Lecture 8: Fast Linear Solvers (Part 6)

Nonsymmetric System of Linear Equations

- The CG method requires to A to be an $n \times n$ symmetric positive definite matrix to solve $A\boldsymbol{x} = \boldsymbol{b}$.
- If A is nonsymmetric:
  - Convert the system to a symmetric positive definite one
  - Modify CG to handle general matrices

Normal Equation Approach

The *normal equations* corresponding to $A\boldsymbol{x} = \boldsymbol{b}$ are $A^T A\boldsymbol{x} = A^T\boldsymbol{b}$

- If A is nonsingular then $A^T A$ is symmetric positive definite and the CG method can be applied to solve $A^T A\boldsymbol{x} = A^T\boldsymbol{b}$ (CG normal residual -- CGNR).
- Alternatively, we can first solve $\mathrm{A}A^T\boldsymbol{y} = \boldsymbol{b}$ for $\boldsymbol{y}$, then $\boldsymbol{x} = A^T\boldsymbol{y}$.
- **Disadvantages:**
  - Each iteration requires $A^T A$ or $\mathrm{A}A^T$
  - Condition number of $A^T A$ or $\mathrm{A}A^T$ is square of that of A. However, CG works well if condition number is small

Arnoldi Iteration

- The Arnoldi method is an orthogonal projection onto a Krylov subspace $\mathrm{K}_m(A, \boldsymbol{r}_0)$ for $n \times n$ nonsymmetric matrix A. Here $m \ll n$.
- Arnoldi reduces A to a *Hessenberg* form.

Upper Hessenberg matrix: zero entries below the first subdiagonal.

$$\begin{bmatrix} 2 & 3 & 4 & 1 \\ 2 & 5 & 1 & 9 \\ 0 & 2 & 1 & 2 \\ 0 & 0 & 3 & 2 \end{bmatrix}$$

Lower Hessenberg matrix: zero entries above the first superdiagonal.

$$\begin{bmatrix} 3 & 2 & 0 & 0 \\ 2 & 5 & 1 & 0 \\ 1 & 2 & 1 & 2 \\ 3 & 4 & 3 & 2 \end{bmatrix}$$

Mechanics of Arnoldi Iteration

- For $A \in R^{n\times n}$, a given vector $\boldsymbol{r}_0 \in R^n$ defines a sequence of Krylov subspaces $\mathrm{K}_m(A, \boldsymbol{r}_0)$. Matrix $\mathrm{K}_m = [\boldsymbol{r}_0 | A\boldsymbol{r}_0 | A^2\boldsymbol{r}_0 | \ldots | A^{m-1}\boldsymbol{r}_0] \in R^{n\times m}$ is the corresponding Krylov matrix.
- The Gram-Schmidt procedure for forming an orthonormal basis for K_m is called the Arnoldi process.
  - **Theorem.** The Arnoldi procedure generates a reduced QR factorization of Krylov matrix K_m in the form $\mathrm{K}_m = V_m R_m$ with $V_m \in R^{n\times m}$ and having orthonormal columns and with a triangular matrix $R_m \in R^{m\times m}$. Furthermore, with the $m \times m - upper$ Hessenberg matrix H_m, we have $V_m^T A V_m = H_m$.

Let H_m be a $m \times m$ Hessenberg matrix:

$$H_m = \begin{bmatrix} h_{11} & h_{12} & \dots & h_{1m} \\ h_{21} & h_{22} & \dots & h_{2m} \\ 0 & \ddots & \ddots & \vdots \\ 0 & \dots & h_{m,m-1} & h_{mm} \end{bmatrix}$$

Let $(m+1) \times m$ $\overline{H}_m$ be the extended matrix of H_m:

$$\overline{H}_m = \begin{bmatrix} h_{11} & h_{12} & \dots & h_{1m} \\ h_{21} & h_{22} & \dots & h_{2m} \\ 0 & \ddots & \ddots & \vdots \\ 0 & \dots & h_{m,m-1} & h_{mm} \\ 0 & \dots & 0 & h_{m+1,m} \end{bmatrix}$$

The Arnoldi iteration produces matrices V_m, V_{m+1} and $\overline{H}_m$ for matrix A satisfying:

$$AV_m = V_{m+1}\overline{H}_m = V_m H_m + \boldsymbol{w}_m \boldsymbol{e}_m^T$$

Here V_m, V_{m+1} have orthonormal columns

$$V_m = [\boldsymbol{v}_1 | \boldsymbol{v}_2 | \dots | \boldsymbol{v}_m], \quad V_{m+1} = [\boldsymbol{v}_1 | \boldsymbol{v}_2 | \dots | \boldsymbol{v}_m | \boldsymbol{v}_{m+1}]$$

The mth column of the equation:

$$\begin{aligned} A\boldsymbol{v}_m = {} & h_{1m}\boldsymbol{v}_1 + h_{2m}\boldsymbol{v}_2 + \cdots + h_{mm}\boldsymbol{v}_m \\ & + h_{m+1,m}\boldsymbol{v}_{m+1} \end{aligned}$$

Therefore,

$$h_{1m} = A\boldsymbol{v}_m \cdot \boldsymbol{v}_1$$

$$\vdots$$

$$h_{m+1,m} = ||A\boldsymbol{v}_m - h_{1m}\boldsymbol{v}_1 \ldots - h_{mm}\boldsymbol{v}_m||$$

$$\boldsymbol{v}_{m+1} = (A\boldsymbol{v}_m - h_{1m}\boldsymbol{v}_1 \ldots - h_{mm}\boldsymbol{v}_m)/h_{m+1,m}$$

Arnoldi Algorithm

$$v_1 = r_0/\|r_0\|_2$$

$$\begin{aligned}
w_1 &= A\,v_1 - (A\,v_1, v_1)\,v_1, && v_2 = w_1/\|w_1\|_2 \\
&\vdots \\
w_j &= A\,v_j - (A\,v_j, v_1)\,v_1 - \ldots - (A\,v_j, v_j)\,v_j, && v_{j+1} = w_j/\|w_j\|_2 \\
&\vdots \\
w_m &= A\,v_m - (A\,v_m, v_1)\,v_1 - \ldots - (A\,v_m, v_m)\,v_m, && v_{m+1} = w_m/\|w_m\|_2
\end{aligned}$$

```
Choose $\boldsymbol{r}_0$ and let $\boldsymbol{v}_1 = \boldsymbol{r}_0/||\boldsymbol{r}_0||$
**for** $j = 1, \ldots, m-1$
    $\boldsymbol{w} = A\boldsymbol{v}_j - \sum_{i=1}^{j}((A\boldsymbol{v}_j)^T \boldsymbol{v}_i)\boldsymbol{v}_i$
    $\boldsymbol{v}_{j+1} = \boldsymbol{w}/||\boldsymbol{w}||_2$
**endfor**
```

Remark: Choose $\boldsymbol{v}_1$. Then for $j = 1, \ldots, m-1$, first multiply the current Arnoldi vector $\boldsymbol{v}_j$ by A, and orthonormalize $A\boldsymbol{v}_j$ against all previous Arnoldi vectors.

$$V_m = \left(\begin{array}{c|c|c|c} | & | & & | \\ | & | & & | \\ v_1 & v_2 & \vdots & v_m \\ | & | & & | \\ | & | & & | \end{array} \right) \in \mathbb{R}^{n\times m}$$

$$\bar{H}_m = \begin{pmatrix} h_{11} & h_{12} & h_{13} & \dots & h_{1m} \\ h_{21} & h_{22} & h_{23} & \dots & h_{2m} \\ & h_{32} & h_{33} & \dots & h_{3m} \\ & & \ddots & \ddots & \vdots \\ & & & h_{m,m-1} & h_{mm} \\ & & & & h_{m+1,m} \end{pmatrix} = \begin{pmatrix} (Av_1, v_1) & (Av_2, v_1) & (Av_3, v_1) & \dots & (Av_m, v_1) \\ (Av_1, v_2) & (Av_2, v_2) & (Av_3, v_2) & \dots & (Av_m, v_2) \\ & (Av_2, v_3) & (Av_3, v_3) & \dots & (Av_m, v_3) \\ & & \ddots & \ddots & \vdots \\ & & & (Av_{m-1}, v_m) & (Av_m, v_m) \\ & & & & (Av_m, v_{m+1}) \end{pmatrix}$$

- $V_m^T V_m = I_{m\times m}$.
- If Arnoldi process breaks down at mth step, $\boldsymbol{w}_m = \boldsymbol{0}$ is still well-defined but not $\boldsymbol{v}_{m+1}$, and the algorithm stop.
- In this case, the last row of $\overline{H}_m$ is set to zero, $h_{m+1,m} = 0$

Stable Arnoldi Algorithm

```
Choose $\boldsymbol{x}_0$ and let $\boldsymbol{v}_1 = \boldsymbol{x}_0/||\boldsymbol{x}_0||$ .
for $j = 1, \dots, m$
    $\boldsymbol{w} = A\boldsymbol{v}_j$
    for $i = 1, \dots, j$
        $h_{ij} =< \boldsymbol{w}, \boldsymbol{v}_i >$
        $\boldsymbol{w} = \boldsymbol{w} - h_{ij}\boldsymbol{v}_i$
    endfor
    $h_{j+1,j} = ||\boldsymbol{w}||_2$
    if $h_{j+1,j} = 0$, then stop
    $\boldsymbol{v}_{j+1} = \boldsymbol{w}/h_{j+1,j}$
endfor
```

Generalized Minimum Residual (GMRES) Method

Let the Krylov space associated with $A\boldsymbol{x} = \boldsymbol{b}$ be $\mathrm{K}_k(A, \boldsymbol{r}_0) = span\{\boldsymbol{r}_0, A\boldsymbol{r}_0, A^2\boldsymbol{r}_0, \ldots, A^{k-1}\boldsymbol{r}_0\}$, where $\boldsymbol{r}_0 = \boldsymbol{b} - A\,\boldsymbol{x}_0$ for some initial guess $\boldsymbol{x}_0$.

