Numerical Linear Algebra: iterative methods

Victor Eijkhout

Two different approaches

Solve $Ax = b$

Direct methods:

- Deterministic
- Exact up to machine precision
- Expensive (in time and space)

Iterative methods:

- Only approximate
- Cheaper in space and (possibly) time
- Convergence not guaranteed

Iterative methods

Choose any x_0 and repeat

$$x^{k+1} = Bx^k + c$$

until $\|x^{k+1} - x^k\|_2 < \epsilon$ or until $\frac{\|x^{k+1} - x^k\|_2}{\|x^k\|} < \epsilon$

Example of iterative solution

Example system

$$\begin{pmatrix} 10 & 0 & 1 \\ 1/2 & 7 & 1 \\ 1 & 0 & 6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 21 \\ 9 \\ 8 \end{pmatrix}$$

with solution $(2, 1, 1)$.

Suppose you know (physics) that solution components are roughly the same size, and observe the dominant size of the diagonal, then

$$\begin{pmatrix} 10 & & \\ & 7 & \\ & & 6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 21 \\ 9 \\ 8 \end{pmatrix}$$

might be a good approximation: solution $(2.1, 9/7, 8/6)$.

Iterative example′

Example system

$$\begin{pmatrix} 10 & 0 & 1 \\ 1/2 & 7 & 1 \\ 1 & 0 & 6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 21 \\ 9 \\ 8 \end{pmatrix}$$

with solution $(2, 1, 1)$.

Also easy to solve:

$$\begin{pmatrix} 10 & & \\ 1/2 & 7 & \\ 1 & 0 & 6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 21 \\ 9 \\ 8 \end{pmatrix}$$

with solution $(2.1, 7.95/7, 5.9/6)$.

Iterative example''

Instead of solving $Ax = b$ we solved $L\tilde{x} = b$. Look for the missing part: $\tilde{x} = x + \Delta x$, then $A\Delta x = A\tilde{x} - b \equiv r$. Solve again $L\widetilde{\Delta x} = r$ and update $\tilde{\tilde{x}} = \tilde{x} - \widetilde{\Delta x}$.

iteration	1	2	3
x_1	2.1000	2.0017	2.000028
x_2	1.1357	1.0023	1.000038
x_3	0.9833	0.9997	0.999995

Two decimals per iteration. *This is not typical*

Exact system solving: $O(n^3)$ cost; iteration: $O(n^2)$ per iteration. Potentially cheaper if the number of iterations is low.

Abstract presentation

- To solve $Ax = b$; too expensive; suppose $K \approx A$ and solving $Kx = b$ is possible
- Define $Kx_0 = b$, then error correction $e_0 = x - x_0$, and $A(x_0 + e_0) = b$
- so $Ae_0 = b - Ax_0 = r_0$; this is again unsolvable, so
- $K\tilde{e}_0$ and $x_1 = x_0 + \tilde{e}_0$.
- now iterate: $e_1 = x - x_1$, $Ae_1 = b - Ax_1 = r_1$ et cetera

Error analysis

- One step

$$
\begin{aligned}
r_1 &= b - Ax_1 = b - A(x_0 + \tilde{e}_0) && (1)\\
&= r_0 - AK^{-1}r_0 && (2)\\
&= (I - AK^{-1})r_0 && (3)
\end{aligned}
$$

- Inductively: $r_n = (I - AK^{-1})^n r_0$ so $r_n \downarrow 0$ if $|\lambda(I - AK^{-1})| < 1$
  Geometric reduction (or amplification!)
- This is 'stationary iteration': every iteration step the same.
  Simple analysis, limited applicability.

Choice of K

- The closer K is to A, the faster convergence.
- Diagonal and lower triangular choice mentioned above: let

$$A = D_A + L_A + U_A$$

  be a splitting into diagonal, lower triangular, upper triangular part, then
- Jacobi method: $K = D_A$ (diagonal part),
- Gauss-Seidel method: $K = D_A + L_A$ (lower triangle, including diagonal)

Computationally

If

$$A = K - N$$

then

$$Ax = b \Rightarrow Kx = Nx + b \Rightarrow Kx_{i+1} = Nx_i + b$$

Equivalent to the above, and you don't actually need to form the residual.

