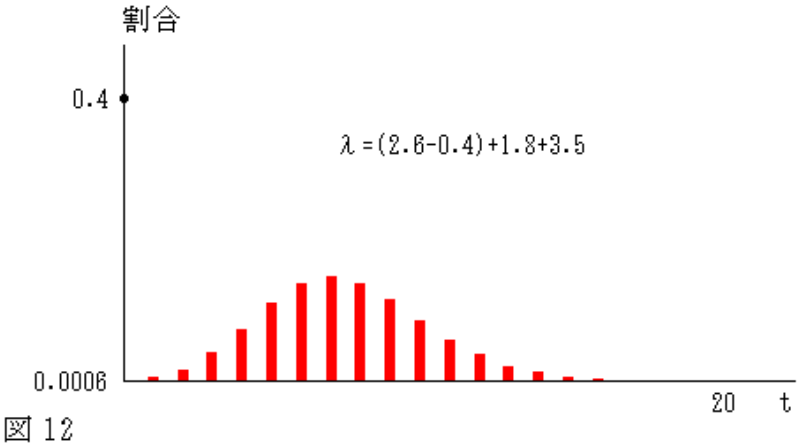


図 12

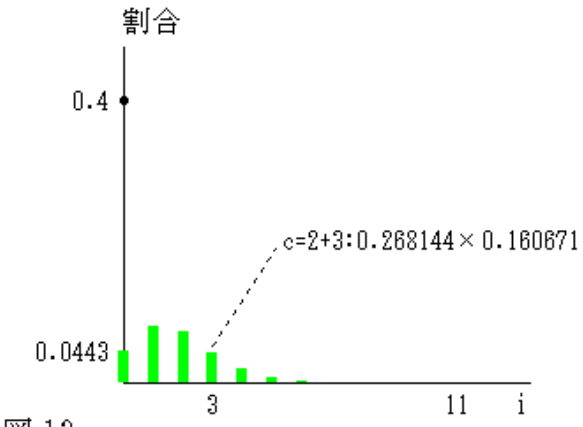


図 13

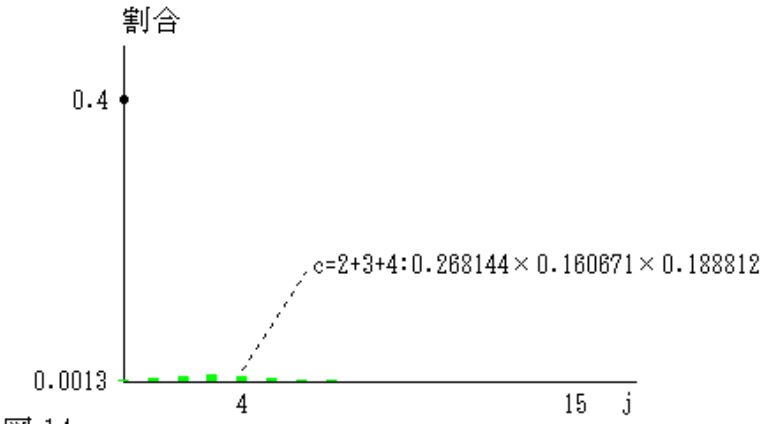


図 14

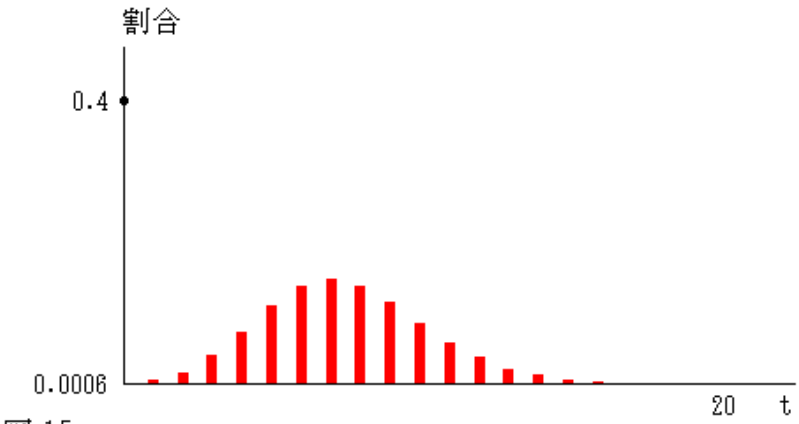


図 15

6. 引き分けの統計学 (ラグビー)

ラグビーの場合もサッカーの場合とほぼ同じ手法で引き分けの割合を求めることができます。簡単のため、ドロップキックによる得点は無視します。以下にサッカーとの対比を示します。

・トライ (5 点) ⟺ 従来のゴール (7 点)
・コンバージョンゴール (2 点) ⟺ LK によるゴール (3 点)
・ペナルティゴール (3 点) ⟺ MK によるゴール (1 点)

トライとペナルティゴールはポアソン分布で表し、これらの事象は独立であると仮定します。コンバージョンゴールはコンバージョンキックによるゴールであり、コンバージョンキックの数はトライの数と同じであるので、コンバージョンゴールは 2 項分布で表します。以下に事象の平均または確率を示します。

・トライ　　：$\lambda_a = 6$　(平均、ポアソン分布)
・コンバージョンゴール：$p = 0.7$　(確率、2 項分布)
・ペナルティゴール　　：$\lambda_c = 3 \times 0.75$　(平均、ポアソン分布)

図 16 はトライの分布を示しています。図 16 のたとえば $h = 3$ にコンバージョンゴールの分布を重ねます。$n = 3,\ p = 0.7$ の 2 項分布によって $h = 3$ は図 17 のように 4 つに分裂します。この 2 項分布の $i = 2$ における値は

$$P = {}_3\mathrm{C}_2 p^2 (1-p)^{3-2} = 0.441000$$

であり、図中に示されているように $h = 3,\ i = 2$ の合成された割合は $c = 3 + 2 : 0.089235 \times 0.441000$ となります。図は省略しますが、これにペナルティゴールの分布、$\lambda_c = 3 \times 0.75$ を重ね合わせると、たとえば $j = 2$ では、$h = 3,\ i = 2,\ j = 2$ の合成された割合は

$$P = \exp(-\lambda_c){\lambda_c}^2/2! = 0.266792$$
$$c = 3 + 2 + 2 : 0.089235 \times 0.441000 \times 0.266792$$

となります。サッカーの場合との違いは、最初の重ね合わせがポアソン分布ではなく 2 項分布である、ということだけです。図 18 はサッカーの場合の図 15 に対応しています。

ポアソン分布における確率は

$$P = \mathrm{e}^{-\lambda}\lambda^N/N!$$

のように表されます。$N!$ は N が大きくなると相乗的に大きくなります。$N!$ だけでなく λ^N も大きくなります。たとえば $\lambda = 7.5$ なら、ある処理系では $N!$ の方が $N = 171$ において先に出力不能に陥ります。そこで上式を N が大きい場合にも使えるようにスターリングの公式を用いて修正しましょう。

$$\begin{aligned} N! &= \sqrt{2\pi N} N^N \mathrm{e}^{-N} \\ &= \sqrt{2\pi N} (N\lambda/\lambda)^N \mathrm{e}^{-N} \\ &= \sqrt{2\pi N} (N/\lambda)^N \lambda^N \mathrm{e}^{-N} \\ \lambda^N/N! &= \mathrm{e}^N / \sqrt{2\pi N} (N/\lambda)^N \\ &= (\mathrm{e}\lambda/N)^N / \sqrt{2\pi N} \end{aligned}$$