The kth ($k \geq 1$) iteration of GMRES is the solution to the least squares problem:

$$minimize_{\boldsymbol{x}\in\boldsymbol{x}_0+\mathrm{K}_k} ||\boldsymbol{b} - A\boldsymbol{x}||_2, \text{ i.e.}$$

Find $\boldsymbol{x}_k \in \boldsymbol{x}_0 + \mathrm{K}_k$ such that

$$||\boldsymbol{b} - A\boldsymbol{x}_k||_2 = min_{\boldsymbol{x}\in\boldsymbol{x}_0+\mathrm{K}_k} ||\boldsymbol{b} - A\boldsymbol{x}||_2$$

- Remark: the GMRES was proposed in "Y. Saad and M. Schultz, *GMRES a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869."

If $\boldsymbol{x} \in \boldsymbol{x}_0 + \mathrm{K}_k$, then $\boldsymbol{x} = \boldsymbol{x}_0 + \sum_{j=0}^{k-1} \gamma_j A^j \boldsymbol{r}_0$.

So $\boldsymbol{b} - A\boldsymbol{x} = \boldsymbol{b} - A\boldsymbol{x}_0 - \sum_{j=0}^{k-1} \gamma_j A^{j+1} \boldsymbol{r}_0 = \boldsymbol{r}_0 - \sum_{j=1}^{k} \gamma_{j-1} A^j \boldsymbol{r}_0$.

- Theorem (Kelly). *Let A be a nonsingular diagonalizable matrix. Assume that A has only k distinct eigenvalues. Then GMRES will terminate in at most k iterations.*
- Least Square via QR factorization

  Let $A \in R^{m\times n}$ $(m \geq n)$, and $\boldsymbol{b} \in R^m$ be given. Find $\boldsymbol{x} \in R^n$ so that the norm of $\boldsymbol{r} = \boldsymbol{b} - A\boldsymbol{x}$ is minimized.

  Algorithm

  1. Compute the QR factorization $A = \hat{Q}\hat{R}$
  2. Compute vector $\hat{Q}^*\boldsymbol{b}$
  3. Solve the upper triangular system $\hat{R}\boldsymbol{x} = \hat{Q}^*\boldsymbol{b}$ for $\boldsymbol{x}$

  Reference: Numerical Linear Algebra, L.N. Trefethen, D. Bau, III

GMRES Implementation

- The kth ($k \geq 1$) iteration of GMRES is the solution to the least squares problem:

$$minimize_{\boldsymbol{x} \in \boldsymbol{x}_0 + \mathrm{K}_k} ||\boldsymbol{b} - A\boldsymbol{x}||_2$$

- Suppose we have used Arnoldi process constructed an orthogonal basis V_k for $\mathrm{K}_k(A, \boldsymbol{r}_0)$.
  - $\boldsymbol{r}_0 = \beta V_k \boldsymbol{e}_1$, where $\boldsymbol{e}_1 = (1,0,0,\dots)^T$, $\beta = ||\boldsymbol{r}_0||_2$
  - Any vector $\boldsymbol{z} \in \mathrm{K}_k(A, \boldsymbol{r}_0)$ can be written as $\boldsymbol{z} = \sum_{l=1}^{k} y_l \boldsymbol{v}_l^k$, where $\boldsymbol{v}_l^k$ is the lth column of V_k. Denote $\boldsymbol{y} = (y_1, y_2, \dots, y_k)^T \in R^k$.

$$\boldsymbol{z} = V_k \boldsymbol{y}$$

Since $\boldsymbol{x} - \boldsymbol{x}_0 = V_k \boldsymbol{y}$ for some coefficient vector $\boldsymbol{y} \in R^k$, we must have $\boldsymbol{x}_k = \boldsymbol{x}_0 + V_k \boldsymbol{y}$ where $\boldsymbol{y}$ minimizes $||\boldsymbol{b} - A(\boldsymbol{x}_0 + V_k \boldsymbol{y})||_2 = ||\boldsymbol{r}_0 - AV_k \boldsymbol{y}||_2$.

- The kth ($k \geq 1$) iteration of GMRES now is equivalent to a least squares problem in R^k, i.e.

$$minimize_{\boldsymbol{x} \in \boldsymbol{x}_0 + \mathrm{K}_k} ||\boldsymbol{b} - A\boldsymbol{x}||_2$$
$$= minimize_{y \in R^k} ||\boldsymbol{r}_0 - AV_k \boldsymbol{y}||_2$$

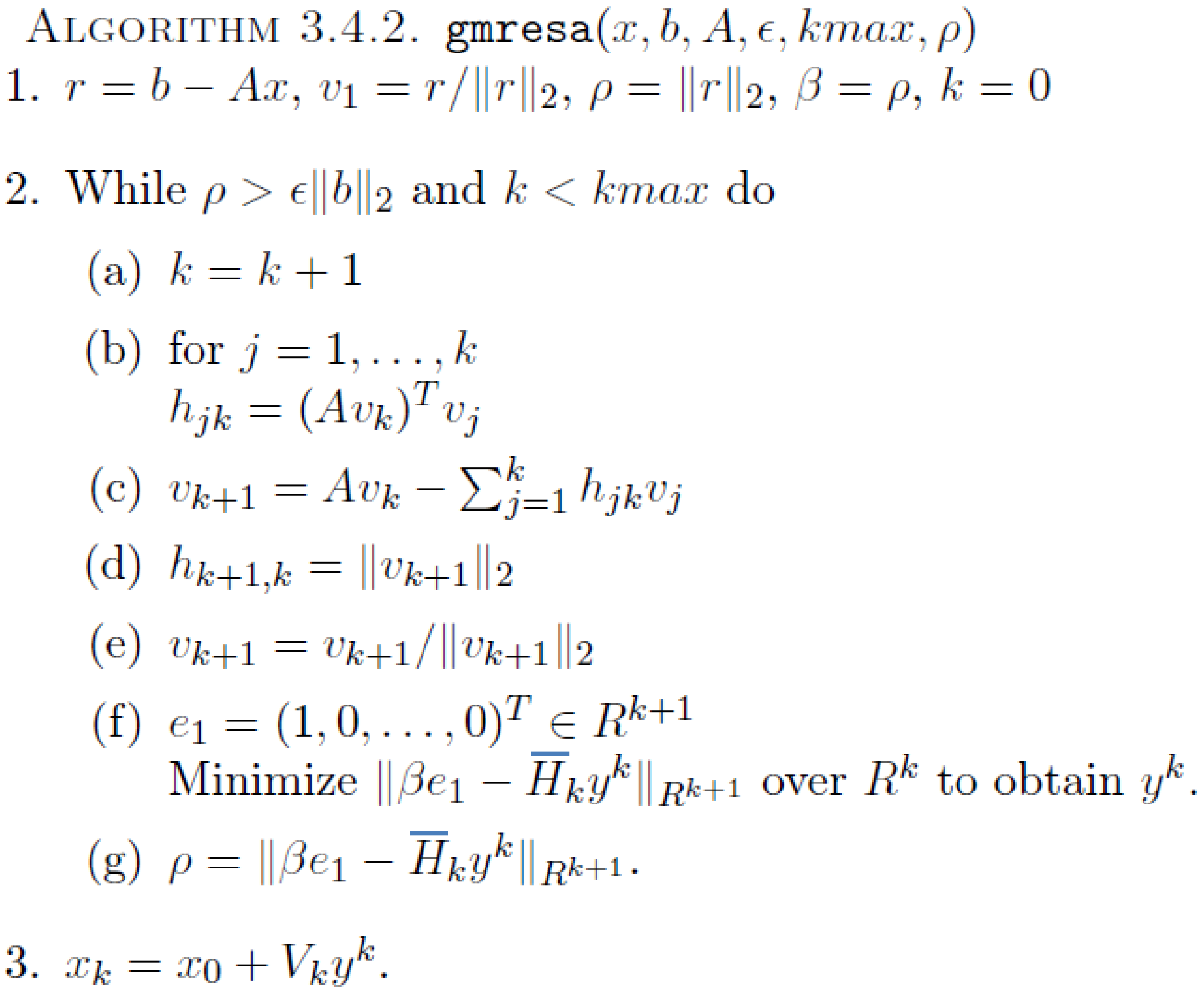
  - Remark: This is a linear least square problem, which can be solved by QR factorization. However, AV_k must be computed at each iteration.
  - The associate normal equation is $(AV_k)^T AV_k \boldsymbol{y} = (AV_k)^T \boldsymbol{r}_0$.
  - But we will solve it differently.

