Petite introduction au chaos

In [1]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
plt.rcParams['figure.figsize'] = (8, 8)
```

Pas de démonstrations dans ce notebook. Juste des illustrations, des suggestions.

1. Introduction

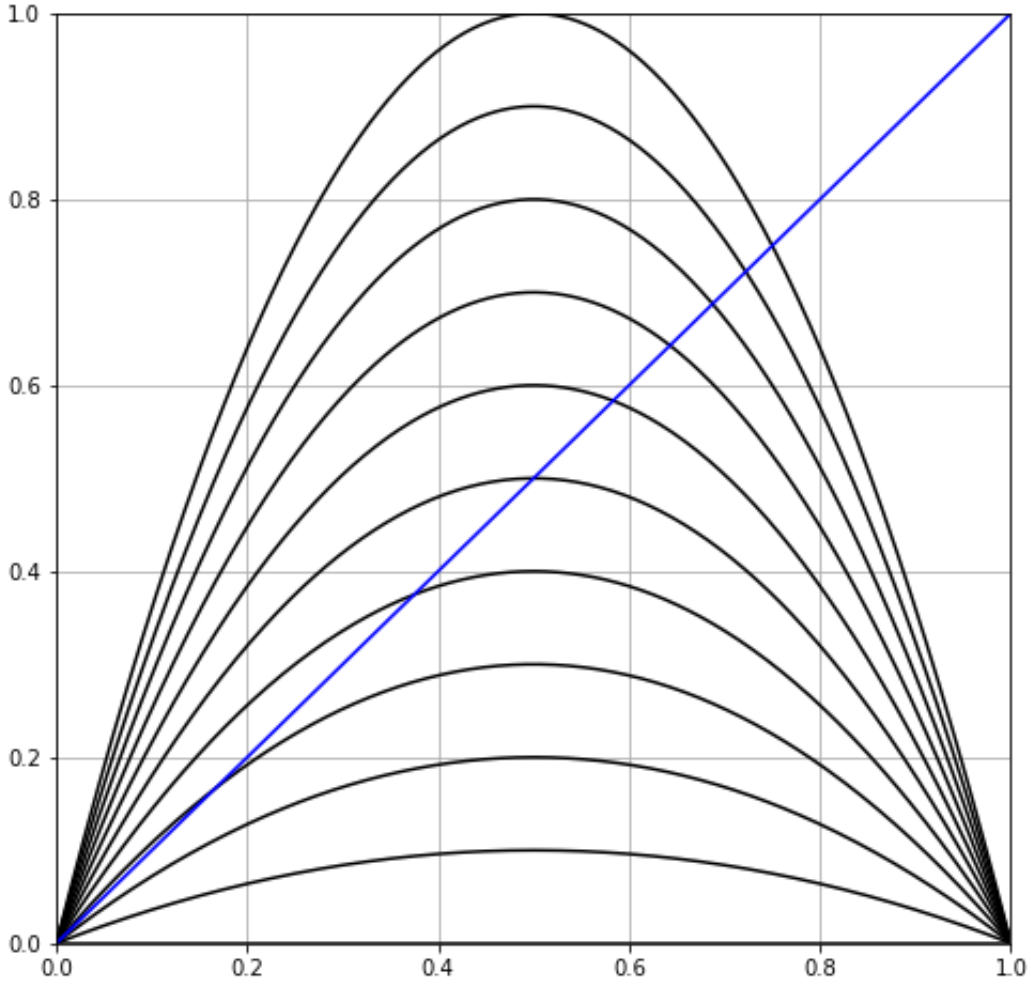
Pour $\mu \in \mathbb{R}$, soit $f : [0, 1] \to \mathbb{R}$ définie par $f(x) = \mu x(1 - x)$.

In [3]:
```
def f(mu, x): return mu * x * (1 - x)
```

Pour $\mu \in [0, 4]$, la fonction f est à valeurs dans $[0, 1]$. Bien qu'il y ait des tas de choses intéressantes à dire dans les autres cas, nous supposerons dorénavant que $0 \leq \mu \leq 4$. Ci-dessous, le tracé de f pour quelques valeurs de μ.

In [4]:
```
def subdivision(a, b, n):
    d = (b - a) / n
    return [a + k * d for k in range(n + 1)]
```

In [5]:
```
xs = subdivision(0, 1, 400)
mus = subdivision(0, 4, 10)
plt.axis([0, 1, 0, 1])
for mu in mus:
    plt.plot(xs, [f(mu, x) for x in xs], 'k')
plt.plot(xs, xs, 'b')
plt.grid()
plt.show()
```



Soit $x_0 \in [0, 1]$. Pour tout $n \in \mathbb{N}$, posons $x_{n+1} = f(x_n)$. Quel est le comportement de la suite (x_n) ? Cela dépend évidemment beaucoup de la valeur du paramètre μ. Nous allons dans ce qui suit regarder quelques cas. Grosso modo, tant que μ est entre 0 et 3, tout va bien. C'est après que ça devient intéressant. Rappelez-vous :

Une propriété mathématique est soit évidente, soit intéressante

La fonction `iter` calcule $f \circ f \circ \ldots \circ f(x)$, k fois. On ne s'en servira pas tout de suite.

In [6]:
```
def iter(f, k, mu, x):
    y = x
    for i in range(k): y = f(mu, y)
    return y
```

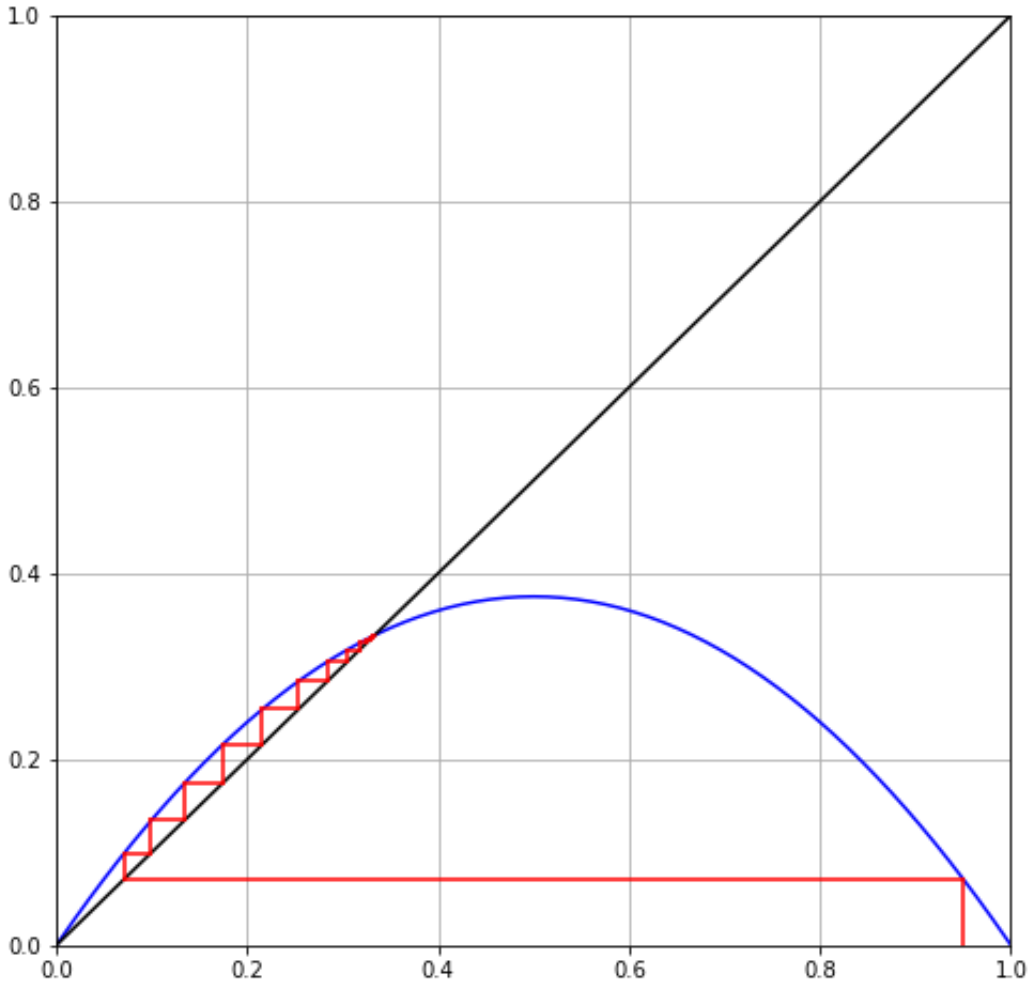
La fonction `escalier` permet de visualiser le comportement de la suite (x_n). Elle prend en paramètres

- Un réel $\mu \in [0, 4]$.
- Un entier n qui est le nombre d'itérations désirées.
- Un réel optionnel $x_0 \in [0, 1]$.
- Un entier optionnel `skip` qui permet de ne pas afficher les premières itérations.
- Un entier optionnel `niter`, nous en parlerons plus loin.

Elle affiche la fonction f, la droite d'équation $y = x$, ainsi que les x_i pour $skip \leq i < skip + n$.

In [7]:
```
def escalier(mu, n, x0=0.4342, skip=0, niter=1):
    g = lambda t: iter(f, niter, mu, t)
    xs = subdivision(0, 1, 200)
    ys = [g(x) for x in xs]
    plt.axis([0, 1, 0, 1])
    plt.plot(xs, ys, 'b')
    plt.plot(xs, xs, 'k')
    x = x0
    for j in range(skip): x = g(x)
    coul = 'r'
    plt.plot([x, x], [0, g(x)], coul)
    for i in range(n):
        plt.plot([x, g(x)], [g(x), g(x)], coul)
        plt.plot([g(x), g(x)], [g(x), g(g(x))], coul)
        x = g(x)
    plt.grid()
    plt.show()
```

In [8]:
```
escalier(1.5, 20, x0=0.95)
```

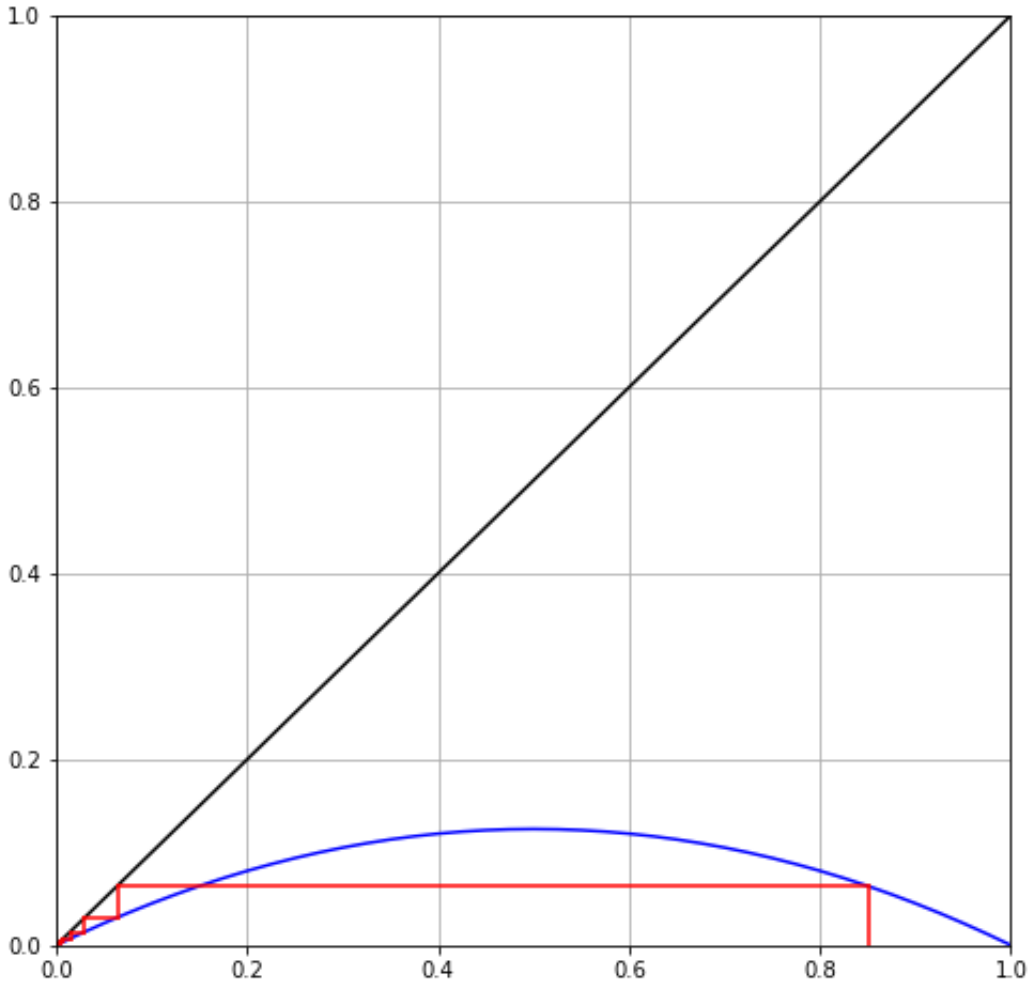


J'entame ci-dessous une discussion selons les valeurs de μ. Je ne prouverai rien, ce notebook est là avant tout pour faire VOIR des choses.

