Debreceni Egyetem
Természettudományi és Technológiai Kar
Matematikai Intézet

Szakdolgozat

Interaktív alkalmazások implementálása SageMath programcsomagba

készítette:

Félegyházi Dávid

témavezető: Dr. Tengely Szabolcs

Debrecen, 2018

Tartalomjegyzék

Köszönetnyilvánítás

Első körben szeretném megköszönni családomnak, hogy támogattak egész tanulmányaim alatt, illetve motiváltak a nehezebb napokon. Szeretném megköszönni a barátaimnak, akik a monoton munkából kizökkentettek és így tették könnyebbé mindennapjaim. Szeretném megköszönni a tanáraimnak, illetve csoporttársaimnak, hogy segítséget nyújtottak az egyetemi tanulmányaim alatt.
Ezenkívül szeretném megköszönni Dr. Tengely Szabolcsnak, hogy vállalta és segített engem, illetve Dsupin Krisztiánnak, aki javaslatokkal és tanácsokkal látott el a teljes munkám ideje alatt.

1. Bevezető

A szakdolgozatom témája a TeamRanking algoritmusokkal és azok matematikájával foglalkozik. A TeamRanking egy rangsorolt táblázat, amely a csapatokat hasonlítja össze és egy rangsort állít fel, a csapat sikerességétől függően. A szakdolgozatomhoz a Bajnokok ligája 2016-2017-es kiírását használtam fel, mivel a felépítésében megtalálható a tabella alapú, illetve kieséses alapú verseny egyaránt. Az algoritmusokat a SageMath nevű matematikai programcsomagba implementáltam. A szakdolgozatban a Colley-módszerrel, a Massey-módszerrel, illetve a Keener-módszerrel foglalkozok. Elsőnek az adatok feldolgozásával kezdtem. A 32 csapatnak, illetve a helyszínnek és a végkimenetelnek adtam egy-egy kódot, amely így néz ki:

Helyszín:1 Hazai 2 Vendég 3 Semleges

Eredmény:1 Győzelem 2 Vereség 3 Döntetlen

Csapatok:

01: Club Brugge	02: Leicester	03: FC Porto
04: FC Kovenhavn	05: Juventus	06: Sevilla
07: Legia Warszawa	08: Borussia Dortmund	09: Bayer Leverkusen
10: CSZKA Moszkva	11: Lyon	12: Dinamo Zagreb
13: Manchester City	14: Monchengladbach	15: Real Madrid
16: Sporting CP	17: Tottenham	18: Monaco
19: Barcelona	20: Celtic	21: Basel
22: Ludogorets	23: Bayern Munchen	24: FK Rosztov
25: Benfica	26: Besiktas	27: Dinamo Kijev
28: Napoli	29: PSG	30: Arsenal
31: PSV Eindhoven	32: Atletico Madrid	

Ezenkívül pedig minden egyes meccsről készítettem két vektort, amely az alábbi elemeket tartalmazza: [Csapat, Ellenfél, Helyszín, Eredmény, Rúgott gól, Kapott gól, Kapuralövés, Kaput találó lövés, Szöglet, Passz, Labdabirtoklás, Szabálytalanság, Sárga lap, Piros lap].

2. Colley-módszer

A legrégebbi értékelő (rating) módszer a százalékos módszer volt, ahol kiszámolták, hogy melyik csapat hány meccset nyert és mennyi volt az összes meccsek száma, és ez alapján értékelték őket. Viszont Colley módszerében [1] a kezdőérték nem 0, ahogy a százalékos értékelésnél lehet, hanem 1/2. Ezzel a aránnyal Colley meg tudta állapítani a csapatok erejét. Meg fogom mutatni, hogy hogyan értékel a Colley-módszer a különböző tabellaeredmények alapján.

2.1. A példa

Vegyük példának a bevezetőben megemlített Bajnokok Ligájának a selejtezőben lévő A-csoportot, amelyre fogok hivatkozni a jövőben: az Arsenal, a PSG, a Basel FC, és a Ludogorets meccseit tartalmazza, ahol az bal oszlopban lévő csapat a hazait jelöli, a felső sorban lévő együttes pedig a vendéget. A meccset végeredménye az alábbi módon alakult:

	Basel	Ludogorets	PSG	Arsenal
Basel	x	1-1	1-2	1-4
Ludogorets	1-3	x	2-3	0-0
PSG	1-1	3-0	x	2-2
Arsenal	2-0	6-0	2-2	x

Ha megfigyeljük a táblázatot, akkor könnyen fel tudjuk sorakoztatni a csapatokat a győzelmeik száma alapján:

1. Arsenal
2. PSG
3. Basel
4. Ludogorets

Az Arsenalnak 4 győzelme volt, a PSG-nek 3, míg a Baselnek és a Ludogoretsnek nem volt egy győzelmük sem.

2.2. A Colley-módszer matematikája

A Colley-módszer egy $Cr = b$ lineáris egyenlet megoldásán alapszik. A C egy $\mathscr{M}_{n\times n}$-es diagonális mátrix, amelynek az a_{ij} eleme $-n$, ahol az n az egymás ellen lejátszott meccsek számát jelenti, a mi esetünkben -2-t, mivel a csapatok oda-vissza játszottak egy-egy meccset. A főátlóban pedig 2+ az összes játszott meccs száma lesz. Mivel 4 csapatunk van, és összesen 6 meccset játszottak, így a C mátrix az alábbi:

$$\begin{pmatrix} 8 & -2 & -2 & -2 \\ -2 & 8 & -2 & -2 \\ -2 & -2 & 8 & -2 \\ -2 & -2 & -2 & 8 \end{pmatrix}$$

A b eredményvektor pedig az alábbi formulából áll: $b_i = 1 + \frac{1}{2}(w_i - l_i)$, ahol a w_i a győztes meccsek számát, az l_i pedig az elvesztett meccsek számát jelöli. Ez alapján könnyen meg tudjuk határozni a r vektort, amelynél az $r_i \in [0, 1]$. Tehát ha kiszámoljuk a $Cr = b$ egyenletet, akkor r-re az alábbi vektort kapjuk:

$$\begin{pmatrix} 0.700 \\ 0.650 \\ 0.350 \\ 0.300 \end{pmatrix}$$

2.3. Előnyei és hátrányai

A módszer egyik legnagyobb előnye, hogy olyan versenyeken, ahol csak győzelem vagy vereség van, könnyen lehet használni. Viszont az algoritmus nem tudja kezelni azon sporteseményeket, ahol a döntetlen lehetősége is fennáll. Például a B csoportban a Benfica és a Besiktas értékei megegyeznek, pedig a Benfica szerzett végül több pontot a táblázatban. A másik nagy hátránya, hogy az algoritmus nem tud hiányos táblázattal foglalkozni, ami azt jelenti,

hogy a Bajnokok Ligájában azon meccseknél nem tudjuk felhasználni az algoritmust, amelyek már a kieséses alapon működtek. Ezáltal a lehetőségek le vannak korlátozva csak tabella alapú eseményekre. Az utolsó komoly gondja ennek, hogy az algoritmus nem értékeli a különböző csapatok elleni egyéb statisztikákat. Például, ha egy gyengébb csapat legyőzz egy erősebb csapatot, azt a metódus ugyanolyan győzelemnek tulajdonítja, mint amikor egy erősebb győz le egy gyengébbet. Ezt a problémát próbálja orvosolni egy fejlesztett Colley-módszer.

2.4. A fejlesztett Colley-módszer

A fejlesztett Colley-módszer kitér a csapatok közötti erőviszonyokra is. Mivel a Colley-módszer esetében az nem volt számottevő, hogy ki hány góllal nyer, vagy mennyi kapura lövése volt, így nem tudjuk mindig rangsorolni, hogy melyik csapat volt legvégül a legerősebb. Erre ad megoldást a fejlesztett Colley-módszer. Az alapja ennek is egy $Cr = b$ lineáris egyenlet, de itt a b vektort máshogy választjuk meg. Különböző, úgynevezett súlyzószorzókat állítunk be, különböző értékekhez, amelyeket módosítva egy más érték jön ki. A kezdeti vektorokban feltüntettem ezeket az értékeket, ennek a jelölése a következő:

- WH:Győzelem otthon
- WA:Győzelem idegenben
- TIE:Döntetlen
- LOSEHOME:Vereség otthon
- LOSEAWAY:Vereség idegenben
- RG:Rúgott gól
- KG:Kapott gól
- KL:Kapuralövés
- KTL:Kaput találó lövés

- CO:Szöglet
- PASS:Passz
- POSS:labdabirtoklás,50 százalékhoz viszonyítva
- SZAB:Szabálytalanság
- YC:Sárga lap
- RC:Piros lap

Ezáltal a b vektor a következő lesz:
$b_i = 1 + \frac{1}{2}(win + c_1 * RG + c_2 * KG + c_3 * KL + c_4 * KTL + c_5 * CO + c_6 * PASS + (c_7 - 50) * POSS - c_8 * SZAB - c_9 * YC - c_{10} * RC)$, ahol a c_i az adatbázis vektorainak különböző értékeit jelöli illetve a $win = k_1 * WH + k_2 * WA + k_3 * TIE - k_4 * LOSE$, ahol k_i a különböző meccsek kimeneteleinek a számát adja meg.
Ezáltal az A csoportban a b vektor:

$$\begin{pmatrix} -1106 \\ -992 \\ 2953.5 \\ 2986 \end{pmatrix}$$

Illetve a rangsoroló vektor:

$$\begin{pmatrix} 273.550 \\ 284.950 \\ 679.550 \\ 682.750 \end{pmatrix}$$

Itt már sokkal látványosabban elkülönülnek azok a csapatok, akik több pontot szereztek azoktól, akik csak néhányat, viszont az súlyzószorzó módosításával érdekesebb dolgokat is kiszűrhetünk. Például ha nagyon magasra állítjuk a sárga lapokért, piros lapokért, illetve a szabálytalanságokért járó büntetőpontokat, és minden mást pedig nullára, akkor egy rangsort kaphatunk a legszabályosabban játszó csapatokról. Viszont ennek a módszernek is ugyanazok a hibái, mint az alap Colley-módszernek: nem értékeli, hogy ki ellen játszott a csapat, nem alkalmazható hiányos táblázattal, illetve a döntetleneket sem kezeli rendesen.

