The Newton Step Method for Algorithmic Differentiation with Implicit Functions

Bradley M. Bell

Applied Physics Laboratory,
Health Metrics and Evaluation,
University of Washington,
bradbell@uw.edu

Kasper Kristensen

Department of Appied Mathematics
and Computer Science,
Technical University of Denmark,
kaskr@imm.dtu.dk

West Coast Optimization Seminar
May 14, 2016

Contents

Contents

Contents

Contents

Contents

Contents

Introduction

Problem

There are many circumstances where variables are defined implicitly and we need to calculate derivatives of functions that depend on these variables.

Problem

There are many circumstances where variables are defined implicitly and we need to calculate derivatives of functions that depend on these variables.

- The implicit function approach to equality constrained optimization; e.g., applied to PDE constrained parameter estimation.

Problem

There are many circumstances where variables are defined implicitly and we need to calculate derivatives of functions that depend on these variables.

- The implicit function approach to equality constrained optimization; e.g., applied to PDE constrained parameter estimation.
- Nonlinear mixed effects models where the optimal random effects are an implicit function of the fixed effects and the fixed effects objective depends on these random effects.

Problem

There are many circumstances where variables are defined implicitly and we need to calculate derivatives of functions that depend on these variables.

- The implicit function approach to equality constrained optimization; e.g., applied to PDE constrained parameter estimation.
- Nonlinear mixed effects models where the optimal random effects are an implicit function of the fixed effects and the fixed effects objective depends on these random effects.
- More generally consider bilevel programming where the current point is such that the implicit function theorem applies to inner variables, and corresponding Lagrange multipliers, as a function of the outer variables.

Approach

If the implicitly dependent variables are defined by nonlinear equations, they are usually solved by an iterative procedure.

Approach

If the implicitly dependent variables are defined by nonlinear equations, they are usually solved by an iterative procedure.

- The naive approach it to apply AD to the iterative procedure.

Approach

If the implicitly dependent variables are defined by nonlinear equations, they are usually solved by an iterative procedure.

- The naive approach it to apply AD to the iterative procedure.
- An AD method that computes these derivatives for any order, using the implicit function instead of the iterative procedure, can be found in [WWS10, Section 4.1].

Approach

If the implicitly dependent variables are defined by nonlinear equations, they are usually solved by an iterative procedure.

- The naive approach it to apply AD to the iterative procedure.
- An AD method that computes these derivatives for any order, using the implicit function instead of the iterative procedure, can be found in [WWS10, Section 4.1].
- We consider the Newton Step Method for computing derivatives of functions that are expressed in terms of the implicitly dependent variables.

Approach

If the implicitly dependent variables are defined by nonlinear equations, they are usually solved by an iterative procedure.

- The naive approach it to apply AD to the iterative procedure.
- An AD method that computes these derivatives for any order, using the implicit function instead of the iterative procedure, can be found in [WWS10, Section 4.1].
- We consider the Newton Step Method for computing derivatives of functions that are expressed in terms of the implicitly dependent variables.
- This enables one to easily use forward or reverse mode and sparsity when calculating derivatives of functions that depend on implicitly defined variables.

Newton Step Theorem

Notation

$Y(x)$

We are given a vector valued function $L : \mathbf{R}^n \times \mathbf{R}^m \to \mathbf{R}^m$, and define the implicit function $Y(x)$ by

$$L(x, Y(x)) = 0 \ .$$

Notation

$Y(x)$

We are given a vector valued function $L : \mathbf{R}^n \times \mathbf{R}^m \to \mathbf{R}^m$, and define the implicit function $Y(x)$ by

$$L(x, Y(x)) = 0 \ .$$

$L_y(x, y)$

Define the matrix valued function $L_y : \mathbf{R}^n \times \mathbf{R}^m \to \mathbf{R}^{m \times m}$ by

$$L_y(x, y)_{[i,j]} = \frac{\partial L(x, y)_{[i]}}{\partial y_{[j]}}; .$$

Notation

$|p|$, $\partial^p f(x)$

Given a matrix valued function $f : \mathbf{R}^\ell \to \mathbf{R}^{n\times m}$, and a multi-index $p \in \mathbf{Z}_+^\ell$, we use the notation $|p| = p_{[1]} + \ldots + p_{[\ell]}$ and $\partial^p f : \mathbf{R}^\ell \to \mathbf{R}^{n\times m}$ is defined by

$$\partial^p f(x) = \frac{\partial^{p[1]}}{\partial x_{[1]}^{p[1]}} \cdots \frac{\partial^{p[\ell]}}{\partial x_{[\ell]}^{p[\ell]}} f(x) \ .$$

Notation

$|p|$, $\partial^p f(x)$

Given a matrix valued function $f : \mathbf{R}^\ell \to \mathbf{R}^{n\times m}$, and a multi-index $p \in \mathbf{Z}_+^\ell$, we use the notation $|p| = p_{[1]} + \ldots + p_{[\ell]}$ and $\partial^p f : \mathbf{R}^\ell \to \mathbf{R}^{n\times m}$ is defined by

$$\partial^p f(x) = \frac{\partial^{p[1]}}{\partial x_{[1]}^{p[1]}} \cdots \frac{\partial^{p[\ell]}}{\partial x_{[\ell]}^{p[\ell]}} f(x) \, .$$

N(Z)(x)

Given an arbitrary function $Z : \mathbf{R}^n \to \mathbf{R}^m$, the corresponding Newton step $N(Z) : \mathbf{R}^n \to \mathbf{R}^m$ is defined by

$$N(Z)(x) = Z(x) - L_y(x, Z(x))^{-1} L(x, Z(x)) \, .$$

Theorem

Notation

Fix $\bar{x} \in \mathbf{R}^n$, define $N_0(x)$ to be the constant function $N_0(x) = Y(\bar{x})$. For $k = 1, \ldots,$ define $N_k : \mathbf{R}^n \to \mathbf{R}^m$ by $N_k(x) = N(N_{k-1})(x)$.

Theorem

Notation

Fix $\bar{x} \in \mathbf{R}^n$, define $N_0(x)$ to be the constant function $N_0(x) = Y(\bar{x})$. For $k = 1, \ldots,$ define $N_k : \mathbf{R}^n \to \mathbf{R}^m$ by $N_k(x) = N(N_{k-1})(x)$.

Conclusion

It follows that for any multi-index $p \in \mathbf{Z}_+^n$, and for any $k \geq |p|$,

$$\partial^p Y(\bar{x}) = \partial^p N_k(\bar{x}) \ .$$

Remarks

One Newton Step

Well Known

This theorem is well know for the case $k = 1$; e.g., [Gil92, eq. 15].

One Newton Step

Well Known

This theorem is well know for the case $k = 1$; e.g., [Gil92, eq. 15].

Standard $N_1(x)$

The reference above and the theorem use the following definition

$$N_1(x) = Y(\bar{x}) - L_y(x, Y(\bar{x}))^{-1} L(x, Y(\bar{x})) \; .$$

This requires differentiating the inversion when computing $N_1^{(1)}(x)$.

One Newton Step

Well Known

This theorem is well know for the case $k = 1$; e.g., [Gil92, eq. 15].

Standard $N_1(x)$

The reference above and the theorem use the following definition

$$N_1(x) = Y(\bar{x}) - L_y(x, Y(\bar{x}))^{-1} L(x, Y(\bar{x})) \ .$$

This requires differentiating the inversion when computing $N_1^{(1)}(x)$.

Alternate $N_1(x)$

The theorem also holds with the alternate definition

$$N_1(x) = Y(\bar{x}) - L_y(\bar{x}, Y(\bar{x}))^{-1} L(x, Y(\bar{x})) \ .$$

This avioids differentiating the inversion when computing $N_1^{(1)}(x)$.

Checkpointing

In General

Often an procedure can be divided into steps where a reduced set of variables are input to each step. Checkpointing [AW00] is a technique for reducing the memory required by AD in this case.

Checkpointing

In General

Often an procedure can be divided into steps where a reduced set of variables are input to each step. Checkpointing [AW00] is a technique for reducing the memory required by AD in this case.

One Newton Step

When computing $N_k(x)$ for $k > 1$, we can divide the computaiton into Newton steps

$$(x, y) \rightarrow \left(x \; , \; y - L_y(x, y)^{-1} L(x, y) \right) \; .$$

We only need to record one such step when computing derivatives of $N_k(x)$ for any k.

Checkpointing

In General

Often an procedure can be divided into steps where a reduced set of variables are input to each step. Checkpointing [AW00] is a technique for reducing the memory required by AD in this case.

One Newton Step

When computing $N_k(x)$ for $k > 1$, we can divide the computaiton into Newton steps

$$(x, y) \rightarrow \left(x \; , \; y - L_y(x, y)^{-1} L(x, y) \right) \; .$$

We only need to record one such step when computing derivatives of $N_k(x)$ for any k.

Example Package

The `cppad_mixed` packages takes advantage of this technique [Bel16].

Nonlinear Mixed Effects Models

Backgound

Model

We use x to denote the fixed effects, y to denote the random effects, and z to denote the data in a non-lienar mixed effects model. We are given a representation for the densities $\mathbf{p}(z|x, y)$, $\mathbf{p}(y|x)$, and $\mathbf{p}(x)$.

Backgound

Model

We use x to denote the fixed effects, y to denote the random effects, and z to denote the data in a non-lienar mixed effects model. We are given a representation for the densities $\mathbf{p}(z|x, y)$, $\mathbf{p}(y|x)$, and $\mathbf{p}(x)$.

Problem

$$\text{minimize } - \int_y \mathbf{p}(z|x, y) \, \mathbf{p}(y|x) \, \mathbf{p}(x) \, \mathbf{d}y \text{ w.r.t } x$$

Often the dimension of y is large and the Hessian w.r.t y is sparse.

Backgound

Model

We use x to denote the fixed effects, y to denote the random effects, and z to denote the data in a non-lienar mixed effects model. We are given a representation for the densities $\mathbf{p}(z|x, y)$, $\mathbf{p}(y|x)$, and $\mathbf{p}(x)$.

Problem

$$\mathrm{minimize} \; - \int_y \mathbf{p}(z|x, y) \, \mathbf{p}(y|x) \, \mathbf{p}(x) \, \mathbf{d}y \text{ w.r.t } x$$

Often the dimension of y is large and the Hessian w.r.t y is sparse.

Approximating Objective

The Laplace approximation for the objective above is expressed in terms of the Hessian of the integrand w.r.t y. Its use has increased with the advent of good AD techniques.

Laplace Approximation

$H(x, y)$

$$H(x, y) = -\mathbf{p}(z|x, y)\,\mathbf{p}(y|x)\,\mathbf{p}(x)\ .$$

Laplace Approximation

$H(x, y)$

$$H(x, y) = -\mathbf{p}(z|x, y)\,\mathbf{p}(y|x)\,\mathbf{p}(x)\ .$$

$H_{y,y}(x, y)$

We use $H_{y,y}(x, y)$ to denote the Hessian of H .w.r.t y and we assume that it is always positive definite.

Laplace Approximation

$H(x, y)$

$H(x, y) = -\mathbf{p}(z|x, y)\ \mathbf{p}(y|x)\ \mathbf{p}(x)$.

$H_{y,y}(x, y)$

We use $H_{y,y}(x, y)$ to denote the Hessian of H .w.r.t y and we assume that it is always positive definite.

$Y(x)$

We use $Y(x)$ for the argmin of $H(x, y)$ with respect to y; i.e.,

$$L(x, Y(x)) = H_y(x, Y(x))^{\mathrm{T}} = 0 \ .$$

Laplace Approximation

$H(x, y)$

$H(x, y) = -\mathbf{p}(z|x, y)\ \mathbf{p}(y|x)\ \mathbf{p}(x)$.

$H_{y,y}(x, y)$

We use $H_{y,y}(x, y)$ to denote the Hessian of H .w.r.t y and we assume that it is always positive definite.

$Y(x)$

We use $Y(x)$ for the argmin of $H(x, y)$ with respect to y; i.e.,

$$L(x, Y(x)) = H_y(x, Y(x))^{\mathrm{T}} = 0 .$$

Approximate Objective

The Laplace approximation of the objective is

$$\frac{1}{2} \log \det \left(H_{y,y}(x, Y(x)) \right) + H(x, Y(x)) .$$

Approximate Objective

Motivation

The Hessian of the approximate objective can be used during optimization. It can also be used as a approximation of the the inverse of the covariance for the optimal solution.

Approximate Objective

Motivation

The Hessian of the approximate objective can be used during optimization. It can also be used as a approximation of the the inverse of the covariance for the optimal solution.

Optimal Value Component

$H(x, Y(x))$ is an optimal value function and has the same Hessian w.r.t to x as its one step representation ([BB08, Theorem 2]):

$$H(x, N_1(x))$$

Approximate Objective

Motivation

The Hessian of the approximate objective can be used during optimization. It can also be used as a approximation of the the inverse of the covariance for the optimal solution.

Optimal Value Component

$H(x, Y(x))$ is an optimal value function and has the same Hessian w.r.t to x as its one step representation ([BB08, Theorem 2]):

$$H(x, N_1(x))$$

Bilevel Programming Component

As direct application of the theorem in the talk,

$$\frac{1}{2} \log \det \left(H_{y,y}(x, Y(x)) \right)$$

has the same Hessian .w.r.t x as the two step approximation

Lemma

Incremental Multi-Index Sequence

Fix $p \in \mathbf{Z}_+^n$, the multi-index in the theorem, and select an incremental sequence

$$\{P(0), P(1), \cdots, P(|p|)\} \subset \mathbf{Z}_+^n$$

such that:

- $P(0) = 0$,

Incremental Multi-Index Sequence

Fix $p \in \mathbf{Z}_+^n$, the multi-index in the theorem, and select an incremental sequence

$$\{P(0), P(1), \cdots, P(|p|)\} \subset \mathbf{Z}_+^n$$

such that:

- $P(0) = 0$,
- $P(|p|) = p$,

Incremental Multi-Index Sequence

Fix $p \in \mathbf{Z}_+^n$, the multi-index in the theorem, and select an incremental sequence

$$\{P(0), P(1), \cdots, P(|p|)\} \subset \mathbf{Z}_+^n$$

such that:

- $P(0) = 0$,
- $P(|p|) = p$,
- $P(j+1) - P(j) \geq 0$,

Incremental Multi-Index Sequence

Fix $p \in \mathbf{Z}_+^n$, the multi-index in the theorem, and select an incremental sequence

$$\{P(0), P(1), \cdots, P(|p|)\} \subset \mathbf{Z}_+^n$$

such that:

- $P(0) = 0$,
- $P(|p|) = p$,
- $P(j+1) - P(j) \geq 0$,
- $|P(j+1) - P(j)| = 1$.

Incremental Multi-Index Sequence

Fix $p \in \mathbf{Z}_+^n$, the multi-index in the theorem, and select an incremental sequence

$$\{P(0), P(1), \cdots, P(|p|)\} \subset \mathbf{Z}_+^n$$

such that:

- $P(0) = 0$,
- $P(|p|) = p$,
- $P(j+1) - P(j) \geq 0$,
- $|P(j+1) - P(j)| = 1$.

Incremental Multi-Index Sequence

Fix $p \in \mathbf{Z}_+^n$, the multi-index in the theorem, and select an incremental sequence

$$\{P(0), P(1), \cdots, P(|p|)\} \subset \mathbf{Z}_+^n$$

such that:

- $P(0) = 0$,
- $P(|p|) = p$,
- $P(j+1) - P(j) \geq 0$,
- $|P(j+1) - P(j)| = 1$.

It follows that $|P(j)| = j$.

Notation

$L_{y,y[k]}(x, y)$

Define the matrix valued function $L_{y,y[k]} : \mathbf{R}^n \times \mathbf{R}^m \rightarrow \mathbf{R}^{m \times m}$ by

$$L_{y,y[k]}(x, y)_{[i,j]} = \partial_{L(x,y)_{[i,j]}} \partial_{y_[k]}$$

Notation

$L_{y,y[k]}(x,y)$

Define the matrix valued function $L_{y,y[k]} : \mathbf{R}^n \times \mathbf{R}^m \to \mathbf{R}^{m\times m}$ by

$$L_{y,y[k]}(x,y)_{[i,j]} = \partial_{L(x,y)_{[i,j]}} \partial_{y[k]}$$

$f^{(k)}$

For a vector valued function $f : \mathbf{R}^n \to \mathbf{R}^m$, and a $k \leq |p|$, define $f^{(k)} : \mathbf{R}^n \to \mathbf{R}^m$

$$f^{(k)}(x) = \partial^{P(k)} f(x) \ .$$

$L_x^{(k)}(x,y)$

Define the vector valued function $L_x^{(k)} : \mathbf{R}^n \times \mathbf{R}^m \to \mathbf{R}^m$ by

$$L_x^{(k)}(x,y) = \partial^{P(k)} L(x,y)$$

where the paritials are w.r.t. the x components.

Notation

$L_{y,y[k]}(x,y)$

Define the matrix valued function $L_{y,y[k]} : \mathbf{R}^n \times \mathbf{R}^m \to \mathbf{R}^{m \times m}$ by

$$L_{y,y[k]}(x,y)_{[i,j]} = \partial_{L(x,y)_{[i,j]}} \partial_{y_[k]}$$

$f^{(k)}$

For a vector valued function $f : \mathbf{R}^n \to \mathbf{R}^m$, and a $k \leq |p|$, define $f^{(k)} : \mathbf{R}^n \to \mathbf{R}^m$

$$f^{(k)}(x) = \partial^{P(k)} f(x) \ .$$

$L_x^{(k)}(x,y)$

Define the vector valued function $L_x^{(k)} : \mathbf{R}^n \times \mathbf{R}^m \to \mathbf{R}^m$ by

$$L_x^{(k)}(x,y) = \partial^{P(k)} L(x,y)$$

where the paritials are w.r.t. the x components.

Lemma

Statement

For $k = 1, \ldots, |p|$, there are functions F_k and G_k such that, for any k order differentiable function $Z : \mathbf{R}^n \to \mathbf{R}^m$, and any $x \in \mathbf{R}^n$

$$\begin{aligned} N(Z)^{(k)}(x) &= F_k\left(x, Z^{(0)}(x), ..., Z^{(k-1)}(x)\right) \\ &+ G_k\left(x, Z^{(0)}(x), ..., Z^{(k)}(x)\right) L(x, Z(x)) \,, \end{aligned}$$

Lemma

Statement

For $k = 1, \ldots, |p|$, there are functions F_k and G_k such that, for any k order differentiable function $Z : \mathbf{R}^n \to \mathbf{R}^m$, and any $x \in \mathbf{R}^n$

$$\begin{aligned} N(Z)^{(k)}(x) &= F_k\left(x, Z^{(0)}(x), ..., Z^{(k-1)}(x)\right) \\ &+ G_k\left(x, Z^{(0)}(x), ..., Z^{(k)}(x)\right) L(x, Z(x)) \ , \end{aligned}$$

Fact

If $A(x)$ is an differentiable invertible matrix value function,

$$\partial^p \left(A(x)^{-1}\right) = -A(x)^{-1} A^p(x) A(x)^{-1} \ .$$

Proof of Lemma: $k = 1$

$$N(Z)(x) = Z(x) - L_y(x, Z(x))^{-1} L(x, Z(x))$$

Proof of Lemma: $k = 1$

$$N(Z)(x) = Z(x) - L_y(x, Z(x))^{-1} L(x, Z(x))$$

$$\begin{aligned} N(Z)^{(1)}(x) &= Z^{(1)}(x) - L_y(x, Z(x))^{-1} L_y(x, Z(x)) Z^{(1)}(x) \\ &- L_y(x, Z(x))^{-1} L_x^{(1)}(x, Z(x)) \\ &- L_y(x, Z(x))^{-1} \left(\partial^{P(1)}(L_y(x, Z(x))) \right) L_y(x, Z(x))^{-1} \\ & L(x, Z(x)) \end{aligned}$$

Proof of Lemma: $k = 1$

$$N(Z)(x) = Z(x) - L_y(x, Z(x))^{-1} L(x, Z(x))$$

$$\begin{aligned}
N(Z)^{(1)}(x) &= Z^{(1)}(x) - L_y(x, Z(x))^{-1} L_y(x, Z(x)) Z^{(1)}(x) \\
&- L_y(x, Z(x))^{-1} L_x^{(1)}(x, Z(x)) \\
&- L_y(x, Z(x))^{-1} \left(\partial^{P(1)}(L_y(x, Z(x))) \right) L_y(x, Z(x))^{-1} \\
& L(x, Z(x))
\end{aligned}$$

$$\begin{aligned}
N(Z)^{(1)}(x) &= - L_y(x, Z(x))^{-1} L_x^{(1)}(x, Z(x)) \\
&- L_y(x, Z(x))^{-1} \left(\partial^{P(1)}(L_y(x, Z(x))) \right) L_y(x, Z(x))^{-1} \\
& L(x, Z(x))
\end{aligned}$$

Induction on k

Assume by induction the lemma holds for index k and let $r = P(k+1) - P(k)$. It follows that $|r| = 1$ and

Induction on k

Assume by induction the lemma holds for index k and let $r = P(k+1) - P(k)$. It follows that $|r| = 1$ and

$$\begin{aligned} N(Z)^{(k+1)}(x) &= \partial^r N(Z)^{(k)}(x) \\ &= \partial^r F_k\left(x, Z^{(0)}(x), \dots, Z^{(k-1)}(x)\right) \\ &+ G_k\left(x, Z^{(0)}(x), \dots, Z^{(k)}(x)\right) \partial^r L(x, Z(x)) \\ &+ \left(\partial^r G_k\left(x, Z^{(0)}(x), \dots, Z^{(k)}(x)\right)\right) L(x, Z(x)) \end{aligned}$$

Induction on k

Assume by induction the lemma holds for index k and let $r = P(k+1) - P(k)$. It follows that $|r| = 1$ and

$$
\begin{aligned}
N(Z)^{(k+1)}(x) &= \partial^r N(Z)^{(k)}(x) \\
&= \partial^r F_k\left(x, Z^{(0)}(x), \ldots, Z^{(k-1)}(x)\right) \\
&+ G_k\left(x, Z^{(0)}(x), \ldots, Z^{(k)}(x)\right) \partial^r L(x, Z(x)) \\
&+ \left(\partial^r G_k\left(x, Z^{(0)}(x), \ldots, Z^{(k)}(x)\right)\right) L(x, Z(x))
\end{aligned}
$$

This completes the inductive step and hence the proof of the lemma.

Bibliography

[AW00] Griewank Andreas and Andrea Walther.
Algorithm 799: Revolve: An implementation of checkpointing for the reverse or adjoint mode of computational differentiation.
ACM Transactions on Mathematical Software, 26:19–45, 2000.

[BB08] Bradley M. Bell and James V. Burke.
Algorithmic differentiation of implicit functions and optimal values.
In Christian H. Bischof, H. Martin BÃijcker, Paul Hovland, Uwe Naumann, and Jean Utke, editors, *Advances in Automatic Differentiation*, pages 67–77. Springer Verlag, 2008.

[Bel16] Bradley M. Bell.
cppad_mixed: Laplace Approximation of Mixed Effects Models.
IHME: Institute for Health Metrics and Evaluation, University of Washington,

http://moby.ihme.washington.edu/bradbell/cppad_mixed/, 2016.

[Gil92] Jean Charles Gilbert.
Automatic differentiation and iterative processes.
Optimization methods and software, 1:13–21, 1992.

[WWS10] Mathias Wagner, Andrea Waltherc, and Bernd-Jochen Schaefer.
On the efficient computation of high-order derivatives for implicitly defined functions.
Computer Physics Communications, 181:756–764, 2010.