On the efficient exploitation of sparsity

Andrea Walther
Institut für Mathematik
Universität Paderborn

Workshop on PDE constrained optimization 2009

Trier, June 3–5, 2009

Outline

Optimal Power Flow Problem

(Fabrice Zaoui, Laure Castaing, RTE France)

Task: Distribute power flow over given network

Difficulty: Unabservable areas due to

- lack of sensors
- error in data transmission
- ...

Approximate required data in unabservable areas ➡

$$\min f(x), \qquad c(x) = 0 \qquad h(x) \leq 0,$$
$$f : \mathbb{R}^n \to \mathbb{R}, \quad c : \mathbb{R}^n \to \mathbb{R}^m, \quad h : \mathbb{R}^n \to \mathbb{R}^p$$

Optimization Problem

- Task: $\min_x f(x)$ s.t. $c(x) = 0$
- Consider Lagrangian $\mathcal{L}(x, \lambda) = f(x) + \lambda^T c(x)$
- Solve

$$0 = [g(x, \lambda), c(x)] \equiv \left[\nabla f(x) + \lambda^T \nabla c(x),\ c(x)\right] \in \mathbb{R}^{n+m},$$

but how?

Optimization Problem

- Task: $\min_x f(x)$ s.t. $c(x) = 0$
- Consider Lagrangian $\mathcal{L}(x,\lambda) = f(x) + \lambda^T c(x)$
- Solve

$$0 = [g(x,\lambda), c(x)] \equiv \left[\nabla f(x) + \lambda^T \nabla c(x),\ c(x)\right] \in \mathbb{R}^{n+m},$$

  but how?

- Apply SQP method, i.e., apply iteration

$$\nabla^2_{x,\lambda}\mathcal{L}(x_k,\lambda_k)\, p_k^N = \begin{bmatrix} B(x_k,\lambda_k) & A(x_k)^T \\ A(x_k) & 0 \end{bmatrix} p_k^N = -\nabla_{x,\lambda}\mathcal{L}(x_k,\lambda_k)$$

- Quite often $B(x,\lambda)$, $A(x)$ are sparse !!

Optimal Power Flow (Discretizations)

n	m	p	$nnz(c)$	$nnz(h)$	$nnz(L)$	time
5,986	2,415	1,575	21,065	6,300	21,068	11
17,958	7,245	11,123	63,179	31,692	64,668	55
29,930	12,075	20,671	105,301	57,084	108,278	129
53,874	21,735	39,767	189,529	107,868	195,478	412
101,762	41,055	77,959	358,025	209,436	369,916	1326

using an interior point method (Zaoui 2008)

Optimal Power Flow (Discretizations)

n	m	p	$nnz(c)$	$nnz(h)$	$nnz(L)$	time
5,986	2,415	1,575	21,065	6,300	21,068	11
17,958	7,245	11,123	63,179	31,692	64,668	55
29,930	12,075	20,671	105,301	57,084	108,278	129
53,874	21,735	39,767	189,529	107,868	195,478	412
101,762	41,055	77,959	358,025	209,436	369,916	1326

using an interior point method (Zaoui 2008)

But:
Large amount of runtime needed for detection of sparsity pattern of Hessian!!

Computation of Sparse Derivative Matrices

$B(x, \lambda)$, $A(x)$ sparse ⟹ Direct sparse solves possible!!

Would like to compute $B(x, \lambda)$ and $A(x)$ *efficiently*

Computation of Sparse Derivative Matrices

$B(x, \lambda)$, $A(x)$ sparse ➡ Direct sparse solves possible!!

Would like to compute $B(x, \lambda)$ and $A(x)$ *efficiently*

Four step procedure:

$$\mathcal{L} \xrightarrow{1} P_{\mathcal{L}} \xrightarrow{2} S_{\mathcal{L}} \xrightarrow{3} \tilde{B} = BS_{\mathcal{L}} \xrightarrow{4} b_{vec}$$

$\mathcal{L}$	$P_{\mathcal{L}}$	$S_{\mathcal{L}}$	$\tilde{B} = BS_{\mathcal{L}}$	b_{vec}
function	sparsity pattern	seed matrix	compressed representation	nonzero entries

Unwrapping the Four-Step Procedure

- Step 1: Sparsity pattern detection ($P_{\mathcal{L}}$)
  Performed only once
  ➡ Algorithmic Differentiation (AD) to compute $P_{\mathcal{L}}$

Unwrapping the Four-Step Procedure

- Step 1: Sparsity pattern detection ($P_{\mathcal{L}}$)
  Performed only once
  ➤ Algorithmic Differentiation (AD) to compute $P_{\mathcal{L}}$

