

Porównanie efektywności metod stochastycznego gradientu i metod globalnych w zadaniach optymalizacji i globalnej

Michał Brzeziński

28 stycznia 2026

Spis treści

Streszczenie	2
1 Wstęp	3
1.1 Cel i zakres pracy	3
2 Algorytmy	4
2.1 Gradientowe	4
2.1.1 AdaGrad	4
2.1.2 Adam	5
2.1.3 L-BFGS-B	5
2.2 Globalne	5
2.2.1 CMAES	5
2.2.2 NL-SHADE-RSP-MID	6
3 Sposoby analizy	7
3.1 Metody testowania	7
3.2 Benchmarki CEC	7
3.2.1 Opis problemu	7
3.2.2 Implementacja	7
3.3 Sieci neuronowe	7
3.3.1 Opis problemu	7
3.3.2 Implementacja	8
4 Wyniki eksperymentów	10
4.1 Wyniki dla benchmarków CEC	10
4.2 Wyniki dla sieci neuronowych	39
4.3 Porównanie wyników	39
5 Podsumowanie i wnioski	41
5.1 Wnioski	41
5.2 Osiągnięte cele	41
5.3 Kierunki dalszego rozwoju	42
6 Załączniki	43
Bibliografia	44

Streszczenie

Celem pracy jest porównanie algorytmów stochastycznych z algorytmami globalnymi, oraz określenie zakresu kompetencji każdego z nich za pomocą prostej sieci neuronowej, i benchmarku CEC2013

Rozdział 1

Wstęp

1.1 Cel i zakres pracy

Praca ma na celu porównanie algorytmów stochastycznych i globalnych. Określić mocne strony każdego algorytmu, oraz wskazanie kierunków na potencjalny rozwój

Rozdział 2

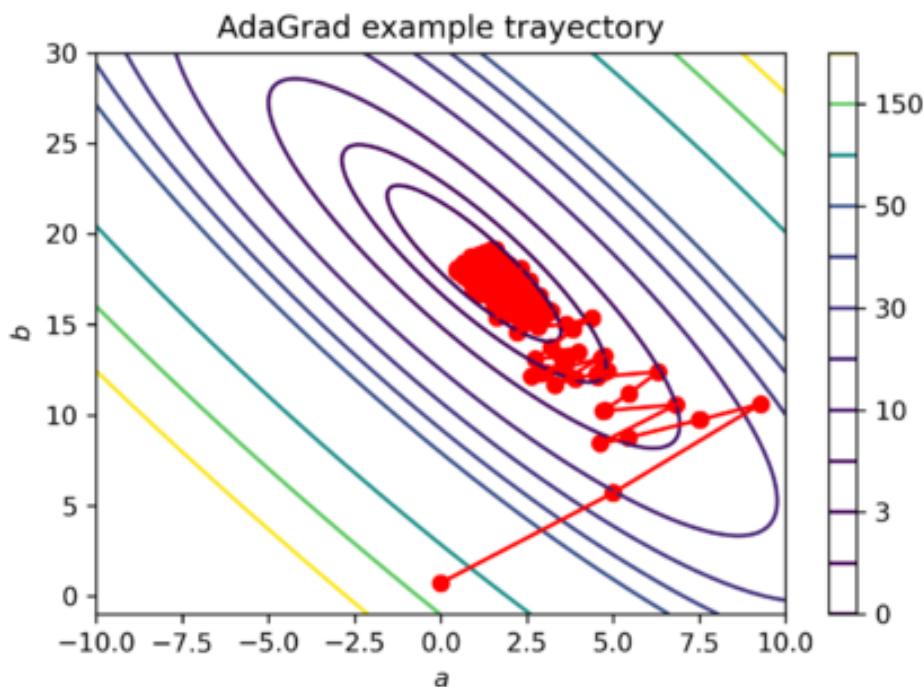
Algorytmy

2.1 Gradientowe

2.1.1 AdaGrad

Algorytm AdaGrad (Adaptive Gradient Algorithm) jest zmodyfikowanym algorytmem stochastycznym spadku gradientu, dla którego współczynnik uczenia jest ciągle aktualizowany na podstawie historii gradientów poprzednich kroków.

Dynamicznie zmieniający się współczynnik uczenia pozwala płynniej przechodzić z fazy eksploracji w fazę eksploatacji.



Dla $g_t = \nabla Q_i(w)$

$$G = \sum_{\tau=1}^t g_\tau g_\tau^T$$

$$w := w - \eta \operatorname{diag}(G)^{-1/2} \odot g_t$$

w - aktualny wektor pozycji

η - współczynnik uczenia

$Q_i(w)$ - funkcja celu w punkcie 'w' i dla zbioru danych treningowych 'i'

g_t - średnia gradientów funkcji celu w momencie 't' dla wszystkich próbek w batchu 'i'

G - historia złożona z sumy kwadratów wektorów poprzednich iteracji

2.1.2 Adam

Algorytm Adam (Adaptive Moment Estimation) wprowadza koncepcję adaptacji współczynnika uczenia za pomocą momentum oraz RMSProp do standardowego algorytmu stochastycznego spadku gradientu.

Adaptuje krok uczenia w każdym wymiarze oddzielnie, łącząc momentum i skalowanie drugiej chwili gradientu.

Dynamiczny współczynnik uczenia algorytmu łączy dwa elementy - momentum obliczane na podstawie średniej ważonej gradientu z poprzednich kroków, oraz średnia ważona kwadratów gradientów (druga chwila gradientu), której zadaniem jest optymalizowanie wartości wzmacniając małe gradienty i wygładzając za duże wartości.

Można porównywać to rozwiązanie do kontrolera PID w automatyce. Momentum określa kierunek, druga chwila gradientu stabilizuje ruch algorytmu

$$\begin{aligned} \text{dla :} \\ m_w^{(t)} &:= \beta_1 m_w^{(t-1)} + (1 - \beta_1) \nabla Q(w)^{(t)} \\ \hat{m}_w^{(t)} &= \frac{m_w^{(t)}}{1 - \beta_1^t} \\ v_w^{(t)} &:= \beta_2 v_w^{(t-1)} + (1 - \beta_2) (\nabla Q(w)^{(t)})^2 \\ \hat{v}_w^{(t)} &= \frac{v_w^{(t)}}{1 - \beta_2^t} \\ w^{(t)} &:= w^{(t-1)} - \eta \frac{\hat{m}_w^{(t)}}{\sqrt{\hat{v}_w^{(t)}} + \epsilon} \end{aligned}$$

Gdzie:

η - współczynnik uczenia

$Q(w)$ - funkcja celu dla wektora 'w'

ϵ - niewielka liczba zapobiegająca dzieleniu przez zero

$\hat{m}_w^{(t)}$ - momentum wektora 'w' podczas kroku 't'

β_1 - współczynnik zapomnienia momentum

$\hat{v}_w^{(t)}$ - druga chwila gradientu wektora 'w' podczas kroku 't'

β_2 - współczynnik zapomnienia drugiej chwili gradientu

2.1.3 L-BFGS-B

Algorytm BFGS wykorzystuje aproksymacje hesjanu funkcji, pozwalając na skalowanie kroku mając wgląd nie tylko w przebieg funkcji, ale też jej gradientu. Pozwala na dokładniejsze wybory kierunku i wielkości kroków.

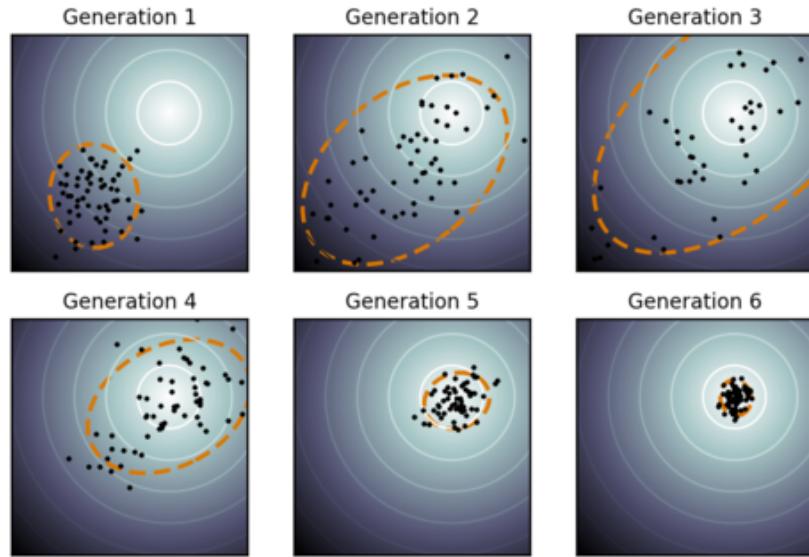
L-BFGS-B to rozszerzenie algorytmu BFGS o ograniczenia przestrzeni (Box constraints) oraz pamięci (Limited memory). Przestrzeń rozwiązań jest ograniczona, a hesjan jest aproksymowany na podstawie macierzy ustalonej liczby ostatnich kroków, pracuje na ograniczonej pamięci nie wykorzystując pełnej macierzy hesjanu.

2.2 Globalne

2.2.1 CMAES

Algorytm CMA-ES to algorytm ewolucyjny, który na podstawie populacji rodziców określa zakres przestrzeni na której generuje nowych potomków.

Tworzy populację kandydatów, ocenia ich jakość i na jej podstawie aktualizuje średnią i macierz kowariancji rozkładu wielowymiarowego, którą wykorzystuje do losowania nowej generacji



Rysunek 2.1: przykładowy przebieg algorytmu w CMAES

2.2.2 NL-SHADE-RSP-MID

NL-SHADE-RSP-MID to rozszerzenie algorytmu DE, adaptacyjnie ustawiające parametry krzyżowania i mutacji Cr i F, na podstawie populacji oraz archiwum rozwiązań.

NL-SHADE:

Dla każdego osobnika generowany jest wektor mutant na podstawie trzech losowo wybranych osobników z populacji i potencjalnie archiwum.

Do operatora różnicowego mutacji może zostać wykorzystany punkt wylosowany z archiwum, które jest złożone z rodziców, którzy zostali zastąpieni potomkami z lepszym wynikiem.

Indywidualnie wartości Cr i F są generowane wykorzystując losową wartość z pamięci Cr i F jako centra rozkładów.

Określamy dla każdego osobnika poprawę wartości funkcji celu względem poprzedniej generacji oraz wartości Cr i F użyte na nich. Na podstawie tych danych, średnią ważoną Lehmera obliczamy najlepszą preferowaną wartość Cr i F dla generacji. Wynik jest zapisywany na jednej z pozycji w pamięci Cr i F.

Pod koniec generacji porównywane są wartości funkcji celu osobników powstały z mutacji, które wykorzystały archiwum z tymi, które tego nie zrobiły i, na podstawie sukcesu potomków, obliczane jest prawdopodobieństwo użycia archiwum w mutacji dla następnej generacji.

Rozmiar populacji oraz archiwum jest zmieniany w zależności od liczby wywołań funkcji celu w stosunku do limitu.

RSP:

Jeśli punkt nie mieści się w podanej przestrzeni danych, jest generowany ponownie.

MID:

Populacja jest klastrowana względem pozycji w przestrzeni rozwiązań, porównywane są wartości funkcji celu żeby wyznaczyć najlepszy podział, oraz porównać go z pełnym zbiorem żeby określić czy populacja ma zgrupowanie kilku punktów. Wyznaczany jest centroid najlepszego zgrupowania. Jeśli centroid się nie zmienia, oznacza to stagnację i algorytm jest zatrzymywany

Rozdział 3

Sposoby analizy

3.1 Metody testowania

Do porównania wyników każdego eksperymentu wykorzystano funkcję ECDF (Empirical Cumulative Distribution Function) oraz wykres średnich wartości w czasie z odchyleniem standardowym, oba wykresy w skali logarytmicznej. ECDF przedstawia skumulowany ułamek progów błędu osiągniętych przez każdy algorytm w kolejnych punktach w czasie. Ponieważ wszystkie algorytmy są oceniane względem tych samych progów, metoda dobrze wizualizuje, jak każdy z algorytmów radził sobie w różnych etapach eksperymentu. Metoda ta uniemożliwia spadek osiągniętych progów, co znacząco poprawia czytelność wykresów

3.2 Benchmarki CEC

3.2.1 Opis problemu

Congress on Evolutionary Computation (CEC), czyli Kongres Optymalizacji Ewolucyjnej, to konferencja naukowa organizowana przez IEEE, której celem jest publikacja standardowych funkcji testowych do oceny algorytmów optymalizacji.

Benchmarki CEC to zestaw standardowych testów optymalizacyjnych opracowanych przez Congress on Evolutionary Computation (CEC). Ich celem jest obiektywne porównanie algorytmów optymalizacyjnych.

3.2.2 Implementacja

Do testów wykorzystano CEC2013, zbiór funkcji używanych na Kongresie Optymalizacji Ewolucyjnej w 2013 roku.

Jest to powszechnie stosowany punkt odniesienia do testowania algorytmów sztucznej inteligencji.

Zastosowano się do wytycznych ustalonych podczas konferencji. Każdy algorytm był uruchamiany 51 razy dla 10, 30 oraz 50 wymiarów, przy ograniczonej liczbie wywołań funkcji ewaluacyjnej.

3.3 Sieci neuronowe

3.3.1 Opis problemu

Benchmarki CEC2013 są przystosowane do pracy na relatywnie niewielkiej liczbie wymiarów.

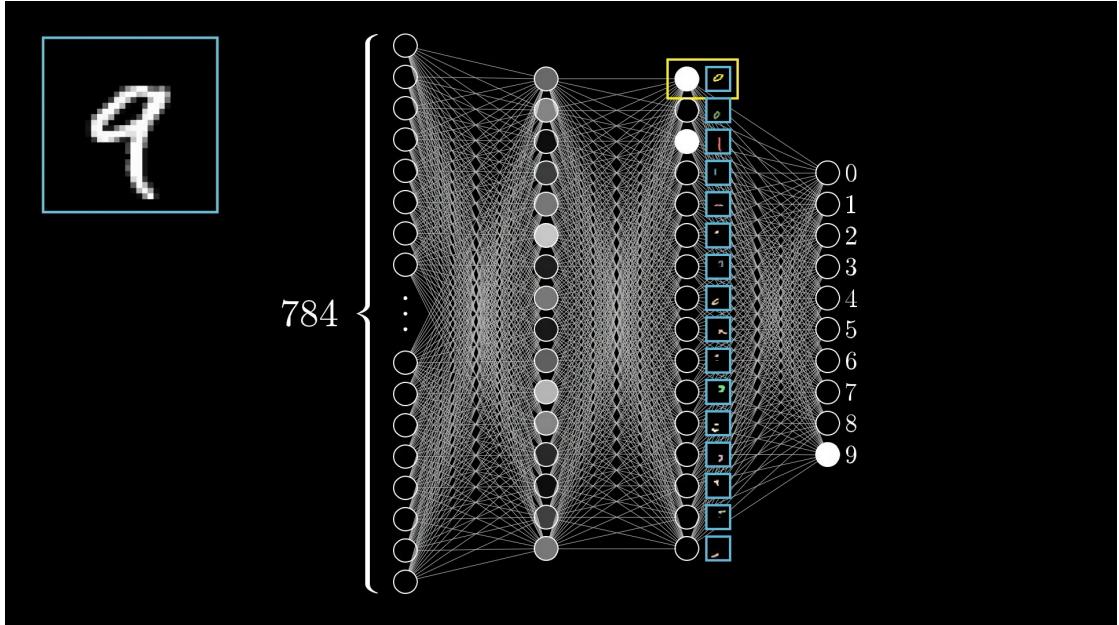
Nie pozwala to na określenie pełnego zakresu możliwych problemów optymalizacyjnych.

W sieciach neuronowych liczba wag liczona jest zwykle w skali tysięcy.

3.3.2 Implementacja

By porównać algorytmy stochastyczne i globalne zaimplementowana została prosta sieć neuronowa do klasyfikacji obrazów z bazy MNIST.

Dla obu rodzajów algorytmów struktura sieci składa się z warstwy wejściowej z 784 neuronami, warstwy losowego embeddingu o wymiarze 200, dwóch warstw ukrytych po 16 neuronów każda oraz warstwy wyjściowej z 10 neuronami. Wielkości warstw ukrytych zostały wybrane na podstawie najczęściej spotykanych wartości w literaturze.



Rysunek 3.1: model przykładowej sieci neuronowej

Wykorzystując lemat Johnsona-Lindenstraussa, jesteśmy w stanie oszacować jak redukcja wymiarów przestrzeni wpływa na zachowanie zależności pomiędzy osobnikami w tej przestrzeni. Dzięki temu możliwe jest osadzenie danych w przestrzeni o mniejszej liczbie wymiarów kontrolując zniekształcenia zależności między osobnikami.

dla danego $\epsilon \in (0, 1)$ i zbioru $X \subset R^{dim}$ o wielkości $|X| = n$ istnieje mapowanie $f : R^{dim} \rightarrow R^m$ gdzie nowy wymiar $m = O(\frac{\log n}{\epsilon^2})$ które osadza zbiór X do przestrzeni R^m ze zniekształceniem maksymalnym równym ϵ .

Warstwa embeddingu losowego Dla 784 wymiarów mapuje osobniki do przestrzeni 200-wymiarowej więc można się spodziewać zniekształceń w zależnościach rzędu 0.12

Jako reprezentant algorytmów stochastycznych został wykorzystany algorytm Adam, natomiast do algorytmów globalnych użyty został algorytm CMAES.

W przypadku rozwiązyania stochastycznego Zastosowano standardową metodę back-propagacji do uczenia i adaptacji wag i biasów. Optymalizowano funkcję kosztu.

Dla algorytmów globalnych problem ma wymiarowość równą sumie wszystkich trenowanych parametrów, czyli sumę wag i biasów warstw ukrytych i wyjścia. Łącznie 3658 wymiarów.

Optymalizowane było accuracy. Każdemu osobnikowi przydzielano ocenę na podstawie 20 losowo wybranych przykładów ze zbioru treningowego.

Jako wspólną jednostkę czasu przyjęto jedno przejście przez sieć neuronową. W przypadku globalnego algorytmu CMA-ES oznaczało to inkrementację licznika po każdej ocenie pojedynczego osobnika, natomiast w przypadku stochastycznego algorytmu licznik był zwiększany po każdej parze operacji przejścia w przód(forward pass), oraz backpropagacji.

Dla każdego algorytmu zapisywano najlepsze rozwiązania po przejściu przez daną część zasobów. Rozwiązania te były następnie oceniane wykorzystując zbiór testowy,

