تمرین شماره ۳

درس تجزیه های تانسوری

امید شرفی

400201518

دکتر سپیده حاجی پور

May 14, 2022

سوال ۱

الف)

```
function [B, C] = nmf_als(A, j, B0, C0, itr)

    B = B0;
    C = C0;

    for i = 1:itr
         B = max(eps, (A * C') * pinv(C * C'));
         C = max(eps, pinv(B' * B)*(B' * A));
    end

end
```

ب)

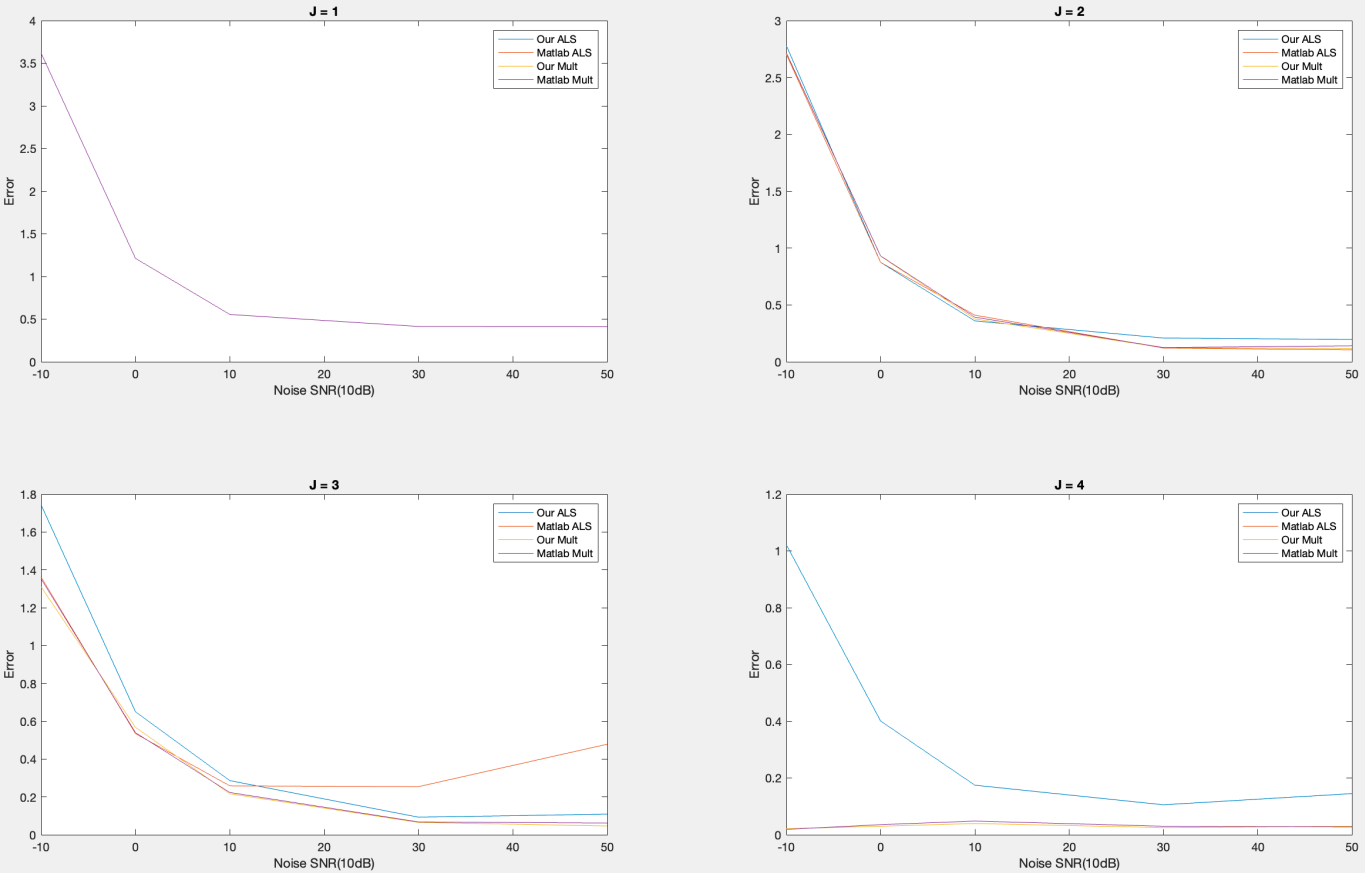
```
function [B, C] = nmf_mul(A, j, B0, C0, itr)

    B = B0;
    C = C0;

    for i = 1:itr
        B = B .* (A * C') ./ (B * C * C' + ones(size(B)) * eps);
        C = C .* (B' * A) ./ (B' * B * C + ones(size(C)) * eps);
    end

end
```