2. Le cas $0 \leq \mu \leq 1$

La suite tend vers 0. Faites varier les valeurs de μ et de x_0 dans la cellule ci-dessous, pour voir comment cela influence la convergence. Essayez en particulier $\mu = 1$ ... Ces incitations à tenter des choses sont évidemment valables dans toute la suite.

In [9]:
```
escalier(0.5, 20, x0=0.85)
```



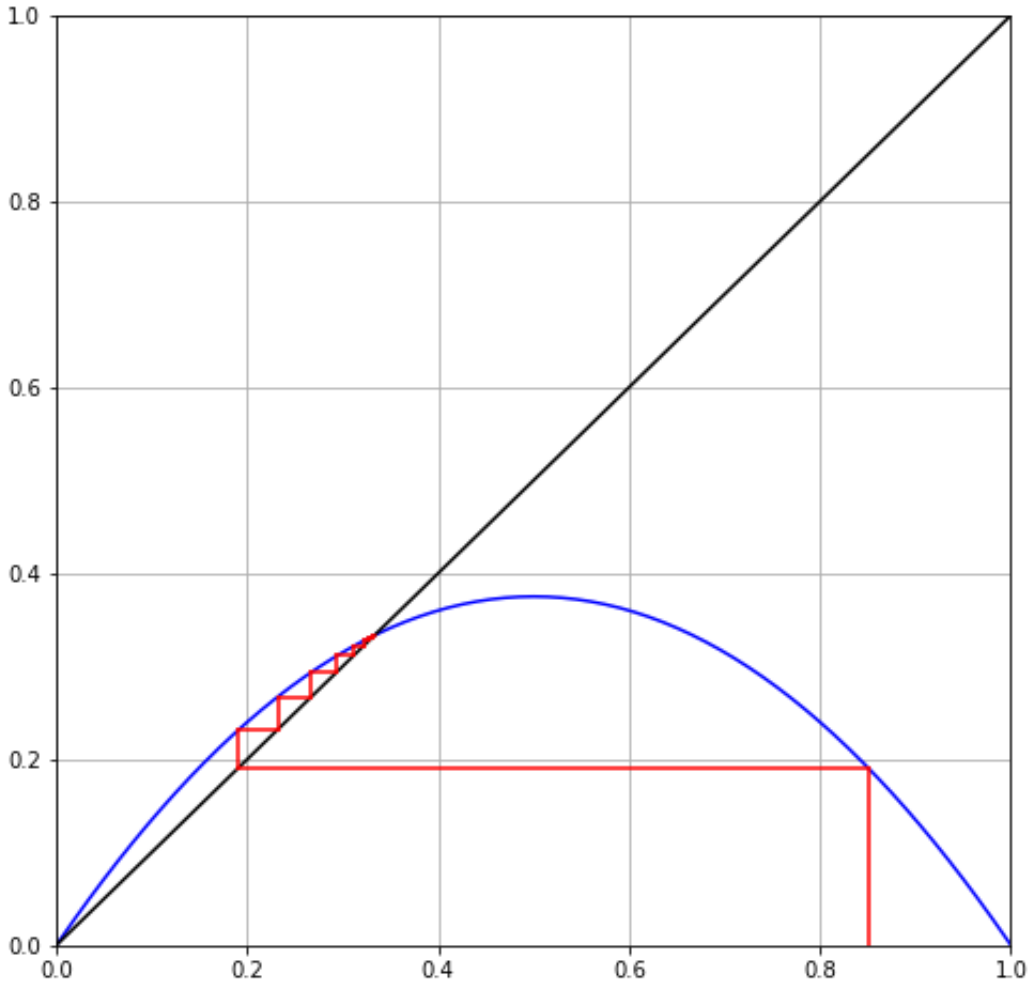
3. Le cas $1 \leq \mu \leq 2$

Sauf cas très particuliers ($x_0 = 0$ par exemple), la suite tend vers $\frac{\mu-1}{\mu}$.

Exercice : Montrez-le, ce n'est pas si difficile.

Pour $\mu = \frac{3}{2}$, la suite (x_n) tend (en général) vers $\frac{1}{3}$. Mais oui, essayez d'autres valeurs de μ entre 1 et 2 ! Le nombre $\frac{3}{2}$ m'est venu assez spontanément comme un nombre quelconque entre 1 et 2 :-).

In [10]:
```
escalier(1.5, 20, x0=0.85)
```

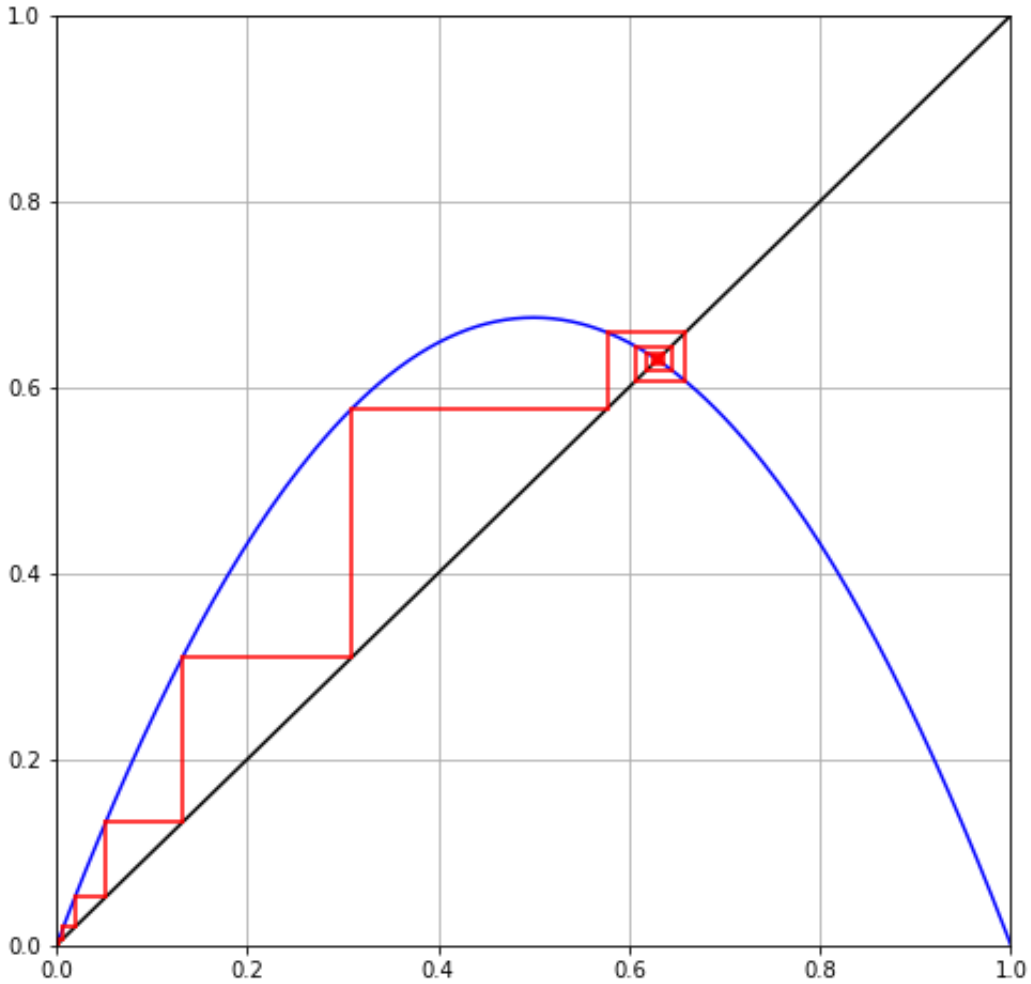


4. Le cas $2 \leq \mu \leq 3$

La suite converge encore vers $\frac{\mu-1}{\mu}$, mais pas de la même façon. Cette fois-ci, les suites (x_{2n}) et (x_{2n+1}) sont adjacentes. Du coup, notre escalier est un escargot.

S'il vous prend l'envie de tester la valeur $\mu = 3$, vous constaterez que pour $\mu = 3$, la suite tend vers $\frac{\mu-1}{\mu} = \frac{2}{3}$ ... comme un escargot. Je n'ai pas réussi à trouver de documentation cohérente sur la vitesse de convergence dans ce cas. Tout ce que racontent les Maîtres est que c'est trèèès lent.