- Let $\boldsymbol{x}_k$ be kth iterative solution of GMRES.

Define: $\boldsymbol{r}_k = \boldsymbol{b} - A\boldsymbol{x}_k = \boldsymbol{r}_0 - A(\boldsymbol{x}_k - \boldsymbol{x}_0) = \beta V_{k+1}\boldsymbol{e}_1 - A(\boldsymbol{x}_0 + V_k\boldsymbol{y} - \boldsymbol{x}_0) = \beta V_{k+1}\boldsymbol{e}_1 - V_{k+1}\bar{H}_k\boldsymbol{y}^k = V_{k+1}(\beta\boldsymbol{e}_1 - \bar{H}_k\boldsymbol{y}^k)$

Using orthonomality of V_{k+1}:

$$minimize_{x\in \boldsymbol{x}_0+\mathrm{K}_k}||\boldsymbol{b} - A\boldsymbol{x}||_2$$
$$= minimize_{y\in R^k}||\beta\boldsymbol{e}_1 - \bar{H}_k\boldsymbol{y}^k||_2$$



```
ALGORITHM 3.4.2. gmresa($x, b, A, \epsilon, kmax, \rho$)
1. $r = b - Ax$, $v_1 = r/\|r\|_2$, $\rho = \|r\|_2$, $\beta = \rho$, $k = 0$

2. While $\rho > \epsilon\|b\|_2$ and $k < kmax$ do
    (a) $k = k + 1$
    (b) for $j = 1, \dots, k$
        $h_{jk} = (Av_k)^T v_j$
    (c) $v_{k+1} = Av_k - \sum_{j=1}^{k} h_{jk} v_j$
    (d) $h_{k+1,k} = \|v_{k+1}\|_2$
    (e) $v_{k+1} = v_{k+1}/\|v_{k+1}\|_2$
    (f) $e_1 = (1, 0, \dots, 0)^T \in R^{k+1}$
        Minimize $\|\beta e_1 - \overline{H}_k y^k\|_{R^{k+1}}$ over $R^k$ to obtain $y^k$.
    (g) $\rho = \|\beta e_1 - \overline{H}_k y^k\|_{R^{k+1}}$.

3. $x_k = x_0 + V_k y^k$.
```

"C.T. Kelley, Iterative Methods for Linear and Nonlinear Equations" .

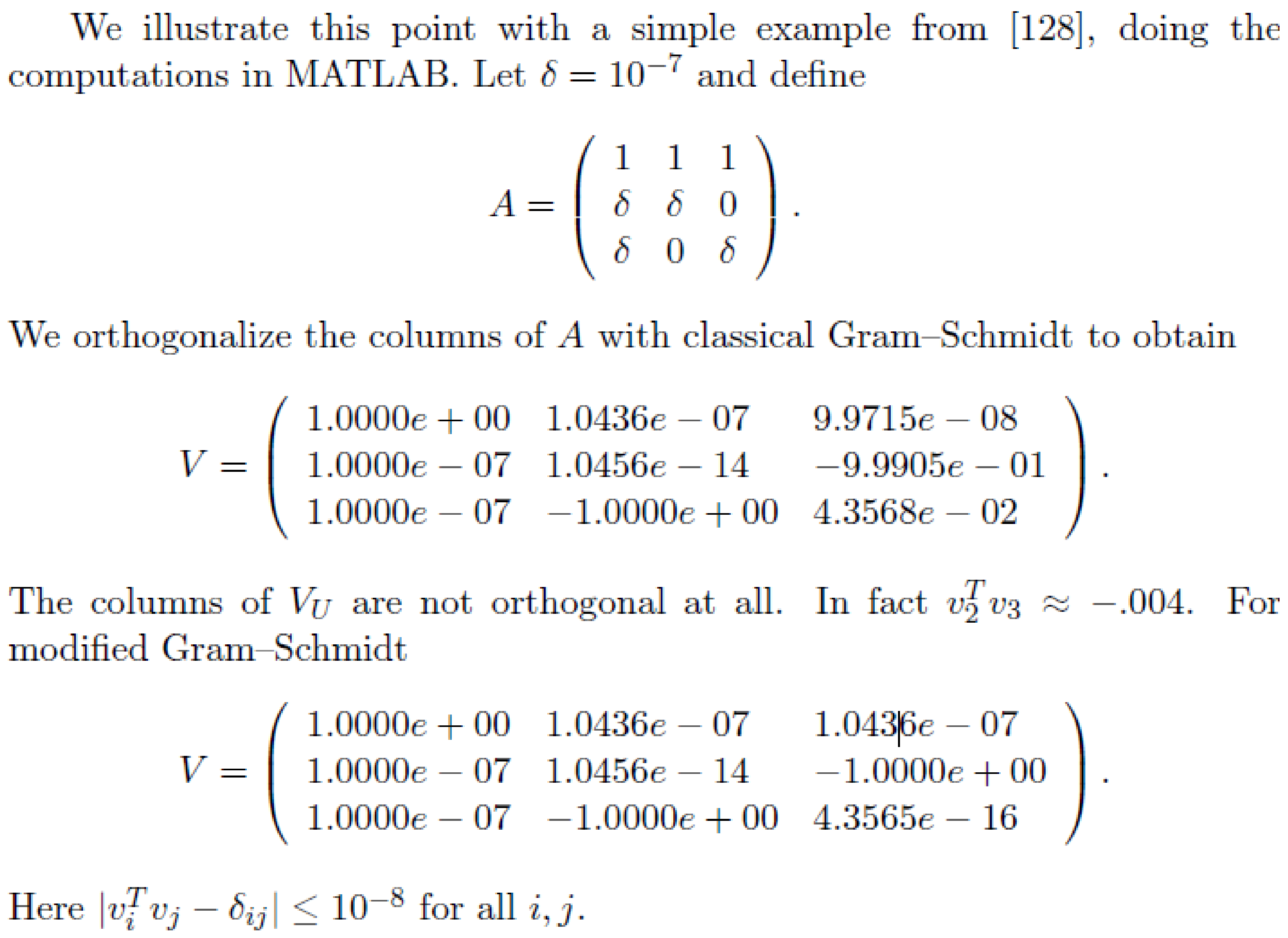
$$minimize_{y \in R^k} ||\beta \boldsymbol{e}_1 - \bar{H}_k \boldsymbol{y}^k||_2$$

Theorem. Let $n \times k$ $(k \leq n)$ matrix B be with linearly independent columns (full column rank). Let $B = QR$ be a QR factorization of B. Then for each $\boldsymbol{b} \in R^n$, the equation $B\boldsymbol{u} = \boldsymbol{b}$ has a unique least-square solution, given by $\hat{\boldsymbol{u}} = R^{-1}Q^T\boldsymbol{b}$.

Using Householder reflection to do QR factorization gives $\bar{H}_k = Q_{k+1}\bar{R}_k$ where $Q_{k+1} \in R^{(k+1)\times(k+1)}$ is orthogonal and $\bar{R}_k \in R^{(k+1)\times k}$ has the form $\bar{R}_k = \begin{bmatrix} R_k \\ 0 \end{bmatrix}$, where $R_k \in R^{k\times k}$ is upper triangular.

