Proiect Metode Numerice

Aplicatie DVS -Compresia de imagini color

Student: Dobran Andrei, 322AC

Cuprins

- Introducere
- Descompunerea valorilor singulare
- Compresia color(RGB) al unei imagini
- Prezentarea aplicatiei
- Grafice
- Bibliografie

Introducere

- Imaginile sunt predominante pe Internet. Deși este un mod excelent de a comunica informații este și o formă costisitoare de date pentru transportul prin retea.
- Pentru a rezolva această problemă, software-ul de compresie este adesea folosit pentru a codifica o imagine , astfel încât să poată fi transportat mai eficient, reducând în același timp modificările de calitate perceptibile din copia originala.
- În acest proiect vom explora un algoritm de compresie a imaginii bazat pe găsirea descompunerii valorii singulare a unei matrice. Pentru o măsurare obiectivă, introducem un indice de similaritate structurală a imaginii cu algoritmul, astfel încât acesta să poată ajusta raportul de compresie printr-o buclă de feedback.
- Compresia fara pierderi este preferata in scopul arhivarii, adesea pentru imaginea medicala,desene tehnice.
- Metodele de compresie cu pierderi mai ales la volum de date reduse introduc artefacte de compresie. Metodele cu pierderi sunt potrivite în special pentru imaginile naturale,fotografiile in aplicatii in care este acceptabila o pierdere minora de fidelitate pentru a obtine o reducere substantiala a volumului de date.

Descompunerea valorilor singulare

- Descoperirea DVS-ului este atribuita multor matematicieni: Beltrami, Jordan, Sylvester, Schmidt si Weyl.
- O multitudine de aplicatii pot fi rezolvate cu DVS: de la tendintele de vot ale parlamentarilor, la procesarea de imagine! DVS reduce eficient sisteme cu date masive la probleme mai mici, prin eliminarea informatiei redundante si retinerea datelor importante
- De asemenea, algoritmul de calcul DVS reprezintă unica modalitate, numeric fiabilă, de determinare a rangului unei matrice
- Calculul DVS este bazat pe ortogonalitate $\Rightarrow$ proprietati numerice remarcabile pentru toate aplicatiile ce apeleaza DVS!

Descompunerea valorilor singulare-formulare

- **Teorema:** Pentru $A \in R^{m*n}$, exista matricele ortogonale $U \in R^{m*m}$ si

$V \in R^{n*n}$ si r = rang(A) astfel încat :

$$U^t A V = \Sigma = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix},$$

unde $\Sigma \in R^{m*n}$, astfel incat $\Sigma_1 = \mathrm{d}iag(\sigma_1, \cdots, \sigma_r) \in R^{r*r}$, si consideram ordinea valorilor singulare $\sigma_1 > \sigma_2$>...$\geq \sigma_r > 0$

$A = U * \Sigma * V^t$ *se numeste descompunerea valorilor singulare,*

Iar numerele nenegative σ_i se numesc valori singulare.

Coloanele matricei ortogonale U se numesc vectori singulari la stanga, iar coloanele lui V se numesc vectori singulari la dreap*ta*.


A
=
U
·
Σ
·
V T

Consecinta:

Orice matrice poate fi scrisa ca o suma de produse externe produse externe de vectori singulari ponderate cu valori singulare

$$A = \sum_{i=0}^{r} (\sigma_i * u_i * v_i^t); \quad \mathrm{u_i}, v_i^t - \text{componente principale}$$

Rangul r al matricei A ∈ R^{m*n} este numarul maxim de coloane liniar independente din A, sau echivalent: r = rangA = dim(Im(A)).

Cum rang(A) = rang (A^t), numarul de coloane sau linii liniar independente intr-o matrice este acelasi. Deci rangul lui A este dimensiunea maxima a unei submatrice inversabile A(I, J) a lui A.

Teorema: Daca A ∈ R^{m*n}, atunci exista matricele ortogonale U ∈ R^{m*m} si V ∈ R^{n*n} a.i. $A = U * \Sigma * V^t$, unde $\Sigma \in R^{m*n}$ este o matrice diagonala Σ = diag($\sigma_1, \cdots, \sigma_{\min(m,n)}$) cu $\sigma_1 > \sigma_2 > \ldots \geq \sigma_{\min(m,n)} > 0$, valorile singulare.

Compresia color(RGB) a unei imagini

- Pixeli – elementul de bază al imaginii. (0-negru,255-alb)
- Rezoluţia imaginii – Numărul de pixeli din imaginea digitală

Imagini ''true color''

1.Se alocă 24(8*3) biţi(cate 8 biti pentru fiecare componenta RGB (Red,Green,Blue).

2. Pot fi reprezentate 256 x 256 x 256 culori (16.777.216)

3 .Pixelul conţine indexul culorii din paleta de culori.

4. O histograma = tabel care da numarul de pixeli care au aceeasi valoare intr-o imagine

RGB(Red-Green-Blue) = model aditiv de culoare, în care culorile albastră, roșie și verde sunt amestecate în diferite moduri pentru a produce o gamă largă de culori

- În această aplicatie vom utiliza algoritmul DVS în analiza imaginii. Vom folosi în principal DVS pe imagini pentru a obține componente principale/vectori singulari care captează imaginea și vom folosi unele dintre ele pentru a compresa imaginea.
- Vom folosi funcția svd a modulului linalg de la NumPy pentru a efectua descompunerea valorii singulare (SVD) pe matricea de imagine.

```
def SVD (B, k):
    U, Sigma, V = np.linalg.svd(B.copy())
    compressed = np.matrix(U[:,:k]) * np.diag(Sigma[:k]) * np.matrix(V[:k,:])
    return U,Sigma,V
```

- In continuare voi prezenta functia de compresare a unei imagini :

Am afisat cu ajutorul functiei nbytes spatiul in bytes de care avem nevoie pentru a stoca imaginea introdusa si cu ajutorul functiei shape .

```
original_bytes = image.nbytes
print("Spatiul(in bytes) pentru a stoca aceasta imagine este = ", original_bytes)
image = image/255
linie,coloana,_= image.shape
```

Pentru imaginile colorate avem o matrice tridimensională cu dimensiunea n × m × 3, unde n și m reprezintă numărul de pixeli pe verticală și respectiv pe orizontală, iar pentru fiecare pixel stocăm intensitatea pentru culorile roșu, verde și albastru. Matricea tridimensională rezultată va fi o bună aproximare a imaginii originale.

- Vom împarti matricea in 3 matrice 2D(RGB), deoarece SVD se aplica doar pe matrice 2D

```
Red = image[:, :, 0]
Green = image[:, :, 1]
Blue = image[:, :, 2]
print(Red)
print(Green)
print(Blue)
```