したがって

$$P = \mathrm{e}^{-\lambda}\lambda^N/N! = \mathrm{e}^{-\lambda}(\mathrm{e}\lambda/N)^N/\sqrt{2\pi N}$$

以下のデータはスコアがいくつの $h,\ i,\ j$ の組によって表されるかを示しています。左がスコア、中が $h,\ i,\ j$ の組の数、右がスコアが実現される割合です。

```
 0   1  0.000261
 3   1  0.000588
 5   1  0.000470
 6   1  0.000661
 7   1  0.001097
 8   1  0.001058
 9   1  0.000496
10   2  0.002892
11   1  0.001190
```

12	2	0.002254
13	2	0.003730
14	2	0.003197
15	3	0.004823
16	2	0.003154
17	3	0.007464
18	3	0.005618
19	3	0.006123
20	4	0.010173
21	4	0.007634
22	4	0.011378
23	4	0.009216
24	5	0.013587
25	5	0.013341
26	5	0.011959
27	6	0.018266
28	6	0.014194
29	6	0.018535
30	7	0.017144
31	7	0.019448
32	7	0.021304
33	8	0.018032
34	8	0.024562
35	9	0.020440
36	9	0.023380
37	9	0.023286
38	10	0.022948
39	10	0.025775
40	11	0.021782
41	11	0.026523
42	12	0.023440
43	12	0.024208
44	12	0.024900
45	13	0.022865
46	13	0.025215
47	14	0.021690
48	14	0.024000
49	15	0.022111
50	15	0.021278
51	15	0.021980
52	16	0.019552
53	16	0.020773
54	17	0.018242
55	17	0.018710
56	18	0.017645
57	18	0.016222

58	18	0.016543
59	19	0.014544
60	19	0.014810
61	20	0.013226
62	20	0.012811
63	21	0.012184
64	21	0.010892
65	21	0.010863
66	22	0.009528
67	22	0.009317
68	23	0.008409
69	23	0.007811
70	24	0.007413
71	24	0.006518
72	24	0.006333
73	25	0.005560
74	25	0.005246
75	26	0.004756
76	26	0.004287
77	27	0.004030
78	27	0.003510
79	27	0.003321
80	28	0.002918
81	28	0.002673
82	29	0.002421
83	29	0.002137
84	30	0.001981
85	30	0.001715
86	30	0.001583
87	31	0.001390
88	31	0.001243
89	32	0.001120
90	32	0.000974
91	33	0.000889
92	33	0.000766
93	33	0.000691
94	34	0.000606
95	34	0.000531
96	35	0.000475
97	35	0.000408
98	36	0.000367
99	36	0.000315
100	36	0.000279
101	37	0.000243
102	37	0.000210
103	38	0.000186

104　38　0.000158
105　39　0.000140
106　39　0.000120
107　39　0.000104
108　40　0.000090
109　40　0.000077
110　40　0.000068
111　41　0.000057
112　41　0.000050
113　41　0.000042
114　41　0.000036
115　41　0.000031
116　41　0.000026
117　42　0.000023
118　41　0.000019
119　42　0.000017

99.99%点はスコア $s = 119$ となっています。
　引き分けの割合その他は以下のようになります。

・引き分けの割合：0.017407
・平均スコア：45.149855
・最頻スコア：41 (割合=0.026523)
・分散：271.039845

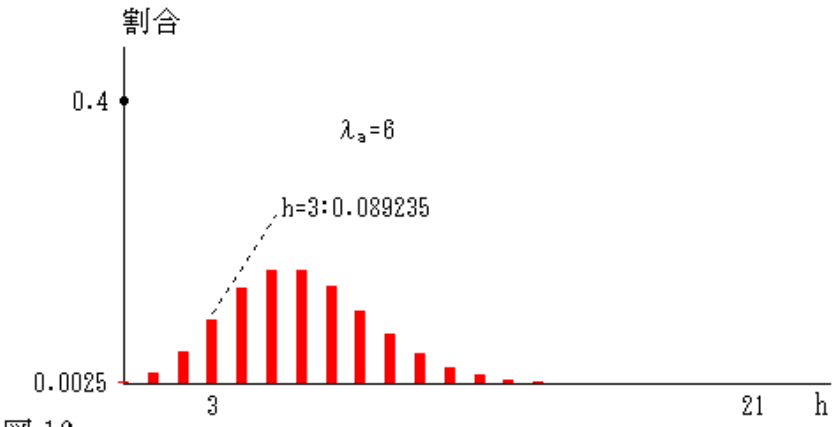


図 16

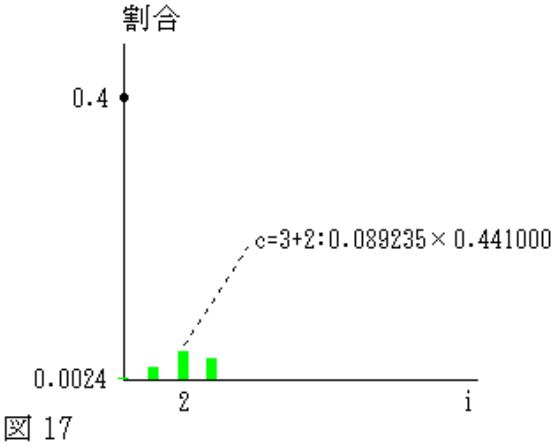


図 17

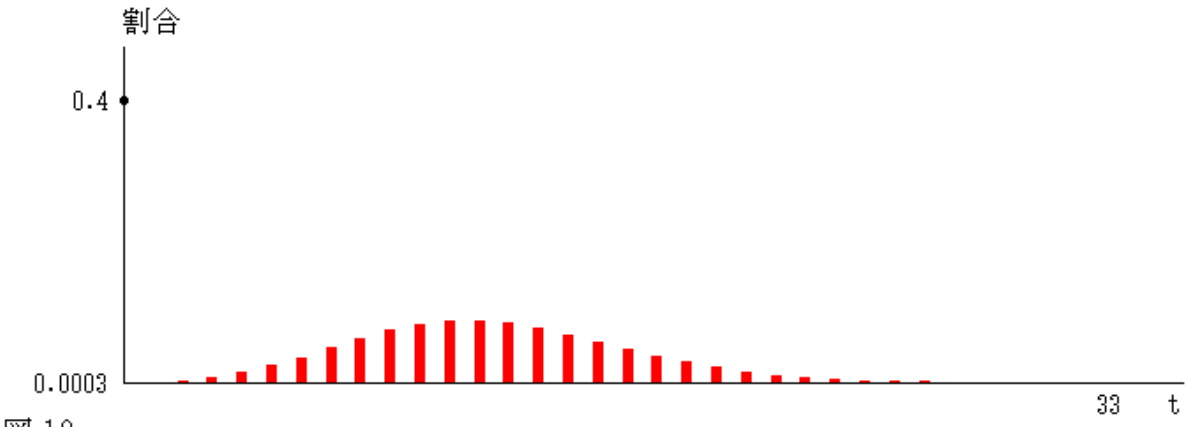


図 18

7. 比較

サッカーとラグビーの前述のデータを修正して比較してみます。サッカーの事象は

・従来のゴール　：$\lambda_a = 6$　(平均、ポアソン分布)
・LK によるゴール：なし
・MK によるゴール：$\lambda_c = 3 \times 0.75$　(平均、ポアソン分布)

とします。ラグビーの事象は

・トライ　　：$\lambda_a = 6$　(平均、ポアソン分布)
・コンバージョンゴール：なし
・ペナルティゴール　　：$\lambda_c = 3 \times 0.75$　(平均、ポアソン分布)

