Nombre: WIlson Rodas

Prueba 2: Generación de números pseudo aleatorios

In [91]:
```
#Importación de los paquetes
import math
import matplotlib.pyplot as pp

%matplotlib inline
```

1. Generación de números aleatorios

A) Método de medios cuadrados

In [140…
```
#Generamos los números aleatorios por medios cuadrados
numeros_aleatorios1 = []

xn = '74731897457'
iteraciones = 100
total_digitos = 7

for i in range(iteraciones):
    xn_pow = str(int(math.pow(int(xn), 2)))
    longitud = len(xn_pow)
    centro = math.ceil(longitud / 2)

    if total_digitos % 2 == 0:
        porcion_izquierda = int(total_digitos / 2)
        porcion_derecha = int(total_digitos / 2)
    else:
        porcion_izquierda = math.floor(total_digitos / 2)
        porcion_derecha = int(total_digitos - porcion_izquierda)

    aux = xn_pow[centro - porcion_izquierda:centro] + xn_pow[centro:centro + porcion_derecha]
    ui = int(aux) / math.pow(10, total_digitos)

    print(f'I: {i + 1}, Xn: {xn}, Xn**2: {xn_pow}, Longitud: {longitud}, Semilla: {aux}, Ui: {ui}')

    xn = aux
    numeros_aleatorios1.append(ui)
```

```
    if aux == 0.0:
        break
```

```
I: 1, Xn: 74731897457, Xn**2: 5584856497523563429888, Longitud: 22, Semilla: 9752356, Ui: 0.9752356
I: 2, Xn: 9752356, Xn**2: 95108447550736, Longitud: 14, Semilla: 8447550, Ui: 0.844755
I: 3, Xn: 8447550, Xn**2: 71361101002500, Longitud: 14, Semilla: 1101002, Ui: 0.1101002
I: 4, Xn: 1101002, Xn**2: 1212205404004, Longitud: 13, Semilla: 2054040, Ui: 0.205404
I: 5, Xn: 2054040, Xn**2: 4219080321600, Longitud: 13, Semilla: 0803216, Ui: 0.0803216
I: 6, Xn: 0803216, Xn**2: 645155942656, Longitud: 12, Semilla: 1559426, Ui: 0.1559426
I: 7, Xn: 1559426, Xn**2: 2431809449476, Longitud: 13, Semilla: 8094494, Ui: 0.8094494
I: 8, Xn: 8094494, Xn**2: 65520833116036, Longitud: 14, Semilla: 0833116, Ui: 0.0833116
I: 9, Xn: 0833116, Xn**2: 694082269456, Longitud: 12, Semilla: 0822694, Ui: 0.0822694
I: 10, Xn: 0822694, Xn**2: 676825417636, Longitud: 12, Semilla: 8254176, Ui: 0.8254176
I: 11, Xn: 8254176, Xn**2: 68131421438976, Longitud: 14, Semilla: 1421438, Ui: 0.1421438
I: 12, Xn: 1421438, Xn**2: 2020485987844, Longitud: 13, Semilla: 4859878, Ui: 0.4859878
I: 13, Xn: 4859878, Xn**2: 23618414174884, Longitud: 14, Semilla: 8414174, Ui: 0.8414174
I: 14, Xn: 8414174, Xn**2: 70798324102276, Longitud: 14, Semilla: 8324102, Ui: 0.8324102
I: 15, Xn: 8324102, Xn**2: 69290674106404, Longitud: 14, Semilla: 0674106, Ui: 0.0674106
I: 16, Xn: 0674106, Xn**2: 454418899236, Longitud: 12, Semilla: 4188992, Ui: 0.4188992
I: 17, Xn: 4188992, Xn**2: 17547653976064, Longitud: 14, Semilla: 7653976, Ui: 0.7653976
I: 18, Xn: 7653976, Xn**2: 58583348608576, Longitud: 14, Semilla: 3348608, Ui: 0.3348608
I: 19, Xn: 3348608, Xn**2: 11213175537664, Longitud: 14, Semilla: 3175537, Ui: 0.3175537
I: 20, Xn: 3175537, Xn**2: 10084035238369, Longitud: 14, Semilla: 4035238, Ui: 0.4035238
I: 21, Xn: 4035238, Xn**2: 16283145716644, Longitud: 14, Semilla: 3145716, Ui: 0.3145716
I: 22, Xn: 3145716, Xn**2: 9895529152656, Longitud: 13, Semilla: 5291526, Ui: 0.5291526
I: 23, Xn: 5291526, Xn**2: 28000247408676, Longitud: 14, Semilla: 0247408, Ui: 0.0247408
I: 24, Xn: 0247408, Xn**2: 61210718464, Longitud: 11, Semilla: 1071846, Ui: 0.1071846
I: 25, Xn: 1071846, Xn**2: 1148853847716, Longitud: 13, Semilla: 8538477, Ui: 0.8538477
