

Adjoint Based Constrained Optimization

Andrea Walther and Andreas Griewank
Technische Universität Dresden

Optimization in Simulation-Based Models
IMA-Workshop, January 9 - 16, 2003

Table of Contents

1. Automatic Differentiation
2. Optimization Problem
3. Total Quasi-Newton Approach
4. Truncated Newton Approach
5. Conclusion and Outlook

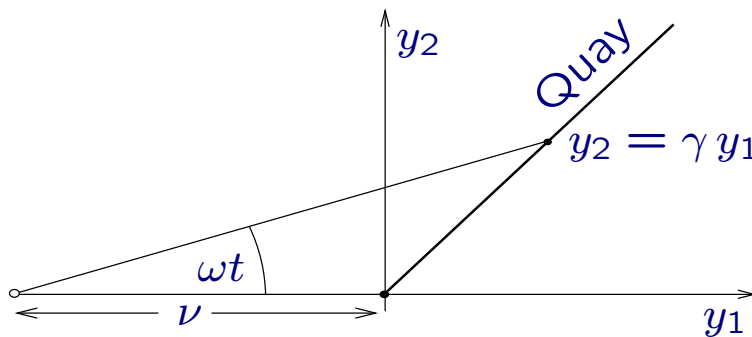
1. Automatic Differentiation

Given: Vector-valued function $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$

Required: Derivatives

Question: Representation of function evaluation?

Example: Lighthouse



$$y_1 = \frac{\nu * \tan(\omega * t)}{\gamma - \tan(\omega * t)}$$

$$y_2 = \frac{\gamma * \nu * \tan(\omega * t)}{\gamma - \tan(\omega * t)}$$

Evaluation Procedure for Lighthouse-Example:

v_{-3}	$=$	$x_1 = \nu$	
v_{-2}	$=$	$x_2 = \gamma$	
v_{-1}	$=$	$x_3 = \omega$	
v_0	$=$	$x_4 = t$	
v_1	$=$	$v_{-1} * v_0$	$\equiv \varphi_1(v_{-1}, v_0)$
v_2	$=$	$\tan(v_1)$	$\equiv \varphi_2(v_1)$
v_3	$=$	$v_{-2} - v_2$	$\equiv \varphi_3(v_{-2}, v_2)$
v_4	$=$	$v_{-3} * v_2$	$\equiv \varphi_4(v_{-3}, v_2)$
v_5	$=$	v_4 / v_3	$\equiv \varphi_5(v_4, v_3)$
v_6	$=$	v_5	$\equiv \varphi_6(v_5)$
v_7	$=$	$v_5 * v_{-2}$	$\equiv \varphi_7(v_5, v_{-2})$
y_1	$=$	v_6	
y_2	$=$	v_7	

Forward Differentiation of Lighthouse-Example

$$v_{-3} = x_1 = \nu$$

$$v_{-2} = x_2 = \gamma$$

$$v_{-1} = x_3 = \omega$$

$$v_0 = x_4 = t$$

$$\dot{v}_{-3} \equiv \dot{x}_1 = 0$$

$$\dot{v}_{-2} \equiv \dot{x}_2 = 0$$

$$\dot{v}_{-1} \equiv \dot{x}_3 = 0$$

$$\dot{v}_0 \equiv \dot{x}_4 = 1$$

$$v_1 = v_{-1} * v_0$$

$$v_2 = \tan(v_1)$$

$$v_3 = v_{-2} - v_2$$

$$v_4 = v_{-3} * v_2$$

$$v_5 = v_4 / v_3$$

$$v_6 = v_5 * v_{-2}$$

$$\dot{v}_1 = \dot{v}_{-1} * v_0 + v_{-1} * \dot{v}_0$$

$$\dot{v}_2 = \dot{v}_1 / \cos(v_1)^2$$

$$\dot{v}_3 = \dot{v}_{-2} - \dot{v}_2$$

$$\dot{v}_4 = \dot{v}_{-3} * v_2 + v_{-3} * \dot{v}_2$$

$$\dot{v}_5 = (\dot{v}_4 - \dot{v}_3 * v_5) * (1/v_3)$$

$$\dot{v}_6 = \dot{v}_5 * v_{-2} + v_5 * \dot{v}_{-2}$$

$$y_1 = v_5$$

$$y_2 = v_6$$

$$\dot{y}_1 = \dot{v}_5$$

$$\dot{y}_2 = \dot{v}_6$$

Adjoint Recursion of Lighthouse-Example

$$v_{-3} = x_1; \quad v_{-2} = x_2; \quad v_{-1} = x_3; \quad v_0 = x_4;$$

$$v_1 = v_{-1} * v_0;$$

$$v_2 = \tan(v_1);$$

$$v_3 = v_{-2} - v_2;$$

$$v_4 = v_{-3} * v_2;$$

$$v_5 = v_4 / v_3;$$

$$v_6 = v_5 * v_{-2};$$

$$y_1 = v_5; \quad y_2 = v_6;$$

$$\bar{v}_5 = \bar{y}_1; \quad \bar{v}_6 = \bar{y}_2;$$

$$\bar{v}_5 \dagger = \bar{v}_6 * v_{-2}; \quad \bar{v}_{-2} \dagger = \bar{v}_6 * v_5;$$

$$\bar{v}_4 \dagger = \bar{v}_5 / v_3; \quad \bar{v}_3 \dagger = \bar{v}_5 * v_5 / v_3;$$

$$\bar{v}_{-3} \dagger = \bar{v}_4 * v_2; \quad \bar{v}_2 \dagger = \bar{v}_4 * v_{-3};$$

$$\bar{v}_{-2} \dagger = \bar{v}_3; \quad \bar{v}_2 \dashv = \bar{v}_3;$$

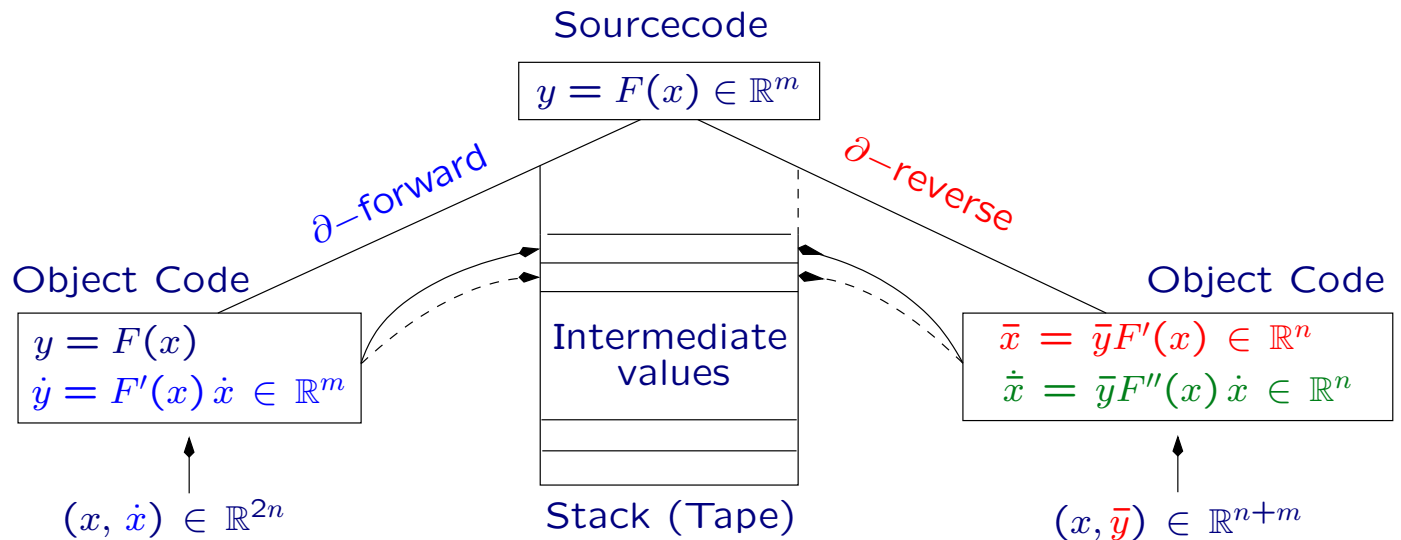
$$\bar{v}_1 \dagger = \bar{v}_2 / \cos^2(v_1);$$

$$\bar{v}_{-1} \dagger = \bar{v}_1 * v_0; \quad \bar{v}_0 \dagger = \bar{v}_1 * v_{-1};$$