- $\boldsymbol{v}_j$ may become nonorthogonal as a result of round off errors.
  - $||\beta \boldsymbol{e}_1 - \bar{H}_k \boldsymbol{y}^k||_2$ which depends on orthogonality, will not hold and the residual could be inaccurate.
  - Replace the loop in Step 2c of Algorithm *gmresa* with

```
$v_{k+1} = Av_k$
for $j = 1, \ldots k$
  $v_{k+1} = v_{k+1} - (v_{k+1}^T v_j) v_j.$
```



We illustrate this point with a simple example from [128], doing the computations in MATLAB. Let $\delta = 10^{-7}$ and define

$$A = \begin{pmatrix} 1 & 1 & 1 \\ \delta & \delta & 0 \\ \delta & 0 & \delta \end{pmatrix}.$$

We orthogonalize the columns of A with classical Gram–Schmidt to obtain

$$V = \begin{pmatrix} 1.0000e+00 & 1.0436e-07 & 9.9715e-08 \\ 1.0000e-07 & 1.0456e-14 & -9.9905e-01 \\ 1.0000e-07 & -1.0000e+00 & 4.3568e-02 \end{pmatrix}.$$

The columns of V_U are not orthogonal at all. In fact $v_2^T v_3 \approx -.004$. For modified Gram–Schmidt

$$V = \begin{pmatrix} 1.0000e+00 & 1.0436e-07 & 1.0436e-07 \\ 1.0000e-07 & 1.0456e-14 & -1.0000e+00 \\ 1.0000e-07 & -1.0000e+00 & 4.3565e-16 \end{pmatrix}.$$

Here $|v_i^T v_j - \delta_{ij}| \leq 10^{-8}$ for all i, j.

“C.T. Kelley, Iterative Methods for Linear and Nonlinear Equations” .

```
ALGORITHM 3.4.3. gmresb($x, b, A, \epsilon, kmax, \rho$)
1. $r = b - Ax$, $v_1 = r/\|r\|_2$, $\rho = \|r\|_2$, $\beta = \rho$, $k = 0$
2. While $\rho > \epsilon\|b\|_2$ and $k < kmax$ do
    (a) $k = k + 1$
    (b) $v_{k+1} = Av_k$
        for $j = 1, \dots k$
          i. $h_{jk} = v_{k+1}^T v_j$
          ii. $v_{k+1} = v_{k+1} - h_{jk} v_j$
    (c) $h_{k+1,k} = \|v_{k+1}\|_2$
    (d) $v_{k+1} = v_{k+1}/\|v_{k+1}\|_2$
    (e) $e_1 = (1, 0, \dots, 0)^T \in R^{k+1}$
        Minimize $\|\beta e_1 - \overline{H}_k y^k\|_{R^{k+1}}$ to obtain $y^k \in R^k$.
    (f) $\rho = \|\beta e_1 - \overline{H}_k y^k\|_{R^{k+1}}$.
3. $x_k = x_0 + V_k y^k$.
```

“C.T. Kelley, Iterative Methods for Linear and Nonlinear Equations” .

Modified Gram-Schmidt Process with Reorthogonalization

```
• $v_{k+1} = Av_k$
  for $j = 1, \ldots, k$
  $h_{jk} = v_{k+1}^T v_j$
  $v_{k+1} = v_{k+1} - h_{jk} v_j$

• $h_{k+1,k} = \|v_{k+1}\|_2$

• If loss of orthogonality is detected
  For $j = 1, \ldots, k$
  $h_{tmp} = v_{k+1}^T v_j$
  $h_{jk} = h_{jk} + h_{tmp}$
  $v_{k+1} = v_{k+1} - h_{tmp} v_j$

• $h_{k+1,k} = \|v_{k+1}\|_2$

• $v_{k+1} = v_{k+1} / \|v_{k+1}\|_2$
```

Test Reorthogonalization

If $||Av_k||_2 + \delta||v_{k+1}||_2 = ||Av_k||_2$ to working precision.

$\delta = 10^{-3}$

Givens Rotations

$minimize_{y \in R^k} ||\beta \boldsymbol{e}_1 - \bar{H}_k \boldsymbol{y}^k||_2$ involves QR factorization.

Do QR factorizations of $\bar{H}_k$ by Givens Rotations.

- A 2×2 Givens rotation is a matrix of the form $G = \begin{bmatrix} c & -s \\ s & c \end{bmatrix}$ where $c = \cos(\theta)$, $s = \sin(\theta)$ for $\theta \in [-\pi, \pi]$. The orthogonal matrix G rotates the vector $(c, -s)^T$, which makes an angle of $-\theta$ with the x-axis, through an angle θ so that it overlaps the x-axis.

$$G \begin{bmatrix} c \\ -s \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

An $N \times N$ Givens rotation $G_j(c, s)$ replaces a 2×2 block on the diagonal of the $N \times N$ identity matrix with a 2×2 Givens rotations. $G_j(c, s)$ is with a 2×2 Givens rotations in rows and columns j and $j + 1$.

$$G = \begin{pmatrix} 1 & 0 & & & \cdots & & & 0 \\ 0 & \ddots & \ddots & & & & & \\ & \ddots & & c & -s & & & \\ \vdots & & & s & c & 0 & & \vdots \\ & & & & 0 & 1 & \ddots & \\ & & & & & \ddots & \ddots & 0 \\ 0 & & & & \cdots & & 0 & 1 \end{pmatrix}$$

- Givens rotations can be used in reducing Hessenberg matrices to triangular form. This can be done in $O(N)$ floating-point operations.

- Let H be an $N \times M (N \geq M)$ upper Hessenberg matrix with rank M. We reduce H to triangular form by first multiplying the matrix by a Givens rotations that zeros h_{21} (values of h_{11} and subsequent columns are changed)

- Step 1: Define $G_1(c_1, s_1)$ by $c_1 = h_{11}/\sqrt{h_{11}^2 + h_{21}^2}$ and $s_1 = -h_{21}/\sqrt{h_{11}^2 + h_{21}^2}$. Replace H by $G_1 H$.
- Step 2: Define $G_2(c_2, s_2)$ by $c_2 = h_{22}/\sqrt{h_{22}^2 + h_{32}^2}$ and $s_2 = -h_{32}/\sqrt{h_{22}^2 + h_{32}^2}$. Replace H by $G_2 H$.

  Remark: G_2 does not affect the first column of H.
- …
- Step j: Define $G_j(c_j, s_j)$ by $c_j = h_{jj}/\sqrt{h_{jj}^2 + h_{j+1,j}^2}$ and $s_j = -h_{j+1,j}/\sqrt{h_{jj}^2 + h_{j+1,j}^2}$. Replace H by $G_j H$.

Setting $Q = G_N \dots G_1$. $R = QH$ is upper triangular.

Let $\bar{H}_k = QR$ by Givens rotations matrices.

$$
\begin{aligned}
&minimize_{y\in R^k}||\beta \boldsymbol{e}_1 - \bar{H}_k \boldsymbol{y}^k||_2 \\
&\qquad = minimize_{y\in R^k}||Q(\beta \boldsymbol{e}_1 - \bar{H}_k \boldsymbol{y}^k)||_2 \\
&\qquad = minimize_{y\in R^k}||\beta Q \boldsymbol{e}_1 - R \boldsymbol{y}^k||_2
\end{aligned}
$$

```
ALGORITHM 3.5.1. gmres($x, b, A, \epsilon, kmax, \rho$)
1. $r = b - Ax$, $v_1 = r/\|r\|_2$, $\rho = \|r\|_2$, $\beta = \rho$,
   $k = 0$; $g = \rho(1, 0, \dots, 0)^T \in R^{kmax+1}$

2. While $\rho > \epsilon\|b\|_2$ and $k < kmax$ do
    (a) $k = k + 1$
    (b) $v_{k+1} = Av_k$
        for $j = 1, \dots k$
          i. $h_{jk} = v_{k+1}^T v_j$
          ii. $v_{k+1} = v_{k+1} - h_{jk}v_j$
    (c) $h_{k+1,k} = \|v_{k+1}\|_2$
    (d) Test for loss of orthogonality and reorthogonalize if necessary.
    (e) $v_{k+1} = v_{k+1}/\|v_{k+1}\|_2$
    (f) i. If $k > 1$ apply $Q_{k-1}$ to the $k$th column of $H$.
        ii. $\nu = \sqrt{h_{k,k}^2 + h_{k+1,k}^2}$.
        iii. $c_k = h_{k,k}/\nu$, $s_k = -h_{k+1,k}/\nu$
             $h_{k,k} = c_k h_{k,k} - s_k h_{k+1,k}$, $h_{k+1,k} = 0$
        iv. $g = G_k(c_k, s_k)g$.
    (g) $\rho = |(g)_{k+1}|$.

3. Set $r_{i,j} = h_{i,j}$ for $1 \le i, j \le k$.
   Set $(w)_i = (g)_i$ for $1 \le i \le k$.
   Solve the upper triangular system $Ry^k = w$.

4. $x_k = x_0 + V_k y^k$.
```

Preconditioning

Basic idea: using GMRES on a modified system such as $M^{-1}A\boldsymbol{x} = M^{-1}\boldsymbol{b}$.

