SkaiWD – Laboratorium 1

Wprowadzenie do wizualizacji danych w Pythonie

Iwo Błądek

13 marca 2019

1 Python

(*Jeżeli znasz Pythona, to możesz przejść do następnej sekcji.*)

Wykonaj przynajmniej jeden z poniższych punktów:

- przeczytaj szybkie omówienie składni języka na: `https://www.stavros.io/tutorials/python/`.
- przerób podstawowe zagadnienia z dość obszernego tutoriala na pyschools: `http://www.pyschools.com` (zakładka *Practices*). Wymaga zalogowania przez konto Google, można szybko utworzyć jakieś specjalnie do tego celu.
- znajdź samemu w internecie odpowiedni dla siebie tutorial.

2 Numpy

(*Jeżeli już znasz tę bibliotekę, to możesz przejść do następnej sekcji.*)

Zapoznaj się z tutorialem na stronie `https://docs.scipy.org/doc/numpy-1.15.1/user/quickstart.html` (do sekcji *Shape Manipulation*). Zwróć uwagę na działanie i sposób wykorzystania następujących funkcji/klas:

- `ndarray` (standardowa macierzowa klasa w numpy): konstruktory, dodawanie, odejmowanie, mnożenie.
- `ones`, `zeros`
- `identity` (do tworzenia macierzy jednostkowej)
- `arange`, `linspace`

Podstawowe informacje na razie wystarczą.

3 Matplotlib

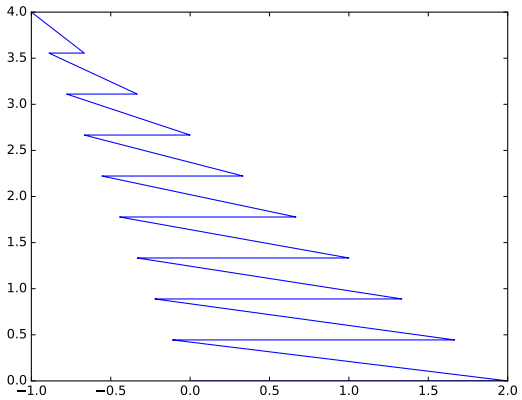
(*Jeżeli już znasz tę bibliotekę, to możesz ~~iść do domu~~ przejść od razu do zadań.*)

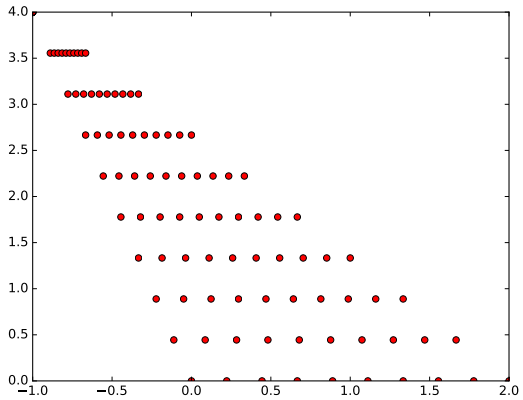
Przeczytaj tutorial na stronie `http://matplotlib.org/users/pyplot_tutorial.html`.

4 Zadania: wizualizacja kombinacji liniowej

Zadanie 4.1: Do tego zadania i następnych wykorzystaj szablon w Pythonie dostępny na stronie.

Dany jest trójkąt o następujących wierzchołkach: $A = \begin{bmatrix}-1 & 4\end{bmatrix}$, $B = \begin{bmatrix}2 & 0\end{bmatrix}$, $C = \begin{bmatrix}0 & 0\end{bmatrix}$. Każdy punkt wewnątrz tego trójkąta może zostać uzyskany przez pewną wypukłą kombinację liniową wierzchołków. Przedstaw na wykresie punkty, które powstaną przez wypukłą kombinację liniową wierzchołków dla współczynników λ_i iterując po ich możliwych wartościach (wykorzystaj `np.linspace` z liczbą wartości 10). Obliczenia zaimplementuj w funkcji `convex_comb_triangle_loop` w szablonie, a rysowanie wykresu w `draw_convex_combination_2d`. Pamiętaj, że współczynniki muszą się sumować do 1. Wykres powinien wyglądać mniej więcej tak jak te poniżej (rysowanie punktów może być w dowolnej kolejności).





