

Exploiting hyper-sparsity when computing preconditioners for conjugate gradients in interior point methods

Julian Hall, Ghussoun Al-Jeiroudi and Jacek Gondzio

School of Mathematics

University of Edinburgh

27th June 2007



What is hyper-sparsity?

What is hyper-sparsity?

- Hyper-sparsity occurs when the result of a sparse matrix-vector computation has sufficiently few nonzero values to be worth exploiting, both when **forming** and **using** the result

What is hyper-sparsity?

- Hyper-sparsity occurs when the result of a sparse matrix-vector computation has sufficiently few nonzero values to be worth exploiting, both when **forming** and **using** the result
- Simple example

$$\mathbf{y} = \mathbf{A}\mathbf{x} \text{ for sparse } \mathbf{A} \text{ and } \mathbf{x}$$

- Can expect $\mathbf{y}$ to be sparse

What is hyper-sparsity?

- Hyper-sparsity occurs when the result of a sparse matrix-vector computation has sufficiently few nonzero values to be worth exploiting, both when **forming** and **using** the result
- Simple example

$$\mathbf{y} = \mathbf{A}\mathbf{x} \text{ for sparse } \mathbf{A} \text{ and } \mathbf{x}$$

- Can expect $\mathbf{y}$ to be sparse
- **Inefficient:** If $\mathbf{A}$ is packed row-wise and $\mathbf{x}$ is full-length, most operations with nonzeros in $\mathbf{A}$ are with zeros in $\mathbf{x}$

What is hyper-sparsity?

- Hyper-sparsity occurs when the result of a sparse matrix-vector computation has sufficiently few nonzero values to be worth exploiting, both when **forming** and **using** the result
- Simple example

$$\mathbf{y} = \mathbf{A}\mathbf{x} \text{ for sparse } \mathbf{A} \text{ and } \mathbf{x}$$

- Can expect $\mathbf{y}$ to be sparse
- **Inefficient:** If $\mathbf{A}$ is packed row-wise and $\mathbf{x}$ is full-length, most operations with nonzeros in $\mathbf{A}$ are with zeros in $\mathbf{x}$
- **Efficient:** Pack $\mathbf{A}$ column-wise and combine the columns corresponding to nonzeros in $\mathbf{x}$

What is hyper-sparsity?

- Hyper-sparsity occurs when the result of a sparse matrix-vector computation has sufficiently few nonzero values to be worth exploiting, both when **forming** and **using** the result
- Simple example

$$\mathbf{y} = \mathbf{A}\mathbf{x} \text{ for sparse } \mathbf{A} \text{ and } \mathbf{x}$$

- Can expect $\mathbf{y}$ to be sparse
- **Inefficient:** If $\mathbf{A}$ is packed row-wise and $\mathbf{x}$ is full-length, most operations with nonzeros in $\mathbf{A}$ are with zeros in $\mathbf{x}$
- **Efficient:** Pack $\mathbf{A}$ column-wise and combine the columns corresponding to nonzeros in $\mathbf{x}$
- More interesting example

$$\text{Solve } \mathbf{B}\mathbf{x} = \mathbf{b} \text{ for sparse } \mathbf{B} \text{ and } \mathbf{b}$$

- Cannot expect $\mathbf{B}^{-1}$ to be sparse so cannot expect $\mathbf{x}$ to be sparse

What is hyper-sparsity?

- Hyper-sparsity occurs when the result of a sparse matrix-vector computation has sufficiently few nonzero values to be worth exploiting, both when **forming** and **using** the result
- Simple example

$$\mathbf{y} = \mathbf{A}\mathbf{x} \text{ for sparse } \mathbf{A} \text{ and } \mathbf{x}$$

- Can expect $\mathbf{y}$ to be sparse
- **Inefficient:** If $\mathbf{A}$ is packed row-wise and $\mathbf{x}$ is full-length, most operations with nonzeros in $\mathbf{A}$ are with zeros in $\mathbf{x}$
- **Efficient:** Pack $\mathbf{A}$ column-wise and combine the columns corresponding to nonzeros in $\mathbf{x}$
- More interesting example

$$\text{Solve } \mathbf{B}\mathbf{x} = \mathbf{b} \text{ for sparse } \mathbf{B} \text{ and } \mathbf{b}$$

- Cannot expect $\mathbf{B}^{-1}$ to be sparse so cannot expect $\mathbf{x}$ to be sparse
- When $\mathbf{B}$ is a simplex method basis matrix, structure often yields sparse $\mathbf{B}^{-1}$ and $\mathbf{x}$

What is hyper-sparsity?