aby określić zmianę accuracy w czasie oraz ECDF

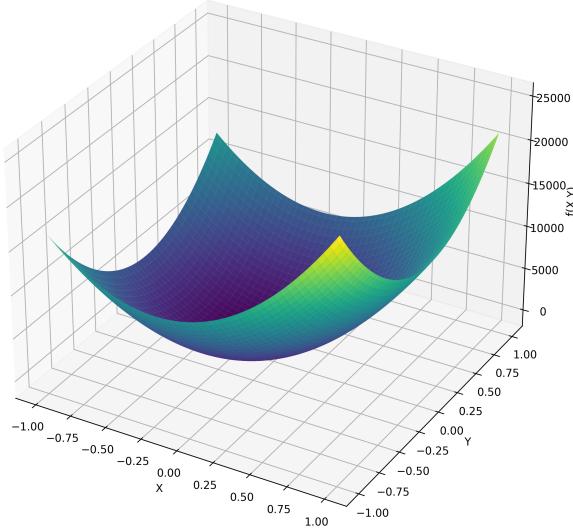
Rozdział 4

Wyniki eksperymentów

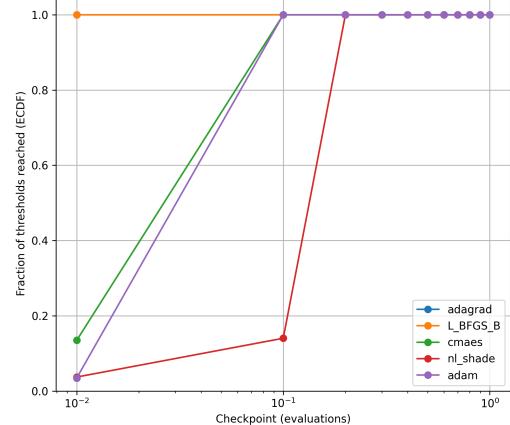
4.1 Wyniki dla benchmarków CEC

Sphere Function

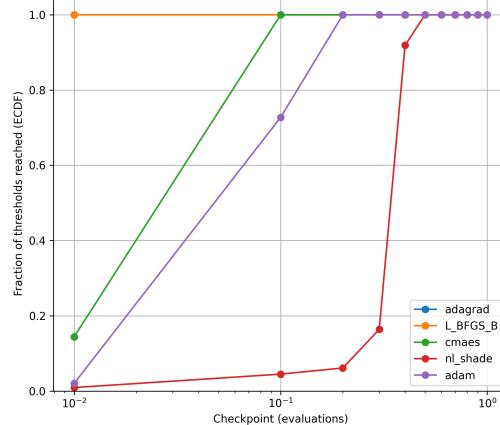
F12013



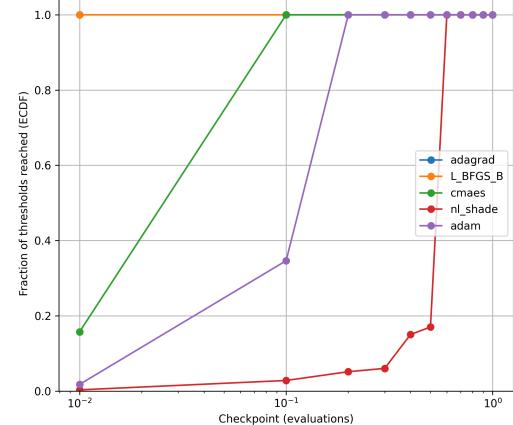
F12013, ECDF, dim=10



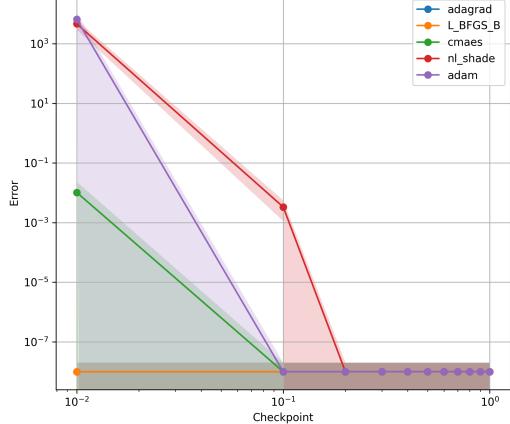
F12013, ECDF, dim=30



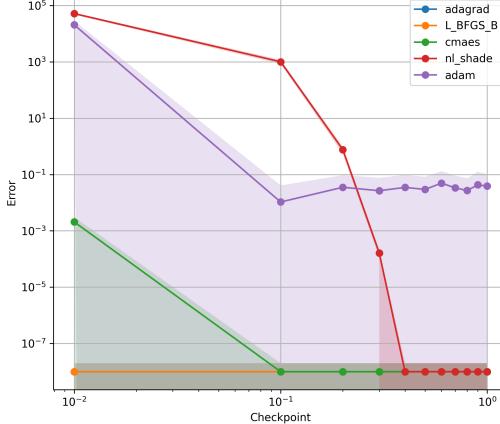
F12013, ECDF, dim=50



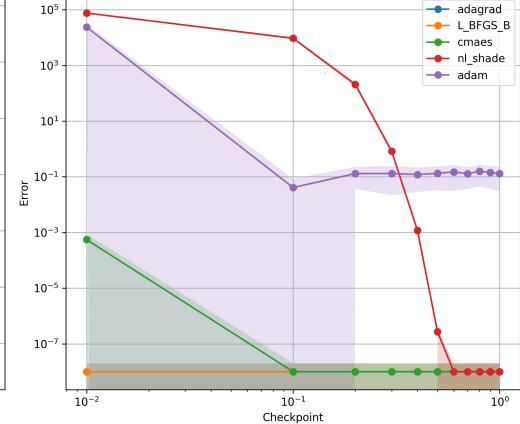
F12013, mean and std, dim=10



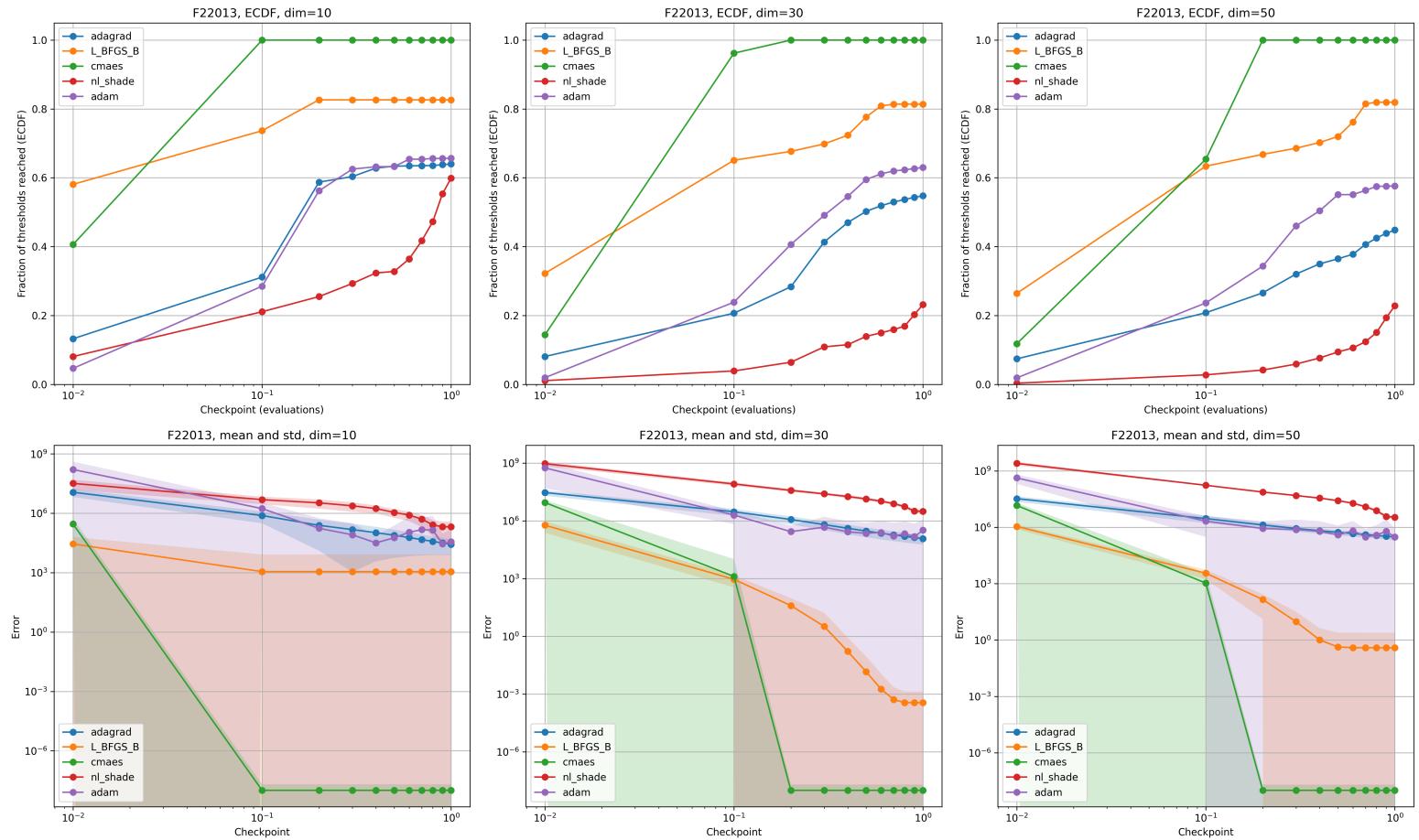
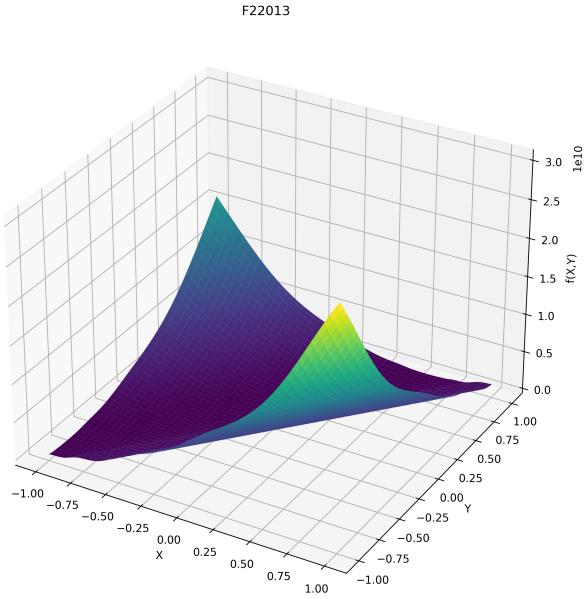
F12013, mean and std, dim=30



F12013, mean and std, dim=50

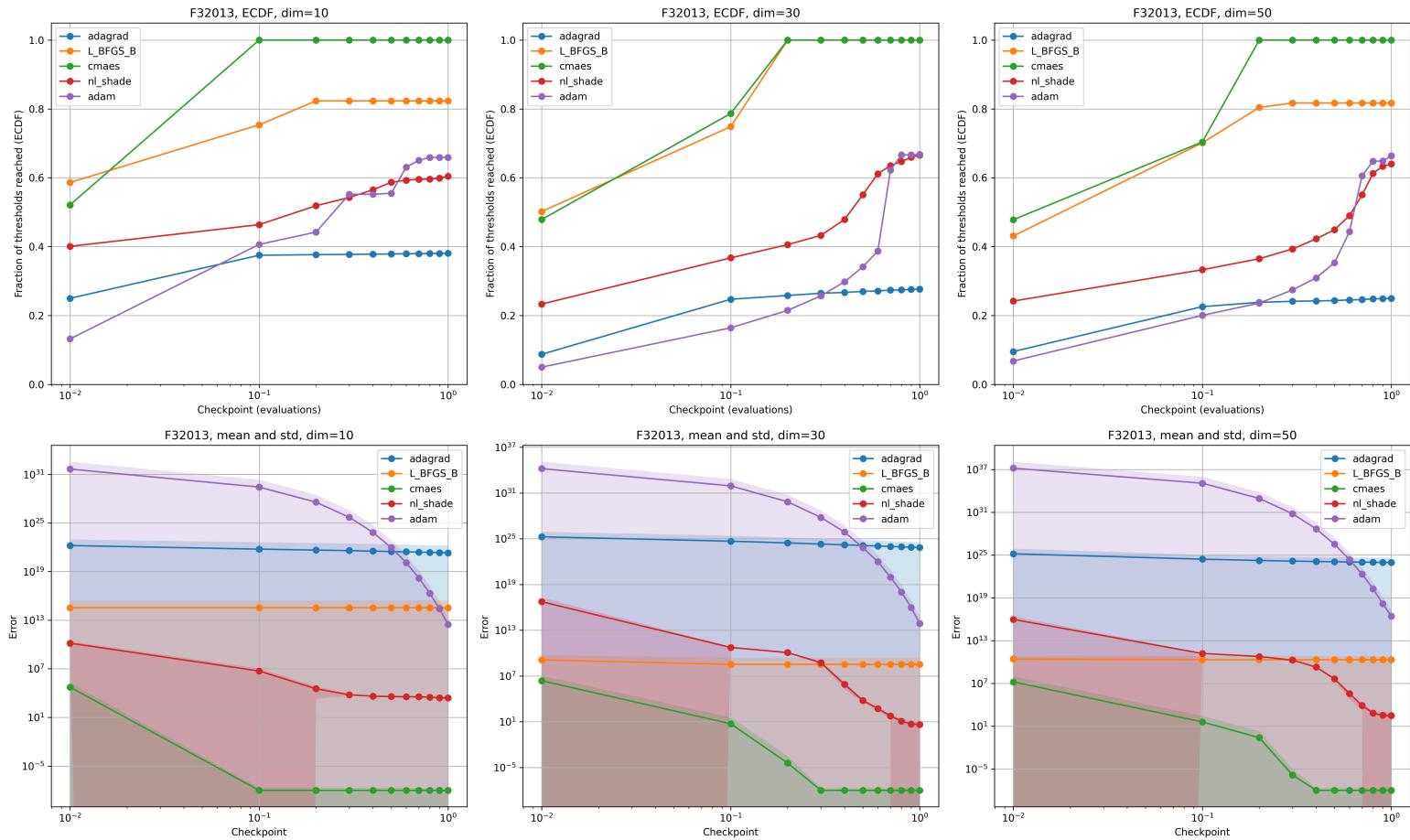
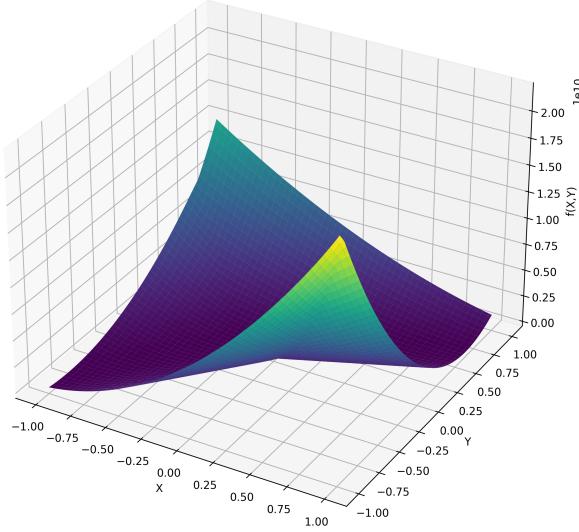


Rotated High Conditioned Elliptic Function



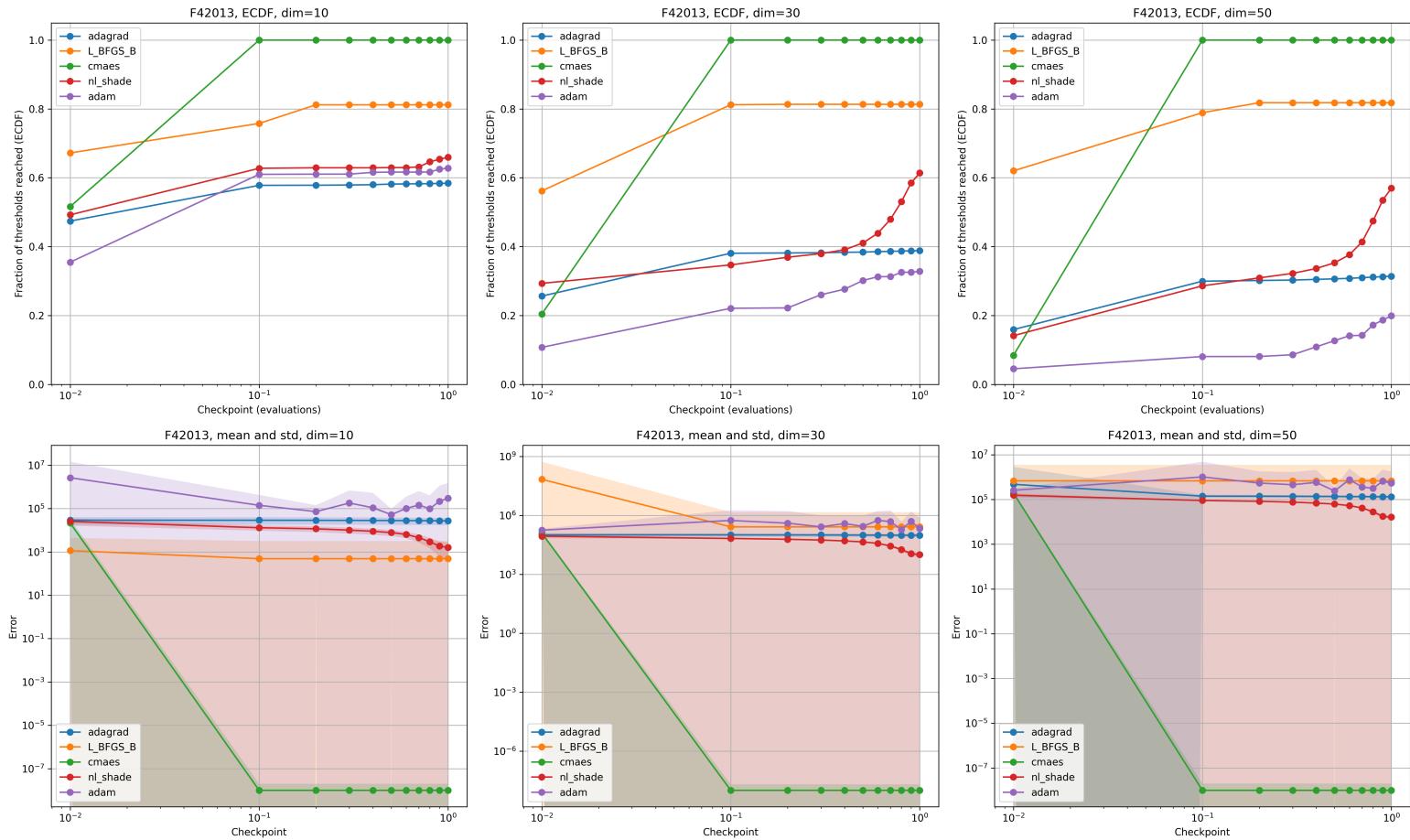
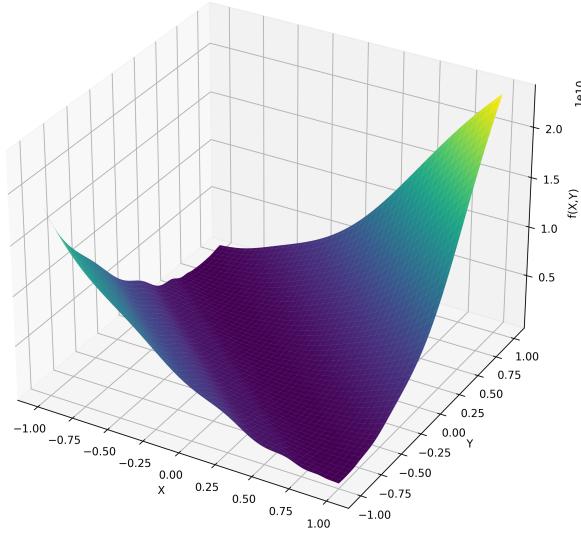
Rotated Bent Cigar Function

F32013



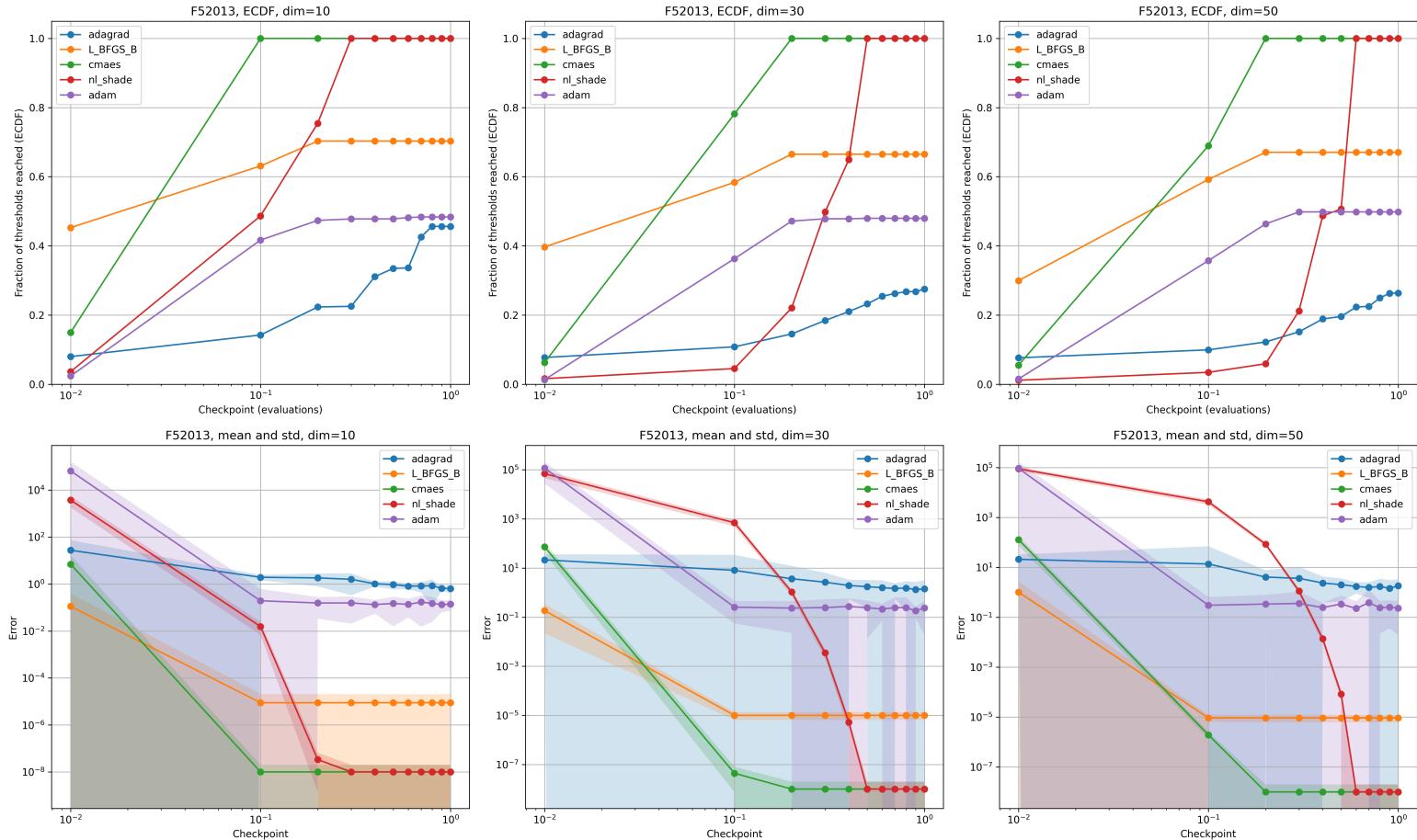
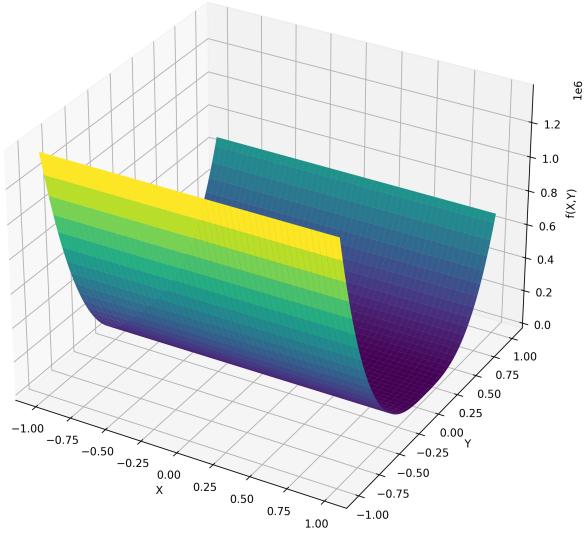
Rotated Discus Function