In [11]:
```
escalier(2.7, 20, x0=0.001)
```



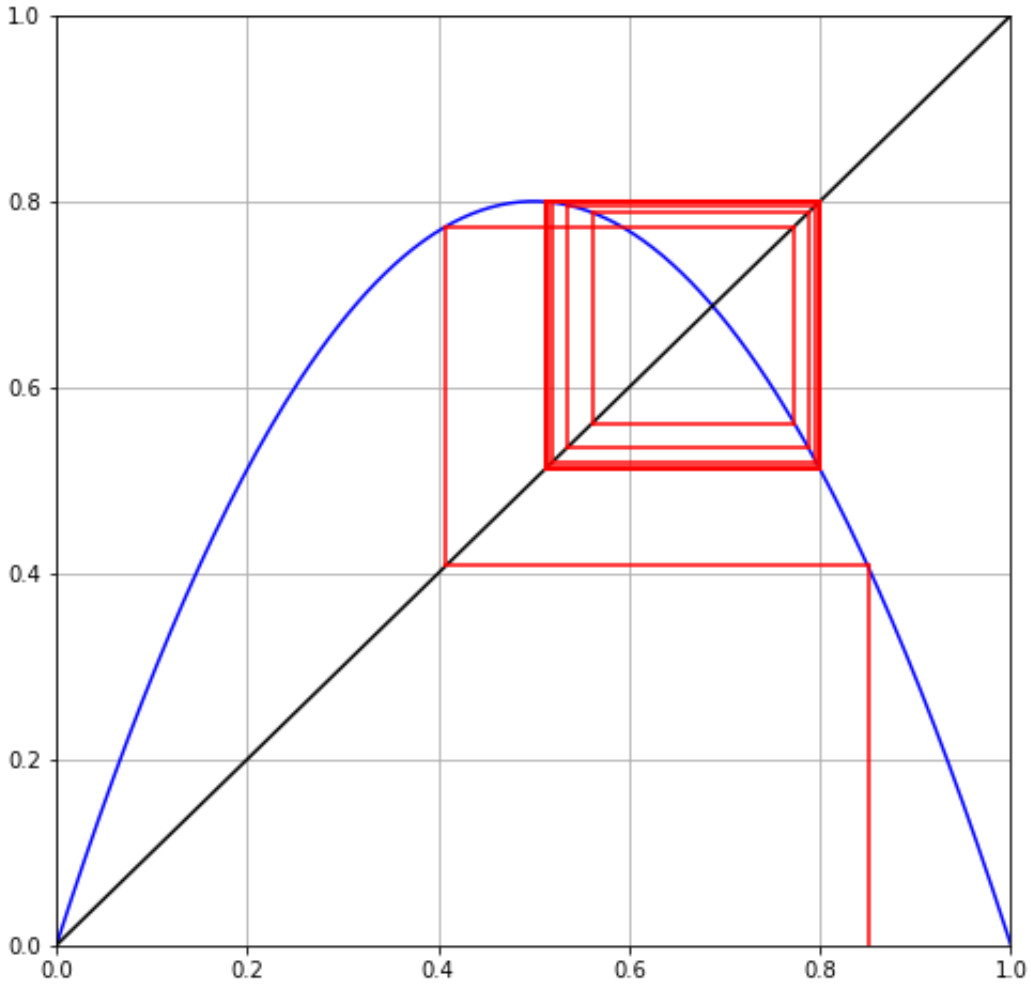
Remarquer, sur l'exemple ci-dessus, le choix d'un x_0 tout petit, très proche de 0 qui est un point fixe. Pourquoi la suite (x_n) ne tend-elle pas bêtement vers 0 ? Eh bien à cause du théorème des accroissements finis. La dérivée de f en 0 est stritement supérieure à 1, ce qui rend ce point **répulsif**. En revanche, $|f'(\frac{\mu-1}{\mu})| < 1$, ce qui en fait un point **attractif**.

5. Le cas $3 < \mu \leq 4$

Là, tout se complique, donc devient intéressant. La suite diverge en général, et plus μ augmente plus son comportement est compliqué (en général). Désolé pour les "en général", mais la vérité est plus compliquée que "c'est de plus en plus compliqué" (cf plus loin, $\mu = 3.84$).

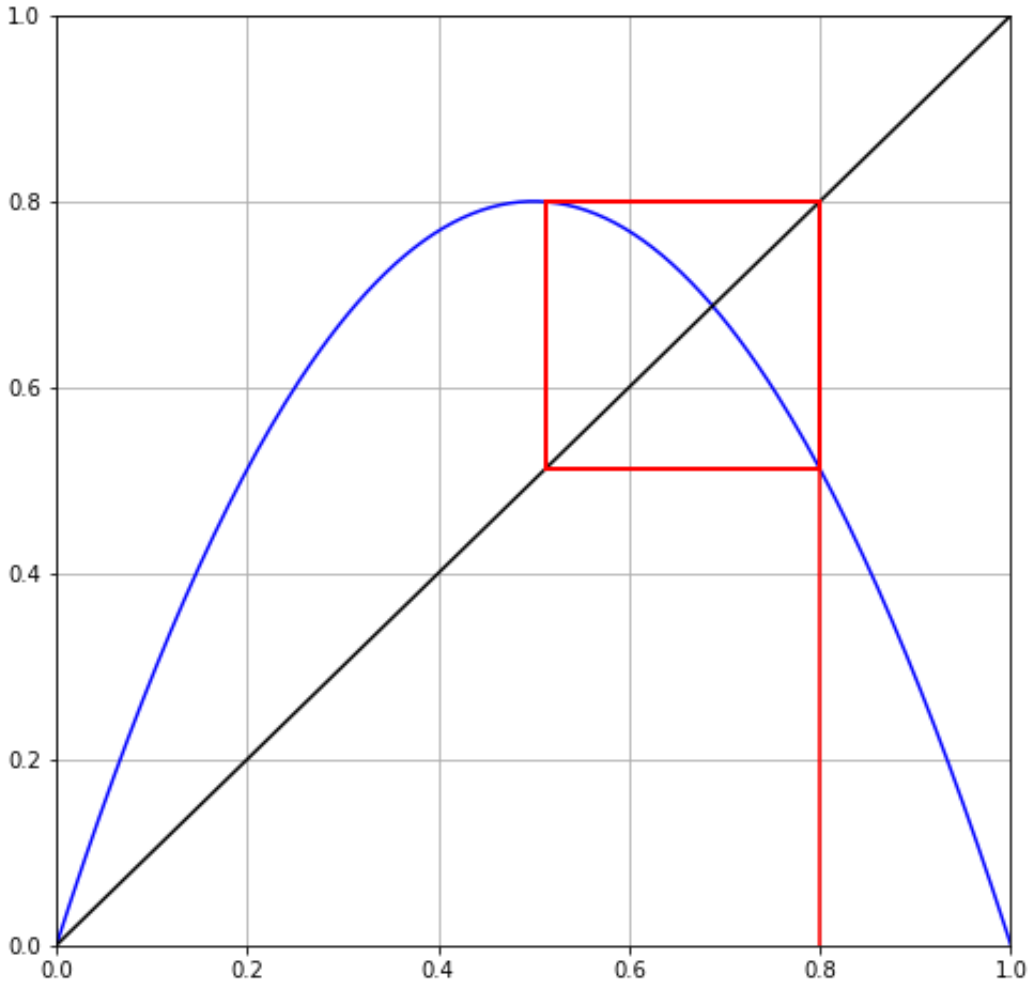
Prenons par exemple $\mu = 3.2$.

In [12]:
```
escalier(3.2, 20, x0=0.85)
```



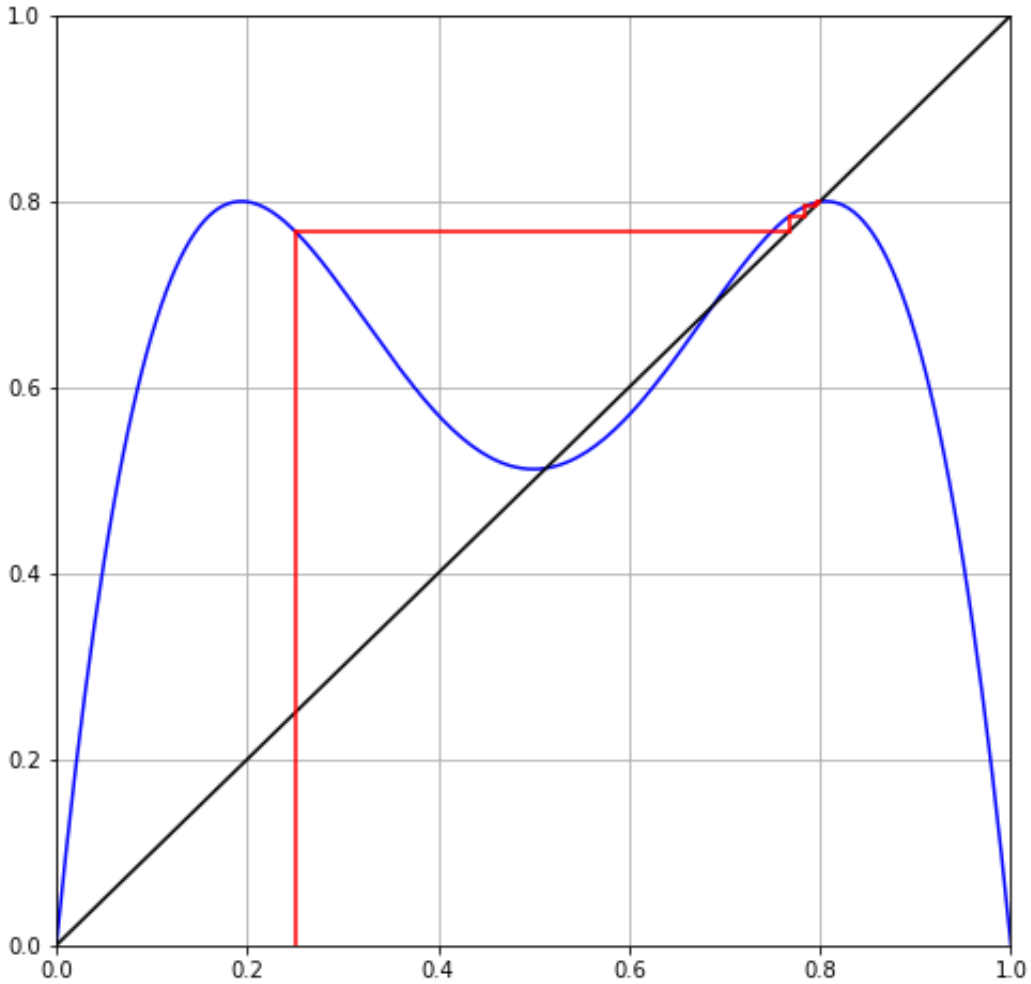
Ne traçons pas les 100 premières valeurs de la suite.

In [13]:
```
escalier(3.2, 20, x0=0.85, skip=100)
```



Que se passe-t-il ? Les suites (x_{2n}) et (x_{2n+1}) convergent, mais vers des limites différentes. Il est temps de parler du paramètre `niter` de la fanction `escalier`. En mettant ce paramètre à 2, on trace $f \circ f$ et la suite (x_{2n}). Allons-y.

In [14]:
```
escalier(3.2, 20, x0=0.25, niter=2)
```



En continuant à augmenter μ, la situation se complique de plus en plus. Pour $\mu = 3.5$, par exemple, il y a 4 valeurs d'adhérence. Euh, c'est quoi une valeur d'adhérence ?

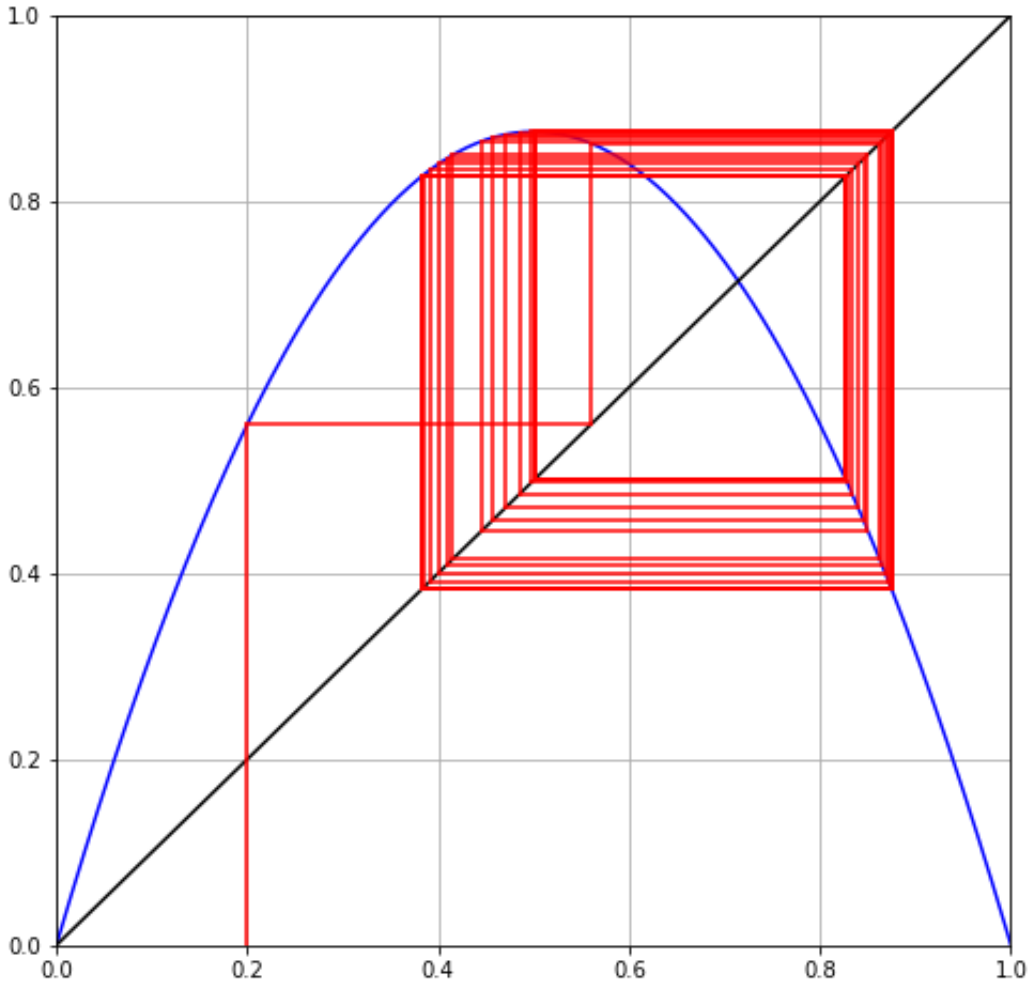
Définition : Les valeurs d'adhérence d'une suite sont les réels dont tout voisinage contient une infinité de termes de la suite.

