**Analiza wpływu metod uwzględniania ograniczeń kostkowych na macierz kowariancji w algorytmie CMA-ES**

Michał Urbański

28.10.2021

* Problem

Wprowadzenie ograniczeń kostkowych do algorytmu CMA-ES znacząco zmienia jego zachowanie.  
W szczególności, w przypadku funkcji unimodalnych, może zmniejszyć tempo zbieżności do optimum. Zadaniem pracy jest znalezienie sposobu poprawy przebiegu algorytmu, aby z ograniczeniami osiągał wyniki tak szybko jak bez nich. Rozważam podstawową wersję algorytmu CMA-ES, optymalizuję funkcję unimodalną – odpowiednio przeskalowaną funkcję kwadratową

* Streszczenie wcześniejszych postępów (13.09.2021):

Próby znalezienia dokładnego, analitycznego rozwiązania nie powiodły się. Wymagałoby to znajomości dokładnej postaci funkcji celu, do której algorytm nie ma dostępu. Zamiast tego, szukam rozwiązania wykorzystującego wewnętrzne parametry CMA-ES. Zacznę od przyjrzenia się dynamice zmian parametrów algorytmu – sigma oraz macierzy kowariancji w zależności od położenia optimum funkcji kwadratowej względem ograniczającej kostki, rozmiaru tej kostki oraz wybranej metody uwzględniania ograniczeń. Spodziewam się, że stosowanie metod uwzględniania ograniczeń powoduje zbyt powolne zbliżanie się środka populacji do optimum, co skutkowałoby zbyt szybkim zanikaniem parametru sigma i macierzy kowariancji. Być może rozwiązaniem będzie obliczanie korekty na podstawie różnicy w średnim położeniu punktów przy stosowaniu ograniczeń i bez stosowania ograniczeń. Korekta ta pozwoliłaby dalej przesuwać się środkowi populacji w jednej iteracji, przez co parametry nie zanikałyby tak szybko.

* Postępy 18.10.2021:

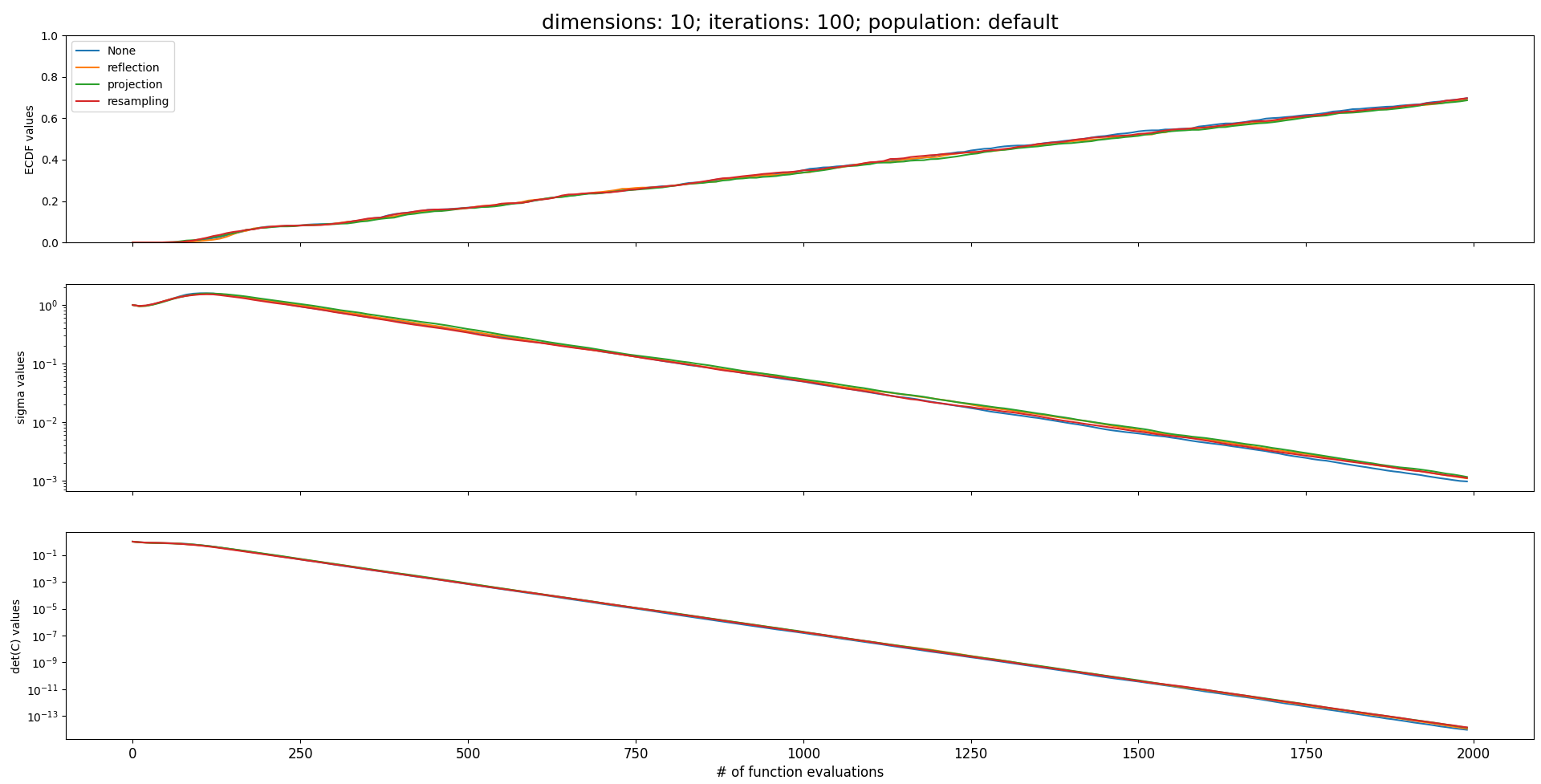
Aby dokładniej przyjrzeć się zmianom wymienionych wcześniej parametrów, wyróżnię trzy fazy przebiegu algorytmu.

- Pierwsza faza odpowiada sytuacji, w której populacja dopiero zmierza ku optimum i jest daleko od granic kostki. Wpływ metod naprawy jest pomijalny, więc algorytm z każdą metodą zachowa się tak samo. Faza ta może nie występować wcale, jeżeli początkowe położenie populacji jest zbyt blisko granic.

- W drugiej fazie algorytm zbliża się do optimum ze znaczącym wpływem metod uwzględniania ograniczeń.

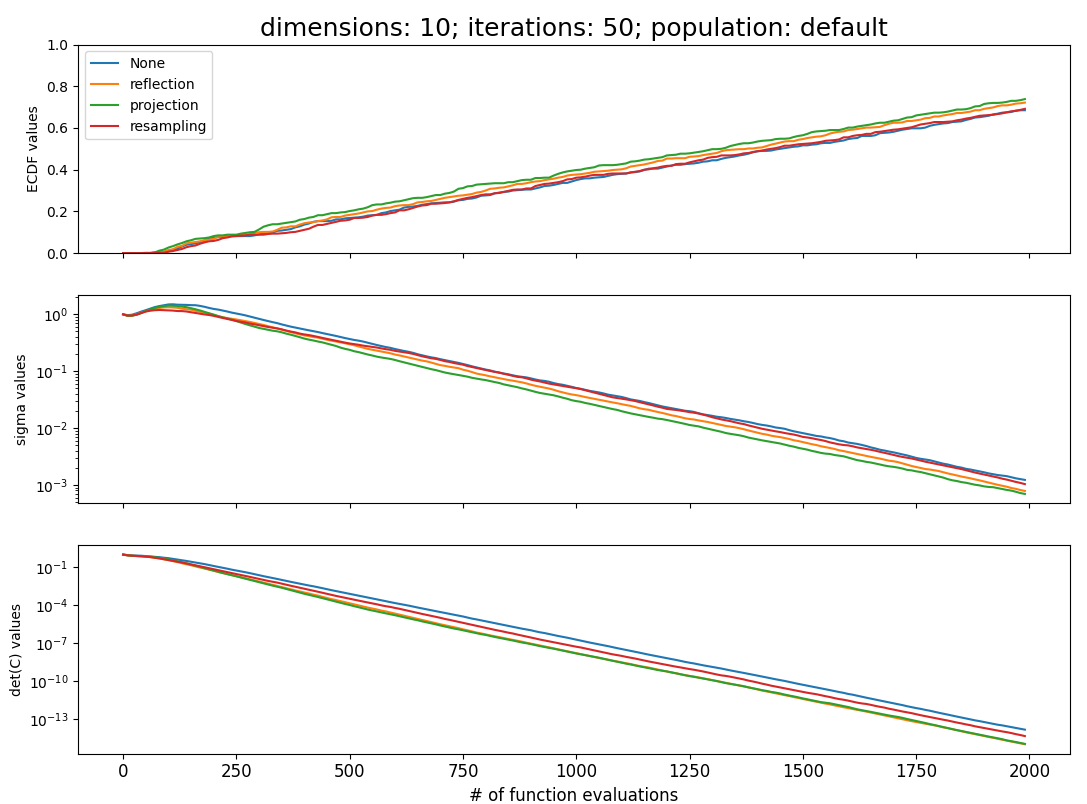
- W trzeciej fazie populacja zbiegła do optimum i rozproszenie punktów jest na tyle małe, że ponownie wpływ ograniczeń jest pomijalny.

Zbadałem szybkość zbieżności algorytmu, wartości sigma oraz wartości macierzy kowariancji w zależności od liczby iteracji algorytmu. Aby łatwiej zrozumieć jak wygląda macierz kowariancji, na wykresie zaznaczam wyznacznik tej macierzy. Z dobrym przybliżeniem opisuje on "wielkość" losowanej populacji. Szczegóły dotyczące badanych przebiegów algorytmu znajdują się na końcu dokumentu.

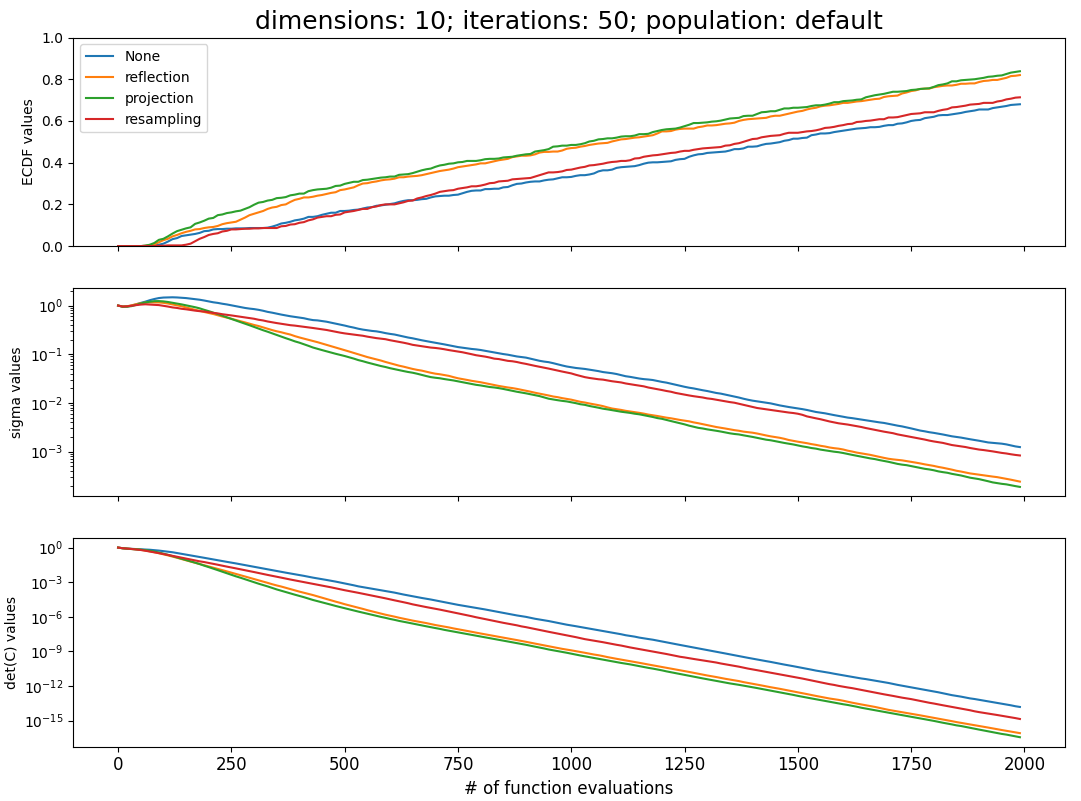
Na początku zbadany został przypadek, gdy algorytm znajduje się od razu w fazie trzeciej. Przedział dopuszczalny jest równy (-100, 100) w każdym wymiarze. Początkowe położenie populacji to punkt 3 w każdym wymiarze.