F42013



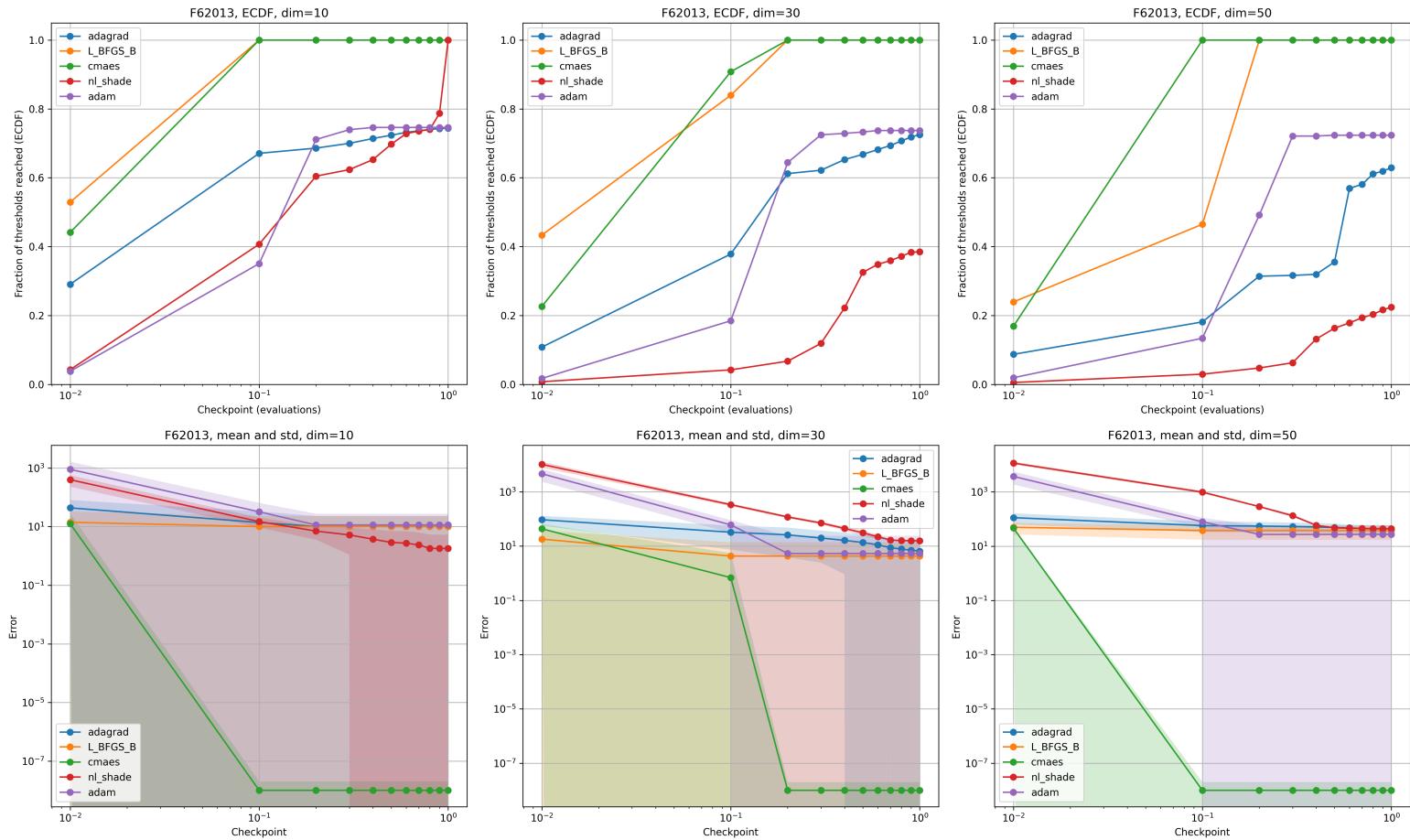
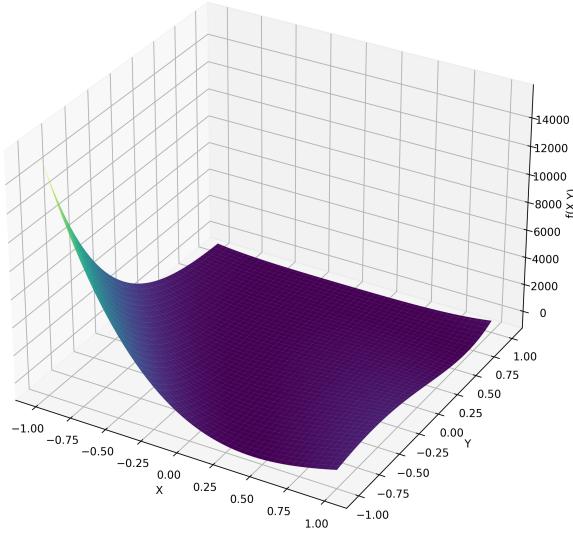
Different Powers Function

F52013



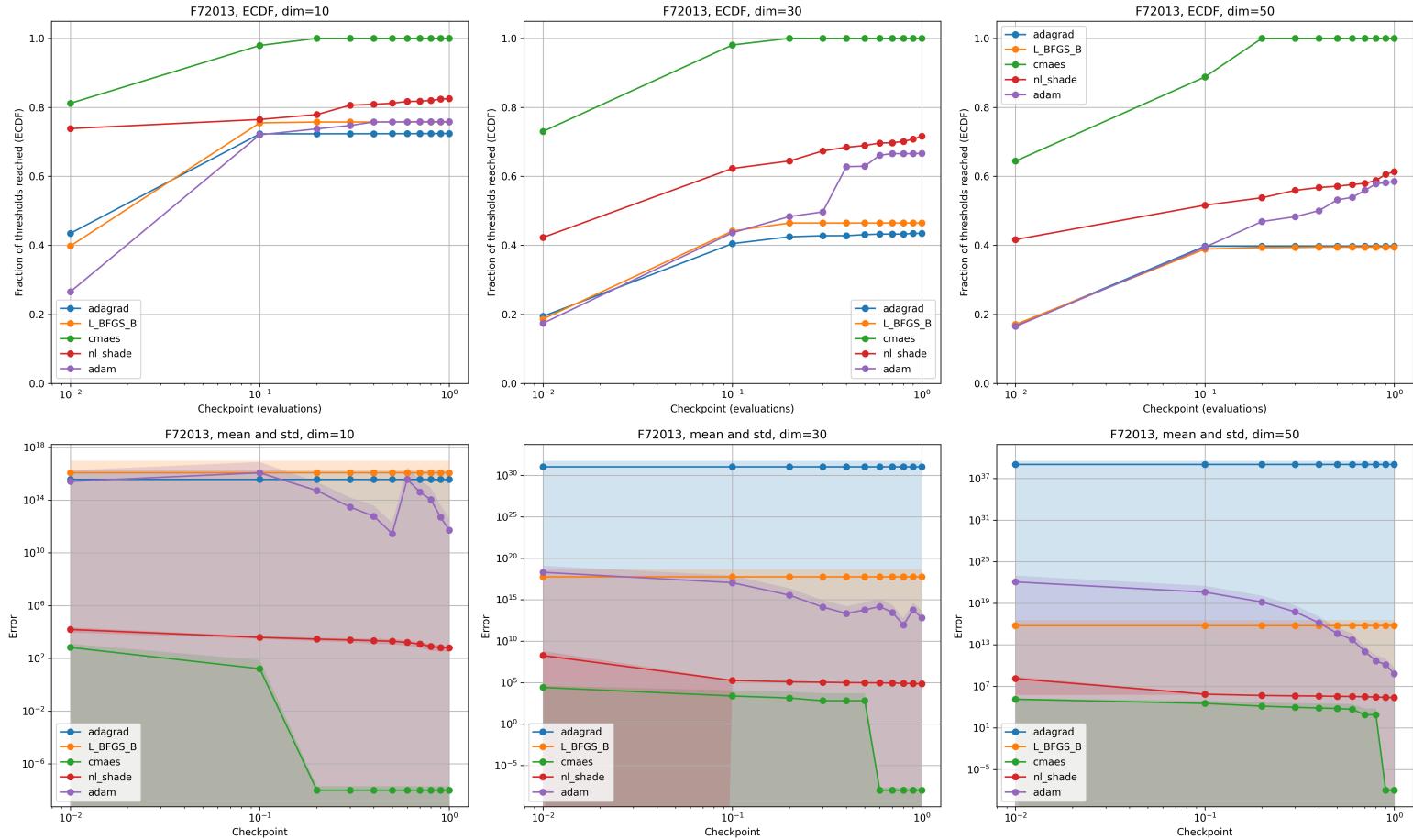
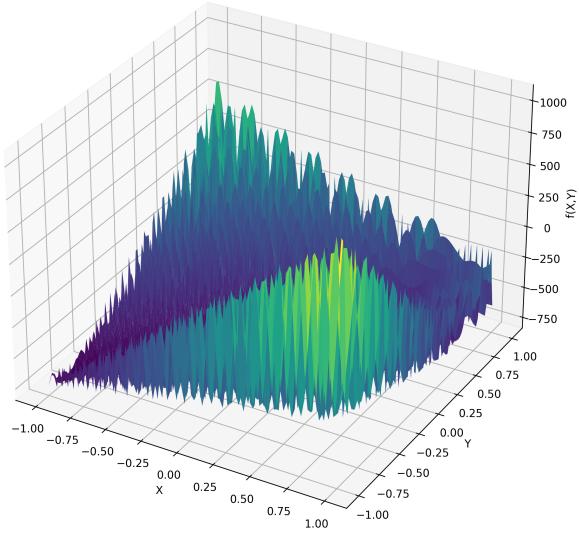
Rotated Rosenbrock's Function

F62013



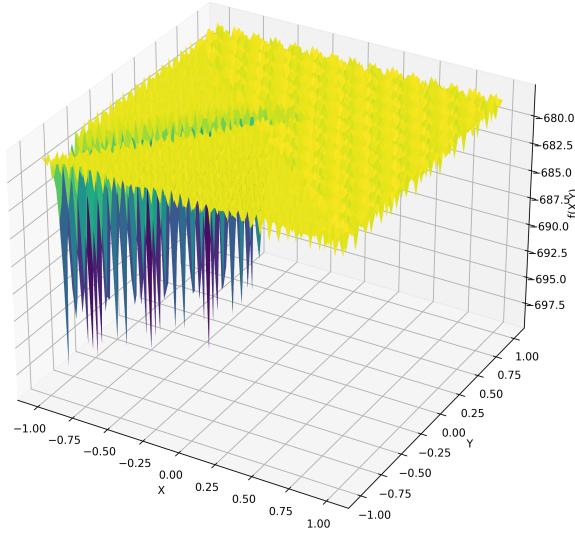
Rotated Schaffers F7 Function

F72013

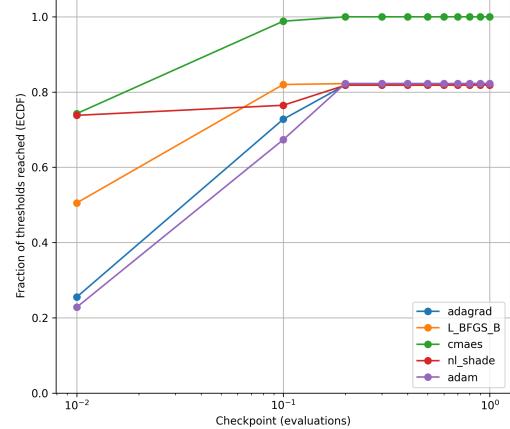


Rotated Ackley's Function

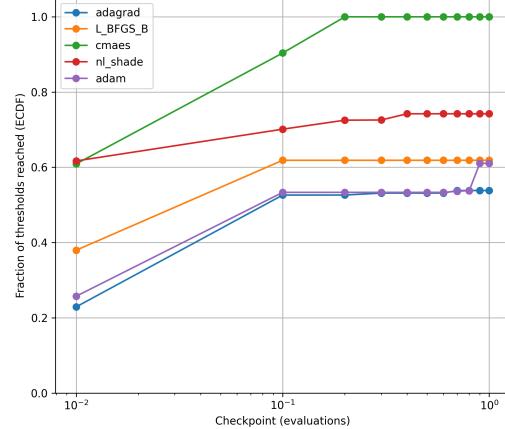
F82013



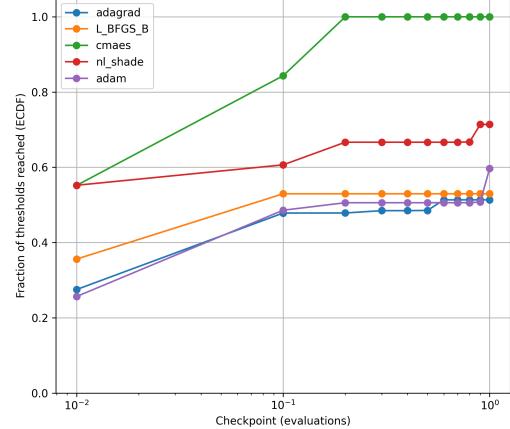
F82013, ECDF, dim=10



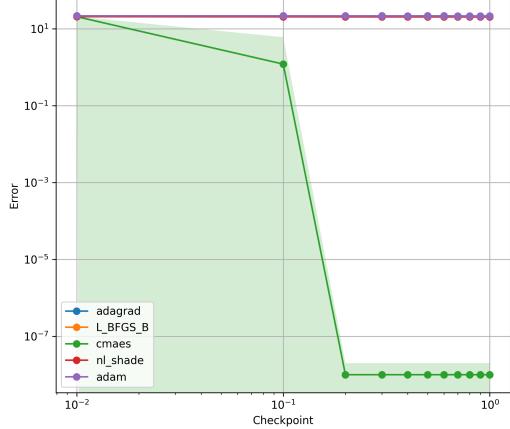
F82013, ECDF, dim=30



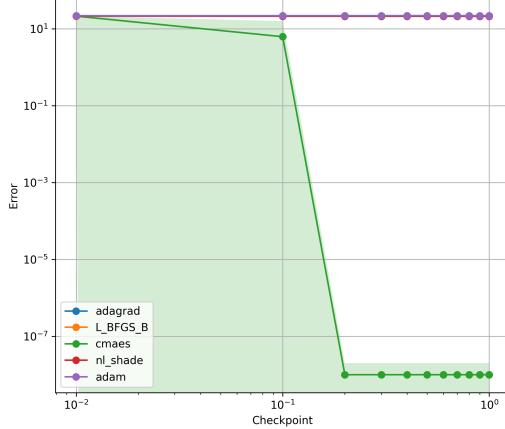
F82013, ECDF, dim=50



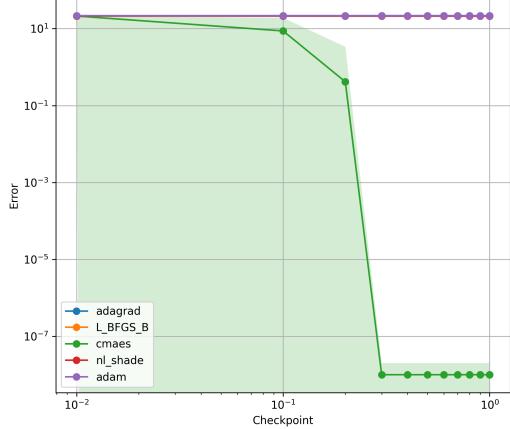
F82013, mean and std, dim=10



F82013, mean and std, dim=30

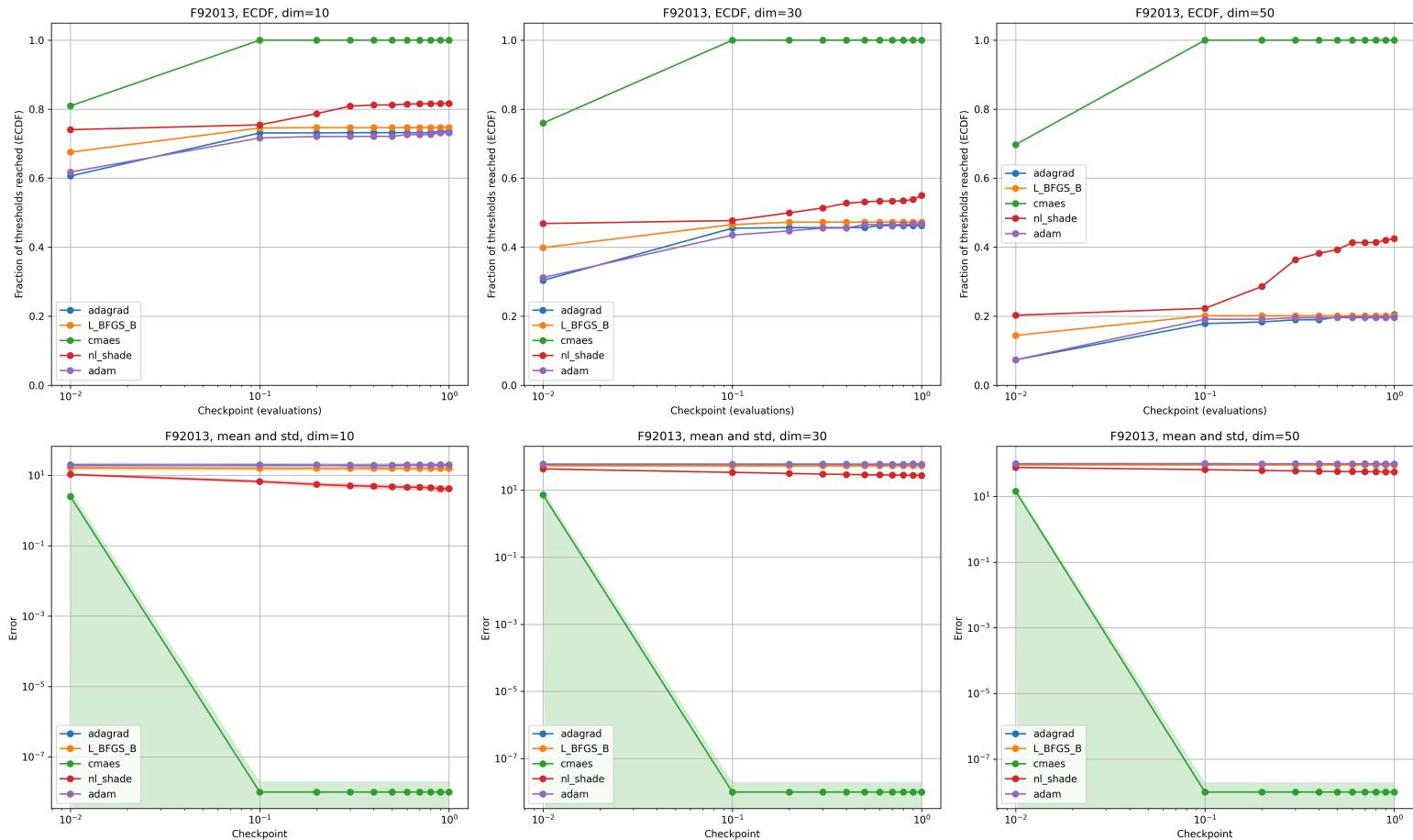
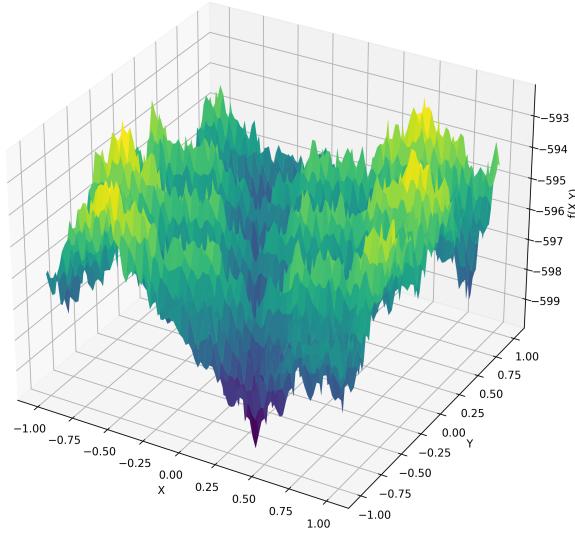


