Appendix A — Code

Note from the Author

There are two algorithms for finding super-log. One is to compute the derivatives, and the other it so compute the matrix. Both methods require solving a system of equations, whether in matrix form or not. The matrix method takes about the same time as computing the derivatives symbolically, so their is not much performance gain. Since it takes so much time to compute the solutions for higher approximations, I made a separate function to compute them rather than include them in the functions themselves. This allows you to wait for the solutions once, and use them over and over rather quickly. Since the base is part of the coefficients, it must be given before solving. I have not made an implementation of the super-root, because it involves solving for any base symbolically, which in general takes much longer than finding it for a numerical base. Of all the bases, ***e*** requires the least time to prepare for, because all the equations being solved become rational. Some sample commands are given after the implementations, some of which demonstrate the numerical derivative functions included to reproduce some of the graphs given in this text. You might have to mess with h for higher derivatives, because the default $h = 0.0001$ and this produces almost random 4th and 5th derivatives. To fix this, let $h = 0.001$ or $h = 0.01$ and they become more smooth.

In Pseudo-code

let $s(x, z, n) = -1 + \text{sum}[k = 1..n]\ z\text{^}k/k!\ v\_k(x)$

solve for $v\_(k+1)(x)$ where $k = 0..(n-1)$ in:

$$(\boldsymbol{D}\_z\text{^}k[s(x, z, n)]@z{=}0 - \boldsymbol{D}\_z\text{^}k[s(x, x\text{^}z, n){-}1]@z{=}0) = 0$$

where $x > 1$, let $\text{slog}(x, z, n)$ = one of:

$s(x, x\text{^}z, n) - 1$	if $z <= 0$
$s(x, z, n)$	if $0 < z <= 1$
$s(x, \log\_x(z), n) + 1$	if $1 < z <= x$
$s(x, \log\_x(\log\_x(z)), n) + 2$	if $x < z <= x\text{^}x$
...	

let $\text{tetrate}(x, y, n) = z$ where $y = \text{slog}(x, z, n)$.

In Maple

```
      ## Usage:
      ##      env := superlog_prepare(n, x):
      ##      superlog(env, z);   -- gives n-th approx. of slog_x(z)
      ##      tetrate(env, y);    -- gives n-th approx. of x^^y
      ## Copyright 2005 Andrew Robbins

with(linalg):

superlog_prepare := proc(n::integer, x)
    [x, linsolve(matrix([seq([seq(
        k^j/k! - `if`(j = k, 1, 0)*log(x)^(-j),
        k = 1..n)], j = 0..(n - 1))]),
        [seq(`if`(k = 1, 1, 0), k = 1..n)])];
end proc;

superlog := proc(v, z) local slog_crit;
    if not (z::numeric) then return 'procname'(args); end if;
    slog_crit := proc(zc) -1 + sum(v[2][k]*zc^k/k!,
        k = 1..(vectdim(v[2]))); end proc;
    piecewise(z = -infinity, -2,
        z < 0, slog_crit(v[1]^z) - 1, z = 0, -1,
        0 < z and z < 1, slog_crit(z), z = 1, 0,
        z > 1, (proc() local a, i; a := z;
            for i from 0 while (evalf(a) > 1) do a := log[v[1]](a); end do;
            slog_crit(evalf(a)) + i; end proc)());
end proc;

tetrate := proc(v, y) local tet_crit;
    if not (y::numeric) then return 'procname'(args); end if;
    tet_crit := proc(yc) local slog_crit;
        slog_crit := proc(zc) -1 + sum(v[2][k]*zc^k/k!,
            k = 1..(vectdim(v[2]))); end proc;
        select((proc(a) evalb(Im(a) = 0 and 0 <= Re(a) and Re(a) <= 1)
            end proc), [solve(evalf(slog_crit(z)) = yc, z)])[1];
    end proc;
    piecewise(y = -2, -infinity,
        -2 < y and y < -1, log[v[1]](tet_crit(y+1)), y = -1, 0,
        -1 < y and y < 0, tet_crit(y), y = 0, 1,
        y > 0, (proc () local a, i; a := tet_crit(y - ceil(y));
            for i from 1 to ceil(y) do a := v[1]^a; end do;
            evalf(a); end proc)());
end proc;
```

```
## Gives the k-th numerical derivative of f(x) at x=c:

    ndiff := proc(f, x, c, k, h)
       if (k = 0) then subs(x = c, f);
       else (ndiff(f, x, c+h, k-1, h) - ndiff(f, x, c, k-1, h))/h;
       end if;
    end proc;

########################################
## A few commands to try for starters: ##
########################################

## Prepares the 10th approx. of slog_e(z):
##  the ':' is used here to suppress display

    env := superlog_prepare(10, exp(1)):

## Shows e^^0.5, and plots of e^^y:

    tetrate(env, 0.5);
    plot(tetrate(env, y), y = -2..2, view = -5..15);
    plot([tetrate(env, y), ndiff(tetrate(env, t), t, y, 1, 0.0001)],
        y = -2..2, view = -5..10);

## Shows slog_e(1.5), and plots of slog_e(z):

    superlog(env, 1.5);
    plot(superlog(env, z), z = -5..15, view = -2..2);
    plot([superlog(env, z), ndiff(superlog(env, t), t, z, 1, 0.0001)],
        z = -5..10, view = -2..2);

## Shows the half-exponential function:
##  f(x) such that f(f(x)) = exp(x)

    half_exp := proc(x) tetrate(env, superlog(env, x) + 1/2); end proc;
    plot([x, half_exp(x), half_exp(half_exp(x))],
        x = -2..4, view = -1..5);

## Plots of slog at different bases:

    plot([superlog(superlog_prepare(5, 2), z),
        superlog(superlog_prepare(5, 10), z)],
        z = -4..8, view = -2..3);
```

In Mathematica

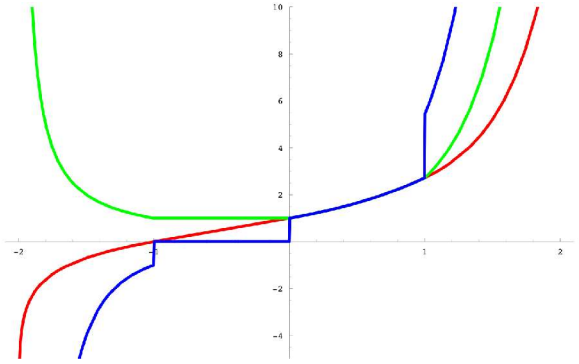
```
      (*
      ** Usage:
      **      env = SuperLogPrepare[n, x];
      **      SuperLog[env, z]    -- gives n-th approx. of slog_x(z)
      **      Tetrate[env, y]     -- gives n-th approx. of x^^y
      ** Copyright 2005 Andrew Robbins
      *)

SuperLogPrepare[n_Integer, x_] := {x, LinearSolve[Table[
    k^j/k! - If[j == k, Log[x]^-k, 0], {j, 0, n - 1},
    {k, 1, n}], Table[If[k == 1, 1, 0], {k, 1, n}]]}

SuperLog[v_, z_?NumericQ] := Block[{(*SlogCrit*)},
    SlogCrit[zc_] := -1 + Sum[v[[2, k]]*zc^k/k!, {k, 1, Length[v[[2]]]}];
    Which[z ≤ 0, SlogCrit[v[[1]]^z] - 1, 0 < z ≤ 1, SlogCrit[z], z > 1,
    Block[{i=-1}, SlogCrit[NestWhile[Log[v[[1]], #]&, z, (i++;#>1)&]]+i]]]

Tetrate[v_, y_?NumericQ] := Block[{(*SlogCrit, TetCrit*)},
    SlogCrit[zc_] := -1 + Sum[v[[2, k]]*zc^k/k!, {k, 1, Length[v[[2]]]}];
    TetCrit[yc_] := FindRoot[SlogCrit[z] == yc, {z, 1}][[1, 2]]; If[y > -1,
    Nest[Power[v[[1]], #]&, TetCrit[y - Ceiling[y]], Ceiling[y]],
    Nest[Log[v[[1]], #]&, TetCrit[y - Ceiling[y]], -Ceiling[y]]]]

(* Gives the k-th numerical derivative of f(x) at x=c: *)
ND[f_, x_, c_] := ND[f, x, c, 1]
ND[f_, x_, c_, k_] := ND[f, x, c, k, 0.0001]
ND[f_, x_, c_, 0, h_] := (f /. x -> c)
ND[f_, x_, c_, k_, h_] /; k != 0 := (
    ND[f, x, c+h, k-1, h] - ND[f, x, c, k-1, h])/h
```

```
(*************************************)
(* A few commands to try for starters: *)
(*************************************)

(* Prepares the 10th approx. of slog_e(z): *)
(*  the '=' is used here for immediate execution *)
(*  the ';' is used here to suppress display *)

    env = SuperLogPrepare[10, E];

(* Shows e^^0.5, and plots of e^^y: *)

    Tetrate[env, 0.5]
    Plot[Tetrate[env, y], {y, -2, 2}, PlotRange -> {-5, 15}]
    Plot[{Tetrate[env, y], ND[Tetrate[env, t], t, y]},
        {y, -2, 2}, PlotRange -> {-5, 10},
        PlotStyle -> {Hue[0], Hue[2/3]}]

(* Shows slog_e(1.5), and plots of slog_e(z): *)

    SuperLog[env, 1.5]
    Plot[SuperLog[env, z], {z, -5, 15}, PlotRange -> {-2, 2}]
    Plot[{SuperLog[env, z], ND[SuperLog[env, t], t, z]},
        {z, -5, 10}, PlotRange -> {-2, 2},
        PlotStyle -> {Hue[0], Hue[2/3]}]

(* Shows the half-exponential function: *)
(*  f(x) such that f(f(x)) = exp(x) *)

    HalfExp[x_] := Tetrate[env, SuperLog[env, x] + 1/2]
    Plot[{x, HalfExp[x], HalfExp[HalfExp[x]]},
        {x, -2, 4}, PlotRange -> {-1, 5}]

(* Plots 5th approx. of slog at different bases: *)

    Plot[{SuperLog[SuperLogPrepare[5, 2], z],
        SuperLog[SuperLogPrepare[5, 10], z]},
        {z, -4, 8}, PlotRange -> {-2, 3},
        PlotStyle -> {Hue[0], Hue[2/3]}]
```

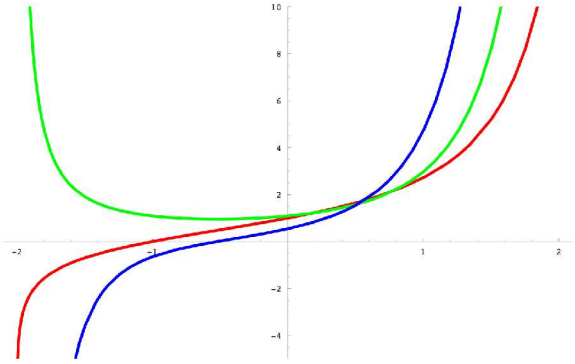
Appendix B — Graphs

Here are graphs of tetration and the super-logarithm, to illustrate the difference between the linear critical extensions, and the new definitions. The red line is the function itself, the green line is the first derivative, and the blue line is the second derivative:

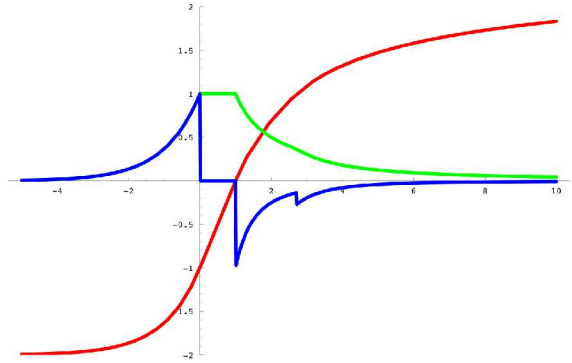
graph of ${}^{y}\boldsymbol{e} \approx \text{slog}_{\boldsymbol{e}}^{-1}(y)_1$



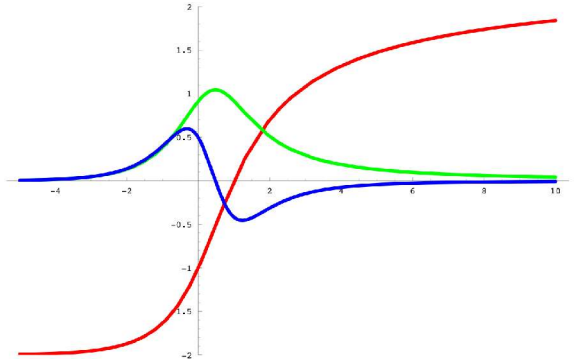
graph of ${}^{y}\boldsymbol{e} \approx \text{slog}_{\boldsymbol{e}}^{-1}(y)_{10}$



graph of $\text{slog}_{\boldsymbol{e}}(z)_1$



graph of $\text{slog}_{\boldsymbol{e}}(z)_{10}$



A graph of $\exp^{y}(x) \approx \text{slog}_{\boldsymbol{e}}^{-1}(\text{slog}_{\boldsymbol{e}}(x)_{10} + y)_{10}$, for $y = \{1, ½, 0, -½, -1\}$: