

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

Michał Brzeziński

19 stycznia 2026

Spis treści

Streszczenie	3
1 Wstęp	4
1.1 Cel i zakres pracy	4
2 Algorytmy	5
2.1 Gradientowe	5
2.1.1 AdaGrad	5
2.1.2 Adam	6
2.1.3 L-BFGS-B	7
2.2 Globalne	7
2.2.1 CMAES	7
2.2.2 NL-SHADE-RSP-MID	8
3 Sposoby analizy	10
3.1 Metody testowania	10
3.2 Benchmarki CEC	10
3.2.1 Opis problemu	10
3.2.2 Implementacja	11
3.3 Sieci neuronowe	11
3.3.1 Opis problemu	11
3.3.2 Implementacja	11
4 Wyniki eksperymentów	14
4.1 Wyniki dla benchmarków CEC	14
4.2 Wyniki dla sieci neuronowych	43
4.3 Porównanie wyników	43
5 Podsumowanie i wnioski	44
5.1 Podsumowanie wniosków obu sposobów	44
5.2 Osiągnięte cele	44
5.3 Kierunki dalszego rozwoju	45

Bibliografia	46
6 Załączniki	47

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

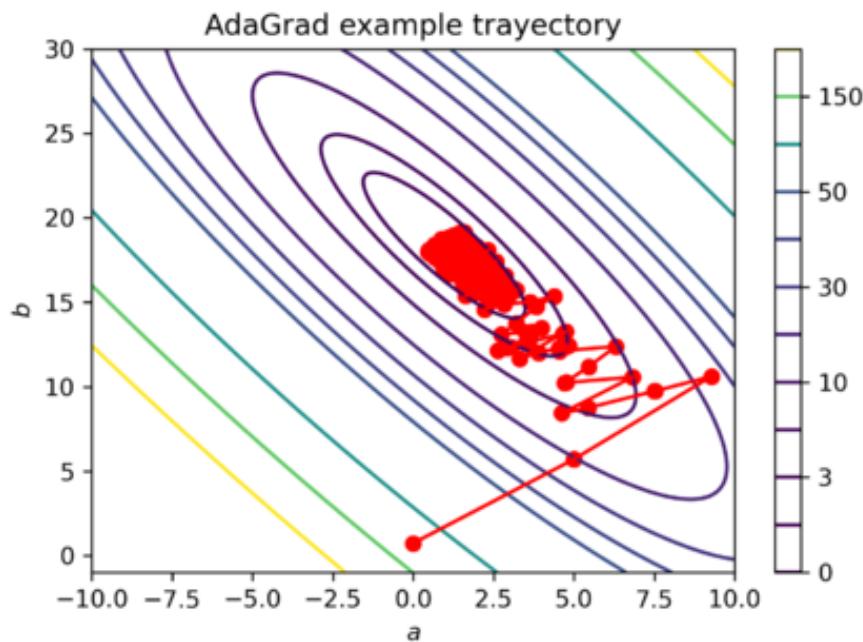
Algorytmy

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 przeходить z fazy eksploracji w fazę eksplotacji.



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 koncept 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 wygłuszając za duże wartości.

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

dla :

$$m_w^{(t)} := \beta_1 m_w^{(t-1)} + (1 - \beta_1) \nabla Q(w)^{(t)}$$

$$\hat{m}_w^{(t)} = \frac{m_w^{(t)}}{1 - \beta_1^t}$$

$$v_w^{(t)} := \beta_2 v_w^{(t-1)} + (1 - \beta_2) (\nabla Q(w)^{(t)})^2$$

$$\hat{v}_w^{(t)} = \frac{v_w^{(t)}}{1 - \beta_2^t}$$

$$w^{(t)} := w^{(t-1)} - \eta \frac{\hat{m}_w^{(t)}}{\sqrt{\hat{v}_w^{(t)}} + \epsilon}$$

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.

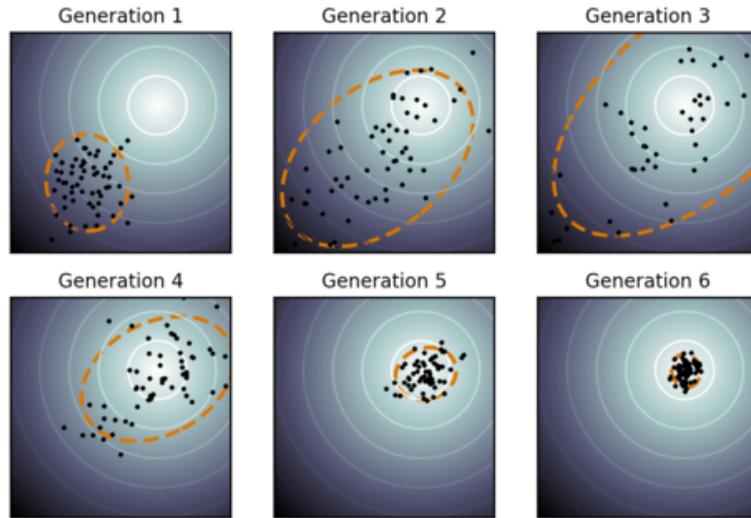
L-BFGS-B to rozszerzenie algorytmu BFGS o ograniczenia przestrzeni (Box constraints) oraz pamięci (Limited memory). Przestrzeń rozwiązań jest ograniczona, a hesjan jest aproksymowany na podstawie macierzy ustalonej liczby ostatnich kroków, pracuje na ograniczonej pamięci nie wykorzystując pełnej macierzy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

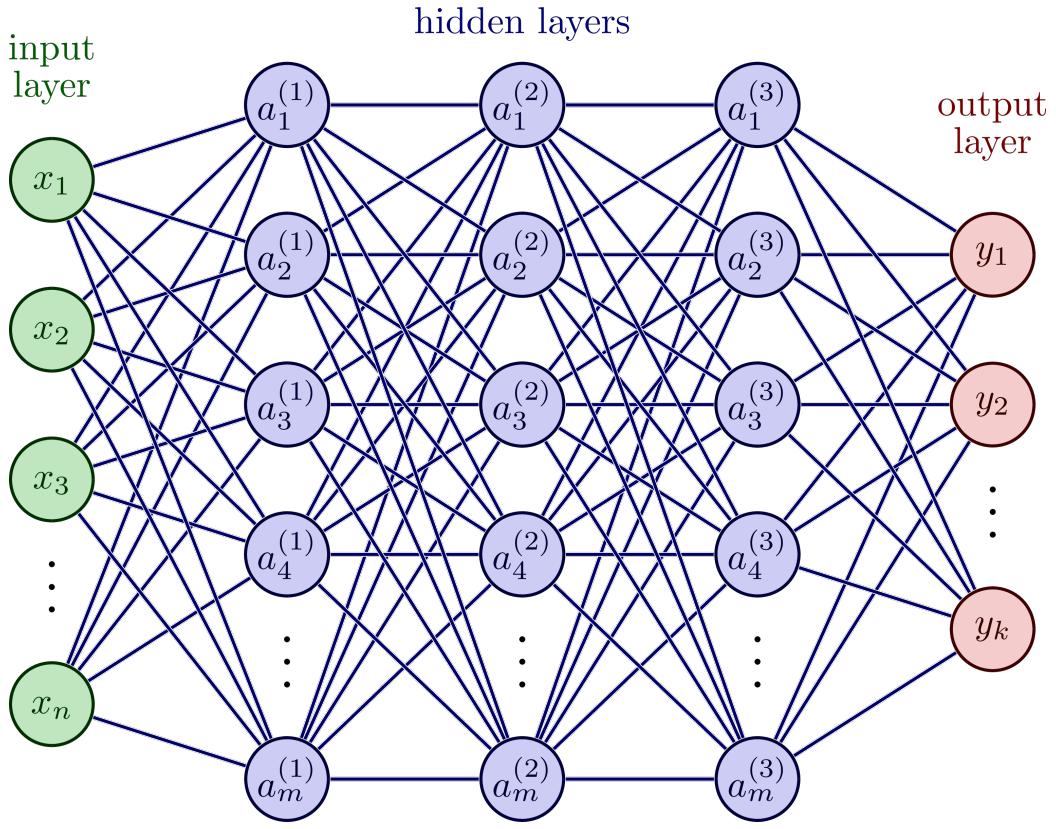
Nie pozwala to na określenie pełnego zakresu możliwych problemów optymalizacyjnych.

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

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 zwiększenie 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 zwiększeniem maksymalnym równym ϵ .

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

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

W przypadku rozwiązania stochastycznego Zastosowano standardową metodę backpropagacji 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.

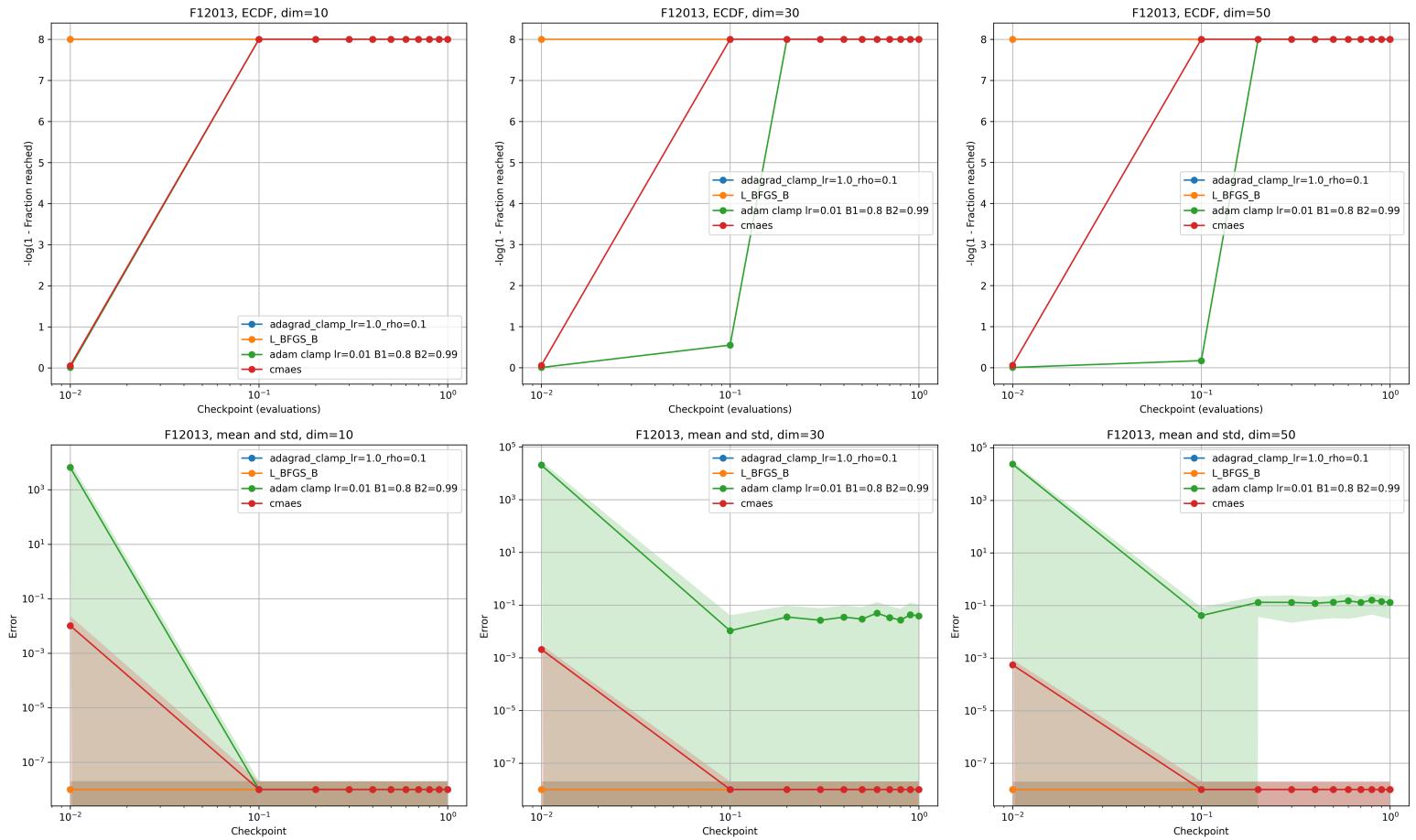
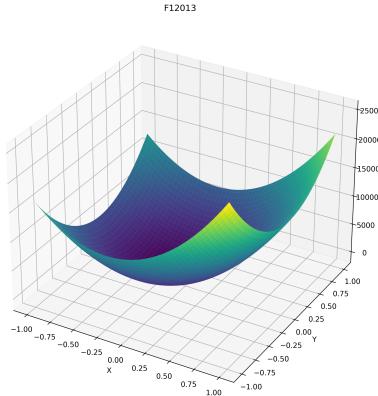
Dla każdego algorytmu zapisywano najlepsze rozwiązania po przejściu przez daną część zasobów. Rozwiązania te były następnie oceniane wykorzystując zbiór testowy, aby określić zmianę accuracy w czasie oraz ECDF