とします。得点は

・従来のゴール、トライ　　：5 点
・MK によるゴール、ペナルティゴール：3 点

とします。当然のことながら、引き分けの割合その他は同じになります。

・引き分けの割合：0.024482
・平均スコア：36.749886
・最頻スコア：36 (割合=0.043398)
・分散：170.243658

これは、サッカーとラグビー、両方のプログラムを作成した場合にプログラムにミスがないことを確かめる一つの手法です。

この分布の重ね合わせによる解析は、サッカー、ラグビー以外のスポーツにも応用できると思われます。アメリカンフットボールの得点体系はラグビーのそれに似ているので、解析は容易でしょう。ラグビーのトライの分布がポアソン分布である、としました。時に一方の守備力が大幅に落ちると、トライの数が異常に増えることがあります。このような試合をどう処理するか、が課題の一つです。

8. プログラム

プログラムはサッカー、ラグビーの両方に使えます。プログラムの結果が信頼性を有するかどうかは

データを恣意的に解釈してみればわかります。サッカーでは以下のように値をセットします。

```
/*#define Rugby*/

/* Soccer */
#define tsum (90.)           /* game time */
#define mnt (90)             /* required time <= tsum */

#define n1CK (10)            /* number(CK) */
#define p1CK (0.04)          /* probability(CK) */
#define n1DFK (0)            /* number(DFK) */
#define p1DFK (0.04)         /* probability(DFK) */
#define lmd1a (2.6-(n1CK*p1CK+n1DFK*p1DFK))  /* mean in Poisson dist.(Usual goal) */
#define n1CK_LK (3)          /* number(CK, LK's share, from GK)
#define p1LK (0.6)           /* probability(LK, "on goal line") */
#define lmd1b (n1CK_LK*p1LK)  /* mean in Poisson dist.(LK goal) */
#define n1CK_MK (n1CK-n1CK_LK)  /* number(CK, MK's share, from FP, Figure 1 left)
#define p1MK (0.5)           /* probability(MK, Figure 3, "on goal line") */
#define lmd1c (n1CK_MK*p1MK)  /* mean in Poisson dist.(MK goal) */
#define p1NK (0.4)           /* probability(NK) */
#define lmd1d (n1DFK*p1NK)   /* --- */
#define p1b (0)              /* --- */

#define n2CK (10)            /* number(CK) */
#define p2CK (0.04)          /* probability(CK) */
#define n2DFK (0)            /* number(DFK) */
#define p2DFK (0.04)         /* probability(DFK) */
#define lmd2a (2.6-(n2CK*p2CK+n2DFK*p2DFK))  /* mean in Poisson dist.(Usual goal) */
#define n2CK_LK (3)          /* number(CK, LK's share, from GK)
#define p2LK (0.6)           /* probability(LK, "on goal line") */
#define lmd2b (n2CK_LK*p2LK)  /* mean in Poisson dist.(LK goal) */
#define n2CK_MK (n2CK-n2CK_LK)  /* number(CK, MK's share, from FP, Figure 1 left)
#define p2MK (0.5)           /* probability(MK, Figure 3, "on goal line") */
#define lmd2c (n2CK_MK*p2MK)  /* mean in Poisson dist.(MK goal) */
#define p2NK (0.4)           /* probability(NK) */
#define lmd2d (n2DFK*p2NK)   /* --- */
#define p2b (0)              /* --- */

#define ra 7                 /* point size(Usual goal) */
#define rb 3                 /* point size(LK goal) */
#define rc 1                 /* point size(MK goal) */
#define rd 0                 /* --- */
#define dgt 0                /* digits after decimal point of r?(ex. 1 <=> 0.5) */

#define dmax_ (100)
#define break_9s (0.999999)  /* standard value for loop break */
#define break9s  (0.9999)    /* standard value for loop break */
```

変数名の中の 1、2 は各々自チーム、相手チームを表し、lmd??はポアソン分布の平均、r?はゴールの得点、dgt は r?の小数点の後の桁数、break_9s と break9s はループを break する基準の値です。上記の値で前述の結果が得られます。

・引き分けの割合：0.025572
・平均スコア：24.299902
・最頻スコア：19 (割合=0.036103)
・分散：127.494322

引き分けの割合は自チーム、相手チームに関する値、他は自チームに関する値です。
上記の値の一部を修正します。LK によるゴール、MK によるゴールを無しにし、得点を 1 とします。

```
#define lmd1a (2.6)
#define lmd1b (0)
#define lmd1c (0)
#define lmd2a (2.6)
#define lmd2b (0)
#define lmd2c (0)
#define ra 1
#define rb 0
#define rc 0
```

この場合は

・引き分けの割合：0.179749
・平均スコア：2.599991
・最頻スコア：2 (割合=0.251045)
・分散：2.599911

ラグビーでは以下のように値をセットします。

```
#define Rugby

/* Rugby */
#define tsum (80.)          /* game time */
#define mnt (80)            /* required time <= tsum */

#define lmd1a (6)           /* mean in Poisson dist.(try) */
#define lmd1b (0)           /* --- */
#define n1PK (3)            /* number(PK) */
#define p1PK (0.75)         /* probability(PK) */
#define lmd1c (n1PK*p1PK)   /* mean in Poisson dist.(penalty goal) */
#define lmd1d (0)           /* --- */
#define p1b (0.7)           /* probability in binomial dist.(conversion goal) */

#define lmd2a (6)           /* mean in Poisson dist.(try) */
#define lmd2b (0)           /* --- */
```

```
#define n2PK (3)              /* number(PK) */
#define p2PK (0.75)           /* probability(PK) */
#define lmd2c (n2PK*p2PK)     /* mean in Poisson dist.(penalty goal) */
#define lmd2d (0)             /* --- */
#define p2b (0.7)             /* probability in binomial dist.(conversion goal) */

#define ra 5                  /* point size(try) */
#define rb 2                  /* point size(conversion goal) */
#define rc 3                  /* point size(penalty goal) */
#define rd 0                  /* --- */
#define dgt 0                 /* digits after decimal point of r?(ex. 1 <=> 2.5) */

#define dmax_ (100)
#define break_9s (0.999999)  /* standard value for loop break */
#define break9s  (0.9999)    /* standard value for loop break */
```

p?b はコンバージョンゴールが成功する確率です。上記の値で前述の結果が得られます。

・引き分けの割合：0.017407
・平均スコア：45.149855
・最頻スコア：41 (割合=0.026523)
・分散：271.039845

コンバージョンゴールを無しにします。

```
#define ra 5
#define rb 0
#define rc 3
```

この場合は

・引き分けの割合：0.024482
・平均スコア：36.749886
・最頻スコア：36 (割合=0.043398)
・分散：170.243658

コンバージョンゴールを無しにすると引き分けの割合は 0.017407 から 0.024482 に増えます。正確にこの通りになるというのではなく、40%程度増えるということです。
コンバージョンゴールを無しにし、トライを 6 点にします。

```
#define ra 6
#define rb 0
#define rc 3
```

この場合は

・引き分けの割合：0.055578
・平均スコア：42.749868

・最頻スコア：42 (割合=0.078142)
・分散：236.241214

トライが 5 点でペナルティゴールが 3 点なら最小公倍数が 15、トライが 6 点でペナルティゴールが 3 点なら最小公倍数が 6 ですから、直観的に後者の方が引き分けが多くなるということがわかります。大体 $0.055578/0.024482 = 2$ 倍強になります。
コンバージョンゴールを無しにし、各得点を 1 とします。

```
#define ra 1
#define rb 0
#define rc 1
```