F82013, mean and std, dim=50



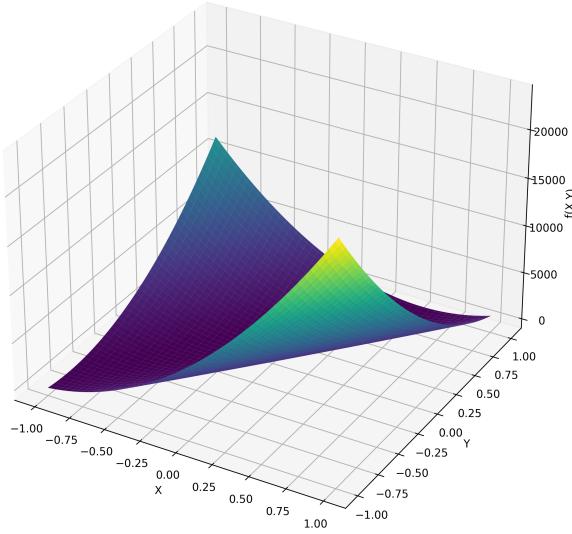
Rotated Weierstrass Function

F92013

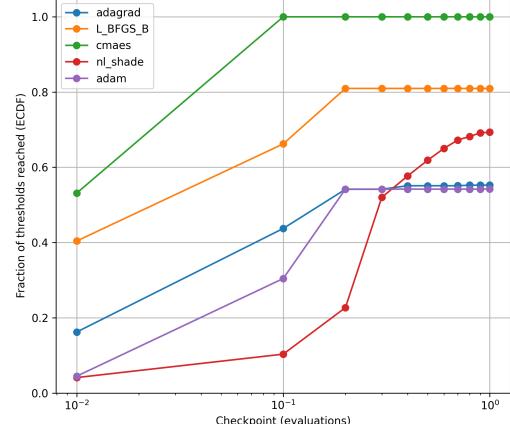


Rotated Griewank's Function

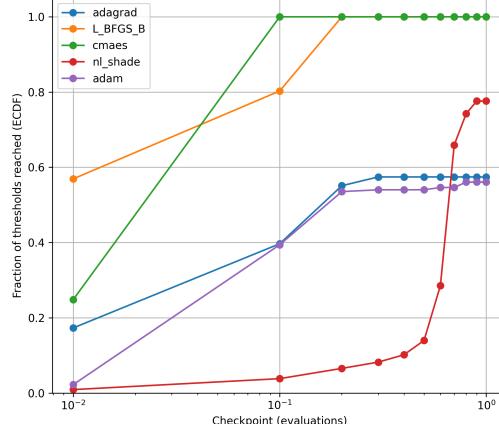
F102013



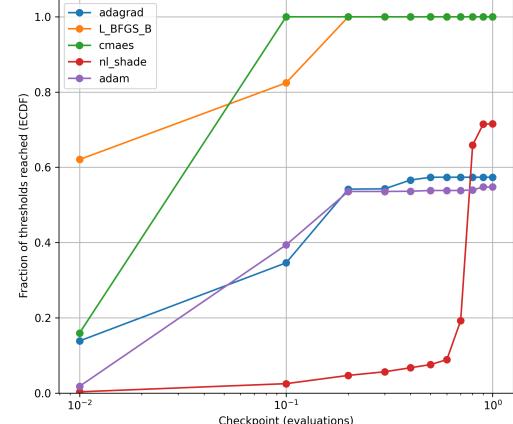
F102013, ECDF, dim=10



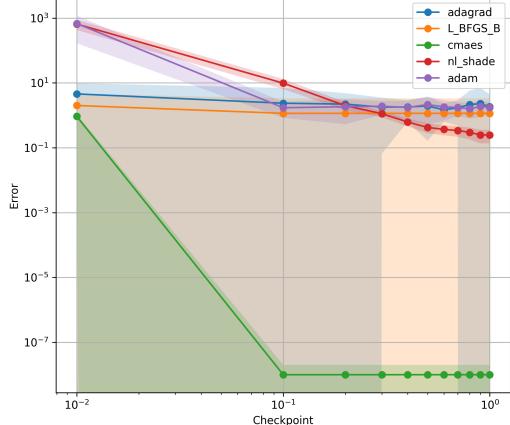
F102013, ECDF, dim=30



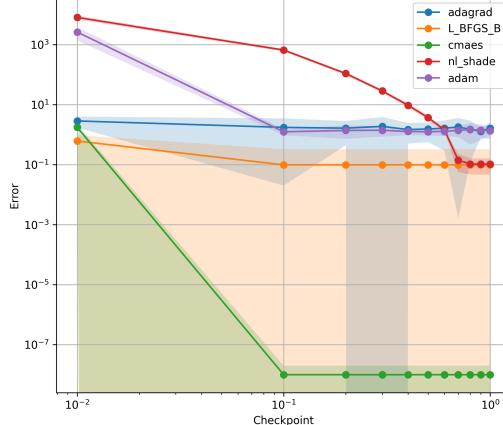
F102013, ECDF, dim=50



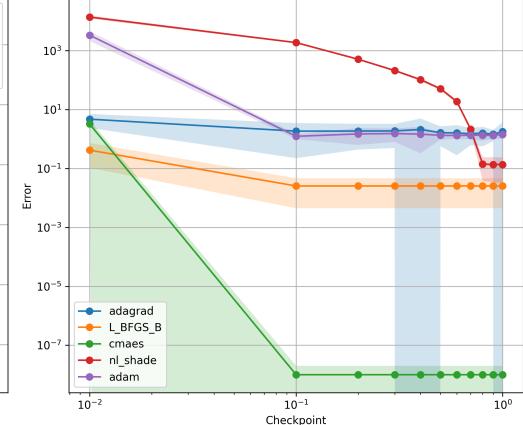
F102013, mean and std, dim=10



F102013, mean and std, dim=30

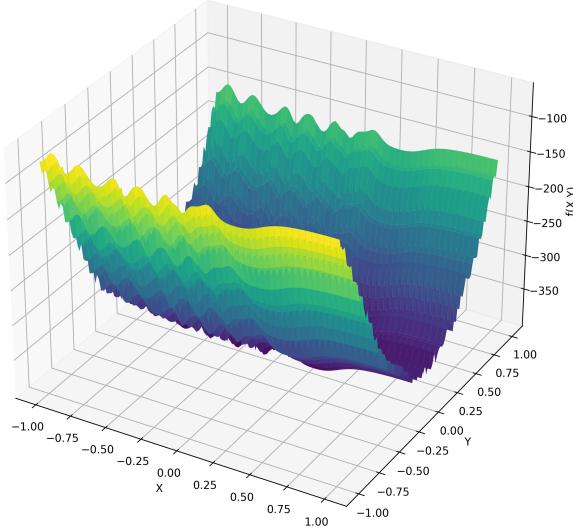


F102013, mean and std, dim=50

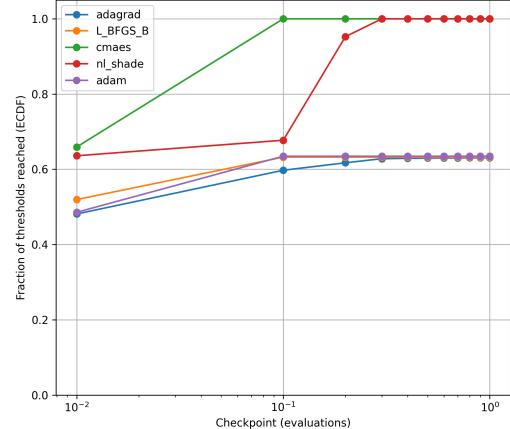


Rastrigin's Function

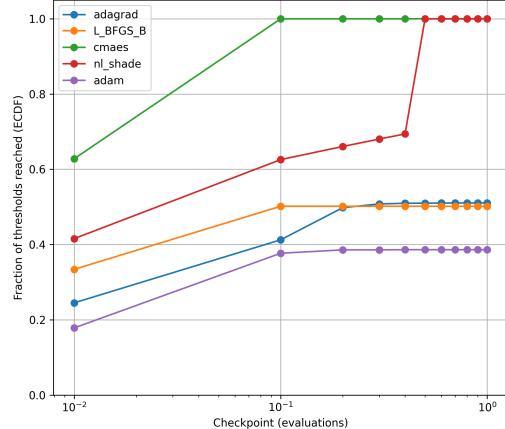
F112013



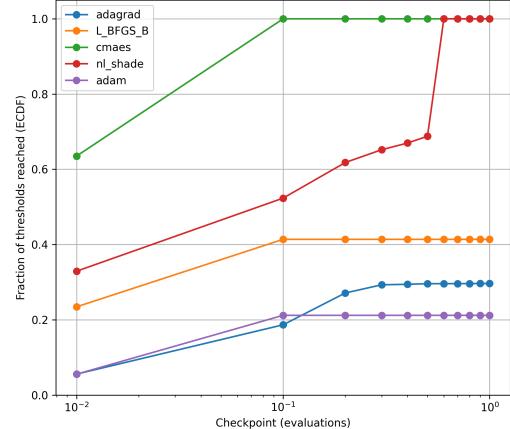
F112013, ECDF, dim=10



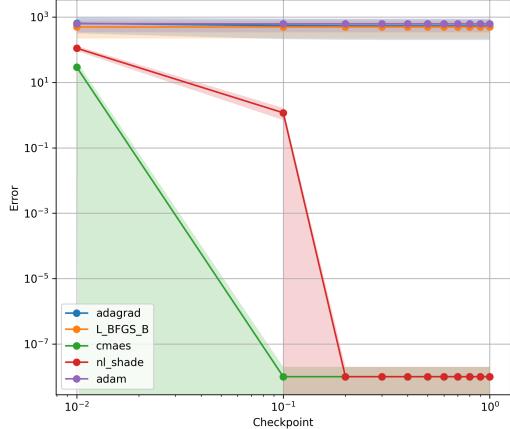
F112013, ECDF, dim=30



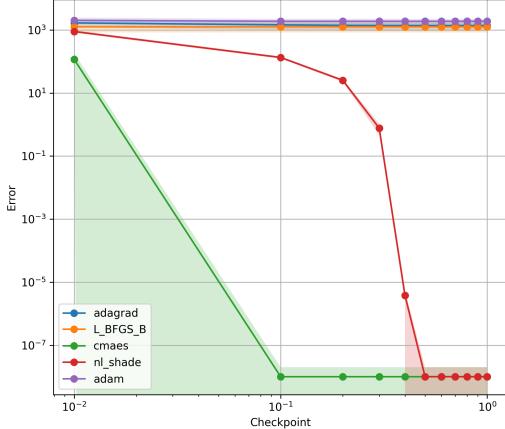
F112013, ECDF, dim=50



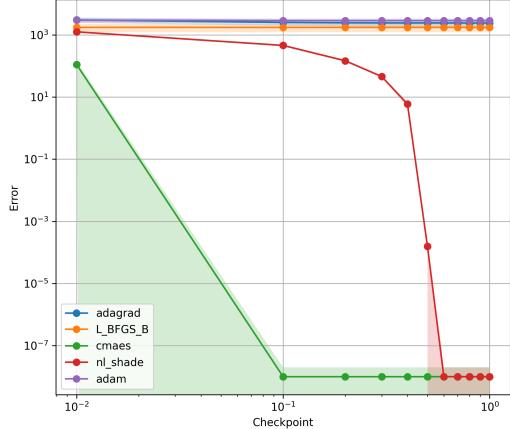
F112013, mean and std, dim=10



F112013, mean and std, dim=30

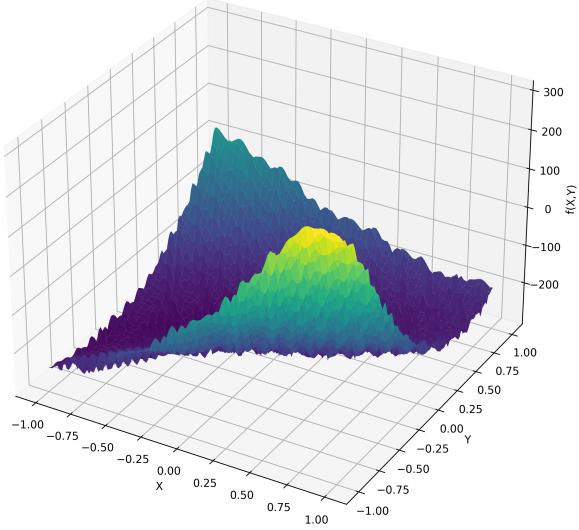


F112013, mean and std, dim=50

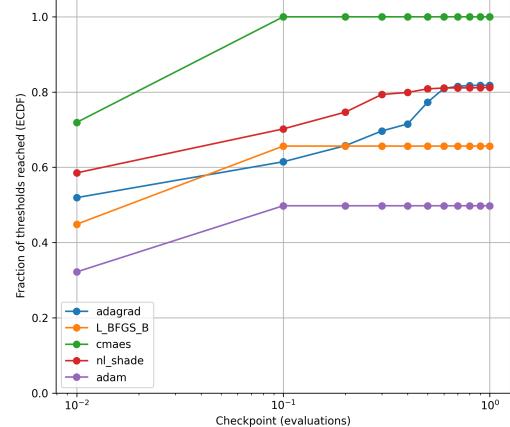


Rotated Rastrigin's Function

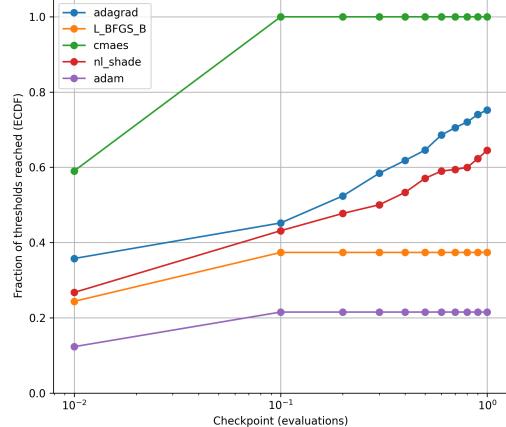
F122013



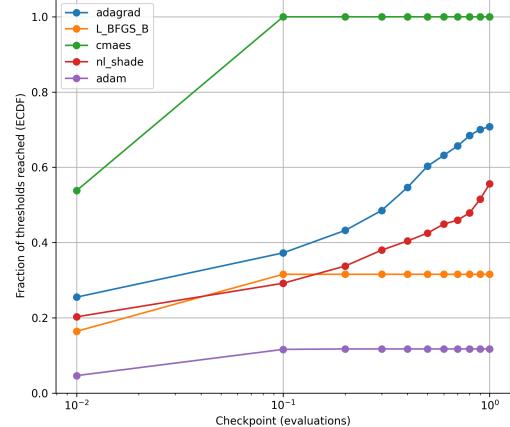
F122013, ECDF, dim=10



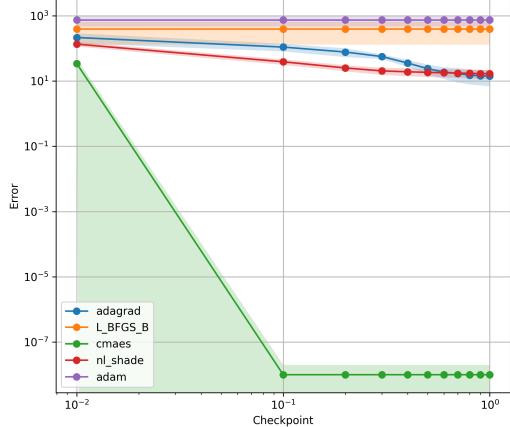
F122013, ECDF, dim=30



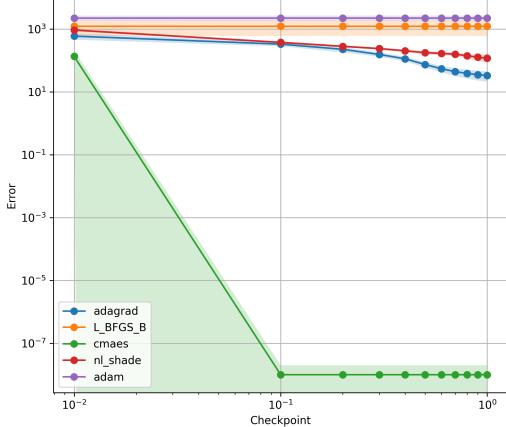
F122013, ECDF, dim=50



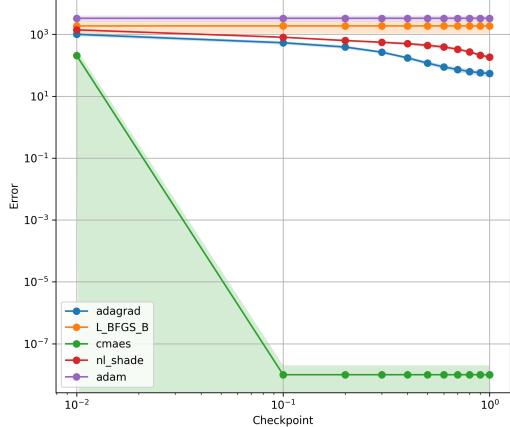
F122013, mean and std, dim=10



F122013, mean and std, dim=30

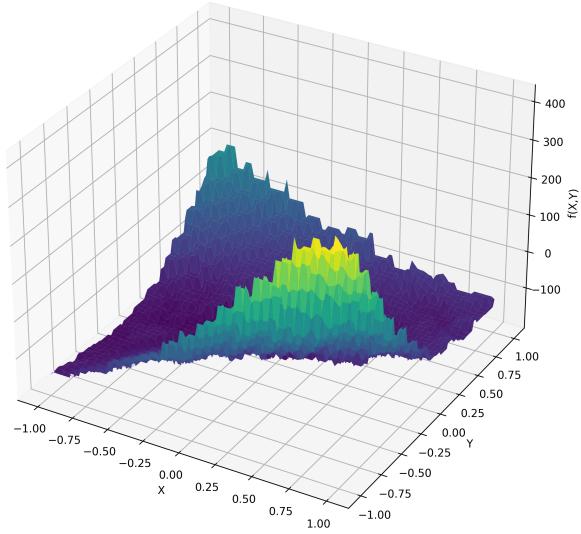


F122013, mean and std, dim=50

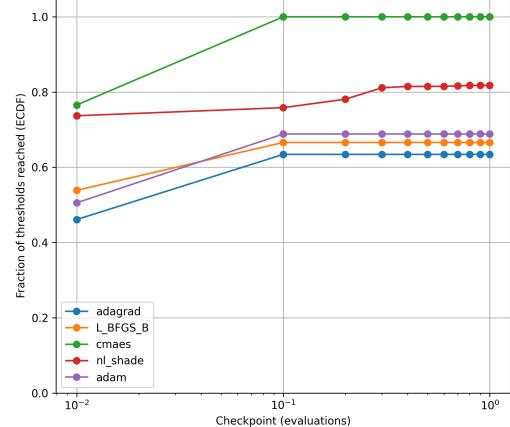


Non-Continuous Rotated Rastrigin's Function

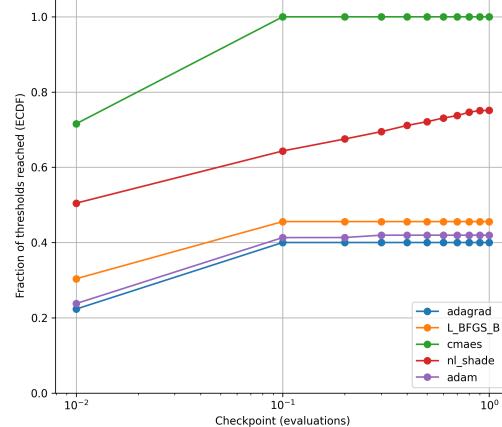
F132013



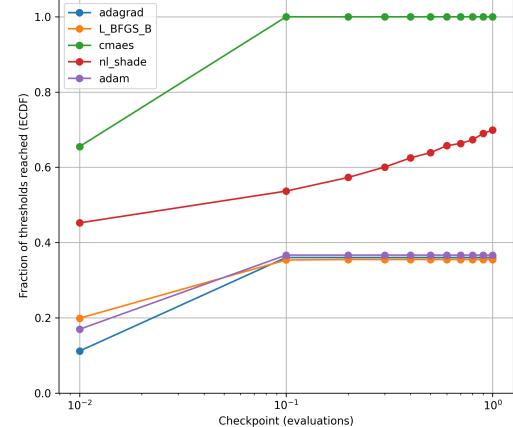
F132013, ECDF, dim=10



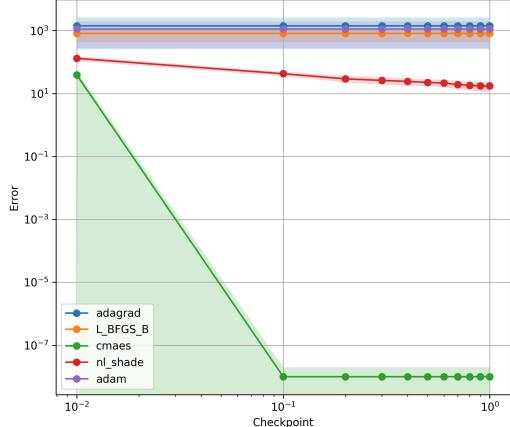
F132013, ECDF, dim=30



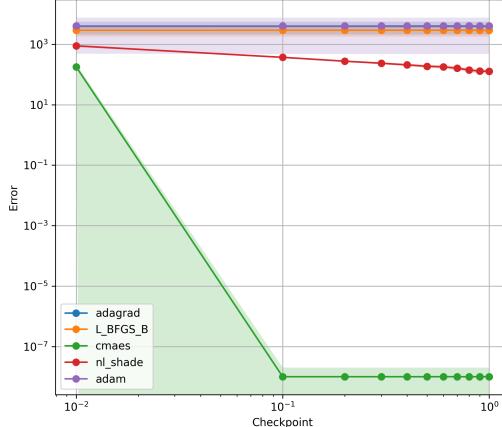
F132013, ECDF, dim=50