Rozdział 4

Wyniki eksperymentów

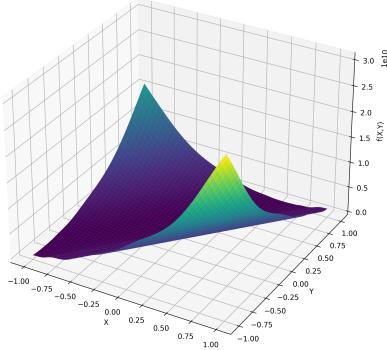
4.1 Wyniki dla benchmarków CEC

Sphere Function

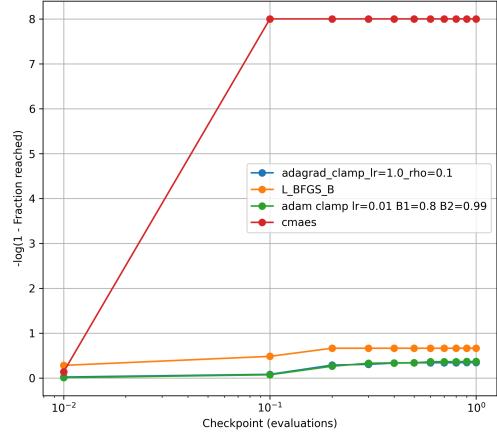


Sphere Function

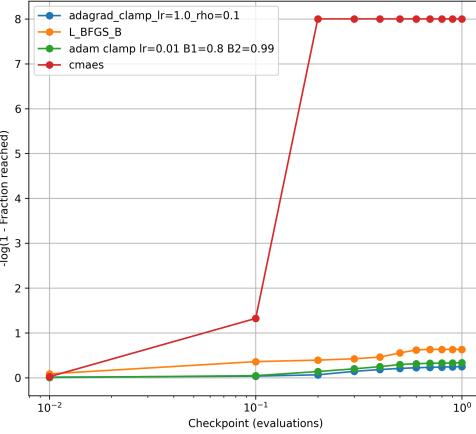
F22013



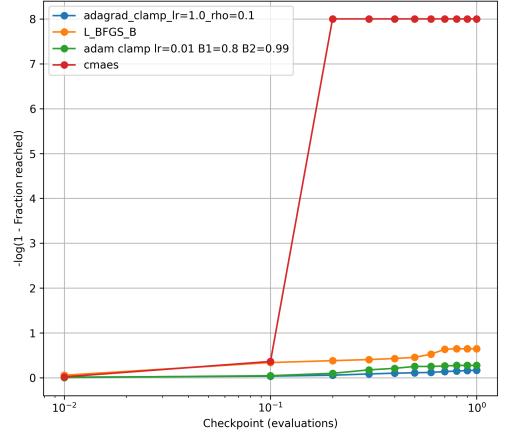
F22013, ECDF, dim=10



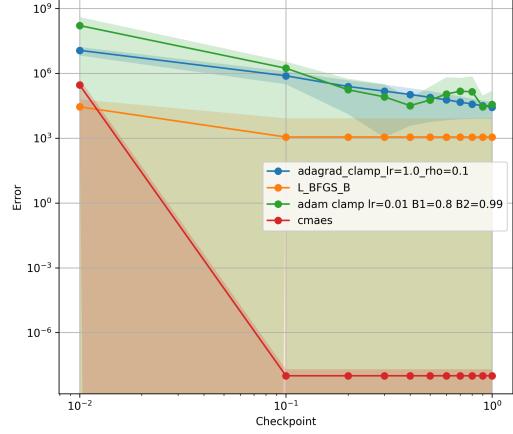
F22013, ECDF, dim=30



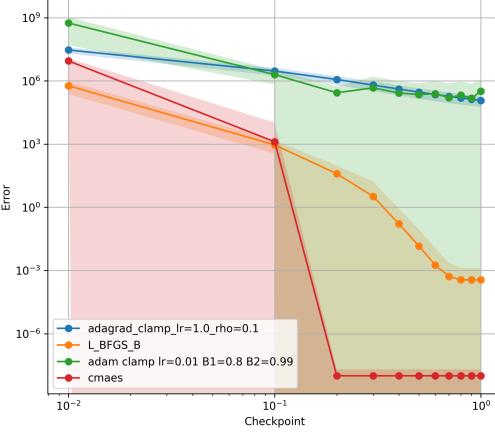
F22013, ECDF, dim=50



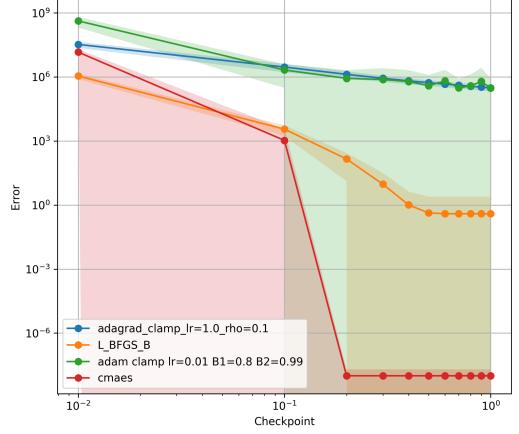
F22013, mean and std, dim=10



F22013, mean and std, dim=30

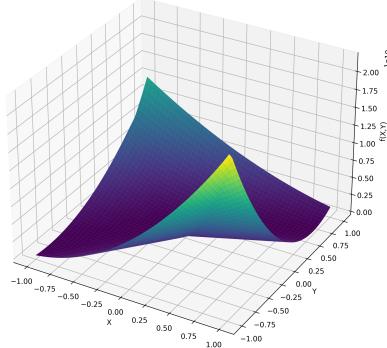


F22013, mean and std, dim=50

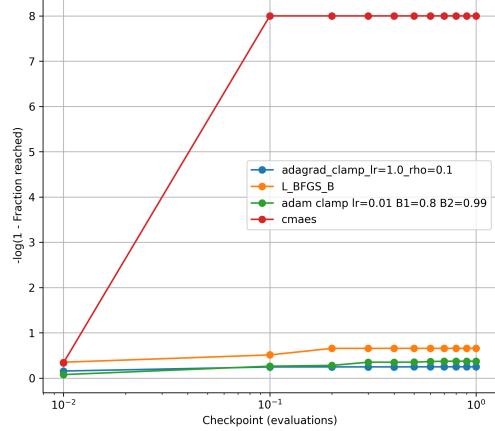


Sphere Function

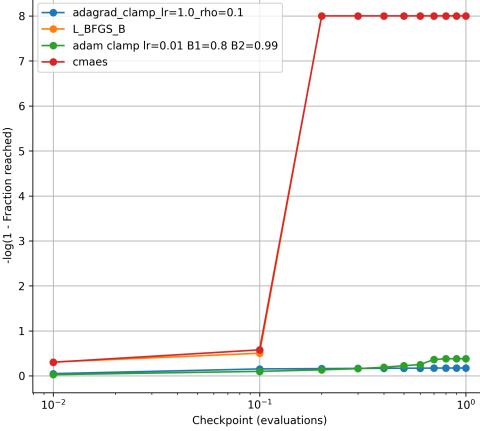
F32013



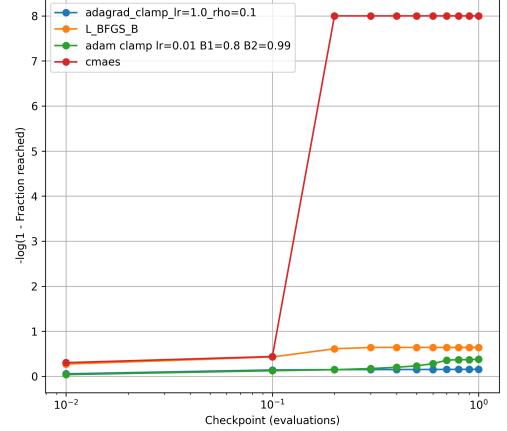
F32013, ECDF, dim=10



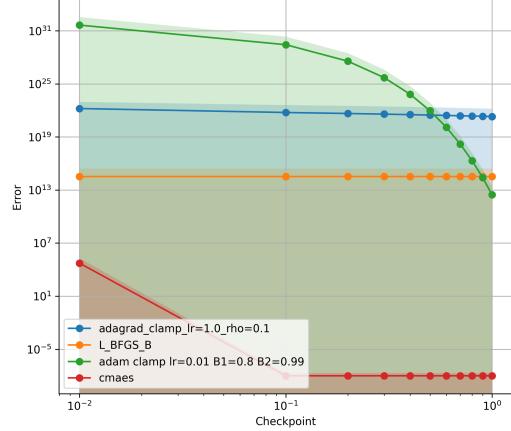
F32013, ECDF, dim=30



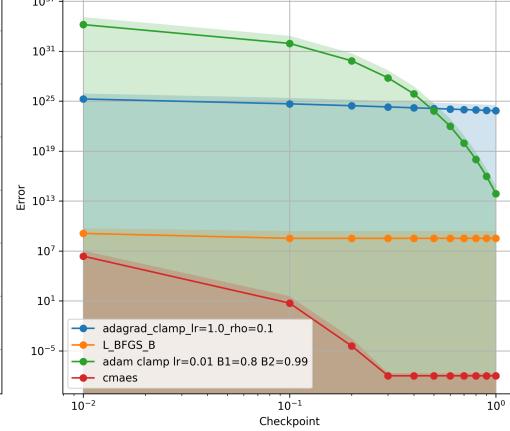
F32013, ECDF, dim=50



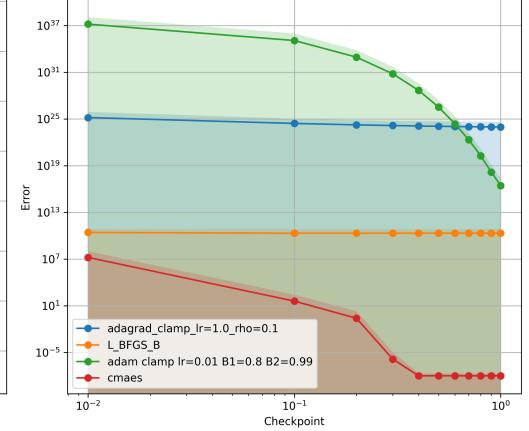
F32013, mean and std, dim=10



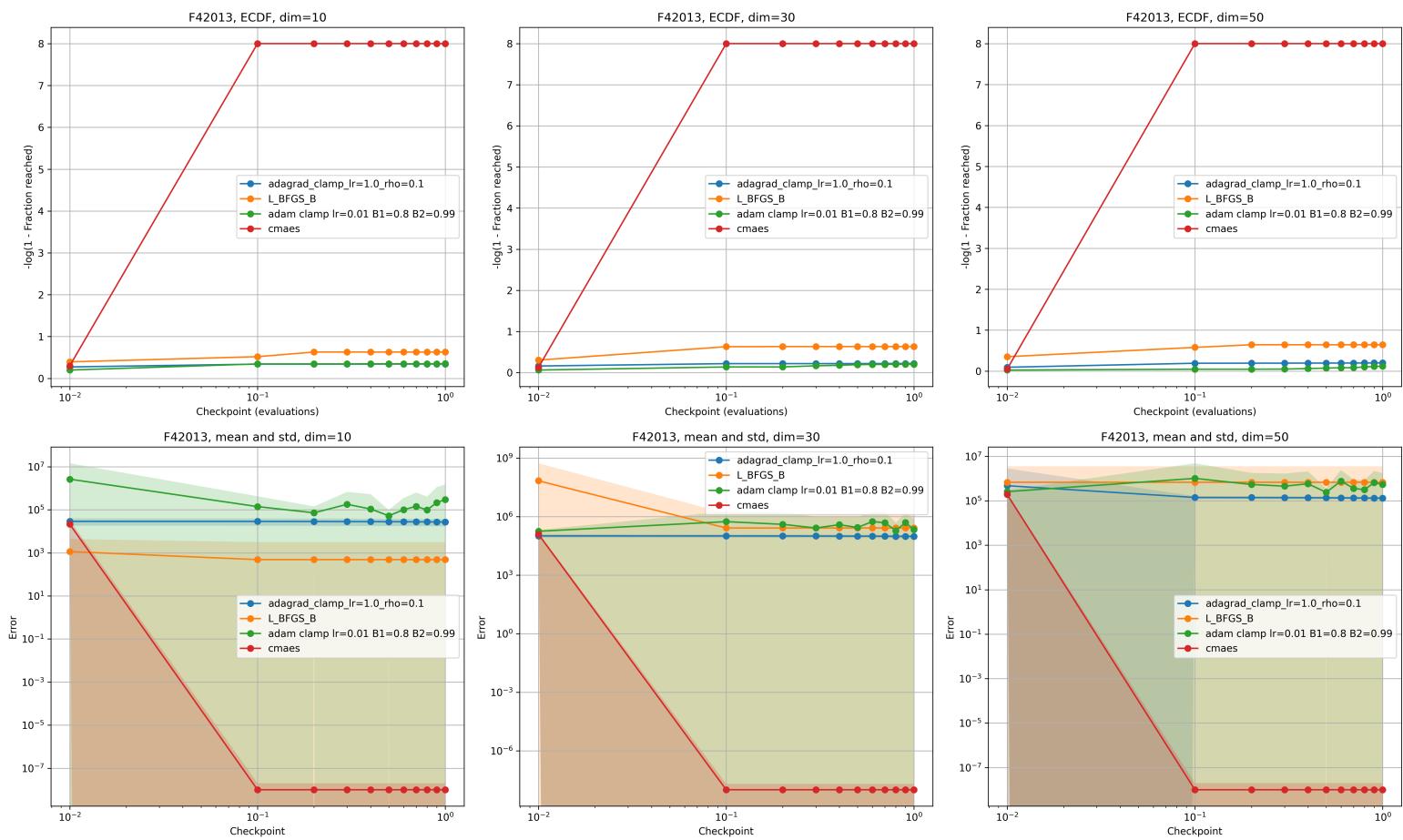
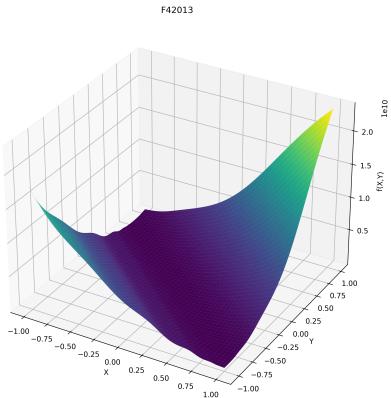
F32013, mean and std, dim=30



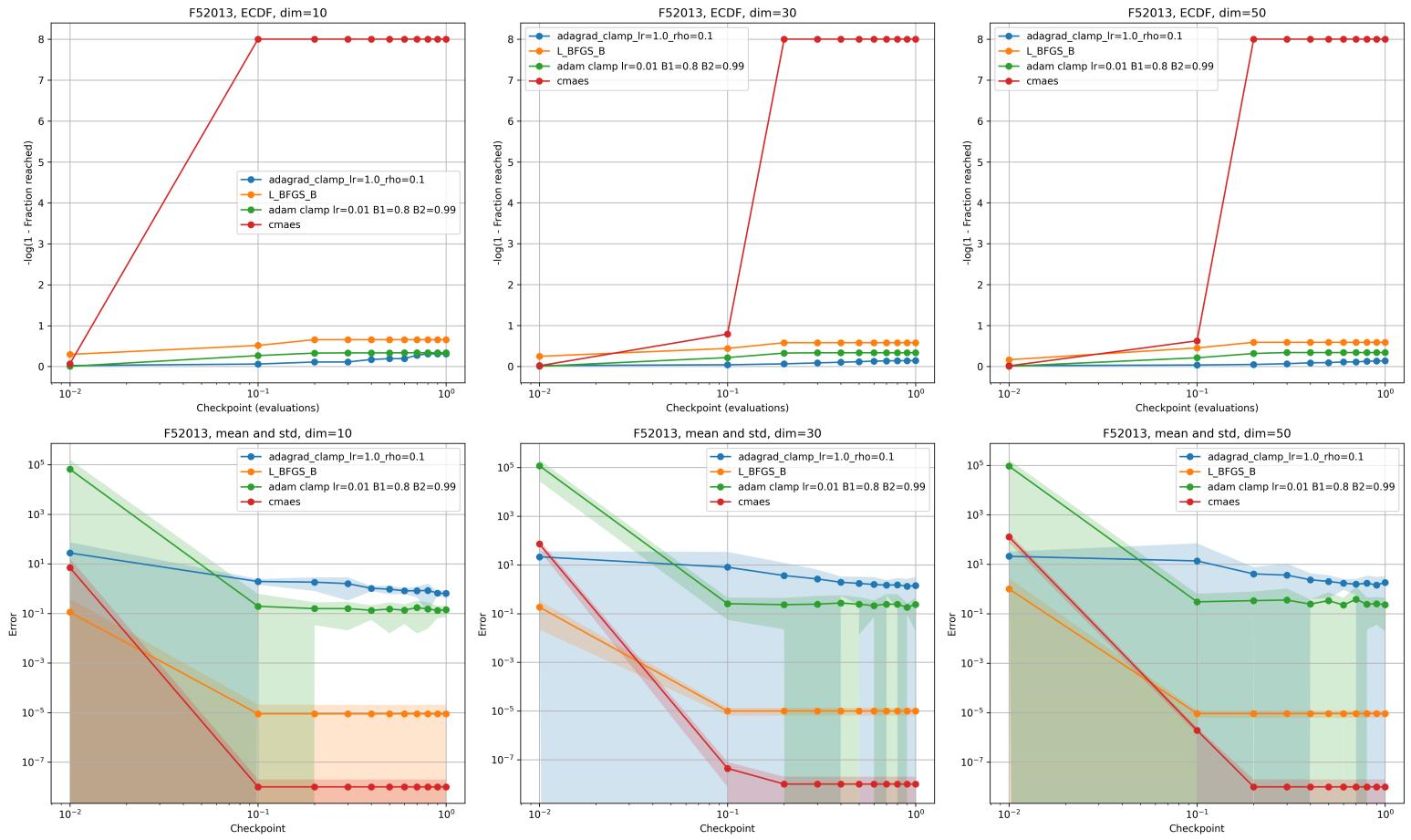
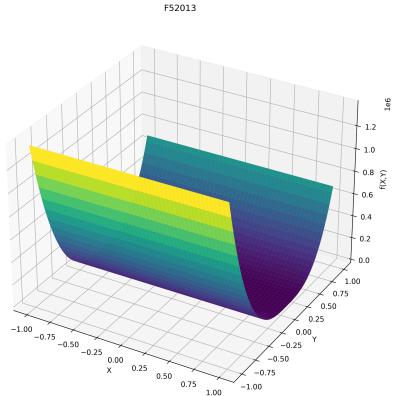
F32013, mean and std, dim=50



