线性方程的迭代解法

代码的一些说明

首先观察到这个矩阵，是弱对角占优矩阵，而且不可约，因此 Jacobi 和 G-S 必然收敛，而 SOR 如果满足 0< <=1 也一定是收敛的，但对于 SOR 这是充分非必要条件，就是说，如果 >1，也可能会收敛。这里默认 SOR 的 为 1.15，之后我们再探究收敛性与 的关系。判停条件是前 3 位数字不变，于是用了 `(round(x,3)!=round(y,3))`。另一种常用的条件是：max(abs(x-y))>1E-3。但这个实际上并不满足需求，可以举出反例，0.3726 与 0.372 虽然相差不到 0.001，但是四舍五入以后的前三位不相同。而 `(round(x,3)!=round(y,3))` 这个条件是更强的，如果满足它，一定也会满足 max(abs(x-y))<1E-3。因此用这个判停条件，而不是 max(abs(x-y))>1E-3，迭代次数和精确度会更高。衡量与精确解的误差用的是：每一个元素的相对误差构成的向量的无穷范数。用代码表达是：`max((solveIter(A$A,A$b,mode = mode)-A$correct)/A$correct)`，不同于 $\frac{\|\Delta x\|}{\|x\|}$。设置了最大迭代次数为 5000(考虑到希望较快得到结果)，这是因为 SOR 可能不收敛，同时使方法更加通用。起点选择的是：全是 1。三种解方程的迭代方法的代码如下：

```
solveIter <- function(A,b,mode,st,max_iter=5000,omega=1.15) {
  #mode=1/2/3 分别是 Jacobi/GS/SOR，而 st 是迭代的起点
    # 对一些参数的设定
    n=nrow(A);
    if(missing(st)) st=rep(1,n);
    getNewJacobix=function(x) {
        y=numeric(n);
        for(i in 1:n)
        y[i]=b[i]/A[i,i]-sum(A[i,-i]*x[-i])/A[i,i];
        return(y);
    }
    getNewGSx=function(x) {
      for(i in 1:n)
            x[i]=b[i]/A[i,i]-sum(A[i,-i]*x[-i])/A[i,i];
      return(x);
    }
    getNewSORx=function(x) {
              for(i in 1:n)
            x[i]=x[i]*(1-omega)+omega*(b[i]/A[i,i]-sum(A[i,-i]*x[-i])/A[i,i]);
```

```
                return(x);
    }
    getnewx=function(x)
      switch(mode,getNewJacobix(x),getNewGSx(x),getNewSORx(x))
x=st;
y=getnewx(x);
count=1;
while(any(round(x,3)!=round(y,3))) {
      x=y;
    y=getnewx(x);
    count=count+1;
    if(count>max_iter) {
      #error('不收敛');
      return(-1) #-1 表示有问题
    }
}
return(list(x=y,count=count));
}
```

用于产生题目中的矩阵 A 和 b 以及准确解的代码如下

```
generateA=function(ep,a,n) {
  # 用于生成方程的 A,b 以及正确的解
   h=1/n
   A=diag(rep(-(2*ep+h),n-1));
   A[(row(A)-col(A))==1]<-ep
   A[(row(A)-col(A))==-1]<-ep+h
   b=rep(rep(a*h*h,n-1));
   b[n-1]=b[n-1]-ep-h;
   x=(1:(n-1))/n;
   correct=(1-a)/(1-exp(-1/ep))*(1-exp(-x/ep))+a*x;
   return(list(A=A,b=b,correct=correct))
}
```

由于题目中 会变化，而 n 和 a 不变。下面这个函数对 进行了封装，计算 a=0.5,n=100，输入 和所用的迭代方法，返回误差和迭代次数。

```
error_count<-function(A,mode,omega){
  # 其中 A 是 generateA 的返回，此函数作用返回误差和迭代次数
  if(missing(omega)) {
    sol=solveIter(A$A,A$b,mode = mode)
```

```
  } else {
    sol=solveIter(A$A,A$b,mode = mode,omega=omega)
  }
  if(class(sol)=="list")
  return(list(error=max(abs((sol$x-A$correct)/A$correct)),count=sol$count))
  else
    return(list(error=-1,count=-1))
  }
testEp<-function(ep,mode) {
  A=generateA(ep,0.5,100)
  if(missing(mode)) {
    return(sapply(1:3,error_count,A=A))
  } else {
    return(error_count(A,mode))
  }
}
```

