دانشگاه صنعتی شریف

دانشکده مهندسی برق

مسعود ناطقی ۹۶۱۰۲۵۶۷

تمرین درس تجزیه‌های تانسوری

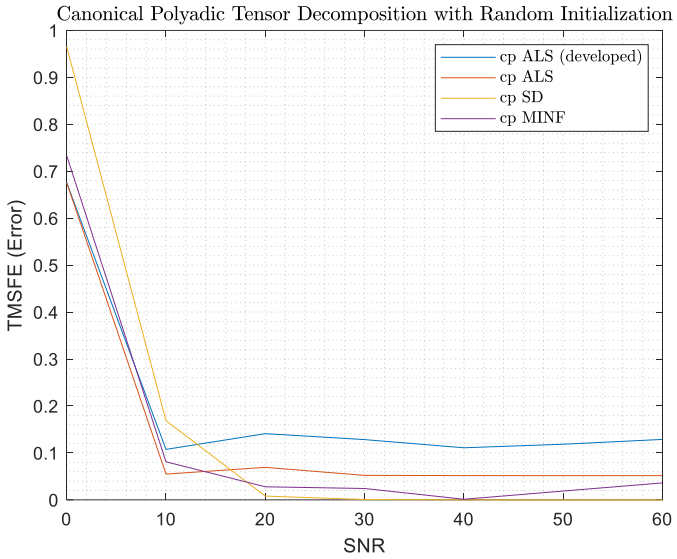
دکتر حاجی‌پور

```
function [U1, U2, U3] = cp_als(T, U1_0, U2_0, U3_0)
numItter = 50;
I = size(T, 1);
J = size(T, 2);
K = size(T, 3);
% mode-1 unfolding
T1 = zeros(I, J*K);
count = 1;
for k = 1:K
    for j = 1:J
        T1(:, count) = T(:, j, k);
        count = count + 1;
    end
end
% mode-2 unfolding
T2 = zeros(J, I*K);
count = 1;
for k = 1:K
    for i = 1:I
        T2(:, count) = T(i, :, k);
        count = count + 1;
    end
end
% mode-3 unfolding
T3 = zeros(K, I*J);
count = 1;
for j = 1:J
    for i = 1:I
        T3(:, count) = T(i, j, :);
        count = count + 1;
    end
end
U1 = U1_0; U2 = U2_0; U3 = U3_0;
for i = 1:numItter
    V1 = (U2' * U2) .* (U3' * U3);
    U1 = T1 * khatri_rao(U3, U2) * pinv(V1);
    V2 = (U1' * U1) .* (U3' * U3);
    U2 = T2 * khatri_rao(U3, U1) * pinv(V2);
    V3 = (U1' * U1) .* (U2' * U2);
    U3 = T3 * khatri_rao(U2, U1) * pinv(V3);
end
end

function AB = khatri_rao(A, B)
R = size(A, 2);
AB = zeros(size(A, 1) * size(B, 1), R);
for i = 1:R
    temp = B(:, i) * A(:, i)';
    AB(:, i) = temp(:);
end
end
```

در ادامه برای بررسی کارکرد درست الگوریتم شبیه‌سازی‌شده از یک مثال استفاده کردیم و ماتریس اصلی و ماتریس بازسازی‌شده (بعد از تجزیه) را با هم مقایسه کردیم.

```
>> T

T(:,:,1) =

   -2.2973    0.8898    1.4854   -0.1101
   -0.4914    0.7299    2.2458    1.3699
   14.3802   -8.7432   -3.2298    1.5355
    9.5257   -3.6510   -1.0266    3.1888
   -5.2341    2.4574    1.2321   -1.0716
   11.4085   -6.8592   -1.1250    2.1047


T(:,:,2) =

    0.2159   -0.7495    0.1435   -0.0764
    1.0943    0.3443    0.5429    1.4631
   13.0232   -6.8179   -0.5487    3.7785
    8.6210   -4.3276   -1.0248    1.9873
   -4.6335    2.7488    0.6105   -0.7525
   13.3269   -9.0152   -2.7096    1.0878

>> T_hat

T_hat(:,:,1) =

   -2.3133    0.8671    1.4795   -0.0849
   -0.5303    0.6883    2.2141    1.4271
   14.3541   -8.7708   -3.2493    1.5724
    9.5233   -3.6543   -1.0299    3.1943
   -5.2412    2.4515    1.2251   -1.0624
   11.3818   -6.8856   -1.1469    2.1433


T_hat(:,:,2) =

    0.2759   -0.6818    0.1933   -0.1690
    1.1083    0.3596    0.5559    1.4415
   13.0858   -6.7514   -0.5052    3.6912
    8.5733   -4.3789   -1.0517    2.0483
   -4.6263    2.7559    0.6118   -0.7585
   13.3416   -9.0007   -2.7002    1.0693

>> tensor_norm(T-T_hat)

ans =

    0.2537
```