- Step 2: Seed matrix computation ($S_{\mathcal{L}} \in \{0, 1\}^{n \times p}$)
  Performed only once
  ➤ Uses graph coloring methods (direct, indirect)
  [Gebremedhin, Manne, Pothen '05]

Unwrapping the Four-Step Procedure

- Step 1: Sparsity pattern detection ($P_{\mathcal{L}}$)
  Performed only once
  ⟶ Algorithmic Differentiation (AD) to compute $P_{\mathcal{L}}$
- Step 2: Seed matrix computation ($S_{\mathcal{L}} \in \{0, 1\}^{n \times p}$)
  Performed only once
  ⟶ Uses graph coloring methods (direct, indirect) [Gebremedhin, Manne, Pothen '05]
- Step 3: Compressed matrix computation ($\tilde{B} = BS_{\mathcal{L}}$)
  Few derivative matrix-vector products evaluated for each x
  ⟶ AD for derivative matrix $\times$ vector products

Unwrapping the Four-Step Procedure

- Step 1: Sparsity pattern detection ($P_{\mathcal{L}}$)
  Performed only once
  ⟹ Algorithmic Differentiation (AD) to compute $P_{\mathcal{L}}$
- Step 2: Seed matrix computation ($S_{\mathcal{L}} \in \{0, 1\}^{n \times p}$)
  Performed only once
  ⟹ Uses graph coloring methods (direct, indirect)
  [Gebremedhin, Manne, Pothen '05]
- Step 3: Compressed matrix computation ($\tilde{B} = BS_{\mathcal{L}}$)
  Few derivative matrix-vector products evaluated for each x
  ⟹ AD for derivative matrix $\times$ vector products
- Step 4: Recovery of values of entries ($B(x, \lambda)$)
  ⟹ non trivial task for indirect methods

Assumptions:

- $f : \mathbb{R}^n \to \mathbb{R}, y = f(x)$, twice continuously differentiable
- function evaluation consists of unary or binary operations

Assumptions:

- $f : \mathbb{R}^n \to \mathbb{R}, y = f(x)$, twice continuously differentiable
- function evaluation consists of unary or binary operations

Algorithm I: Function Evaluation

$$
\begin{array}{l}
\textbf{for } i = 1, \ldots, n \\
\qquad v_{i-n} \quad = \quad x_i \\
\textbf{for } i = 1, \ldots, l \\
\qquad v_i \quad\;\;\; = \quad \varphi_i(v_j)_{j \prec i} \\
\; y = v_l
\end{array}
$$

with precedence relation $j \prec i$:

$$\varphi_i(v_j)_{j \prec i} = \varphi_i(v_j) \quad \text{or} \quad \varphi_i(v_j)_{j \prec i} = \varphi_i(v_j, v_l) \quad \text{with} \quad j, l < i$$

Nonlinear Interaction Domains

Index domains [Griewank 2000]:

$$\mathcal{X}_i \equiv \{j \leq n \, : \, j - n \prec^* i\} \quad \text{for} \quad i = 1-n, \ldots, l$$

One has:

$$\left\{ j \leq n \, : \, \frac{\partial v_i}{\partial x_j} \neq 0 \right\} \subseteq \mathcal{X}_i$$

Nonlinear Interaction Domains

Index domains [Griewank 2000]:

$$\mathcal{X}_i \equiv \{j \leq n : j - n \prec^* i\} \quad \text{for} \quad i = 1 - n, \ldots, l$$

One has:

$$\left\{ j \leq n : \frac{\partial v_i}{\partial x_j} \neq 0 \right\} \subseteq \mathcal{X}_i$$

For sparse Hessians additionally nonlinear interaction domains

$$\left\{ j \leq n : \frac{\partial^2 y}{\partial x_i \partial x_j} \neq 0 \right\} \subseteq \mathcal{N}_i$$

for $i = 1, \ldots, n$.

Theorem (Numerical Stability of Hessian Calculation)

The recovery routines for the computation of the compressed representation of the Hessians are numerical stable, i.e. the magnitude of the error associated with the computation of $H[i,j]$ is bounded by the product of $n_{T(h_i)}$, the number of vertices in the subtree *$T(h_i)$ of T, and a constant independent of T.*

Proof: [Gebremedhin, Pothen, Tarafdar, Walther 2009]

Theorem (Complexity result of Sparsity Pattern)

Let $OPS(NID)$ denote the number of operations needed to generate all $\mathcal{N}_i$, $1 \leq i \leq n$. Then, the inequality

$$OPS\big(NID\big) \leq 6(1 + \hat{n}) \sum_{i=1}^{l} \bar{n}_i$$

is valid, where l is the number of elemental functions evaluated to compute the function value, $\bar{n}_i = |\mathcal{X}_i|$, and $\hat{n} = \max_{1 \leq i \leq n} |\mathcal{N}_i|$.

