دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر و فناوری اطلاعات

گزارش تمرین هفتم درس جبر خطی کاربردی

نگارش

آرش حاجی صفی - 9631019

استاد درس

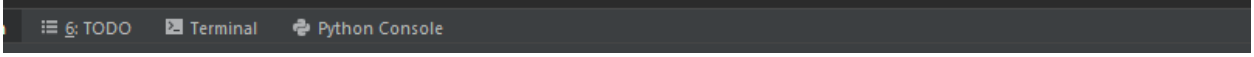
استاد امیرمزلقانی

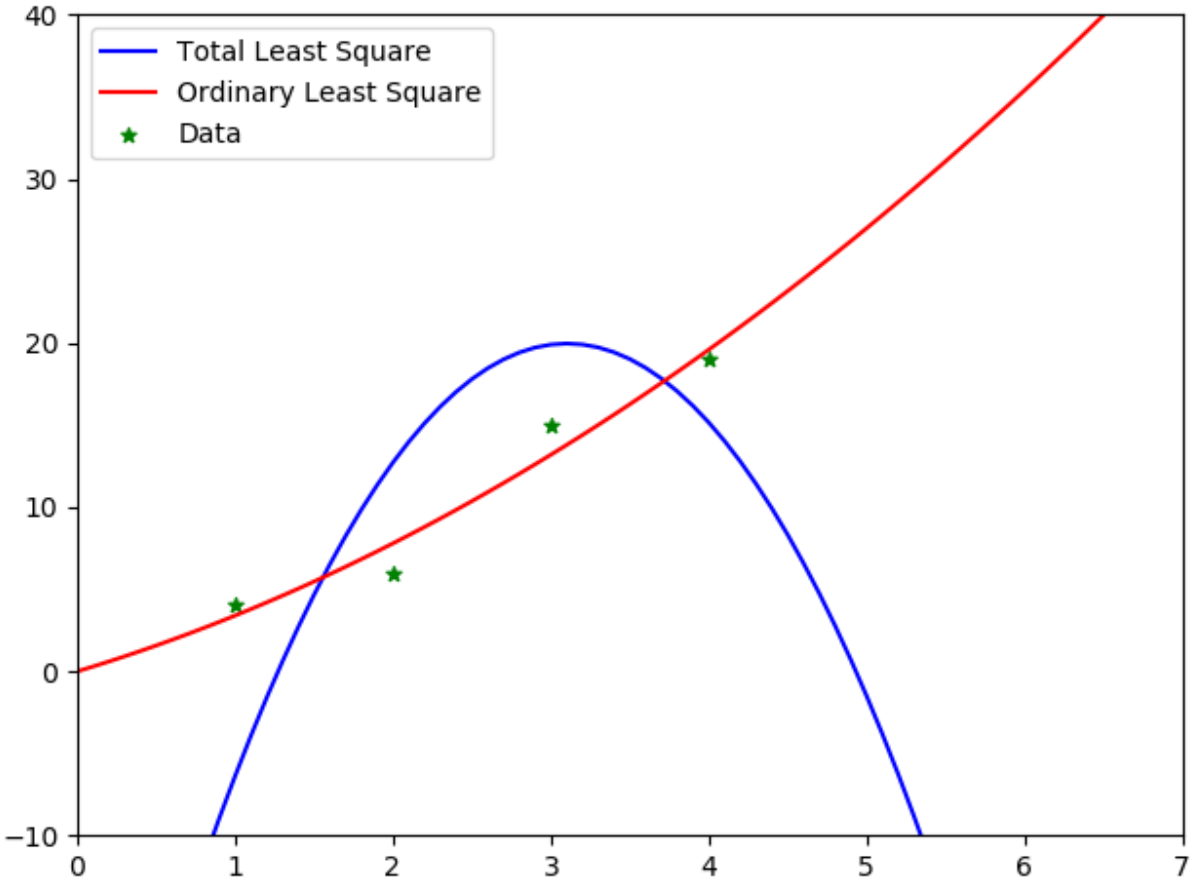
تیر 1398

در این تمرین به سوال‌های 1، 2 و 3 پاسخ داده شده است.

سوال 1. تابع totalLeastSquare(X, Y) را تعریف کرده‌ام که ماتریس X (در مثال صورت سوال همان A) و بردار Y (در مثال صورت سوال همان b) را به عنوان ورودی گرفته و الگوریتم پیدا کردن Total Least Square را اجرا کرده و بردار B که در مثال صورت سوال همان ضرایب چندجمله‌ای (بردار θ) است را برمی‌گرداند. ماتریس A و بردار b را با مقادیر گفته‌شده تشکیل داده و در کنسول پرینت کرده‌ام، سپس حاصل Least Square عادی را هم حساب کرده‌ام به روشی که در شبیه‌سازی تمرین 5 سوال 2 توضیح داده‌ام، در نهایت نقاط را به همراه چندجمله‌ای حاصل از Total Least Square و چندجمله‌ای حاصل از Least Square معمولی (Ordinary) با کتابخانه‌ی matplotlib رسم کرده‌ام.

```
C:\Users\Arash\venv\Scripts\python.exe "I:/Liniear Algebra/HW7/Codes/Q1/TLS.py"
A is:
 [[ 1  1  1]
 [ 1  2  4]
 [ 1  3  9]
 [ 1  4 16]]

b is:
 [[ 4]
 [ 6]
 [15]
 [19]]

Ordinary Least Square Answer is
 [[7.57912251e-14]
 [2.90000000e+00]
 [5.00000000e-01]]

Total Least Square Answer is:
 [[-37.40759389]
 [ 36.99079461]
 [ -5.96118439]]

Process finished with exit code 0
```





همینطور که از روی نمودار می‌بینیم، منحنی قرمز یا همان Least Square معمولی به داده‌های ما نزدیکتر است و خطای کمتری نسبت به Total Least Square دارد. علت این است که در Ordinary Least Square فرض می‌کنیم که یکی از متغیرها (در اینجا x) بدون خطا است و فقط y دارای خطا است که می‌خواهیم آنرا کمترین مقدار کنیم؛ در صورتی که در Total Least Square هم برای x و هم برای y خطا در نظر گرفته و آنرا کمترین مقدار می‌کنیم.

