Numerical Optimization Assignment 2

Justin Cata 10/6/2022

Written Exercises

2.1

For each of the two cost functions on $\mathbb{R}^2$, *i. e.*, $x^T = (\xi_1, \xi_2)$, determine the gradient, the Hessian and the stationary points. Identify which stationary points are minima, maxima or neither.

a. $f(x) = 8\xi_1 + 12\xi_2 + {\xi_1}^2 - 2{\xi_2}^2$

$\nabla f = (8 + 2\xi_1, 12 - 4\xi_2)^T$

$\nabla f = 0 \implies x^* = (-4, 3)^T$

$H_f = \begin{pmatrix} 2 & 0 \\ 0 & -4 \end{pmatrix} \implies x^*$

is a saddle point.

b. $f(x) = 100({\xi_2 - {\xi_1}^2})^2 + (1 - \xi_1)^2$

$\nabla f = (-400\xi_1(\xi_2 - {\xi_1}^2) + 2(\xi_1 - 1), 200(\xi_2 - {\xi_1}^2))^T$

$\nabla f = 0 \implies x^* = (1, 1)^T$

$H_f = \begin{pmatrix} 802 & -400 \\ -400 & 200 \end{pmatrix} \implies x^*$

is a minimum.

2.2

Let $f(x) : \mathbb{R}^n \implies \mathbb{R}$ be a convex function. Let the set Γ be the set of global minimizers of f. Show that Γ must be a convex set.

If f is convex function then there is a set of global minimizers s.t. for some radius, all x in the neighborhood of a global minimizer $x^* \epsilon \Gamma$, $f(x^*) \leq f(x)$. Because the radius is arbitrary the set of global minimizers must be a convex set.

Programming Exercises

1) Statement of Problem

We empirically compare the performance of numerous iterative methods used to solve convex quadratic unconstrained optimization probelms. The following iterative methods were implemented and tested:

- Gauss-Siedel (Richardson Family)
- Conjugate Gradient
- Steepest Decent
- Slow Steepest Decent
- Gauss-Southwell

2) Description of Mathematics

3) Description of the Implementation

An object oriented approach was used for this implementation. The user supplies the A matrix, b, and x vectors via text files which are then read into their corresponding data types. For stability purposes, the matrix is stored as a vector of vectors as opposed to an nxm array. The main method then constructs an IterativeMethod object (Iterate1) which takes the A matrix, b, and x vectors as input.

This object internally constructs empty solution vectors for each possible iterative method. The user can then call the public method SolveSystem which is supplied a string character representing the iterative method of interest. It will then solve an "exact" solution in double precision for metric purposes followed by a single precision solution regarding the method of interest. This method is characterized by the generalized decent method. The unique aspects of the each family of solutions is contained particularly in get_direction method which runs a conditional statement over the string value that defines the method.

The solution vectors are stored as private attributes in the object and can only be accessed via the public print methods. Performance metrics of the method are also calculated at each iteration and stored in in their own vector whereby the index represents the iteration number.

Its worth noting that his object implementation also includes the option to apply a preconditioner to the system. If so, cholesky factorization is utilized to solve for the direction vectors.

4) Results

Iterative Methods on a Diagnoal Matrix

For our first test we want to measure the relative error of a well and ill conditioned diagnol matrices. The following well conditioned matrix was used and its corresponding condition number $K(A) = 6$.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 6 \end{pmatrix}$$

The following ill conditioned matrix was used and its corresponding condition number $K(A) = 600$.

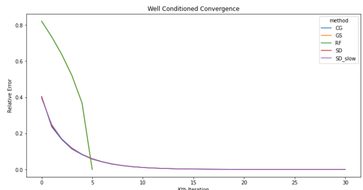
$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 600 \end{pmatrix}$$

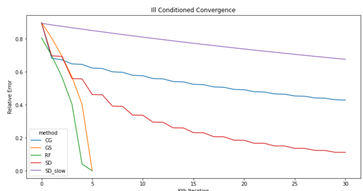
The most direct way to test correctness is via the relative error of X_k with the exact solution X_{exact}.

$$\frac{||X_k - X_{exact}||}{||X_{exact}||}$$

In [3]:
```
import pandas as pd, matplotlib.pyplot as plt
from scipy.stats import linregress
df = pd.read_csv('analysis_data.csv').iloc[: , :-1]
df[df['conditioning']==0].pivot(index='kth_iteration', columns='method', values='metric_known').plot(xlabel='Kth Iteration', ylabel='Relative Error', title='Well Conditioned Convergence', figsize=(12,6))
df[df['conditioning']==1].pivot(index='kth_iteration', columns='method', values='metric_known').plot(xlabel='Kth Iteration', ylabel='Relative Error', title='Ill Conditioned Convergence', figsize=(12,6))
```

Out[3]: <AxesSubplot:title={'center':'Ill Conditioned Convergence'}, xlabel='Kth Iteration', ylabel='Relative Error'>





As expected, the two descent methods never fully reach the exact solution and have quite poor performance when aplied to an ill conditioned diagnol matrix. Both the Richardson Method (Gauss Siedel) and Gauss-Southwell converge exactly and within an iteration count equivalent to the rank of the system.

