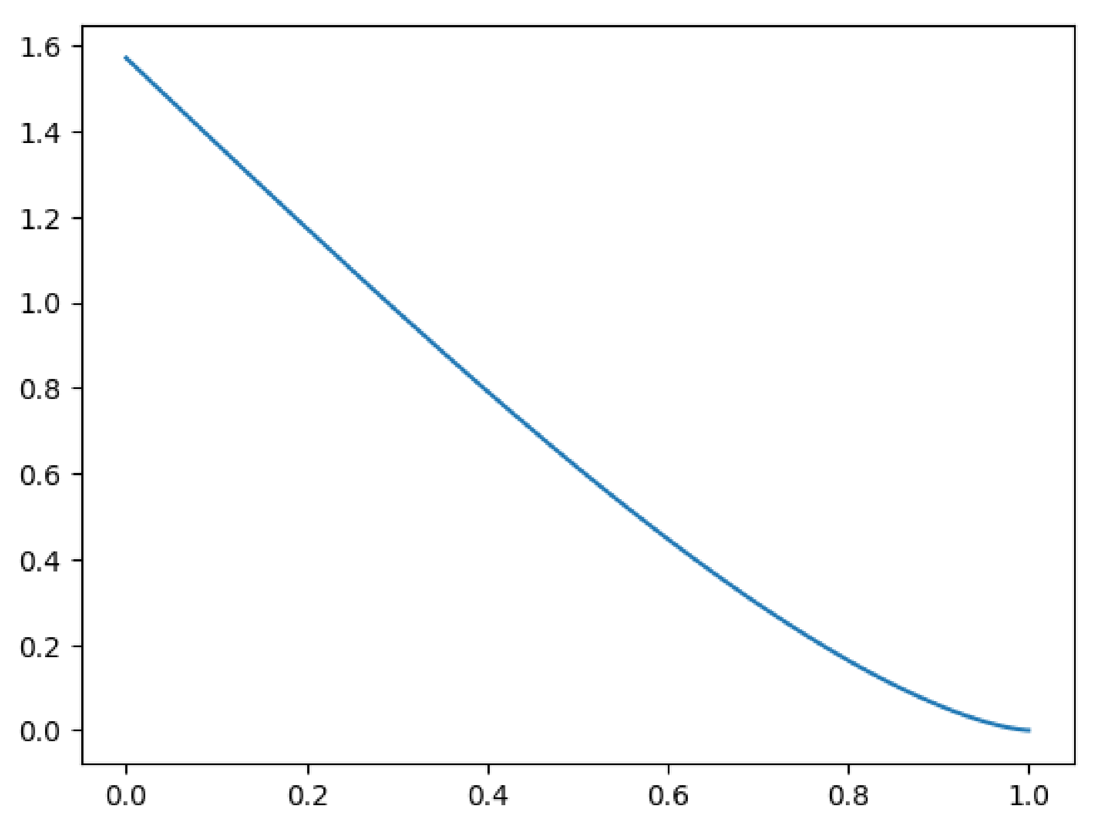
In [1]:
```
import numpy as np
import imtools as im
import matplotlib.pyplot as plt

%matplotlib inline
```

TP 1 - Exercice 2

In [2]:
```
# Visualize radial profile
def gamma(_r):
    return np.arccos(_r) - _r * np.sqrt(1 - _r**2)
```

In [3]:
```
r = np.linspace(0, 1, 400)
plt.plot(r, gamma(r))
```

Out[3]:
```
[<matplotlib.lines.Line2D at 0x7f4b0a453fd0>]
```



Exercice 4

In [4]:
```
room = im.load('room.pgm').astype('double')
im.View(room)
```

Out[4]:
```
<imtools.View at 0x7f4b0839ecd0>
```

1.