Affirmation : Les valeurs d'adhérence sont aussi les limites des suites extraites de la suite. Bien entendu, une suite convergente a une unique valeur d'adhérence, qui est sa limite.

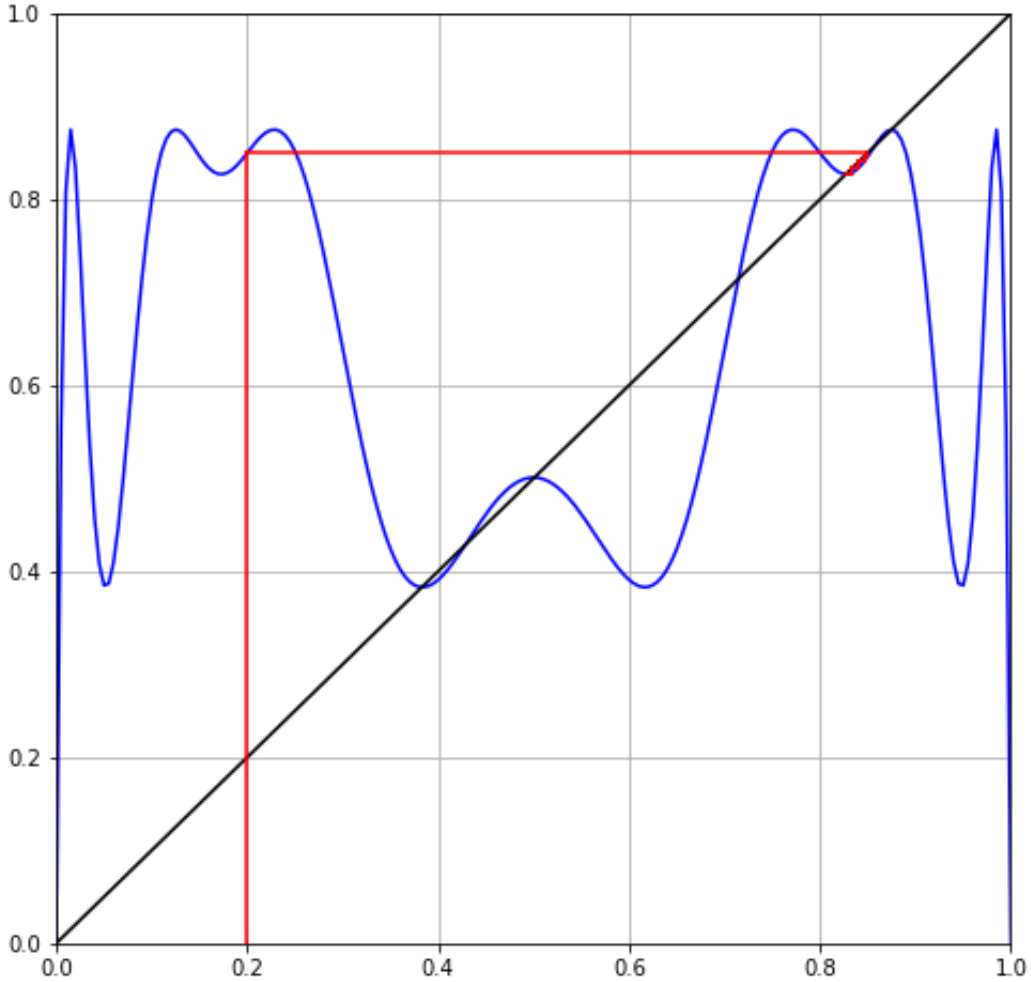
À vous tester, avec `skip` nul ou pas, pour différentes valeurs de μ.

Valeurs conseillées : 3.2, 3.74, 3.84, 4. Et bien d'autres ...

In [15]:
```
escalier(3.5, 100, x0=0.2, skip=0, niter=1)
```

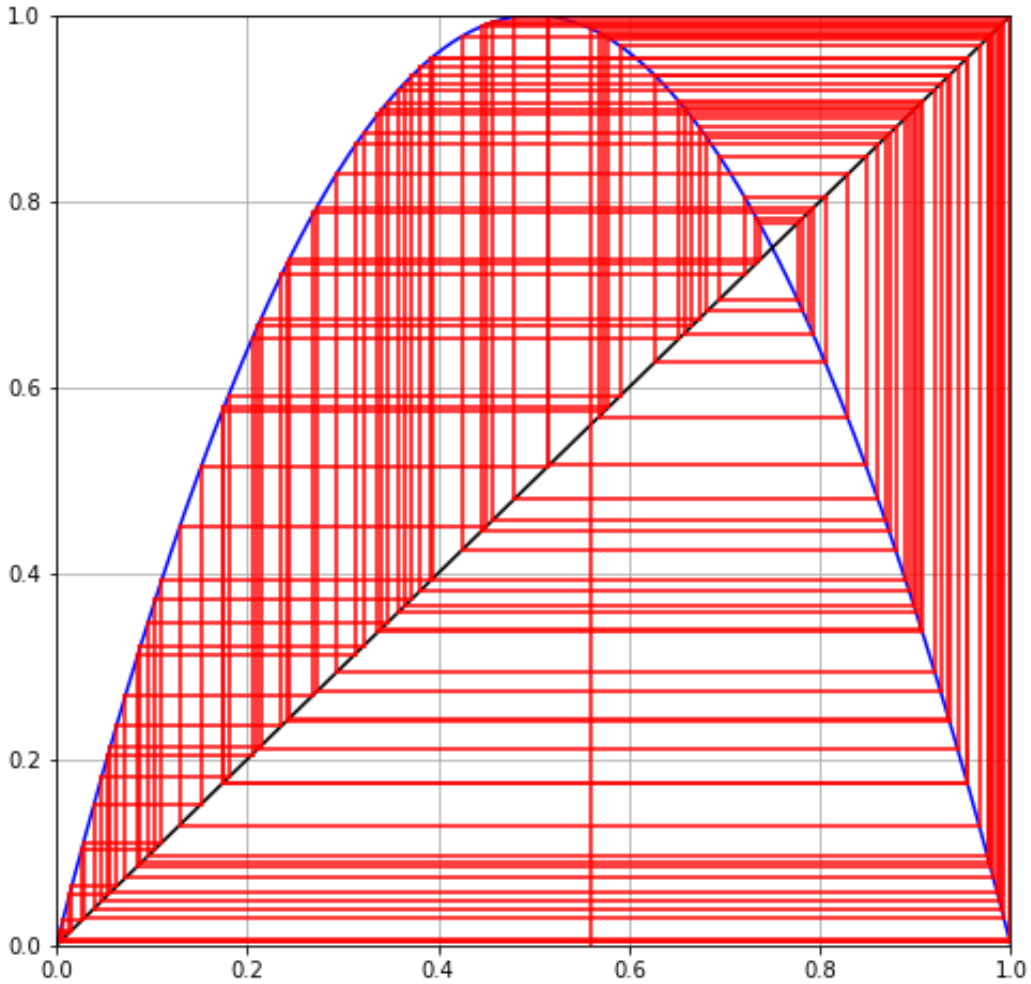


In [16]:
```
escalier(3.5, 100, x0=0.2, niter=4)
```



Et pour finir, le cas $\mu = 4$, où l'on peut montrer que TOUT réel de $[0, 1]$ est valeur d'adhérence de la suite (x_n) (en général)

In [17]:
```
escalier(4, 100, x0=0.1, skip=50, niter=1)
```



Il reste beaucoup, beaucoup de choses à dire, mais je ne veux pas multiplier les graphiques. Il est temps de passer à une autre vision, plus globale, et aussi plus esthétique, de la **dynamique de** f.

6. La cascade harmonique

In [18]:
```
plt.rcParams['figure.figsize'] = (12, 8)
```

Au lieu de tracer la suite (x_n) pour une valeur de μ à la fois, nous allons tracer toutes les suites en même temps. Précisément, nous allons tracer le graphique suivant : En abscisse, les valeurs du paramètre μ. En ordonnées, les valeurs d'adhérence de la suite (x_n) pour le paramètre μ correspondant.

La fonction `cascade` prend en paramètres :

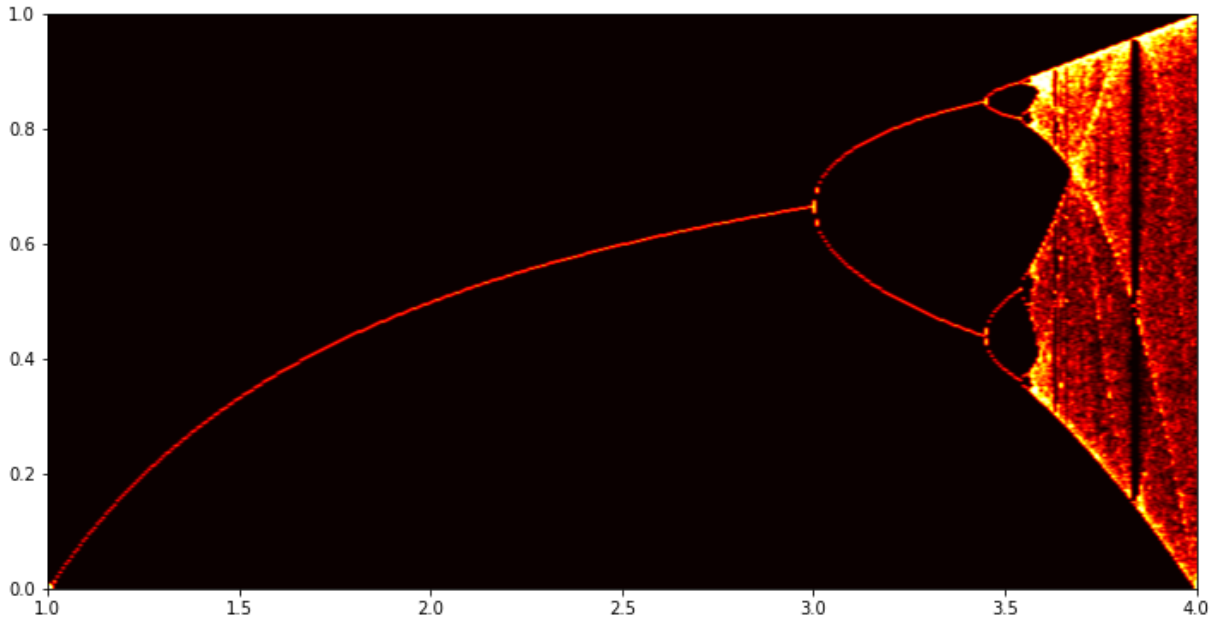
- deux réels μ_1 et μ_2 qui sont les bornes de variations de μ.
