My sample book

The Jupyter Book Community

Apr 19, 2024

CONTENTS

Exploring Gradient Descent On Image Reconstruction with EHT data

Abstract: The Event Horizon Telescope (EHT) has revolutionized our understanding of black holes by using technology like Very Long Baseline Interfermetry (VLBI) and General Relativistic Magnetohydrodynamics (GRMHD) simulations to create high-resolution images. This Juypter Book delves into the process of reconstructing images while showcasing images from EHT data. We investigate some key components within this process such as interpolation, calculating loss, and different gradient calculation methods.

Interpolation schemes play a pivotal role in connecting the dots between sparse observational data and reconstructing an image. It does so by estimating unknown data that fall in between existing, know data points. By studying how the real domain relates to the fourier domain, we can understand what factors can go into recreating a high-resolution image. Loss functions are a important part of the optimization process. This is because they quantify the difference between a reconstructed image and the real image. In order to do so we must understand how the data's peculiarities affects the loss function and it's regularization.

Finally, this notebook focuses on studying two distinct methods for computing gradients: finite differences and "dirtying the image." By evaluating their efficiency and accuracy, we hope to provide insights into selecting suitable gradient calculation methods for image reconstruction tasks. Additionally we attempt to combine all of these topics and perform gradient descent on sample test images.

CHAPTER

ONE

INTRO AND SETUP USE DIFF

These are all the libraries that are needed

```
import pandas as pd
import numpy as np
import cmath
import math
from scipy.optimize import fmin, minimize
from astropy import units as u
from scipy.interpolate import RegularGridInterpolator
import matplotlib.pyplot as plt
import copy
%matplotlib inline
```

As the earth spins, we trace out a track in the Fourier domain in order to give us one snapshot of the black hole image. On earth we create a sinusoidal wave in order to amplify the data that we recieve from our snapshot. This create two different sine bands. One being hi and the other being low. We can use these two data sets to create two different images for the same day in practice. Theoretically they should create the same image

1.1 Paste images here

1.2 Data Details

The data set from EHT (The Event Horizon Telescope) and from the HOPS pipeline (software from MIT), comes in multiple sets. From one data collection, we are given a hi band and low band sets.

Suppose that the sky signal is some radio wave $Sky = \epsilon \sin(\omega t)$. The signal however is extremely small which make it hard to detect. In order to pick out this signal, we use a local oscillator which generates another wave (i.e. $LO = A \sin(\omega' t)$) where A is some amplitude. Once these two signals are multiplied together, we get something like so: $Sky \times LO = A\epsilon \sin(\omega t) \times \sin(\omega' t) = A\epsilon \sin[(\omega + \omega')t] + sin[(\omega - \omega')t]$.

Here we have $A\epsilon$ be an amplitude such that the signals are big enough for our instruments to detect. The $\omega + \omega'$ and $\omega - \omega'$ results in two bands which the data is taken from

1.3 Some Details about the Table

time is the time that it was taken. This time stamp is not used because we assume things are static.

T1 and T2 are the two telescopes.

U and V are the fourier location of the fourier domain. When we plot each point in the fourier domain, we should obtain something like below #HEREEE They are given in terms of lambda.

Iamp is the Amplitude of the Fourier Coefficient

IPhase is the Phase of the Fourier Coefficient in degrees

ISigma is the error or the noisiness of the data point

Here, we will start to preprocess the data. First we will create a class to hold the information is a more accessible manner

```
class data:
    u: float
    v: float
    phase: float
    amp: float
    sigma: float
    vis_data: complex
    def __init__(self, u, v, phase, amp, sigma):
        self.u = u
        self.v = v
        self.phase = phase
        self.amp = amp
        self.sigma = sigma
        self.vis_data = amp * np.exp(1j * math.radians(phase))

    def __repr__(self):
        return f"[u: {self.u}, v: {self.v}]"

    def __str__(self):
        return f"[u: {self.u}, v: {self.v}]"
```

The next few cells read the data from a csv file

```
def process_data(data_df):
    """
    Processes the data in the dataframe into a coords list and data objects
    Args:
        data_df is a pandas data frame of the data
    Returns:
        a list of coordinates in u,v space
        a list of data objects
    """

    coords = []
    data_list = []
    for i in range(len(data_df)):
        data_list.append(data(data_df.loc[i, 'U(lambda)'], data_df.loc[i, 'V(lambda)
↪'], data_df.loc[i, 'Iphase(d)'], data_df.loc[i, 'Iamp(Jy)'], data_df.loc[i,
↪'Isigma(Jy)']))
        coords.append([data_df.loc[i, 'U(lambda)'], data_df.loc[i, 'V(lambda)']])
    coords = np.array(coords)
    return coords, data_list
```

```
def read_data(filename):
    """
    reads the data from a file into a pandas dataframe
    Args:
        filename is a string that represents a csv file
    Returns:
        a pandas dataframe
    """
    df = pd.read_csv(filename)
    return df
```

```
df = read_data("./data/SR1_M87_2017_095_hi_hops_netcal_StokesI.csv")
coords, data_list = process_data(df)
df
```

```
      #time(UTC)  T1  T2     U(lambda)     V(lambda)  Iamp(Jy)  Iphase(d)  \
0       0.768056  AA  LM  1.081710e+09 -3.833722e+09  0.014292  -118.9454
1       0.768056  AA  PV -4.399933e+09 -4.509480e+09  0.136734     5.8638
2       0.768056  AA  AP  8.349088e+05 -1.722271e+06  1.119780    58.1095
3       0.768056  AP  LM  1.080840e+09 -3.832004e+09  0.018448  -137.6802
4       0.768056  AP  PV -4.400757e+09 -4.507747e+09  0.139619   -57.1724
...          ...  ..  ..           ...           ...       ...        ...
6453    8.165278  AZ  LM -1.078324e+09  1.029597e+09  0.315983    10.9377
6454    8.165278  AZ  JC  3.392180e+09  9.968579e+08  0.058864    46.0474
6455    8.165278  JC  LM -4.470504e+09  3.273711e+07  0.108582  -178.7050
6456    8.165278  JC  SM  1.745735e+04 -1.192282e+05  1.123722   -29.5589
6457    8.165278  LM  SM  4.470522e+09 -3.285633e+07  0.104931    96.6936

      Isigma(Jy)
0       0.005847
1       0.004968
2       0.005243
3       0.044576
4       0.032591
...          ...
6453    0.030449
6454    0.090288
6455    0.043965
6456    0.091870
6457    0.028783

[6458 rows x 8 columns]
```

```
%run ./utility.ipynb
```

CHAPTER

TWO

INTERPOLATION AND REGULARIZERS

In this notebook, we will talk about interpolating coordinate values from an image as well as calculating the regularizer for gradient descent.

2.1 Background on FOV and k-space

FOV stands for Field of view and it refers to the distance over which an image is acquired or displayed. These images are usually captured by some sort of signal processing like telescopes and MRIs. The FOV and pixel width determine the number of units in the fourier domain that must be obtained in order to reconstruct an image.

For the sakes of this research and example, FOVx = FOVy = FOV and $\Delta x = \Delta y = \Delta w$. Where w is the pixel width.

The Fourier projection of spatial frequencies all follow a similar pattern. Instead of regular sine waves, we see complex exponentials , cos n ω t + i sin n ω t. Each pair of samples lines differ by exactly 1 cycle over the FOV. This means that $\Delta k = k_{n+1} - k_n = \frac{n+1}{FOV} - \frac{n}{FOV} = \frac{1}{FOV}$

2.2 Interpolation

In this notebook, we have an image that is being linearly transformed using fourier transforms. We do this in order to allow us to use direct comparisons between our image (in real space but being transformed into fourier space) and the data obtained by the telescopes (which are in the spectral or fourier domain.)

When doing this transformation, we obtain a grid of points extracted from the image which do not necessarily line up with the U and V coordinates from the dataset. Instead, we must interpolate or estimate the values of the data points using existing known points in our grid.

Furthermore, in order to translate correctly from image space to fourier space, we need to multiply the grids (kx and ky) by k_FOV. The calculation for this is explained above.

It is important to note that the more grid points we have, the bigger the image. Thus the size of our Fourier Domain is the Number of pixels of the image by k_FOV. If our FOV is very small, then the width of the fourier grid becomes large.

Next we want to interpolate the complex value at each coordinate. RegularGridInterpolator however cannot interpolate complex numbers so we interpolate the real and imaginary parts separately and then combined them for the final result.

Finally, we use a linear method for interpolation due to it's local interpolation scheme. The other schemes that RegularGridInterpolator offers uses a C^2-smooth split which is a non-local scheme. Since we wanted to optimize the computation time, we chose the linear method.

```
def interpolate(image, coords, FOV):
    """
    Interpolates the values of each coordinate in coords in the fourier domain of␣
 ↪image
    Args:
        image is a 80x80 pixel image that represents our reconstructed image
        coords is a list of u,v coordinates that we obtained from our data
        FOV is the Field of view from the telescopes. For the EHT data, our FOV is␣
 ↪100 micro ascs.
    Returns:
        The interpolated values at the coordinates based on image
    """

    ft_image = np.fft.fftshift(np.fft.fft2(image))

    k_FOV = 1/FOV

    kx = np.fft.fftshift(np.fft.fftfreq(ft_image.shape[0], d = 1/(k_FOV*ft_image.
 ↪shape[0])))
    ky = np.fft.fftshift(np.fft.fftfreq(ft_image.shape[1], d = 1/(k_FOV*ft_image.
 ↪shape[1])))

    interp_real = RegularGridInterpolator((kx, ky), ft_image.real, bounds_error=False,
 ↪ method="linear")
    interp_imag = RegularGridInterpolator((kx, ky), ft_image.imag, bounds_error=False,
 ↪ method="linear")

    real = interp_real(coords)
    imag = interp_imag(coords)

    return real + imag * 1j
```

Here we do some tests using interp_real and interp_imag to verify that they do interpolate a value using a linear scheme. First we start with a very basic 5x5 grid.

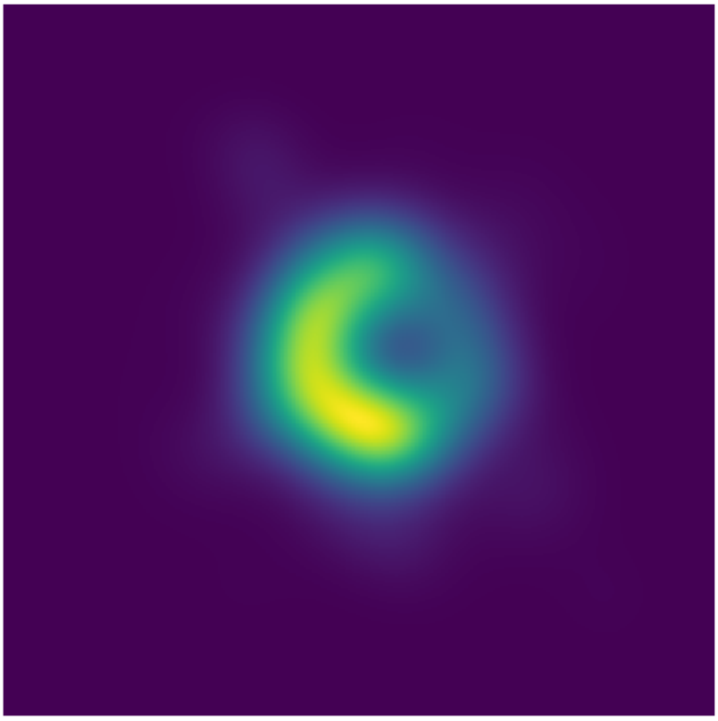
```
x = [0,1,2,3,4]
y = [0,1,2,3,4]
z = [[1,1,1,1,1],[2,2,2,2,2],[3,3,3,3,3],[4,4,4,4,4],[5,5,5,5,5]]
interp = RegularGridInterpolator((x,y),z)
print("Expected: 1, Actual:", interp([0,3]))
print("Expected: 2.5, Actual:", interp([1.5,2.5]))
```

```
Expected: 1, Actual: [1.]
Expected: 2.5, Actual: [2.5]
```

Now, here is a test on a fourier transformed image

```
sample = np.loadtxt("images/interpolate_test.csv", delimiter=",")
plt.figure()
plt.imshow(sample)
plt.axis('off')
```

```
(-0.5, 179.5, 179.5, -0.5)
```



```
ft_image = np.fft.fftshift(np.fft.fft2(sample))
interpolated_points = interpolate(sample,[[-9.0e+09,-9.0e+09], [-8.9e+09,-9.0e+09] ,[-
↪8.95e+09,-9.0e+09]],180*u.uas.to(u.rad))
print("Expected:",ft_image[0,0],"\nActual:", interpolated_points[0])
print("\nExpected:",ft_image[1,0],"\nActual:", interpolated_points[1])
print("\nExpected: ",(ft_image[1,0]+ft_image[0,0])/2,"\nActual:", interpolated_
↪points[2])
```

```
Expected: (3.0000000019953994-2.1431905139479568e-09j)
Actual: (-335133.52213257825+690198.3841637481j)

Expected: (38.20989061676637+0.6177808031820264j)
Actual: (-231640.3460875617+376131.9723567965j)

Expected:  (20.604945309380888+0.30889400519417796j)
Actual: (-283386.93411007+533165.1782602723j)
```

2.2.1 Cubic Splines

Below is an interpolation subroutine written by Misha Stepanov.

The code provides an insight into a local interpolation scheme using 2D cubic splines. It is local in a sense that only 12 grid points around the point at which we are interpolating is used to contribute the the estimation. The code the interpolation is done with data being wrapped into a 2D torus and the xy-coordinates are supplied assuming that the step of the grid in both the x and y directions are equal to 1.

We call a piec piecewise cubic function a cubic spline if it interpolates a set of data points while also guaranteeing the smoothness at the data points observed. Splining are often used instead of polynomial interpolations because it yields similar results and avoids the Runge's phenomenon for higher degree polynomials.

```
def cubf1(x, y):
  cf1 = (1. + x - x*x)*(1. + 2.*y)*(1. - y)
  cf2 = (1. + y - y*y)*(1. + 2.*x)*(1. - x)
  return 0.5*(cf1 + cf2)*(1. - x)*(1. - y)
def cubf2(x, y):
  return -0.5*x*(1. - x)*(1. - x)*(1. + 2.*y)*(1. - y)*(1. - y)
def cubic12(A, X):
  N = len(A);  F = np.zeros(len(X))
  for i in range(len(X)):
    m = np.mod(X[i], N);  x, y = m - np.floor(m)
    m1, m2 = np.floor(np.mod(m + np.array([2., 2.]), N)).astype(int)
    F[i]  = A[m1 - 2, m2 - 2]*cubf1(x, y)
    F[i] += A[m1 - 1, m2 - 2]*cubf1(1. - x, y)
    F[i] += A[m1 - 2, m2 - 1]*cubf1(x, 1. - y)
    F[i] += A[m1 - 1, m2 - 1]*cubf1(1. - x, 1. - y)
    F[i] += A[m1 - 3, m2 - 2]*cubf2(x, y)
    F[i] += A[m1 - 2, m2 - 3]*cubf2(y, x)
    F[i] += A[m1    , m2 - 2]*cubf2(1. - x, y)
    F[i] += A[m1 - 1, m2 - 3]*cubf2(y, 1. - x)
    F[i] += A[m1 - 3, m2 - 1]*cubf2(x, 1. - y)
    F[i] += A[m1 - 2, m2    ]*cubf2(1. - y, x)
    F[i] += A[m1    , m2 - 1]*cubf2(1. - x, 1. - y)
    F[i] += A[m1 - 1, m2    ]*cubf2(1. - y, 1. - x)
  return F

def func(x, y):
  return np.sin(2*np.pi*x) + np.cos(2*np.pi*(x + y))
```

```
A = np.zeros((10, 10))
for i in range(10):
  x = i / 10.
  for j in range(10):
    y = j / 10.
    A[i, j] = func(x, y)

X = np.zeros((10000, 2))
for i in range(10000):
  X[i, 0] = i / 1000.
  X[i, 1] = i / 2000.
F = cubic12(A, X)
for i in range(10000-10,10000):
  print(X[i, 0], X[i, 1], F[i], func(X[i, 0] / 10., X[i, 1] / 10.))
```

```
9.99 4.995 -1.0058598083263428 -1.0062387310745087
9.991 4.9955 -1.0052755185370548 -1.0056188621460636
9.992 4.996 -1.004690839850659 -1.0049981027528014
9.993 4.9965 -1.0041057771382442 -1.0043764531360648
9.994 4.997 -1.0035203352965503 -1.0037539135379836
9.995 4.9975 -1.0029345192480468 -1.0031304842014765
9.996 4.998 -1.002348333941004 -1.002506165370251
9.997 4.9985 -1.001761784349569 -1.0018809572888083
9.998 4.999 -1.001174875473833 -1.0012548602024367
9.999 4.9995 -1.0005876123399073 -1.0006278743572115
```

2.3 Regularization

Here we start to calculate regularizers. We do so in order to smooth image and add some bias into the model to prevent it from overfitting the training data. Regularizers allow machine learning models to generalize to new examples that it has not seen during training.

The regularizer below implements a regularization method called "total squared variation." This method favors smooth edges and is often used for astronomical image reconstruction.

The formula for a TSV regularizer is

$-\sum_l \sum_m [(I_{l+1,m} - I_{l,m})^2 + (I_{l,m+1} - I_{l,m})^2]$

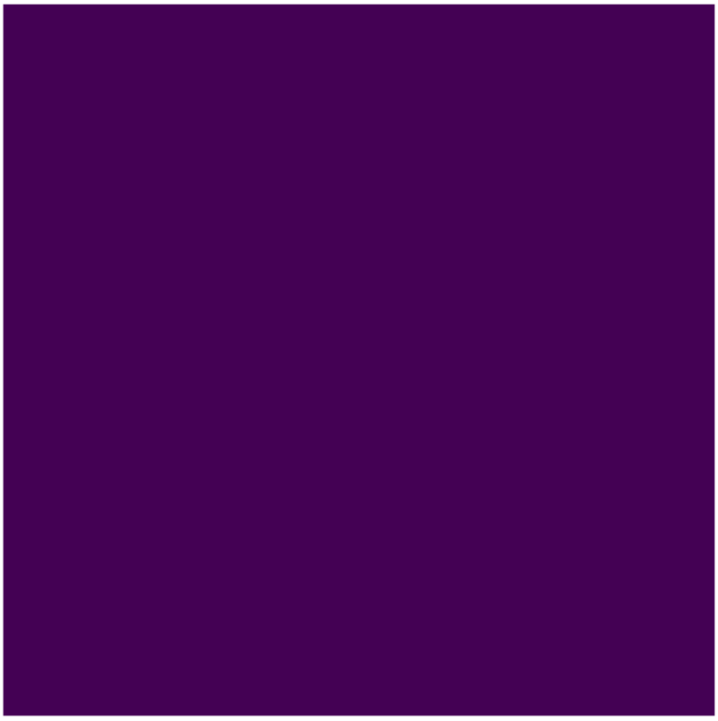
where l and m are pixel coordinates and I is the image. There are however other similar regularizers like Total variation that favors pixel-to-pixel smoothness.

```
def calc_regularizer(image: np.array, tsv=False, p=None):
    """
    Calculates the regularizer according to total squared variation
    Args:
        image is a 80x80 pixel image that represents our reconstructed image
        p is the kind of norm to be used
        tsv is the flag for total squared variation
    Returns:
        the regularizer
    """
    if tsv and p == None:
        raise Exception("p value not set")
    reg = 0
    if tsv:
        image_lshift = np.copy(image, subok=True)
        image_lshift = np.roll(image_lshift, -1,axis=1)
        image_lshift[:,-1] = image_lshift[:,-2]
        image_upshift = np.copy(image, subok=True)
        image_upshift = np.roll(image_upshift, -1, axis=0)
        image_upshift[-1] = image_upshift[-2]

        term_1 = np.power(np.absolute(np.subtract(image_lshift, image)),p)
        term_2 = np.power(np.absolute(np.subtract(image_upshift, image)),p)
        reg = np.sum(np.add(term_1,term_2))
    return -1 * reg
```

Here we test the calc_regulaizer function using an empty image, a monotone image, and a noisy image.

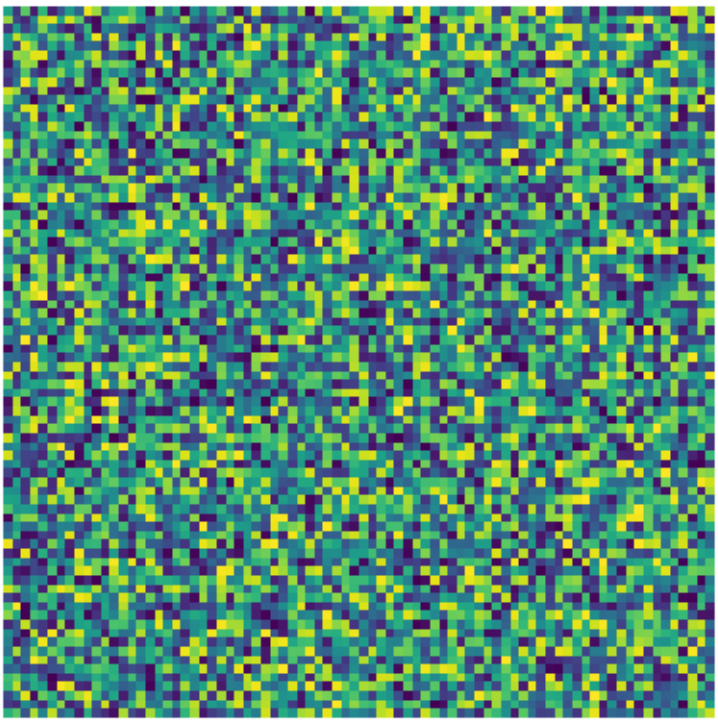
```
empty_image = np.zeros((80,80))
plt.figure()
plt.imshow(empty_image, vmin=0, vmax=1)
plt.axis('off')
```

```
(-0.5, 79.5, 79.5, -0.5)
```



```
noisy_image = np.random.rand(80,80)
plt.figure()
plt.imshow(noisy_image, vmin=0, vmax=1)
plt.axis('off')
```

```
(-0.5, 79.5, 79.5, -0.5)
```



```
monotone_image = np.full((80,80), 1)
plt.figure()
plt.imshow(monotone_image, vmin=0, vmax=1)
plt.axis('off')
```

```
(-0.5, 79.5, 79.5, -0.5)
```

```
print("Empty:",calc_regularizer(empty_image,True,2))
print("Noisy:",calc_regularizer(noisy_image,True,2))
print("Monotone:",calc_regularizer(monotone_image,True,2))
```

```
Empty: -0.0
Noisy: -2063.5561274280326
Monotone: 0
```

Here we see that the regularizer favors smooth images over noisy/rough ones. It is important to note that both the empty image and the monotone image have a regularizer of 0. This is because we weight the images towards smoother images.

This is the gradient of the regularizer. We calculate each point in the picture by doing $x_{i+1,j} + x_{i-1,j} + x_{i,j+1} + x_{i,j-1} - 4x_{i,j}$ for each i,j pixels

```
def gradient_regularizer(image: np.array):
    """
    Calculates the gradient of the regularizer
    Args:
        image is a 80x80 pixel image that represents our reconstructed image
    Returns:
        the gradient of the regularizer
    """
    image_lshift = np.copy(image, subok=True)
    image_lshift = np.roll(image_lshift, -1,axis=1)
    image_lshift[:,-1] = image_lshift[:,-2]
    image_upshift = np.copy(image, subok=True)
    image_upshift = np.roll(image_upshift, -1, axis=0)
    image_upshift[-1] = image_upshift[-2]
    image_rshift = np.copy(image, subok=True)
    image_rshift = np.roll(image_rshift, 1,axis=1)
```

(continues on next page)

(continued from previous page)

```
    image_rshift[:,0] = image_rshift[:,1]
    image_dshift = np.copy(image, subok=True)
    image_dshift = np.roll(image_dshift, 1, axis=0)
    image_dshift[0] = image_lshift[1]
    g_reg = 4 * image - image_lshift - image_upshift - image_rshift - image_dshift
    return g_reg
```

```
print("Empty:\n",gradient_regularizer(empty_image))
print("Noisy:\n",gradient_regularizer(noisy_image))
print("Monotone:\n",gradient_regularizer(monotone_image))
```

```
Empty:
 [[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
Noisy:
 [[ 0.51716531 -1.01700674  2.1734534  ...  0.6395222  -1.43350586
   2.34380415]
 [ 1.57802559 -0.80556637 -1.15188101 ...  0.93849329  1.2060943
  -2.47353726]
 [-1.48665407  0.72562006 -0.94508536 ...  1.57231842 -0.93047286
   2.19143682]
 ...
 [-0.61619555 -0.42364072  2.5512162  ...  0.67583614 -1.72396528
   0.16083916]
 [-0.46116163 -1.34493264  0.26653847 ...  0.74803333  1.45852493
  -0.58074494]
 [ 0.36681888  1.17858589 -1.31396134 ... -0.01283615 -0.16154421
   0.47977924]]
Monotone:
 [[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

```
%run ./utility.ipynb
```

CHAPTER

THREE

LOSS

The Loss function is trying to compare the data with the image and see if they agree with each other.

Let $I(x, y)$ be the image and we transform it into $\hat{I}(x, y)$ by using fourier transfroms described in the the interpolate section of the previous notebook.

Next we compare each interpolated point with its data term counterpart and get the equation for loss to be:

$$J = \sum(\frac{|\hat{I} - D|^2}{\sigma^2})$$

σ here is the error for each data point.

Finally, we add the regularizer to the loss using the method in the previous notebook.

3.1 Data Term calculation

We have above that $\hat{I}$ is in the Fourier Domain. In order to compute the data term, we calculate the term as $D = \text{Amp} * e^{i*\text{phase}}$.

It is important to note that the phase in Z is in radians, not degrees

```
def loss(image, data_list: list[data], coords, p = 2, reg_weight = 1, FOV = 100*u.uas.
↪to(u.rad)):
    """
    calculates the loss of an image compared to the data given
    Args:
        image is a 80x80 pixel image that represents our reconstructed image
        coords is a list of u,v coordinates that we obtained from our data
        p is the kind of norm to be used
        reg_weight is the regularizer weight
        FOV is the Field of view from the telescopes. For the EHT data, our FOV is␣
↪100 micro ascs.
    Returns:
        a loss value
    """
    error_sum = 0
    vis_images = interpolate(image, coords, FOV)

    for i in range(len(data_list)):
        vis_data = data_list[i].amp * np.exp(1j * math.radians(data_list[i].phase))
        vis_image = vis_images[i]
        error = (abs(vis_image-vis_data) / data_list[i].sigma) ** 2
```

(continues on next page)

(continued from previous page)

```
        error_sum += error

    return error_sum + reg_weight * calc_regularizer(image=image, tsv=True, p=2)
```

3.2 Furthermore into the data

What we are doing here is a simplified version of the loss function. In reality, the loss is more complicated due to other factors.

Because of the method of combining signal data from many telescopes, we get errors that may throw off the data terms.

How would we change our loss function?

3.2.1 Method One: Gain

We add a new function called the Gain which represents the telescoping errors. It is a smooth function in time that has a phase and amplitude. We multiply it with the data term to result in:

$$J = \sum(\frac{|\hat{I} - GD|^2}{\sigma^2})$$

Now when we try to minimize the image, we are also trying to minimize G.

3.2.2 Method Two: Cosher Phase

The other way is to take into the account the Cosher Phase.

Using three telescopes we get:

$$\sum(\frac{|\phi_{1,2} + \phi_{2,3} + \phi_{3,1} - \phi_{\hat{I}}|^2}{\sigma^2})$$

$\phi_{1,2} + \phi_{2,3} + \phi_{3,1}$ is called the Cosher Phase CP which is independent of the telescope error and is taken from the data. $\phi_{\hat{I}}$ is a Cosher Phase computed from the image.

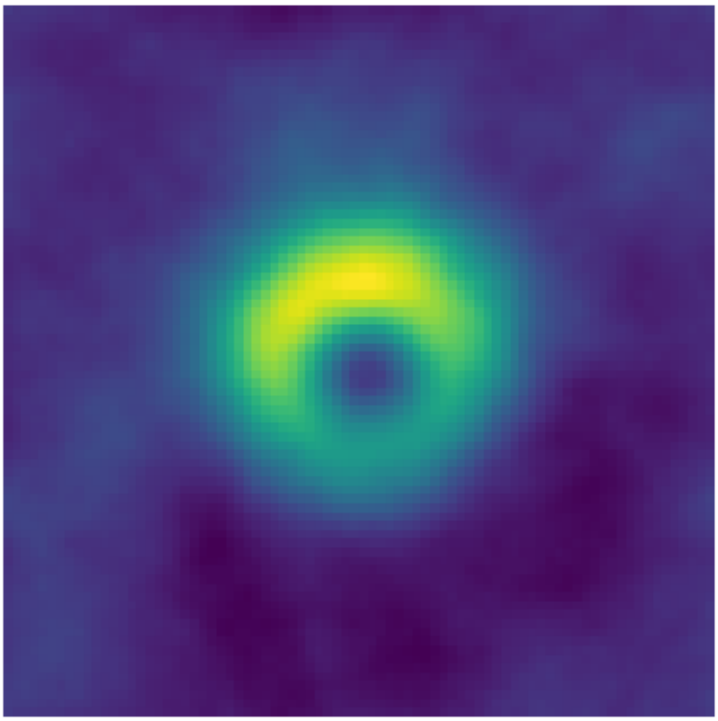
In practice, these are the two major methods in order to fit the image to the data.

3.3 Testing the function

Below we have a image that was generated for testing purposes. Sometimes we want to shift the image in someway so that the loss function can be optimized further. In EHT, they have tested with many variations and come up with the loss being the best with the ring centered in the middle. We will test this claim below.