In [5]:
```
step = 5
v = room[::step, ::step]  # Sub-sample room
# Enlarge pixels of the subsampled image, so that it reaches the same
# size (mod step) as the original room
w = np.kron(v, np.ones(shape=(step, step)))
ny, nx = room.shape
```

In [6]:
```
im.View(np.hstack((room, w[:ny, :nx])))  # Display images side by side
im.View(w)
```

Out[6]:
```
<imtools.View at 0x7f4b083ce490>
```

2.

Le phénomène observé est de l'aliasing.

On observe notamment les lignes de l'image se dégrader : le fil en travers de l'image perd sa continuité ($(x, y) = (71, 72)$ sur l'image w), les tranches des livres deviennent indistinguables. Les rayures de la serviette, proches d'une onde pure, font apparaître un nouveau motif de la forme d'une onde pure mais de vecteur d'onde complètement différent ($(x, y) = (175, 386)$). Les contours sont crénelés ($(x, y) = (34, 51)$)

3.

In [7]:
```
f = np.zeros((512, 512))
f[189, 49] = 2
onde = np.real(np.fft.ifft2(f))
```

In [8]:
```
im.View(onde)
```

Out[8]:
```
<imtools.View at 0x7f4b0843cac0>
```

In [9]:
```
mod_fft_onde = np.abs(np.fft.fftshift(np.fft.fft2(onde)))
im.View(mod_fft_onde)
```

Out[9]:
```
<imtools.View at 0x7f4b0843c850>
```

In [10]:
```
onde_subsampled = onde[::2, ::2]
im.View(onde_subsampled)
```

Out[10]:
```
<imtools.View at 0x7f4b083b83d0>
```

In [11]:
```
mod_fft_onde_s = np.abs(np.fft.fftshift(np.fft.fft2(onde_subsampled)))
im.View(mod_fft_onde_s)
```

Out[11]:
```
<imtools.View at 0x7f4b0a4c4fd0>
```

Les coordonnées des pics dans le domaine de Fourier sont :

- Pour l'onde de base, sur un carré de 512×512 :
  - en indices : $(305, 445)$ et son symétrique par rapport au centre $(207, 67)$
  - en coordonnées de Fourier (fftshift) : $(49, 189)$ et son symétrique $(-49, -189)$
- Pour l'onde sous-échantillonée, sur un carré de 256×256 :
  - en indices : $(177, 61)$ et son symétrique $(79, 195)$
  - après fftshift : $(-49, 67)$ et son symétrique $(49, -67)$

La raison pour laquelle le sous-échantillonage provoque un phénomène d'aliasing est que les coordonnées du vecteur d'onde initial excèdent la taille de la fenêtre de calcul utilisée pour la transformée de Fourier de la deuxième image : les coordonnées supérieures à 128 sont aliasées, c'est-à-dire que le pic de la transformée de Fourier apparaîtra à la valeur modulo 256 comprise entre -128 et 128. Ainsi, le vecteur d'onde $(49, 189)$ est aliasé au vecteur d'onde $(49, 189 - 256) = (49, -67)$, et de façon équivalente son symétrique est aliasé à $(-49, 67)$. L'onde est ainsi complètement transformée par ce sous-échantillonage.

Exercice 5

1.