Theorem (Complexity result of Sparsity Pattern)

Let $OPS(NID)$ denote the number of operations needed to generate all $\mathcal{N}_i$, $1 \leq i \leq n$. Then, the inequality

$$OPS\big(NID\big) \leq 6(1 + \hat{n}) \sum_{i=1}^{l} \bar{n}_i$$

is valid, where l is the number of elemental functions evaluated to compute the function value, $\bar{n}_i = |\mathcal{X}_i|$, and $\hat{n} = \max_{1 \leq i \leq n} |\mathcal{N}_i|$.

Proof: [Walther 2008]

Theorem (Complexity result of Sparsity Pattern)

Let $OPS(NID)$ denote the number of operations needed to generate all $\mathcal{N}_i$, $1 \leq i \leq n$. Then, the inequality

$$OPS(NID) \leq 6(1 + \hat{n}) \sum_{i=1}^{l} \bar{n}_i$$

is valid, where l is the number of elemental functions evaluated to compute the function value, $\bar{n}_i = |\mathcal{X}_i|$, and $\hat{n} = \max_{1 \leq i \leq n} |\mathcal{N}_i|$.

Proof: [Walther 2008]

For $\mathcal{L}(x, \lambda) = f(x) + \lambda^T c(x)$ detailed examination of $\sum_{i=1}^{l} \bar{n}_i$

Required runtime?

Required runtime?

For Lagrangian $\mathcal{L}(x, \lambda) = f(x) + \lambda^T c(x)$ usually

$\bar{n}_i = n$ for a considerable amount of intermediates

Required runtime?

For Lagrangian $\mathcal{L}(x, \lambda) = f(x) + \lambda^T c(x)$ usually

$\bar{n}_i = n$ for a considerable amount of intermediates

Alternative: Exploit additional structure!

- Compute sparsity pattern of objective S_f
- Detect linear constraints together with sparsity pattern of Jacobian
- If required, compute "sparsity pattern" of constraints S_c
- Compute sparsity pattern $S = S_f$ v S_c

with $\bar{n}_i \ll n$ for S_c in PDE constrained context!

Automatic differentiation by overloading in C++

➛ ADOL-C version 2.0

- reorganization of taping
  tape dependent information kept in separate structure
- different differentiation contexts ⇒
  - documented external function facility
  - documented fixpoint iteration facility
  - documented checkpointing facility based on revolve
- documented parallelization of derivative calculation
- coupled with ColPack for exploitation of sparsity
- available at COIN-OR since May 2009

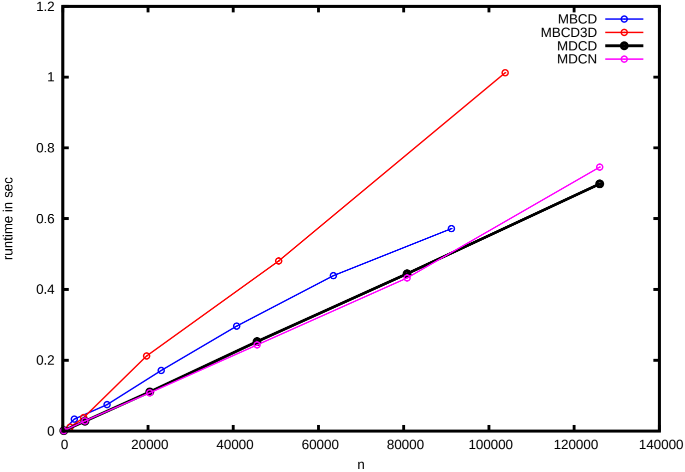
Testproblems

- Boundary Control with Dirichlet boundary conditions (2D)
- Boundary Control with Dirichlet boundary conditions (3D)
- Distributed Control with Dirichlet boundary conditions (2D)
- Distributed Control with Neumann boundary conditions (2D)

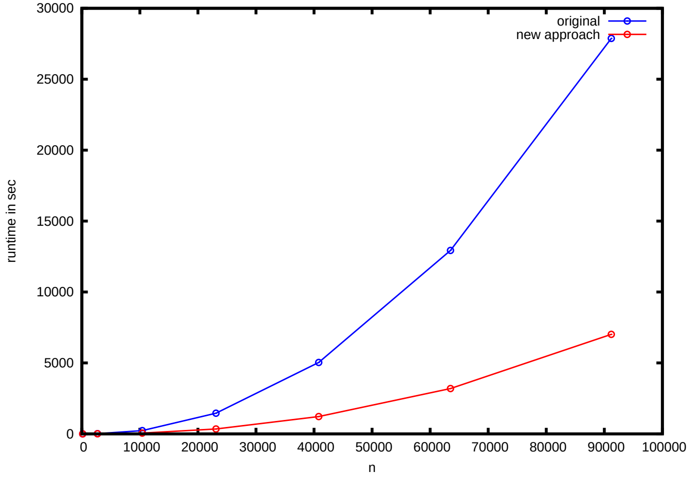
out of

Hans Mittelmann:
Optimization Techniques for Solving Elliptic Control Problems with Control and State Constraints. Part 1+2