```
sample = np.loadtxt("images/data.csv", delimiter=",")
plt.figure()
plt.imshow(sample)
plt.axis('off')
```

```
(-0.5, 79.5, 79.5, -0.5)
```



Here we generate a coords and data_list list by sampling data from the ring (not the noise around the ring).

```
def do_sample(n):
    """
    Collects n samples from a sample image
    Args:
        n is the number of samples as an integer
    Returns:
        a list of coordinates in u,v space
        a list of data objects
    """
    coords = []
    data_list = []
    ft_image = np.fft.fftshift(np.fft.fft2(sample))
    for i in range(n):
        coords.append((int(np.random.rand()*10-5), int(np.random.rand()*10-5)))
        data_list.append(data(coords[i][0], coords[i][1], 0, ft_
↪image[coords[i][0]+40][coords[i][1]+40], 1))
    return coords, data_list
```

```
coords, data_list = do_sample(25)
data_list
```

```
[[u: 4, v: 0],
 [u: 0, v: -1],
 [u: 3, v: 3],
 [u: 1, v: 1],
```

(continues on next page)

(continued from previous page)

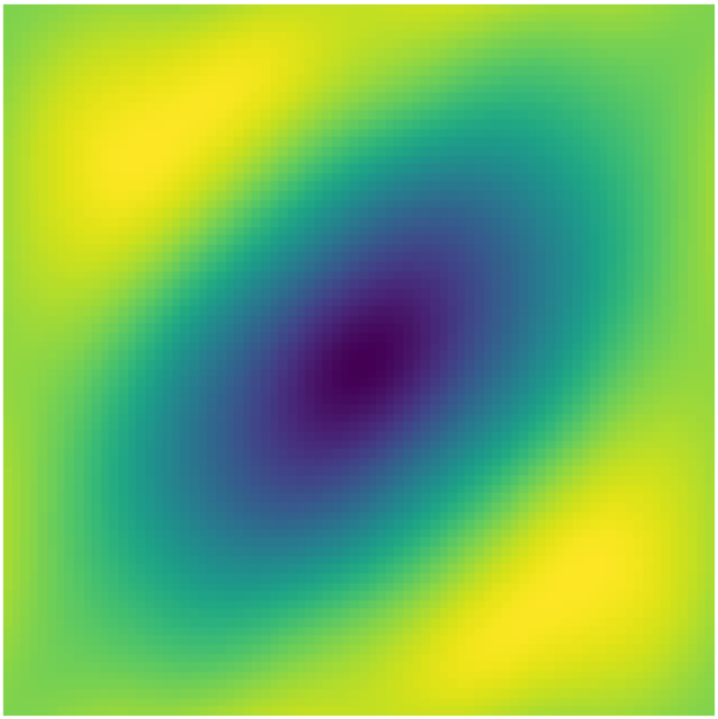
```
 [u: 1, v: 1],
 [u: 0, v: 2],
 [u: 0, v: -1],
 [u: -4, v: 3],
 [u: -2, v: 1],
 [u: 2, v: 0],
 [u: -3, v: -2],
 [u: -2, v: -1],
 [u: -1, v: 4],
 [u: -4, v: -2],
 [u: -4, v: -2],
 [u: 2, v: -4],
 [u: 1, v: 0],
 [u: -4, v: -2],
 [u: 0, v: -4],
 [u: -4, v: 4],
 [u: -2, v: -1],
 [u: 1, v: 3],
 [u: 3, v: 1],
 [u: 2, v: -4],
 [u: 2, v: 4]]
```

It is important to note here that the data points in data_list is complex. Since we sampled from the fourier transform, we have no need to calcuate the data term like we do in the loss function above. Below is a altered version of the loss function above.

Finally, below we calcuate the loss of the sample image we started with and then shift the image in all directions. Then we plot the array of losses that we obtained.

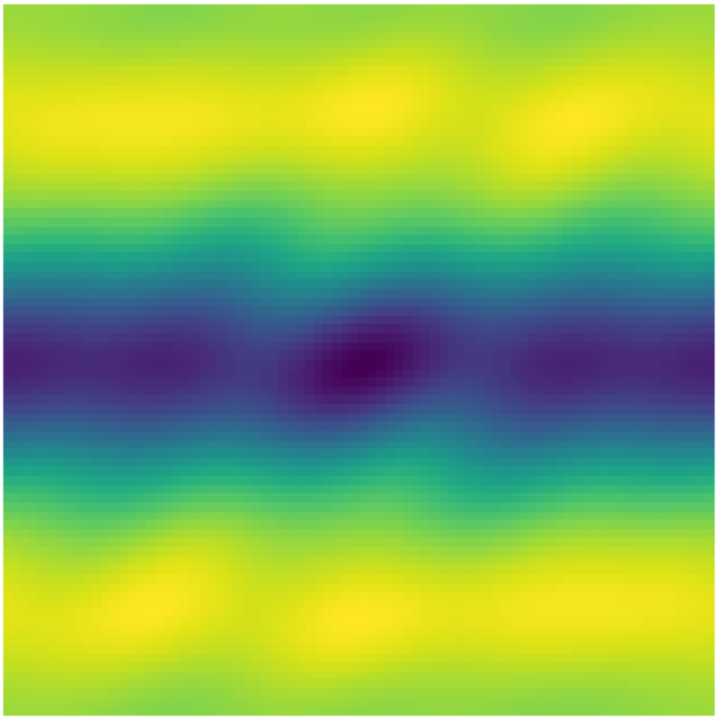
```
def calculate_losses():
    """
    Calculates the losses of the image by shifting it around
    Args:
        None
    Returns:
        an array of losses where the index is how much the image is shifted starting␣
 ↪with -40 to 40
    """
    loss_arr = np.zeros((len(sample),len(sample[0])))
    for i in range(len(sample)):
        image_1 = np.roll(sample, i, axis=1) # Right shifts
        for j in range(len(sample[i])):
            image_2 = np.roll(image_1, j, axis = 0) # Up shifts
            loss_arr[i][j] = loss(image_2, data_list, coords, reg_weight=0, FOV=1)

    loss_arr1 = np.roll(loss_arr, 40, axis=1)
    loss_arr2 = np.roll(loss_arr1, 40, axis=0)

    plt.figure()
    plt.imshow(loss_arr2)
    plt.axis('off')
    return loss_arr
```

```
loss_arr = calculate_losses()
```

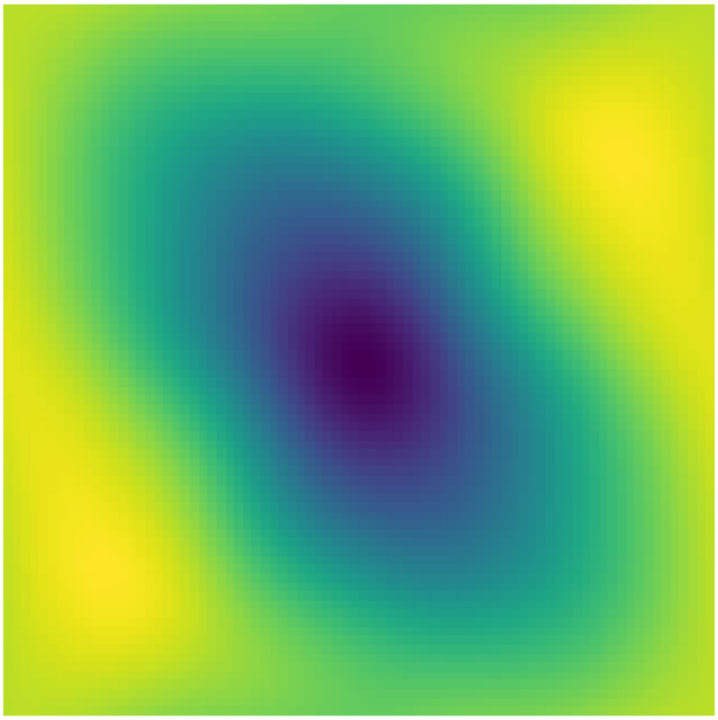


The dark spots in this picture are low points while the yellow spots are high points. Here we see that there are a few spots where the image produces low loss. If ran again, we will get differing behaviors

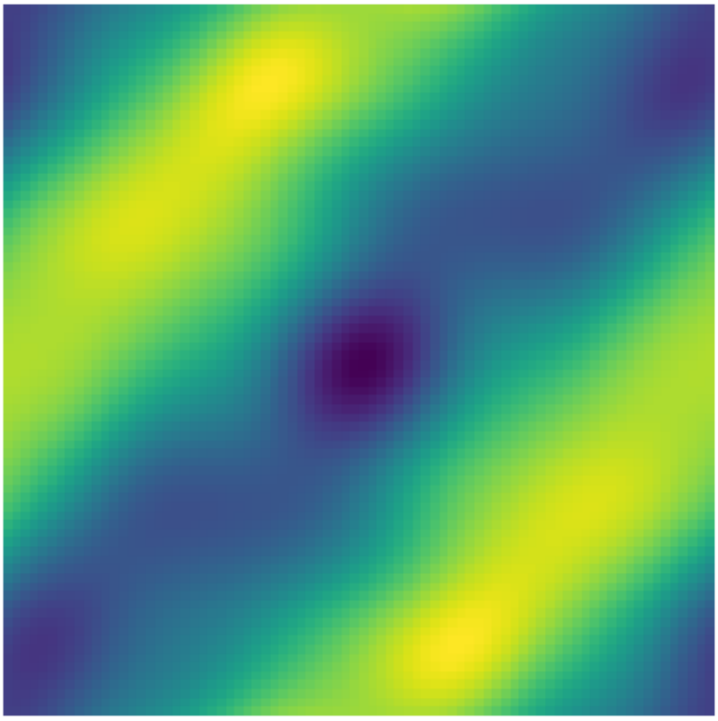
```
coords, data_list = do_sample(25)
loss_arr = calculate_losses()
```



```
coords, data_list = do_sample(25)
loss_arr = calculate_losses()
```




```
coords, data_list = do_sample(25)
loss_arr = calculate_losses()
```



In all four of these trials (and other reruns that we test), we generally see dark spots in the centers suggesting that the best area to put the ring is centered in the middle of the image

```
%run ./utility.ipynb
```

CHAPTER

FOUR

GRADIENT CALCULATIONS: FINITE DIFFERENCES

We have two different methods of computing gradients and in this notebook we will explore the first one.

4.1 Gradient's Background

A gradient is the direction of greatest change when looking at a scalar function. It is usually described as a generalized derivative or the rate of change. In our model, the gradient will be positive or negative depending on if a pixel's value must in increased or decreased. Additionally, each pixel will have its own gradient.

Here's an intuitive way of looking at gradients: Image you are standing in Tucson at Alvernon and Grant and we are interested in the function f that tells you your elevation. The gradient of f will always tell you which direction you should travel in in order to rise in elevation the quickest.

Since gradients are consistent (meaning if initial conditions are constant and the function doesn't change then we always get the same result), we can use gradients to find local maximums and minimums by simply following the gradient until either the gradient is 0 or we never finish (in the event of infinity end behaviors).

Why are gradient's important? In the real world, gradients are used in many fields like physics, robotics, and optimizations. We use it all the time in order to quanify the net rate of change in multi-variable functions!

4.2 Method one: Finite Differences Methods

The Finite Differences Method are a numerical analysis technique for solving differential equations by approximating derivatves using finite differences. Using these finite differences we can approximate a gradient of a function which we will denote as ∇f.

So what is Finite Differences exactly? It is a mathematical expression of the form $f(x+b) - f(x+a)$.

There are three basic types that are commonly considered for this method: Forward, Backward, and Central.

Forward differences is calcuated by $\nabla f = \frac{f(x+h)-f(x)}{h}$

Backward differences is calcuated by $\nabla f = \frac{f(x)-f(x-h)}{h}$

Central differences is calcuated by $\nabla f = \frac{f(x+\frac{h}{2})-f(x-\frac{h}{2})}{h}$

4.2.1 The Function Implemented

Below we have all three methods implemented controlled by a mode flag.

For Gradient Descent, we consider f to be our loss function and h to be a minute difference from a pixel's value. We iterate over every pixel and calculate the loss needed for equation used ($f(x+h)$, $f(x-h)$, or $f(x+\frac{h}{2}) - f(x-\frac{h}{2})$). This gives us the gradient for each pixel.

```
def gradient_finite_differences(data_list: list[data], coords, image, mode = 1, FOV =␣
↪100*u.uas.to(u.rad)):
    """
    Calculates a gradient based on finite differences
    Args:
        data_list is a list of data objects
        coords is a list of u,v coordinates that we obtained from our data
        image is a 80x80 pixel image that represents our reconstructed image
        mode is the type of difference used: 0 For central, -1 for backward, 1 for␣
↪forward
        FOV is the Field of view from the telescopes. For the EHT data, our FOV is␣
↪100 micro ascs.
    Returns:
        the gradient of the loss function
    """
    image_copy = np.copy(image, subok=True)
    upper_diff: float
    lower_diff: float
    h: float
    gradient_arr = np.empty(np.shape(image),dtype=np.complex_)
    if (mode == 0): # Central difference
        for row in range(len(image)):
            for col in range(len(image[row])):
                image_copy[row,col] += 1e-6 / 2
                upper_diff = loss(image_copy, data_list, coords, FOV=FOV)
                image_copy[row,col] -= 1e-6
                lower_diff = loss(image_copy, data_list, coords, FOV=FOV)
                image_copy[row,col] = image[row,col] # Reset that pixel to original␣
↪value
                gradient_arr[row,col] = (upper_diff - lower_diff) / 1e-6
    elif (mode == -1): # Backward difference
        upper_diff = loss(image, data_list, coords, FOV=FOV)
        for row in range(len(image)):
            for col in range(len(image[row])):
                image_copy[row,col] -= 1e-8
                lower_diff = loss(image_copy, data_list, coords, FOV=FOV)
                gradient_arr[row,col] = (upper_diff - lower_diff) / 1e-8
                image_copy[row,col] = image[row,col]
    elif (mode == 1) : # Forward difference is default
        lower_diff = loss(image, data_list, coords, FOV=FOV)
        for row in range(len(image)):
            for col in range(len(image[row])):
                image_copy[row,col] += 1e-8
                upper_diff = loss(image_copy, data_list, coords, FOV=FOV)
                gradient_arr[row,col] = (upper_diff - lower_diff) / 1e-8
                image_copy[row,col] = image[row,col]
    else:
        raise ValueError('Incorrect mode for finite differences')
    return gradient_arr.real
```

Final Note: Why do we use 1e-6 and 1e-8.

When we want to estimate our loss function we consider the equation below

$$\kappa(f_h') = max_{\delta x} \frac{|\frac{f(x+\delta x+h)-f(x+\delta x)}{h} - f'(x)|}{|f'(x)|} |\frac{x}{\delta x}|$$

$$\sim \frac{|f(x+h) - f(x) - f'(x)h|}{\epsilon_{machine} h |f'(x)|}$$

$$\sim \frac{\epsilon_{machine} f + (\frac{fh^2}{L^2})}{\frac{\epsilon_{machine} hf}{L}}$$

$$= \frac{L}{h} + \frac{h}{\epsilon_{machine} L}$$

Here, L is a characteristic scale of x and $\epsilon_{machine}$ is the error of the machine. By minimizing $\kappa(f_h')$ with respect to h, we find that $h \sim \sqrt{\epsilon_{machine}} L$. With $L \sim 1$, $h \sim \sqrt{\epsilon_{machine}}$.

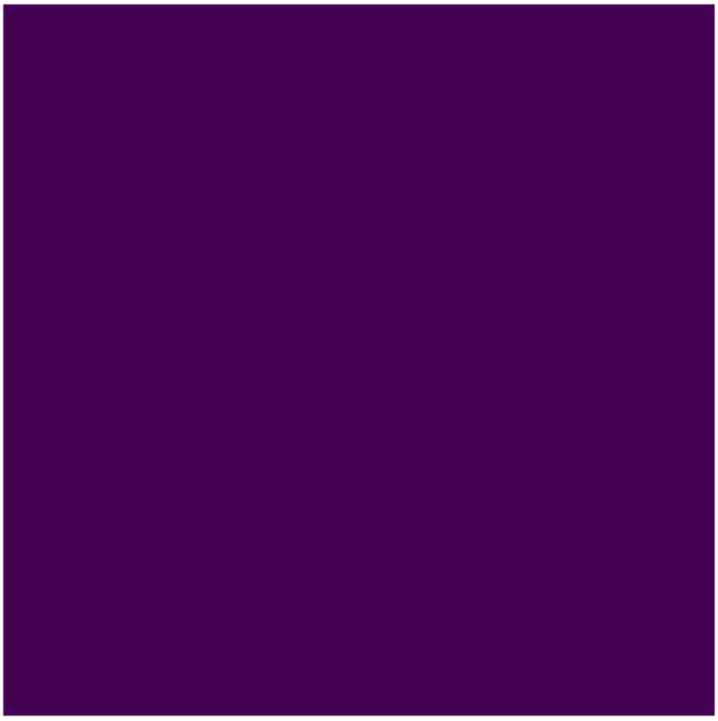
This is because of the maximum precision for floats in python. Floats in Python use IEEE Double Precision format which gives allows for it to compare up to 10^{-16} precision between two values. In forward and backward differences, the h value uses the square root of 10^{-16} which is 10^{-8}. In central differences, the h value uses a cubic root which means it uses roughly 10^{-6}

4.2.2 Demo Walkthrough

Start with an empty or blank image

```
emp = np.zeros((80,80))
plt.figure()
plt.imshow(emp)
plt.axis('off')
```

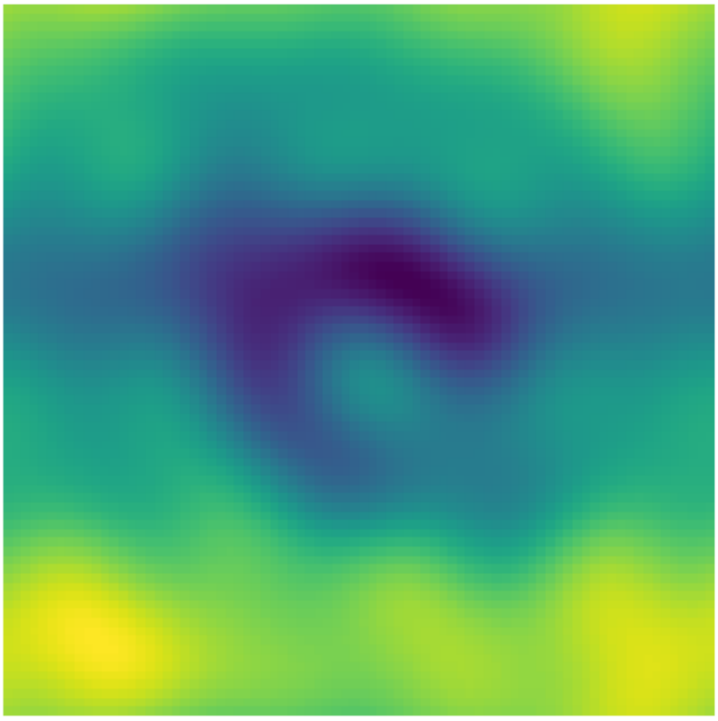
```
(-0.5, 79.5, 79.5, -0.5)
```



Next we get the sample data points from our synthetic data. Afterwards we will run our gradient code.

```
sample = np.loadtxt("images/data.csv", delimiter=",")
coords, data_list = do_sample(50)
x = gradient_finite_differences(data_list, coords, emp, FOV = 1)
plt.figure()
plt.imshow(x)
plt.axis('off')
```

```
(-0.5, 79.5, 79.5, -0.5)
```



Above is the image representation of the gradient, darker areas represent smaller gradients and lighter areas represent bigger gradients.

It is specifically a gradient when we start our "image" as an empty or blank image. The gradient above shows which pixels are different than what we would expect given a data set (whether some should be brighter, darker or stay the same).

4.3 Gradient Descent

Now that we have computed a gradient, we can now start doing gradient descent. Gradient Descent is an optimization algorithm for finding local minimum within a differentiable function. We can use this to find values of parameters that minimize some sort of cost function. Here, the parameters as simply each pixel in our image, and the cost functions that we are trying to minimize is the loss function.

Heres how it works. We first calcualte the loss at the current position. The algorithm then iteratively calculates the next image by using the gradient at the current position. We scale the gradient so that we can obtain a loss less than the current loss and then subtract the scaled gradient from the image. This is called a single step. We subtract the gradient because we are looking to minimise the function rather than maximizing it.

We take a bunch of these steps until we hit some stopping condition (here it is np.min(np.abs(grad)) ≤ 0.0000001).

```
def gradient_descent(image, data_list, coords, coeffs = None, FOV = 100*u.uas.to(u.
↪rad), stopper = None, dirty = False):
    """
    Performs gradient descent to reconstruct the image
    Args:
        image is a 80x80 pixel image that represents our reconstructed image
        data_list is a list of data objects
        coords is a list of u,v coordinates that we obtained from our data
        coeffs is the precomputed coefficients
```

(continues on next page)

(continued from previous page)

```
        FOV is the Field of view from the telescopes. For the EHT data, our FOV is␣
↪100 micro ascs.
        stopper allows the descent to stop at 20 iterations
        dirty changes the gradient mode to dirty kernel
    Returns:
        the reconstructed image
    """
    image_copy = np.copy(image, subok=True) # Uses copy of the image due to lists␣
↪being mutable in python
    i = 0
    grad = None
    # Can also use max here, min just makes it finish quicker
    while grad is None or np.min(np.abs(grad)) > 0.00001:
        t = 10000000 # Initial Step size which resets each iteration
        prev_loss = loss(image_copy, data_list, coords, FOV=FOV)

        if dirty:
            grad = dirty_gradient(data_list, coords, coeffs, image_copy)
        else:
            grad = gradient_finite_differences(data_list, coords, image_copy, FOV=FOV)

        new_image = image_copy - t * grad.real
        new_loss = loss(new_image, data_list, coords, FOV=FOV)

        while new_loss > prev_loss: # Only run when new_loss > prev_loss
            new_image = image_copy - t * grad.real
            new_loss = loss(new_image, data_list, coords, FOV=FOV)
            t /= 2

        image_copy -= t * 2 * grad.real # Multiply by 2 to undo last divide in the␣
↪while loop
        i += 1
        if stopper != None:
            if i == stopper: # Hard stop here for notebook purposes
                return image_copy
        print("loss:",new_loss)
    return image_copy
```

Here is what the gradient descent's output looks like after just 15 iterations. The code should be ran using

```
reconstructed_img = gradient_descent(emp, data_list, coords)
```

You will notices that the loss decreases very rapidly

```
reconstructed_img = gradient_descent(emp, data_list, coords, FOV = 1, stopper = 15)
plt.figure()
plt.imshow(reconstructed_img, vmin=0, vmax=np.max(sample))
plt.axis('off')
plt.title("Reconstructed Image")

# Use a graph here
```

```
loss: 96950.54834705345
```

```
loss: 5756.369467490237
```

```
loss: 2443.326037681071
```

```
loss: 1161.8645100715955
```

```
loss: 575.5540242488119
```

```
loss: 295.8916945789288
```

```
loss: 157.12341834262693
```

```
loss: 85.67156073552526
```

```
loss: 47.66638910484922
```

```
loss: 39.858293701477656
```

```
loss: 17.990132545746114
```

```
loss: 8.359641740332783
```

```
loss: 3.9768029269520992
```

```
loss: 1.9070653793590802
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[6], line 1
----> 1 reconstructed_img = gradient_descent(emp, data_list, coords, FOV = 1,␣
 ↪stopper = 15)
      2 plt.figure()
      3 plt.imshow(reconstructed_img, vmin=0, vmax=np.max(sample))

Cell In[5], line 26, in gradient_descent(image, data_list, coords, coeffs, FOV,␣
 ↪stopper, dirty)
     24     grad = dirty_gradient(data_list, coords, coeffs, image_copy)
     25 else:
---> 26     grad = gradient_finite_differences(data_list, coords, image_copy,␣
 ↪FOV=FOV)
     28 new_image = image_copy - t * grad.real
     29 new_loss = loss(new_image, data_list, coords, FOV=FOV)

Cell In[2], line 40, in gradient_finite_differences(data_list, coords, image, mode,
 ↪ FOV)
     38 for col in range(len(image[row])):
     39     image_copy[row,col] += 1e-8
```

(continues on next page)

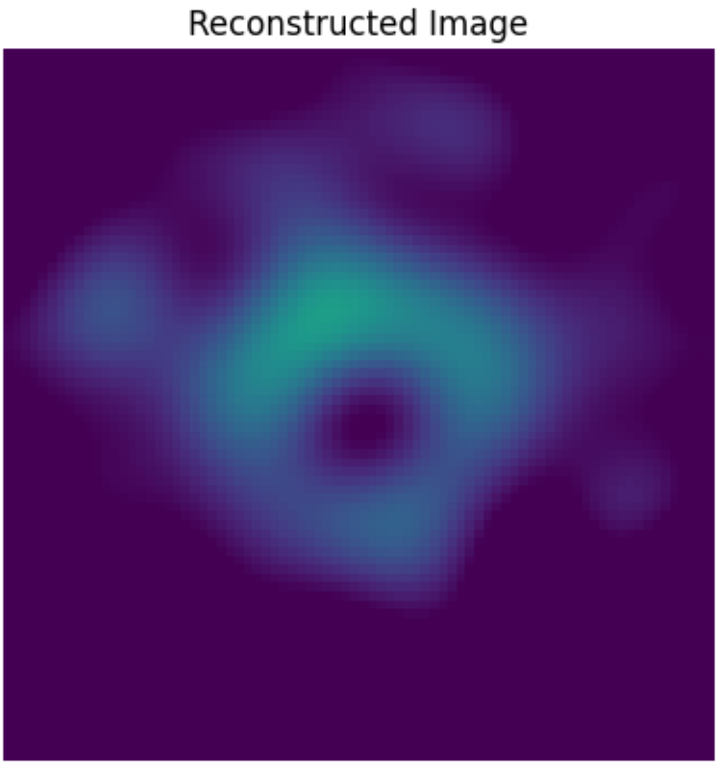
(continued from previous page)

```
---> 40      upper_diff = loss(image_copy, data_list, coords, FOV=FOV)
     41      gradient_arr[row,col] = (upper_diff - lower_diff) / 1e-8
     42      image_copy[row,col] = image[row,col]

File /var/folders/_b/trlmhkgj5xq968yccj4vtg1c0000gn/T/ipykernel_80100/353158488.
 ↪py:22, in loss(image, data_list, coords, p, reg_weight, FOV)
     19      error = (abs(vis_image-vis_data) / data_list[i].sigma) ** 2
     20      error_sum += error
---> 22 return error_sum + reg_weight * calc_regularizer(image=image, tsv=True,␣
 ↪p=2)

File /var/folders/_b/trlmhkgj5xq968yccj4vtg1c0000gn/T/ipykernel_80100/469347306.
 ↪py:23, in calc_regularizer(image, tsv, p)
     20      image_upshift[-1] = image_upshift[-2]
     22      term_1 = np.power(np.absolute(np.subtract(image_lshift, image)),p)
---> 23      term_2 = np.power(np.absolute(np.subtract(image_upshift, image)),p)
     24      reg = np.sum(np.add(term_1,term_2))
     25 return -1 * reg

KeyboardInterrupt:
```

After another 20 iterations, you should see this:

```
reconstructed_img = np.loadtxt("./images/reconstructed_img.csv", delimiter=",")
plt.figure()
plt.imshow(reconstructed_img, vmin=0, vmax=np.max(sample))
plt.axis('off')
plt.title("Reconstructed Image")
```

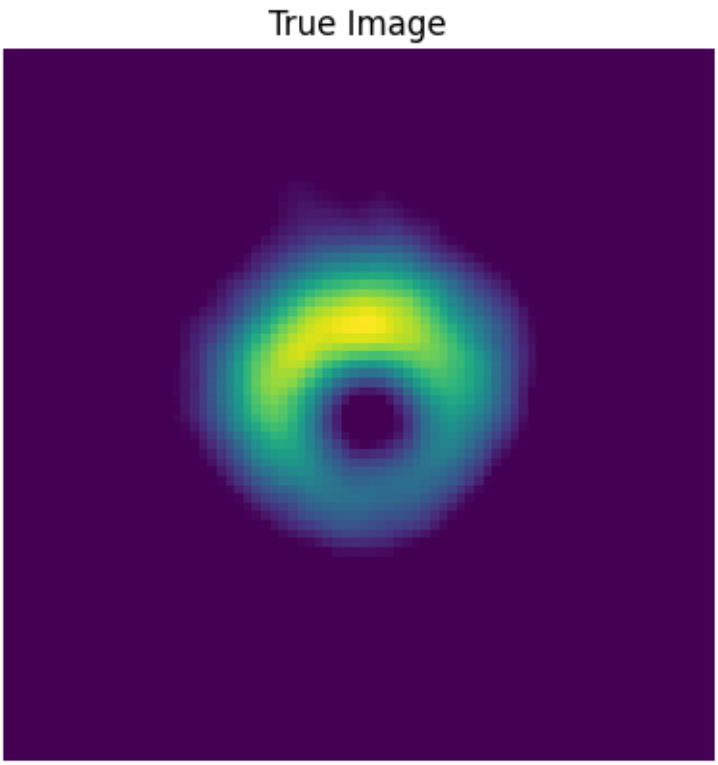
```
Text(0.5, 1.0, 'Reconstructed Image')
```



Here is the original image that the data was taken from

