Symbolic–Numeric Nonlinear Equation Solving

Kelly Roach
Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1

Abstract

A numerical equation–solving algorithm employing differentiation and interval arithmetic is presented which finds all solutions of $f(z) = 0$ on an interval I when f is holomorphic and has simple zeros. A two dimensional generalization of this algorithm is discussed. Finally, aspects of a broader symbolic–numeric algorithm which uses the first algorithm as a foundation are considered.

1. Introduction

Computer algebra systems such as Maple and Mathematica include numerical equation solvers such as **fsolve** and **NSolve**. However, it seems that existing equation solvers are limited because they do not return all solutions and they are not guaranteed to find a solution if one exists. Outside of polynomial systems, symbolic equation solving algorithms of such systems seem to be in even worse shape. Published numerical algorithms like Brent [2] start out with local assumptions about equations $f(z) = 0$ such as a change of sign in $f(z)$ has been detected and make it their goal to find a single root of $f(z)$. Nerinckx and Haegemans [10] give a comparative study of 10 FORTRAN and ALGOL programs nonlinear equation solvers. Rice [13] presents a slightly more up to date comparison of 8 nonlinear equation solvers.

We seek a more global approach which is guaranteed to find all solutions of an equation $f(z) = 0$ by assuming f is holomorphic (a smooth function) and the domain I of the unknown z is finite. The last restriction is needed because Richardson [12] shows the problem is otherwise unsolvable. We assume the zeros of $f(z)$ are simple (meaning $f(a) = 0$ implies $f'(a) \neq 0$), though we would like to lift this restriction. Methods for escaping this restriction in some cases with multiple zeros (not simple) are discussed later.

Although "nonlinear equation" in general has a very broad meaning (see Saaty [15]), in this paper a nonlinear equation is an equation $f(z) = 0$ where f is a univariate function that is not of the form $f(z) = a\,z + b$. Efficient algorithms for equations where $f(z)$ is a polynomial in z, such as Uspensky's algorithm or Sturm sequences are known and are described in Collins and Loos [4]. Our emphasis is on solving more difficult non–polynomial equations like

$$e^{\cos(z)} - \sin(z \sin(z)) = 0$$

and at the same time offering a guarantee that we have found all the solutions.

2. Simple Zeros of a Holomorphic Function

In this section we sketch ideas we have implemented to compute zeros of a smooth real-valued function on a real interval. Let $f : \Omega \to C$ be holomorphic on $\Omega \subset C$, $I = [a_0, a_1]$ be a closed real interval contained in Ω, f be real valued on I, f be not constant on I, and f have simple zeros on I.

Since f is holomorphic, f is infinitely differentiable. (Corollary 2.12, page 73, Conway [5]). Since I is compact, f and f' have finitely many zeros on I. (Theorem 3.7, page 78, Conway [5]). The zeros of f' on I are distinct from the zeros of f on I because the zeros of f are simple. Therefore, recursively bisecting I into increasingly smaller subintervals eventually isolates each zero r of f on I in a subinterval $I_r \subset I$ on which f is also strictly monotone.

Now, we assume the existence of an interval arithmetic package Φ such that given any derivative $f^{(n)}$ of f and subinterval $[b_0, b_1] \subset I$, we have

$$\Phi(f^{(n)}, [b_0, b_1]) = [c_0, c_1]$$

with either $[c_0, c_1] \subset R$ or $[c_0, c_1] = [-\infty, \infty]$. Given any set $S \subset R$, define open neighborhoods $N_\epsilon(S)$ about S by

$$N_\epsilon(S) = \{x \mid \exists y \in S \quad \text{such that} \quad |x - y| < \epsilon\}$$

We let $\overline{N}_\epsilon(S)$ represent the closure of $N_\epsilon(S)$.

We will assume Φ has the following continuity property: Given any $\epsilon > 0$, $x \in I$, and derivative $f^{(n)}$ of f, there exists $\delta > 0$ (δ may depend on ϵ, x, and $f^{(n)}$) such that if $J = \overline{N}_\delta(x)$, then $\Phi(f^{(n)}, J) \subset N_\epsilon(f^{(n)}(J))$. In fact, by the compactness of I we can suppose δ is independent of x (see Rudin [14]).

All of the above leads us to the following primitive algorithm for detecting zeros of f.

```
proc Zeros(f, $[a_0, a_1]$)
if not CouldBeZero(f, $[a_0, a_1]$) then
  return $\emptyset$;
elif not CouldBeZero($f'$, $[a_0, a_1]$) then
  if f($a_0$) f($a_1$) $> 0$ then
    return $\emptyset$;
  else
    return Locate(f, $[a_0, a_1]$);
  fi;
else
  $a := (a_0 + a_1)/2$;
  if f($a$) $\neq 0$ then
    return Zeros(f, $[a_0, a]$) $\cup$ Zeros(f, $[a, a_1]$);
  else
    $\delta := (a_1 - a_0)/4$;
    while CouldBeZero($f'$, $[a - \delta, a + \delta]$) do
      $\delta := \delta/2$;
    od;
    return Zeros(f, $[a_0, a - \delta]$) $\cup \{a\}$
      $\cup$ Zeros(f, $[a + \delta, a_1]$);
  fi;
fi;

proc CouldBeZero(f, $[a_0, a_1]$)
return (f($a_0$) f($a_1$) $\leq 0$ or $0 \in \Phi$(f, $[a_0, a_1]$));

proc Locate(f, $[a_0, a_1]$)
if $0 \in [a_0, a_1]$ and f(0) $= 0$ then
  return $\{0\}$;
else
  while true do
    $a := (a_0 + a_1)/2$;
    if $a_0\, a_1 > 0$
      and $|a_0 - a_1|/\min(|a_0|, |a_1|) < PREC$ then
      break;
    elif f($a$) f($a_1$) $> 0$ then
      $a_1 := a$;
    else
      $a_0 := a$;
    fi;
  od;
  return $\{a\}$;
fi;
```