اگر هدف این باشد که با یک منحنی نزدیکترین تقریب ممکن به داده را به دست آوریم Ordinary Least Square مناسبتر است، اگر هدف این باشد که x و y دو متغیر مستقل فرض شوند و بخواهیم با یک منحنی آنهارا تقریب بزنیم Total Least Square مناسبتر است چون هر ضریبی یک معنای فیزیکی برای داده‌ها دارد.

سوال 2.

ابتدا ماتریس A و بردار b را با همان داده های سوال 1 تشکیل داده‌ام، سپس ماتریس [A b] را تشکیل داده‌ام و آنرا Z نامیده‌ام و تجزیه SVD ماتریس Z را پیدا کرده‌ام. حاصل Z^TZ را حساب کرده‌ام و آنرا ماتریس C نامیده‌ام. تابع inverse_power_method(A, a) را که در شبیه‌سازی تمرین سری پنجم نوشته بودم در اینجا استفاده کرده‌ام، به این صورت که ماتریس C و یک تقریب نزیک به مقدارویژه و بردار ویژه‌ی خواسته شده را می‌گیرد و نتیجه را برمی‌گرداند. چون کوچکترین مقدار ویژه و بردار ویژه‌ی متناظر با آنرا می‌خواهیم، برای تقریب نزدیک به مقدار ویژه، مقدار 1e-10 را داده‌ام که همان 10^{-10} می‌باشد. به این ترتیب کوچکترین مقدار ویژه و بردار ویژه متناظر با آنرا بدست آورده و حاصل آنرا پرینت می‌کنم:

```
C:\Users\Arash\venv\Libarary\Scripts\python.exe "I:/Liniear Algebra/HW7/Codes/Q2-/Q2.py"
A is:
 [[ 1  1  1]
 [ 1  2  4]
 [ 1  3  9]
 [ 1  4 16]]

b is:
 [[ 4]
 [ 6]
 [15]
 [19]]

[A b] is:
 [[ 1  1  1  4]
 [ 1  2  4  6]
 [ 1  3  9 15]
 [ 1  4 16 19]]

C is:
 [[  4  10  30  44]
 [ 10  30 100 137]
 [ 30 100 354 467]
 [ 44 137 467 638]]
--Inverse power Iteration Method--

a guess selected for 'a' is:  1e-10


Minimum Eigen Value for [A b] using inverse power method is:
 0.06838080102270264

Minimum Eigen Vector for [A b] using inverse power method is:
 [[ 0.70640965]
 [-0.69853876]
 [ 0.11257175]
 [ 0.01888412]]

Process finished with exit code 0
```

سوال 3.

تابع compressImage(imgLocation, n) را نوشته‌ام که آدرس فایل و بعد rank مورد نظر برای فشرده‌سازی عکس را گرفته، عکس مورد نظر را ابتدا سیاه و سفید کرده و سایزش را به 100*100 تغییر می‌دهد و سپس ماتریس متناظر آنرا که یک ماتریس 100*100 است تشکیل می‌دهد که آنرا A می‌نامم.

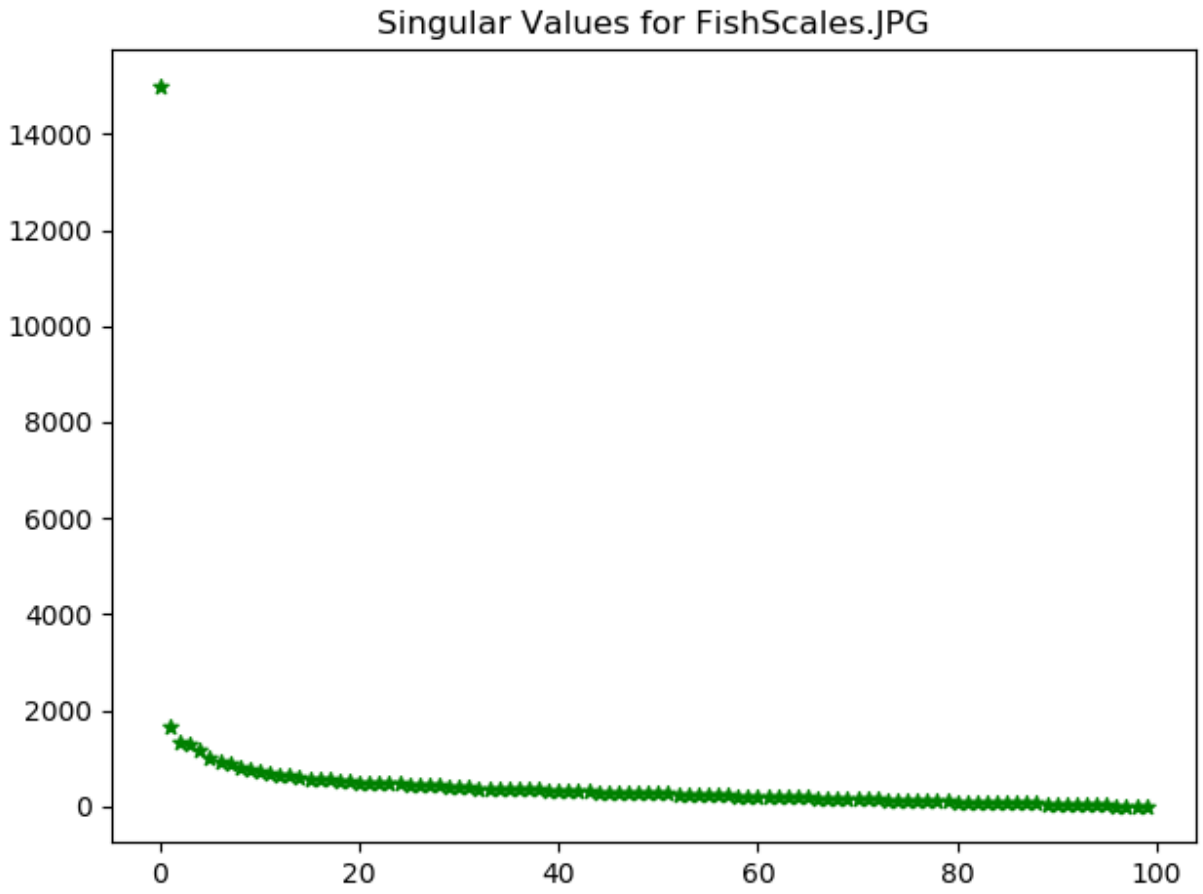
عکس اصلی را نشان داده و سپس تجزیه SVD ماتریس A را انجام داده‌ام. تجزیه SVD را اگر ضرب کرده و باز کنیم به این صورت است:

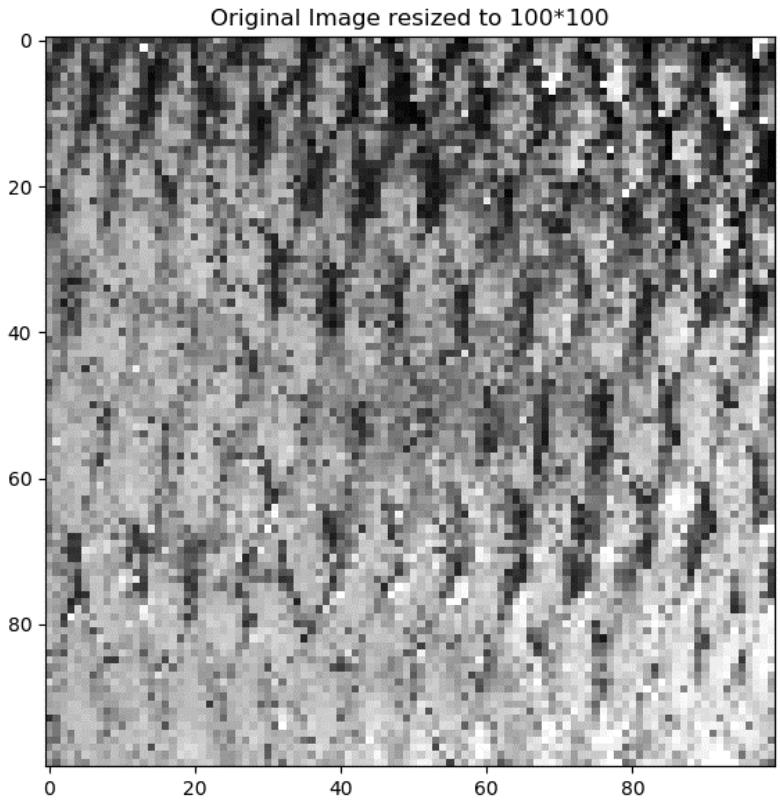
$$\boldsymbol{A = \sum_{i=1}^{100} \sigma_i u_i v_i^T}$$

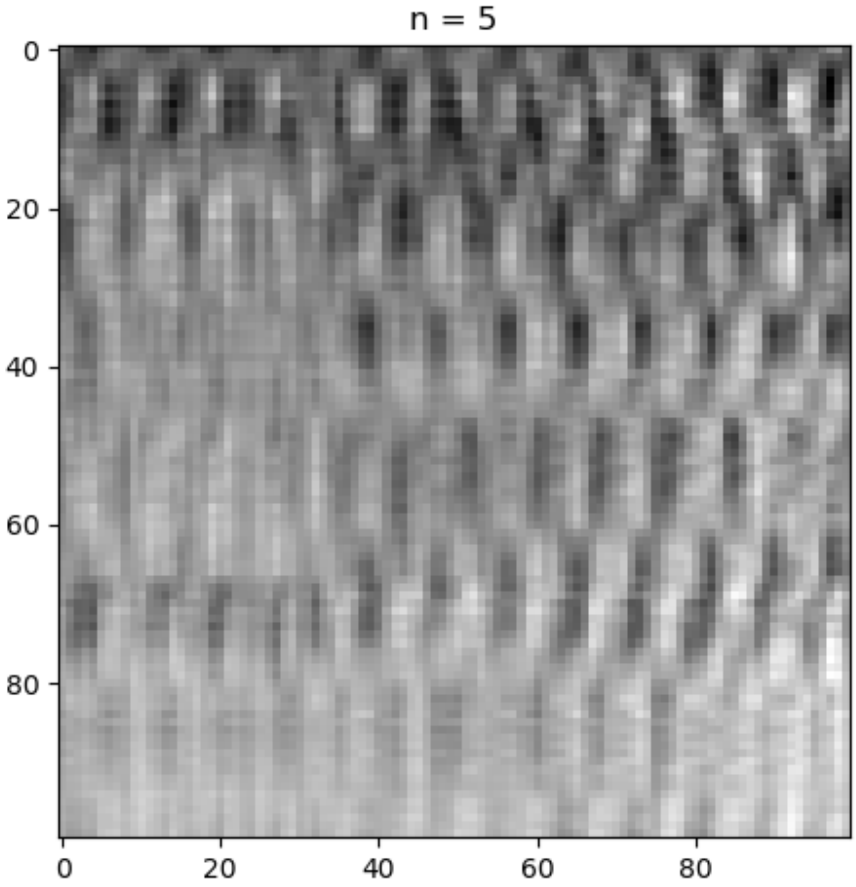
ماتریس S که مقادیر $\boldsymbol{\sigma_i}$ روی آن می‌باشند به صورت نزولی تشکیل شده، یعنی بزرگترین مقدار تکین درایه اول است و u_1 و v_1 متناظر با بزرگترین مقدار تکین هستند و الی آخر. بنابراین برای تقریب ماتریس A با n ستون که در صورت سوال گفته شده، کافی است که n جمله‌ی اول سیگمای گفته شده در بالا را حساب کنیم که بیشترین اثرگذاری را دارند. در ادامه در تابع compressImage پس از تجزیه SVD، طبق گفته‌ی صورت سوال مقادیر تکین را که همان درایه های ماتریس S هستند، رسم کرده و سپس با توجه به n گرفته شده در argumentهای تابع، سیگمای گفته شده تا n جمله‌ی اول را تشکیل می‌دهم و ماتریس تصویر تشکیل شده را نمایش می‌دهم.

