MASSACHUSETTS INSTITUTE OF TECHNOLOGY
DEPARTMENT OF MATHEMATICS
18.065/18.0651 - MATRIX METHODS (SPRING 2018)

GRADIENT DESCENT FOR LEAST SQUARES PROBLEMS

---

1. Gradient Descent

Recall that the least squares problem:

$$x_{ls} = argmin_x ||Ax - b||_2,$$

has the solution $x_{ls} = A^+ b$ where A^+ is the pseudo-inverse. When the null-space of A is not empty, then x_{ls} is the minimum norm solution. When A is large, it can be computationally prohibitive to compute the SVD of A and then its pseudo-inverse before computing x_{ls}. In such settings, it can be shown that the iteration given by:

$$x_{k+1} = x_k - \mu A^H (Ax_k - b),$$

will minimize $||Ax - b||_2^2$ whenever $0 < \mu < 2/\sigma_1^2(A)$. Note that the iteration reaches a fixed point, i.e, $x_{k+1} = x_k$ when

$$A^H (Ax_k - b) = 0,$$

which are exactly the normal equations – so, the solution minimizes the least squares objective function! Your first task for this problem is to write a function called lsgd that implements the above least squares gradient descent algorithm. In Julia, your file should be named lsgd.jl and should contain the following function:

```
function lsgd(A, b, mu, x0, nIters)
#
# Syntax: x = lsgd(A, b, mu, x0, nIters)
#
# Inputs:   A is an m x n matrix
#           b is a vector of length m
#           mu is the step size to use, and must satisfy
#           0 < mu < 2 / norm(A)^2 to guarantee convergence
#           x0 is the initial starting vector (of length n) to use
#           nIters is the number of iterations to perform
#
# Outputs: x is a vector of length n containing the approximate solution
#
# Description: Performs gradient descent to solve the least squares problem
#
#  \min x \|b - A x\|_2
#
```

To check your solution, send your lsgd.jl file to eecs551@autograder.eecs.umich.edu. Once your solution is correct, forward your confirmation email to 18065-code-submission@mit.edu to receive credit for the lab.

After your code passes and you've forwarded your confirmation, use your solution to generate a plot of $||x_{ls} - x_k||$ as a function of k for A and b generated as:

```
m = 100; n = 50; sigma = 0.1
A = randn(m, n); xtrue = rand(n)
b = A * xtrue + sigma * randn(m);
```

Repeat the above experiment for $\sigma = 0.5, 1, 2$ and submit the plots with the problem set on Gradescope. Does $||x_{ls} - x_k||$ decrease monotonically with k in the plots?

2. Nesterov-Accelerated Gradient Descent

We now describe an accelerated gradient descent method due to Nesterov for the least squares problem:

$$x_{ls} = argmin_x ||Ax - b||_2,$$

that converges provably faster than the standard gradient descent method. The method consists of the following iteration:

$$\begin{aligned} t_{k+1} &= \frac{1 + \sqrt{1 + 4t_k^2}}{2} \\ z_{k+1} &= x_k + \frac{t_k - 1}{t_{k+1}}(x_k - x_{k-1}) \\ x_{k+1} &= z_{k+1} - \mu A^T(Az_{k+1} - b) \end{aligned}$$

initialized with $t_0 = 0$ and $x_{-1} = x_0$.

One can now show that the Nesterov-acceperated algorithm converges to x_{ls} when $0 < \mu < 1/\sigma_1^2(A)$. Your first task for this problem is to write a function called lsngd that implements the above Nesterov-accelerated least squares gradient descent algorithm.
In Julia, your file should be named lsngd.jl and should contain the following function:

```
function lsngd(A, b, mu, x0, nIters)
#
# Syntax: x = lsngd(A, b, mu, x0, nIters)
#
# Inputs:   A is an m x n matrix
#           b is a vector of length m
#           mu is the step size to use, and must satisfy
#           0 < mu < 1 / norm(A)^2 to guarantee convergence
#           x0 is the initial starting vector (of length n) to use
#           nIters is the number of iterations to perform
#
# Outputs: x is a vector of length n containing the approximate solution
#
# Description: Performs Nesterov-accelerated gradient descent to solve
#              the least squares problem
```

```
#
#  \min x \|b - A x\|_2
#
```

To check your solution, send your lsngd.jl file to eecs551@autograder.eecs.umich.edu. Once your solution is correct, forward your confirmation email to 18065-code-submission@mit.edu to receive credit for the lab.

After your code passes an you've forwarded your confirmation, use your solution to generate a plot of $||x_{ls} - x_k||$ as a function of k for A and b generated as:

```
m = 100; n = 50; sigma = 0.1
A = randn(m, n); xtrue = rand(n)
b = A * xtrue + sigma * randn(m);
```

Repeat the above experiment for $\sigma = 0.5, 1, 2$ and submit the plots with the problem set on Gradescope.

Compare the convergence characteristics for the accelerated gradient descent method to the standard gradient descent method. For a given step-size μ, which converges faster to the true solution? Does this hold for different values of (allowable) μ? Turn in your plots illustrating the rate of convergence. Note that the convergence of the accelerated gradient descent method is not necessarily monotonic.