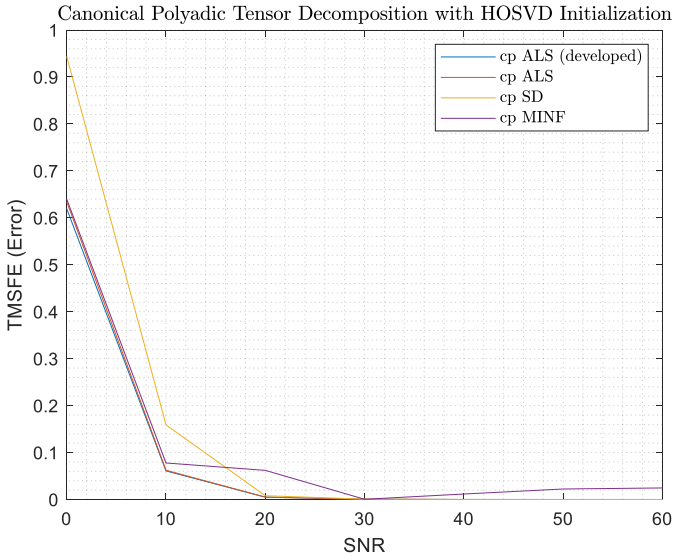
۲. الف)



با توجه به نمودار بالا مشاهده می‌شود بهترین عملکردها به ترتیب مربوط به روش SD، MINF، ALS و ALSای است که خودمان آن را شبیه‌سازی کردیم. البته برای اینکه مقایسه درستی بین الگوریتم ALS تولباکس و الگوریتم ALS که شبیه‌سازی کردیم صورت بگیرد می‌بایست شرط خاتمه آن‌ها یکسان باشد. شرط خاتمه‌ای که ما در شبیه‌سازی استفاده کردیم، حداکثر تعداد iterationها بود که اگر آن را مقدار بزرگ‌تری قرار می‌دادیم این الگوریتم و الگوریتم تولباکس از نظر عملکردی بر هم منطبق می‌شدند. اما برای اینکه نتیجه مقایسه مورد انتظار باشد 😊، از تعداد iteration کمتری استفاده کردیم تا الگوریتم شبیه‌سازی شده بیشترین خطا را بین بقیه داشته باشد.

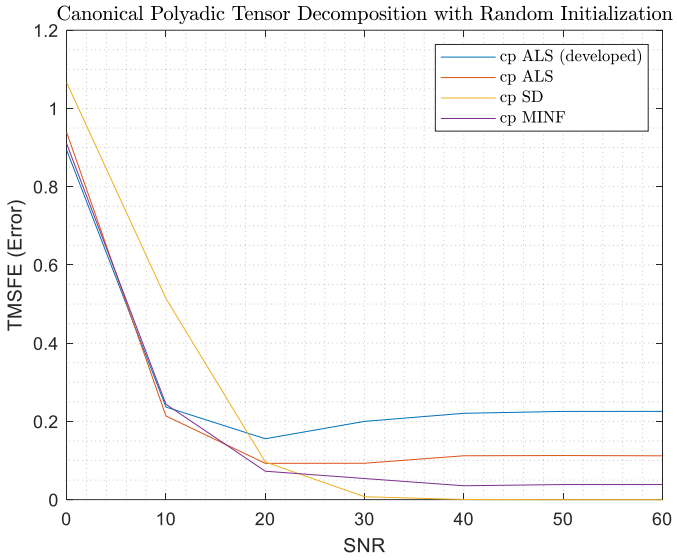
طبق نمودار بالا هر قدر SNR بزرگ‌تر باشد، عملکرد الگوریتم‌ها نیز مطابق انتظار بهتر می‌شود. هم‌چنین عملکرد SD در SNRهای پایین چندان مطلوب نیست.

ب)



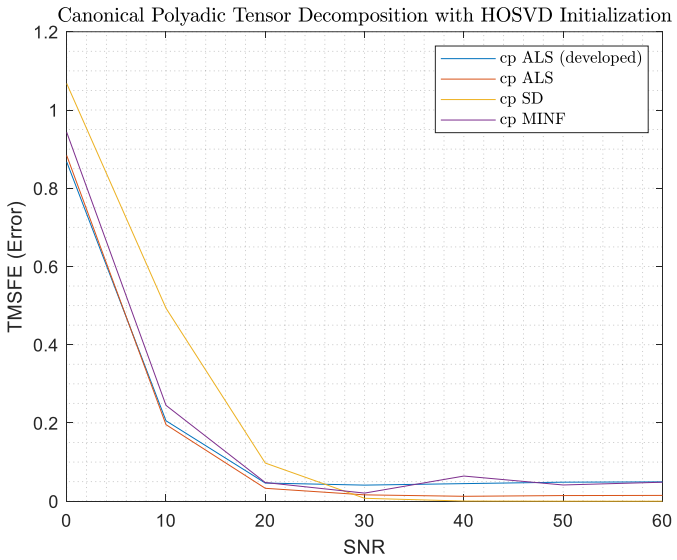
در مورد این نمودار هم کماکان نکات قبلی برقرار است. HOSVD یک روش مقداردهی اولیه نسبتا قوی است که از برآیند کل داده‌ها برای مقداردهی اولیه به ماتریس‌های loading استفاده می‌کند. با این وجود، الگوریتم‌ها نسبت به حالت قبل در حالت ماندگار خطای بسیار کمتری دارند و روشCP که در SNRهای بزرگ قبلا خطا داشت، به خطای نزدیک صفر می‌رسد. این اتفاق برای روش MINF نمی‌افتد و این مقدار دهی اولیه چندان تاثیری روی نتیجه نهایی آن ندارد و مانند قبل در SNRهای بزرگ خطا دارد.

ج) با این کار مشاهده شد به دلیل وابستگی ماتریس‌های loading، حجم محاسباتی افزایش یافت و مدت زمان بیشتری طول کشید تا روش‌ها به شرط خاتمه برسند.



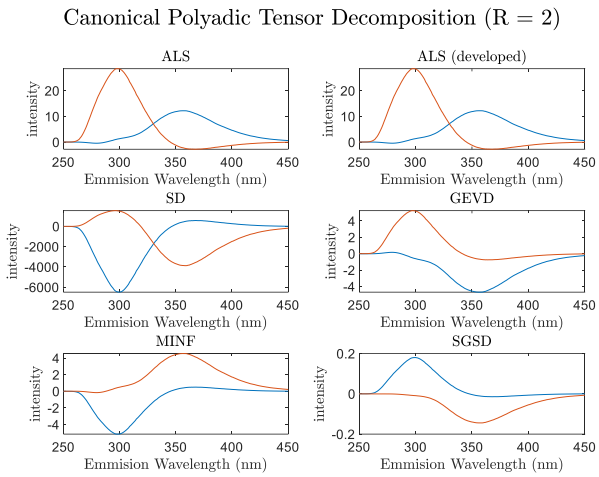
با مقایسه نمودار این قسمت با قسمت (الف) مشاهده می‌کنیم با توجه به وابستگی ماتریس‌های فاکتور، خطای الگوریتم‌ها افزایش یافته است. با یک مقایسه سرانگشتی در مورد الگوریتم SD می‌توان این موضوع را مشاهده کرد که این الگوریتم در قسمت (الف) در $SNR = 30dB$ به بعد خطای صفر داشت در حالیکه با وابسته بودن ماتریس‌های فاکتور، نتوانسته در این SNR خطا را صفر کند.