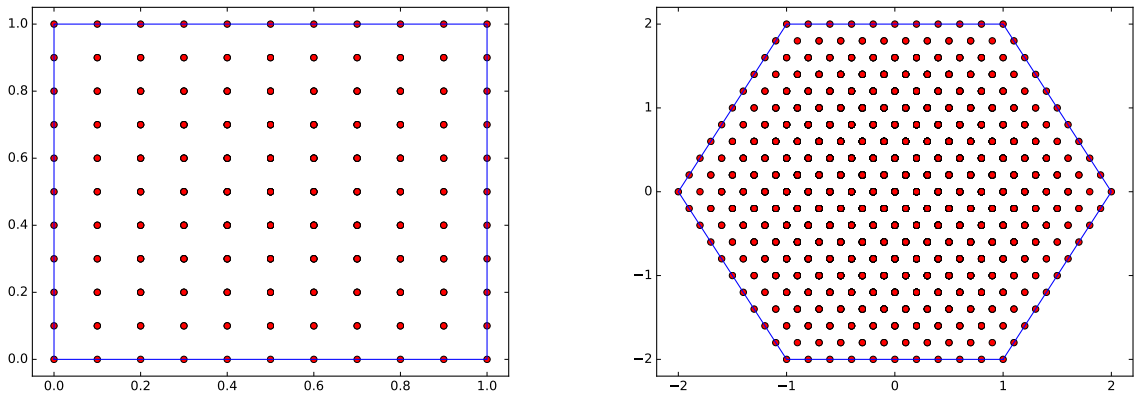
Wykres po prawej powstał przez dodanie do funkcji `np.plot` argumentu "ro". Matplotlib w funkcji `plot` domyślnie łączy punkty linią, co widać po lewej stronie (kształt krzywej zależy od kolejności iteracji po współczynnikach liniowych). Osie wykresu nakładają się na rysowane wewnątrz elementy, co nie wygląda dobrze. Problem można rozwiązać dodając w kodzie:

- `plt.margins(0.05)` – obszar wewnętrzny wykresu zostanie poszerzony we wszystkich kierunkach o wskazaną wartość.

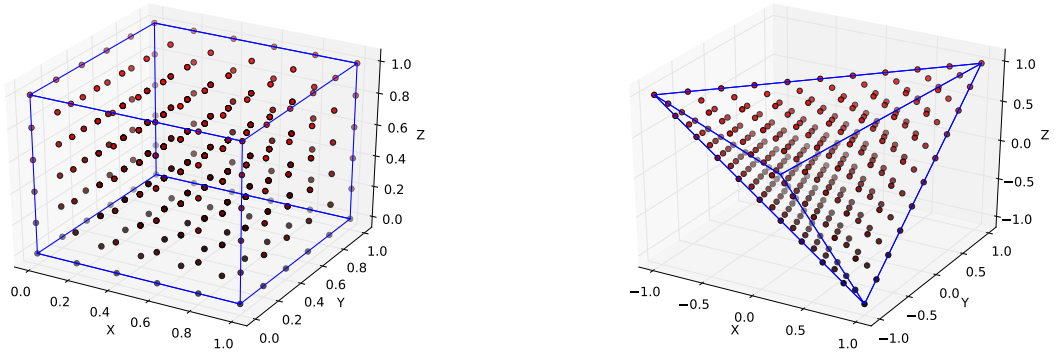
Zadanie 4.2: Uogólnimy teraz program z poprzedniego zadania dla dowolnego wielokąta wypukłego zadanego zbiorem wierzchołków. Zaimplementuj w szablonie funkcję `convex_comb_general`, a następnie przetestuj ją dla różnych figur.