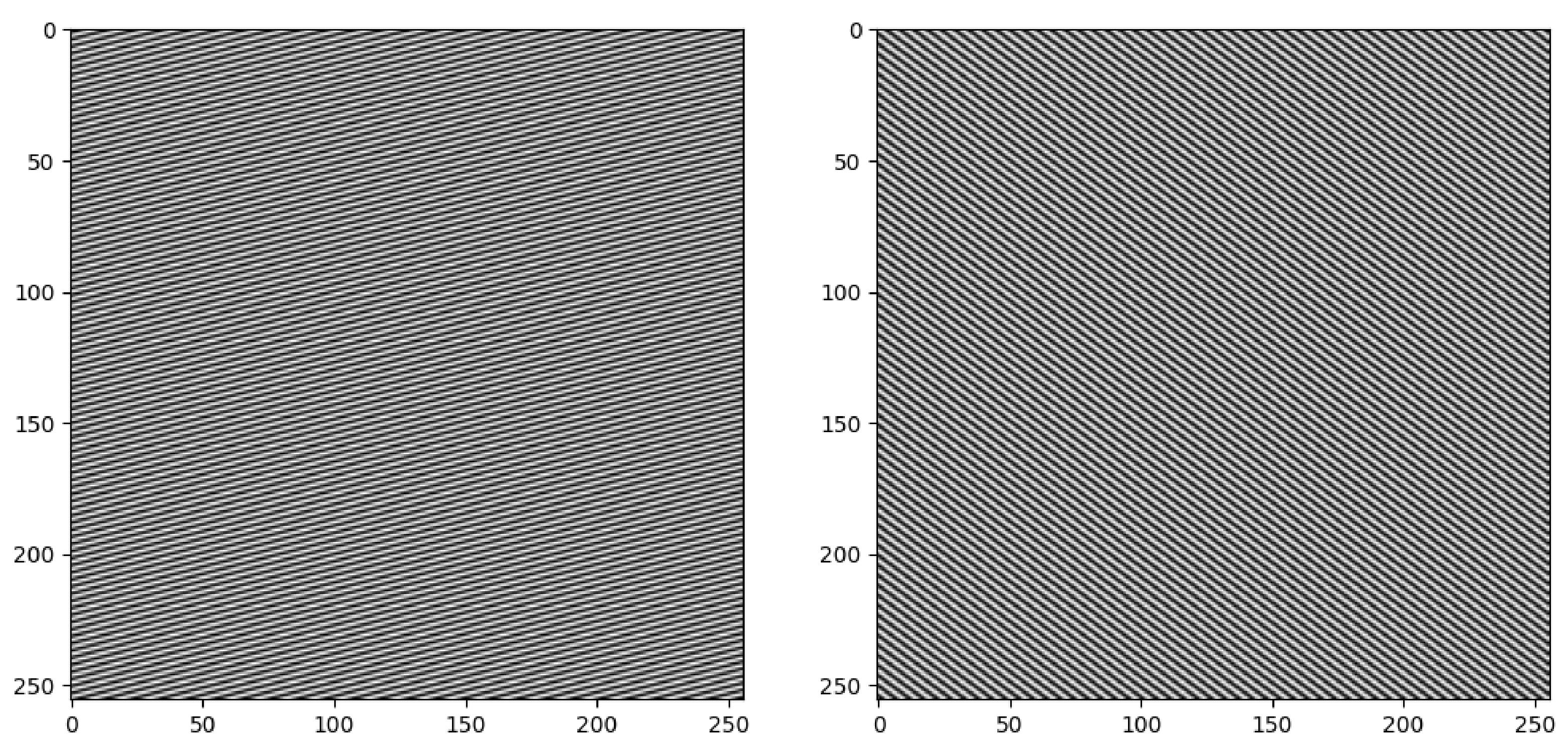
In [12]:
```
onde_sqr = onde ** 2
# Normalize the waves, for plotting,
# otherwise onde_sqr is invisible because it takes very small values
ondes_nrm = np.hstack((onde / np.max(onde), onde_sqr / np.max(onde_sqr)))
```

In [13]:
```
im.View(ondes_nrm)
```

Out[13]:
```
<imtools.View at 0x7f4b083b8b50>
```

In [14]:
```
_, axes = plt.subplots(1, 2, figsize=(12, 6))
axes[0].imshow(onde[:len(onde)//2, :len(onde)//2], cmap='gray')  # Slice to zoom in
axes[1].imshow(onde_sqr[:len(onde)//2, :len(onde)//2], cmap='gray')
```

