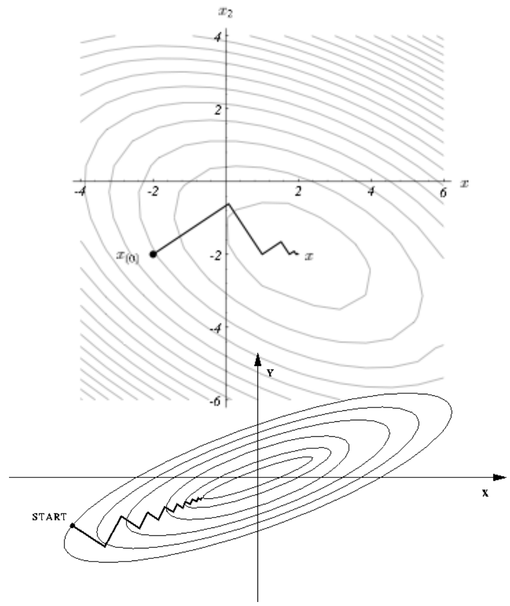
ME/AerE 546

Conjugate Gradient

see p.163 Pletcher, et al.

Search Directions



The idea is explained by steepest descents; but, steepest descents is not a good method! Preconditioned Conjugate gradient is summarized on page 3.

The discretized Poisson equation is written in Δ form

$$\boldsymbol{A} \cdot \Delta\psi = R$$

where
$$R = \omega - A \cdot \psi$$

Solving the problem can be posed as: *Find* $\min_\psi |R|$.

If there is a solution, the minimum is 0. For the Laplacian (or, a symmetric positive definite A), it is equivalent to: *Minimize*
$$F = ({}^1\!/_2\, \psi \cdot \boldsymbol{A} \cdot \psi - \omega \cdot \psi)$$

Note $\nabla_\psi F = -\boldsymbol{R}$, so the minimum of F solves $\boldsymbol{R} = 0$. The steepest descent method consists of searching down the gradient of F, which is $-R$:

$$\boldsymbol{d} = \alpha \nabla_\psi F = -\alpha \boldsymbol{R}$$

Start with in initial guess, $\psi = \psi_0$; $\boldsymbol{R}_0 = \omega - A \cdot \psi_0$. Update to new guess as

$$\psi_1 = \psi_0 - \alpha_0 \boldsymbol{R}_0$$

iterate as

$$\psi_n = \psi_{n-1} - \alpha_{n-1} \boldsymbol{R}_{n-1} \tag{1}$$

The only thing left is choosing α. That is selected to minimize F for each step of the search:

$$\begin{aligned} F_n &= {}^1\!/_2\, (\psi_{n-1} - \alpha_{n-1}\boldsymbol{R}_{n-1}) \cdot \boldsymbol{A} \cdot (\psi_{n-1} - \alpha_{n-1}\boldsymbol{R}_{n-1}) - \omega \cdot (\psi_{n-1} - \alpha_{n-1}\boldsymbol{R}_{n-1}) \\ &= \underbrace{{}^1\!/_2\, \psi_{n-1} \cdot \boldsymbol{A} \cdot \psi_{n-1} - \omega \cdot \psi_{n-1}}_{const} - \underbrace{\alpha_{n-1}\boldsymbol{R}_{n-1} \cdot (\boldsymbol{A} \cdot \psi_{n-1} - \omega)}_{linear} + \underbrace{{}^1\!/_2\, \alpha_{n-1}^2 \boldsymbol{R}_{n-1} \cdot \boldsymbol{A} \cdot \boldsymbol{R}_{n-1}}_{quadratic} \qquad (2) \\ &= F_{n-1} + \alpha_{n-1}\boldsymbol{R}_{n-1} \cdot \boldsymbol{R}_{n-1} + {}^1\!/_2\, \alpha_{n-1}^2 \boldsymbol{R}_{n-1} \cdot \boldsymbol{A} \cdot \boldsymbol{R}_{n-1} \end{aligned}$$

using the fact that A is symmetric. Then $dF_n/d\alpha = 0$ gives

$$\alpha_{n-1} = -\frac{\boldsymbol{R}_{n-1} \cdot \boldsymbol{R}_{n-1}}{\boldsymbol{R}_{n-1} \cdot \boldsymbol{A} \cdot \boldsymbol{R}_{n-1}} \tag{3}$$

The algorithm is get α from (3), update ψ from (1). Then update the residual; which can be done as $\quad \boldsymbol{R}_n = \boldsymbol{R}_{n-1} + \alpha_{n-1}\boldsymbol{A} \cdot \boldsymbol{R}_{n-1}$ (4)

algorthm:

`WHILE |R|>`ε

$\alpha_{n-1} = -\boldsymbol{R}_{n-1} \cdot \boldsymbol{R}_{n-1} / \boldsymbol{R}_{n-1} \cdot \boldsymbol{A} \cdot \boldsymbol{R}_{n-1}$[†]

$\psi_n = \psi_{n-1} - \alpha_{n-1} \boldsymbol{R}_{n-1}$

$\boldsymbol{R}_n = \boldsymbol{R}_{n-1} + \alpha_{n-1} \boldsymbol{A} \cdot \boldsymbol{R}_{n-1}$

$n = n + 1$

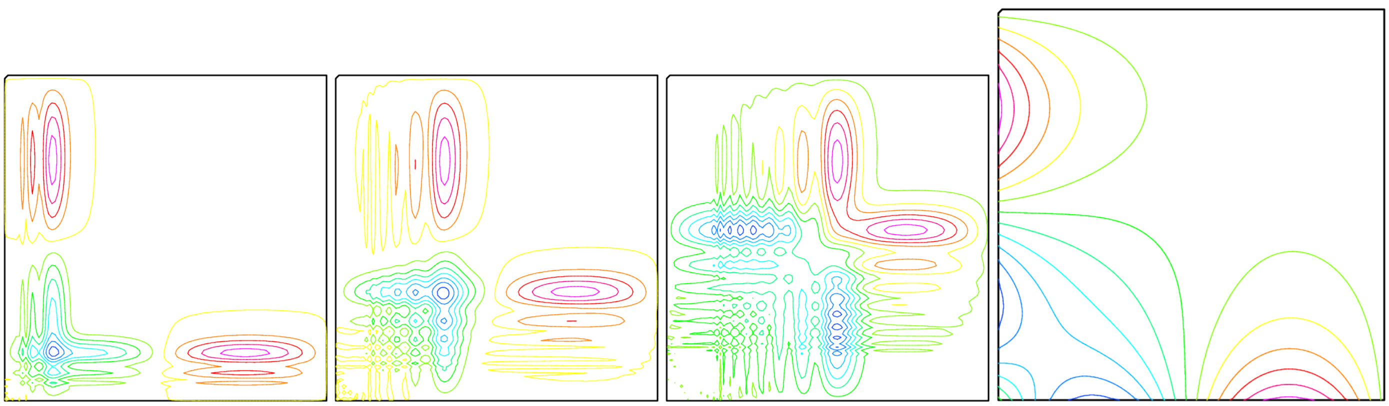
`END WHILE`

Multiplication by A is the finite difference formula for the Laplacian $\delta_x^2 R + \delta_y^2 R$, not a matrix multiplication, *per se*[†] (which would be $O(N^2)$)

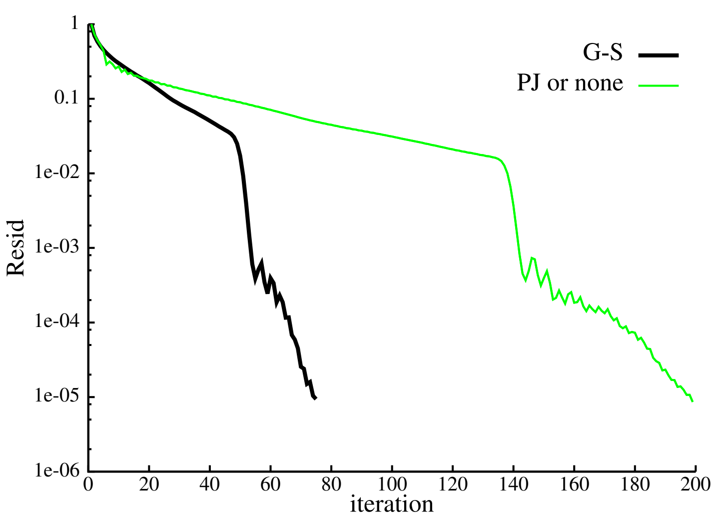
Note that search directions are **orthogonal** $\boldsymbol{d}_n \cdot \boldsymbol{d}_{n-1} = \alpha_{n-1}\alpha_n[\boldsymbol{R}_{n-1}\boldsymbol{R}_{n-1} - \alpha_{n-1}\boldsymbol{R}_{n-1}\boldsymbol{A}\cdot\boldsymbol{R}_{n-1}] = 0$. But $\boldsymbol{d}_n \cdot \boldsymbol{d}_{n-2} \neq 0$

In conjugate gradient they are made **conjugate** $\boldsymbol{d}_n \cdot \boldsymbol{A} \cdot \boldsymbol{d}_{n-1} = 0$. Then it can be shown that $\boldsymbol{d}_n \cdot \boldsymbol{A} \cdot \boldsymbol{d}_{n-i} = 0, i > 0$.

Pre-condition: Replace residual by $Z^n = \widetilde{A}^{-1} \cdot R^{n-1}$; this means one Gauss-Seidel (or SGS) sweep, then find $\min(Z)$



10^{th} ε=0.024 20^{th} ε=0.0058 30^{th} search vectors ε=0.0015 Solution $\Psi(x,y)$



[†] $[A \cdot P](j,k) = A_1 * P(j-1,k) + A_2 * P(j+1,k) + A_3 * P(j,k) + A_4 * P(j,k-1) + A_5 * P(j,k+1)$

Preconditioned C-G

Discrete equation is

$$\boldsymbol{A} \cdot \Delta\psi = \boldsymbol{R}; \quad \boldsymbol{R} = \omega - \boldsymbol{A} \cdot \psi$$

N.B.: Initialize all arrays to zero then impose boundary conditions. Evaluate residual from initial guess $\boldsymbol{R}^0 = \omega - \boldsymbol{A} \cdot \psi^0$; if latter is zero $\boldsymbol{R}^0 = \omega$. Also, start with $s^0 = 10^{15}$, i.e., very large; $\boldsymbol{p}^0 = 0$.

Pseudo-code for Preconditioned C-G algorithm:

While $\varepsilon <$ *tolerance* Do

Pre-condition: $\boldsymbol{z}^n = \widetilde{A}^{-1}\boldsymbol{R}^{n-1}$; this means one Gauss-Seidel (or SGS) sweep (replace $\boldsymbol{z}$ by $\boldsymbol{R}$ for C-G without preconditioning)

$s^n = \boldsymbol{R}^{n-1} \cdot \boldsymbol{z}^n$
$\beta^n = s^n / s^{n-1}$
$\boldsymbol{p} = \boldsymbol{z}^n + \beta^n \boldsymbol{p}$! $\boldsymbol{p}^n = \boldsymbol{z}^n + \beta^n \boldsymbol{p}^{n-1}$; but, no need to save old search directions

$\alpha^n = s^n / \boldsymbol{p} \cdot \boldsymbol{A} \cdot \boldsymbol{p}$
$\boldsymbol{\psi}^n = \boldsymbol{\psi}^{n-1} + \alpha^n \boldsymbol{p}$! update solution
$\boldsymbol{R}^n = \boldsymbol{R}^{n-1} - \alpha^n \boldsymbol{A} \cdot \boldsymbol{p}$! update residual

Evaluate error ε

End While

Example:

```
  ψ(:,1)  = BCx ; ψ(:,NK) = BCx ; ψ(1,:)  = BCy ; ψ(NJ,:) = BCy
Coefficients of 5-point stencil (A must be symmetric, positive definite)
  A1 = 1/Δy^2  ;   A2 = 1/Δx^2  ;   A4 = 1/Δx^2  ;    A5 = 1/Δy^2
  A3 = -(A1+A2+A4+A5)
Initialize residual (zero on boundaries)
  R(2:NJ-1,2:NK-1) = ω(2:NJ-1,2:NK-1)  !  Residual=ω because ψ0 = 0
  sk = 1.e15  ; iter = 0
While(err.gt.errmx .and.iter.lt.ITMAX)
 Z(:,:)  = 0.0; iter = iter+1
Precondition residual
IF(GS)THEN                                  ! Symmetric Gauss-Seidel
DO k=2,NK-1
 DO j=2,NJ-1
    Z(j,k) = (R(j,k) -A2*Z(j-1,k)-A1*Z(j,k-1) )/A3
 ENDDO
ENDDO
DO k=NK-1,2,-1
 DO j=NJ-1,2,-1
   Z(j,k) = Z(j,k) -(A4*Z(j+1,k)+A5*Z(j,k+1))/A3
 ENDDO
ENDDO
ELSEIF(PJ)THEN                              ! Point Jacobi
  Z(2:NJ-1,2:NK-1) = R(2:NJ-1,2:NK-1)/A3
```

```
ELSE                                          ! No Preconditioner
  Z(2:NJ-1,2:NK-1) = R(2:NJ-1,2:NK-1)
ENDIF
     Conjugate Gradient
  skm = sk    !  Save previous value
  sk  = Z*R   ! N.B. this is the inner product of vectors
  β   = sk/skm
Next conjugate search direction
   P(2:NJ-1,2:NK-1) = Z(2:NJ-1,2:NK-1)+β*P(2:NJ-1,2:NK-1)
Matrix multiplication:  note matrix multiplication=O(N) ops
 DO k=2,NK-1
   DO j=2,NJ-1
     AP(j,k) = A_2*P(j-1,k)+A_4*P(j+1,k)+A_3*P(j,k)+A_1*P(j,k-1)+A_5*P(j,k+1)
   ENDDO
 ENDDO
 den = dP*AP      ! N.B. this is the inner product of vectors
 α   = sk/den
Advance solution and residual: (eliminates P component of R)
ψ(:,:)  = ψ(:,:)+α*P(:,:)
R(:,:)  = R(:,:) - α*AP(:,:)

Monitor convergence
   err = 0.  ;  pnorm = 0.
  DO k=2,NK-1
    DO j=2,NJ-1
     err = err+P(j,k)**2; pnorm = pnorm+ψ(j,k)**2
    ENDDO
   ENDDO
    err = sqrt(err/pnorm) ;   PRINT *,iter,err
EndWhile

Plot solution
```