この場合は

・引き分けの割合：0.098984
・平均スコア：8.249977
・最頻スコア：8 (割合=0.139053)
・分散：8.249738

得点はトライだけとし、その得点を 1 とします。

```
#define ra 1
#define rb 0
#define rc 0
```

この場合は

・引き分けの割合：0.116426
・平均スコア：5.999991
・最頻スコア：5 (割合=0.160623)
・分散：5.999895

サッカーと比較すると、平均スコアが大きいので引き分けの割合は小さくなっています。
得点はペナルティゴールだけとし、その得点を 1 とします。

```
#define ra 0
#define rb 0
#define rc 1
```

この場合は

・引き分けの割合：0.194198
・平均スコア：2.249990
・最頻スコア：2 (割合=0.266792)
・分散：2.249909

サッカーと比較すると、平均スコアが小さいので引き分けの割合は大きくなっています。

9. 平均スコア

サッカーでは三種類のゴールの分布は図 9、10、11 のように表されます。各ゴールの得点の平均は λ_c を各ゴールの平均として $\lambda_c r_c$ です ($c = a,\ b,\ c$)。ここで全ゴールの得点の平均が $\Sigma\lambda_c r_c$ であると仮定します。”8. プログラム”のパラメーターを用いると

$$\text{平均スコア} = \lambda_{1a} r_a + \lambda_{1b} r_b + \lambda_{1c} r_c = (2.6 - 0.4) \times 7 + 1.8 \times 3 + 3.5 \times 1 = 24.3$$

ラグビーでは最初の重ね合わせが 2 項分布なので (図 17)

$$\text{平均スコア} = \lambda_{1a} r_a + (\lambda_{1a} p_{1b}) r_b + \lambda_{1c} r_c = 6 \times 5 + (6 \times 0.7) \times 2 + (3 \times 0.75) \times 3 = 45.15$$

10. 直接フリーキック

コーナーキックは選択的に LK、MK に置き換えることができます。直接フリーキック (DFK) も選択的に別種のフリーキックに置き換えることができ、これを NK と略称します。NK と LK(MK) との違いは得点と非関与プレーヤーの排除ゾーンです。

・得点例：2 点
・非関与プレーヤーの排除ゾーン：LK の排除ゾーン＋ボールの位置、ゴールラインとペナルティエリアを示すラインの交点が作る三角形 (図 19)
・他：LK と同じ

得点は MK<NK<LK とします。攻撃側は審判に対して、DFK ならばゴールを指し示し、NK ならば図 8 のように両腕で’L’ を表現します。審判はキックするボールを置く位置において、前者なら掌を合わせて両腕でゴールを指し示し (ラグビーのペナルティキックの選択の表現のように)、後者ならゴールを向いて両腕を大きく開きます (相撲の不知火型のように)。

一試合平均の DFK の数とそのゴールの確率を 5、0.04 とし、NK によるゴールの確率を 0.4 とします。DFK をすべて NK に置き換えるとすればパラメーターは以下のようになります。

```
/*#define Rugby*/

/* Soccer */
#define tsum (90.)           /* game time */
#define mnt (90)             /* required time <= tsum */

#define n1CK (10)            /* number(CK) */
#define p1CK (0.04)          /* probability(CK) */
#define n1DFK (5)            /* number(DFK) */
#define p1DFK (0.04)         /* probability(DFK) */
#define lmd1a (2.6-(n1CK*p1CK+n1DFK*p1DFK))  /* mean in Poisson dist.(Usual goal) */
#define n1CK_LK (3)          /* number(CK, LK's share, from GK)
#define p1LK (0.6)           /* probability(LK, "on goal line") */
#define lmd1b (n1CK_LK*p1LK)  /* mean in Poisson dist.(LK goal) */
#define n1CK_MK (n1CK-n1CK_LK)  /* number(CK, MK's share, from FP, Figure 1 left)
#define p1MK (0.5)           /* probability(MK, Figure 3, "on goal line") */
#define lmd1c (n1CK_MK*p1MK)  /* mean in Poisson dist.(MK goal) */
#define p1NK (0.4)           /* probability(NK) */
#define lmd1d (n1DFK*p1NK)   /* mean in Poisson dist.(NK goal) */
```

```
#define p1b (0)              /* --- */

#define n2CK (10)            /* number(CK) */
#define p2CK (0.04)          /* probability(CK) */
#define n2DFK (5)            /* number(DFK) */
#define p2DFK (0.04)         /* probability(DFK) */
#define lmd2a (2.6-(n2CK*p2CK+n2DFK*p2DFK))  /* mean in Poisson dist.(Usual goal) */
#define n2CK_LK (3)          /* number(CK, LK's share, from GK)
#define p2LK (0.6)           /* probability(LK, "on goal line") */
#define lmd2b (n2CK_LK*p2LK)  /* mean in Poisson dist.(LK goal) */
#define n2CK_MK (n2CK-n2CK_LK)  /* number(CK, MK's share, from FP, Figure 1 left)
#define p2MK (0.5)           /* probability(MK, Figure 3, "on goal line") */
#define lmd2c (n2CK_MK*p2MK)  /* mean in Poisson dist.(MK goal) */
#define p2NK (0.4)           /* probability(NK) */
#define lmd2d (n2DFK*p2NK)   /* mean in Poisson dist.(NK goal) */
#define p2b (0)              /* --- */

#define ra 7                 /* point size(Usual goal) */
#define rb 3                 /* point size(LK goal) */
#define rc 1                 /* point size(MK goal) */
#define rd 2                 /* point size(NK goal) */
#define dgt 0                /* digits after decimal point of r?(ex. 1 <=> 0.5) */

#define dmax_ (100)
#define break_9s (0.999999)  /* standard value for loop break */
#define break9s  (0.9999)    /* standard value for loop break */
```