ج)

در ابتدا برای بررسی کلی الگوریتم، روی یک ماتریس رندوم ۱۰ در ۶ و بدون ترم E چهار روش را اعمال میکنیم و نتایج به شرح زیر به دست می آید. در J های کوچک خطای روش ها به یکدیگر نزدیک بوده و در ادامه برای J های بزرگتر خطای روش ما بیشتر از خطای متلب بوده و در عین حال نوسان خطا برای J ها بزرگتر بسته به شرایط اولیه بالاتر می‌رود.

```
Error J = 1
- Our ALS : 2.135645e+00
- Matlab ALS : 2.135645e+00
- Our Mult : 2.135645e+00
- Matlab Mult : 2.135645e+00

Error J = 2
- Our ALS : 1.480617e+00
- Matlab ALS : 1.481054e+00
- Our Mult : 1.480496e+00
- Matlab Mult : 1.485693e+00

Error J = 3
- Our ALS : 1.126601e+00
- Matlab ALS : 1.123553e+00
- Our Mult : 1.124126e+00
- Matlab Mult : 1.130523e+00

Error J = 4
- Our ALS : 7.350151e-01
- Matlab ALS : 7.321576e-01
- Our Mult : 7.881583e-01
- Matlab Mult : 8.004588e-01

Error J = 5
- Our ALS : 4.971142e-01
- Matlab ALS : 3.173935e+00
- Our Mult : 5.434737e-01
- Matlab Mult : 5.730635e-01

Error J = 6
- Our ALS : 4.859509e-01
- Matlab ALS : 6.133401e+00
- Our Mult : 2.647497e-01
- Matlab Mult : 2.351649e-01
```

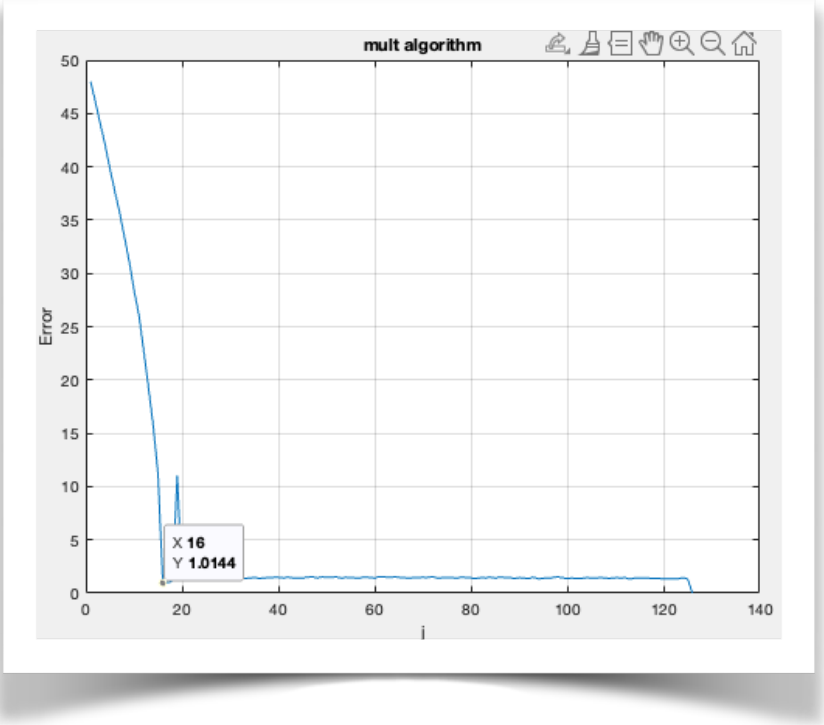
حال متناسب با خواسته‌ی سوال برای ۵ سطح نویز متفاوت و ۴ مقدار j (برای حفظ تقارن در شکل‌های subplot به جای ۲ تا ۴ از ۱ تا ۴ استفاده شد) هر کدام با ۱۰ مقدار دهی اولیه ی رندوم و میانگین گیری از خطای تخمین نمودارهای زیر حاصل می‌گردد.