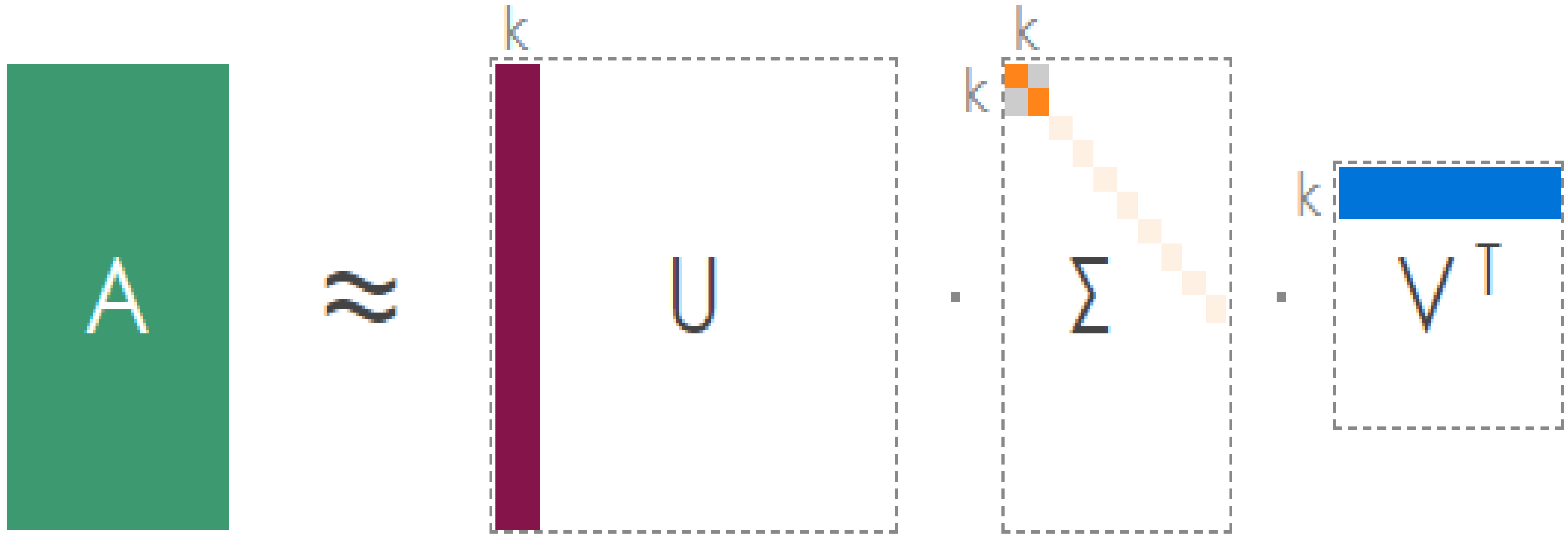
In continuare vom aplica functia SVD pe cele 3 matrice:

```
U_Red, Sigma_Red, V_Red = SVD(Red, k)
U_Green, Sigma_Green, V_Green =SVD(Green, k)
U_Blue, Sigma_Blue, V_Blue = SVD(Blue, k)
```

- In continuare vom afisa dimensiunea totala in bytes a matricelor pe care le stocam:

```
bytes_matrices =sum([matrix.nbytes for matrix in [U_Red, Sigma_Red, V_Red, U_Green, Sigma_Green,
                                                   V_Green ,U_Blue, Sigma_Blue, V_Blue]])
print("Matricile pe care le stocăm au dimensiunea totală (în bytes)",bytes_matrices)
```

Selectam doar k valori unice pentru fiecare matrice pentru a forma o imagine urmand sa fie excluse unele informatii din imagine , pastrandu-si dimensiunile intiale dar rezultand formarea imaginii de comprimare cu informatii reduse. Cu cât acest număr este mai mare, cu atât devine mai bună calitatea aproximării, dar, de asemenea, sunt necesare mai multe date pentru a o codifica.


A
≈
k
U
·
k
k
Σ
·
k
V T

- Datele din matricile U, Σ și V sunt sortate după cât contribuie la matricea A din produs. Acum luăm doar primele k linii respectiv coloane ale lui U și V transpus și pătratul din stânga sus (k × k) din Σ, conținând cele mai mari k valori singulare.
- Am ales ,,k'' , adică construim cele mai bune aproximări de rang ,,k'' ale matricilor de intensitate pentru fiecare culoare.

```
U_red_k= U_Red[:,0:k]
V_red_k= V_Red[0:k,:]
U_green_k= U_Green[:,0:k]
V_green_k= V_Green[0:k,:]
U_blue_k= U_Blue[:,0:k]
V_blue_k= V_Blue[0:k,:]
Sigma_Red_k = Sigma_Red[0:k]
Sigma_Green_k= Sigma_Green[0:k]
Sigma_Blue_k= Sigma_Blue[0:k]
```

- In continuare vom afisa dimensiune totala a matricelor comprimate pe care le stocam:

```
compressedBytes =sum([matrix.nbytes for matrix in [U_red_k, Sigma_Red_k, V_red_k, U_green_k,
                                                   Sigma_Green_k, V_green_k ,U_blue_k, Sigma_Blue_k, V_blue_k]])
print("Matricele comprimate pe care le stocăm acum au o dimensiune totală ",compressedBytes)
```

Ca bonus vom putea calcula dimensiunea totala a matricelor comprimate bucățile de aproximare rang ,,k’’ comparativ cu dimensiunea imaginii originale.De exemplu am obținut o rată de compresie ,,x’’ ceea ce înseamnă că doar x% din spațiul de stocare original este utilizat pentru acele matrici din care putem restabili o matrice care este apropiată de cea originala.

```
rata = compressedBytes/original_bytes
print("Raportul de compresie dintre dimensiunea originală a imaginii și dimensiunea totală a factorilor comprimați= ",rata
```

Vom construi imaginea aproximata pentru fiecare valoare in parte urmand sa le fuzionam, cu ajutorul teoremei: $A = U * \Sigma \ * V^t$

```
image_red_aproximation = np.matrix(U_Red[:, :k]) * np.diag(Sigma_Red[:k])* np.matrix(V_Red[:k, :])
image_green_aproximation = np.matrix(U_Green[:, :k]) * np.diag(Sigma_Green[:k]) * np.matrix(V_Green[:k, :])
image_blue_aproximation = np.matrix(U_Blue[:, :k]) * np.diag(Sigma_Blue[:k])* np.matrix(V_Blue[:k, :])
```

- Vom crea o noua matrice cu zerouri cu dimensiunile imaginei originale în care vom adauga: image_red_aproximation,

image_green_aproximation, image_blue_aproximation.

```
compressed_image = np.zeros((linie,coloana,3))

compressed_image[:, :, 0] = image_red_aproximation
compressed_image[:, :, 1] = image_green_aproximation
compressed_image[:, :, 2] = image_blue_aproximation
```

De asemenea vom converti valoarea intensitatii pixelilor ; daca valoarea este negativa la valoarea 0,respective daca valoarea este mai mare decat 255 o convertim la 255.

```
np.clip(compressed_image,0,255,out=compressed_image)
```

- In continuare vom afisa imaginea compresata si îi vom seta un titlu

```
plt.title("Image Name: " + "compressed_image_jpg" + "\n")
plt.imshow(compressed_image)
plt.axis('on')
plt.show()
compressed_image = Image.fromarray(compressed_image)
```

Am folosit si o functie pentru a afisa imaginea:

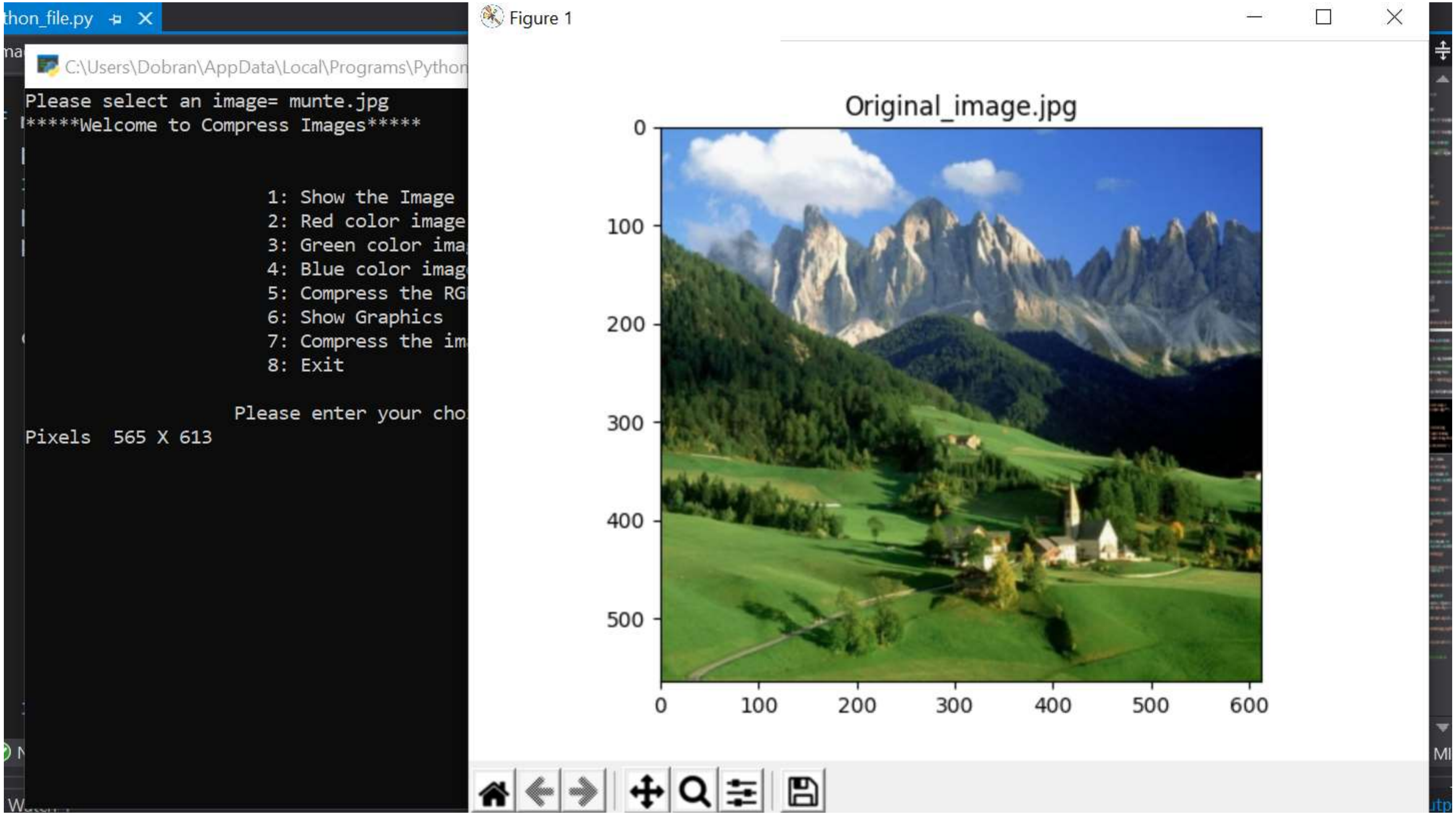
```
def show_images(image_name):

    plt.title("Original_image.jpg")
    plt.imshow(image_name)
    plt.axis('on')
    plt.show()
```

- Cu ajutorul librariei importate imageio putem deschide imaginea dorita pe care vrem sa o compresăm:

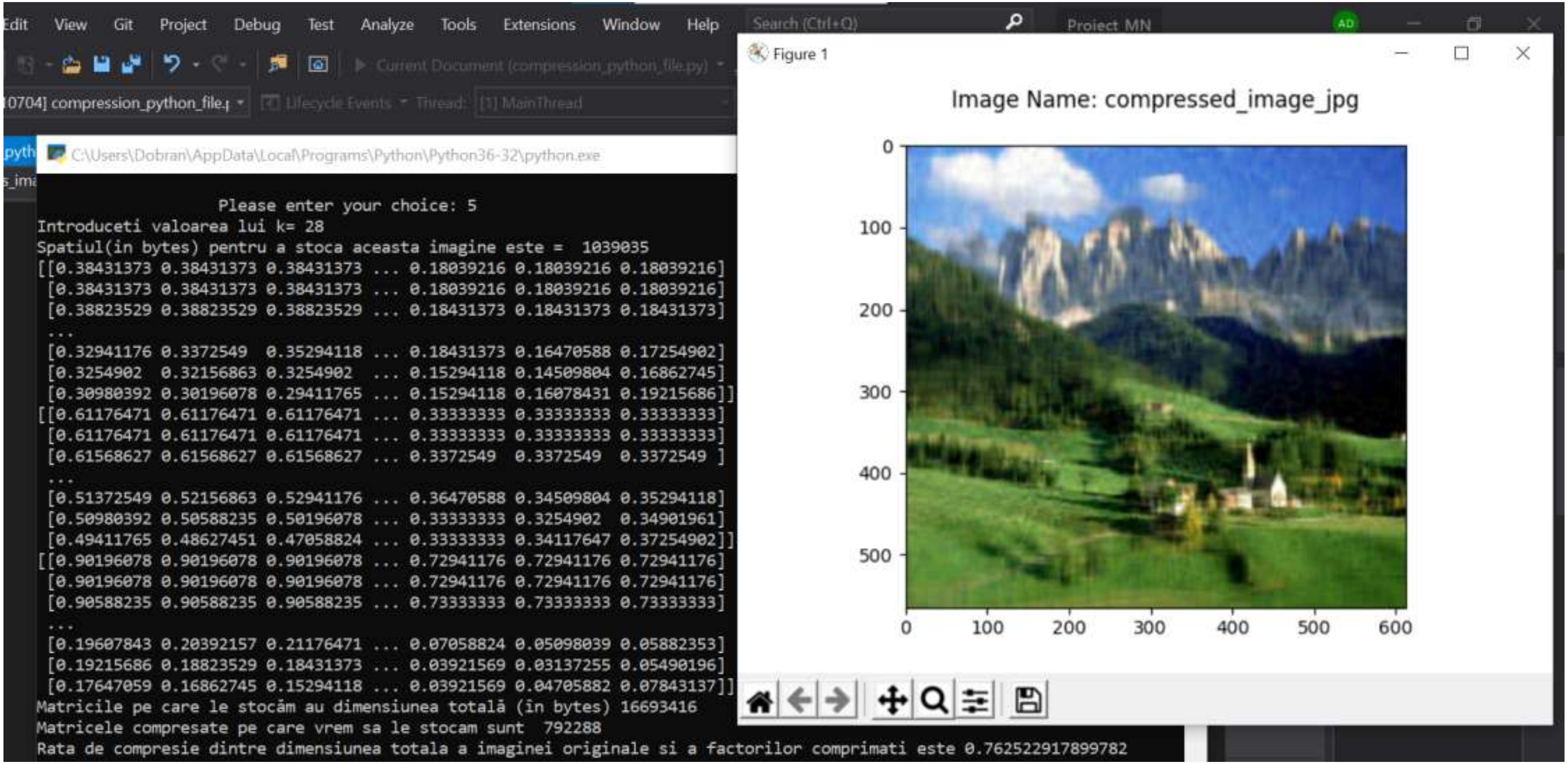
```
path = input("Please select an image= ")
image = imageio.imread(path)
print("*****Welcome to Compress Images*****")
print()

choice = input("""
                  1: Show the Image
                  2: Red color image
                  3: Green color image
                  4: Blue color image
                  5: Compress the RGB image
                  6: Show Graphics
                  7: Compress the image with 4 different values
                  8: Adjust Image Brightness
                  9: Exit

               Please enter your choice: """)
```

Vom selecta 1 pentru a vizualiza imaginea aleasa:

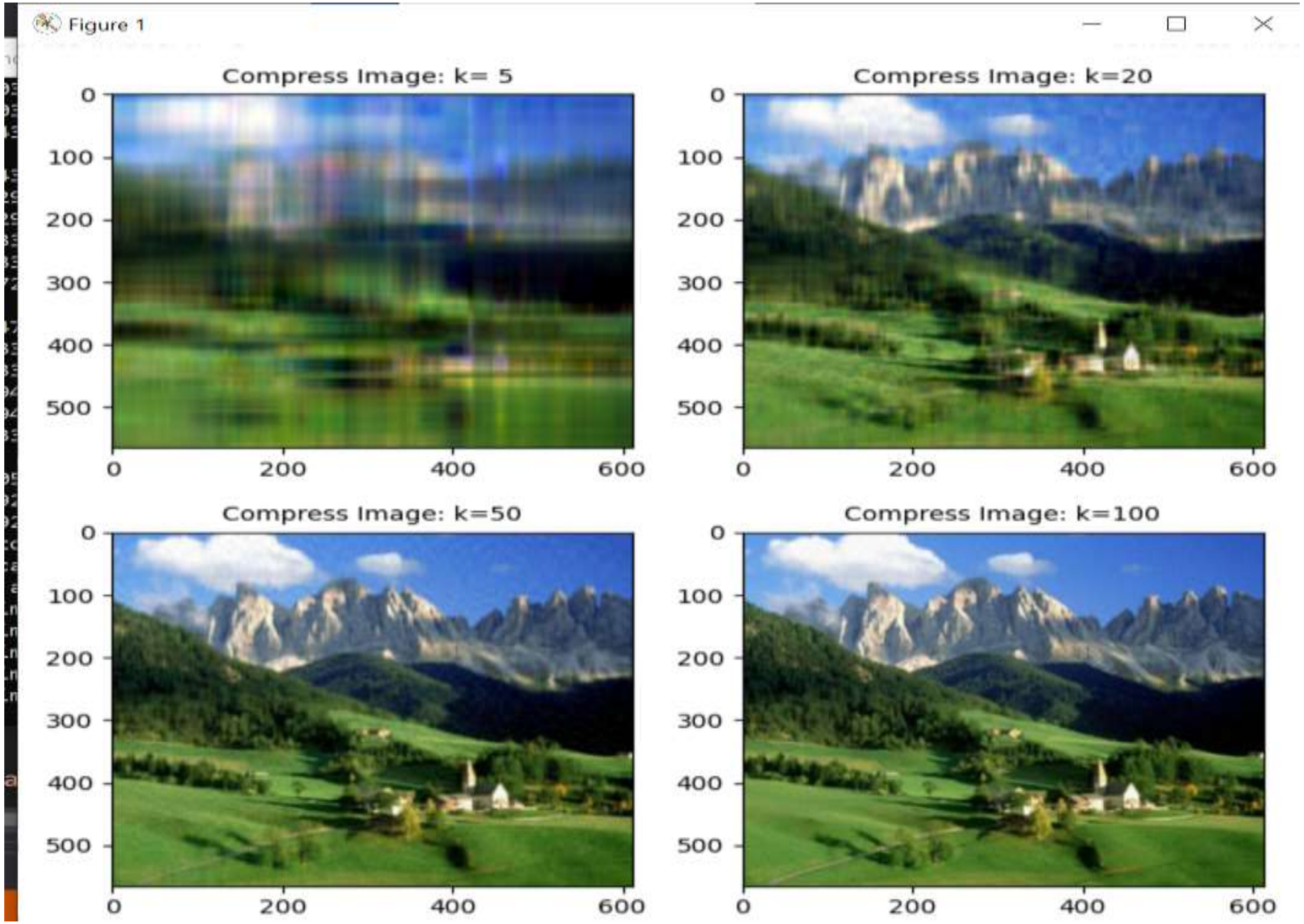
```
if choice == "1":
    image=image/255
    rand,coloana,_=image.shape
    print("Pixels ",rand,"X",coloana)
    show_images(image)
```



- In continuare vom selecta 5 pentru a compresa imaginea color cu o valoare ,,k’’ pe care o introducem de la tastatura:



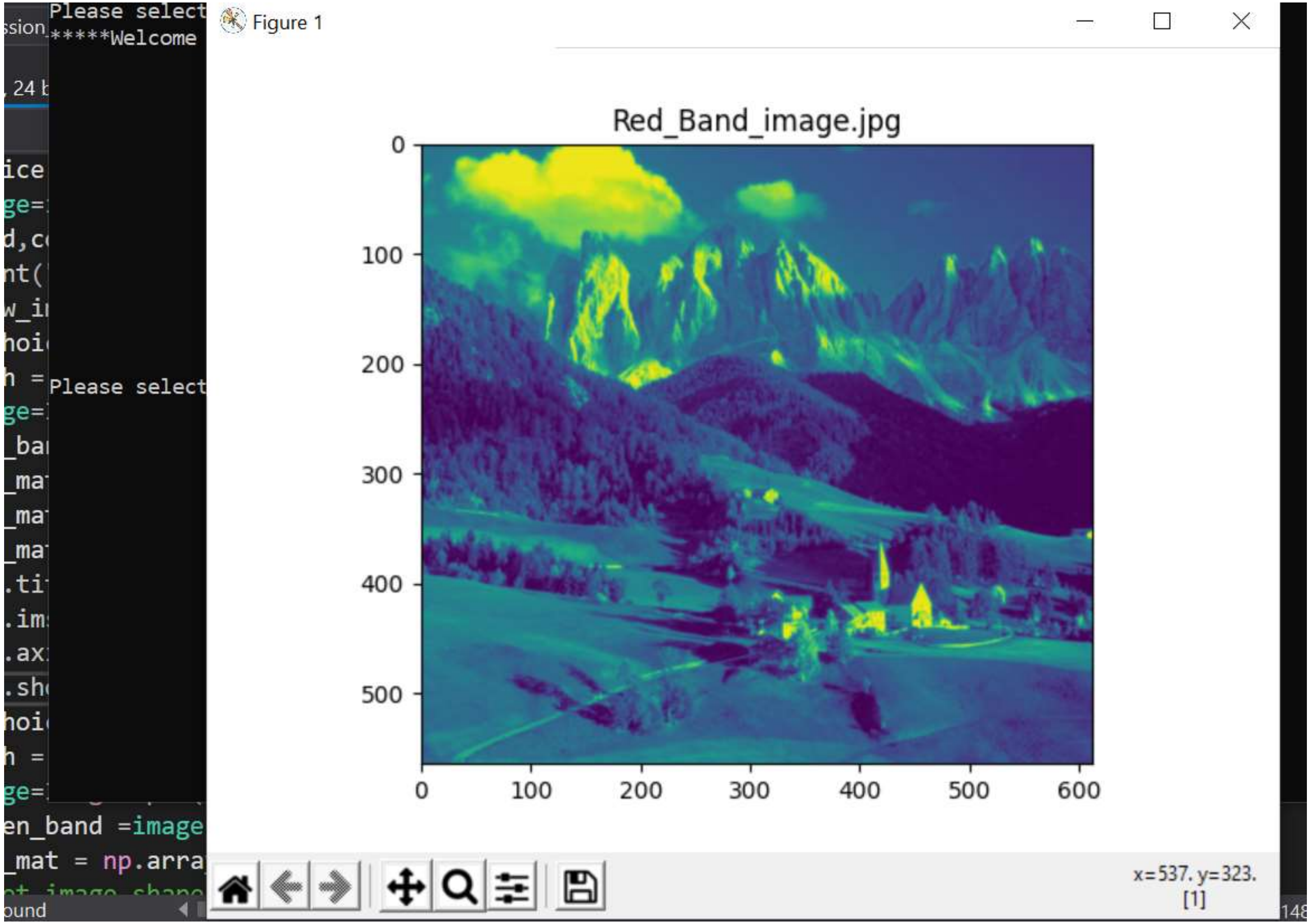
- Calitatea imaginii reconstituite s-ar îmbunătăți pe măsură ce vom folosi mai mulți vectori singulari de top. Iată o comparație a imaginii reconstituite utilizând un număr diferit de componente de top,selectand tasta 7:



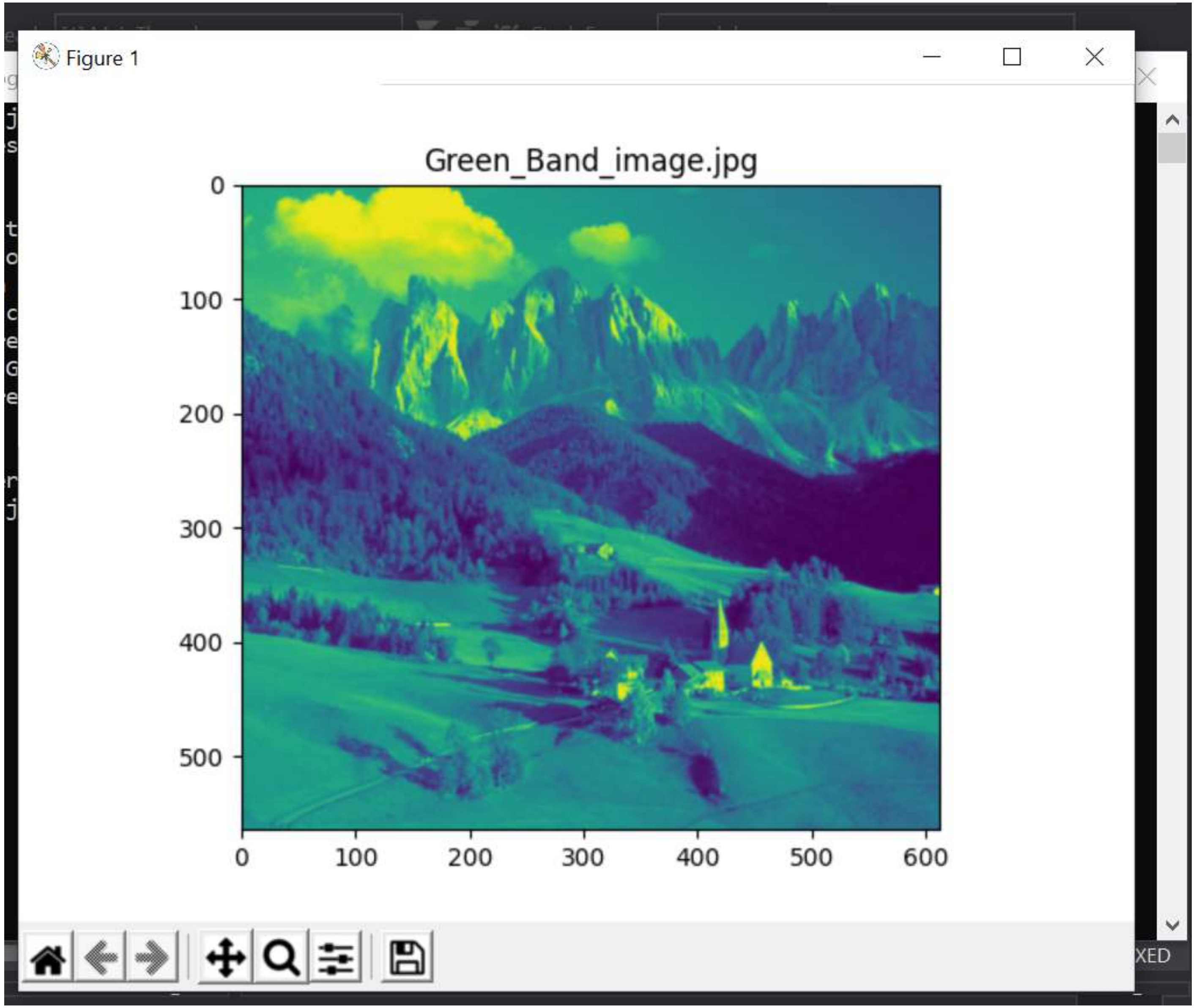
- In continuare ne vom ocupa pe rand doar de doar de una dintre matricile nu de toate trei. Intai extragem mai întâi imaginea corespunzătoare matricei de culoare roșie. Cu ajutorul functiei getdata() pe care o regasim in pachetul PIL d putem specifica banda = 0 pentru a obține o imagine de culoare roșie
- Vom converti matricea de culoare rosie ca o matrice care contine valorile fiecarui pixel.In continuare vom converti matricea unidimensionala respectiva matricea bidimensionala utilizand dimensiunile imaginei initiale:

```
elif choice == "2":
    path = input("Please select an image= ")
    image=Image.open(path)
    red_band =image.getdata(band=0)
    img_mat = np.array(list(red_band), float)
    img_mat.shape = (image.size[1], image.size[0])
    img_mat = np.matrix(img_mat)
    plt.title("Red_Band_image.jpg")
    plt.imshow(img_mat)
    plt.axis('on')
    plt.show()
```

Utilizand tasta 2 vom obtine:



• Analog pentru imaginea de culoare verde respective albastru:



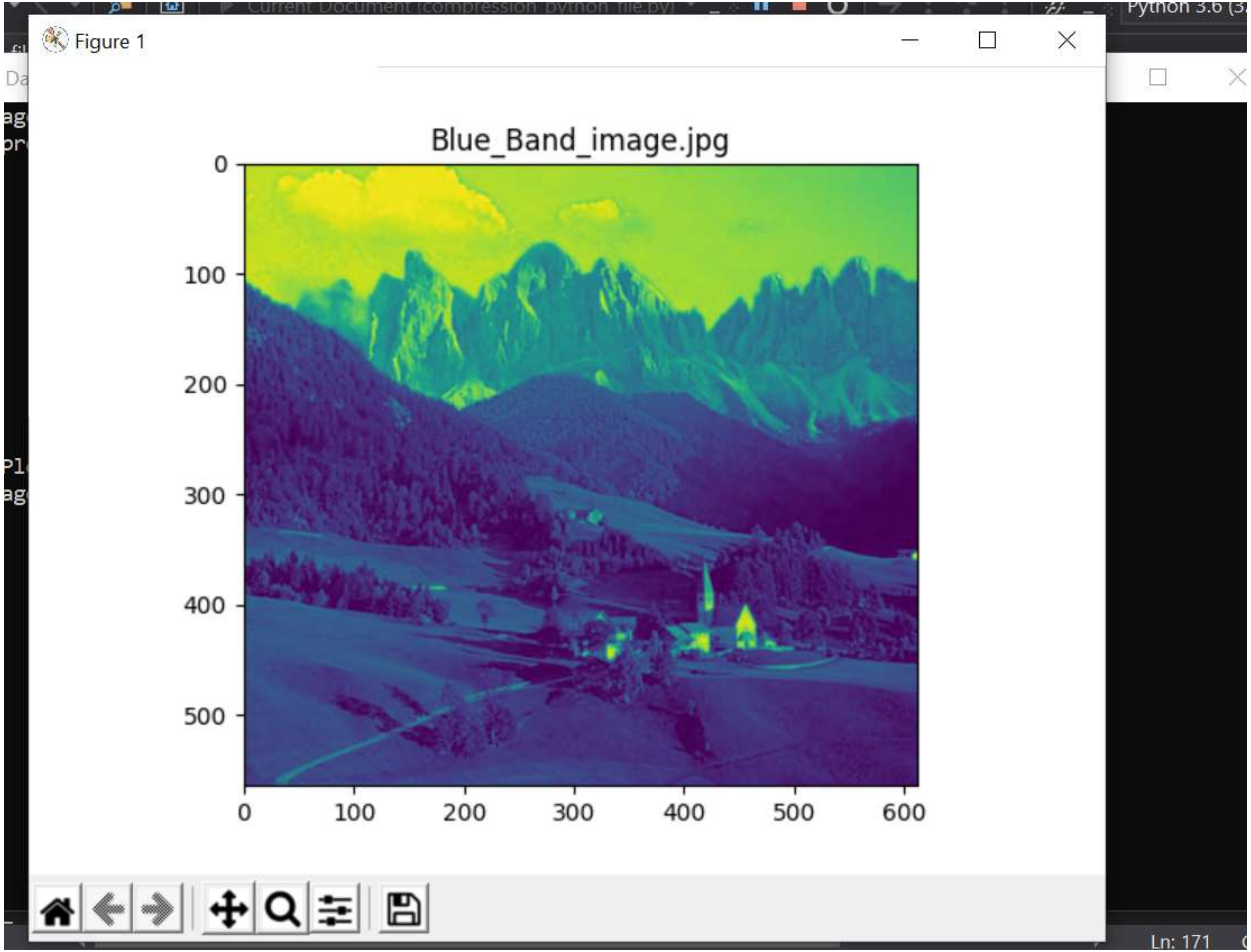
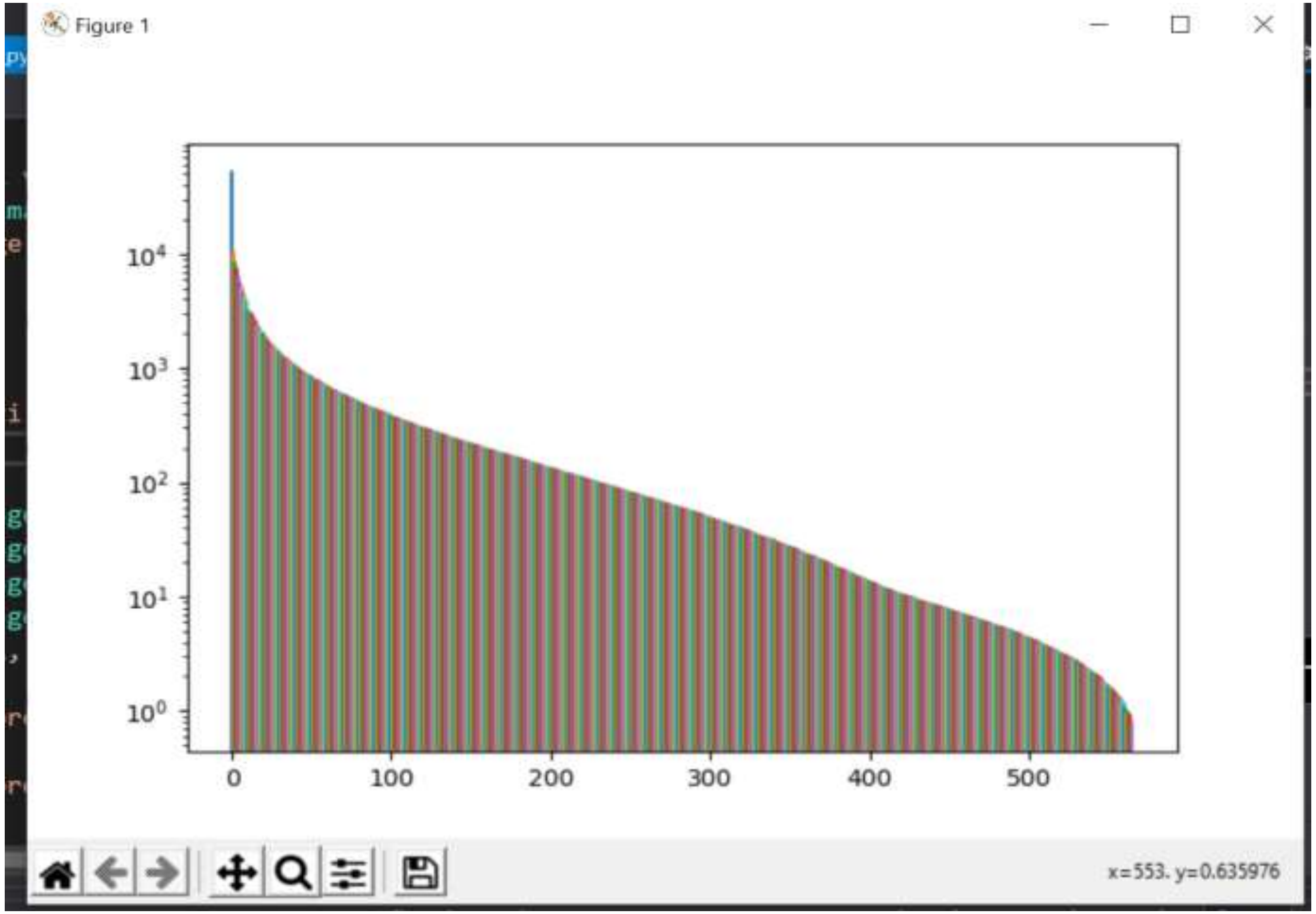