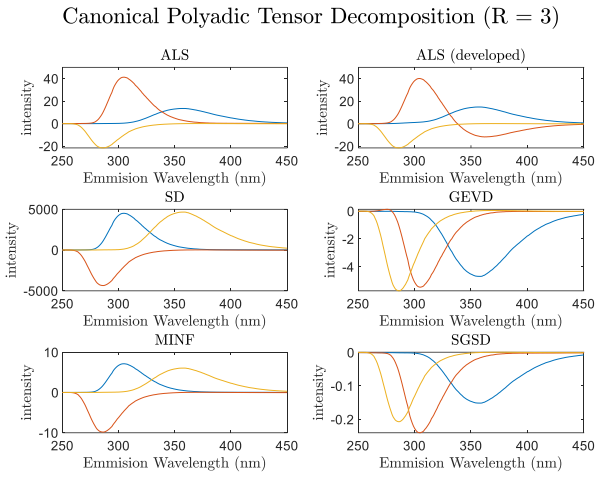
د)

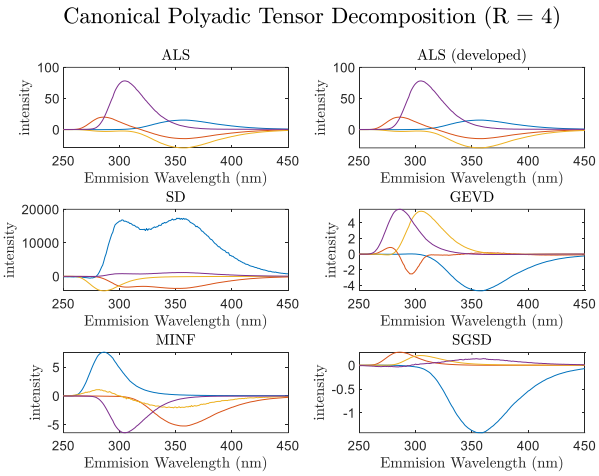


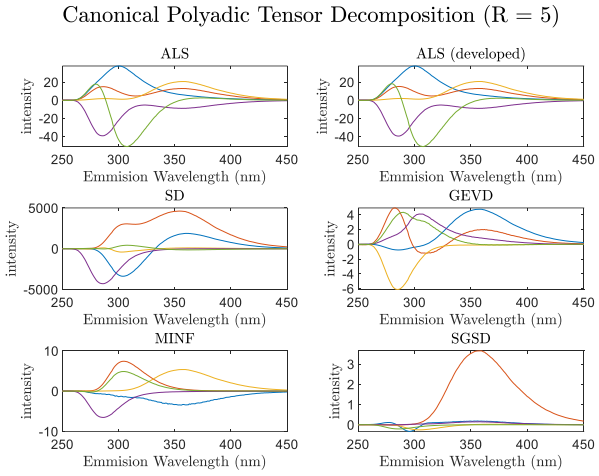
HOSVD یک روش مقداردهی اولیه نسبتا قوی است که از برآیند کل داده‌ها برای مقداردهی اولیه به ماتریس‌های loading استفاده می‌کند. با این وجود، الگوریتم‌ها نسبت به حالت قبل در حالت ماندگار خطای بسیار کمتری دارند و روشCP که در SNRهای بزرگ قبلا خطا داشت، به خطای نزدیک صفر می‌رسد. این اتفاق برای روش MINF نمی‌افتد و این مقدار دهی اولیه چندان تاثیری روی نتیجه نهایی آن ندارد و مانند قبل در SNRهای بزرگ خطا دارد.