```
plt.figure()
plt.imshow(sample, vmin=0, vmax=np.max(sample))
plt.axis('off')
plt.title("True Image")
```

```
Text(0.5, 1.0, 'True Image')
```



As you can see here, the reconstructed image doesn't replicate the original image directly but it will still reconstruct some of the big important features that EHT Analysis uses. Here we can see the general shape of the reconstructed image matches up with the original image

zig zagging of gradient descent

Run gradient on actual data for comparison with dirty kernel

```
%run ./utility.ipynb
```

CHAPTER

FIVE

GRADIENT CALCULATIONS: "DIRTYING" THE IMAGE

5.1 Method two: Dirtying the Image

Here, we use a different method of computing gradients in order to speed up the gradient calculation process. By doing so, we should also see an increase in speed for the gradient descent.

First we start with the Gradient calculation. Consider the following loss term $$L_{data}(I) = \sum_a \frac{|I(u_a,v_a)-O_a|^2}{\sigma_a^2}$$

Here a represents which data point, O_a represents the data point itself, i and j are pixel coordinates, I_{ij} is the image. By doing this, we get the gradient to be: $$G_{ij} = \frac{\partial L_{data}}{\partial I_{ij}} = \sum_a \frac{1}{\sigma_a^2}(\frac{\partial I(u_a,v_a)}{\partial I_{ij}}(I^*(u_a,v_a) - O_a^*) + (I(u_a,v_a) - O_a)\frac{\partial I^*(u_a,v_a)}{\partial I_{ij}})$$

Ideally we can compute the partial derivates as $$\frac{\partial I(u_a,v_a)}{\partial I_{ij}} = exp(\frac{2\pi i}{N}(iu_a + jv_a)), \frac{\partial I^*(u_a,v_a)}{\partial I_{ij}} = exp(-\frac{2\pi i}{N}(iu_a + jv_a))$$

N is the size of the image in pixels

We can further define things like so:

We introduce the so-called "dirty" image from observations as $$DO_{ij} = \sum_a \frac{1}{\sigma_a^2}\frac{\partial I^*(u_a,v_a)}{\partial I_{ij}} O_a$$

Here $I^*(u_a, v_a)$ is the fourier domain of the image at an observed data point.

We also define the "dirtying" of the image I as $$DI_{ij} = \sum_a \frac{1}{\sigma_a^2}\frac{\partial I^*(u_a,v_a)}{\partial I_{ij}} I(u_a, v_a)$$

This then gives us that $$G_{ij} = 2Re(DI_{ij} - DO_{ij})$$

```
sample = np.loadtxt("images/data.csv", delimiter=",")
coords, data_list = do_sample(25)
emp = np.zeros((80,80))
```

Here we preprocess and compute $\frac{\partial I(u_a,v_a)}{\partial I_{ij}}$, $\frac{\partial I^*(u_a,v_a)}{\partial I_{ij}}$

```
def preprocess_gradient(data_list, coords, image):
    """
    Precomputed the coefficients decribed in dirty gradient
    Args:
        data_list is a list of data objects
        coords is a list of u,v coordinates that we obtained from our data
        image is a 80x80 pixel image that represents our reconstructed image
    Returns:
        a 4d list of coefficients
    """
    r, c = np.shape(image)
```

(continues on next page)

(continued from previous page)

```
    preprocessed = np.empty([r,c,len(data_list),2], dtype=np.complex_)
    for row in range(len(image)):
        for col in range(len(image[row])):
            for datum in range(len(data_list)):
                term = ((2*np.pi*1j)/image.size)*(row*coords[datum][0] +␣
↪col*coords[datum][1]) #.size for numpy array returns # of rows * # of cols
                term_1 = np.exp(term)
                term_2 = np.exp(-1*term)
                preprocessed[row,col,datum,0] = term_1
                preprocessed[row,col,datum,1] = term_2
    return preprocessed
```

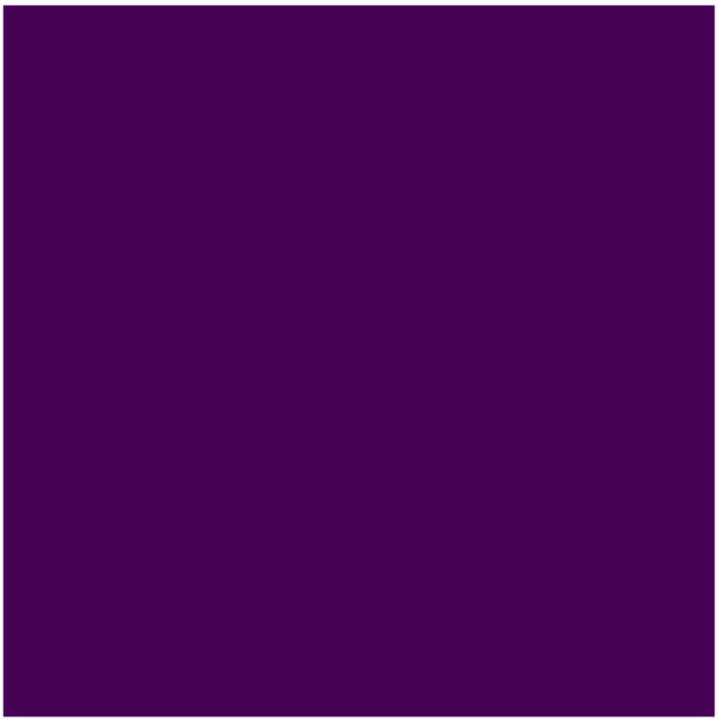
```
coeffs = preprocess_gradient(data_list, coords, emp)
```

Now, we can use the formula above to compute the gradient of the image directly

```
def dirty_gradient(data_list: list[data], coords, coeffs, image, FOV = 100*u.uas.to(u.
↪rad)):
    """
    Calculates a gradient based on dirting the image
    Args:
        data_list is a list of data objects
        coords is a list of u,v coordinates that we obtained from our data
        coeffs is the precomputed coefficients
        image is a 80x80 pixel image that represents our reconstructed image
        FOV is the Field of view from the telescopes. For the EHT data, our FOV is␣
↪100 micro ascs.
    Returns:
        the gradient of the loss function
    """
    gradient_arr = np.empty(np.shape(image)) # Because we are in real space
    vis_images = interpolate(image, coords, FOV)
    for row in range(len(image)):
        for col in range(len(image[row])):
            gradient_sum = 0
            for i in range(len(data_list)):
                vis_data = data_list[i].vis_data
                vis_image = vis_images[i] # Ask about this on Wednesday
                term_1 = coeffs[row,col,i,0] * (np.conj(vis_image) - np.conj(vis_
↪data))
                term_2 = coeffs[row,col,i,1] * (vis_image - vis_data)
                gradient_sum += (term_1 + term_2)/(data_list[i].sigma ** 2)
            gradient_arr[row,col] = gradient_sum.real
    return gradient_arr


# Why is it made of fourier harmonics
# Setting Visibilities of 1 for the purposes of this work
```

```
x = dirty_gradient(data_list, coords, coeffs, sample, FOV = 1)
plt.figure()
plt.imshow(x)
plt.axis('off')
```

```
(-0.5, 79.5, 79.5, -0.5)
```



Below is the gradient descent routine done in the fine gradient code

```
def gradient_descent(image, data_list, coords, coeffs = None, FOV = 100*u.uas.to(u.
↪rad), stopper = None, dirty = False):
    """
    Performs gradient descent to reconstruct the image
    Args:
        image is a 80x80 pixel image that represents our reconstructed image
        data_list is a list of data objects
        coords is a list of u,v coordinates that we obtained from our data
        coeffs is the precomputed coefficients
        FOV is the Field of view from the telescopes. For the EHT data, our FOV is␣
↪100 micro ascs.
        stopper allows the descent to stop at 20 iterations
        dirty changes the gradient mode to dirty kernel
    Returns:
        the reconstructed image
    """
    image_copy = np.copy(image, subok=True) # Uses copy of the image due to lists␣
↪being mutable in python
    i = 0
    grad = None
    # Can also use max here, min just makes it finish quicker
    while grad is None or np.min(np.abs(grad)) > 0.00001:
        t = 10000000 # Initial Step size which resets each iteration
        prev_loss = loss(image_copy, data_list, coords, FOV=FOV)

        if dirty:
```

(continues on next page)

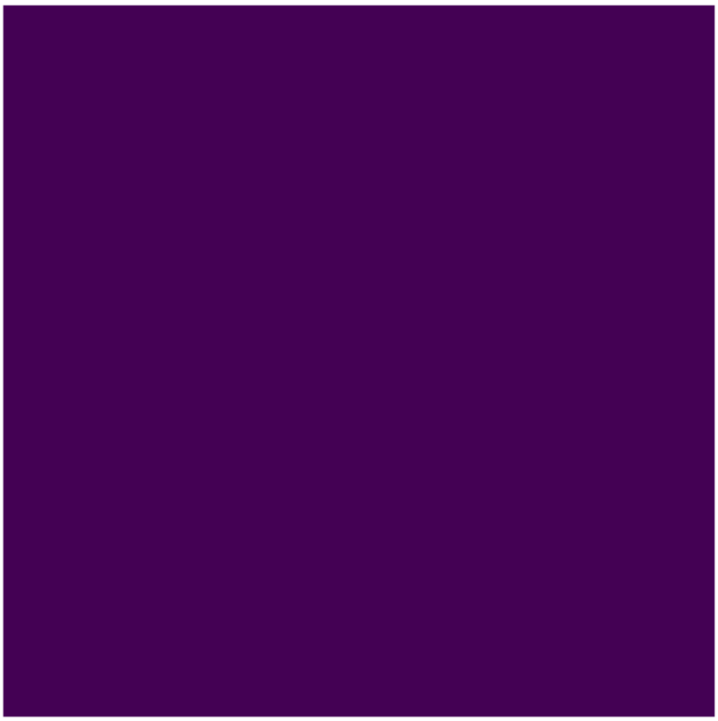
(continued from previous page)

```
            grad = dirty_gradient(data_list, coords, coeffs, image_copy)
        else:
            grad = gradient_finite_differences(data_list, coords, image_copy, FOV=FOV)

        new_image = image_copy - t * grad.real
        new_loss = loss(new_image, data_list, coords, FOV=FOV)

        while new_loss > prev_loss: # Only run when new_loss > prev_loss
            new_image = image_copy - t * grad.real
            new_loss = loss(new_image, data_list, coords, FOV=FOV)
            t /= 2

        image_copy -= t * 2 * grad.real # Multiply by 2 to undo last divide in the␣
↪while loop
        i += 1
        if stopper != None:
            if i == stopper: # Hard stop here for notebook purposes
                return image_copy
        print("loss:",new_loss)
    return image_copy
```

```
# x = gradient_descent(emp,data_list,coords,coeffs)
```

```
plt.figure()
plt.imshow(x)
plt.axis('off')
```

```
(-0.5, 79.5, 79.5, -0.5)
```



5.1.1 Continuing Further

Now lets say that DFT (Discrete fourier transform) is defined as $$I(u,v) = \frac{1}{N}\sum_{i,j=0}^{N-1} exp(-\frac{2\pi i}{N}(iu+jv))I_{ij}I_{ij} = \frac{1}{N}\sum_{i,j=0}^{N-1} exp(\frac{2\pi i}{N}(iu+jv))I(u,v)$$

The interpolation scheme computes $I(u_a, v_a)$ as $I(u_a, v_a) = \sum_{u,v=0}^{N-1} W(u_a, v_a|u, v)I(u, v)$ where $W(u_a, v_a|u, v)$ stands for the weights of u_a, v_a at u and v. The interpolation scheme will compute with only a few weights being non-zero for each a (assuming the interpolation is done in a local way). Putting this expression into $I(u, v)$ we get

$$\frac{\partial I(u_a, v_a)}{\partial I_{ij}} = \frac{1}{N}\sum_{u,v=0}^{N-1} W(u_a, v_a|u, v)exp(-\frac{2\pi i}{N}(iu+jv))$$

We this, we see that the partial derivative $\frac{\partial I(u_a,v_a)}{\partial I_{ij}}$ is the interpolation of the Fourier transform kernel.

Now, we can rewrite our formula for the Dirty Image.

$$DI(u,v) = \frac{1}{N}\sum_{i,j} exp(-\frac{2\pi i}{N}(iu+jv))DI_{ij}$$

$$= \frac{1}{N}\sum_{i,j} exp(-\frac{2\pi i}{N}(iu+jv))\sum_a \frac{1}{\sigma_a^2}\frac{\partial I^*(u_a,v_a)}{\partial I_{ij}}I(u_a,v_a)$$

$$= \frac{1}{N}\sum_{i,j} exp(-\frac{2\pi i}{N}(iu+jv))\sum_a \frac{1}{\sigma_a^2}\frac{1}{N}\sum_{u',v'=0}^{N-1} W(u_a,v_a|u',v')exp(-\frac{2\pi i}{N}(iu'+jv'))I(u_a,v_a)$$

$$= \sum_a \frac{1}{\sigma_a^2}W(u_a,v_a|u,v)I(u_a,v_a)$$

$$= \sum_a \frac{1}{\sigma_a^2}W(u_a,v_a|u,v)\sum_{u',v'=0}^{N-1} W(u_a,v_a|u',v')I(u',v')$$

Now lets finally introduce the "dirty kernel" DK. $$DK(u,v|u',v') = \sum_a \frac{1}{\sigma_a^2}W(u_a,v_a|u,v)W(u_a,v_a|u',v')$$

This transforms the dirty image equation to be $$DI(u,v) = \sum_{u',v'=0}^{N-1} DK(u,v|u',v')I(u',v')$$

Notice here that for each (u, v) there are only a few points (u', v') close by where $DK(u, v|u', v') \neq 0$. Thus for each point (u, v) in the Fourier domain, we can create a small list of (u', v') points with tabulated DK values.

Another important note is that we can dirty the image quickly since the dirty kernel is able to be precomputed. We can do so since it only depends on observational data. The time complexity of going from the image to the fourier domain using DFT is about $O(N^2 lnN)$ whereas the time complexity of going from image to dirty image by using the dirty kernel method is about $O(N^2)$.

The code below was written by Misha Stepanov in order to quickly calculate DI, DO, and DK.

```
import numpy as np
N = 80;  coeff = 4.84813681109536e-10;  Breg = 10000.;  Nreg = 100.

# reading data
list_of_strings = []
with open('./data/SR1_M87_2017_095_hi_hops_netcal_StokesI.csv', 'r') as f:
  for line in f:
    list_of_strings.append(line)
list_of_strings.pop(0)
Na = len(list_of_strings)
```

(continues on next page)

(continued from previous page)

```
print(len(list_of_strings))
UVa, sigma = np.zeros((2*Na, 2)), np.zeros(2*Na)
Oa = np.zeros(2*Na).astype(complex)
for a in range(Na):
  current_string = (list_of_strings[-1]).split(",") # Split on commas
  list_of_strings.pop()
  UVa[a, 0], UVa[a, 1] = float(current_string[3]), float(current_string[4])
  Oa[a] = float(current_string[5])*np.exp(1j*float(current_string[6])*np.pi/180.)
  sigma[a] = float(current_string[7])
  UVa[a + Na], Oa[a + Na], sigma[a + Na] = -UVa[a], np.conj(Oa[a]), sigma[a]
del list_of_strings, current_string

# local cubic interpolation
def cubf1(x, y):                                #      8  11
  cf1 = (1. + x - x*x)*(1. + 2.*y)*(1. - y)    #   8  2  3  10
  cf2 = (1. + y - y*y)*(1. + 2.*x)*(1. - x)    #   4  0  1  6
  return 0.5*(cf1 + cf2)*(1. - x)*(1. - y)     #      5  7
def cubf2(x, y):  return -0.5*x*(1. - x)*(1. - x)*(1. + 2.*y)*(1. - y)*(1. - y)
IND = np.array([[-2, -2], [-1, -2], [-2, -1], [-1, -1], [-3, -2], [-2, -3], \
  [0, -2], [-1, -3], [-3, -1], [-2, 0], [0, -1], [-1, 0]]).astype(int)
def cubfun12(x, y):  return np.array([cubf1(x, y), cubf1(1. - x, y), \
    cubf1(x, 1. - y), cubf1(1. - x, 1. - y), cubf2(x, y), cubf2(y, x), \
    cubf2(1. - x, y), cubf2(y, 1. - x), cubf2(x, 1. - y), cubf2(1. - y, x), \
    cubf2(1. - x, 1. - y), cubf2(1. - y, 1. - x)])

"""
Calculate DO
Compare G = -2DO
"""

# computing DO, the dirty image from observations
DOuv = np.zeros((N, N)).astype(complex)
for a in range(2*Na):
  m = np.mod(coeff*UVa[a], N);  x, y = m - np.floor(m)
  m1, m2 = np.floor(np.mod(m + np.array([2., 2.]), N)).astype(int)
  C, CUBFUN = Oa[a] / sigma[a]**2, cubfun12(x, y)
  for c in range(12):  DOuv[m1 + IND[c, 0], m2 + IND[c, 1]] += C*CUBFUN[c]
DO = N*np.real(np.fft.ifft2(DOuv))
del DOuv

# computing DK, the dirty kernel
DK49, DK = np.zeros((N, N, 7, 7)), []
for a in range(2*Na):
  m = np.mod(coeff*UVa[a], N);  x, y = m - np.floor(m)
  m1, m2 = np.floor(np.mod(m + np.array([2., 2.]), N)).astype(int)
  C, CUBFUN = 1. / sigma[a]**2, cubfun12(x, y)
  for c in range(12):
    for cp in range(12):
      DK49[m1 + IND[c, 0], m2 + IND[c, 1], 3 + IND[cp, 0] - IND[c, 0], \
           3 + IND[cp, 1] - IND[c, 1]] += CUBFUN[c]*CUBFUN[cp]*C
for u in range(N):
  for v in range(N):
    for du in range(7):
      for dv in range(7):
        if (DK49[u, v, du, dv] != 0.):
          DK.append([u, v, ((u + du) % N) - 3, ((v + dv) % N) - 3, \
```

(continues on next page)

(continued from previous page)

```
            DK49[u, v, du, dv]])
del DK49, IND, CUBFUN

# computing DI, the dirtying of the image I
I12, Iuv = np.zeros((2, N, N)), np.zeros((N, N)).astype(complex)
DI, DIuv = np.zeros((N, N)), np.zeros((N, N)).astype(complex)
def calc_DI():
  global Iuv, DI, DIuv
  Iuv, DIuv = np.fft.fft2(I), np.zeros((N, N)).astype(complex)
  for q in DK:  DIuv[q[0], q[1]] += q[4]*Iuv[q[2], q[3]]
  DI = np.real(np.fft.ifft2(DIuv))
  return

# computing G, the gradient of the loss function
def neighbor(Iat, Inear):  return np.sign(Iat - Inear)
def calc_G():
  global G
  calc_DI()
  G = 2.*(DI - DO)
  for i in range(N):
    for j in range(N):
      if (I[i, j] < 0.):  G[i, j] -= Breg
      if (I[i, j] > 0.):  G[i, j] += Nreg
  return

I, dt = (0.*DO), 0.00001

calc_DI()
for i in range(N):
  for j in range(N):
    print(DO[i, j], ' ', end='')
  print('')
```

```
6458
```

```
58323.23350943204  64183.049640411664  70342.19688473325  76463.96037976317  82220.
↪16597132056  87305.73622052035  91452.51766566507  94441.89074861264  96115.
↪64259028548  96384.5643224987  95234.24157287803  92727.55055779137  89003.
↪46182561584  84271.8933431409  78804.5430845263  72921.86116885504  66976.
↪57946460739  61334.483778300266  56353.36785037754  52361.32520517919  49635.
↪69081987981  48384.01898300869  48728.46133707091  50694.78195464391  54207.
↪015032441996  59088.44528819557  65069.190279815084  71800.21437073575  78873.
↪13862691575  85844.76562534264  92264.8494482344  97705.34272347736  101789.
↪17231835447  104216.55259778662  104786.94895689753  103415.05197293212  100139.
↪49958876801  95123.56595481075  88647.58658445961  81093.4695900302  72922.
↪20808565989  64645.81592598703  56795.518366710385  49888.30886180076  44394.
↪1109272239  40705.74954130665  39113.742389092025  39787.58239324253  42764.
↪72567910187  47947.95873747794  55111.2365366803  63913.503840376696  73919.
↪47821806502  84625.9239720032  95491.61274585147  105968.97037271332  115535.
↪3606766145  123722.05375312216  130139.15571903337  134495.1162118896  136609.
↪84864778892  136420.9612824536  133983.06772195833  129460.58846817125  123114.
↪83950768145  115286.50572520071  106374.80025048862  96814.70908092597  87053.
↪71588114378  77529.30534883417  68648.37214613013  60769.43781317713  54188.
↪32409417533  49127.67118937375  45730.444540740325  44057.36047695973  44087.
↪99045163279  45725.18068693345  48802.34783536612  53093.175725431065
```

(continues on next page)

(continued from previous page)

```
69268.61646341992  74102.41231054289  78897.38264590115  83341.52181703526  87143.
↪27663147583  90045.551840139  91838.58371658273  92371.17138744085  91559.
↪73476230988  89394.66681449201  85943.47890954457  81350.31117960578  75831.
↪50213486183  69667.08414232201  63188.28937743438  56761.403374394424  50768.
↪573587620835  45586.447060227685  41563.75021915356  38999.11028982266  38120.
↪52910518491  39067.93785643205  41880.173902569695  46487.524751196586  52710.
↪68568402082  60266.59163594924  68781.1349827288  77808.30011190605  86854.
↪76908274583  95408.61822513201  102970.3697354065  109084.41715863881  113368.
↪73351204777  115540.81016118536  115437.96630613948  113030.50375824387  108426.
↪63846633483  101868.6874832218  93720.58822545008  84447.43152786019  74588.
↪25533618289  64723.828519627394  55441.51680959373  47299.536971141206  40792.
↪95415574243  36323.65702231886  34176.26492825879  34501.50301029944  37308.
↪05665241402  42463.32695785696  49702.89827444962  58647.94333002596  68829.
↪2740615262  79716.33346866752  90749.14342148192  101371.09185029275  111060.
↪46459063372  119358.79470499212  125894.39668146493  130399.8467514119  132722.
↪6294757055  132828.657195727  130798.8453404565  126819.35839732236  121166.
↪49984084666  114187.48356653632  106278.4826107054  97861.40086122142  89360.
↪76179182739  81181.96940788876  73691.9907902648  67203.2605395588  61961.
↪33968320186  58136.598381288386  55819.952673017186  55022.48504228426  55678.
↪62520729358  57652.46347084936  60746.71067479303  64713.7981156942
81696.29625642695  85341.27603222559  88591.28893947971  91166.58227661085  92819.
↪56396436229  93347.99156874693  92606.6476589365  90516.97669704602  87074.
↪14895873857  82351.03369559073  76498.61862953188  69742.5146981215  62375.
↪337899930906  54744.962446437145  47238.88312226474  40265.195417536015  34230.
↪97983704157  29519.13815909857  26464.94890076508  25333.76213820692  26301.
↪31871211577  29438.140028538513  34699.2842274399  41920.50370331004  50821.
↪47889267959  61016.36447300263  72031.39553299343  83328.79802016518  94335.
↪76851208572  104476.87208628631  113207.88972839697  120048.95797947049  124614.
↪80393948455  126639.99765520428  125997.41824748096  122708.54421913858  116944.
↪70492497188  109019.03218349899  99369.48460999908  88533.9352104975  77118.
↪86829004505  65763.68226099777  55102.905743899915  45728.78137337584  38156.
↪64358450008  32795.31563328227  29924.392916968318  29679.792150528116  32048.
↪36676532308  36871.76232736727  43859.05913151086  52607.16885767307  62627.
↪46012266049  73376.71789358441  84290.31758043906  94815.42767860103  104442.
↪1439570564  112730.69016315378  119333.17125388086  124008.80329160493  126632.
↪03217907046  127193.4522827111  125793.90807590753  122632.57409450576  117990.
↪13461886288  112208.4072432434  105667.86619790248  98764.52392550965  91887.
↪53369302861  85398.69974277019  79614.84734032457  74793.73812768453  71123.
↪94131861944  68718.8108243942  67614.49025683294  67771.68403387169  69080.
↪79924155188  71369.97896842222  74415.50757360147  77954.06177509871
95329.43646191097  97701.07848548933  99310.0548408521  99914.55914385623  99316.
↪26810676866  97372.49006662502  94006.44533848175  89215.1463864503  83074.
↪34767030797  75740.06959230505  67446.27915492962  58498.43879575681  49262.
↪816057884884  40151.675602096904  31604.740257855774  24067.59187725554  17967.
↪963295721354  13691.12424807082  11555.760466626689  11791.86214759995  14522.
↪15581055888  19748.519503858966  27344.611292059242  37055.6208295532  48505.
↪63950031913  61212.66138278243  74610.70820898865  88078.05483835538  100970.
↪05808892549  112654.7004041821  122548.68547474181  130151.7924404373  135077.
↪2250760977  137075.88657631556  136052.86033801545  132074.86093313078  125368.
↪0044378568  116305.89177582122  105388.65548083537  93214.24021036661  80443.
↪72352854273  67762.89503051052  55842.56730204883  45300.17201865957  36665.
↪092838269295  30349.911890726187  26629.320276957944  25627.89829913348  27317.
↪350183242612  31523.12816198824  37939.7507000146  46153.55555501295  55671.
↪170237885126  65951.66091114469  76440.15494983643  86600.72863481451  95946.
↪50325721661  104065.18193301311  110638.65741636606  115455.79276893973  118417.
↪98259287572  119537.60291248038  118929.91581192022  116799.37951934655  113421.
↪60275363007  109122.36029873871  104255.15142509385  99178.73999100282  94235.
↪97917753321  89735.01551315279  85933.71070938991  83027.84170941805  81143.
↪3638142162  80332.76983199161  80575.36568826954  81781.11934889117  83797.
↪6281926473  86419.68696111714  89400.91594912692  92466.91563442982
```

(continues on next page)

(continued from previous page)

```
109806.45314062026  110897.64175596692  110854.67238331687  109477.45697247807 ␣
↪106620.13458030505  102201.99990066176  96216.24188215408  88735.96041429423 ␣
↪79916.94269130702  69996.73293374313  59289.62950286488  48177.39724678964 ␣
↪37095.68931414145  26516.42406497802  16926.644791800583  8804.682521866773 ␣
↪2594.720629445561  -1318.9024441864312  -2636.1458173903593  -1162.888017119134 ␣
↪3171.8393440633595  10304.416726308353  20032.146453719797  32016.480145282956 ␣
↪45794.27647533005  60796.88356913318  76376.30036774816  91837.15044346041 ␣
↪106472.74162367417  119603.12434391452  130612.83372810623  138985.92861512623 ␣
↪144336.03716977837  146429.38237346942  145199.17725515945  140750.3222897011 ␣
↪133353.9683688298  123432.1827186171  111533.62241736519  98301.73081758653 ␣
↪84437.48006032515  70659.04945600967  57661.02704428191  46075.73546563442 ␣
↪36439.112991570844  29163.24006609539  24517.118005366134  22616.717397137618 ␣
↪23424.664700549714  26759.27591077267  32312.025092706303  39671.99813464367 ␣
↪48355.46556962992  57838.43958596477  67589.97383349854  77104.02159043532 ␣
↪85927.87673040939  93685.55978084618  100094.9462178217  104977.9280314324 ␣
↪108263.41198622044  109983.44930092352  110263.22629067232  109305.99540009345 ␣
↪107374.27147322109  104768.7495175924  101806.41819982274  98799.25773391128 ␣
↪96034.7392507868  93759.10818206299  92164.16252687741  91377.95414601595  91459.
↪57115624336  92397.92137388284  94114.24398089838  96467.93592629809  99265.
↪19119800808  102269.90933770358  105216.32429766971  107822.82317939089
124694.9056346577  124571.2213606769  122946.96419451728  119665.63014578054 ␣
↪114634.90812243936  107836.25445692486  99331.99367745934  89269.42059478417 ␣
↪77881.40410354578  65483.06192127118  52464.19637007553  39277.35742678955 ␣
↪26421.62726998885  14422.49008993675  3808.4449896085425  -4914.684161516496  -
↪11290.51631947676  -14938.51284788677  -15578.421866067612  -13051.13848415851  -
↪7334.42579161624  1447.86291521167  13024.106379524856  26981.410146505426 ␣
↪42780.40474627589  59777.9345845099  77256.67391143  94460.19031962263  110631.
↪53719418128  125053.13227047556  137085.50035245536  146202.44435840892  152020.
↪3677943296  154319.79759011552  153057.62960202567  148369.20839736806  140560.
↪01669451254  130087.43982557434  117533.73573611982  103571.93150582636  88926.
↪83866452084  74333.69617830086  60497.088298538285  48052.73476748072  37534.
↪51805683237  29348.715461872926  23756.87461063615  20868.150771602086  20641.
↪26114173936  22895.55537755921  27330.101608448575  33549.186061205575  41092.
↪25629551243  49466.1254973555  58177.207932060155  66761.66983443499  74811.
↪63949582855  81995.99818358291  88074.73486563044  92906.35282691845  96448.
↪32425814547  98751.06115248053  99946.27414928291  100230.9000020134  99847.
↪97736795526  99065.93413874025  98157.72208028116  97381.10949900944  96961.
↪240298357  97076.31292413875  97846.95189404098  99329.5635376542  101513.
↪70860903405  104323.3047515403  107621.30217353093  111217.36027839378  114877.
↪98884005715  118338.59664443939  121316.90151722216  123527.18464869722
139509.69896139827  138301.23477049952  135240.52918369597  130214.7173870718 ␣
↪123184.11323681366  114190.34070360941  103361.75025705429  90915.60932150061 ␣
↪77156.59263280887  62471.18275625224  47317.73031362692  32212.118759650715 ␣
↪17709.22407630176  4380.642646826978  -7210.538527909616  -16536.860636365855  -
↪23134.17164831229  -26628.336589208502  -26759.339759840048  -23401.13508853929 ␣
↪-16575.71699375818  -6460.123802118462  6614.558445280469  22172.670104542827 ␣
↪39611.884184848335  58228.940486758656  77252.05154948687  95878.32225600454 ␣
↪113314.11400178977  128816.00062880896  141729.83435580705  151525.4819093857 ␣
↪157825.00712705014  160422.45585881532  159293.9181581707  154597.16600491497 ␣
↪146660.8476463409  135963.91209389863  123106.58789996445  108774.80029383543 ␣
↪93700.33800650321  78619.34375156733  64231.7812330601  51164.421939589316 ␣
↪39939.60676799208  30951.594413568306  24451.745949315504  20543.157332498373 ␣
↪19184.688009386708  20203.694586714533  23316.21081325975  28152.859083579457 ␣
↪34288.464562857225  41273.188891033744  48663.01105014022  56047.549975089896 ␣
↪63073.526461503425  69462.5706623584  75022.5588811914  79652.16892234268  83338.
↪83653085149  86150.73955765364  88223.80099266737  89744.96516430574  90933.
↪15144255909  92019.32526526105  93227.05549679551  94754.766505055  96760.
↪66543982828  99351.0560056586  102572.46623964253  106407.74419711313  110776.
↪03260641808  115536.3362351453  120494.25191471659  125411.34201449466  130016.
↪5925151468  134019.39664522407  137123.53112220162  139041.63012225792
```

(continues on next page)

(continued from previous page)