Sparsity Pattern of Jacobian

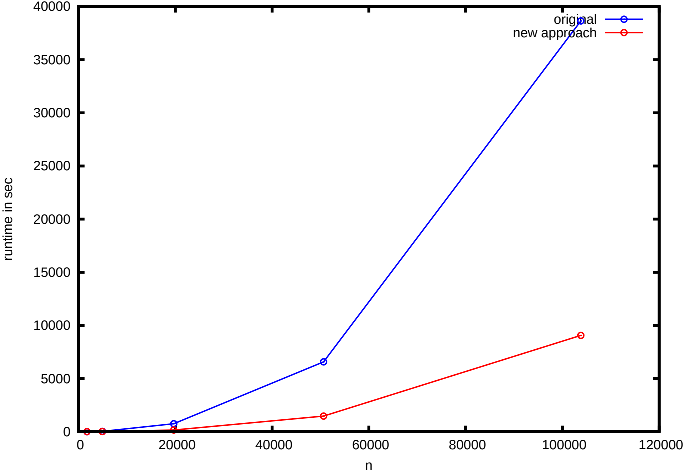


Boundary Control + Dirichlet conditions (2D)



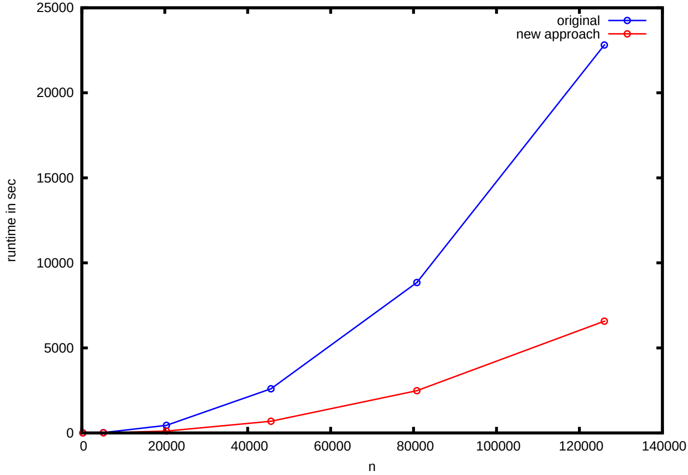
Linear constraints!

Boundary Control + Dirichlet conditions (3D)

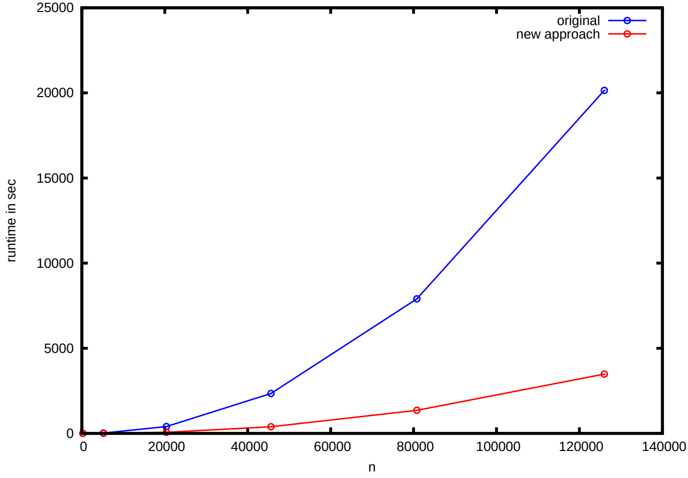


Linear constraints!

Distributed Control + Dirichlet conditions (2D)



Distributed Control + Neumann conditions (2D)



Conclusion and Outlook

- Analysis of sparsity detection routines + recovery
- ADOL-C coupled with COLPACK for graph coloring
- Runtimes for sparsity pattern detection of Jacobian OK
  numerical tests confirm majority of theoretical results
- Similar study on Hessian computation
  (Gebremedhin, Pothen, Tarafdar, Walther, 2009)
- Efficient sparsity detection for Hessians
  requires additional exploitation of structure for PDE-constrained optimization
- Coupling of ADOL-C with IPOPT for large scale optimization