The matrix $M^{-1}A$ need not to be formed explicitly. However, $M\boldsymbol{w} = \boldsymbol{v}$ need to be solved whenever needed.

Left preconditioning

$$M^{-1}A\boldsymbol{x} = M^{-1}\boldsymbol{b}$$

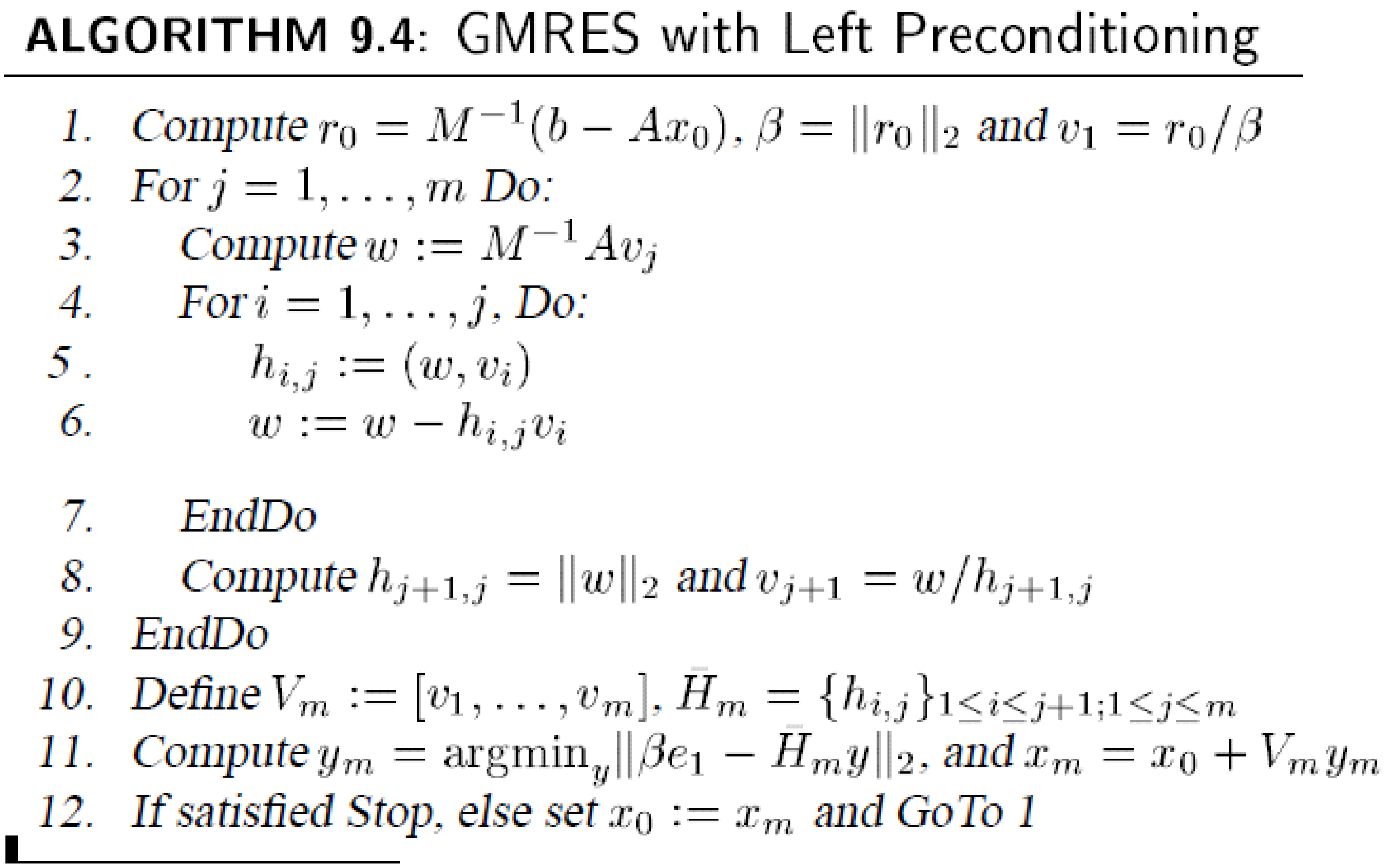
Right preconditioning

$$AM^{-1}\boldsymbol{u} = \boldsymbol{b} \ \ with \ \ \boldsymbol{x} = M^{-1}\boldsymbol{u}$$

Split preconditioning: M is factored as $M = M_L M_R$

$$M_L^{-1}AM_R^{-1}\boldsymbol{u} = M_L^{-1}\boldsymbol{b} \ \ with \ \ \boldsymbol{x} = M_R^{-1}\boldsymbol{u}$$

GMRES with Left Preconditioning



ALGORITHM 9.4: GMRES with Left Preconditioning

```
1.  Compute $r_0 = M^{-1}(b - Ax_0)$, $\beta = \|r_0\|_2$ and $v_1 = r_0/\beta$
2.  For $j = 1, \ldots, m$ Do:
3.      Compute $w := M^{-1}Av_j$
4.      For $i = 1, \ldots, j$, Do:
5 .         $h_{i,j} := (w, v_i)$
6.          $w := w - h_{i,j}v_i$

7.      EndDo
8.      Compute $h_{j+1,j} = \|w\|_2$ and $v_{j+1} = w/h_{j+1,j}$
9.  EndDo
10. Define $V_m := [v_1, \ldots, v_m]$, $\bar{H}_m = \{h_{i,j}\}_{1 \le i \le j+1; 1 \le j \le m}$
11. Compute $y_m = \mathrm{argmin}_y \|\beta e_1 - \bar{H}_m y\|_2$, and $x_m = x_0 + V_m y_m$
12. If satisfied Stop, else set $x_0 := x_m$ and GoTo 1
```

The Arnoldi process constructs an orthogonal basis for Span$\{\boldsymbol{r}_0, M^{-1}A\boldsymbol{r}_0, (M^{-1}A)^2\boldsymbol{r}_0, \ldots (M^{-1}A)^{k-1}\boldsymbol{r}_0\}$.

Sadd. Iterative Methods for Sparse Linear Systems

GMRES with Right Preconditioning

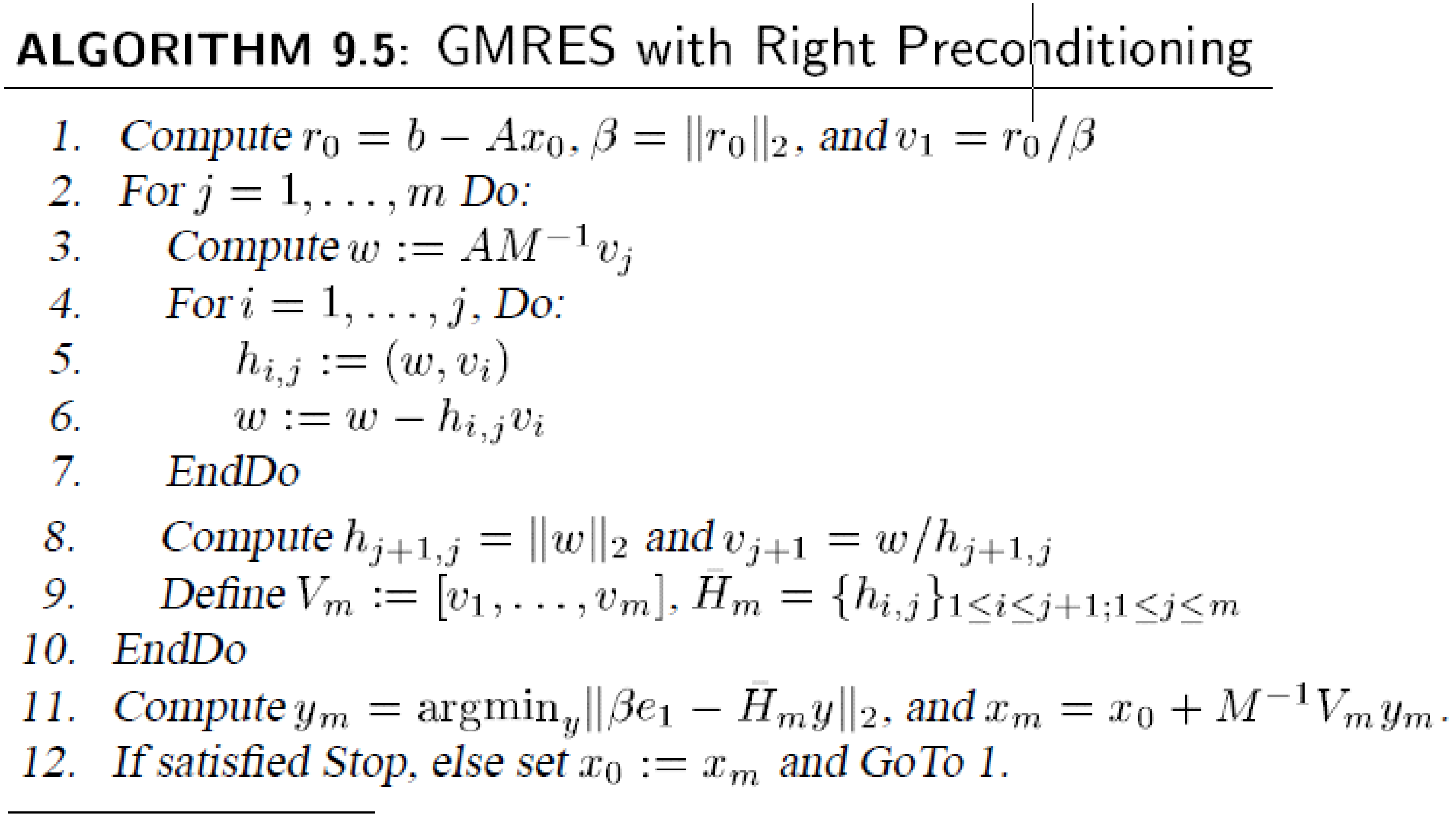
Right preconditioned GMRES is based on solving $AM^{-1}\boldsymbol{u} = \boldsymbol{b}\ \ with\ \ \boldsymbol{x} = M^{-1}\boldsymbol{u}$.