I: 26, Xn: 8538477, Xn**2: 72905589479529, Longitud: 14, Semilla: 5589479, Ui: 0.5589479
I: 27, Xn: 5589479, Xn**2: 31242275491441, Longitud: 14, Semilla: 2275491, Ui: 0.2275491
I: 28, Xn: 2275491, Xn**2: 5177859291081, Longitud: 13, Semilla: 8592910, Ui: 0.859291
I: 29, Xn: 8592910, Xn**2: 73838102268100, Longitud: 14, Semilla: 8102268, Ui: 0.8102268
I: 30, Xn: 8102268, Xn**2: 65646746743824, Longitud: 14, Semilla: 6746743, Ui: 0.6746743
I: 31, Xn: 6746743, Xn**2: 45518541108049, Longitud: 14, Semilla: 8541108, Ui: 0.8541108
I: 32, Xn: 8541108, Xn**2: 72950525867664, Longitud: 14, Semilla: 0525867, Ui: 0.0525867
I: 33, Xn: 0525867, Xn**2: 276536101689, Longitud: 12, Semilla: 5361016, Ui: 0.5361016
I: 34, Xn: 5361016, Xn**2: 28740492552256, Longitud: 14, Semilla: 0492552, Ui: 0.0492552
I: 35, Xn: 0492552, Xn**2: 242607472704, Longitud: 12, Semilla: 6074727, Ui: 0.6074727
I: 36, Xn: 6074727, Xn**2: 36902308124529, Longitud: 14, Semilla: 2308124, Ui: 0.2308124
I: 37, Xn: 2308124, Xn**2: 5327436399376, Longitud: 13, Semilla: 4363993, Ui: 0.4363993
I: 38, Xn: 4363993, Xn**2: 19044434904049, Longitud: 14, Semilla: 4434904, Ui: 0.4434904
I: 39, Xn: 4434904, Xn**2: 19668373489216, Longitud: 14, Semilla: 8373489, Ui: 0.8373489
I: 40, Xn: 8373489, Xn**2: 70115318033121, Longitud: 14, Semilla: 5318033, Ui: 0.5318033
I: 41, Xn: 5318033, Xn**2: 28281474989089, Longitud: 14, Semilla: 1474989, Ui: 0.1474989
I: 42, Xn: 1474989, Xn**2: 2175592550121, Longitud: 13, Semilla: 5925501, Ui: 0.5925501
I: 43, Xn: 5925501, Xn**2: 35111562101001, Longitud: 14, Semilla: 1562101, Ui: 0.1562101
```

```
I: 44, Xn: 1562101, Xn**2: 2440159534201, Longitud: 13, Semilla: 1595342, Ui: 0.1595342
I: 45, Xn: 1595342, Xn**2: 2545116096964, Longitud: 13, Semilla: 1160969, Ui: 0.1160969
I: 46, Xn: 1160969, Xn**2: 1347849018961, Longitud: 13, Semilla: 8490189, Ui: 0.8490189
I: 47, Xn: 8490189, Xn**2: 72083309255721, Longitud: 14, Semilla: 3309255, Ui: 0.3309255
I: 48, Xn: 3309255, Xn**2: 10951168655025, Longitud: 14, Semilla: 1168655, Ui: 0.1168655
I: 49, Xn: 1168655, Xn**2: 1365754509025, Longitud: 13, Semilla: 7545090, Ui: 0.754509
I: 50, Xn: 7545090, Xn**2: 56928383108100, Longitud: 14, Semilla: 8383108, Ui: 0.8383108
I: 51, Xn: 8383108, Xn**2: 70276499739664, Longitud: 14, Semilla: 6499739, Ui: 0.6499739
I: 52, Xn: 6499739, Xn**2: 42246607068121, Longitud: 14, Semilla: 6607068, Ui: 0.6607068
I: 53, Xn: 6607068, Xn**2: 43653347556624, Longitud: 14, Semilla: 3347556, Ui: 0.3347556
I: 54, Xn: 3347556, Xn**2: 11206131173136, Longitud: 14, Semilla: 6131173, Ui: 0.6131173
I: 55, Xn: 6131173, Xn**2: 37591282355929, Longitud: 14, Semilla: 1282355, Ui: 0.1282355
I: 56, Xn: 1282355, Xn**2: 1644434346025, Longitud: 13, Semilla: 4343460, Ui: 0.434346
I: 57, Xn: 4343460, Xn**2: 18865644771600, Longitud: 14, Semilla: 5644771, Ui: 0.5644771
I: 58, Xn: 5644771, Xn**2: 31863439642441, Longitud: 14, Semilla: 3439642, Ui: 0.3439642
I: 59, Xn: 3439642, Xn**2: 11831137088164, Longitud: 14, Semilla: 1137088, Ui: 0.1137088
I: 60, Xn: 1137088, Xn**2: 1292969119744, Longitud: 13, Semilla: 9691197, Ui: 0.9691197
I: 61, Xn: 9691197, Xn**2: 93919299292809, Longitud: 14, Semilla: 9299292, Ui: 0.9299292
I: 62, Xn: 9299292, Xn**2: 86476831701264, Longitud: 14, Semilla: 6831701, Ui: 0.6831701