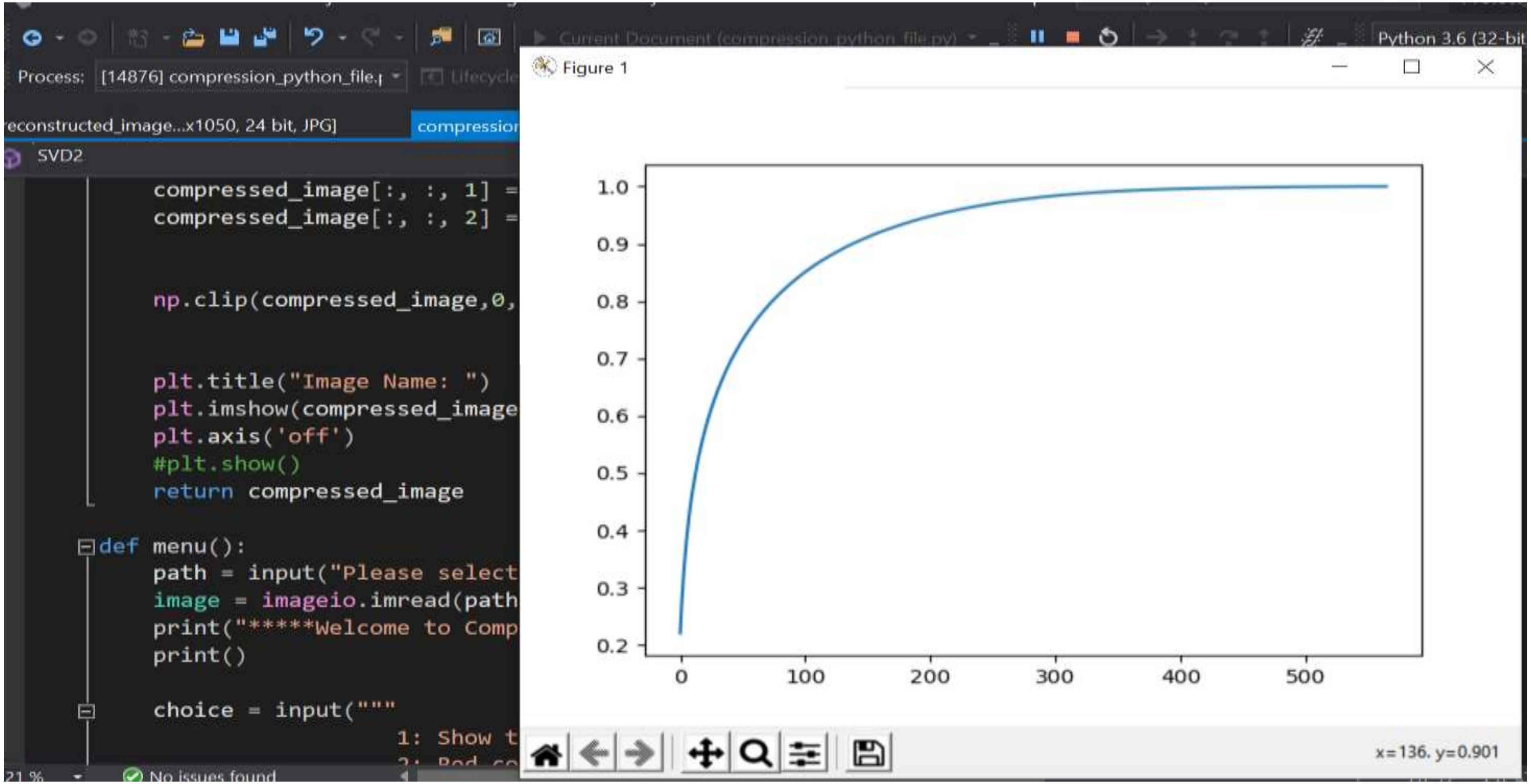
Figure 1
Blue_Band_image.jpg
0
100
200
300
400
500
0
100
200
300
400
500
600
Ln: 171

Grafice

- In continuare vom plota valorile singulare pe scala logaritmica utilizand plt.semilogy:

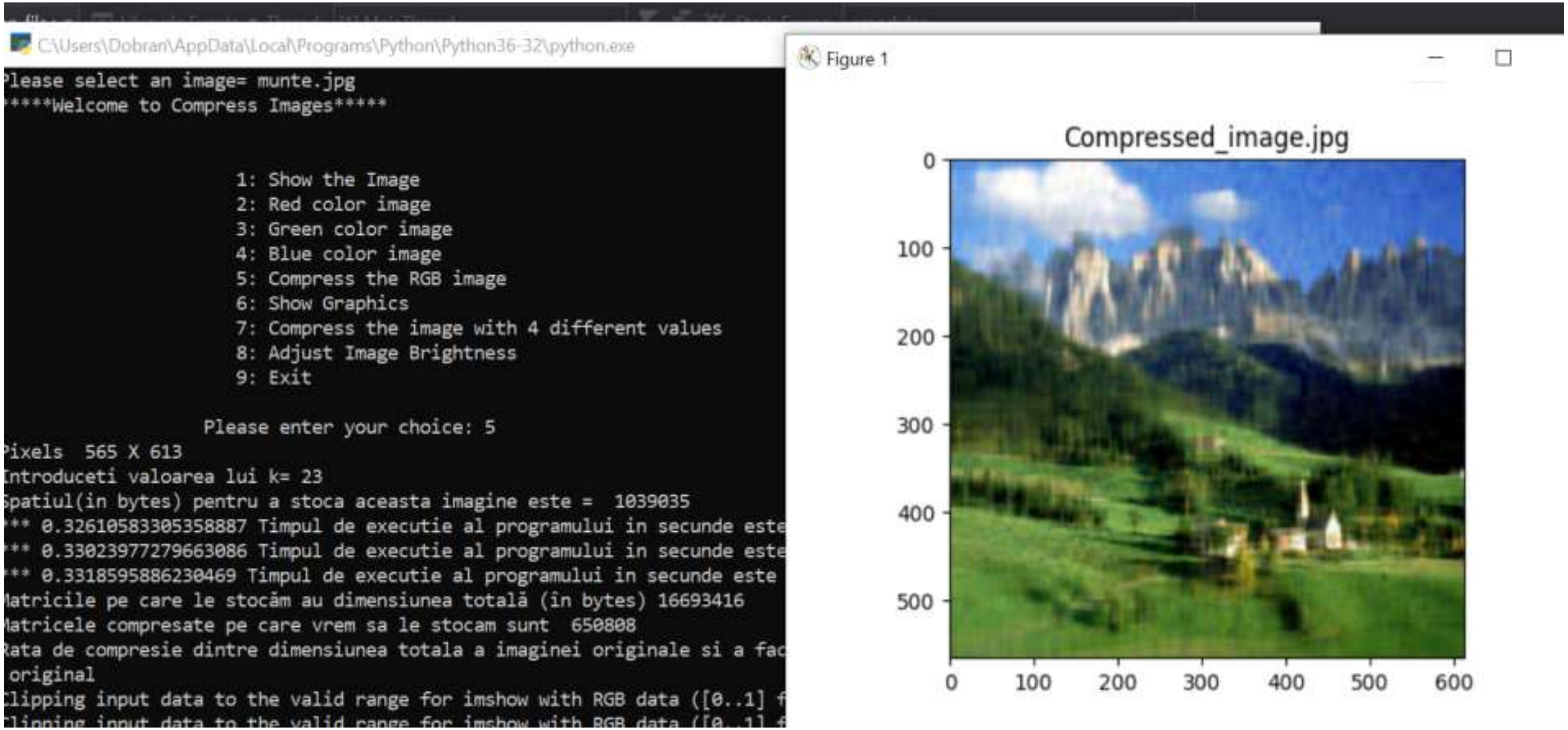


In continuare vom plota graficul procent informatie vs valori singulare pentru canalul rosu al imaginii. Dupa cum se poate observa peste 90% din informatie este cuprinsa in primele 136 de valori. Acest fapt este valabil si pentru celelalte 2 canale(albastru,verde) .