- The initial residual is: $\boldsymbol{b} - AM^{-1}\boldsymbol{u}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$.
  - This means all subsequent vectors of the Krylov subspace can be obtained without any references to the $\boldsymbol{u}$.
- At the end of right preconditioned GMRES:

$$\boldsymbol{u}_m = \boldsymbol{u}_0 + \sum_{i=1}^{m} \boldsymbol{v}_i \eta_i \quad with \quad \boldsymbol{u}_0 = M\boldsymbol{x}_0$$

$$\boldsymbol{x}_m = \boldsymbol{x}_0 + M^{-1} \sum_{i=1}^{m} \boldsymbol{v}_i \eta_i$$

GMRES with Right Preconditioning



ALGORITHM 9.5: GMRES with Right Preconditioning

```
1.  Compute $r_0 = b - Ax_0$, $\beta = \|r_0\|_2$, and $v_1 = r_0/\beta$
2.  For $j = 1, \ldots, m$ Do:
3.      Compute $w := AM^{-1}v_j$
4.      For $i = 1, \ldots, j$, Do:
5.          $h_{i,j} := (w, v_i)$
6.          $w := w - h_{i,j}v_i$
7.      EndDo
8.      Compute $h_{j+1,j} = \|w\|_2$ and $v_{j+1} = w/h_{j+1,j}$
9.      Define $V_m := [v_1, \ldots, v_m]$, $\bar{H}_m = \{h_{i,j}\}_{1\le i\le j+1; 1\le j\le m}$
10. EndDo
11. Compute $y_m = \operatorname{argmin}_y \|\beta e_1 - \bar{H}_m y\|_2$, and $x_m = x_0 + M^{-1}V_m y_m$.
12. If satisfied Stop, else set $x_0 := x_m$ and GoTo 1.
```

The Arnoldi process constructs an orthogonal basis for Span$\{\boldsymbol{r}_0, AM^{-1}\boldsymbol{r}_0, (AM^{-1})^2\boldsymbol{r}_0, \ldots (AM^{-1})^{k-1}\boldsymbol{r}_0\}$.

Sadd. Iterative Methods for Sparse Linear Systems.

Split Preconditioning

- M can be a factorization of the form $M = LU$.

- Then $L^{-1}AU^{-1}\boldsymbol{u} = L^{-1}\boldsymbol{b}, \quad with \quad \boldsymbol{x} = U^{-1}\boldsymbol{u}$.
  - Need to operate on the initial residual by $L^{-1}(\boldsymbol{b} - A\boldsymbol{x_0})$
  - Need to operate on the linear combination $U^{-1}(V_m\boldsymbol{y}_m)$ in forming the approximate solution

Comparison of Left and Right Preconditioning

- Spectra of $M^{-1}A$, AM^{-1} and $L^{-1}AU^{-1}$ are identical.
- In principle, one should expect convergence to be similar.
- When M is ill-conditioned, the difference could be substantial.

Jacobi Preconditioner

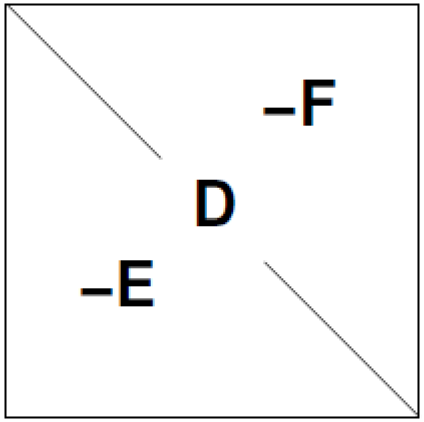
Iterative method for solving $Ax = b$ takes the form: $\boldsymbol{x}_{k+1} = M^{-1}N\boldsymbol{x}_k + M^{-1}\boldsymbol{b}$ where $M\ and\ N$ split A into $A = M - N$.

- Define $G = M^{-1}N = M^{-1}(M - A) = I - M^{-1}A$ and $\boldsymbol{f} = M^{-1}\boldsymbol{b}$.
- Iterative method is to solve $(I - G)\boldsymbol{x} = \boldsymbol{f}$, which can be written as $M^{-1}A\boldsymbol{x} = M^{-1}\boldsymbol{b}$.

Jacobi iterative method: $\boldsymbol{x}_{k+1} = G_{JA}\boldsymbol{x}_k + \boldsymbol{f}$ where $G_{JA} = (I - D^{-1}A)$ and $\boldsymbol{f} = D^{-1}\boldsymbol{b}$

- $M = D$ for Jacobi method.

SOR/SSOR Preconditioner



- Define: $A = D - E - F$
- Gauss-Seidel: $G_{GS} = I - (D - E)^{-1}A$
- $M_{SOR} = \frac{1}{w}(D - wE)$

A symmetric SOR (SSOR) consists of:

$$(D - wE)\boldsymbol{x}_{k+\frac{1}{2}} = [wF + (1-w)D]\boldsymbol{x}_k + w\boldsymbol{b}$$

$$(D - wF)\boldsymbol{x}_{k+1} = [wE + (1-w)D]\boldsymbol{x}_{k+\frac{1}{2}} + w\boldsymbol{b}$$

This gives

$$\boldsymbol{x}_{k+1} = G_{SSOR}\boldsymbol{x}_k + \boldsymbol{f}$$

Where

$G_{SSOR} = (D - wF)^{-1}(wE + (1-w)D)(D - wE)^{-1}(wF +$
$(1-w)D)$

- $M_{SSOR} = (D - wE)D^{-1}(D - wF)$; $M_{SGS} = (D - E)D^{-1}(D - F)$;
- Note: SSOR usually is used when A is symmetric

Take symmetric GS for example:

$M_{SGS} = (D - E)D^{-1}(D - F)$

- Define: $L = (D - E)D^{-1} = I - ED^{-1}$ and $U = D - F$.
- L is a lower triangular matrix and U is a upper triangular matrix.
- To solve $M_{SGS}\boldsymbol{w} = \boldsymbol{x}$ for $\boldsymbol{w}$, a forward solve and a backward solve are used:
  - Solve $(I - ED^{-1})\boldsymbol{z} = \boldsymbol{x}$ for $\boldsymbol{z}$
  - Solve $(D - F)\boldsymbol{w} = \boldsymbol{z}$ for $\boldsymbol{w}$

Incomplete LU(0) Factorization

Define: $NZ(X) = \{(i,j) | X_{i,j} \neq 0\}$

Incomplete LU (ILU(0)):

- $A = LU + R$ with $NZ(L) \cup NZ(U) = NZ(A)$

$$r_{ij} = 0 \quad for\ (i,j) \in NZ(A)$$

I.e. L and U have no fill-ins at the entries $a_{ij} = 0$.

```
for $i = 1$ to $n$
    for $k = 1$ to $i - 1$ and if $(i,k) \in NZ(A)$
        $a_{ik} = a_{ik}/a_{kj}$
        for j $= k + 1$ to $n$ and if $(i,k) \in NZ(A)$
            $a_{ij} = a_{ij} - a_{ik}a_{kj}$
        end;
    end;
end;
```

ILU(0)