I: 63, Xn: 6831701, Xn**2: 46672138553401, Longitud: 14, Semilla: 2138553, Ui: 0.2138553
I: 64, Xn: 2138553, Xn**2: 4573408933809, Longitud: 13, Semilla: 4089338, Ui: 0.4089338
I: 65, Xn: 4089338, Xn**2: 16722685278244, Longitud: 14, Semilla: 2685278, Ui: 0.2685278
I: 66, Xn: 2685278, Xn**2: 7210717937284, Longitud: 13, Semilla: 7179372, Ui: 0.7179372
I: 67, Xn: 7179372, Xn**2: 51543382314384, Longitud: 14, Semilla: 3382314, Ui: 0.3382314
I: 68, Xn: 3382314, Xn**2: 11440047994596, Longitud: 14, Semilla: 0047994, Ui: 0.0047994
I: 69, Xn: 0047994, Xn**2: 2303424036, Longitud: 10, Semilla: 0342403, Ui: 0.0342403
I: 70, Xn: 0342403, Xn**2: 117239814409, Longitud: 12, Semilla: 2398144, Ui: 0.2398144
I: 71, Xn: 2398144, Xn**2: 5751094644736, Longitud: 13, Semilla: 0946447, Ui: 0.0946447
I: 72, Xn: 0946447, Xn**2: 895761923809, Longitud: 12, Semilla: 7619238, Ui: 0.7619238
I: 73, Xn: 7619238, Xn**2: 58052787700644, Longitud: 14, Semilla: 2787700, Ui: 0.27877
I: 74, Xn: 2787700, Xn**2: 7771271290000, Longitud: 13, Semilla: 2712900, Ui: 0.27129
I: 75, Xn: 2712900, Xn**2: 7359826410000, Longitud: 13, Semilla: 8264100, Ui: 0.82641
I: 76, Xn: 8264100, Xn**2: 68295348810000, Longitud: 14, Semilla: 5348810, Ui: 0.534881
I: 77, Xn: 5348810, Xn**2: 28609768416100, Longitud: 14, Semilla: 9768416, Ui: 0.9768416
I: 78, Xn: 9768416, Xn**2: 95421951149056, Longitud: 14, Semilla: 1951149, Ui: 0.1951149
I: 79, Xn: 1951149, Xn**2: 3806982420201, Longitud: 13, Semilla: 9824202, Ui: 0.9824202
I: 80, Xn: 9824202, Xn**2: 96514944936804, Longitud: 14, Semilla: 4944936, Ui: 0.4944936
I: 81, Xn: 4944936, Xn**2: 24452392044096, Longitud: 14, Semilla: 2392044, Ui: 0.2392044
I: 82, Xn: 2392044, Xn**2: 5721874497936, Longitud: 13, Semilla: 8744979, Ui: 0.8744979
I: 83, Xn: 8744979, Xn**2: 76474657710441, Longitud: 14, Semilla: 4657710, Ui: 0.465771
I: 84, Xn: 4657710, Xn**2: 21694262444100, Longitud: 14, Semilla: 4262444, Ui: 0.4262444
I: 85, Xn: 4262444, Xn**2: 18168428853136, Longitud: 14, Semilla: 8428853, Ui: 0.8428853
I: 86, Xn: 8428853, Xn**2: 71045562895609, Longitud: 14, Semilla: 5562895, Ui: 0.5562895
I: 87, Xn: 5562895, Xn**2: 30945800781025, Longitud: 14, Semilla: 5800781, Ui: 0.5800781
I: 88, Xn: 5800781, Xn**2: 33649060209961, Longitud: 14, Semilla: 9060209, Ui: 0.9060209
I: 89, Xn: 9060209, Xn**2: 82087387123681, Longitud: 14, Semilla: 7387123, Ui: 0.7387123
I: 90, Xn: 7387123, Xn**2: 54569586217129, Longitud: 14, Semilla: 9586217, Ui: 0.9586217
```

```
I: 91, Xn: 9586217, Xn**2: 91895556371089, Longitud: 14, Semilla: 5556371, Ui: 0.5556371
I: 92, Xn: 5556371, Xn**2: 30873258689641, Longitud: 14, Semilla: 3258689, Ui: 0.3258689
I: 93, Xn: 3258689, Xn**2: 10619053998721, Longitud: 14, Semilla: 9053998, Ui: 0.9053998
I: 94, Xn: 9053998, Xn**2: 81974879784004, Longitud: 14, Semilla: 4879784, Ui: 0.4879784
I: 95, Xn: 4879784, Xn**2: 23812291886656, Longitud: 14, Semilla: 2291886, Ui: 0.2291886
I: 96, Xn: 2291886, Xn**2: 5252741436996, Longitud: 13, Semilla: 7414369, Ui: 0.7414369
I: 97, Xn: 7414369, Xn**2: 54972867668161, Longitud: 14, Semilla: 2867668, Ui: 0.2867668
I: 98, Xn: 2867668, Xn**2: 8223519758224, Longitud: 13, Semilla: 5197582, Ui: 0.5197582
I: 99, Xn: 5197582, Xn**2: 27014858646724, Longitud: 14, Semilla: 4858646, Ui: 0.4858646
I: 100, Xn: 4858646, Xn**2: 23606440953316, Longitud: 14, Semilla: 6440953, Ui: 0.6440953
```