- Hyper-sparsity occurs when the result of a sparse matrix-vector computation has sufficiently few nonzero values to be worth exploiting, both when **forming** and **using** the result
- Simple example

$$\mathbf{y} = \mathbf{A}\mathbf{x} \text{ for sparse } \mathbf{A} \text{ and } \mathbf{x}$$

- Can expect $\mathbf{y}$ to be sparse
- **Inefficient:** If $\mathbf{A}$ is packed row-wise and $\mathbf{x}$ is full-length, most operations with nonzeros in $\mathbf{A}$ are with zeros in $\mathbf{x}$
- **Efficient:** Pack $\mathbf{A}$ column-wise and combine the columns corresponding to nonzeros in $\mathbf{x}$
- More interesting example

$$\text{Solve } \mathbf{B}\mathbf{x} = \mathbf{b} \text{ for sparse } \mathbf{B} \text{ and } \mathbf{b}$$

- Cannot expect $\mathbf{B}^{-1}$ to be sparse so cannot expect $\mathbf{x}$ to be sparse
- When $\mathbf{B}$ is a simplex method basis matrix, structure often yields sparse $\mathbf{B}^{-1}$ and $\mathbf{x}$
- Simplex method performance has been transformed by exploiting hyper-sparsity
Hall and McKinnon: COAP **32** (2005) 259–283



Hyper-sparsity in operations with B^{-1}

Represent B^{-1} in product form with pivot η_k in row p_k and multipliers η_k for $k = 1, r$

Hyper-sparsity in operations with B^{-1}

Represent B^{-1} in product form with pivot η_k in row p_k and multipliers η_k for $k = 1, r$

- Traditional technique for solving $Bx = b$ by transforming b into x :

For $k = 1, r$ do

$$b_{p_k} := b_{p_k} / \eta_k; \quad b := b - b_{p_k} \eta_k$$

End do

Hyper-sparsity in operations with B^{-1}

Represent B^{-1} in product form with pivot η_k in row p_k and multipliers η_k for $k = 1, r$

- Traditional technique for solving $Bx = b$ by transforming b into x :

For $k = 1, r$ do

$$b_{p_k} := b_{p_k} / \eta_k; \quad b := b - b_{p_k} \eta_k$$

End do

- When b is sparse there is no need to apply η_k if b_{p_k} is zero:

For $k = 1, r$ do

$$\text{If } (b_{p_k} \neq 0) \quad b_{p_k} := b_{p_k} / \eta_k; \quad b := b - b_{p_k} \eta_k$$

End do

Hyper-sparsity in operations with B^{-1}

Represent B^{-1} in product form with pivot η_k in row p_k and multipliers η_k for $k = 1, r$

- Traditional technique for solving $Bx = b$ by transforming b into x :

For $k = 1, r$ do

$$b_{p_k} := b_{p_k} / \eta_k; \quad b := b - b_{p_k} \eta_k$$

End do

- When b is sparse there is no need to apply η_k if b_{p_k} is zero:

For $k = 1, r$ do

$$\text{If } (b_{p_k} \neq 0) \quad b_{p_k} := b_{p_k} / \eta_k; \quad b := b - b_{p_k} \eta_k$$

End do

- When x is sparse the dominant cost is the test for zero

Hyper-sparsity in operations with B^{-1}

Represent B^{-1} in product form with pivot η_k in row p_k and multipliers η_k for $k = 1, r$

- Traditional technique for solving $Bx = b$ by transforming b into x :

For $k = 1, r$ do

$$b_{p_k} := b_{p_k} / \eta_k; \mathbf{b} := \mathbf{b} - b_{p_k} \boldsymbol{\eta}_k$$

End do

- When b is sparse there is no need to apply η_k if b_{p_k} is zero:

For $k = 1, r$ do

$$\text{If } (b_{p_k} \neq 0) \quad b_{p_k} := b_{p_k} / \eta_k; \mathbf{b} := \mathbf{b} - b_{p_k} \boldsymbol{\eta}_k$$

End do

- When x is sparse the dominant cost is the test for zero
- (Aim to) identify vectors η_k to be applied at a cost proportional to arithmetic operations
Gilbert and Peierls: SISC **9** (1988) 862–874
Hall and McKinnon: COAP **32** (2005) 259–283