```
153734.76724530727  151625.00570658254  147336.17149328877  140797.4508595139 ␣
↪132019.01668840443  121098.67076709909  108225.57630992768  93680.59070856898 ␣
↪77832.75595582991  61131.60270541215  44095.07905829488  27293.126029796476 ␣
↪11327.180219781128  -3193.8246267378418  -15680.08301518554  -25589.007958583334␣
↪ -32453.37866223525  -35907.96309383758  -35712.952347853876  -31772.57602377348␣
↪ -24147.42297156802  -13059.26969640964  1112.3913196217545  17842.45046422376 ␣
↪36480.693429846084  56279.950865143444  76430.56611550144  96099.38219901183 ␣
↪114471.0726143757  130789.40093005987  144395.91552516576  154763.68053876382 ␣
↪161523.91082703698  164483.80259946373  163634.40462760342  159148.01885460498 ␣
↪151365.30734578092  140772.96406740902  127973.43410931797  113648.6828685306 ␣
↪98520.39447130913  83309.18451230608  68695.4325921063  55284.1752287664  43576.
↪16405426091  33946.71550294744  26633.394959946036  21732.936578961686  19207.
↪149366771922  18896.94986544035  20543.136574725755  23812.11786256546  28324.
↪549424270153  33684.742680202995  39508.77187348903  45449.42226413215  51216.
↪460688197214  56591.14006273179  61434.3327392642  65688.18338374936  69371.
↪64114456568  72570.63878729488  75424.00610452144  78106.41837560457  80809.
↪7802821926  83724.43398142446  87021.46871130948  90837.21750616272  95260.
↪77857218604  100325.12124077204  106002.05544671715  112201.08275361803  118771.
↪9245138912  125510.35091348008  132166.81855390675  138457.36221774944  144076.
↪1710260901  148709.2984655092  152048.9954846258  153808.2015243268
166847.20503204092  164059.63074592856  158801.0751537517  151039.77137309077 ␣
↪140831.38247198204  128324.14838483342  113760.97841280326  97478.02666038451 ␣
↪79899.34513914789  61527.31999973229  42928.764364977644  24716.763690699972 ␣
↪7528.636389464242  -7999.333387070437  -21259.43518225901  -31701.111655062883  -
↪38859.11944873747  -42379.663836643915  -42042.850264509034  -37779.86261749385 ␣
↪-29683.467790868875  -18010.75579627711  -3177.44030977344  14256.457847888787 ␣
↪33609.175149840106  54107.09200681332  74920.49183850395  95203.2933519822 ␣
↪114134.52016417531  130959.07886103923  145025.38764084995  155817.54094478808 ␣
↪162980.00555477783  166333.30173592822  165879.69812159296  161798.60028886338 ␣
↪154431.99275135837  144260.9519125057  131874.8341006111  117935.21413375725 ␣
↪103136.97081619606  88169.06301061044  73677.50381051347  60232.82563247234 ␣
↪48303.95449105738  38239.90834511964  30260.14287286842  24453.735171347398 ␣
↪20786.970739000608  19118.32891100787  19219.388233194342  20799.82870086324 ␣
↪23534.51359968983  27090.598337098836  31152.73257035278  35444.67836443389 ␣
↪39746.033991449  43903.1956666024  47834.169035319384  51527.31880745559  55034.
↪58143334872  58460.03066559638  61944.95585395458  65650.77410009493  69741.
↪14638249329  74364.61070335444  79638.8970155452  85637.87051425157  92381.
↪78722741379  99831.26542141428  107885.1034855815  116381.83125311072  125104.
↪68316294208  133789.53757581674  142135.27921026078  149816.00677696505  156494.
↪51578152948  161836.52377489035  165525.15707416122  167275.26969840762
178342.65734146867  175125.884896988  169190.76659391035  160540.43459060258 ␣
↪149270.36141435275  135572.04122417944  119733.52467410392  102136.37827402703 ␣
↪83248.70533124599  63613.98632019417  43835.674139313734  24557.709483702234 ␣
↪6441.3916050021  -9860.66846756627  -23732.495395411337  -34623.32011356097  -
↪42075.29200964149  -45748.72966222775  -45443.28880327537  -41113.52456268077  -
↪32877.54439219355  -21017.78227005821  -5973.35725529529  11676.01612744372 ␣
↪31234.056113352097  51918.83088383715  72899.01879342251  93333.55373907246 ␣
↪112412.4056486729  129396.10118710545  143651.60873728473  154682.39852597186 ␣
↪162150.8341523688  165891.5335885046  165914.92246968378  162400.84791570687 ␣
↪155682.77952213457  146223.74626709858  134585.69709171518  121394.388359602 ␣
↪107302.16248931133  92951.0702026401  78938.69917799358  65788.81363039167 ␣
↪53928.50484435931  43673.03557705497  35218.97346619342  28645.596810318377 ␣
↪23923.96784897385  20932.54835632724  19477.817666388815  19318.07221225673 ␣
↪20188.454626345934  21825.28306233651  23987.918750707227  26476.701628450493 ␣
↪29145.870740858045  31910.832244885125  34749.604518487744  37698.71884825308 ␣
↪40844.25092767273  44308.9749860379  48236.849422457344  52776.15076785904 ␣
↪58062.572080189144  64203.50223376685  71264.52148665143  79258.90925747351 ␣
↪88140.68824271925  97801.45112980262  108070.95615212797  118721.2550238357 ␣
↪129473.94473859153  140010.01976180242  149981.74278414145  159025.94371193956 ␣
↪166778.18638765192  172887.29613072693  177029.80318283485  178923.9134227706
```

(continues on next page)

(continued from previous page)

```
187760.67945244213  184372.9436000082  178072.73123679883  168893.08382161413 ␣
↪156962.6188677611  142507.80650274473  125852.06211062039  107411.26520768268 ␣
↪87685.38836443062  67246.04834440589  46719.97530646369  26768.628536648994 ␣
↪8064.4553567916955  -8735.426003879496  -23017.062978283328  -34237.72741314897 ␣
↪-41952.72049405113  -45839.354965392995  -45716.557294002516  -41558.65914380552␣
↪ -33502.19086386386  -21844.84086355861  -7036.184874615742  10339.709740155697 ␣
↪29589.172268889506  49940.16326328819  70578.24909522131  90685.34182635006 ␣
↪109478.98575739504  126249.87586837818  140395.3575794208  151446.88188426814 ␣
↪159089.76211157744  163174.07342640503  163716.11848836544  160890.5096852348 ␣
↪155013.54320624657  146519.11577642628  135928.91762654623  123818.98893545056 ␣
↪110784.92550630125  97408.04916664353  84224.71913254775  71700.66558517763 ␣
↪60211.80142415167  50032.44801652217  41331.3381010295  34175.17948420171  28539.
↪021884143804  24322.207068394786  21368.332549707935  19487.444655965788  18478.
↪609098561632  18151.085183130956  18342.540638315906  18933.064455028438  19854.
↪133954911915  21092.13341347943  22686.46718960666  24722.724400828985  27321.
↪70342352192  30625.368811468477  34780.97548599881  39924.6498883401  46165.
↪66946776623  53572.543292463444  62161.78744100882  71890.03340286925  82649.
↪832113173  94269.24582313519  106515.07692101173  119099.38408356109  131688.
↪79232734523  143916.01823406146  155393.0016980391  165725.05207287282  174525.
↪4664600182  181430.1457293892  186111.80423371276  188293.4285782175
194708.74099212355  191402.61897780522  185050.4291682578  175709.60452917917 ␣
↪163534.6074049012  148777.88853852675  131787.67994367564  113002.27615081717 ␣
↪92940.53795983572  72188.48613522202  51382.038520949  31186.174973167334  12271.
↪07576850726  -4713.949505576718  -19167.668414827618  -30563.97645055982  -38477.
↪34977255594  -42605.193771606384  -42785.62094569209  -39009.34271506492  -31424.
↪621119217976  -20334.58666734909  -6186.667273846993  10445.638689139394  28887.
↪909625025644  48395.653879584606  68189.2379047769  87490.93917402707  105561.
↪98023820836  121737.35241747973  135456.34360684553  146286.94076556066  153942.
↪66791317362  158290.9177939083  159352.40364343784  157291.95433550072  152401.
↪45647544367  145076.26658214425  135786.83583743678  125047.57740398614  113385.
↪14101598389  101308.2327382599  89280.93375984622  77701.1476769393  66885.
↪36880509845  57060.45111389539  48362.510311131344  40842.554186425536  34477.
↪95048938706  29188.443994568865  24855.153370503365  21340.831976127047  18509.
↪67106675278  16245.053600724432  14463.915441558673  13126.712636356568  12242.
↪396394305086  11868.225593279923  12104.663991131769  13085.982528132474  14967.
↪488502093775  17910.5130571852  22066.395532916693  27560.706502489502  34478.
↪85861196674  42854.081239419  52658.50299758814  63797.82010835301  76109.
↪75440621533  89366.24618004863  103279.1042852709  117508.66335379572  131674.
↪8832389447  145370.26988774858  158173.99405739113  169666.62388182883  179444.
↪95492018719  187136.5011511114  192413.28719802905  195004.6432301791
198883.58047891033  195891.7945357529  189786.39645638532  180643.47375768184 ␣
↪168635.74809929013  154032.32499629035  137195.3502015068  118573.27604735098 ␣
↪98690.52489418855  78133.47229679207  57532.85713371335  37542.95065255181 ␣
↪18818.065829648316  1987.242183817878  -12371.824550612619  -23758.34854573477  -
↪31772.134254883968  -36133.95451435733  -36701.275320533379  -33478.14740543057  -
↪26618.357752588632  -16421.295839438222  -3320.417700433631  12135.343344827386 ␣
↪29306.40151193019  47492.3254768455  65965.00717783024  84003.30589246869 ␣
↪100927.13961372878  116128.99463613806  129100.96773876512  139455.73612050439 ␣
↪146940.2497121849  151441.43009126073  152983.703635649  151718.75262834  147908.
↪39418643332  141901.95267996175  134109.84178861935  124975.29104071452  114946.
↪22279498624  104449.20535835731  93867.18473748348  83522.35132950451  73665.
↪0583020985  64469.21224026802  56034.04452324512  48391.6849643506  41519.
↪53560218953  35356.114100709805  29818.826191748063  24822.047688970524  20293.
↪95025414893  16190.681680630487  12506.791464972895  9281.148486317936  6597.
↪997103925068  4583.206122724754  3396.147996303284  3217.9733148389864  4237.
↪294349735566  6634.445583116794  10565.542196217364  16147.512047601356  23445.
↪143762606924  32460.99108764158  43128.724337759166  55310.2486314459  68796.
↪64051372396  83312.71178356334  98524.8095463423  114051.31638179686  129475.
↪2289959777  144358.1660462881  158255.1782556774  170729.79411164369  181368.
↪81720579654  189796.48065766698  195687.64539339274  198779.79071428618
```

(continues on next page)

(continued from previous page)

```
200088.71733011102  197611.7982846341  192023.1285401704  183411.78162457564 ␣
↪171961.20289786515  157947.87424585834  141737.01193661703  123775.0532284395 ␣
↪104578.7735944487  84721.01119334823  64813.1546947382  45484.76470098812  27360.
↪93204311679  11038.20798435004  -2939.8509375608337  -14106.320214630732  -22092.
↪43146638693  -26645.68095746692  -27643.308647434882  -25100.120380703447  -
↪19169.904145350716  -10140.046608647266  1580.634334512204  15480.390407742952 ␣
↪30968.49926343015  47403.41252775638  64123.51981750645  80478.74221541994 ␣
↪95861.07944823698  109732.27819787244  121646.9649770942  131269.8853411707 ␣
↪138386.29117950605  142904.99004481034  144854.07919380552  144369.893079165 ␣
↪141680.1571458251  137082.72681206523  130921.56855757369  123561.78897581351 ␣
↪115365.5269236902  106670.39441004192  97771.89582639321  88910.89498936196 ␣
↪80266.76593508633  71956.39275234647  64038.714662460465  56524.08286419508 ␣
↪49387.339093882845  42583.26926682507  36062.94681478478  29789.4669387875 ␣
↪23751.681773873148  17974.764095344693  12526.73168244285  7520.427288826169 ␣
↪3110.8380340631993  -511.9802461707732  -3133.764668199001  -4529.8050927683435 ␣
↪-4480.519425197948  -2787.6625074289404  710.0147767844464  6122.749397513995 ␣
↪13498.31829409366  22813.857792403975  33971.38852988077  46797.21316938187 ␣
↪61045.09675899393  76402.91588658563  92502.28792006086  108930.57453038445 ␣
↪125244.59661557438  140985.39654623086  155693.4287607678  168923.63761820493 ␣
↪180259.97642996447  189329.0180040666  195812.39095242837  199457.83704295754
198247.0961537471  196443.57936845487  191600.52236730093  183814.63614420034 ␣
↪173272.90816344123  160250.31771875542  145104.76205578653  128269.00367978933 ␣
↪110239.53604627302  91562.39603628221  72816.12231891765  54592.25817142905 ␣
↪37474.00977936154  22013.876995283397  8711.251655220382  -2008.8922130714127  -
↪9815.030124076264  -14484.674091975132  -15915.456299543448  -14130.847813658751␣
↪ -9279.927938563502  -1630.9613747863625  8441.07666550337  20472.49085775621 ␣
↪33933.456194826016  48253.96241083895  62851.612591155295  77159.62232276425 ␣
↪90653.32753080514  102873.5931794579  113445.71796700344  122092.73877656888 ␣
↪128642.43070358994  133027.74442490254  135280.88806685098  135521.70911037977 ␣
↪133941.42783798475  130783.08606141034  126320.27882540475  120835.816807834 ␣
↪114601.9178152336  107863.35238664746  100824.68656806718  93642.39875799163 ␣
↪86422.22822297632  79221.6755228788  72057.15572598549  64914.93787508368  57764.
↪717641564  50574.48605096658  43325.28806667223  36024.51319648549  28716.
↪51870315034  21489.637956574657  14478.948078113346  7864.533312948851  1865.
↪3521262816432  -3270.834141620239  -7280.7129977441  -9900.866668010887  -10884.
↪285047513711  -10016.772155782208  -7132.118893670224  -2125.0447338717204  5038.
↪891809656503  14316.970432132346  25586.86398320057  38648.095547074336  53227.
↪17941846844  68986.02582252712  85533.04055967323  102436.26152615284  119237.
↪844001867  135469.22949233284  150666.39770408525  164384.6936003072  176212.
↪82569885196  185785.7326426434  192796.0997299422  197004.36647977182
193408.06674423628  192387.7438972173  188468.8031889821  181750.76787691342 ␣
↪172417.5989077691  160734.6038933234  147042.78176629797  131750.46978356925 ␣
↪115322.24485377476  98265.15602192392  81112.52346359132  64405.72043435615 ␣
↪48674.542855901855  34416.94823331597  22079.091893432666  12036.68551469769 ␣
↪4578.734932992127  -105.32959998472506  -1934.241757769978  -938.5192214331983 ␣
↪2741.301355276373  8861.91030493227  17089.698118824144  27017.850664583966 ␣
↪38187.13611472098  50109.181823119405  62290.84584729944  74258.19072799402 ␣
↪85578.57762466738  95879.51941136338  104863.15515404161  112315.51999397628 ␣
↪118110.1606475416  122206.05846412435  124640.2367936061  125515.81414984896 ␣
↪124986.58832216896  123239.47303252782  120476.23894032811  116896.02453917476 ␣
↪112679.97926015998  107979.19015056673  102906.74305870177  97534.40492843946 ␣
↪91894.01634428898  85983.2860997506  79775.31512209924  73230.87519670965  66312.
↪25249627371  58997.35314484491  51292.76518075282  43244.57614935061  34945.
↪94693453559  26540.721231913663  18222.681153177895  10230.414018774849  2838.
↪1035544728225  -3657.127871671153  -8950.444994977239  -12746.035508813293  -
↪14774.209696435348  -14807.720441301544  -12676.249538363554  -8278.169792357054␣
↪ -1588.9123493855368  7334.476565456345  18352.739066087146  31247.663516978424 ␣
↪45728.866951908145  61443.594008961045  77988.9024930301  94925.54346333333 ␣
↪111792.83868852966  128123.90234771927  143460.63509704292  157368.0218896818 ␣
↪169447.37534323533  179348.26916852978  186778.98984044822  191515.39155863164
```

(continues on next page)

(continued from previous page)

```
185748.18266841114  185568.7541456042  182696.0748214724  177228.3243727162 ␣
↪169340.68416941297  159281.58661783778  147366.6674057492  133970.34374648123 ␣
↪119515.0235985275  104458.06670131271  89276.7615163266  74451.74165833805 ␣
↪60449.42643216463  47704.21578505117  36601.2830404229  27460.873329529008 ␣
↪20525.019068585167  15947.518346978999  13787.885877532564  14009.783969585333 ␣
↪16484.18412262865  20997.21546851963  27262.346152581726  34936.24235672249 ␣
↪43637.38219333724  52966.291659386545  62526.136901462814  71942.36507931592 ␣
↪80880.14174784793  89058.48443343415  96260.2302713791  102337.28291593863 ␣
↪107210.93688338588  110867.44814462493  113349.37824914168  114743.55620645976 ␣
↪115166.7515685902  114750.31287711833  113625.08438255441  111907.86574918151 ␣
↪109690.52836130337  107032.66056783627  103958.30306911693  100456.98084421347 ␣
↪96488.86930679268  91993.58053783697  86901.74959860406  81148.36605312802 ␣
↪74686.65063978403  67501.2324340381  59619.44010386508  51119.67553089799  42136.
↪07470518917  32858.958346466505  23530.907462228468  14438.639012299027  5901.
↪176173005475  -1744.9188715113523  -8162.274689776561  -13030.301653804609  -
↪16062.504186860033  -17022.421968510476  -15737.234283797181  -12108.
↪252338830673  -6117.760170266149  2168.0760186537373  12600.250037888227  24951.
↪68901024066  38925.33452407901  54164.95577995781  70268.02368678615  86799.
↪93822521572  103308.91876372647  119340.92903287584  134454.10229277017  148232.
↪24316183716  160297.0957899548  170319.17030208814  178027.00046329113  183214.
↪75913013506
175565.61826277897  176232.8902274804  174469.89963877012  170370.07936438342 ␣
↪164095.03215396532  155870.26816941577  145978.94875687765  134753.61486275162 ␣
↪122565.95503242598  109814.77237444522  96912.43515600712  84270.23094734045 ␣
↪72283.17580173933  61314.942800121484  51683.65357766441  43649.30924931222 ␣
↪37403.6139702355  33062.860432251924  30664.402692101197  30167.044507278377 ␣
↪31455.43307513013  34348.28574026462  38610.01125877914  43965.03963265248 ␣
↪50113.96705508098  56750.47445505119  63577.904540242016  70324.39232309765 ␣
↪76755.54019037682  82683.80560058661  87974.01527609644  92544.71582652145 ␣
↪96365.39381065145  99449.92215105688  101846.88775547486  103627.70207210763 ␣
↪104873.5709444737  105662.4872123466  106057.40096611076  106096.6181865501 ␣
↪105787.28679945081  105102.5655362228  103982.75718427272  102340.34991475187 ␣
↪100068.57653977699  97052.79956428488  93183.78479069745  88371.75813657741 ␣
↪82560.06275899961  75737.2523486435  67946.56922391565  59291.95243808005  49939.
↪984466551905  40117.49283620795  30104.849690498857  20225.33114510526  10831.
↪184076598634  2287.2788490497187  -5046.614194560279  -10833.808973180065  -
↪14777.840046864128  -16637.7298146018  -16240.52212295962  -13490.428410156674  -
↪8374.177012251253  -962.4133690184972  8592.75248015247  20063.733408564774 ␣
↪33156.491150293165  47521.977832179255  62769.30111814784  78479.90796335481 ␣
↪94222.1188268579  109565.42057564533  124094.02910445271  137419.3475784043 ␣
↪149191.05930538525  159106.6936400231  166919.58026891557  172445.15698815984
163268.34195962502  164739.90016286215  164092.6238741817  161412.56762356666 ␣
↪156843.98094528646  150584.68100951164  142879.76367160794  134013.68417017374 ␣
↪124300.80808147904  114074.62401254935  103675.91460963953  93440.29147169711 ␣
↪83685.60020882716  74699.78123437228  66729.8177884862  59972.4052713545  54566.
↪92884909015  50591.23760434187  48060.5563792809  46929.68910565774  47098.
↪4526211609  48420.05412929927  50711.90719543931  53768.18966815145  57373.
↪3001169291  61315.28223227186  65398.26955101648  69453.060548367  73345.
↪06489405638  76979.0573352092  80300.42209093308  83292.84875736863  85972.
↪72809867267  88380.7688499964  90571.59151380487  92602.23151382903  94520.
↪58596862631  96354.85580710926  98104.96505931861  99736.78640980484  101179.
↪77810613053  102328.35986981884  103047.04706950941  103179.04839505034  102557.
↪738560726  101020.16851525368  98421.59190723908  94649.8833520404  89638.
↪70982972177  83378.39233840407  75923.55428469504  67396.88270929741  57988.
↪60937227222  47951.62805350032  37592.47730931271  27258.709719257273  17323.
↪417005129573  8167.86709584064  163.32161455628378  -6346.865761954941  -11063.
↪817415576503  -13747.434112027086  -14227.67948830522  -12412.317593227754  -
↪8290.82036889455  -1934.4105873894443  6507.562316252472  16814.517413864233 ␣
↪28704.259460869187  41844.669368794566  55866.77513253651  70378.51840082774 ␣
↪84978.58505854683  99269.75515773434  112871.33552349336  125430.35371505553 ␣
↪136631.30212018202  146204.31539540697  153931.7360698619  159653.0685806793
```

(continues on next page)

(continued from previous page)

```
149356.5247242045  151548.60568565037  151970.499228503  150698.97934183967 ␣
↪147858.1948540819  143614.82190556993  138171.907610216  131761.4851271808 ␣
↪124636.10091112068  117059.47040187402  109296.56162817017  101603.48768668452 ␣
↪94217.65817541475  87348.6857925827  81170.55814619301  75815.55954262367  71370.
↪35980620784  67874.577834891  65321.98181122882  63664.31512881678  62817.
↪55004456969  62670.18517220543  63093.03463814122  63949.822034261924  65107.
↪805216095934  66447.62903979627  67871.63819338007  69309.98181452171  70724.
↪0003268224  72106.59209534343  73479.4975769278  74887.6925675984  76391.
↪3289289984  78055.87967625818  79941.31614156578  82091.25221213614  84523.
↪02350312157  87219.62297120123  90124.29074495047  93138.36329490188  96122.
↪73993642553  98903.04234172545  101278.24777105238  103032.29300706102  103947.
↪89670049031  103821.65355755626  102479.33064452949  99790.25437702781  95679.
↪72015010481  90138.48173274484  83228.57452530757  75084.97982256667  65912.
↪92621228921  55980.9263997031  45609.93961681363  35159.30917176254  25010.
↪33255318182  15548.463404483735  7145.211926698921  140.8004452096502  -5171.
↪452145119176  -8558.189967958839  -9855.892657210148  -8977.157383462407  -5913.
↪227475574269  -732.8387956031656  6422.334427336129  15345.179628582031  25773.
↪011519878874  37399.14968526158  49885.53352381999  62875.72650583413  76007.
↪72751711734  88926.10094606181  101293.04661463977  112798.1431804644  123166.
↪60307633303  132165.96465641778  139611.21242350107  145368.35703015383
134399.9865663716  137197.0682676927  138596.9952643075  138665.99101866473 ␣
↪137506.32251789322  135251.37449244587  132059.7791635014  128108.72556632996 ␣
↪123586.62273802664  118685.34974089032  113592.38571045214  108483.1665335055 ␣
↪103514.05259452744  98816.30613729486  94491.46074088904  90608.41541606175 ␣
↪87202.50142731862  84276.65435917288  81804.68404774848  79736.48089749257 ␣
↪78004.84134040566  76533.4516570432  75245.45199468787  74071.92378875421  72959.
↪61387182772  71877.2335214832  70819.75214709913  69810.23984298331  68898.
↪9922845462  68159.88262590079  67684.11190084885  67571.75349950773  67921.
↪68955340247  68820.69954831048  70332.56893443089  72488.12679318132  75277.
↪0908663267  78642.49547888464  82478.30917490441  86630.62599859439  90902.
↪55385094127  95062.64526491366  98856.4421285841  102020.45820495981  104297.
↪72176770898  105453.8617694951  105292.65621554159  103669.9764333722  100505.
↪15491326187  95788.97022024673  89587.66731706643  82042.69805179558  73366.
↪15399856927  63832.15035684372  53764.68372562398  43522.708998995906  33483.
↪34585824949  24024.222786367605  15505.991302022725  8255.996204930481  2553.
↪9756578398956  -1379.500636575642  -3391.3509298252325  -3400.7794559649415  -
↪1400.5019850075623  2545.4380799241626  8306.91491179369  15695.533714212725 ␣
↪24474.421752152135  34369.3584901296  45080.60641662938  56294.84612267249 ␣
↪67696.69415630188  78979.37916169222  89854.25918718893  100058.96968896793 ␣
↪109364.08817072451  117578.28057923193  124551.95254789272  130179.46458237029
119011.77226174931  122278.23312373782  124531.02925097824  125825.05432445598 ␣
↪126239.76281655612  125874.30482010072  124842.08272035641  123264.89117895211 ␣
↪121266.84058969653  118968.30503691075  116480.17249934943  113898.70060056534 ␣
↪111301.28885745461  108743.46253983778  106257.3203528799  103851.62728693736 ␣
↪101513.63730310093  99212.61332705404  96904.88245882755  94540.13277786446 ␣
↪92068.53640694737  89448.18364871695  86652.24616537322  83675.26238884557 ␣
↪80537.96155312374  77290.11594444126  74011.03181018491  70807.45106637973 ␣
↪67808.82763938498  65160.14980303629  63012.68685399308  61513.22767957976 ␣
↪60792.53362458472  60953.83409168161  62062.23952480752  64135.92639608093 ␣
↪67139.86138727426  70982.6815087655  75517.14314164767  80544.31027547675  85821.
↪38844053172  91072.84600879927  96004.21915170678  100317.79006780303  103729.
↪17657513586  105983.78703470698  106872.08482232482  106242.6723498419  104012.
↪3411733962  100172.43170756454  94791.08852325784  88011.26677479564  80044.
↪62173859365  71161.67631297636  61678.89164860126  51943.44784599158  42316.
↪66306342878  33157.03365952968  24803.863820379553  17562.3745168202  11691.
↪047450171056  7391.7826130924595  4803.243618110806  3997.549684101898  4980.
↪263570537558  7693.435683330696  12021.308175497543  17798.167660171748  24817.
↪765583997334  32843.701642809174  41620.18425739018  50882.63624410272  60367.
↪694286588  69822.24728368773  79011.26065174805  87724.23168180045  95780.
↪20741122228  103031.36594269525  109365.21228606012  114705.4709329078
```

(continues on next page)

(continued from previous page)

