浙江大学数学科学学院
School of Mathematical Sciences, Zhejiang University

专业：自动化控制
姓名：林皓泓
学号：3170105156
日期：2020 年 3 月 30 日
地点：在家网课

课程名称：优化实用算法　指导老师：王何宇　成绩：

1 Wolfe 条件下的线搜索

Algorithm 1 基于 Wolfe 条件的线搜索

Require: α_0, $c_1 = 1e-4, c_2 = 0.9$.

Ensure: Iteration Data: x_k, f_k, $error_k$ $f_0 \leftarrow f(x_0), g_0 \leftarrow \nabla f(x_0)$

while $k < max_iter$ & $error < tolerance$ **do**
  $\alpha \leftarrow \alpha_0$, $f_k \leftarrow f(x_k)$, $p_k = -\nabla f_k$ *or* $-\nabla^2 f_k^{-1} \nabla f_k$
  if $f(x_k + \alpha p_k) > f(x_k) + c\alpha \nabla f_k^T p_k$ **then**
    $\alpha \leftarrow Interpolation(\alpha)$
  end if
  $error \leftarrow f(x_k + \alpha p_k)$
  $x_{k+1} \leftarrow x_k + \alpha p_k$
end while

由于三次多项式的极值点不一定能取到，此处我们统一采用二次多项式进行步长选取，也即：

$$\alpha^* = -\frac{\phi'(0)\alpha_0^2}{2[\phi(\alpha_0) - \phi(0) - \phi'(0)\alpha_0]}$$

除此以外，我们也同样可以用一种类似于“有生之年”的方法，迭代选择适合 Wolfe 条件的步长。

Algorithm 2 “有生之年”的步长选择

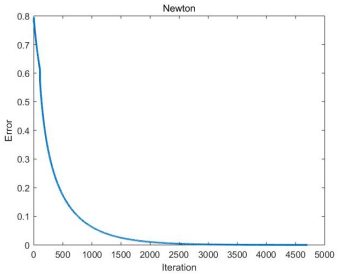
Require: $b \leftarrow Inf, a \leftarrow 0$

while True **do**
  if Armijo not satisfied **then**
    $b \leftarrow \alpha$
    $\alpha = (\alpha + a)/2$
    Update x_n, continue
  end if
  if Curve Condition not satisfied **then**
    $\alpha = min(2\alpha.(b + \alpha)/2)$
    $\alpha = Interpolation(\alpha), break$
  end if
end while

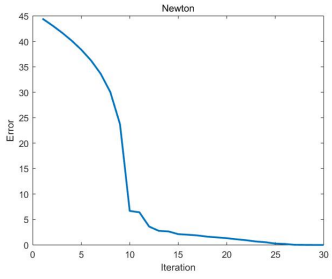
注意事项：

- 使用最速下降的时候，$c_2 = 0.1$, Newton 法的时候，$c_2 = 0.9$；
- 最速下降的步长选择中，每次新的迭代 $\alpha_0 = 1.01\alpha^*$，可以达到比较好的鲁棒性；
- Newton 法中，每次 $\alpha_0 = 1$，方向向量不需要归一化.
- 比起 BLS 甚至 α 取常数，局部收敛变慢，全局稳定性变好。

基于 Wolfe 条件，可以得到如下的误差曲线，：



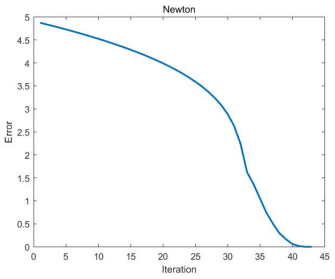
(a) GD，初值在 0 1 之间随机 5 维



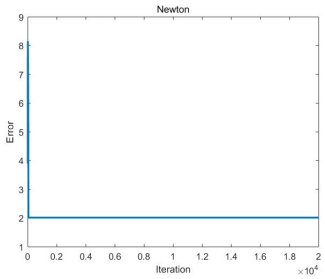
(b) Newton，初值在-10 10 之间随机 5 维

2 基于单位修正法的牛顿方向

上述牛顿法存在这样的问题，当我们把初值拉远以后：



(c) newton 在-10 10 的初值，5 维，很艰险



(d) newton 在-10 10 的初值，5 维，收敛到了局部极小 (-0.9621,0.9357, 0.8807, 0.7779, 0.6051)