Zgodnie z oczekiwaniami, wszystkie badane metody zachowują się identycznie (z dokładnością do losowości). Sigma oraz macierz kowariancji zmniejszają się, a wartości ECDF rosną. Chwilowe spowolnienie w chwili 250 odpowiada znalezieniu przez populację optimum i dopasowaniu się do niej konturem.

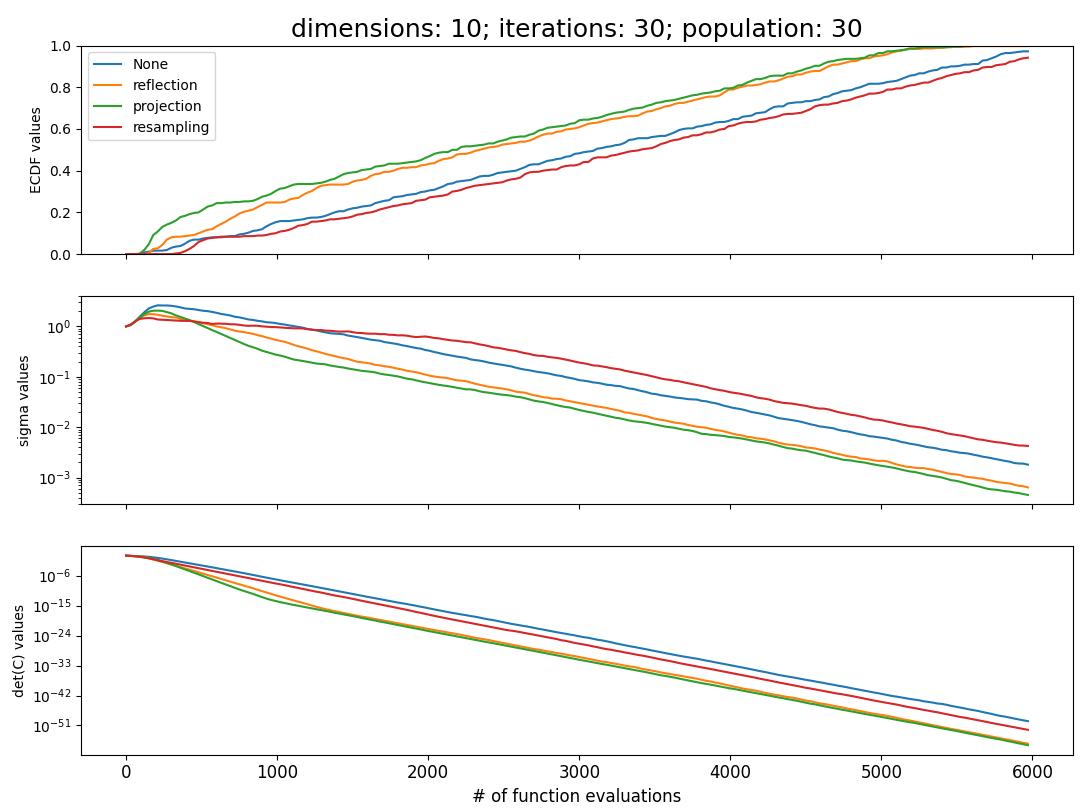
Teraz ograniczenia kostki zostaną przesunięte tak, by znajdować się blisko granic kostki. W połowie liczby wymiarów dolna granica kostki przesunięta zostaje do wartości -0.1.

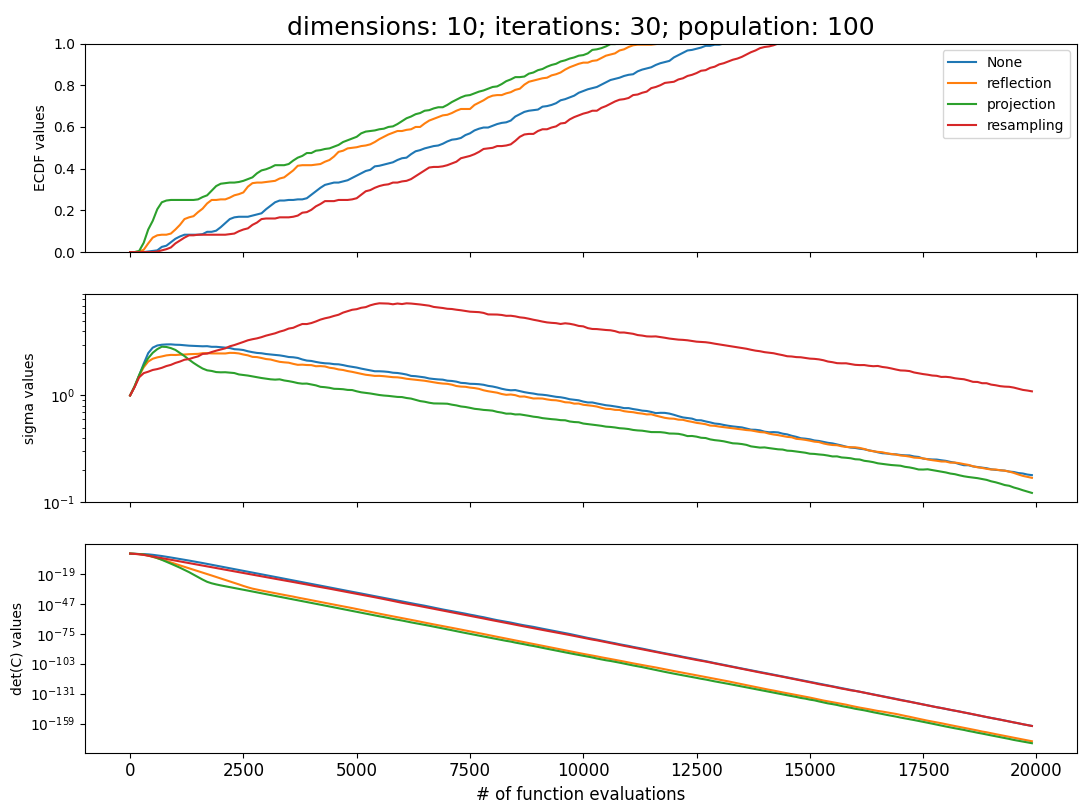


Wpływ kostki zaczyna być zauważalny, szczególnie w fazie drugiej – około chwili 120 na osi poziomej. Wpływ ten zwiększy się, gdy przesuniemy pozostałą połowę dolnych granic kostki do wartości -0.1:

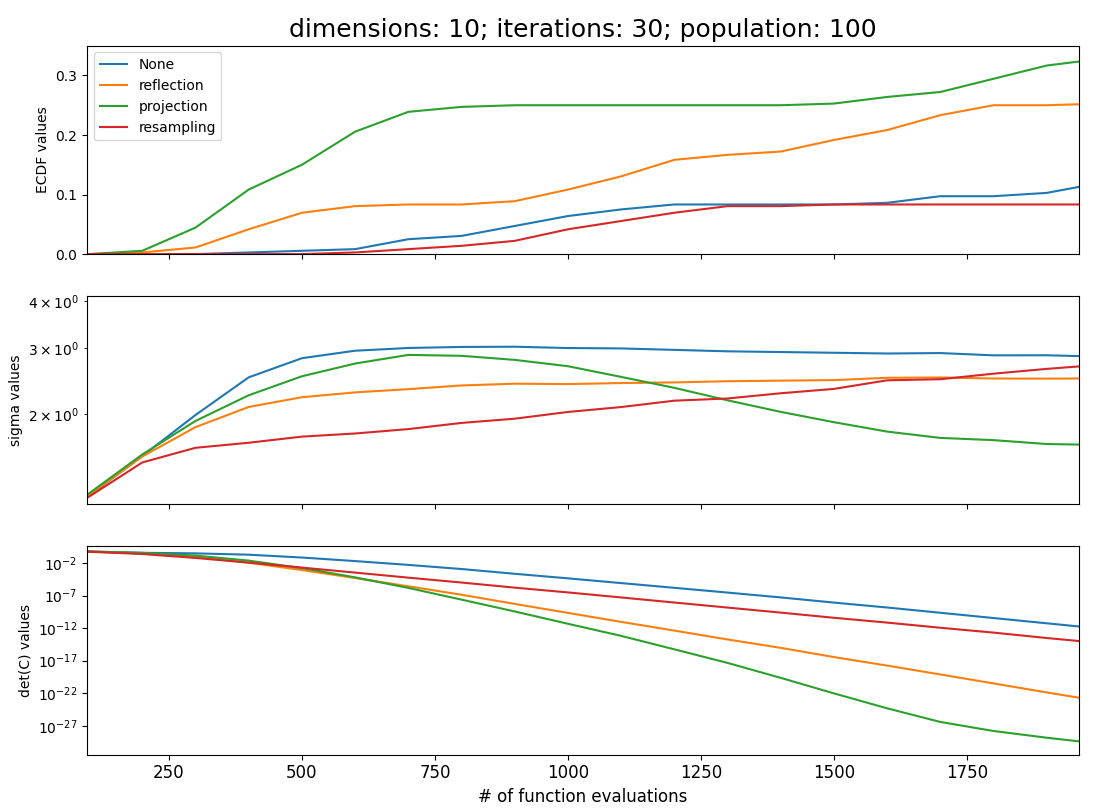


Podobnie, wpływ ten zwiększy się gdy zwiększymy liczebność populacji:



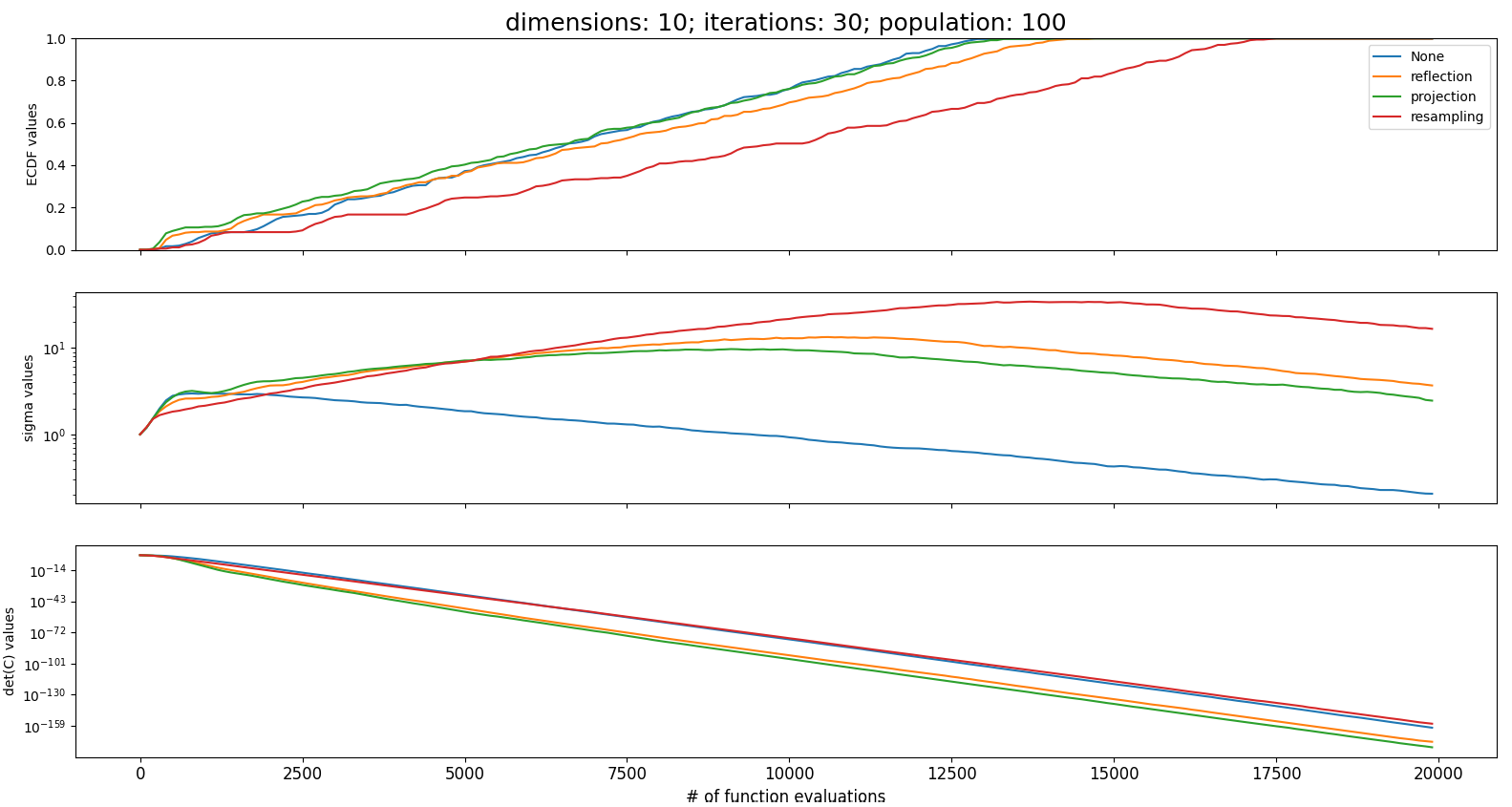


Przyjrzę się teraz wartościom sigma w interesującym przedziale. Jest to przedział odpowiadający fazie drugiej algorytmu, czyli wartościom mniej więcej 100-2000 na osi poziomej. Metoda resamplingu zachowuje się dużo inaczej od pozostałych, dlatego omówię ją osobno. Przy wykorzystaniu tej metody narożnik kostki "odpycha" populację, co powoduje że przybliża się ona do optimum małymi krokami. Macierz kowariancji zanika, ale przez to że populacja zbiega w jedną stronę, wartość sigma rośnie. Przez takie powolne przesuwanie, metoda ta osiąga fazę trzecią dopiero około chwili 5000 na osi poziomej i dalej zachowuje się jak inne metody. Pozostałe trzy metody są słabo widoczne na powyższym wykresie. Ten sam, powiększony wykres:



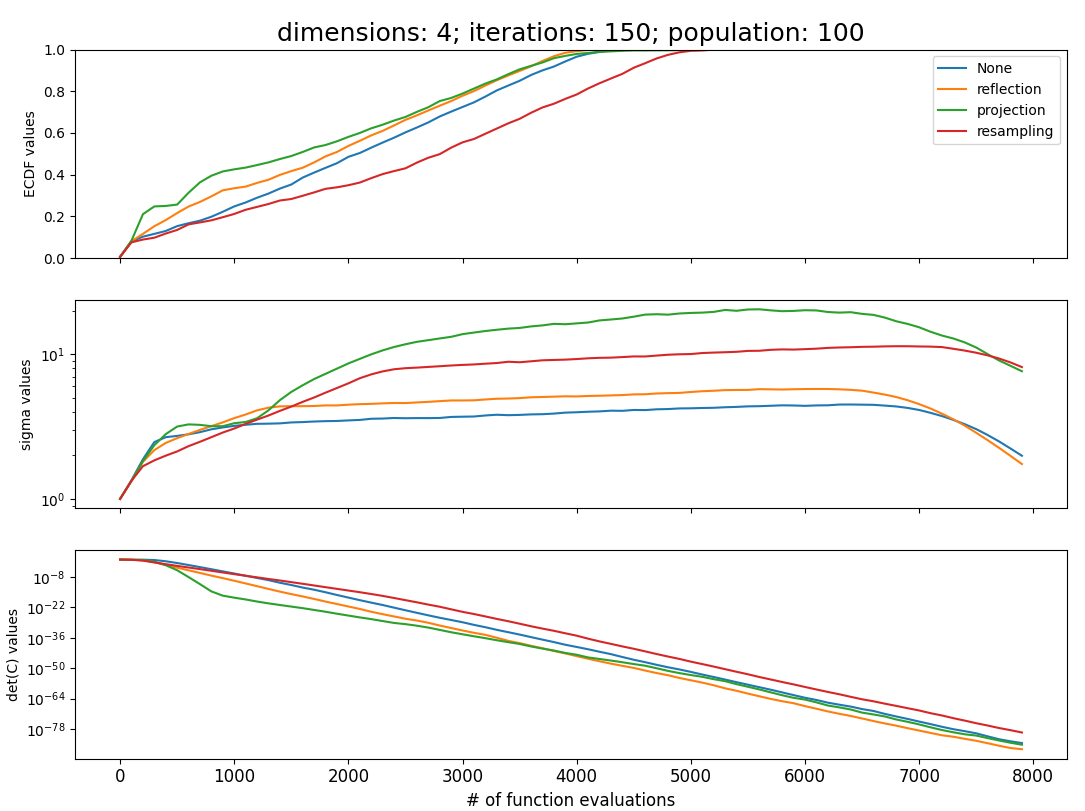
Wszystkie metody do chwili ~200 na osi poziomej zachowują się podobnie – odpowiada to fazie pierwszej. Metody odbijania i rzutowania (zielona i żółta) przesuwają wiele punktów w okolice optimum, przez co metody te szybciej osiągają lepsze wartości ECDF. Rzutowane punkty lądują bardzo często w punkcie (-0.1, -0.1, …), co powoduje bardzo szybkie przesuwanie populacji do optimum, przez co wartość sigma szybko rośnie. Gdy środek populacji znajdzie się w optimum, wartości sigma i C zmniejszają się i algorytm przechodzi (jako pierwszy z czterech) do fazy trzeciej. Odbijanie podobnie zwiększa liczbę punktów w okolicy optimum, jednak nie tak szybko. Algorytm bez ograniczeń (niebieski) zachowuje się tak, jak zwykle – jak w pierwszym badanym przypadku, będącym punktem odniesienia.

