Allgemeine Hinweise zur Toolbox GEATbx

- Die Toolbox GEATbx (*Genetic and Evolutionary Algorithm Toolbox for MATLAB*) besteht aus einer Menge von M-files, die unter

  `\Programme\matlabR12\toolbox\geatbx`

  direkt oder in Unterverzeichnissen zu finden sind. In diesen sind Funktionen und Scripts, aus denen sich individuelle Evolutionäre Algorithmen zusammenbauen lassen, implementiert.

  Sie können sich die einzelnen M-files im MATLAB-Editor anschauen, indem Sie im Fenster "Current Directory" (meist links unten im MATLAB-Desktop) den oben genannten Pfad einstellen und auf einen Dateinamen doppelklicken.

- Unter `\\Merkur\LEHRE\Erben\GenAlg\geatbx3` finden Sie
  - ein Tutorial, Beispiele und einführende Beschreibungen als pdf-Dateien.
  - im Ordner `geadocu` eine ausführliche Dokumentation in Form von html-Dateien. Starten Sie diese mit `index.html`.

  Eine erste Übersicht über die Architektur der Toolbox verschafft man sich am besten mit Hilfe des Abschnitts *Quick Start* im pdf-Tutorial und der Analyse der dort beschriebenen Skripts `demofun1` und `demogeatbx`.

  Die Beschreibungen der von diesen Beispielprogrammen aufgerufenen Funktionen und Parameteroptionen können Sie gezielt in `geadocu` nachschlagen. (Klicken Sie insbesondere auf den Hyperlink `Parameter/Options` und blättern Sie hierin mit `Page` durch die verschiedenen Options.)

- In der Toolbox `\Programme\matlabR12\toolbox\geatbx` sind vor allem folgende Bausteine wichtig:
  - Die zentrale Funktion `geamain2` steuert den evolutionären Algorithmus und ruft im Wesentlichen die genetischen Operatoren auf.
  - Die Parameter für `geamain2` werden sowohl in den Skripts `demo*` (Unterverzeichnis `skripts`) als auch in den so genannten Toolboxfunktionen `tbx*` bereitgestellt.
  - In den Funktionen `init*`, `sel*`, `rec*` und `mut*` sind die genetischen Operatoren (Initialisierung, Selektion, Rekombination und Mutation) in verschiedenen Varianten implementiert.
  - Unter den Dateinamen `obj*` finden sich im Unterverzeichnis `objfun` einige Standard-Zielfunktionen (objective functions).

  Natürlich können alle diese vorgegebenen (Standard-)Funktionen den eigenen Bedürfnissen angepasst werden. Insbesondere wird man im Allgemeinen seine eigene Zielfunktion implementieren.

- Beachten Sie bei den Probeläufen von `demofun1` und `demogeatbx`, dass stets das Minimum der Zielfunktion gesucht wird. Machen Sie sich auch mit der Form des Outputs vertraut: Es werden Zwischen- und Endergebnisse protokolliert und auch grafisch visualisiert.

Übungsaufgabe 2

Erstellen Sie einen Genetischen Algorithmus, der auf dem klassischen Ansatz *einer* Population *binärcodierter* Chromosomen beruht und das *Roulette-Wheel-Selektionsverfahren*, *den One-Point-Crossover* und die *Bitflip-Mutation* verwendet. Nehmen Sie das Script `demofun1` als Vorlage und modifizieren Sie dieses entsprechend.

Führen Sie etliche Testläufe für die Schwefelsche Zielfunktion (vgl. Aufgabe 1) mit unterschiedlichen Parameter-Settings (Populationsgröße, Crossover- und Mutationswahrscheinlichkeit) durch.