B) Método de congruencia lineal

In [141…

```python
#Generamos los números aleatorios por congruencia lineal
xn = 7
a = 74731897457 #Multiplicador
b = 37747318974 #Incremento
m = 19 #Modulo

iteraciones = 100
numeros_aleatorios2 = []

for i in range(iteraciones):
    aux = (a * xn + b) % m

    if i > 0:
        ui = xn / m
        print(f'Paso: {i}, Xn: {xn}, Ui: {ui}')
    else:
        print(f'Paso: {i}, Xn: {xn}, Ui: ---')

    xn = aux
    numeros_aleatorios2.append(ui)
```

```
Paso: 0, Xn: 7, Ui: ---
Paso: 1, Xn: 17, Ui: 0.8947368421052632
Paso: 2, Xn: 16, Ui: 0.8421052631578947
Paso: 3, Xn: 18, Ui: 0.9473684210526315
Paso: 4, Xn: 14, Ui: 0.7368421052631579
Paso: 5, Xn: 3, Ui: 0.15789473684210525
Paso: 6, Xn: 6, Ui: 0.3157894736842105
Paso: 7, Xn: 0, Ui: 0.0
Paso: 8, Xn: 12, Ui: 0.631578947368421
Paso: 9, Xn: 7, Ui: 0.3684210526315789
Paso: 10, Xn: 17, Ui: 0.8947368421052632
Paso: 11, Xn: 16, Ui: 0.8421052631578947
```

```
Paso: 12, Xn: 18, Ui: 0.9473684210526315
Paso: 13, Xn: 14, Ui: 0.7368421052631579
Paso: 14, Xn: 3, Ui: 0.15789473684210525
Paso: 15, Xn: 6, Ui: 0.3157894736842105
Paso: 16, Xn: 0, Ui: 0.0
Paso: 17, Xn: 12, Ui: 0.631578947368421
Paso: 18, Xn: 7, Ui: 0.3684210526315789
Paso: 19, Xn: 17, Ui: 0.8947368421052632
Paso: 20, Xn: 16, Ui: 0.8421052631578947
Paso: 21, Xn: 18, Ui: 0.9473684210526315
Paso: 22, Xn: 14, Ui: 0.7368421052631579
Paso: 23, Xn: 3, Ui: 0.15789473684210525
Paso: 24, Xn: 6, Ui: 0.3157894736842105
Paso: 25, Xn: 0, Ui: 0.0
Paso: 26, Xn: 12, Ui: 0.631578947368421
Paso: 27, Xn: 7, Ui: 0.3684210526315789
Paso: 28, Xn: 17, Ui: 0.8947368421052632
Paso: 29, Xn: 16, Ui: 0.8421052631578947
Paso: 30, Xn: 18, Ui: 0.9473684210526315
Paso: 31, Xn: 14, Ui: 0.7368421052631579
Paso: 32, Xn: 3, Ui: 0.15789473684210525
Paso: 33, Xn: 6, Ui: 0.3157894736842105
Paso: 34, Xn: 0, Ui: 0.0
Paso: 35, Xn: 12, Ui: 0.631578947368421
Paso: 36, Xn: 7, Ui: 0.3684210526315789
Paso: 37, Xn: 17, Ui: 0.8947368421052632
Paso: 38, Xn: 16, Ui: 0.8421052631578947
Paso: 39, Xn: 18, Ui: 0.9473684210526315
Paso: 40, Xn: 14, Ui: 0.7368421052631579
Paso: 41, Xn: 3, Ui: 0.15789473684210525
Paso: 42, Xn: 6, Ui: 0.3157894736842105
Paso: 43, Xn: 0, Ui: 0.0
Paso: 44, Xn: 12, Ui: 0.631578947368421
Paso: 45, Xn: 7, Ui: 0.3684210526315789
Paso: 46, Xn: 17, Ui: 0.8947368421052632
Paso: 47, Xn: 16, Ui: 0.8421052631578947
Paso: 48, Xn: 18, Ui: 0.9473684210526315
Paso: 49, Xn: 14, Ui: 0.7368421052631579
Paso: 50, Xn: 3, Ui: 0.15789473684210525
Paso: 51, Xn: 6, Ui: 0.3157894736842105
Paso: 52, Xn: 0, Ui: 0.0
Paso: 53, Xn: 12, Ui: 0.631578947368421
Paso: 54, Xn: 7, Ui: 0.3684210526315789
Paso: 55, Xn: 17, Ui: 0.8947368421052632
Paso: 56, Xn: 16, Ui: 0.8421052631578947
Paso: 57, Xn: 18, Ui: 0.9473684210526315
Paso: 58, Xn: 14, Ui: 0.7368421052631579
```

```
Paso: 59, Xn: 3, Ui: 0.15789473684210525
Paso: 60, Xn: 6, Ui: 0.3157894736842105
Paso: 61, Xn: 0, Ui: 0.0
Paso: 62, Xn: 12, Ui: 0.631578947368421
Paso: 63, Xn: 7, Ui: 0.3684210526315789
Paso: 64, Xn: 17, Ui: 0.8947368421052632
Paso: 65, Xn: 16, Ui: 0.8421052631578947
Paso: 66, Xn: 18, Ui: 0.9473684210526315
Paso: 67, Xn: 14, Ui: 0.7368421052631579
Paso: 68, Xn: 3, Ui: 0.15789473684210525
Paso: 69, Xn: 6, Ui: 0.3157894736842105
Paso: 70, Xn: 0, Ui: 0.0
Paso: 71, Xn: 12, Ui: 0.631578947368421
Paso: 72, Xn: 7, Ui: 0.3684210526315789
Paso: 73, Xn: 17, Ui: 0.8947368421052632
Paso: 74, Xn: 16, Ui: 0.8421052631578947
Paso: 75, Xn: 18, Ui: 0.9473684210526315
Paso: 76, Xn: 14, Ui: 0.7368421052631579
Paso: 77, Xn: 3, Ui: 0.15789473684210525
Paso: 78, Xn: 6, Ui: 0.3157894736842105
Paso: 79, Xn: 0, Ui: 0.0
Paso: 80, Xn: 12, Ui: 0.631578947368421
Paso: 81, Xn: 7, Ui: 0.3684210526315789
Paso: 82, Xn: 17, Ui: 0.8947368421052632
Paso: 83, Xn: 16, Ui: 0.8421052631578947
Paso: 84, Xn: 18, Ui: 0.9473684210526315
Paso: 85, Xn: 14, Ui: 0.7368421052631579
Paso: 86, Xn: 3, Ui: 0.15789473684210525
Paso: 87, Xn: 6, Ui: 0.3157894736842105
Paso: 88, Xn: 0, Ui: 0.0
Paso: 89, Xn: 12, Ui: 0.631578947368421
Paso: 90, Xn: 7, Ui: 0.3684210526315789
Paso: 91, Xn: 17, Ui: 0.8947368421052632
Paso: 92, Xn: 16, Ui: 0.8421052631578947
Paso: 93, Xn: 18, Ui: 0.9473684210526315
Paso: 94, Xn: 14, Ui: 0.7368421052631579
Paso: 95, Xn: 3, Ui: 0.15789473684210525
Paso: 96, Xn: 6, Ui: 0.3157894736842105
Paso: 97, Xn: 0, Ui: 0.0
Paso: 98, Xn: 12, Ui: 0.631578947368421
Paso: 99, Xn: 7, Ui: 0.3684210526315789
```