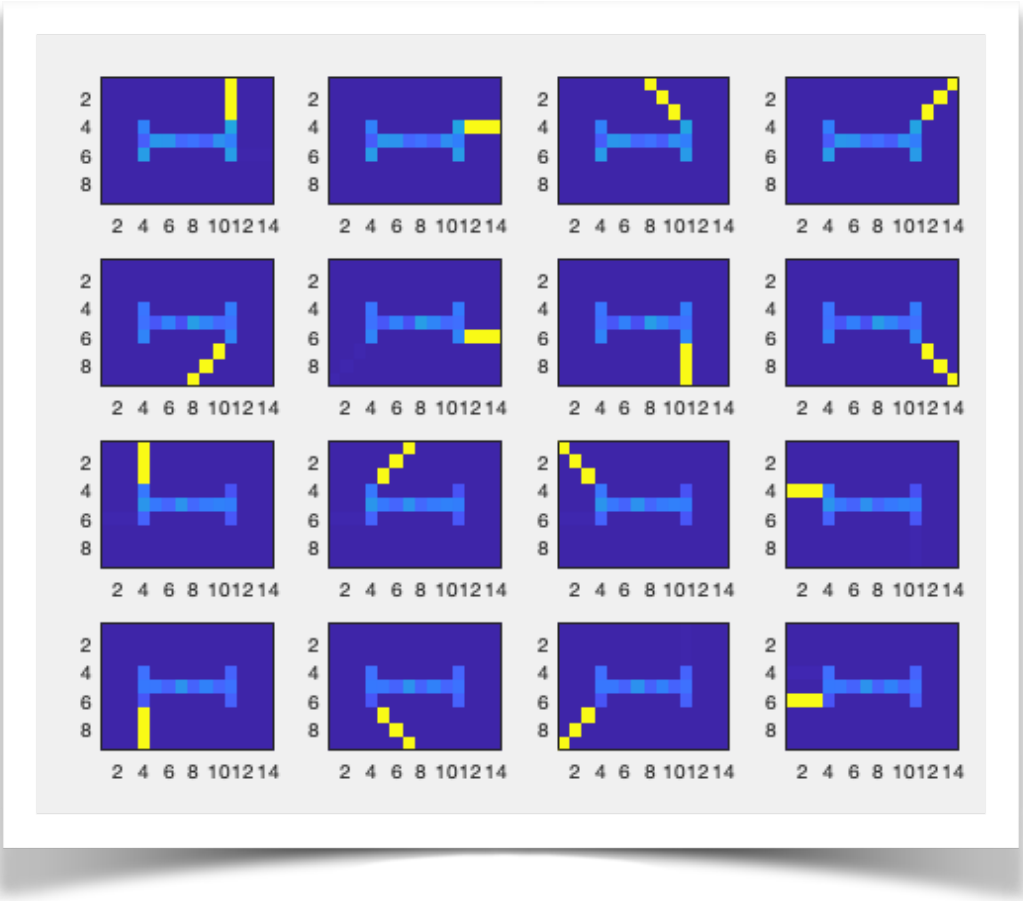
برای حجم نویز کمتر که SNR بیشتری داریم همانطور که انتظار داریم روند تغییر خطا به طور کل نزولی شده است. همچنین خطای تجزیه‌های تقریبا به یکدیگر نزدیک می باشند. با این حال برای J برابر ۳ که عملا حالت ساخت مساله‌ی ما است، روش Mul ما و متلب بهتر از روش های ALS بهتر عمل می‌کنند. همچنین در خطاهای کم روش پیاده سازی شده‌ی ما برای ALS بهتر از تابع متلب عمل می‌کنند. نکته‌ی نهایی، روش ALS متلب برای J برابر با ۴ که از بعد ۳ اصلی ساخته شده داده بزرگتر است مقدار NaN برمی‌گرداند.

سوال ۲

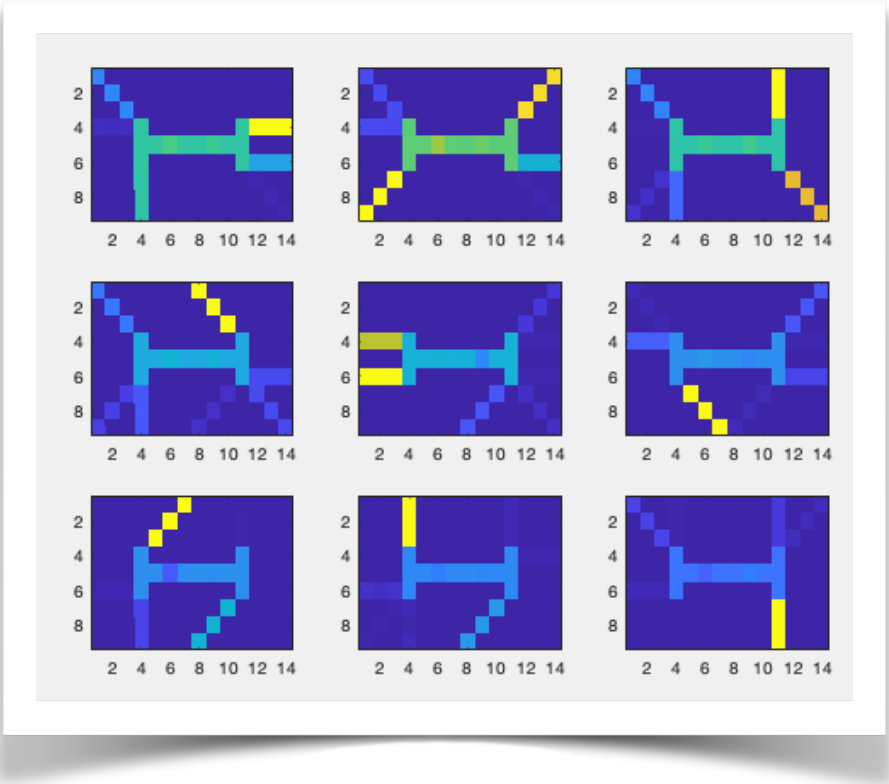
در ابتدا پس از reshape ماتریس‌های داخل هر سلول و ساخت ماتریس ۲۵۶ در ۱۲۶ که هر سطر آن یک تصویر وضعیت شناگر میباشد، با سرچ بر روی پارامتر j از ۱ تا ۱۲۶، نمودار زیر حاصل می‌شود.