Sphere Function

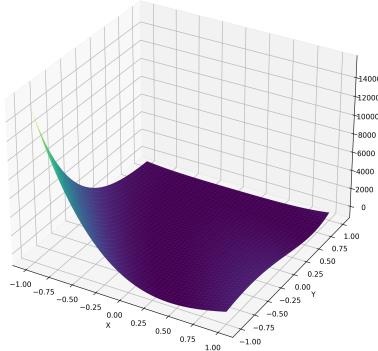


Sphere Function

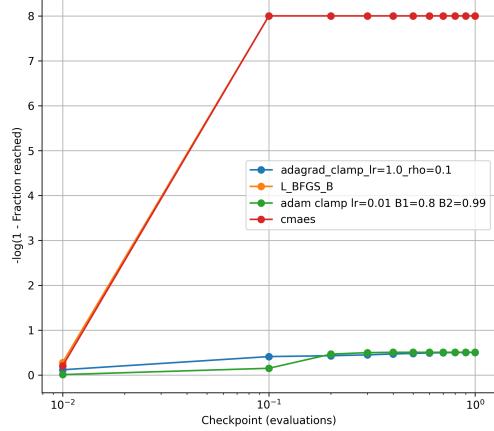


Sphere Function

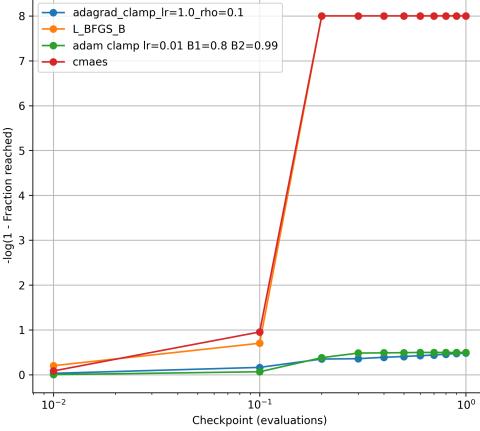
F62013



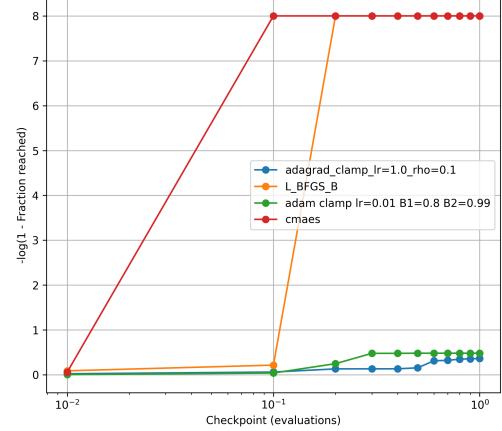
F62013, ECDF, dim=10



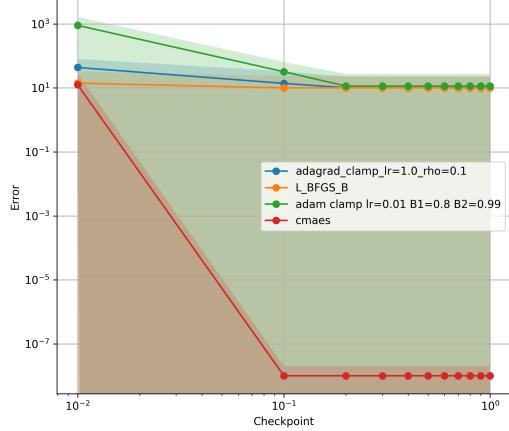
F62013, ECDF, dim=30



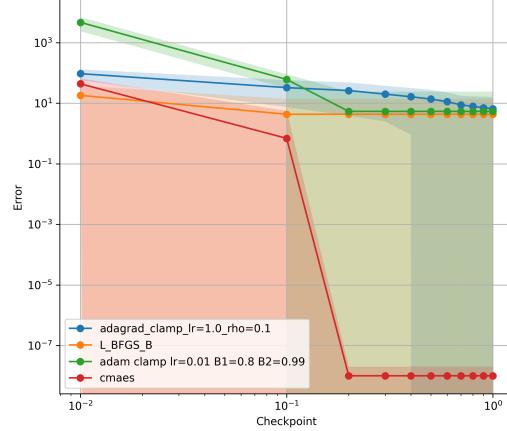
F62013, ECDF, dim=50



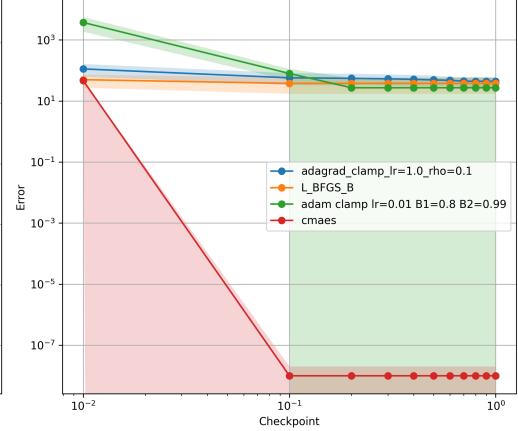
F62013, mean and std, dim=10



F62013, mean and std, dim=30

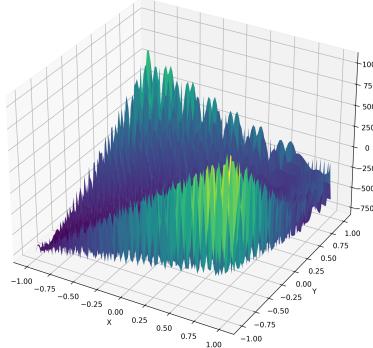


F62013, mean and std, dim=50

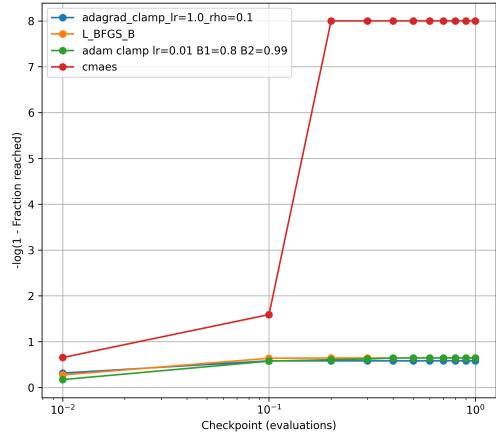


Sphere Function

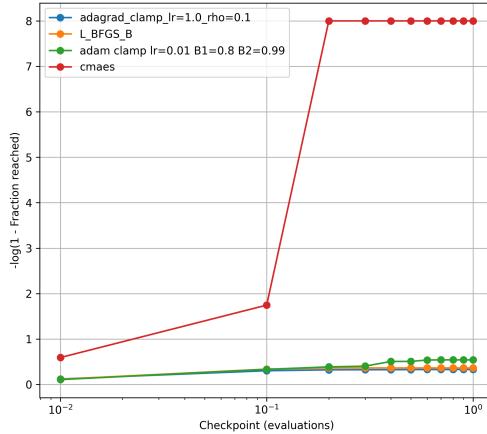
F72013



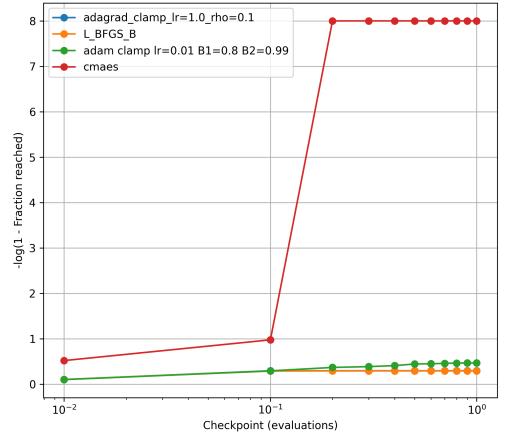
F72013, ECDF, dim=10



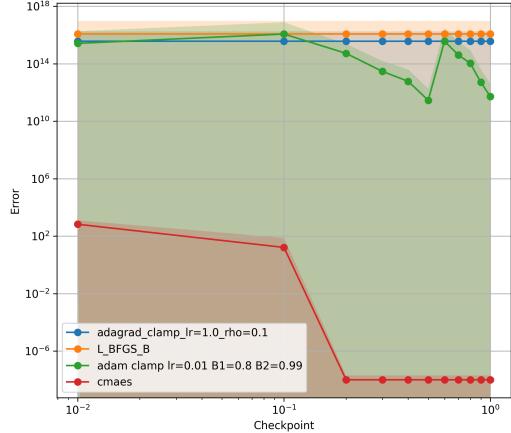
F72013, ECDF, dim=30



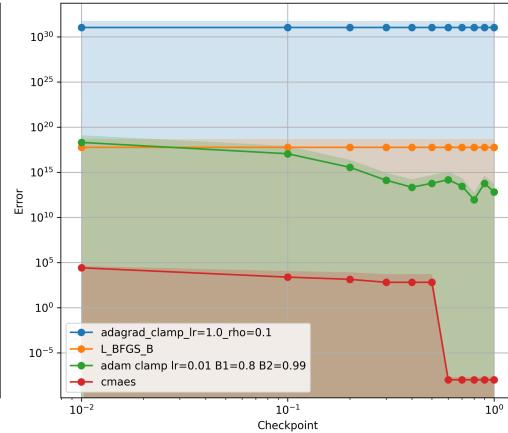
F72013, ECDF, dim=50



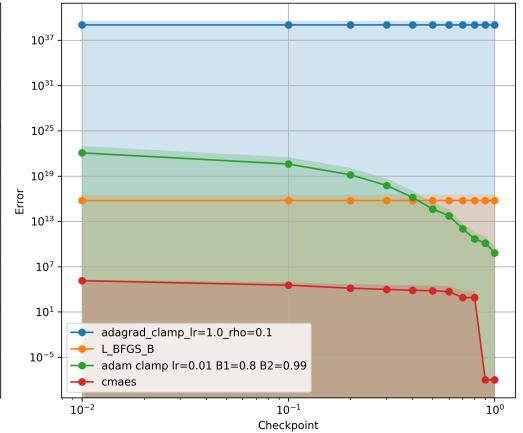
F72013, mean and std, dim=10



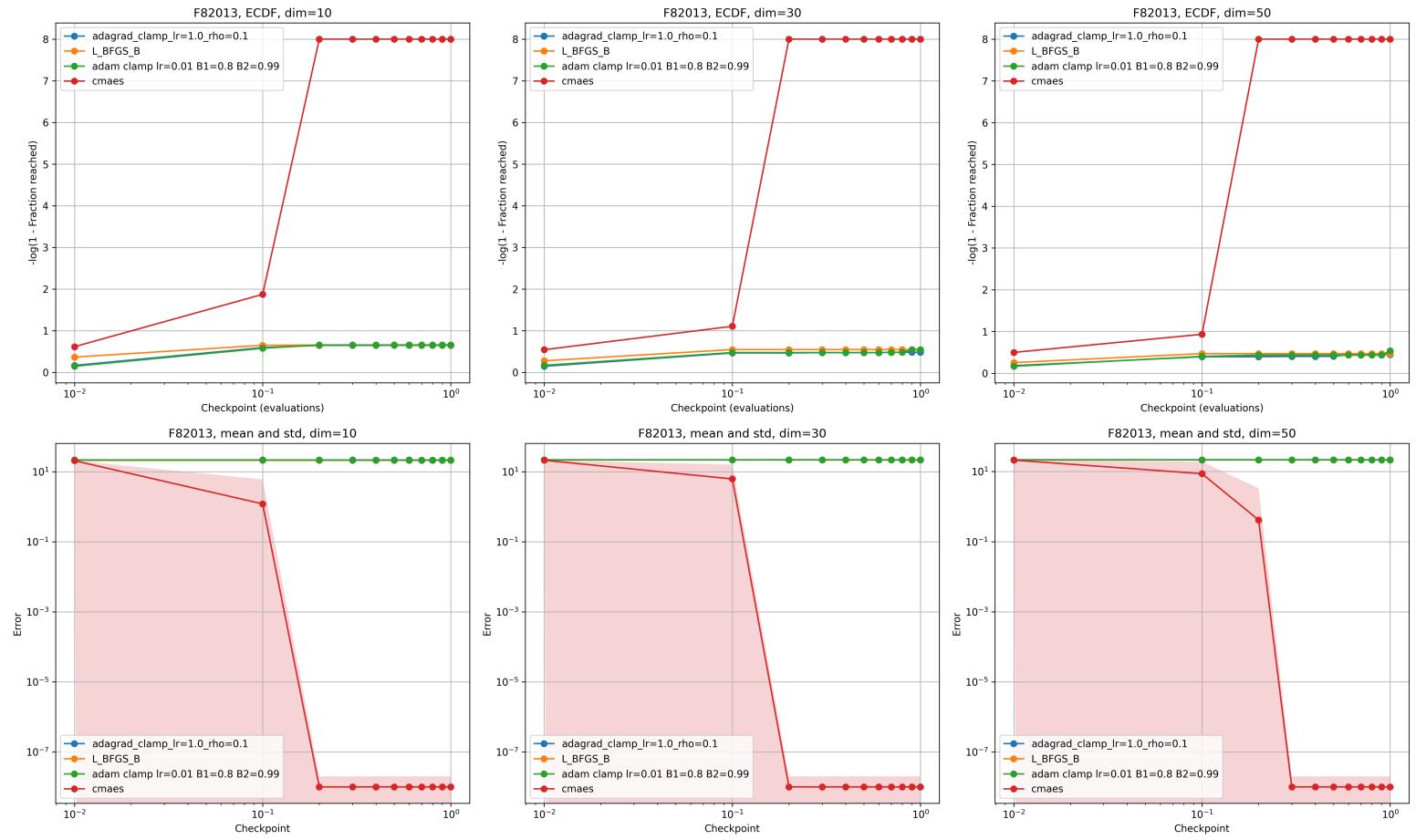
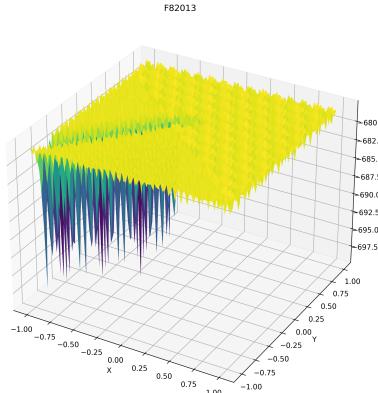
F72013, mean and std, dim=30



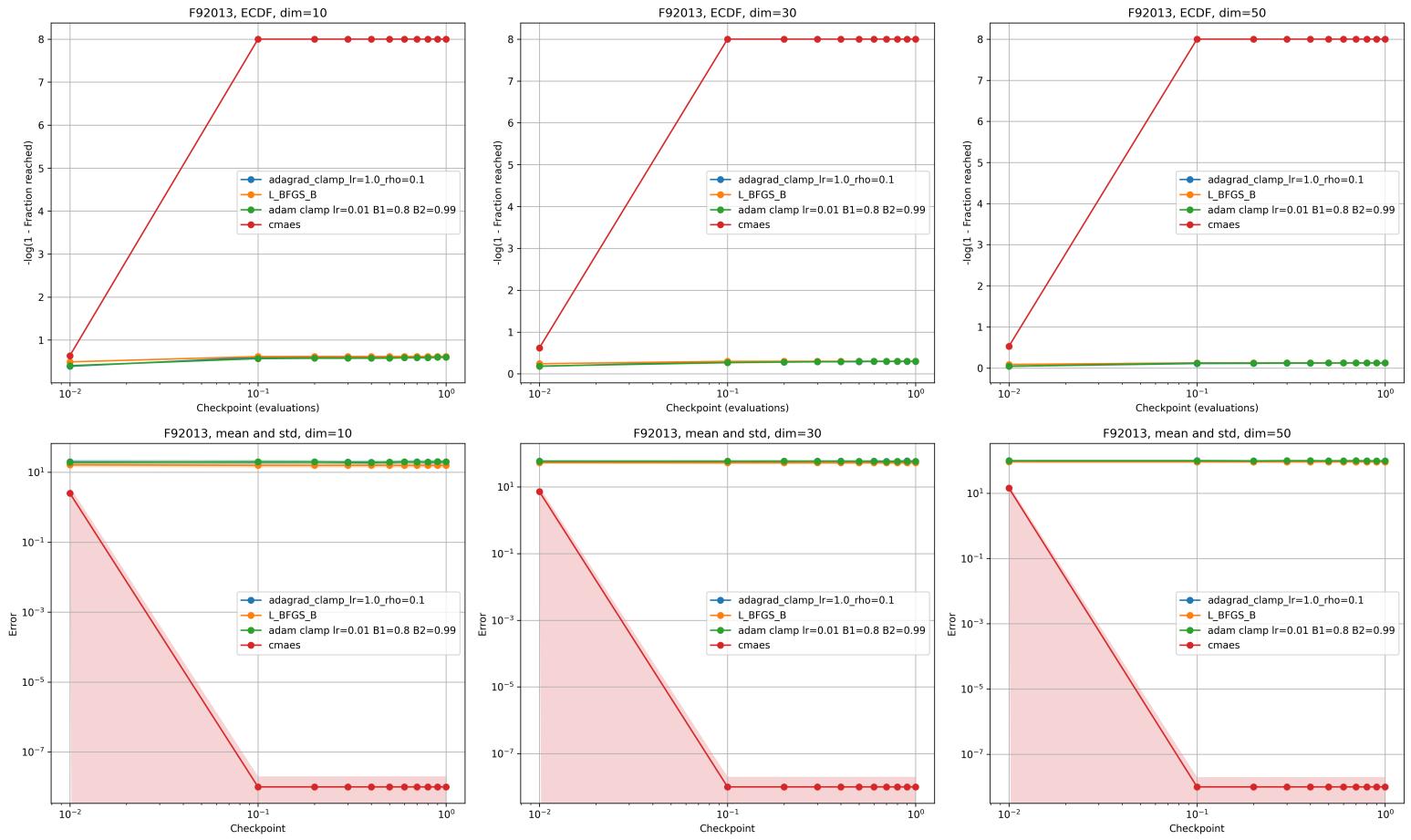
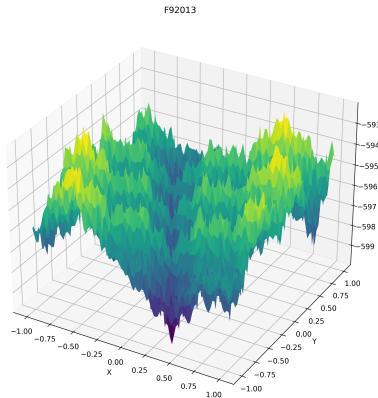
F72013, mean and std, dim=50