```
103819.18163338903  107412.23948217563  110371.14148025599  112738.99051377566 ␣
↪114572.18895467835  115935.76997921226  116898.5091152739  117528.00739652399 ␣
↪117885.9636688978  118023.87610229212  117979.42627527662  117773.79785774791 ␣
↪117410.16130724507  116873.51310876652  116131.99211230218  115139.70822370402 ␣
↪113841.01454031351  112176.03995267338  110087.18423687127  107526.17199544956 ␣
↪104461.1760667733  100883.46511049694  96813.01213899045  92302.52625633219 ␣
↪87439.44080693318  82345.5056531976  77173.78352403057  72103.03069582809  67329.
↪63797051451  63057.50413032274  59486.39494379855  56799.4909154942  55150.
↪93259464709  54654.222667461545  55372.332578532965  57310.28615183699  60410.
↪85680028082  64553.826608840165  69559.02749006147  75193.13290566606  81179.
↪91175676054  87213.41319957966  92973.3408146686  98141.7129395921  102419.
↪80564856657  105544.34374789677  107301.94585761378  107540.9392391583  106179.
↪83029353934  103211.93478331288  98705.9214810795  92802.28558875553  85706.
↪02470680415  77676.02203807086  69011.83288563593  60038.70889040001  51091.
↪77183444369  42500.26213445862  34572.73830279714  27583.999259468786  21764.
↪35196090326  17291.665782691496  14286.45751244315  12810.052053548206  12865.
↪678418142888  14402.200594408423  17320.057703292157  21478.903230214128  26706.
↪39095149529  32807.55374341164  39574.255850070185  46794.2620311096  54259.
↪54941299664  61773.58039050386  69157.34856325897  76254.09672739496  82932.
↪6805928025  89089.61071035439  94649.84705387423  99566.44704812719
89433.74466885455  93216.79607129436  96726.89611212938  99995.02618708859  103054.
↪79956254613  105938.12217615616  108670.97420095169  111269.52521999239  113736.
↪80883082573  116060.18655782928  118209.821586068  120138.35625735679  121781.
↪94108054342  123062.6967115006  123892.60572773876  124178.73221056064  123829.
↪56053315487  122762.13852339877  120909.61366731545  118228.67390445228  114706.
↪35588150029  110365.6707448022  105269.52569481489  99522.49049127765  93270.
↪07021006988  86695.29343594078  80012.59986795855  73459.20121855303  67284.
↪28041751792  61736.571601734315  57051.01251749389  53435.268613053086  51056.
↪983760121235  50032.609398254244  50418.59952510573  52205.63569533063  55316.
↪37110134809  59606.967233527525  64872.455369736716  70855.7053859308  77259.
↪5442777773  83761.35406495351  90029.30927329698  95739.30065990318  100591.
↪5427799627  104325.88203172287  106734.90743222713  107674.11201945189  107068.
↪54741940311  104915.64317418562  101284.10895387844  96309.08384328714  90183.
↪92502855574  83149.22275441073  75479.77673732562  67470.36258020552  59421.
↪150896562795  51623.617221345696  44347.70219087893  37830.85758395791  32269.
↪456252737382  27812.86621814022  24560.305401255086  22560.41721311525  21813.
↪3505109346  22274.999725335263  23862.968634039276  26463.76726230972  29940.
↪73531851798  34142.203983571635  38909.45488963938  44084.10321506193  49514.
↪61267723261  55061.735612399025  60602.75397426584  66034.47117269738  71274.
↪96636170689  76264.17032937029  80963.35581375344  85353.6568362874
76421.71576710802  80277.15751634924  84188.97412027842  88175.63851418164  92248.
↪53892182134  96408.09915902044  100640.33275272536  104914.05605173552  109178.
↪9849340698  113364.92570938566  117382.24007344429  121123.71449660699  124467.
↪8960134151  127283.87093239275  129437.36462626532  130797.93548577829  131246.
↪93219872715  130685.78969229419  129044.16458125015  126287.36443264694  122422.
↪51373380091  117502.92796606288  111630.23749995569  104953.91359781854  97667.
↪99456683811  90004.98263130308  82227.07012917394  74615.04387384234  67455.
↪39461105585  61026.31022089612  55583.343768854334  51345.61026623862  48483.
↪37206093201  47107.81929295742  47263.74048946793  48925.61534885381  51997.
↪4574419087  56316.50273592945  61660.59671701271  67758.89578372383  74305.
↪28463164961  80973.73608625442  87434.71599678668  93371.67227008379  98496.
↪64825800847  102564.12600789423  105382.32903641238  106821.3876513594  106817.
↪97925395533  105376.28589859509  102565.34489142447  98513.08877095551  93397.
↪56350265745  87435.96562227316  80872.24138305098  73964.03880097093  66969.
↪79626994787  60136.69273197342  53690.08152038884  47824.8931481559  42699.
↪33361905928  38431.03726437746  35095.66934107642  32727.82506180551  31323.
↪947606937967  30846.89448746456  31231.722805239584  32392.239821022602  34227.
↪87328747205  36630.451404374544  39490.53853491583  42703.04265341169  46171.
↪88648515511  49813.60972874393  53559.83916786678  57358.623098971475  61174.
↪67438612181  64988.60246209995  68795.24011505207  72601.18842247868
```

(continues on next page)

(continued from previous page)

```
65276.65753741801  69117.29015459334  73299.53306680414  77828.7390971185  82694.
↪7388996961  87868.54732761777  93299.7801684629  98915.00918074229  104617.
↪26924075525  110286.90017665517  115783.85559212827  120951.54151382175  125622.
↪16097293788  129623.44077893154  132786.50993549934  134954.59322468436  135992.
↪08735038433  135793.5098974949  134291.76187658438  131465.12999446504  127342.
↪47997464304  122006.15929727073  115592.23534928134  108287.83840352116  100325.
↪54999515138  91974.96520635998  83531.74909289855  75304.6888700524  67601.
↪40042453099  60713.4673309556  54901.86224022925  50383.51696253355  47319.
↪86547423858  45808.0845140344  45875.60491298157  47478.27294597293  50502.
↪31784568742  54770.04468346047  60048.937851984956  66063.64608711547  72510.
↪14067421535  79071.2069485001  85432.35458022049  91297.21929659782  96401.
↪57783919216  100525.20466299022  103500.95449930712  105220.64729150828  105637.
↪54656552583  104765.44314292727  102674.56737962487  99484.74021135109  95356.
↪3239668109  90479.63917694933  85063.5683006287  79324.0703151825  73473.
↪28436817031  67709.81257717307  62210.650850646474  57125.09345359719  52570.
↪78401256601  48631.93463895786  45359.59660846594  42773.749382002505  40866.
↪885973211545  39608.71519172835  38951.57566478333  38836.16065187073  39197.
↪18221394079  39968.65241547994  41088.52123042042  42502.478914216816  44166.
↪798622486014  46050.15799209  48134.432784884484  50414.49976670912  52897.
↪11965155125  55598.995488943074  58544.11961727661  61760.53604004959
56395.59106142256  60173.77947760928  64524.42977292738  69438.80389015912  74885.
↪76846061499  80809.20925858879  87126.38799447055  93727.46243453772  100476.
↪36349516833  107213.17577046952  113758.10047099384  119916.99368441901  125488.
↪37219077622  130271.66988860902  134076.41809495282  136731.92162455257  138096.
↪9190899798  138068.65928223176  136590.8034113057  133659.58082738693  129327.
↪68614058208  123705.50760279257  116959.41588118776  109307.01122422621  101009.
↪41484531295  92360.88426920492  83676.21830766641  75276.5810399371  67474.
↪50251348496  60558.89588125677  54780.958552175834  50341.79476312278  47382.
↪509239413645  45977.3811760689  46130.54365447964  47776.37830819419  50783.
↪60361402657  54962.80403830693  60076.93293100195  65854.13983443272  72002.
↪13563277142  78223.22614822308  84229.12165792963  89754.66708242573  94569.
↪73144944839  98488.63764965783  101376.69298025481  103153.5835295901  103793.
↪60577261305  103322.91142601459  101814.1224007013  99378.81957456928  96158.
↪51319033498  92314.75875860525  88019.08916269675  83443.3938266341  78751.
↪29537730913  74090.96202419802  69589.66058723422  65350.2122878736  61449.
↪37249033127  57938.026848716036  54842.988197613675  52170.09697162267  49908.
↪276135939894  48034.1699374444  46517.0020599549  45323.31869905934  44421.
↪32989381026  43784.62164107532  43395.07533001197  43244.89397953814  43337.
↪691948807056  43688.65308727334  44323.80027554487  45278.44720471345  46594.
↪92265532806  48319.67103112932  50499.8434743718  53179.50441048077
50060.0101409905  53773.90614948126  58228.83302178492  63400.545073310124  69237.
↪32246344164  75658.20940643427  82552.37527061158  89779.80310466522  97173.
↪47024869692  104543.12420673041  111680.67500295218  118367.12634878712  124380.
↪85796338359  129506.95813819586  133547.19832393256  136330.14995965175  137720.
↪87748171523  137629.6087371527  136018.79105938  132907.99136527328  128376.
↪19174687077  122561.16449288862  115655.77509181514  107901.24822840451  99577.
↪62737003886  90991.8490801618  82464.0242020953  74312.65591532184  66839.
↪61781046177  60315.75506601447  54967.95391729581  50968.44862087215  48427.
↪00525631146  47386.4464279951  47821.772316837945  49642.906373304126  52700.
↪86457270982  56796.932152868656  61694.24662676564  67131.04379724732  72834.
↪73432305008  78535.94779707206  83981.71045817311  88947.00814870125  93244.
↪12029945396  96729.28237700877  99306.4296008579  100927.97866344871  101592.
↪80135561555  101341.72019840131  100250.99940159926  98424.40579870286  95984.
↪46863346736  93063.57294177747  89795.48140705808  86307.80005406849  82715.
↪79283995648  79117.81982648683  75592.53474601576  72197.84178789466  68971.
↪48855815094  65933.07067124051  63087.14905146589  60427.13664672582  57939.
↪59692313241  55608.60964473476  53419.89535816299  51364.44254717469  49441.
↪44374127558  47660.41216379606  46042.4127615147  44620.395973379615  43438.
↪66637602281  42551.55024731402  42021.346748528784  41915.65895706023  42304.
↪206526314876  43255.22490530555  44831.560264212974  47086.57726706667
```

(continues on next page)

(continued from previous page)

```
46422.800952794714  50119.10892904103  54657.596187862  59996.61779514892  66063.
↪93455443071  72755.88166378147  79937.78406849041  87445.79763507367  95090.
↪30385214242  102660.91199864684  109933.02934741891  116675.85227154945  122661.
↪51661971984  127675.03363963681  131524.53818394901  134051.29904457374  135138.
↪8961407706  134720.9630869122  132786.93075689982  129385.28881959159  124624.
↪00510314855  118667.90060553816  111732.96119211856  104077.7630090632  95992.
↪38314977274  87785.34529392663  79769.29768397087  72246.22556031519  65493.
↪052654914136  59748.48099168582  55201.85391451989  51984.707418130354  50165.
↪50736836933  49747.86678799141  50672.31271441575  52821.44264900593  56028.
↪09320798722  60085.95423274364  64761.914088931415  69809.32662375303  74981.
↪3535518248  80043.55956911675  84785.01820968022  89027.31692546543  92631.
↪0189358498  95499.33317967626  97578.9467918052  98858.17126585281  99362.
↪72925664214  99149.65144278132  98299.85280000947  96910.00962715148  95084.
↪36142048055  92927.01794432497  90535.26794645275  87994.271164706  85373.
↪3807720963  82724.20130224104  80080.34926357011  77458.76072996011  74862.
↪29073379596  72283.27911958791  69707.71943512501  67119.66116981971  64505.
↪498127288534  61857.84159913933  59178.739386678295  56482.07295496741  53795.
↪037559818295  51158.677399477056  48627.5045498722  46268.27341326934  44158.
↪01052002906  42381.41368193405  41027.73734122521  40187.27635857513  39947.
↪55290158558  40389.30491527851  41582.37352046617  43581.59294748869
45501.78980023353  49274.77167062618  53921.52150823345  59380.664058956165  65559.
↪15951232673  72332.4968674591  79546.19749296416  87018.77451391667  94546.
↪23200399101  101908.10420471602  108874.93348312113  115216.97337906035  120713.
↪78884718189  125164.31993308157  128396.88827049546  130278.56790523024  130723.
↪32129272494  129698.32349469574  127227.96487143032  123395.1333851723  118339.
↪52655008905  112252.92099709844  105371.52244772973  97965.71637586012  90327.
↪72457744664  82757.83037444201  75549.95196396696  68977.40900191902  63279.
↪73517864231  58651.337058975434  55232.68933715001  53104.596071913  52285.
↪84773776838  52734.379642804415  54351.80511166995  56990.97424395426  60466.
↪01277018926  64564.1402015798  69058.46348433987  73720.89889392836  78334.
↪39333093204  82703.69396851989  86664.04514341679  90087.36226315898  92885.
↪63009371876  95011.48108851629  96456.11200800046  97244.87850989428  97431.
↪05444327708  97088.34530128786  96302.7977194782  95164.74744802907  93761.
↪3996000889  92170.54382611749  90455.78320369577  88663.51106700518  86821.
↪71777033065  84940.56237023903  83014.51402503668  81025.76391324674  78948.
↪53708801378  76753.89799575173  74414.64307856961  71909.90549822651  69229.
↪15452647657  66375.34769106624  63367.07858966863  60239.64866436876  57045.
↪06936778658  53851.06590822566  50739.20117175101  47802.26694977016  45141.
↪10011778014  42860.976882563584  41067.72318493894  39863.659243095426  39343.
↪47659292377  39590.13172415551  40670.835106264996  42633.21981521645
47180.27847072025  51166.933346651094  55990.341038277315  61566.72692957971 ␣
↪67781.64419788815  74491.26953835753  81525.01063840937  88689.53068156808 ␣
↪95774.22484281268  102558.09250002283  108817.8426397333  114336.95704994077 ␣
↪118915.32649779928  122378.98014659651  124589.3586947531  125451.54648896876 ␣
↪124920.88415469829  123007.43483509614  119777.87384103735  115354.50922679904 ␣
↪109911.31142382702  103667.02208026734  96875.61160587915  89814.54613261622 ␣
↪82771.49208566951  76030.2159943869  69856.51679504693  64485.04959793345  60107.
↪860105857006  56865.34889952618  54840.23090107509  54054.858257326545  54472.
↪04910336608  55999.32715712771  58496.24597290949  61784.26468831119  65658.
↪47511673649  69900.36575337332  74290.75510631976  78622.038212698  82708.
↪9645068851  86397.29558972402  89569.86669501255  92149.78112092923  94100.
↪68579085272  95424.29092069504  96155.49076266369  96355.60099812446  96104.
↪34033657177  95491.24221652442  94607.18484359214  93536.67651046478  92351.
↪4349491574  91105.66470939784  89833.2781587748  88547.1379265969  87240.
↪23576887784  85888.57799753874  84455.431550813  82896.50507822791  81165.
↪59999122175  79220.26738677619  77027.044582425  74565.91315309689  71833.
↪71004742963  68846.32455738666  65639.61641696971  62269.084757631295  58808.
↪39642517698  55346.94003138993  51986.60660340295  48838.00915930685  46016.
↪344908540916  43637.080031113524  41811.60443218024  40642.969265754655  40221.
↪789727055235  40622.374759192106  41899.137171869115  44083.343215269335
```

(continues on next page)

(continued from previous page)

```
51214.54021961018  55586.14390024056  60692.88099596645  66425.73484566653  72648.
↪00452991332  79197.75922440682  85891.54433358299  92529.39899524032  98901.
↪16736641698  104793.99003420629  110000.75433207038  114329.17292080517  117611.
↪05971780766  119711.29241935108  120535.90186506286  120038.71872143474  118226.
↪04276637752  115158.88102278393  110952.425518947  105772.60275229908  99829.
↪71439520363  93369.38876458566  86661.25949465192  79985.96549615254  73621.
↪20925915526  67827.70568095683  62835.891171182564  58834.237547121535  55959.
↪92698257833  54292.49803159495  53850.87845688693  54593.992378699586  56424.
↪88431069374  59198.0598021247  62729.520569524255  66808.78855792657  71212.
↪0826929891  75715.74430285093  80109.00732751397  84205.27698794562  87851.
↪20957512032  90933.06551343716  93380.02293732736  95164.37208628011  96298.
↪74309048464  96830.73283583128  96835.47418801469  96406.81987120907  95647.
↪88504878507  94661.7034487419  93542.70319952541  92369.6067772692  91200.
↪21519452099  90068.36350514727  88983.14841837372  87930.34527286151  86875.
↪76577604606  85770.17243554902  84555.26986177429  83170.2424806267  81558.
↪30376866623  79672.7608701232  77482.17369873103  74974.2898823875  72158.
↪55494280098  69067.11911437634  65754.37700430435  62295.1743551061  58781.
↪890475640364  55320.6514912768  52026.94771414488  49020.92053606746  46422.
↪55531152886  44346.97362411821  42899.968985912885  42173.88225343392  42243.
↪87372873062  43164.623131367036  44967.47907957197  47658.08643453657
57247.86240003509  62198.296416726735  67724.48641579054  73689.37645894213  79933.
↪06472462256  86276.44627953574  92525.98864288883  98479.65050560818  103933.
↪87189885869  108691.4659272361  112570.13663215163  115411.24531106376  117088.
↪36026021003  117515.06369327333  116651.46423302918  114508.88091404364  111152.
↪2285088554  106699.74343545664  101319.83935134225  95225.06243367678  88663.
↪31538627011  81906.72097222798  75238.68368860317  68939.86566322696  63273.
↪90530392869  58473.76315951755  54729.57166412925  52178.79184667847  50899.
↪344079475886  50906.19024176267  52151.614744838655  54529.19885056632  57881.
↪22604745012  62009.01602480645  66685.480057407  71669.03774845011  76717.
↪94634171392  81604.07629118444  86125.22175155142  90115.15748286585  93450.
↪83417271415  96056.32753918113  97903.40458097393  99008.8229714214  99428.
↪71673922362  99250.62454031802  98583.87065934515  97549.10245134367  96267.
↪8155635595  94852.65990653659  93399.22040853568  91979.81740353492  90639.
↪68623689434  89395.69099575977  88237.52082832178  87131.12615065  86023.
↪99112200832  84851.71975887263  83545.34340965  82038.73980009703  80275.
↪58654722798  78215.34868945827  75837.91064799581  73146.5959577855  70169.
↪45978078009  66958.87651715372  63589.56606760474  60155.298256076276  56764.
↪57961117926  53535.65776197225  50591.17738931502  48052.792498329174  46035.
↪98980317037  44645.315915872794  43970.13601377541  44080.992371419554  45026.
↪5846745427  46831.36513586796  49493.73201547139  52984.81412694109
64830.357221307335  70561.88051594535  76661.38420054023  82961.28251250004  89277.
↪61441612258  95414.87297147492  101171.79405520677  106348.06403771865  110751.
↪82116319172  114207.7277885659  116565.29024063393  117707.01132001125  117555.
↪88924297664  116081.73695314399  113305.79594173832  109303.16438473646  104202.
↪65183965888  98183.8087286636  91471.05064821008  84324.99343217505  77031.
↪31994996005  69887.69702897297  63189.43340533515  57214.70113186115  52210.
↪2200003586  48378.317889854065  45866.22520573452  44758.340032761436  45072.
↪01931382781  46757.222602179965  49700.07540251296  53730.148533950574  58630.
↪98943071018  64153.21156485985  70029.26776900142  75988.91689237542  81774.
↪35057305805  87153.9816082402  91934.00478506932  95967.01630266782  99157.
↪2048908678  101461.88805326418  102889.43956468627  103493.917656344  103366.
↪93648788198  102627.50858842226  101410.70928342218  99856.06742402291  98096.
↪56770299682  96249.06224408493  94406.74236368874  92634.12946247571  90964.
↪8242735983  89402.02526597313  87921.6089830712  86477.37494597517  85007.
↪9097979739  83444.42968468435  81718.92153087792  79771.92274859962  77559.
↪34986735921  75057.90021125044  72268.69444219292  69218.98703933386  65961.
↪93165140616  62574.53479252563  59154.052864638936  55813.17554655316  52674.
↪388633340364  49863.920971418156  47505.656617396955  45715.34142785856  44595.
↪34199368734  44230.13442971092  44682.62130414675  45991.30608836305  48168.
↪30293403372  51198.12949108272  55037.22277621301  59614.13046403554
```

(continues on next page)

(continued from previous page)

```
73443.44175532447  80150.74569447925  86981.27797234863  93735.02605083783  100203.
↪42202443839  106175.32036263568  111443.71361618905  115813.09556584415  119107.
↪29580315275  121177.51533771669  121910.20043633558  121234.31351347877  119127.
↪50727762024  115620.69168388975  110800.50991301064  104809.3131156872  97842.
↪34306223775  90141.99143251359  81989.19411353012  73692.22462612725  65573.
↪35612582923  57954.04856451547  51139.46922266805  45403.25565508348  40973.
↪46844774828  38020.650196159055  36648.805581898036  36889.95014097875  38702.
↪65287947574  41974.73595633908  46530.012722365165  52138.665231232386  58530.
↪60609469911  65410.95792177184  72476.63394485207  79432.92833513545  86009.
↪03070424512  91971.46633402404  97134.6250734244  101367.76477420257  104598.
↪14174411201  106810.20958702007  108041.11587907522  108372.99056881995  107922.
↪7401594234  106830.22094013808  105245.7513970472  103317.93326817575  101182.
↪68387538288  98954.24702810233  96718.75912776001  94530.71853397108  92412.
↪45985211337  90356.4914650978  88330.33424958037  86283.31916678947  84154.
↪67435571007  81882.1664143065  79410.55803846353  76699.20180076387  73728.
↪1992266805  70502.70301376774  67055.11302874364  63445.09716624674  59757.
↪54017326436  56098.67288404151  52590.74980241774  49365.71741483694  46558.
↪34625428247  44299.28823463549  42708.472799046714  41889.17979898891  41923.
↪03461018542  42866.073425706265  44745.93502513625  47560.15893686724  51275.
↪515417708724  55828.26336677394  61125.22789974186  67045.60617956644
82527.63117736802  90381.15761487126  98089.1201492297  105418.06936357467  112134.
↪83307075538  118013.56373599192  122843.26278391278  126435.64401109173  128633.
↪11436388918  129316.5615781408  128412.55628348957  125899.51338957086  121812.
↪3253015608  116244.98696202066  109350.78568208043  101339.72870423524  92473.
↪02514187577  83054.61828302663  73419.96683467049  63922.48376675002  54918.
↪24121231192  46749.72118515615  39729.51776810801  34124.96293573503  30144.
↪645744252266  27927.71919039809  27536.742675393805  28954.59897129824  32085.
↪76683674426  36761.94259494166  42751.707874794985  49773.65949571951  57512.
↪17370195723  65634.79059392287  73810.0912771178  81724.90979598247  89099.
↪77748580003  95701.63506281277  101353.05691207477  105937.49605183862  109400.
↪35606664096  111746.00409862067  113031.13226230818  113355.1304873917  112848.
↪33203027691  111659.11896370983  109940.92029396689  107840.09825903071  105485.
↪60401215225  102981.10380146354  100400.0476364977  97783.89466118245  95143.
↪44521292805  92462.98086090252  89706.7005352401  86826.779386572  83772.
↪2784102656  80498.10243790297  76973.24103894769  73187.62458911404  69157.
↪07467888691  64926.008870771  60567.757119610586  56182.54303216956  51893.
↪360851070865  47840.12459113982  44172.56898690849  41042.43729693457  38595.
↪49790262876  36963.89403956077  36259.25688508854  36566.91249918676  37941.
↪40029937615  40403.40757936165  43938.12262428492  48494.92744704933  53988.
↪29603695528  60299.73749298107  67280.62391397129  74755.76539326916
91512.11053129708  100641.69310803834  109346.71595024534  117360.44809079947 ␣
↪124425.90421372658  130303.83964933179  134780.9312254212  137677.96403524728 ␣
↪138857.7638449035  138232.53393034445  135770.18578412296  131499.20755901138 ␣
↪125511.60308030913  117963.46572645895  109072.82949945307  99114.56356648786 ␣
↪88412.24087795445  77327.10583677379  66244.47606939799  55558.12193311566 ␣
↪45653.35595054996  36889.71473491403  29584.2121505604  23996.172219002998 ␣
↪20314.606536028194  18648.982656518434  19024.042395123688  21379.083860456536 ␣
↪25571.83528968293  31386.743493458252  38547.198460113956  46730.9423053151 ␣
↪55587.68752049029  64757.814963623976  73890.94952474705  82663.22760688847 ␣
↪90792.17498551693  98048.29847005355  104262.74564244617  109330.68386023675 ␣
↪113210.36895050923  115918.19006459932  117520.26487850831  118121.39610724327 ␣
↪117852.3679918916  116856.64781183244  115277.55730495672  113246.89418225076 ␣
↪110875.8237568358  108248.64025468274  105419.73652187175  102413.84236864281 ␣
↪99229.31932422529  95844.05553086352  92223.30815633034  88328.70659731876 ␣
↪84127.56656768927  79601.67516992221  74754.78608229104  69618.20210299439 ␣
↪64254.004832963255  58755.700427954835  53246.26685146154  47873.79304006224 ␣
↪42805.07707453918  38217.685597997886  34291.06169485464  31197.299761670347 ␣
↪29092.185287097982  28107.03137595388  28341.74258635865  29859.413049310988 ␣
↪32682.63386198658  36791.55793973239  42123.660621070805  48575.0501314112 ␣
↪56003.12856626988  64230.38230141233  73049.08764513148  82226.74666700077
```

(continues on next page)

(continued from previous page)

```
99844.46046391153  110324.36490346103  120104.59780758459  128886.71000612521 ␣
↪136391.70020457244  142368.79057585535  146604.07813517767  148928.8470283306 ␣
↪149227.2521221253  147443.0130834781  143584.70288268238  137729.18722200464 ␣
↪130022.78162221852  120679.74731819797  109977.84790635633  98250.8331184237 ␣
↪85877.89643039329  73270.35715181872  60856.02925555079  49061.94031677199 ␣
↪38296.23574652299  28930.228442426567  21281.617420234066  15599.890845117916 ␣
↪12054.844782727567  10728.99098439487  11614.403689115163  14614.281390866718 ␣
↪19549.19438300342  26167.675667300486  34160.51601677211  43177.86751283675 ␣
↪52848.064930843364  62796.957496991614  72666.51520864296  82131.53736477383 ␣
↪90913.44191267113  98790.3411277879  105602.8941016042  111255.74675030517 ␣
↪115714.69987125209  119000.05871595026  121176.88868577013  122343.10980343049 ␣
↪122616.4917312797  122121.65222515978  120978.11344006873  119290.33838406295 ␣
↪117140.4669141255  114584.21555259207  111650.12064961593  108342.01457382023 ␣
↪104644.3540296478  100529.79031487987  95968.2012088247  90936.30545821943 ␣
↪85426.95920311533  79457.28801831821  73074.93088068627  66361.85004078565 ␣
↪59435.375987769454  52446.38946415963  45574.772235607736  39022.46577008419 ␣
↪33004.6458485361  27739.639482946346  23438.27166641556  20293.332326318567 ␣
↪18469.802694395432  18096.38412686008  19258.743721969167  21994.744505836665 ␣
↪26291.77860858952  32086.1837856684  39264.608449555875  47667.10613072237 ␣
↪57091.69086353872  67300.06991299197  78024.28498900485  88974.02980240216
107018.91637630668  118855.30387118968  129734.4798273158  139329.43459805893 ␣
↪147342.13138646522  153512.85094474675  157629.0946292839  159533.80519164988 ␣
↪159132.59733371047  156399.63013395766  151381.71332513925  144200.2303286147 ␣
↪135050.49163158663  124198.20744793225  111972.88890063945  98758.14742569224 ␣
↪84979.05302719443  71086.9194565317  57542.09127189995  44795.49539083634  33269.
↪86967650545  23341.676796504165  15324.740279646125  9456.593747617124  5888.
↪412248458975  4679.201612295976  5794.669386582318  9110.906476199001  14422.
↪69376481355  21455.936974191616  29883.450607405815  39343.08116757462  49457.
↪00038873023  59850.924912860864  70172.03683238568  80104.48936543064  89381.
↪57557743869  97793.90040026326  105193.20631001216  111491.83593107507  116658.
↪14362752657  120708.46629021733  123696.5071480634  125701.15634664355  126813.
↪85540036937  127126.603991473  126721.60932708337  125663.40027309259  123993.
↪98724672655  121731.36592757865  118871.36292448311  115392.53022777644  111263.
↪53688209965  106452.30172648531  100935.97600425774  94710.82847696013  87801.
↪11065827217  80266.0811305559  72204.53440816939  63756.395031316366  55101.
↪181487336646  46453.39546692894  38055.128512030096  30166.381188932486  23053.
↪744296881603  16978.18716767858  12182.730479754953  8880.751753737168  7245.
↪58781333045  7401.97146783994  9419.68411263961  13309.637988537304  19022.
↪437467871285  26449.32216881262  35425.27706788511  45734.013153065855  57114.
↪47905173332  69268.5574553137  81869.62398868073  94571.69128054753
112601.63759691227  125723.35841863416  137660.56823654976  148062.56143058982 ␣
↪156616.5492601808  163057.32007040147  167176.14513764018  168828.67464751948 ␣
↪167941.50989798852  164517.09034446493  158636.5095758093  150459.88311770122 ␣
↪140223.94045843818  128236.6070903501  114868.47802240677  100541.25546907344 ␣
↪85713.41888074674  70863.59994092016  56472.33088237204  43003.0028597909  30882.
↪994489839883  20485.994113892608  12116.532311907853  5997.6588014625595  2262.
↪541618013929  950.5445270069176  2008.0652823442215  5294.112086593144  10590.
↪281496413154  17614.502791964325  26037.65575543805  35501.97250731998  45640.
↪01654054437  56093.00416352083  66527.29870719516  76648.0618241223  86209.
↪2772695206  95019.65238795901  102944.2277529283  109901.85955291976  115859.
↪05510276831  120820.91353635667  124820.12932493543  127905.13944735476  130128.
↪5265608284  131536.72896537877  132161.96001606304  132017.01869748824  131093.
↪39914254996  129362.80399665861  126781.86083743512  123299.55828270242  118866.
↪68241293453  113446.36368382678  107024.75255463828  99620.83450868304  91294.
↪47049739791  82151.8985239655  72348.14139055932  62086.01520442131  51611.
↪700915619105  41207.10405068508  31179.464347565852  21848.868831396474  13534.
↪455367092278  6540.16100811182  1140.8688642935304  -2430.256869359866  -3992.
↪57590463019  -3426.402728234302  -678.7302292690438  4233.900017260785  11225.
↪35041167697  20142.480596051566  30770.081954197336  42837.77767177518  56028.
↪478096546765  69987.98462946729  84335.3693863719  98673.81202256744
```

(continues on next page)

(continued from previous page)