(1)

得到的 Jacobi,GS,SOR 分别如下，这里 SOR 的 取值为 1.15

```
testEp(ep=1)
```

```
##         [,1]      [,2]      [,3]
## error 0.440734 0.2477789 0.2236505
## count 2700     1907      1484
```

SOR 的表现是最好的，其次是 G-S，Jacobi 表现最差。而且迭代次数不与误差成反比，Jacobi 误差最大，迭代次数最多，SOR 迭代次数最少，但误差是最小的。这符合所学：G-S 迭代更快，SOR 在大部分时候都比两种迭代法迭代次数少。

(2)

```
testEp(ep=0.1)
```

```
##         [,1]      [,2]     [,3]
## error 0.124345 0.043093 0.03767969
## count 1493     1035     793
```

```
testEp(ep=0.01)
```

```
##         [,1]      [,2]     [,3]
```

```
## error 0.199564 0.199448 0.2023387
## count 360      225      183
```

```r
testEp(ep=0.0001)
```

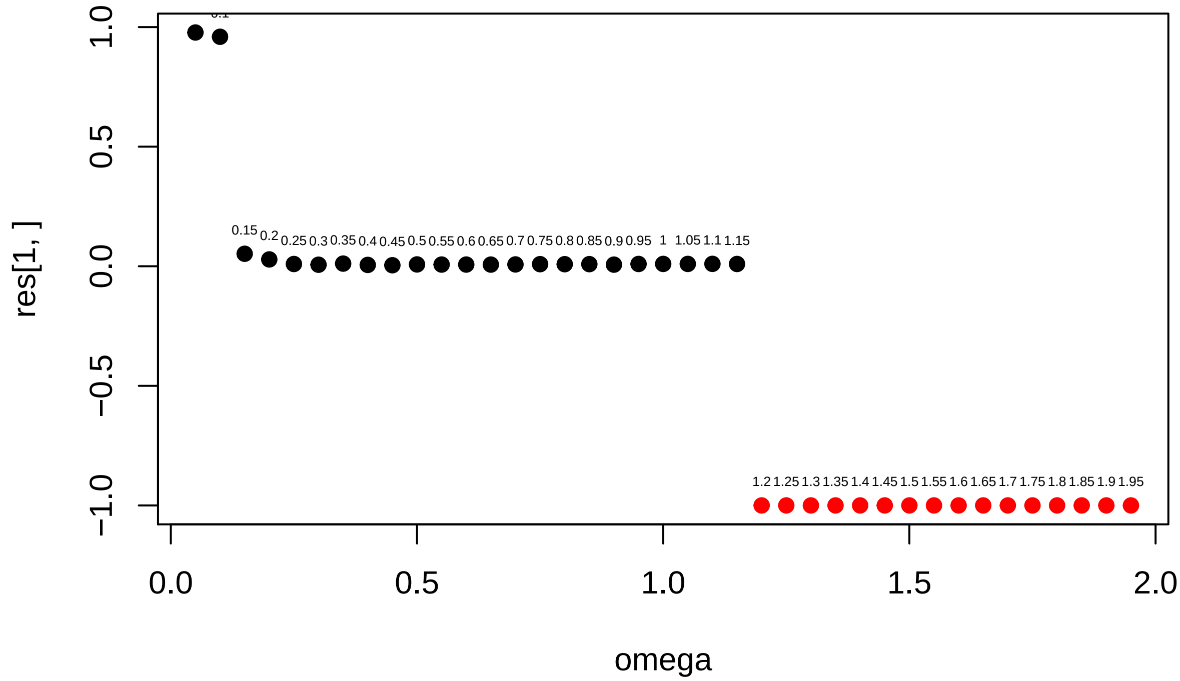
```
##       [,1]        [,2]        [,3]
## error 0.009521098 0.009755514 0.009543466
## count 106         103         154
```

纵向对比：随着 减小，三种方法的迭代次数都在不断减小，但是误差并不是随 单减的 (=0.0 时 1 的误差大于 =0.1)。 =0.01 处的误差 >0.1 处的误差。横向对比：在 =1 和 0.1 的时候 (比较大的时候)，Jacobi 表现得最差，误差和迭代次数都是最大的。但是当 更小 (=0.01 以及 0.0001) 时，三种方法的误差就差不多了。G-S 的迭代次数总是比 Jacobi 更少。但 SOR 与它们比起来，就很不确定了。当 =1 和 0.1 时，无论是迭代次数还是误差，SOR 都是很好的，但是当 =0.0001，SOR 的迭代次数最大。