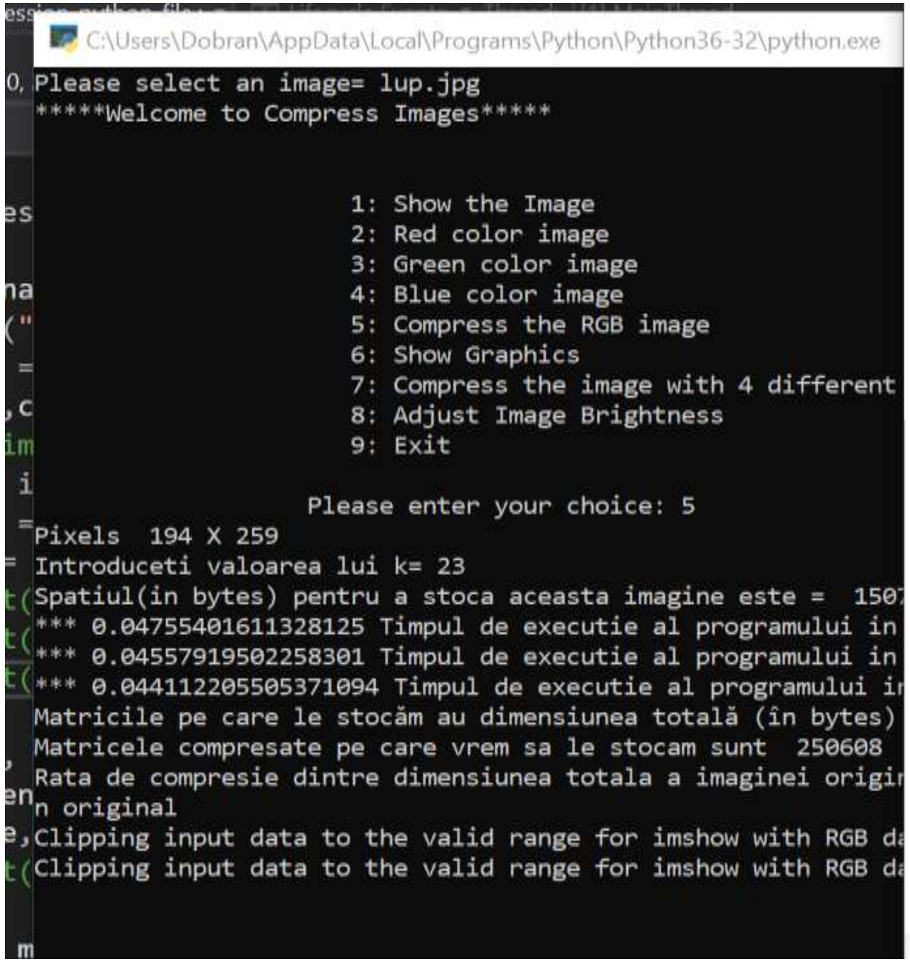


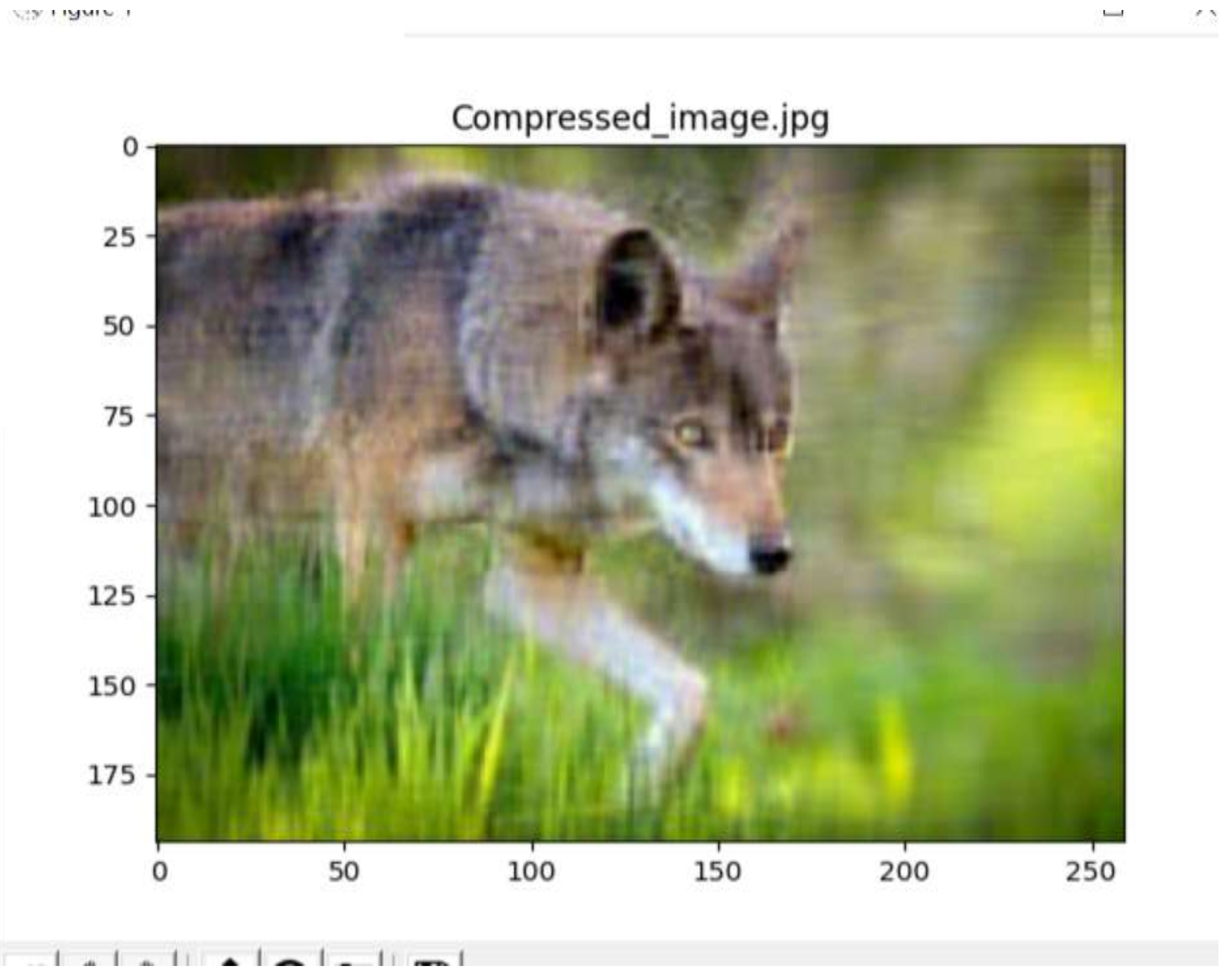
Algoritmul SVD necesita aproximativ 0.32 secunde pentru a fi executat,iar pentru toata imaginea ce continue cele 3 matrici, de dimensiune 565 X 613 pixeli, iar tot procesul va dura aproximativ 0.98 secunde

```
start_time = time.time()
U, Sigma, V = np.linalg.svd(B.copy())
print("*** %s Timpul de executie al programului in secunde este ***" % (time.time() - start_time))
```



Algoritmul SVD necesita aproximativ 0.0475 secunde pentru a fi executat,iar pentru toata imaginea ce continue cele 3 matrici, de dimensiune 194X259 pixeli, iar tot procesul va dura aproximativ 0.136 secunde





Concluzie

In concluzie, descompunerea valorii singulare este un instrument matematic foarte puternic, cu aplicații în multe domenii, cum ar fi compresia imaginilor pe care tocmai le-am abordat ,dar există multe altele.

Descompunerea SVD este, în general, o alegere bună atunci când trebuie să comprimăm seturi de date mari în așa fel încât structura interioară și relațiile de corespondență dintre punctele de date să fie păstrate într-un fel. De asemenea, algoritmul de calcul DVS reprezintă unica modalitate, numeric fiabilă, de determinare a rangului unei matrice.

Calitatea unei metode de comprimare este adesea masurata prin raportul semnal-zgomot El măsoară cantitatea de zgomot introdusă printr-o comprimare cu pierderi a imaginii dar judecata subiectivă a privitorului este considerată probabil cea mai importantă măsură.

Bibliografie

- Cursuri Metode Numerice
- Laboratoare MN
- https://numpy.org/doc/stable/reference/generated/numpy.matrix.html
- http://timbaumann.info/svd-image-compression-demo/ slide 6,10
- http://fourier.eng.hmc.edu/e161/lectures/svdcompression.html
- http://www.cs.ubbcluj.ro/~per/Scs_Per/PrelImg/Prel_Img%20C12.pdf
- https://ro.wikipedia.org/wiki/Pixel
- https://www.digi24.ro/regional/digi24-constanta/sejururi-mai-ieftine-la-munte-313009

VA MULTUMESC PENTRU ATENTIE