この場合は

・引き分けの割合：0.025707
・平均スコア：26.899902
・最頻スコア：24 (割合=0.036131)
・分散：125.694766

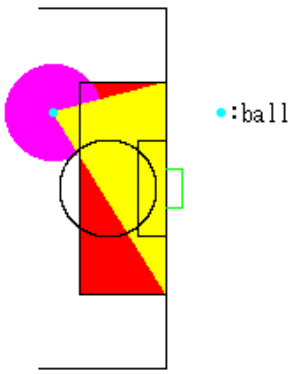


図 19

****

List 1:draw.c

```
/* draw.c                                                                    */
/* by Morio Kikuchi 2017.1.1                                                 */
/* COMPILER:gpp(dos)                                                         */
/* COMMANDLINE:gcc -Dfar= -o draw.exe draw.c                                 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <malloc.h>

/*#define Rugby*/

#ifndef Rugby
/* Soccer */
#define mnt (90)

#define lmd1a (0.015*mnt)
#define lmd1b (0)
#define lmd1c (0)
#define lmd1d (0)
#define p1b (0)

#define lmd2a (0.012*mnt)
#define lmd2b (0)
#define lmd2c (0)
#define lmd2d (0)
#define p2b (0)

#define ra 1
#define rb 0
#define rc 0
#define rd 0
#define dgt 0
/*
#define tsum (90.)
#define mnt (89)

#define n1CK (10)
#define p1CK (0.04)
#define n1DFK (0)
#define p1DFK (0.04)
```

```
#define lmd1a (((2.6-(n1CK*p1CK+n1DFK*p1DFK))/tsum)*mnt)
#define n1CK_LK (3)
#define p1LK (0.6)
#define lmd1b (((n1CK_LK*p1LK)/tsum)*mnt)
#define n1CK_MK (n1CK-n1CK_LK)
#define p1MK (0.5)
#define lmd1c (((n1CK_MK*p1MK)/tsum)*mnt)
#define p1NK (0.4)
#define lmd1d (((n1DFK*p1NK)/tsum)*mnt)
#define p1b (0)

#define n2CK (10)
#define p2CK (0.04)
#define n2DFK (0)
#define p2DFK (0.04)
#define lmd2a (((2.6-(n2CK*p2CK+n2DFK*p2DFK))/tsum)*mnt)
#define n2CK_LK (3)
#define p2LK (0.6)
#define lmd2b (((n2CK_LK*p2LK)/tsum)*mnt)
#define n2CK_MK (n2CK-n2CK_LK)
#define p2MK (0.5)
#define lmd2c (((n2CK_MK*p2MK)/tsum)*mnt)
#define p2NK (0.4)
#define lmd2d (((n2DFK*p2NK)/tsum)*mnt)
#define p2b (0)

#define ra 7
#define rb 3
#define rc 1
#define rd 0
#define dgt 0*/
/*
#define mnt (90)

#define lmd1a (2.6-0.4*1.)
#define lmd1b (1.8)
#define lmd1c (3.5)
#define lmd1d (0)
#define p1b (0)

#define lmd2a (2.6-0.4*1.)
#define lmd2b (1.8)
#define lmd2c (3.5)
#define lmd2d (0)
#define p2b (0)
```

```
#define ra 7
#define rb 3
#define rc 1
#define rd 0
#define dgt 0*/
#else
/* Rugby ************************************************/
#define tsum (80.)
#define mnt (79.9)

#define lmd1a ((4/tsum)*mnt)
#define lmd1b (0)
#define lmd1c ((1/tsum)*mnt)
#define lmd1d (0)
#define p1b (0.8)

#define lmd2a ((3.2/tsum)*mnt)
#define lmd2b (0)
#define lmd2c ((0.8/tsum)*mnt)
#define lmd2d (0)
#define p2b (0.7)

#define ra 5
#define rb 2
#define rc 3
#define rd 0
#define dgt 0
/*
#define tsum (80.)
#define mnt (79.9)

#define lmd1a ((6/tsum)*mnt)
#define lmd1b (0)
#define n1PK (3)
#define p1PK (0.75)
#define lmd1c (((n1PK*p1PK)/tsum)*mnt)
#define lmd1d (0)
#define p1b (0.7)

#define lmd2a ((6/tsum)*mnt)
#define lmd2b (0)
#define n2PK (3)
#define p2PK (0.75)
#define lmd2c (((n2PK*p2PK)/tsum)*mnt)
#define lmd2d (0)
#define p2b (0.7)
```

```
#define ra 5
#define rb 2
#define rc 3
#define rd 0
#define dgt 0*/
#endif

#define dmax_ (100)
#define break_9s (0.999999)
#define break9s  (0.9999)
#define READ 0
#define WRITE 1

int Umax,LKmax,MKmax,NKmax;
int lmt_Ph,lmt_Ps,lmt_Psum;
double mgf;
double /***Pji,**Qji,*/*Ph,*Qh,*Pi,*Qi,*Pj,*Qj,*Psum,*Qsum;
typedef struct {
int /*k,j,i,*/n;double P;} nP;
nP *Ps,*Qs;

FILE *fp;

void figs(void);


int main(int argc,unsigned char **argv)
{
mgf=pow(10,dgt);

figs();

return 0;
}/** main **/


double times(int n)
{
int val;
double t;

if(n<=1) return 1.;
else{
t=n;val=n;
while(1){
```

```
val--;
t=t*val;
if(val<=1) break;
}

return t;
}
}/** times **/

int getmax(int flag,double lmd)
{
int k;
double val[2];

if(lmd>0){
val[0]=0;
for(k=0;;k++){
val[1]=exp(-lmd)*pow(lmd,k)/times(k);  /* Po:3 */
val[0]+=val[1];
if(flag==0) {if(val[0]>break9s) break;}
else        {if(val[0]>break_9s) break;}
}

printf(" lmd=%f k=%d\n",lmd,k);
return (k+2);
}

return 0;
}/** getmax **/


int mallocs(void)
{
int Umax_,LKmax_;

if(lmd1a>lmd2a)  Umax=getmax(1,lmd1a);else  Umax=getmax(1,lmd2a);
#ifndef Rugby
if(lmd1b>lmd2b) LKmax=getmax(1,lmd1b);else LKmax=getmax(1,lmd2b);
#endif
if(lmd1c>lmd2c) MKmax=getmax(1,lmd1c);else MKmax=getmax(1,lmd2c);
#ifndef Rugby
if(lmd1d>lmd2d) NKmax=getmax(1,lmd1d);else NKmax=getmax(1,lmd2d);
#endif

#ifndef Rugby
Umax_=LKmax;                        /* for LK */
```

```
LKmax_=LKmax;
#else
Umax_=Umax;
LKmax_=Umax;                          /* for conv. goal */
#endif

lmt_Ph=Umax_+1;

Ph=(double *)malloc(sizeof(double)*(lmt_Ph));
Qh=(double *)malloc(sizeof(double)*(lmt_Ph));
Pi=(double *)malloc(sizeof(double)*(MKmax+1));
Qi=(double *)malloc(sizeof(double)*(MKmax+1));
Pj=(double *)malloc(sizeof(double)*(NKmax+1));
