# Tests Algorithmen

1. Wenn 0 eingesetzt, muss sofort 0 rauskommen (bei Parabel, norm…)
2. Wenn nahe 0, wenig Iterationen
3. Alle Größen, die als ≥/≤0 definiert sind, werden im Algo auch so berechnet
4. Verschiedene Abbruchbedingungen testen
5. Tests schreiben?
6. Welche Veränderung bringt t? (Laufzeitverlängerung wenn kleiner?)

## Conv, exact

1. Passt in beiden Versionen (sparse, nicht sparse)
2. Passt (teilweise zu wenige 🡪) (in sparse und nicht sparse)
   1. In MATLAB-Algo zur Lösung des Subproblems stimmt was nicht (wird zu „schnell“ =0) Abbruchbedingung stimmt dann nicht.🡪 wenn man [a, 0,…,0] eingibt, kommt nur a als f-Wert raus. 🡪 auch bei sehr großen Zahlen für a  
      wenn eine Komponente des Startwertes = 0: Abbruch gleich nach erster Iteration 🡪 Problem mit quadprog
   2. wenn Eingabewert zu weit weg 🡪 E oft < 0
   3. Kein Problem, wenn nur eindimensionales x0
3. G und alpha nur in sehr wenigen Fällen unpassend (tol 10^-7);  
   alpha jetzt immer >= 0  
    E sehr häufig nicht passend 🡪 liegt an e (aber tol besser als 10^-14)  
   muss e „anpassen“ ähnlich wie für nonconv gamma dazuzählen, sodass e nicht negativ wird.  
   Kommt von Rundungsfehlern oder kann man das noch anders begründen?  
   In 1D e nie < 0, warum?  
   Welche Auswirkungen hat das Addieren von gamma? – wenn zu groß, dann längere Laufzeit (Abbruc hbedingung wird verfälscht, weil diese Norm(E) berechnet und E aus e berechnet wird  
   bei tol = 10^-6: G passend, tol = 10^-5: alpha passend (wenn e auf 0 gesetzt wird)  
   Komponenten von alpha immer >= 0  
   Komponenten von alpha scheinen nicht kleiner als 10^-5 zu werden  
   Bundle scheint nicht gekürzt zu werden, immer length(alpha) == k , da Komponenten von alpha nie klein genug werden  
   Entscheidung wann alpha klein genug???
4. Abbruchbed 3 gibt die wenigsten Funktionsauswertungen und im Mittel anscheinend auch keine schlechteren Ergebnisse
5. Wenn t kleiner E deutlich häufiger schlecht, Vermutung: Laufzeit länger  
   ??? genau Gegenteil von Erwartung, vielleicht damals mit \*t anstatt 1/t gerechnet dann: würde Sinn machen, Funktion dann „weniger konvex“ = flacher? Dann kann „Tangente“ einen größeren Bereich schneiden

* Irgendwann konnte e nicht berechnet werden; weiß nicht warum??? (nach weiteren Tests nicht mehr aufgetreten)