Jacobi

$$K = D_A$$

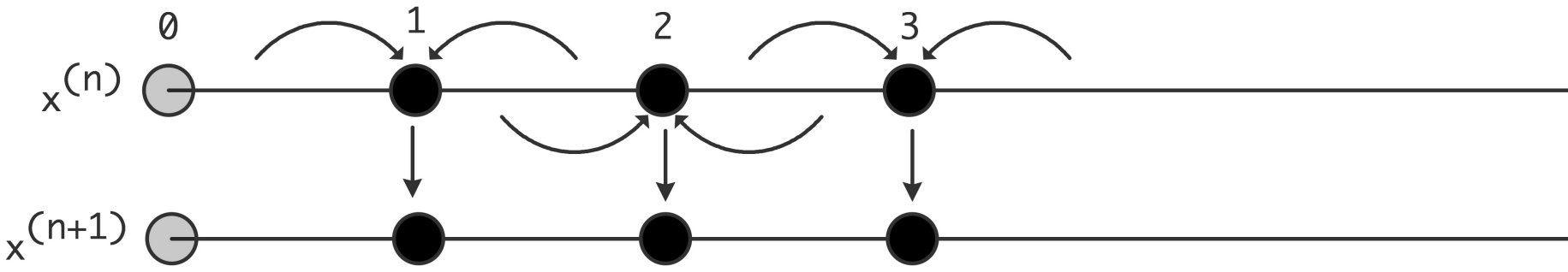
Algorithm:

for $k = 1, \ldots$ *until convergence, do:*
  for $i = 1 \ldots n$*:*
    $x_i^{(k+1)} = a_{ii}^{-1}(\sum_{j \neq i} a_{ij} x_j^{(k)} + b_i)$

Implementation:

for $k = 1, \ldots$ *until convergence, do:*
  for $i = 1 \ldots n$*:*
    $t_i = a_{ii}^{-1}(-\sum_{j \neq i} a_{ij} x_j + b_i)$
  copy $x \leftarrow t$

Jacobi in pictures:



Gauss-Seidel

$$K = D_A + L_A$$

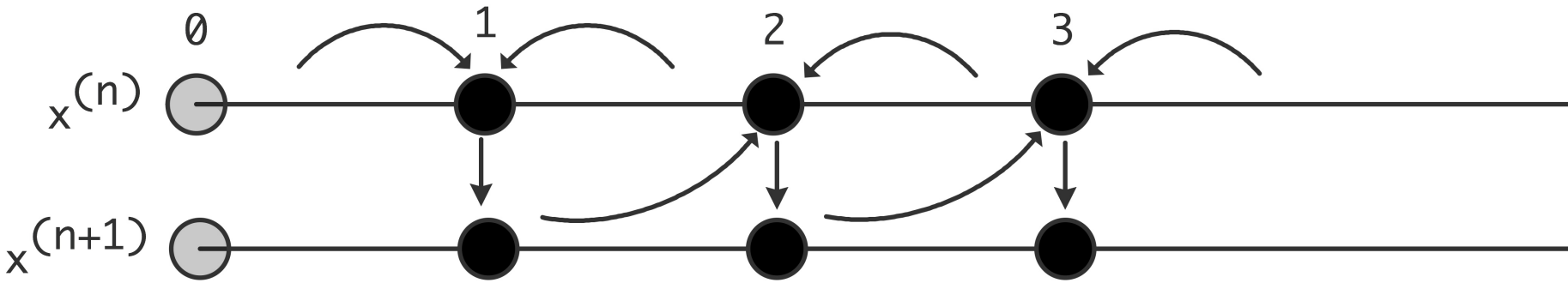
Algorithm:

for $k = 1, \ldots$ *until convergence, do:*
  for $i = 1 \ldots n$*:*
    $x_i^{(k+1)} = a_{ii}^{-1}(-\sum_{j<i} a_{ij} x_j^{(k+1)}) - \sum_{j>i} a_{ij} x_j^{(k)} + b_i)$