Qj=(double *)malloc(sizeof(double)*(NKmax+1));

lmt_Ps=(int)((ra*Umax+rb*LKmax_+rc*MKmax+rd*NKmax+dmax_)*mgf);
Ps=(nP *)malloc(sizeof(nP)*(lmt_Ps));
if(Ps==NULL) {printf(" ?Ps:m\n");return 1;}
Qs=(nP *)malloc(sizeof(nP)*(lmt_Ps));
if(Qs==NULL) {printf(" ?Qs:m\n");return 1;}

lmt_Psum=Umax+LKmax_+MKmax+NKmax+1;
Psum=(double *)malloc(sizeof(double)*(lmt_Psum));
if(Psum==NULL) {printf(" ?Psum:m\n");return 1;}
Qsum=(double *)malloc(sizeof(double)*(lmt_Psum));
if(Qsum==NULL) {printf(" ?Qsum:m\n");return 1;}

return 0;
}/** mallocs **/


int reallocs(int flag)
{
int i,j,old;

if(flag==0){
old=lmt_Ph;lmt_Ph*=2;
Pj=(double *)realloc(Pj,sizeof(double)*(lmt_Ph));
}
else if(flag==1){
old=lmt_Ps;lmt_Ps*=2;
Ps=(nP *)realloc(Ps,sizeof(nP)*(lmt_Ps));
if(Ps==NULL) {printf(" ?Ps:r\n");return 1;}
Qs=(nP *)realloc(Qs,sizeof(nP)*(lmt_Ps));
if(Qs==NULL) {printf(" ?Qs:r\n");return 1;}
```

```
for(i=old;i<lmt_Ps;i++){
Ps[i].n=0;Ps[i].P=0;
Qs[i].n=0;Qs[i].P=0;
}
}
else{
old=lmt_Psum;lmt_Psum*=2;
Psum=(double *)realloc(Psum,sizeof(double)*(lmt_Psum));
if(Psum==NULL) {printf(" ?Psum:r\n");return 1;}
Qsum=(double *)realloc(Qsum,sizeof(double)*(lmt_Psum));
if(Qsum==NULL) {printf(" ?Qsum:r\n");return 1;}

for(i=old;i<lmt_Psum;i++){
Psum[i]=0;
Qsum[i]=0;
}
}

return 0;
}/** reallocs **/


void frees(void)
{
int j;

/*for(j=0;j<lmt_Ph;j++){
if(Pji[j]!=NULL) free(Pji[j]);
if(Qji[j]!=NULL) free(Qji[j]);
}
if(Pji!=NULL) free(Pji);
if(Qji!=NULL) free(Qji);*/

if(Ph!=NULL) free(Ph);
if(Qh!=NULL) free(Qh);
if(Pi!=NULL) free(Pi);
if(Qi!=NULL) free(Qi);
if(Pj!=NULL) free(Pj);
if(Qj!=NULL) free(Qj);

if(Ps!=NULL) free(Ps);
if(Qs!=NULL) free(Qs);

if(Psum!=NULL) free(Psum);
if(Qsum!=NULL) free(Qsum);
}/** frees **/
```

```
double Phij(int h,int i,int j,double vh,double vi,double vj,int RW)
{
if(RW==0){
return Ph[h]*Pi[i]*Pj[j];
}
else{
Ph[h]=vh;Pi[i]=vi;Pj[j]=vj;
}

return vh*vi*vj;
}/** Phij **/


double Qhij(int h,int i,int j,double vh,double vi,double vj,int RW)
{
if(RW==0){
return Qh[h]*Qi[i]*Qj[j];
}
else{
Qh[h]=vh;Qi[i]=vi;Qj[j]=vj;
}

return vh*vi*vj;
}/** Qhij **/


int get_Ps_S(void)
{
int h,i,j,k,endflag=0;
int score,ra_,rb_,rc_,rd_;
double val[8],wal[8];

ra_=ra*mgf;rb_=rb*mgf;rc_=rc*mgf;rd_=rd*mgf;

for(j=0;j<lmt_Psum;j++){
Psum[j]=0;
Qsum[j]=0;
}
for(j=0;j<lmt_Ps;j++){
Ps[j].n=0;Ps[j].P=0;
Qs[j].n=0;Qs[j].P=0;
}

val[4]=0;val[5]=0;val[6]=0;
```

```
wal[4]=0;wal[5]=0;wal[6]=0;

for(h=0;h<Umax+1;h++){
val[1]=exp(-lmd1a)*pow(lmd1a,h)/times(h);  /* Ps:7 */
wal[1]=exp(-lmd2a)*pow(lmd2a,h)/times(h);  /* Ps:7 */
val[7]+=val[1]*val[1];
wal[7]+=wal[1]*wal[1];

if(1){
/* LK is Poisson */

for(i=0;i<=LKmax;i++)
for(j=0;j<=MKmax;j++)
for(k=0;k<=NKmax;k++){
Phij(i,j,k,0,0,0,WRITE);
Qhij(i,j,k,0,0,0,WRITE);
}

if((int)(rb*mgf)==0 && (int)(rc*mgf)==0 && (int)(rd*mgf)==0){
/*Pji[0][0]=val[1];Qji[0][0]=wal[1];*/
Phij(0,0,0,val[1],1,1,WRITE);
Qhij(0,0,0,wal[1],1,1,WRITE);
}
else if((int)(rb*mgf)==0 && (int)(rc*mgf)==0){
for(k=0;k<=NKmax;k++){
val[4]=exp(-lmd1d)*pow(lmd1d,k)/times(k);  /* Ps:rd */
wal[4]=exp(-lmd2d)*pow(lmd2d,k)/times(k);  /* Ps:rd */
Phij(0,0,k,1,1,val[1]*val[4],WRITE);
Qhij(0,0,k,1,1,wal[1]*wal[4],WRITE);
}
}
else if((int)(rb*mgf)==0 && (int)(rd*mgf)==0){
for(j=0;j<=MKmax;j++){
val[3]=exp(-lmd1c)*pow(lmd1c,j)/times(j);  /* Ps:rc */
wal[3]=exp(-lmd2c)*pow(lmd2c,j)/times(j);  /* Ps:rc */
Phij(0,j,0,1,val[1]*val[3],1,WRITE);
Qhij(0,j,0,1,wal[1]*wal[3],1,WRITE);
}
}
else if((int)(rc*mgf)==0 && (int)(rd*mgf)==0){
for(i=0;i<=LKmax;i++){
val[2]=exp(-lmd1b)*pow(lmd1b,i)/times(i);  /* Ps:rb */
wal[2]=exp(-lmd2b)*pow(lmd2b,i)/times(i);  /* Ps:rb */
Phij(i,0,0,val[1]*val[2],1,1,WRITE);
Qhij(i,0,0,wal[1]*wal[2],1,1,WRITE);
}
```

```
}
else if((int)(rb*mgf)==0){
for(j=0;j<=MKmax;j++)
for(k=0;k<=NKmax;k++){
val[3]=exp(-lmd1c)*pow(lmd1c,j)/times(j);  /* Ps:rc */
wal[3]=exp(-lmd2c)*pow(lmd2c,j)/times(j);  /* Ps:rc */
val[4]=exp(-lmd1d)*pow(lmd1d,k)/times(k);  /* Ps:rd */
wal[4]=exp(-lmd2d)*pow(lmd2d,k)/times(k);  /* Ps:rd */
Phij(0,j,k,1,val[1]*val[3],val[4],WRITE);
Qhij(0,j,k,1,wal[1]*wal[3],wal[4],WRITE);
}
}
else if((int)(rc*mgf)==0){
for(i=0;i<=LKmax;i++)
for(k=0;k<=NKmax;k++){
val[2]=exp(-lmd1b)*pow(lmd1b,i)/times(i);  /* Ps:rb */
wal[2]=exp(-lmd2b)*pow(lmd2b,i)/times(i);  /* Ps:rb */
val[4]=exp(-lmd1d)*pow(lmd1d,k)/times(k);  /* Ps:rd */
wal[4]=exp(-lmd2d)*pow(lmd2d,k)/times(k);  /* Ps:rd */
Phij(i,0,k,val[1]*val[2],1,val[4],WRITE);
Qhij(i,0,k,wal[1]*wal[2],1,wal[4],WRITE);
}
}
else if((int)(rd*mgf)==0){
for(i=0;i<=LKmax;i++)
for(j=0;j<=MKmax;j++){
val[2]=exp(-lmd1b)*pow(lmd1b,i)/times(i);  /* Ps:rb */