Sphere Function

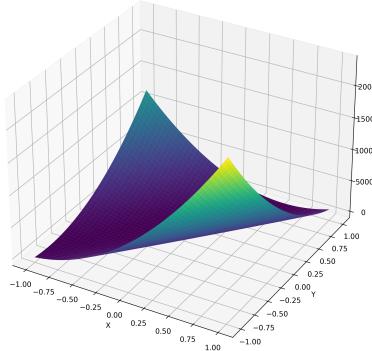


Sphere Function

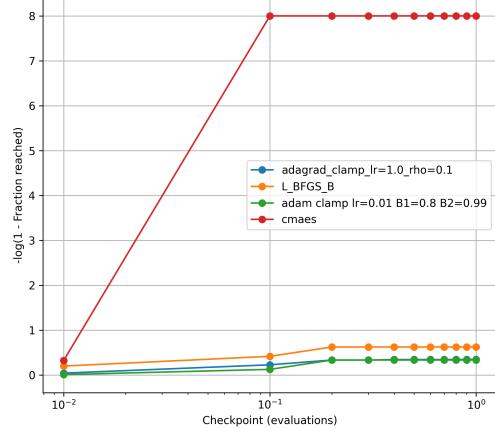


Sphere Function

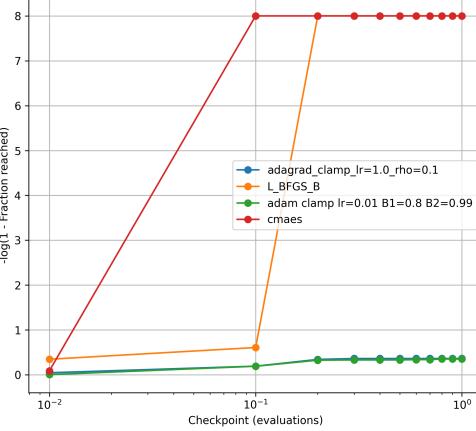
F102013



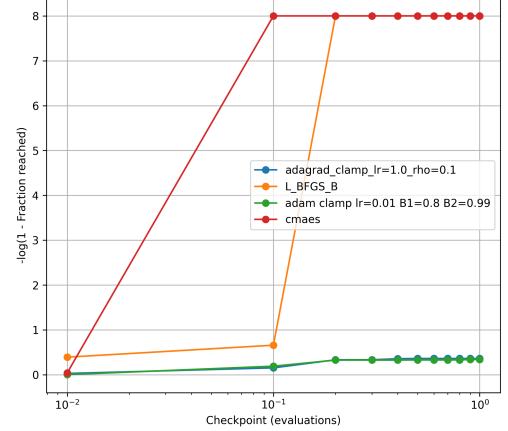
F102013, ECDF, dim=10



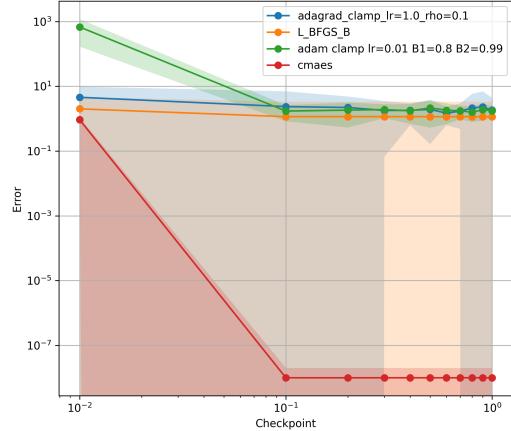
F102013, ECDF, dim=30



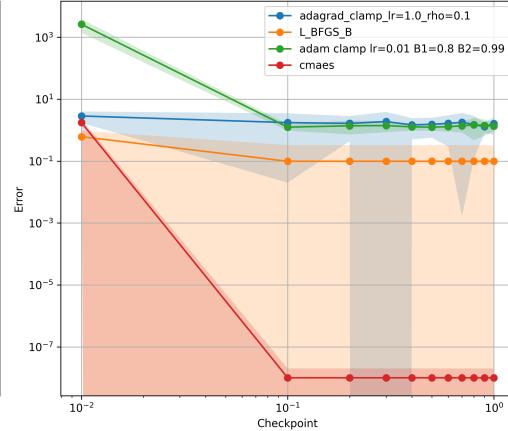
F102013, ECDF, dim=50



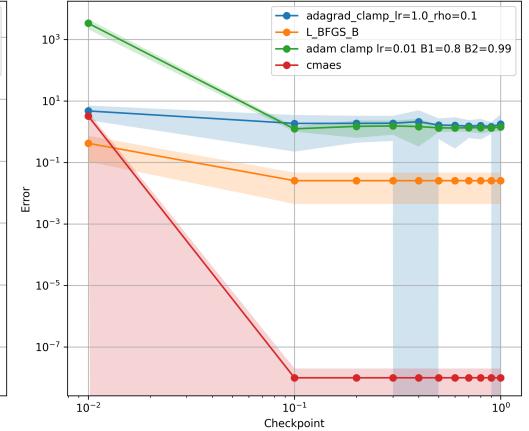
F102013, mean and std, dim=10



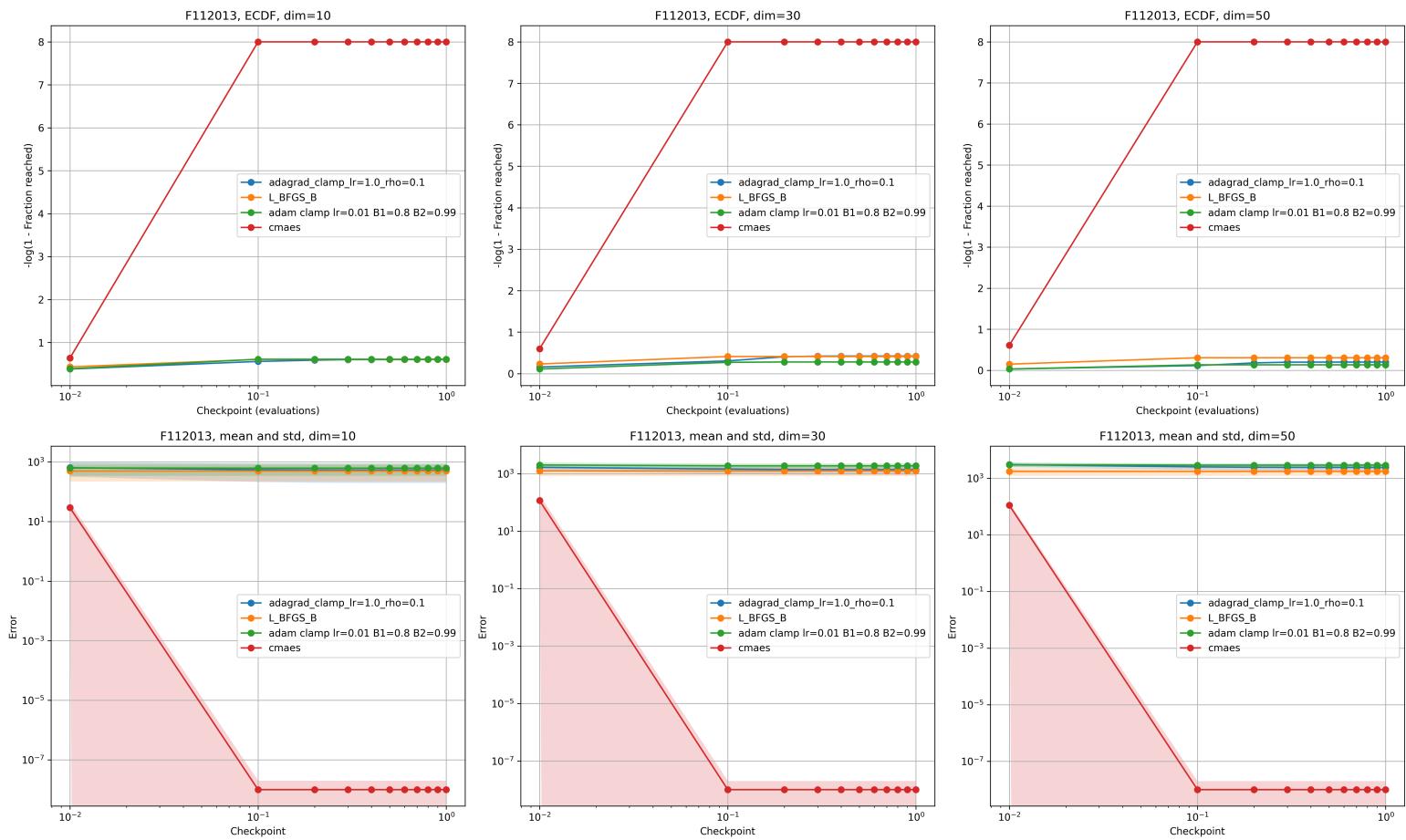
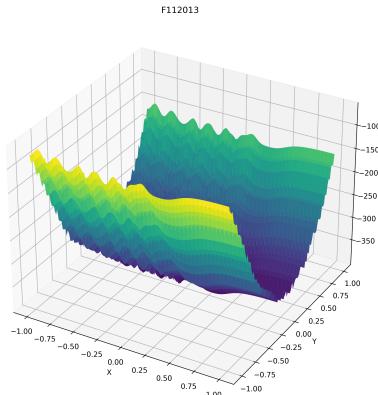
F102013, mean and std, dim=30



F102013, mean and std, dim=50

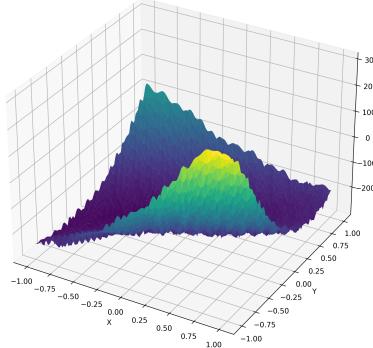


Sphere Function

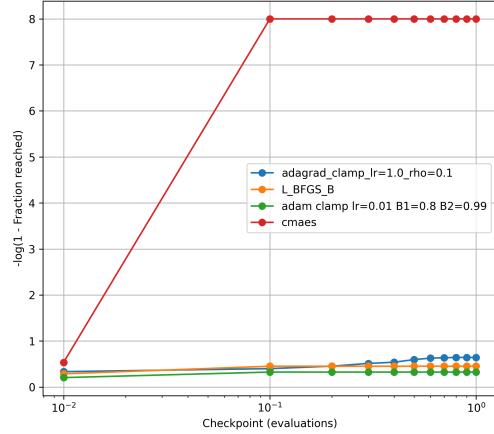


Sphere Function

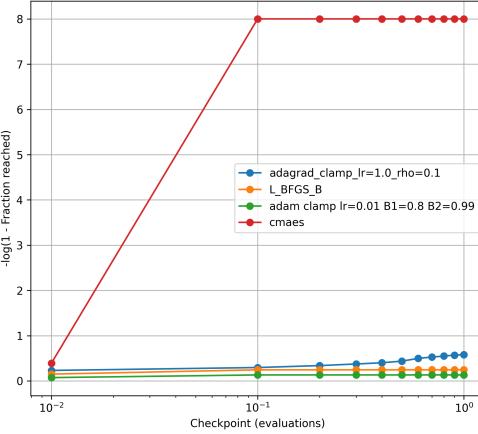
F122013



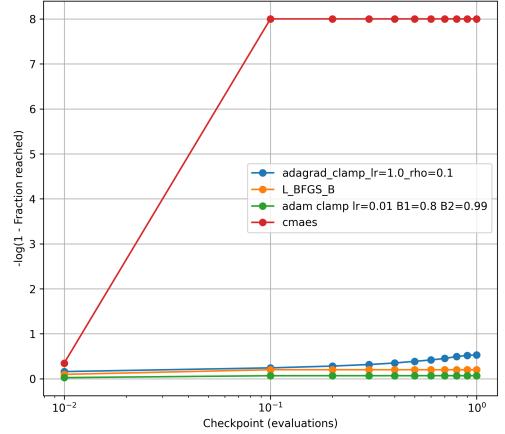
F122013, ECDF, dim=10



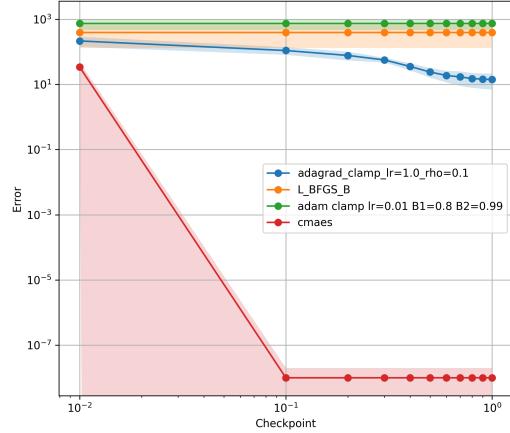
F122013, ECDF, dim=30



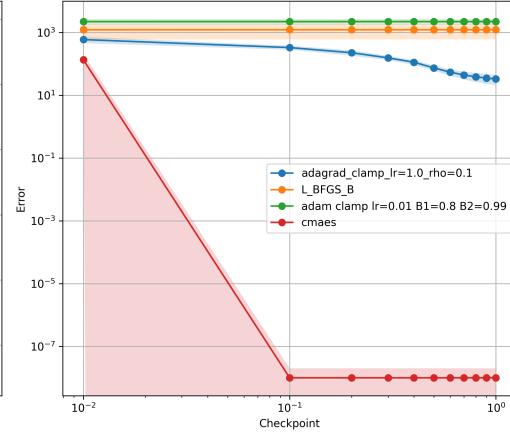
F122013, ECDF, dim=50



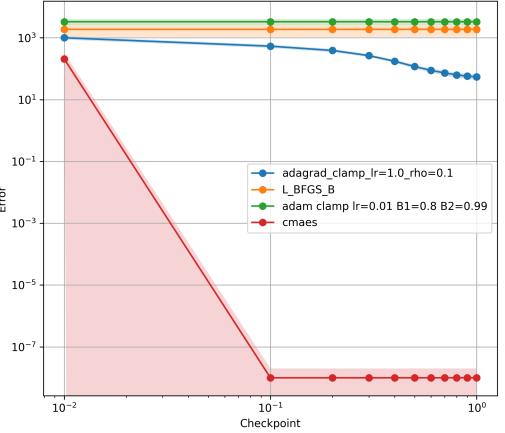
F122013, mean and std, dim=10



F122013, mean and std, dim=30

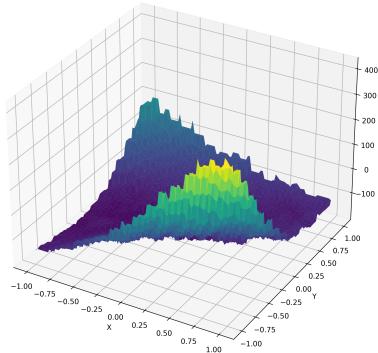


F122013, mean and std, dim=50

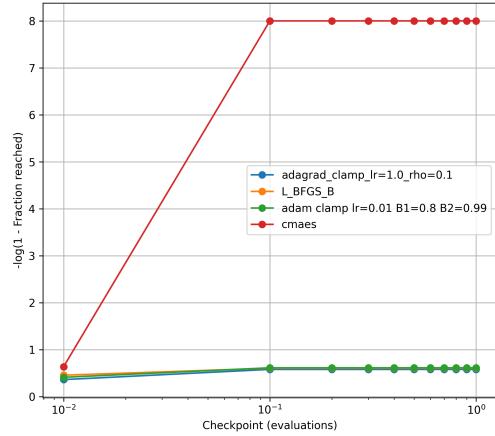


Sphere Function

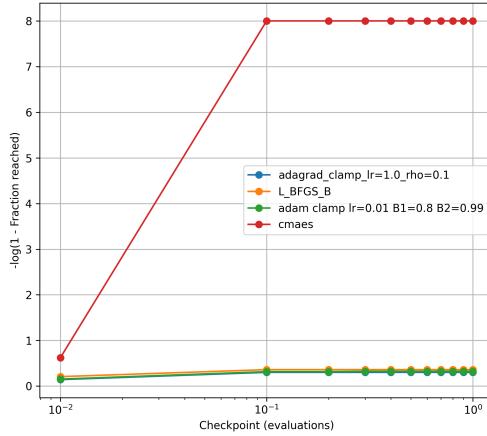
F132013



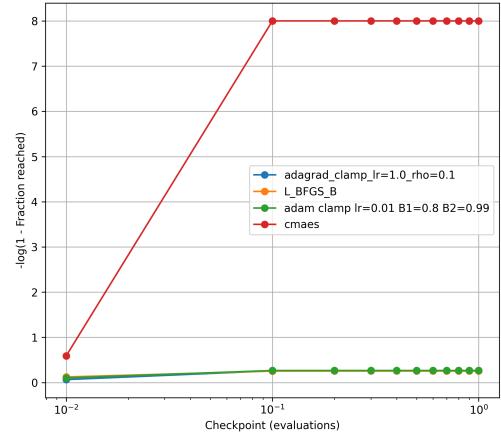
F132013, ECDF, dim=10



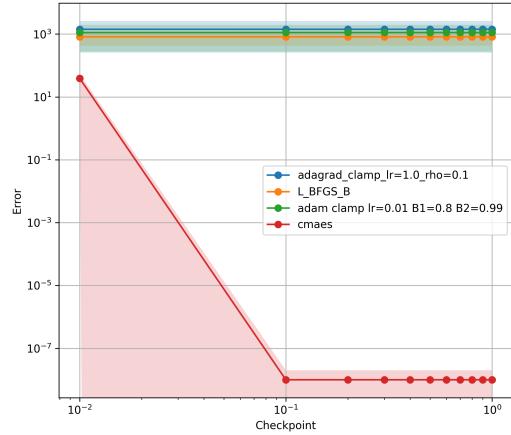
F132013, ECDF, dim=30



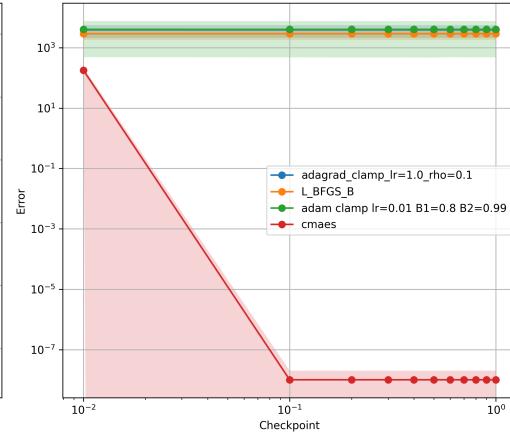
F132013, ECDF, dim=50



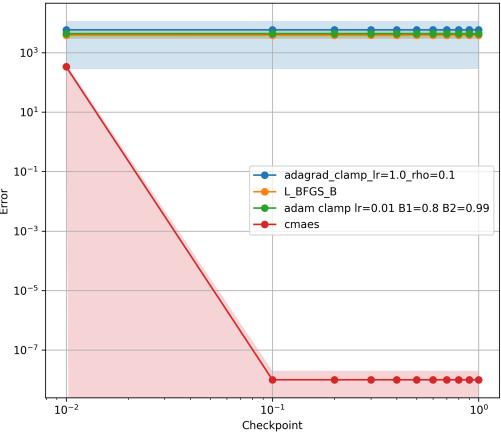
F132013, mean and std, dim=10



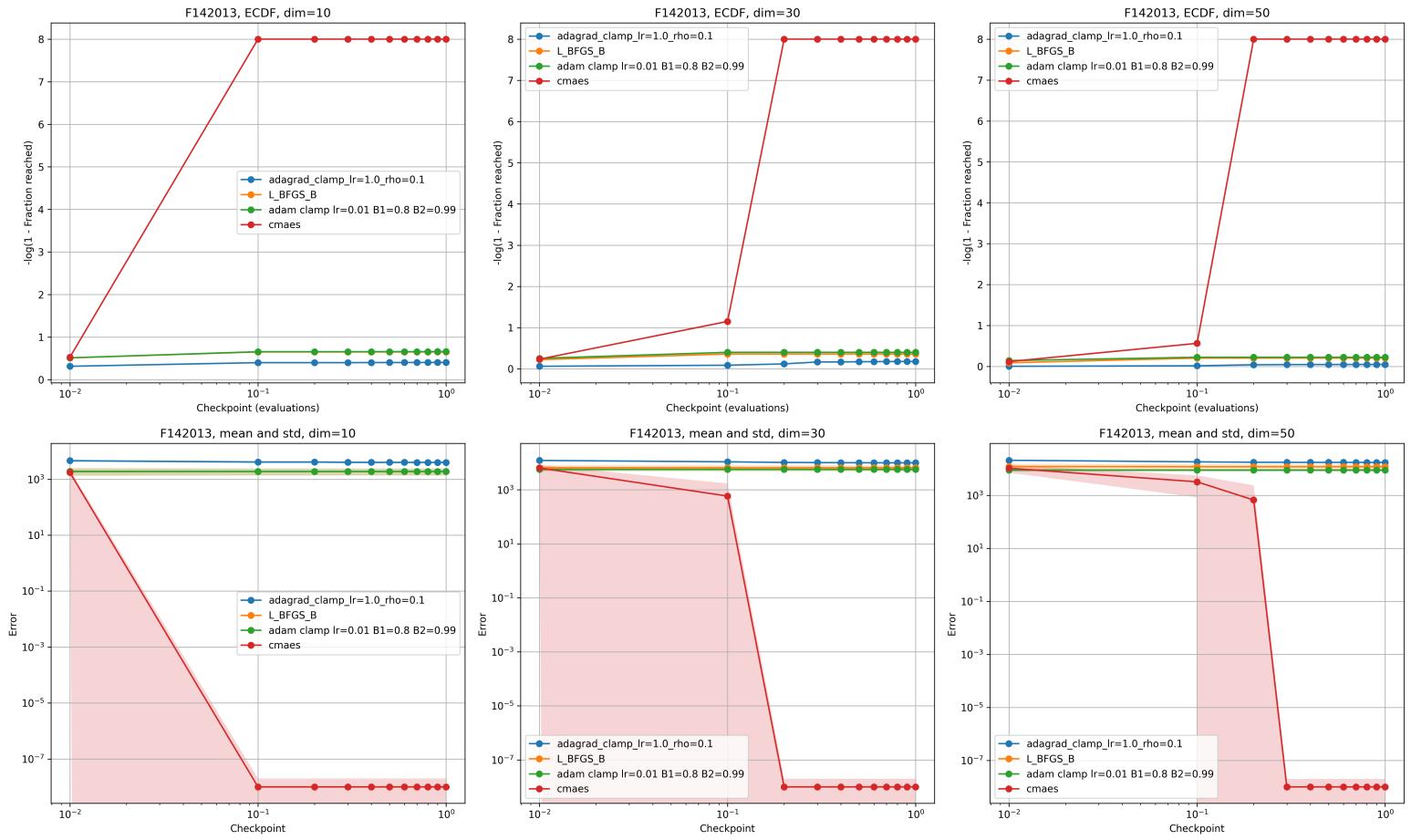
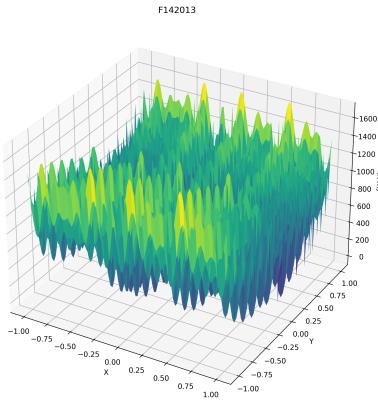
F132013, mean and std, dim=30