Out[14]:
```
<matplotlib.image.AxesImage at 0x7f4b07721fa0>
```



L'orientation du vecteur d'onde a changé. On s'attend à un doublement de la fréquence de l'onde initiale, car mettre au carré un onde sinusoïdale revient à "replier" la partie négative de $sin\left(\langle k, x\rangle\right)$ vers des valeurs positives (i.e. "prendre la valeur absolue"). La valeur moyenne de l'onde est alors strictement positive, et la fréquence est doublée (on peut s'en convaincre à l'aide de l'identité trigonométrique $sin^2(x) = \frac{1}{2}(1 - cos(2x))$)

Or, si la fréquence double, il est possible qu'on observe de l'aliasing : passant d'un vecteur d'onde k à $2k$, l'onde obtenue peut ne pas respecter les conditions de Shannon. Ici, le vecteur d'onde est $2k = 2 \cdot (49, 189) = (98, 378)$. On voit bien que ce vecteur est en-dehors du carré de Shannon situé entre $(-256, -256)$ et $(256, 256)$. Le vecteur d'onde $2k$ sera donc replié après aliasing en un vecteur $\hat{k} = (98, 378 - 512) = (98, -134)$.

On peut vérifier ceci à l'aide de la transformée de Fourier de la nouvelle onde obtenue :

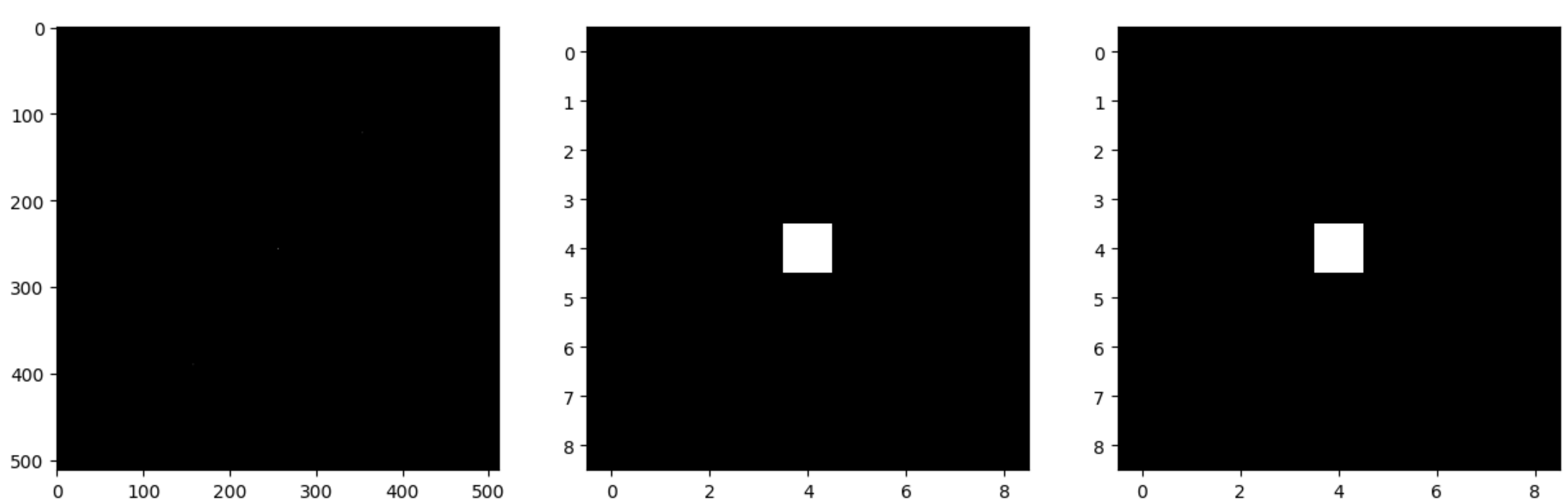
In [15]:
```
onde_sqr_fft = np.abs(np.fft.fftshift(np.fft.fft2(onde_sqr)))
```