SOR 的效果与 的关系

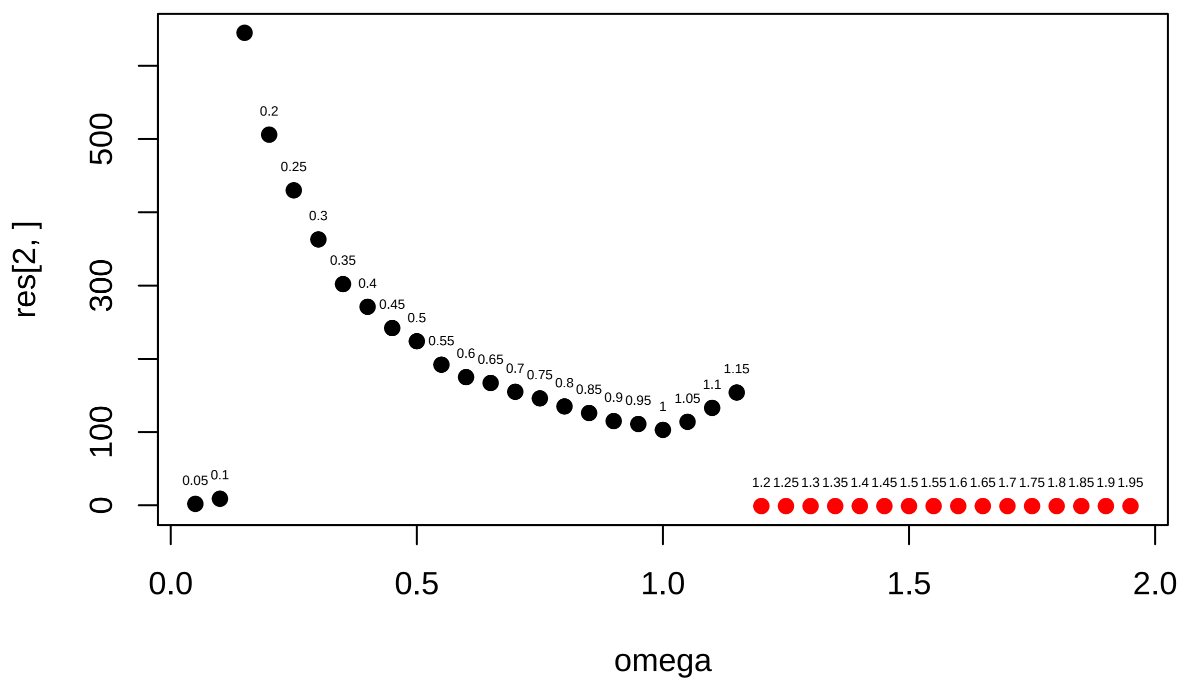
```r
A=generateA(0.0001,0.5,100) # 由于事实上也可变，可以在外面再做一次封装。
# 需要测试的 omega
omega=seq(0.05,1.95,by=0.05)
res<-sapply(omega,function(ome)error_count(A=A,mode=3,omega = ome))
```

```r
getcol<-function(v) {
  #v 是一个向量
  col=rep(1,length(v))
  col[v==-1]=2 #2 是红色
  return(col)
}
plot(x=omega,res[1,],col=getcol(res[1,]),pch=19)
text(omega,res[1,],omega,pos = 3,cex=0.4)
```



图中红色的点表示不收敛，可以看到， =0.0001 时，当 在 [0.05,1.15] 区间上的时候，是收敛的，但是对于 <0.15 的 ，误差很大。>=1.2 则完全不收敛。至于迭代次数

```r
plot(x=omega,res[2,],col=getcol(res[2,]),pch=19)
text(omega,res[2,],omega,pos = 3,cex=0.4)
```



可以看到，之所以 0.05 和 0.1 的误差那么大，是因为迭代次数太少。迭代次数随 先增后降，在 1.0 时迭代次数最小，此时也就是 G-S 方法。

通过这个实验，看到 SOR 的收敛性与 有很大的关系，看来 SOR 并不是很可靠，有时候 SOR 的表现是最优的，但一般情况我会选择 G-S 方法，它的收敛性可靠 (相比 SOR)，而且迭代次数和误差通常优于 Jacobi。