The Lifted Newton Method and Its Use in Optimization

Moritz Diehl

Optimization in Engineering Center (OPTEC), K.U. Leuven, Belgium

joint work with Jan Albersmeyer (U. Heidelberg)

ENSIACET, Toulouse, February 3-5, 2010

Overview

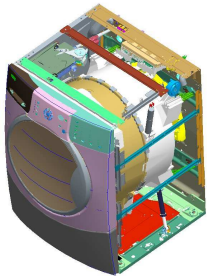
- **Nonlinear Optimization and Control Applications at OPTEC**
- Idea of Lifted Newton Method
- An algorithmic trick to exploit structure "for free"
- Application to optimization
- Convergence analysis
- Numerical Tests

Lifted Newton Method described in:

J.Albersmeyer & M.D. The Lifted Newton Method and Its Application in Optimization. *SIAM J. Opt.* (2010) 20:3,1655-1684

Nonlinear Optimization and Control Applications at OPTEC

Distillation column (Stuttgart; Leuven)



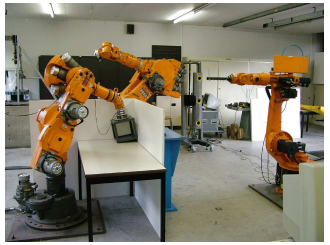
Washing machine design and control (with Bauknecht)

Chemical Process Control (with IPCOS)

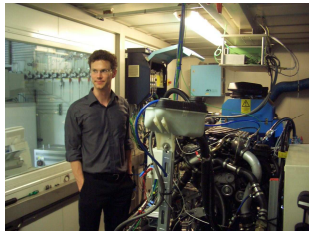
Combine Harvester Control (with Mebios/CNH)

Walking Robots (with Grenoble/Tsukuba)

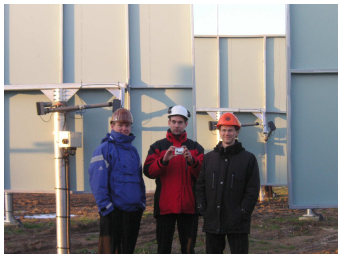
Power generating kites (Leuven)



Robot arm time optimal motion (Leuven)



Combustion Engines (Leuven; Linz; Hoerbiger, US)

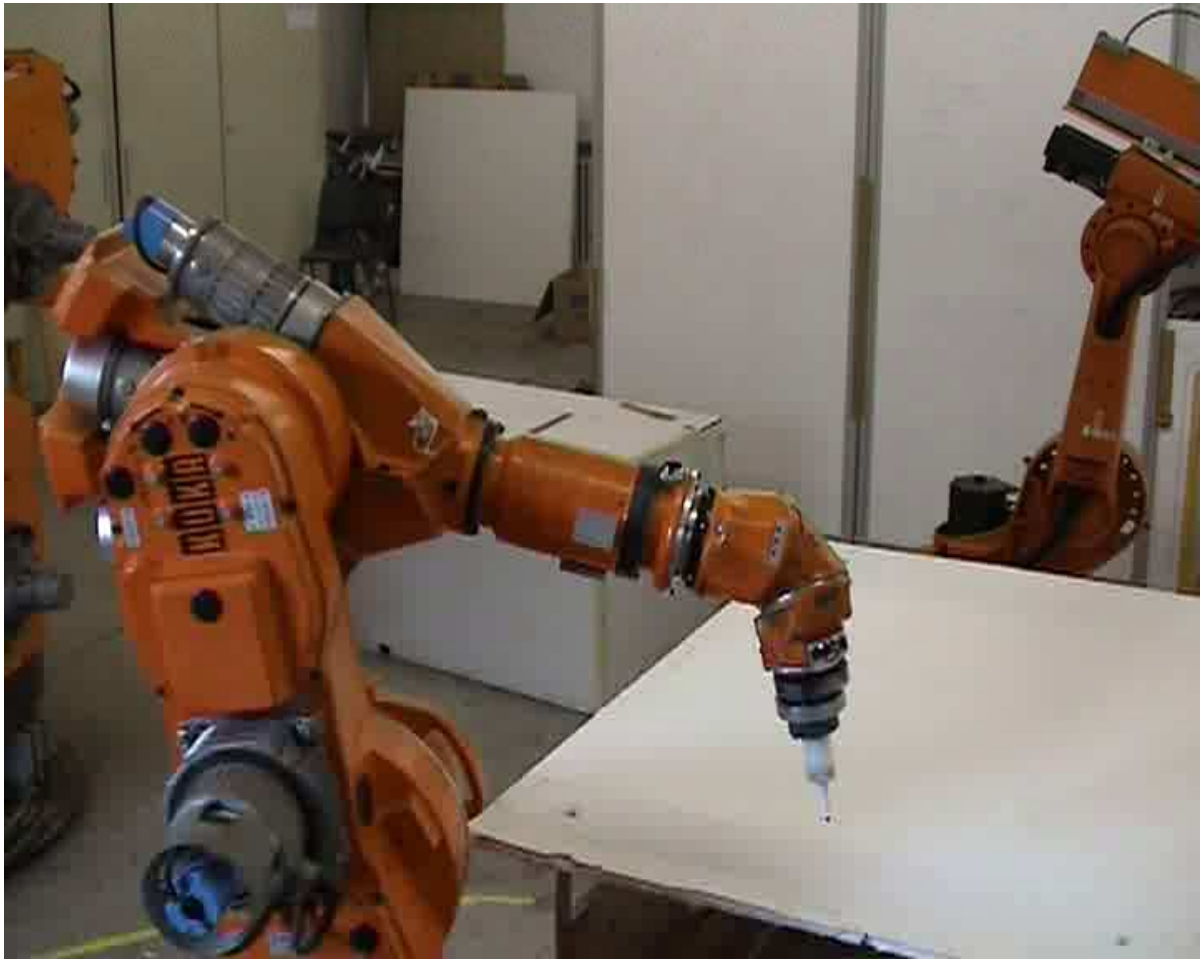


Solar Thermal Power Plant (Jülich)

Idea: use nonlinear dynamic simulation models for optimization!

Example: Time Optimal Robot Motions

Diederik Verscheure in front of Leuven's robots



Convex Reformulation allows computation of globally optimal paths in 2 milliseconds →

Each model based optimization application needs particular tricks...

Overview

- Nonlinear Optimization and Control Applications at OPTEC
- **Idea of Lifted Newton Method**
- An algorithmic trick to exploit structure "for free"
- Application to optimization
- Convergence analysis
- Numerical Tests

Lifted Newton Method described in:

J.Albersmeyer & M.D. The Lifted Newton Method and Its Application in Optimization. *SIAM J. Opt.* (2010) 20:3,1655-1684

Simplified Setting: Root Finding Problem

Aim: solve root finding problem

$$F(u) = 0$$

where $F \in \mathcal{C}^1(\mathbb{R}^{n_u}, \mathbb{R}^{n_u})$ is composed of several nonlinear subfunctions:

Simplified Setting: Root Finding Problem

Aim: solve root finding problem

$$F(u) = 0$$

where $F \in \mathcal{C}^1(\mathbb{R}^{n_u}, \mathbb{R}^{n_u})$ is composed of several nonlinear subfunctions:

Algorithm 1: Function with output of intermediate variables

Input : $u \in \mathbb{R}^{n_u}$
Output: $x_1 \in \mathbb{R}^{n_1}, \ldots, x_m \in \mathbb{R}^{n_m}, F \in \mathbb{R}^{n_u}$
begin
  for $i = 1, 2, \ldots, m$ **do**
    $x_i = f_i(u, x_1, x_2, \ldots, x_{i-1})$;
  end for
  $F := f_F(u, x_1, x_2, \ldots, x_m)$;
end

Idea: "Lift" root finding problem into a higher dimensional space…

Lifted Problem

The equivalent, "lifted" problem is:

$$G(u, x) = 0$$

with

$$G(u,x) = \begin{pmatrix} f_1(u) & - & x_1 \\ f_2(u, x_1) & - & x_2 \\ \vdots & & \\ f_m(u, x_1, \ldots, x_{m-1}) & - & x_m \\ f_F(u, x_1, \ldots, x_m) & & \end{pmatrix}$$

Why to lift and increase the system size?

- Lifting is a generalization of the well-known "multiple shooting" technique for solution of
  - boundary value problems [Osborne 1969]
  - parameter estimation problems in ODE [Bock 1987,Schloeder1988]
  - optimal control problems [Bock and Plitt 1984, Gill et al. 2000, Schaefer2005]

- Lifting often offers advantages in terms of
  - sparsity exploitation (not today's focus)
  - larger region of convergence
  - **more freedom for initialization (e.g. measurements)**
  - **faster local contraction rate**

Motivating Toy Example

- Original scalar root finding problem:

$$F(u) := u^{16} - 2 = 0$$

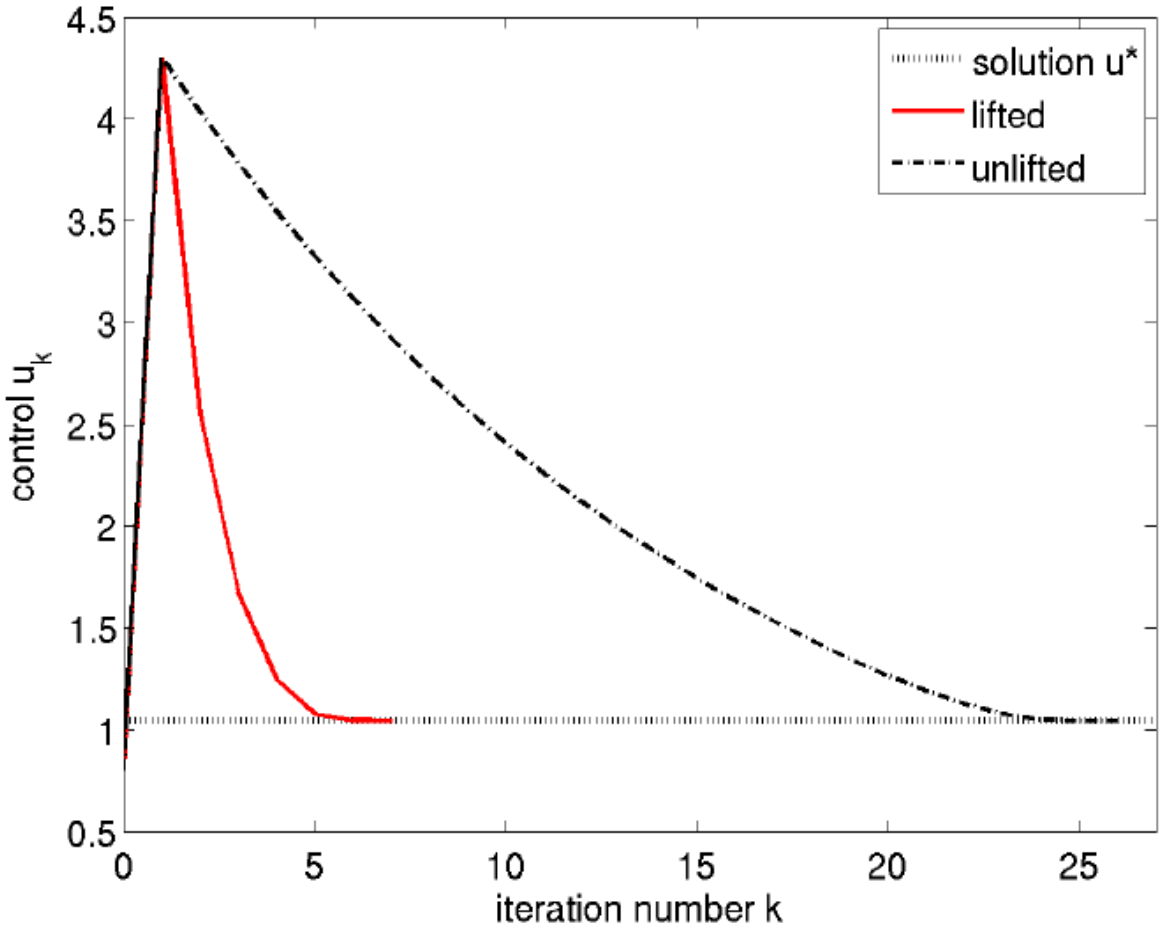
- Lifted formulation yields a nonlinear equation system with 5 equations in 5 variables:

$$\begin{aligned} x_1 &:= u^2 & x_2 &:= x_1^2 \\ x_3 &:= x_2^2 & x_4 &:= x_3^2 \\ F &:= x_4 - 2 \end{aligned}$$

- Compare lifted and unlifted Newton iterates with same initial guess. Obtain intermediate lifted variables by function evaluation.

Motivating Toy Example

- First iteration is identical, as we initialized identically
- Lifted Newton method converges in 7 instead of 26 iterations!



Lifted Problem is much larger ... is it more expensive?

In each lifted Newton iteration

$$\begin{pmatrix} x^{k+1} \\ u^{k+1} \end{pmatrix} = \begin{pmatrix} x^{k} \\ u^{k} \end{pmatrix} + \begin{pmatrix} \Delta x^{k} \\ \Delta u^{k} \end{pmatrix}$$

we have to solve a large linear system:

$$\begin{pmatrix} \Delta x^{k} \\ \Delta u^{k} \end{pmatrix} = - \left[\frac{\partial G}{\partial (u,x)} (x^k, u^k) \right]^{-1} G(x^k, u^k)$$

It is large and structured,..

… but exploitation algorithms are difficult to develop and implement, cf. Dissertations [Schloeder1988, Schaefer2005] in context of multiple shooting

Overview

- Nonlinear Optimization and Control Applications at OPTEC
- Idea of Lifted Newton Method
- **An algorithmic trick to exploit structure "for free"**
- Application to optimization
- Convergence analysis
- Numerical Tests

Algorithmic Trick: Preliminaries 1

Original "user function":

Algorithm 1: Function with output of intermediate variables

Input : $u \in \mathbb{R}^{n_u}$
Output: $x_1 \in \mathbb{R}^{n_1}, \ldots, x_m \in \mathbb{R}^{n_m}, F \in \mathbb{R}^{n_u}$
begin
  for $i = 1, 2, \ldots, m$ **do**
    $x_i = f_i(u, x_1, x_2, \ldots, x_{i-1})$;
  end for
  $F := f_F(u, x_1, x_2, \ldots, x_m)$;
end

"Lifted residual" (after minor code additions):

Algorithm 2: Residual function $G(u, y)$

Input : $u, y_1, \ldots, y_m$
Output: $G_1, \ldots, G_m, F$
begin
  for $i = 1, 2, \ldots, m$ **do**
    $x_i = f_i(u, x_1, x_2, \ldots, x_{i-1})$;
    $G_i = x_i - y_i$;
    $x_i = y_i$;
  end for
  $F = f_F(u, x_1, x_2, \ldots, x_m)$;
end

Algorithmic Trick: Preliminaries 2

Write

$$G(u,x) = \begin{pmatrix} H(u,x) - x \\ f_F(u,x) \end{pmatrix}$$

with

$$H(u,x) := \begin{pmatrix} f_1(u) \\ f_2(u,x_1) \\ \vdots \\ f_m(u,x_1,\ldots,x_{m-1}) \end{pmatrix}$$

in each lifted Newton iteration, we have to solve:

$$H(u,x) - x + \left(\frac{\partial H}{\partial x}(u,x) - \mathbb{I}_{n_x} \right) \Delta x + \frac{\partial H}{\partial u}(u,x)\Delta u = 0$$

$$f_F(u,x) + \frac{\partial f_F}{\partial x}(u,x)\Delta x + \frac{\partial f_F}{\partial u}(u,x)\Delta u = 0$$

How to solve this linear system efficiently ?

Well-Known Technique: "Condensing"

To solve $$H(u,x) - x + \left(\frac{\partial H}{\partial x}(u,x) - \mathbb{I}_{n_x} \right) \Delta x + \frac{\partial H}{\partial u}(u,x) \Delta u = 0$$

$$f_F(u,x) + \frac{\partial f_F}{\partial x}(u,x) \Delta x + \frac{\partial f_F}{\partial u}(u,x) \Delta u = 0$$

can eliminate $$\Delta x = \underbrace{-\left(\frac{\partial H}{\partial x}(u,x) - \mathbb{I}_{n_x} \right)^{-1} (H(u,x) - x)}_{=:a} + \underbrace{-\left(\frac{\partial H}{\partial x}(u,x) - \mathbb{I}_{n_x} \right)^{-1} \frac{\partial H}{\partial u}(u,x)}_{=:A} \Delta u$$

and solve "condensed" system in original degrees of freedom only: $\Delta u = -B^{-1}\, b$

and expand solution again: $\Delta x = a + A\, \Delta u$

But how to compute a, b, A, B efficiently ?

Basis of new trick: an auxiliary function

Define $Z(u,d)$

as implicit function satisfying a perturbed fixed point equation (by vector d)

$$H(u,z) - z - d = 0$$

PROPOSITION: if $d = H(u,x) - x$ then

$$\frac{\partial Z}{\partial u}(u,d) = -\left(\frac{\partial H}{\partial x}(u,x) - \mathbb{I}_{n_x}\right)^{-1} \frac{\partial H}{\partial u}(u,x)$$

and

$$\frac{\partial Z}{\partial d}(u,d) = \left(\frac{\partial H}{\partial x}(u,x) - \mathbb{I}_{n_x}\right)^{-1}$$

Partial derivatives of Z deliver part of solution of Newton system ! But how to obtain Z ?

Obtain Z by another minor code modification

- Can show that Z is obtained by one evaluation of the following function:

Algorithm 3: Modified function $Z(u, d)$

Input : $u, d_1, \ldots, d_m$
Output: $z_1, \ldots, z_m, F$
begin
  for $i = 1, 2, \ldots, m$ **do**
    $x_i = f_i(u, x_1, x_2, \ldots, x_{i-1})$;
    $z_i = x_i - d_i$;
    $x_i = z_i$;
  end for
  $F = f_F(u, x_1, x_2, \ldots, x_m)$;
end

Costs of generating and solving linear system

Using Z, can easily compute via directional derivates a, A, b, B as

$$a = -\frac{\partial Z}{\partial d}(u,d)\ d \qquad A = \frac{\partial Z}{\partial u}(u,d)$$

$$b = \quad f_F(u,x) + \frac{\partial f_F}{\partial x}(u,x)a \qquad B = \frac{\partial f_F}{\partial u}(u,x) + \frac{\partial f_F}{\partial x}(u,x)A$$

and then compute the Newton step

$$\Delta u = -B^{-1}\ b$$
$$\Delta x = a + A\ \Delta u$$

Computational effort per iteration:

- computing A & B (in one combined forward sweep)
- factoring B

Extra efforts for lifting:

- computing vector a (one extra directional derivative)
- matrix vector product $A\ \Delta u$ (could also be done w/o evaluating A)

Overview

- Nonlinear Optimization and Control Applications at OPTEC
- Idea of Lifted Newton Method
- An algorithmic trick to exploit structure "for free"
- **Application to optimization**
- Convergence analysis
- Numerical Tests

Two lifted algorithms

- Lifted Gauss-Newton Method
- Lifted SQP Method

Lifted Gauss-Newton

- Original Problem:

$$\begin{aligned} \min_{u} \quad & \frac{1}{2}\|F_1(u)\|_2^2 \\ \text{s.t.} \quad & \\ & F_2(u) \left\{ \begin{matrix} = \\ \geq \end{matrix} \right\} 0 \end{aligned}$$

- Lifted Problem:

$$\begin{aligned} \min_{u,x} \quad & \frac{1}{2}\|f_{F_1}(u,x)\|_2^2 \\ \text{s.t.} \quad & \\ & f_{F_2}(u,x) \left\{ \begin{matrix} = \\ \geq \end{matrix} \right\} 0 \\ & H(u,x) - x \quad = \quad 0 \end{aligned}$$

(obtained by lifting $F(u) := (F_1(u)^T, F_2(u)^T)^T$)

Lifted Gauss-Newton: Quadratic Subproblems

- Linearized lifted problem = structured QP subproblem:

$$
\begin{aligned}
\min_{\Delta u, \Delta x} \quad & \frac{1}{2} \left\| f_{F_1}(u_k, x_k) + \frac{\partial f_{F_1}}{\partial (u,x)}(u_k, x_k) \begin{pmatrix} \Delta u \\ \Delta x \end{pmatrix} \right\|_2^2 \\
\text{s.t.} \quad & \\
& f_{F_2}(u_k) + \frac{\partial f_{F_2}}{\partial (u,x)}(u_k, x_k) \begin{pmatrix} \Delta u \\ \Delta x \end{pmatrix} \left\{ \begin{matrix} = \\ \geq \end{matrix} \right\} 0 \\
& H(u_k, x_k) - x_k + \frac{\partial H}{\partial (u,x)}(u_k, x_k) \begin{pmatrix} \Delta u \\ \Delta x \end{pmatrix} - \Delta x \quad = \quad 0
\end{aligned}
$$

Lifted Gauss-Newton: Quadratic Subproblems

- Linearized lifted problem = structured QP subproblem:

$$\begin{aligned} \min_{\Delta u, \Delta x} \quad & \frac{1}{2} \left\| f_{F_1}(u_k, x_k) + \frac{\partial f_{F_1}}{\partial (u,x)}(u_k, x_k) \begin{pmatrix} \Delta u \\ \Delta x \end{pmatrix} \right\|_2^2 \\ \text{s.t.} \quad & \\ & f_{F_2}(u_k) + \frac{\partial f_{F_2}}{\partial (u,x)}(u_k, x_k) \begin{pmatrix} \Delta u \\ \Delta x \end{pmatrix} \left\{ \begin{matrix} = \\ \geq \end{matrix} \right\} 0 \\ H(u_k, x_k) - x_k + \frac{\partial H}{\partial (u,x)}(u_k, x_k) \begin{pmatrix} \Delta u \\ \Delta x \end{pmatrix} - \Delta x \quad & = \quad 0 \end{aligned}$$

- Condensed QP (solved by dense QP solver):

$$\begin{aligned} \min_{\Delta u} \quad & \frac{1}{2} \| b_1 + B_1 \Delta u \|_2^2 \\ \text{s.t.} \quad & \\ & b_2 + B_2 \Delta u \left\{ \begin{matrix} = \\ \geq \end{matrix} \right\} 0 \end{aligned}$$

easily get $b = (b_1^T, b_2^T)^T$ and $B = (B_1^T, B_2^T)^T$ as lifted derivatives of

$$F(u) := (F_1(u)^T, F_2(u)^T)^T$$

Lifted SQP Method: Problem Statement

- For notational simplicity, regard first only unconstrained optimization problem:

$$\min_u \ \varphi(u)$$

- Aim: find root of first order necessary conditions for optimality:

$$F(u) := \nabla_u \varphi(u) \ = 0$$

Lifted Problem

- Lifted problem formulation:

$$\begin{aligned} \min_{u,w} \quad & f_\varphi(u, w_1, w_2, \ldots, w_m) \\ \text{s.t.} \quad & \\ & g(u,w) = \begin{pmatrix} f_1(u) & - & w_1 \\ f_2(u, w_1) & - & w_2 \\ \vdots & & \\ f_m(u, w_1, \ldots, w_{m-1}) & - & w_m \end{pmatrix} = 0 \end{aligned}$$

- First order necessary conditions for optimality of lifted problem

$$\begin{aligned} \nabla_u \mathcal{L}(u,w,\lambda) &= \nabla_u f_\varphi(u,w) + \nabla_u g(u,w)\lambda &&= 0 \\ \nabla_w \mathcal{L}(u,w,\lambda) &= \nabla_w f_\varphi(u,w) + \nabla_w g(u,w)\lambda &&= 0 \\ \nabla_\lambda \mathcal{L}(u,w,\lambda) &= g(u,w) &&= 0 \end{aligned}$$

- Lifted Newton = full space SQP method (with exact Hessian) ?

Gradient evaluation via adjoint differentiation

To compute $\boxed{F(u) := \nabla_u \varphi(u)}$ perform the following code:

$$w_1 = f_1(u) \tag{3.8a}$$
$$w_2 = f_2(u, w_1) \tag{3.8b}$$
$$\vdots$$
$$w_m = f_m(u, w_1, \ldots, w_n - 1) \tag{3.8c}$$

function value →

$$y \equiv f_\varphi(u, w_1, \ldots, w_m) \tag{3.8d}$$
$$\bar{w}_m = \nabla_{w_m} f_\varphi \tag{3.8e}$$
$$\bar{w}_{m-1} = \nabla_{w_{m-1}} f_\varphi + \nabla_{w_{m-1}} f_m \bar{w}_m \tag{3.8f}$$
$$\vdots$$
$$\bar{w}_1 = \nabla_{w_1} f_\varphi + \sum_{i=2}^{m} \nabla_{w_1} f_i \bar{w}_i \tag{3.8g}$$

gradient →

$$\bar{u} = \nabla_u f_\varphi + \sum_{i=1}^{m} \nabla_u f_i \bar{w}_i. \tag{3.8h}$$

Gradient evaluation via adjoint differentiation

To compute $F(u) := \nabla_u \varphi(u)$ perform the following code:

function value

$$w_1 = f_1(u) \tag{3.8a}$$
$$w_2 = f_2(u, w_1) \tag{3.8b}$$
$$\vdots$$
$$w_m = f_m(u, w_1, \ldots, w_n - 1) \tag{3.8c}$$
$$y \equiv f_\varphi(u, w_1, \ldots, w_m) \tag{3.8d}$$
$$\bar{w}_m = \nabla_{w_m} f_\varphi \tag{3.8e}$$
$$\bar{w}_{m-1} = \nabla_{w_{m-1}} f_\varphi + \nabla_{w_{m-1}} f_m \bar{w}_m \tag{3.8f}$$
$$\vdots$$
$$\bar{w}_1 = \nabla_{w_1} f_\varphi + \sum_{i=2}^{m} \nabla_{w_1} f_i \bar{w}_i \tag{3.8g}$$

gradient

$$\bar{u} = \nabla_u f_\varphi + \sum_{i=1}^{m} \nabla_u f_i \bar{w}_i. \tag{3.8h}$$

Lifted Newton = Full Space SQP

THEOREM: If we lift $F(u) := \nabla_u \varphi(u)$ with respect to all intermediate variables: $x = (w_1, \ldots, w_m, \bar{w}_m, \ldots, \bar{w}_1)$

then full space SQP and lifted Newton iterations are identical, with $\lambda \equiv \bar{w}$

COROLLARY: Same equivalence holds for lifting of constrained problems:

$$\begin{aligned} &\min_u \varphi(u) \\ &\text{subject to} \\ &\qquad h(u) = 0, \end{aligned} \quad \longrightarrow \quad \begin{aligned} &\min_{u,w} f_\varphi(u,w) \\ &\text{subject to} \\ &\qquad g(u,w) = 0, \\ &\qquad f_h(u,w) = 0, \end{aligned}$$

if we lift Lagrange gradient and constraint:

$$\begin{pmatrix} \nabla_u \mathcal{L}^{\text{orig}} \\ h \end{pmatrix}$$

(need full derivatives only w.r.t. u, not μ, due to symmetry of KKT systems)

Overview

- Nonlinear Optimization and Control Applications at OPTEC
- Idea of Lifted Newton Method
- An algorithmic trick to exploit structure "for free"
- Application to optimization
- **Convergence analysis**
- Numerical Tests

Why does lifting often improve convergence speed ?

Simple model problem: chain of scalar functions

- Regard a sequence of scalar functions:

$$x_1 = f_1(u),\ x_2 = f_2(x_1), \ldots, x_m = f_m(x_{m-1}), \text{and } f_F(x_m) \equiv f_{m+1}(x_m)$$

- after affine transformations, can assume that solution is zero and

$$f_i(x) = x + b_i(x)^2 + O(|x|^3)$$

- under these circumstances, the non-lifted function is:

$$F(u) = f_{m+1}(f_m(\ldots f_1(u)\ldots)) = u + \Big(\sum_{i=1}^{m+1} b_i\Big)u^2 + O(|u|^3)$$

Convergence speed of Non-Lifted Newton

- Derivative is given by

$$F'(u) = 1 + 2\Big(\sum_{i=1}^{m+1} b_i\Big)u + O(|u|^2)$$

- It is easy to show that non-lifted Newton iterations contract like:

$$u^{[k+1]} = \Big(\sum_{i=1}^{m+1} b_i\Big)(u^{[k]})^2 + O(|u^{[k]}|^3)$$

i.e. quadratic convergence with contraction constant $\left(\sum_{i=1}^{m+1} b_i\right)$

Lifted Newton Convergence

- Lifted residual is

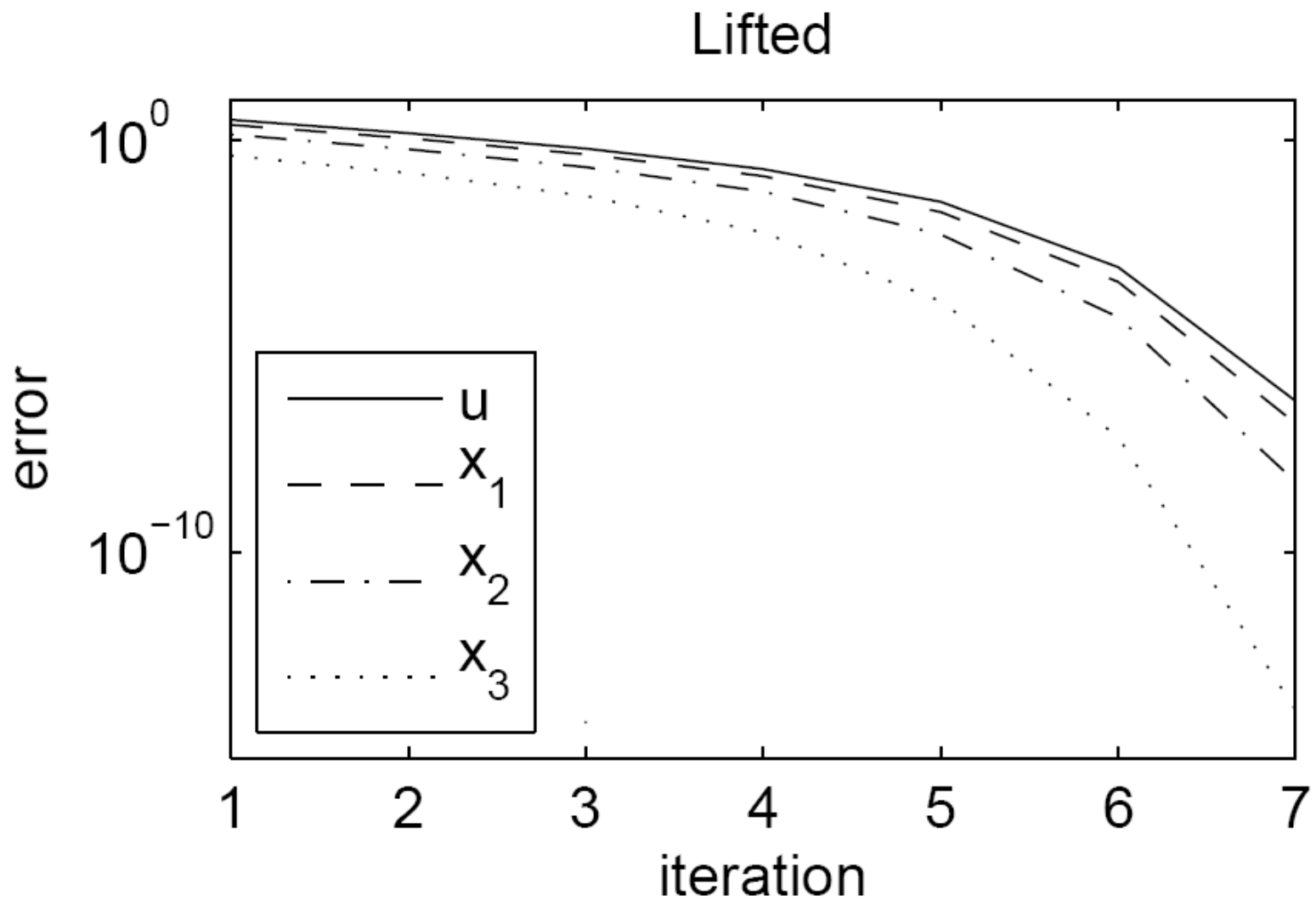
$$G(u,x) = \begin{pmatrix} u + b_1 u^2 & - & x_1 \\ x_1 + b_2 x_1^2 & - & x_2 \\ \vdots & & \\ x_{m-1} + b_m x_{m-1}^2 & - & x_m \\ x_m + b_{m+1} x_m^2 & & \end{pmatrix} + O\left(\left\| \begin{pmatrix} u \\ x \end{pmatrix} \right\|^3 \right)$$

THEOREM: Lifted Newton iterations contract in a "staggered" way:

$$\begin{pmatrix} u \\ x_1 \\ \vdots \\ x_{m-1} \\ x_m \end{pmatrix}^{[k+1]} = \begin{pmatrix} b_1 \left(u^{[k]}\right)^2 + \sum_{i=2}^{m+1} b_i \left(x_{i-1}^{[k]}\right)^2 \\ \sum_{i=2}^{m+1} b_i \left(x_{i-1}^{[k]}\right)^2 \\ \vdots \\ b_m \left(x_{m-1}^{[k]}\right)^2 + b_{m+1} \left(x_m^{[k]}\right)^2 \\ b_{m+1} \left(x_m^{[k]}\right)^2 \end{pmatrix} + O\left(\left\| \begin{pmatrix} u \\ x \end{pmatrix}^{[k]} \right\|^3 \right)$$

(if all b_i have the same sign, last variable is "leader" and contracts fastest)

Convergence for motivating toy example



Practical Conclusions from Theorem

Two cases:

- Same curvature → lifted Newton better.
  E.g. in toy example, or dynamic simulation codes where several times
  - same time stepping function is called
  - at similar values

- Opposite curvature → unlifted Newton better.
  Example: $b_1 = 1$ and $b_2 = -1$, e.g. $F(u) = u$ decomposed as

$$f_1(u) = \tfrac{1}{2}(1+u)^2 - \tfrac{1}{2} \text{ and } f_2(x) = \sqrt{1+2x} - 1$$

  After first iteration, unlifted Newton is already converged, lifted not!

Toy Optimal Control Problem for Illustration

$$\min_{x(\cdot),u(\cdot)} \int_0^3 |x(t)|^2 + |u(t)|^2 dt$$

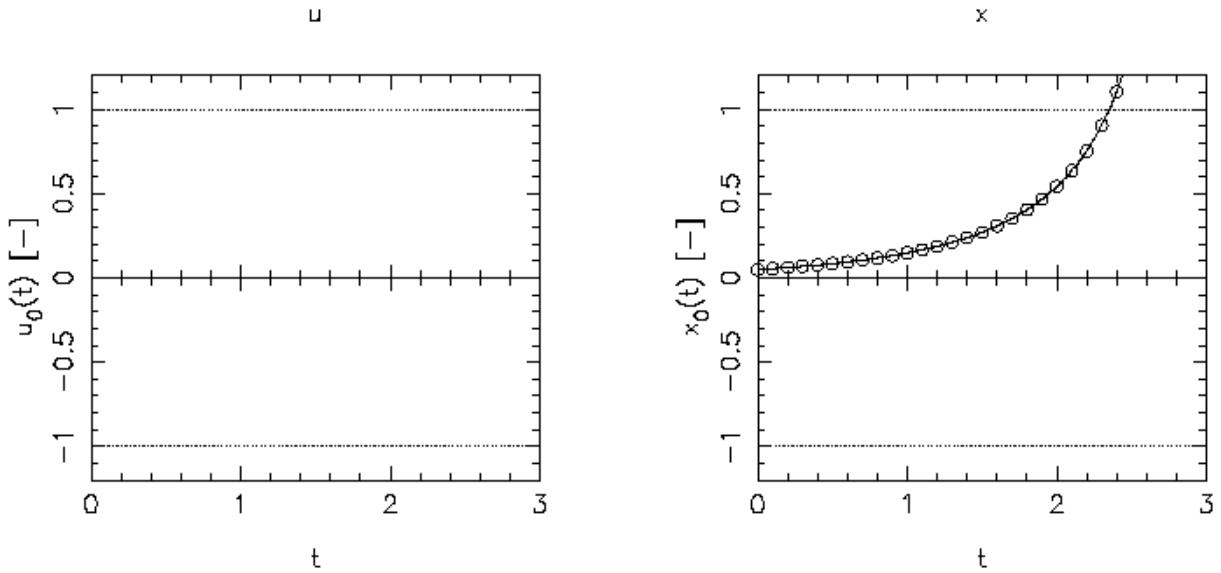
s.t.

$$\begin{aligned} \dot{x}(t) &= x(t)\,(x(t)+1) + u(t) \\ x(0) &= x_0 \\ x(3) &= 0 \\ |x(t)| &\leq 1 \\ |u(t)| &\leq 1. \qquad t \in [0,3] \end{aligned}$$

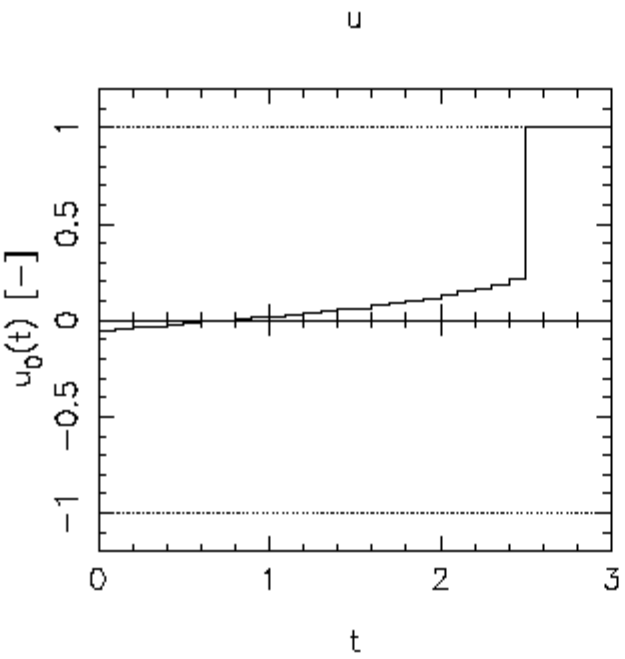
Mildly nonlinear and unstable system.
Use ODE solver with sensitivities (e.g. in ACADO or MUSCOD-II).
Compare unlifted and lifted Gauss-Newton method.

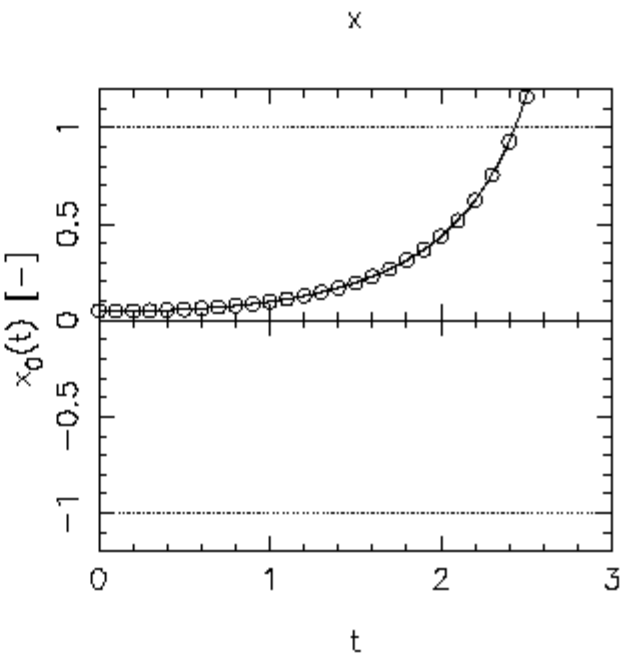
Unlifted Gauss-Newton

- Choose $N = 30$ equal control intervals.
- Initialize with steady state controls $u(t) \equiv 0$.

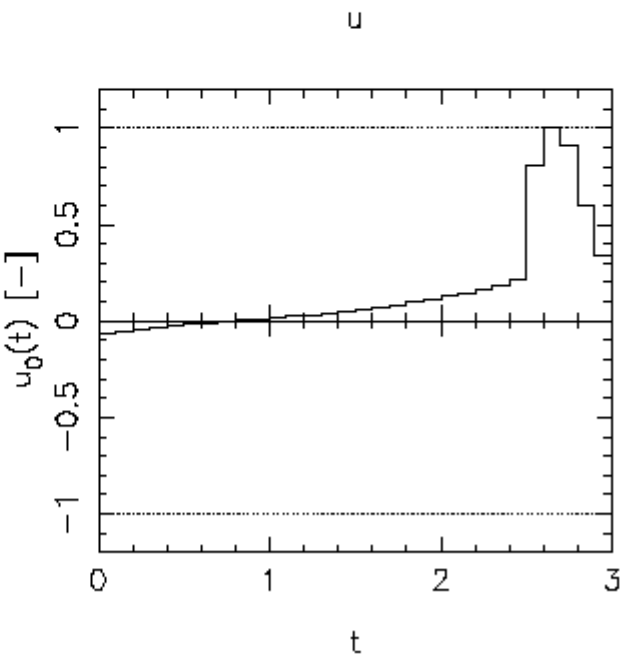


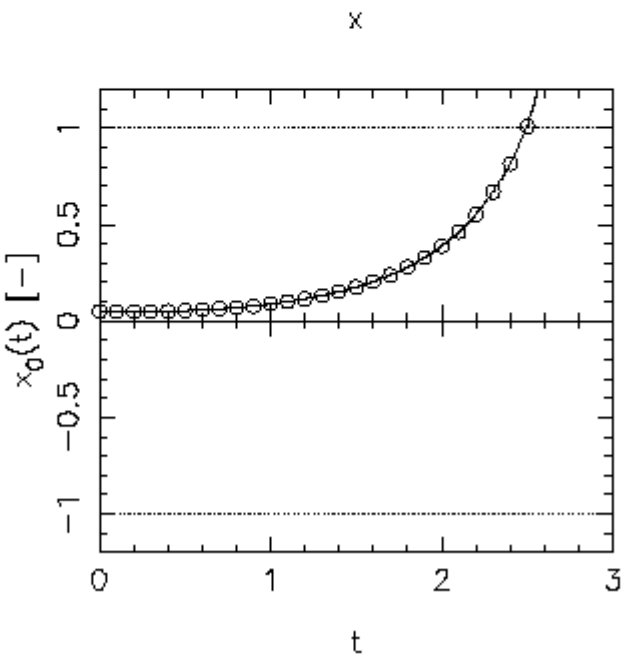
Unlifted: First Iteration



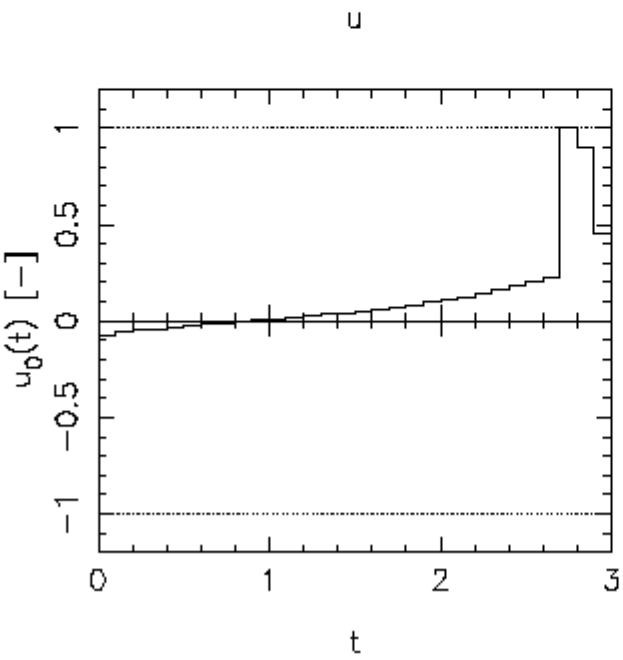


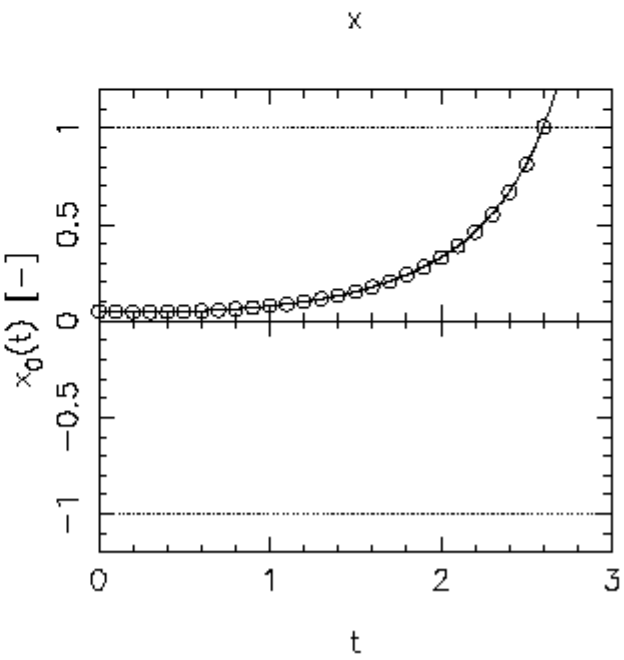
Unlifted: Second Iteration



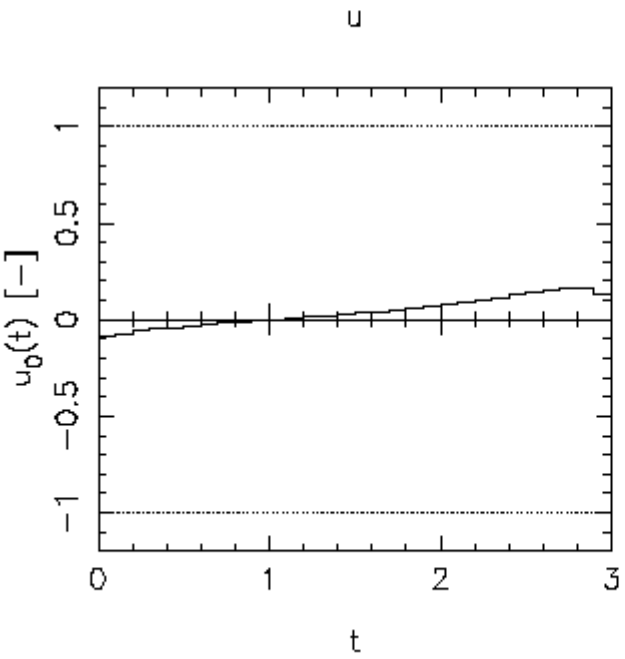


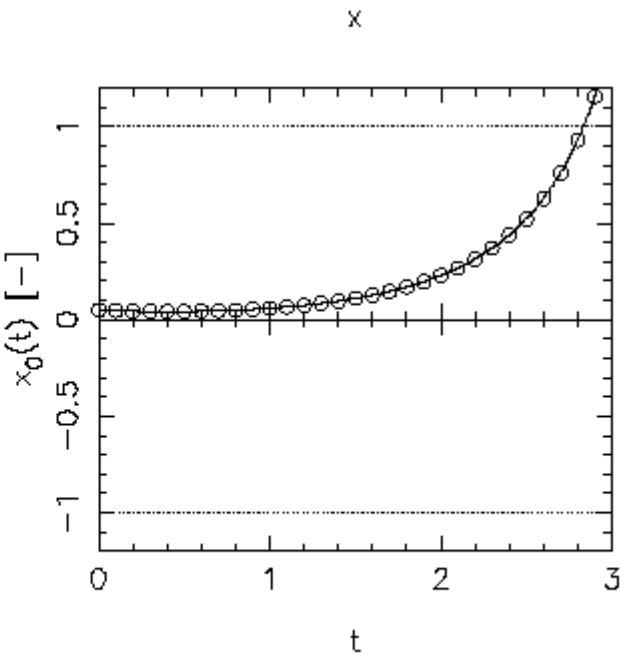
Unlifted: Third Iteration



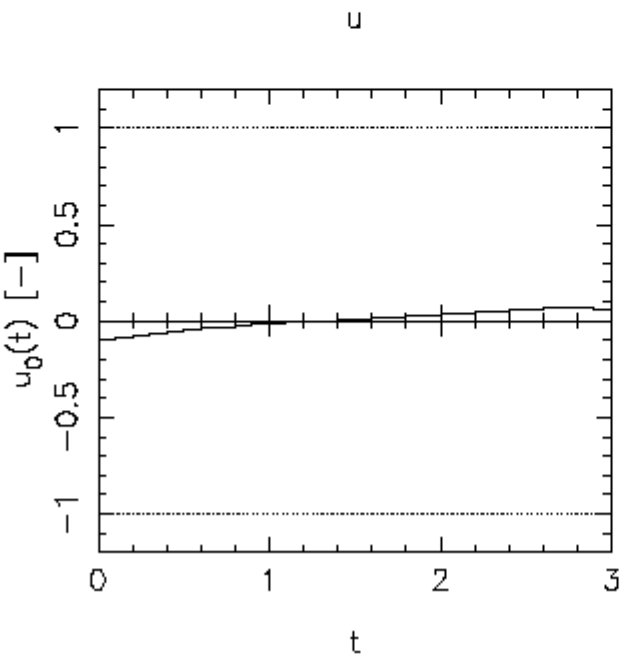


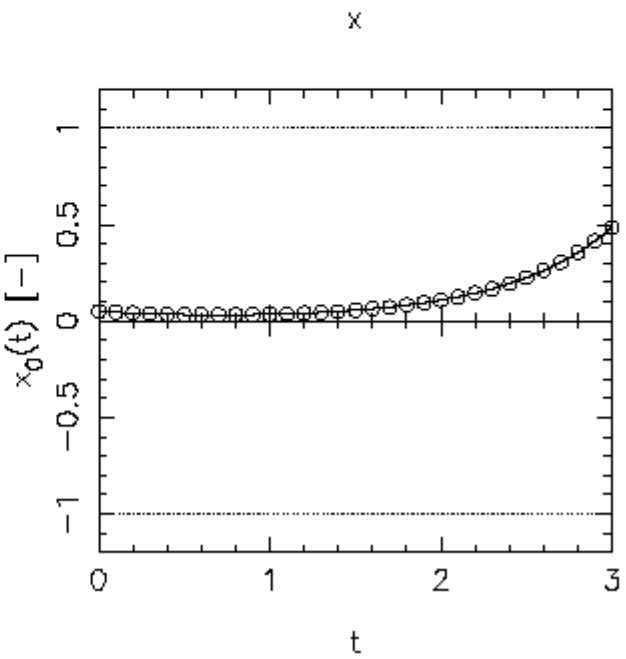
Unlifted: 4th Iteration



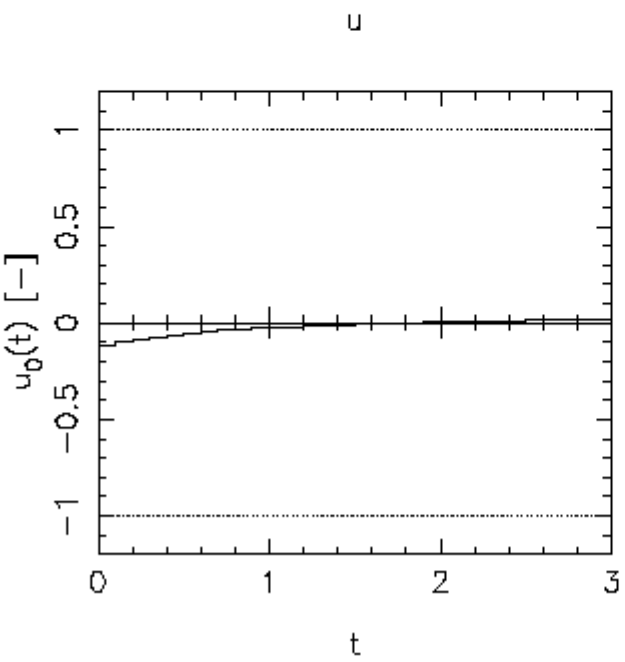


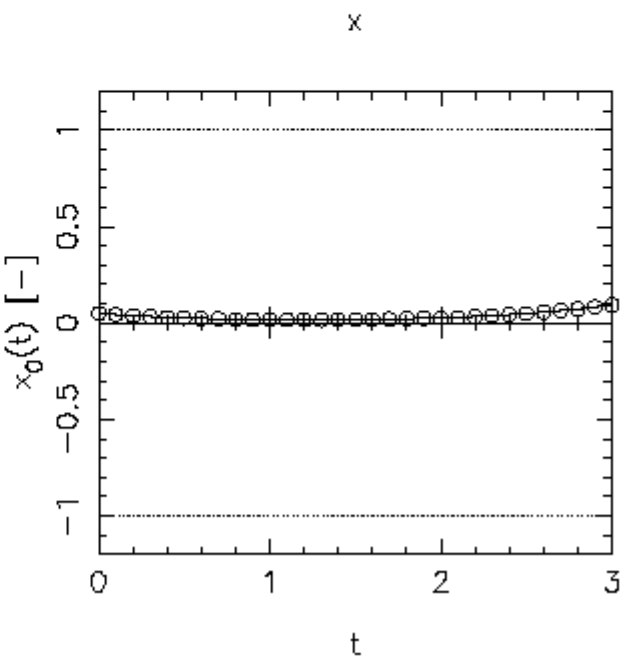
Unlifted: 5th Iteration



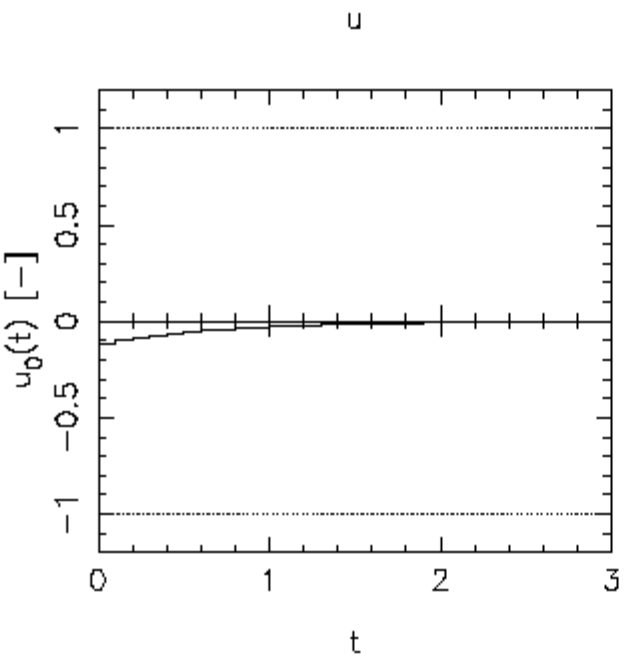


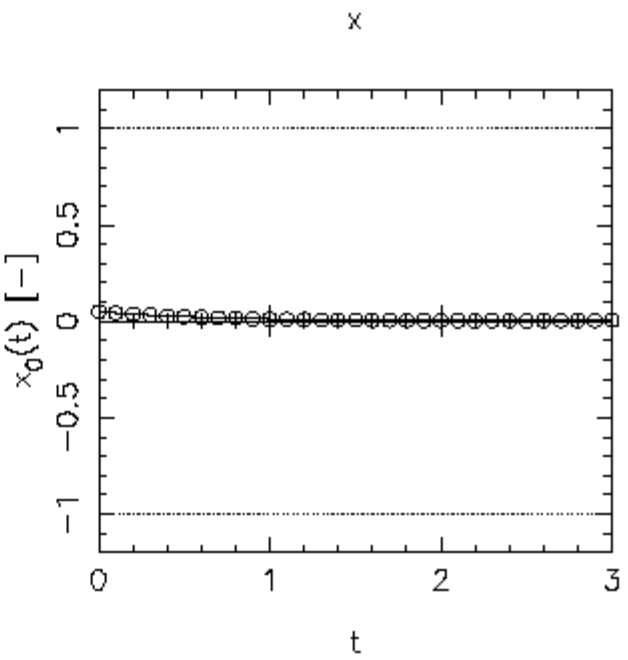
Unlifted: 6th Iteration



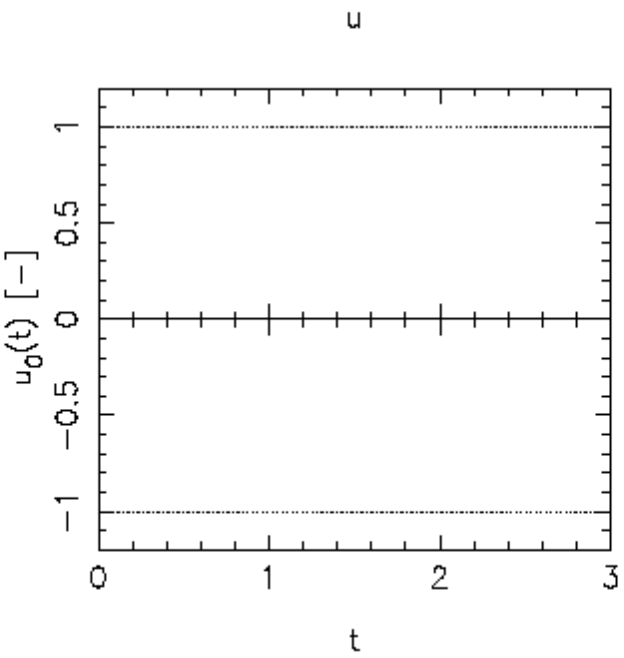


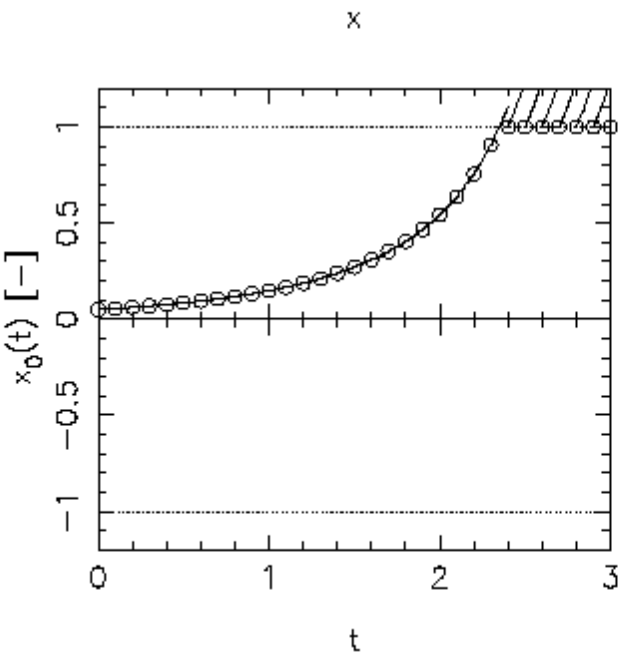
Unlifted: 7th Iteration (Solution)



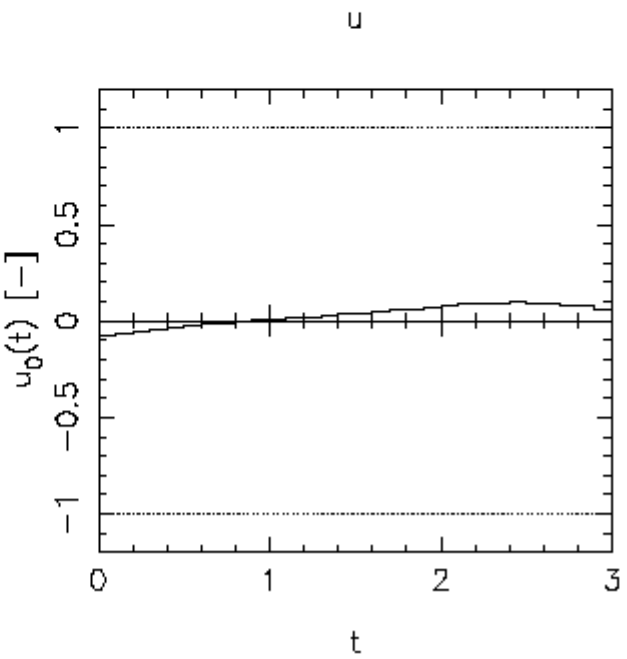


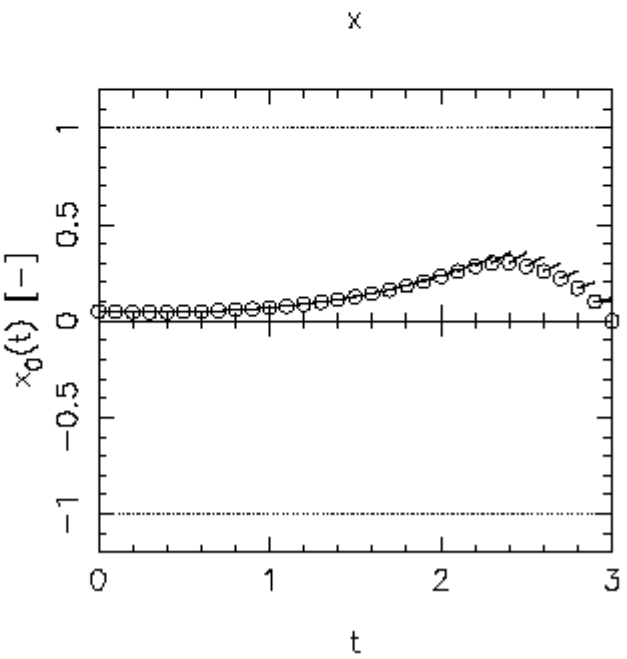
Toy Example: Lifted Initialization



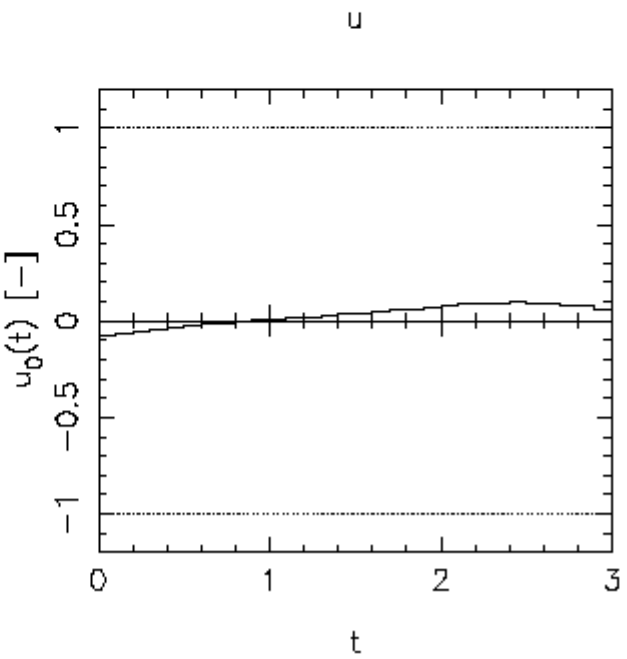


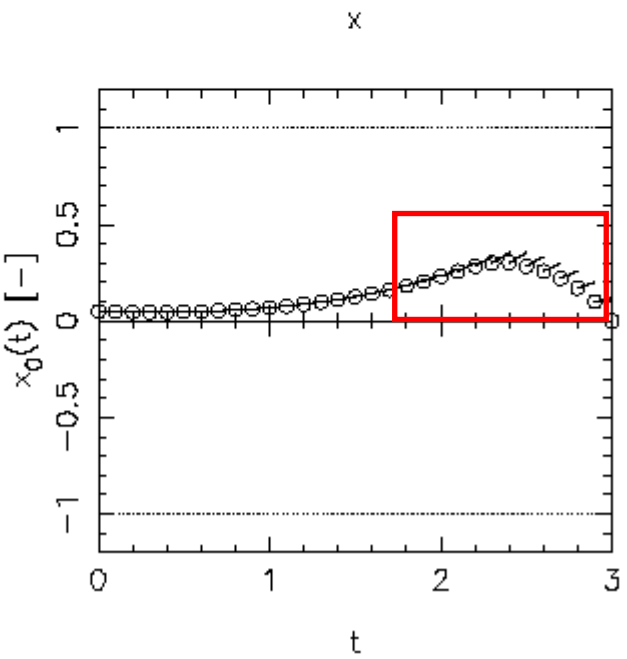
Lifted: First Iteration



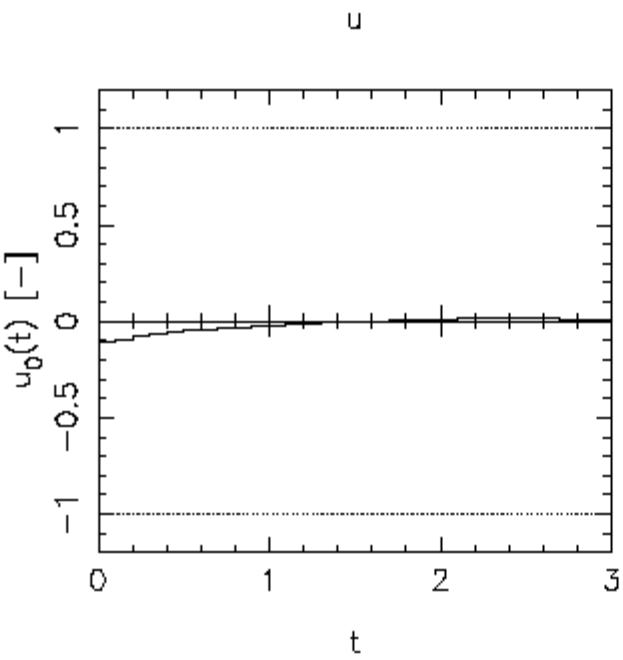


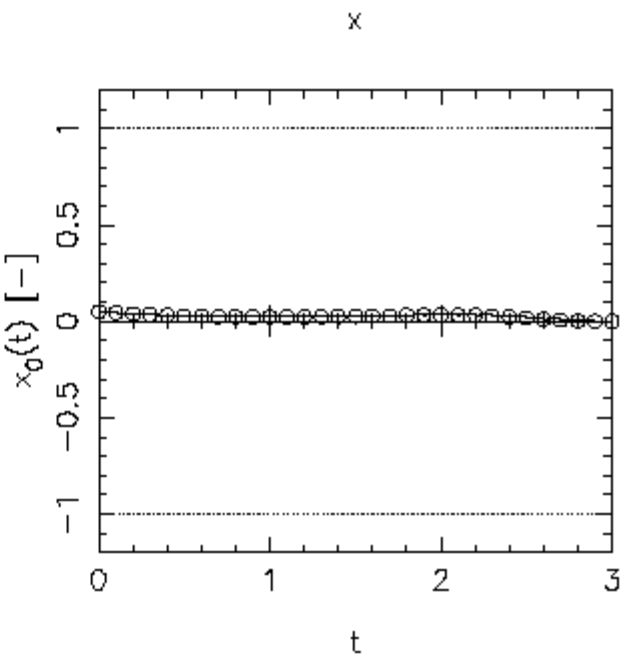
Lifted: First Iteration



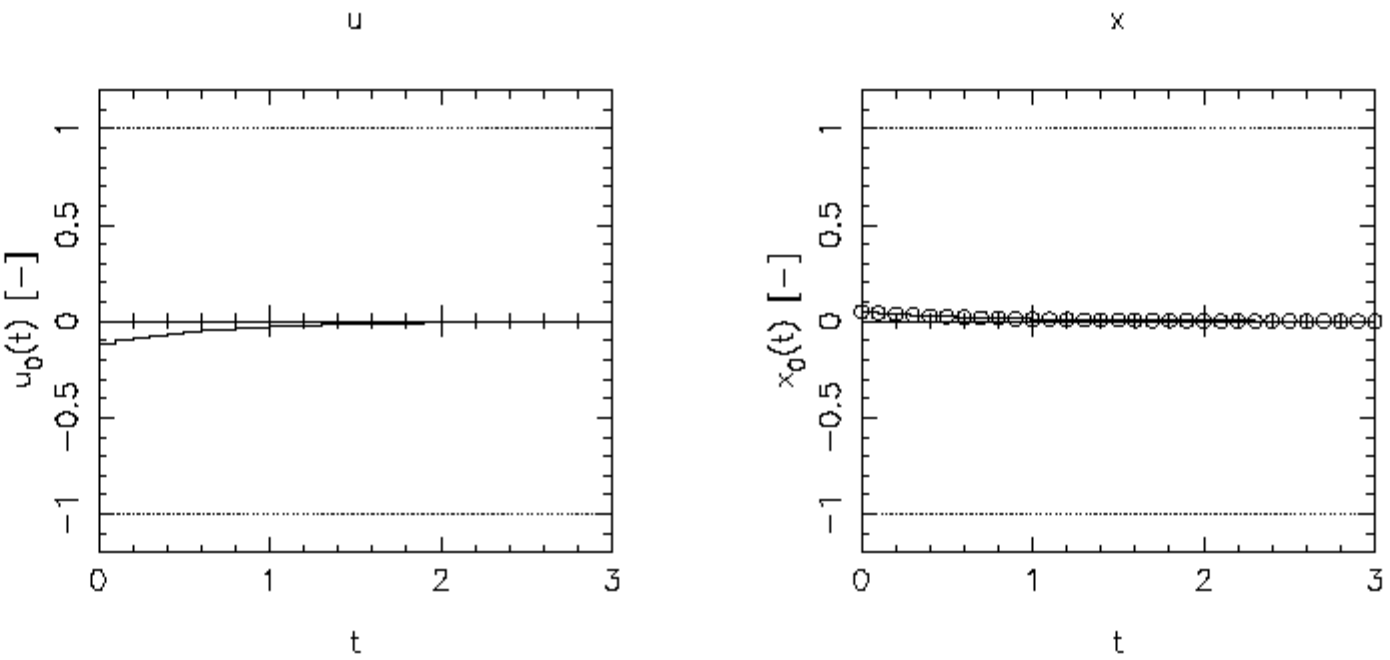


Lifted: Second Iteration





Lifted: 3rd Iteration (already solution!)



Large scale Gauss-Newton example 1

- Regard 2-D heat equation with 3 unknown diffusion parameters c

$$\Omega = (0,1) \times (0,1)$$

$$\begin{aligned}
\partial_t u(t,x_1,x_2) - d(x_1,x_2)\Delta u(t,x_1,x_2) &= 0, \qquad (x_1,x_2) \in \Omega, t \in [0,1] \\
d(x_1,x_2) &= c_0 + c_1 x_1 + c_2 x_2 \\
u(t,x_1,x_2) &= 100, \qquad (x_1,x_2) \in \{0\} \times [0,1], t \in [0,1] \\
u(t,x_1,x_2) &= 0, \qquad (x_1,x_2) \in \partial\Omega \setminus (\{0\} \times [0,1]), t \in [0,1]
\end{aligned}$$

- Discretize on 9 x 9 grid and perform 200 timesteps
  (= 16200 hidden variables)

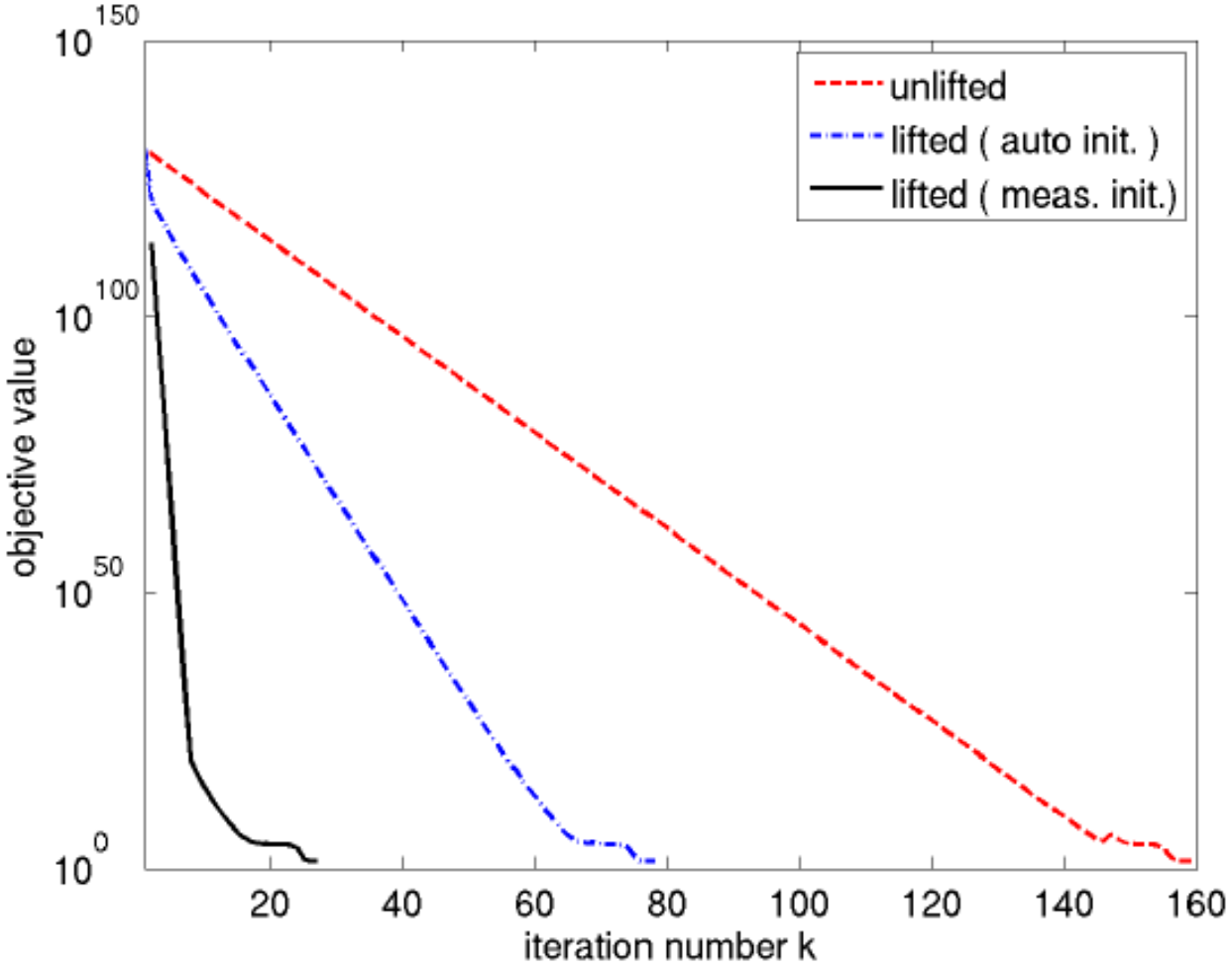
- Measure state u every 10 time steps (= 1620 measurements)

- Aim: estimate unknown parameters

Compare three Gauss-Newton variants

- Non-lifted Newton with 3 variables only
- Lifted Newton (with 1620 lifted node variables), initialized like non-lifted variant via a forward simulation
- Lifted Newton, but all nodes initialized with measurements

Convergence for PDE parameter estimation example 1



Large Scale Example 2

- Regard 2-D shallow water wave equation (with unknown parameters)

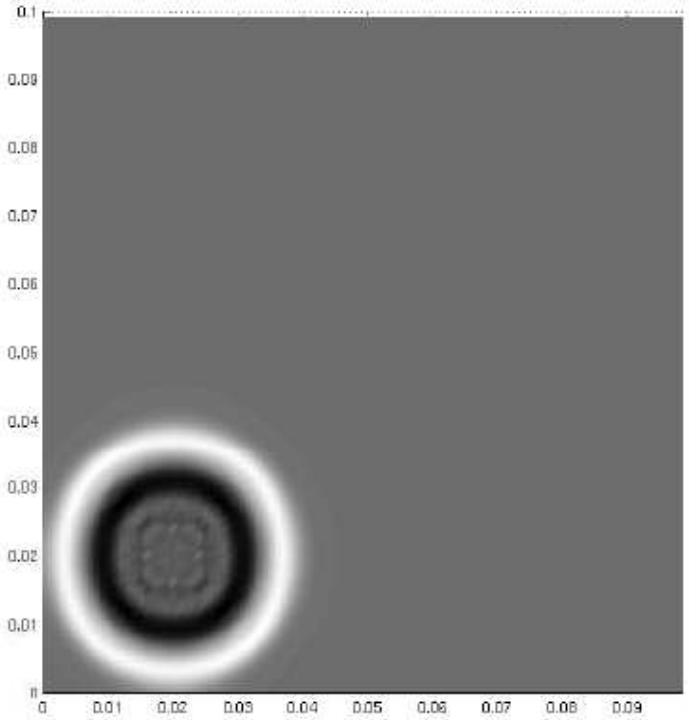
$$\begin{aligned}
\partial_t u(t,x,y) &= -g\partial_x h(t,x,y) - bu(t,x,y) \\
\partial_t v(t,x,y) &= -g\partial_y h(t,x,y) - bv(t,x,y) \\
\partial_t h(t,x,y) &= -H\left[\partial_x u(t,x,y) + \partial_y v(t,x,y)\right]
\end{aligned}$$

- Discretize all 3 states u, v, h on 30 x 30 grid and perform 10000 timesteps (= 27 million variables !)

- Measure **only height** h every 100 time steps (= 90 000 measurements)

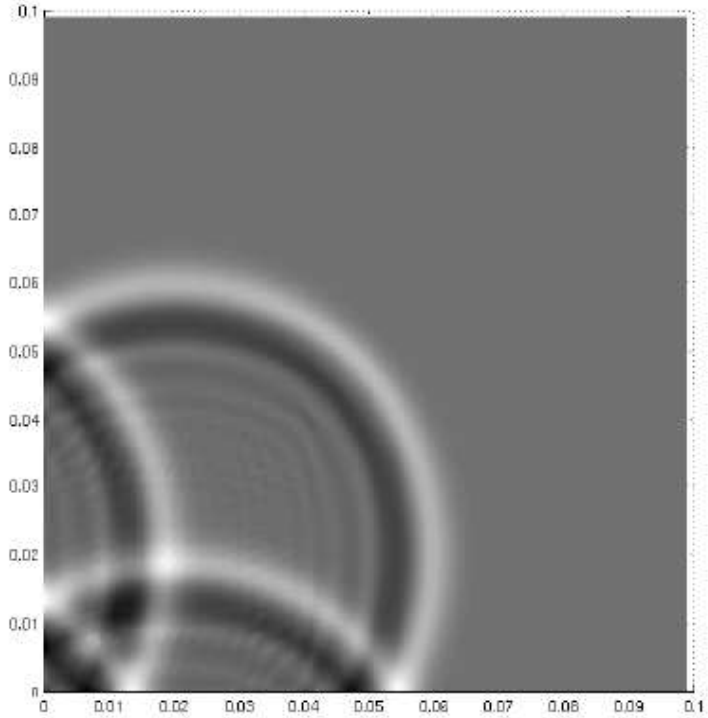
- Aim: estimate 2 unknown parameters, water depth H and viscous friction coefficient b

Large Scale Example 2: Measurements



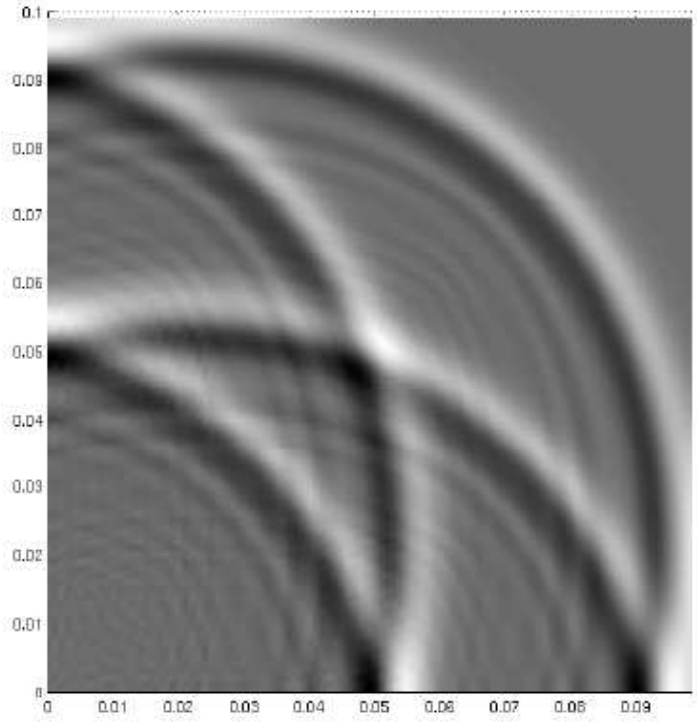
(a)

Large Scale Example 2: Measurements



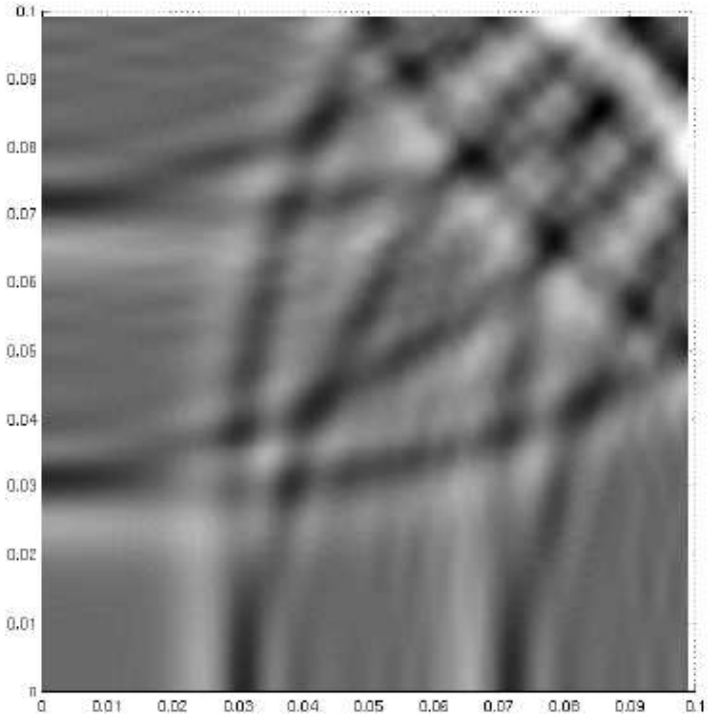
(b)

Large Scale Example 2: Measurements



(c)

Large Scale Example 2: Measurements



(d)

Compare three Gauss-Newton variants for example 2

- Non-lifted Newton with 2 variables only
- Lifted Newton (with 90 000 lifted variables), initialized like non-lifted variant via a forward simulation
- Lifted Newton, but initialize height with measurements

Performance of 3 variants for different initial guesses

b	H	# iterations unlifted	#iterations lifted (autom. init.)	#iterations lifted (meas init.)
0.5	0.01	5	5	4
5	0.01	6	5	4
15	0.01	17	7	6
30	0.01	27	7	6
2	0.005	31	9	5
2	0.02	38	12	5
2	0.1	44	13	8
0.2	0.001	33	12	7
1	0.005	47	10	5
4	0.02	56	10	5
1	0.02	44	9	6
20	0.001	24	10	6

true values $b = 2$ and $H = 0.01$

CPU time per iteration: 8.9 s for unlifted, 11.8 s for lifted (~1 min total)

(Note: formulation e.g. in AMPL would involve 27 million variables)

Summary

- Lifting offers advantages for Newton type optimization:
  - faster local convergence rate (→proven in simplified setting)
  - more freedom for initialization

- Structure exploiting "Lifted Newton Methods" can easily be generated for any given user functions and any Newton type method:
  - only minor code additions
  - nearly no additional costs per iteration
  - compatible with SQP, BFGS, Gauss-Newton, …
  - compatible with any linear solver for condensed systems e.g. GMRES, cf. [Shimizu, Ohtsuka & D. 2009]

- Code and examples available in C++ package LiftOpt [Albersmeyer&D. 2010]

References

- **M.R. Osborne, On shooting methods for boundary value problems, Journal of Mathematical Analysis and Applications, 27 (1969), pp. 417–433.**
- **H.G. Bock, Randwertproblemmethoden zur Parameteridentifizierung in Systemen nichtlinearer Differentialgleichungen, vol. 183 of Bonner Mathematische Schriften, University of Bonn, Bonn, 1987.**
- J.P. Schloeder, Numerische Methoden zur Behandlung hochdimensionaler Aufgaben der Parameteridentifizierung, vol. 187 of Bonner Mathematische Schriften, University of Bonn, Bonn, 1988.
- P. E. Gill, L. O. Jay, M. W. Leonard, L. R. Petzold, and V. Sharma. An SQP method for the optimal control of large-scale dynamical systems. Journal Comp. Appl. Math.,120:197–213, 2000.
- A.A.S. Schaefer, Efficient reduced Newton-type methods for solution of large-scale structured optimization problems with application to biological and chemical processes, PhD thesis, University of Heidelberg, 2005.
- Y. Shimizu, T. Ohtsuka & M.D, A real-time algorithm for nonlinear receding horizon control using multiple shooting and continuation/Krylov method. *Int. J. Robust Nonlinear Control (*2009) **19**:919–936
- **J. Albersmeyer & M. D., The Lifted Newton Method and Its Application in Optimization. *SIAM J. Opt.* (2010) 20:3,1655-1684**

Open Positions at Optimization in Engineering Center OPTEC, Leuven

- OPTEC successfully secured funding until 2017
- Two open post doc positions in MD's group:
  - **Embedded Optimization for Control (3 years)**
  - **Distributed Optimization (2 years)**
- One open PhD position:
  - **Modelling and Optimal Control of Kite Energy Systems (4 years)**