Implementation:

for $k = 1, \ldots$ *until convergence, do:*
  for $i = 1 \ldots n$*:*
    $x_i = a_{ii}^{-1}(-\sum_{j \neq i} a_{ij} x_j + b_i)$

GS in pictures:



Choice of K through incomplete LU

Gauss elimination $LU = A$:

```
for k,i,j:
   a[i,j] = a[i,j] - a[i,k] * a[k,j] / a[k,k]
```

Incomplete variant $K = LU \approx A$:

```
for k,i,j:
  if a[i,j] not zero:
    a[i,j] = a[i,j] - a[i,k] * a[k,j] / a[k,k]
```

$\Rightarrow$ sparsity of $L + U$ the same as of A
it is possible to allow some fill-in

Stopping tests

When to stop converging? Can size of the error be guaranteed?

- Direct tests on error $e_n = x - x_n$ impossible; two choices
- Relative change in the computed solution small:

$$\|x_{n+1} - x_n\| / \|x_n\| < \epsilon$$

- Residual small enough:

$$\|r_n\| = \|Ax_n - b\| < \epsilon$$

Without proof: both imply that the error is less than some other ϵ'.

General form of iterative methods 1.

System $Ax = b$ has the same solution as $K^{-1}Ax = K^{-1}b$.

Let $\tilde{x}$ be a guess and

$$\tilde{r} = K^{-1}A\tilde{x} - K^{-1}b.$$

then

$$x = A^{-1}b = \tilde{x} - A^{-1}K\tilde{r} = \tilde{x} - (K^{-1}A)^{-1}\tilde{r}.$$

Using Cayley-Hamilton theorem:

$$x = \tilde{x} - \pi(K^{-1}A)K^{-1}\tilde{r} = \tilde{x} - K^{-1}\pi(AK^{-1})\tilde{r}.$$

Iterative scheme:

$$x_{i+1} = x_0 + K^{-1}\pi^{(i)}(AK^{-1})r_0 \qquad (4)$$

Convergence theory for residuals

$$x_{i+1} = x_0 + K^{-1}\pi^{(i)}(AK^{-1})r_0$$

Multiply by A and subtract b:

$$r_{i+1} = r_0 + \tilde{\pi}^{(i)}(AK^{-1})r_0$$

So:

$$r_i = \hat{\pi}^{(i)}(AK^{-1})r_0$$

where $\hat{\pi}^{(i)}$ is a polynomial of degree i with $\hat{\pi}^{(i)}(0) = 1$.

What polynomial sequence minimizes the residual?

Juggling polynomials

Lots of induction proves

$$(AK^{-1})^i r_0 \in [\![r_i, \ldots, r_0]\!]. \tag{5}$$

and

$$r_i \in [\![(AK^{-1})^0 r_0 \ldots, (AK^{-1})^{i-1} r_0]\!]. \tag{6}$$

General form of iterative methods 3.

$$x_{i+1} = x_0 + \sum_{j \leq i} K^{-1} r_j \alpha_{ji}.$$

or equivalently:

$$x_{i+1} = x_i + \sum_{j \leq i} K^{-1} r_j \alpha_{ji}.$$

More residual identities

$$x_{i+1} = x_i + \sum_{j \le i} K^{-1} r_j \alpha_{ji}.$$

gives

$$r_{i+1} = r_i + \sum_{j \le i} A K^{-1} r_j \alpha_{ji}.$$

More throwing of formulas:

$$r_{i+1} \gamma_{i+1,i} = A K^{-1} r_i + \sum_{j \le i} r_j \gamma_{ji}$$

where $\gamma_{i+1,i} = \sum_{j \le i} \gamma_{ji}$.

General form of iterative methods 4.

$$r_{i+1}\gamma_{i+1,i} = AK^{-1}r_i + \sum_{j\le i} r_j\gamma_{ji}$$

and $\gamma_{i+1,i} = \sum_{j\le i}\gamma_{ji}$.