Wejściem funkcji `convex_comb_general` jest lista wierzchołków, z których każdy jest tablicą w numpy (wektorem). Spróbuj tak zaimplementować tę funkcję, by działała dla dowolnej liczby wymiarów (w następnym zadaniu będzie tworzony wykres trójwymiarowy). Tym razem do iteracji po współczynnikach użyj funkcji `arange`.

Oczekiwane rezultaty:

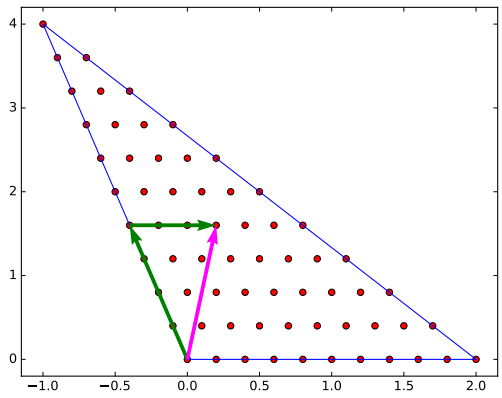


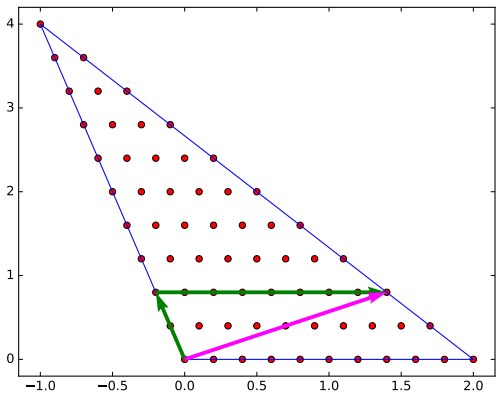
Zadanie 4.3: Dobrze napisany program z poprzedniego zadania powinien działać bez problemu również dla trzech wymiarów. Zaimplementuj więc w funkcji `draw_convex_combination_3d` wizualizację wypukłych kombinacji liniowych wektorów trójwymiarowych na wykresie trójwymiarowym. Generalnie operacje na wykresie 3D wyglądają w numpy praktycznie tak samo jak na dwuwymiarowym, tylko należy utworzyć wykres w specyficzny sposób (`projection='3d'`) i podawać w funkcjach rysujących dodatkową listę współrzędnych punktów (oś z). Zalecana do użycia funkcja to `ax.scatter`. Poniżej przykład sześcianu oraz czworościanu foremnego (ang. *regular tetrahedron*):



Możesz pokusić się o pokolorowanie punktów zgodnie z wartością współrzędnej z. W funkcji `scatter` można podać dodatkową listę zawierającą informację o kolorze dla każdego wierzchołka. Przykład: `ax.scatter(xs, ys, zs, c=colors)`.

Zadanie 4.4: Zajmiemy się teraz wizualizacją składania wektorów na płaszczyźnie. Uzupełnij w kodzie funkcję `draw_vector_addition`, która pokaże za pomocą strzałek wynik kombinacji liniowej wektorów w zależności od dobranych współczynników. Wykorzystaj `plt.arrow` (alternatywnie `plt.quiver`; po angielsku *quiver* to kołczan). Możesz nałożyć strzałki na wcześniej utworzone wykresy by zobaczyć, w jaki sposób otrzymane zostały poszczególne punkty.





By strzałki były rysowane nad elementami wykresu a nie pod spodem, trzeba dodać argument `zorder=value` do poszczególnych elementów wykresu, przy czym elementy z niższymi wartościami `value` są rysowane wcześniej. Domyślnie elementy wykresu okupują wartości mniejsze lub równe 3.