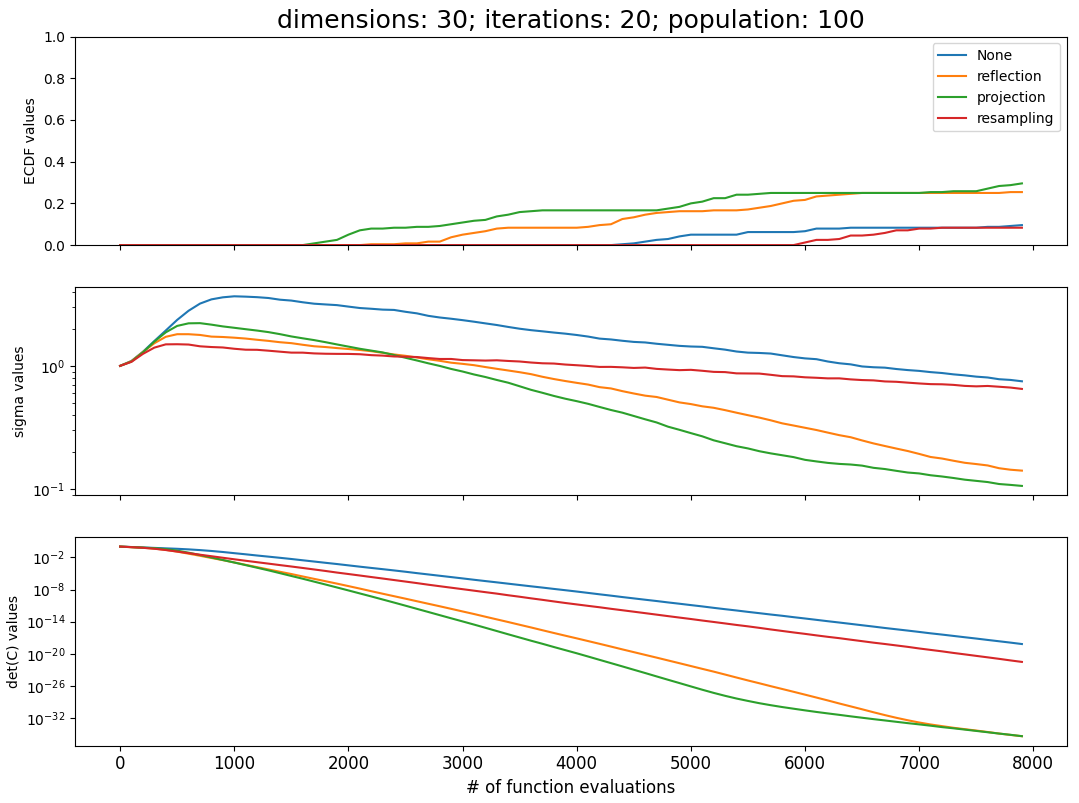
Zbadane zostało jeszcze kilka innych przypadków. Pierwszym z nich jest sytuacja, gdy dolne granice każdego kolejnego wymiaru przesuniemy bliżej i bliżej optimum. Górne granice nadal mają wartość 100. Dolne przyjmują 10 kolejnych wartości: -10^(i/2 - 4) dla i =0,1,…,8,9.



Różnice między metodami są widoczne i podobne do tych otrzymanych wcześniej. Rzutowanie i odbijanie dają gorsze wyniki niż wcześniej, być może dlatego, że przestrzeń przestała być symetryczna w każdym wymiarze i algorytm ma trudność z dopasowaniem się do konturu.

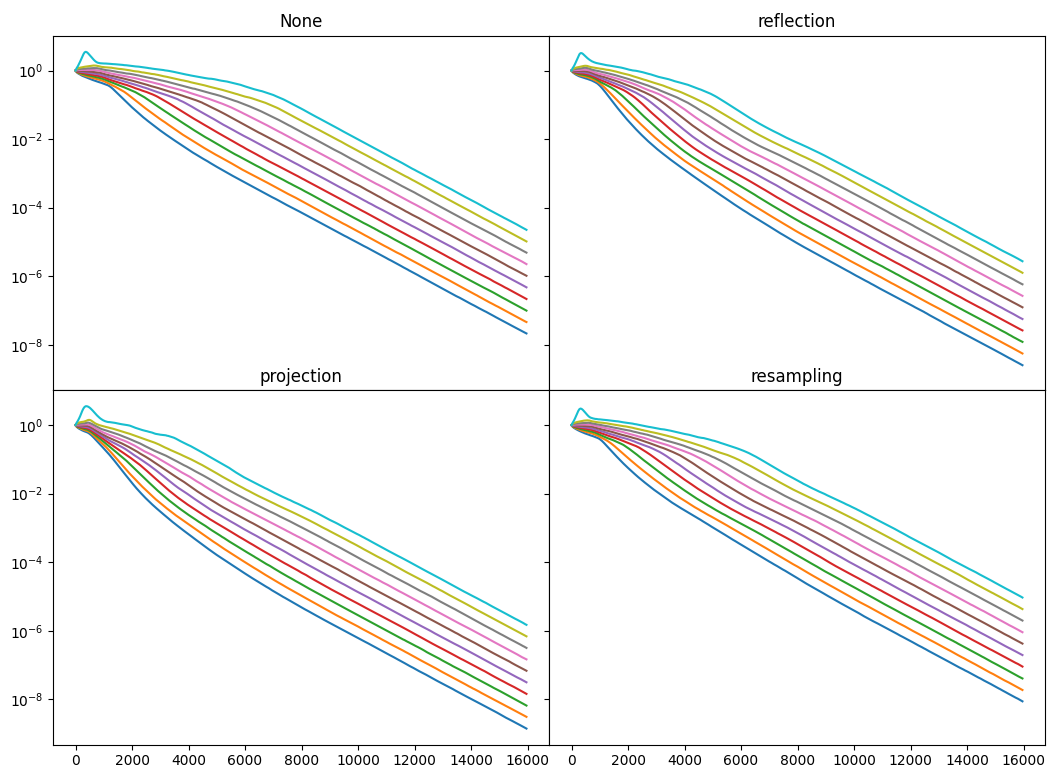
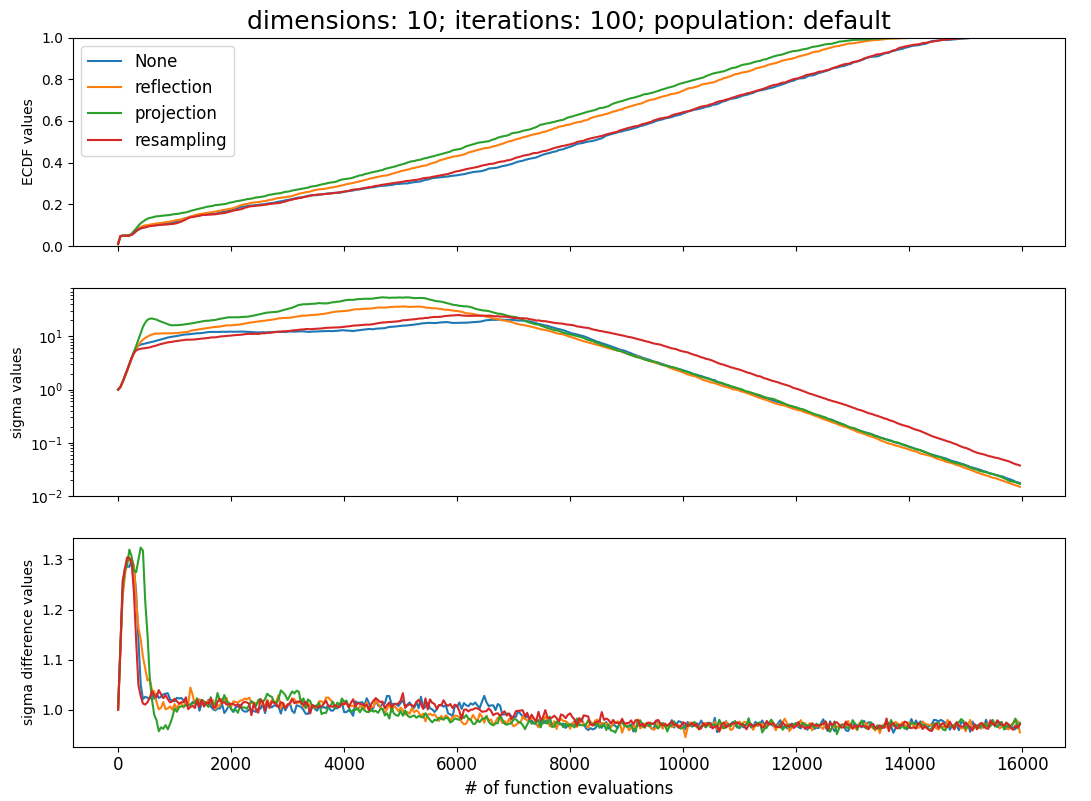
Inną wartą zbadania zmienną jest liczba wymiarów. Granice kostki z powrotem zostały ustawione na wartości (-0.1, 100).