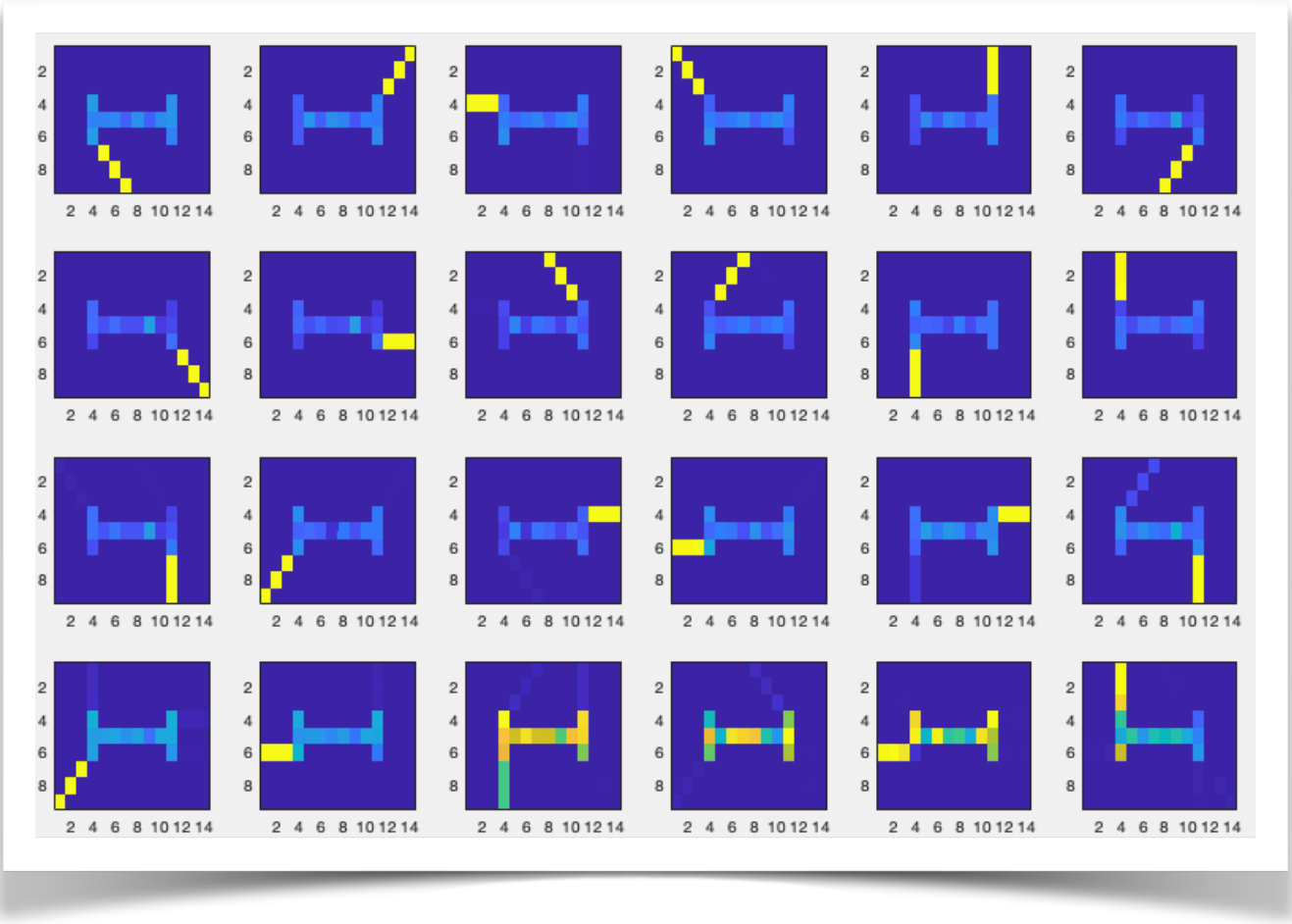
با چندبار اجرای کد، تقریبا مقدار پارامتر j برابر با ۱۶ به عنوان بهترین پارامتر روش به دست می‌آید. با اجرای کد برای j بهینه، ۱۶ وضعیت پایه ای که الگوریتم برای شناگر به دست می‌آورد به شرح زیر است. همانطور که انتظار داشتیم هر تصویر تقریبا وضعیت یک دست یا پا در یک زاویه مشخص میباشد و داده عملا از ترکیب این وضعیت ها ساخته می‌شود.



همچنین به طور مثال اگر J=9 را اجرا کنیم خطای تخمین بالاتر رفته و انگار تکرییبی از وضعیت شناگر در هر تصویر قرار گرفته است.

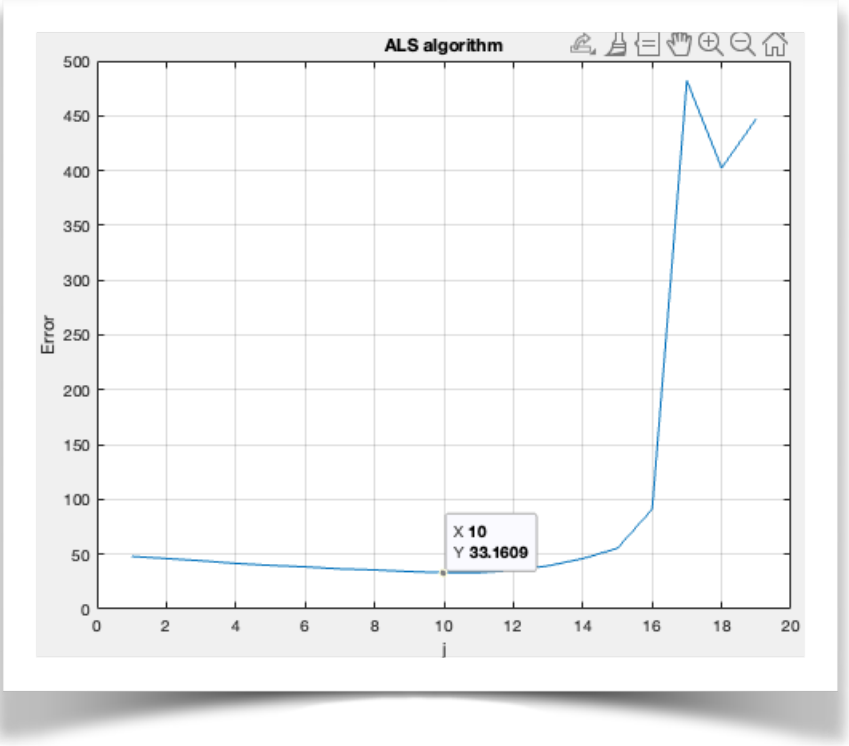


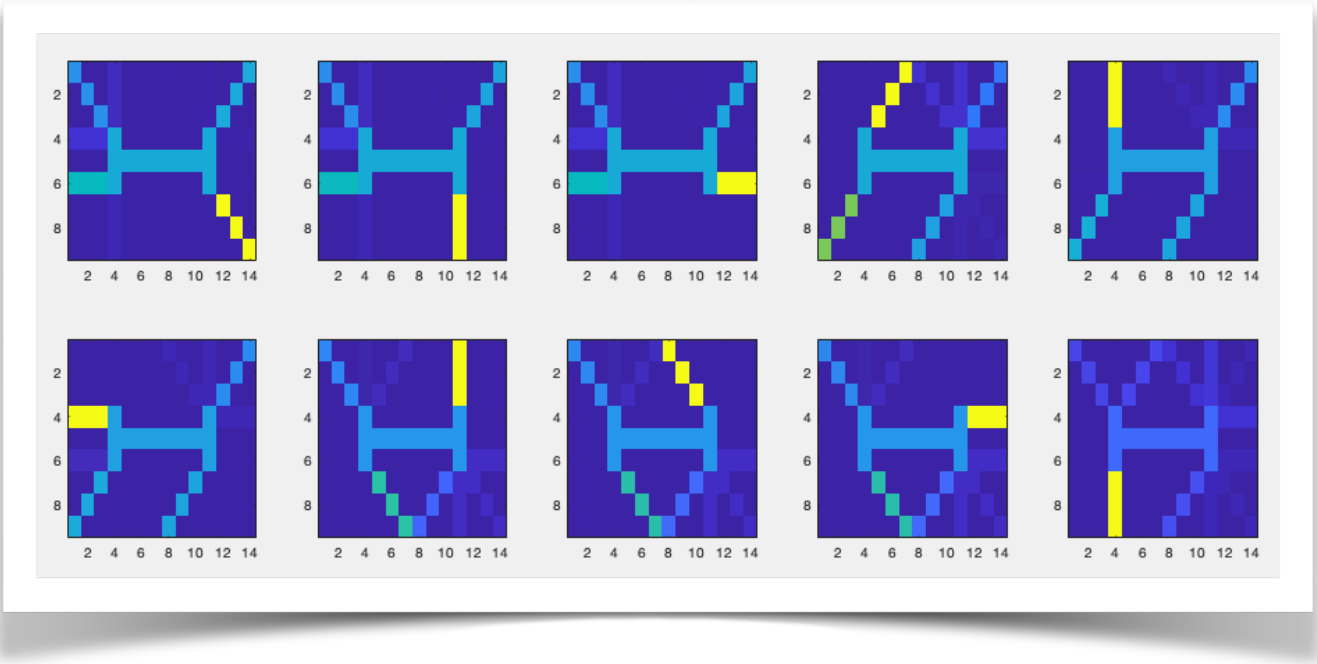
یا اگر J=24 قرار دهیم خطا مانند همان وضعیت ۱۶ بوده و صرفا تصاویر پایه ترکیبی اضافه‌تر ای در سطرهای C قرار گرفته است.