wal[2]=exp(-lmd2b)*pow(lmd2b,i)/times(i);  /* Ps:rb */
val[3]=exp(-lmd1c)*pow(lmd1c,j)/times(j);  /* Ps:rc */
wal[3]=exp(-lmd2c)*pow(lmd2c,j)/times(j);  /* Ps:rc */
Phij(i,j,0,val[1]*val[2],val[3],1,WRITE);
Qhij(i,j,0,wal[1]*wal[2],wal[3],1,WRITE);
}
}
else{
for(i=0;i<=LKmax;i++)
for(j=0;j<=MKmax;j++)
for(k=0;k<=NKmax;k++){
val[2]=exp(-lmd1b)*pow(lmd1b,i)/times(i);  /* Ps:rb */
wal[2]=exp(-lmd2b)*pow(lmd2b,i)/times(i);  /* Ps:rb */
val[3]=exp(-lmd1c)*pow(lmd1c,j)/times(j);  /* Ps:rc */
wal[3]=exp(-lmd2c)*pow(lmd2c,j)/times(j);  /* Ps:rc */
val[4]=exp(-lmd1d)*pow(lmd1d,k)/times(k);  /* Ps:rd */
wal[4]=exp(-lmd2d)*pow(lmd2d,k)/times(k);  /* Ps:rd */
Phij(i,j,k,val[1]*val[2],val[3],val[4],WRITE);
Qhij(i,j,k,wal[1]*wal[2],wal[3],wal[4],WRITE);
```

```
}
}

for(i=0;i<=LKmax;i++){
for(j=0;j<=MKmax;j++){
for(k=0;k<=NKmax;k++){
val[6]+=Phij(i,j,k,-1,-1,-1,READ)*Phij(i,j,k,-1,-1,-1,READ);
wal[6]+=Qhij(i,j,k,-1,-1,-1,READ)*Qhij(i,j,k,-1,-1,-1,READ);

while(1){
if(h+i+j>=lmt_Psum){
printf(" ?m_Psum:%d\n",lmt_Psum-(h+i+j));  /* ?m */
if(reallocs(2)) {endflag=1;goto next_mem;}
}
else break;
}

Psum[h+i+j+k]+=Phij(i,j,k,-1,-1,-1,READ);
Qsum[h+i+j+k]+=Qhij(i,j,k,-1,-1,-1,READ);

score=ra_*h+rb_*i+rc_*j+rd_*k;            /* important technique */
while(1){
if(score>=lmt_Ps){
printf(" ?m_Ps:%d\n",lmt_Ps-score);  /* ?m */
if(reallocs(1)) {endflag=1;goto next_mem;}
}
else break;
}

/*Ps[score].h=h;
Ps[score].i=i;
Ps[score].j=j;*/
if(Phij(i,j,k,-1,-1,-1,READ)>0) Ps[score].n++;
if(Qhij(i,j,k,-1,-1,-1,READ)>0) Qs[score].n++;
Ps[score].P+=Phij(i,j,k,-1,-1,-1,READ);
Qs[score].P+=Qhij(i,j,k,-1,-1,-1,READ);

val[5]+=Phij(i,j,k,-1,-1,-1,READ);
wal[5]+=Qhij(i,j,k,-1,-1,-1,READ);
if(val[5]>break_9s) {/*printf(" OK\n");*/goto next_S;}
if(wal[5]>break_9s) {/*printf(" OK\n");*/goto next_S;}
}/**for(k)**/
}/**for(j)**/
}/**for(i)**/
}/**if**/
else{
```

```
}/**else**/

/*projection(0+h*width,0,val[0]*mpl,&xs,&ys);
ellipse(xs,ys,-2,2,0,0,9);*/
}/**for(h)**/

next_S:
printf(" Soccer:h=%d",h);
/*printf(" draw_U=%f draw_min=%f",val[7],val[6]);*/
next_mem:

return endflag;
}/** get_Ps_S **/


int get_Ps_R(void)
{
int i,j,endflag=0;
int h,score,ra_,rb_,rc_;
double val[7],wal[7];

ra_=ra*mgf;rb_=rb*mgf;rc_=rc*mgf;

for(j=0;j<lmt_Psum;j++){
Psum[j]=0;
Qsum[j]=0;
}
for(j=0;j<lmt_Ps;j++){
Ps[j].n=0;Ps[j].P=0;
Qs[j].n=0;Qs[j].P=0;
}

val[4]=0;val[5]=0;val[6]=0;
wal[4]=0;wal[5]=0;wal[6]=0;

for(h=0;h<Umax+1;h++){
val[1]=exp(-lmd1a)*pow(lmd1a,h)/times(h);  /* Ps:7 */
wal[1]=exp(-lmd2a)*pow(lmd2a,h)/times(h);  /* Ps:7 */
val[6]+=val[1]*val[1];
wal[6]+=wal[1]*wal[1];

if(0){
}/**if**/
else{
if(h>=lmt_Ph){
printf(" ?m_Pj:%d\n",lmt_Ph-h);     /* ?m */
```

```
if(reallocs(0)) {endflag=1;goto next_mem;}
}

for(i=0;i<lmt_Ph;i++)
for(j=0;j<=MKmax;j++){
Phij(i,j,-0,0,0,0,WRITE);
Qhij(i,j,-0,0,0,0,WRITE);
}

if((int)(rb*mgf)==0 && (int)(rc*mgf)==0){
/*Pji[0][0]=val[1];Qji[0][0]=wal[1];*/
Phij(0,0,-0,val[1],1,1,WRITE);
Qhij(0,0,-0,wal[1],1,1,WRITE);
}
else if((int)(rb*mgf)==0){
for(j=0;j<=MKmax;j++){
val[3]=exp(-lmd1c)*pow(lmd1c,j)/times(j);  /* Ps:rc */
wal[3]=exp(-lmd2c)*pow(lmd2c,j)/times(j);  /* Ps:rc */
/*Pji[0][j]=val[1]*val[3];
Qji[0][j]=wal[1]*wal[3];*/
Phij(0,j,-0,1,val[1]*val[3],1,WRITE);
Qhij(0,j,-0,1,wal[1]*wal[3],1,WRITE);
}
}
else if((int)(rc*mgf)==0){
for(i=0;i<=h;i++){
val[2]=(times(h)/(times(h-i)*times(i)))*pow(p1b,i)*pow(1-p1b,h-i);  /* B */
wal[2]=(times(h)/(times(h-i)*times(i)))*pow(p2b,i)*pow(1-p2b,h-i);  /* B */
/*Pji[i][0]=val[1]*val[2];
Qji[i][0]=wal[1]*wal[2];*/
Phij(i,0,-0,val[1]*val[2],1,1,WRITE);
Qhij(i,0,-0,wal[1]*wal[2],1,1,WRITE);
}
}
else{
for(i=0;i<=h;i++)
for(j=0;j<=MKmax;j++){
val[2]=(times(h)/(times(h-i)*times(i)))*pow(p1b,i)*pow(1-p1b,h-i);  /* B */
wal[2]=(times(h)/(times(h-i)*times(i)))*pow(p2b,i)*pow(1-p2b,h-i);  /* B */
val[3]=exp(-lmd1c)*pow(lmd1c,j)/times(j);  /* Ps:rc */
wal[3]=exp(-lmd2c)*pow(lmd2c,j)/times(j);  /* Ps:rc */
Phij(i,j,-0,val[1]*val[2],val[3],1,WRITE);
Qhij(i,j,-0,wal[1]*wal[2],wal[3],1,WRITE);
}
}
```

```
for(i=0;i<=h;i++){
for(j=0;j<=MKmax;j++){
val[5]+=Phij(i,j,-0,-1,-1,-1,READ)*Phij(i,j,-0,-1,-1,-1,READ);
wal[5]+=Qhij(i,j,-0,-1,-1,-1,READ)*Qhij(i,j,-0,-1,-1,-1,READ);

while(1){
if(h+i+j>=lmt_Psum){
printf(" ?m_Psum:%d\n",lmt_Psum-(h+i+j));  /* ?m */
if(reallocs(2)) {endflag=1;goto next_mem;}
}
else break;
}

Psum[h+i+j]+=Phij(i,j,-0,-1,-1,-1,READ);
Qsum[h+i+j]+=Qhij(i,j,-0,-1,-1,-1,READ);

score=ra_*h+rb_*i+rc_*j;            /* important technique */
while(1){
if(score>=lmt_Ps){
printf(" ?m_Ps:%d\n",lmt_Ps-score);  /* ?m */
if(reallocs(1)) {endflag=1;goto next_mem;}