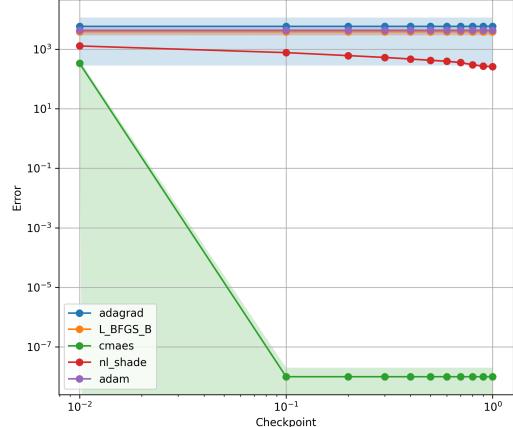
F132013, mean and std, dim=10



F132013, mean and std, dim=30

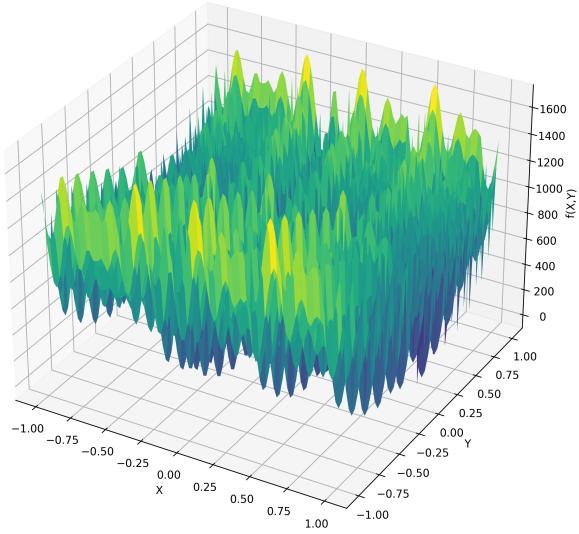


F132013, mean and std, dim=50

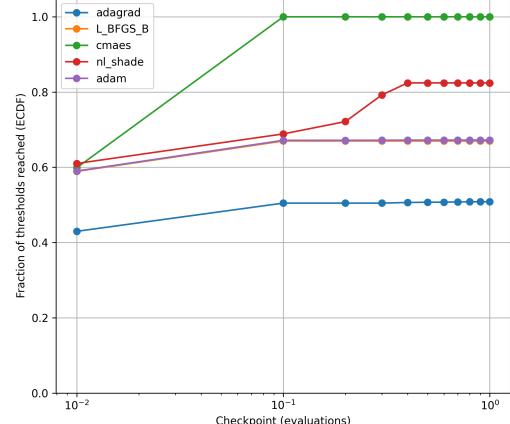


Schwefel's Function

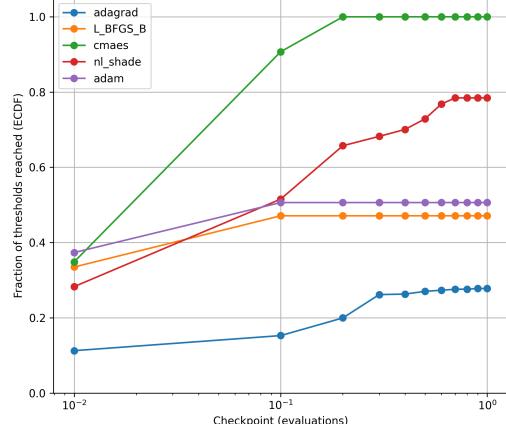
F142013



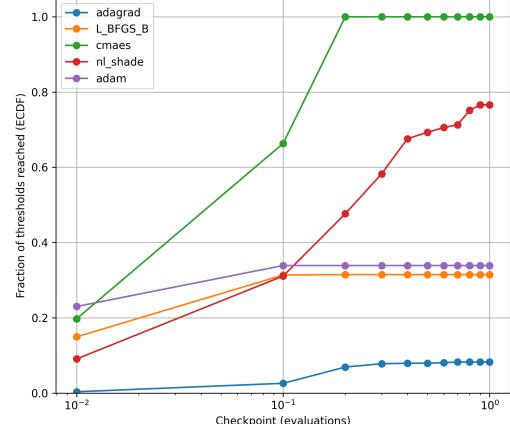
F142013, ECDF, dim=10



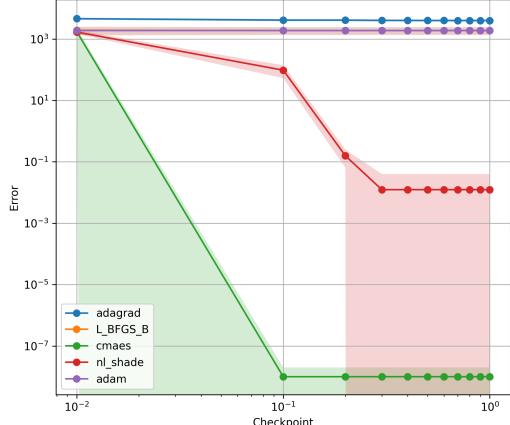
F142013, ECDF, dim=30



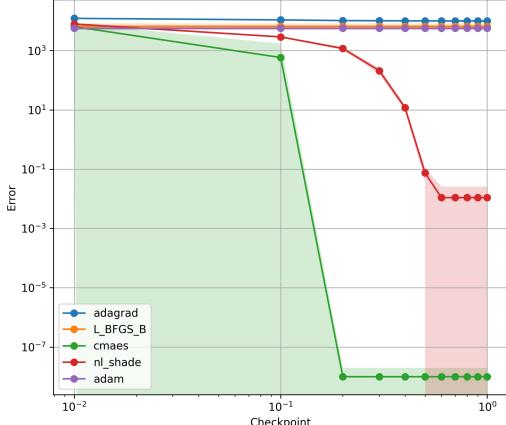
F142013, ECDF, dim=50



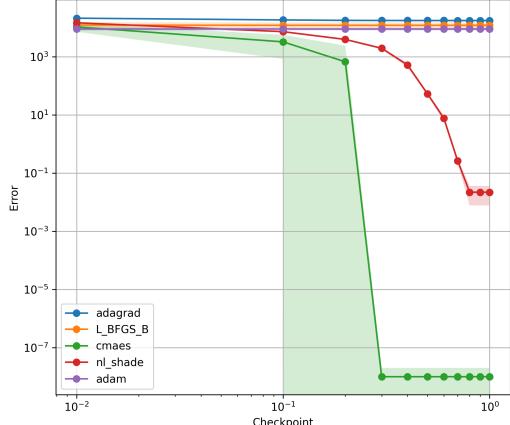
F142013, mean and std, dim=10



F142013, mean and std, dim=30

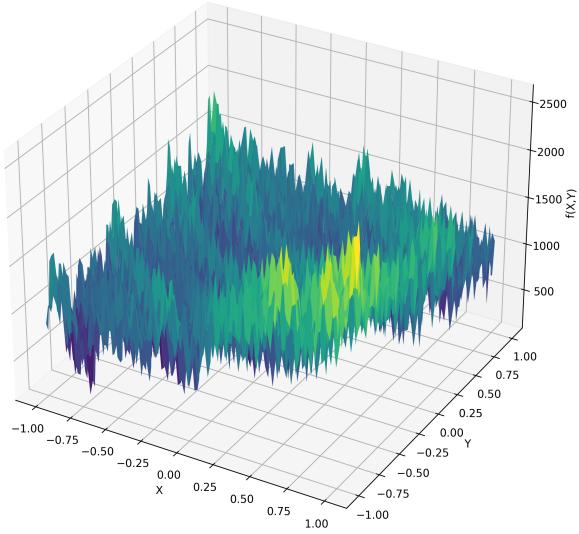


F142013, mean and std, dim=50

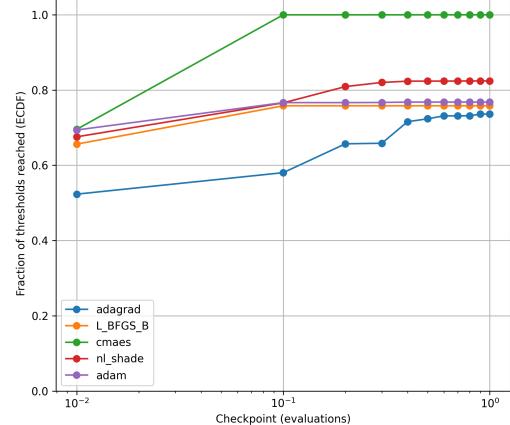


Rotated Schwefel's Function

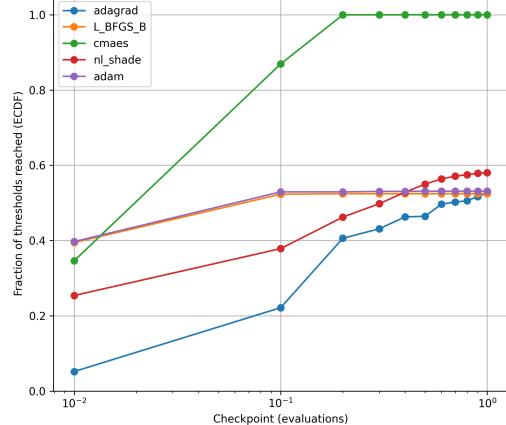
F152013



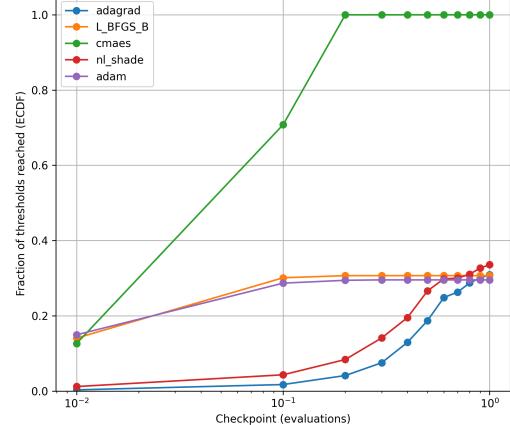
F152013, ECDF, dim=10



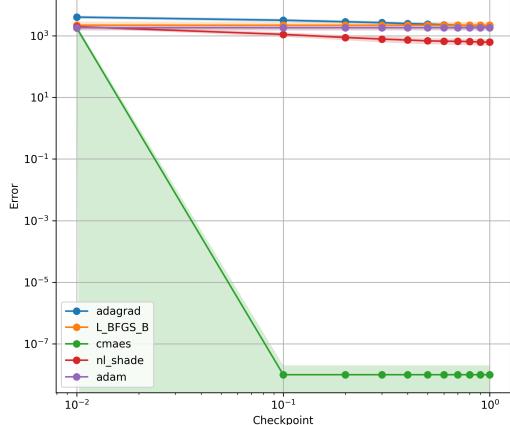
F152013, ECDF, dim=30



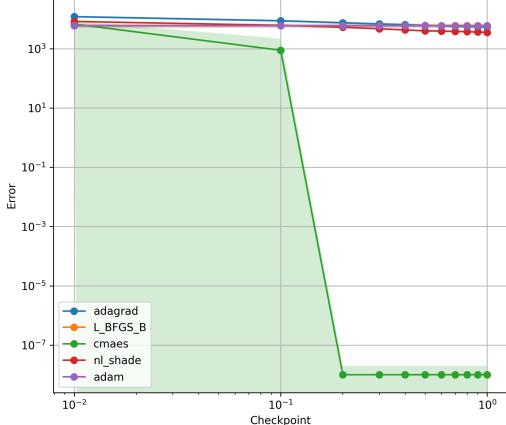
F152013, ECDF, dim=50



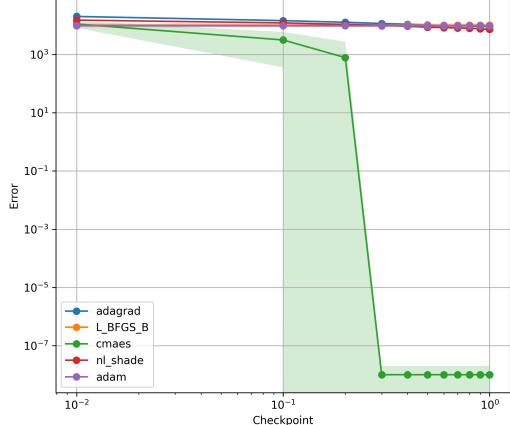
F152013, mean and std, dim=10



F152013, mean and std, dim=30

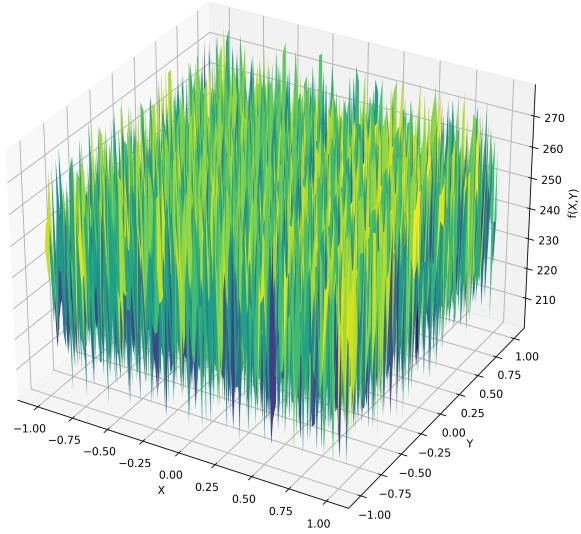


F152013, mean and std, dim=50

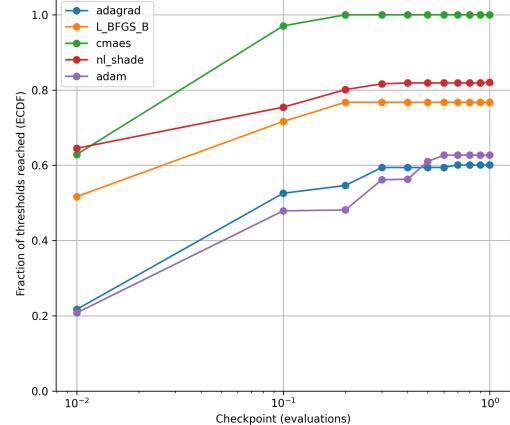


Rotated Katsuura Function

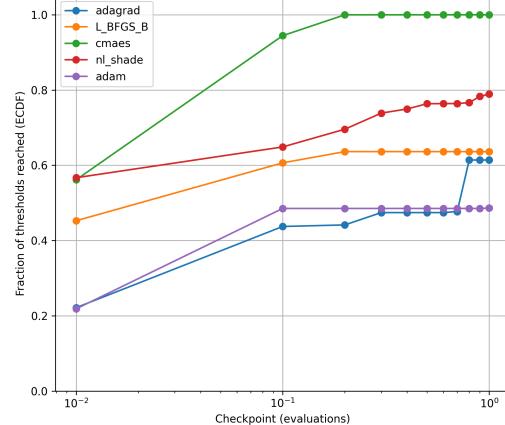
F162013



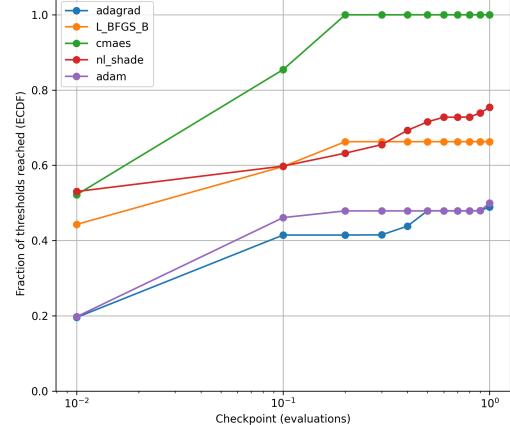
F162013, ECDF, dim=10



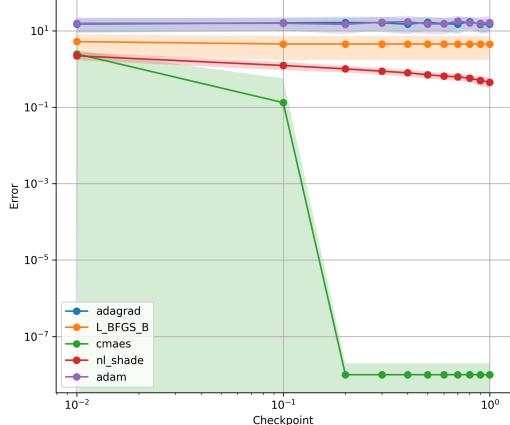
F162013, ECDF, dim=30



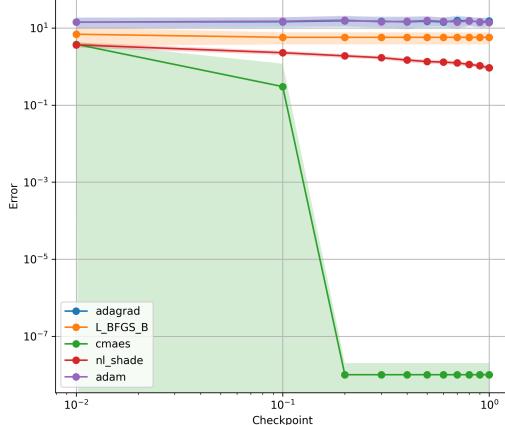
F162013, ECDF, dim=50



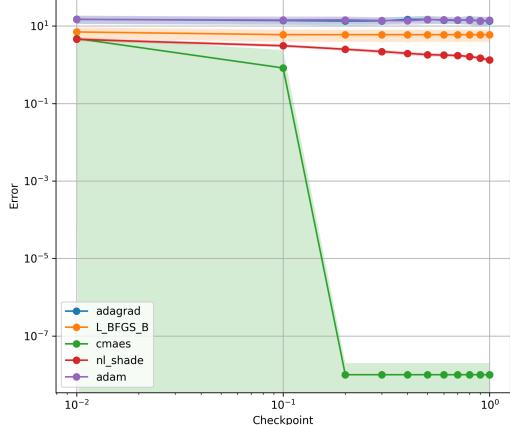
F162013, mean and std, dim=10



F162013, mean and std, dim=30

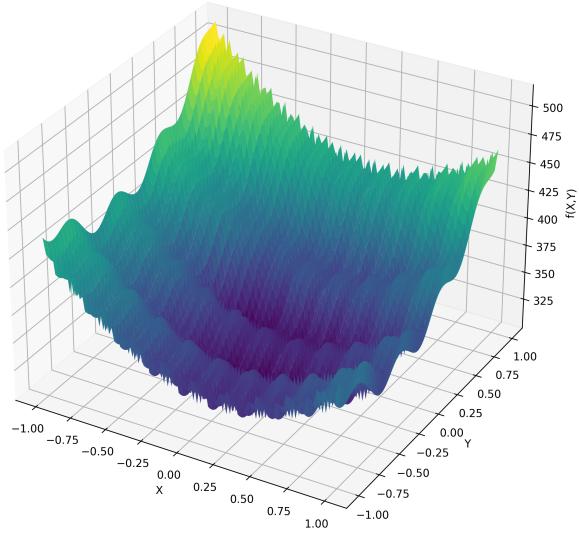


F162013, mean and std, dim=50

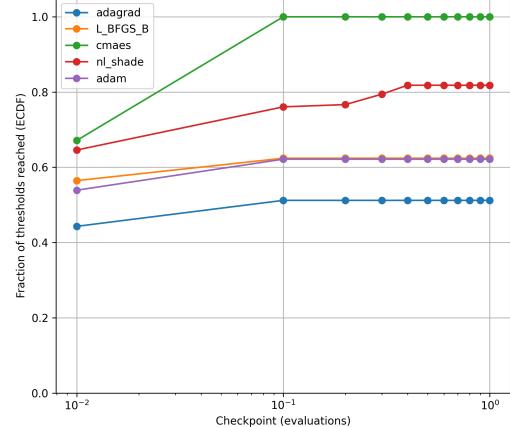


Lunacek Bi-Rastrigin Function

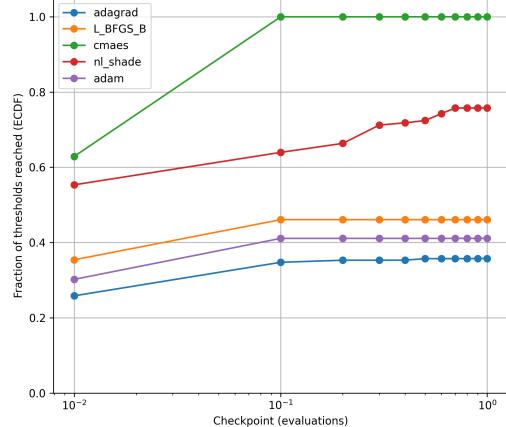
F172013



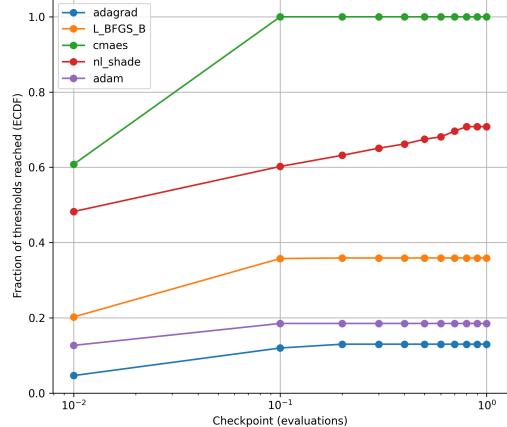
F172013, ECDF, dim=10



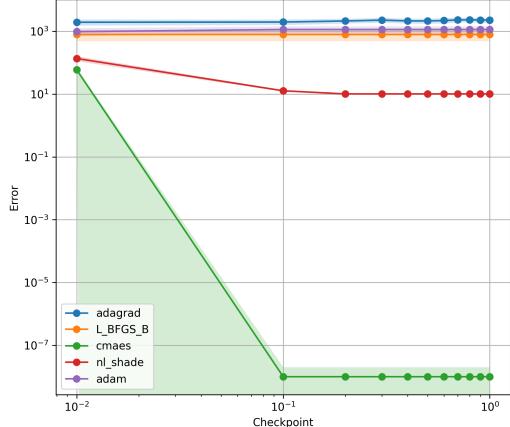
F172013, ECDF, dim=30



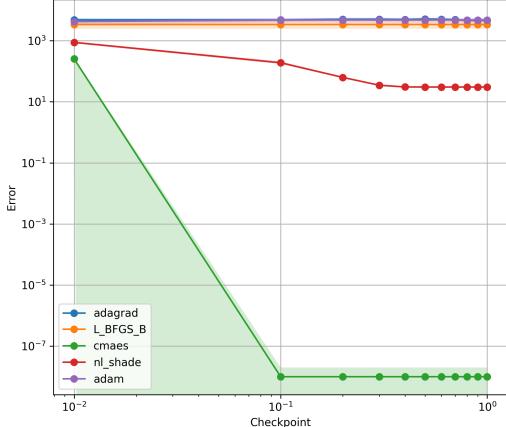