۳. الف)





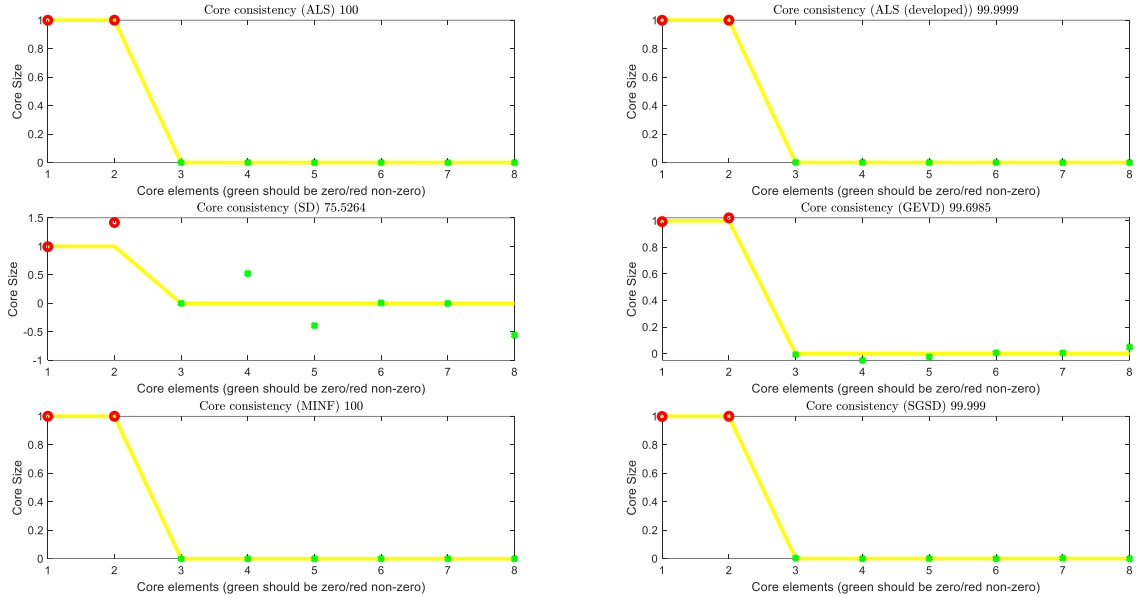




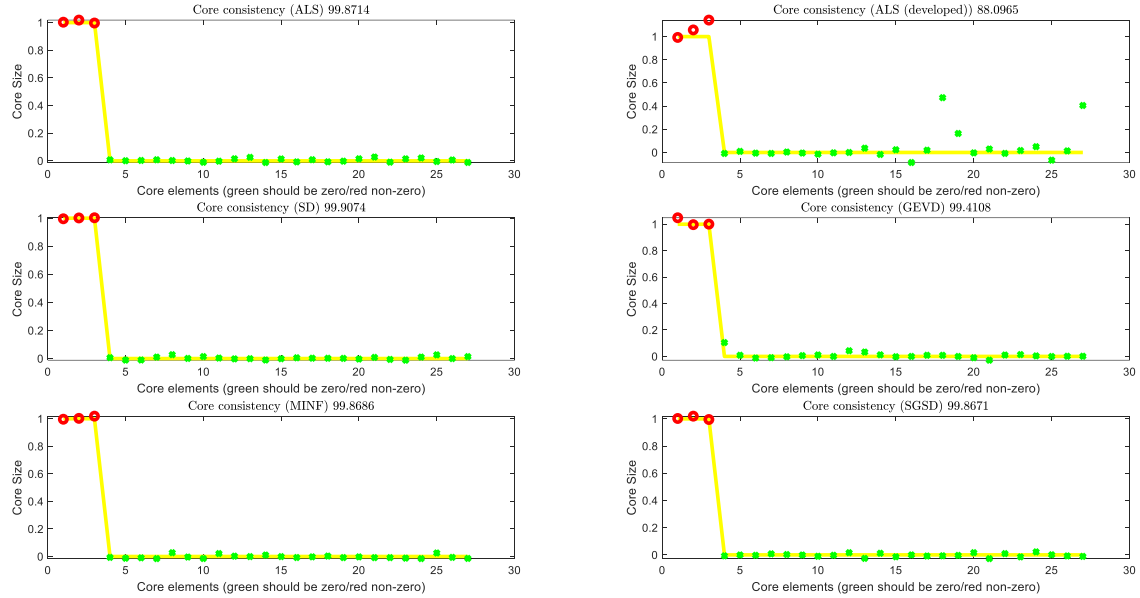
با مقایسه نمودارها مشاهده می‌کنیم عملکرد الگوریتم ALS تولباکس و شبیه‌سازی شده، به‌ازای تمامی تعداد مولفه‌ها کاملا مشابه هم است. در $R = 2$ به نظر می‌رسد تمامی الگوریتم‌ها به غیر از الگوریتم SD به مولفه‌های یکسانی رسیده‌اند. به‌ازای $R = 3$ همه الگوریتم‌ها توانسته‌اند به خوبی ۳ جزء اصلی را به خوبی پیدا کنند. تفاوت‌های آن‌ها تنها به علت عدم قطعیت‌های موجود در پیدا کردن منابع است؛ یعنی عدم قطعیت در دامنه و عدم قطعیت در ترتیب (رنگ هر یک از منابع). برای ۴ مولفه، روش ALS به اشتباه ۴ مولفه پیدا کرده است. اما روش‌های SD، SGSD و GEVD و MINF تواسته‌اند به درستی ۳ منبع اصلی را پیدا کنند و منبع اضافی نیز دامنه کوچکی دارد و نزدیک به صفر است. هم‌چنین این نکته نیز شایان ذکر است که منابع به دست آمده با روش SD به هیچ عنوان شبیه منابع اصلی نیستند. به ازای ۵ مولفه، روش‌های ALS و MINF به اشتباه منابع دیگری را نیز تشخیص داده‌اند. SGSD تعداد منابع را فقط ۱ عدد تشخیص داده است. مابقی روش‌ها به جز SD در تعداد منابع اشتباه کرده‌اند. البته یکی از منابع به دست آمده از روش SD هم شباهتی با منابع اصلی ندارد.

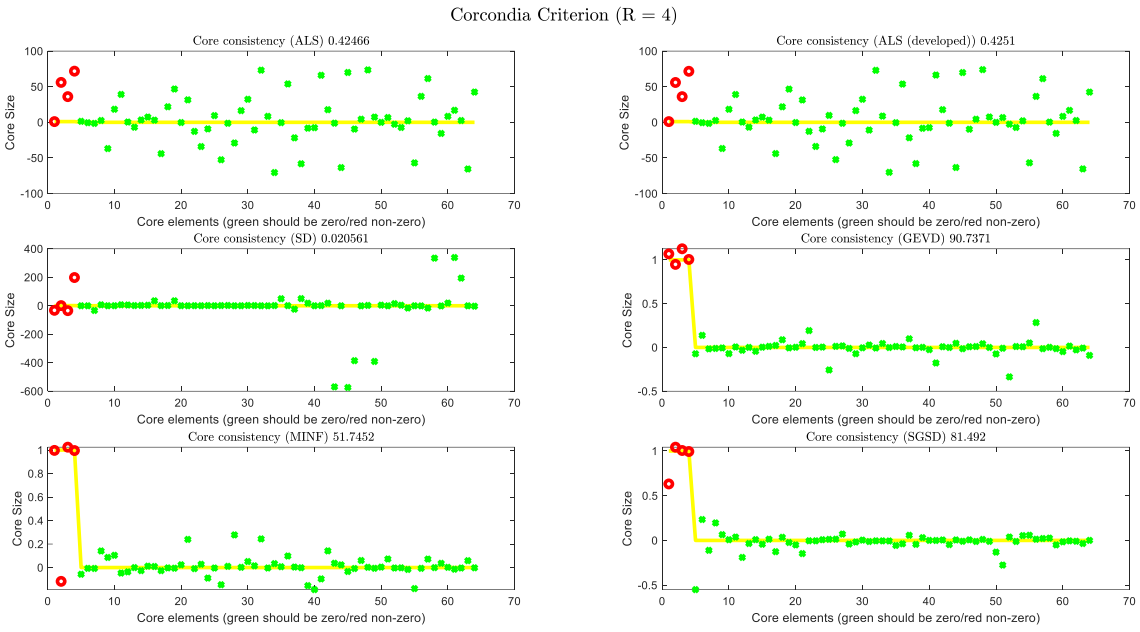
(ب

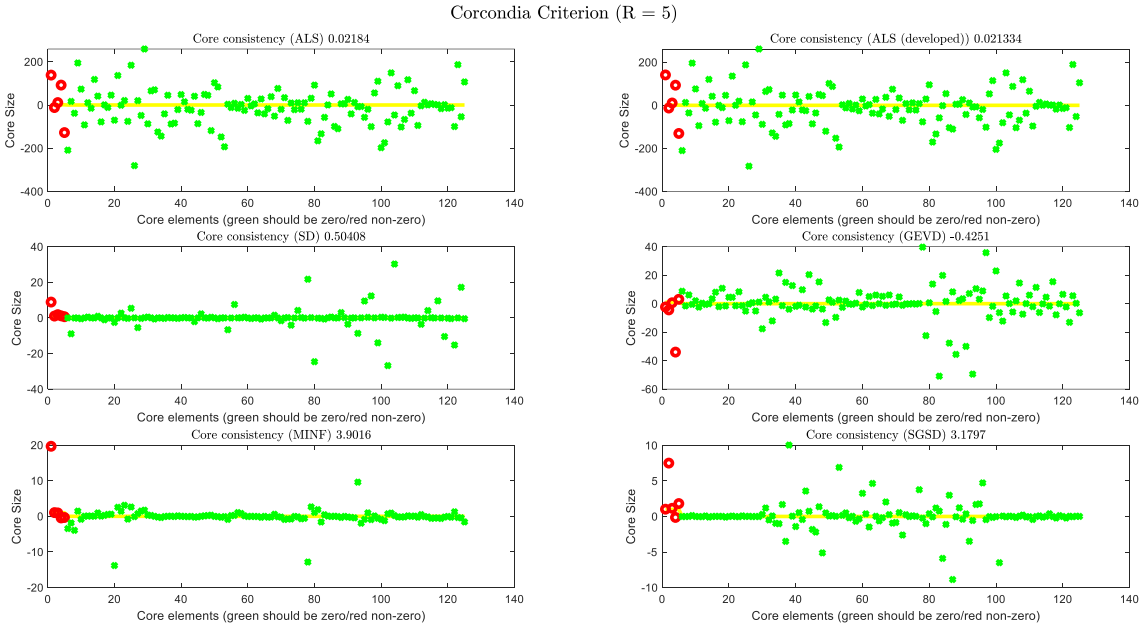
Corcondia Criterion (R = 2)



Corcondia Criterion (R = 3)







در جدول زیر مقادیر معیار corcondia را برای روش‌های مختلق و تعداد مولفه‌های مختلف ذکر کرده‌ایم.

R\algorithm	ALS	ALS (dev)	SD	GEVD	MINF	SGSD
R = 2	100	99.99	75.53	99.6985	100	99.99
R = 3	99.87	88.10	99.91	99.41	99.87	99.87
R = 4	0.42	0.43	0.02	90.74	51.75	81.49
R = 5	0.02	0.02	0.50	-0.43	3.90	3.18

با توجه به جدول بالا مشاهده می شود این معیار به ازای $R = 2,3$ برای همه الگوریتم‌ها مقدار خوبی دارد و با افزایش تعداد مولفه‌ها همانطور که در قسمت قبل هم بررسی کردیم، الگوریتم‌های ALS و SD کارایی خود را از دست می‌دهند. اگر تعداد مولفه‌ها را خیلی پرت انتخاب کنیم، تقریبا هیچ الگوریتمی نمی‌تواند منابع را به خوبی شناسایی کند. از بین روش‌های فوق، روش GEVD بهترین عملکرد را بین بقیه به ازای تعداد مختلف مولفه‌ها داشته است.