In [16]:
```
im.View(onde_sqr_fft)
```

Out[16]:
```
<imtools.View at 0x7f4b07697d60>
```

In [17]:
```
_, axes = plt.subplots(1, 3, figsize=(15, 5))
axes[0].imshow(onde_sqr_fft, cmap='gray')
axes[1].imshow(onde_sqr_fft[118:127, 350:359], cmap='gray')
axes[2].imshow(onde_sqr_fft[386:395, 154:163], cmap='gray')
```

Out[17]:
```
<matplotlib.image.AxesImage at 0x7f4b05db6760>
```



Les pics d'intensité étant quasiment invisibles sur le graphe complet de la FFT, on zoome sur les pics anticipés, aux indices translatés du fait de l'opération fftshift : $(98 + 256, -134 + 256) = (354, 122)$ et $(-98 + 256, 134 + 256) = (158, 390)$. On observe par ailleurs un pic centré en $(0, 0)$ dans les coordonnées de Fourier, qui correspond au terme constant de l'onde.

Les résultats sont donc tels qu'attendus : repliement du vecteur d'onde dans la fenêtre de Fourier.

2.

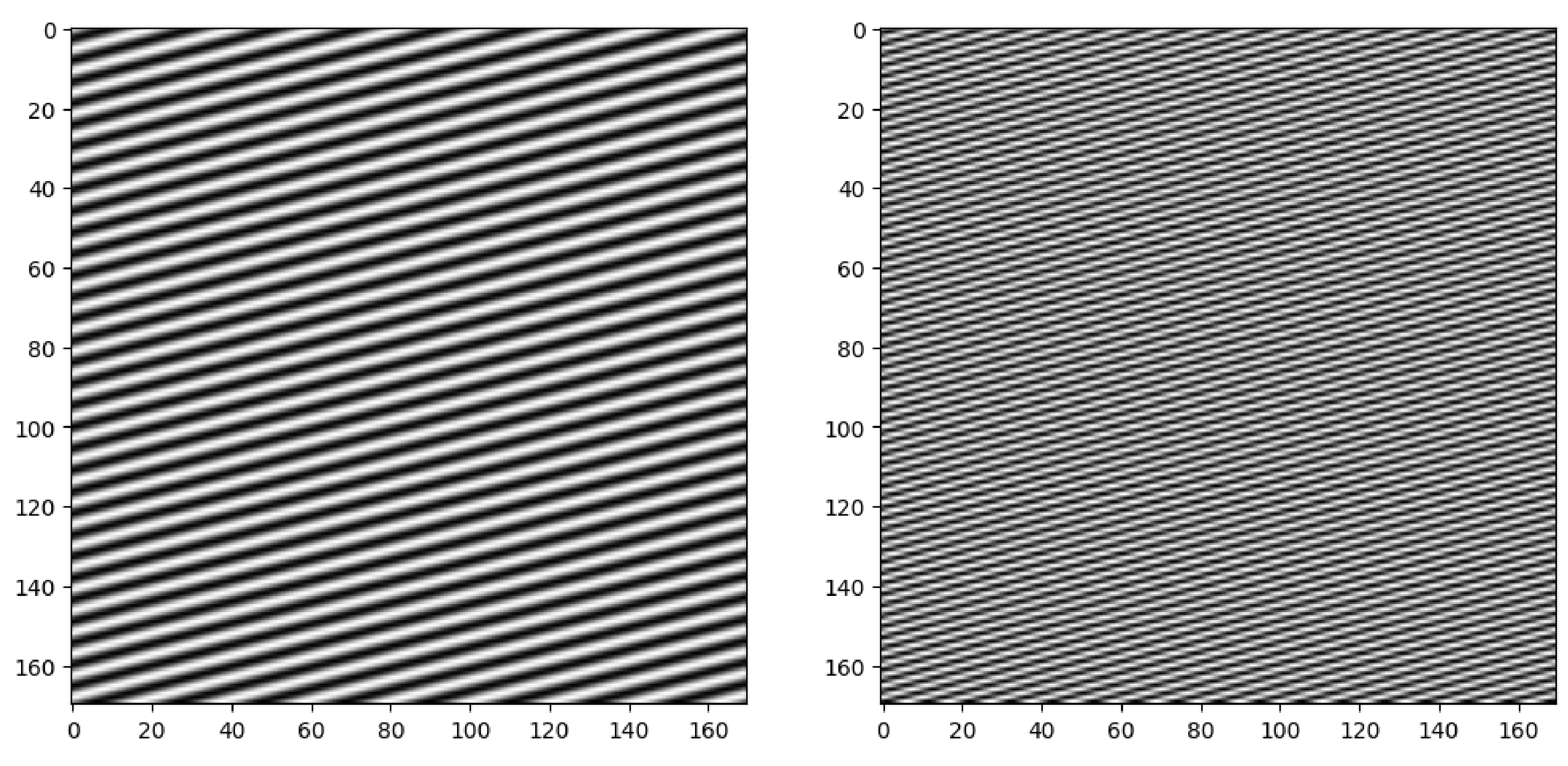
In [18]:
```
onde_z = im.fftzoom(onde, 2)
onde_z_sqr = onde_z ** 2
ondes_z_nrm = np.hstack((onde_z / np.max(onde_z), onde_z_sqr / np.max(onde_z_sqr)))
```