F172013, ECDF, dim=50



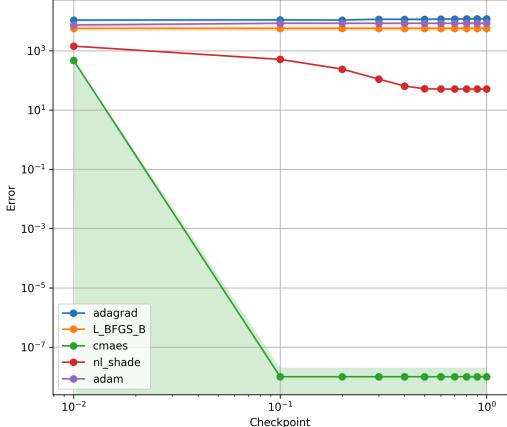
F172013, mean and std, dim=10



F172013, mean and std, dim=30

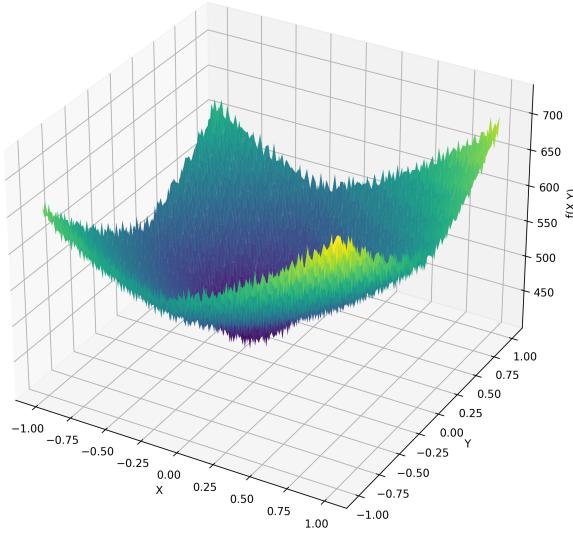


F172013, mean and std, dim=50

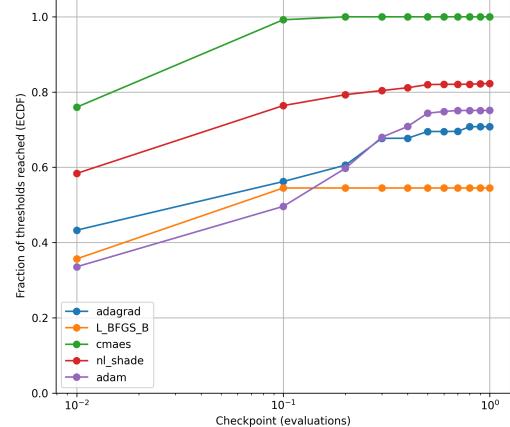


Rotated Lunacek Bi-Rastrigin Function

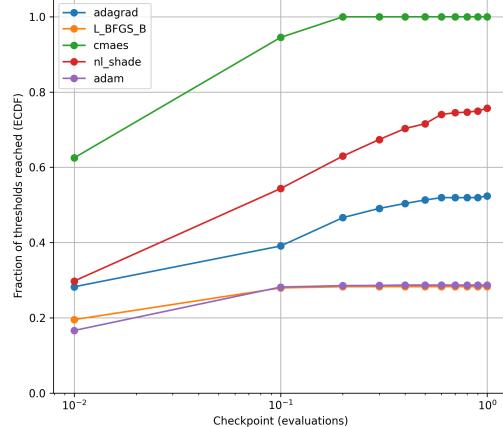
F182013



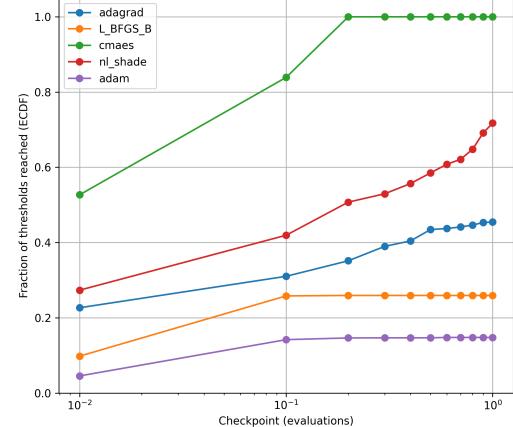
F182013, ECDF, dim=10



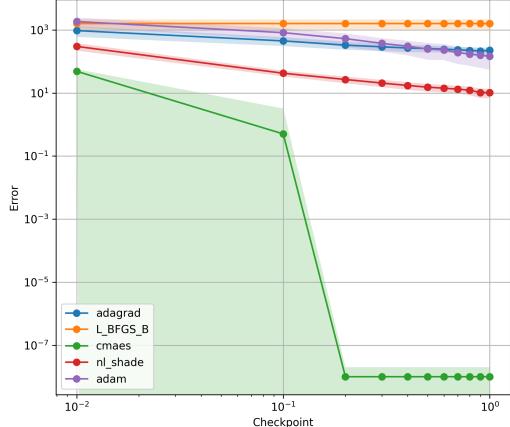
F182013, ECDF, dim=30



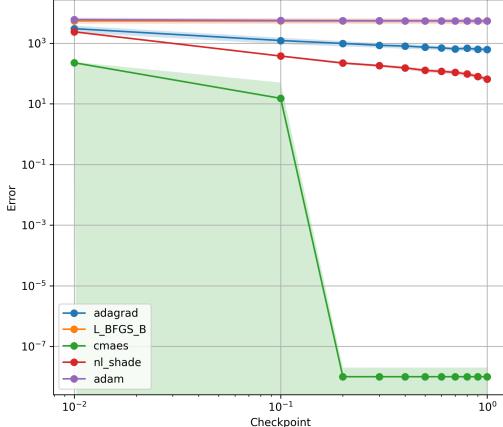
F182013, ECDF, dim=50



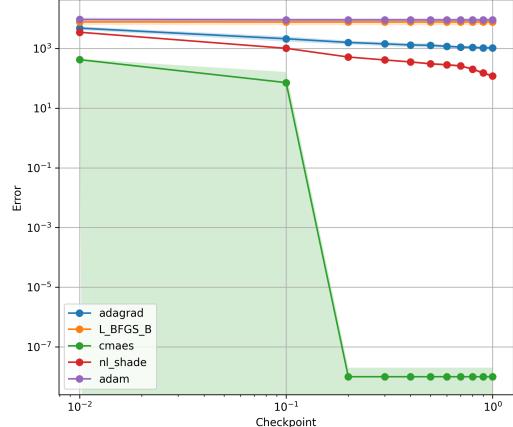
F182013, mean and std, dim=10



F182013, mean and std, dim=30

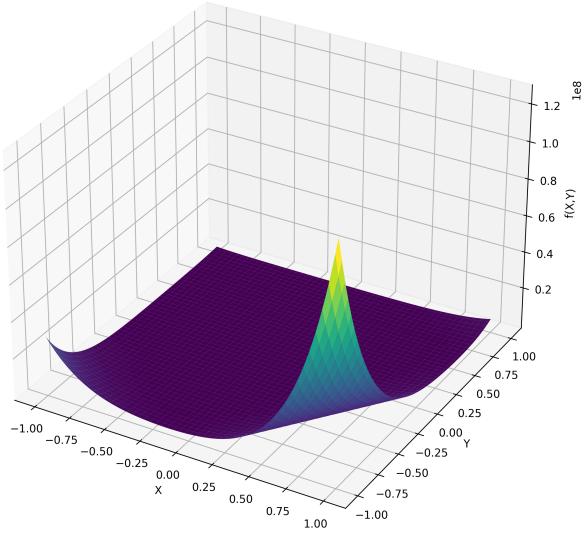


F182013, mean and std, dim=50

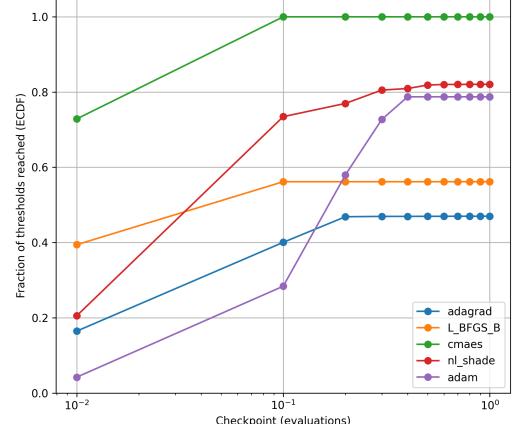


Expanded Griewank's plus Rosenbrock's Function

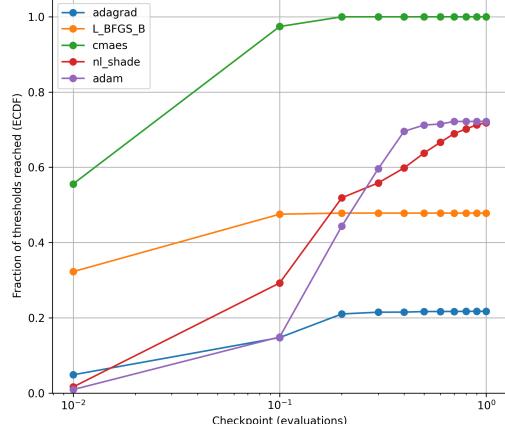
F192013



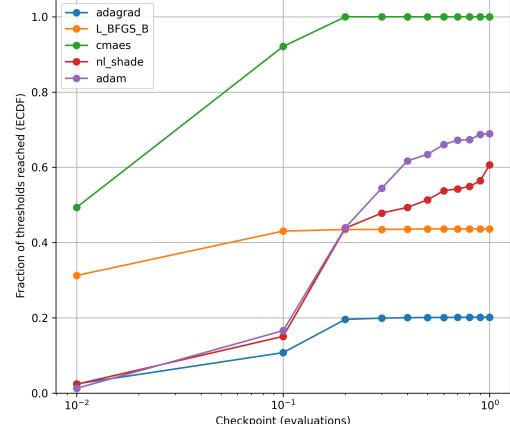
F192013, ECDF, dim=10



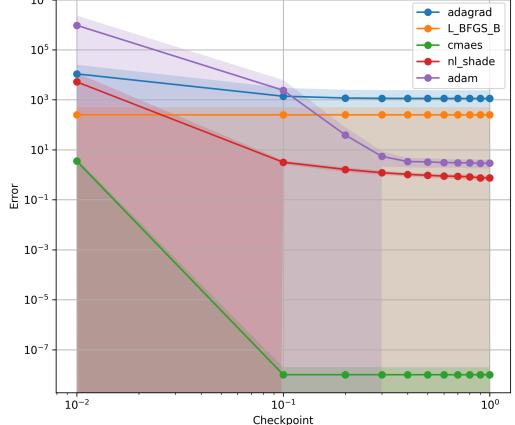
F192013, ECDF, dim=30



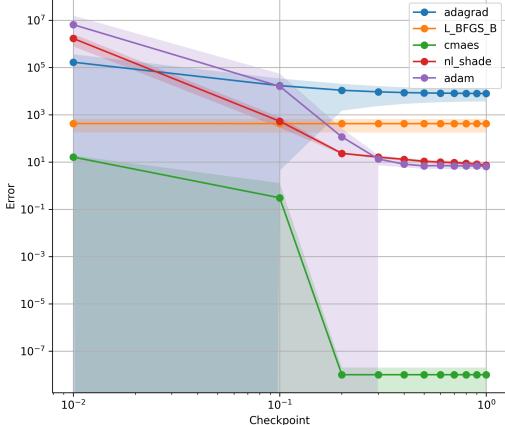
F192013, ECDF, dim=50



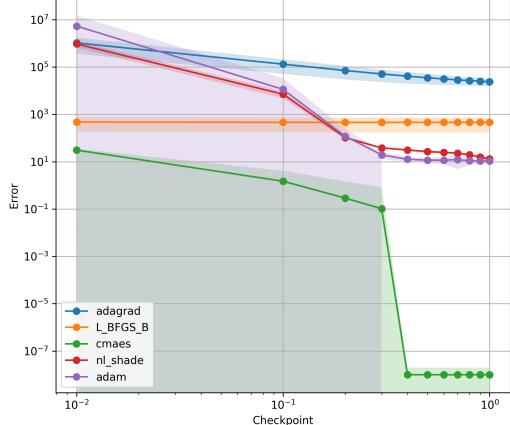
F192013, mean and std, dim=10



F192013, mean and std, dim=30

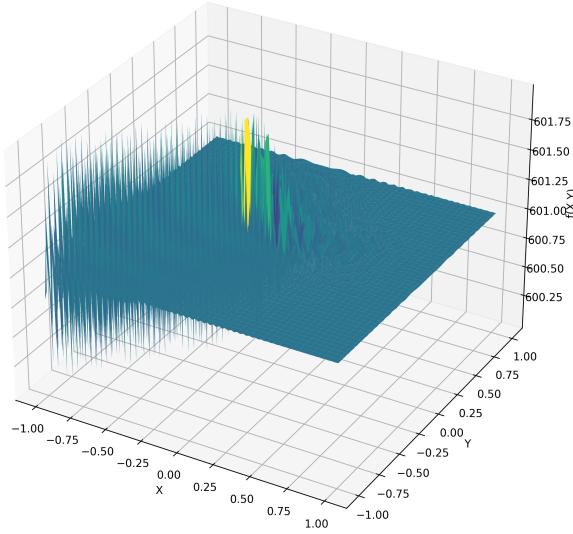


F192013, mean and std, dim=50

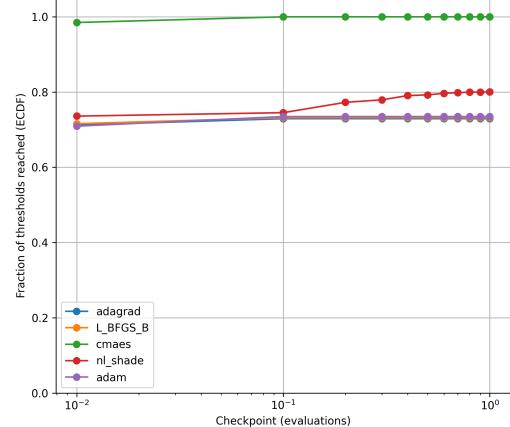


Expanded Scaffer's F6 Function

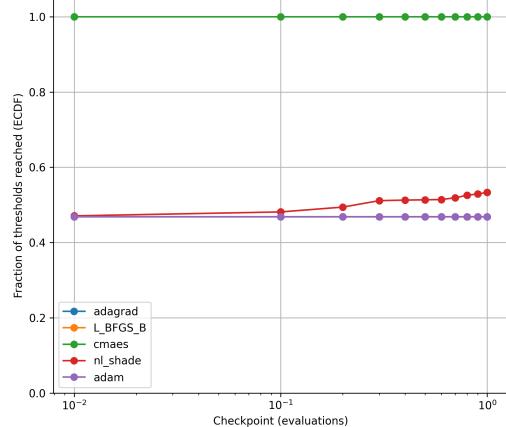
F202013



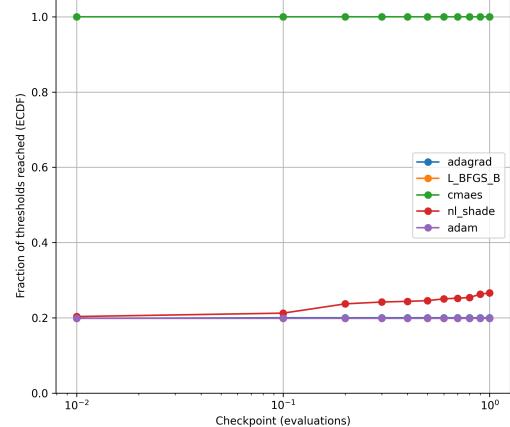
F202013, ECDF, dim=10



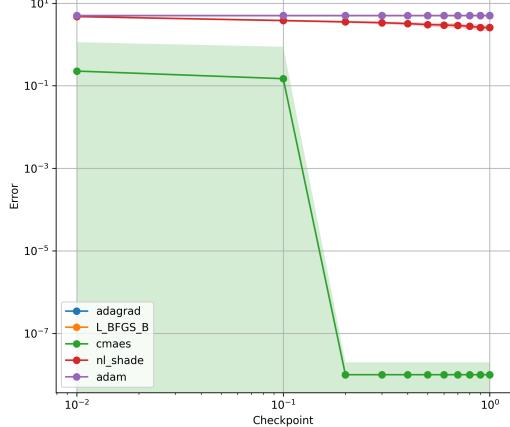
F202013, ECDF, dim=30



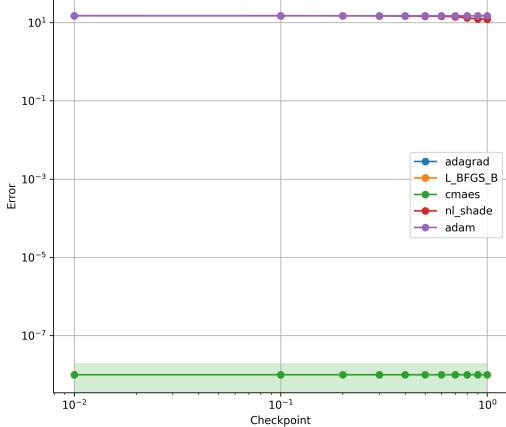
F202013, ECDF, dim=50



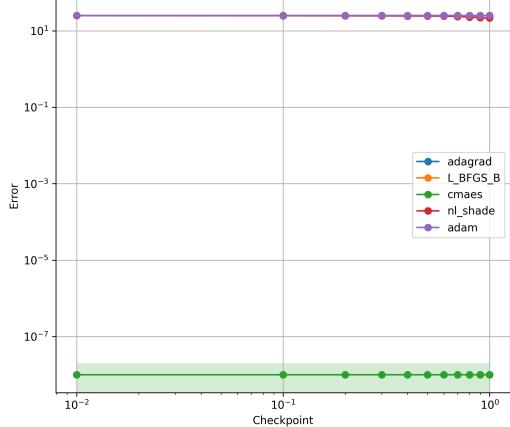
F202013, mean and std, dim=10



F202013, mean and std, dim=30

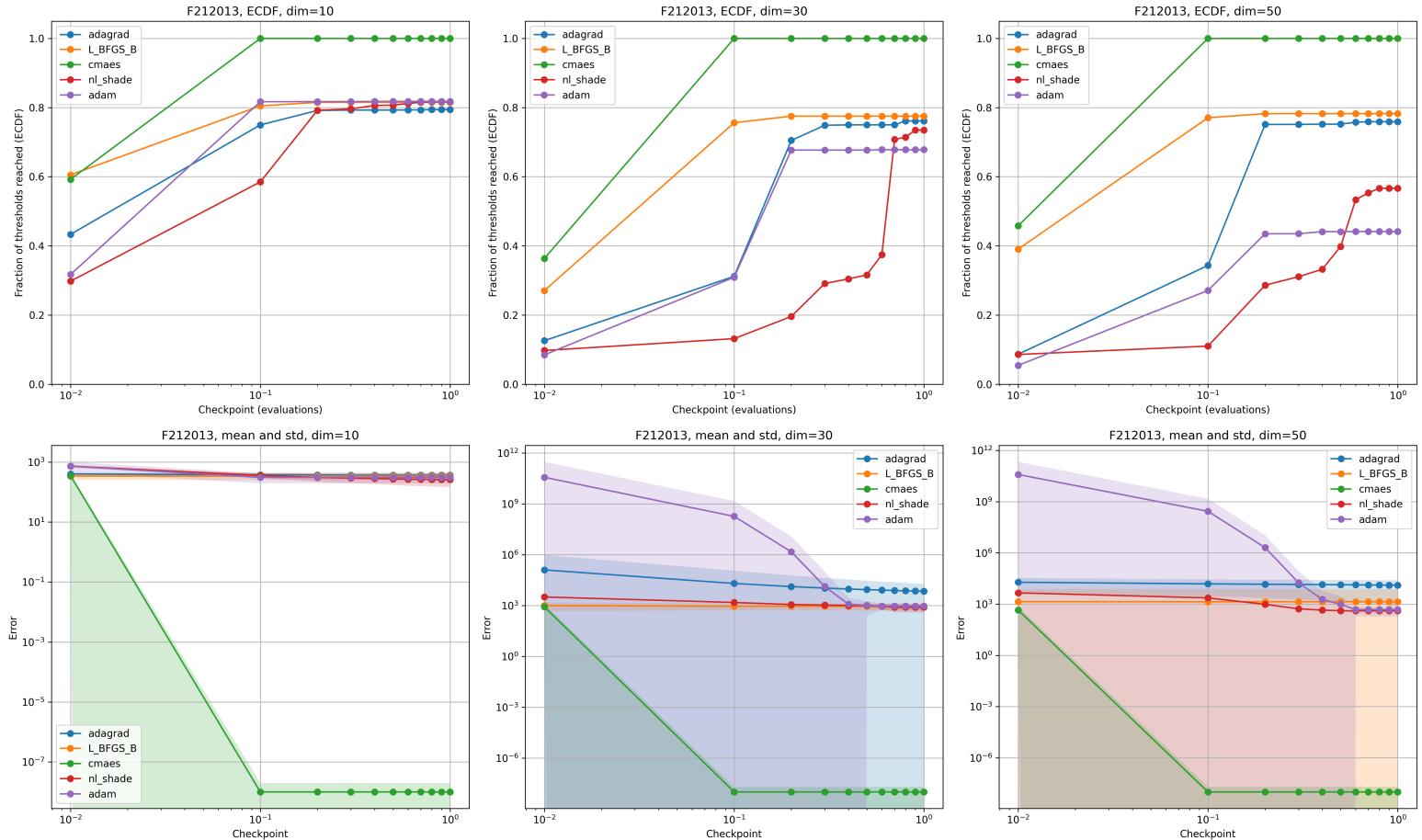
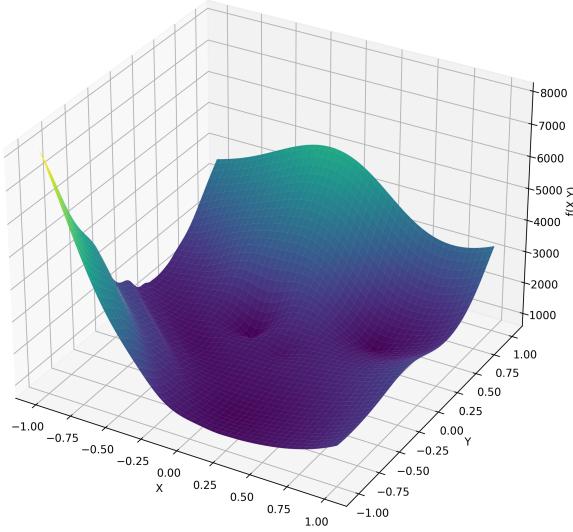


F202013, mean and std, dim=50



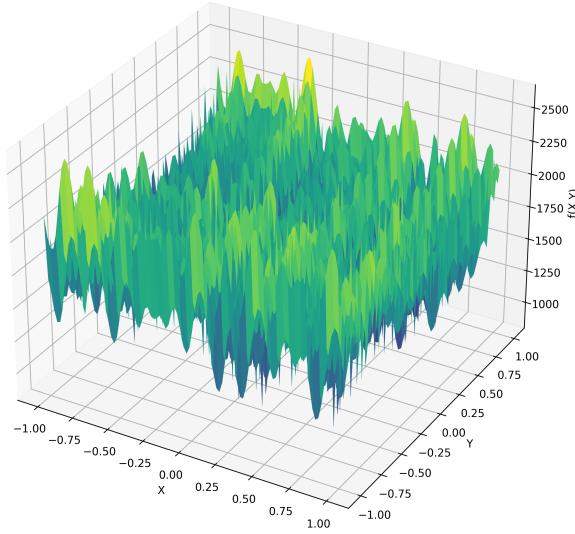
Composition Function 1 (n=5, Rotated)

F212013

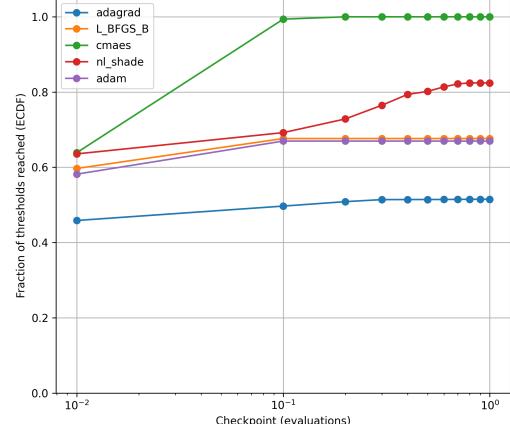


Composition Function 2 (n=3,Unrotated)