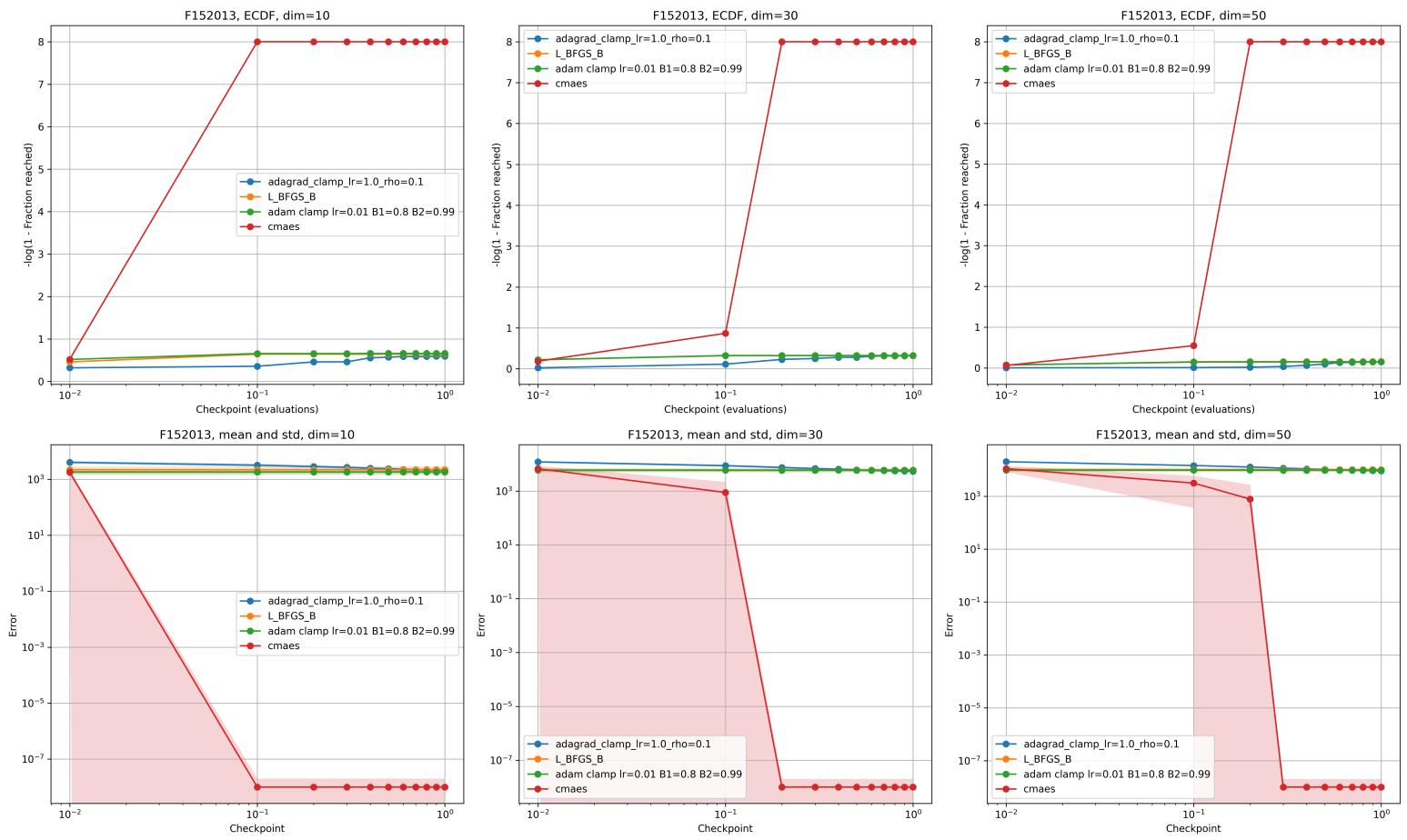
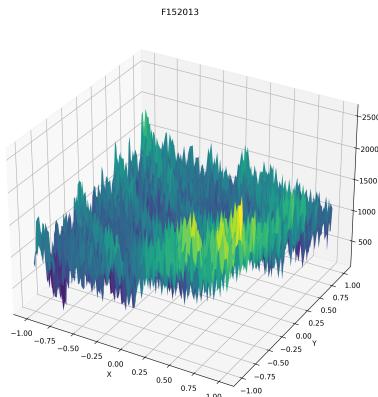
F132013, mean and std, dim=50



Sphere Function

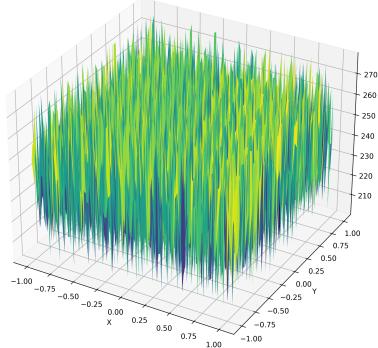


Sphere Function

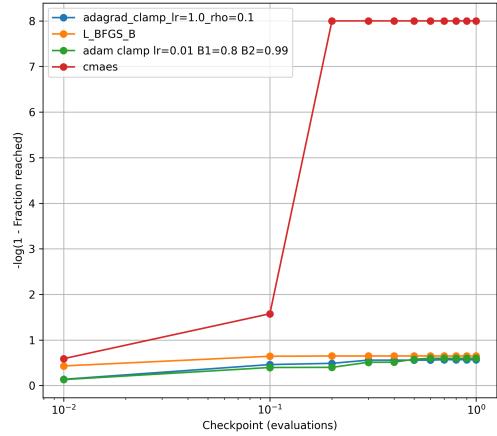


Sphere Function

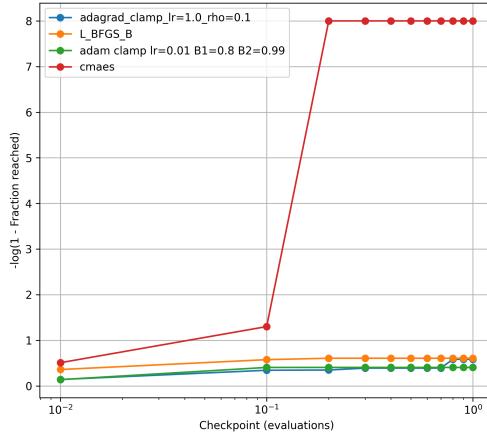
F162013



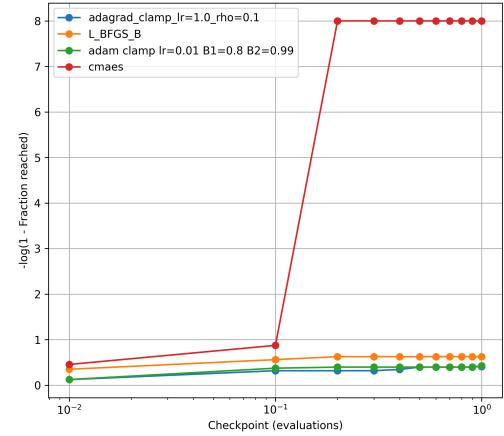
F162013, ECDF, dim=10



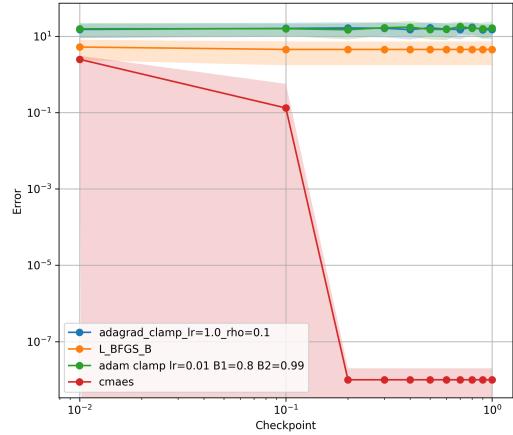
F162013, ECDF, dim=30



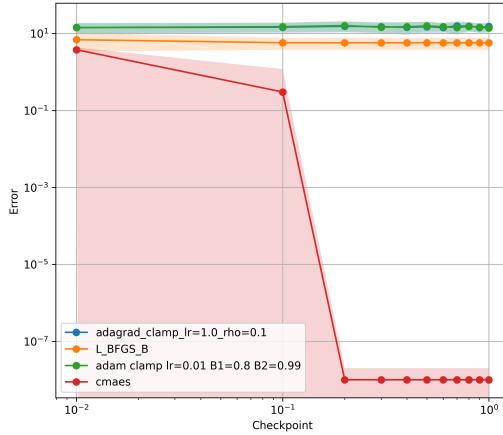
F162013, ECDF, dim=50



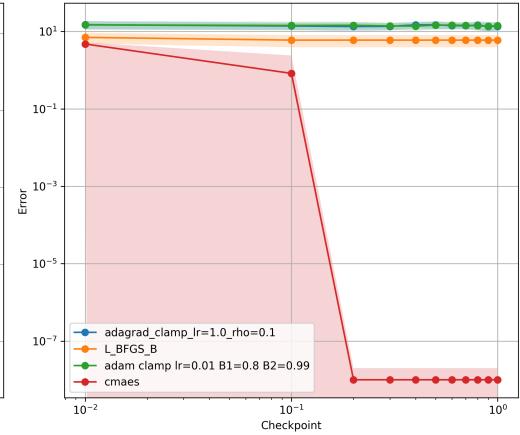
F162013, mean and std, dim=10



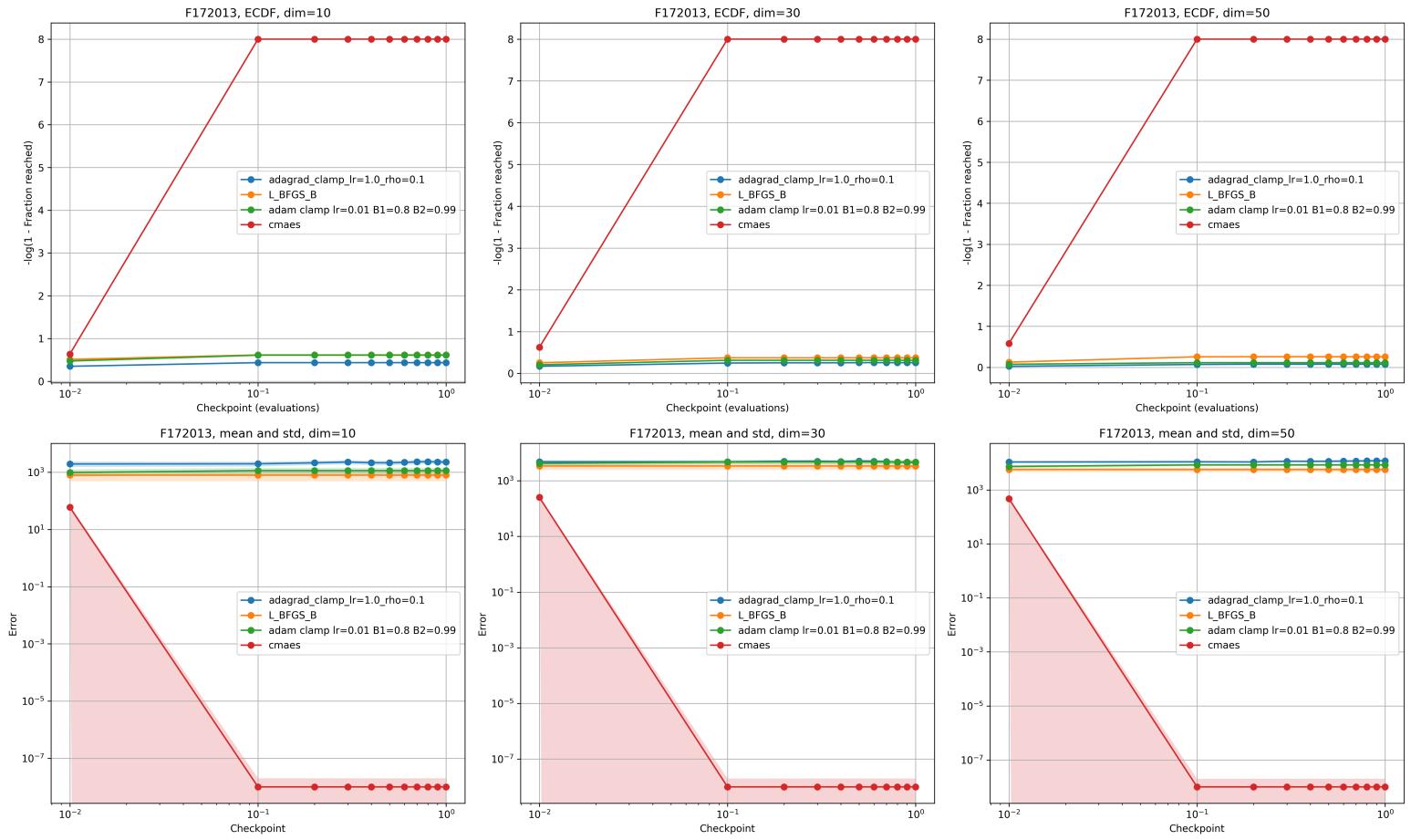
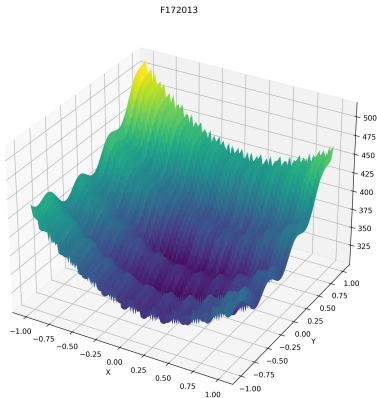
F162013, mean and std, dim=30



F162013, mean and std, dim=50

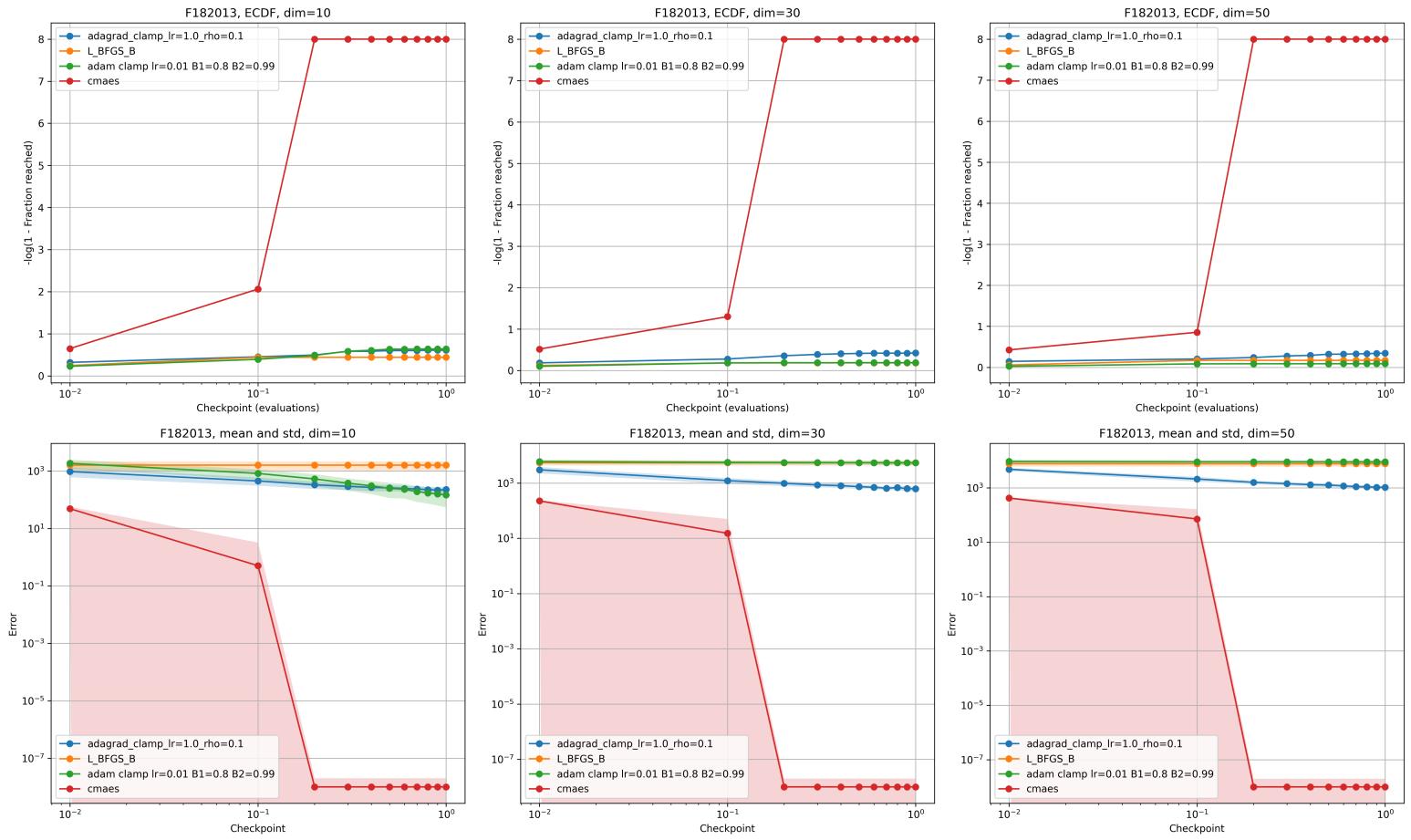
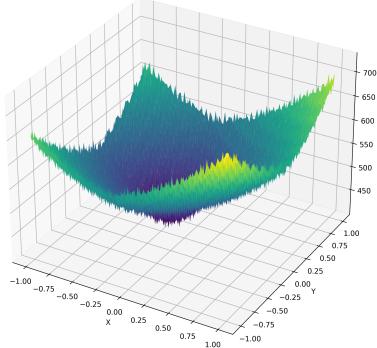


Sphere Function



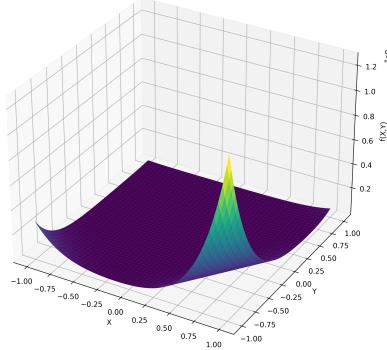
Sphere Function

F182013

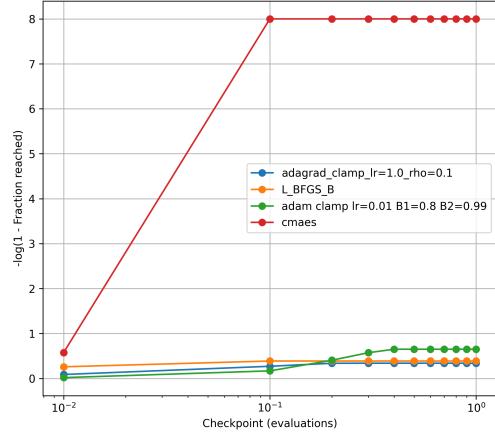


Sphere Function

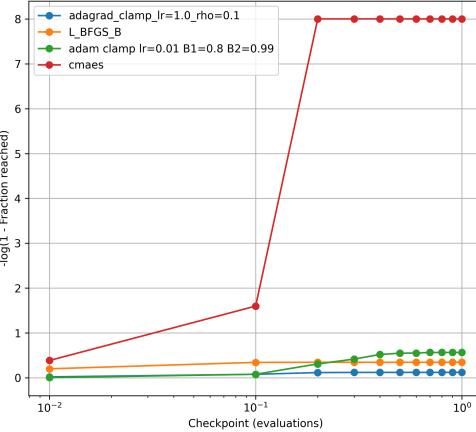
F192013



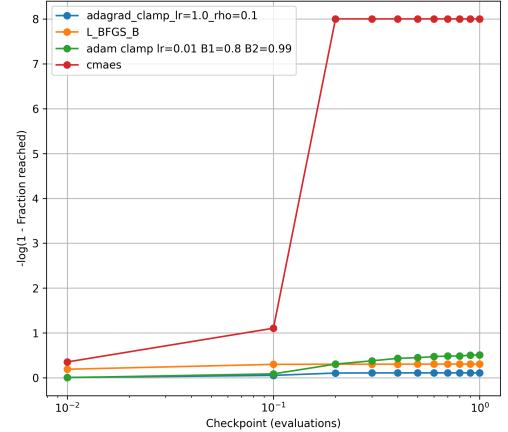
F192013, ECDF, dim=10



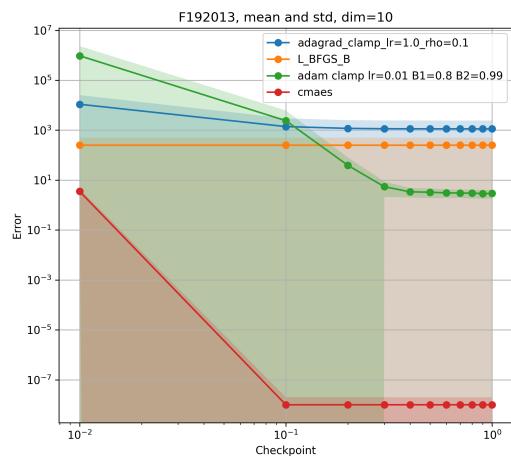
F192013, ECDF, dim=30



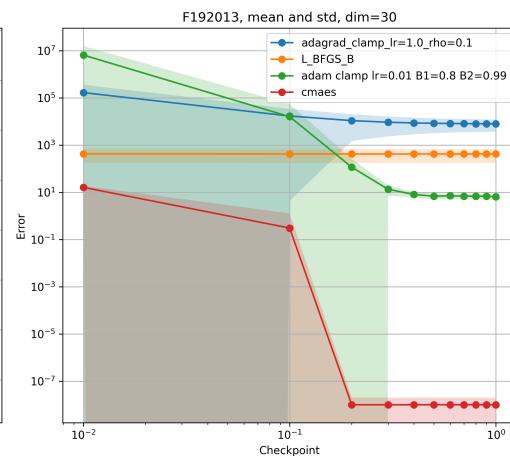
F192013, ECDF, dim=50



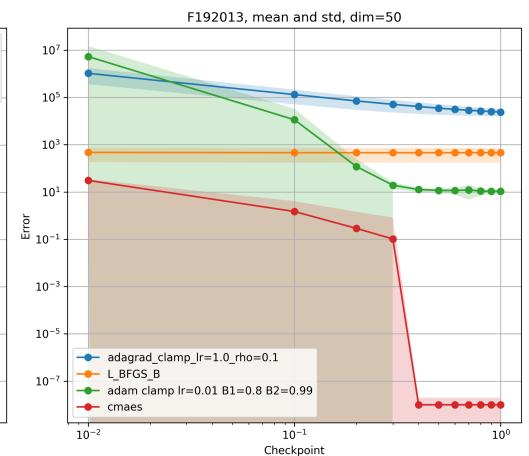
F192013, mean and std, dim=10



F192013, mean and std, dim=30

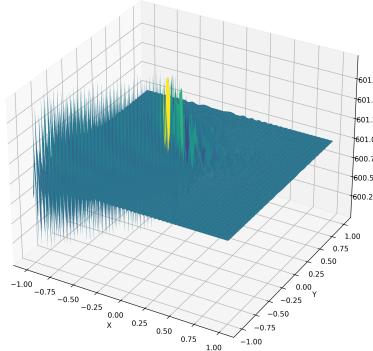


F192013, mean and std, dim=50

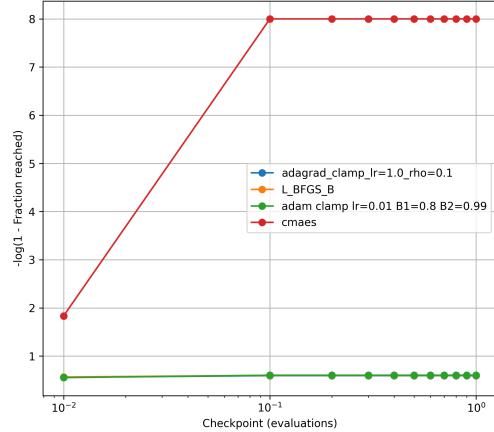


Sphere Function

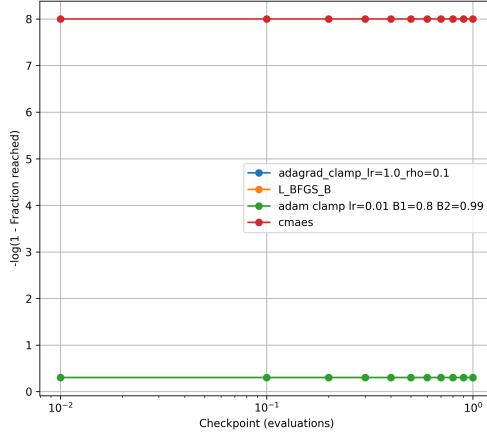
F202013



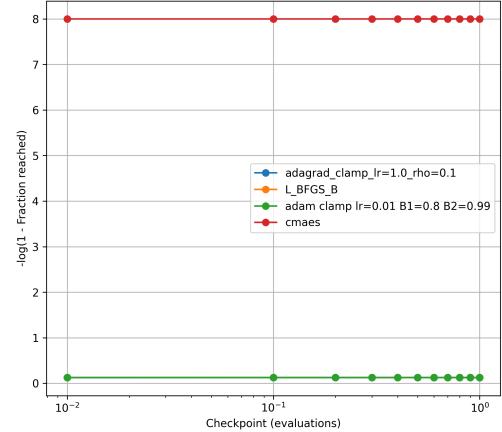
F202013, ECDF, dim=10



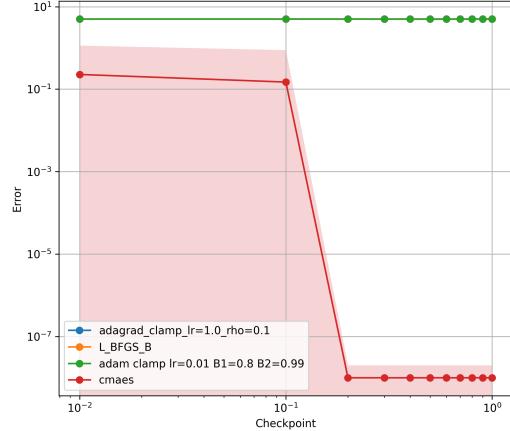
F202013, ECDF, dim=30



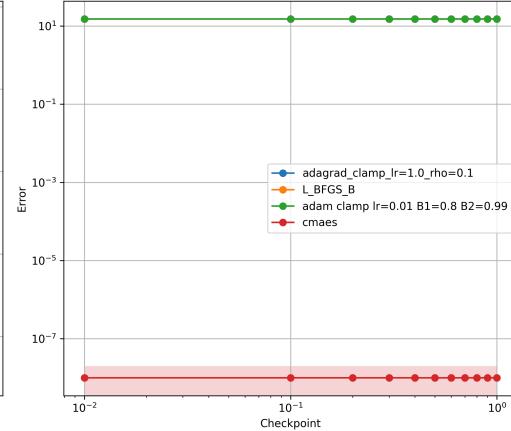
F202013, ECDF, dim=50



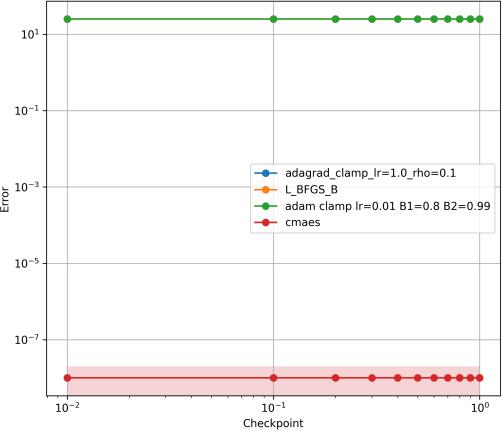
F202013, mean and std, dim=10



F202013, mean and std, dim=30

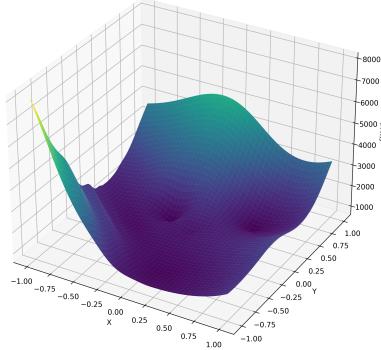


F202013, mean and std, dim=50

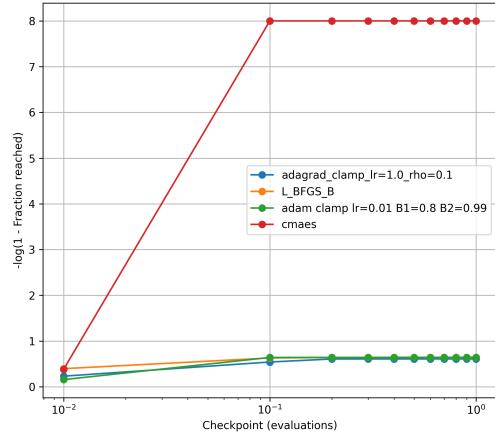


Sphere Function

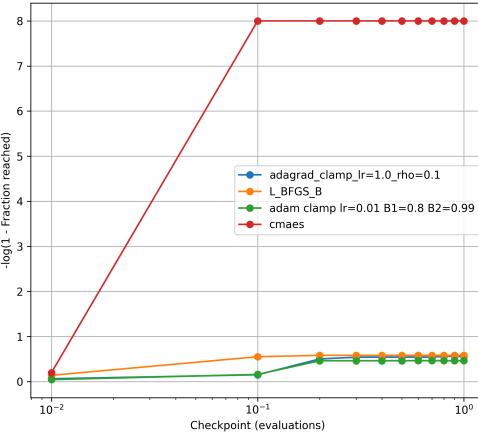
F212013



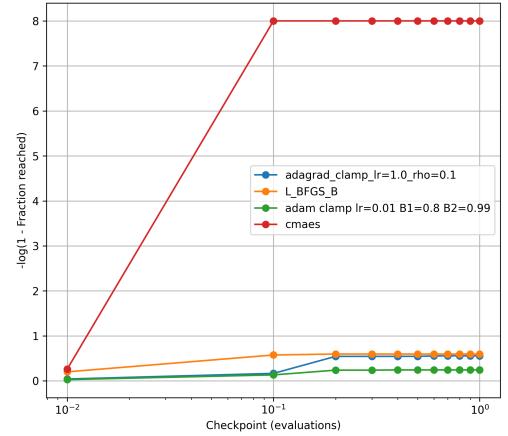
F212013, ECDF, dim=10



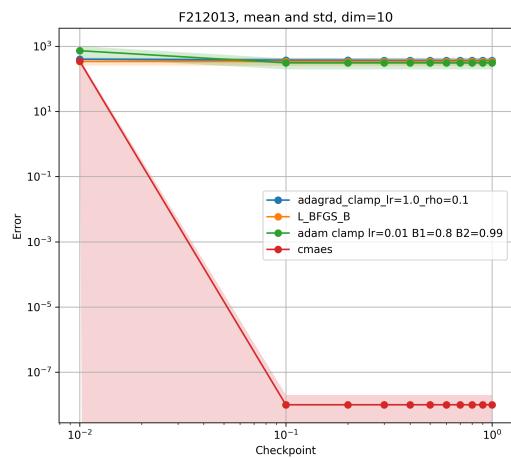
F212013, ECDF, dim=30