Write this as $AK^{-1}R = RH$ where

$$H = \begin{pmatrix} -\gamma_{11} & -\gamma_{12} & \cdots & & \\ \gamma_{21} & -\gamma_{22} & -\gamma_{23} & \cdots & \\ 0 & \gamma_{32} & -\gamma_{33} & -\gamma_{34} & \\ \emptyset & \ddots & \ddots & \ddots & \ddots \end{pmatrix}$$

H is a Hessenberg matrix, and note zero column sums.

Divide A out:

$$x_{i+1}\gamma_{i+1,i} = K^{-1}r_i + \sum_{j\le i} x_j\gamma_{ji}$$

General form of iterative methods 5.

$$\begin{cases} r_i = Ax_i - b \\ x_{i+1}\gamma_{i+1,i} = K^{-1}r_i + \sum_{j\le i} x_j\gamma_{ji} \\ r_{i+1}\gamma_{i+1,i} = AK^{-1}r_i + \sum_{j\le i} r_j\gamma_{ji} \end{cases}$$

where $\gamma_{i+1,i} = \sum_{j\le i} \gamma_{ji}$.

Orthogonality

Idea one:

> *If you can make all your residuals orthogonal to each other, and the matrix is of dimension n, then after n iterations you have to have converged: it is not possible to have an $n+1$-st residuals that is orthogonal and nonzero.*

Idea two:

> *The sequence of residuals spans a series of subspaces of increasing dimension; by orthogonalizing the error is the distance between r_0 and these spaces. This means that the error will be decreasing.*

Full Orthogonalization Method

Let r_0 *be given*
For $i \geq 0$*:*
    let $s \leftarrow K^{-1} r_i$
    let $t \leftarrow AK^{-1} r_i$
    for $j \leq i$*:*
        let γ_j *be the coefficient so that* $t - \gamma_j r_j \perp r_j$
    for $j \leq i$*:*
        form $s \leftarrow s - \gamma_j x_j$
        and $t \leftarrow t - \gamma_j r_j$
    let $x_{i+1} = (\sum_j \gamma_j)^{-1} s$, $r_{i+1} = (\sum_j \gamma_j)^{-1} t$.

Modified Gramm-Schmidt

Let r_0 *be given*
For $i \geq 0$*:*
    let $s \leftarrow K^{-1} r_i$
    let $t \leftarrow AK^{-1} r_i$
    for $j \leq i$*:*
        let γ_j *be the coefficient so that* $t - \gamma_j r_j \perp r_j$
        form $s \leftarrow s - \gamma_j x_j$
        and $t \leftarrow t - \gamma_j r_j$
    let $x_{i+1} = (\sum_j \gamma_j)^{-1} s$, $r_{i+1} = (\sum_j \gamma_j)^{-1} t$.

Coupled recurrences form

$$x_{i+1} = x_i - \sum_{j \le i} \alpha_{ji} K^{-1} r_j \tag{7}$$

This equation is often split as

- Update iterate with search direction: direction:

$$x_{i+1} = x_i - \delta_i p_i,$$

- Construct search direction from residuals:

$$p_i = K^{-1} r_i + \sum_{j<i} \beta_{ij} K^{-1} r_j.$$

  Inductively:

$$p_i = K^{-1} r_i + \sum_{j<i} \gamma_{ij} p_j,$$

Conjugate Gradients

Basic idea:

$$r_i^t K^{-1} r_j = 0 \quad \text{if } i \neq j.$$

Split recurrences:

$$\begin{cases} x_{i+1} = x_i - \delta_i p_i \\ r_{i+1} = r_i - \delta_i A p_i \\ p_i = K^{-1} r_i + \sum_{j<i} \gamma_{ij} p_j, \end{cases} \tag{8}$$

Derivation 1.

Let

- x_1, r_1, p_1 are the current iterate, residual, and search direction. Note that the subscript 1 does not denote the iteration number here.
- x_2, r_2, p_2 are the iterate, residual, and search direction that we are about to compute. Again, the subscript does not equal the iteration number.
- X_0, R_0, P_0 are all previous iterates, residuals, and search directions bundled together in a block of vectors.