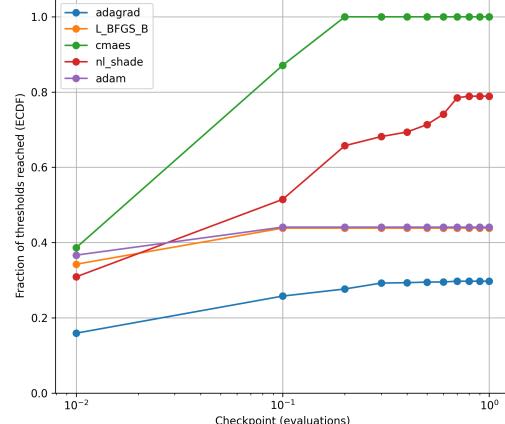
F222013



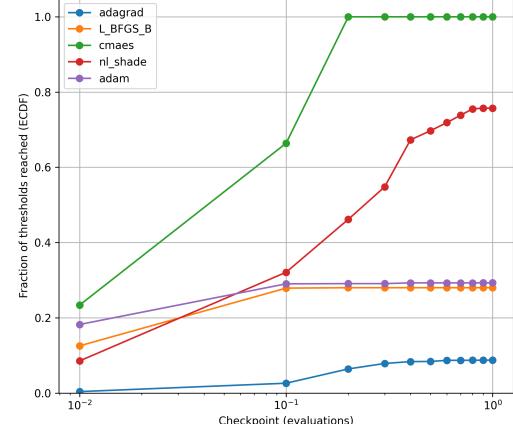
F222013, ECDF, dim=10



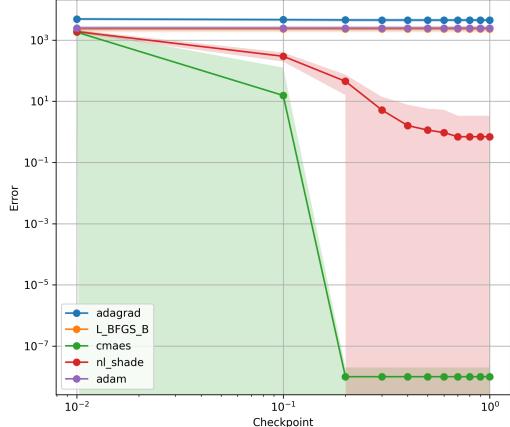
F222013, ECDF, dim=30



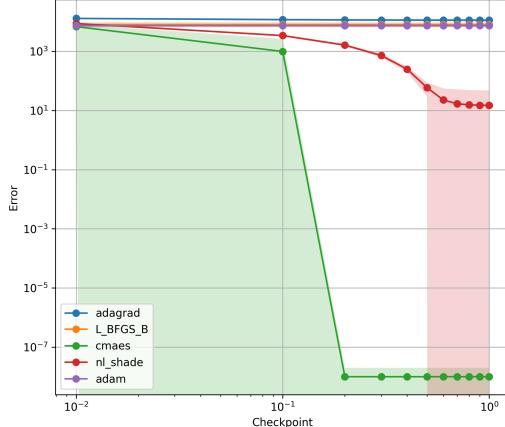
F222013, ECDF, dim=50



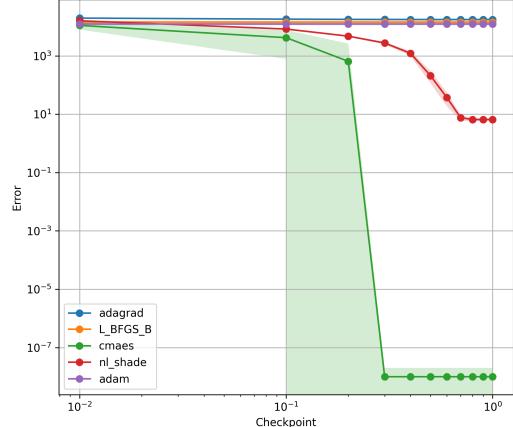
F222013, mean and std, dim=10



F222013, mean and std, dim=30

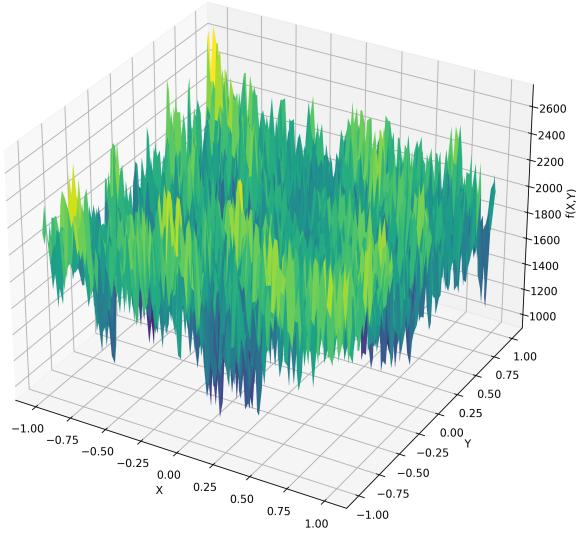


F222013, mean and std, dim=50

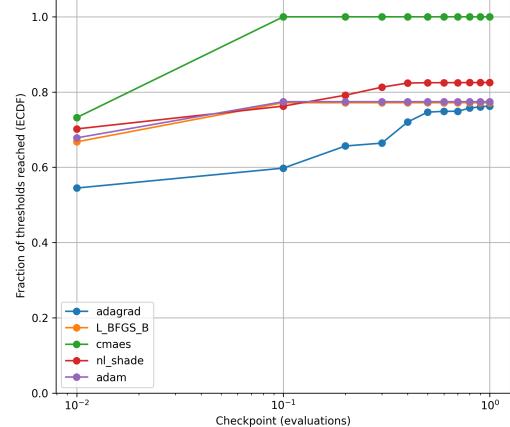


Composition Function 3 (n=3, Rotated)

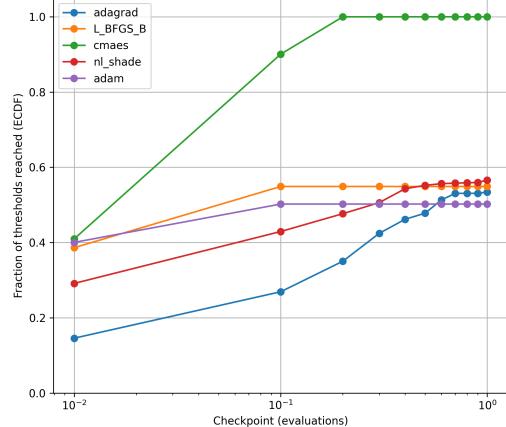
F232013



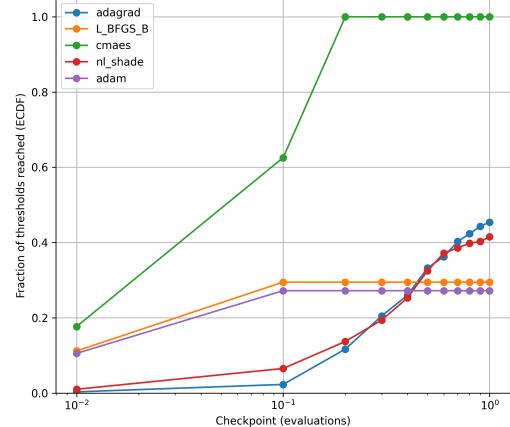
F232013, ECDF, dim=10



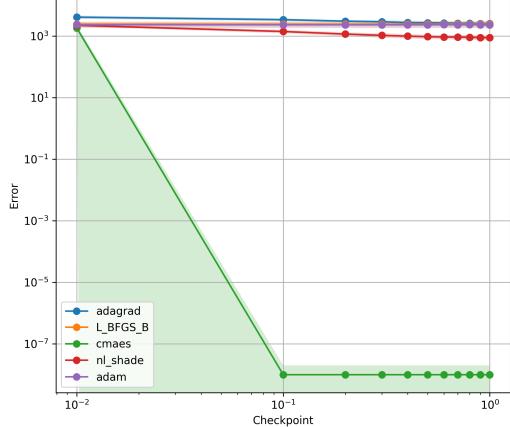
F232013, ECDF, dim=30



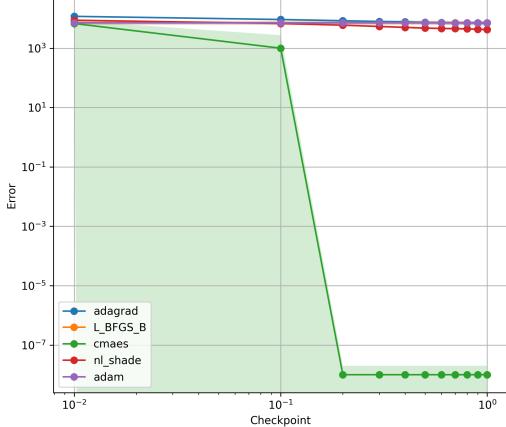
F232013, ECDF, dim=50



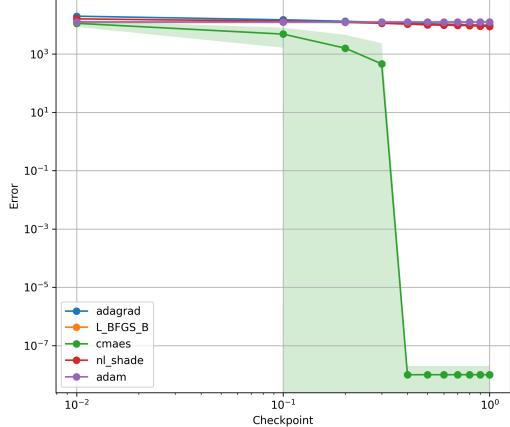
F232013, mean and std, dim=10



F232013, mean and std, dim=30

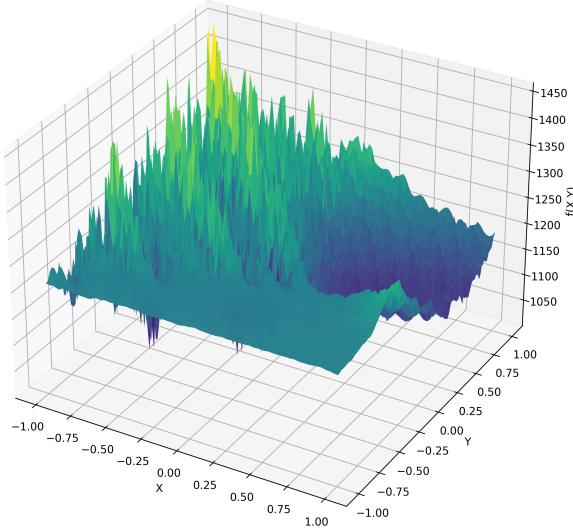


F232013, mean and std, dim=50

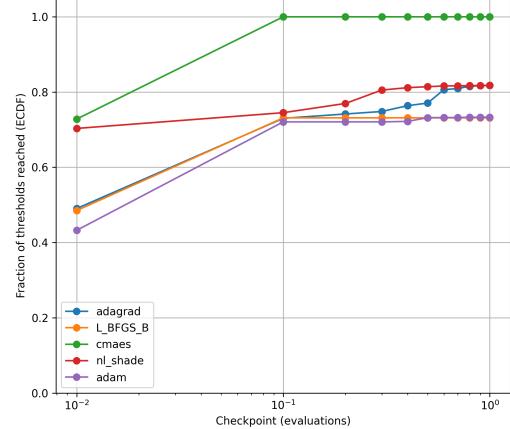


Composition Function 4 (n=3, Rotated)

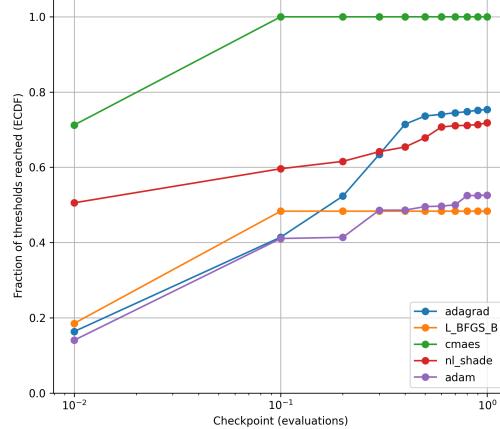
F242013



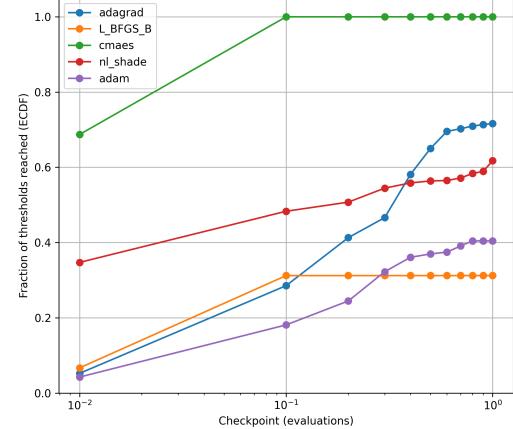
F242013, ECDF, dim=10



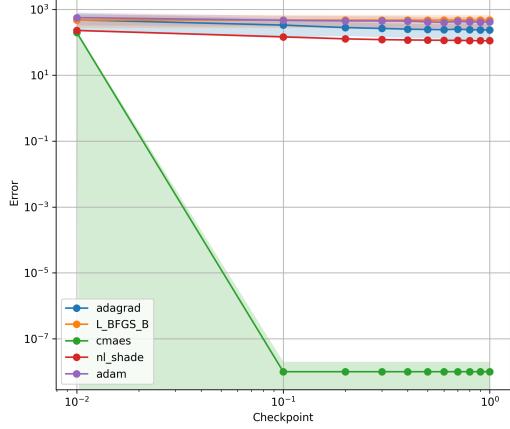
F242013, ECDF, dim=30



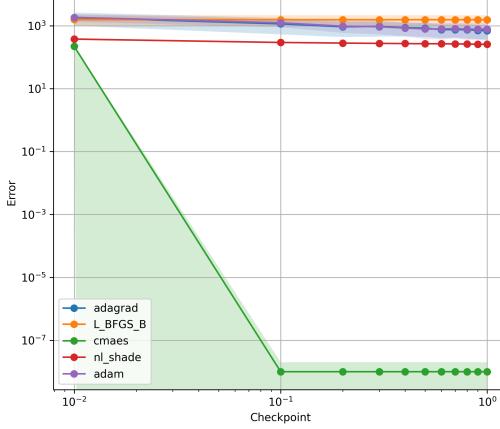
F242013, ECDF, dim=50



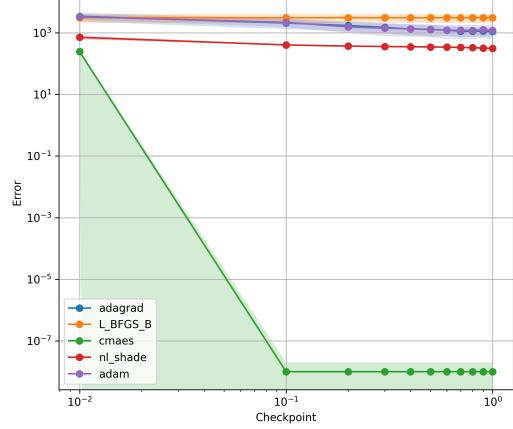
F242013, mean and std, dim=10



F242013, mean and std, dim=30

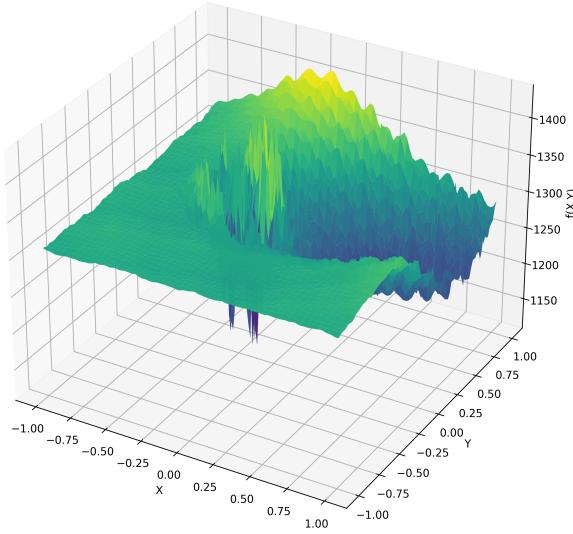


F242013, mean and std, dim=50

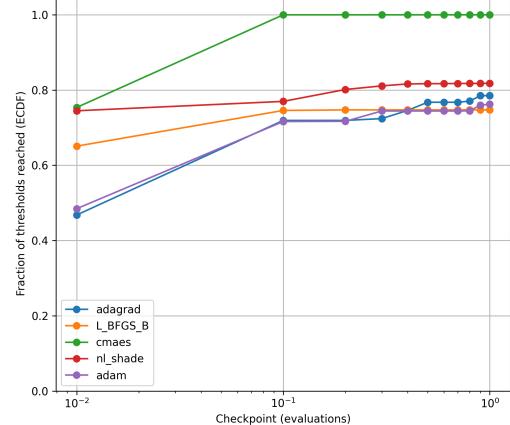


Composition Function 5 (n=3, Rotated)

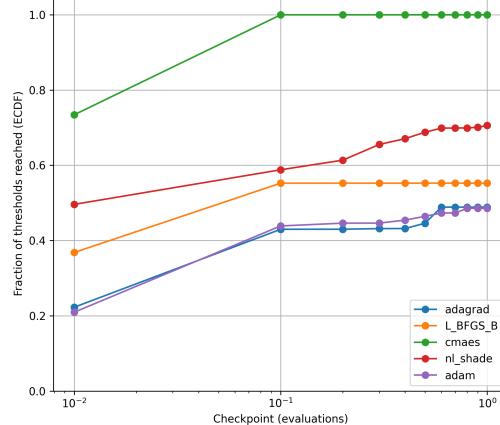
F252013



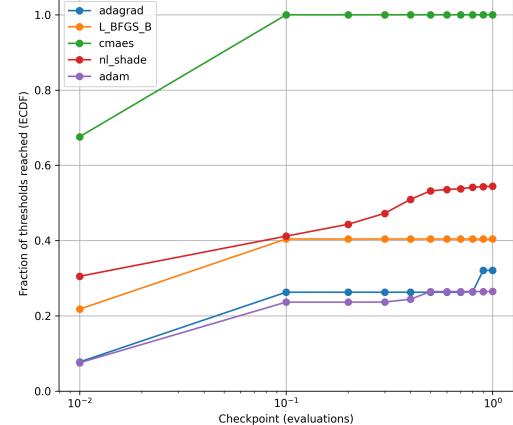
F252013, ECDF, dim=10



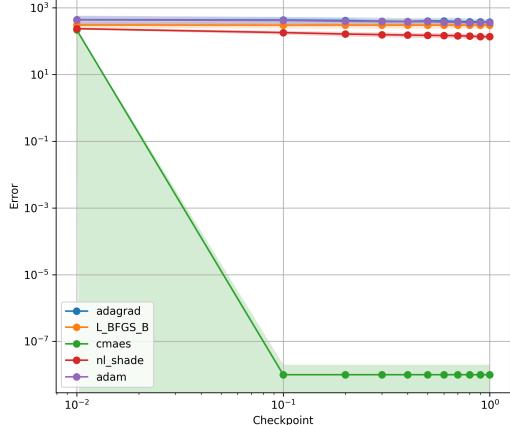
F252013, ECDF, dim=30



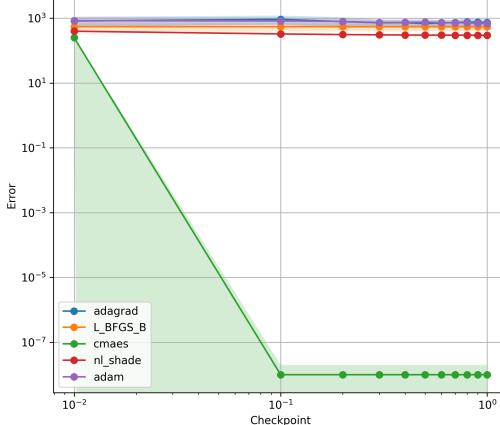
F252013, ECDF, dim=50



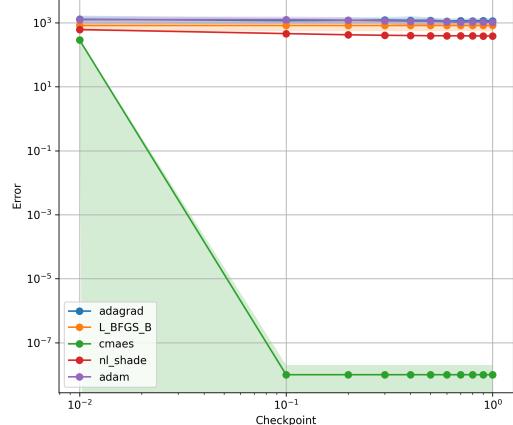
F252013, mean and std, dim=10



F252013, mean and std, dim=30

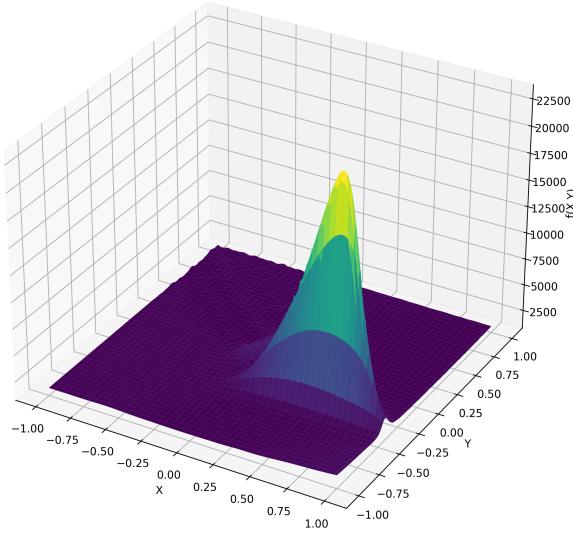


F252013, mean and std, dim=50

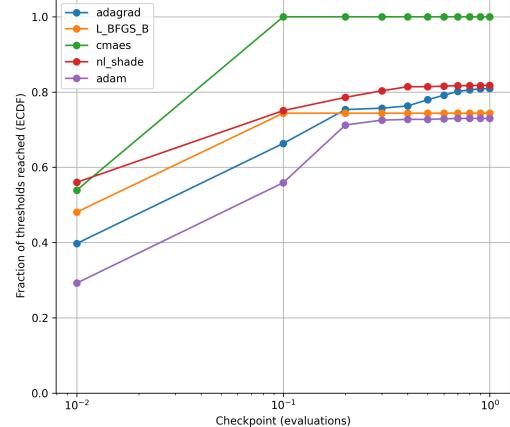


Composition Function 6 (n=5, Rotated)

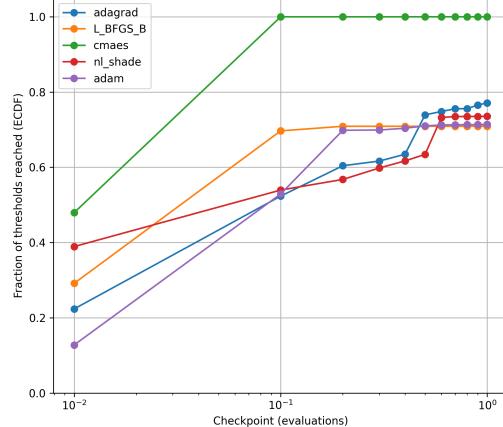
F262013



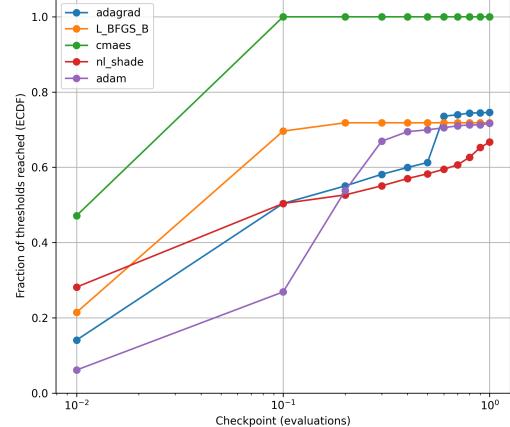
F262013, ECDF, dim=10



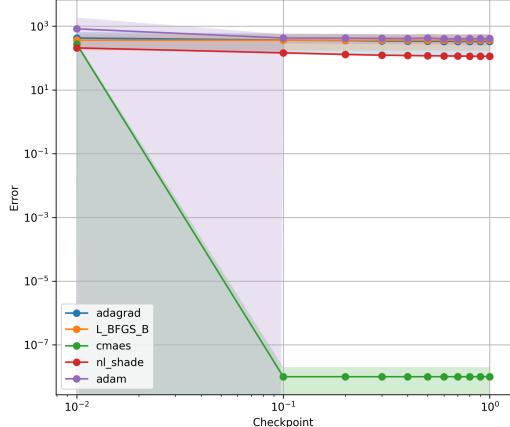
F262013, ECDF, dim=30



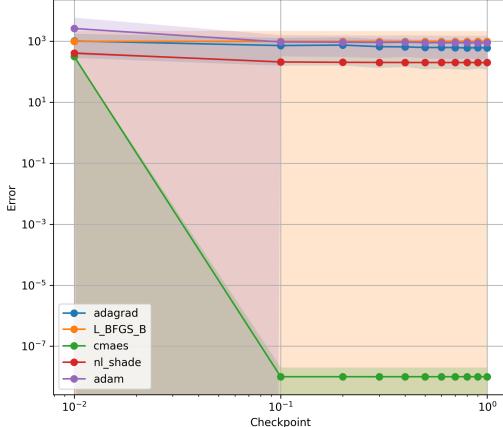
F262013, ECDF, dim=50



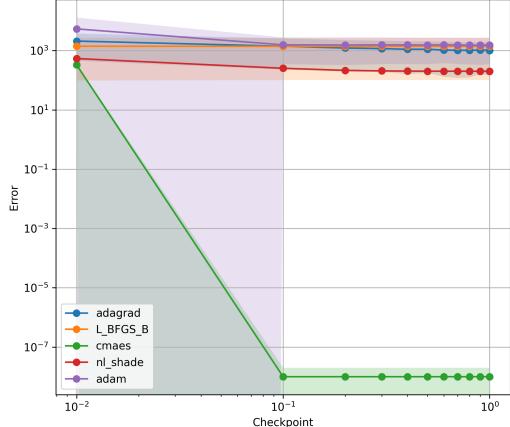
F262013, mean and std, dim=10



F262013, mean and std, dim=30

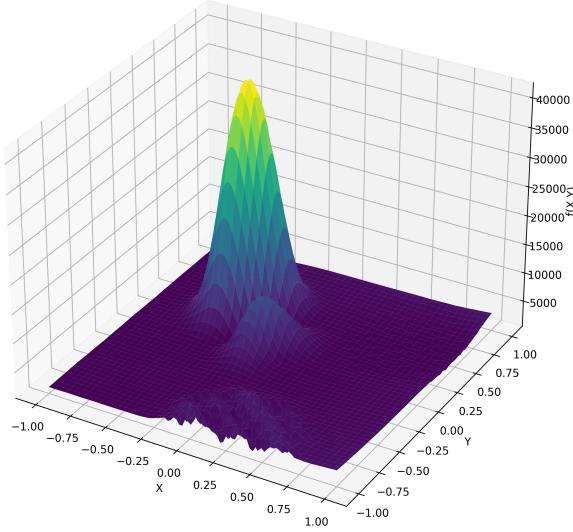


F262013, mean and std, dim=50

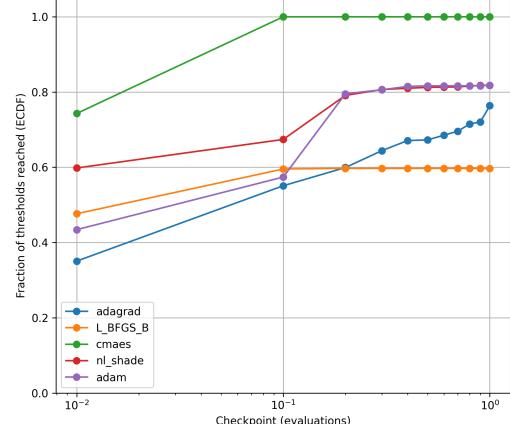


Composition Function 7 (n=5, Rotated)

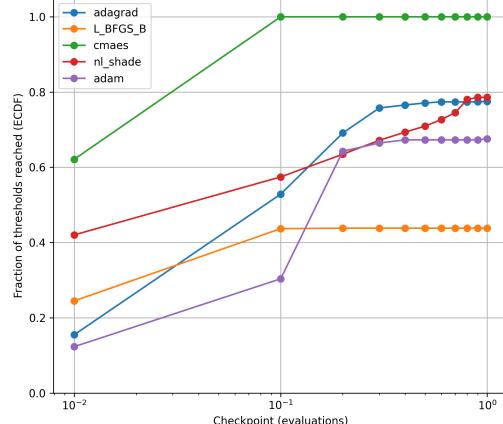
F272013



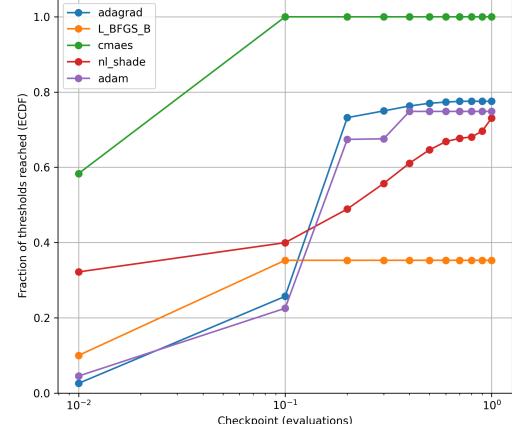
F272013, ECDF, dim=10



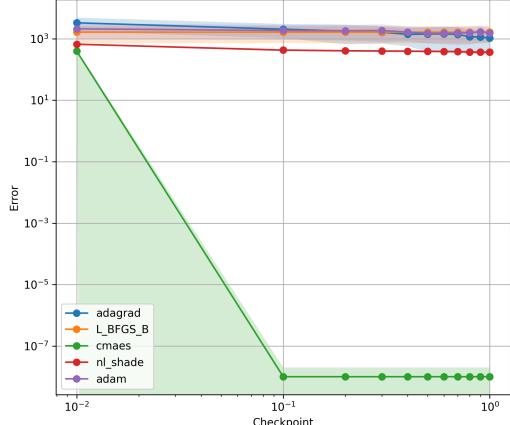
F272013, ECDF, dim=30



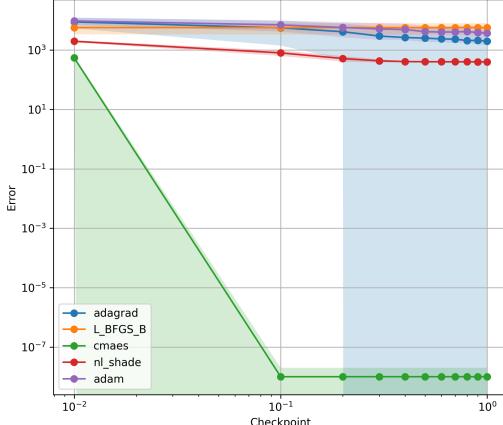
F272013, ECDF, dim=50



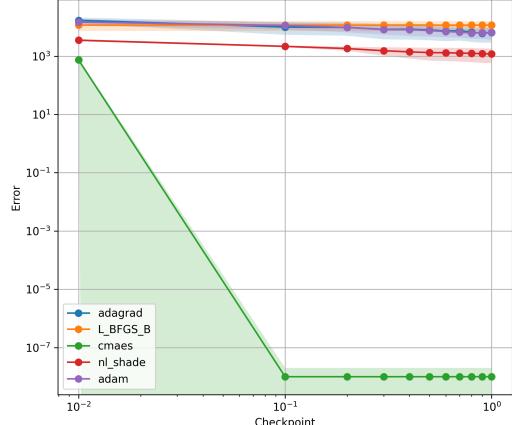
F272013, mean and std, dim=10



F272013, mean and std, dim=30

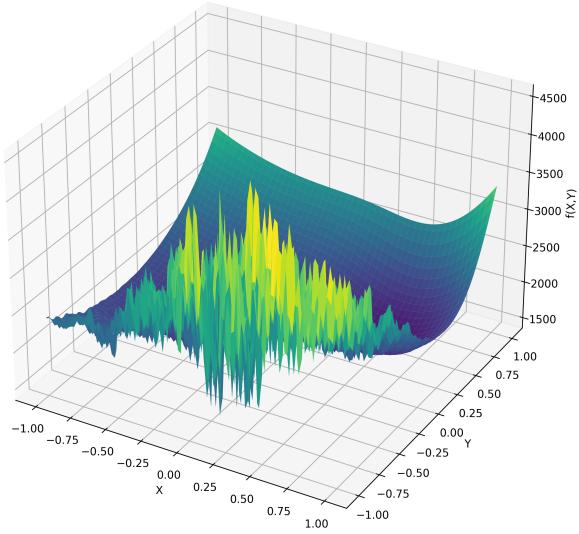


F272013, mean and std, dim=50

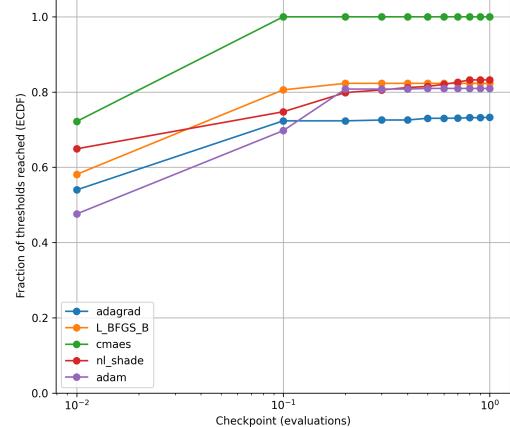


Composition Function 8 (n=5, Rotated)

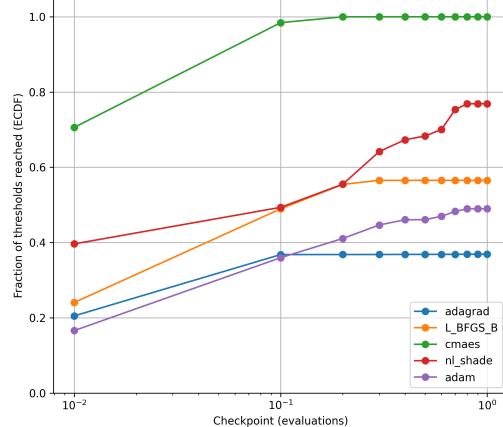
F282013



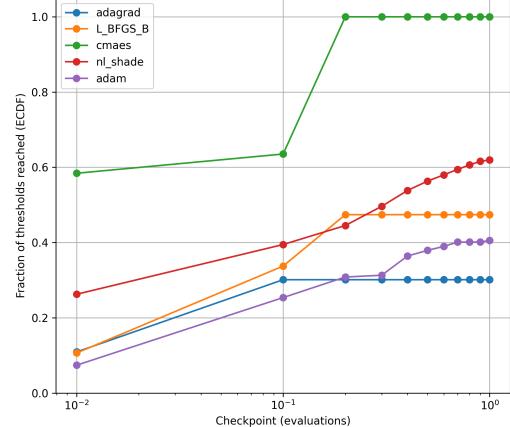
F282013, ECDF, dim=10



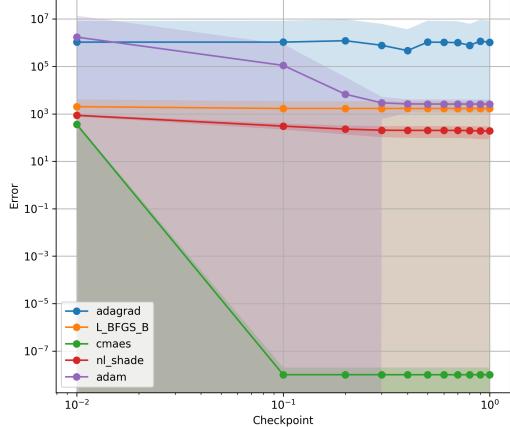
F282013, ECDF, dim=30



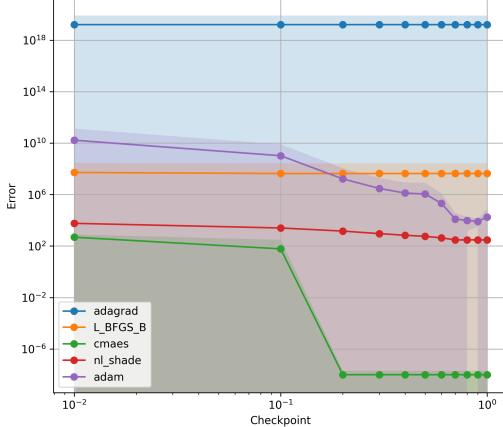
F282013, ECDF, dim=50



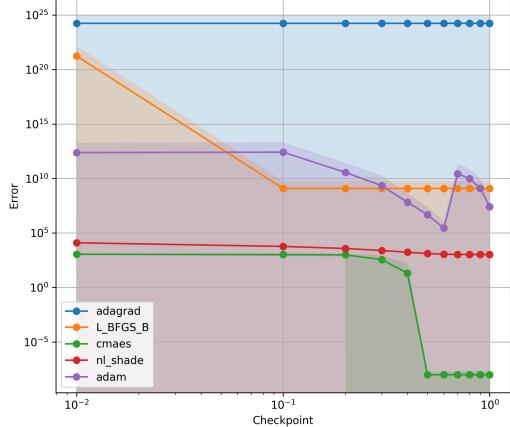
F282013, mean and std, dim=10



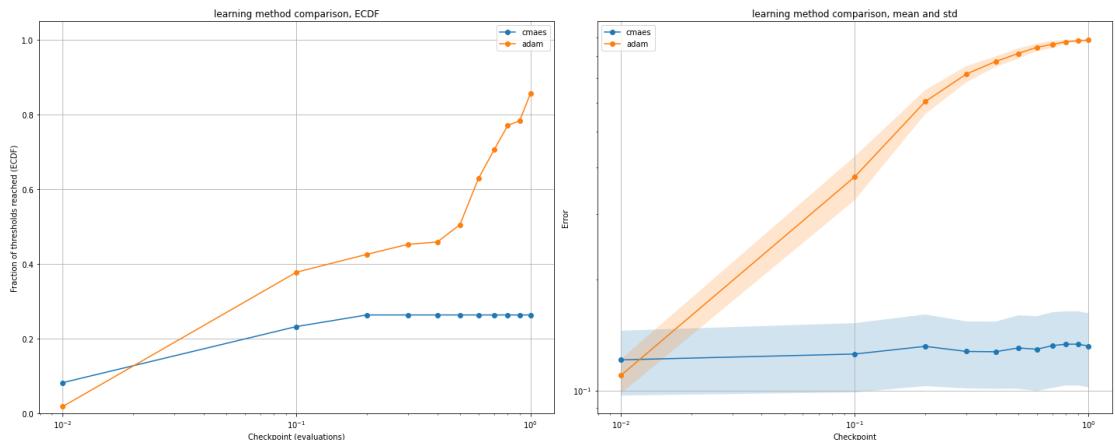
F282013, mean and std, dim=30



F282013, mean and std, dim=50



4.2 Wyniki dla sieci neuronowych



Rysunek 4.1: model przykładowej sieci neuronowej

4.3 Porównanie wyników

Algorytmy globalne vs stochastyczne

- Algorytmy globalne osiągają wyraźną przewagę w zadaniach optymalizacyjnych CEC, zwłaszcza w niskich i średnich wymiarach, natomiast algorytmy stochastyczne sprawdzają się lepiej w optymalizacji sieci neuronowych.

Adam i Adagrad

- Pomimo zastosowania momentum, oba algorytmy osiągają podobne wyniki.
- Nie działają optymalnie, gdy zbiory minimów lokalnych funkcji (doliny funkcji) tworzą „rynnę” nierównoległe do osi układu współrzędnych. Można porównać funkcję F5 - “Different Powers Function” z “doliną” równoległą do osi, oraz F3 - “Rotated Bent Cigar Function”.
- Algorytm Adam, pomimo wielu zalet, nie zawsze radzi sobie lepiej względem algorytmu Adagrad. Widać problemy gdy funkcja ma gwałtowne spadki jak funkcja F21 - “Composition Function 1 (n=5, Rotated)”. Można zauważyc, że radzi sobie znaczco gorzej i różnica w wynikach jest większa im więcej jest wymiarów.

L-BFGS-B

- Wykazuje podobne ograniczenia co Adam i Adagrad. Można zauważyc te same problemy dla funkcji o „rynnowej” strukturze minimów. Jednak zwykle osiąga lepsze wyniki, czasem porównywalne z algorytmami globalnymi.
- Dla większości przypadków krzywe ECDF tego algorytmu często przypominają wykresy Adagrad, przesunięte o około 20% w zakresie przebitych progów jakości.
- Jest bardzo wrażliwy na szum i chaotyczność funkcji, ponieważ aproksymacja hejsjanu staje się niestabilna (F9 - “Rotated Weierstrass Function”).
- Dla funkcji “Rotated Lunacek Bi_Rastrigin Function” widać, że czasami Adagrad i Adam potrafią przebić jakość rozwiązań algorytmu L-BFGS-B, jeśli funkcja ma wystarczajaco dużo szumu i jest wystarczajaco chaotyczna wizualnie.

CMA-ES

- Osiąga najlepsze wyniki w funkcjach CEC2013 w porównaniu do pozostałych algorytmów.
- Dzięki rozproszeniu populacji i estymacji średnich wartości parametrów, pozwala na dokładniejszą eksplorację krajobrazu funkcji i szybsze znalezienie ekstremów przy mniejszej liczbie wymiarów.
- Jak teoretyzowano tworząc nl-shade-rsp-mid, prawdopodobnie ma to związek z teorią, że środek populacji ma większe szanse być lepszą aproksymacją ekstremum od pojedynczego punktu.
- Czas obliczeń algorytmów stochastycznych rośnie liniowo względem ilości wymiarów, natomiast algorytmów globalnych potęgowo. Z wyników CEC widać, że algorytmy globalne radzą sobie o wiele lepiej od stochastycznych dla większej liczby wymiarów, jednak dla problemu optymalizacji sieci neuronowej algorytmy globalne przestają być opłacalne.

nl-shade-rsp-mid

- Działa wyraźnie gorzej niż CMA-ES jako algorytm globalny. Optymalizacja zajmuje znaczco dłużej, a punkty nie są równie dobrej jakości.
- Zmiana metody krzyżowania w połowie przebiegu sugeruje, że algorytm był zaprojektowany z nadzieją na dłuższy czas działania, w przeciwieństwie do CMA-ES, który konsekwentnie wykorzystuje tę samą strategię przez cały przebieg.

Sieci neuronowe

- Klasyczne sieci z algorytmem stochastycznym minimalizującym funkcję straty nie zachowują zależności między neuronami — każda aktualizacja wagi jest lokalna, liniowa i niezależna.
- Algorytmy globalne zmieniają wszystkie wagi jednocześnie, co zwiększa złożoność obliczeniową i ogranicza skalowalność w zastosowaniach sieciowych.

Rozdział 5

Podsumowanie i wnioski

5.1 Wnioski

Algorytmy stochastyczne

- Poruszają się niezależnie w każdym wymiarze, co sprawia, że są wrażliwe na kierunki przekątne w krajobrazie funkcji.

Wpływ szumu

- Im bardziej złożony algorytm stochastyczny, tym bardziej wrażliwy jest na szum i chaotyczność funkcji.

Charakter algorytmów

- Adam, Adagrad i L-BFGS-B aktualizują parametry jeden wymiar na raz, co powoduje problemy w funkcjach, których minima lokalne tworzą obiekt nierównoległy do osi układu (np. F3 - “Rotated Bent Cigar Function”).
- CMA-ES i inne algorytmy globalne, dzięki rozproszonej populacji, radzą sobie lepiej z kierunkami przekątnymi i wykrywaniem ekstremów w krajobrazie funkcji.

Skalowalność

- Algorytmy stochastyczne skalują się liniowo względem liczby wymiarów.
- Algorytmy globalne, jak CMA-ES, skalują się potęgowo, co ogranicza ich opłacalność przy dużych wymiarach, mimo ich dokładności dla mniejszych wymiarów.

Cechy specyficzne algorytmów

- Zmiana strategii w trakcie działania, np. w nl-shade-rsp-mid, sugeruje nastawienie na długotrwałą eksplorację, w odróżnieniu od CMA-ES, który utrzymuje stałą strategię.

5.2 Osiągnięte cele

- Zaimplementowane środowisko do testowania algorytmów sztucznej inteligencji, z możliwością rozszerzenia o kolejne w przyszłości.
- Opisano działania algorytmów.
- Określone zostały zakresy kompetencji podanych algorytmów.
- Zostały porównane ich wady oraz zalety.

5.3 Kierunki dalszego rozwoju

- Aktualne rozwiązania wykorzystujące algorytmy globalne do sieci neuronowej nie przenoszą informacji o zależności pomiędzy neuronami, wagi są spłaszczone do jednego wymiaru, ignorując powiązania pomiędzy neuronami. Potencjalnie możliwe jest zaprojektowanie algorytmu globalnego dostosowanego specyficznie pod naukę sieci neuronowych, który poradziłby sobie lepiej.
- Implementacja kolejnych algorytmów i porównanie ich z obecnymi wynikami.
- Rozszerzenie testów o kolejne benchmarki, takie jak CEC2017.
- Badanie hybrydowych metod optymalizacji, łączących algorytmy globalne z lokalnymi lub stochastycznymi.
- Opracowanie metod redukcji wymiarów dla zadań optymalizacji sieci neuronowych metodami globalnymi.
- Badanie wpływu szumu w danych wejściowych lub funkcji kosztu na skuteczność algorytmów.

Rozdział 6

Załączniki

Bibliografia

- <https://optimization.cbe.cornell.edu/index.php?title=AdaGrad>
- https://en.wikipedia.org/wiki/Stochastic_gradient_descent#AdaGrad
- <https://docs.pytorch.org/docs/stable/generated/torch.optim.Adam.html>
- https://en.wikipedia.org/wiki/Stochastic_gradient_descent#Adam
- <https://www.geeksforgeeks.org/deep-learning/adam-optimizer/>
- https://en.wikipedia.org/wiki/Limited-memory_BFGS
- <https://dl.acm.org/doi/pdf/10.1145/279232.279236>
- <https://esezam.okno.edu.pl/mod/book/view.php?id=1245&chapterid=1950>
- https://www.researchgate.net/publication/353782316_NL-SHADE-RSP_Algorithm_with_Adaptive_Archive_and_Selective_Pressure_for_CEC_2021_Numerical_Optimization/link/6226e6573c53d31ba4b03f4a/download?tp=eyJjb250ZXh0Ijp7ImZpcnN0UGFr