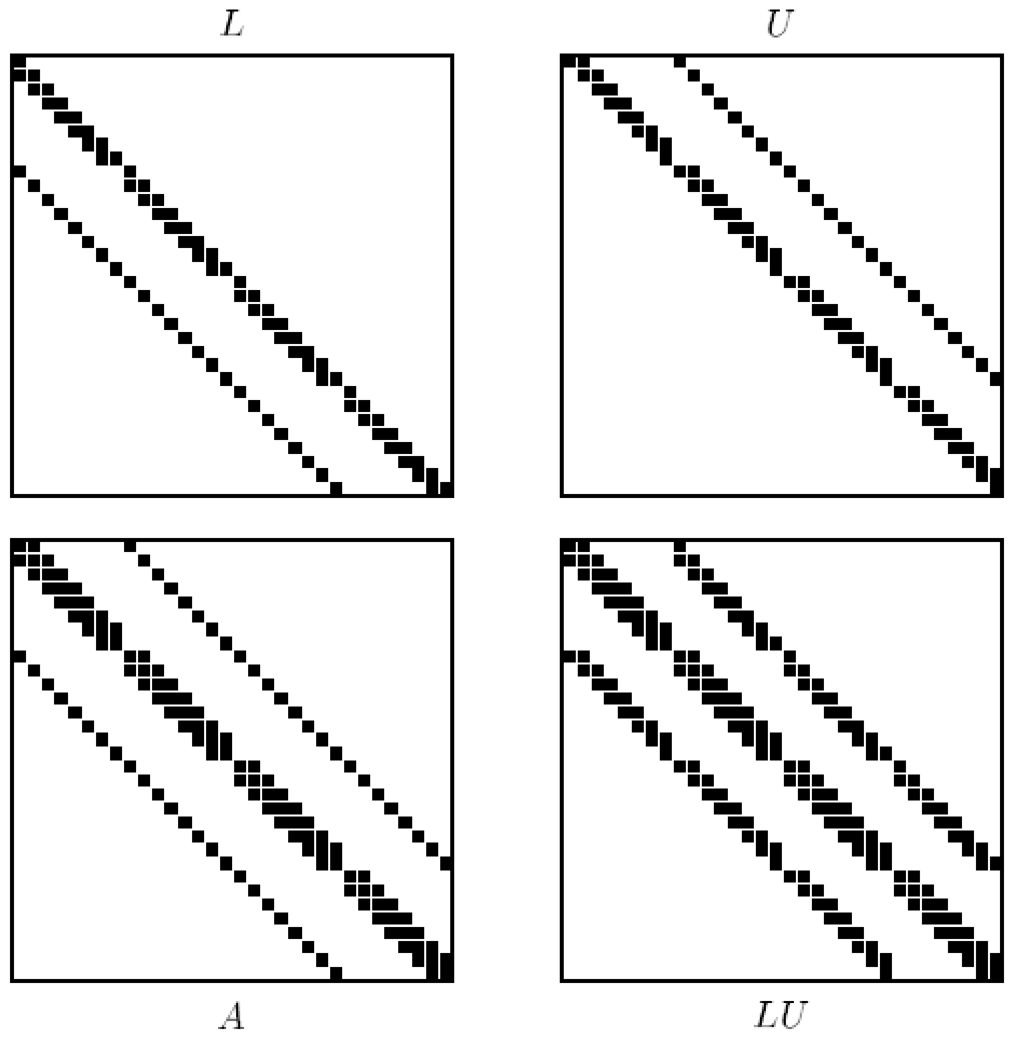
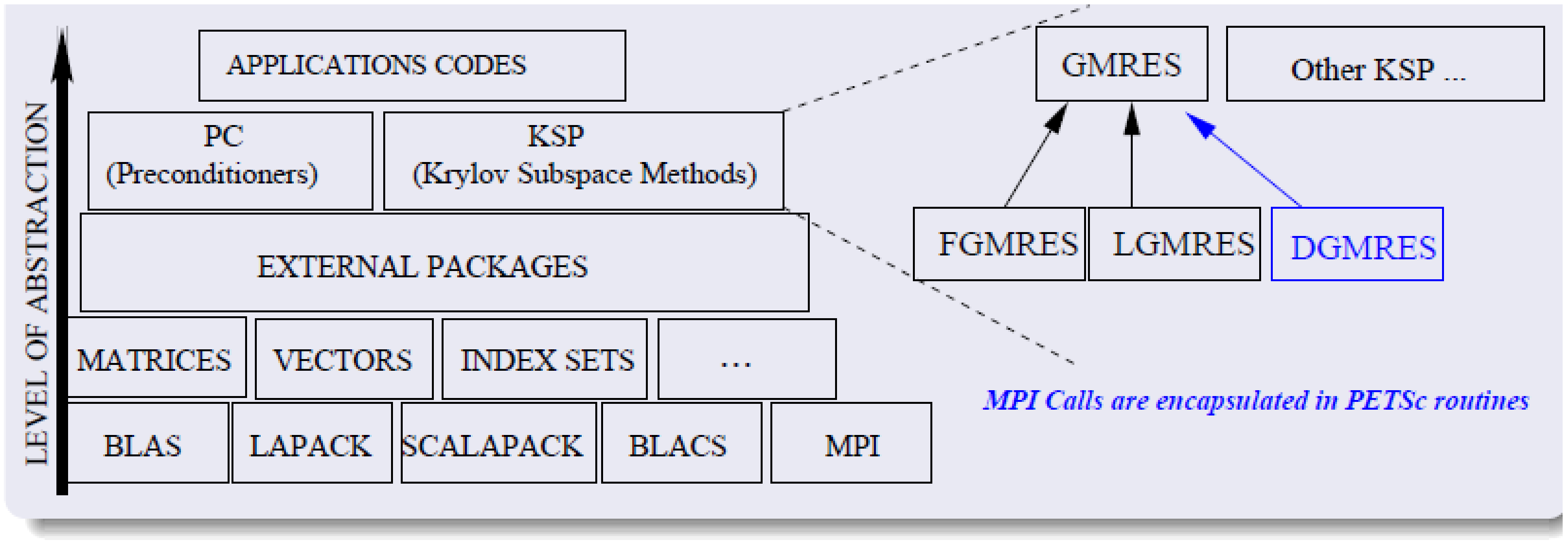


Figure 10.2 *The ILU(0) factorization for a five-point matrix.*

Sadd. Iterative Methods for Sparse Linear Systems.

Parallel GMRES

- J. Erhel. A parallel GMRES version for general sparse matrices. Electronic Transactions on Numerical Analyis. 3:160-176, 1995.
- Implementation in PETSc (Portable, Extensible Toolkit for Scientific Computation)
  - http://www.mcs.anl.gov/petsc/



