Test problems for PSO, GSA and GA optimization algorithms:

1. Convex quadratic optimization (or linear regression): Find

$$\min_{\boldsymbol{x}} \frac{1}{2}\boldsymbol{x}^T A\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x}$$

where A is symmetric ($A^T = A$) positive definite ($\boldsymbol{z}^T A\boldsymbol{z} > 0$ for all $\boldsymbol{z} \neq \boldsymbol{0}$). The exact minimizer is $\boldsymbol{x}^* = A^{-1}\boldsymbol{b}$. The main issues are: how do the algorithms perform depending on

   (a) the dimension of $\boldsymbol{x}$, and

   (b) the condition number of A: $\kappa_2(A) = \|A\|_2 \|A^{-1}\|_2$.

   The condition number is the ratio of the largest to smallest eigenvalue of A (provided A symmetric positive definite).

   We can generate random A with (approximately) specified condition numbers as follows: generate diagonal matrices D with diagonal entries $d_{ii} = \exp((\ln \kappa)u_i)$ with u_i uniformly distributed over $[0, 1]$. Generate random orthogonal matrix Q as follows: generate pseudo-random $n \times n$ matrix Z with normally distributed entries. Then form the QR factorization $Z = QR$ with Q orthogonal. Then set $A = Q^T DQ$.

   The vector $\boldsymbol{b}$ should be generated using a normally distributed random entries.

2. Highly non-convex optimization: Use rotated separable functions

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} g_i((Q\boldsymbol{x})_i)$$

where Q is a randomly generated orthogonal matrix. The functions g_i should be non-convex functions where $\lim_{z\to\pm\infty} g_i(z) = +\infty$. Suitable choices are

$$g_i(z) = \frac{1}{2}z^2 + \sum_{i=1}^{m} \alpha_i \cos(\beta_i z + \gamma_i)$$

with randomly chosen α_i, β_i and γ_i. We choose α_i uniformly over $[0, M]$, β_i uniformly over $[1, B]$, and γ_i uniformly over $[0, 2\pi]$. We can take the value of m to be something modest, say, 3 to 10. The minimizer is $\boldsymbol{x}^* = Q^T\boldsymbol{z}^*$ where z_i is the global minimizer of g_i. If g_i has p_i local minimizers

(including the global minimizer) then the number of local minimizers of f is $\prod_{i=1}^{n} p_i$. Provided the global minimizer of each g_i is unique, there is only one global minimizer of f.

We want to see how the performance depends on

(a) the dimension of $\boldsymbol{x}$, and

(b) the values of m, M and B (which ultimately control the number of local minimizers of each g_i).

Note: When we say "random" we really mean "pseudo-random". The pseudo-random number generator (and seeds) should be fixed so that results can be more easily compared across runs and codes and even programming languages. There should be separate seeds for generating the function (so that we are comparing the same function across different methods) from the seeds for each method and each run.