2.5. Program és programkód

A módszerekhez a SageMath [5] matematikai programcsomagban készítettünk interaktív alkalmazást. A programkód:

```
BL1617= matrix(QQ,[
[1,2,1,2,0,3,11,2,5,585,59,7,4,0],[2,1,2,1,3,0,10,7,4,348,41,10,2,0],
[3,4,1,3,1,1,10,2,6,596,56,5,1,0],[4,3,2,3,1,1,6,2,3,470,44,11,2,1],
[5,6,1,3,0,0,16,4,4,519,51,16,0,0],[6,5,2,3,0,0,3,1,2,417,49,22,3,0],
[7,8,1,2,0,6,7,2,8,284,36,12,3,0],[8,7,2,1,6,0,30,15,7,627,64,9,1,0],
[9,10,1,3,2,2,18,9,4,679,62,5,2,0],[10,9,2,3,2,2,8,4,5,336,38,11,0,0],
[11,12,1,1,3,0,22,10,7,534,55,9,1,0],[12,11,2,2,0,3,14,2,6,391,45,7,3,0],
[13,14,1,1,4,0,22,11,6,561,58,8,1,0],[14,13,2,2,0,4,3,1,6,415,42,14,1,0],
[15,16,1,1,2,1,15,5,5,628,59,9,1,0],[16,15,2,2,1,2,9,4,2,431,41,14,3,0],
[17,18,1,2,1,2,15,5,11,582,58,11,1,0],[18,17,2,1,2,1,7,2,0,336,42,4,2,0],
[19,20,1,1,7,0,15,9,9,852,67,8,2,0],[20,19,2,2,0,7,4,2,2,324,33,3,1,0],
[21,22,1,3,1,1,13,5,1,560,57,26,1,0],[22,21,2,3,1,1,4,2,0,432,43,17,2,0],
[23,24,1,1,5,0,19,8,8,797,72,13,0,0],[24,23,2,2,0,5,5,2,2,247,28,12,0,0],
[25,26,1,3,1,1,14,6,6,416,47,17,2,0],[26,25,2,3,1,1,12,5,5,470,53,11,2,0],
[27,28,1,2,1,2,9,2,8,551,54,13,3,1],[28,27,2,1,2,1,14,2,6,580,46,17,1,0],
[29,30,1,3,1,1,14,5,11,581,56,10,3,1],[30,29,2,3,1,1,11,3,7,504,44,12,3,1],
[31,32,1,2,0,1,12,4,2,514,50,6,2,0],[32,31,2,1,1,0,16,3,4,603,50,10,2,0],
[30,21,1,1,2,0,18,7,8,787,61,4,0,0],[21,30,2,2,0,2,7,3,7,411,39,12,2,0],
[32,23,1,1,1,0,16,5,4,325,37,13,1,0],[23,32,2,2,0,1,13,4,6,692,63,17,4,0],
[26,27,1,3,1,1,14,5,10,373,50,18,2,0],[27,26,2,3,1,1,12,4,4,358,50,19,3,0],
[20,13,1,3,3,3,8,3,4,359,43,8,0,0],[13,20,2,3,3,3,21,9,9,567,57,8,0,0],
[24,31,1,3,2,2,8,4,4,427,53,15,2,0],[31,24,2,3,2,2,9,4,3,344,47,12,2,0],
[22,29,1,2,1,3,8,4,0,445,41,9,3,0],[29,22,2,1,3,1,17,7,4,779,59,14,2,0],
[14,19,1,2,1,2,8,1,2,445,38,8,0,0],[19,14,2,1,2,1,11,8,5,683,62,10,3,0],
[28,25,1,1,4,2,14,7,6,624,55,7,1,0],[25,28,2,2,2,4,7,3,0,505,45,19,4,0],
[8,15,1,3,2,2,18,11,6,655,59,14,4,0],[15,8,2,3,2,2,11,4,6,446,41,11,1,0],
[10,17,1,2,0,1,8,2,4,341,35,5,1,0],[17,10,2,1,1,0,22,2,8,698,65,6,1,0],
[12,5,1,2,0,4,4,0,0,299,34,10,0,0],[5,12,2,1,4,0,16,6,6,808,66,13,0,0],
[4,1,1,1,4,0,17,10,7,534,58,14,1,0],[1,4,2,2,0,4,6,2,1,347,42,14,3,0],
[2,3,1,1,1,0,6,3,2,314,41,24,3,0],[3,2,2,2,0,1,12,4,6,493,59,16,3,0],
[18,9,1,3,1,1,6,1,6,468,55,12,1,0],[9,18,2,3,1,1,10,3,7,397,45,15,2,0],
[6,11,1,1,1,0,17,6,3,473,56,12,2,0],[11,6,2,2,0,1,15,4,6,307,44,24,3,0],
```

```
[16,7,1,1,2,0,15,5,8,535,55,10,0,0],[7,16,2,2,0,2,6,0,3,408,45,16,3,0],
[30,22,1,1,6,0,19,12,6,474,43,12,0,0],[22,30,2,2,0,6,10,3,1,622,57,5,0,0],
[19,13,1,1,4,0,12,8,3,449,54,15,2,1],[13,19,2,2,0,4,10,5,4,405,46,17,3,1],
[23,31,1,1,4,1,26,13,6,819,63,7,1,0],[31,23,2,2,1,4,7,4,0,385,37,5,0,0],
[20,14,1,2,0,2,9,1,4,626,58,9,1,0],[14,20,2,1,2,0,13,6,5,512,42,7,1,0],
[27,25,1,2,0,2,16,6,9,493,56,14,1,0],[25,27,2,1,2,0,12,5,1,419,44,15,1,0],
[24,32,1,2,0,1,3,3,0,317,34,11,1,0],[32,24,2,1,1,0,15,6,5,712,66,6,2,0],
[28,26,1,2,2,3,18,9,17,578,55,11,1,0],[26,28,2,1,3,2,8,3,1,351,45,16,3,0],
[29,21,1,1,3,0,13,6,3,711,64,14,1,0],[21,29,2,2,0,3,10,1,5,355,36,18,2,0],
[1,3,1,2,1,2,7,5,3,387,41,14,2,0],[3,1,2,1,2,1,20,8,8,556,59,9,1,0],
[10,18,1,3,1,1,8,3,2,383,42,14,1,0],[18,10,2,3,1,1,15,6,8,562,58,14,2,0],
[12,6,1,2,0,1,5,0,3,221,28,16,2,0],[6,12,2,1,1,0,12,5,5,846,72,13,2,0],
[2,4,1,1,1,0,10,2,3,384,45,8,1,0],[4,2,2,2,0,1,9,4,7,525,55,9,1,0],
[9,17,1,3,0,0,16,6,5,381,44,15,2,0],[17,9,2,3,0,0,5,1,5,493,56,20,1,0],
[11,5,1,2,0,1,13,4,10,425,44,15,5,0],[5,11,2,1,1,0,14,5,4,557,56,7,3,1],
[15,7,1,1,5,1,29,11,9,638,56,10,1,0],[7,15,2,2,1,5,12,5,3,407,44,8,2,0],
[16,8,1,2,1,2,10,5,2,395,41,10,4,0],[8,16,2,1,2,1,10,3,4,537,59,12,3,0],
[8,16,1,1,1,0,10,5,3,603,53,6,1,0],[16,8,2,2,0,1,10,2,2,518,47,18,3,0],
[4,2,1,3,0,0,20,2,4,530,57,3,0,0],[2,4,2,3,0,0,10,1,4,407,43,17,4,0],
[3,1,1,1,1,0,20,6,10,437,49,16,2,0],[1,3,2,2,0,1,11,4,4,459,51,17,4,0],
[5,11,1,3,1,1,10,2,2,592,52,14,4,0],[11,5,2,3,1,1,12,3,4,457,48,20,2,0],
[7,15,1,3,3,3,10,4,2,520,45,8,0,0],[15,7,2,3,3,3,28,11,7,690,55,16,0,0],
[18,10,1,1,3,0,20,7,6,655,60,8,1,0],[10,18,2,2,0,3,6,3,6,404,40,13,1,0],
[6,12,1,1,4,0,30,12,10,720,67,10,0,0],[12,6,2,2,0,4,3,1,0,268,33,12,4,1],
[17,9,1,2,0,1,9,2,4,533,59,12,0,0],[9,17,2,1,1,0,11,2,5,368,41,11,0,0],
[32,24,1,1,2,1,33,8,11,618,63,8,3,0],[24,32,2,2,1,2,5,4,1,292,37,11,4,0],
[21,29,1,2,1,2,9,2,3,353,36,14,3,1],[29,21,2,1,2,1,17,8,10,716,64,8,2,0],
[25,27,1,1,1,0,17,4,5,419,51,15,1,0],[27,25,2,2,0,1,9,2,6,394,49,15,6,0],
[22,30,1,2,2,3,8,4,4,452,46,10,1,0],[30,22,2,1,3,2,11,4,5,587,54,23,3,0],
[13,19,1,1,3,1,13,4,5,323,40,17,2,0],[19,13,2,2,1,3,8,2,7,592,60,16,3,0],
[14,20,1,3,1,1,18,5,9,547,47,10,2,1],[20,14,2,3,1,1,10,2,1,611,53,11,1,0],
[31,23,1,2,1,2,4,2,0,265,31,10,0,0],[23,31,2,1,2,1,23,9,7,777,69,8,1,0],
[26,28,1,3,1,1,7,2,3,480,48,8,2,0],[28,26,2,3,1,1,14,5,8,626,52,20,1,0],
[30,29,1,3,2,2,5,1,3,532,48,11,2,0],[29,30,2,3,2,2,15,3,6,629,52,9,2,0],
[32,31,1,1,2,0,11,4,9,672,54,6,0,0],[31,32,2,2,0,2,4,2,2,484,46,8,0,0],
[20,19,1,2,0,2,7,3,1,413,39,13,3,0],[19,20,2,1,2,0,9,4,5,723,61,12,4,0],
[22,21,1,3,0,0,5,0,3,491,55,12,3,0],[21,22,2,3,0,0,11,5,5,452,45,30,4,0],
[14,13,1,3,1,1,12,5,2,305,31,13,5,1],[13,14,2,3,1,1,11,7,9,709,69,10,2,1],
```

```
[28,27,1,3,0,0,17,5,17,513,48,18,1,0],[27,28,2,3,0,0,7,0,2,491,52,15,1,0],
[26,25,1,3,3,3,10,5,5,588,56,11,2,0],[25,26,2,3,3,3,15,4,2,446,44,13,2,0],
[24,23,1,1,3,2,7,4,3,194,29,9,2,0],[23,24,2,2,2,3,23,10,7,708,71,8,2,0],
[8,7,1,1,8,4,17,9,1,794,68,11,1,0],[7,8,2,2,4,8,11,7,1,320,32,11,2,0],
[12,11,1,2,0,1,8,3,1,361,37,9,3,0],[11,12,2,1,1,0,17,8,5,763,63,16,1,0],
[4,3,1,3,0,0,8,1,3,440,51,16,3,0],[3,4,2,3,0,0,16,5,5,412,49,13,3,0],
[2,1,1,1,2,1,13,9,11,464,51,5,1,0],[1,2,2,2,1,2,8,3,7,417,49,8,0,0],
[18,17,1,1,2,1,17,11,5,307,40,11,2,0],[17,18,2,2,1,2,7,2,7,559,60,14,3,0],
[6,5,1,2,1,3,6,1,5,359,45,13,4,1],[5,6,2,1,3,1,12,5,5,526,55,18,4,0],
[16,15,1,2,1,2,10,2,2,441,41,16,3,1],[15,16,2,1,2,1,11,2,8,643,59,9,0,2],
[10,9,1,3,1,1,13,5,5,478,54,14,2,0],[9,10,2,3,1,1,8,4,4,365,46,12,3,0],
[1,4,1,2,0,2,8,4,6,433,50,15,1,0],[4,1,2,1,2,0,7,5,4,459,50,12,1,0],
[3,2,1,1,5,0,18,8,4,613,64,8,0,0],[2,2,2,2,0,5,6,0,1,277,36,11,1,0],
[5,12,1,1,2,0,17,9,7,701,62,9,2,0],[12,5,2,2,0,2,3,2,3,405,38,12,1,0],
[7,16,1,1,1,0,10,4,2,349,41,14,5,0],[16,7,2,2,0,1,19,5,7,632,59,14,3,1],
[9,18,1,1,3,0,12,6,7,443,50,18,1,0],[18,9,2,2,0,3,12,0,5,472,50,13,3,0],
[11,6,1,3,0,0,24,5,8,570,58,10,2,0],[6,11,2,3,0,0,2,0,3,396,42,13,2,0],
[15,8,1,3,2,2,15,8,6,518,47,9,2,0],[8,15,2,3,2,2,12,7,5,608,53,15,1,0],
[17,10,1,1,3,1,23,12,11,724,67,7,0,0],[10,17,2,2,1,3,6,3,1,298,33,15,2,0],
[19,14,1,1,4,0,14,8,5,1023,68,7,0,0],[14,19,2,2,0,4,1,0,2,411,32,5,1,0],
[21,30,1,2,1,4,15,4,7,483,45,9,1,0],[30,21,2,1,4,1,12,7,2,728,55,13,1,0],
[23,32,1,1,1,0,14,4,6,826,68,11,0,0],[32,23,2,2,0,1,5,3,1,318,32,9,1,0],
[25,28,1,2,1,2,13,3,7,515,50,15,1,0],[28,25,2,1,2,1,14,7,2,559,50,8,1,0],
[27,26,1,1,6,0,17,13,1,569,53,12,1,0],[26,27,2,2,0,6,6,0,3,523,47,9,4,2],
[13,20,1,3,1,1,10,2,8,590,55,7,1,0],[20,13,2,3,1,1,15,6,6,448,45,16,2,0],
[29,22,1,3,2,2,27,7,19,616,60,10,1,0],[22,29,2,3,2,2,5,2,3,373,40,18,0,0],
[31,24,1,3,0,0,17,4,6,695,65,12,2,0],[24,31,2,3,0,0,6,3,1,332,35,8,1,0],
[32,9,1,3,0,0,13,4,7,420,45,6,2,0],[9,32,2,3,0,0,18,7,4,548,55,8,2,0],
[18,13,1,1,3,1,8,4,3,321,41,12,3,0],[13,18,2,2,1,3,6,3,6,534,59,13,3,0],
[5,3,1,1,1,0,15,3,4,775,63,13,1,0],[3,5,2,2,0,1,9,2,1,382,37,12,2,1],
[2,6,1,1,2,0,13,4,2,195,33,10,4,0],[6,2,2,2,0,2,13,5,10,531,67,10,3,1],
[19,29,1,1,6,1,17,7,6,591,66,16,5,0],[29,19,2,2,1,6,7,3,4,251,34,25,5,0],
[8,25,1,1,4,0,13,6,5,473,54,13,3,0],[25,8,2,2,0,4,5,2,5,389,46,17,1,0],
[30,23,1,2,1,5,9,3,5,413,44,13,3,2],[23,30,2,1,5,1,16,7,8,556,56,10,2,0],
[28,15,1,2,1,3,13,4,9,633,51,17,2,0],[15,28,2,1,3,1,13,5,4,530,49,8,0,0],
[3,5,1,2,0,2,3,0,1,232,30,12,5,1],[5,3,2,1,2,0,18,4,8,744,70,13,1,0],
[6,2,1,1,2,1,22,8,8,654,68,6,2,0],[2,6,2,2,1,2,8,3,1,234,32,9,2,0],
[9,32,1,2,2,4,14,3,6,550,59,14,4,0],[32,9,2,1,4,2,10,7,4,326,41,9,3,0],
```

```
[13,18,1,1,5,3,8,6,5,590,61,12,4,0],[18,13,2,2,3,5,15,6,6,275,39,15,6,0],
[23,30,1,1,5,1,24,11,9,767,69,12,2,0],[30,23,2,2,1,5,8,5,2,273,31,11,3,0],
[15,28,1,1,3,1,19,6,5,600,54,9,2,0],[28,15,2,2,1,3,8,2,1,632,46,12,2,0],
[25,8,1,1,1,0,4,1,3,303,36,15,1,0],[8,25,2,2,0,1,14,4,10,680,64,10,3,0],
[29,19,1,1,4,0,16,10,2,428,43,14,1,0],[19,29,2,2,0,4,6,1,4,591,57,11,3,0],
[19,5,1,3,0,0,17,1,13,587,61,10,2,0],[5,19,2,3,0,0,12,4,4,329,39,19,2,0],
[18,8,1,1,3,1,15,9,5,280,37,15,0,0],[8,18,2,2,1,3,12,7,3,661,63,8,0,0],
[2,32,1,3,1,1,21,4,7,488,51,9,0,0],[32,2,2,3,1,1,6,3,1,400,49,7,0,0],
[15,23,1,2,4,2,26,12,7,638,47,7,1,0],[23,15,2,1,2,4,19,2,10,692,53,22,5,1],
[32,2,1,1,1,0,15,4,8,632,64,10,0,0],[2,32,2,2,0,1,6,0,1,293,36,16,3,0],
[23,15,1,2,1,2,11,3,8,535,49,11,4,1],[15,23,2,1,2,1,22,12,4,589,51,7,2,0],
[8,18,1,2,2,3,15,4,7,644,64,9,2,0],[18,8,2,1,3,2,7,2,1,285,36,9,5,0],
[5,19,1,1,3,0,14,8,5,280,34,21,4,0],[19,5,2,2,0,3,15,3,8,614,66,18,3,0],
[32,15,1,1,2,1,17,7,7,320,40,23,4,0],[15,32,2,2,1,2,19,6,8,582,60,7,2,0],
[5,18,1,1,2,1,15,6,6,407,47,11,2,0],[18,5,2,2,1,2,11,2,8,400,53,21,2,0],
[18,5,1,2,0,2,14,6,7,448,52,11,1,0],[5,18,2,1,2,0,10,5,3,452,48,13,3,0],
[15,32,1,1,3,0,16,8,9,673,61,10,1,0],[32,15,2,2,0,3,4,1,3,422,39,15,3,0],
[5,15,2,2,1,4,11,4,1,418,44,23,5,1],[15,5,2,1,4,1,18,5,1,541,56,18,4,0]])

Codes=[['Club Brugge',1,7],['Leicester',2,7],['FC Porto',3,7],
    ['FC Kobenhavn',4,7],['Juventus',5,8],['Sevilla',6,8],
    ['Legia Warszawa',7,6],['Borussia Dortmund',8,6],['Bayer Leverkusen',9,5],
    ['CSZKA Moszkva',10,5],['Lyon',11,8],['Dinamo Zagreb',12,8],
    ['Machester City',13,3],['Monchengladbach',14,3],['Real Madrid',15,6],
    ['Sporting CP',16,6],['Tottenham',17,5],['Monaco',18,5],
    ['Barcelona',19,3],['Celtic',20,3],['Basel',21,1],
    ['Ludogorets',22,1],['Bayern Munchen',23,4],['FK Rosztov',24,4],
    ['Benfica',25,2],['Besiktas',26,2],['Dinamo Kijev',27,2],
    ['Napoli',28,2],['PSG',29,1],['Arsenal',30,1],
    ['PSV Eindhoven',31,4],['Atletico Madrid',32,4]]

html("<h1 align=center>Colley-módszer</h1>")
html("<h2 align=center>Alap formula</h2>")
C=matrix(QQ,[[8,-2,-2,-2],[-2,8,-2,-2],[-2,-2,8,-2],[-2,-2,-2,8]])
b=vector(QQ,C.ncols())
badv=vector(QQ,C.ncols())

def matches(groupcode):
```

```
137     A=[]
138     eredmeny=[]
139     ellenfel=''
140     for i in [0..len(Codes)-1]:
141         if Codes[i][2]==groupcode:
142             A.append([Codes[i][0],Codes[i][1]])
143     for i in [0..3]:
144         for j in [0..BL1617.nrows()-59]:
145             if A[i][1]==BL1617[j][0]:
146                 for k in [0..3]:
147                     if BL1617[j][1]==A[k][1]:
148                         ellenfel=A[k][0]
149                 if BL1617[j][2]==1:
150                      eredmeny.append([A[i][0],BL1617[j][4],
                      ↪  BL1617[j][5],ellenfel])
151     return eredmeny
152 
153 def basicColley(groupcode):
154     A=[]
155     eredmeny=[]
156     for i in [0..len(Codes)-1]:
157         if Codes[i][2]==groupcode:
158             A.append([Codes[i][0],Codes[i][1],0])
159     for i in [0..3]:
160         for j in [0..BL1617.nrows()-59]:
161             if A[i][1]==BL1617[j][0]:
162                 for k in [0..3]:
163                     if BL1617[j][1]==A[k][1]:
164                         kod=A[k][1]
165                 if BL1617[j][2]==1:
166                     if(BL1617[j][3]==1):
167                         A[i][2]=A[i][2]+1
168                         for l in [0..3]:
169                             if(A[l][1]==kod):
170                                 A[l][2]=A[l][2]-1
171                     elif(BL1617[j][3]==2):
172                         A[i][2]=A[i][2]-1
173                         for l in [0..3]:
```

```
                                if(A[l][1]==kod):
                                    A[l][2]=A[l][2]+1
                        else:
                            A[i][2]=A[i][2]+0
                            for l in [0..3]:
                                if(A[l][1]==kod):
                                    A[l][2]=A[l][2]+0

    for i in [0..3]:
        b[i]=1+0.5*(A[i][2])
        Y=C.solve_right(b)
        eredmeny.append([Y[i].n(digits=3),A[i][0]])
    eredmenysorted=sorted(eredmeny,reverse=1)
    eredmenysorted=[[basicColley_r_vektora,csapatnev]]
    return eredmenysorted

@interact
def colley(a=selector(label="csoportok száma",buttons=True,
 ↪  nrows=1, values=[1..8], default=1)):
    seged=vector(QQ,C.ncols())
    html("<center>")
    for i in [0..len(basicColley(a))-1]:
        seged[i]=basicColley(a)[i][0]
        html("<center><font size='5'>%s<font
         ↪  color='red'>%s</font>
        Colley pontot szerzett</font></center>"
        %(basicColley(a)[i][1],basicColley(a)[i][0]))
    html("<center><font size='5'> Meccseredmények
     ↪  </font></center>")
    for i in [0..len(matches(a))-1]:
        home=matches(a)[i][0]
        homescore=matches(a)[i][1]
        guestscore=matches(a)[i][2]
        guest=matches(a)[i][3]

    #táblázatos alak
    tablew=[]
```

