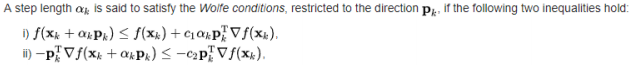


**Kod w języku python:**

|  |
| --- |
| *import* numpy # operacje macierzowe *from* scipy.optimize *import* minimize # metoda szukania minimum *import* random # losowanie *import* matplotlib.pyplot *as* plt # wykres *from* mpl\_toolkits.mplot3d *import* Axes3D # wykres 3d *from* matplotlib *import* cm # kolorowanie wykresu *import* sys # argumenty wywolania programu   # zwraca wartosc funkcji rosenbrocka *def* rosenbrock(*p*):  *return* ((1 - *p*[0]) \* (1 - *p*[0]) + 100 \* (*p*[1] - *p*[0] \* *p*[0]) \* (*p*[1] - *p*[0] \* *p*[0]))   # dodaje punkt do listy krokow metody CG *def* add\_point(*x*):  *global* steps  steps.append(*x*)   # globalna zmienna pokazujaca kolejne kroki minimalizacji steps = []   *if* (len(sys.argv) != 3):  print("Opcje wywolania programu")  print("<nazwa> <liczba punktow do wylosowania> <granica> <tryb>")  print("<granica>- w jakich granicach program ma rysowac wykres- na przyklad 20 da granice [-20,20]")  exit(0)  # 10 punktow maksymalnie dla trybu rysowania ile\_punktow = int(sys.argv[1]) granica = int(sys.argv[2])  # przygotowanie do wykresu fig = plt.figure() ax = fig.add\_subplot(111, projection='3d')  # rysowanie funkcji rosenbrocka X = numpy.linspace(-granica, granica, 1000) Y = numpy.linspace(-granica, granica, 1000) X, Y = numpy.meshgrid(X, Y) Z = rosenbrock([X, Y]) surf = ax.plot\_surface(X, Y, Z, cmap=cm.coolwarm, linewidth=0, antialiased=*False*)  color = ['black', 'green', 'gray', 'red', 'cyan', 'magenta', 'yellow', 'white', 'blue', 'orange']  *for* i *in* range(ile\_punktow):  # wylosowanie punktu  p = [random.uniform(-granica, granica), random.uniform(-granica, granica)]  print(f"Wylosowano punkt ({p[0]},{p[1]})")  steps.append(p)   # zastosowanie minimalizacji metoda gradientow sprzezonych  found = minimize(rosenbrock, p, method='CG', callback=add\_point)  print(f"Znalezione minimum ({found.x[0]}, {found.x[1]}) wartosc funkcji wynosi= {rosenbrock(found.x)}, liczba iteracji={len(steps)}")  print()   # dodawanie sciezki do wykresu  temp\_x = [i[0] *for* i *in* steps]  temp\_y = [i[1] *for* i *in* steps]  temp\_z = [rosenbrock([temp\_x[i], temp\_y[i]]) *for* i *in* range(0, len(temp\_x))]  ax.plot3D(temp\_x, temp\_y, temp\_z, color=color[i], linestyle='dashed', linewidth=2, markersize=12)  steps.clear()  plt.show()  **Przykładowe wywołanie programu:**    **Wyniki:**        **Omówienie wyników:**  Metodą gradientów sprzężonych daje szybkie i dokładne wyniki. Nie zawsze jednak znajduje minimum globalne(w niektórych przypadkach znajduje minimum lokalne – drugi wykres). Algorytm ten pozwala znaleźć minimum w dość małej liczbie iteracji.  **Opis metody:**  Przy szukaniu minimum funkcji Rosenbrocka skorzystałem z biblioteki SciPy. Algorytm korzysta z wariantu metody Fletchera-Reevesa do wyznaczania minimum funkcji wielowymiarowej. |

Następnie algorytm wylicza alfę dopóki nie będą spełnione dwa warunki Wolfe’a:



Kolejne alfy wyliczane są za pomocą algorytmu wyszukiwania More’a i Thuente’a.