Parallel Libraries

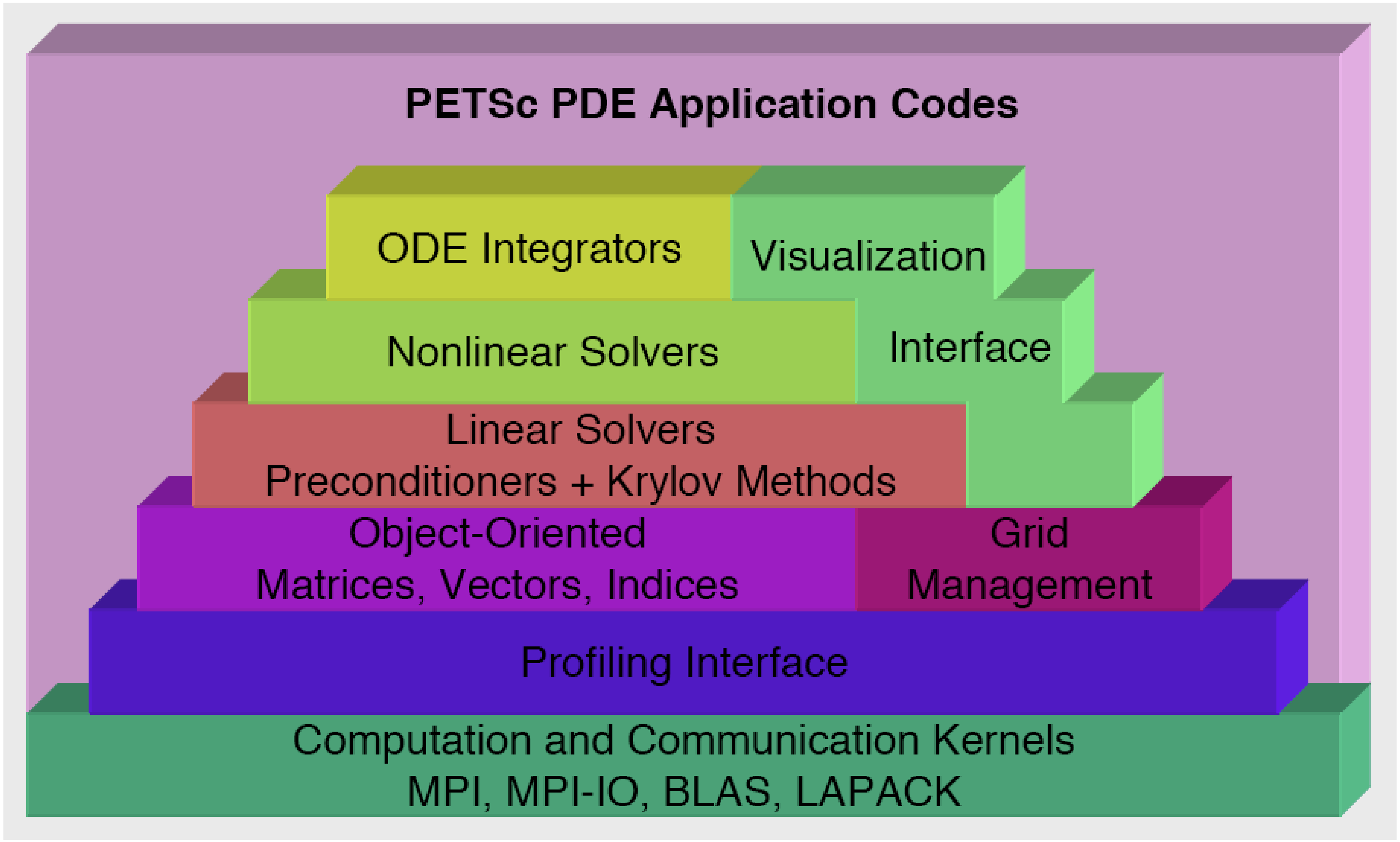
ScaLAPACK

- http://www.netlib.org/scalapack/
- Based on LAPACK (Linear Algebra PACKage) and BLAS (Basic Linear Algebra Subroutines)
- Parallelized by “divide and conquer” or block distribution
- Written in Fortran 90
- Successor of LINPACK, which was originally written for vector supercomputers in the 1970s
- Implemented on top of MPI using MIMD, SPMD, and used explicit message passing

PETSc (Portable, Extensible Toolkit for Scientific Computation)

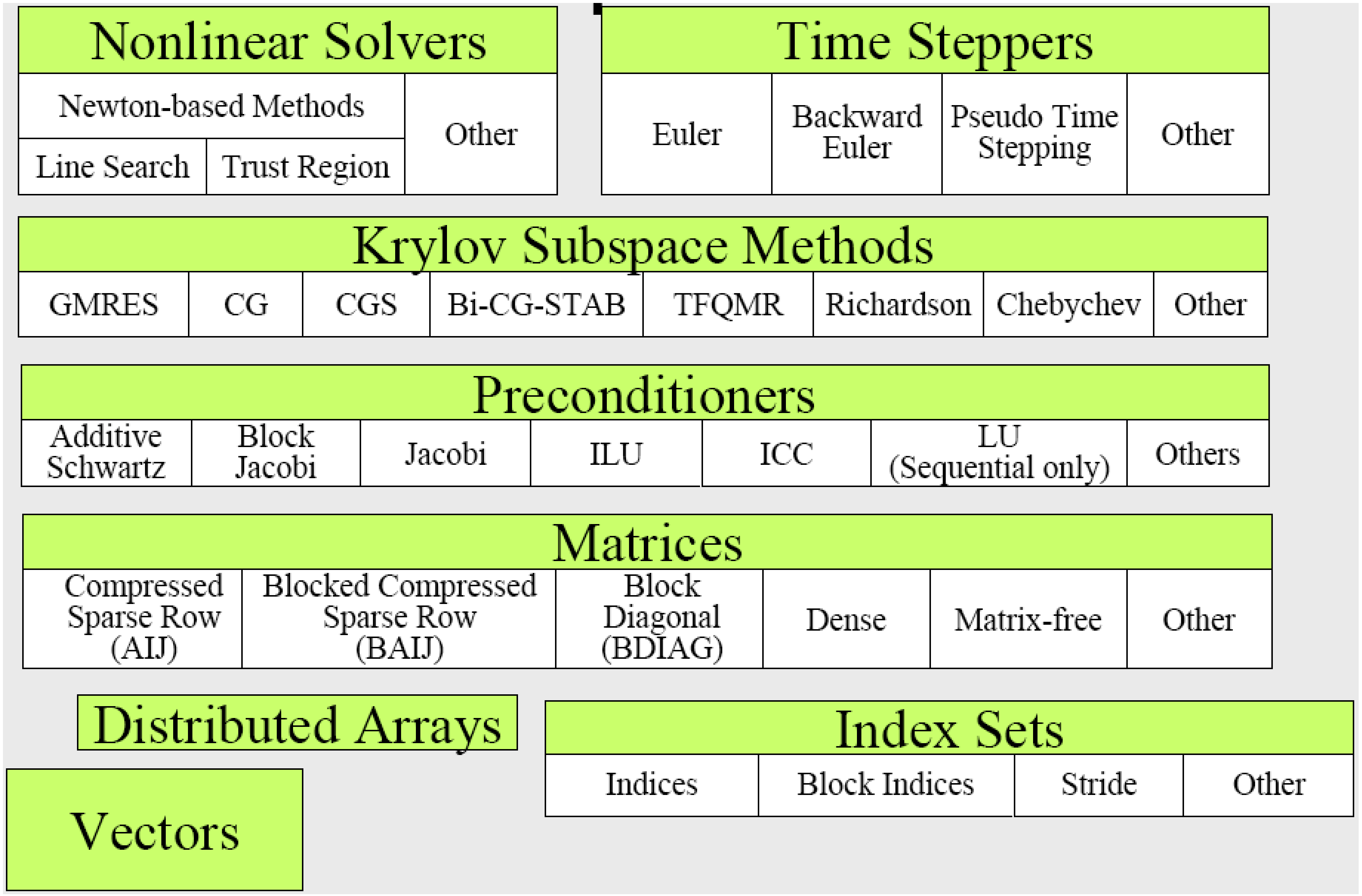
- http://www.mcs.anl.gov/petsc/
- Suite of data structures (core: distributed vectors and matrices) and routines for linea and non-linear solvers
- User (almost) never has to call MPI himself when using PETSc
- Uses two MPI communicators: PETSC_COMM_SELF for the library-internal communication and PETSC_COMM_WORLD for user processes
- Written in C, callable from Fortran
- Has been used to solve systems with over 500 millions unknowns
- Has been shown to scale up to over 6000 processors

PETSc Structure



PETSc PDE Application Codes
ODE Integrators
Visualization
Interface
Nonlinear Solvers
Linear Solvers
Preconditioners + Krylov Methods
Object-Oriented
Matrices, Vectors, Indices
Grid
Management
Profiling Interface
Computation and Communication Kernels
MPI, MPI-IO, BLAS, LAPACK

PETSc Numerical Solvers



Parallel Random Number Generator

SPRNG (The Scalable parallel random number generators library)

- http://sprng.cs.fsu.edu/
- Random number sequence does not depend on the number of processors used, but only on the seed
  a reproducible Monte Carlo simulations in parallel
- SPRNG implements parallel-safe, high-quality random number generators
- C++/Fortran (used to be C/Fortran in previous versions)

Parallel PDE Solver

POOMA (**P**arallel **O**bject-**O**riented **M**ethods and **A**pplications)

- http://acts.nersc.gov/formertools/pooma/index.html
- Collection of templated C++ classes for writing parallel PDE solvers
- Provides high-level data types (abstractions) for fields and particles using data-parallel arrays
- Supports finite-difference simulations on structured, unstructured, and adaptive grids. Also supports particle simulations, hybrid particle-mesh simulations, and Monte Carlo
- Uses mixed message-passing/thread parallelism

Many more…

- Aztec (iterative solvers for sparse linear systems)
- SuperLU (LU decomposition)
- Umfpack (unsymmetric multifrontal LU)
- EISPACK (eigen-solvers)
- Fishpack (cyclic reduction for 2nd & 4th order FD)
- PARTI (Parallel run-time system)
- Bisect (recursive orthogonal bisection)
- ROMIO (parallel distributed file I/O)
- KINSol (solves the nonlinear algebraic systems) https://computation.llnl.gov/casc/sundials/main.html
- SciPy (Scientific Tools for Phython) http://www.scipy.org/
- …

References:

– C.T. Kelley. Iterative Methods for Linear and Nonlinear Equations.

– Yousef Sadd. Iterative methods for Sparse Linear Systems

– G. Karypis and V. Kumar. Parallel Threshold-based ILU Factorization. *Technical Report #96-061. U. of Minnesota*, Dept. of Computer Science, 1998.

– P.-O. Persson and J. Peraire. Newton-GMRES Preconditioning for Discontinuous Galerkin Discretizations of the Navier-Stokes Equations. *SIAM J. on Sci. Comput.* 30(6), 2008.