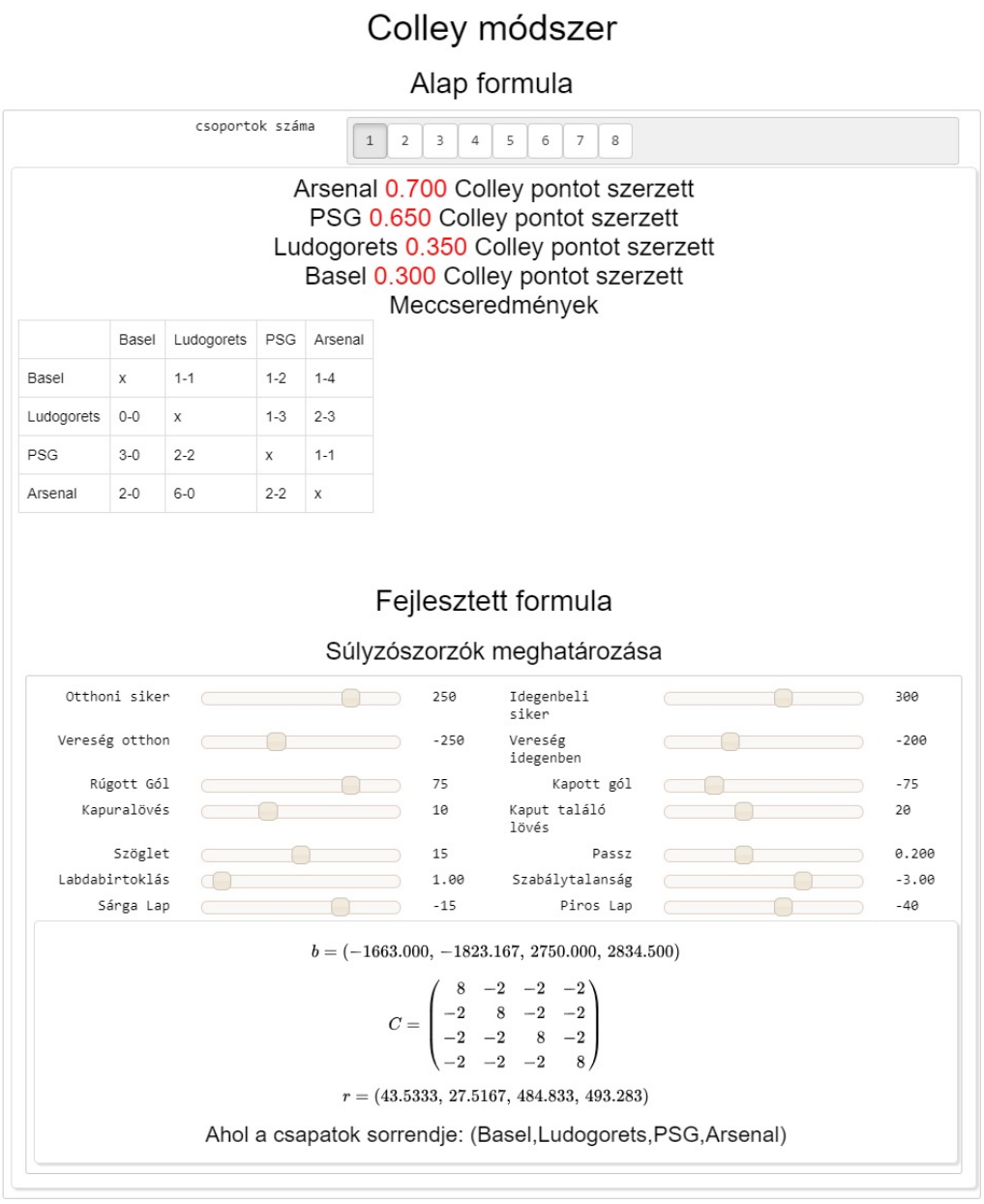
```
208     table1=[' ',matches(a)[0][0],matches(a)[0][3],
        ↪  matches(a)[1][3],matches(a)[2][3]]
209     table2=[table1[1]]
210     table3=[table1[2]]
211     table4=[table1[3]]
212     table5=[table1[4]]
213     for i in [1..4]:
214         for j in [0..11]:
215             if matches(a)[j][0]==table2[0]:
216                 if matches(a)[j][3]==table1[i]:
217                     table2.append(matches(a)[j][1].str()+
218                     '-'+matches(a)[j][2].str())
219     for i in [1..4]:
220         for j in [0..11]:
221             if matches(a)[j][0]==table3[0]:
222                 if matches(a)[j][3]==table1[i]:
223                     table3.append(matches(a)[j][1].str()+
224                     '-'+matches(a)[j][2].str())
225     for i in [1..4]:
226         for j in [0..11]:
227             if matches(a)[j][0]==table4[0]:
228                 if matches(a)[j][3]==table1[i]:
229                     table4.append(matches(a)[j][1].str()+
230                     '-'+matches(a)[j][2].str())
231     for i in [1..4]:
232         for j in [0..11]:
233             if matches(a)[j][0]==table5[0]:
234                 if matches(a)[j][3]==table1[i]:
235                     table5.append(matches(a)[j][1].str()+
236                     '-'+matches(a)[j][2].str())
237
238     table2.append('x')
239     table2[4]=table2[3]
240     table2[3]=table2[2]
241     table2[2]=table2[1]
242     table2[1]='x'
243     table3.append('x')
244     table3[4]=table3[3]
```

```
245     table3[3]=table3[2]
246     table3[2]='x'
247     table4.append('x')
248     table4[4]=table4[3]
249     table4[3]='x'
250     table5.append('x')
251 
252     tablew.append(table1)
253     tablew.append(table2)
254     tablew.append(table3)
255     tablew.append(table4)
256     tablew.append(table5)
257     html("<h2 align=center>%s</h2>"%html.table(tablew))
258 
259     html("<h2 align=center>Fejlesztett formula</h2>")
260     html('<h3 align=center>Súlyzószorzók meghatározása</h3>')
261 
262 
        ↪ @interact(layout=[['WH','WA'],['LOSEHOME','LOSEAWAY'],['RG','KG'],
263     ['KL','KTL'],['CO','PASS'],['POSS','SZAB'],['YC','RC']])
264     def sliders(
265     WH=slider(100,300,5,250, label='Otthoni siker'),
266     WA=slider(150,400,5,300, label='Idegenbeli siker'),
267     LOSEHOME=slider(-400,0,5,-250, label='Vereség otthon'),
268     LOSEAWAY=slider(-300,0,5,-200, label='Vereség idegenben'),
269     RG=slider(0,100,5,75, label='Rúgott Gól'),
270     KG=slider(-100,0,5,-75, label='Kapott gól'),
271     KL=slider(0,30,1,10, label='Kapuralövés'),
272     KTL=slider(0,50,1,20, label='Kaput találó lövés'),
273     CO=slider(0,30,1,15, label='Szöglet'),
274     PASS=slider(0,0.5.n(digits=3),.01,0.2, label='Passz'),
275     POSS=slider(0,10.n(digits=3),0.5,1,
        ↪ label='Labdabirtoklás'),
276     SZAB=slider(-10,0.n(digits=3),0.5,-3,
        ↪ label='Szabálytalanság'),
277     YC=slider(-50,0,1,-15,label='Sárga Lap'),
278     RC=slider(-100,0,5,-40,label='Piros Lap')):
279         def Completepoint(kod):
```

```
280             Complete=0
281             win=0
282             s=0
283             while (s!=6):
284                 for i in [0..BL1617.nrows()-59]:
285                     if BL1617[i][0]==kod:
286                         if BL1617[i][2]==1:
287                             if BL1617[i][3]==1:
288                                 win=win+WH
289                             else:
290                                 win=win+LOSEHOME
291                         elif BL1617[i][2]==2:
292                             if BL1617[i][3]==1:
293                                 win=win+WA
294                             else:
295                                 win=win+LOSEAWAY
296                         Complete=Complete+win+BL1617[i][4]*RG+
297                         BL1617[i][5]*KG+BL1617[i][6]*KL+
298                         BL1617[i][7]*KTL+BL1617[i][8]*CO+
299 
                             ↪  BL1617[i][9]*PASS+(BL1617[i][10]-50)*POSS+
300                         BL1617[i][11]*SZAB+BL1617[i][12]*YC+
301                         BL1617[i][13]*RC
302                         s=s+1
303             return Complete
304         def TiePointCalc(groupcode):
305             A=[]
306             TiePoints=[]
307             tiecalc=[0,0,0,0]
308             for i in [0..len(Codes)-1]:
309                 if Codes[i][2]==groupcode:
310 
                         ↪  A.append([Completepoint(Codes[i][1]),Codes[i][1]])
311             for i in [0..len(A)-1]:
312                 for j in [0..BL1617.nrows()-59]:
313                     if A[i][1]==BL1617[j][0] and
                         ↪  BL1617[j][3]==3:
314                         tiecalc[i]+=1
```

```
315             for i in [0..len(Codes)-1]:
316                 if Codes[i][2]==groupcode:
317                     TiePoints.append([0,Codes[i][1]])
318             for i in [0..len(TiePoints)-1]:
319                 for j in [0..BL1617.nrows()-59]:
320                     if TiePoints[i][1]==BL1617[j][0] and
                        ↪ BL1617[j][2]==1 and BL1617[j][3]==3:
321                         for k in [0..3]:
322                             if BL1617[j][1]==TiePoints[k][1]:
323                                 TiePoints[i][0]+=((A[i][0]*
                                    ↪ (1/(tiecalc[i]^3))+
                                    ↪ A[k][0]*(1/(tiecalc[k]^3)))*0.5).abs()
324                                 TiePoints[k][0]+=((A[k][0]*
                                    ↪ (1/(tiecalc[k]^3))+
                                    ↪ A[i][0]*(1/(tiecalc[i]^3)))*0.5).abs()
325             return TiePoints
326
327         def advancedColley_b(groupcode):
328             A=[]
329             for i in [0..len(Codes)-1]:
330                 if Codes[i][2]==groupcode:
331                     A.append(Codes[i][1])
332             for i in [0..3]:
333                 badv[i]=1+.5*(Completepoint(A[i])+
334                 TiePointCalc(groupcode)[i][0])
335
336         TiePointCalc(a)
337         advancedColley_b(a)
338         html('$$b=%s$$'%latex(badv.n(digits=7)))
339         html('$$C=%s$$'%latex(C))
340         r=C.solve_right(badv).n(digits=6)
341         html('$$r=%s$$'%latex(r))
342         html('<h3 align=center>Ahol a csapatok sorrendje:
            ↪ (%s,%s,%s,%s)</h3>'%(tablew[0][1],
            ↪ tablew[0][2],tablew[0][3],tablew[0][4]))
```

Futási kép az alapbeállításokkal:



Illetve ha a győzelmeket és a vereségeket súlyozzuk ki, a többi adatot pedig kevésbé a 6-os csoportban:

Colley módszer

Alap formula

csoportok száma: 1 2 3 4 5 **6** 7 8

Borussia Dortmund 0.700 Colley pontot szerzett
Real Madrid 0.650 Colley pontot szerzett
Legia Warszawa 0.350 Colley pontot szerzett
Sporting CP 0.300 Colley pontot szerzett

Meccseredmények

	Legia Warszawa	Borussia Dortmund	Real Madrid	Sporting CP
Legia Warszawa	x	0-6	3-3	1-0
Borussia Dortmund	8-4	x	2-2	1-0
Real Madrid	5-1	2-2	x	2-1
Sporting CP	2-0	1-2	1-2	x

Fejlesztett formula

Súlyzószorzók meghatározása

Otthoni siker	300	Idegenbeli siker	400
Vereség otthon	-400	Vereség idegenben	-300
Rúgott Gól	50	Kapott gól	-50
Kapuralövés	5	Kaput találó lövés	10
Szöglet	10	Passz	0.000
Labdabirtoklás	0.000	Szabálytalanság	0.000
Sárga Lap	-5	Piros Lap	-50

$$b = (-1706.250,\ 2855.000,\ 3290.333,\ -1701.500)$$

$$C = \begin{pmatrix} 8 & -2 & -2 & -2 \\ -2 & 8 & -2 & -2 \\ -2 & -2 & 8 & -2 \\ -2 & -2 & -2 & 8 \end{pmatrix}$$

$$r = (103.133,\ 559.258,\ 602.792,\ 103.608)$$

Ahol a csapatok sorrendje: (Legia Warszawa,Borussia Dortmund,Real Madrid,Sporting CP)

A továbbiakban a többi programkód esetén a BL1617 és a Codes listák ugyanazok, így azokat nem írtam bele a programkódokba.

3. Massey-módszer

1997-ben Kenneth Massey találta ki [4], egyetemista korában, és elsőnek az egyetemi focibajnokságban alkalmazta. Ezt az eljárást később tovább fejlesztette azzal a kritériummal, hogy a csapatokat az alapján is értékelte, hogy melyik csapat ellen mikor játszott. A Massey-módszert a mai napig használják az NFL-ben.

3.1. Példa és az eredeti Massey módszer matematikája

Ismét a Bajnokok Ligájának A csoportját vegyük alapul. Az Arsenalnak 4 győzelme és két döntetlenje, a PSG-nek 3 győzelme és 3 döntetlenje, a Ludororetsnek 3 döntetlenje és 3 veresége, míg a Baselnek 2 döntetlenje, illetve 4 veresége volt. A Massey-módszer szintén egy lineáris egyenlet megoldásán alapszik, de most a mátrixot és a vektorokat egy más algoritmussal határozzuk meg. Legyen az egyenletrendszer a következő: $Xr = b$. Elsőnek konstruáljuk meg az X mátrixot. Mivel 4 csapat összesen 12 meccset játszott, így legyen az X egy 12 soros és 4 oszlopos mátrix, melynek elemei csak -1,0 vagy 1. Legyen az egyik csapat $i.$, a másik pedig $j.$. Ha $i.$ csapat legyőzte $j.$-t, akkor az $k.$sorban a $i.$-nek megfelelő oszlophoz tartozó elem 1, $j.$-nek pedig -1 lesz, minden más pedig 0 (döntetlen esetén vagy $i.$-nek vagy $j.$-nek 1 lesz az értéke, a másiknak -1, és a sorban lévő többi elem pedig 0). Így

alakul ki az alábbi mátrix:

$$\begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \\ 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ -1 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 1 \end{pmatrix}$$

Ezután készítsük el a b vektort is. Legyen $b_k = \sum |r_i - r_j| * c$, ahol r_i és r_j az k. meccsen elért statisztikáját jelenti i.-nek és j.-nek, c pedig a Colley-módszernél felhasznált súlyzószorzókat jelenti. Az így kialakult vektor:

$$b = \begin{pmatrix} 162 \\ 632 \\ 511 \\ 138 \\ 564 \\ 310 \\ 482 \\ 589 \\ 151 \\ 498 \\ 617 \\ 202 \end{pmatrix}.$$

Ezután legyen $M = X^T X$ illetve $p = X^T * b$. Ekkor észre lehet venni, hogy

$$M_{i,j} = \begin{cases} i. \text{ és } j. \text{ csapat egymás elleni meccsek száma negatív előjellel} & \text{ha } i \neq j \\ \text{az összes lejátszott meccs} & \text{ha } i = j \end{cases}$$

és p_i pedig az i.csapat összes szerzett pontját jelenti. Az így kialakult lineáris rendszer a következő:

$$Mr = p$$

amelynek a megoldásvektora:

$$r = \begin{pmatrix} -536 \\ -404 \\ -15 \\ 0 \end{pmatrix}$$

Ezekből az adatokból pedig a következő módszerrel két fontos információ is kinyerhető. Elsőnek legyen az $M = D - A$, ahol D egy diagonális mátrix, amelynek a főátlójában a csapatok lejátszott meccsei találhatóak, míg $A_{i,j}$ elemei a csapatok egymás ellen lejátszott meccseinek a számát taglalja, ha $i \neq j$, a főátlóban pedig csupa nullák találhatóak. Így az $Mr = p$ lineáris egyenlet a következőképpen módosul:

$$Dr - Ar = p$$

és ebből fejezzük ki r-t:

$$r = D^{-1}(Ar + p) = D^{-1}Ar + D^{-1}p.$$

Így, i.csapat pontszáma a következő:

$$r_i = \frac{1}{D_{i,i}} \sum_j A_{i,j} r_j + \frac{p_i}{D_{i,i}}.$$

amelyet át tudjuk alakítani $r_i^{(1)} + r_i^{(2)}$ összegre, amely a következő információval rendelkezik:

1. Az algoritmus értékelése az i. csapatnak

$$r_i^{(1)} = \frac{1}{D_{i,i}} \sum_j A_{i,j} r_j$$

$$r^{(1)} = \left(-\frac{587}{10}, -\frac{587}{15}, -\frac{2852}{5}, -\frac{1095}{2} \right)$$

2. Statisztikákból kapott pontja az i. csapatnak

$$r_i^{(2)} = \frac{p_i}{D_{i,i}}$$

$$r^{(2)} = \left(-\frac{1115}{2}, -\frac{1319}{3}, 453, \frac{1095}{2} \right)$$

3.2. Előnyei és hátrányai

Az első, amit észre lehet venni, hogy ez az algoritmus sem tud mit kezdeni a döntetlenekkel. Ha az X mátrixból kihagynánk azon sorokat, amelynél a meccs kimenetele döntetlen lenne, akkor az M mátrix sem pontos értékeket adna vissza, mivel eltűnne a szimmetria és azon meccseket, amelyek döntetlennel végződtek, nem is nézné, viszont a súlyozásokkal ezt lehet korrigálni, emiatt tarthatjuk meg ezt a formulát. A Massey-algoritmusnak van egy másik formája is, amely már foglalkozik döntetlenekkel, illetve értékeli a csapatokat aszerint is, hogy a mennyire erős ellenfél ellen játszotta, illetve mikor játszotta le azt a meccset.

3.3. A szezonális Massey-módszer

Ennél a módszernél az eredeti Massey-módszert alkalmazzuk azzal a változtatással, hogy a lejátszott meccsek "időértékét" is figyelembe vesszük [2]. Elsőnek is jelentse $s_i(t)$ azt a függvényt, amely megadja, hogy mennyi pontot szerzett i. csapat a j. csapat ellen a t. játéknapon. Ha azon a napon egy csapat nem játszott meccset, akkor ennek az értéke legyen 0. Legyen $m_{i,t}$ az i. csapat a j. időpontig lejátszott meccsek számát. Legyen $j_1, \ldots, j_{m_{i,t}}$ a megfelelő napokon az i. csapat ellenfeleit és legyen $t_1, \ldots, t_{m_{i,t}}$ a napokat, amikor játszott az i. csapat. Ekkor az alábbi függvény megadja, hogy így mennyi pontot szerzett i. csapat a j. játéknap után:

$$r_i(t) = \frac{1}{m_{i,t}} \sum_{k=1}^{m_{i,t}} (r_{j_k}(t_k - 1) + s_i(t_k)).$$

Természetesen $r_i(t) = 0$, ha az i. csapat a t. időpontig nem játszott mérkőzést. Ebben az esetben is fel tudjuk bontani $r_i^{(1)} + r_i^{(2)}$ összegre, ahol:

1. A szezonális értékelése az i. csapatnak:

$$r_i^{(1)}(t) = \frac{1}{m_{i,t}} \sum_{k=1}^{m_{i,t}} (r_{j_k}(t_k - 1)$$

2. Statisztikákból kapott pontja az i. csapatnak a j. mérkőzésen:

$$r_i^{(1)}(t) = \frac{1}{m_{i,t}} \sum_{k=1}^{m_{i,t}} s_i(t_k)$$

3.4. Példa a szezonális Massey-módszerhez

Vegyük alapul újra az 1.csoport kimeneteleit. Mint tudjuk, az Arsenal lett az első, majd a PSG, utána a Basel, az utolsó pedig a Ludogorets. A fordulók eredményei a következők:

1.forduló	2.forduló	3.forduló	4.forduló	5.forduló	6.forduló
Basel 1-1 Ludogorets	Arsenal 2-0 Basel	Arsenal 6-0 Ludogorets	Basel 1-2 PSG	Arsenal 2-2 PSG	Basel 1-4 Arsenal
PSG 1-1 Arsenal	Ludogorets 1-3 PSG	PSG 3-0 Basel	Ludogorets 2-3 Arsenal	Ludogorets 0-0 Basel	PSG 2-2 Ludogorets

Viszont a szezonális pontozása nem ebben a sorrendben alakult alakult:

	1.forduló	2.forduló	3.forduló	4.forduló	5.forduló	6.forduló
Basel	242.6	-67.19	-142.2	-260.5	-222.2	-260.3
Ludogorets	-142.6	-258.3	-360.3	-322.2	-356.6	-444.6
PSG	213.4	439.5	520.0	677.1	736.7	845.6
Arsenal	-113.4	311.0	549.2	582.7	581.2	650.4

Ahogy lehet látni, az első fordulóban mindkét csapat döntetlent játszott. Viszont az egyéb statisztikák miatt van a PSG és a Basel plusz ponttal, míg az Arsenal és a Ludogorets mínusszal. Ezzel az első forduló után egy erősorrend már ki is alakult, viszont ebből még nem lehet tudni, hogy melyik csapat a valódi esélyes. A második fordulótól kezdve viszont egyértelműen látszik. Viszont annak ellenére, hogy a PSG végül 845.6 pontot szerzett, ő lett a

második, mivel a statisztikái lehet, hogy jobbak lettek, mint az első helyezett Arsenalnak, mégis játszott egy döntetlent a legutolsó meccsén. Viszont az algoritmus nem azt mutatja meg, hogy matematikailag mely csapat nyerte meg a csoportkört, hanem mely csapat számít a lejátszott meccsek után a legerősebbnek.
Ezenkívül az algoritmus nem csak azon csapatokat értékeli, amely szerzett pontot vagy pontokat az adott meccsen. Például a harmadik fordulóban az Arsenal 6-0-ra győzte le a Ludogorets csapatát, így a harmadik fordulóig az Arsenalnak 549.2 pontja volt, míg a Ludogoretsnek -360.3. Viszont a következő fordulóban már az Arsenal 2-3-ra tudott diadalmaskodni a Ludogoretsen, mégis mindketten plusz pontokat könyvelhettek el. Az Arsenal azért, mivel meg tudta nyerni a mérkőzést, míg a Ludogorets annak ellenére, hogy kikapott, de az aktuális legerősebb csapat ellen vesztett egy szoros mérkőzésen.