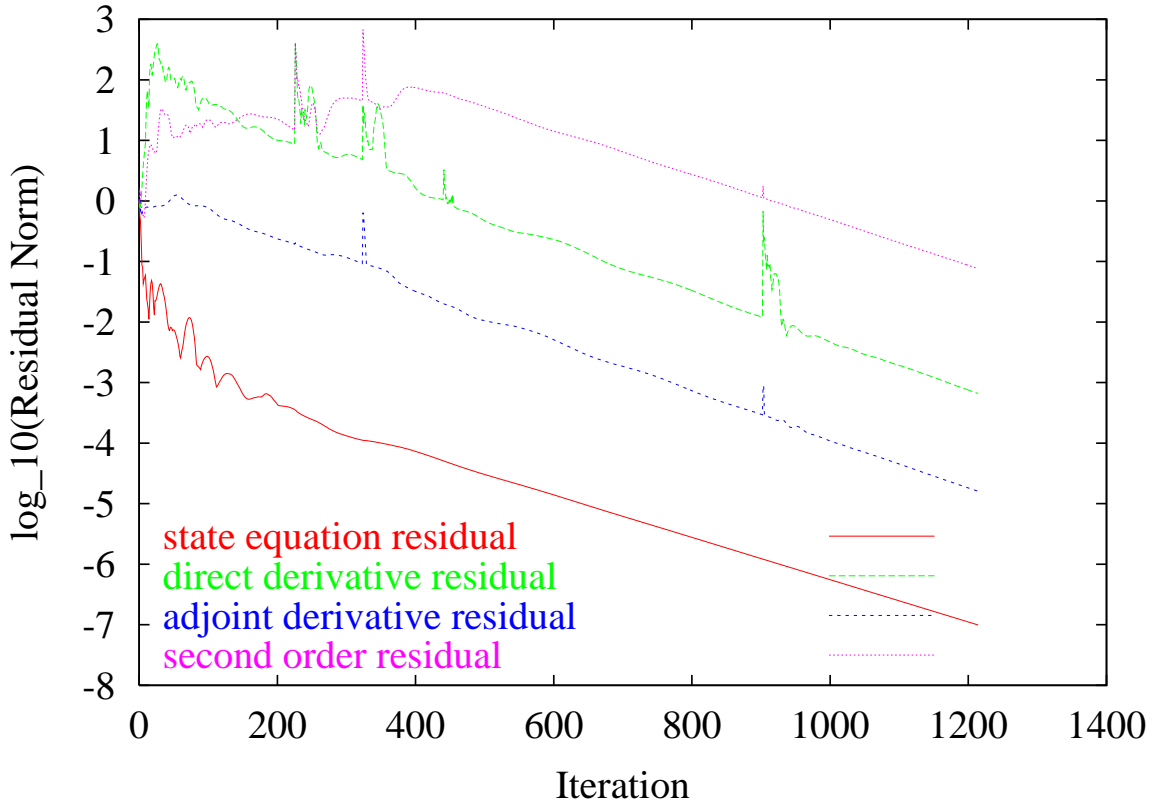
$$\bar{x}_4 = \bar{v}_0; \quad \bar{x}_3 = \bar{v}_{-1}; \quad \bar{x}_2 = \bar{v}_{-2}; \quad \bar{x}_1 = \bar{v}_{-3};$$

Second Derivatives ?

Combine forward and reverse differentiation



Convergence of Derivatives for Euler Code



⇒ AD for fixed point iterations

Conclusion I: Automatic Differentiation (AD)

- Evaluation of derivatives with working accuracy
- Forward mode: $\text{OPS}(F'(x)\dot{x}) \leq c \text{OPS}(F)$, $c \in [3, 5]$
- Reverse mode: $\text{OPS}(\bar{y} F'(x)) \leq c \text{OPS}(F)$, $c \in [3, 5]$
- Combination: $\text{OPS}(\bar{y} F''(x)\dot{x}) \leq c \text{OPS}(F)$, $c \in [7, 10]$
- Tools: ADIFOR, ADIC, ADOL-C, ODYSSEE, TAF, ...

⇒ Computation of “tangent” $F'(x)\dot{x}$, “gradient” $\bar{y} F'(x)$, and $\bar{y} F''(x)\dot{x}$ is cheap and facilitated by existing tools.

Question: New algorithms based on this information ??

2. Optimization Problem

Task:

$$\text{Min } f(x) \quad \text{subject to } c(x) = 0,$$

with nonlinear and sufficiently smooth

- $f : \mathbb{R}^n \rightarrow \mathbb{R}$ objective,
- $c : \mathbb{R}^n \rightarrow \mathbb{R}^m$ constraints.