}
else break;
}

if(Phij(i,j,-0,-1,-1,-1,READ)>0) Ps[score].n++;
if(Qhij(i,j,-0,-1,-1,-1,READ)>0) Qs[score].n++;
Ps[score].P+=Phij(i,j,-0,-1,-1,-1,READ);
Qs[score].P+=Qhij(i,j,-0,-1,-1,-1,READ);

val[4]+=Phij(i,j,-0,-1,-1,-1,READ);
wal[4]+=Qhij(i,j,-0,-1,-1,-1,READ);
if(val[4]>break_9s) goto next_R;
if(wal[4]>break_9s) goto next_R;
}/**for(j)**/
}/**for(i)**/
}/**else**/

/*projection(0+h*width,0,val[0]*mpl,&xs,&ys);
ellipse(xs,ys,-2,2,0,0,9);*/
}/**for(h)**/

next_R:
printf(" Rugby:h=%d",h);
/*printf(" draw_U=%f draw_min=%f",val[6],val[5]);*/
next_mem:
```

```
return endflag;
}/** get_Ps_R **/


void figs(void)
{
int i,j,dflag,d,fnum;
double lmd,val[7];

#ifndef Rugby
fnum=1;
#else
fnum=2;
#endif

if(mallocs()) goto end;

if(abs(fnum)==1){
if(get_Ps_S()) goto end;

if(1){
lmd=lmd1a+lmd1b+lmd1c+lmd1d;
val[0]=0;
for(i=0;i<lmt_Psum;i++){
/*projection(0+i*width,0,Psum[i]*mpl,&xs,&ys);
ellipse(xs,ys,-2,2,0,0,10);*/
/*val[1]=exp(-lmd)*pow(lmd,i)/times(i);
printf("  t=%2d  %f  %f\n",i,sum[i],val[1]);*/
val[0]+=Psum[i];
if(val[0]>break9s) break;
}
printf(" t=%d S_Psum=%f\n",i,val[0]);
}

if(1){
val[0]=0;val[1]=0;val[2]=0;val[3]=0;val[4]=0;
for(i=0;i<lmt_Ps;i++){
val[0]+=Ps[i].P;
if(Ps[i].P>0) val[1]+=i*Ps[i].P;
if(Ps[i].P>val[3]) {val[2]=i;val[3]=Ps[i].P;}
val[4]+=Ps[i].P*Qs[i].P;
if(val[0]>break_9s) break;
}
printf(" draw=%f\n",val[4]);
printf(" mean=%f mode=%f(P=%f)\n",val[1]/mgf,val[2]/mgf,val[3]);
```

```
val[6]=val[1];

val[0]=0;/*val[1]=0;*/val[2]=0;
for(i=0;i<lmt_Ps;i++){
val[0]+=Ps[i].P;
val[2]+=pow((val[6]-i)/mgf,2)*Ps[i].P;
if(val[0]>break_9s) break;
}
printf(" vrc=%f\n",val[2]);
}

if(0){
val[0]=0;
for(i=0;i<lmt_Ps;i++){
val[0]+=Ps[i].P;
/*if(Po[i].n>0) printf("  %3d  %2d  %f\n",(int)(i/(double)mgf),Po[i].n,Po[i].P);*/
if(val[0]>break9s) break;
}
printf(" S_Ps[].P=%f at %d\n",val[0],i/(double)mgf);
}

if(1){
val[0]=0;val[1]=0;val[2]=0;val[3]=0;
 dflag=-1;
d=1;
for(i=0;i<lmt_Ps;i++){
val[0]+=Ps[i].P;

/*99*/
if(dflag==0){

}/**if()**/
else if(dflag==1){

j=i-d;
if(i>=d) val[1]+=Ps[i].P*Qs[j].P;
}/**else if()**/
else{

j=i+d;
if(i>=0) val[1]+=Ps[i].P*Qs[j].P;
}/**else**/

if(val[0]>break_9s) break;
}
```

```
/*printf(" w=%f l=%f wld=%f\n",val[1],val[2],val[1]+val[2]+val[4]);*/
if(dflag==0)      val[0]=val[4];
else if(dflag==1) val[0]=val[1];
else              val[0]=val[1];
printf(" mnt=%f d=%d p(d)=%f\n",(double)mnt,dflag*d,val[0]);
}

printf(" mean_=%f\n",(double)(lmd1a*ra+lmd1b*rb+lmd1c*rc+lmd1d*rd));
printf(" ra=%f rb=%f rc=%f rd=%f\n",(double)ra,(double)rb,(double)rc,(double)rd);
}/**else if(fnum)**/
else if(abs(fnum)==2){
/*999*/
/* Rugby ************************************************/
if(get_Ps_R()) goto end;

if(1){
lmd=lmd1a+lmd1b+lmd1c+lmd1d;
val[0]=0;
for(i=0;i<lmt_Psum;i++){
/*projection(0+i*width,0,Psum[i]*mpl,&xs,&ys);
ellipse(xs,ys,-2,2,0,0,10);*/
/*val[1]=exp(-lmd)*pow(lmd,i)/times(i);
if((int)(rb*mgf)==0)
printf("  t=%2d  %f  %f\n",i,sum[i],val[1]);*/
val[0]+=Psum[i];
if(val[0]>break9s) break;
}
printf(" t=%d S_Psum=%f\n",i,val[0]);
}

if(1){
val[0]=0;val[1]=0;val[2]=0;val[3]=0;val[4]=0;
for(i=0;i<lmt_Ps;i++){
val[0]+=Ps[i].P;
if(Ps[i].P>0) val[1]+=i*Ps[i].P;
if(Ps[i].P>val[3]) {val[2]=i;val[3]=Ps[i].P;}
val[4]+=Ps[i].P*Qs[i].P;
if(val[0]>break_9s) break;
}
printf(" draw=%f\n",val[4]);
printf(" mean=%f mode=%f(P=%f)\n",val[1]/mgf,val[2]/mgf,val[3]);

val[6]=val[1];

val[0]=0;/*val[1]=0;*/val[2]=0;
for(i=0;i<lmt_Ps;i++){
```

```
val[0]+=Ps[i].P;
val[2]+=pow((val[6]-i)/mgf,2)*Ps[i].P;
if(val[0]>break_9s) break;
}
printf(" vrc=%f\n",val[2]);
}

if(0){
val[0]=0;
for(i=0;i<lmt_Ps;i++){
val[0]+=Ps[i].P;
/*if(Po[i].n>0) printf("  %3d  %2d  %f\n",(int)(i/(double)mgf),Po[i].n,Po[i].P);*/
if(val[0]>break9s) break;
}
printf(" S_Ps[].P=%f at %d\n",val[0],i/(double)mgf);
}

if(1){
val[0]=0;val[1]=0;val[2]=0;val[3]=0;
 dflag=-1;
d=2;
for(i=0;i<lmt_Ps;i++){
val[0]+=Ps[i].P;

/*99*/
if(dflag==0){

}/**if()**/
else if(dflag==1){

j=i-d;
if(i>=d) val[1]+=Ps[i].P*Qs[j].P;
}/**else if()**/
else{

j=i+d;
if(i>=0) val[1]+=Ps[i].P*Qs[j].P;
}/**else**/

if(val[0]>break_9s) break;
}
/*printf(" w=%f l=%f wld=%f\n",val[1],val[2],val[1]+val[2]+val[4]);*/
if(dflag==0)      val[0]=val[4];
else if(dflag==1) val[0]=val[1];
else              val[0]=val[1];
printf(" mnt=%f d=%d p(d)=%f\n",(double)mnt,dflag*d,val[0]);
```

```
}

printf(" mean_=%f\n",(double)(lmd1a*ra+(lmd1a*p1b)*rb+lmd1c*rc+lmd1d*rd));
printf(" ra=%f rb=%f rc=%f rd=%f\n",(double)ra,(double)rb,(double)rc,(double)rd);
}/**else if(fnum)**/
else if(abs(fnum)==3){


}/**else if(fnum)**/

end:
frees();
}/** figs **/
```