3.5. Előnyei és hátrányai a szezonális Massey-módszernek

Ez az algoritmus már tudja kezelni a döntetleneket, sőt ezt már lehet alkalmazni hiányos adattáblázaton is, csak abban az esetben egy újabb függvényt kell kialakítani, amely megadja, hogy egy csapat hány meccset játszott összesen. Ezenkívül az eljárás már figyelembe veszi azt is, hogy aki ellen játszott az i. csapat, az erősebb csapatnak számít-e, vagy sem. Ezáltal egy sokkal valósághűbb értékelést mutat be egy bajnokságról.
A hátránya viszont, hogy előzetes rangsorolással már nem tud mit kezdeni. Ezalatt azt értem, hogy annak ellenére, hogy tudjuk, hogy i. csapat erősebb j.-nél, az első fordulóban ugyanannyi esélyt adnak i.-nek, mint j.-nek. Emiatt muszáj néhány mérkőzés, hogy meg tudjuk mondani, hogy melyik csapatnak van nagyobb esélye a másikkal szemben.

3.6. Programok és programkódok

Eredeti Massey-módszer:

A programkód:

```
def matches(groupcode):
    A=[]
    eredmeny=[]
    ellenfel=''
    for i in [0..len(Codes)-1]:
        if Codes[i][2]==groupcode:
            A.append([Codes[i][0],Codes[i][1]])
    for i in [0..3]:
        for j in [0..BL1617.nrows()-59]:
            if A[i][1]==BL1617[j][0]:
                for k in [0..3]:
                    if BL1617[j][1]==A[k][1]:
                        ellenfel=A[k][0]
                if BL1617[j][2]==1:
                     eredmeny.append([A[i][0],BL1617[j][4],
                      ↪  BL1617[j][5],ellenfel])
    return eredmeny

def Xmatrix(groupcode):
    B=[]
    X=matrix(QQ,12,4)
    A=vector(QQ,X.ncols())
    ellenfel=''
    s=0
    t=0
    r=0
    indextarto=[]
    for i in [0..len(Codes)-1]:
        if Codes[i][2]==groupcode:
            B.append([Codes[i][0],s])
             ↪  ]
```

```
30              s=s+1
31      for i in [0..len(matches(groupcode))-1]:
32          for j in [0..3]:
33              if matches(groupcode)[i][0]==B[j][0]:
34                  r=B[j][1]
35                  for k in [0..3]:
36                      if B[k][0]==matches(groupcode)[i][3]:
37                          t=B[k][1]
38                  if
                     ↪ matches(groupcode)[i][1]>matches(groupcode)[i][2]:
39                      X[i,r]=1
40                      X[i,t]=-1
41                  elif
                     ↪ matches(groupcode)[i][1]<matches(groupcode)[i][2]:
42                      X[i,r]=-1
43                      X[i,t]=1
44                  else:
45                     X[i,r]=1
46                     X[i,t]=-1
47      return X
48
49  html("<h1 align=center>Massey's method</h1>")
50  @interact
51  def massey(csoportszam=selector(label='Csoportok
     ↪ száma:',buttons=True, nrows=1, values=[1..8], default=1)):
52      #táblázatos alak
53      tablew=[]
54
         ↪ table1=[' ',matches(csoportszam)[0][0],matches(csoportszam)[0][3],
         ↪ matches(csoportszam)[1][3],matches(csoportszam)[2][3]]
55      table2=[table1[1]]
56      table3=[table1[2]]
57      table4=[table1[3]]
58      table5=[table1[4]]
59      for i in [1..4]:
60          for j in [0..11]:
61              if matches(csoportszam)[j][0]==table2[0]:
62                  if matches(csoportszam)[j][3]==table1[i]:
```

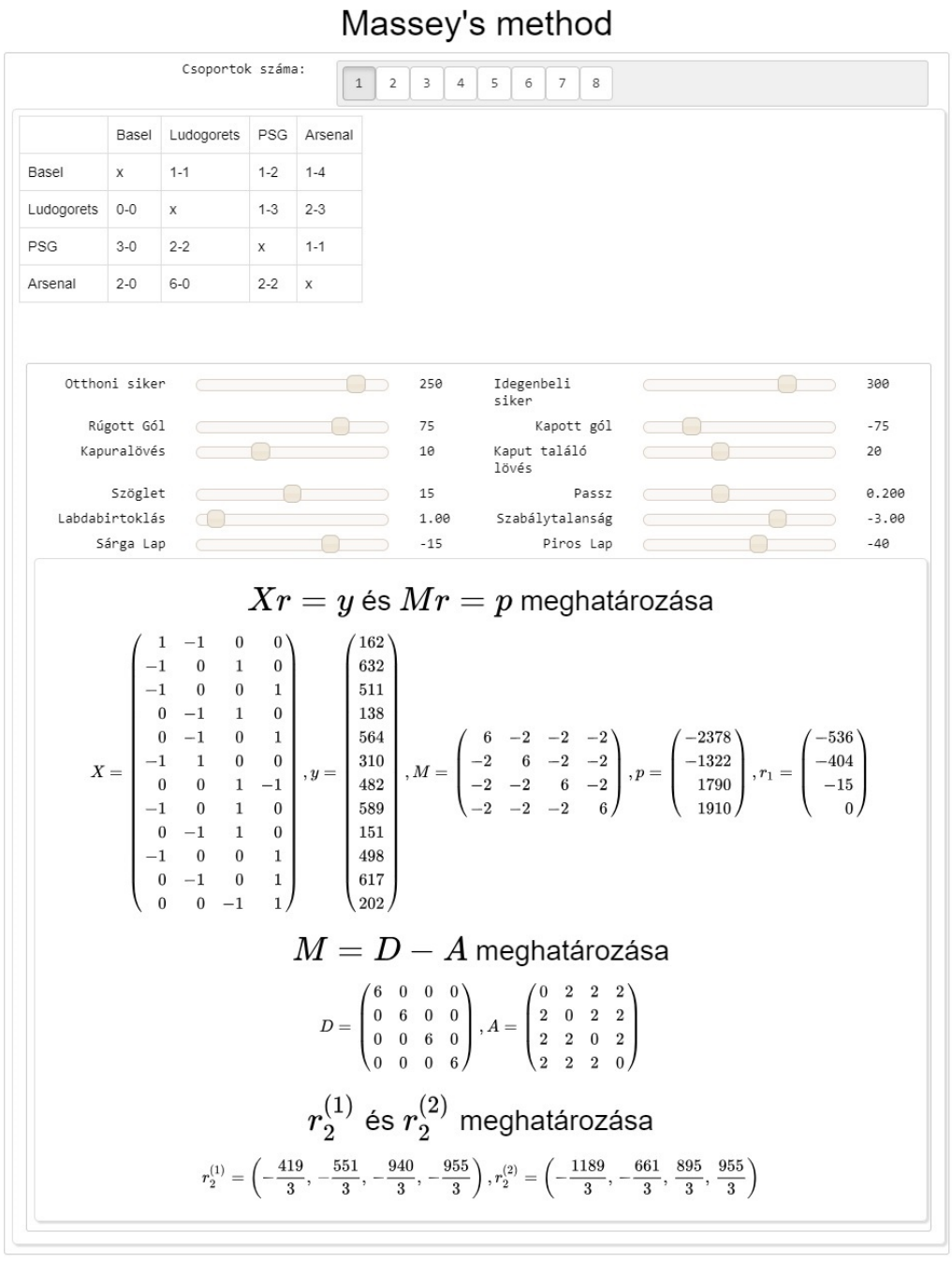
```
63
                        ↪  table2.append(matches(csoportszam)[j][1].str()+
                        ↪  '-'+matches(csoportszam)[j][2].str())
64      for i in [1..4]:
65          for j in [0..11]:
66              if matches(csoportszam)[j][0]==table3[0]:
67                  if matches(csoportszam)[j][3]==table1[i]:
68
                        ↪  table3.append(matches(csoportszam)[j][1].str()+
                        ↪  '-'+matches(csoportszam)[j][2].str())
69      for i in [1..4]:
70          for j in [0..11]:
71              if matches(csoportszam)[j][0]==table4[0]:
72                  if matches(csoportszam)[j][3]==table1[i]:
73
                        ↪  table4.append(matches(csoportszam)[j][1].str()+
                        ↪  '-'+matches(csoportszam)[j][2].str())
74      for i in [1..4]:
75          for j in [0..11]:
76              if matches(csoportszam)[j][0]==table5[0]:
77                  if matches(csoportszam)[j][3]==table1[i]:
78
                        ↪  table5.append(matches(csoportszam)[j][1].str()+
                        ↪  '-'+matches(csoportszam)[j][2].str())
79
80      table2.append('x')
81      table2[4]=table2[3]
82      table2[3]=table2[2]
83      table2[2]=table2[1]
84      table2[1]='x'
85      table3.append('x')
86      table3[4]=table3[3]
87      table3[3]=table3[2]
88      table3[2]='x'
89      table4.append('x')
90      table4[4]=table4[3]
91      table4[3]='x'
92      table5.append('x')
```

```
93
94      tablew.append(table1)
95      tablew.append(table2)
96      tablew.append(table3)
97      tablew.append(table4)
98      tablew.append(table5)
99      html("<h2 align=center>%s</h2>"%html.table(tablew))
100     @interact(layout=[['WH','WA'],['RG','KG'],['KL','KTL'],
        ↪  ['CO','PASS'],['POSS','SZAB'],['YC','RC']])
101     def sliders(WH=slider(0,300,5,250, label='Otthoni siker'),
        ↪  WA=slider(0,400,5,300, label='Idegenbeli siker'),
        ↪  RG=slider(0,100,5,75, label='Rúgott Gól'),
        ↪  KG=slider(-100,0,5,-75, label='Kapott gól'),
        ↪  KL=slider(0,30,1,10, label='Kapuralövés'),
        ↪  KTL=slider(0,50,1,20, label='Kaput találó lövés'),
        ↪  CO=slider(0,30,1,15, label='Szöglet'),
        ↪  PASS=slider(0,0.5.n(digits=3),.01,0.2, label='Passz'),
        ↪  POSS=slider(0,10.n(digits=3),0.5,1,label='Labdabirtoklás'),
        ↪  SZAB=slider(-10,0.n(digits=3),0.5,-3,label='Szabálytalanság'),
        ↪  YC=slider(-50,0,1,-15,label='Sárga Lap'),
        ↪  RC=slider(-100,0,5,-40,label='Piros Lap')): #csúszkák
102
103         def Bvector(groupcode):
104             s=0
105             B=[]
106             C=[]
107             point=0
108             win=0
109             for i in [0..len(Codes)-1]:
110                 if Codes[i][2]==groupcode:
111                     B.append([s,Codes[i][1]])
112                     s=s+1
113             for j in [0..3]:
114                 for i in [0..BL1617.nrows()-59]:
115                     if B[j][1]==BL1617[i][0] and
                        ↪  BL1617[i][2]==1:
116                         C.append([B[j][0],B[j][1],BL1617[i][1],
                            ↪  BL1617[i][3]])
```

```
117                     C=sorted(C)
118             Eredmeny=matrix(len(C),1)
119             for j in [0..len(C)-1]:
120                 for i in [0..BL1617.nrows()-59]:
121                     if C[j][3]==1:
122                         win=WH
123                     elif C[j][3]==2:
124                         win=WA
125                     else:
126                         win=0
127                     if C[j][1]==BL1617[i][0] and
                        ↪ BL1617[i][2]==1 and
                        ↪ BL1617[i][1]==C[j][2]:
128                         point=win+
                            ↪ ((BL1617[i][4]-BL1617[i+1][4]).abs())*RG+
                            ↪ ((BL1617[i][5]-BL1617[i+1][5]).abs())*KG+
                            ↪ ((BL1617[i][6]-BL1617[i+1][6]).abs())*KL+
                            ↪ ((BL1617[i][7]-BL1617[i+1][7]).abs())*KTL+
                            ↪ ((BL1617[i][8]-BL1617[i+1][8]).abs())*CO+
                            ↪ (floor((BL1617[i][9]-BL1617[i+1][9]).abs()*PASS))+
                            ↪ ((((BL1617[i][10]-50)-
                            ↪ (BL1617[i+1][10]-50)).abs())*POSS)+
                            ↪ (((BL1617[i][11]-BL1617[i+1][11]).abs())*SZAB)+
                            ↪ (((BL1617[i][12]-BL1617[i+1][12]).abs())*YC)+
                            ↪ (((BL1617[i][13]-BL1617[i+1][13]).abs())*RC)
129                         Eredmeny[j,0]=point
130                 point=0
131                 win=0
132             return Eredmeny
133
134         M=Xmatrix(csoportszam).transpose()*Xmatrix(csoportszam)
135         p=Xmatrix(csoportszam).transpose()*Bvector(csoportszam)
136         r1=M.solve_right(p)
137
138         html("<h2 align=center>$Xr=y$ és $Mr=p$
            ↪ meghatározása</h2>")
```

```
139         html('$$X=%s,y=%s, M=%s, p=%s,
             ↪  r_1=%s$$'%(latex(Xmatrix(csoportszam)),latex(Bvector(csoportszam))
             ↪  ,latex(M),latex(p),latex(r1)))
140         D=matrix(M.ncols(),M.ncols())
141         A=matrix(M.ncols(),M.ncols())
142
143         for i in [0..M.ncols()-1]:
144             for j in [0..M.ncols()-1]:
145                 if(i==j):
146                     D[i,j]=M[i,j]
147                 else:
148                     A[i,j]=M[i,j]*-1
149         html("<h2 align=center>$M=D-A$ meghatározása</h2>")
150         html('$$D=%s, A=%s$$'%(latex(D),latex(A)))
151         r2=D.inverse()*(A*r1+p)            #r_2==r_1
152         html("<h2 align=center>$r_2^{(1)}$ és $r_2^{(2)}$
             ↪  meghatározása </h2>")
153         r21=vector(QQ,M.ncols())
154         r22=vector(QQ,M.ncols())
155         for i in [0..M.ncols()-1]:
156             asd=0
157             for j in [0..A.ncols()-1]:
158                 asd+=(A[i,j]*r1[j])
159             asd=asd[0]
160             r21[i]=asd/D[i,i]
161             r22[i]=(p[i,0]/D[i,i])
162         html('$$r_2^{(1)}=%s,
             ↪  r_2^{(2)}=%s$$'%(latex(r21),latex(r22)))
```

Futási kép (az alapbeállítások mellett):



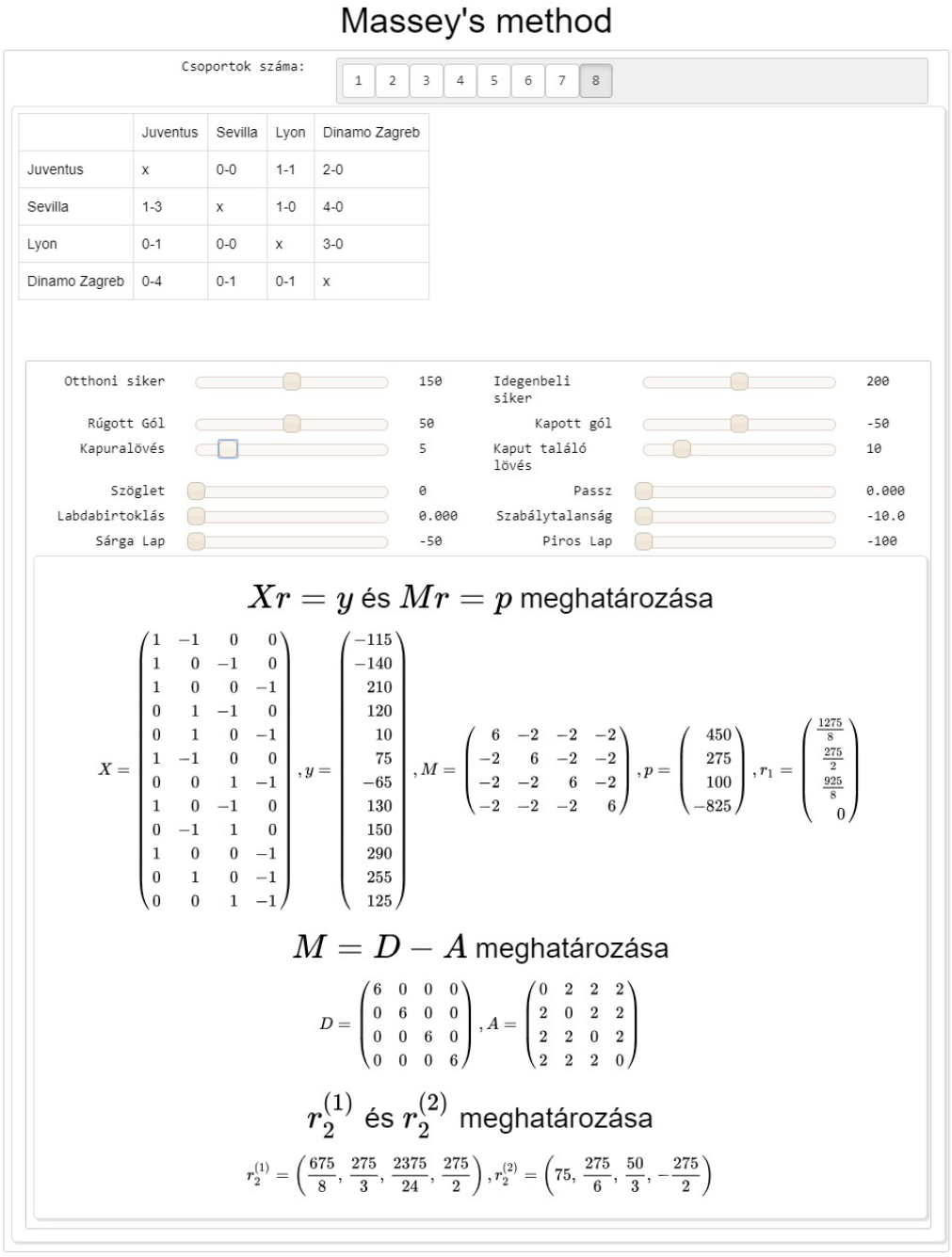
	Basel	Ludogorets	PSG	Arsenal
Basel	x	1-1	1-2	1-4
Ludogorets	0-0	x	1-3	2-3
PSG	3-0	2-2	x	1-1
Arsenal	2-0	6-0	2-2	x

$$X = \begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \\ 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ -1 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 1 \end{pmatrix}, y = \begin{pmatrix} 162 \\ 632 \\ 511 \\ 138 \\ 564 \\ 310 \\ 482 \\ 589 \\ 151 \\ 498 \\ 617 \\ 202 \end{pmatrix}, M = \begin{pmatrix} 6 & -2 & -2 & -2 \\ -2 & 6 & -2 & -2 \\ -2 & -2 & 6 & -2 \\ -2 & -2 & -2 & 6 \end{pmatrix}, p = \begin{pmatrix} -2378 \\ -1322 \\ 1790 \\ 1910 \end{pmatrix}, r_1 = \begin{pmatrix} -536 \\ -404 \\ -15 \\ 0 \end{pmatrix}$$

$$D = \begin{pmatrix} 6 & 0 & 0 & 0 \\ 0 & 6 & 0 & 0 \\ 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & 6 \end{pmatrix}, A = \begin{pmatrix} 0 & 2 & 2 & 2 \\ 2 & 0 & 2 & 2 \\ 2 & 2 & 0 & 2 \\ 2 & 2 & 2 & 0 \end{pmatrix}$$

$$r_2^{(1)} = \left(-\frac{419}{3}, -\frac{551}{3}, -\frac{940}{3}, -\frac{955}{3}\right), r_2^{(2)} = \left(-\frac{1189}{3}, -\frac{661}{3}, \frac{895}{3}, \frac{955}{3}\right)$$

Futási kép a 8.csoportban úgy, hogy a szabálytalanságokért, illetve sárga és piros lapokért járó szorzót kihangsúlyozzuk



	Juventus	Sevilla	Lyon	Dinamo Zagreb
Juventus	x	0-0	1-1	2-0
Sevilla	1-3	x	1-0	4-0
Lyon	0-1	0-0	x	3-0
Dinamo Zagreb	0-4	0-1	0-1	x

$$X = \begin{pmatrix} 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 1 & 0 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{pmatrix}, y = \begin{pmatrix} -115 \\ -140 \\ 210 \\ 120 \\ 10 \\ 75 \\ -65 \\ 130 \\ 150 \\ 290 \\ 255 \\ 125 \end{pmatrix}, M = \begin{pmatrix} 6 & -2 & -2 & -2 \\ -2 & 6 & -2 & -2 \\ -2 & -2 & 6 & -2 \\ -2 & -2 & -2 & 6 \end{pmatrix}, p = \begin{pmatrix} 450 \\ 275 \\ 100 \\ -825 \end{pmatrix}, r_1 = \begin{pmatrix} \frac{1275}{8} \\ \frac{275}{2} \\ \frac{925}{8} \\ 0 \end{pmatrix}$$

$$D = \begin{pmatrix} 6 & 0 & 0 & 0 \\ 0 & 6 & 0 & 0 \\ 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & 6 \end{pmatrix}, A = \begin{pmatrix} 0 & 2 & 2 & 2 \\ 2 & 0 & 2 & 2 \\ 2 & 2 & 0 & 2 \\ 2 & 2 & 2 & 0 \end{pmatrix}$$

$$r_2^{(1)} = \left(\frac{675}{8}, \frac{275}{3}, \frac{2375}{24}, \frac{275}{2}\right), r_2^{(2)} = \left(75, \frac{275}{6}, \frac{50}{3}, -\frac{275}{2}\right)$$

A szezonális Massey-módszer:

A programkód:

```
def matches(groupcode):
    A=[]
    eredmeny=[]
    ellenfel=''
    for i in [0..len(Codes)-1]:
        if Codes[i][2]==groupcode:
            A.append([Codes[i][0],Codes[i][1]])
    for i in [0..3]:
        for j in [0..BL1617.nrows()-59]:
            if A[i][1]==BL1617[j][0]:
                for k in [0..3]:
                    if BL1617[j][1]==A[k][1]:
                        ellenfel=A[k][0]
                if BL1617[j][2]==1:
                     eredmeny.append([A[i][0],BL1617[j][4],
                     ↪  BL1617[j][5],ellenfel])
    return eredmeny

def matches_by_days(groupcode):
    A=[]
    tmp=[]
    eredmeny=[]
    day=1
    for i in [0..len(Codes)-1]:
        if Codes[i][2]==groupcode:
            A.append(Codes[i][1])
    for i in [0..len(A)-1]:
        for j in [0..BL1617.nrows()-59]:
            if A[i]==BL1617[j][0]:
                tmp.append([j,A[i],BL1617[j][3],BL1617[j][4],
                 ↪  BL1617[j][5],BL1617[j][6],BL1617[j][7],BL1617[j][8],
                 ↪  BL1617[j][9],BL1617[j][10],BL1617[j][11],
                 ↪  BL1617[j][12],BL1617[j][13]])
```

```
30      tmp=sorted(tmp)
31      for i in [0..len(tmp)-1,step=4]:
32
            ↪ eredmeny.append([day,tmp[i][1],tmp[i+1][1],tmp[i][2],tmp[i][3],
            ↪ [i][4],tmp[i][5],tmp[i+1][5],tmp[i][6],tmp[i+1][6],
            ↪ tmp[i][7],tmp[i+1][7],tmp[i][8],tmp[i+1][8],tmp[i][9],
            ↪ tmp[i+1][9],tmp[i][10],tmp[i+1][10],tmp[i][11],
            ↪ tmp[i+1][11],tmp[i][12],tmp[i+1][12]])
33
            ↪ eredmeny.append([day,tmp[i+2][1],tmp[i+3][1],tmp[i+2][2],
            ↪ tmp[i+2][3],tmp[i+2][4],tmp[i+2][5],tmp[i+3][5],
            ↪ tmp[i+2][6],tmp[i+3][6],tmp[i+2][7],tmp[i+3][7],
            ↪ tmp[i+2][8],tmp[i+3][8],tmp[i+2][9],tmp[i+3][9],
            ↪ tmp[i+2][10],tmp[i+3][10],tmp[i+2][11],
            ↪ tmp[i+3][11],tmp[i+2][12],tmp[i+3][12]])
34          day+=1
35      return eredmeny
36
37  def opponents(teamcode,day):
38      groupcode=0
39      eredmeny=[]
40      for i in [0..len(Codes)-1]:
41          if teamcode==Codes[i][1]:
42              groupcode=Codes[i][2]
43      for i in [0..len(matches_by_days(groupcode))-1]:
44              if (matches_by_days(groupcode)[i][1]==teamcode):
45                  if (matches_by_days(groupcode)[i][0]<=day):
46
                        ↪ eredmeny.append(matches_by_days(groupcode)[i][2])
47              if (matches_by_days(groupcode)[i][2]==teamcode):
48                  if (matches_by_days(groupcode)[i][0]<=day):
49
                        ↪ eredmeny.append(matches_by_days(groupcode)[i][1])
50      return eredmeny
51
52  def turn_results(groupcode):
53      table=[['1.forduló','2.forduló','3.forduló',
        ↪ '4.forduló','5.forduló','6.forduló'] ,[],[]]
```

```
54      for k in [0..len(matches_by_days(groupcode))-1,step=2]:
55          table[1].append([matches_by_days(groupcode)[k][1],
            ↪ matches_by_days(groupcode)[k][2]])
56          table[2].append([matches_by_days(groupcode)[k+1][1],
            ↪ matches_by_days(groupcode)[k+1][2]])
57      for i in [1..len(table)-1]:
58          for j in [0..len(table[i])-1]:
59              for k in [0..len(table[i][j])-1]:
60                  for l in [0..len(Codes)-1]:
61                      if table[i][j][k]==Codes[l][1]:
62                          table[i][j][k]=Codes[l][0]
63      for i in [1..len(table)-1]:
64          for j in [0..len(table[i])-1]:
65              for l in [0..len(matches(groupcode))-1]:
66                  if table[i][j][0]==matches(groupcode)[l][0] and
                    ↪ table[i][j][1]==matches(groupcode)[l][3]:
67                      table[i][j]=matches(groupcode)[l][0]+ '
                        ↪ '+matches(groupcode)[l][1].str()+
                        ↪ '-'+matches(groupcode)[l][2].str()+ '
                        ↪ '+matches(groupcode)[l][3]
68      return table
69
70  html("<h1 align=center>Szezonális Massey-módszer</h1>")
71  @interact
72  def massey(csoportszam=selector(label='Csoportok
    ↪ száma:',buttons=True, nrows=1, values=[1..8], default=1)):
73      html("<h2
        ↪ align=center>%s</h2>"%html.table(turn_results(csoportszam)))
74      @interact(layout=[['WH','WA'],['TIE','RG'],['KL','KTL'],
        ↪ ['CO','PASS'],['POSS','SZAB'],['YC','RC']])
```

```
75      def sliders( RG=slider(0,100,5,75, label='Rúgott Gól'),
        ↪  KL=slider(0,30,1,10, label='Kapuralövés'),
        ↪  KTL=slider(0,50,1,20, label='Kaput találó lövés'),
        ↪  CO=slider(0,30,1,15, label='Szöglet'),
        ↪  PASS=slider(0,0.5.n(digits=3),.01,0.2, label='Passz'),
        ↪  POSS=slider(0,10.n(digits=3),0.5,1,label='Labdabirtoklás'),
        ↪  SZAB=slider(-10,0.n(digits=3),0.5,-3,
        ↪  label='Szabálytalanság'),
        ↪  YC=slider(-50,0,1,-15,label='Sárga Lap'),
        ↪  RC=slider(-100,0,5,-40,label='Piros Lap'),
        ↪  WH=slider(0,300,5,100,label='Otthoni siker'),
        ↪  WA=slider(0,300,5,150,label='Idegenbeli siker'),
        ↪  TIE=slider(0,100,5,50, label='Döntetlen')):
76
77          def point_difference(teamcode,day):
78              groupcode=0
79              opponent=0
80              win=0
81              for i in [0..len(Codes)-1]:
82                  if teamcode==Codes[i][1]:
83                      groupcode=Codes[i][2]
84              for i in [0..len(matches_by_days(groupcode))-1]:
85                  if teamcode==matches_by_days(groupcode)[i][1]:
86                      if matches_by_days(groupcode)[i][0]==day:
87
                            ↪  opponent=matches_by_days(groupcode)[i][2]
88                  elif
                    ↪  teamcode==matches_by_days(groupcode)[i][2]:
89                      if matches_by_days(groupcode)[i][0]==day:
90
                            ↪  opponent=matches_by_days(groupcode)[i][1]
91              for i in [0..len(matches_by_days(groupcode))-1]:
92                  if
                    ↪  (matches_by_days(groupcode)[i][1]==teamcode):
93
                        ↪  if(matches_by_days(groupcode)[i][2]==opponent):
94                      if
                        ↪  (matches_by_days(groupcode)[i][0]==day):
```

```
95                          if
                             ↪ matches_by_days(groupcode)[i][3]==1:
96                              win=WH
97                          elif
                             ↪ matches_by_days(groupcode)[i][3]==3:
98                              win=TIE
99                          return
                             ↪ win+(matches_by_days(groupcode)[i][4]-
                             ↪ matches_by_days(groupcode)[i][5])*RG+
                             ↪ (matches_by_days(groupcode)[i][6]-
                             ↪ matches_by_days(groupcode)[i][7])*KL+
                             ↪ (matches_by_days(groupcode)[i][8]-
                             ↪ matches_by_days(groupcode)[i][9])*KTL+
                             ↪ (matches_by_days(groupcode)[i][10]-
                             ↪ matches_by_days(groupcode)[i][11])*CO+
                             ↪ (matches_by_days(groupcode)[i][12]-
                             ↪ matches_by_days(groupcode)[i][13])*PASS+
                             ↪ (matches_by_days(groupcode)[i][14]-
                             ↪ matches_by_days(groupcode)[i][15])*POSS+
                             ↪ (matches_by_days(groupcode)[i][16]-
                             ↪ matches_by_days(groupcode)[i][17])*SZAB+
                             ↪ (matches_by_days(groupcode)[i][18]-
                             ↪ matches_by_days(groupcode)[i][19])*YC+
                             ↪ (matches_by_days(groupcode)[i][20]-
                             ↪ matches_by_days(groupcode)[i][21])*RC
100                 elif
                     ↪ (matches_by_days(groupcode)[i][2]==teamcode):
101
                         ↪ if(matches_by_days(groupcode)[i][1]==opponent):
102                        if
                            ↪ (matches_by_days(groupcode)[i][0]==day):
103                             if
                                 ↪ matches_by_days(groupcode)[i][3]==2:
104                                 win=WA
105                             elif
                                 ↪ matches_by_days(groupcode)[i][3]==3:
106                                 win=TIE
```

```
107                             return
                                ↪ win+(matches_by_days(groupcode)[i][5]-
                                ↪ matches_by_days(groupcode)[i][4])*RG+
                                ↪ (matches_by_days(groupcode)[i][7]-
                                ↪ matches_by_days(groupcode)[i][6])*KL+
                                ↪ (matches_by_days(groupcode)[i][9]-
                                ↪ matches_by_days(groupcode)[i][8])*KTL+
                                ↪ (matches_by_days(groupcode)[i][11]-
                                ↪ matches_by_days(groupcode)[i][10])*CO+
                                ↪ (matches_by_days(groupcode)[i][13]-
                                ↪ matches_by_days(groupcode)[i][12])*PASS+
                                ↪ (matches_by_days(groupcode)[i][15]-
                                ↪ matches_by_days(groupcode)[i][14])*POSS+
                                ↪ (matches_by_days(groupcode)[i][17]-
                                ↪ matches_by_days(groupcode)[i][16])*SZAB+
                                ↪ (matches_by_days(groupcode)[i][19]-
                                ↪ matches_by_days(groupcode)[i][18])*YC+
                                ↪ (matches_by_days(groupcode)[i][21]-
                                ↪ matches_by_days(groupcode)[i][20])*RC
108
109         def total_earned_point(teamcode,day):
110             opponent=0
111             groupcode=0
112             if day==0:
113                 return 0
114             else:
115                 for i in [0..len(Codes)-1]:
116                     if teamcode==Codes[i][1]:
117                         groupcode=Codes[i][2]
118                 for i in
                    ↪ [0..len(matches_by_days(groupcode))-1]:
119                     if
                        ↪ teamcode==matches_by_days(groupcode)[i][1]:
120                         if
                            ↪ matches_by_days(groupcode)[i][0]==day:
121
                                ↪ opponent=matches_by_days(groupcode)[i][2]
```

```
122                     elif
                         ↪ teamcode==matches_by_days(groupcode)[i][2]:
123                         if
                             ↪ matches_by_days(groupcode)[i][0]==day:
124
                                 ↪ opponent=matches_by_days(groupcode)[i][1]
125                 return (1/day)*(point_difference(teamcode,day)+
                     ↪ total_earned_point(opponent,day-1))
126
127         def szumpoints(teamcode,day):
128             eredmeny=0
129             for i in [1..day]:
130                 eredmeny+=total_earned_point(teamcode,i)
131             return eredmeny
132
133         def table(groupcode):
134             table=[['','1.forduló','2.forduló','3.forduló',
                     ↪ '4.forduló','5.forduló','6.forduló'],
                     ↪ [matches(groupcode)[0][0]],[matches(groupcode)[0][3]],
                     ↪ [matches(groupcode)[1][3]],[matches(groupcode)[2][3]]]
135             teams=[]
136             teams.append(matches_by_days(groupcode)[0][1])
137             teams.append(matches_by_days(groupcode)[0][2])
138             teams.append(matches_by_days(groupcode)[1][1])
139             teams.append(matches_by_days(groupcode)[1][2])
140             M=matrix(QQ,4,6)
141             for i in [0..M.nrows()-1]:
142                 for j in [0..M.ncols()-1]:
143                     M[i,j]=szumpoints(teams[i],j+1)
144             M=M.n(digits=4)
145             for i in [1..4]:
146                 for j in [1..6]:
147                     table[i].append(M[i-1,j-1])
148             return table
149         html("<h2
             ↪ align=center>%s</h2>"%html.table(table(csoportszam)))
150         html("Ahol az értékek az addigi fordulóig elért
             ↪ értékelések")
```

```
151 html("<h2 align=center>Csapatokra osztott szezonális
    ↪ Massey-módszer</h2>")
152 @interact
153 def group_temp_massey(teamname=selector([
    ↪ 'Basel','Ludogorets','PSG','Arsenal',
    ↪ 'Dinamo Kijev','Benfica','Besiktas',
    ↪ 'Napoli','Machester City','Monchengladbach','Celtic',
    ↪ 'Barcelona','Bayern Munchen','FK Rosztov',
    ↪ 'PSV Eindhoven','Atletico Madrid','Bayer Leverkusen',
    ↪ 'CSZKA Moszkva','Tottenham','Monaco','Legia Warszawa',
    ↪ 'Borussia Dortmund','Real Madrid','Sporting CP',
    ↪ 'Club Brugge','Leicester','FC Porto','FC Kobenhavn',
    ↪ 'Juventus','Dinamo Zagreb','Sevilla','Lyon'],
    ↪ width=25,ncols=4,nrows=8,label='csapatnevek:',default='PSV
    ↪ Eindhoven')):
154
155     def teammatches(teamname):
156         eredmeny=[]
157         teamcode=0
158         for i in [0..len(Codes)-1]:
159             if Codes[i][0]==teamname:
160                 teamcode=Codes[i][1]
161         for i in [0..BL1617.nrows()-1,step=2]:
162             if teamcode==BL1617[i][0] or
                ↪ teamcode==BL1617[i][1]:
163                 eredmeny.append([BL1617[i],BL1617[i+1]])
164         return eredmeny
165     @interact(layout=[['WH','WA'],['TIE','RG'],['KL','KTL'],
      ↪ ['CO','PASS'],['POSS','SZAB'],['YC','RC']])
```

```
166     def sliders( RG=slider(0,100,5,75, label='Rúgott Gól'),
        ↪  KL=slider(0,30,1,10, label='Kapuralövés'),
        ↪  KTL=slider(0,50,1,20, label='Kaput találó lövés'),
        ↪  CO=slider(0,30,1,15, label='Szöglet'),
        ↪  PASS=slider(0,0.5.n(digits=3),.01,0.2, label='Passz'),
        ↪  POSS=slider(0,10.n(digits=3),0.5,1,label='Labdabirtoklás'),
        ↪  SZAB=slider(-10,0.n(digits=3),0.5,-3,label='Szabálytalanság'),
        ↪  YC=slider(-50,0,1,-15,label='Sárga Lap'),
        ↪  RC=slider(-100,0,5,-40,label='Piros Lap'),
        ↪  WH=slider(0,300,5,100,label='Otthoni siker'),
        ↪  WA=slider(0,300,5,150,label='Idegenbeli siker'),
        ↪  TIE=slider(0,100,5,50,label='Döntetlen')):
167
168         def teampoint(teamname,day):
169             eredmeny=0
170             teamcode=0
171             win=0
172             for i in [0..len(Codes)-1]:
173                 if Codes[i][0]==teamname:
174                     teamcode=Codes[i][1]
175             if (teammatches(teamname)[day-1][0][2]==1 and
                ↪  teamcode==teammatches(teamname)[day-1][0][0]):
176                 if teammatches(teamname)[day-1][0][3]==1:
177                     win=WH
178                 elif teammatches(teamname)[day-1][0][3]==3:
179                     win=TIE
```