我们不能坐以待毙，为了防止“越过山脊”，我们需要对 Hessian 矩阵 $\nabla^2 f_k$ 作修正，保证其正定。

- 参照书本算法 3.3，可以得到一类复杂度较高的方法。代码实现中注意几个细节：
  - $a_{ii} \geq 0$ 的条件是不够的，需要 $a_{ii} > \beta > 0$
  - 编程过程中，楚列斯基分解可以用矩阵操作提升效率。
  - 我们此处给定 $\beta = 10^{-3}$。
- 基于书本方法，参照有关文献 [1]，设计一种基于 LDL 楚列斯基分解的修正。
  - 其中设定 $\delta = 10^{-3}, \beta = 0.8$
  - 此方法复杂度低于前者（避免多次楚列斯基分解），适当设置参数效果较好。

3 算法评价与反思

3.1 初值依赖

在实验中，对所有初值 x0，当 $x_i > 0$ 对所有 x 成立的时候，无论 $||x||$ 有多大，均可以收敛。牛顿法都能够收敛到 global optimal。

对于空间内的任意点，我们通过 rand*A-A/2 生成一系列随机向量作为 $(-A/2, A/2)$ 区间内的输入初值，放入迭代中，统计其中收敛的比例。由于篇幅限制，不把改进和没改进的牛顿法作比较（鞭尸）了，基本上对于改进牛顿法可以收敛的东西，有 25% 及以上（4 维以上情况）是原始牛顿法没法收敛的。

下图可以看出，随着维数的提高，收敛的比例也在逐渐下降，初值依赖逐渐上升。且使用单位修正法对收敛率有提升作用。

(-100, 100) 区间内的收敛比例：

变量数	2	3	4	5	6	7	8	9	10
初值 (0 100) 不收敛比例	0	0	0	0	0	0	0	0	0
初值 (-100 100) 不收敛比例	0	0.01	0.20	0.21	0.25	0.28	0.31	0.35	0.37

查阅文献发现，之所以在 3D 4D 的时候发生“跳变”，是因为在 4D 以上，Rosenbrock 函数在 (-1, 1, ..., 1) 附近有局部极小值。故 Rosenbrock 的初值依赖主要是在 x_1 上，我们给一个函数：其中 $x_1 \in (0, 1000)$, $x_{rest} \in (-1000, 1000)$。

```
x0 = [converge*rand(1000, 1), converge*rand(1000, num-1)-converge/2];
```

如下表，初值依赖显著下降。

变量数	2	3	4	5	6	7	8	9	10
初值 (0 100) 不收敛比例	0	0	0.05	0.06	0.05	0.05	0.10	0.12	0.12
初值 (-1000 1000) 不收敛比例	0	0	0.06	0.07	0.08	0.10	0.11	0.12	0.15

表 1: The actual convergence rate

3.2 实际收敛阶

实际收敛阶的判断需要借助 $||x - x^*||$ 的信息。

此处收敛阶 k 的定义是 $||x_{k+1} - x^*|| = o(||x_{k+1} - x^*||^c)$. 为了我们把所有 $error < 1$ 的数值打印出来，取对数后即可得到收敛阶的一个估计。也即收敛率估计如下；

$$\hat{c} = mean|\frac{log(error_{k+1})}{log(error_k)}|$$

由于这里 error 要小于 1 比较便于计算，我们取初值 $x_0 \in (0.5, 1.5)$, 作多次修正牛顿法，从 error<1 开始将后续步骤里的 error 代入计算，记录平均值如下：

变量数	2	3	4	5	6	7	8	9	10
初值 0.5 1.5	1.6901	1.6795	1.6689	1.6443	1.6460	1.6306	1.6281	1.6372	1.5939

表 2: The actual convergence rate

由于仿真次数的限制（这个实际收敛率波动还是比较大），有些部分存在偶然误差；我们这里设置仿真次数 1000，记录数据如上。可以看出改进牛顿法的实际收敛率会有所损失，在 1.6 左右。这可能是因为为了保证收敛和正定性，牺牲了部分效率。

本次实验中，在修正的位置花了比较多时间，尝试了 LL^T 和 LDL^T 型的方法，参考文献的方法在实际使用过程中最好用。评估上各个修正方法的表现几乎不在一个量级上，比较意义似乎不大。比较有发现的点就是初值依赖对不同维度有着不同的取值。譬如对第一维应该尽量避免是负的，否则会对最终的收敛带来很大影响。

参考文献

[1] S. H. Cheng and N. J. Higham. A modified Cholesky algorithm based on a symmetric indefinite factorization. SIAM J. Matrix Anal. Appl., 19(4):1097-1110, 1998. doi:10.1137/S0895479896302898,