In [19]:
```
im.View(ondes_z_nrm)
```

Out[19]:
```
<imtools.View at 0x7f4b05d03520>
```

In [20]:
```
_, axes = plt.subplots(1, 2, figsize=(12, 6))
axes[0].imshow(onde_z[:len(onde_z)//6, :len(onde_z)//6], cmap='gray')
axes[1].imshow(onde_z_sqr[:len(onde_z)//6, :len(onde_z)//6], cmap='gray')
```

Out[20]:
```
<matplotlib.image.AxesImage at 0x7f4b05d0bf70>
```



Le phénomène d'aliasing ne se reproduit plus : le carré de Shannon s'étend maintenant de $(-512, -512)$ à $(512, 512)$, et contient donc le vecteur $2k$. On observe donc l'onde attendue, c'est-à-dire l'onde originale avec une fréquence deux fois plus élevée

3.

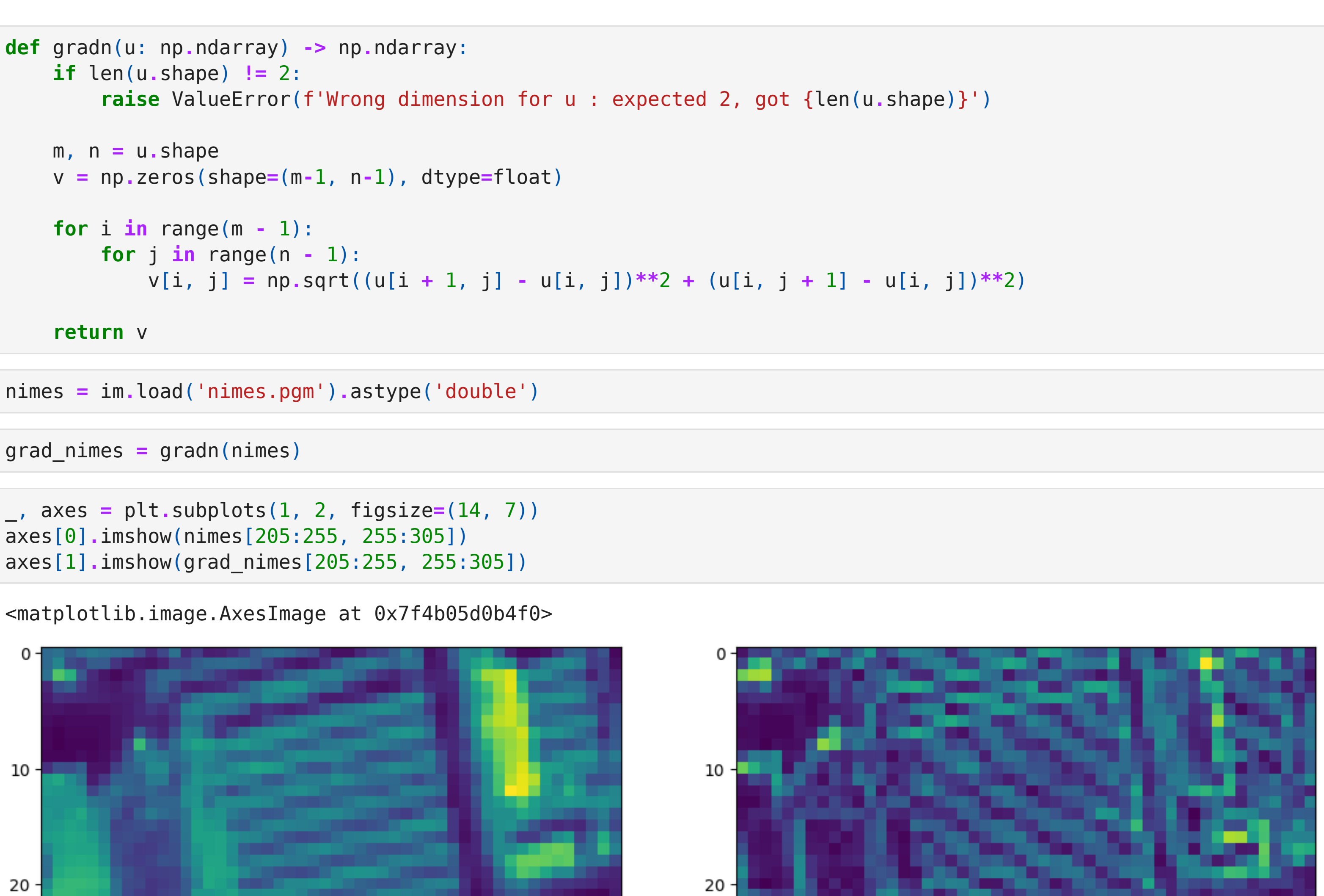
In [21]:
```
def gradn(u: np.ndarray) -> np.ndarray:
    if len(u.shape) != 2:
        raise ValueError(f'Wrong dimension for u : expected 2, got {len(u.shape)}')

    m, n = u.shape
    v = np.zeros(shape=(m-1, n-1), dtype=float)

    for i in range(m - 1):
        for j in range(n - 1):
            v[i, j] = np.sqrt((u[i + 1, j] - u[i, j])**2 + (u[i, j + 1] - u[i, j])**2)

    return v
```

In [22]:
```
nimes = im.load('nimes.pgm').astype('double')
```

In [23]:
```
grad_nimes = gradn(nimes)
```

In [24]:
```
_, axes = plt.subplots(1, 2, figsize=(14, 7))
axes[0].imshow(nimes[205:255, 255:305])
axes[1].imshow(grad_nimes[205:255, 255:305])
```