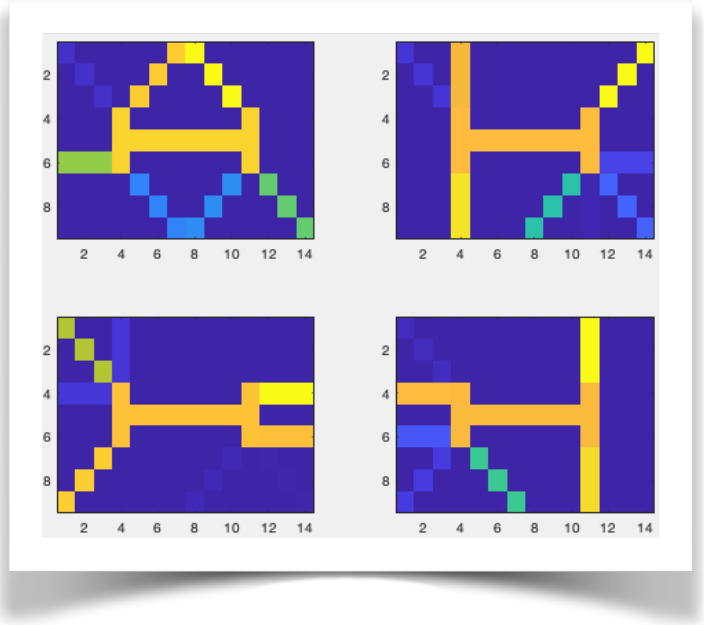
در روش ALS اوردر خطا به شدت بالاتر از روش Mul است و نوسانات شدیدی دارد. نمودار زیر رسم خطا برای J های کمتر از ۱۹ بوده که عملا مقدار ۱۰ بهترین پاسخ این روش می‌باشد.

برای حالت J=24 در این الگوریتم با توجه به آن که رنک ۱۷ را برای ماتریس به دست می‌آورد، عملا NaN شده و خطا شروع به افزایش می‌کند.

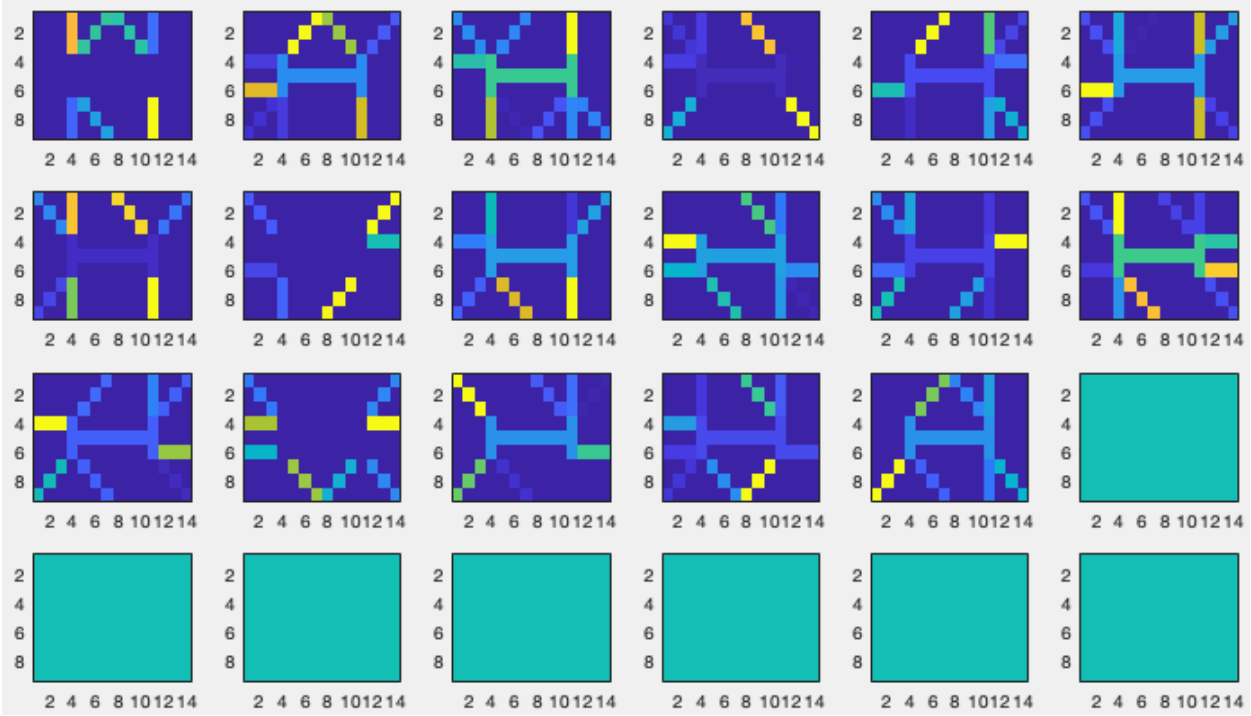




پاسخ حالات سطرهای ماتریس C برای الگوریتم ALS و J=۱۰



پاسخ حالات سطرهای ماتریس C برای الگوریتم ALS و J=۴



پاسخ حالات سطرهای ماتریس C برای الگوریتم ALS و J=۲۴