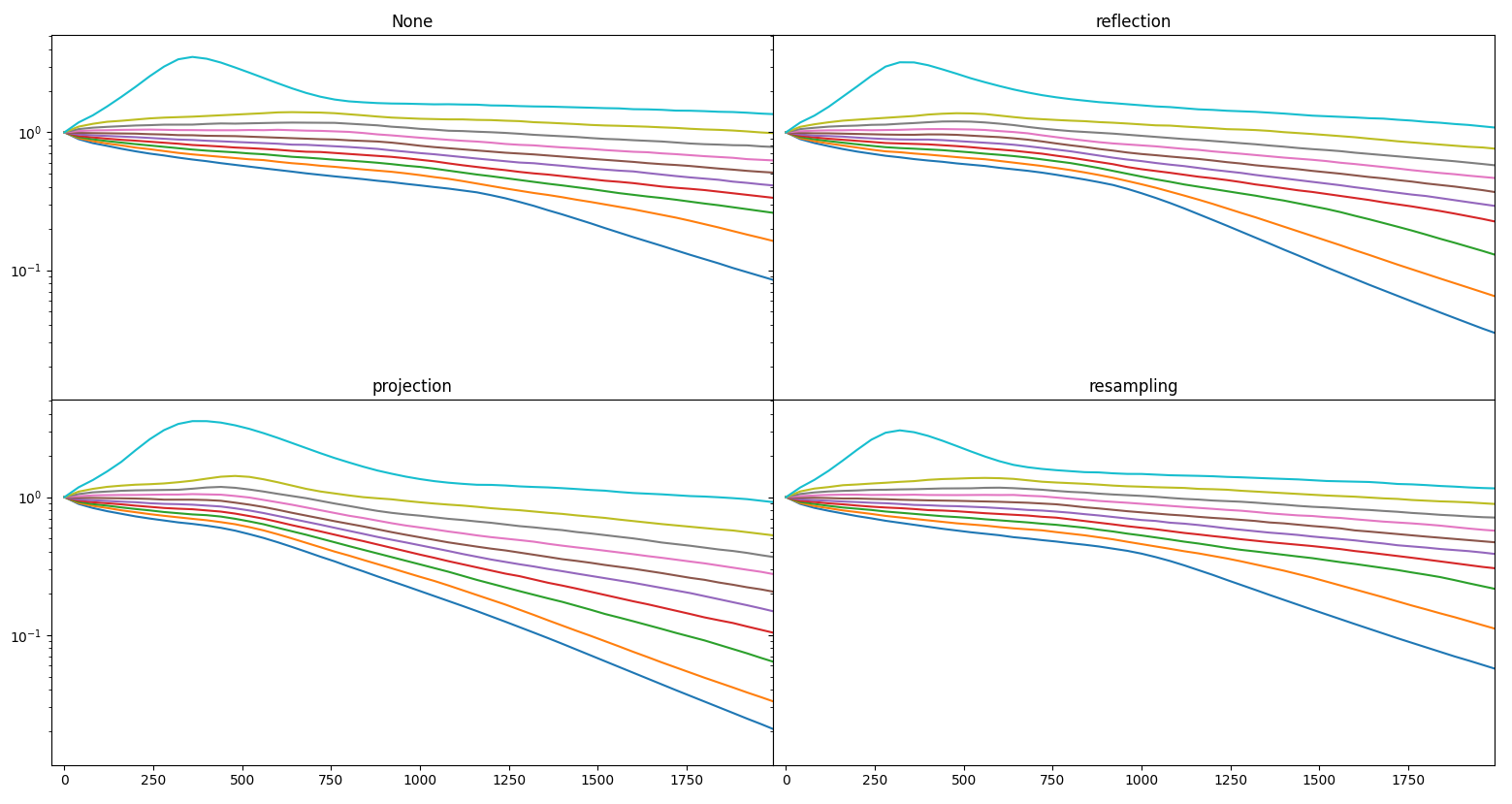
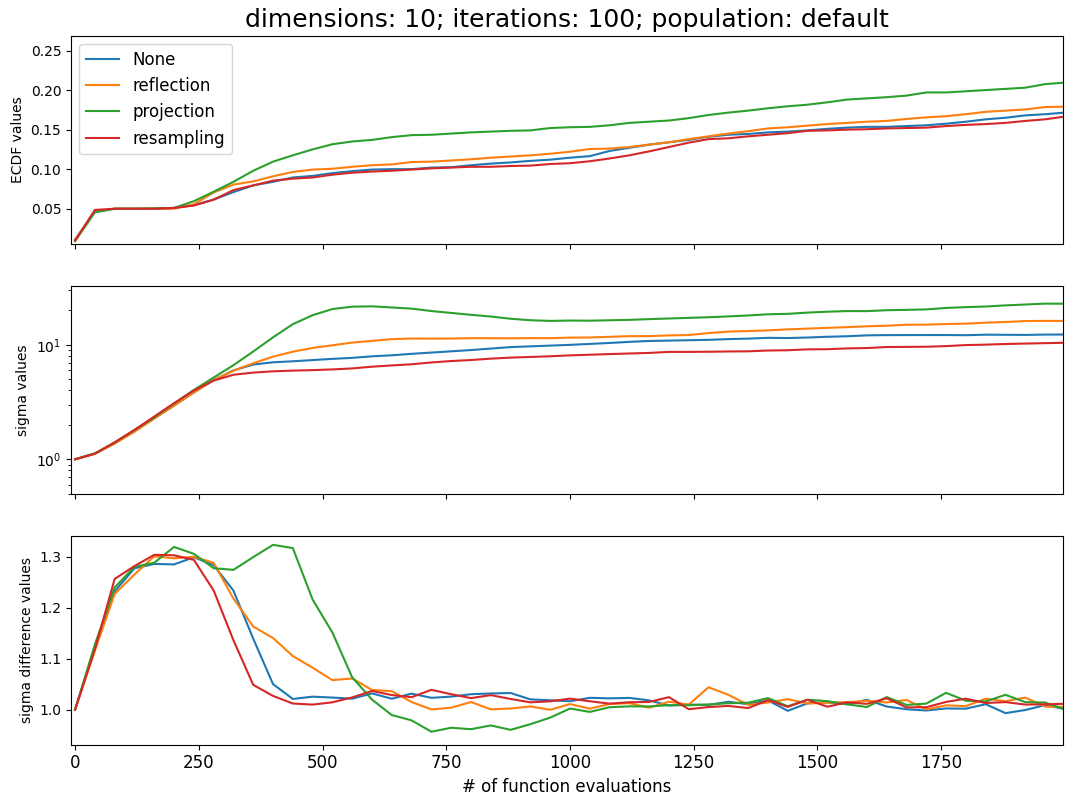


