View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by University of Hertfordshire Research Archive

Reverse Accumulation and Implicit Functions.

Bruce Christianson
(B.Christianson@herts.ac.uk)

Faculty of Information Sciences, University of Hertfordshire : Hatfield, England, Europe

June 1996 revised February, August 1997

Abstract. We begin by introducing a simple technique for using reverse accumulation to obtain the first derivatives of target functions which include in their construction the solution of systems of linear or nonlinear equations. In the linear case solving $Ay = b$ for y corresponds to the adjoint operations $\bar{b} := \bar{b} + v$ and $\bar{A} := \bar{A} - yv$ where v is the solution to the adjoint equations $vA = \bar{y}$. A more sophisticated construction applies in the nonlinear case.

We apply these techniques to obtain automatic numerical error estimates for calculated function values. These error estimates include the effects of inaccurate equation solution as well as rounding error.

Our basic techniques can be generalized to functions which contain several (linear or nonlinear) implicit functions in their construction, either serially or nested. In the case of scalar-valued target functions that include equation solution as part of their construction, our algorithms involve at most the same order of computational effort as the computation of the target function value, regardless of the number of independent variables or the size of the systems of equations.

1. Introduction. Automatic differentiation [8][10][11] is a set of techniques for obtaining derivatives of numerical functions to the same order of accuracy as the function values themselves, but without the labour of forming explicit symbolic expressions for the derivative functions. Automatic differentiation works by repeated use of the chain rule, but applied to numerical expressions rather than to symbolic values.

The forward accumulation technique of automatic differentiation associates with each program variable a vector containing partial derivatives of that variable with respect to the independent variables. The reverse accumulation technique builds a computational graph for the function evaluation, with one graph node (or *Wengert variable*) corresponding to each successive value held by a program variable, and associates with each node an *adjoint* variable containing the partial derivatives of the dependent variables with respect to that node. The adjoint values are calculated numerically in the opposite order to the function evaluation, whence the term 'reverse accumulation'. Reverse accumulation is of particular interest when the gradient (or sensitivities) of a single scalar-valued target (loss) function is required, as it allows the evaluation of the entire gradient vector of a scalar function at a computational cost of three function evaluations in total, regardless of the number of independent variables. For further details, see [8], [10], [11] and the references therein.

Many functions include in their computation the solution of a set of equations of the form $\Psi(y,u)=0$ where $u \in R^p, y \in R^q$ are (column) vectors of known and unknown quantities respectively and Ψ is a mapping $R^q \times R^p \to R^q$. Provided the function Ψ is sufficiently well-behaved, we can regard Ψ as the implicit definition at u of (a branch of) the function $y_*(u)$ defined by $\Psi(y_*(u),u)=0$. The eversion of this equation for a particular value of u, so as to construct a corresponding value for $y_*(u)$, is typically part of a larger computation of a dependent variable $z=f(x,y)$, where the x are the independent variables, $y=y_*(u)$, and the u are also functions of the x.

It is clearly desirable to extend the automatic differentiation techniques so as to determine such quantities as the gradient (row) vector $\nabla_x\, z$. The paper [7] addresses these issues and proposes an approach to the automatic evaluation of first derivatives, in the context of equation solution by iterative fixed point constructions. More detailed analyses in this context were given by Griewank et al [9] in the case where forward accumulation is used, and by Christianson [4] in the reverse case.

In this paper we examine a simple technique for using reverse accumulation to obtain first derivatives of a target function (dependent variable) which includes implicit function eversions in its construction. Some of the results and techniques presented here are generalizations of those given in [4], others are completely new. A corresponding analysis for the forward case has been given by Bartholomew-Biggs [1].

In the next section, following some mathematical preliminaries, we show that the adjoint quantities $\bar{u}$ can also be defined implicitly. In section 3, we use this to suggest an implementation approach which could be built on top of existing code. In section 4 we show how to obtain accurate error bounds for the effects of inexact equation solution on target function values which are calculated via an implicit constructor. In section 5 we extend this to provide a unified analysis of the effects of inexact solution of nonlinear equations and of rounding error and give an algorithm for calculating these error bounds numerically. In section 6 we discuss possible generalizations.

2. Adjoint Implicit Equations.

In what follows, we assume a norm $\|b\|$ for column vectors b, and denote by the same symbol the corresponding operator norm $\|A\| = \sup\{\|Ab\| \ st\ \|b\|=1\}$. We write $\|a\|$ for the norm of a row vector a considered as a linear operator on column vectors, so that $\|a\| = \|a^T\|_{adj}$ where T denotes the adjoint (transpose). Usually in practice we shall use the Euclidean norm, in which case $\|a\|_2 = \|a^T\|_2$ and the corresponding operator norm is the spectral norm, but we do not assume this in what follows. Because we are dealing with finite dimensional spaces, all complete norms are equivalent and we can speak without ambiguity of an open "neighbourhood" of a point. However the equivalence bounds can be arbitrarily large.

We regard the gradient operator ∇ as an adjoint operator, mapping scalars into row vectors and column vectors into Jacobian matrices.

If Ψ is a mapping of the form $\Psi(y,u) : R^q \times R^p \to R^q$ then we write Ψ_y, Ψ_u to denote $\nabla_y \Psi, \nabla_u \Psi$ respectively, when these derivatives exist. At any such point (y,u), $\Psi'(y,u) = (\Psi_y(y,u), \Psi_u(y,u))$ is a map $R^q \times R^p \to \mathcal{L}(R^q \times R^p \to R^q)$. On a region where Ψ' is everywhere defined, we say that Ψ' is Lipschitz with respect to a given norm if the map $(y,u) \to \Psi'(y,u)$ is Lipschitz with respect to the induced operator norm.

The results we give below remain true in the case where Ψ' is uniformly continuous (rather than Lipschitz). The methods of proof are the same, but the construction of bounds is notationally more cumbersome.

Definition 2.1 Fix the norm $\|.\|$ on $R^q \times R^p$, this induces norms on $R^q = R^q \times \{0\}, R^p = \{0\} \times R^p$ and on $\mathcal{L}(R^q \times R^p \to R^q)$. Choose $u_0 \in R^p, y_0 \in R^q$ and let U, Y be bounded open convex subsets of R^p, R^q respectively, such that $u_0 \in U, y_0 \in Y$. Since we are interested in local behaviour with respect to u, we lose nothing by restricting our attention to domains of this form.

We say that $\Psi : Y \times U \to R^q$ is a *well behaved implicit mapping* if the following four conditions are satisfied:
(a) $\Psi(y_0, u_0) = 0$
(b) Ψ is continuously differentiable throughout $Y \times U$
(c) Ψ' is Lipschitz with constant C throughout $Y \times U$ for some constant C
(d) Ψ_y is uniformly invertible throughout $Y \times U$,
ie there is some constant λ such that $\|\Psi_y^{-1}(y,u)\| < \lambda$ for all u in U and y in Y

Note that under these conditions there will always exist a constant κ with $\|\Psi_u(y,u)\| < \kappa$ for all u in U and y in Y. We could relax condition (d) in Definition 2.1, as the following Lemma shows.

Lemma 2.2 Suppose that $\Psi : Y \times U \to R^q$ satisfies conditions (a), (b) and (c) in Definition 2.1, and that Ψ_y is invertible at (y_0, u_0). Then there exist convex sets Y_0, U_0 with $y_0 \in Y_0 \subseteq Y$ and $u_0 \in U_0 \subseteq U$ such that Ψ restricted to $Y_0 \times U_0$ is a well-behaved implicit mapping.

Proof: Choose Y_0 and U_0 small enough to ensure that $\|y - y_0\|$ and $\|u - u_0\|$ are both less than

$$\frac{1}{3\ C\ \|\Psi_y^{-1}(y_0, u_0)\|}$$

for all $y \in Y_0, u \in U_0$. Define

$$A = \Psi_y(y,u), \qquad A_0 = \Psi_y(y_0,u_0), \qquad D = -(A - A_0)A_0^{-1}$$

Then $A = (I - D)A_0$ but

$$\|D\| \leq \|A - A_0\|.\|A_0^{-1}\| < \frac{2}{3}$$

so the series

$$A^{-1} = A_0^{-1}(I + D + D^2 + D^3 + ...)$$

converges in operator norm with $\|A^{-1}\| < 3\|A_0^{-1}\|$ for all $u \in U_0, y \in y_0$, and condition (d) is recovered.

QED.

By the implicit function theorem, a well-behaved implicit mapping defines a continuous function $y_* : U \to Y$ such that $\Psi(y_*(u), u) = 0$ for all $u \in U$, provided U is small enough relative to Y.

Lemma 2.3 Suppose Ψ a well behaved implicit mapping on $Y \times U$, with Ψ' Lipschitz with constant C, and $\|\Psi_y^{-1}\|, \|\Psi_u\|$ bounded throughout $Y \times U$ by λ, κ respectively. Suppose that U and Y are related in such a way that if $u \in U$ and $y \in R^q$ such that $\|y - y_0\| < 2\lambda\kappa\|u - u_0\|$ then $y \in Y$.

Then there is a unique continuous function $y_* : U \to Y$ such that

$$y_*(u_0) = y_0 \quad \text{and} \quad \Psi(y_*(u), u) = 0 \text{ for all } u \in U$$

Furthermore $y_*(u)$ is continuously differentiable with respect to u, with gradient given by

$$\nabla_u \ y_*(u) = \frac{dy_*}{du}(u) = -\Psi_y^{-1}(y_*(u), u)\Psi_u(y_*(u), u)$$

Proof: This follows from the implicit function theorem, defining $y_*(u)$ by integrating the given equation for the derivative along a line from u_0 to u. Taking norms gives $\|y_*(u) - y_0\| < \lambda\kappa\|u - u_0\|$ so we maintain the condition that $(y_*(u), u) \in Y \times U$ at each point along the line.

QED.

Theorem 2.4 Suppose Ψ a well behaved implicit mapping for y_* on the open domain $Y \times U$, with Ψ' Lipschitz with constant C, and $\|\Psi_y^{-1}\|, \|\Psi_u\|$ bounded throughout $Y \times U$ by λ, κ respectively.

Then the map $(y, u) \to \Psi_y^{-1}(y, u)$ is Lipschitz with constant

$$C \ \lambda^2$$

the map $(y, u) \to \Psi_y^{-1}(y, u)\Psi_u(y, u)$ is Lipschitz with constant

$$C \ \lambda(1 + \lambda\kappa)$$

and the map $u \to \nabla_u \; y_*(u)$ is Lipschitz with constant

$$C \; \lambda(1+\lambda\kappa)^2$$

Proof: Suppose $u_0, u \in U$; $y_0, y \in Y$ with $\|(y,u) - (y_0,u_0)\| < \epsilon$ and define

$$A = \Psi_y(y,u), \qquad A_0 = \Psi_y(y_0,u_0),$$

$$B = \Psi_u(y,u), \qquad B_0 = \Psi_u(y_0,u_0),$$

Since

$$A^{-1} - A_0^{-1} = -A^{-1}(A - A_0)A_0^{-1}$$

and the map $\Psi' : (y,u) \to (\Psi_y(y,u), \Psi_u(y,u))$ is Lipschitz with constant C throughout $Y \times U$, we have

$$\|A^{-1} - A_0^{-1}\| \le \lambda^2 \|A - A_0\| \le \lambda^2 C\epsilon$$

Also

$$A^{-1}B - A_0^{-1}B_0 = A^{-1}(B - B_0) + (A^{-1} - A_0^{-1})B_0$$

so

$$\|A^{-1}B - A_0^{-1}B_0\| \le \lambda\|B - B_0\| + \lambda^2\kappa\|A - A_0\|$$

whence

$$\|A^{-1}B - A_0^{-1}B_0\| \le (\lambda + \lambda^2\kappa)C\epsilon$$

Finally, as in Lemma 2.3 a line integral of $\nabla_u \; y_*(u)$ from u_0 to u shows that $\|y_*(u) - y_*(u_0)\| < \lambda\kappa\|u - u_0\|$ whence

$$\|(y_*(u),u) - (y_*(u_0),u_0)\| \le \|y_*(u) - y_*(u_0)\| + \|u - u_0\| \le \lambda\kappa\|u - u_0\| + \|u - u_0\|$$

which with the previous result gives the final assertion.

QED.

Theorem 2.5 Suppose Ψ a well behaved implicit mapping and let r be a fixed arbitrary adjoint (row) vector in $\mathcal{L}(R^q \to R) = (R^q)^T$.

Suppose second order derivatives of Ψ exist and are Lipschitz, then the operator $\Omega : R^{2q+p} \times R^{p+q} \to R^{2q+p}$ given by

$$\Omega((y, \zeta^T, \xi^T), (u, r)) = (\Psi(y,u), r - \zeta\Psi_y(y,u), \xi + \zeta\Psi_u(y,u))$$

is a well-behaved implicit mapping, and solving $\Omega = 0$ gives $\xi = r \; \nabla_u \; y_*(u)$.

Proof: Since $\Psi = 0$ we have $y = y_*(u)$, since $r = \zeta\Psi_y$ we have $\zeta = r\Psi_y^{-1}$ and since $\xi = -\zeta\Psi_u$ we have $\xi = -r\Psi_y^{-1}\Psi_u = r \; \nabla_u \; y_*(u)$ by Theorem 2.4

QED.

Note that if reverse accumulation is used to construct the directional derivatives $\zeta\Psi_y$ and $\zeta\Psi_u$ then the evaluation of Ω involves the same order of computational effort as the evaluation of Ψ.

3. Implementation Strategy.

3. Implementation Strategy. We seek to use the technique of reverse accumulation to calculate row vectors of the form $r\ \nabla_u\ y_*(u)$ for fixed row (adjoint) vectors r. We can use the construction in Theorem 2.5 to do this, which leads to the following.

Algorithm 3.1 Suppose as before that Ψ is a well-behaved implicit mapping for y_*, and assume now that the computation of $y_*(u)$ is part of a larger computation of a dependent variable $z = f(x, y)$, where the x are the independent variables, $y = y_*(u)$, and the u are also functions of the x.

To evaluate $\nabla_x\ z$ proceed as follows:
Step 1. Build the part of the computational graph corresponding to the calculation of u from the independent variables x.
Step 2. Switch off building the graph and construct y by solving the equations $\Psi(y, u) = 0$.
Step 3. Switch on graph construction and compute the variables $w = \Psi(y, u)$ (which should have value zero).
Step 4. Build the graph for the dependent variable $z = f(x, y)$.
Step 5. Initialize $\bar{z}$ to 1 as usual.
Step 6. Reverse through the graph from Step 4, accumulating the adjoint quantities $\bar{x} = f_x$ and $\bar{y} = f_y$.
Step 7. Solve the adjoint equations $\zeta\Psi_y = \bar{y}$ and set $\bar{w} = -\zeta$.
Step 8. Reverse through the graph from Step 3. (This has the effect of setting $\bar{u} := \bar{u} - \zeta\Psi_u$.)
Step 9. Reverse through the graph from Step 1, accumulating the adjoints $\bar{x}$ of the independent variables x.

Writing $\prec$ to mean "has a functional dependence upon", we have

$$x \prec u \prec y \prec z, \quad x \prec z, \quad y \prec w, \quad u \prec w.$$

Algorithm 3.1 is a template from which a number of well-known algorithms can be recovered by filling in particular strategies for performing Steps 2 and 7. It is not our intention here to advocate a preferred way of doing this: rather we wish to ensure that our results in §5 will apply to all algorithms which can be derived from this general template.

In practice we may have a linking between the strategies used in the two steps: for example there may exist some differentiable contractive iteration of the form $y := \Phi(y, u)$ with the property that the fixed point to which this converges satisfies the implicit equation $\Psi(y, u) = 0$. Whether or not this iteration is used in Step 2, the corresponding adjoint operator may be iterated to solve the adjoint equations in Step 7, as described in [4]. However even in this case the de-linked form of Algorithm 3.1 is useful, because Ψ may have very different numerical stability properties to $id_{\text{left}} - \Phi$.

Alternatively we may have a (possibly non-differentiable) method for eversion of the implicit equations in Step 2 which produces by-products that can be used in the solution of the (linear) equations in Step 7. We consider this case further in §6. Or it may be that there is a complete disconnection between the two steps, with the Jacobian Ψ_y being explicitly calculated and factorised to solve the linear equations for ζ.

Algorithm 3.1 has a particularly simple form in the case where the implicit mapping Ψ is itself linear.

Corollary 3.2 Suppose the computation of a dependent variable z contains the intermediate step

$$\text{solve } Ay = b \text{ for } y$$

where A and b are functions of the independent variables.

Then the corresponding adjoint steps on the reverse accumulation pass are

$$\text{solve } vA = \bar{y} \text{ for } v, \qquad \bar{b} := \bar{b} + v, \qquad \bar{A} := \bar{A} - yv.$$

where

$$\bar{A}_{ij} = \frac{\partial z}{\partial A_{ji}}$$

Proof: In Algorithm 3.1 set $u = (A|b)$ and $\Psi(y,u) = Ay - b$. Then $\Psi_y = A$ so

$$\nabla_u \, y = -\Psi_y^{-1}\Psi_u = -A^{-1}\Psi_u$$

For fixed row (adjoint) vector r we have $r\Psi_u = \left(\frac{yr}{-r}\right)$ so if $v = rA^{-1}$ then

$$r\nabla_u \, y = -rA^{-1}\Psi_u = -v\Psi_u = \left(\frac{-yv}{v}\right)$$

Now set $r = \bar{y}$ and the result follows.

QED.

4. Error Estimates for Function Values. There are a number of potential sources of numerical error in Algorithm 3.1. Rounding error introduces slight perturbations into the calculated values for u in Step 1. In Step 2 we may not solve the equations exactly for y. Even if the calculated value of w in Step 3 is exactly zero, this fact may itself be due to rounding errors in the evaluation of Ψ. These slight perturbations in y, together with the further effects of rounding error in Step 4, will perturb the calculated value of z. This in turn will have an effect on $\bar{y}$ and hence on $\bar{u}$ and $\bar{x}$. Again, in Step 7 we may not solve the adjoint equations exactly, and this will also affect the values calculated for $\bar{u}$ and $\bar{x}$.

We investigate further the effects of these errors in this section. We begin by investigating the effect of inexact solution in Step 2 of Algorithm 3.1 upon the calculated value for the dependent variable z.

Lemma 4.1 In the set-up of Algorithm 3.1, suppose that we are using accurate arithmetic (ie with no rounding error), but that we have $w \neq 0$ in Step 3, as a result of inaccurate equation solution in Step 2. Let $\hat{z}$ be the true value of the dependent variable (corresponding to the correct values $\hat{y}$ for y) and let z be the value calculated in Step 4. Assume that ζ is exactly calculated in Step 7, but using the incorrect values for y.

Let B be the ball with diameter $2\lambda\|w\|$ around (y, u) and assume that $B \subseteq Y \times U$. Let K be a Lipschitz constant for f_y on B. Then

$$\|z - \hat{z} - \zeta w\| \leq (K + C\lambda\|\bar{y}\|)\,(\lambda\|w\|)^2$$

Hence to first-order

$$\hat{z} = z - \zeta w.$$

Proof: By the inverse function theorem, treating u as fixed, we have

$$\frac{dy}{dw} = \Psi_y^{-1}.$$

Integrating this from 0 to w gives

$$\|y - \hat{y}\| \leq \lambda\|w\|$$

where y is the calculated (incorrect) value. Also

$$\frac{dz}{dw} - \zeta = f_y\Psi_y^{-1} - \zeta = (f_y - \bar{y})\Psi_y^{-1} + (\bar{y}\Psi_y^{-1} - \zeta)$$

where $\bar{y}$ is $f_y(y, u)$ evaluated at the calculated (incorrect) value for y and ζ is $\bar{y}\Psi_y^{-1}$ evaluated at this point. Certainly

$$\|f_y - \bar{y}\| \leq K\|y - \hat{y}\|$$

and

$$\|\Psi_y^{-1}(y, u) - \Psi_y^{-1}(\hat{y}, u)\| \leq C\lambda^2\|y - \hat{y}\|$$

by Theorem 2.4. Integrating as before now gives

$$\|z - \hat{z} - \zeta w\| \leq K\|y - \hat{y}\| \cdot \lambda\|w\| + \|\bar{y}\| \cdot C\lambda^2\|y - \hat{y}\| \cdot \|w\|$$

$$\leq (K + C\lambda\|\bar{y}\|)\,(\lambda\|w\|)^2$$

QED.

Note that if ζ is also calculated inexactly in Step 7, the effect on the value of $z - \zeta w$ is bounded by $\lambda \|\bar{y} - \zeta \Psi_y\| \cdot \|w\|$.

In the next Lemma, we examine the case where the equations for y are solved exactly, but for a perturbed value of u. The crucial point is that by Theorem 2.4, all the relevant operators are Lipschitz, and so the error growth is contained, in a fashion similar to the propagation of rounding error.

Lemma 4.2 In the set-up of Algorithm 3.1, suppose that we are using accurate arithmetic and exact equation solution, but that we slightly perturb the values of u. Let $\hat{z}$ be the true value of the dependent variable (corresponding to the correct values $\hat{u}$ for u) and let z be the value calculated in Step 4 by using $u = \hat{u} + \Delta_u$ in Step 2.

Let B be the ball with diameter $2\lambda\kappa\|\Delta u\|$ around (y, u), assume that $B \subseteq Y \times U$ and let K be a Lipschitz constant for $f' = (f_y, f_u)$ on B. Then

$$\|z - \hat{z} - \bar{u}\Delta u\| \leq \lambda(1 + \lambda\kappa)\left(\kappa K + C(1 + \lambda\kappa)\|\bar{y}\|\right)\|\Delta u\|^2$$

where ζ and $\bar{u} = -\zeta\Psi_u$ are calculated using $u = \hat{u} + \Delta u$.

Proof: Applying Lemma 2.3 to the equation $\nabla_u\, y_* = -\Psi_y^{-1}\Psi_u$ gives

$$\|\Delta y\| = \|y - \hat{y}\| \leq \lambda\kappa\|\Delta u\|$$

Pre-multiplying the same equation by f_y we have

$$\frac{dz}{du} - \bar{u} = -(f_y - \bar{y})\Psi_y^{-1}\Psi_u - (\bar{y}\Psi_y^{-1}\Psi_u + \bar{u})$$

and integrating this from $\hat{u}$ to u gives (using Theorem 2.4)

$$\|z - \hat{z} - \bar{u}\Delta u\| \leq \left(K\lambda\kappa + \|\bar{y}\|C\lambda(1 + \lambda\kappa)\right)\|(\Delta y, \Delta u)\| \cdot \|\Delta u\|$$

Now inserting the bound for Δy and rearranging gives the result.

QED.

Essentially the same analysis can be made of the potential error in $\bar{y}$ and $\bar{u}$.

Corollary 4.3 Under the conditions of Lemma 4.1 and Lemma 4.2, the sources of error in the values for $\bar{u}$ calculated by Algorithm 3.1 and bounds on their magnitudes are:

Error caused by using an inaccurate value $u + \Delta u$ in place of u in Steps 2, 7 and 8 at most

$$\|\bar{y}\|C\lambda(1 + \lambda\kappa)^2\|\Delta u\|$$

Error caused by inexact solution in Step 2 at most

$$\|\bar{y}\| C \lambda^2 (1 + \lambda\kappa) \|w\|$$

Error caused by inexact solution in Step 7 at most

$$\lambda\kappa \|\zeta \Psi_y - \bar{y}\|$$

Error caused by using inaccurate values of u and y in Steps 4 and 6 at most

$$\lambda\kappa \|\Delta\bar{y}\| \leq \lambda\kappa K \|(\Delta y, \Delta u)\|$$

where $\|\Delta y\|$ can be estimated using the bound

$$\|\Delta y\| \leq \lambda \|w\| + \lambda\kappa \|\Delta u\|$$

Proof: Theorem 2.4 and Lemmas 4.1, 4.2 *QED.*

If estimates of the constants λ, κ and C are available, then they can be used to check that the equation solution in Steps 2 and 7 of Algorithm 3.1 is sufficiently accurate to ensure the desired accuracy for z and $\bar{u}$.

From now on, we shall assume that the errors in $\bar{u}$ are "small" relative to $\max(\|\bar{u}\|, \epsilon)$ where ϵ is a machine constant which is "small" relative to one, and that consequently a first-order analysis suffices.

5. Rounding Error. An important application of reverse accumulation is the analysis of rounding error. The conventional method of reverse accumulation allows an analysis of the rounding error in a dependent variable to be made automatically. We summarize these results in the following.

Proposition 5.1 (M. Iri) Let z be a scalar dependent variable produced by a sequence of elementary operations, indexed by i. (We assume for convenience that the indexing includes the independent variables at the beginning and z itself at the end.) For each node v_i in the computational graph of the calculation for z, let $\bar{v_i} = \partial z / \partial v_i$ be the corresponding adjoint quantity calculated by reverse accumulation, and let δ_i be an upper bound on the absolute rounding error in v_i introduced by the elementary operation i which produced v_i. Define

$$e = \sum_i \delta_i \cdot |\bar{v}_i|, \qquad s = \sum_i \delta_i^2$$

Then the rounding error for z is bounded by $e + O(s)$.

Proof: (Outline.) We perform a reverse induction, in decreasing order of i. At stage i of the induction, we assume that all the values v_j with $j \leq i$

are calculated using inaccurate operations subject to rounding error, but that the values $\hat{v}_j$ for $j > i$ are calculated using exact arithmetic (with no rounding error) although the accurate operations are applied to possibly inaccurate arguments. To proceed from stage i to stage $i-1$, we replace the value v_i by the accurate value $\hat{v}_i$ obtained by applying the accurate elementary operation i to the (inaccurate) arguments, and we then re-calculate all the $\hat{v}_j$ with $j > i$.

The total effect of this on $\hat{z}$ is to add $\bar{v}_i(\hat{v}_i - v_i)$ together with a remainder term which is bounded by some multiple of $|\hat{v}_i - v_i|^2$. Since $|\hat{v}_i - v_i| < \delta_i$, completing the induction gives our result. For further details see [3, §7], [10] and the references cited there.

QED.

We now show how to provide a combined analysis of rounding and equation solution error where the computation of z involves constructing a value for an implicit function. We do this by applying Proposition 5.1 to the graph produced by Steps 1, 3 and 4 of Algorithm 3.1. For the purpose of this analysis, we treat the y as additional independent variables.

Algorithm 5.2 Augment Algorithm 3.1 by introducing a new scalar variable e. Initialize e to zero in Step 5, and add to Steps 6, 8 and 9 the operation

$$e := e + \delta_i \cdot |\bar{v}_i|$$

for each of the graph nodes v_i visited in the reverse accumlation, where δ_i is an upper bound on the absolute rounding error in v_i introduced by the elementary operation i which produced v_i.

Theorem 5.3 After the completion of Algorithm 5.2, assume that

$$\|\zeta\Psi_y - \bar{y}\| \leq \|w\|$$

Then

$$|\hat{z} - z + \zeta w| < e + O(s')$$

where $\hat{z}$ denotes the true value for the dependent variable, with accurate equation solution and no rounding error; z, ζ and w denote the values actually calculated in Algorithm 3.1, including the effects of rounding error and of inaccurate equation solution in Steps 2 and 7; and

$$s' = \|w\|^2 + \sum_i \delta_i^2$$

Proof: Lemma 4.1 tells us that in the absence of rounding error $z - \zeta w$ is an $O(\|w\|^2)$ estimate for $\hat{z}$. By our assumption, $\|\zeta - \bar{y}\Psi_y^{-1}\| \leq \lambda\|w\|$, so the effect upon $z - \zeta w$ of inaccurate equation solution for the calculated value of ζ in Step 7 will be $O(\|w\|^2)$ as well. Therefore it suffices to consider the effects of rounding error on the calculated value of $z - \zeta w$, treating the values of y as fixed constants.

The rounding errors which occur during Step 4 affect only the calculated value for z, by an amount which is already contained in $e + O(s')$ by Proposition 5.1. The rounding errors in Step 3 affect only the calculated values of w. By Step 7, the effect on $z - \zeta w$ of replacing v_i by $\hat{v}_i$ in Step 3 is

$$-\zeta \, \frac{dw}{dv_i} \, (\hat{v}_i - v_i) = \bar{v}_i(\hat{v}_i - v_i)$$

plus a remainder term of second order in $\hat{v}_i - v_i$. Since $|\hat{v}_i - v_i| < \delta_i$, the magnitude of the first order term is bounded by $\delta_i \cdot |\bar{v}_i|$ which is contained in e, and the remainder term by $O(\delta_i^2)$ which is contained in $O(s')$. The rounding error in Step 1 may affect both z and ζw. However the value $\bar{v}_i$ contains a component for each of these effects and once again the total effect on $z - \zeta w$ will be bounded by $\delta_i \cdot |\bar{v}_i|$ which is contained in e, plus a remainder term which is of the required order.

QED.

Theorem 5.3 says in essence that $e + \|\zeta w\|$ is, to first order, a tight worst case error bound for the entire calculation of z, including both rounding error and equation solution error. It is a worst case estimate, because it implicitly assumes that all rounding errors will affect z in the same direction, rather than cancelling.

An alternative analysis of rounding error can be made under the assumption that absolute rounding errors for v_i are independently distributed with mean zero and standard deviation δ_i rather than being worst case additive, see [10] and [8]. Analagous results can be proved in our set-up under this assumption. Given reasonable conditions on the distribution of the $|v_i|$ the error for z will be approximately normally distributed, with mean zero and standard deviation

$$\sigma = \sqrt{\sum_i (\delta_i \cdot \bar{v}_i)^2}$$

Consequently $|\hat{z} - z + \zeta w|$ is (very probably) bounded by 4σ.

6. Discussion. The main contributions of this paper are Algorithms 3.1 and 5.2. The total computational effort involved in these algorithms is typically of the same order as that required for the computation of the dependent variable z. Note that algorithms of this type do not involve forming $\Psi_y^{-1}\Psi_u$ explicitly. In particular, the computational effort of our algorithms does not explicitly depend upon the dimensions p and q. Although the cost of solving the adjoint equations will usually depend implicitly upon q, this cost is typically no worse than that of solving the original implicit equations and may indeed be very much less.

For example, if by-products of the original equation solution (such as LU decompositions) are available on the reverse pass, then the time required to do the reverse pass may be an order of magnitude less than that required to calculate the function value, see [5] and [12]. It may also be possible to precondition the equations $\Psi = 0$ so that the component equations have similar sensitivities.

If the target function (dependent variable) is evaluated frequently in the same region (typically true in the later stages of optimization where accurate solutions are required) and preconditioning is used together with some improvement algorithm to obtain the new solution from the old then the implicit dependence on q may disappear altogether.

The techniques and algorithms discussed in this paper can be applied recursively or iteratively to cope with nested or serial iterative implicit function constructions, and the fact that the adjoint equations are linear makes it easy in principle to extend the method to higher order directional derivatives or Taylor series. The paper [5] does just this in the case of a particular inverse diffusion problem. This in turn enables the use of conjugate gradient or truncated Newton method, see [6]. Other recent advances in the use of reverse accumulation are discussed in [2].

The paper [4] considered the iterative constructor case where

$$\|I - \Psi_y\| < \tau < 1 \quad \text{throughout } Y \times U, \quad \text{whence} \quad \lambda \leq \frac{1}{1-\tau}.$$

Here we make no such restriction on the norm of $I - \Psi_y$. Even in the case where the restriction is satisfied, the results in §5 of the present paper give powerful alternative accurate error bounds for truncation and rounding error, particularly when $\tau < 1/2$ in which case $\lambda < 2$. User code for these algorithms can be written in pretty much the same style as in the forward case, see [1].

References.

References

[1] M. Bartholomew-Biggs, 1997, Automatic Differentiation of Implicit Functions using Forward Accumulation, Computational Optimization and Applications, to appear

[2] M. Bartholomew-Biggs, S. Brown, B. Christianson *and* L.C.W. Dixon, 1996, The Efficient Calculation of Gradients, Jacobians and Hessians, Technical Report **301**, Numerical Optimisation Centre, University of Hertfordshire : Hatfield

[3] Bruce Christianson, 1992, Reverse Accumulation and Accurate Rounding Error Estimates for Taylor Series Coefficients, Optimization Methods and Software **1**(1), 81–94

[4] Bruce Christianson, 1994, Reverse Accumulation and Attractive Fixed Points, Optimization Methods and Software **3**(4), 311–326

[5] B. Christianson, A.J. Davies, L.C.W. Dixon, R. Roy *and* P. van der Zee, 1997, Giving Reverse Differentiation a Helping Hand, Optimization Methods and Software, to appear

[6] A.J. Davies, B. Christianson, L.C.W. Dixon, R. Roy *and* P. van der Zee, 1997, Reverse Differentiation and the Inverse Diffusion Problem, Advances in Engineering Software, to appear

[7] Jean Charles Gilbert, 1992, Automatic Differentiation and Iterative Processes, Optimization Methods and Software **1**(1), 13–21

[8] Andreas Griewank, 1989, On Automatic Differentiation, *in* Mathematical Programming 88, Kluwer Academic Publishers, Japan

[9] Andreas Griewank *et al*, 1993, Derivative Convergence for Iterative Equation Solvers, Optimization Methods and Software **2**(4) 321–355

[10] Masao Iri, 1991, History of Automatic Differentiation and Rounding Error Estimation, *in* Automatic Differentiation of Algorithms, SIAM, Philadelphia

[11] Louis B. Rall *and* George F. Corliss, 1996, An Introduction to Automatic Differentiation, *in* Computational Differentiation : Techniques, Applications and Tools, SIAM, Philadelphia

[12] Stephen P. Smith, Sparse Matrix Tools for Gaussian Models on Lattices, 1997, Computational Statistics and Data Analysis, to appear