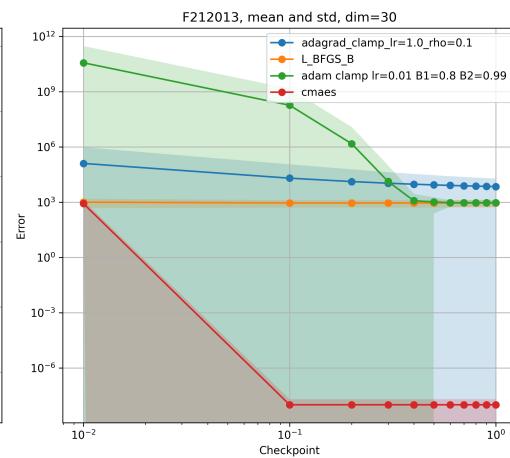
F212013, ECDF, dim=50



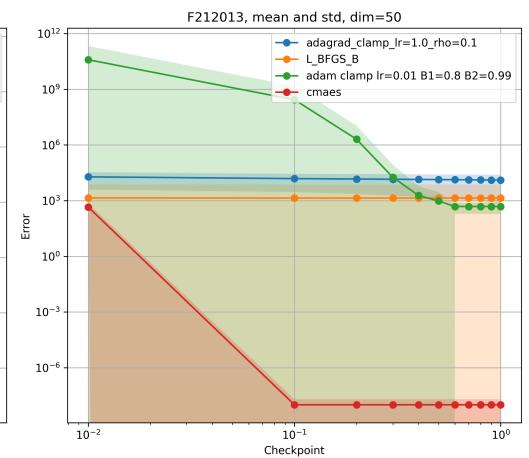
F212013, mean and std, dim=10



F212013, mean and std, dim=30

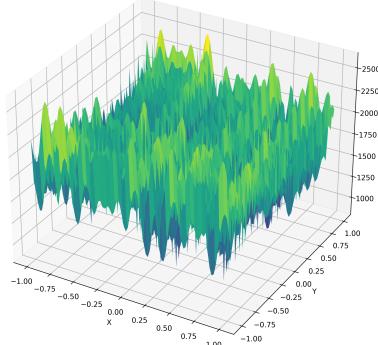


F212013, mean and std, dim=50

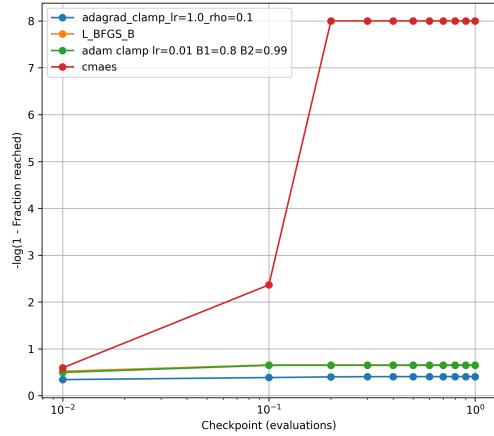


Sphere Function

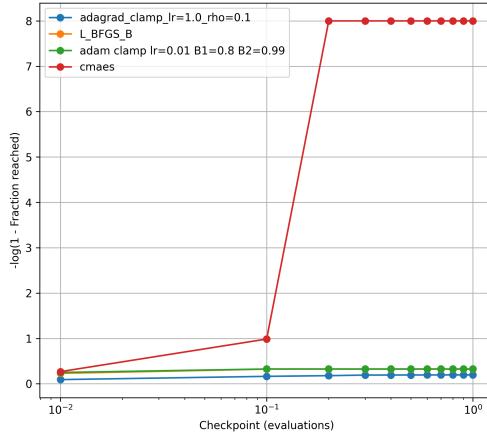
F222013



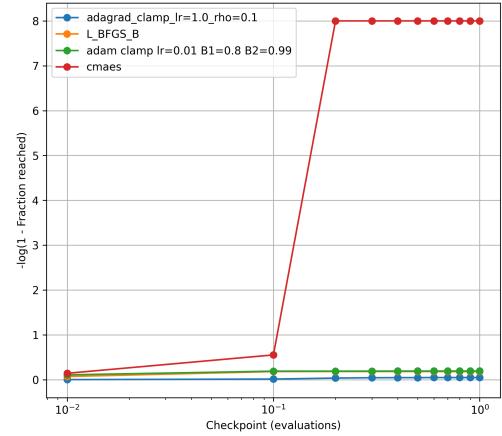
F222013, ECDF, dim=10



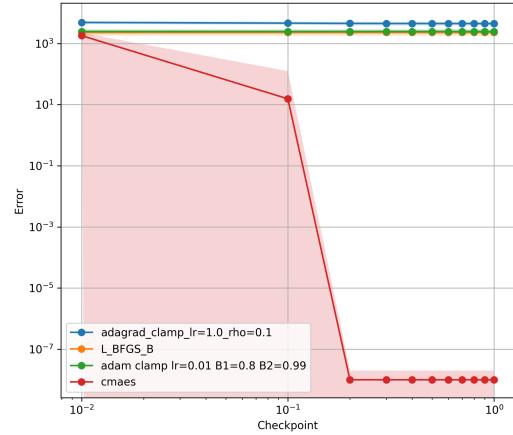
F222013, ECDF, dim=30



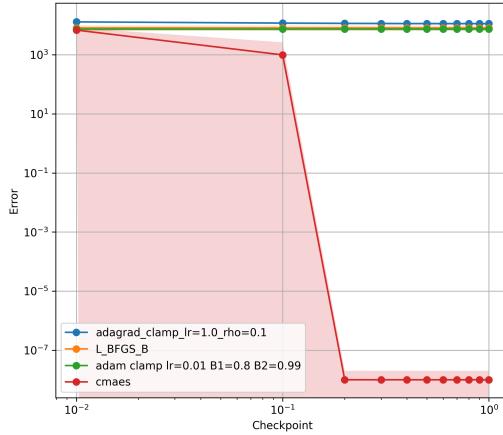
F222013, ECDF, dim=50



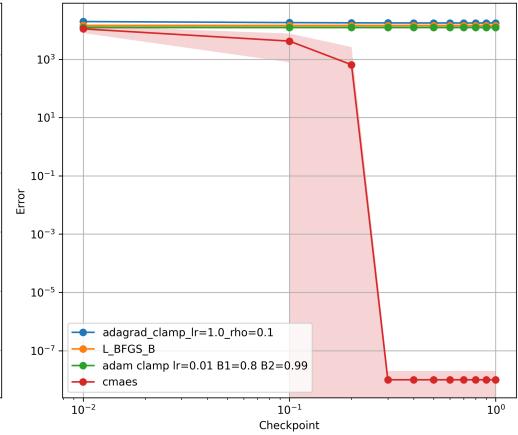
F222013, mean and std, dim=10



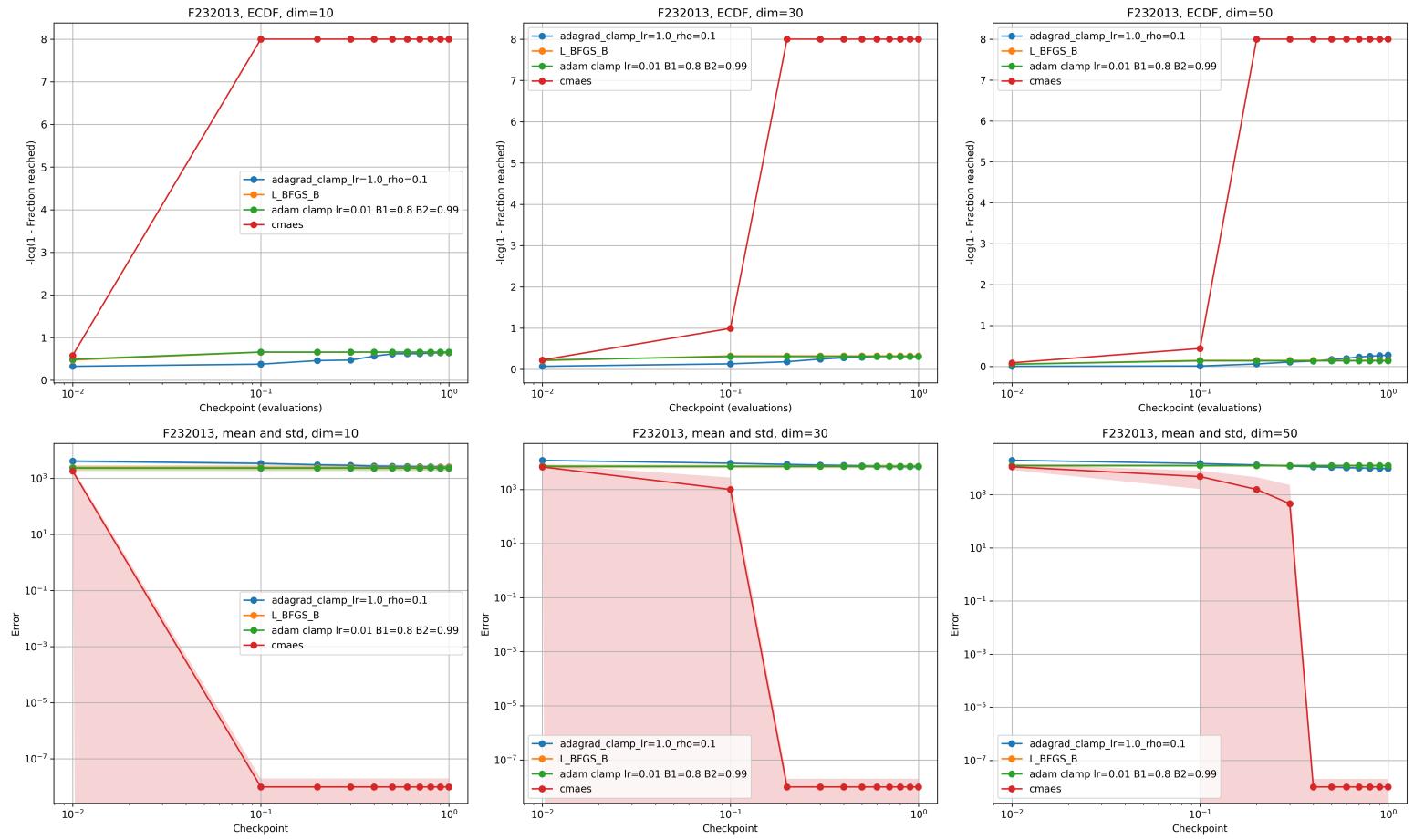
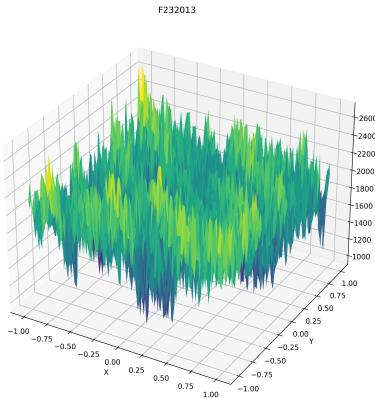
F222013, mean and std, dim=30



F222013, mean and std, dim=50

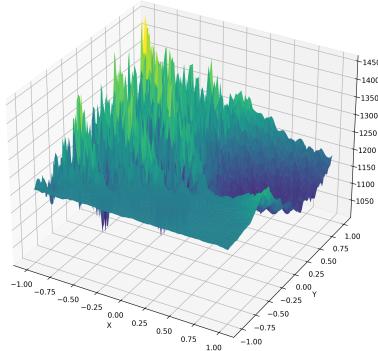


Sphere Function

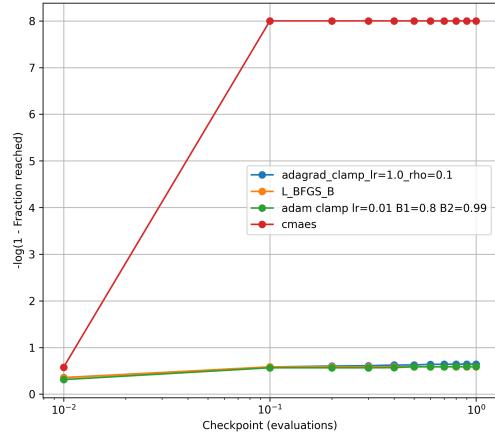


Sphere Function

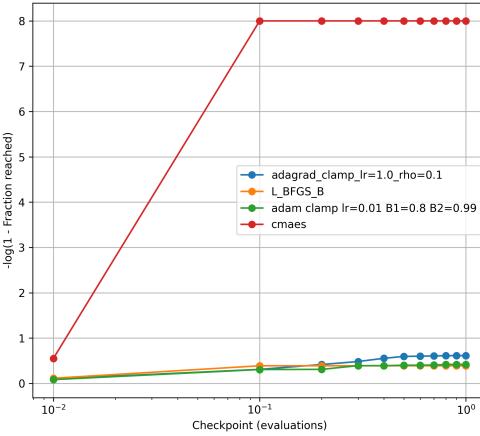
F242013



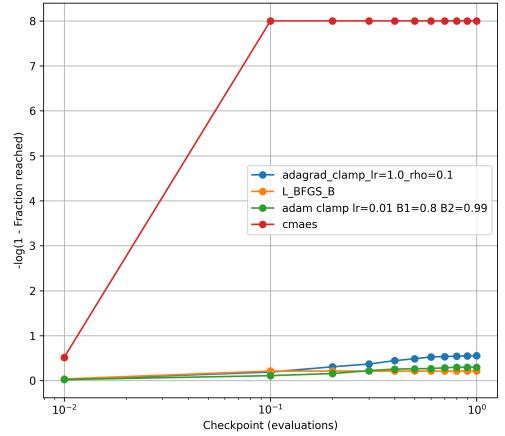
F242013, ECDF, dim=10



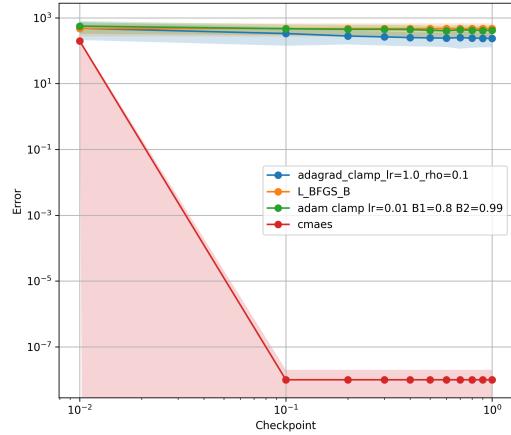
F242013, ECDF, dim=30



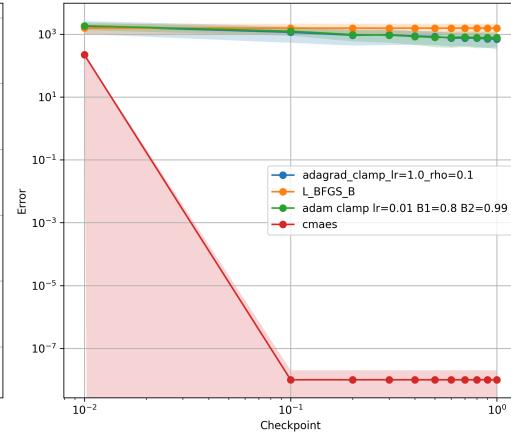
F242013, ECDF, dim=50



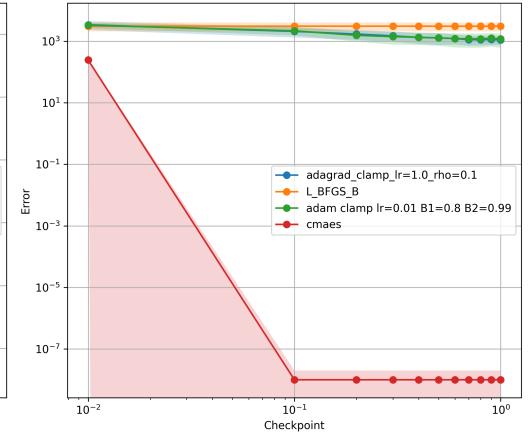
F242013, mean and std, dim=10



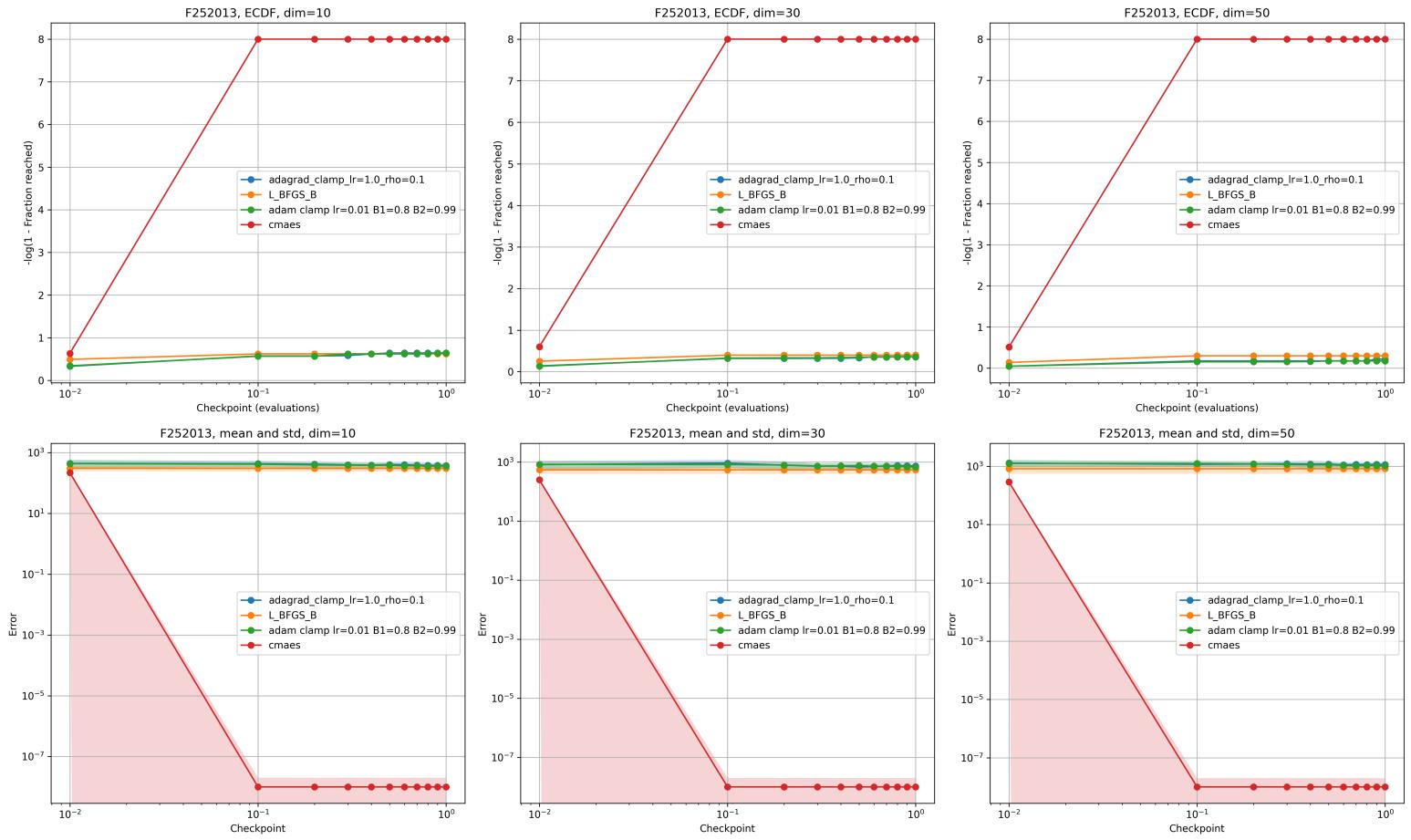
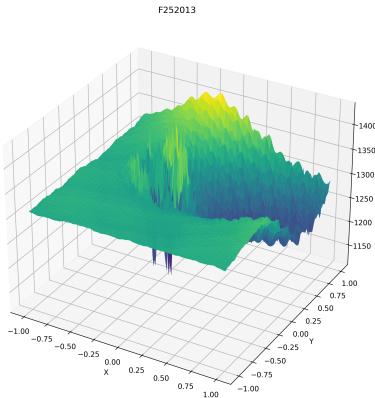
F242013, mean and std, dim=30