Exploiting hyper-sparsity when solving $Bx = b$

Let $\mathcal{R}$ be the set of indices i of nonzeros in b

Let $\mathcal{E}$ be the set of indices j of pivotal rows p_j with nonzeros in b

Exploiting hyper-sparsity when solving $Bx = b$

Let $\mathcal{R}$ be the set of indices i of nonzeros in b

Let $\mathcal{E}$ be the set of indices j of pivotal rows p_j with nonzeros in b

1 Scan $\mathcal{E}$ for the index k of the next vector η_k to be applied

If (k is undefined) $Bx = b$ has been solved; stop

Exploiting hyper-sparsity when solving $Bx = b$

Let $\mathcal{R}$ be the set of indices i of nonzeros in b

Let $\mathcal{E}$ be the set of indices j of pivotal rows p_j with nonzeros in b

1 Scan $\mathcal{E}$ for the index k of the next vector η_k to be applied

If (k is undefined) $Bx = b$ has been solved; stop

$$b_{p_k} := b_{p_k} / \eta_k$$

$$b := b - b_{p_k} \eta_k$$

Exploiting hyper-sparsity when solving $Bx = b$

Let $\mathcal{R}$ be the set of indices i of nonzeros in b

Let $\mathcal{E}$ be the set of indices j of pivotal rows p_j with nonzeros in b

1 Scan $\mathcal{E}$ for the index k of the next vector η_k to be applied

If (k is undefined) $Bx = b$ has been solved; stop

$$b_{p_k} := b_{p_k} / \eta_k$$

$$b := b - b_{p_k} \eta_k$$

For all nonzeros b_i created in b do

 Add i to $\mathcal{R}$

 Add any p_j to $\mathcal{E}$ if $p_j = i$ and $j > k$

End do

Go to 1

Exploiting hyper-sparsity when solving $Bx = b$

Let $\mathcal{R}$ be the set of indices i of nonzeros in b

Let $\mathcal{E}$ be the set of indices j of pivotal rows p_j with nonzeros in b

1 Scan $\mathcal{E}$ for the index k of the next vector η_k to be applied

If (k is undefined) $Bx = b$ has been solved; stop

$$b_{p_k} := b_{p_k} / \eta_k$$

$$b := b - b_{p_k} \eta_k$$

For all nonzeros b_i created in b do

 Add i to $\mathcal{R}$

 Add any p_j to $\mathcal{E}$ if $p_j = i$ and $j > k$

End do

Go to 1

- Gilbert and Peierls analyse the elimination tree to identify all vectors η_k to be applied *before* applying any



Interior point methods (IPM)

- When solving the LP problem

$$\begin{array}{ll} \text{minimize} & f = \mathbf{c}^T \mathbf{x} \\ \text{subject to} & \mathbf{Ax} = \mathbf{b} \quad \mathbf{x} \geq \mathbf{0} \end{array}$$

Interior point methods (IPM)

- When solving the LP problem

$$\begin{array}{ll} \text{minimize} & f = \mathbf{c}^T \mathbf{x} \\ \text{subject to} & \mathbf{Ax} = \mathbf{b} \quad \mathbf{x} \geq 0 \end{array}$$

- Each iteration, interior point methods require the solution of the **augmented system**

$$\begin{bmatrix} \Theta^{-1} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} -\Delta \mathbf{x} \\ \Delta \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix}$$

Interior point methods (IPM)

- When solving the LP problem

$$\begin{array}{ll} \text{minimize} & f = \mathbf{c}^T \mathbf{x} \\ \text{subject to} & \mathbf{Ax} = \mathbf{b} \quad \mathbf{x} \geq 0 \end{array}$$

- Each iteration, interior point methods require the solution of the **augmented system**

$$\begin{bmatrix} \Theta^{-1} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} -\Delta \mathbf{x} \\ \Delta \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix}$$

- Usually achieved by solving $A\Theta A^T \Delta \mathbf{y} = A\Theta \mathbf{f} - \mathbf{g}$ using $A\Theta A^T = L^T L$
- $A\Theta A^T$ can fill in badly so look for alternatives

Interior point methods (IPM)

- When solving the LP problem

$$\begin{array}{ll} \text{minimize} & f = \mathbf{c}^T \mathbf{x} \\ \text{subject to} & \mathbf{Ax} = \mathbf{b} \quad \mathbf{x} \geq 0 \end{array}$$