2. Cálculo de chi cuadrado

In [142…

```
#Definimos una función para calcular varias veces chi cuadrado
def calcular_chi_cuadrado(numeros_aleatorios):
```

```
    N = len(numeros_aleatorios) #Cantidad de números generados
    n = math.sqrt(N) #Grados de libertad

    Ei = int(N / n)
    tamanio_intervalo = 1 / Ei

    inicio_intervalo = 0
    fin_intervalo = tamanio_intervalo

    chi = 0

    for intervalo in range(Ei):
        rango = [inicio_intervalo, fin_intervalo]

        Oi = 0

        for numero in numeros_aleatorios:
            if numero >= inicio_intervalo and numero <= fin_intervalo:
                Oi =+ 1

        chi += math.pow(Oi - Ei, 2) / Ei
        inicio_intervalo += tamanio_intervalo
        fin_intervalo += tamanio_intervalo

    print(f'a: 0.05, Grado de libertad: {n}, k-1: {n - 1},Chi: {chi} <= 16,919')


print('Método de medio cuadrados:')
calcular_chi_cuadrado(numeros_aleatorios1)

print('\nMétodo de congruencia lineal:')
calcular_chi_cuadrado(numeros_aleatorios2)
```

```
Método de medio cuadrados:
a: 0.05, Grado de libertad: 10.0, k-1: 9.0,Chi: 80.99999999999999 <= 16,919

Método de congruencia lineal:
a: 0.05, Grado de libertad: 10.0, k-1: 9.0,Chi: 86.69999999999999 <= 16,919
```