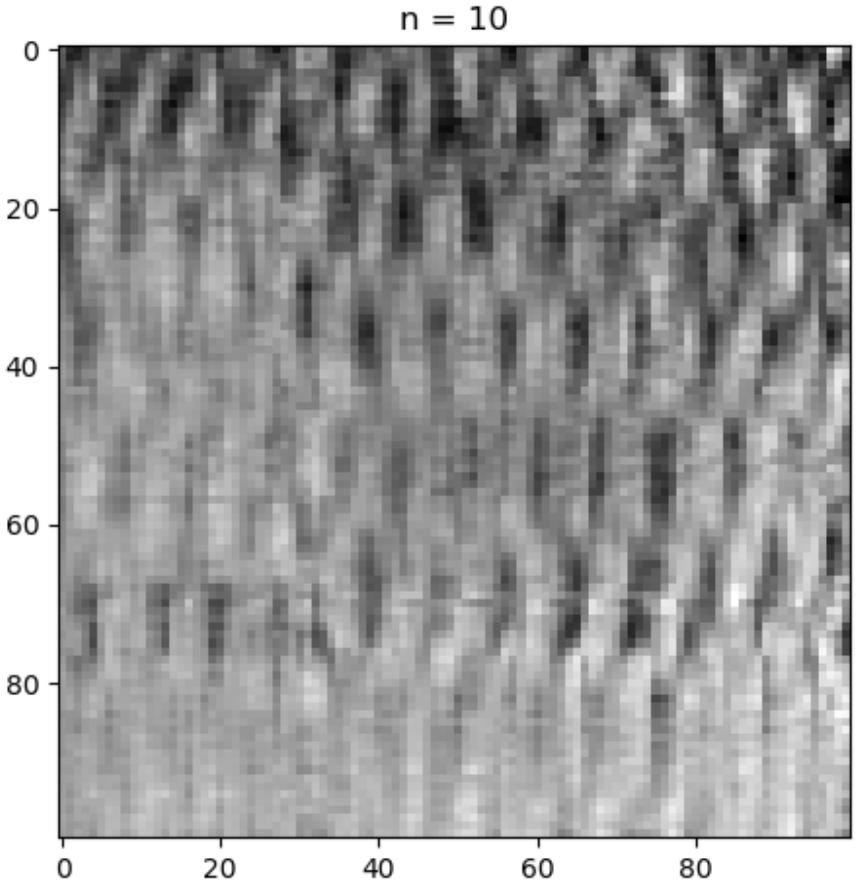
نتایج: (رابطه‌ی بین مقادیر تکین و کیفیت عکس هم در صفحه‌ی آخر گزارش گفته شده)

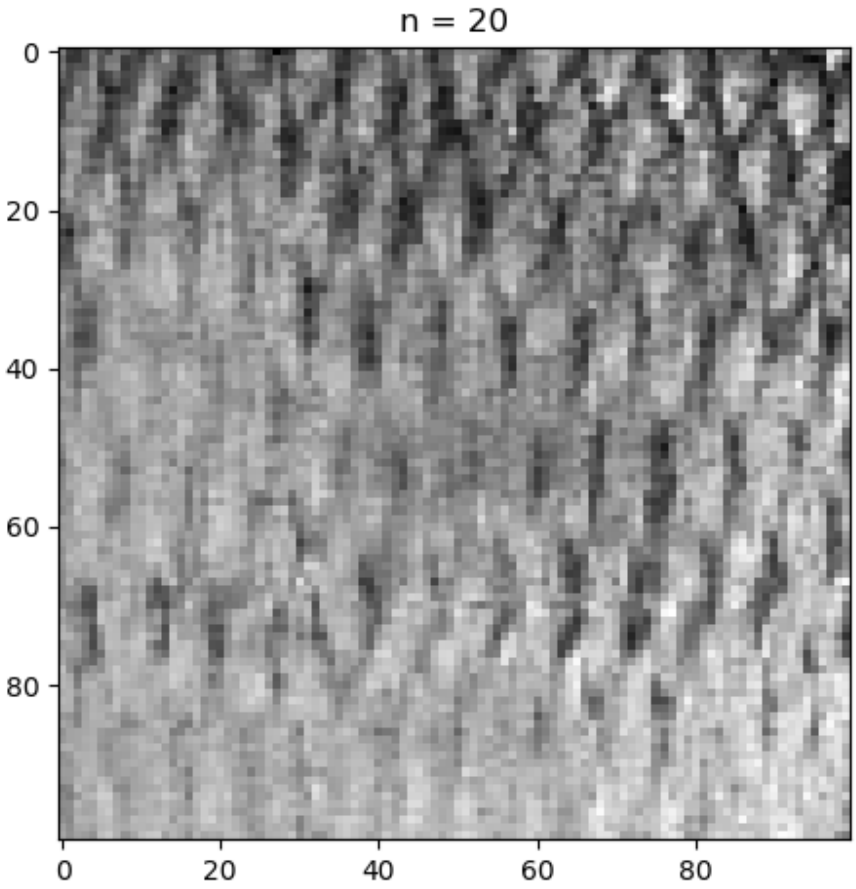
تصویر اول (FishScales.JPG):



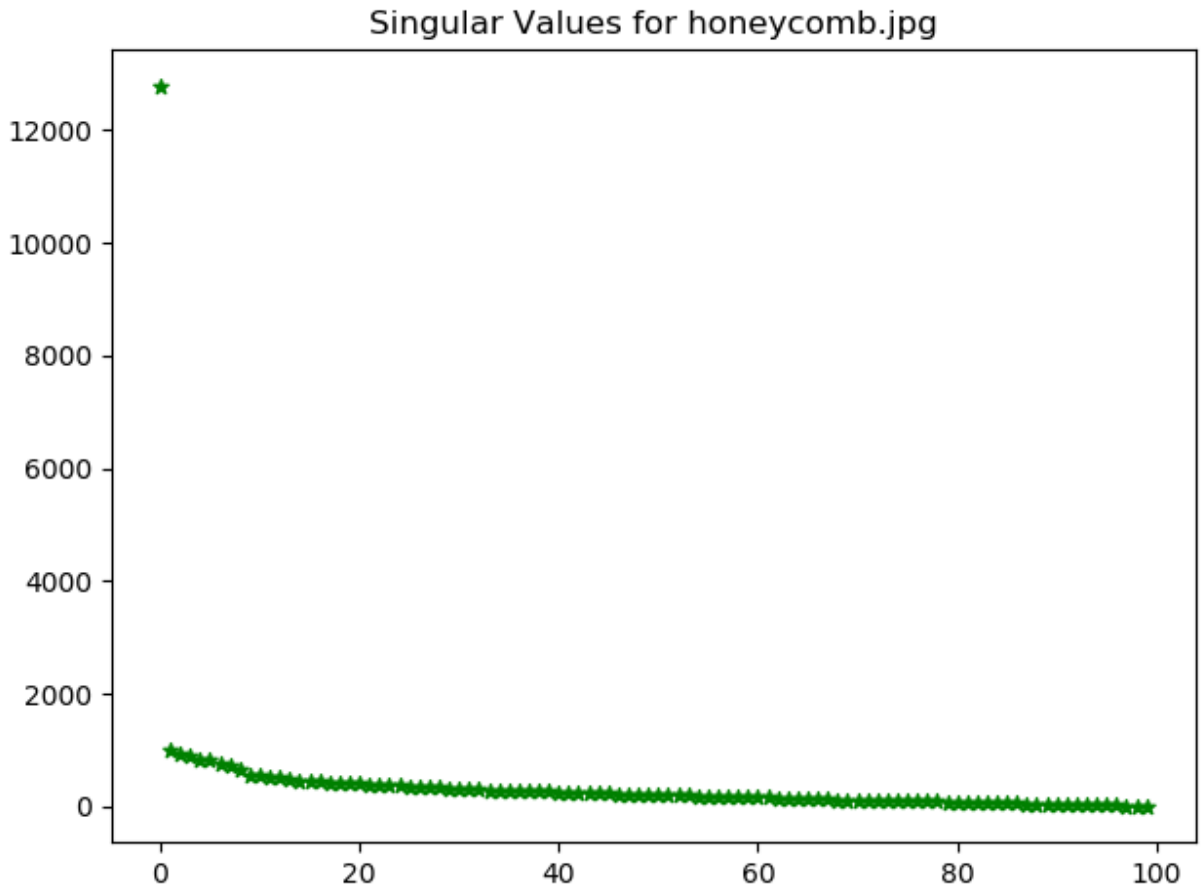

Original Image resized to 100*100
0
20
40
60
80
0
20
40
60
80

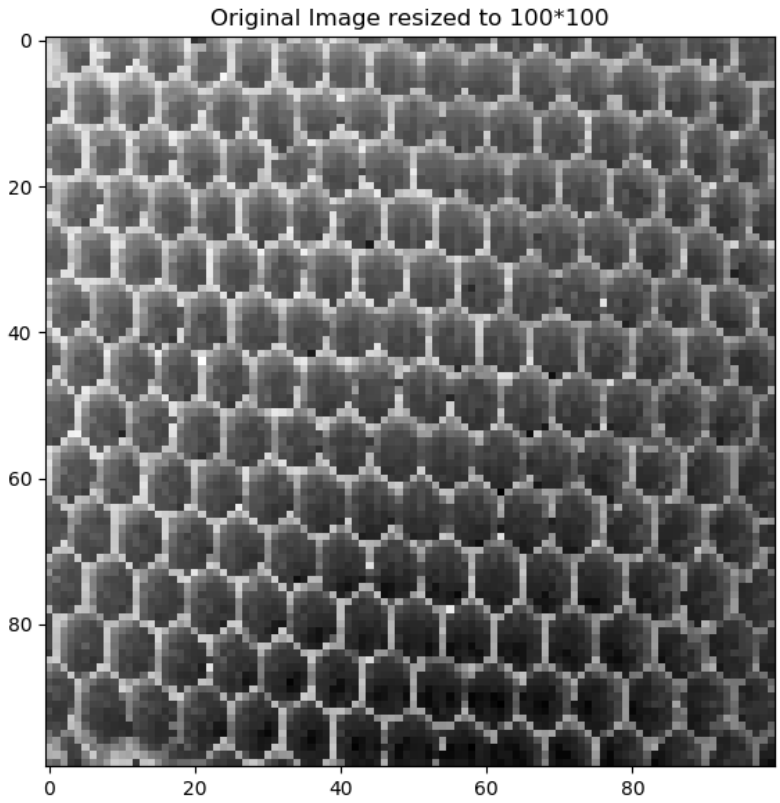

n = 5
0
20
40
60
80
0
20
40
60
80

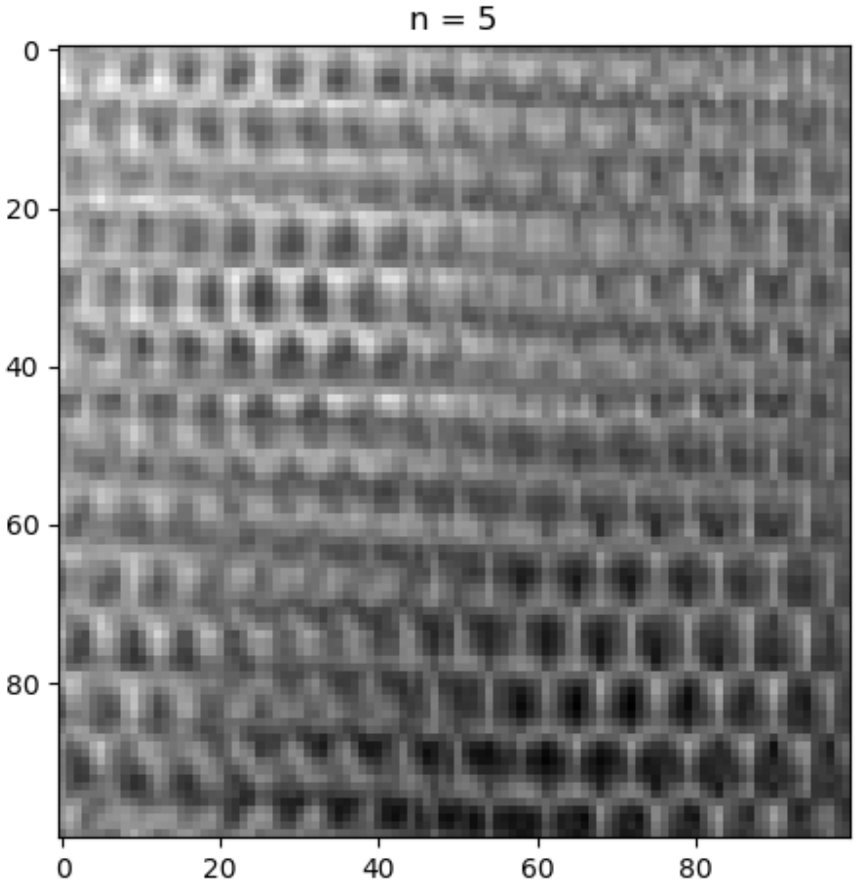