- Each iteration, interior point methods require the solution of the **augmented system**

$$\begin{bmatrix} \Theta^{-1} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} -\Delta \mathbf{x} \\ \Delta \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix}$$

- Usually achieved by solving $A\Theta A^T \Delta \mathbf{y} = A\Theta \mathbf{f} - \mathbf{g}$ using $A\Theta A^T = L^T L$
- $A\Theta A^T$ can fill in badly so look for alternatives
- $K = \begin{bmatrix} \Theta^{-1} & A^T \\ A & 0 \end{bmatrix}$ is indefinite but preconditioned CG works if $\mathbf{x}^{(0)}$ in appropriate space

Lukšan and Vlček: NLAA **5** (1998) 219–247

Rozložník and Simoncini: SIMAX **24** (2002) 368–391



Design of the preconditioner

- **Observation:**

- Convergence of IPM yields (simplex-like) partition $\Theta^{-1} = [\Theta_B^{-1} \quad \Theta_N^{-1}]$
- $\Theta_B^{-1} \rightarrow 0$

Design of the preconditioner

- **Observation:**

- Convergence of IPM yields (simplex-like) partition $\Theta^{-1} = [\Theta_B^{-1} \quad \Theta_N^{-1}]$
- $\Theta_B^{-1} \rightarrow 0$

- **In theory:**

- Order A by increasing values of Θ^{-1}
- Partition ordered A as $[B \quad N]$ and precondition

$$K = \begin{bmatrix} \Theta_B^{-1} & \Theta_N^{-1} & B^T \\ B & N & 0 \end{bmatrix} \text{ with } P = \begin{bmatrix} & & B^T \\ B & N & 0 \end{bmatrix}$$

Design of the preconditioner

- **Observation:**

- Convergence of IPM yields (simplex-like) partition $\Theta^{-1} = [\Theta_B^{-1} \quad \Theta_N^{-1}]$
- $\Theta_B^{-1} \rightarrow 0$

- **In theory:**

- Order A by increasing values of Θ^{-1}
- Partition ordered A as $[B \quad N]$ and precondition

$$K = \begin{bmatrix} \Theta_B^{-1} & \Theta_N^{-1} & B^T \\ B & N & 0 \end{bmatrix} \text{ with } P = \begin{bmatrix} & \Theta_N^{-1} & B^T \\ B & N & 0 \end{bmatrix}$$

- **In practice:**

- First m columns of ordered A are typically rank-deficient
- Form B from the first m linearly-independent columns of ordered A

Design of the preconditioner

- **Observation:**

- Convergence of IPM yields (simplex-like) partition $\Theta^{-1} = [\Theta_B^{-1} \quad \Theta_N^{-1}]$
- $\Theta_B^{-1} \rightarrow 0$

- **In theory:**

- Order A by increasing values of Θ^{-1}
- Partition ordered A as $[B \quad N]$ and precondition

$$K = \begin{bmatrix} \Theta_B^{-1} & & B^T \\ & \Theta_N^{-1} & N^T \\ B & N & 0 \end{bmatrix} \text{ with } P = \begin{bmatrix} & & B^T \\ & \Theta_N^{-1} & N^T \\ B & N & 0 \end{bmatrix}$$

- **In practice:**

- First m columns of ordered A are typically rank-deficient
- Form B from the first m linearly-independent columns of ordered A
- Once B is identified, exploit hyper-sparsity in forming a factored representation of B^{-1}



Identifying B

- Assume A is ordered by increasing values of Θ^{-1}
- Identify B by applying Gaussian elimination to A

Identifying B

- Assume A is ordered by increasing values of Θ^{-1}
- Identify B by applying Gaussian elimination to A
 - After k pivots have been found, let L_k be the unit lower triangular matrix of multipliers
 - Let $\mathbf{a}_j$ be the next column of A to be considered
 - Compute

$$L_k \hat{\mathbf{a}}_j = \mathbf{a}_j$$

Identifying B

- Assume A is ordered by increasing values of Θ^{-1}
- Identify B by applying Gaussian elimination to A
 - After k pivots have been found, let L_k be the unit lower triangular matrix of multipliers
 - Let $\mathbf{a}_j$ be the next column of A to be considered
 - Compute

$$L_k \hat{\mathbf{a}}_j = \mathbf{a}_j$$

- Determine the non-pivotal row p for which $|\hat{\mathbf{a}}_{ij}|$ is maximal

Identifying B

- Assume A is ordered by increasing values of Θ^{-1}
- Identify B by applying Gaussian elimination to A
 - After k pivots have been found, let L_k be the unit lower triangular matrix of multipliers
 - Let $\mathbf{a}_j$ be the next column of A to be considered
 - Compute

$$L_k \hat{\mathbf{a}}_j = \mathbf{a}_j$$

- Determine the non-pivotal row p for which $|\hat{\mathbf{a}}_{ij}|$ is maximal
If $|\hat{\mathbf{a}}_{pj}|$ is too small then reject $\mathbf{a}_j$ as a column of B

Identifying B

- Assume A is ordered by increasing values of Θ^{-1}
- Identify B by applying Gaussian elimination to A
 - After k pivots have been found, let L_k be the unit lower triangular matrix of multipliers
 - Let $\mathbf{a}_j$ be the next column of A to be considered
 - Compute

$$L_k \hat{\mathbf{a}}_j = \mathbf{a}_j$$

- Determine the non-pivotal row p for which $|\hat{\mathbf{a}}_{ij}|$ is maximal
If $|\hat{\mathbf{a}}_{pj}|$ is too small then reject $\mathbf{a}_j$ as a column of B

Otherwise,

Let $\mathbf{a}_j$ be column $k + 1$ of B

Row p is now pivotal

Form L_{k+1} from L_k , using the values of $\hat{\mathbf{a}}_{ij} / \hat{\mathbf{a}}_{pj}$ in non-pivotal rows

Identifying B

- Assume A is ordered by increasing values of Θ^{-1}
- Identify B by applying Gaussian elimination to A
 - After k pivots have been found, let L_k be the unit lower triangular matrix of multipliers
 - Let $\mathbf{a}_j$ be the next column of A to be considered
 - Compute

$$L_k \hat{\mathbf{a}}_j = \mathbf{a}_j$$

- Determine the non-pivotal row p for which $|\hat{\mathbf{a}}_{ij}|$ is maximal
If $|\hat{\mathbf{a}}_{pj}|$ is too small then reject $\mathbf{a}_j$ as a column of B

Otherwise,

Let $\mathbf{a}_j$ be column $k + 1$ of B

Row p is now pivotal

Form L_{k+1} from L_k , using the values of $\hat{\mathbf{a}}_{ij} / \hat{\mathbf{a}}_{pj}$ in non-pivotal rows

- Questions
 - How can this be done efficiently?
 - What effect does this have on the PCG-based IPM solver?



Exploiting hyper-sparsity when identifying B

For LP problem pds-10: 16558 rows, 66490 columns, 157790 nonzeros

Operation	CPU	%
Solve $L_k \hat{\mathbf{a}}_j = \mathbf{a}_j$	64.93	47.0
Find $\hat{\mathbf{a}}_{pj}$	72.61	52.6
Other	0.22	0.2
Factor B	0.24	0.2
Total	138.00	

Exploiting hyper-sparsity when identifying B

For LP problem pds-10: 16558 rows, 66490 columns, 157790 nonzeros

Operation	CPU	%
Solve $L_k \hat{\mathbf{a}}_j = \mathbf{a}_j$	64.93	47.0
Find $\hat{\mathbf{a}}_{pj}$	72.61	52.6
Other	0.22	0.2
Factor B	0.24	0.2
Total	138.00	

Operation	CPU	%
Solve $L_k \hat{\mathbf{a}}_j = \mathbf{a}_j$	65.20	99.1
Find $\hat{\mathbf{a}}_{pj}$	0.15	0.2
Other	0.21	0.3
Factor B	0.24	0.4
Total	65.81	

Exploiting hyper-sparsity when identifying B

For LP problem pds-10: 16558 rows, 66490 columns, 157790 nonzeros

Operation	CPU	%
Solve $L_k \hat{\mathbf{a}}_j = \mathbf{a}_j$	64.93	47.0
Find $\hat{\mathbf{a}}_{pj}$	72.61	52.6
Other	0.22	0.2
Factor B	0.24	0.2
Total	138.00	

Operation	CPU	%
Solve $L_k \hat{\mathbf{a}}_j = \mathbf{a}_j$	65.20	99.1
Find $\hat{\mathbf{a}}_{pj}$	0.15	0.2
Other	0.21	0.3
Factor B	0.24	0.4
Total	65.81	

Operation	CPU	%
Solve $L_k \hat{\mathbf{a}}_j = \mathbf{a}_j$	0.71	58.1
Find $\hat{\mathbf{a}}_{pj}$	0.09	7.3
Other	0.18	15.0
Factor B	0.24	19.7
Total	1.22	

Exploiting hyper-sparsity when identifying B

For LP problem pds-10: 16558 rows, 66490 columns, 157790 nonzeros

Operation	CPU	%
Solve $L_k \hat{\mathbf{a}}_j = \mathbf{a}_j$	64.93	47.0
Find $\hat{\mathbf{a}}_{pj}$	72.61	52.6
Other	0.22	0.2
Factor B	0.24	0.2
Total	138.00	

Operation	CPU	%
Solve $L_k \hat{\mathbf{a}}_j = \mathbf{a}_j$	65.20	99.1
Find $\hat{\mathbf{a}}_{pj}$	0.15	0.2
Other	0.21	0.3
Factor B	0.24	0.4
Total	65.81	

Operation	CPU	%
Solve $L_k \hat{\mathbf{a}}_j = \mathbf{a}_j$	0.71	58.1
Find $\hat{\mathbf{a}}_{pj}$	0.09	7.3
Other	0.18	15.0
Factor B	0.24	19.7
Total	1.22	

Exploiting hyper-sparsity has reduced the time to identify B by a factor of 141



Speed-up in the time to identify B

Model	Rows	Columns	Nonzeros	Speed-up
dbir2	18879	64729	1186221	18
deteq8	20678	76905	181085	112
deteq27	68672	255600	602600	232
nsct2	22983	60526	731941	40
pds-10	16558	66490	157790	141
qap12	3192	12048	47436	1
route	20894	63913	250015	146
storm-27	14387	51872	132731	91

Cost of forming preconditioner within IPM solution time

Model	Before (%)	After (%)
dbir2	36	3
deteq8	84	3
deteq27	98	4
nsct2	27	1
pds-10	87	2
qap12	41	42
route	83	2
storm-27	75	2

Effect of exploiting hyper-sparsity on PCG-based IPM solver

Model	Before	After
dbir2	0.74	1.00
deteq8	0.17	0.68
deteq27	0.21	1.72
nsct2	1.65	2.97
pds-10	0.57	1.97
qap12	0.30	0.28
route	0.53	1.61
storm-27	0.65	1.53

Speed of PCG-based IPM solver relative to solver based on factoring $A\Theta A^T$



Conclusions

- Without exploiting hyper-sparsity, identifying B for the preconditioner can dominate the cost of the PCG-based IPM solver
- By exploiting hyper-sparsity, the cost of identifying B is reduced dramatically
- The PCG-based IPM solver can out-perform the solver based on factoring $A\Theta A^T$

Conclusions

- Without exploiting hyper-sparsity, identifying B for the preconditioner can dominate the cost of the PCG-based IPM solver
- By exploiting hyper-sparsity, the cost of identifying B is reduced dramatically
- The PCG-based IPM solver can out-perform the solver based on factoring $A\Theta A^T$

See

- **Slides:** <http://www.maths.ed.ac.uk/hall/Talks>
- **Paper:** Al-Jeiroudi, Gondzio and Hall, *Preconditioning Indefinite Systems in Interior Point Methods for Large Scale Linear Optimization*, Optimization Methods and Software (Accepted for publication)

Thank you



Conclusions

- Without exploiting hyper-sparsity, identifying B for the preconditioner can dominate the cost of the PCG-based IPM solver
- By exploiting hyper-sparsity, the cost of identifying B is reduced dramatically
- The PCG-based IPM solver can out-perform the solver based on factoring $A\Theta A^T$

Conclusions

- Without exploiting hyper-sparsity, identifying B for the preconditioner can dominate the cost of the PCG-based IPM solver
- By exploiting hyper-sparsity, the cost of identifying B is reduced dramatically
- The PCG-based IPM solver can out-perform the solver based on factoring $A\Theta A^T$

See

- **Slides:** <http://www.maths.ed.ac.uk/hall/Talks>
- **Paper:** Al-Jeiroudi, Gondzio and Hall, *Preconditioning Indefinite Systems in Interior Point Methods for Large Scale Linear Optimization*, Optimization Methods and Software (Accepted for publication)

Thank you