```
116251.64719348573  130505.09553425734  143388.06731052778  154532.7566674561 ␣
↪163617.25729319412  170375.24700122824  174604.62196047668  176174.8223766609 ␣
↪175032.54327168502  171205.48972806046  164803.82696111896  156019.001125902 ␣
↪145119.67270056147  132444.61205939975  118392.5529928983  103409.1760876726 ␣
↪87971.58733588136  72570.85208946973  57693.32264927565  43801.64139401324 ␣
↪31316.39411405046  20599.41725633209  11939.720156659707  5542.867051509034 ␣
↪1524.4782354249473  -91.733915288336  628.7364372240263  3538.3800768009246 ␣
↪8418.876603964061  14995.558233100326  22954.156989961717  31958.704747694064 ␣
↪41669.38547850605  51759.16017435357  61928.09670173152  71914.53132041454 ␣
↪81502.45052457541  90524.79018171724  98862.67845261292  106440.97250634288 ␣
↪113220.729809399  119189.48868841102  124350.39067967949  128711.24612346612 ␣
↪132274.6193874447  135029.89391160687  136948.08062746088  137979.87369406887 ␣
↪138057.1576288824  137097.85572859336  135013.708039742  131720.30349405293 ␣
↪127148.48699188197  121256.13455106693  114039.24768414398  105541.36395643305 ␣
↪95860.40870385757  85152.310755716  73630.95468024649  61564.32160941087  49266.
↪95629641588  37089.166194576974  25403.58785283065  14589.929779928047  5018.
↪8076137120515  -2964.378276206422  -9054.609450809337  -13000.647625469202  -
↪14616.003673234372  -13787.003739876944  -10477.683003425656  -4731.440640478257␣
↪ 3330.4252301011074  13513.029993568482  25554.67649219864  39135.63631488766 ␣
↪53888.460262081535  69409.35485178369  85270.20678731604  101030.90094110397
117736.36244466928  132884.89819945183  146527.3774728683  158287.14852185053 ␣
↪167840.13800463607  174924.2380597784  179347.37529490655  180994.01392253413 ␣
↪179829.80835048173  175904.0997788141  169349.95726683192  160381.50425145996 ␣
↪149288.35082837736  136427.06992995  122209.80669443411  107090.28504864364 ␣
↪91547.66017838482  76068.84382496722  61130.083791230936  47178.69286667476 ␣
↪34615.88119096386  23781.639487205248  14942.543531619376  8283.203723453174 ␣
↪3901.8750846496846  1810.4858660191824  1939.0556294791168  4144.178253349193 ␣
↪8220.965514659401  13917.606095949435  20951.513887945708  29025.93468864732 ␣
↪37845.86214152386  47132.18516904032  56633.14579637038  66132.41629991024 ␣
↪75453.38975362976  84459.59500248474  93051.46957790815  101160.02505814962 ␣
↪108738.19345917231  115750.82887249866  122164.43996852307  127937.73756147356 ␣
↪133013.99654618045  137316.06055527553  140744.57542128355  143179.7448204988 ␣
↪144486.58343405332  144523.32673928302  143152.36896173522  140252.8659318344 ␣
↪135733.97701913494  129547.64300857257  121699.81036440922  112259.11463009623 ␣
↪101362.21709448656  89215.23318396992  76090.9772336613  62322.05226714481 ␣
↪48290.11026719504  34411.87461280585  21122.731736413465  8858.848496631668  -
↪1961.1536825144576  -10953.93941662567  -17787.661506088087  -22195.80910415316 ␣
↪-23988.071452247867  -23057.801041924264  -19385.884252726508  -13041.
↪042818116359  -4176.779472673457  6974.666814814977  20110.88441762141  34869.
↪84218268926  50841.89135970454  67582.70332213593  84626.68677603992  101500.
↪50146169537
116940.9549048762  132669.1368693994  146812.72396952682  158995.9582681559 ␣
↪168900.74368738013  176275.39182057354  180941.82080305993  182800.98349683138 ␣
↪181836.27619590695  178114.67173338856  171785.34059290861  163075.57681117163 ␣
↪152283.93459283235  139770.60460901307  125945.209260603  111252.36290554574 ␣
↪96155.51204500654  81119.72565209283  66594.23077570878  52995.56820531713 ␣
↪40692.26533022823  29991.880843608615  21131.166634421494  14269.920163268487 ␣
↪9488.87595092447  6791.723043564125  6111.0561919371685  7317.794298518261 ␣
↪10233.353369662742  14643.664216956488  20313.99546103028  27003.492745198746 ␣
↪34478.38165313052  42522.90352510073  50947.2516556811  59592.03487633312  68329.
↪09530479579  77058.82223162224  85704.40811923362  94203.75949073669  102499.
↪98193987993  110531.48640269513  118222.80159767426  125477.12124354346  132171.
↪46845804417  138155.1355451811  143251.77386357618  147265.1894505215  149988.
↪57244767464  151216.5797665032  150759.42681687704  148457.94757253444  144198.
↪4690503606  137926.32558165543  129656.91133985136  119483.32987540084  107579.
↪93311862061  94201.32978343155  79676.7605950706  64400.05957606318  48815.
↪721364457524  33401.851295320914  18650.969221077357  5049.757086033715  -6941.
↪12186120209  -16906.046623610426  -24491.33402830799  -29419.209732854797
↪31498.274183278976  -30630.121287974067  -26812.007707080575  -20135.
↪693836751598  -10782.768015846115  983.0882264064859  14826.340211895149  30351.
↪74726018925  47117.821404488976  64651.04352301427  82460.35138319898  100051.
↪49700218608
```

(continues on next page)

(continued from previous page)

```
113871.13813376831  129793.73159572999  144114.26663016557  156468.83275015219 ␣
↪166554.4390934493  174136.76402491776  179056.17719080855  181231.84784758385 ␣
↪180663.74043068563  177432.30393300572  171695.6936402783  163684.42685718124 ␣
↪153693.46918558434  142071.87103247823  129210.21741190474  115526.30619926943 ␣
↪101449.61665414006  87405.25593436213  73798.16160632411  60998.38002986122 ␣
↪49328.22478118807  39052.04160950595  30369.168031692534  23410.483620845975 ␣
↪18238.71390837083  14852.393772345167  13193.13561857243  13155.605415936969 ␣
↪14599.40703659071  17361.931152231453  21271.15355533224  26157.37774920828 ␣
↪31863.009667676615  38249.6228875754  45201.80816295327  52627.58297210418 ␣
↪60455.44191760423  68628.43154975158  77095.90742119076  85803.8527854043  94684.
↪78728905853  103648.35618308044  112573.6591651798  121304.25410977368  129646.
↪56374405704  137372.13914392804  144223.91499052156  149926.25408463262  154198.
↪25064715312  156769.47076232074  157397.07825816728  155883.14458089852  152090.
↪88429913681  145958.5985078064  137510.24322933835  126861.75784041238  114222.
↪57180283357  99892.03368342946  84250.8486941063  67747.94306089813  50883.
↪47022143958  34188.91352484419  18205.406457715042  3461.4746694308415  -9548.
↪59842573457  -20384.664502984757  -28677.897980606707  -34144.52365601261  -
↪36595.573414374914  -35942.41901497819  -32198.079196026683  -25474.525417243018␣
↪ -15976.396753945082  -3991.670894857205  10120.07955886668  25941.18723321346 ␣
↪43010.50648933185  60838.56182352156  78922.72426152992  96762.00397446228
108649.35903804001  124324.79608089461  138443.10553396947  150664.03101579635 ␣
↪160709.50649248168  168370.05572399867  173509.3395526735  176066.71542725235 ␣
↪176057.67285611798  173572.0261382066  168769.78801380759  161874.717748079 ␣
↪153165.63358870358  142965.69739767787  131630.00964492816  119531.98399256449 ␣
↪107049.08872666015  94548.6333026616  82374.32948442982  70834.35794829993 ␣
↪60191.61682069814  50656.71744995521  42384.1292813772  35471.66991241509  29963.
↪302930355465  25854.96329230111  23102.898187703555  21633.811159637597  21355.
↪94768245607  22170.176765305485  23980.115674208886  26700.417405773245  30262.
↪49001522603  34617.13326924131  39733.84504086105  45596.84603945904  52198.
↪17224676256  59528.46401019116  67566.31441530038  76267.2059131669  85553.
↪14708145341  95304.11151990766  105352.27664953875  115479.86806101544  125421.
↪14910535619  134868.77619486678  143484.39314033953  150912.99081809662  156800.
↪23985751095  160811.74006651173  162652.94281571664  162088.4072174658  158959.
↪05577658617  153196.20018771675  144831.30491850805  134000.72959326842  120945.
↪0194447329  106002.67060850421  89598.65629089362  72228.33379530723  54437.
↪63669426953  36800.67139436235  19895.96943759376  4282.689685391728  -9521.
↪981020212028  -21063.106102084173  -29965.0262577518  -35944.445603796004  -
↪38819.154719303464  -38512.2407546367  -35051.89155063403  -28567.124228877714  -
↪19279.94569543461  -7494.574941694432  6415.577475281593  22022.465853185844 ␣
↪38858.829546439774  56433.87486501269  74248.7087201548  91811.114715399
101504.74302399649  116452.4434998528  129948.99265648477  141690.01177449804 ␣
↪151432.5082996763  158999.58534947032  164283.2363299433  167245.1515831254 ␣
↪167915.47507197177  166389.47759566462  162822.16338541236  157420.90022245655 ␣
↪150436.25672308705  142151.33772889996  132870.02025088808  122904.59637313371 ␣
↪112563.41322772068  102139.15143008444  91898.39203279375  82073.08127072362 ␣
↪72854.40967438444  64389.48009050955  56780.95550777649  50089.66485340244 ␣
↪44339.91948049523  39527.073792566094  35626.670063559635  32604.35869633479 ␣
↪30425.696735280904  29064.91069682963  28511.77016552969  28775.854955898816 ␣
↪29887.702957401656  31896.58364318298  34864.93383415695  38859.79405776862 ␣
↪43941.87027123234  50153.09173444155  57503.71890771858  65960.15755057977 ␣
↪75434.64518785657  85777.88977746945  96775.56178616878  108149.2818468238 ␣
↪119562.42562200432  130630.70982657751  140937.15647147584  150050.68537925533 ␣
↪157547.2857386453  163032.49016228275  166163.73799763457  166671.17991540235 ␣
↪164375.54630850116  159201.8726269404  151188.13237755926  140488.15328847175 ␣
↪127368.55917488188  112199.8615106361  95442.19218025114  77626.49568534592 ␣
↪59332.262435747405  41163.067561553835  23721.271723349597  7583.238950152532  -
↪6723.663701824175  -18741.187489463075  -28096.575275130075  -34514.5857724527  -
↪37824.85330785428  -37964.548555421934  -34976.5618770577  -29003.64602963948 ␣
↪20279.117669265484  -9114.821430396332  4112.893830548026  18978.444794978903 ␣
↪35022.83972982027  51769.455432078976  68739.26263469676  85465.10558749954
```

(continues on next page)

(continued from previous page)

```
92757.4918955902  106478.2136410141  118910.96281063189  129799.38697140744 ␣
↪138945.62773346255  146213.0282867761  151526.92901026903  154873.5510354147 ␣
↪156296.98983316316  155894.3786951644  153809.33791616146  150223.89870338037 ␣
↪145349.17700284068  139415.16404379386  132660.0875480128  125319.86899008014 ␣
↪117618.24677618613  109758.14287013911  101914.81409594466  94231.24577796969 ␣
↪86816.11582137676  79744.48795556951  73061.19451349135  66786.65661794877 ␣
↪60924.68060029328  55471.58274600745  50425.84839839882  45797.44208945402 ␣
↪41615.86473442419  37936.108657638964  34841.791606714265  32444.95052925193 ␣
↪30882.23179804376  30307.508186473347  30881.26136532123  32757.366753112598 ␣
↪36068.17966434409  40909.0242799983  47323.310517987586  55289.53483334066 ␣
↪64711.3530334779  75411.74808701295  87132.06346345978  99536.35004756681 ␣
↪112221.10556025246  124730.09742215138  136573.58289920882  147250.90355483437 ␣
↪156275.16104209475  163198.49959239282  167636.44184079688  169289.7554628958 ␣
↪167962.4664877688  163574.87071203085  156170.70950276082  145918.04661766984 ␣
↪133103.7807189085  118122.12461109058  101457.74827359243  83664.59293880106 ␣
↪65341.597310527686  47106.72060538565  29570.693989742802  13311.883440190431  -
↪1146.4883783544647  -13355.721156459025  -22956.83697913424  -29691.111831825143␣
↪ -33405.74657339182  -34054.76810081443  -31695.500410948414  -26481.
↪145970813614  -18650.161052282405  -8513.19100571498  3561.645081947922  17164.
↪359354358137  31858.898525515397  47198.55240459056  62740.48708988166  78059.
↪03661920347
82798.73654851716  94796.9298609378  105721.48505918958  115375.625623878  123616.
↪14899683445  130353.86126221201  135552.17019271915  139223.9179986609  141426.
↪56742657  142255.90021913836  141838.44582788978  140322.92790676176  137871.
↪09068176866  134648.3384592436  130814.67939667338  126516.49892377958  121879.
↪68951250629  117004.62474332436  111963.38307847371  106799.50093032408  101530.
↪37065101373  96152.206700589  90647.29623220481  84993.04541410877  79172.
↪14831161793  73183.05929300946  67049.85902467475  60830.580902470654  54623.
↪11687592476  48567.95057034347  42847.16622984223  37679.44250034651  33311.
↪042708473105  30003.135644384583  28016.097553191976  27591.730847580962  28934.
↪563124181477  32193.54013544358  37445.48272586397  44681.631763109115  53798.
↪45620885836  64593.65544872613  76767.96351720407  89932.9823924808  103624.
↪86165729474  117323.23298583814  130474.43102814673  142517.71615251293  152912.
↪98356330695  161168.3154812276  166865.71838563093  169683.48708190466  169413.
↪84407937204  165974.80081382324  159415.5541794898  149915.1402762257  137774.
↪48719623784  123402.40982800539  107296.4440094629  90019.70127399103  72175.
↪12140392468  54378.59787823763  37232.448790470924  21300.608366359353  7086.
↪734209528354  -4983.819187542555  -14577.497534154363  -21461.157941355956  -
↪25505.740309069923  -26685.222369736108  -25071.31678794595  -20824.548909772697␣
↪ -14182.472600952271  -5445.838246547501  5036.474163100342  16882.994066627514 ␣
↪29694.995923418974  43071.066648668995  56620.46109719851  69974.92059933335
72067.08686712384  81874.09258704989  90865.08594264799  98913.28690246366  105938.
↪66958291284  111905.90687311272  116820.61047681898  120724.03536310107  123686.
↪45881842717  125799.49528507356  127167.66729452887  127899.61544321387  128099.
↪38935753437  127858.30806081272  127247.90253714543  126314.44664437449  125075.
↪53781425406  123519.1022124415  121605.07020618433  119269.80159249855  116433.
↪1451217131  113007.8065797847  108910.49044070613  104074.0901128367  98460.
↪04956733756  92069.92160354674  84955.11883306745  77223.90146131949  69044.
↪77388226184  60645.6660163838  52308.54441650612  44359.41533365006  37154.
↪02489187142  31059.90446278443  26435.724986522582  23609.18589028607  22854.
↪84904127231  24373.417676915393  28273.943254522877  34560.316541637796  43123.
↪16854600223  53737.98600041316  66069.85650485249  79684.82742413718  94067.
↪42137717007  108643.43210087583  122806.75882734735  135948.75248839665  147488.
↪36446217942  156901.32187076053  163746.60747041376  167688.69261325293  168514.
↪24548895325  166142.39311763272  160628.02769100422  152158.08547921217  141041.
↪15803884348  127691.19057592697  112606.35439842407  96344.42845390992  79496.
↪1751330776  62658.24192573629  46407.065330358084  31275.106370606223  17730.
↪524275093394  6161.117177485685  -3136.9502952466064  -9965.428687979162  -14225.
↪370924329429  -15913.985936923471  -15117.533552530185  -12000.989186265659  -
↪6795.29666008833  216.94484556161842  8716.541279649202  18362.765527941592 ␣
↪28807.778560478226  39709.89020671343  50745.26719916504  61617.81747349465
```

(continues on next page)

(continued from previous page)

```
61023.28564316299  68220.14689793179  74892.68836577167  80992.90947501117  86511.
↪03336131854  91470.80897536993  95923.30969082918  99939.48869445102  103601.
↪80009090481  106995.24921839489  110198.29235562358  113274.05842088163  116262.
↪40513217833  119173.34023504482  121982.32596602765  124627.9346862909  127012.
↪23121540979  129004.1222756725  130445.73954026014  131161.71862635086  130971.
↪0148710852  129700.67426126255  127200.77308015339  123359.5719406246  118117.
↪81697361839  111481.07811376505  103529.05207365875  94420.88011129093  84395.
↪73557545926  73768.21380661675  62918.390827425676  52276.78514222073  42304.
↪83243377781  33471.837380849815  26229.67174711335  20986.717387603057  18082.
↪685786476388  17765.967855288633  20175.072856321014  25325.50630972473  33103.
↪125430386295  43264.61711978088  55445.29473427628  69173.93747747902  83893.
↪93435532476  98989.57685673588  113816.00149774474  127731.04021324596  140127.
↪11091795526  150461.28111996024  158281.76344983664  163249.3431519317  165152.
↪57525364545  163915.99764886193  159601.0552846807  152399.88731094662  142622.
↪56105212367  130678.71446691737  117054.86794588793  102288.86927831444  86943.
↪03278196733  71577.52370924104  56725.42899473374  42870.75920849021  30430.
↪363868866516  19740.43640475797  11047.960976190534  4507.135438275502  180.
↪51480664677  -1955.6246616106982  -2002.3868328119136  -126.63017657463207  3452.
↪520532139184  8478.165160314205  14670.316969725873  21740.078791206357  29402.
↪716234242707  37389.16947153387  45455.67679324056  53391.304828924556
50124.453314359795  54363.112480101714  58393.12839173539  62251.965435309394 ␣
↪66005.31109498114  69739.65680283369  73553.63564964395  77548.46943277374 ␣
↪81817.93201047918  86438.29195096283  91458.74911936624  96892.91911326672 ␣
↪102711.93713545185  108839.73981148383  115151.0318082865  121472.34873083544 ␣
↪127586.48726287662  133240.39099163414  138156.36377756303  142046.2445165876 ␣
↪144627.93425325642  145643.4381371771  144877.39111385983  142174.89724002028 ␣
↪137457.44520825517  130735.67968527779  122117.91603688018  111813.48447547432 ␣
↪100130.27059359709  87466.1672093815  74294.54546939716  61144.26418897656 ␣
↪48575.13536530368  37150.119898724035  27405.81188249269  19822.957782006953 ␣
↪14798.830360988004  12623.22650426793  13459.682621259919  17333.210192911705 ␣
↪24125.46547538584  33577.80748556685  45302.1991789645  58799.40366490246  73483.
↪45655080467  88710.99106362957  103813.68359781473  118131.89575747782  131047.
↪52788844831  142014.17176117582  150582.84965996566  156421.93745982106  159330.
↪26614987306  159242.84956761164  156229.16225815506  150484.35604355155  142314.
↪22472597618  132115.07545232098  120349.92121953939  107522.5583056192  94151.
↪1302253328  80742.7099184924  67770.26595905326  55653.134830221905  44741.
↪82262592024  35307.63132527287  27537.27274515189  21532.321263183556  17313.
↪084597809953  14826.255445018218  13955.554813989893  14534.493547651344  16360.
↪359036812297  19208.57188278018  22846.64090215834  27047.06089764592  31598.
↪631951722236  36315.818453921325  41045.89962235695  45673.78294939691
39799.40251847992  40821.12729827249  41963.44459130746  43353.485836199325  45135.
↪4154808355  47460.291956148096  50474.99857024333  54310.68710078183  59071.
↪235475511625  64822.27628934531  71581.39710340957  79310.13701381788  87908.
↪39699292934  97211.83512465478  106992.72561436359  116964.6195482655  126790.
↪95760857014  136097.5572396072  144488.64113249988  151565.80690989218  156949.
↪07952459232  160298.9599606793  161338.2081137731  159871.9938918217  155805.
↪03415852267  149154.41378052055  140056.96913527703  128770.38637831314  115667.
↪52085526219  101223.8571290172  85998.47392822905  70609.32351742737  55704.
↪04741226754  41927.897818376805  29890.588087384764  20134.033396896088  13102.
↪95046704975  9120.157695544265  8368.160124194437  10878.23268094525  16527.
↪75471909882  25046.030761517948  36028.293108352584  48957.06009275616  63229.
↪5567080117  78189.52549238861  93161.49169705741  107485.41588520572  120549.
↪67711643182  131820.47802051262  140866.03632462834  147374.3036180336  151163.
↪40195747686  152184.45822428475  150517.00887478812  146357.60874963168  140002.
↪6754637906  131826.91014646532  122258.83829192528  111755.10243300168  100775.
↪11162077221  89757.52030964314  79099.78769702816  69141.78004596678  60154.
↪048771269314  52331.07305024961  45789.42329119834  40570.503665997494  36647.
↪285997064246  33934.26559036233  32299.75774401977  31579.611003838218  31591.
↪433610095017  32148.502449169573  33072.63578805429  34205.44750912984  35417.
↪54725004328  36615.39537131504  37745.654490340516  38796.99443015334
```

(continues on next page)

(continued from previous page)

```
30426.407799582026  28076.429209519672  26179.57428089785  24954.07008849848 ␣
↪24623.105942787122  25402.0594263365  27485.149965917768  31032.048373311052 ␣
↪36155.03172944927  42907.32562361998  51273.31029231904  61161.27287052479 ␣
↪72399.35448845624  84735.26004130681  97840.16529519051  111317.07035840402 ␣
↪124713.61561249832  137539.10673876622  149285.20550611237  159449.45278934616 ␣
↪167560.5229116682  173203.88774429896  176046.41761628413  175858.38301199532 ␣
↪172531.3596964566  166090.68648566172  156701.3771825675  144666.73526595638 ␣
↪130419.34236798953  114504.5627395114  97557.19366009979  80272.36093018463 ␣
↪63372.17373197502  47569.98143808665  33534.288972795446  21854.467522573395 ␣
↪13010.333855349156  7347.4650885791125  5059.778226219052  6180.457165176908 ␣
↪10581.784796611166  17983.87126753857  27971.70185760221  40019.39998820974 ␣
↪53520.15041600989  67819.886563551  82252.63769922151  96175.3695879563  109000.
↪23843210963  120222.4033618898  129441.88806962238  136378.41947795017  140878.
↪66645523405  142915.81705436524  142581.93076665414  140073.94776087403  135674.
↪60031347588  129729.73046148277  122623.65914518788  114754.27220839527  106509.
↪39349109432  98245.81925726231  90272.11255146532  82835.92666176255  76116.
↪27183370473  70220.78666160736  65187.750580441614  60992.29753150145  57556.
↪078244831195  54759.47809628983  52455.43083479464  50483.87109851236  48685.
↪93087387678  46917.09357156666  45058.658932193095  43027.027076783495  40780.
↪466646319706  38323.1788082382  35706.59820389251  33027.97987156216
22314.643292568304  16552.185470156444  11569.036831831945  7672.998083736384 ␣
↪5164.205104243883  4319.888753786392  5378.954991941297  8526.992137543843 ␣
↪13882.373573032964  21484.172155064647  31282.62552714256  43132.87764222617 ␣
↪56792.66067277193  71924.46570493317  88102.57723074361  104825.11803362827 ␣
↪121530.97623136066  137621.17959202308  152483.96360172398  165522.47284042416 ␣
↪176183.7657096327  183987.5858988646  188553.24297292047  189622.9269269561 ␣
↪187079.87830442138  180960.0488038825  171456.21017815336  158913.88525815675 ␣
↪143818.9591784019  126777.34948278329  108487.63427547194  89708.01967631135 ␣
↪71219.43454866638  53786.83916486743  38120.999724218505  24842.996489357654 ␣
↪14453.594847554268  7309.322629034586  3606.681723859743  3375.4057778906867 ␣
↪6481.094888905114  12636.95435396019  21423.781176697397  32316.821271216177 ␣
↪44717.69931313414  57989.331443273295  71491.58793197843  84615.48637376355 ␣
↪96813.86192832893  107626.76399519359  116700.24289824071  123797.68224986401 ␣
↪128803.36428186316  131718.48641029245  132650.33895325055  131795.77206178498 ␣
↪129420.39744932269  125835.16927661974  121372.06046704805  116360.49791182697 ␣
↪111106.05412994113  105872.63369272319  100869.06581086121  96240.64910745565 ␣
↪92065.82079895967  88357.76857764338  85070.49374386878  82108.58713650174 ␣
↪79339.80685658993  76609.45310790587  73755.5181579119  70623.63989570833  67080.
↪99258420186  63028.39221035687  58410.05945084266  53220.65471645785  47509.
↪36350563482  41380.95647617104  34993.87155082874  28555.46360872512
15690.261436028653  6593.386739594418  -1412.959276337584  -7935.927381953588  -
↪12603.188814785492  -15080.39231445406  -15088.383238944953  -12419.509136589639␣
↪ -6952.27593317171  1336.422586632034  12362.2890336672  25928.572804067553 ␣
↪41725.64975307137  59335.744157607995  78243.09407545907  97849.5930273614 ␣
↪117495.62854740405  136485.4998567131  154116.45609819828  169710.07995757554 ␣
↪182644.47693539082  192385.54446359113  198515.50992197864  200756.95846575144 ␣
↪198990.7281527485  193266.32924934488  183803.93475496635  170987.46833367337 ␣
↪155348.85373921087  137544.04987376306  118322.03763405232  98488.4080535783 ␣
↪78865.58824613193  60252.00071314309  43382.560016391  28892.856960225854  17289.
↪164819702688  8926.03778169769  3992.783245582709  2509.511384610145  4332.
↪8389940598645  9170.695643422012  16605.094290629684  26121.22822632242  37140.
↪877176171474  49057.873730067  61273.31156032272  73228.27111205406  84432.
↪08598987576  94484.55199048402  103090.95897005146  110069.36503017324  115350.
↪09086287333  118967.94745546918  121048.18398047594  121787.52217182372  121431.
↪90511279437  120252.718704726  118523.24063504991  116496.94250766379  114389.
↪03342678152  112362.3126721086  110518.02504902334  108892.01706872389  107456.
↪10664879228  106124.23156829801  104762.65505992952  103203.29630221166  101259.
↪12706505129  98740.53348192872  95471.5769591404  91305.18798263701  86136.
↪47454180164  79913.50468410779  72645.11215391949  64405.458996505484  55335.
↪25677299746  45639.69043468601  35583.2025493752  25481.381606514995
```

(continues on next page)

(continued from previous page)

```
10687.794113195261  -1547.2375230994658  -12402.772093113483  -21408.362827141114 ␣
↪-28126.848104637644  -32173.70163174306  -33235.713126477975  -31088.
↪266737368744  -25610.427156576043  -16797.010004802185  -4766.817868018625 ␣
↪10233.720579910558  27832.15377077893  47537.314984447876  68752.54909888923 ␣
↪90794.76886033508  112918.90654664805  134346.96334141956  154300.5028550695 ␣
↪172035.11631874545  186875.1344860508  198246.7020418905  205707.2858435857 ␣
↪208969.7723904187  207919.52677053737  202623.12860837177  193327.95313471666 ␣
↪180452.30041771499  164566.3578620832  146364.8695453674  126632.93761775458 ␣
↪106206.8537121407  85932.21455229736  66621.78600864032  49015.62454507113 ␣
↪33745.83749482088  21308.070260179935  12041.368644880415  6117.508970391449 ␣
↪3540.257501319859  4154.360199675422  7663.422539981431  13655.264153548109 ␣
↪21632.86626329999  31048.704436380085  41340.09710444101  51963.210667415035 ␣
↪62423.54041563027  72301.01566071129  81268.32873346104  89101.62349090693 ␣
↪95683.25672815711  100996.92053261682  105115.94202513836  108186.0211855348 ␣
↪110403.99775061084  111994.43493397016  113185.8628083679  114188.44107028886 ␣
↪115174.59379710148  116263.86098088755  117512.83297341775  118910.61813531144 ␣
↪120379.8752545327  121783.05270617797  122933.14301860842  123608.00478109583 ␣
↪123567.13586444761  122569.70640481997  120392.67249886374  116847.88070599592 ␣
↪111797.22267404929  105165.08883142304  96947.57962711103  87218.14308849491 ␣
↪76129.50252100293  63911.906528900145  50867.86897135589  37363.668059659656 ␣
↪23817.945336391094
7347.232293056209  -7717.634998010734  -21142.019279539694  -32388.047610200916  -
↪40963.50480169457  -46442.68809479843  -48486.15788515213  -46858.607846062965  -
↪41444.02506541791  -32257.28692267131  -19451.36168038049  -3319.3574189535434 ␣
↪15709.19183508023  37078.199846698284  60121.178660762234  84083.44786226319 ␣
↪108149.81380871563  131476.7482342908  153227.73222439515  172610.11789412855 ␣
↪188911.62663274733  201534.47612320434  210025.12850954942  214097.79009417706 ␣
↪213650.0692143521  208769.60251280386  199730.9685915298  186982.79007012886 ␣
↪171125.54075881466  152881.17974081042  133056.28324130864  112500.79545936665 ␣
↪92064.83454333595  72556.14258032547  54700.74381344109  39109.171781280325 ␣
↪26250.256102819218  16433.94877872661  9804.05498808399  6341.0596711401095 ␣
↪5874.558748204399  8104.163445399732  12627.195349145244  18971.068779004716 ␣
↪26627.995722048614  35089.56401389472  43878.834752691895  52577.869051671645 ␣
↪60849.00312571118  68448.70949327062  75233.46811062298  81157.67777299174 ␣
↪86264.21814969135  90668.78304580785  94539.50966151374  98073.7003309735 ␣
↪101473.5576799846  104922.8289543678  108566.09011388986  112492.11554094109 ␣
↪116722.40374030611  121205.49767809972  125817.2872817309  130367.04706595767 ␣
↪134608.57591563137  138255.49437657418  140999.5348175308  142530.53984825063 ␣
↪142556.8636028869  140824.93983350994  137136.92441245535  131365.51743622264 ␣
↪123465.29910932716  113480.15169655031  101546.56731226598  87892.84271532923 ␣
↪72834.32763020074  56765.018406947194  40145.87574995666  23490.300247191175
5616.801079347214  -11868.870909645688  -27485.11885631629  -40639.20969029718  -
↪50795.77449864359  -57498.726143398395  -60391.64652085513  -59235.837378379474 ␣
↪-53925.181843677456  -44496.950575650495  -31137.72271209935  -14183.
↪690732727455  5885.2118234948575  28457.67862969467  52809.90530128262  78130.
↪62082234997  103551.5355866571  128182.07338758641  151146.8915292009  171624.
↪39229209197  188884.22112248227  202321.6579253392  211486.8556802149  216107.
↪07311552975  216100.38171029068  211579.786232983  202847.25546682748  190377.
↪77920597058  174794.20559022226  156834.22236404946  137311.379950676  117072.
↪47078378948  96953.84350807275  77739.31885278621  60122.27589792252  44674.
↪1972423094  31821.517802759685  21832.04629946068  14811.563295331944  10710.
↪494684380305  9339.867833825243  10395.131266245206  13485.904679812913  18169.
↪36205780352  23984.76232640502  30486.641079161185  37274.35995995561  44016.
↪05956200764  50465.5465161546  56471.225072262714  61976.81062972929  67014.
↪18752278297  71689.34771822052  76162.82829616201  80626.41992522044  85278.
↪12368749137  90297.38008803569  95822.48558315527  101931.86447124572  108630.
↪50392040094  115842.42137626265  123409.55515354939  131096.9901352856  138603.
↪98814536503  145579.91814362025  151643.8984308623  156406.7856333155  159494.
↪0774669647  160568.332689033  159349.8384562884  155634.4524539397  149307.
↪79072482995  140355.19704537775  128867.19232711899  115040.3429280651  99173.
↪69038240425  81661.04371746708  62979.54760739722  43675.00948665207  24344.
↪50552033977
```

(continues on next page)

(continued from previous page)

```
5361.114315308507  -14051.636608405586  -31400.449871656998  -46052.261217929925  -
↪57442.17029614675  -65095.94093030476  -68650.5631410506  -67872.05654707644  -
↪62669.65523835315  -53105.51250448662  -39399.116296360095  -21925.725758322922 ␣
↪-1208.332582407631  22097.082108618415  47222.89287090393  73317.2950484521 ␣
↪99477.00516883022  124783.91623773496  148344.07237426046  169327.0462255381 ␣
↪187003.62637546772  200779.67490896236  210224.11280933957  215089.23727821512 ␣
↪215321.96247360547  211065.08335976122  202648.2599025952  190569.06516693791 ␣
↪175465.08935806513  158078.69354716683  139216.51451377774  119706.19441162184 ␣
↪100353.01388141897  81899.12529701425  64987.90893473354  50135.61904259799 ␣
↪37711.973710532504  27930.709572816035  20850.417069192605  16385.24770170868 ␣
↪14324.39663736398  14358.664514515986  16111.936699518948  19175.1209725581 ␣
↪23139.976543413028  27630.35400453415  32328.637335901993  36995.61081238308 ␣
↪41482.52869174058  45734.79781729264  49787.34153463029  53752.34556120822 ␣
↪57800.644132863235  62138.44679219764  66981.40212745251  72528.12700390871 ␣
↪78935.29461433545  86296.18195147546  94624.24932163165  103842.89343486221 ␣
↪113782.02033082911  124181.56695325943  134701.60196395905  144938.1946366756 ␣
↪154443.88551950714  162751.34483316913  169398.67460483813  173954.79796539102 ␣
↪176043.4736540887  175364.6572331341  171712.17871187138  164986.99214004935 ␣
↪155205.54908985362  142503.13025553938  127132.21794941975  109456.19376077011 ␣
↪89938.79392690575  69129.85096322664  47647.90089226269  26160.252667672718
6374.085931136666  -14407.210015599821  -32965.27207272658  -48642.67826315153  -
↪60859.86141385783  -69137.68878334606  -73117.72378736219  -72579.35051416414  -
↪67452.91046335894  -57828.011389878375  -43956.2367165645  -26247.622752919375  -
↪5260.485479520823  18315.53353499448  43682.98260694148  69966.06498029378 ␣
↪96245.03381539986  121594.10644208394  145121.14754944437  166007.11664818987 ␣
↪183543.13586694538  197163.03234913942  206469.35760938085  211251.1847130493 ␣
↪211492.42085338538  207369.92317632178  199241.33393715433  187623.21323195507 ␣
↪173160.6945905594  156590.4706087716  138699.3857460661  120281.23198658497 ␣
↪102094.4817593277  84823.63650674584  69046.61901567507  55210.209096153776 ␣
↪43614.94597578858  34410.23967728749  27599.699324182075  23055.955094527857 ␣
↪20543.57876410029  19748.147163105634  20309.08620098044  21853.71071464797 ␣
↪24029.85264827908  26534.646221825118  29137.397082634012  31694.971838298996 ␣
↪34158.76206060821  36572.9521094082  39064.49863107075  41825.85799863802  45092.
↪02833063158  49113.86635407561  54129.869792051795  60338.670760513305  67874.
↪36711691273  76786.54303718568  87026.42559892882  98440.12829993642  110769.
↪3883422426  123659.65695172227  136674.8935497111  149317.98205311786  161055.
↪3591447689  171344.2376686032  179660.72958777775  185527.2172314112  188537.
↪4739517336  188378.27311408485  184846.51977298886  177861.26265731506  167470.
↪26644020193  153851.1207245066  137307.11369074226  118258.29321762123  97228.
↪27260968107  74827.41539392789  51733.06372588864  28667.471852945728
8395.71143970111  -13153.561329327742  -32354.893699542914  -48543.342656888126  -
↪61140.24061935104  -69675.34472576372  -73806.32553197793  -73334.79195328019  -
↪68218.0271468249  -58575.632972556064  -44690.36485879452  -27002.595398636477  -
↪6098.077738122141  17311.015797707936  42411.441923968334  68319.88502262019 ␣
↪94118.46418956268  118893.22779271351  141773.7976146054  161972.1030482509 ␣
↪178818.04791463725  191789.99820324648  200538.17744675418  204899.40406632103 ␣
↪204902.08538271367  200760.9672617387  192861.78726045683  181736.64582180092 ␣
↪168031.54387064537  152468.085848079  135801.7692996425  118779.53896122315 ␣
↪102099.35047034721  86374.35666482219  72104.00410255505  59653.83087257237 ␣
↪49245.1251065404  40954.88440701597  34725.76485787677  30384.982639043392 ␣
↪27670.488056902643  26262.221091419906  25815.91877263131  25996.80332072814 ␣
↪26510.54547106564  27129.162458923973  27709.951416619224  28206.139131679483 ␣
↪28668.600214887832  29238.7031448565  30133.03087957818  31621.336058849345 ␣
↪33999.5838022332  37560.2722180149  42562.380096305096  49203.26606470938  57594.
↪642130376946  67744.38939108729  79545.50873546946  92772.94680484904  107088.
↪45393779878  122053.06303052683  137146.26916937524  151790.57516149143  165379.
↪77335301414  177309.17339444856  187005.9607703173  193957.97234472854  197739.
↪38346345315  198032.08956499945  194641.90279855492  187509.03885071937  176712.
↪71124434465  162469.9548958232  145129.04975774995  125158.09911877137  103129.
↪43412612534  79700.57253227528  55592.46785790772  31565.761390498697
```

(continues on next page)

(continued from previous page)

```
11131.626740480235  -10567.559460161481  -29826.889689259657  -45990.98215507161  -
↪58497.760713569114  -66899.6744920754  -70881.95873681584  -70277.16424623196  -
↪65075.17032021293  -55427.93165859119  -41648.308632450564  -24202.50264342646  -
↪3695.866003800693  19147.827822989944  43513.198508819114  68525.30454129583 ␣
↪93285.6710433644  116910.69082316931  138570.5018452991  157526.2606712596 ␣
↪173163.67829816704  185020.77921594575  192808.09179050205  196419.87072309502 ␣
↪195935.46970796157  191610.59186896  183858.8046914675  173224.36605631735 ␣
↪160348.0171890785  145927.90728710004  130678.17994829688  115287.94018001971 ␣
↪100383.31298655775  86495.09607199955  74034.112058481  63275.807437921714 ␣
↪54354.9674893122  47270.670223334535  41900.84587616808  38025.100798954925 ␣
↪35353.86148371656  33561.44342106179  32320.38576915018  31334.336504232542 ␣
↪30366.927034632536  29264.426319935235  27970.482629221828  26531.903230308108 ␣
↪25095.13602036455  23893.844834183226  23228.6541724689  23440.726296980254 ␣
↪24881.279948806074  27879.433746382656  32710.841820265166  39569.48348648326 ␣
↪48544.68724611361  59605.0403624231  72590.29783030333  87211.80500885367 ␣
↪103061.33578841109  119627.67080085233  136319.7403645937  152494.7683291961 ␣
↪167489.59829085073  180653.27281242702  191378.96669462835  199133.53283052772 ␣
↪203483.1800589015  204114.13638803017  200847.5243314982  193648.05431931646 ␣
↪182626.49696403887  168036.20108776403  150264.16542314726  129817.34015564057 ␣
↪107304.93107329626  83417.51285074015  58903.74363068621  34545.42880079792
14274.213223086444  -6964.435032895622  -25701.431539873418  -41307.657650150206  -
↪53252.860008026946  -61125.467791210816  -64649.20372495474  -63695.760028832534␣
↪ -58292.80809990417  -48626.65858239062  -35039.00116157154  -18017.339267990305␣
↪ 1821.0059134511105  23751.124040284034  46968.40845485787  70621.15394131716 ␣
↪93846.52020875218  115808.18809651244  135733.78241945384  152949.99394110544 ␣
↪166913.3269108929  177234.54158674984  183695.15470938786  186254.7933849102 ␣
↪185048.74472690758  180375.66888937712  172676.1018934092  162503.01668979746 ␣
↪150486.2848442672  137293.3390445587  123588.63792536364  109994.65054656268 ␣
↪97056.99362814866  85216.07174893771  74787.10690519775  65949.83208010343 ␣
↪58748.40953564719  53101.37327259143  48820.64601883563  45638.00339815958 ␣
↪43236.80550237546  41286.432980434976  39476.68139571597  37549.39820935004 ␣
↪35324.88813371697  32721.04445722331  29763.750602241897  26587.789690366117 ␣
↪23428.24412296673  20603.102607066572  18488.460745222605  17488.251765649155 ␣
↪18000.83530830487  20384.97732954759  24927.761828965013  31816.79023135657 ␣
↪41118.66729931854  52765.27707153591  66548.76214333833  82125.48416961607 ␣
↪99028.61380520584  116688.42252542466  134458.86813190382  151648.71141441932 ␣
↪167555.19234761514  181498.23611094803  192853.24500056833  201080.74339386134 ␣
↪205751.4517837511  206565.73913289106  203366.80469833536  196147.33654497526 ␣
↪185049.75410187175  170360.44317161175  152498.6192596772  132000.6037297958 ␣
↪109500.37083392797  85707.23346307845  61381.49841620188  37308.85793790383
17523.954274987453  -2675.7844603217527  -20338.981223750794  -34878.49597788146  -
↪45810.10005094898  -52770.45993243999  -55531.722942428096  -54012.031579976465 ␣
↪-48281.301196439235  -38561.655343410304  -25222.03814536734  -8766.719960842616␣
↪ 10182.320387898872  30908.827367757374  52630.83947913311  74533.3463570496 ␣
↪95803.62234744329  115667.52455551917  133424.82988884766  148481.58956345456 ␣
↪160377.5216463565  168806.65250290872  173629.75405010785  174877.58990377025 ␣
↪172744.5517172373  167572.898377741  159828.4585733662  150069.27085439436 ␣
↪138909.1642594511  126978.68118578954  114885.97573976884  103180.36109418476 ␣
↪92321.0192229942  82653.03289948539  74392.3763004257  67620.84479750902  62291.
↪16605583601  58241.770614663175  55219.97089978865  52911.661250433375  50975.
↪159188469515  49076.49960667507  46923.393815363415  44295.18252593186  41066.
↪436269729675  37222.3620051055  32864.819714945865  28208.4855116807  23567.
↪459363431924  19333.34557120211  15946.47549009601  13862.44573150523  13516.
↪474301140743  15288.209821905419  19469.559584142466  26237.84156692362  35636.
↪13981339481  47562.19013656005  61766.49143262008  77859.67877164009  95328.
↪5597749798  113559.65300472119  131868.615346588  149533.63261427678  165830.
↪68875394296  180068.62539691484  191622.04350718122  199960.35984821245  204671.
↪6820104155  205480.5709557624  202259.18237118894  195031.682821649  183972.
↪19444013736  169396.811185234  151750.43828681746  131589.33177167608  109560.
↪26327074032  86377.22039942624  62796.49368236401  39590.91838560792
```

(continues on next page)

(continued from previous page)

```
20609.76033590357  1972.5488742995515  -14116.715707531128  -27127.04130136814  -
↪36632.869821868255  -42329.85140320234  -44047.087503244395  -41755.235532026956␣
↪ -35569.88554096306  -25749.69032439141  -12688.86595719806  3096.1149330900885 ␣
↪20985.543845803244  40280.793884427505  60233.1616037489  80076.10236320138 ␣
↪99059.39941116209  116483.54151321031  131732.41381924797  144302.35797857103 ␣
↪153825.74762950733  160087.46049809825  163033.0019980073  162767.52808059595 ␣
↪159545.5958348118  153752.09787439508  145875.46315491866  136474.78215943754 ␣
↪126142.98972464538  115468.5716146409  104998.41887866148  95204.41800205379 ␣
↪86456.1313170853  79001.5038245928  72956.9579146548  68307.54854923842  64917.
↪10075105396  62547.49733055085  60885.58754250895  59575.60287524121  58254.
↪541946823156  56587.75735466652  54301.962438114824  51213.077015178606  47246.
↪73239562963  42449.82522493118  36992.20128373767  31158.308872946905  25329.
↪42647061802  19957.78065549919  15534.472670373034  12553.579916954695  11475.
↪059898857333  12689.142165346759  16484.748944049745  23024.15403137711  32325.
↪603311732644  44255.021668587935  58527.270238995516  74716.74820320592  92276.
↪50628511094  110564.5006838842  128875.2025699275  146474.5137123444  162635.
↪83381166405  176675.17647883116  187983.42264968832  196054.10620303705  200505.
↪51288888266  201096.30275563948  197734.30036511144  190478.50189124572  179534.
↪69579402852  165245.36477317492  148074.7214716021  128589.82903552087  107438.
↪77909524388  85326.85993854929  62991.56701035747  41177.20839588657
23307.064760225912  6681.4066193519975  -7405.081536642407  -18489.689320646692  -
↪26216.158106266856  -30347.93535959498  -30777.825798433692  -27533.506116878998␣
↪ -20778.397663099437  -10807.472389494607  1962.2881339574599  17006.
↪850093143687  33715.299497283675  51414.710449845305  69398.85138178861  86959.
↪53715685711  103419.13938084959  118162.54262548994  130666.71016352405  140526.
↪02227622003  147471.6869409589  151383.79895805585  152295.02898745198  150385.
↪43571727345  145968.4784962368  139468.9215459906  131393.9163482159  122299.
↪07676690606  112751.77584388887  103294.15530607168  94408.42108314006  86486.
↪88698183438  79808.92621105928  74526.51484115265  70659.43559695115  68100.
↪50022474618  66630.39854739606  65941.05119468237  65665.68935756115  65413.
↪3618125604  64805.21993847849  63509.78479368175  61274.469492161916  57950.
↪91053198899  53512.13159569718  48060.185445344476  41823.644370998576  35145.
↪0796016819  28459.424276795977  22264.793976847584  17087.891149757244  13446.
↪503789905219  11811.797467691202  12573.082582988474  16007.521674386186  22256.
↪84683086449  31312.620461081977  43010.939892652576  57036.809605929404  72937.
↪73795972922  90145.50857005097  108004.57336344989  125805.14746929564  142818.
↪87539583142  158334.88929707493  171694.1858631635  182320.4885524156  189746.
↪10695578088  193631.7190762263  193779.44654987394  190139.02970504013  182807.
↪30568218633  172021.52269330784  158147.27008317536  141661.960345741  123134.
↪8685742377  103204.728419085  82555.81936129436  61893.37962078278  41919.
↪06478980086
25452.643977276854  11196.851271470967  -545.8312323275459  -9390.674090621425  -
↪15058.954056233692  -17388.452489311298  -16340.33998003656  -12001.994400858966␣
↪ -4585.336415731559  5579.643706685747  18056.342936477806  32317.94770762183 ␣
↪47768.475860368824  63767.84743294058  79660.0540991133  94803.1925784124 ␣
↪108599.86929501613  120526.30281253225  130158.3739666878  137192.92079656094 ␣
↪141462.7561603188  142944.19854688845  141756.3369921372  138151.77327382172 ␣
↪132499.16260686587  125258.46434045263  116950.36923028338  108121.84249789455 ␣
↪99310.06935493262  91007.27774521655  83628.91945405629  77487.50754510402 ␣
↪72774.04301242018  69548.43975709482  67739.71164445979  67155.96675052619 ␣
↪67503.51753892275  68413.71990752961  69475.55489602103  70271.51344672339 ␣
↪70414.07480624609  69580.00620842172  67539.86197524969  64180.41334597395 ␣
↪59518.268901064075  53703.607814736315  47013.691974727175  39836.58948321607 ␣
↪32646.27096983855  25970.874850070126  20356.429749111558  16328.635564421975 ␣
↪14355.417854300586  14812.878722901525  17956.98299715027  23902.868323905 ␣
↪32613.09146115451  43895.46896711555  57410.49170382862  72687.64198403427 ␣
↪89149.36818374181  106141.01416161946  122964.68835834328  138914.9051425217 ␣
↪153313.84001537703  165544.1985787606  175077.9831419972  181499.8182490411 ␣
↪184523.9297726203  184004.32258192045  179938.1323646569  172462.50662633765 ␣
↪161845.67481139387  148473.08404725313  132829.6014240893  115478.82173440412 ␣
↪97040.4849698215  78166.9201154159  59519.31338368875  41744.47252067462
```

(continues on next page)

(continued from previous page)

```
26955.322281405643  15325.26278147615  6167.2297285565  -218.99962133247755  -3637.
↪8048243339435  -4005.3083202741113  -1353.3978533209302  4170.182671313407 ␣
↪12305.996616057326  22691.557782321648  34874.523126539905  48330.29221256418 ␣
↪62483.61079387007  76733.48664117836  90480.4312726476  103154.7702972305 ␣
↪114244.54789438665  123321.41612761171  130062.87104969917  134269.29082512835 ␣
↪135874.4521885592  134948.5447914208  131693.1508420967  126428.18140017799 ␣
↪119571.32318021142  111611.10708891225  103075.215860803  94496.05861102251 ␣
↪86375.91617923442  79154.07357989714  73178.28800601358  68682.6904728956  65773.
↪79941169315  64425.7630613768  64485.28481703536  65685.97183130276  67671.
↪13775447087  70023.44209412888  72299.21397394687  74064.93136930073  74933.
↪14021322614  74595.11798320359  72847.81367158108  69613.01373287305  64947.
↪25947317441  59041.73029971207  52212.055063590116  44878.7616154951  37539.
↪76421257823  30736.866340371143  25018.679216089437  20902.593194205114  18838.
↪4754255006  19176.60378161117  22142.00158757896  27816.84161624251  36131.
↪98377880372  46868.04897439531  59665.76449253898  74044.6953910432  89428.
↪94668628686  105178.01878908838  120620.74708281971  135090.166000767  147957.
↪2048129683  158661.33030219618  166736.5738016515  171831.7823837282  173724.
↪37806883524  172327.3562661452  167689.67006273044  159990.5015559974  149528.
↪1948247334  136704.8067841992  122007.32146156905  105986.57866552906  89234.
↪90548046623  72363.32930315609  55979.11718796096  40664.2502266438
27801.970811425475  18944.173808359366  12503.745329332714  8691.444972681973 ␣
↪7617.056321149168  9284.787563554768  13592.126766190144  20332.4323673353 ␣
↪29201.486862596343  39808.138577578175  51688.998714942725  64326.95203465425 ␣
↪77172.99106853275  89670.6144767269  101281.76455414601  111513.04492395409 ␣
↪119940.78558326575  126233.43458083746  130169.77396953087  131651.594281795 ␣
↪130709.71840623193  127502.63295238277  122307.44236838711  115503.37762432866 ␣
↪107548.62883517548  98951.78675303122  90239.6276343793  81923.31862694131 ␣
↪74465.32245477207  68249.31702850462  63555.30676355127  60541.79132751202 ␣
↪59236.39237104436  59535.75121752233  61214.8434672604  63945.160889657825 ␣
↪67320.54145968254  70888.8387427657  74187.16041579521  76778.11127072157  78284.
↪37426004105  78419.06429117273  77009.5874160345  74013.21061003776  69523.
↪15804037765  63764.75087503267  57081.84428801188  49914.52969977345  42769.
↪70691770653  36186.64091712539  30699.963472165662  26802.736421342433  24912.
↪152330732348  25340.217028826883  28271.359360151822  33748.38185272693  41667.
↪5467371946  51782.9353923631  63719.577498633924  76994.26743165165  91042.
↪5110359242  105249.7068334469  118984.48036470174  131632.0636377787  142625.
↪7352394879  151474.5903990855  157786.2646846404  161283.65435745276  161815.
↪12240513403  159358.1146353999  154016.50201240578  146012.28780781757  135672.
↪55422792258  123412.66590878482  109716.79965893962  95116.8422196451  80170.
↪6083233234  65440.201709219  51471.194952519116  38773.161333911776
28058.486844840572  22008.282816501858  18307.762451985545  17077.56579605463 ␣
↪18341.78808673998  22026.359898567738  27960.71700959623  35882.95256370593 ␣
↪45448.585217913744  56242.969893720794  67797.22051571382  79607.31224842451 ␣
↪91155.79724231231  101935.32233289414  111472.90386004592  119353.7205155014 ␣
↪125243.0561353248  128904.98341797222  130216.44318980571  129175.55000087486 ␣
↪125903.2413146035  120637.77140087179  113722.00898506076  105583.9976642384 ␣
↪96711.74163248358  87623.64475821012  78836.41743680282  70832.53612357375 ␣
↪64029.466138821124  58752.8212057522  55215.42819340015  53503.90163567889 ␣
↪53573.83248210251  55254.094174903265  58260.111157013 78  62215.2709995965 ␣
↪66679.04410134448  71179.85441004258  75250.36403930806  78462.62653075127 ␣
↪80460.54679414604  80987.26403559765  79905.43501895381  77208.91163208094 ␣
↪73024.93904226093  67606.69937778742  61316.73619244377  54602.46152342369 ␣
↪47965.51856922255  41927.20544384356  36992.42731763223  33614.71790198964 ␣
↪32164.754457616382  32904.49641421724  35968.6333211371  41354.471595730196 ␣
↪48920.767652455164  58395.37750496245  69390.98932345991  81427.68051483264 ␣
↪93960.63121487515  106411.05715547071  118198.30949907641  128771.1270182115 ␣
↪137636.20403535824  144382.53221362777  148700.35395041143  150393.993368333 ␣
↪149388.27010654434  145728.61594314934  139575.37471174652  131193.0497124229 ␣
↪120935.45676985016  109227.84187435768  96547.03577191831  83400.6575479368 ␣
↪70306.26442857918  57771.19799494422  46273.72032188246  36245.887374033184
```

(continues on next page)

(continued from previous page)

```
27865.727534648926  24550.370259053474  23503.61621183646  24757.786578620915 ␣
↪28254.25326275303  33844.71320204668  41295.53540806862  50295.286286272254 ␣
↪60465.47492304501  71374.45090454613  82554.23308285393  93519.85593982716 ␣
↪103790.60509967504  112912.29508916463  120479.54440858052  126156.85044149071 ␣
↪129697.18331697487  130956.82290122929  129905.26921490223  126629.26759602029 ␣
↪121330.29981528319  114315.2849643991  105980.68385437323  96790.67495217481 ␣
↪87250.53074898628  77876.73145894658  69165.67059996667  61563.00231123669 ␣
↪55435.72992280839  51049.02725778691  48549.51920462516  47956.340360694914 ␣
↪49160.76675385517  51934.61328167759  55946.953913068806  60788.10240535833 ␣
↪65999.2378812903  71105.61747622651  75651.0246536138  79230.98248534281  81522.
↪32816684232  82306.99522723476  81488.26531737021  79098.3005851608  75296.
↪40797050443  70358.16830207527  64656.23259116983  58634.19328095486  52775.
↪432796421876  47569.19825230314  43476.32465468957  40897.01868392514  40142.
↪92542154763  41415.34976129851  44791.02379553774  50216.24188382183  57509.
↪57305551533  66372.75444235955  76408.81592304006  87146.02538895217  98065.
↪90709120172  108633.39201102861  118327.11607428262  126667.98362354109  133244.
↪3426623515  137732.44836998882  139911.2890697553  139671.2778003277  137016.
↪73668337817  132062.48766028153  125025.18535265901  116210.26745664058  105995.
↪54616986339  94812.52103116532  83126.46814102147  71416.26883907552  60154.
↪802980062814  49790.57040419696  40731.04078559422  33328.08535757078
27430.655840629173  26677.137448959096  28096.528202653826  31637.360454804908 ␣
↪37163.88843169131  44460.09908738898  53236.531221875695  63139.92385708567 ␣
↪73765.64740990014  84672.7620884318  95401.39953436745  105491.98516225105 ␣
↪114505.6251761631  122044.79390738002  127773.29667745245  131434.37387294383 ␣
↪132865.7739430061  132010.6722560199  128923.45768124127  123769.6491877895 ␣
↪116819.53101563484  108435.48815169951  99053.4569100268  89159.34526249126 ␣
↪79261.68776174937  69862.14390159684  61425.69332661341  54352.49973590141 ␣
↪48953.3902638113  45430.72257818581  43866.09437560201  44215.908254595255 ␣
↪46315.269253863604  49890.10184089063  54576.77296630617  59947.94533753158 ␣
↪65542.90591943833  70900.25873437653  75590.66896727422  79247.31582709544 ␣
↪81591.85942851084  82454.04260469865  81783.50876700038  79652.986506499  76252.
↪62637447056  71875.92560990274  66898.29170409901  61749.827819333375  56884.
↪33140688772  52746.75142686159  49741.43165445354  48203.37405018527  48374.
↪497643077695  50386.46845261213  54251.16911322996  59859.30523013678  66987.
↪05524557535  75310.10790578263  84423.93860913077  93868.78801871679  103157.
↪54839810682  111804.64853931835  119354.05792784841  125404.69411073  129631.
↪79294711015  131803.16041876544  131789.63288684195  129569.49436118543  125227.
↪00038165454  118945.50896541956  110995.99717909006  101721.9332146876  91521.
↪573545761  80828.76689098674  70093.28315438471  59761.5639681285  50258.
↪63317894492  41971.73275670982  35236.08270649484  30323.019333894234
27013.22078485548  28560.27428594409  32168.002636210753  37708.134460605375 ␣
↪44975.77722134671  53695.94275492881  63532.574401778606  74100.015858781  84976.
↪79234780406  95721.4677223021  105890.2011397422  115055.46452505494  122825.
↪21318052642  128861.64625801015  132898.5727479839  134756.33272853968  134353.
↪23016028514  131712.52426198017  126964.20583178732  120341.04864286594  112168.
↪76125211618  102850.44997143271  92846.0113199758  82647.46939832461  72751.
↪62594444846  63631.66537686229  55709.525484812824  49330.885613902836  44744.
↪52700800451  42087.58447195911  41377.846590912624  42513.79653518699  45282.
↪54984769173  49375.27959834915  54409.166899707205  59954.420293236035  65564.
↪5117022798  70807.51317644445  75296.31107307381  78715.53386659684  80843.
↪2542049448  81565.89943111624  80885.29954348193  78917.37941212783  75882.
↪6180075968  72089.00325653417  67908.75957917947  63750.573638897324  60029.
↪358114357325  57135.75027618665  55407.53163642614  55104.98040018357  56391.
↪84649548175  59323.19774575386  63840.86240786787  69776.63062824996  76862.
↪82012553126  84749.30249633179  93025.66347354309  101246.86238462398  108960.
↪58157342675  115734.42258276194  121181.2075790883  124980.8661331694  126897.
↪7043923571  126792.23487708672  124627.1562954154  120467.4795026559  114475.
↪16662832232  106898.95965098687  98060.30409685074  88336.4135475944  78141.
↪57067681821  67907.72827293439  58065.37365796681  49025.4713638372  41163.
↪123529992314  34803.40673028142  30209.676802382488  27574.49426375334
```

(continues on next page)

(continued from previous page)