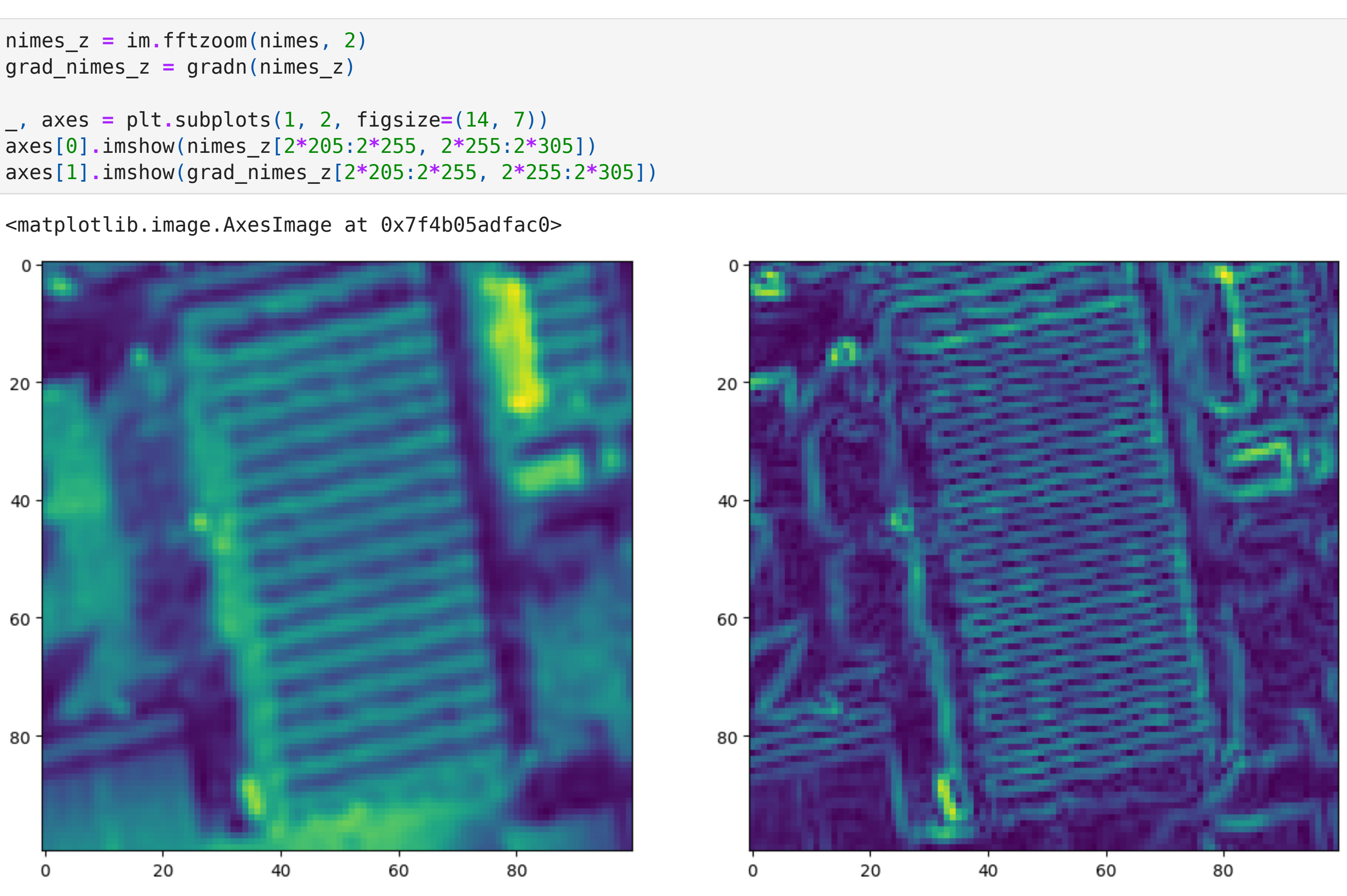
Out[24]:
```
<matplotlib.image.AxesImage at 0x7f4b05d0b4f0>
```



On observe de l'aliasing au niveau d'une portion de l'image ressemblant à une onde pure, car on considère dans le calcul du gradient le carré de l'image. La fréquence de cette onde au carré est certainement repliée dans le domaine spectral. Pour bien calculer le gradient, on peut procéder à un zoom par zéro-padding sur l'image originale.

In [25]:
```
nimes_z = im.fftzoom(nimes, 2)
grad_nimes_z = gradn(nimes_z)

_, axes = plt.subplots(1, 2, figsize=(14, 7))
axes[0].imshow(nimes_z[2*205:2*255, 2*255:2*305])
axes[1].imshow(grad_nimes_z[2*205:2*255, 2*255:2*305])
```

Out[25]:
```
<matplotlib.image.AxesImage at 0x7f4b05adfac0>
```



In [ ]:
```
```