- deux réels x_1 et x_2 qui sont les valeurs minimale et maximale des ordonnées de notre graphique.
- Une paramètre de contraste sur lequel nous allons revenir.
- Un paramètre `niter` sur lequel nous allons revenir aussi.

Voici le code que j'expliquerai petit à petit. Mais lançons d'abord, juste pour voir.

In [19]:
```
def cascade(mu1, mu2, x1, x2, contrast=100, niter=1000):
    Nmu = 400
    Nx = 400
    x = 0.3432
    M = (Nx + 1) * [None]
    for i in range(Nx + 1): M[i] = (Nmu + 1) * [0]
    for i in range(Nmu + 1):
        mu = mu1 + i * (mu2 - mu1) / Nmu
        for j in range(100): x = f(mu, x)
        for j in range(niter):
            x = f(mu, x)
            v = int(Nx * (x - x1) / (x2 - x1))
            if v >= 0 and v <= Nx: M[v][i] += 1
        s = sum([M[v][i] for v in range(Nx)])
        for v in range(Nx):
            M[v][i] = M[v][i] / s * contrast
            if M[v][i] > 1: M[v][i] = 1
    aspect = 0.5 * (mu2 - mu1) / (x2 - x1)
    plt.imshow(M, origin='lower',interpolation='bicubic', extent=(mu1,
    plt.savefig('cascade.png')
    plt.show()
```

6.1 Vue globale

In [20]:
```
cascade(1, 4, 0, 1)
```



Que voit-on ? Pour $\mu \in [1, 3]$, une simple courbe. On est dans la zone où la suite converge. Précisément, c'est la courbe d'équation $x = \frac{\mu-1}{\mu}$.

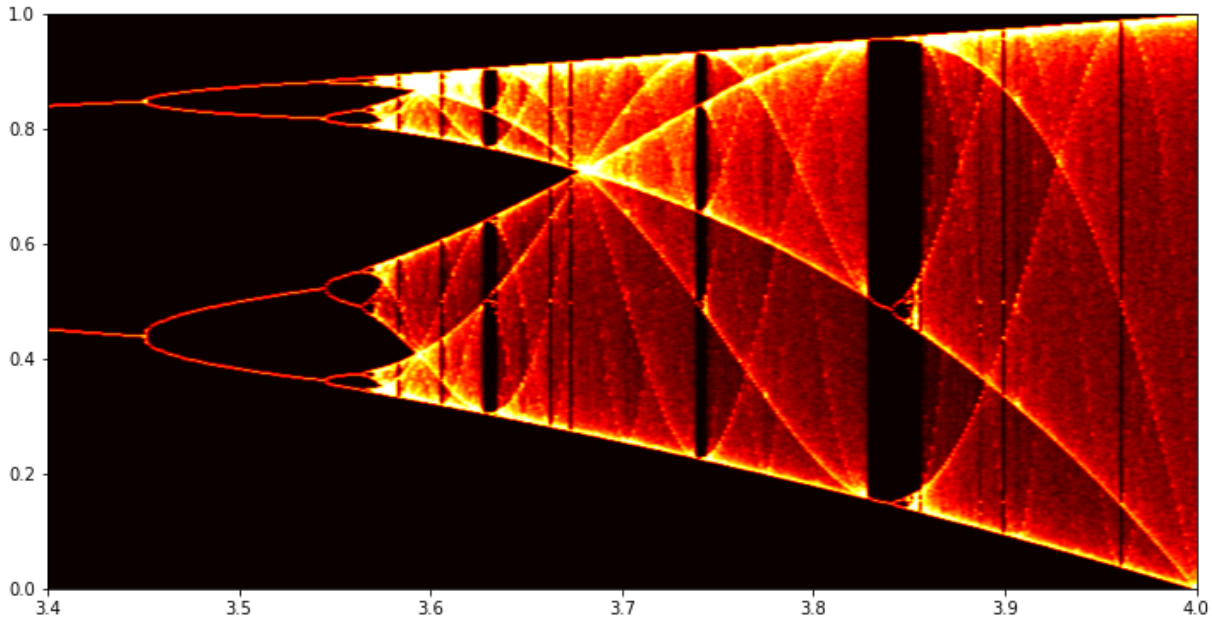
Pour μ entre 3 et environ 3.4, deux courbes. Entre ces deux valeurs, la suite a deux valeurs d'adhérence. Ensuite, tout s'accélère. On distingue une zone "à 4 courbes" puis, si on a de bons yeux, une zone à 8 courbes. Après c'est le fouillis. C'est donc intéressant :-).

Faisons des zooms. Je commenterai peu ou pas, à vous de regarder.

6.2 La zone d'intérêt

Laissons tomber les "petites" valeurs de μ. Tout devient intéressant pour $\mu > 3$.

In [21]:
```
cascade(3.4, 4, 0, 1, niter=10000)
```



Bon choix de `niter` , bon choix du contraste. Et bon choix des couleurs. Un peu flamboyant peut-être, mais si cela vous perturbe, changez la palette de couleurs. Que voit-on ?

- Des endroits chauffés à blanc. Ce sont les points d'adhérence de la suite (x_n) très très adhérents.
- Conséquence du premier point, des "courbes" de haute température. C'est quoi ces courbes ?
- Des zones toutes noires au mileu d'autres zones très agitées. Où cela exactement ?
- Une structure "fractale", des motifs se répètent.
- Omniprésence de 1, 2, 4, 8, 16 ...

Est-ce tout ? Non, mais ce notebook est trop étroit pour tout contenir.

Que fait la fonction `cascade` ? Une matrice M initialisée avec des zéros représente les pixels du graphe à tracer. Pour "chaque" valeur de μ,

- elle écarte les 100 premiers termes de la suite (x_n).
- elle calcule les `niter` termes suivants. Pour chaque terme calculé, elle augmente de 1 la case de M correspondante.
- enfin, la ligne `M[v][i] = M[v][i] / s * contrast` fait "une mise à l'échelle". Puis la ligne suivante met à 1 toutes les valeurs de M qui dépassent 1.

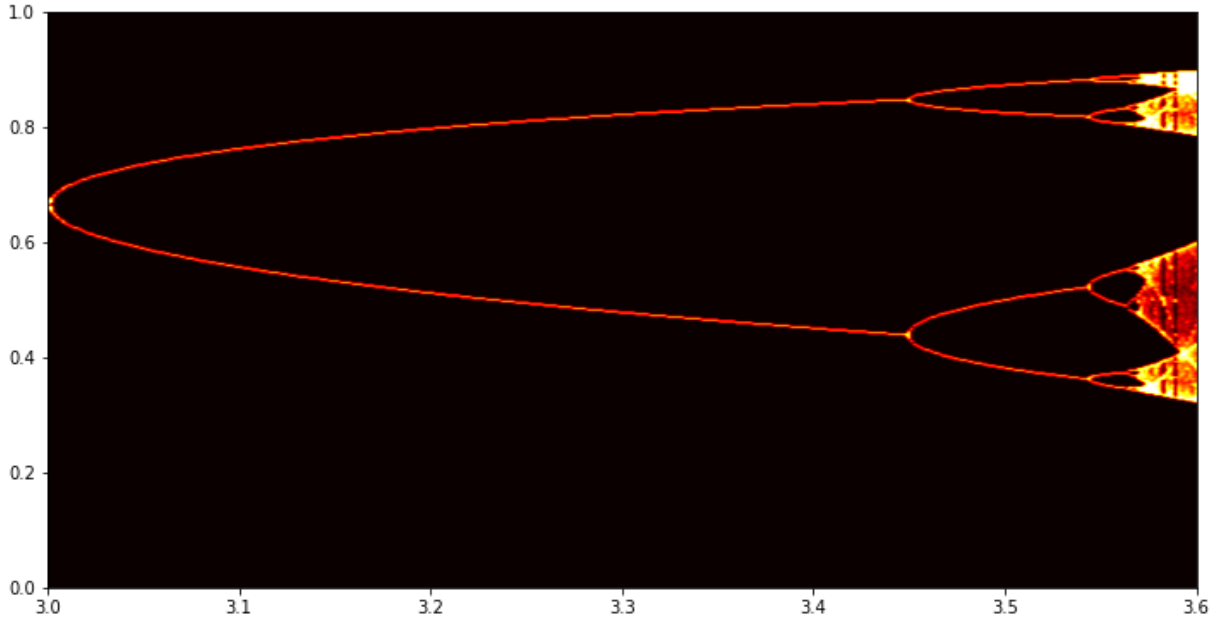
La paramètre `contrast` permet de faire apparaître plus fortement les parties du graphique trop sombres. Une valeur de 100 donne d'assez bons résultats, sauf si l'on zoome sur certaine parties du graphique. Le mieux est de faire des essais.

Le paramètre `niter` est un compromis entre la qualité de l'image et le temps de calcul. Plus `niter` est grand, plus l'image sera belle. Mais plus on attendra ... Une valeur de 1000 donne assez vite une image correcte. Pour une image de qualité "édition", une valeur de 50000 ou plus sera préférable.

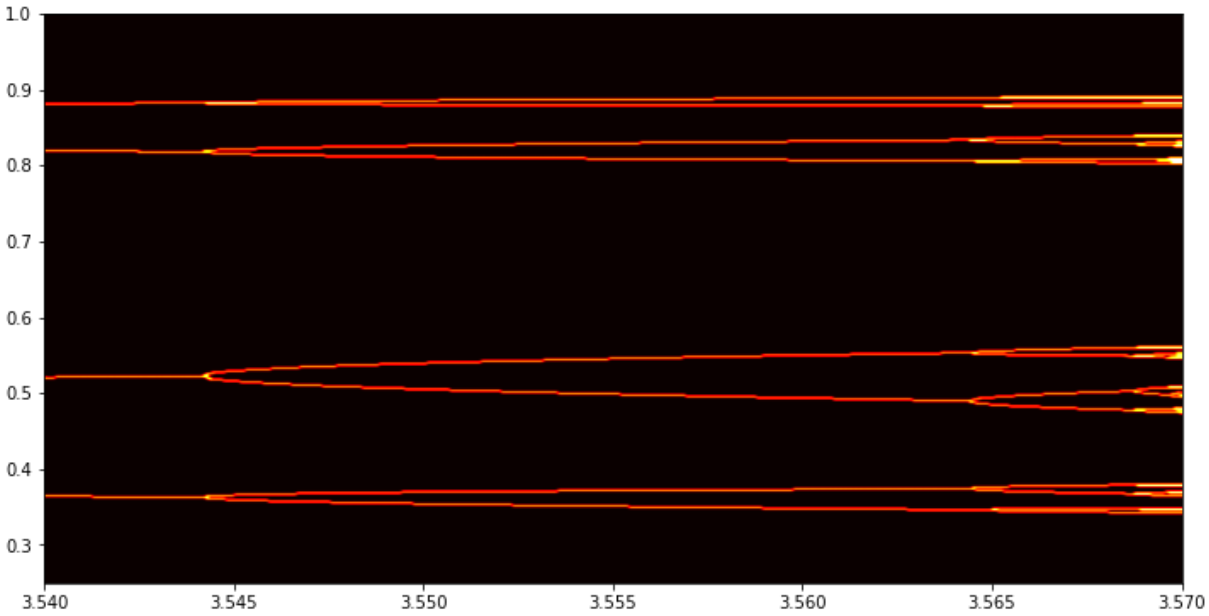
Il y aurait énormément à dire. Je me contente de faire un zoom sur quelques zones intéressantes. Vous avez l'outil, à vous de vous documenter, d'expérimenter ... ou de vous contenter d'admirer.

6.3 Doublements de période : 1, 2, 4, 8, ...

In [22]:
```
cascade(3, 3.6, 0, 1, niter=5000)
```

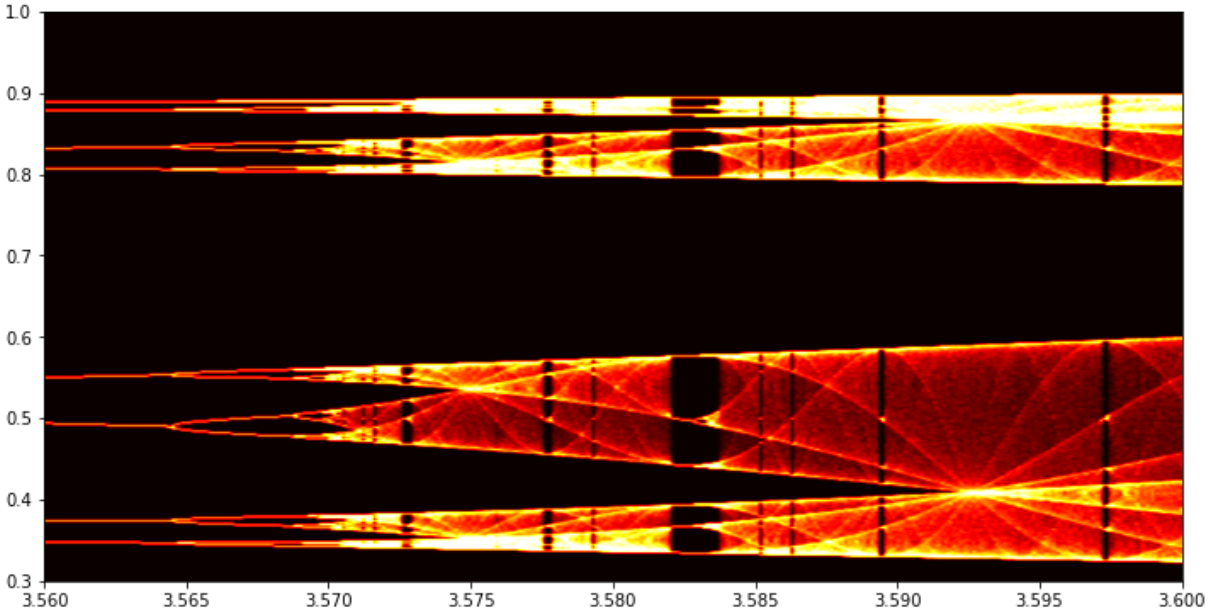


In [23]:
```
cascade(3.54, 3.57, 0.25, 1, niter=5000)
```



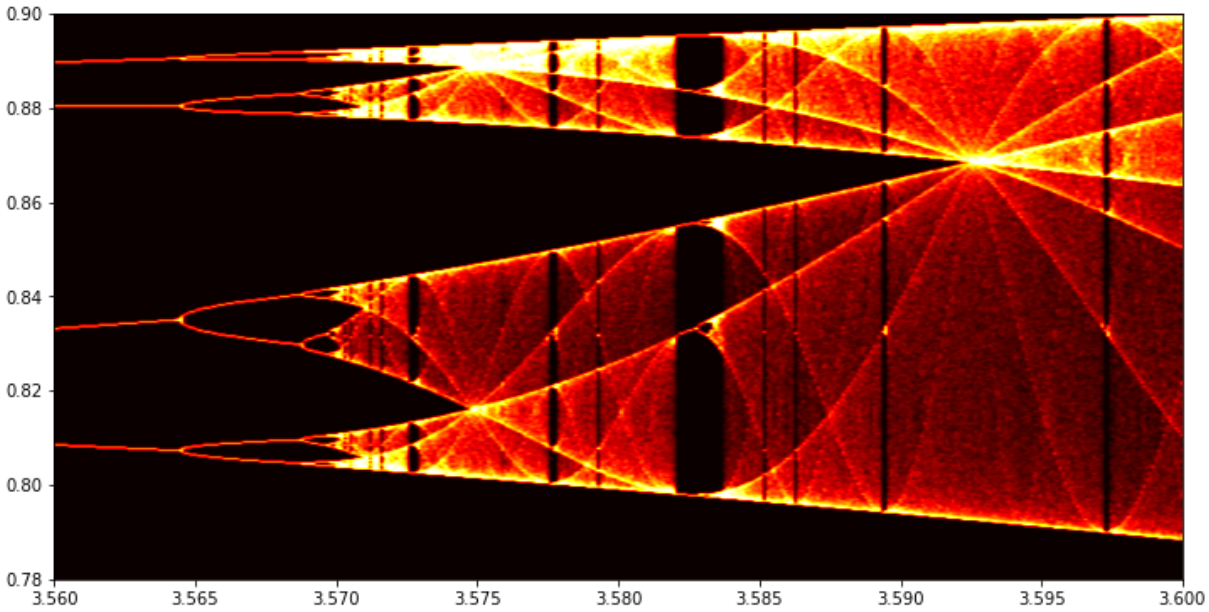
6.4 Entrée dans le chaos : 16, 32, 64, ..., ∞

In [24]:
```
cascade(3.56, 3.6, 0.3, 1, niter=10000)
```



Zoom sur la moitié haute du graphique ci-dessus. Structure fractale, les parties sont "semblables" au tout.

In [25]:
```
cascade(3.56, 3.6, 0.78, 0.9, niter=10000)
```

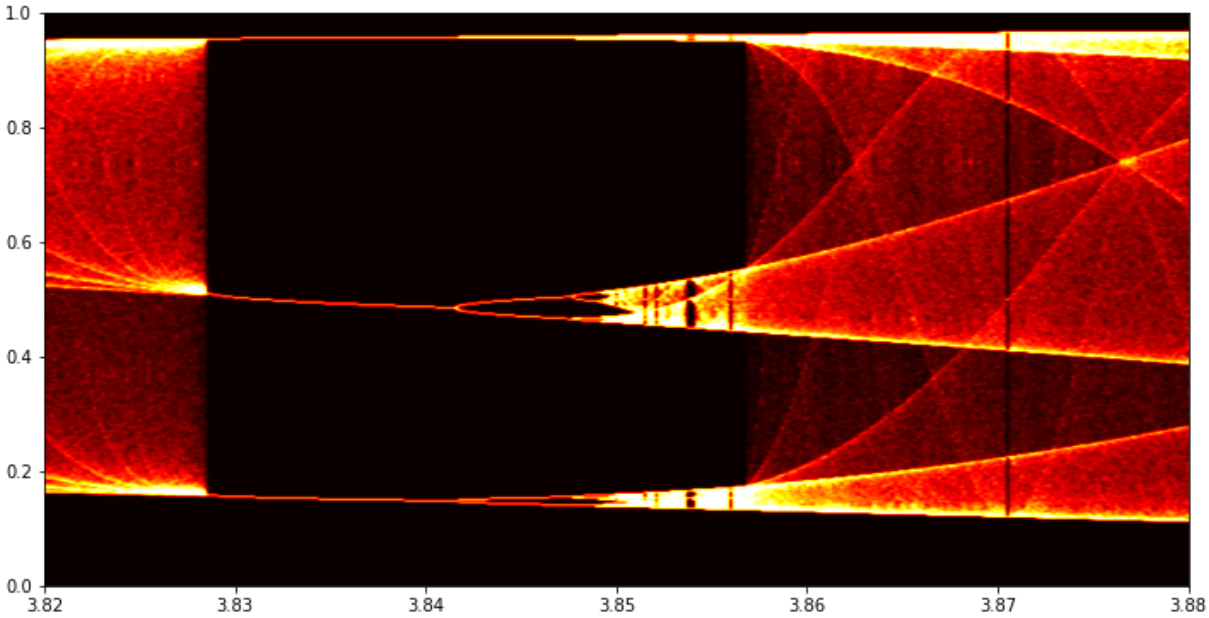


6.5 Le calme au milieu de la tempête - points 3-périodiques et 5-périodiques

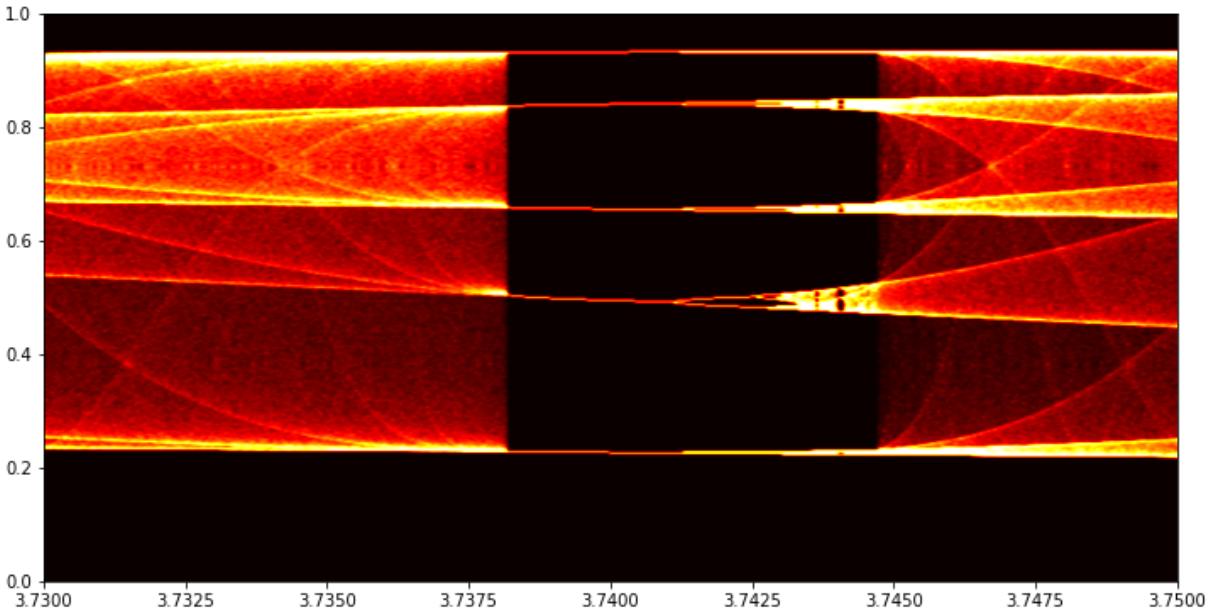
En plein milieu des zones en feu, le calme ... apparent. Pour tout entier n, on peut trouver (en cherchant bien) des points de période n.

Théorème (Li et Yorke) : période 3 $\Rightarrow$ chaos.

In [26]:
```
cascade(3.82, 3.88, 0, 1, niter=5000)
```



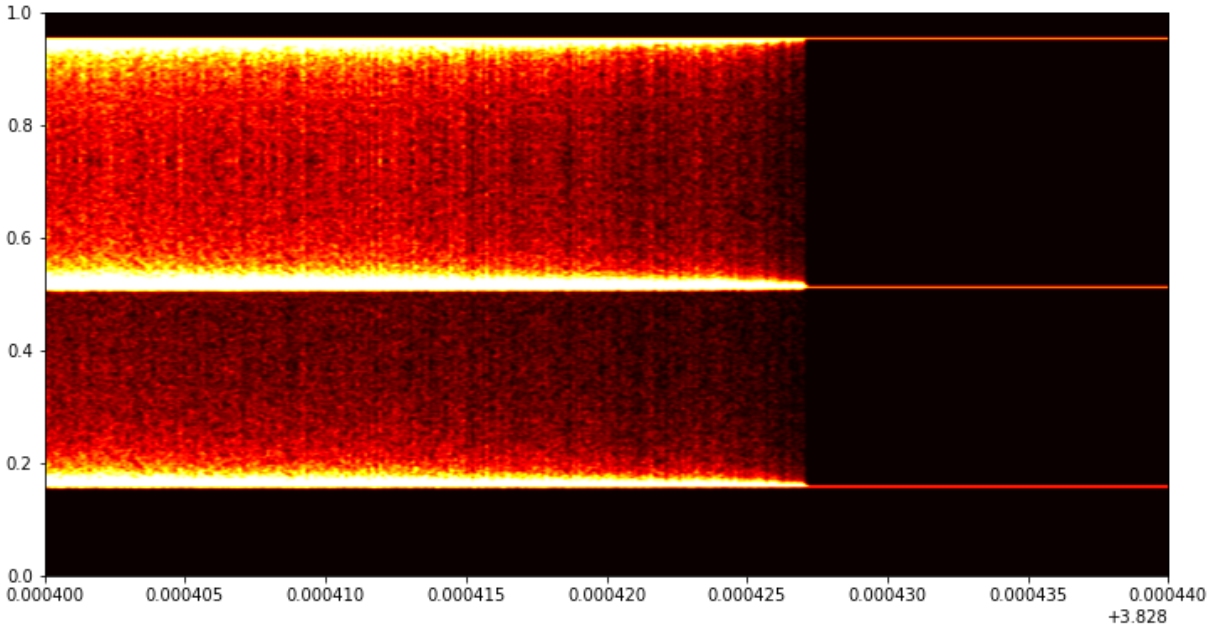
In [27]:
```
cascade(3.73, 3.75, 0, 1, niter=10000)
```



6.6 Sortie du chaos

Chaos ... et puis plus de chaos. Transition brutale (vraiment si brutale ?).

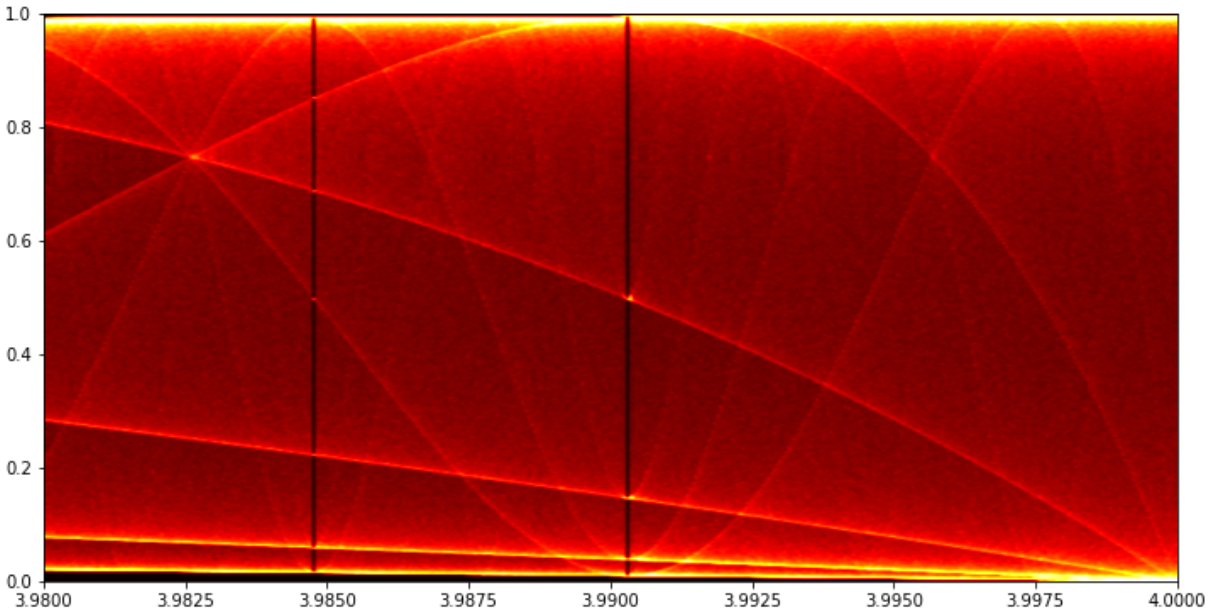
In [28]:
```
cascade(3.8284, 3.82844, 0, 1, contrast=500, niter=10000)
```



6.7 Le chaos

Du côté de chez $\mu = 4$. Soyez patient, j'ai mis `niter` à 50000.

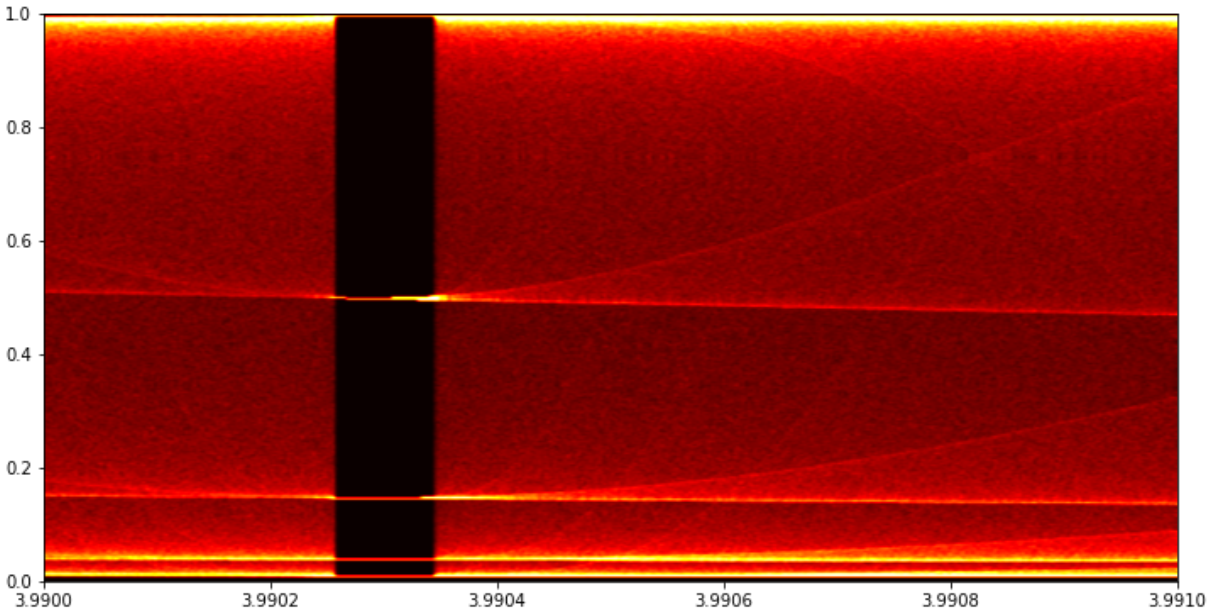
In [29]:
```
cascade(3.98, 4, 0, 1, contrast=100, niter=50000)
```



6.8 Le chaos ?

On peut montrer que les zones de calme sont **denses**. Ci-dessous, choix d'une zone au hasard.

In [30]:
```
cascade(3.99, 3.991, 0, 1, contrast=100, niter=20000)
```



7. Mais c'est quoi le chaos ?

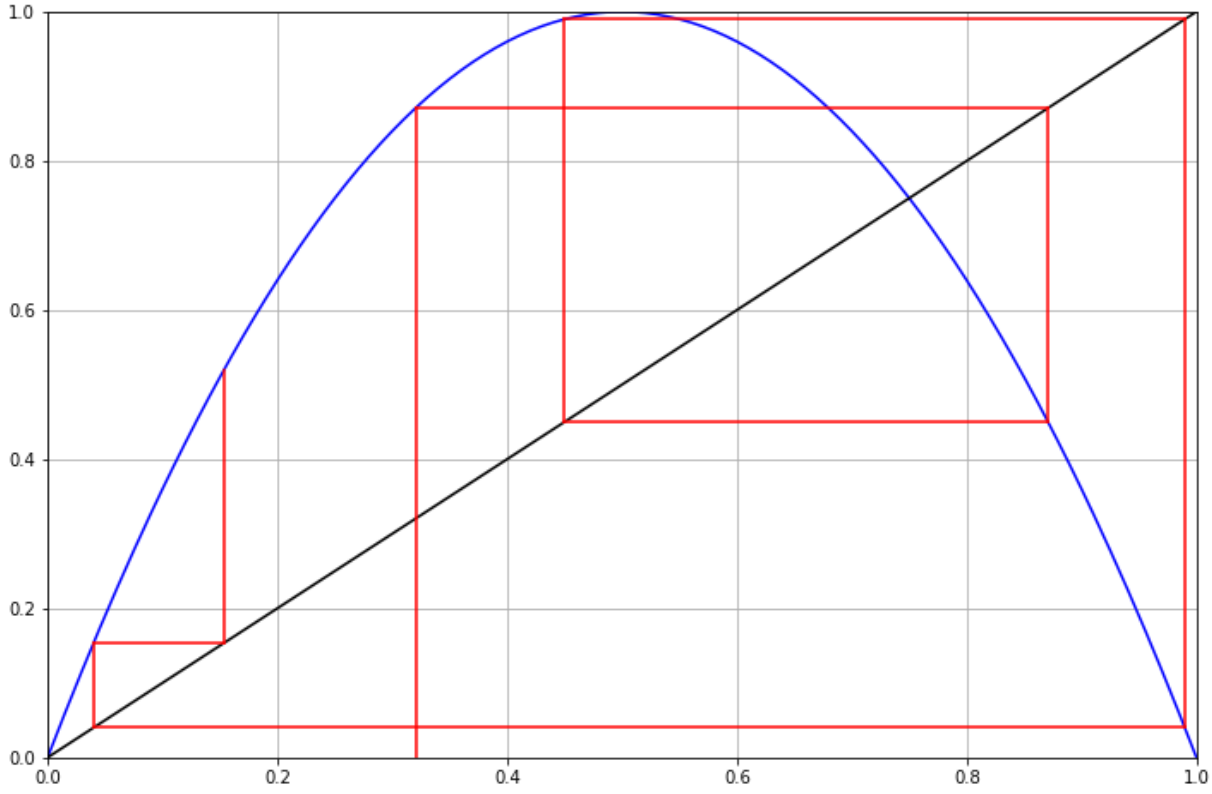
Pas facile de ***définir*** le chaos, mais on s'accorde à dire que les trois points ci-dessous caractérisent la présence du chaos. Dans la suite je suppose que $\mu = 4$.