Calculus-based methods:

- Consider Lagrangian $\mathcal{L}(x, \lambda) = f(x) + \lambda^T c(x)$
- Solve

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

but how?

Solution procedure involves Newton-direction p_k^N with

$$\nabla_{x,\lambda}^2 \mathcal{L}(x_k, \lambda_k) p_k^N = -\nabla_{x,\lambda} \mathcal{L}(x_k, \lambda_k)$$

or approximation p_k where

$$\nabla_{x,\lambda}^2 \mathcal{L}(x, \lambda) = \begin{bmatrix} \nabla^2 f(x) + \sum_{i=1}^m \lambda_i \nabla^2 c_i(x) & \nabla c(x)^T \\ \nabla c(x) & 0 \end{bmatrix} \in \mathbb{R}^{(n+m) \times (n+m)}$$

$$\nabla_{x,\lambda} \mathcal{L}(x, \lambda) = [g(x, \lambda), c(x)] = [\nabla f(x) + \lambda^T \nabla c(x), c(x)]^T \in \mathbb{R}^{n+m}$$

Computation of p_k^N

- by direct solver
 - Cost: cubic in dimension $n + m$
- by iterative solver
 - Take block structure into account !?
 - So far constraint Jacobian $\nabla c(x_k)$ required exactly
 - Need of suitable preconditioners

Question:

Are there alternatives such that forming and factoring the blocks of $\nabla_{x,\lambda}^2 \mathcal{L}(x, \lambda)$ at each iteration can be avoided?

3. Total Quasi-Newton Approach

So far: Partial Quasi-Newton Approach

$$\begin{bmatrix} B_k & A_k^T \\ A_k & 0 \end{bmatrix} \begin{bmatrix} s_k \\ \sigma_k \end{bmatrix} = - \begin{bmatrix} g_k \\ c_k \end{bmatrix}$$

with

- $A_k = \nabla c(x_k)$ assumed available, possibly by FD.
- Hessian approximation for example by

$$B_{k+1} = B_k + \varepsilon_k \frac{(w_k - B_k s_k)(w_k - B_k s_k)^T}{(w_k - B_k s_k)^T s_k}$$

with $s_k \equiv x_{k+1} - x_k$ to satisfy secant condition

$$B_{k+1} s_k = w_k \equiv g(x_k + s_k, \lambda_k) - g(x_k, \lambda_k) \in \mathbb{R}^n$$

Now: Two-sided Rank-One Update of A_k (TR1)

$$A_{k+1} = A_k + \delta_k \frac{(y_k - A_k s_k)(\mu_k^T - \sigma_k^T A_k)}{\mu_k^T s_k - \sigma_k^T A_k s_k}$$

to simultaneously satisfy direct secant condition

$$A_{k+1} s_k = y_k \equiv c(x_k + s_k) - c(x_k) \approx \nabla c(x_k) s_k$$

as well as adjoint = transposed secant condition

$$\sigma_k^T A_{k+1} = \mu_k^T \equiv g^T(x_k + s_k, \lambda_k + \sigma_k) - g^T(x_k + s_k, \lambda_k)$$

with $\sigma_k \equiv \lambda_{k+1} - \lambda_k \in \mathbb{R}^m$ and use of AD.

Assume affine constraints $c : \mathbb{R}^n \rightarrow \mathbb{R}^m$

Then one has:

- Exact consistency of secant conditions as

$$\sigma_k^T y_k = \sigma_k^T A s_k = \mu_k^T s_k.$$

Otherwise: Discrepancy of order $O(\|s_k\|^2)$

- Premature termination occurs if exactly

$$\sigma_k^T D_k s_k = 0 \quad \text{for} \quad D_k = A_k - A.$$

- Otherwise heredity and error rank reduction as

$$D_{k+1} = D_k - D_k s_k \sigma_k^T D_k / \sigma_k^T D_k s_k.$$

⇒ Exact Jacobian after m steps.

Termination after n steps in combination with SR1.

- Jacobian reduced to tridiagonal form as

$$\sigma_k^T A s_j = 0 \quad \text{when} \quad |j - k| > 1.$$

- Everything is invariant w.r.t. linear transformations !!

Provided damping factors ε_k and δ_k chosen accordingly.

First numerical results:

	n	m	Exact	T & W/G	STR1
Wright	2	1	5	6	10
Himmelblau	3	2	3	5	9
Powell	5	3	5	6	15
Wright	5	3	**	13	9
HS100	7	2	11	13	19
Byrd	2	1	9	7	10
Byrd	2	1	8	7	9
Yuan	2	1	6	9	9
Yuan	2	1	5	7	7
HS93	6	2	4	18	18*

Matlab function, no line search

Efficient C implementation?

Factorized Nullspace Representation of KKT matrix !!

One has with $d = n - m$

- For the Jacobian approximation

$$Y_k \in \mathbb{R}^{m \times n}, \quad L_k \in \mathbb{R}^{m \times m},$$

with Y_k orthonormal, L_k lower triangular, and $L_k Y_k = A_k$.

- For the Nullspace representation of Hessian approximation

$$Z_k \in \mathbb{R}^{d \times n}, \quad R_k \in \mathbb{R}^{d \times d},$$

with Z_k orthonormal, R_k lower triangular, and

$$R_k^T R_k = Z_k B_k Z_k^T, \quad Y_k Z_k^T = 0$$

- A cross term $C_k \in \mathbb{R}^{m \times n}$ with $C_k = Y_k B_k$.

One obtains

$$s_k = Y_k^T \tilde{c}_k - Z_k^T R_k^{-1} R_k^{-T} Z_k (g_k + C_k^T \tilde{c}_k) \quad ,$$

$$\sigma_k = -L_k^{-T} (C_k s_k + Y_k g_k) \quad \text{with} \quad \tilde{c}_k \equiv L_k^{-1} c_k \quad .$$

In addition:

- All matrices can be updated with quadratic cost.
- Results in rank-three update of complete KKT-matrix.
- Total storage $(m^2 + 3n^2)/2$ reducable to $mn + n^2/2$.

Choice of damping factors ε_k and δ_k ?

One has

$$\det \begin{bmatrix} B_k & A_k^T \\ A_k & 0 \end{bmatrix} = (-1)^m [\det(R_k) \det(L_k)]^2$$

⇒ Nonsingularity of KKT-system by monitoring $\det R_k$, $\det L_k$

Current implementation STR1:

- Uses ADOL-C to provide required derivatives
- Computes appropriate initialization of A_0 and sets $B_0 = I_n$
- Ensures that determinant of KKT-matrix varies only by factor in given interval $[\tau, 1/\tau]$
- Performs cubic line search

Numerical Results: Some Cute test problems

Problem	type	n	m	STR1		Lancelot	
				init	iter.	o.it.	CG
bt3	SLR	3	3	3	3	16	12
bt5	QQR	3	2	2	7	21	23
bt7	OQR	5	3	3	*	55	53
bt12	QQR	5	3	3	6	22	18
aug2d	QLR	212	96	96	11	19	404
dtoc1nc	OQR	300	200	*	*	47	415
dtoc3	QLR	296	197	197	22	24	56
dtoc5	QQR	298	149	149	19	21	64

Lancelot: Default preconditioning, exact first derivatives + SR1

Conclusion II: Total Quasi-Newton Approach STR1

- Optimization without constraint Jacobian feasible.
Thanks to reverse mode of AD $\Rightarrow \lambda^T \nabla c(x)$.
- Cost per step amounts to a few evaluations of (f, c)
and linear algebra of $O(n + m)^2$.
- Implementation with $O(k(n + m))$ storage possible.
- Lagrange multipliers become true iterates.
- Identification of active inequality constraints
probably a serious challenge.

4. Truncated Newton Approach

Apparently not yet widely used in constrained case:

Start: Initial point x_0

for $k = 0, 1, \dots$

 Compute $p_k = (s_k, \sigma_k)^T$ using ?? for

$$\nabla_{x,\lambda}^2 \mathcal{L}(x_k, \lambda_k) p_k = -\nabla_{x,\lambda} \mathcal{L}(x_k, \lambda_k)$$

 Terminate when $\|r_k\|$ small

 Set $x_{k+1} = x_k + \alpha_k s_k$, α_k chosen by line search

Questions:

- Hessian $\nabla_{x,\lambda}^2 \mathcal{L}(x_k, \lambda_k)$ symmetric, indefinite \Rightarrow Solver ?
- Computation of $\nabla_{x,\lambda}^2 \mathcal{L}(x_k, \lambda_k) p_k$?

Solver: QMR, MINRES

Product $\nabla_{x,\lambda}^2 \mathcal{L}(x_k, \lambda_k) p_k$:

- LKNS [Biros, Ghattas]:
Adjoint PDE + QN-approximation,
solve complete system (+ Preconditioning)
- Second order adjoint procedure of AD:
Choosing $\bar{\mathcal{L}} = 1$, $\dot{x} = s_k \in \mathbb{R}^n$, $\dot{\lambda} = \sigma_k \in \mathbb{R}^m$ yields

$$(\bar{x}, \bar{\lambda}) = \bar{\mathcal{L}} \mathcal{L}'(x, \lambda) = (\nabla f(x) + \sum_{i=1}^m \lambda_i \nabla c_i(x), c(x))$$

and

$$(\dot{\bar{x}}, \dot{\bar{\lambda}}) = \bar{\mathcal{L}} \mathcal{L}''(x, \lambda) \begin{pmatrix} \dot{x} \\ \dot{\lambda} \end{pmatrix} = (\dot{x}^T \nabla_x^2 \mathcal{L}(x, \lambda) + \dot{\lambda}^T \nabla c(x), \dot{x}^T \nabla c(x)^T)$$

without any need to form or factor Jacobian $\nabla c(x)$.

AD approaches:

- Keyes et al. [2000]: LNKSS
 - MATLAB/ADMAT implementation with full A for preconditioning (as far as we know)
- Griewank/W.: ABICO
 - C code with ADOL-C for second-order adjoint
 - MINRES + truncated version
 - cubic line search on l_1 -merit function
 - so far: no preconditioning (\Rightarrow STR1?)

First Numerical Results I:

Problem	type	n	m	ABICO (tr)			same with FD		
				o.it.	i.it.	#gxp	o.it.	i.it.	#gxp
bt3	SLR	5	3	2	16	2	4	35	4
bt5	QQR	3	2	7	35	7	7	39	7
bt7	OQR	5	3	*	*	*	*	*	*
bt12	QQR	5	3	5	29	5	*	*	*
aug2d1	QLR	212	96	3	213	3	3	309	3
dtoc1nc	OQR	300	200	*	*	*	*	*	*
dtoc3	QLR	296	197	4	1027	2	3	1359	3
dtoc5	QQR	298	149	6	1523	5	*	*	*

First Numerical Results II:

Problem	type	n	m	ABICO (tr)			Lancelot		
				o.it.	i.it.	#gxp	o.it.	#J	CG
bt3	SLR	5	3	2	16	2	16	17	12
bt5	QQR	3	2	7	35	7	21	17	23
bt7	OQR	5	3	*	*	*	55	51	53
bt12	QQR	5	3	5	29	4	22	19	18
aug2d	QLR	212	96	3	213	3	19	20	404
dtoc1nc	OQR	300	200	*	*	*	47	37	415
dtoc3	QLR	296	197	4	1027	4	24	25	56
dtoc5	QQR	298	149	6	1523	6	21	22	64

Lancelot: Default preconditioning, exact first derivatives + SR1

Comparison STR1 vs. ABICO:

Problem	type	n	m	STR1		ABICO	
				init	iter.	o.it.	i.it.
bt3	SLR	3	3	3	3	2	16
bt5	QQR	3	2	2	7	7	35
bt7	OQR	5	3	3	*	*	*
bt12	QQR	5	3	3	6	5	29
aug2d	QLR	212	96	96	11	3	213
dtoc1nc	OQR	300	200	*	*	*	*
dtoc3	QLR	296	197	197	22	4	1027
dtoc5	QQR	298	149	149	19	6	1523

Conclusion III: Truncated Newton Approach ABICO

- Optimization without forming any derivative matrix.
Thanks to second order adjoint of AD $\Rightarrow \bar{x}, \bar{\lambda}, \dot{\bar{x}}, \dot{\bar{\lambda}}$.
- Cost per inner step amounts to a few evaluations of (f, c) and linear algebra of $O(n + m)$.
- Inexact solution using descent direction.
- Identification of active inequalities serious challenge.
- Theoretical convergence conditions?
Jäger/Sachs [1997], Heinkenschloss/Vicente [2001]

5. Conclusion and Outlook

- Derivation of Jacobian-free methods:
 - Total Quasi-Newton (STR1)
 - Truncated Newton (ABICO)

based on reverse mode of AD = vector adjoints.

- Promising numerical results for test problems.
- Vision: Combine both
 - Limited memory STR1 for preconditioning
 - ABICO for solving the preconditioned system