n = 10
0
20
40
60
80
0
20
40
60
80

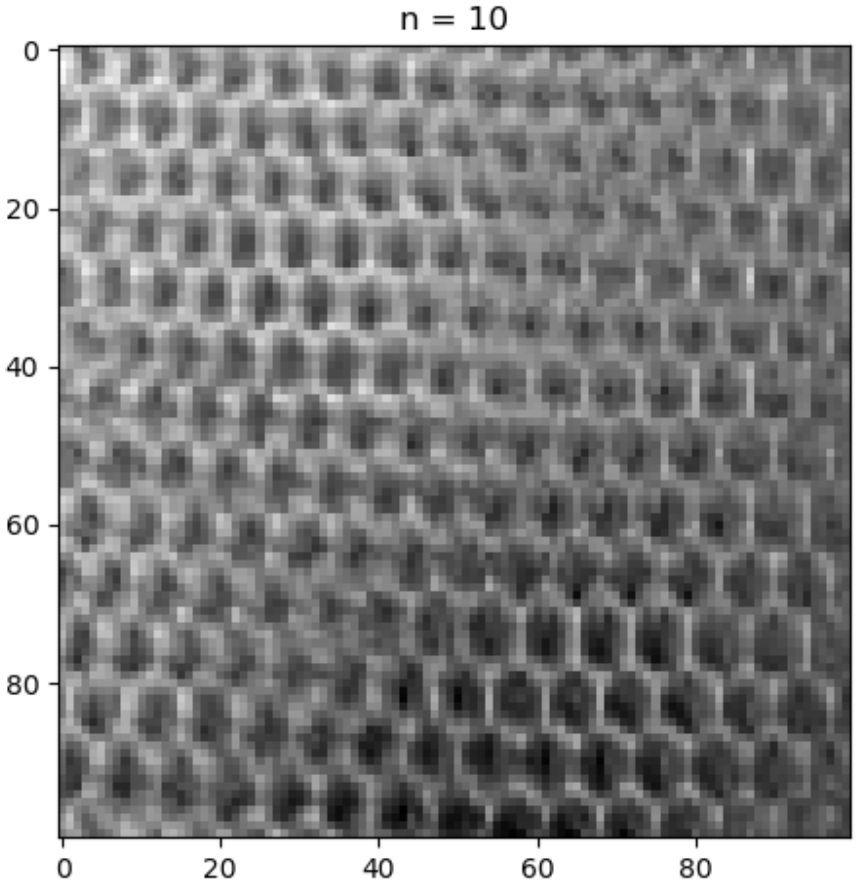

n = 20
0
20
40
60
80
0
20
40
60
80

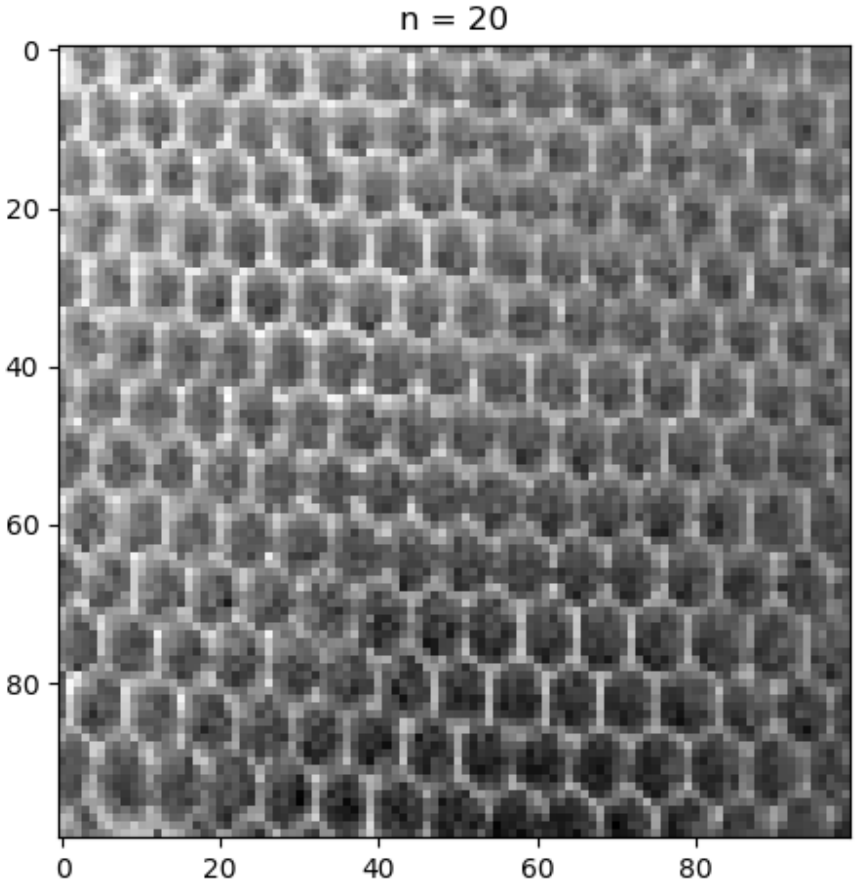
تصویر دوم (honeycomb.jpg):



