Early works dealing with the reverse mode are [225] and [92] (see also [93]), while Griewank [63, 64] has studied the process more recently. Shiriaev [223] has included vector operations in the expanded system, thus reducing the number of intermediate results and increasing the efficiency considerably for many problems.

1.4.3 Slope Arithmetic

Any interval Lipschitz matrix, such as $\boldsymbol{F}'(\boldsymbol{X})$ of Definition 1.11, is also a slope bound matrix $\mathbf{S}(F, \boldsymbol{X}, \check{X})$ for any $\check{X} \in \boldsymbol{X}$. However, tighter bounds can be computed automatically for particular $\check{X}$ or for particular $\check{\boldsymbol{X}} \subset \boldsymbol{X}$. This is done with slope arithmetic that is similar to the differentiation arithmetic of §1.4. Formulas analogous to those of §1.4 will be presented here; slopes of multivariate functions can be computed from these and from Algorithm 3.

As with differentiation arithmetic, *slope arithmetic* is based on defining operations and standard functions on ordered triplets of the form $\ll\check{\boldsymbol{u}}, \boldsymbol{u}, \boldsymbol{u}^{(s)}\gg$. Addition and subtraction are defined the same way as with differentiation arithmetic, but multiplication and division differ somewhat: the rules[23] corresponding to formulas (1.29) are

$$\begin{array}{rcl} \ll\check{\boldsymbol{u}}, \boldsymbol{u}, \boldsymbol{u}^{(s)}\gg + \ll\check{\boldsymbol{v}}, \boldsymbol{v}, \boldsymbol{v}^{(s)}\gg & = & \ll\check{\boldsymbol{u}} + \check{\boldsymbol{v}}, \boldsymbol{u} + \boldsymbol{v}, \boldsymbol{u}^{(s)} + \boldsymbol{v}^{(s)}\gg, \\ \ll\check{\boldsymbol{u}}, \boldsymbol{u}, \boldsymbol{u}^{(s)}\gg - \ll\check{\boldsymbol{v}}, \boldsymbol{v}, \boldsymbol{v}^{(s)}\gg & = & \ll\check{\boldsymbol{u}} - \check{\boldsymbol{v}}, \boldsymbol{u} - \boldsymbol{v}, \boldsymbol{u}^{(s)} - \boldsymbol{v}^{(s)}\gg, \\ \ll\check{\boldsymbol{u}}, \boldsymbol{u}, \boldsymbol{u}^{(s)}\gg \times \ll\check{\boldsymbol{v}}, \boldsymbol{v}, \boldsymbol{v}^{(s)}\gg & = & \ll\check{\boldsymbol{u}}\check{\boldsymbol{v}}, \boldsymbol{u}\boldsymbol{v}, \boldsymbol{u}\boldsymbol{v}^{(s)} + \boldsymbol{u}^{(s)}\check{\boldsymbol{v}}\gg, \\ \ll\check{\boldsymbol{u}}, \boldsymbol{u}, \boldsymbol{u}^{(s)}\gg \div \ll\check{\boldsymbol{v}}, \boldsymbol{v}, \boldsymbol{v}^{(s)}\gg & = & \ll\check{\boldsymbol{u}}/\check{\boldsymbol{v}}, \boldsymbol{u}/\boldsymbol{v}, \frac{\boldsymbol{u}^{(s)} - \check{\boldsymbol{u}}\boldsymbol{v}^{(s)}/\check{\boldsymbol{v}}}{\boldsymbol{v}}\gg. \end{array} \tag{1.33}$$

As with differentiation arithmetic, rules could be defined for the standard functions according to the rules of differentiation arithmetic. However, the resulting slope bounds are not the sharpest possible. For standard functions $\omega(x)$ for which $\omega''(x) \neq 0$ for $x \in \boldsymbol{x}$, a difference formula computes the *exact* slope range $\mathbf{S}^\sharp(\omega, \boldsymbol{x}, \check{x})$ to within roundout error:

Theorem 1.13 (A slight generalization of a theorem of Rump in [208, 211]) *Suppose $\omega : \mathbb{R} \to \mathbb{R}$, $\boldsymbol{x} \in \mathbb{IR}$, $\check{\boldsymbol{x}} \in \mathbb{IR}$, and ω is either convex or concave (i.e. $\omega''(x) \neq 0$ if ω'' is continuous) for all x in the convex hull $\boldsymbol{x} \underline{\cup} \check{\boldsymbol{x}}$. Then $\mathbf{S}^{(d)}(\omega, \boldsymbol{x}, \check{\boldsymbol{x}})$ defined by*

$$\mathbf{S}^{(d)}(\omega, \boldsymbol{x}, \check{\boldsymbol{x}}) = h(\underline{x}) \underline{\cup} h(\overline{x}) \quad \textit{with} \quad h(x) = \begin{cases} \frac{\omega(x) - \omega(\check{\boldsymbol{x}})}{x - \check{\boldsymbol{x}}} & \textit{for } x \notin \check{\boldsymbol{x}} \\ \boldsymbol{\omega}'(\check{\boldsymbol{x}}) & \textit{otherwise} \end{cases} \tag{1.34}$$

[23] These rules were presented in [141], with additional insight in [208].

is a slope set for ω over $\boldsymbol{x}$ and centered on $\check{\boldsymbol{x}}$. If $\check{\boldsymbol{x}}$ is a point, then $\mathbf{S}^{(d)}(\omega, \boldsymbol{x}, \check{\boldsymbol{x}})$ is the smallest possible such slope set.

It was proven in [208] that $\mathbf{S}^{(d)}(\omega, \boldsymbol{x}, \check{x})$ is a slope set if $\omega'' \neq 0$, and it was pointed out in [211] that $\mathbf{S}^{(d)}(\omega, \boldsymbol{x}, \check{x})$ is a slope set when ω is convex over $\boldsymbol{x} \underline{\cup} \check{\boldsymbol{x}}$. The fact that this slope set is sharp for points $\check{x}$ follows from the fact that any slope set must contain $h(\underline{x})$ and $h(\overline{x})$. The fact that $\mathbf{S}^{(d)}(\omega, \boldsymbol{x}, \check{\boldsymbol{x}})$ is a slope set when $\check{\boldsymbol{x}}$ is non-degenerate interval follows from elementary properties of interval arithmetic, assuming $\boldsymbol{\omega}(\check{\boldsymbol{x}})$ contains the range of ω over $\check{\boldsymbol{x}}$.

In actual implementations, h is not defined exactly as in Equation (1.34), since roundout error becomes severe when x is close to $\check{\boldsymbol{x}}$, even though $x \notin \check{\boldsymbol{x}}$. In such implementations, $h(x)$ can be defined to be $\boldsymbol{\omega}'(\check{\boldsymbol{x}})$ whenever

$$<\check{\boldsymbol{x}} - x> \; < \sqrt{\epsilon_m}, \tag{1.35}$$

where ϵ_m is the machine epsilon.

Theorem 1.13 can be applied directly when $\omega \in \left\{\exp, \log, \sqrt{\ }\right\}$. It can also be applied directly to $\omega(x) = x^n$ with n even, since such ω are limiting cases of ω satisfying the hypotheses. It can be applied to $\omega(x) = x^n$ with n odd by considering the algebraic signs of the endpoints of $\boldsymbol{x}$ and $\check{\boldsymbol{x}}$, and it can be applied to functions such as $\omega(x, y) = x^y$, since they can be computed by composing log and exp. Theorem 1.13 can be applied when ω is an inverse trigonometric function by checking for the positions of the inflection points relative to $\boldsymbol{x}$ and $\check{\boldsymbol{x}}$.

It is somewhat more difficult to apply Theorem 1.13 when $\omega \in \{\sin, \cos\}$. For these functions, the interval derivative or other techniques can be used.

For example, a slope bound for $f(x) = x^4 + x^3 + x$, $\boldsymbol{x} = [.99, 1.01]$, $\check{\boldsymbol{x}} = \check{x} = 1$ can be evaluated as in Example 1.3 on page 37:

$$\begin{array}{rcl}
\ll \check{x}_2, x_2, x_2^{(s)} \gg & \leftarrow & \ll \check{x}, x, 1 \gg^4 = \ll 1, [.99, 1.01]^4, \mathbf{S}^\sharp(x^4, [.99, 1.01], 1) \gg \\
& \subset & \ll 1, [.9606, 1.041], [3.940, 4.061] \gg, \\
\ll \check{x}_3, x_3, x_3^{(s)} \gg & \leftarrow & \ll \check{x}, x, 1 \gg^3 = \ll 1, [.99, 1.01]^3, \mathbf{S}^\sharp(x^3, [.99, 1.01], 1) \gg \\
& \subset & \ll 1, [.9702, 1.031], [2.970, 3.031] \gg, \\
\ll \check{x}_4, x_4, x_4^{(s)} \gg & \leftarrow & \ll \check{x}_2, x_2, x_2^{(s)} \gg + \ll \check{x}_3, x_3, x_3^{(s)} \gg \\
& \subset & \ll 2, [1.930, 2.072], [6.910, 7.092] \gg, \\
\ll \check{x}_5, x_5, x_5^{(s)} \gg & \leftarrow & \ll \check{x}_4, x_4, x_4^{(s)} \gg + \ll \check{x}, x, 1 \gg \\
& \subset & \ll 3, [2.920, 3.082], [7.910, 8.092] \gg.
\end{array}$$

Thus, an interval enclosure for $\{f(x) \mid x \in [.99, 1.01]\}$ is $[2.920, 3.082]$, the value $f(1)$ is 3, and an interval enclosure for the slope is:

$$\{\mathbf{S}^\sharp(f, x, \check{x}) \mid x \in [.99, 1.01], \check{x} = 1\} \subset [7.910, 8.092].$$

Note that $[7.910, 8.092] \subset [7.821, 8.183]$, where $[7.821, 8.183]$ is the interval bound on the derivative obtained in Example 1.3. In fact, $\mathrm{w}([7.910, 8.092]) = 0.182 \approx \frac{1}{2}\mathrm{w}([7.821, 8.183]) = 0.362$, as theory of $\mathbf{S}^\sharp$ would predict.

A final note: Neumaier has suggested that slope arithmetic can be done with matrices, rather than with numbers. This would obviate the need for Hansen's slope technique, and also be more efficient that computing derivative matrices.

1.4.4 Operator Overloading and Code List Generation

In the software described in this work, both the forward mode and reverse mode will be used, as well as a *symbolic* version of each. To do so, methods explained early in [187] (and earlier in [131]) will be used to first create an internal representation, termed a *code list*, of the function from a user-created program. Generic routines are then used, with this code list as input, to compute interval and floating point function and derivative values, and to symbolically differentiate a code list. In this work[24], the code list will be created with *operator overloading*.

[24] Alternatives are to use operator overloading directly in automatic differentiation, without creating a code list, to implement a special programming language with a derivative data type, or to create a special computer program that generates a code list. Each has advantages and disadvantages.