优化实用算法: 第二次作业

2022 年 4 月 22 日

指导老师：徐翔

徐圣泽 3190102721

Problem 1

1.Implement CG algorithm to solve linear systems in which A is the Hilbert matrix, whose elements are $A(i,j) = \frac{1}{i+j-1}$. Set the right-hand-side to $b = (1, 1, \cdots, 1)^T$ and the initial point to $x_0 = 0$. Try dimensions $n = 5, 8, 12, 20$ and show the performance of residual with respect to iteration numbers to reduce the residual below 10^{-6}.

解：首先利用 Python 写出共轭梯度法的函数，代码如下：

```
def CGmethod(A, b, n, x0, max, epsilon):
#求解Ax=b的共轭梯度法，n为维数，x0为初值，max为最大迭代次数，epsilon为精度
        k = 0
        x = x0
        r = np.dot(A, x)-b
        p = -r
        while k <= max:
                if np.sqrt(np.dot(r.T, r)) <= epsilon:
                        break
                alpha = np.dot(-r.T, p)/np.dot(p.T, np.dot(A, p))
                x = x+np.array(alpha)[0][0]*p
                r0 = r
                r = r+np.dot(np.array(alpha)[0][0]*A, p)
                beta = np.dot(r.T, r)/np.dot(r0.T, r0)
                p = -r+np.array(beta)[0][0]*p
                k = k+1
        return x, k
```

下面利用该函数求解题目指定的 Hilbert 矩阵，代码如下：

```
dimension=np.array([5,8,12,20])#四种维数情况
m=1
epsilon=1e-6
for index in range(4):
        n = dimension[index]
        A = [[random.random() for col in range(n)] for row in range(n)]
        x0 = [[random.random() for col in range(m)] for row in range(n)]
        b = [[random.random() for col in range(m)] for row in range(n)]
        for i in range(n):
                for j in range(n):
                        A[i][j] = 1/(i+j+1)
        #A为n阶Hilbert矩阵，初始化矩阵并赋值
        for i in range(n):
                for j in range(m):
                        x0[i][j] = 0
                        b[i][j] = 1
        #初始化x0和b
        A = np.mat(A)
        b = np.mat(b)
```

```
        x0 = np.mat(x0)
        #将A,b,x0转化为矩阵类型
        (x, k) = CGmethod(A, b, n, x0, 10000, epsilon)
        print(x)
        print(k)
```

5 阶 Hilbert 矩阵使用 CG 法，迭代次数为 6，解如下：

```
[[    4.99999996]
[ -119.99999998]
[  630.00000003]
[-1119.99999997]
[  630.00000003]]
```

8 阶 Hilbert 矩阵使用 CG 法，迭代次数为 20，解如下：

```
[[-8.00001170e+00]
[ 5.04000300e+02]
[-7.56000196e+03]
[ 4.62000058e+04]
[-1.38600011e+05]
[ 2.16216014e+05]
[-1.68168011e+05]
[ 5.14800039e+04]]
```

12 阶 Hilbert 矩阵使用 CG 法，迭代次数为 44，解如下：

```
[[-9.60760022e+00]
[ 8.15348107e+02]
[-1.64961591e+04]
[ 1.35509221e+05]
[-5.36480720e+05]
[ 1.02540139e+06]
[-6.42580101e+05]
[-6.57592000e+05]
[ 8.04244085e+05]
[ 6.63073788e+05]
[-1.24127915e+06]
[ 4.65505726e+05]]
```

20 阶 Hilbert 矩阵使用 CG 法，迭代次数为 85，解如下：

```
[[-1.09717066e+01]
[ 1.05074723e+03]
[-2.39540259e+04]
[ 2.20415205e+05]
[-9.65326279e+05]
[ 1.99009026e+06]
```

```
[-1.25270412e+06]
[-1.34347342e+06]
[ 8.83234198e+05]
[ 1.68796915e+06]
[ 3.88214600e+05]
[-1.30552508e+06]
[-1.71054478e+06]
[-5.28252009e+05]
[ 1.20867559e+06]
[ 2.00288477e+06]
[ 9.44593867e+05]
[-1.43403647e+06]
[-2.65094537e+06]
[ 1.88784341e+06]]
```

我们发现，本题中 n 阶的 Hilbert 矩阵在求解时迭代次数均大于 n，且随着维数的上升迭代次数增长非常迅速，这与一般对 n 阶矩阵应用共轭梯度法求解时迭代次数不超过 n 的结论不符，我们推测这一点可能与 Hilbert 矩阵的条件数有关。
经过计算发现，5 阶矩阵的条件数为 4.77×10^5，8 阶矩阵的条件数为 1.53×10^{10}，12 阶矩阵的条件数为 1.62×10^{16}，20 阶矩阵的条件数为 2.11×10^{18}。Hilbert 矩阵的条件数较大，因此共轭梯度法迭代时无法达到预想的准确程度，则 CG 法的收敛速度无法达到理论预期。

Problem 2

2. Derive Preconditioned CG Algorithm by applying the standard CG method in the variables $\hat{x}$ and transforming back into the original variables x to see the expression of preconditioner M.