Derivation 2.

In terms of these quantities, the update equations are then

$$\begin{cases} x_2 = x_1 - \delta_1 p_1 \\ r_2 = r_1 - \delta_i A p_1 \\ p_2 = K^{-1} r_2 + \upsilon_{12} p_1 + P_0 u_{02} \end{cases} \tag{9}$$

where δ_1, υ_{12} are scalars, and u_{02} is a vector with length the number of iterations before the current.

Derivation of scalars

We want:

$$r_2^t K^{-1} r_1 = 0, \qquad r_2^t K^{-1} R_0 = 0.$$

Combining these relations gives us, for instance,

$$\left.\begin{array}{l} r_1^t K^{-1} r_2 = 0 \\ r_2 = r_1 - \delta_i A K^{-1} p_1 \end{array}\right\} \Rightarrow \delta_1 = \frac{r_1^t r_1}{r_1^t A K^{-1} p_1}.$$

Finding υ_{12}, u_{02} is a little harder.

Preconditioned Conjugate Gradients

Compute $r^{(0)} = b - Ax^{(0)}$ for some initial guess $x^{(0)}$
for $i = 1, 2, \ldots$
  solve $Mz^{(i-1)} = r^{(i-1)}$
  $\rho_{i-1} = r^{(i-1)^T} z^{(i-1)}$
  if $i = 1$
    $p^{(1)} = z^{(0)}$
  else
    $\beta_{i-1} = \rho_{i-1}/\rho_{i-2}$
    $p^{(i)} = z^{(i-1)} + \beta_{i-1} p^{(i-1)}$
  endif
  $q^{(i)} = Ap^{(i)}$
  $\alpha_i = \rho_{i-1}/p^{(i)^T} q^{(i)}$
  $x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$
  $r^{(i)} = r^{(i-1)} - \alpha_i q^{(i)}$
  check convergence; continue if necessary
end

Observations on iterative methods

- Conjugate gradients: constant storage and inner products; works only for symmetric systems
- GMRES (like FOM): growing storage and inner products: restarting and numerical cleverness
- BiCGstab and QMR: relax the orthogonality

CG derived from minimization

Special case of SPD:

For which vector x with $\|x\| = 1$ is $f(x) = 1/2x^t Ax - b^t x$ minimal? (10)

Taking derivative:

$$f'(x) = Ax - b.$$

Update

$$x_{i+1} = x_i + p_i \delta_i$$

optimal value:

$$\delta_i = \operatorname*{argmin}_{\delta} \|f(x_i + p_i \delta)\| = \frac{r_i^t p_i}{p_1^t A p_i}$$

Other constants follow from orthogonality.

Parallism

- Vector operations, including inner products
- Matrix vector product
- Preconditioner (K) application

Parallelism in preconditioners: the problem

Mvp:

```
for i=1..n
  y[i] = sum over j=1..n a[i,j]*x[j]
```

In parallel:

```
for i=myfirstrow..mylastrow
  y[i] = sum over j=1..n a[i,j]*x[j]
```

Preconditioner ILU:

```
for i=1..n
  x[i] = (y[i] - sum over j=1..i-1 ell[i,j]*x[j]) / a[i,i]
```