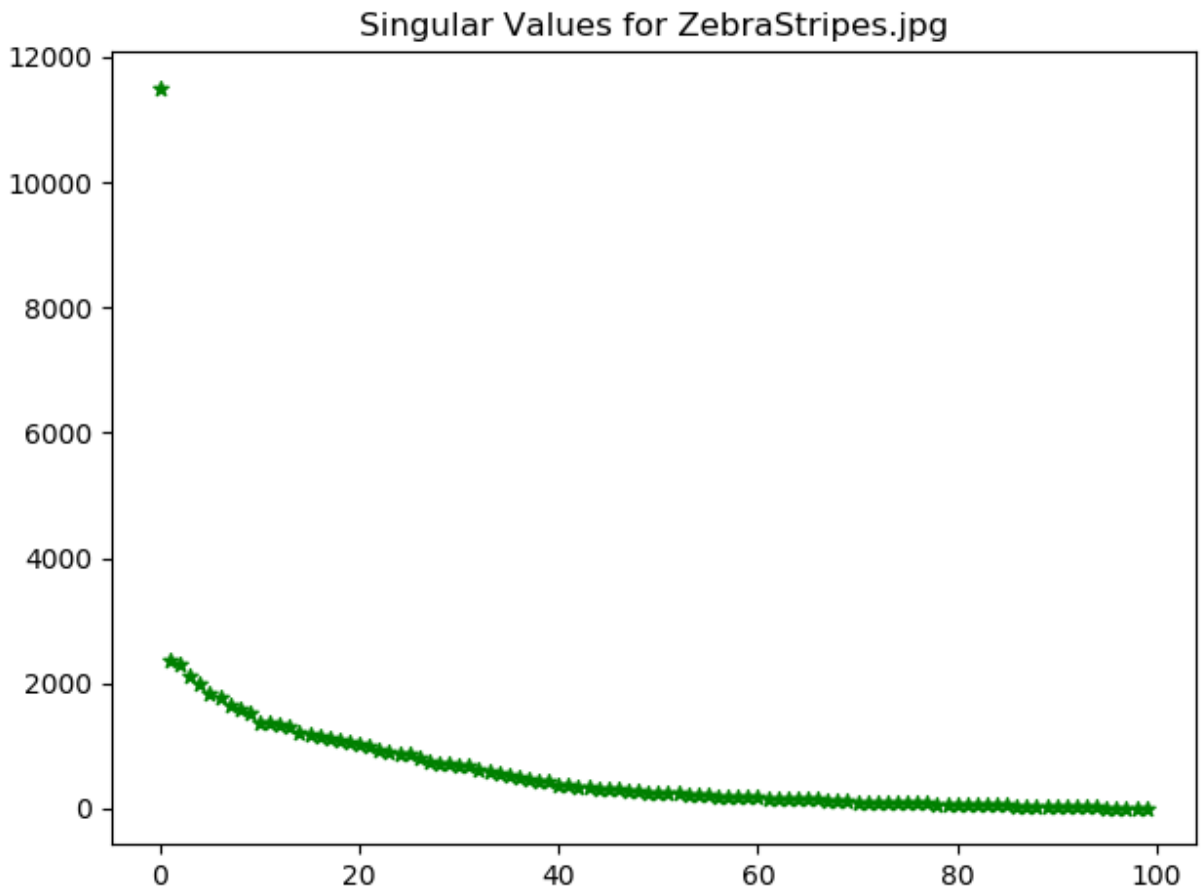

Original Image resized to 100*100
0
20
40
60
80
0
20
40
60
80

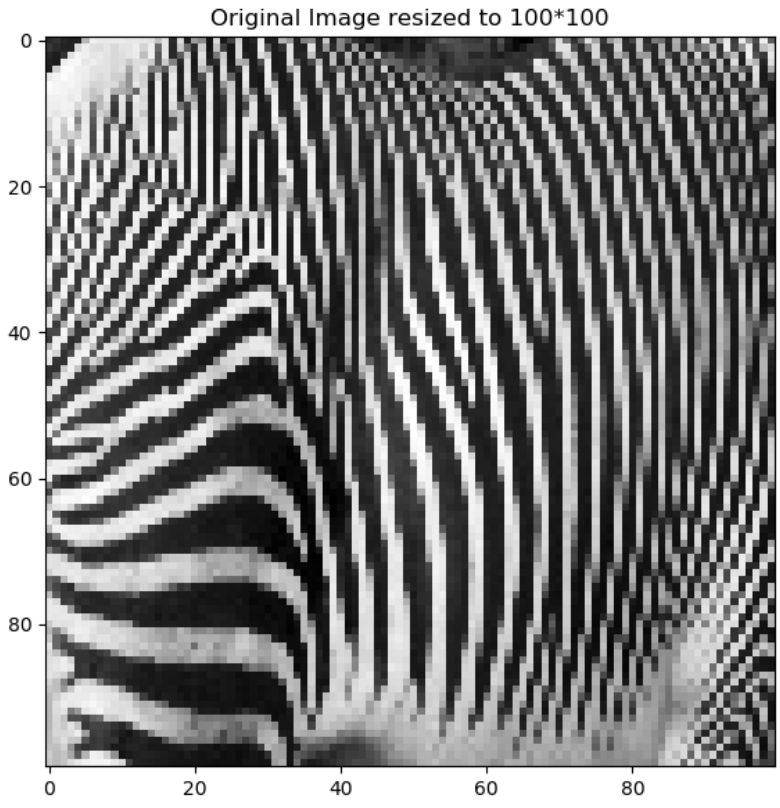

n = 5
0
20
40
60
80
0
20
40
60
80

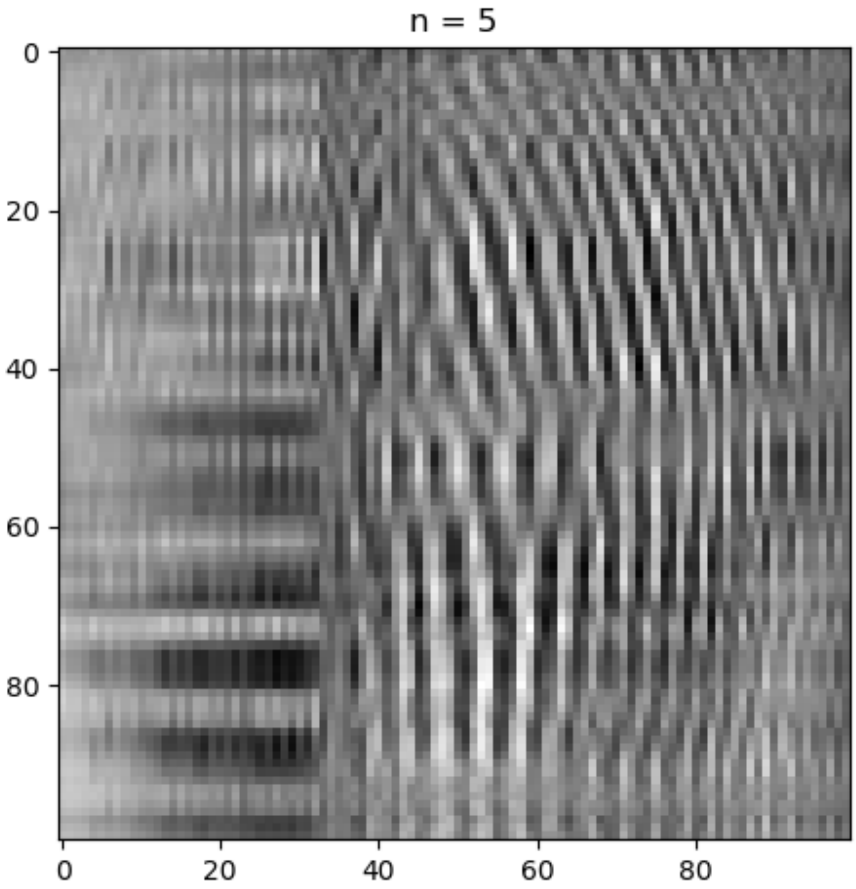

n = 10
0
20
40
60
80
0
20
40
60
80

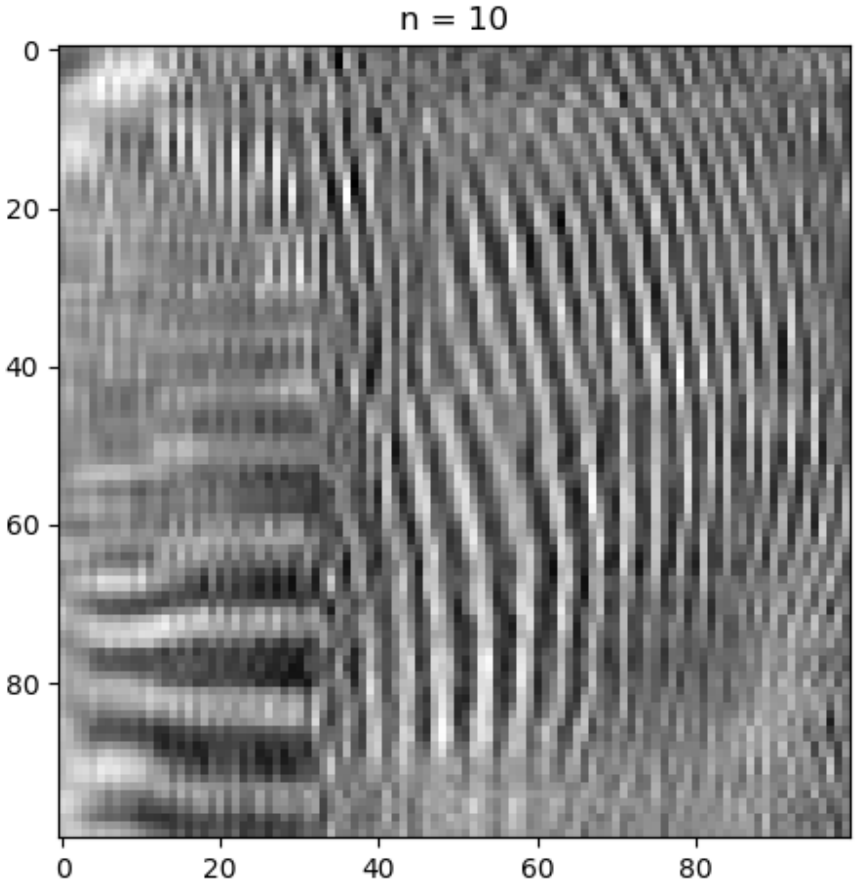

n = 20
0
20
40
60
80
0
20
40
60
80

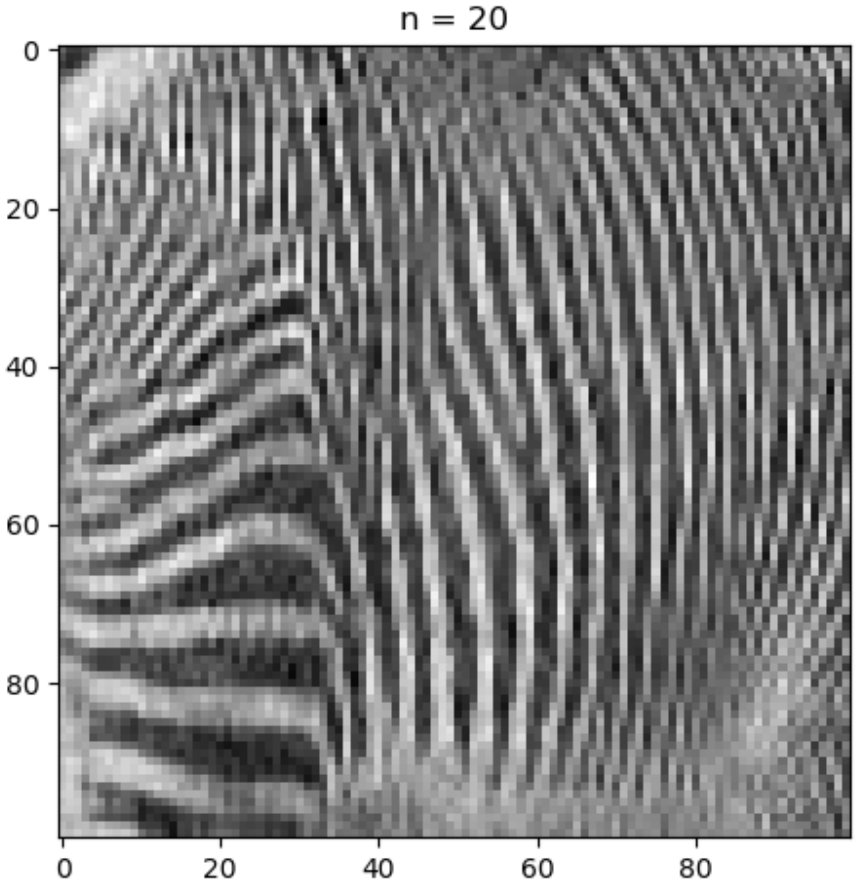
تصویر سوم (ZebraStripes.jpg):




Original Image resized to 100*100
0
20
40
60
80
0
20
40
60
80


n = 5
0
20
40
60
80
0
20
40
60
80


n = 10
0
20
40
60
80
0
20
40
60
80



رابطه‌ی بین مقادیر تکین و کیفیت عکس:

همانطور که مشاهده می‌شود، هرچه تعداد بیشتری جمله از سیگمای گفته شده در تجزیه SVD حساب شود، یعنی n بیشتر شود، کیفیت عکس بیشتر می‌شود. همچنین جمله‌های ابتدای که مقادیر تکین متناظر آنها بزرگتر است، خیلی تاثیر بیشتری روی کیفیت عکس می‌گذارند تا جمله‌های بالاتر. یعنی هرچه مقدار

تکین بیشتر باشد، جمله‌ی متناظر آن کلیت عکس را بهتر نشان می‌دهد. در ضمن تفاوت بین عکسی که در آن $n=5$ است با $n=10$، بیشتر از تفاوت بین $n=10$ تا $n=20$ است، چون همانطور که گفته شده جمله‌های ابتدایی تاثیر بیشتری در کیفیت عکس دارند تا جمله های انتهایی.