解：对变量 x 乘上非奇异矩阵 C 得到新变量 $\hat{x}$，即 $\hat{x}=Cx$，此时目标函数表达式如下

$$\phi(\hat{x})=\frac{1}{2}\hat{x}^T(C^{-T}AC^{-1})^{-1}\hat{x}-(C^{-1}b)^T\hat{x}$$

记 $\hat{A}=C^{-T}AC^{-1}$，$\hat{b}=C^{-T}b$，首先写出对于 $\hat{x}$ 的 CG 算法。

Algorithm 1 CG

Require: $\hat{x_0}=Cx_0$;
  $\hat{r_0}\leftarrow C^{-T}(Ax_0-b)$, $\hat{p_0}\leftarrow-\hat{r_0}$, $k\leftarrow0$;
  while $r_k\neq0$ **do**
    $\hat{\alpha_k}\leftarrow-\frac{\hat{r_k}^T\hat{p_k}}{\hat{p_k}^T\hat{A}\hat{p_k}}$;
    $\hat{x}_{k+1}\leftarrow\hat{x_k}+\hat{\alpha_k}\hat{p_k}$;
    $\hat{r}_{k+1}\leftarrow\hat{r_k}+\hat{\alpha_k}\hat{A}\hat{p_k}$;
    $\hat{\beta}_{k+1}\leftarrow\frac{\hat{r}_{k+1}^T\hat{r}_{k+1}}{\hat{r_k}^T\hat{r_k}}$;
    $\hat{p}_{k+1}\leftarrow-\hat{r}_{k+1}+\hat{\beta}_{k+1}\hat{p_k}$;
    $k\leftarrow k+1$;
  end while

以上就是对 $\hat{x}$ 应用标准 CG 算法的流程，下面我们对上述算法的每一步进行分析，改为对 x 进行过预处理后的形式。

- 首先由 $\hat{x_0} = Cx_0$ 合理推测每一步均有 $\hat{x_k} = Cx_k$，同理由 $\hat{r_0} = C^{-T}(Ax_0 - b) = C^{-T}r_0$ 可知每一步也均有 $\hat{r_k} = C^{-T}r_k$。

- 对于 $\hat{x}_{k+1} \leftarrow \hat{x_k} + \hat{\alpha_k}\hat{p_k}$，代入后可以得到 $x_{k+1} \leftarrow x_k + C^{-1}\hat{\alpha_k}\hat{p_k}$，为了保证算法的形式不变，新算法的形式也应为 $x_{k+1} \leftarrow x_k + \alpha_k p_k$，因此有 $\alpha_k = \hat{\alpha_k}$，$p_k = C^{-1}\hat{p_k}$。

- 对于 $\alpha_k = \hat{\alpha_k}$，代入化简 $\hat{\alpha_k} = -\frac{\hat{r_k}^T\hat{p_k}}{\hat{p_k}^T\hat{A}\hat{p_k}}$ 后，得到 $\alpha_k = -\frac{r_k^T C^{-1}Cp_k}{p_k^T C^T C^{-T}AC^{-1}Cp_k} = -\frac{r_k^T p_k}{p_k^T Ap_k}$

- 对于 $\hat{r}_{k+1} \leftarrow \hat{r_k} + \hat{\alpha_k}\hat{A}\hat{p_k}$，代入得到 $C^{-T}r_{k+1} = C^{-T}r_k + \alpha_k C^{-T}AC^{-1}Cp_k$，化简得到 $r_{k+1} = r_k + \alpha_k Ap_k$。

- 对于 $\hat{p_0} = -\hat{r_0}$，代入得到 $Cp_0 = -C^{-T}r_0$，化简得到 $p_0 = -(C^TC)^{-1}r_0$，引入新变量 $y_0 = (C^TC)^{-1}r_0$。设预处理器 $M = C^TC$，则有 $y_0 = M^{-1}r_0$，$p_0 = -y_0$。

- 对于 $\hat{\beta}_{k+1} \leftarrow \frac{\hat{r}_{k+1}^T\hat{r}_{k+1}}{\hat{r_k}^T\hat{r_k}}$，保持 $\beta_{k+1} = \hat{\beta}_{k+1}$，则有 $\beta_{k+1} = \frac{r_{k+1}^T C^{-1}C^{-T}r_{k+1}}{r_k^T C^{-1}C^{-T}r_k} = \frac{r_{k+1}^T M^{-1}r_{k+1}}{r_k^T M^{-1}r_k} = \frac{r_{k+1}^T y_{k+1}}{r_k^T y_k}$。

- 对于 $\hat{p}_{k+1} \leftarrow -\hat{r}_{k+1} + \hat{\beta}_{k+1}\hat{p_k}$，代入得到 $Cp_{k+1} = -C^{-T}r_{k+1} + \beta_{k+1}Cp_k$，化简得到 $p_{k+1} = -M^{-1}r_{k+1} + \beta_{k+1}p_k = -y_{k+1} + \beta_{k+1}p_k$。

Algorithm 2 Preconditioned CG

Require: x_0, preconditioner M;
  $r_0 \leftarrow Ax_0 - b$;
  Solve $My_0 = r_0$, $p_0 \leftarrow -y_0$, $k \leftarrow 0$;
  while $r_k \neq 0$ **do**
    $\alpha_k \leftarrow -\frac{r_k^T p_k}{p_k^T Ap_k}$;
    $x_{k+1} \leftarrow x_k + \alpha_k p_k$;
    $r_{k+1} \leftarrow r_k + \alpha_k Ap_k$;
    Solve $My_{k+1} = r_{k+1}$;
    $\beta_{k+1} \leftarrow \frac{r_{k+1}^T y_{k+1}}{r_k^T y_k}$;
    $p_{k+1} \leftarrow -y_{k+1} + \beta_{k+1}p_k$;
    $k \leftarrow k + 1$;
  end while

上面算法中的 M 为预处理器，$M = C^TC$，是一个对称正定矩阵，以上就是求解 x 的经过 M 预处理的 CG 算法的流程。