* Postępy 25.10.2021:

Badana funkcja celu jest nadal funkcją kwadratową, ale odpowiednio przeskalowaną w każdym wymiarze. Dokładny wzór podany jest na końcu dokumentu. Domyślna wartość wielkości populacji zostaje zmieniona na 4\*liczba wymiarów, czyli dla dziesięciu wymiarów 40. Początkowe położenie populacji oraz początkowa wartość sigma według zaleceń autora powinny być wybierane ręcznie, w zależności od funkcji celu. Początkowe położenie zostaje "odsunięte" do punktu [30,30,…30,30], a początkowa wartość sigma nadal równa jest 1. Z wykresu zostaje usunięty niepotrzebny fragment ilustrujący zmiany wyznacznika macierzy kowariancji. Zamiast niego powstaną wykresy opisujące zmiany wartości własnych tej macierzy. Oprócz wykresu opisującego zmiany sigma, powstanie wykres opisujący wartość ilorazu wartości sigma pomiędzy pokoleniami algorytmu.



Ten sam wykres, przybliżony:



* Postępy 28.10.2021:

Badam i funkcję kwadratową, i funkcję eliptyczną (przeskalowaną kwadratową). Dodam wykres opisujący wskaźnik uwarunkowania macierzy kowariancji oraz wykres opisujący wartość funkcji celu punktu będącego środkiem populacji punktów po selekcji.

* Hiperparametry algorytmu i szczegóły dotyczące wykresów:

Wartość początkowa macierzy kowariancji to macierz jednostkowa.

Wartość początkowa sigma to 1.

Mu jest równe lambda/2, czyli wynikiem selekcji jest lepsza połowa populacji. Wybrane punkty brane są pod uwagę z wagą 1/mu, a pozostałe z wagą 0.

Liczba wymiarów w większości testów jest równa 10. Wartość ta jest podana na każdym wykresie.

Lambda w większości testów jest równa wartości domyślnej 4\*liczba wymiarów, czyli 40.

Algorytm zatrzymuje się po wykonaniu 200 iteracji. Jest to wystarczająca liczba, żeby algorytm przeszedł do fazy trzeciej i wpływ ograniczeń był pomijalny.

Progi wykorzystane do nakreślenia krzywych ECDF są równe: 1e2, 1e1, 1e0, 1e-1, … 1e-9, 1e-10.

Pozostałe hiperparametry algorytmu zostały ustawione zgodnie z poradnikiem autora algorytmu: <http://arxiv.org/pdf/1604.00772.pdf>

Testy zostały przeprowadzone z wykorzystaniem implementacji własnej w języku Python.

Jeden wykres przedstawia najczęściej uśrednione wartości z 20 różnych przebiegów algorytmu. Wartość ta jest podana na każdym wykresie.

Wartości na osi poziomej odpowiadają liczbie wykonanych obliczeń funkcji celu, co jest wprost proporcjonalne do liczebności populacji i liczby wykonanych iteracji.

Badane zostały cztery warianty algorytmu – bez ograniczeń (None), metoda rzutowania (projection), odbijania (reflection) oraz ponownego próbkowania (resampling). W przypadku, gdy resampling nie uda się 10 razy z rzędu, używane zostaje rzutowanie.

Badaną funkcją jest funkcja kwadratowa przeskalowana odpowiednio w każdym wymiarze. W pierwszym wymiarze jest to 1, w ostatnim 10^6. Iloraz skali między kolejnymi wyrazami jest stały. Przykładowo, dla siedmiu wymiarów, wzór funkcji jest postaci: f(x) = 1\*(x1^2) +10\*(x2^2) +100\*(x3^2) +1000\*(x4^2) +10000\*(x5^2) +100000\*(x6^2) +1000000\*(x7^2).