3. Generando gráficas de distribución de probabilidad

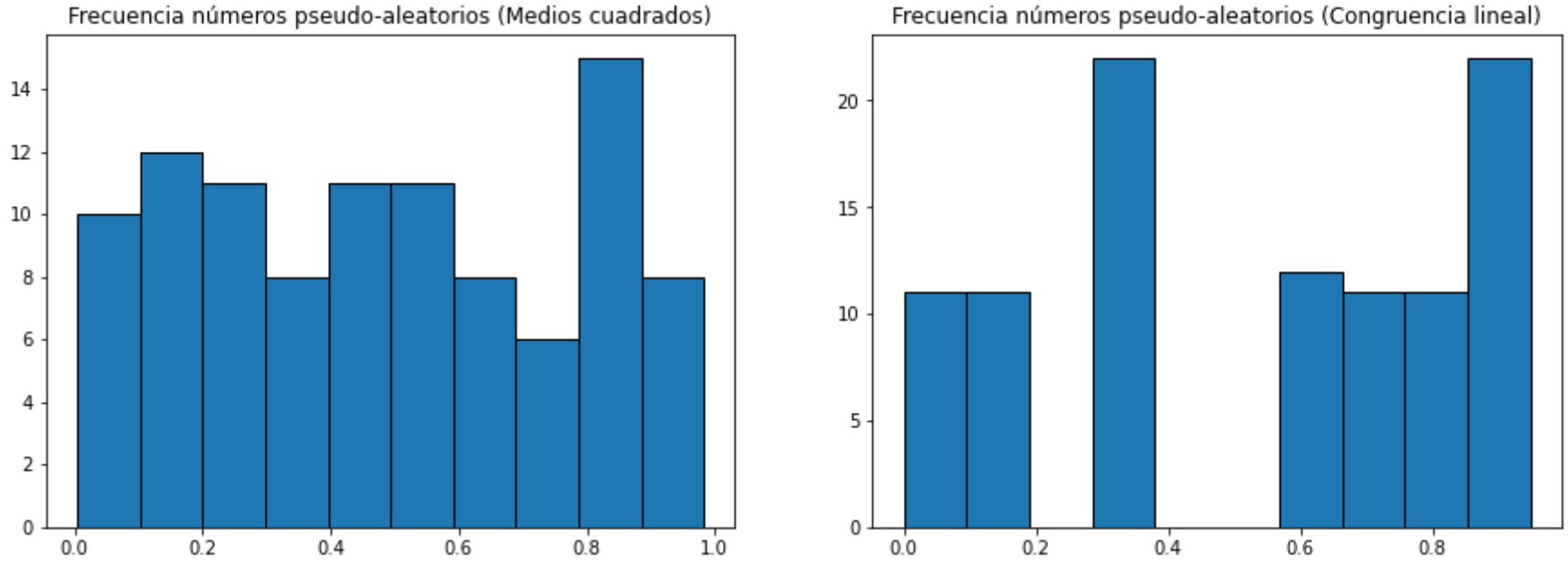
In [148…

```
#Graficamos las distribuciones de probabilidad
grafico = pp.figure(figsize = (15, 5))
```

```
grafica1 = grafico.add_subplot(1, 2, 1)
grafica1.set_title('Frecuencia números pseudo-aleatorios (Medios cuadrados)')
pp.hist(numeros_aleatorios1, edgecolor = 'black')

grafica2 = grafico.add_subplot(1, 2, 2)
grafica2.set_title('Frecuencia números pseudo-aleatorios (Congruencia lineal)')
pp.hist(numeros_aleatorios2, edgecolor = 'black')

pp.show()
```



Como se puede observar en la gráfica, la distribución de los números pseudo-aleatorios generados a partir de los parámetros iniciales para ambos métodos no es uniforme. En el caso del método por medios cuadrados los números aparecen con diversas frecuencias. Por otra parte, en el caso por congruencia lineal existen cambios demasiados bruscos en los picos de frecuencia.

4. Conclusiones

Mediante la presente prueba realizada se puede afirmar que para los datos iniciales entregados, ningún grupo de números generados por los métodos propuestos cumple con las normas de uniformidad. Esto se puede verificar calculando Chi cuadrado de ambos, y de los cuales ninguno se encuentra dentro de las condiciones establecidas por los grados de libertad.

La distribución ideal para los números pseudo-aleatorios es la uniforme, la cual tiene forma de rectangulo y donde cada número tiene la misma probabilidad de aparecer. En el caso de los números generados en esta prueba, ninguno de ellos se ajusta a la forma rectangular.

In [ ]: