タイトル

名前

日付

はじめに

- 項目 1
- 項目 2
  - 1 階層下の項目 1
  - 1 階層下の項目 2
- このページの最後の項目

次のスライド

- Ficititious play の説明 1
- Ficititious play の説明 2

  $x_0(t)$ は

  $$x_0(t+1) = x_0(t) + \frac{1}{t+2}(a_1(t) - x_0(t))$$

  と再帰的に書くことができる.
- Ficititious play の説明 3
- “pause” をつけると overlay ができる.
- ファイルの冒頭の documentclass のオプションで handout を指定すると，overlay にならずいっぺんに表示される.

  Web にのせるときや，印刷して配るときなどは handout を指定しておく.

次のスライド

- Ficititious play の説明 1
- Ficititious play の説明 2

  $x_0(t)$ は

  $$x_0(t+1) = x_0(t) + \frac{1}{t+2}(a_1(t) - x_0(t))$$

  と再帰的に書くことができる.
- Ficititious play の説明 3
- “pause” をつけると overlay ができる.
- ファイルの冒頭の `documentclass` のオプションで `handout` を指定すると，overlay にならずいっぺんに表示される.

  Web にのせるときや，印刷して配るときなどは `handout` を指定しておく.

次のスライド

- Ficititious play の説明 1
- Ficititious play の説明 2

  $x_0(t)$ は

  $$x_0(t+1) = x_0(t) + \frac{1}{t+2}(a_1(t) - x_0(t))$$

  と再帰的に書くことができる.
- Ficititious play の説明 3
- “pause” をつけると overlay ができる.
- ファイルの冒頭の documentclass のオプションで handout を指定すると，overlay にならずいっぺんに表示される.

  Web にのせるときや，印刷して配るときなどは handout を指定しておく.

次のスライド

- Ficititious play の説明 1
- Ficititious play の説明 2

  $x_0(t)$ は

  $$x_0(t+1) = x_0(t) + \frac{1}{t+2}(a_1(t) - x_0(t))$$

  と再帰的に書くことができる.
- Ficititious play の説明 3
- “pause” をつけると overlay ができる.
- ファイルの冒頭の `documentclass` のオプションで `handout` を指定すると，overlay にならずいっぺんに表示される.

  Web にのせるときや，印刷して配るときなどは `handout` を指定しておく.

コードの説明

- ライブラリのインポートとクラスの定義

```
import matplotlib.pyplot as plt
import random
import numpy as np
```

- matplotlib.pyplot はグラフのプロットに使用
- random は初期信念と行動を定めるのに使用
- numpy は行列計算に使用

コードの説明

- ライブラリのインポートとクラスの定義

```
import matplotlib.pyplot as plt
import random
import numpy as np
```

- matplotlib.pyplot はグラフのプロットに使用
- random は初期信念と行動を定めるのに使用
- numpy は行列計算に使用

コードの説明

- ライブラリのインポートとクラスの定義

```
import matplotlib.pyplot as plt
import random
import numpy as np
```

- matplotlib.pyplot はグラフのプロットに使用
- random は初期信念と行動を定めるのに使用
- numpy は行列計算に使用

コードの説明

- ライブラリのインポートとクラスの定義

```
import matplotlib.pyplot as plt
import random
import numpy as np
```

- matplotlib.pyplot はグラフのプロットに使用
- random は初期信念と行動を定めるのに使用
- numpy は行列計算に使用

コードの説明

- ライブラリのインポートとクラスの定義

```
import matplotlib.pyplot as plt
import random
import numpy as np
```

- matplotlib.pyplot はグラフのプロットに使用
- random は初期信念と行動を定めるのに使用
- numpy は行列計算に使用

コードの説明とか

- ```
  class FP:

      def __init__(self, profits):
          self.pro = profits
          self.cu_xs = [0, 0]
          self.x0s = []
          self.x1s = []
  ```

- いくつかのメソッドをもつクラスを FP として定義
- FP を使う際には f = FP(profits) のように打つ必要がある
- profits は利得表を表す行列である
- init メソッド中でインスタンス変数をいくつか定義している　これはクラス内でのみ共有される　複数のメソッドで用いる変数をここに記した
  ```

コードの説明とか

- ```
  class FP:

      def __init__(self, profits):
          self.pro = profits
          self.cu_xs = [0, 0]
          self.x0s = []
          self.x1s = []
  ```

- いくつかのメソッドをもつクラスを FP として定義
- FP を使う際には f = FP(profits) のように打つ必要がある
- profits は利得表を表す行列である
- init メソッド中でインスタンス変数をいくつか定義している　これはクラス内でのみ共有される　複数のメソッドで用いる変数をここに記した
  ```

コードの説明とか

- ```
  class FP:

      def __init__(self, profits):
          self.pro = profits
          self.cu_xs = [0, 0]
          self.x0s = []
          self.x1s = []
  ```

- いくつかのメソッドをもつクラスを FP として定義
- FP を使う際には f = FP(profits) のように打つ必要がある
- profits は利得表を表す行列である
- init メソッド中でインスタンス変数をいくつか定義している　これはクラス内でのみ共有される　複数のメソッドで用いる変数をここに記した
  ```

コードの説明とか

- ```
  class FP:

      def __init__(self, profits):
          self.pro = profits
          self.cu_xs = [0, 0]
          self.x0s = []
          self.x1s = []
  ```

- いくつかのメソッドをもつクラスを FP として定義
- FP を使う際には f = FP(profits) のように打つ必要がある
- profits は利得表を表す行列である
- init メソッド中でインスタンス変数をいくつか定義している
  これはクラス内でのみ共有される　複数のメソッドで用いる変数をここに記した
  ```

コードの説明とか

- ```
  class FP:

      def __init__(self, profits):
          self.pro = profits
          self.cu_xs = [0, 0]
          self.x0s = []
          self.x1s = []
  ```

- いくつかのメソッドをもつクラスを FP として定義
- FP を使う際には f = FP(profits) のように打つ必要がある
- profits は利得表を表す行列である
- init メソッド中でインスタンス変数をいくつか定義している
  これはクラス内でのみ共有される　複数のメソッドで用いる変数をここに記した
  ```

コードの説明とか

- ```
  class FP:

      def __init__(self, profits):
          self.pro = profits
          self.cu_xs = [0, 0]
          self.x0s = []
          self.x1s = []
  ```

- いくつかのメソッドをもつクラスを FP として定義
- FP を使う際には f = FP(profits) のように打つ必要がある
- profits は利得表を表す行列である
- init メソッド中でインスタンス変数をいくつか定義している
  　これはクラス内でのみ共有される　複数のメソッドで用いる変数をここに記した
  ```

コードの説明とか

- コードの表示の例

```
def oneplay(self, ts_length):
    pro_cal = (np.transpose(self.pro)[0],
               np.transpose(np.transpose(self.pro)[1]))
    self.x0s = []
    self.x1s = []
    self.cu_xs = [random.uniform(0, 1), random.uniform(0, 1)]
    cu_es = [[0, 0], [0, 0]]
    for i in range(ts_length):
        self.x0s.append(self.cu_xs[0])
        self.x1s.append(self.cu_xs[1])
        exp = ((1-self.cu_xs[0], self.cu_xs[0]),
               (1-self.cu_xs[1], self.cu_xs[1]))
        cu_es[0] = np.dot(pro_cal[0], exp[0])
        cu_es[1] = np.dot(pro_cal[1], exp[1])
        cu_as = [0, 0]  # the act of players
        for j in range(2):
            if cu_es[j][0] == cu_es[j][1]:
                cu_as[j] = random.randint(0, 1)
            else:
                cu_as[j] = cu_es[j].argmax()
        for k in range(2):
            self.cu_xs[k] = (self.cu_xs[k]*(i+1)+cu_as[1-k])/(i+2)
            # x(i) to x(i+1)
```

図

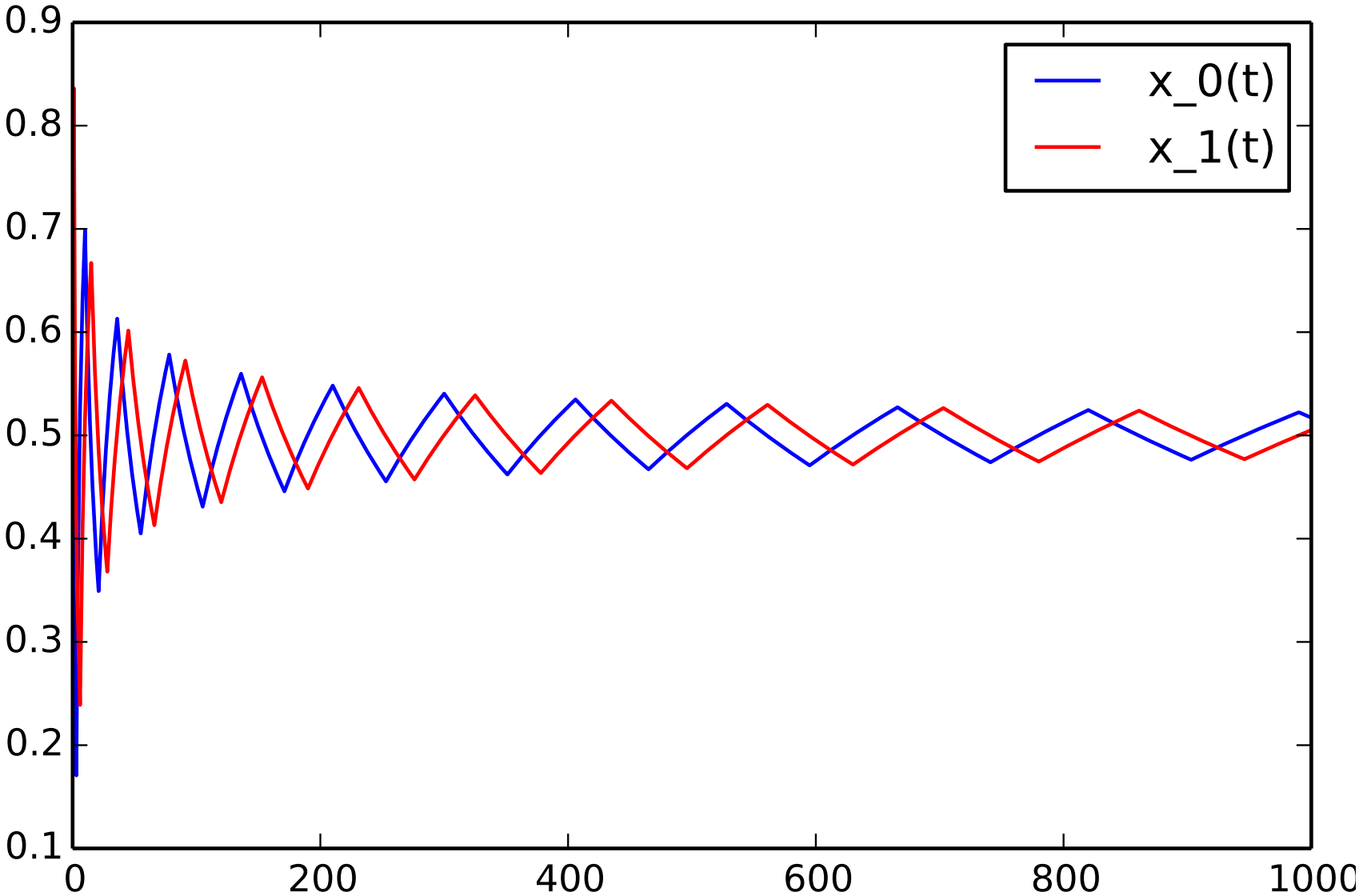

Figure : 図の表示

まとめ

- まとめ
- よくわかっていない点とか
- 今後の課題とか