$PREC$ is a small positive number like 10^{-29} determined by the current working precision. The next two sections sketch ideas that improve the time efficiency of this algorithm. The Mean-Value Theorem speeds up procedure Zeros. Newton's method speeds up procedure Locate.

3. Application of the Mean-Value Theorem

From calculus we know that if $N_\epsilon(x) \subset I = [a_0, a_1]$ then

$$f(x + \epsilon) = f(x) + f'(\xi)\epsilon$$

for some $\xi \in N_\epsilon(x) \subset I$. If $|f'(\xi)| < B$ for all $\xi \in I$ and $f(x) \neq 0$, then $f(y) \neq 0$ for all $y \in N_\epsilon(x) \cap I$ where $\epsilon = |f(x)|/B$. Thus, while searching for zeros of f on I, we can eliminate all of $N_\epsilon(x) \cap I$ from consideration if we know $f(x) \neq 0$ and a finite bound B. If interval I is sufficiently small, then $\Phi(f', I)$ will determine a finite bound B on f$'$. This leads to procedure **Narrow**(f,I) which trims the ends of interval I by this technique before calling **Zeros** (if it must). The recursive calls in **Zeros** are changed to be calls to procedure **Narrow**.

We change lines 13 and 19-20 of procedure **Zeros** to be:

```
return Narrow(f, $[a_0, a]$) $\cup$ Narrow(f, $[a, a_1]$);

return Narrow(f, $[a_0, a - \delta]$) $\cup \{a\}$
  $\cup$ Narrow(f, $[a + \delta, a_1]$);
```

and add procedure **Narrow** below:

```
proc Narrow(f, $[a_0, a_1]$)
$B$ := Bound($f'$, $[a_0, a_1]$);
if $B \neq \infty$ then
  for $i$ from 1 to $FUDGE_1$ do
    $a_0 := a_0 + |f(a_0)|/B$;
    $a_1 := a_1 - |f(a_1)|/B$;
    if $a_1 < a_0$ then
      return $\emptyset$;
    fi;
  od;
fi;
return Zeros(f, $[a_0, a_1]$);

proc Bound(f, $[a_0, a_1]$)
$[b_0, b_1] := \Phi$(f, $[a_0, a_1]$);
return max($|b_0|$, $|b_1|$);
```

$FUDGE_1$ is a "fudge factor" which we currently set as $FUDGE_1 = 20$.

4. Newton's Method

Once a root is sufficiently isolated by bisection and narrowing, a faster numerical method can be applied to obtain remaining digits of the root. Newton's method, the secant method, Muller's algorithm, third–order Newton iteration, and quadratic inverse interpolation all have superlinear convergence and are described in Rice [13]. We use Newton's method because it is well–known, is efficient, has low–overhead, and is easy to analyze.

Given function f, define function g by

$$g(z) = z - \frac{f(z)}{f'(z)}$$

If r is a root of f on I_r, then $g(r) = r$. Hence r is a fixed point of g. Newton's method for calculating root r of f starts with an approximation r_0 and then successively computes $r_{n+1} = g(r_n)$. Under the right conditions, explained below, the sequence r_0, r_1, r_2, ... converges to r.

Formulas for the derivatives of g are:

$$g'(z) = \frac{f(z)\,f''(z)}{f'(z)^2}$$

$$g''(z) = \frac{f''(z)}{f'(z)} - \frac{2\,f''(z)^2}{f'(z)^2} + \frac{f(z)\,f'''(z)}{f'(z)^2}$$

Letting $\epsilon_n = r_n - r$ and using $g(r) = r$ and $g'(r) = 0$ we obtain via Taylor series expansions the results

$$\epsilon_{n+1} = g'(\xi_1)\,\epsilon_n$$

$$\epsilon_{n+1} = \frac{g''(\xi_2)}{2}\,\epsilon_n{}^2$$

for some $\xi_1, \xi_2 \in N_{\epsilon_n}(r)$. If $|g'(\xi_1)| < B_1$ and $|g''(\xi_2)| < B_2$ for all $\xi_1, \xi_2 \in I_r$ and $f'(z) \neq 0$ for all $z \in I_r$, then Newton's method is safe to use and converges faster than bisection if either

$$B_1 < \frac{1}{2} \quad \text{or} \quad B_2 < \frac{1}{2\,\epsilon_0} \leq \frac{1}{2\,(a_1 - a_0)}$$

and we can guarantee that all $r_i \in I_r$. The latter can be arranged by tweaking Newton's method so that if $r_1 = \mathrm{g}(r_0)$ overshoots the boundaries of interval I_r, then r_1 is shoved back towards the root r. Subsequent r_i computed by $r_{n+1} = \mathrm{g}(r_n)$ then proceeds without incident. Hence, we use

$$r_{n+1} = \begin{cases} \min(a_0, \max(\mathrm{g}(r_0), a_1)), & \text{if } n = 0 \\ \mathrm{g}(r_n), & \text{if } n \geq 1 \end{cases}$$

Now we assume Φ's continuity property extends to derivatives $\mathrm{g}^{(n)}$ of g on subinterval $I_r \subset I$. If interval $J \subset I_r$ is sufficiently small, then $\Phi(\mathrm{g}', J)$ and $\Phi(\mathrm{g}'', J)$ will determine finite bounds B_1 and B_2.

We modify procedure **Locate** and introduce procedures **NewtonTest** and **Newton**. Procedure **NewtonTest** tests whether Newton's method should be used and procedure **Newton** actually implements Newton's method. These are listed below.

```
proc Locate(f, [$a_0, a_1$])
if $0 \in [a_0, a_1]$ and f(0) = 0 then
  return {0};
else
  for $i$ from 1 do
    $a := (a_0 + a_1)/2$;
    if ($i$ mod $FUDGE_2$) = 0
      and NewtonTest(f, [$a_0, a_1$]) then
      $a :=$ Newton(f, [$a_0, a_1$]);
      break;
    elif $a_0\, a_1 > 0$
      and $|a_0 - a_1|/\min(|a_0|, |a_1|) < PREC$ then
      break;
    elif f($a$) f($a_1$) > 0 then
      $a_1 := a$;
    else
      $a_0 := a$;
    fi;
  od;
  return {$a$};
fi;

proc NewtonTest(f, [$a_0, a_1$])
global $B_1, B_2$;
if CouldBeZero($f'$, [$a_0, a_1$]) then
  return false;
else
  $\epsilon := a_1 - a_0$;
  $a := (a_1 + a_0)/2$;
  $B_1 := g'(a)$;
  $B_2 := g''(a)/2$;
  if $B_1 \geq 1/2$ and $B_2 \geq 1/(2\,\epsilon)$ then
    return false;
  else
    $B_1 :=$ Bound($g'$, [$a_0, a_1$]);
    $B_2 :=$ Bound($g''/2$, [$a_0, a_1$]);
    if $B_1 \geq 1/2$ and $B_2 \geq 1/(2\,\epsilon)$ then
      return false;
    else
      return true;
    fi;
  fi;
fi;

proc Newton(f, [$a_0, a_1$])
global $B_1, B_2$;
$a := (a_1 + a_0)/2$;
$a := \max(a_0, \min(\mathrm{g}(a), a_1))$;
$\epsilon := (a_1 - a_0)/2$;
while $\epsilon/(|a| - \epsilon) \geq PREC$ and $B_1 \leq B_2\,\epsilon$ do
  $a := \mathrm{g}(a)$;
  $\epsilon := B_1\,\epsilon$;
od;
while $\epsilon/(|a| - \epsilon) \geq PREC$ do
  $a := \mathrm{g}(a)$;
  $\epsilon := B_2\,\epsilon^2$;
od;
return $a$;
```

$FUDGE_2$ is another "fudge factor" which in our algorithm is currently set to the value $FUDGE_2 = 6$.

5. Results

The preceding algorithm has been implemented in Maple and run on various inputs including the following examples:

(1) $e^z - 6\,z \quad$ for $z \in [0, 4]$

(2) $\sin(z^2)\log(1 + z) - \cos(\sqrt{2}\,z) \quad$ for $z \in [0, 4]$

Figures 1.1–1.2 picture these two functions.

The operations of procedures **Zeros** and **Narrow** are represented in Figures 2.1–2.2. Each right triangle represents a narrowing step by procedure Narrow. A triangle with vertices $(a, \mathrm{f}(a))$, $(a, 0)$, $(a + |\mathrm{f}(a)|/B, 0)$ represents **Narrow** detecting $\mathrm{f}(a) \neq 0$ and $B = \mathrm{Bound}(f', [a_0, a_1]) < \infty$ allowing **Narrow** to eliminate subinterval $(a, a + |\mathrm{f}(a)|/B)$ as a possible location for any zeros of f.

The effects of procedure **Zeros** can be seen wherever there is a change in the slope of the hypotenuses of the triangles or two right triangles' altitudes merge together.

Gaps on the z-axis between the triangles (most visible in Figure 2.2), represent procedures **Locate** and **Newton** taking over from procedures **Zeros** and **Narrow** to locate an isolated root of f. Closeups of this process for selected roots are shown in Figures 3.-1-3.2. Points $(a, \mathrm{f}(a))$ in the trajectories computed by **Locate** and **Newton** are also shown. The points converge onto the z-axes where the roots are located.

In Maple, we choose **nsolve** as the name of the main routine to resemble Maple's routine **fsolve**. Procedure **nsolve** returns Nonlin(...) objects and these, like E or Pi, can be evaluated via Maple's **evalf** to as many decimal places as desired. Thus,

```
> nsolve(E^z-6*z,z,0..4);

   [Nonlin(...), Nonlin(...)]

> evalf(",30);

   [.204481449339915533617757754510,
    2.83314789204934214261167464234]
```

Each Nonlin(...) object records the nonlinear equation which is solved, the Newton iteration formula, a bounding interval for the root, and the two bounds B_1, B_2 needed by procedure Newton to terminate properly.

6. Two Dimensional Systems

Let $f_1, f_2 : \Omega \to C \times C$ be holomorphic on $\Omega \subset C \times C$; $I = [a_0, a_1] \times [b_0, b_1]$ be a closed real parallelepiped contained in Ω; f_1, f_2 be real valued on I; curves $f_1(x_1, x_2) = 0$ and $f_2(x_1, x_2) = 0$ intersect in a finite number of points; the partial derivatives $\partial f_i / \partial x_j \neq 0$ at each such point; the Jacobian $J = [\partial f_i / \partial x_j]$ is nonsingular at each such point.

The two dimensional algorithm is like the one dimensional algorithm in that it uses bisection. However in this case, bisection means cutting rectangles in half, either widthwise or lengthwise depending on which way is most profitable in improving precision. After each bisection, $\Phi(f_i, I)$ and $\Phi(\partial f_i / \partial x_j, I)$ are computed. If $0 \notin \Phi(f_1, I)$ or $0 \notin \Phi(f_2, I)$, then I can be eliminated. Otherwise, continued bisection produces I such that $0 \notin \Phi(\partial f_i / \partial x_j, I)$, in which case the extrema of f_1 and f_2 must occur at the corners of I. It is then possible to determine if $f_i \neq 0$ on some edge E of I and if so to use $\Phi(\partial f_i / \partial x_j, E) \leq B < \infty$ to perform narrowing.

Bisection and narrowing ultimately produce a parallelepiped I such that $0 \notin \Phi(\det(J), I)$ and both f_1 and f_2 change sign on I. A nonsingular Jacobian J implies that $f_1(x_1, x_2) = 0$ and $f_2(x_1, x_2) = 0$ can intersect in at most one point p in I. Each curve $f_i(x_1, x_2) = 0$ intersects the boundary ∂I of I twice at points p_{i1} and p_{i2}. Supposing p is in the interior of I, then all 4 intersection points p_{ij} are distinct and can be isolated from each other. If the p_{1j} points alternate with the p_{2j} points around the perimeter of I, then the two curves actually do intersect and point p exists. Otherwise, the two curves do not intersect and p does not exist.

The generalization of Newton's formula to two dimensions is

$$\vec{g}(\vec{z}) = \vec{z} - J^{-1} \cdot \vec{f}(\vec{z})$$

and bounds $\|D\vec{g}(\vec{\xi}_1)\|_2 < B_1$ and $\|D^2\vec{g}(\vec{\xi}_1)\|_2 < B_2$ must be determined.

7. Results

The two dimensional algorithm has been implemented in Maple and run on some inputs including

(1) $e^x - 6\,y, \quad e^y - 6\,x \quad$ for $(x, y) \in [0,4] \times [0,4]$

(2) $x - 3\cos(3\,x) - y, \;\; x - y - 2\cos(2\,y)$
for $(x, y) \in [0,4] \times [0,4]$

Figures 4.1–4.2 plot the curves $f_1(x, y) = 0$ and $f_2(x, y) = 0$ for these examples. The solutions being sought are the intersection points of these curves.

The operations of the two dimensional **Zeros** and **Narrow** procedures are represented graphically in Figures 5.1–5.2. Most rectangles represent regions of the xy–plane eliminated by steps in either **Zeros** or **Narrow**. Only the very smallest rectangles represent regions that have not been eliminated and these surround the solutions we seek.

Finally, Figures 6.1–6.2 are closeups for selected solutions of these two systems which picture the trajectories of approximate solutions (x_n, y_n) computed by two dimensional versions of the **Locate** and **Newton** procedures. The Newton iteration formulas for example (1) are

$$g_1(x, y) = \frac{e^y\,(e^x\,x - e^x + 6\,y - 6)}{e^x\,e^y - 36}$$

$$g_2(x, y) = \frac{e^x\,(e^y\,y - 6 - e^y + 6\,x)}{e^x\,e^y - 36}$$

and these lead to solution (x,y)=(2.8331478920493421425, 2.8331478920493421425) as shown in Figure 6.1.

8. Multiple Zeros, Non-Holomorphic Functions, Infinite Intervals

Hoenders and Slump [7] describe a method for determining number and multiplicities of zeros of a function based on a numerical quadrature technique, but as they show in their Tables 4–5, this technique is unstable. It seems unlikely to us that any numerical technique could decisively solve this problem. Symbolic methods can come to our aid in some instances. First, if $f(z) \in K(z)$ is a rational function over a suitable computable extension K of Q, then square–free factorization is applicable. Second, if $f(z) \in K(z, \theta(z))$ where $K(z, \theta(z))$ is a purely transcendental extension of a computable extension K of Q, $z = a$ is a multiple zero, and $f(z) = h(z, \theta(z))$ where h is a rational function, then

$$r_1(a, \theta(a)) = h(a, \theta(a)) = 0$$

$$r_2(a, \theta(a)) = h_1(a, \theta(a)) + h_2(a, \theta(a))\,\theta'(a) = 0$$

are rational functions in a and $\theta(a)$ identical to zero. Since $y = \theta(a)$ is a common root of both $r_1(a, y)$ and $r_2(a, y)$, the resultant

$$r(a) = \mathrm{res}(r_1(a, y), r_2(a, y), y)$$

must be zero. Hence a is a root of $r(x) = \mathrm{res}(r_1(x, y), r_2(x, y), y)$ and is algebraic over K. The equation $r(x)$ can be solved symbolically for $x = a$ and then this solution substituted into $f(x)$ and $f'(x)$ to ascertain if $f(a) = f'(a) = 0$ proving $x = a$ is a multiple zero. Third, if $f(z)$ can be decomposed as a composition $f(z) = u(v(z))$, $z = a$ is a root of $u(z)$ with multiplicity m, $z = b$ is a root of $v(z) - a$ with multiplicity n, then $z = a$ is a root of $f(z)$ with multiplicity $m\,n$.

A piecewise holomorphic function defined by

$$f = \begin{cases} f_1(z), & \text{for } z \in I_1 \\ f_2(z), & \text{for } z \in I_2 \\ \dots & \\ f_n(z), & \text{for } z \in I_n \end{cases}$$

consisting of finitely many holomorphic $f_i : \Omega \to C$ may be treated as n separate inputs to our algorithm.

An expression f containing non–holomorphic subexpressions can sometimes be rewritten to become holomorphic or piecewise holomorphic. As an example, suppose $f(z) = u(|v(z)|)$ where u and v are holomorphic. Then solve $u(z) = 0$ and replace $|u(z)|$ by better expressions $\pm u(z)$.

Our algorithm is restricted to a finite interval I. In some cases, such as $f(z) = \sin(z)$, this is necessary, for otherwise there would be an infinite number of solutions. In other cases, such as $f(z) = e^z - 6\,z$, there are only finitely many solutions even on $I = R$. The strategy proposed in this case is to use some asymptotic analysis to find $a, b \in R$ such that $f(z)$ is non-zero on $(-\infty, a)$ and (b, ∞) leaving only $I = [a, b]$ to contend with.

9. Elementary Solutions

Bundy and Welham [3] describe concepts of **attraction**, **collection**, and **isolation** which sometimes lead to solutions of nonlinear equations. Taking a somewhat related algorithmic approach, suppose $f(z) \in K(z, \theta_1, \dots, \theta_n)$ is an elementary expression where each θ_i is exponential, logarithmic, or algebraic. Factoring, we suppose $f(z) = p(z, \theta_1, \dots, \theta_n)$ is irreducible in $R = K[z, \theta_1, \dots, \theta_n]$. Apply rules

(1) $(e^{\eta_1(z)})^a\,(e^{\eta_2(z)})^b \to e^{a\,\eta_1(z)+b\,\eta_2(z)}$

(2) $a\log(\eta_1(z)) + b\log(\eta_2(z))$
$\quad \to \log\left((\eta_1(z))^a\,(\eta_2(z))^b\right) + 2\,\pi\,i\,\kappa(z)$

(3) $e^{\eta_1(z)+a\log(\eta_2(z))} \to (\eta_2(z))^a\,e^{\eta_1(z)}$

(4) $\log(\eta_1(z)\,(e^{\eta_2(z)})^a)$
$\quad \to \log(\eta_1(z)) + a\,\eta_2(z) + 2\,\pi\,i\,\kappa(z)$

(5) $\mathrm{p}(\theta) \to \theta - \mathrm{rootof}(\mathrm{p}(x), x, \kappa(z))$

(6) $e^{\eta} - c \to \eta - \log(c) + 2\,\pi\,i\,\kappa(z)$

(7) $\log(\eta) - c \to \eta - e^c$

(8) $\mathrm{rootof}(\mathrm{p}(x), x, a) - c \to \mathrm{p}(c)$

where $a, b \in Z$ and $\kappa : C \to Z$ is an integer–valued piecewise constant function of z.

For rule (1), let θ_i and θ_j be exponentials. Try to find integers a, b, c such that $\mathrm{f}(z) = \mathrm{p}(\theta_i, \theta_j) = {\theta_j}^c\,\mathrm{q}({\theta_i}^a\,{\theta_j}^b)$ (consider $\mathrm{p}(0, y)$ and degrees) and simplify $\mathrm{f}(z)$ to $\mathrm{q}(\theta)$ where $\theta = \exp(a\,\eta_i(z) + b\,\eta_j(z))$.

For rule (2), let θ_i and θ_j be logarithms. Try to find a rational μ such that $\mathrm{p}(\theta_i, \theta_j) = \mathrm{q}(\theta_i + \mu\,\theta_j)$ (consider $\mathrm{p}(x, 1)$, $\mathrm{p}(1, y)$, and lcoeff's) and simplify $\mathrm{f}(z)$ to $\mathrm{q}((\theta + 2\,\pi\,i\,k)/a)$ where $\theta = \log\left(\pm(\eta_1(z))^a\,(\eta_2(z))^b\right)$, $\mu = b/a$, and $k \in Z$ is chosen according to a given root $z = r$ (given by our numerical algorithm).

Rules (1)-(2) are applied first. Rules (3)-(8) are applied with new factorization. This scheme will solve some equations.

10. Conclusion

The algorithm in this paper promises to return all solutions of an equation $\mathrm{f}(z) = 0$ over an interval I. The generalization to two dimensions does the same in two dimensions and we are confident the same method can be generalized to n dimensions.

Computer algebra is moving out of the era where symbolic expressions like $(a\,b)^c$, $\sqrt{x^2}$, Maple's RootOf, and Mathematica's InverseFunction have had poor or no relation to actual numbers into an era where proper semantics establishing a mapping from symbolic expressions to complex numbers is fully appreciated and enforced. Symbolic expressions are names for numbers. In this paper, we introduce a new kind of name, the Nonlin(...) object. Rather than a vague unknown solution of a nonlinear equation (cf. Mathematica's InverseFunction), a Nonlin(...) object represents a particular number which can be evaluated to as many decimal places as the user likes, just as it is possible to do with Pi or E. The future promises to bring more such objects into the realm of computer algebra: implicit functions, implicit solutions of differential equations, etc., and such objects will always have a definite meaning over the field of complex numbers. The mathematical impetus to study the algebra of such objects more carefully will likely emerge as these objects come on line, can be routinely calculated, and are equipped with sensible semantics.

References

[1] Alefeld, G., and Herzberger, J. (1983) *Introduction to Interval Computations*, Academic Press.

[2] Brent, R. P. (1971), "An Algorithm with Guaranteed Convergence for Finding a Zero of a Function", *The Computer Journal*, **14**, 422–425.

[3] Bundy, A., and Welham, B. (1981) "Using Meta-Level Inference for Selective Application of Multiple Rewrite Rules in Algebraic Manipulation", *Artificial Intelligence* **16(2)**.

[4] Collins, G. E. and Loos, R. (1983) "Real Zeros of Polynomials", *Computer Algebra Symbolic and Algebraic Computation, Second Edition*, (ed. B. Buchberger, G. E. Collins, and R. Loos) Springer-Verlag, 83–94.

[5] Conway, John B. (1978), *Functions of One Complex Variable, Second Edition*, Springer-Verlag.

[6] Hardy, G. H. (1910), Orders of infinity. *Cambridge Tracts in Mathematics* **12**.

[7] Hoenders, B. J. and Slump, C. H. (1992) "On the Determination of the Number and Multiplicity of Zeros of a Function", *Computing*, **47(3–4)**, 323–336.

[8] Ioakimidis, N. I. and Anastasselou, E. G. (1986) "On the Simultaneous Determination of Zeros of Analytic or Sectionally Analytic Functions", *Computing* **36(3)** 239–247.

[9] Kearfott, R. Baker (1991) "Decomposition of Arithmetic Expressions to Improve the Behaviour of Interval Arithmetic for Nonlinear Systems", *Computing* **47(2)**, 169–191.

[10] Nerinckx, D., and Haegemans, A. (1976), "A Comparison of Non-Linear Equation Solvers", *Journal of Computational and Applied Mathematics* **2**, 145–148.

[11] Ratschek, H., and Rokne, J. (1984) *Computer Methods for the Range of Functions*, Ellis Horwood Limited.

[12] Richardson, Daniel (1968) "Some Undecidable Problems Involving Elementary Functions of a Real Variable", *The Journal of Symbolic Logic* **33(4)**, 514–520.

[13] Rice, John R. (1993) *Numerical Methods, Software, and Analysis*, Academic Press, 323–410.

[14] Rudin, Walter (1976), *Principles of Mathematical Analysis*, McGraw–Hill.

[15] Saaty, Thomas L. (1981), *Modern Nonlinear Equations*, Dover.

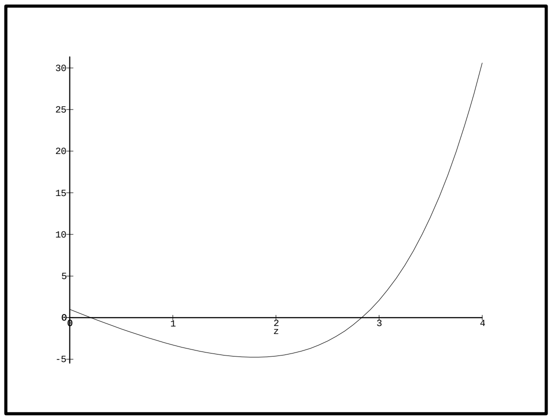

Figure 1.1 $e^z - 6\,z$

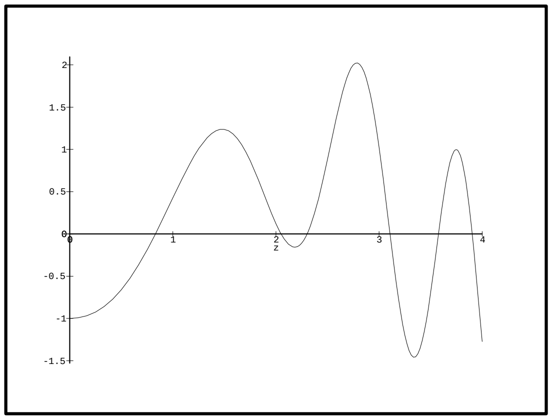

Figure 1.2 $\sin(z^2)\log(1+z) - \cos(\sqrt{2}\,z)$

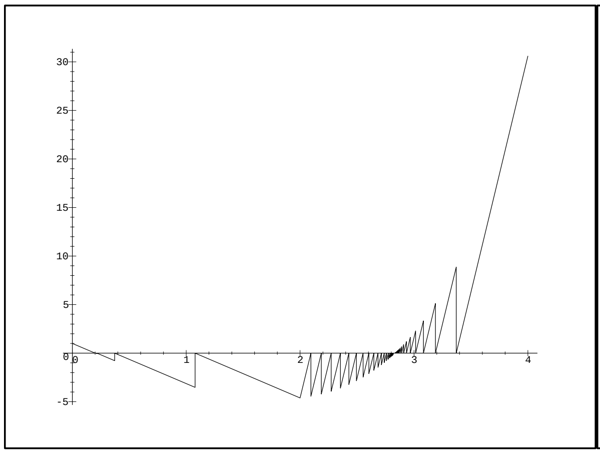

Figure 2.1 $e^z - 6\,z$

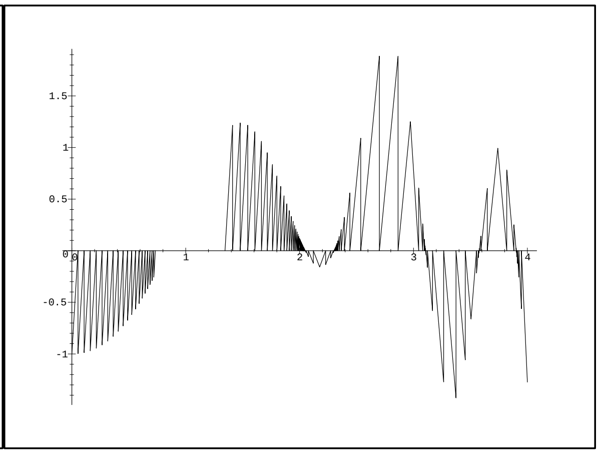

Figure 2.2 $\sin(z^2)\log(1+z) - \cos(\sqrt{2}\,z)$

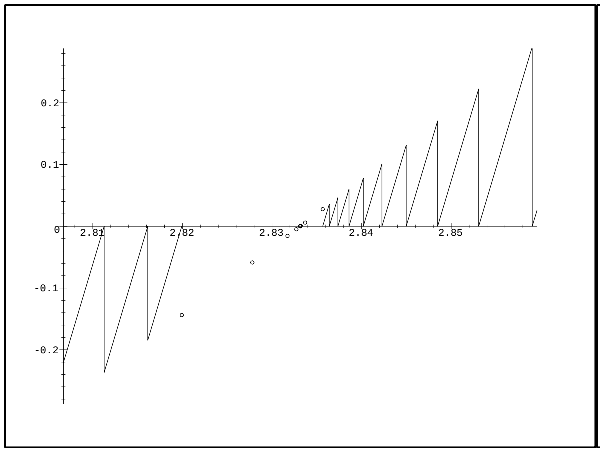

Figure 3.1 $e^z - 6\,z$

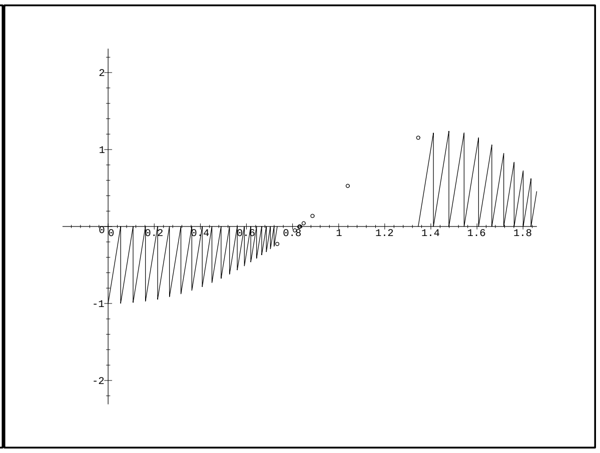

Figure 3.2 $\sin(z^2)\log(1+z) - \cos(\sqrt{2}\,z)$

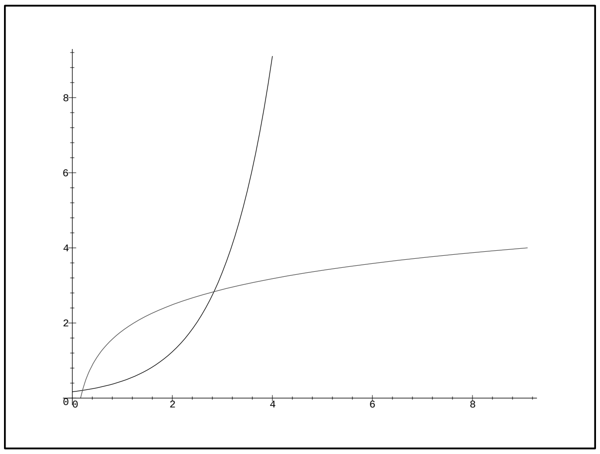

Figure 4.1 $e^x - 6\,y, \quad e^y - 6\,x$

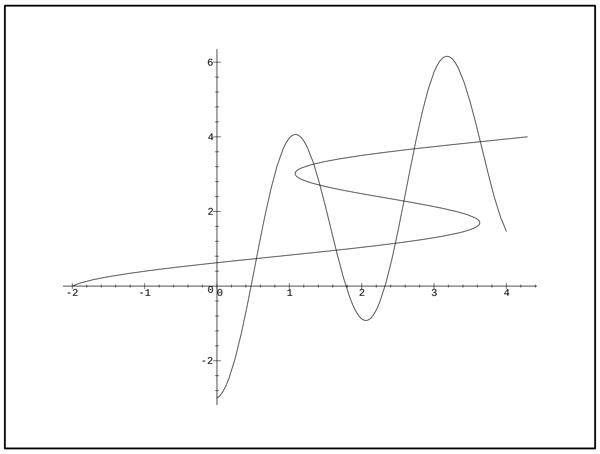

Figure 4.2 $x - 3\cos(3\,x) - y, \;\; x - y - 2\cos(2\,y)$

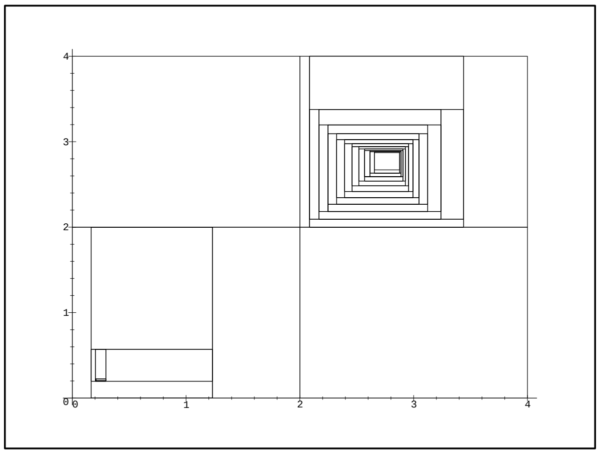

Figure 5.1 $e^x - 6\,y, \quad e^y - 6\,x$

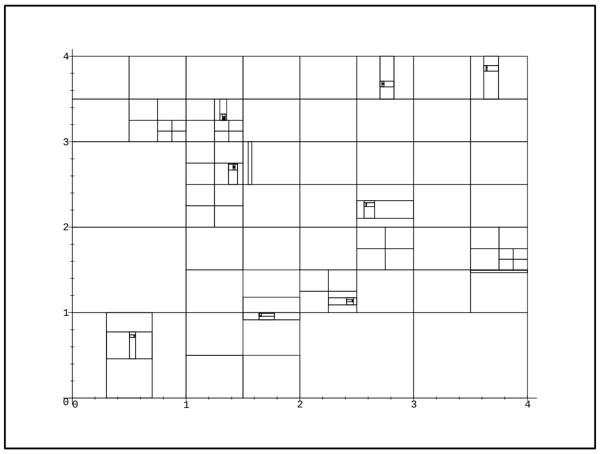

Figure 5.2 $x - 3\cos(3\,x) - y, \;\; x - y - 2\cos(2\,y)$

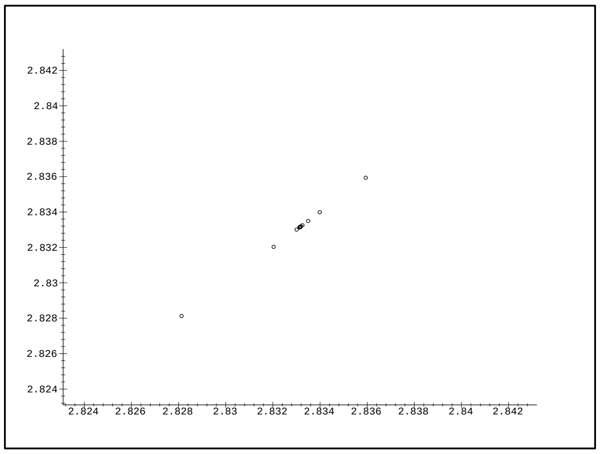

Figure 6.1 $e^x - 6\,y, \quad e^y - 6\,x$

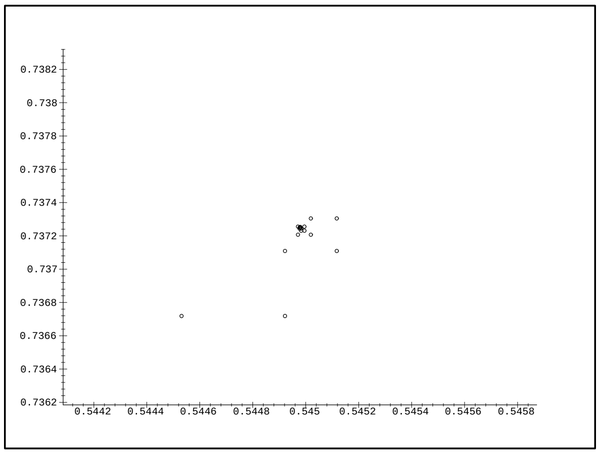

Figure 6.2 $x - 3\cos(3\,x) - y, \;\; x - y - 2\cos(2\,y)$