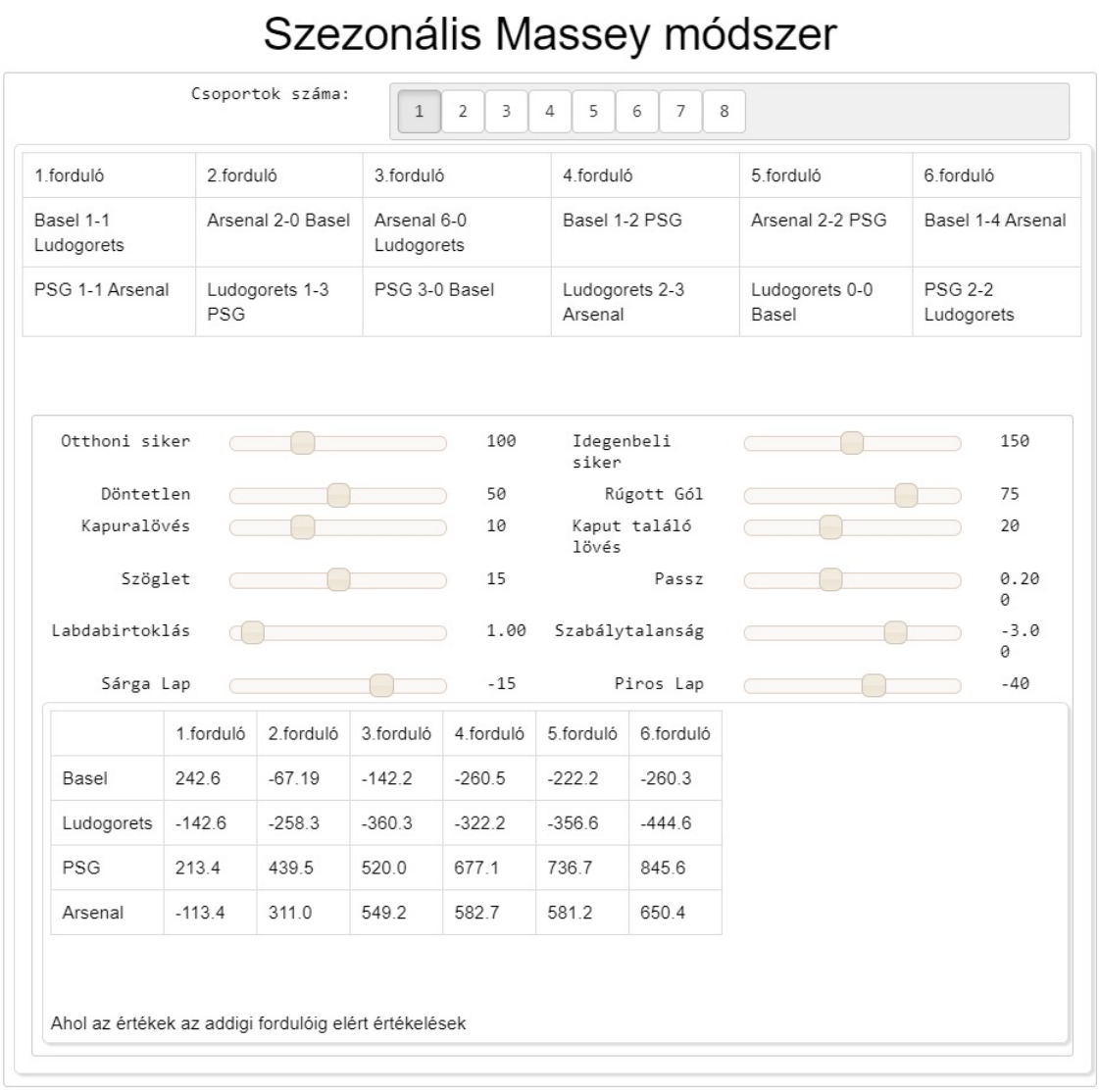
```
180                 eredmeny=win+
                     ↪  (teammatches(teamname)[day-1][0][4]-
                     ↪  teammatches(teamname)[day-1][1][4])*RG+
                     ↪  (teammatches(teamname)[day-1][0][6]-
                     ↪  teammatches(teamname)[day-1][1][6])*KL+
                     ↪  (teammatches(teamname)[day-1][0][7]-
                     ↪  teammatches(teamname)[day-1][1][7])*KTL+
                     ↪  (teammatches(teamname)[day-1][0][8]-
                     ↪  teammatches(teamname)[day-1][1][8])*CO+
                     ↪  (teammatches(teamname)[day-1][0][9]-
                     ↪  teammatches(teamname)[day-1][1][9])*PASS+
                     ↪  (teammatches(teamname)[day-1][0][10]-
                     ↪  teammatches(teamname)[day-1][1][10])*POSS+
                     ↪  (teammatches(teamname)[day-1][0][11]-
                     ↪  teammatches(teamname)[day-1][1][11])*SZAB+
                     ↪  (teammatches(teamname)[day-1][0][12]-
                     ↪  teammatches(teamname)[day-1][1][12])*YC+
                     ↪  (teammatches(teamname)[day-1][0][13]-
                     ↪  teammatches(teamname)[day-1][1][13])*RC
181             elif (teammatches(teamname)[day-1][1][2]==2 and
                 ↪  teamcode==teammatches(teamname)[day-1][1][0]):
182                 if teammatches(teamname)[day-1][0][3]==1:
183                     win=WA
184                 elif teammatches(teamname)[day-1][0][3]==3:
185                     win=TIE
```

```
186                 eredmeny=win+
                     ↪  (teammatches(teamname)[day-1][1][4]-
                     ↪  teammatches(teamname)[day-1][0][4])*RG+
                     ↪  (teammatches(teamname)[day-1][1][6]-
                     ↪  teammatches(teamname)[day-1][0][6])*KL+
                     ↪  (teammatches(teamname)[day-1][1][7]-
                     ↪  teammatches(teamname)[day-1][0][7])*KTL+
                     ↪  (teammatches(teamname)[day-1][1][8]-
                     ↪  teammatches(teamname)[day-1][0][8])*CO+
                     ↪  (teammatches(teamname)[day-1][1][9]-
                     ↪  teammatches(teamname)[day-1][0][9])*PASS+
                     ↪  (teammatches(teamname)[day-1][1][10]-
                     ↪  teammatches(teamname)[day-1][0][10])*POSS+
                     ↪  (teammatches(teamname)[day-1][1][11]-
                     ↪  teammatches(teamname)[day-1][0][11])*SZAB+
                     ↪  (teammatches(teamname)[day-1][1][12]-
                     ↪  teammatches(teamname)[day-1][0][12])*YC+
                     ↪  (teammatches(teamname)[day-1][1][13]-
                     ↪  teammatches(teamname)[day-1][0][13])*RC
187             return eredmeny
188
189         def total_team_point(teamname,day):
190             teamcode=0
191             opponentcode=0
192             opponentname=''
193             if day==0 :
194                 return 0
195             else:
196                 for i in [0..len(Codes)-1]:
197                     if Codes[i][0]==teamname:
198                         teamcode=Codes[i][1]
199                 for i in [0..len(teammatches(teamname))-1]:
200                     if
                         ↪  teamcode==teammatches(teamname)[i][0][0]:
201
                             ↪  opponentcode=teammatches(teamname)[i][0][1]
202                     elif
                         ↪  teamcode==teammatches(teamname)[i][0][1]:
```

```
203
                        ↪ opponentcode=teammatches(teamname)[i][0][0]
204             for i in [0..len(Codes)-1]:
205                 if opponentcode==Codes[i][1]:
206                     opponentname=Codes[i][0]
207             return (1/day)*(teampoint(teamname,day)+
                ↪ total_team_point(opponentname,day-1))
208
209         def szumteampoint(teamname,day):
210             eredmeny=0
211             for i in [1..day]:
212                 eredmeny+=total_team_point(teamname,i)
213             return eredmeny
214
215         def teamopponents(teamname):
216             opponentcode=[]
217             result=[]
218             teamcode=0
219             for i in [0..len(Codes)-1]:
220                 if Codes[i][0]==teamname:
221                     teamcode=Codes[i][1]
222             for i in [0..len(teammatches(teamname))-1]:
223                     if
                        ↪ teamcode==teammatches(teamname)[i][0][0]:
224
                            ↪ opponentcode.append(teammatches(teamname)[i][0][1])
225                     elif
                        ↪ teamcode==teammatches(teamname)[i][0][1]:
226
                            ↪ opponentcode.append(teammatches(teamname)[i][0][0])
227             for i in [0..len(opponentcode)-1]:
228                 for j in [0..len(Codes)-1]:
229                     if opponentcode[i]==Codes[j][1]:
230
                            ↪ result.append([opponentcode[i],Codes[j][0]])
231             return result
232
233         def teamtable(teamname):
```

```
234             table=[[],[],[],[]]
235             fordulok=['1.forduló','2.forduló','3.forduló',
                ↪  '4.forduló','5.forduló','6.forduló', 'Legjobb
                ↪  tizenhat 1.meccs','Legjobb tizenhat
                ↪  2.meccs','Legjobb nyolc 1.meccs', 'Legjobb
                ↪  nyolc 2.meccs', 'Legjobb négy 1. meccs',
                ↪  'Legjobb négy 2.meccs','Finálé']
236             for i in [0..len(teammatches(teamname))-1]:
237                 table[0].append(fordulok[i])
238             for i in [0..len(teammatches(teamname))-1]:
239
                    ↪  table[1].append([teammatches(teamname)[i][0][0],
                    ↪  teammatches(teamname)[i][0][1],
                    ↪  teammatches(teamname)[i][0][4],
                    ↪  teammatches(teamname)[i][0][5]])
240             for i in [0..len(table[1])-1]:
241                 for j in [0..1]:
242                     for k in [0..len(Codes)-1]:
243                         if table[1][i][j]==Codes[k][1]:
244                             table[1][i][j]=Codes[k][0]
245             for i in [0..len(table[1])-1]:
246                 table[1][i]=table[1][i][0]+' '+
                    ↪  table[1][i][2].str()+'-'+
                    ↪  table[1][i][3].str()+' '+table[1][i][1]
247             for i in [1..len(teamopponents(teamname))]:
248                     table[2].append('Összontszáma:
                        ↪  '+szumteampoint(teamname,i).str())
249             for i in [1..len(teamopponents(teamname))]:
250                     table[3].append('Forduló pontszáma:
                        ↪  '+total_team_point(teamname,i).str())
251             return table
252         html("<h2
            ↪  align=center>%s</h2>"%html.table(teamtable(teamname)))
```

Futási kép az 1.csoporttal, illetve az Arsenal eredményeivel:



1.forduló	2.forduló	3.forduló	4.forduló	5.forduló	6.forduló
Basel 1-1 Ludogorets	Arsenal 2-0 Basel	Arsenal 6-0 Ludogorets	Basel 1-2 PSG	Arsenal 2-2 PSG	Basel 1-4 Arsenal
PSG 1-1 Arsenal	Ludogorets 1-3 PSG	PSG 3-0 Basel	Ludogorets 2-3 Arsenal	Ludogorets 0-0 Basel	PSG 2-2 Ludogorets

	1.forduló	2.forduló	3.forduló	4.forduló	5.forduló	6.forduló
Basel	242.6	-67.19	-142.2	-260.5	-222.2	-260.3
Ludogorets	-142.6	-258.3	-360.3	-322.2	-356.6	-444.6
PSG	213.4	439.5	520.0	677.1	736.7	845.6
Arsenal	-113.4	311.0	549.2	582.7	581.2	650.4

Csapatokra osztott szezonális Massey módszer

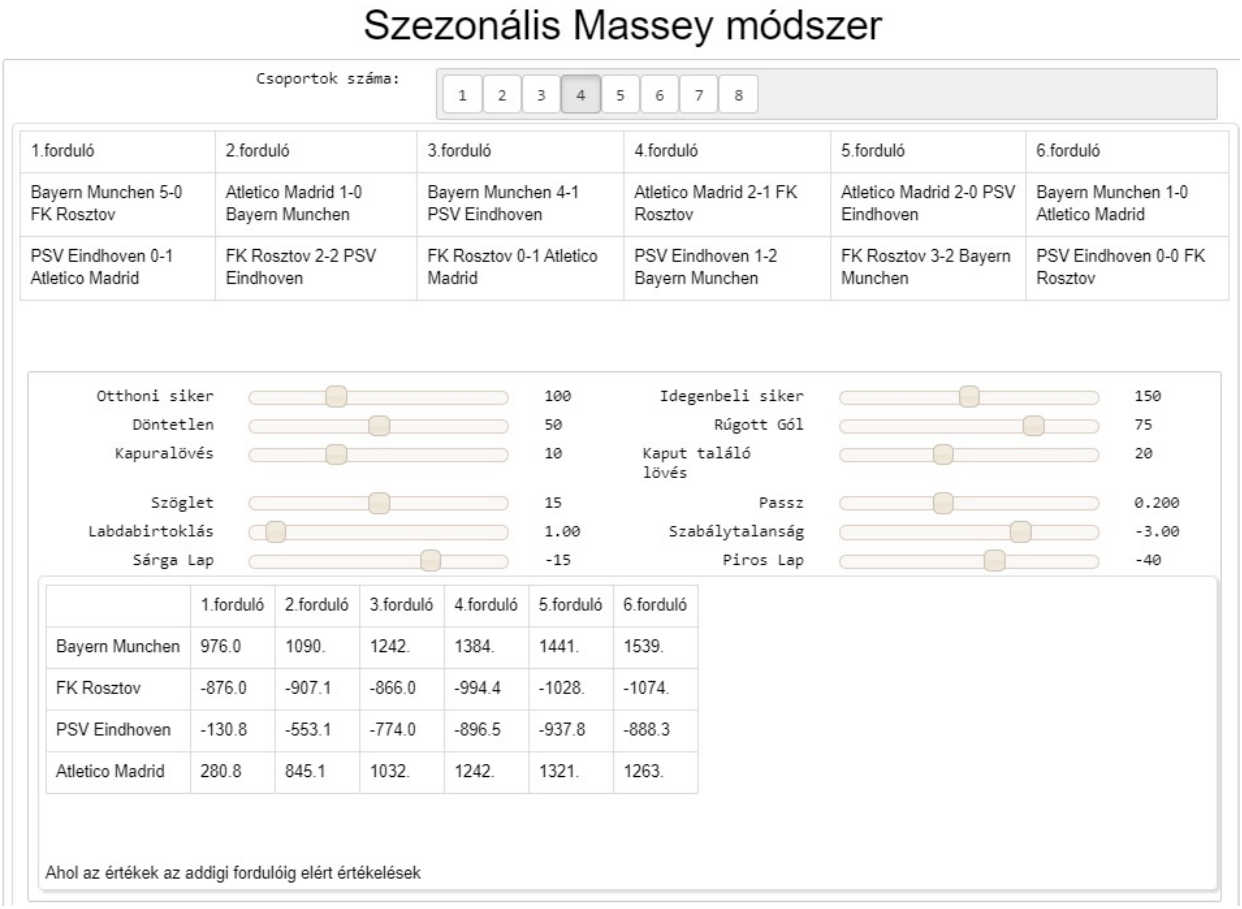
csapatnevek:

Basel	Ludogorets	PSG	Arsenal
Dinamo Kijev	Benfica	Besiktas	Napoli
Machester City	Monchengladbach	Celtic	Barcelona
Bayern Munchen	FK Rosztov	PSV Eindhoven	Atletico Madrid
Bayer Leverkusen	CSZKA Moszkva	Tottenham	Monaco
Legia Warszawa	Borussia Dortmund	Real Madrid	Sporting CP
Club Brugge	Leicester	FC Porto	FC Kobenhavn
Juventus	Dinamo Zagreb	Sevilla	Lyon

Otthoni siker	100	Idegenbeli siker	150
Döntetlen	50	Rúgott Gól	75
Kapuralövés	10	Kaput találó lövés	20
Szöglet	15	Passz	0.200
Labdabirtoklás	1.00	Szabálytalanság	-3.00
Sárga Lap	-15	Piros Lap	-40

1.forduló	2.forduló	3.forduló	4.forduló	5.forduló	6.forduló
PSV Eindhoven 0-1 Atletico Madrid	FK Rosztov 2-2 PSV Eindhoven	Bayern Munchen 4-1 PSV Eindhoven	PSV Eindhoven 1-2 Bayern Munchen	Atletico Madrid 2-0 PSV Eindhoven	PSV Eindhoven 0-0 FK Rosztov
Összontszáma: -130.797	Összontszáma: -478.125	Összontszáma: -697.438	Összontszáma: -923.500	Összontszáma: -1015.31	Összontszáma: -978.188
Forduló pontszáma: -130.797	Forduló pontszáma: -347.312	Forduló pontszáma: -219.312	Forduló pontszáma: -226.031	Forduló pontszáma: -91.7891	Forduló pontszáma: 37.1406

Ezenkívül pedig ha a 4.csoport, illetve a PSV Eindhoven eredményei:



1.forduló	2.forduló	3.forduló	4.forduló	5.forduló	6.forduló
Bayern Munchen 5-0 FK Rosztov	Atletico Madrid 1-0 Bayern Munchen	Bayern Munchen 4-1 PSV Eindhoven	Atletico Madrid 2-1 FK Rosztov	Atletico Madrid 2-0 PSV Eindhoven	Bayern Munchen 1-0 Atletico Madrid
PSV Eindhoven 0-1 Atletico Madrid	FK Rosztov 2-2 PSV Eindhoven	FK Rosztov 0-1 Atletico Madrid	PSV Eindhoven 1-2 Bayern Munchen	FK Rosztov 3-2 Bayern Munchen	PSV Eindhoven 0-0 FK Rosztov

	1.forduló	2.forduló	3.forduló	4.forduló	5.forduló	6.forduló
Bayern Munchen	976.0	1090.	1242.	1384.	1441.	1539.
FK Rosztov	-876.0	-907.1	-866.0	-994.4	-1028.	-1074.
PSV Eindhoven	-130.8	-553.1	-774.0	-896.5	-937.8	-888.3
Atletico Madrid	280.8	845.1	1032.	1242.	1321.	1263.

Csapatokra osztott szezonális Massey módszer

csapatnevek:

Basel	Ludogorets	PSG	Arsenal
Dinamo Kijev	Benfica	Besiktas	Napoli
Machester City	Monchengladbach	Celtic	Barcelona
Bayern Munchen	FK Rosztov	PSV Eindhoven	Atletico Madrid
Bayer Leverkusen	CSZKA Moszkva	Tottenham	Monaco
Legia Warszawa	Borussia Dortmund	Real Madrid	Sporting CP
Club Brugge	Leicester	FC Porto	FC Kobenhavn
Juventus	Dinamo Zagreb	Sevilla	Lyon

Otthoni siker	100	Idegenbeli siker	150
Döntetlen	50	Rúgott Gól	75
Kapuralövés	10	Kaput találó lövés	20
Szöglet	15	Passz	0.200
Labdabirtoklás	1.00	Szabálytalanság	-3.00
Sárga Lap	-15	Piros Lap	-40

1.forduló	2.forduló	3.forduló	4.forduló	5.forduló	6.forduló	Legjobb tizenhat 1.meccs	Legjobb tizenhat 2.meccs
PSG 1-1 Arsenal	Arsenal 2-0 Basel	Arsenal 6-0 Ludogorets	Ludogorets 2-3 Arsenal	Arsenal 2-2 PSG	Basel 1-4 Arsenal	Arsenal 1-5 Bayern Munchen	Bayern Munchen 5-1 Arsenal
Összontszáma: -113.398	Összontszáma: 677.750	Összontszáma: 1037.88	Összontszáma: 1140.38	Összontszáma: 1163.38	Összontszáma: 1224.25	Összontszáma: 1146.75	Összontszáma: 1073.25
Forduló pontszáma: -113.398	Forduló pontszáma: 791.125	Forduló pontszáma: 360.094	Forduló pontszáma: 102.469	Forduló pontszáma: 22.9492	Forduló pontszáma: 60.9180	Forduló pontszáma: -77.5156	Forduló pontszáma: -73.4844

4. Keener-módszer

James P. Keener 1993-ban[3] publikálta a rangsorolási módszerét. Az alapötletet az akkori egyetemi futball bajnokság ihlette, amikor a rivális egyetem nyert, mert ők voltak az egyedüli csapat, akiket nem győztek le. A kérdés, amely megfogalmazódott magában, hogy biztos az a csapat nyerte meg a bajnokságot, amelyik a legjobb volt? Amit megpróbált matematikai módszerekkel bebizonyítani, hogy különböző értékelési formák segítségével milyen rangsorolást eredményeztek a csapatok. A metódusok alapja a Perron-Frobenius tétel, amely segítségével végül négy különböző rangsorolási algoritmust sikerült publikálnia. Az első egy direkt módszer, mely a Perron-Frobenius tételt direktben használja és így alkot konklúziót. A második már figyelembe veszi, hogy a különböző erősségű csapatok mikor játszottak egymás ellen. A harmadik és a negyedik pedig a meccsek végkimenetelét próbálja megtippelni, itt már indirekt módon használjuk a Perron-Frobenius tételt.

4.1. Az alapötlet

Első körben minden esetben kell egy r vektor, amely a csapatok erősségét reprezentálja. Legyen

$$r_i = \frac{W_i}{W_i + L_i} = \frac{W_i}{N_i},$$

ahol W_i a győzelmek számát, L_i a vereségek számát és N_i pedig a lejátszott meccsek számát jelenti. Ez nem teljesen adja vissza a valós rangsort, mivel a döntetlenek esetén nem értékel. Viszont vegyünk egy S mátrixot, amely az alábbi módon legyen értelmezve:

$$S_{i,j} = \begin{cases} i \text{ csapat összpontszáma } j \text{ csapat ellen} & \text{ha játszottak} \\ 0 & \text{ha nem játszottak, vagy } i = j \end{cases}$$

Legyen egy Q mátrix, amely a különböző módszerek esetén máshogy van legenerálva és legyen

$$r = Ar^0 \text{ ,ahol } r^0 = 1.$$

Ezt erősítsük meg azzal, hogy az A mátrixot még n-nel hatványozzuk, és úgy szorozzuk meg r_0-lal, illetve osszuk el a normájával, azaz

$$r = \frac{A^n r^0}{\| A^n r^0 \|}$$

Már csak az a kérdés, hogy mi van akkor, ha $n \to \infty$. A Peron-Frobenius tétel viszont erre ad választ.

4.1. Tétel (Perron-Frobenius)**.** Legyen egy (nemtriviális) A mátrix, úgy, hogy $\forall A_{i,j} \geqslant 0$. Ekkor létezik egy olyan r sajátvektor, amelynek elemei nemnegatívak, és egy hozzátartozó pozitív λ sajátérték. Ezenkívül, ha A irreducibilis, akkor r elemei szigorúan pozitívak és r egyértelmű, az r-hez tartozó λ sajátérték a legnagyobb abszolút értékű és algebrai multiplicitása 1.

4.2. A direkt módszer

Ez a módszer a legegyszerűbb. A csapatok értékelését az egymás ellen elért értékeik segítségével határozzuk meg megszorozva a rangjukkal. Legyen r azon vektor, mely tartalmazza a csapatok erősségét, n_i jelentse az i. csapat által játszott meccsek számát, N a csapatok számát, illetve legyen $a_{i,j}$ a csapatok értékelését. Ekkor legyen az értékelés a következő:

$$s_i = \frac{1}{n_i} \sum_{j=1}^{N} a_{i,j} r_j.$$

Ha viszont $s_i = \lambda r_i$, akkor az alábbi egyenlőséget kapjuk:

$$Ar = \lambda r$$

Így a Perron-Frobenius tétel szerint, ha A irreducibilis és az elemei nemnegatívak, akkor létezik pozitív értékű r.
Elsőnek definiáljuk A mátrixot. Ezt többféle módszerrel is meg tudjuk tenni. A legegyszerűbb a következő:

$$A_{i,j} = \begin{cases} 1 & \text{ha } i. \text{ csapat legyőzte } j. \text{ csapatot} \\ 1/2 & \text{ha } i. \text{ csapat döntetlent ért el } j. \text{ csapat ellen} \\ 0 & \text{ha } i. \text{ csapat vereséget szenvedett } j. \text{ csapat ellen} \end{cases}$$

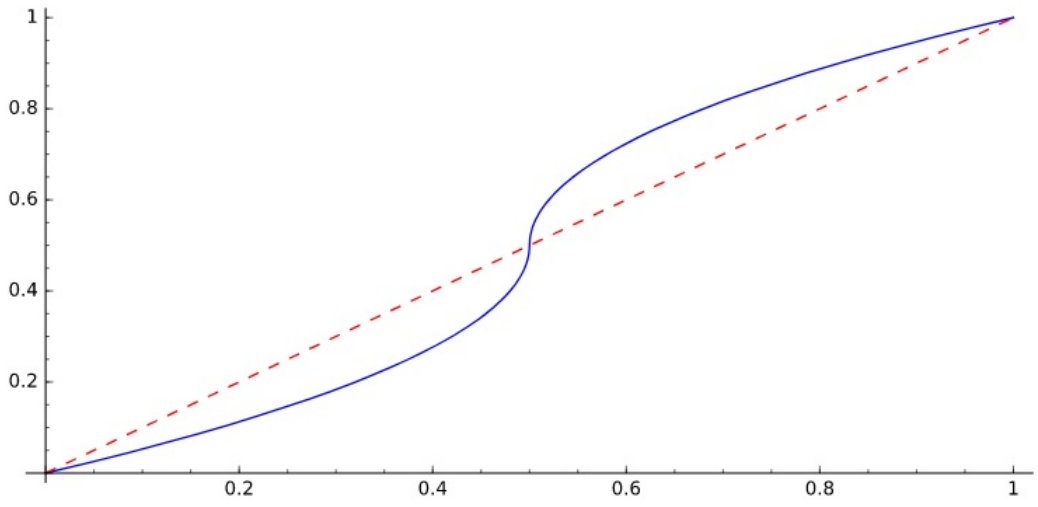
Viszont ez a fajta mátrix nem adná vissza a csapatok teljesítményét, illetve azon csapatok, amelyek több győzelmet szereztek gyengébb csapatok ellen, míg a rangadókat elvesztették, azok több pontot kapnának, mint azok, akik a rangadókat nyerték meg, de a gyengébb csapatok ellen botlanának meg. Így célszerűbb egy másik megoldást választani, amely nem csak a győzelmek száma alapján értékel, ezenkívül a függvény korlátos is lenne, például $[0, 1]$ között lenne. Ehhez elsőnek készítsük el S-t, amelynek $S_{i,j}$ legyen az i. csapat j. csapat elleni statisztikái. Ezután legyen A az alábbi módon definiálva:

$$A_{i,j} = \frac{S_{i,j}}{S_{i,j} + S_{j,i}}$$

Ez viszont azért nem a legideálisabb, ha egy olyan statisztikánk van, amely kevés adattal rendelkezik, akkor nagyon nem releváns értéket fog mutatni. Például, ha csak a gólok számát vesszük, akkor egy 6-0-val végződő mérkőzésen ugyanannyi pont jár a győztes csapatnak, mint egy 1-0-val végződőn. Ezenkívül ha ez lenne az alapja a csapatok rangsorolásának, akkor az erősebb csapatok a gyengébbek elleni meccsre fókuszálnának, hogy ott minél több pontot szeretve nyerjék meg a bajnokságot. Mivel a BL nem az meccsen elért teljesítményből vonatkoztat a bajnokra, illetve több, különféle adattal rendelkezünk, így érdemesebb egy nem lineáris függvényt használni. Erre a legjobb a következő

$$a_{i,j} = h\left(\frac{S_{i,j} + 1}{S_{i,j} + S_{j,i} + 2}\right)$$

$$h(x) = \frac{1}{2} + \frac{1}{2}\mathrm{sgn}(x - \frac{1}{2})\sqrt{|2x - 1|}$$



A $h(x)$ függvény (folytonos vonal) összehasonlítva az $f(x) = x$ függvénnyel (szaggatott vonal).

Azon csapatok, amelyek nem játszottak egymás ellen, $\frac{1}{2}$ pontot szereztek,a függvény értékkészlete még mindig $[0, 1]$ között van, illetve egy nagyobb győzelem esetén a győztes nem húzz el a többiektől. Miután megvan az A mátrix, meg kell határozni egy sajátvektorát. A Sage direktben tud számolni sajátvektort, de egy 32×32-es mátrixban memóriaigényes, és emiatt, ha egyáltalán ki tudja számolni, pontatlan értéket ad vissza, így érdemesebb egy iterációs módszert használni. Ezt az iterációt fogalmaztam meg az alapötletben. Vegyünk egy nemnegatív kezdővektor, legyen ez r_0. Ezt szorozzuk be balról A-val, illetve vegyük ennek a normáját, azaz

$$\frac{Ar_0}{\parallel Ar_0 \parallel},$$

Ezt tegyük meg n-szer, és kapunk egy jól közelített pozitív sajátvektort, illetve ha

$$\lim_{n\to\infty} \frac{A^n r_0}{\parallel A^n r_0 \parallel} = r.$$

Mivel az iteráció csak egy jól közelített becslést ad a sajátvektornak, így egy bizonyos hibahatárt eltekintve vehetjük ezt a sajátvektornak. Mivel az n tart a végtelenbe, ez megint vagy végtelen ciklushoz, vagy memóriatúlcsorduláshoz vezethet, így az iteráció befejeződik, ha nem történik értékmódosítás. Ha

megvan a sajátvektor, abból az elején megadott s_i-vel pedig meg is kapjuk az értékelést.