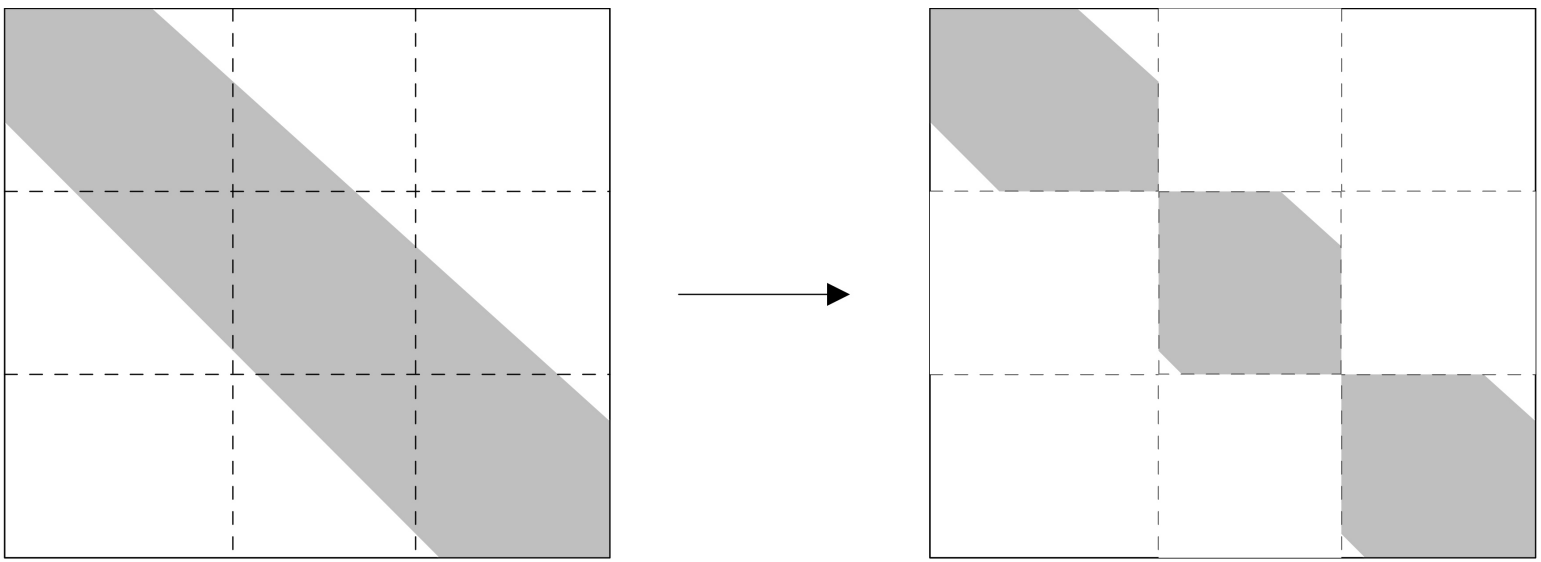
parallel:

```
for i=myfirstrow..mylastrow
  x[i] = (y[i] - sum over j=1..i-1 ell[i,j]*x[j]) / a[i,i]
```

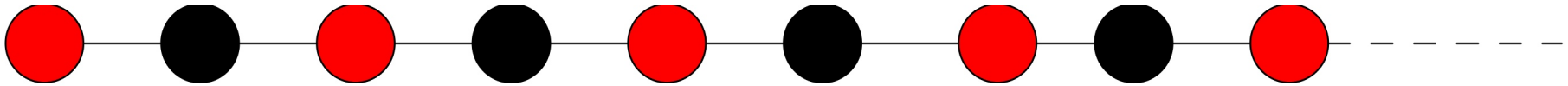
Not the same!

Block Jacobi

```
for i=myfirstrow..mylastrow
  x[i] = (y[i] - sum over j=myfirstrow..i-1 ell[i,j]*x[j])
         / a[i,i]
```



Multicolouring



$$\begin{pmatrix} a_{11} & & & & a_{12} & & \\ & a_{33} & & & a_{32} & a_{34} & \\ & & a_{55} & & & \ddots & \ddots \\ & & & \ddots & & & \\ a_{21} & a_{23} & & & a_{22} & & \\ & a_{43} & a_{45} & & & a_{44} & \\ & & \ddots & \ddots & & & \ddots \end{pmatrix} \begin{pmatrix} x_1 \\ x_3 \\ x_5 \\ \vdots \\ x_2 \\ x_4 \\ \vdots \end{pmatrix} = \begin{pmatrix} y_1 \\ y_3 \\ y_5 \\ \vdots \\ y_2 \\ y_4 \\ \vdots \end{pmatrix}$$

Parallelism through multicolouring