libcKrylov algorithm collection

Pengliang Yang
Harbin Institute of Technology, China
Email: ypl.2100@gmail.com

June 25, 2023

Algorithm 1 Conjugate gradient algorithm for solving $Ax = b$ ($A^H = A$) (Shewchuk, 1994)

$r_0 := b - Ax_0$
$p_0 := r_0$
for $k = 0, 1, \cdots$ until convergence **do**
  $\alpha_k = \frac{r_k^H r_k}{p_k^H A p_k}$
  $x_{k+1} = x_k + \alpha_k p_k$
  $r_{k+1} = r_k - \alpha_k A p_k$
  $\beta_k = \frac{r_{k+1}^H r_{k+1}}{r_k^H r_k}$
  $p_{k+1} = r_{k+1} + \beta_k p_k$
end for

Algorithm 2 CGNR for solving $A^H Ax = A^H b$ (Saad, 2003, algorithm 8.4)

$r_0 = b - Ax_0$
$z_0 = A^H r_0$
$p_0 = z_0$
for $k = 0, 1, \cdots$ until convergence **do**
  $w_k = Ap_k$
  $\alpha_k = (z_k, z_k)/(w_k, w_k)$
  $x_{k+1} = x_k + \alpha_k p_k$
  $r_{k+1} = r_k - \alpha_k w_k$
  $z_{k+1} = A^H r_{k+1}$
  $\beta_k = (z_{k+1}, z_{k+1})/(z_k, z_k)$
  $p_{k+1} = z_{k+1} + \beta_k p_k$
end for

Algorithm 3 CGNE (Craig's method) for solving $AA^H y = b, x = A^H y$(Saad, 2003, algorithm 8.5)

$r_0 = b - Ax_0$
$p_0 = A^H r_0$
for $k = 0, 1, \cdots$ until convergence **do**
  $\alpha_k = (r_k, r_k)/(p_k, p_k)$
  $x_{k+1} = x_k + \alpha_k p_k$
  $r_{k+1} = r_k - \alpha_k A p_k$
  $\beta_k = (r_{k+1}, r_{k+1})/(r_k, r_k)$
  $p_{k+1} = A^H r_{k+1} + \beta_k p_k$
end for

Algorithm 4 Preconditioned Conjugate gradient algorithm for solving $Ax = b$ (Saad, 2003, algorithm 9.1)

$r_0 := b - Ax_0 = b$
$z_0 := M^{-1} r_0$
$p_0 := z_0$
for $k = 0, 1, \cdots$ until convergence **do**
    $\alpha_k = \frac{r_k^H p_k}{p_k^H A p_k}$
    $x_{k+1} = x_k + \alpha_k p_k$
    $r_{k+1} = r_k - \alpha_k A p_k$
    $z_{k+1} := M^{-1} r_{k+1}$
    $\beta_k = \frac{r_{k+1}^H z_{k+1}}{r_k^H z_k}$
    $p_{k+1} = z_{k+1} + \beta_k p_k$
end for

Algorithm 5 BiCGStab (Chen et al., 2016), improved version from (Van der Vorst, 1992)

$r_0 = b - Ax_0$, $\tilde{r}_0$ arbitrary but $(\tilde{r}_0, r_0) \neq 0$
$p_0 = r_0$
for $k = 0, 1, \cdots$ until convergence **do**
    $\alpha_k = (r_k, \tilde{r}_0)/(Ap_k, \tilde{r}_0)$
    $s_k = r_k - \alpha_k A p_k$
    $\omega_k = (As_k, s_k)/(As_k, As_k)$
    $x_{k+1} = x_k + \alpha_k p_k + \omega_k s_k$
    $r_{k+1} = s_k - \omega_k A s_k$
    $\beta_k = (r_{k+1}, \tilde{r}_0)/(r_k, \tilde{r}_0) \cdot \alpha_k/\omega_k$
    $p_{k+1} = r_{k+1} + \beta_k (p_k - \omega_k A p_k)$
end for

Algorithm 6 BiCGStab with right preconditioning (Flexible BiCGStab) (Chen et al., 2016)

$r_0 = b - Ax_0$, $\tilde{r}_0$ arbitrary but $(\tilde{r}_0, r_0) \neq 0$
$p_0 = r_0$
for $k = 0, 1, \cdots$ until convergence **do**
    $\tilde{p}_k = M^{-1} p_k$
    $\alpha_k = (r_k, \tilde{r}_0)/(A\tilde{p}_k, \tilde{r}_0)$
    $s_k = r_k - \alpha_k A \tilde{p}_k$
    $\tilde{s}_k = M^{-1} s_k$
    $\omega_k = (A\tilde{s}_k, s_k)/(A\tilde{s}_k, A\tilde{s}_k)$
    $x_{k+1} = x_k + \alpha_k p_k + \omega_k s_k$
    $r_{k+1} = s_k - \omega_k A \tilde{s}_k$
    $\beta_k = (r_{k+1}, \tilde{r}_0)/(r_k, \tilde{r}_0) \cdot \alpha_k/\omega_k$
    $p_{k+1} = r_{k+1} + \beta_k (p_k - \omega_k A \tilde{p}_k)$
end for

Algorithm 7 GMRES (Saad, 2003, algorithm 6.9)

$r_0 := b - Ax_0 = b$
$\beta = \|r_0\|_2$
$v_1 := r_0/\beta$
for $j = 1, \cdots, m$ **do**
  compute $w_j = Av_j$
  for $i = 1, \cdots, j$ **do**
    $h_{ij} = (w_j, v_i)$
    $w_j = w_j - h_{ij}v_i$
  end for
  $h_{j+1,j} = \|w_j\|_2$. If $h_{j+1,j} = 0$, set $m = j$ and go to
  $v_{j+1} = w_j/h_{j+1,j}$
  solve least-squares problem $\min_y \| \begin{bmatrix} \beta \\ 0 \\ \cdots \\ 0 \end{bmatrix} - \tilde{H}_m y\|_2$ by Givens rotation
end for
$x_m = x_0 + V_m y_m$

Algorithm 8 GMRES with right preconditioning (Flexible GMRES)(Saad, 2003, algorithm 9.5)

$r_0 := b - Ax_0 = b$
$\beta = \|r_0\|_2$
$v_1 := r_0/\beta$
for $j = 1, \cdots, m$ **do**
  compute $w_j = AM^{-1}v_j$
  for $i = 1, \cdots, j$ **do**
    $h_{ij} = (w_j, v_i)$
    $w_j = w_j - h_{ij}v_i$
  end for
  $h_{j+1,j} = \|w_j\|_2$. If $h_{j+1,j} = 0$, set $m = j$ and go to
  $v_{j+1} = w_j/h_{j+1,j}$
  solve least-squares problem $\min_y \| \begin{bmatrix} \beta \\ 0 \\ \cdots \\ 0 \end{bmatrix} - \tilde{H}_m y\|_2$ by Givens rotation
end for
$x_m = x_0 + M^{-1}V_m y_m$

References

Chen, J., McInnes, L. C., and Zhang, H. (2016). Analysis and practical use of flexible bicgstab. *Journal of Scientific Computing*, 68(2):803–825.

Saad, Y. (2003). *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia.

Shewchuk, J. R. (1994). An introduction to the conjugate gradient method without the agonizing pain. Technical Report Computer Science Technical Report CMU-CS-94-125, School of computer science, Carnegie Mellon University.

Van der Vorst, H. A. (1992). Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM Journal on scientific and Statistical Computing*, 13(2):631–644.