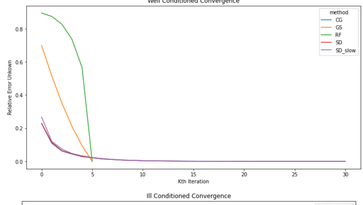
Regarding well conditioned convergence, the descent methods seem to perform much better with fewer iterations but by around the 4th or 5th iteration the descent methods make marginal progress while Richardson and G-S reach the exact solution.
Note that the 30th iteration upper bound for the well and ill conditioned matrices are $4.132e-5$ and 0.905 respectively which all iterative methods abide by.

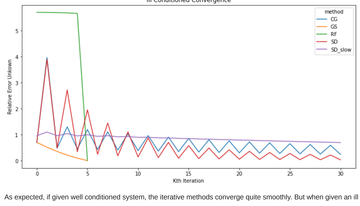
More often than not we do not know the exact solution to a system and so we have to use the relative error to test for correction. The metric is given by the formula below where r_k is the residual and b is the b vector.

$$\frac{||r_k||}{||b||}$$

In [4]:
```
df[df['conditioning']==0].pivot(index='kth_iteration', columns='method', values='metric_unknown').plot(xlabel='Kth Iteration', ylabel='Relative Error Unkown', title='Well Conditioned Convergence', figsize=(12,6))
df[df['conditioning']==1].pivot(index='kth_iteration', columns='method', values='metric_unknown').plot(xlabel='Kth Iteration', ylabel='Relative Error Unkown', title='Ill Conditioned Convergence', figsize=(12,6))
```

Out[4]: <AxesSubplot:title={'center':'Ill Conditioned Convergence'}, xlabel='Kth Iteration', ylabel='Relative Error Unkown'>



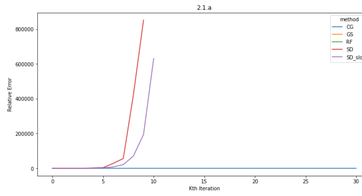


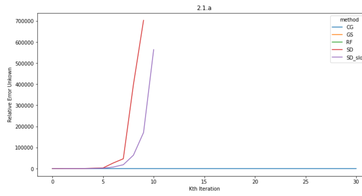
As expected, if given well conditioned system, the iterative methods converge quite smoothly. But when given an ill conditioned matrix, the residual of the non-stationary descent methods appear to oscillate downward in a simialr fashion as a damped oscillator.

Below we test the iterative methods over the quadratic system in problem 2.1.a where $\nabla f = (8 + 2\xi_1, 12 - 4\xi_2)^T \implies \begin{pmatrix} 2 & 0 \\ 0 & -4 \end{pmatrix}(x_1, x_2)^T = (-8, -12)^T$

In [7]:
```
df = pd.read_csv('analysis_data.csv').iloc[: , :-1]
df[df['conditioning']==0].pivot(index='kth_iteration', columns='method', values='metric_known').plot(xlabel='Kth Iteration', ylabel='Relative Error', title='2.1.a', figsize=(12,6))
df[df['conditioning']==1].pivot(index='kth_iteration', columns='method', values='metric_unknown').plot(xlabel='Kth Iteration', ylabel='Relative Error Unkown', title='2.1.a', figsize=(12,6))
```

Out[7]: <AxesSubplot:title={'center':'2.1.a'}, xlabel='Kth Iteration', ylabel='Relative Error Unkown'>





5) Conclusion

Although all the iterative methods eventually converged to the exact solution, ill conditioning had a drastic impact on the perfomance of the descent methods as expected. It was reassuring to see that the relative errors fell within the theoretical bounds given the associated condition numbers.

With the exception of the steepest descent method, all the iterative methods converge to the minimizer $(-4, 3)^T$. The steepest decents inability to converge to the minimizer in this case because the quadratic system is not strictly convex and the minimizer defines a saddle point rather than a true global minimum.