F242013, mean and std, dim=50

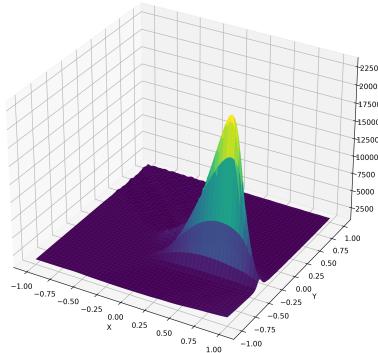


Sphere Function

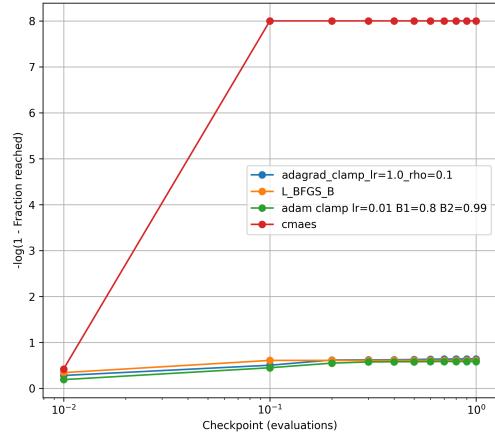


Sphere Function

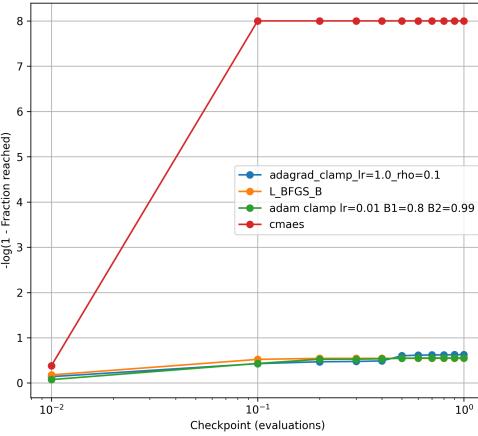
F262013



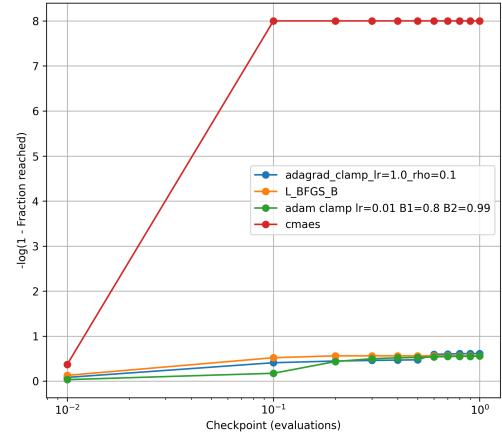
F262013, ECDF, dim=10



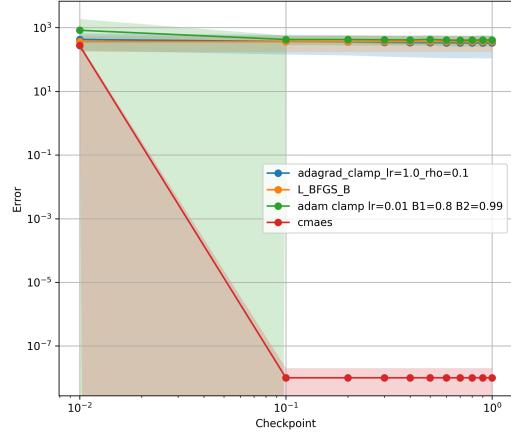
F262013, ECDF, dim=30



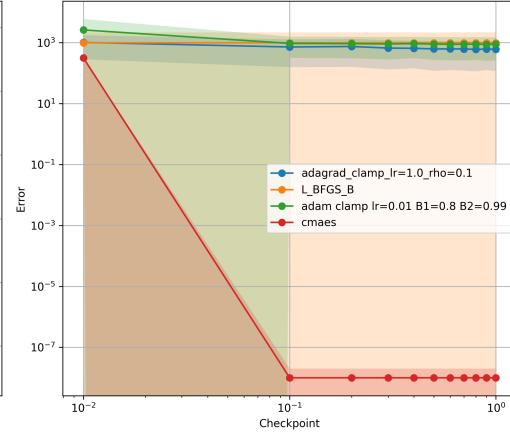
F262013, ECDF, dim=50



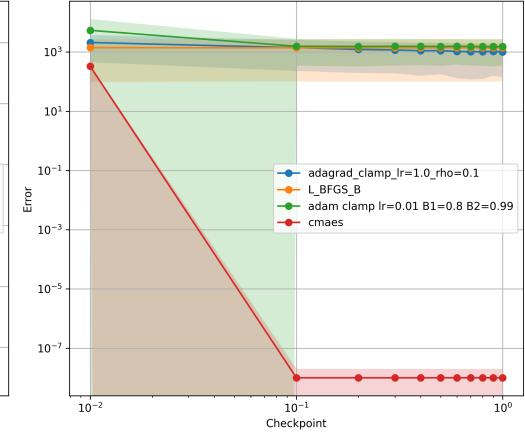
F262013, mean and std, dim=10



F262013, mean and std, dim=30

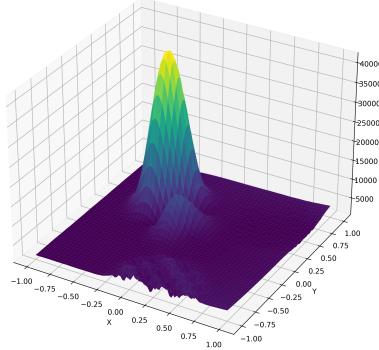


F262013, mean and std, dim=50

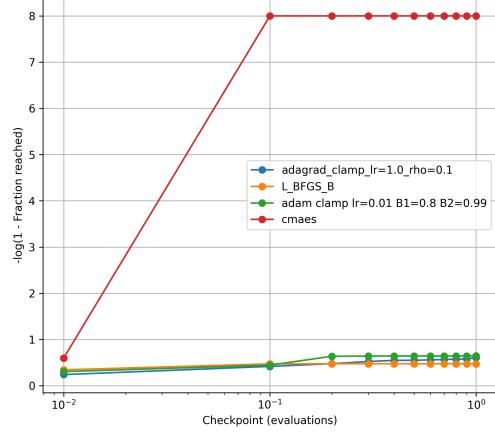


Sphere Function

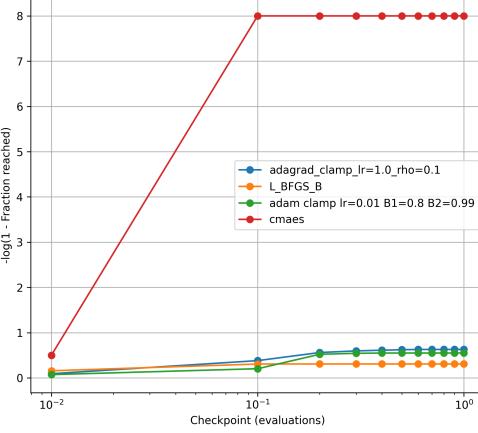
F272013



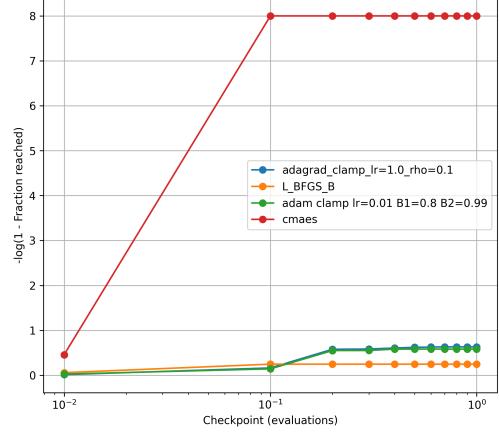
F272013, ECDF, dim=10



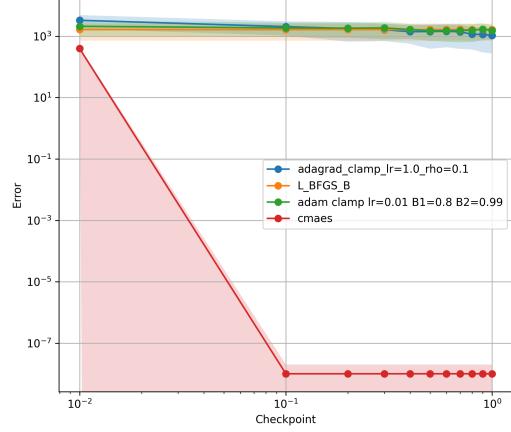
F272013, ECDF, dim=30



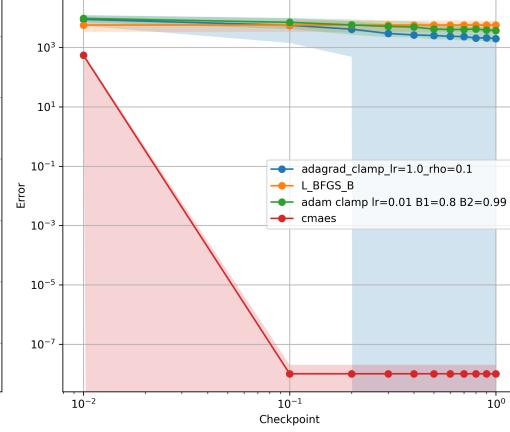
F272013, ECDF, dim=50



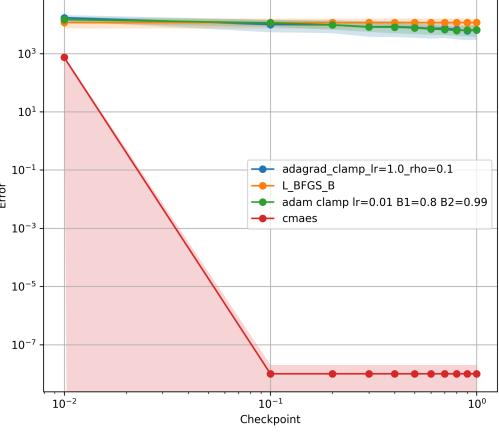
F272013, mean and std, dim=10



F272013, mean and std, dim=30

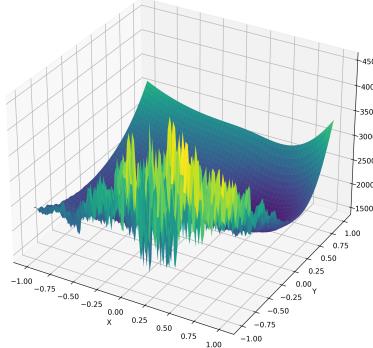


F272013, mean and std, dim=50

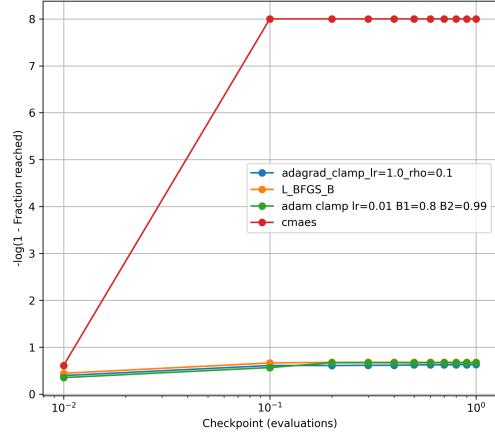


Sphere Function

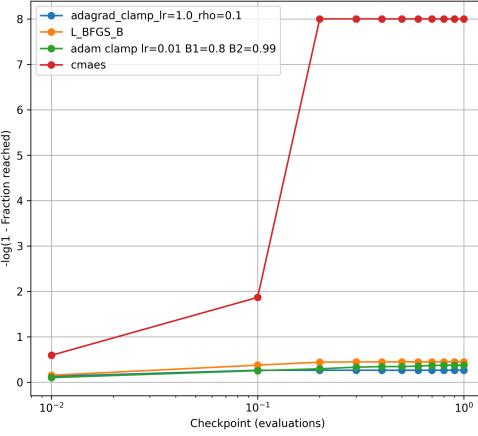
F282013



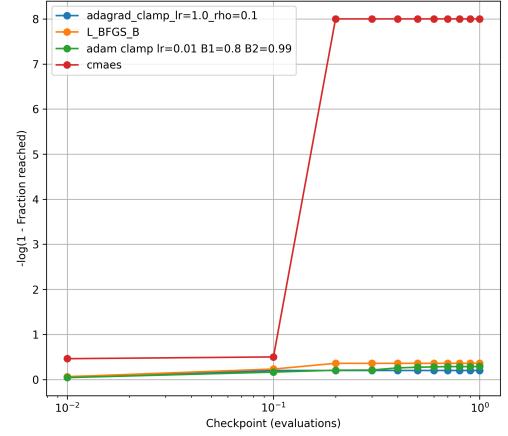
F282013, ECDF, dim=10



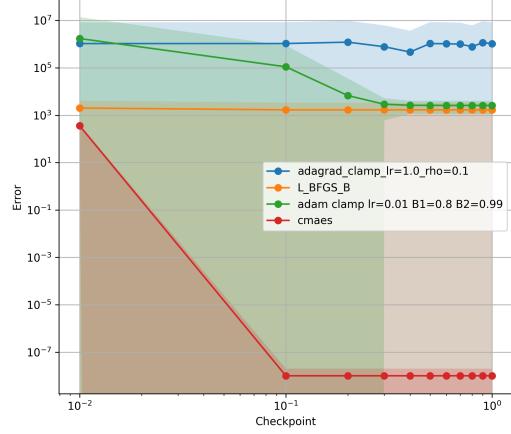
F282013, ECDF, dim=30



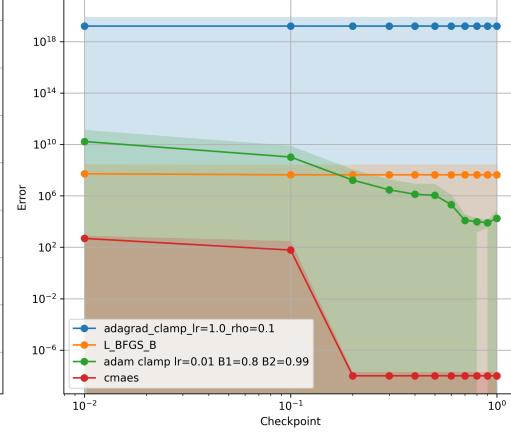
F282013, ECDF, dim=50



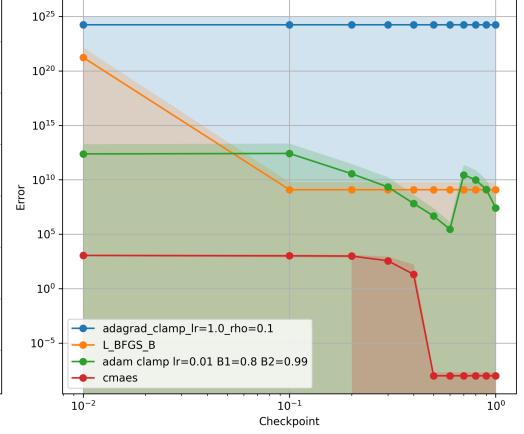
F282013, mean and std, dim=10



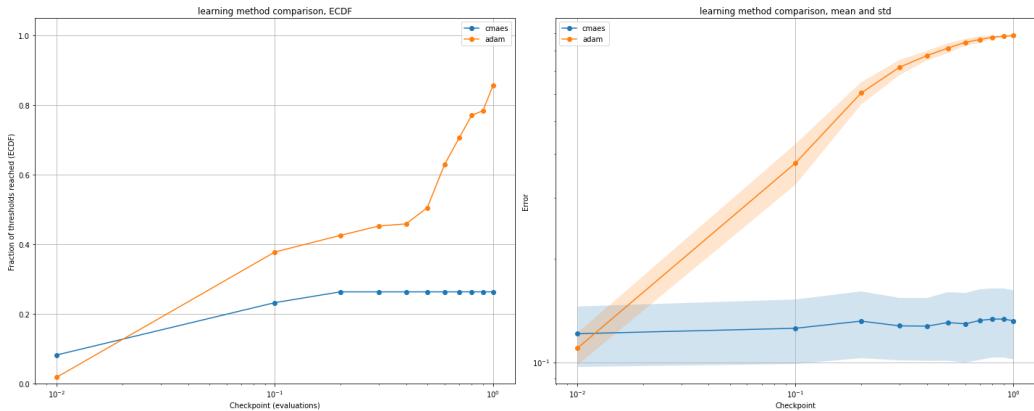
F282013, mean and std, dim=30



F282013, mean and std, dim=50



4.2 Wyniki dla sieci neuronowych



Rysunek 4.1: Enter Caption

4.3 Porównanie wyników

Można zauważać przewagę algorytmów globalnych w zadaniach optymalizacyjnych CEC, oraz algorytmów stochastycznych dla sieci neuronowych

Dodatkowo L-BFGS-B wykazuje zachowanie odmienne względem pozostałych użytych algorytmów stochastycznych, Adam i Adagrad.

Adam, chociaż jest rozszerzeniem Adagrad, nie zawsze odnosi lepsze wyniki.

Zarówno Adam jak i Adagrad nie działają optymalnie gdy funkcje mają elementy nierównoległe z osiami.

CMA-ES osiąga lepsze wyniki dla funkcji CEC2013, jednak czas obliczeń jest wyraźnie większy w porównaniu do algorytmów stochastycznych, mimo wykorzystania tej samej liczby wywołań funkcji celu.

Algorytm Adam, pomimo wielu zalet, nie zawsze radzi sobie lepiej względem pozostałych algorytmów stochastycznych. Dla funkcji (21 : Composition Function 1 (n=5,Rotated)), można zauważać że radzi sobie najgorzej z nich wszystkich

Można zauważać że Adam i Adagrad radzą sobie zaskakująco podobnie dla stosunkowo prostych funkcji, drugi momentum nie wydaje się być dla nich ogromną zmianą

Adagrad potrafi poradzić sobie lepiej od Adama, ale wydaje się przy tym mniej stabilny (27: Composition Function 7 (n=5,Rotated)). Potencjalnie dруgie momentum w algorytmie Adam zwiększa krok ponad potrzebę i dochodzi do chwilowego przestrzelenia względem Adagrad

Rozdział 5

Podsumowanie i wnioski

5.1 Podsumowanie wniosków obu sposobów

Algorytmy Adam, L-BFGS-B i Adagrad, poruszają w jednym wymiarze na raz. Można zauważyć patrząc na wyniki funkcji której najniższe punkty znajdują się na linii po przekątnej do osi (3 Rotated Bent Cigar Function), a funkcji z linią najwyższych punktów równoległą z osią (5: Different Powers Function), że jest to poważny problem dla funkcji stochastycznych

przewidywalnie algorytm CMAES, dzięki wykorzystywaniu rozproszonej po mapie populacji, pozwala na znaczco szybsze i dokładniejsze wyniki przy mniejszych ilościach wymiarów, jednak skalowalność okazała się o wiele goręsza w porównaniu do algorytmów stochastycznych

Ponieważ L-BFGS-B wyłącznie aproksymuje hesjan, im więcej "szumów" posiada funkcja, oraz im mniej jest przewidywalna, tym gorzej sobie radzi (9: Rotated Weierstrass Function)

sieci klasyczne, z algorytmem stochastycznym minimalizującym funkcję straty, nie zachowują zależności pomiędzy neuronami, każda aktualizacja wagi jest procesem lokalnym, liniowym i niezależnym od siebie.

natomiast algorytmy globalne mają znaczący problem ze skalowalnością. wszystkie wagi są zmieniane na raz, zależnie od siebie, co znacząco zwiększa złożoność obliczeniową.

5.2 Osiągnięte cele

Zaimplementowane środowisko do testowania algorytmów sztucznej inteligencji, z możliwością rozszerzenia o kolejne w przyszłości.

Opisanie działania algorytmów, oraz porównanie ich wad oraz zalet.

5.3 Kierunki dalszego rozwoju

Aktualne rozwiązania wykorzystujące algorytmy globalne do sieci neuronowej nie przenoszą informacji o zależności pomiędzy neuronami, wagi są spłaszczane do jednego wymiaru, ignorując powiązania pomiędzy neuronami. Potencjalnie możliwe jest zaprojektowanie algorytmu globalnego dostosowanego specyficznie pod naukę sieci neuronowych, który poradziłby sobie lepiej.

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.pw.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=eyJjb250ZXh0Ijp7ImZpcnN0UGFnZSI6InB1YmxpY2F0aW9uIiwicGFnZSI6InI

Rozdział 6

Załączniki