```
26909.726097653736  30423.325835920583  35866.396706734726  43042.94385009247 ␣
↪51688.903456393586  61480.90672803479  72047.20229282859  82980.60106886801 ␣
↪93853.23752788617  104232.83864212259  113700.0628356485  121866.32784556824 ␣
↪128391.40480152355  132999.9350246486  135495.94556406536  135774.41709548625 ␣
↪133829.00722277653  129755.16115765226  123748.05046989408  116095.06120327362 ␣
↪107162.88918363019  97379.66973764922  87212.94285322673  77144.60161418792 ␣
↪67644.25993000151  59142.676367726264  52006.96118143008  46519.25848973751 ␣
↪42860.43002277699  41099.97694072242  41193.03885078976  42984.83117485243 ␣
↪46222.35826894026  50572.71021303903  55646.75793892581  61026.644661039  66295.
↪16657238937  71064.9693096191  75005.47483953922  77865.59987190687  79490.
↪62247100608  79831.97673741498  78949.27414783438  77004.42423593448  74248.
↪3119216355  71001.03812872994  67627.20140887787  64508.05459624168  62012.
↪5850032804  60469.62395377246  60142.98900215169  61211.41039342031  63754.
↪61443383615  67746.46255249016  73055.51498330054  79452.84478290801  86626.
↪41382301225  94200.87630251014  101761.32925607287  108879.30625161412  115139.
↪22185698683  120163.52094201678  123634.95786878184  125314.70588835284  125055.
↪34895563516  122808.20438790177  118624.8314305798  112652.9654404607  105127.
↪45190203677  96357.0172204928  86707.89057939101  76585.37821041526  66414.
↪49179638011  56620.657498512155  47611.39812730029  39759.70872543171  33389.
↪656945363255  28764.554512168328  26077.88185577126  25447.016793352905
27433.629854389506  32525.155768809545  39393.30524532809  47785.108159947544 ␣
↪57388.88104060634  67844.90988699778  78757.91794706834  89711.11187632111 ␣
↪100281.53073715388  110056.32610125616  118649.48611223264  125718.3945447049 ␣
↪130979.50390018395  134222.31726465162  135320.8343651857  134241.6376499674 ␣
↪131047.88420630826  125898.6325209917  119043.1657217846  110810.26362809452 ␣
↪101592.70616674965  91827.6361241704  81973.74159145917  72486.50821149157 ␣
↪63793.010231550536  56267.833334553245  50211.733884146175  45834.52988952116 ␣
↪43243.48997440569  42438.149613178466  43312.06041995375  45661.49827259699 ␣
↪49200.65554726992  53582.36092912881  58422.946224214116  63329.548887018245 ␣
↪67927.93048098436  71888.82456030193  74950.91084099437  76938.74220404879 ␣
↪77774.3111846605  77481.40640597297  76182.44140500607  74087.99710207869  71479.
↪86098062035  68688.82829738318  66068.91555218035  63969.8941782679  62710.
↪163552263104  62551.93952864113  63680.54284631807  66189.24795460189  70070.
↪72446402127  75215.60588092462  81418.19371543106  88388.79103423655  95771.
↪6975227709  103167.52222466779  110158.20637590217  116333.01318619  121313.
↪73908051326  124777.52582510584  126475.88909273194  126248.90208809178  124033.
↪85278954344  119868.09720220299  113886.22591212932  106312.017320112  97445.
↪94379741247  87649.20949737947  77325.4218548213  66901.03262367615  56805.
↪63627363837  47453.09788332404  39224.31817248714  32452.251227279412  27409.
↪592014050973  24299.364949455106  23248.486519722428  24304.25376085015
28894.854490063917  35140.9861753843  42986.623012411575  52133.76206402274  62235.
↪75149700856  72909.54126242315  83749.32126324295  94341.27604755494  104279.
↪11877967847  113179.97961947223  120700.1241734359  126549.87979819693  130507.
↪06708076141  132428.18715351087  132256.6174434319  130027.13027092877  125866.
↪17639687493  119987.56865257898  112683.45156270254  104310.73681922487  95273.
↪50096546672  86002.1556248886  76930.48468011664  68471.86981475502  60996.
↪17148653884  54808.77780671275  50133.26757819617  47098.95355339234  45734.
↪2852443521  45966.714411490604  47629.186305260126  50472.94816265635  54185.
↪89981588733  58415.2870940342  62793.19239054101  66963.03845350358  70605.
↪21322449463  73459.95735434286  75345.8324014146  76172.39551199877  75946.
↪12346676966  74769.12395277983  72830.70663985274  70392.4193451025  67767.
↪64336508463  65297.24841956803  63323.0996654926  62161.36425770375  62077.
↪57144987601  63265.23879018643  65829.60003739949  69777.58123869693  75014.
↪70191129306  81349.06631717947  88502.09594412093  96125.17827693898  103821.
↪00447570854  111168.06845334737  117746.62170305444  123164.33120171525  127079.
↪97003127275  129223.66987941816  129412.56024914619  127560.98322448067  123684.
↪87314858685  117900.29458910512  110416.50829823443  101524.25672584088  91580.
↪20770302608  80988.65474245489  70181.64101239396  59598.65565004133  49660.
↪957092593555  40783.425468976915  33298.65520674978  27503.79140288205  23620.
↪409579818206  21793.55543730138  22087.91163795822  24486.949385519467
```

(continues on next page)

(continued from previous page)

```
31578.766092709342  38542.12038262273  46901.320611567855  56325.96916278832 ␣
↪66447.69363033772  76873.59510237924  87200.66903663203  97030.87002036044 ␣
↪105986.42967051777  113724.95988127339  119953.79227674268  124442.93280477819 ␣
↪127035.962994459  127658.21101053015  126321.55826081126  123125.34862839617 ␣
↪118253.02938974372  111964.37098848473  104583.37581608328  96482.27663032949 ␣
↪88062.32098303057  79732.31378814526  71886.1199898808  64880.489035753126 ␣
↪59014.63265317082  54512.95388689601  51512.183277758464  50053.93143012392 ␣
↪50083.32936475186  51454.02098687146  53939.32485165054  57248.92919568365 ␣
↪61050.06103057276  64991.712082180544  68730.24254269019  71954.54210863516 ␣
↪74408.92153091122  75912.04108709063  76370.44781953386  75785.67287152342 ␣
↪74254.30591930171  71960.98016684038  69164.72864503061  66179.66932110766 ␣
↪63351.403590266746  61030.83632057667  59547.32024163242  59183.07798593258 ␣
↪60150.75802545569  62575.74449238924  66484.48494079297  71799.65339725232 ␣
↪78342.46435552178  85841.93631165841  93950.41110988741  102264.20501796022 ␣
↪110347.93043482488  117760.80647445051  124083.1853675202  128941.56188152573 ␣
↪132030.49636926013  133130.15068782854  132118.48452833632  128977.55696299602 ␣
↪123793.7912574417  116752.45785024593  108126.98188478891  98263.96449691789 ␣
↪87565.00529612188  76466.51950664958  65418.75783998916  54865.168850326  45223.
↪107118108026  36866.70538646292  30112.51622887128  25208.310358387676  22325.
↪21347718547  21553.186790116408  22899.71791349262  26291.493022914783
35726.00145890332  42975.523079074876  51389.08474581542  60616.72655853903  70281.
↪69140219953  79994.69331706934  89368.73277326988  98034.07935070881  105652.
↪98603591174  111933.63472295574  116642.75119180488  119616.28352371883  120767.
↪52456604429  120092.08902101486  117669.23791224741  113659.18153119108  108296.
↪18444420992  101877.53429631477  94748.7053849553  87285.32852744269  79872.
↪8472584649  72884.97202139616  66662.21478970259  61491.87582598006  57590.
↪84699439525  55092.48455837256  54038.589752451044  54377.22777555362  55966.
↪734122129725  58585.82778924403  61949.305871343706  65728.36836714271  69574.
↪25070163031  73143.55651515232  76123.51077574244  78255.31093402804  79353.
↪84919442429  79322.30844327722  78160.48327865207  75966.1211409907  72929.
↪08373734541  69318.65718017469  65464.84948257997  61734.966188317565  58507.
↪113208110895  56142.511950142325  54958.60685812881  55204.89208005549  57043.
↪1866541674  60533.76227063409  65628.30039971604  72170.16028319  79901.
↪91485285884  88479.59865470299  97492.6492305482  106488.14520036546  114997.
↪67700343495  122565.04684984924  128772.98917511763  133267.22718002342  135776.
↪42016102554  136126.88706977884  134251.38463888707  130191.64055407645  124094.
↪76008514091  116204.00718287587  106844.78203782276  96406.8570408521  85324.
↪08063610425  74052.81090760618  63050.303271879224  52754.16182600688  43563.
↪78957296455  35824.56003269969  29815.203742098383  25738.678676438452  23716.
↪591286741394  23787.06751387182  25905.8480634449  29950.30081158619
41514.281166536486  48644.43830282334  56678.02619523869  65258.066127039085 ␣
↪74012.34762036061  82568.14352517735  90567.0421802635  97679.47350597841 ␣
↪103618.45674478947  108152.04701577983  111113.91817378491  112411.50397613963 ␣
↪112031.14077904313  110039.72279988004  106582.50102260774  101876.82924001725 ␣
↪96201.87952320097  89884.60244006716  83282.47700795981  76763.85994211382 ␣
↪70686.97930411724  65378.80024172101  61115.098658805815  58103.09563809418 ␣
↪56467.921293089326  56243.989696381555  57372.08382159468  59702.58713125334 ␣
↪63004.88059267049  66982.48081331435  71293.06084566173  75572.10526252282 ␣
↪79458.63859982777  82621.25948510933  84782.63238173562  85740.6458803408  85384.
↪64093457397  83705.4331876036  80798.27877920948  76858.4319479557  72169.
↪47791323208  67085.1548940355  62005.86341332784  57351.46116076024  53532.
↪225940793396  50920.01497228941  49821.644351134164  50456.3581102356  52938.
↪964198002184  57269.80790462236  63332.26376143055  70897.89289435922  79638.
↪87540279812  89146.82731930107  98956.68609372343  108574.02764064525  117503.
↪98305705917  125279.86487698622  131489.69077946641  135798.9958775073  137968.
↪63253238593  137866.64114621872  135473.7039442754  130882.13193814585  124288.
↪74982101037  115982.40517595559  106327.11396221313  95742.04846472482  84679.
↪67004672781  73603.30903477478  62965.40759657557  53187.48457590425  44642.
↪674300407714  37641.45682539466  32420.95712040398  29137.965162917295  27865.
↪633463684848  28593.6535768753  31231.60314489877  35615.088757656325
```

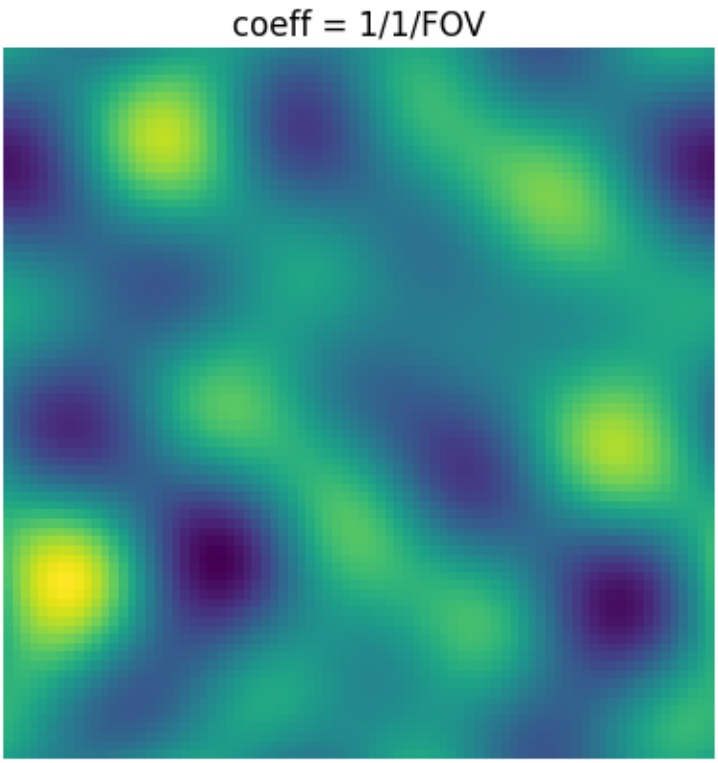
(continues on next page)

(continued from previous page)

```
49043.24959920242  55691.17142811592  62953.64832644641  70478.49174421749  77910.
↪1075112974  84904.29887621177  91142.75923642008  96346.79503998332  100289.
↪7786041339  102807.79552534463  103807.93455288462  103273.68187471096  101266.
↪93749129372  97926.27592952746  93461.22919891989  88142.57295790303  82288.
↪83752169627  76249.5280382151  70385.80266024118  65049.600654022506  60562.
↪410072536906  57194.99439350333  55149.44081759063  54544.83726925463  55407.
↪72584981279  57668.221684649245  61162.34097098055  65640.67221438934  70783.
↪07881033834  76218.67281296631  81549.88461951316  86379.1061858806  90336.
↪13711982459  93104.53761562028  94445.0046025346  94214.04184073878  92376.
↪48372289645  89010.83776226526  84306.90378384845  78555.6725240581  72132.
↪06161024749  65471.57007217454  59042.38289989157  53314.7992949506  48730.
↪06504449509  45670.74433939989  44434.66479378252  45214.21920342458  48082.
↪42820902802  52986.68906934222  59750.59433387286  68083.64193713709  77598.
↪1177363372  87831.95293022908  98275.97657898351  108403.72365875899  117701.
↪83749111983  125699.12643935284  131992.49159071187  136268.21750049392  138317.
↪4861458502  138045.40311779757  135473.2796811948  130734.35945498658  124063.
↪58174569026  115782.30804414807  106279.18427234654  95988.45801932568  85367.
↪11515237353  74872.15023323207  64939.15377859892  55963.205703386484  48282.
↪830640127584  42167.52047392034  37809.08418097302  35316.86331089672  34716.
↪66670346421  35953.13799575039  38895.175902031195  43343.97639324675
```

```
plt.figure()
plt.imshow(-2*DO)
plt.axis('off')
plt.title("coeff = 1/1/FOV")
```

```
Text(0.5, 1.0, 'coeff = 1/1/FOV')
```



CHAPTER

SIX

UTILITY APPENDIX

This is the Utility File with all of the functions put in so that all the Juypter Notebooks can use any of these functions

```
import pandas as pd
import numpy as np
import cmath
import math
from scipy.optimize import fmin, minimize
from astropy import units as u
from scipy.interpolate import RegularGridInterpolator
import matplotlib.pyplot as plt
import copy
%matplotlib inline
```

```
class data:
    u: float
    v: float
    phase: float
    amp: float
    sigma: float
    vis_data: complex
    def __init__(self, u, v, phase, amp, sigma):
        self.u = u
        self.v = v
        self.phase = phase
        self.amp = amp
        self.sigma = sigma
        self.vis_data = amp * np.exp(1j * math.radians(phase))

    def __repr__(self):
        return f"[u: {self.u}, v: {self.v}]"

    def __str__(self):
        return f"[u: {self.u}, v: {self.v}]"
```

```
def process_data(data_df):
    """
    Processes the data in the dataframe into a coords list and data objects
    Args:
        data_df is a pandas data frame of the data
    Returns:
        a list of coordinates in u,v space
        a list of data objects
```

(continues on next page)

(continued from previous page)

```
    """

    coords = []
    data_list = []
    for i in range(len(data_df)):
        data_list.append(data(data_df.loc[i, 'U(lambda)'], data_df.loc[i, 'V(lambda)
↪'], data_df.loc[i, 'Iphase(d)'], data_df.loc[i, 'Iamp(Jy)'], data_df.loc[i,
↪'Isigma(Jy)']))
        coords.append([data_df.loc[i, 'U(lambda)'], data_df.loc[i, 'V(lambda)']])
    coords = np.array(coords)
    return coords, data_list
```

```
def read_data(filename):
    """
    reads the data from a file into a pandas dataframe
    Args:
        filename is a string that represents a csv file
    Returns:
        a pandas dataframe
    """
    df = pd.read_csv(filename)
    return df
```

```
df = read_data("./data/SR1_M87_2017_095_hi_hops_netcal_StokesI.csv")
coords, data_list = process_data(df)
```

```
def loss(image, data_list: list[data], coords, p = 2, reg_weight = 1, FOV = 100*u.uas.
↪to(u.rad)):
    """
    calculates the loss of an image compared to the data given
    Args:
        image is a 80x80 pixel image that represents our reconstructed image
        coords is a list of u,v coordinates that we obtained from our data
        p is the kind of norm to be used
        reg_weight is the regularizer weight
        FOV is the Field of view from the telescopes. For the EHT data, our FOV is␣
↪100 micro ascs.
    Returns:
        a loss value
    """
    error_sum = 0
    vis_images = interpolate(image, coords, FOV)

    for i in range(len(data_list)):
        vis_data = data_list[i].amp * np.exp(1j * math.radians(data_list[i].phase))
        vis_image = vis_images[i]
        error = (abs(vis_image-vis_data) / data_list[i].sigma) ** 2
        error_sum += error

    return error_sum + reg_weight * calc_regularizer(image=image, tsv=True, p=2)
```

```
def do_sample(n):
    """
```

(continues on next page)

(continued from previous page)

```
    Collects n samples from a sample image
    Args:
        n is the number of samples as an integer
    Returns:
        a list of coordinates in u,v space
        a list of data objects
    """
    coords = []
    data_list = []
    ft_image = np.fft.fftshift(np.fft.fft2(sample))
    for i in range(n):
        coords.append((int(np.random.rand()*10-5), int(np.random.rand()*10-5)))
        data_list.append(data(coords[i][0], coords[i][1], 0, ft_
↪image[coords[i][0]+40][coords[i][1]+40], 1))
    return coords, data_list
```

```
def calculate_losses():
    """
    Calculates the losses of the image by shifting it around
    Args:
        None
    Returns:
        an array of losses where the index is how much the image is shifted starting␣
↪with -40 to 40
    """
    loss_arr = np.zeros((len(sample),len(sample[0])))
    for i in range(len(sample)):
        image_1 = np.roll(sample, i, axis=1) # Right shifts
        for j in range(len(sample[i])):
            image_2 = np.roll(image_1, j, axis = 0) # Up shifts
            loss_arr[i][j] = loss(image_2, data_list, coords, reg_weight=0, FOV=1)

    loss_arr1 = np.roll(loss_arr, 40, axis=1)
    loss_arr2 = np.roll(loss_arr1, 40, axis=0)

    plt.figure()
    plt.imshow(loss_arr2)
    plt.axis('off')
    return loss_arr
```

```
def interpolate(image, coords, FOV):
    """
    Interpolates the values of each coordinate in coords in the fourier domain of␣
↪image
    Args:
        image is a 80x80 pixel image that represents our reconstructed image
        coords is a list of u,v coordinates that we obtained from our data
        FOV is the Field of view from the telescopes. For the EHT data, our FOV is␣
↪100 micro ascs.
    Returns:
        The interpolated values at the coordinates based on image
    """

    ft_image = np.fft.fftshift(np.fft.fft2(image))
```

(continues on next page)

(continued from previous page)

```
    k_FOV = 1/FOV

    kx = np.fft.fftshift(np.fft.fftfreq(ft_image.shape[0], d = 1/(k_FOV*ft_image.
↪shape[0])))
    ky = np.fft.fftshift(np.fft.fftfreq(ft_image.shape[1], d = 1/(k_FOV*ft_image.
↪shape[1])))

    interp_real = RegularGridInterpolator((kx, ky), ft_image.real, bounds_error=False,
↪ method="linear")
    interp_imag = RegularGridInterpolator((kx, ky), ft_image.imag, bounds_error=False,
↪ method="linear")

    real = interp_real(coords)
    imag = interp_imag(coords)

    return real + imag * 1j
```

```
def calc_regularizer(image: np.array, tsv=False, p=None):
    """
    Calculates the regularizer according to total squared variation
    Args:
        image is a 80x80 pixel image that represents our reconstructed image
        p is the kind of norm to be used
        tsv is the flag for total squared variation
    Returns:
        the regularizer
    """
    if tsv and p == None:
        raise Exception("p value not set")
    reg = 0
    if tsv:
        image_lshift = np.copy(image, subok=True)
        image_lshift = np.roll(image_lshift, -1,axis=1)
        image_lshift[:,-1] = image_lshift[:,-2]
        image_upshift = np.copy(image, subok=True)
        image_upshift = np.roll(image_upshift, -1, axis=0)
        image_upshift[-1] = image_upshift[-2]

        term_1 = np.power(np.absolute(np.subtract(image_lshift, image)),p)
        term_2 = np.power(np.absolute(np.subtract(image_upshift, image)),p)
        reg = np.sum(np.add(term_1,term_2))
    return -1 * reg
```

```
def gradient_regularizer(image: np.array):
    """
    Calculates the gradient of the regularizer
    Args:
        image is a 80x80 pixel image that represents our reconstructed image
    Returns:
        the gradient of the regularizer
    """
    image_lshift = np.copy(image, subok=True)
    image_lshift = np.roll(image_lshift, -1,axis=1)
    image_lshift[:,-1] = image_lshift[:,-2]
    image_upshift = np.copy(image, subok=True)
```

(continues on next page)

(continued from previous page)

```
    image_upshift = np.roll(image_upshift, -1, axis=0)
    image_upshift[-1] = image_upshift[-2]
    image_rshift = np.copy(image, subok=True)
    image_rshift = np.roll(image_rshift, 1,axis=1)
    image_rshift[:,0] = image_rshift[:,1]
    image_dshift = np.copy(image, subok=True)
    image_dshift = np.roll(image_dshift, 1, axis=0)
    image_dshift[0] = image_lshift[1]
    g_reg = 4 * image - image_lshift - image_upshift - image_rshift - image_dshift
    return g_reg
```

```
def gradient_finite_differences(data_list: list[data], coords, image, mode = 1, FOV =␣
↪100*u.uas.to(u.rad)):
    """
    Calculates a gradient based on finite differences
    Args:
        data_list is a list of data objects
        coords is a list of u,v coordinates that we obtained from our data
        image is a 80x80 pixel image that represents our reconstructed image
        mode is the type of difference used: 0 For central, -1 for backward, 1 for␣
↪forward
        FOV is the Field of view from the telescopes. For the EHT data, our FOV is␣
↪100 micro ascs.
    Returns:
        the gradient of the loss function
    """
    image_copy = np.copy(image, subok=True)
    upper_diff: float
    lower_diff: float
    h: float
    gradient_arr = np.empty(np.shape(image),dtype=np.complex_)
    if (mode == 0): # Central difference
        for row in range(len(image)):
            for col in range(len(image[row])):
                image_copy[row,col] += 1e-6 / 2
                upper_diff = loss(image_copy, data_list, coords, FOV=FOV)
                image_copy[row,col] -= 1e-6
                lower_diff = loss(image_copy, data_list, coords, FOV=FOV)
                image_copy[row,col] = image[row,col] # Reset that pixel to original␣
↪value
                gradient_arr[row,col] = (upper_diff - lower_diff) / 1e-6
    elif (mode == -1): # Backward difference
        upper_diff = loss(image, data_list, coords, FOV=FOV)
        for row in range(len(image)):
            for col in range(len(image[row])):
                image_copy[row,col] -= 1e-8
                lower_diff = loss(image_copy, data_list, coords, FOV=FOV)
                gradient_arr[row,col] = (upper_diff - lower_diff) / 1e-8
                image_copy[row,col] = image[row,col]
    elif (mode == 1) : # Forward difference is default
        lower_diff = loss(image, data_list, coords, FOV=FOV)
        for row in range(len(image)):
            for col in range(len(image[row])):
                image_copy[row,col] += 1e-8
                upper_diff = loss(image_copy, data_list, coords, FOV=FOV)
```

(continues on next page)

(continued from previous page)

```
                gradient_arr[row,col] = (upper_diff - lower_diff) / 1e-8
                image_copy[row,col] = image[row,col]
    else:
        raise ValueError('Incorrect mode for finite differences')
    return gradient_arr.real
```

```
def gradient_descent(image, data_list, coords, coeffs = None, FOV = 100*u.uas.to(u.
↪rad), stopper = None, dirty = False):
    """
    Performs gradient descent to reconstruct the image
    Args:
        image is a 80x80 pixel image that represents our reconstructed image
        data_list is a list of data objects
        coords is a list of u,v coordinates that we obtained from our data
        coeffs is the precomputed coefficients
        FOV is the Field of view from the telescopes. For the EHT data, our FOV is␣
↪100 micro ascs.
        stopper allows the descent to stop at 20 iterations
        dirty changes the gradient mode to dirty kernel
    Returns:
        the reconstructed image
    """
    image_copy = np.copy(image, subok=True) # Uses copy of the image due to lists␣
↪being mutable in python
    i = 0
    grad = None
    # Can also use max here, min just makes it finish quicker
    while grad is None or np.min(np.abs(grad)) > 0.00001:
        t = 10000000 # Initial Step size which resets each iteration
        prev_loss = loss(image_copy, data_list, coords, FOV=FOV)

        if dirty:
            grad = dirty_gradient(data_list, coords, coeffs, image_copy)
        else:
            grad = gradient_finite_differences(data_list, coords, image_copy, FOV=FOV)

        new_image = image_copy - t * grad.real
        new_loss = loss(new_image, data_list, coords, FOV=FOV)

        while new_loss > prev_loss: # Only run when new_loss > prev_loss
            new_image = image_copy - t * grad.real
            new_loss = loss(new_image, data_list, coords, FOV=FOV)
            t /= 2

        image_copy -= t * 2 * grad.real # Multiply by 2 to undo last divide in the␣
↪while loop
        i += 1
        if stopper != None:
            if i == stopper: # Hard stop here for notebook purposes
                return image_copy
        print("loss:",new_loss)
    return image_copy
```

```
def preprocess_gradient(data_list, coords, image):
    """
```

(continues on next page)

(continued from previous page)

```
    Precomputed the coefficients decribed in dirty gradient
    Args:
        data_list is a list of data objects
        coords is a list of u,v coordinates that we obtained from our data
        image is a 80x80 pixel image that represents our reconstructed image
    Returns:
        a 4d list of coefficients
    """
    r, c = np.shape(image)
    preprocessed = np.empty([r,c,len(data_list),2], dtype=np.complex_)
    for row in range(len(image)):
        for col in range(len(image[row])):
            for datum in range(len(data_list)):
                term = ((2*np.pi*1j)/image.size)*(row*coords[datum][0] +␣
↪col*coords[datum][1]) #.size for numpy array returns # of rows * # of cols
                term_1 = np.exp(term)
                term_2 = np.exp(-1*term)
                preprocessed[row,col,datum,0] = term_1
                preprocessed[row,col,datum,1] = term_2
    return preprocessed
```

```
def dirty_gradient(data_list: list[data], coords, coeffs, image, FOV = 100*u.uas.to(u.
↪rad)):
    """
    Calculates a gradient based on dirting the image
    Args:
        data_list is a list of data objects
        coords is a list of u,v coordinates that we obtained from our data
        coeffs is the precomputed coefficients
        image is a 80x80 pixel image that represents our reconstructed image
        FOV is the Field of view from the telescopes. For the EHT data, our FOV is␣
↪100 micro ascs.
    Returns:
        the gradient of the loss function
    """
    gradient_arr = np.empty(np.shape(image)) # Because we are in real space
    vis_images = interpolate(image, coords, FOV)
    for row in range(len(image)):
        for col in range(len(image[row])):
            gradient_sum = 0
            for i in range(len(data_list)):
                vis_data = data_list[i].vis_data
                vis_image = vis_images[i] # Ask about this on Wednesday
                term_1 = coeffs[row,col,i,0] * (np.conj(vis_image) - np.conj(vis_
↪data))
                term_2 = coeffs[row,col,i,1] * (vis_image - vis_data)
                gradient_sum += (term_1 + term_2)/(data_list[i].sigma ** 2)
            gradient_arr[row,col] = gradient_sum.real
    return gradient_arr
```