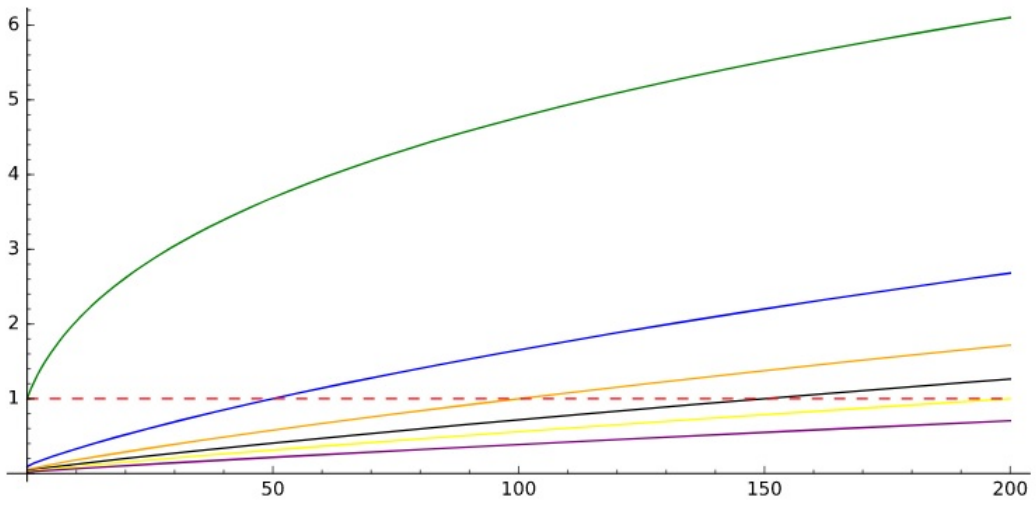
4.3. A nemlineáris módszer

Ez a módszer leginkább a szezonális Massey-módszerre hasonlít. Az alapötlet e mögött is az, hogy számít-e az, hogy egy csapat egy erősebb vagy egy gyengébb ellen érte el a győzelmet, illetve ha vesztett, akkor is milyen meccsvégi statisztikát tud felmutatni. Eredetileg az amerikai bajnokságokban használták, ahol nem minden csapat játszott minden csapat ellen. Ott a tabella állását befolyásolta a meccsen elért fontosabb statisztikai adatok, mint a meccs végén szerzett gólok/pontok. Így az edzők jobban készültek azon meccsekre, ahol egy gyengébb csapat ellen játszhatnak, és a rangadókra pedig kevésbé. Mivel ez az algoritmus az értékelésnél számításba veszi, hogy ki ellen nyert, így az edzők nem vehették félvállról a rangadókat sem, viszont a kisebb csapat elleni meccseket sem, mivel egy ellenük elszenvedett vereség sok pontot le tudna vonni tőlük.
Az alapötlet az, hogy számoljuk ki a csapatok erősségét az alábbi módon:

$$r_i = \frac{1}{n_i} \sum_{j=1}^{N} f(e_{i,j} r_j).$$

Ahol az $e_{i,j}$ a meccsek végkimenetelei, amelyet i. csapat játszott j. ellen, r_j a j csapat erőssége, n_i pedig az i. csapat meccseinek a számát, illetve N az összes csapat számát jelöli.
Elsőként értelmezzük az E mátrixot az alábbi módon:

$$e_{i,j} = \frac{5 + S_{i,j} + S_{i,j}^{2/3}}{5 + S_{j,i} + S_{i,j}^{2/3}}$$

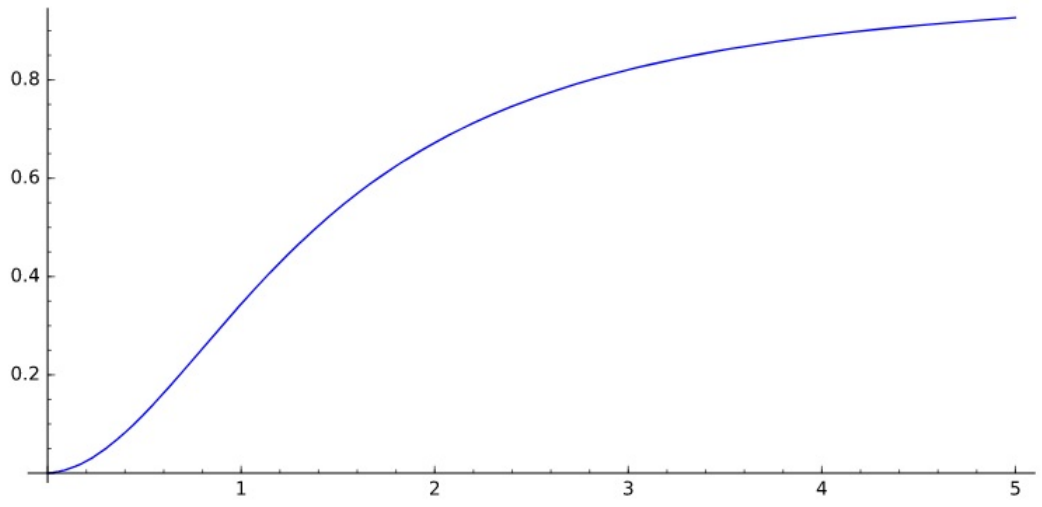


Értelmezzük az $e_{i,j}(S_{i,j}, S_{j,i})$-t, mint egy függvényt, és helyettesítsük be a 0 (zöld), 50 (kék), 100(narancssárga), 150 (fekete), 200 (citromsárga), illetve 300 (lila) értékeket $S_{j,i}$ helyére, és $e_{i,i}(x, x) = 1$ függvény (szaggatott vonal)

Az előnye ennek a választásnak a következő: ha valamelyik csapat nagy előnnyel nyer, akkor az az E mátrixban meg fog mutatkozni, de nem annyira számottevő módon, hogy a kisebb csapatok ne tudják utolérni. Ez annyit tesz, hogy a mátrix minden eleme pozitív, ha nem játszott két csapat egymás ellen, akkor azon értékek nullák, illetve korlátosak az értékek, és a maximum érték pedig 8,45. Ezután vegyünk egy olyan f függvényt, amely pozitív, növekvő és konkáv $[0, \infty[$ között. Legyen ez a

$$f(x) = \frac{0.05x + x^2}{2 + 0.05x + x^2}$$

függvény.



Az $f(x)$ függvény.

Ez alapján már minden adatunk meglenne az r_i elkészítéséhez, de még kell r_j, amit nem ismerünk. Emiatt írjuk fel az r vektort egy függvényként:

$$F_i(r) = \frac{1}{n_i} \sum_{j=1}^{N} f(e_{i,j} r_j).$$

Ez a függvény is pozitív, növekvő és konkáv, így megadható egy alábbi iteráció: $r^k = F(r^{k-1}), r^0 = 1$ és ez a monoton csökkenő sorozat, amely $\lim_{n\to\infty} r^n = r$. Ezenkívül pedig $F(r) = r$. Ezt az iteráció pedig létezik és véges a Perron-Frobenius tétel szerint.

4.4. Program és programkód

A programkód:

```
1  html("<h1 align=center>Keener-módszer</h1>")
2
3  S=matrix(RR,32,32)
4  for k in [0..BL1617.nrows()-1]:
5      for i in [0..S.nrows()-1]:
6          for j in [0..S.ncols()-1]:
7              if i+1==BL1617[k][0] and j+1==BL1617[k][1] and
                ↪  BL1617[k][2]==1:
8                 S[i,j]=S[i,j]+BL1617[k][4]+BL1617[k][6]+
                   ↪  BL1617[k][7]+BL1617[k][8]+BL1617[k][9]+
                   ↪  BL1617[k][10]+BL1617[k][11]+
                   ↪  BL1617[k][12]+BL1617[k][13]
9                 S[j,i]=S[j,i]+BL1617[k+1][4]+BL1617[k+1][6]+
                   ↪  BL1617[k+1][7]+BL1617[k+1][8]+BL1617[k+1][9]+
                   ↪  BL1617[k+1][10]+BL1617[k+1][11]+
                   ↪  BL1617[k+1][12]+BL1617[k+1][13]
10 A=matrix(RR,32,32)
11 for i in [0..S.nrows()-1]:
12     for j in [0..S.ncols()-1]:
13         x=(S[i,j])/(S[i,j]+S[j,i]+1)
14         A[i,j]=((1/2)+(1/2)*(sgn(x-(1/2)))*sqrt((2*x-1).abs()))
15 AA=matrix(RR,32,32)
16 for i in [0..S.nrows()-1]:
17     for j in [0..S.ncols()-1]:
18         x=(S[i,j]+1)/(S[i,j]+S[j,i]+2)
19
            ↪  AA[i,j]=((1/2)+(1/2)*(sgn(x-(1/2)))*sqrt((2*x-1).abs()))
20
21 def PFiter1(m):
22     a = random()
23     V = vector([a, 1-a] + [0 for i in range(len(m[0])-2)])
24     while True:
25         W = m*V
```

```
26          W = W/sum(W)
27          if V == W: #or s>100:
28              return V
29          else:
30              V = W
31
32  n=[]
33  counter=0
34  for i in [0..S.nrows()-1]:
35      for j in [0..S.ncols()-1]:
36          if S[i,j]>0:
37              counter+=2
38      n.append(counter)
39      counter=0
40  n[4]-=1
41  n[14]-=1
42
43  html("<h3 align=center>1.módszer: A direkt eljárás</h3>")
44  pf1=PFiter1(A)
45  print
46  r=[(k/pf1[0]).n(digits=3)  for k in pf1]
47  html('Az direkt iteráció megoldása, amely  a
    ↪  $h((S_{i,j})/(S_{i,j}+S_{j,i}+1))$ módon lett definiálva:
    ↪  ')
48  print r
49
50  pf1A=PFiter1(AA)
51  print
52  rA=[(k/pf1A[0]).n(digits=3) for k in pf1A]
53  html('Az direkt iteráció megoldása, amely  a
    ↪  $h((S_{i,j}+1)/(S_{i,j}+S_{j,i}+2))$ módon lett definiálva:
    ↪  ')
54  print rA
55
56  for i in [0..len(r)-1]:
57      r[i]=[r[i],i+1]
58  r=sorted(r,reverse=1)
59  for i in [0..len(r)-1]:
```

```
60      r[i]=[r[i][0],r[i][1],i+1]
61
62  for i in [0..len(rA)-1]:
63      rA[i]=[rA[i],i+1]
64  rA=sorted(rA,reverse=1)
65  for i in [0..len(rA)-1]:
66      rA[i]=[rA[i][0],rA[i][1],i+1]
67
68  E=matrix(RR,32,32)
69  for i in [0..E.nrows()-1]:
70      for j in [0..E.ncols()-1]:
71          E[i,j]=(5+S[i,j]+(S[i,j]^(2/3)))/
             ↪  (5+S[j,i]+(S[i,j]^(2/3)))
72
73  def f(u):
74      return (0.05*u+u^2)/(2+0.05*u+u^2)
75
76  F=matrix(RR,32,32,lambda i, j:f((5+S[i,j]+
    ↪  (S[i,j]^(2/3)))/(5+S[j,i]+(S[i,j]^(2/3)))))
77
78  def PFiter2(m):
79      a = random()
80      V = vector([a,1-a] + [0 for i in  range(len(m[0])-2)])
81      W=[1.00000000000 for i in F]
82      s=0
83      while True:
84          W=m*V
85          W=W/sum(W)
86          if V == W or s>100:
87              return V
88          else:
89              V = W
90              s+=1
91  html("<h3 align=center>2.módszer: A nemlineáris eljárás</h3>")
92  pf2=PFiter2(E)
93  html('Az nemlineáris iteráció megoldása: ')
94  print  pf2
95  print
```

```
96
97  M2=[]
98  for i in [0..len(pf2)-1]:
99      M2.append([pf2[i].n(digits=3),i+1])
100 M2=sorted(M2,reverse=1)
101 for i in [0..len(M2)-1]:
102     M2[i]=[M2[i][0],M2[i][1],i+1]
103
104 table=[[' ','Direkt módszer $(S_{i,j})/ (S_{i,j}+S_{j,i}+1)$
    ↪ módon adva: (Helyezés,Pontszám)','Direkt módszer
    ↪ $(S_{i,j}+1)/(S_{i,j}+S_{j,i}+2)$ módon adva:
    ↪ (Helyezés,Pontszám)', 'Nemlineáris fixpont módszer:
    ↪ (Helyezés,Pontszám)']]
105 for i in [0..len(r)-1]:
106     for j in [0..len(Codes)-1]:
107         if r[i][1]==Codes[j][1]:
108             table.append([Codes[j][0],r[i][2].str()+',
                ↪ '+r[i][0].str()])
109 for i in [0..len(rA)-1]:
110     for j in [0..len(Codes)-1]:
111         if rA[i][1]==Codes[j][1]:
112             for k in [0..len(table)-1]:
113                 if Codes[j][0]==table[k][0]:
114                     table[k].append(rA[i][2].str()+',
                        ↪ '+rA[i][0].str())
115 for i in [0..len(M2)-1]:
116     for j in [0..len(Codes)-1]:
117         if M2[i][1]==Codes[j][1]:
118             for k in [0..len(table)-1]:
119                 if Codes[j][0]==table[k][0]:
120                     table[k].append(M2[i][2].str()+',
                        ↪ '+M2[i][0].str())
121
122 html("<h3 align=center>Összesített táblázat</h3>")
123 html("<h2 align=center>%s</h2>"%html.table(table))
```

Futási kép:

Keener módszer

1.módszer: A direkt eljárás

Az direkt iteráció megoldása, amely a $h((S_{i,j})/(S_{i,j}+S_{j,i}+1))$ módon lett definiálva:

```
[1.00, 1.45, 1.87, 1.20, 6.74, 2.66, 3.11, 10.1, 3.80, 1.80, 2.34, 1.16, 4.19, 1.45, 10.8, 4.43, 4.17, 5.62, 5.94, 1.68, 0.527, 0.584,
5.81, 1.03, 3.63, 2.78, 3.20, 6.39, 1.61, 1.30, 1.52, 3.93]
```

Az direkt iteráció megoldása, amely a $h((S_{i,j}+1)/(S_{i,j}+S_{j,i}+2))$ módon lett definiálva:

```
[1.00, 0.941, 1.02, 1.03, 1.04, 1.04, 0.963, 1.09, 1.02, 0.967, 1.02, 0.953, 1.04, 0.967, 1.05, 1.00, 1.06, 0.980, 1.09, 0.986, 0.986,
0.984, 1.09, 0.956, 0.972, 1.00, 1.02, 1.05, 1.04, 1.00, 0.991, 1.00]
```

2.módszer: A nemlineáris eljárás

Az nemlineáris iteráció megoldása:

```
(0.0307518821335089, 0.0293887329812827, 0.0312606858951576, 0.0313187486526315, 0.0327725516740734, 0.0332655386282311,
0.0297862545376904, 0.0337470705446254, 0.0312178749079655, 0.0299264879336708, 0.0313474633540792, 0.0293842103108236, 0.0321659875067926,
0.0299273481762381, 0.0321238824602724, 0.0308874575348924, 0.0326848349233686, 0.0301446501088562, 0.0342000042239715, 0.0302585477164372,
0.0302474247254661, 0.0304182564188067, 0.0353558589769034, 0.0294919547940690, 0.0302459366721729, 0.0307708379608389, 0.0309774985729551,
0.0315686121437138, 0.0320235401948468, 0.0308562040214403, 0.0304137100212937, 0.0310699512929244)
```

Illetve az összesített táblázat:

Összesített táblázat

	Direkt módszer $(S_{i,j})/(S_{i,j}+S_{j,i}+1)$ módon adva: (Helyezés,Pontszám)	Direkt módszer $(S_{i,j}+1)/(S_{i,j}+S_{j,i}+2)$ módon adva: (Helyezés,Pontszám)	Nemlineáris fixpont módszer: (Helyezés,Pontszám)
Real Madrid	1, 10.8125	5, 1.04626	8, 0.0321236
Borussia Dortmund	2, 10.0615	3, 1.08582	3, 0.0337486
Juventus	3, 6.73682	8, 1.04175	5, 0.0327721
Napoli	4, 6.38721	6, 1.04578	10, 0.0315704
Barcelona	5, 5.93799	1, 1.09436	2, 0.0341988
Bayern Munchen	6, 5.81348	2, 1.08948	1, 0.0353546
Monaco	7, 5.61963	25, 0.979858	26, 0.0301437
Sporting CP	8, 4.42969	18, 1.00085	17, 0.0308876
Machester City	9, 4.19141	7, 1.04236	7, 0.0321655
Tottenham	10, 4.16992	4, 1.05774	6, 0.0326843
Atletico Madrid	11, 3.92822	19, 1.00073	15, 0.0310707
Bayer Leverkusen	12, 3.79834	14, 1.01685	14, 0.0312176
Benfica	13, 3.63086	26, 0.971924	24, 0.0302467
Dinamo Kijev	14, 3.20410	15, 1.01648	16, 0.0309772
Legia Warszawa	15, 3.11182	29, 0.962585	29, 0.0297871
Besiktas	16, 2.77979	17, 1.00195	19, 0.0307713
Sevilla	17, 2.65820	10, 1.03528	4, 0.0332642
Lyon	18, 2.34448	13, 1.02136	11, 0.0313492
FC Porto	19, 1.86938	12, 1.02356	13, 0.0312614
CSZKA Moszkva	20, 1.80469	28, 0.966919	28, 0.0299263
Celtic	21, 1.68481	23, 0.985962	23, 0.0302582
PSG	22, 1.60962	9, 1.03711	9, 0.0320244
PSV Eindhoven	23, 1.51904	21, 0.990662	22, 0.0304146
Leicester	24, 1.45300	32, 0.941467	31, 0.0293884
Monchengladbach	25, 1.45203	27, 0.967346	27, 0.0299282
Arsenal	26, 1.30334	16, 1.00269	18, 0.0308571
FC Kobenhavn	27, 1.19812	11, 1.02612	12, 0.0313187
Dinamo Zagreb	28, 1.15857	31, 0.953064	32, 0.0293846
FK Rosztov	29, 1.02637	30, 0.956238	30, 0.0294914
Club Brugge	30, 1.00000	20, 1.00000	20, 0.0307522
Ludogorets	31, 0.584229	24, 0.983765	21, 0.0304184
Basel	32, 0.527100	22, 0.986023	25, 0.0302467

Irodalomjegyzék

[1] W. N. Colley. Colley's Bias Free College Football Ranking Method:The Colley Matrix Explained, 2003. http://www.colleyrankings.com/matrate.pdf.

[2] M. Franceschet, E. Bozzo, and P. Vidoni. The temporalized Massey's method. *Journal of Quantitative Analysis in Sports*, 13(2):37–48, June 2017.

[3] J. P. Keener. The Perron–Frobenius Theorem and the Ranking of Football Teams. *SIAM Review*, 35(1):80–93, 1993.

[4] K. Massey. Statistical Models Applied to the Rating of Sports Teams. Master's thesis, Bluefield College, 1997. `https://www.masseyratings.com/theory/massey97.pdf`.

[5] W. A. Stein et al. *Sage Mathematics Software (Version 8.1)*. The Sage Development Team, 2018. `http://www.sagemath.org`.