7.1 Sensibilité aux conditions initiales

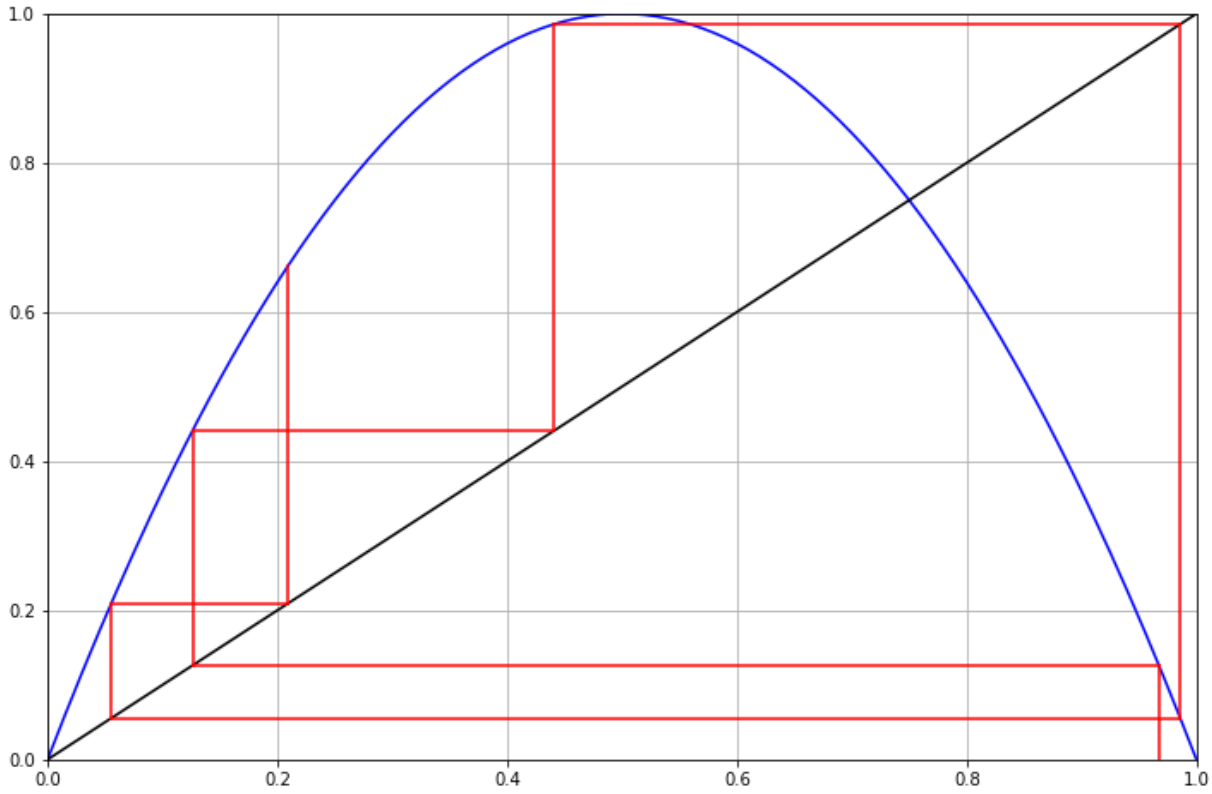
SCI : Il existe $\delta > 0$ tel que pour "tout" $x \in [0, 1]$, il existe des $y \in [0, 1]$ aussi proches de x qu'on le désire, mais tels que pour un certain k, $|f^k(y) - f^k(x)| \geq \delta$.

Je ne détaillerai pas les guillemets de pour "tout" ... Disons que sauf cas très particulier, la moindre perturbation de la valeur initiale x_0 change complètement les valeurs des x_n , même pour de petites valeurs de n. Ci-dessous, deux graphes sans grand rapport. Et pourtant, les valeurs initiales diffèrent de 10^{-5}. Dans un cas x_{30} vaut à peu près 0.3, dans l'autre cas, il vaut presque 1.

In [31]:
```
escalier(4, 5, x0=0.1, skip=30)
```



In [32]:
```
escalier(4, 5, x0=0.10001, skip=30)
```



7.2 Mélange

Mélange : pour tous intervalles ouverts I, J inclus dans $[0, 1]$, il existe $x_0 \in I$ tel que, pour un certain entier n, on ait $x_n \in J$.

Et même une infinité de $x_0 \in I$, et une infinité de $x_n \in J$! Imaginez des intervalles I et J de longueur 1 micron, choisis au hasard. Il existe une infinité de $x_0 \in I$ tels qu'une infinité de termes de la suite (x_n) soient dans J.

Dit autrement, en partant de n'importe où, on peut arriver où on veut.

7.3 Points périodiques denses

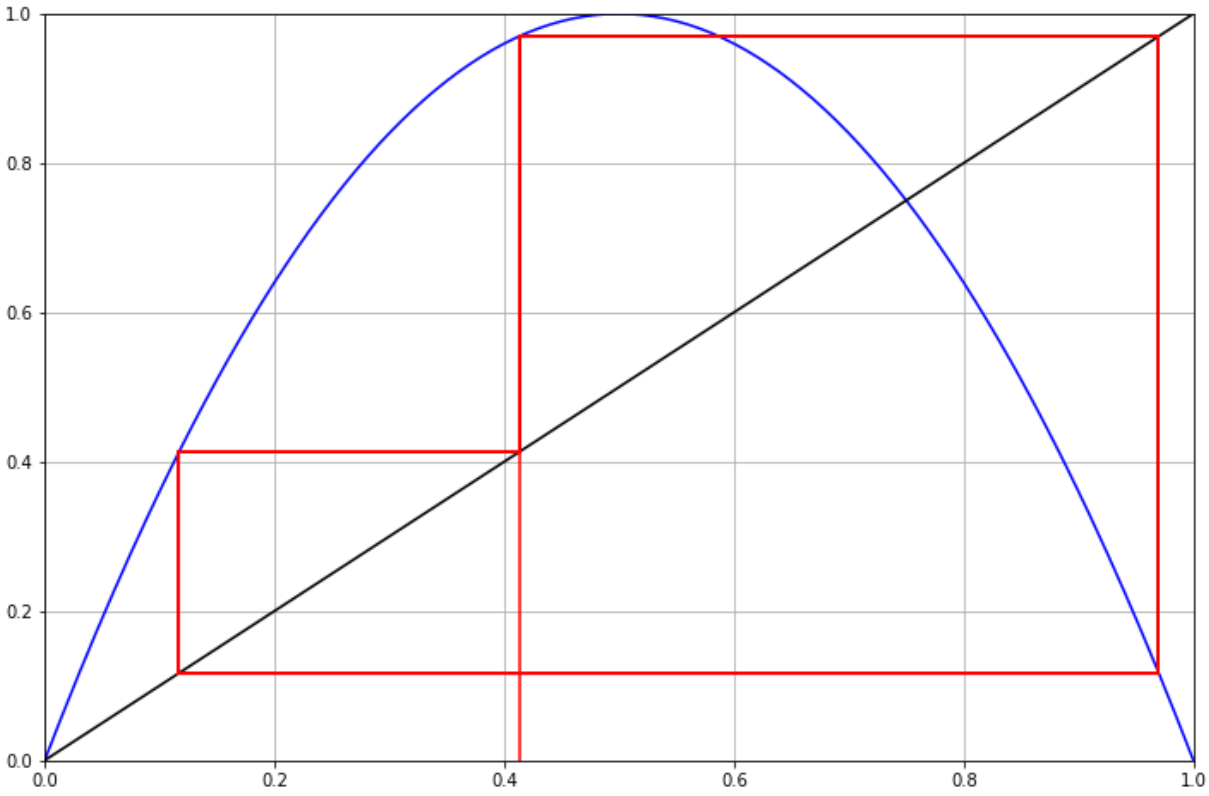
Soit $x \in [0, 1]$. La période de x (si elle existe) est le plus petit entier $n > 0$ tel que $f^n(x) = x$ (où f^n désigne $f \circ \ldots \circ f$). Par exemple, les points fixes sont les points de période 1.

Densité : l'ensemble des points périodiques est dense dans $[0, 1]$.

Propriété impossible à visualiser. En effet, les points périodiques sont tous répulsifs ; la sensibilité aux conditions initiales enpêche de voir la période d'un point. La moindre erreur sur la valeur d'un point le rend non-périodique, si j'ose dire.

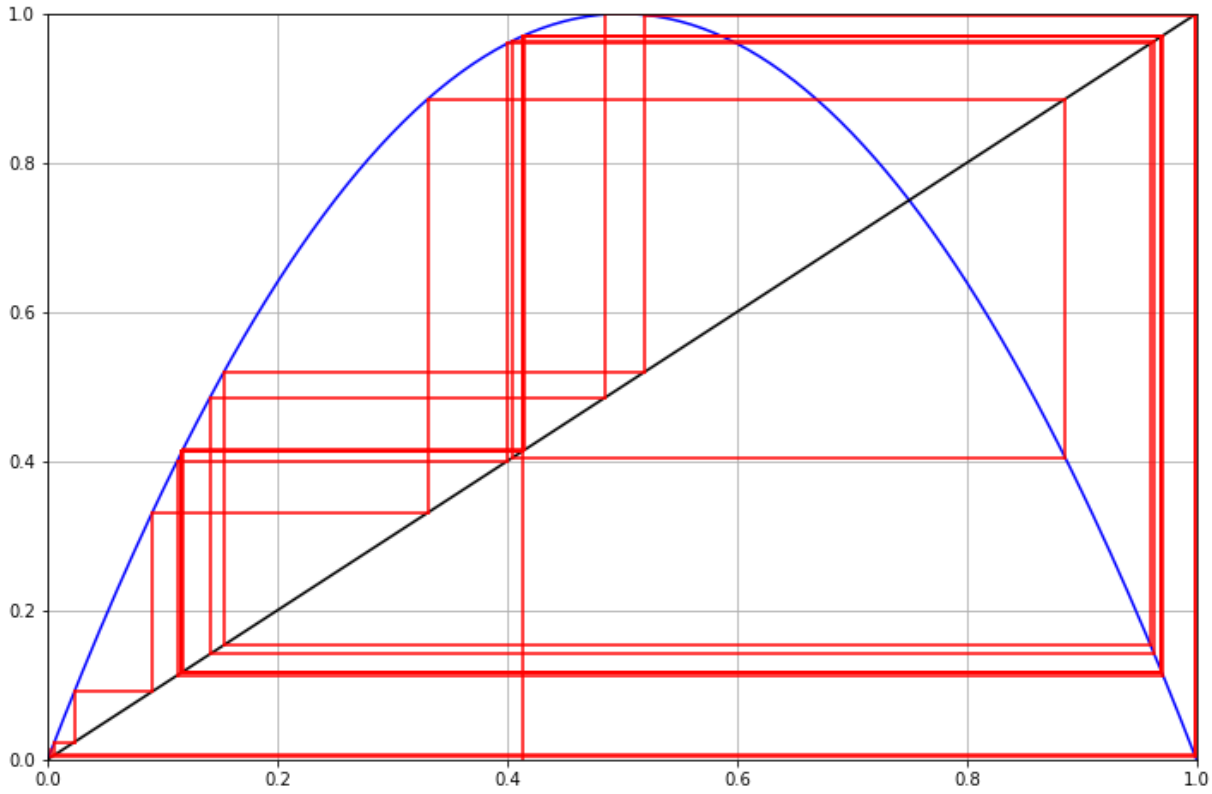
J'essaie quand même. Il y a un point de période 3 pas loin de 0.413175911166535 :

In [33]:
```
escalier(4, 10, x0=0.413175911166535,niter=1)
```



Mais voici ce qui arrive après 60 itérations. La sensibilité aux conditions initiales rend toute simulation numérique nulle et non avenue. Quoique ... mais ceci est une autre histoire.

In [34]:
```
escalier(4, 60, x0=0.413175911166535,niter=1)
```



Théorème : s'il y a un point de période 3, alors pour tout $n \geq 1$ il y a un point de période n.

Cherchez bien, il y a un point de période 2018 :-).

In [ ]:
```
```