Research and Communications in Mathematics and Mathematical Sciences
Vol. 12, Issue 1, 2020, Pages 1-15
ISSN 2319-6939
Published Online on January 28, 2020

http://jyotiacademicpress.org

ON THE ENUMERATION OF THE ROOTS OF ARBITRARY SEPARABLE EQUATIONS USING HW HYPER-LAMBERT MAPS

IOANNIS GALIDAKIS

Department of Mathematics
Agricultural University of Athens
Greece
e-mail: jgal@aua.gr

Abstract

In this article we use the HW maps to solve arbitrary equations $f = 0$, by providing an effective enumeration of the roots of f, as these project on and at the branches of the HW maps. This is just an enumeration of the projection points (roots) of a pin-line on the Riemann surface of f through HW.

1. Introduction

The HW maps have been used to determine the attractors of the infinite exponential whenever it falls into a p-cycle in [3] and in [2] to solve certain transcendental equations such as Kepler's equation. They have also been used in [4] to solve in closed form the generalized Abel differential equation. Here we display a simple algebraic scheme which can be used to utilize the solution of arbitrary equations using the HW maps, by providing an effective enumeration of all the roots of arbitrary

2020 Mathematics Subject Classification: 3002, 30A99, 30E15.
Keywords and phrases: polynomial, rational, inverse function, Lambert root analytic function, Riemann surface.
Communicated by Suayip Yuzbasi.
Received November 22, 2019; January 15, 2020

equations $f = 0$, using the branches of the maps HW. Imagine an arbitrary multivalued f, for which we force $f(x) = 0$. We line-pin the entire Riemann surface of f from top to bottom starting at the complex origin. The local projection pin points z_i will be exactly the roots of $f = 0$. Because the branches of HW can be enumerated starting at the origin, all the roots z_i of $f = 0$ can therefore be enumerated and referenced by approximating just an ϵ pin through the origin.

2. Definitions

Suppose $f_n(z)$ are non-vanishing identically complex functions, with $n \leq n_0 \in \mathbb{N}$. We define $F_n(z) : \mathbb{N} \times \mathbb{C} \to \mathbb{C}$ as:

Definition 2.1.

$$F_n(z) = \begin{cases} 1, & \text{if } n = 1, \\ e^{f_{n-1}(z)F_{n-1}(z)}, & \text{if } n > 1. \end{cases}$$

Definition 2.2. $G(f_1, f_2, \ldots, f_n; z) = z \cdot F_{n+1}(z)$.

If $n = 0$, then $G(z) = z$. If $n = 1$, then $G(f_1; z) = ze^{f_1(z)}$. If $n = 2$, then $G(f_2, f_1, ; z) = ze^{f_2 e^{f_1(z)}}$. When we write about the HW, we can use the terminology $G(\ldots; z)$, meaning that the corresponding function includes meaningful terms-parameters. The order of the functions is immaterial and we can re-order them to get to the function of interest here, which is the inverse of $G(\ldots; z)$, denoted by,

$$\mathrm{HW}(f_1, f_2, \ldots, f_n; y). \tag{1}$$

In other words G and HW satisfy the functional relation:

$$G(\ldots; \mathrm{HW}(\ldots; y)) = y \tag{2}$$

by supposing always that the list of parameters is identical on both sides. These maps have been called generalized hyper-Lambert HW functions

and in general they are multivalued. We note that when $n = 1$, $\mathrm{HW}(y)$ satisfies a more general form which comes from the Lambert function W, i.e., $ze^{f_1(z)} = y$. The Lambert function satisfies $ze^z = y$. The existence of all the HW is guaranteed in all cases by the Lagrange Inversion Theorem (see ([5], p. 201-202)).

3. An Indexing Scheme for the HW Maps

3.1. An algebraic scheme

For the complex maps log and W, their indexing scheme is the simplest possible, that is $\log(k, z)$ and $W(k, z)$, $k \in \mathbb{Z}$. There exists an indexing scheme which indexes identically the mappings HW but it is not integral. Dubinov and Galidakis in [1] solve Kepler's equation, using the following algebraic inversion:

$$E - \epsilon \cdot \sin(E) = M \Rightarrow$$

$$E\left(1 - \epsilon \frac{\sin(E)}{E}\right) = M \Rightarrow$$

$$E \cdot e^{\log(1-\epsilon \cdot \operatorname{sinc} E)} = M \Rightarrow$$

$$E = \mathrm{HW}[\log(1 - \epsilon \cdot \operatorname{sinc}(x)); M]. \tag{3}$$

The inversion above can be generalized producing a removable pole at z_0 of multiplicity n. Setting $w = (z - z_0)^n$, with z_0 such that $f(z_0) = y$, we have:

$$f(z) = y \Rightarrow$$

$$(z - z_0)^n \cdot \frac{f(z)}{(z - z_0)^n} = y \Rightarrow$$

$$w \cdot e^{\log(\frac{f(z)}{w})} = y \Rightarrow$$

$$w = \mathrm{HW}\left[\log\left(\frac{f(z)}{w}\right);\ y\right] \Rightarrow$$

$$(z - z_0)^n = \mathrm{HW}\left[\log\left(\frac{f(z)}{(z - z_0)^n}\right);\ y\right] \Rightarrow$$

$$z = \mathrm{HW}\left[\log\left(\frac{f(z)}{(z - z_0)^n}\right);\ y\right]^{\frac{1}{n}} + z_0. \tag{4}$$

The scheme above gives an index into the set of the HW functions, in the form of a functional parameter as $\log\left(\frac{f(z)}{(z - z_0)^n}\right)$. Now, if we know $f(z)$, this scheme can give identities which must hold identifying this way the corresponding function.

We can now list how the most important categories of complex functions are solved based on this index.

3.2. Polynomial functions

Suppose then that $f(z) = \prod_{k=1}^{N} (z - z_k)^{n_k}$. Keeping k fixed and setting $w = (z - z_k)^{n_k}$, we have

$$w = \mathrm{HW}\left[\log\left(\frac{f(z)}{w}\right);\ y\right] \Rightarrow$$

$$z = \mathrm{HW}\left[\log\left(\frac{f(z)}{(z - z_k)^{n_k}}\right);\ y\right]^{\frac{1}{n_k}} + z_k \Rightarrow$$

$$z = \mathrm{HW}[\log(f(z)) - n_k \log(z - z_k);\ y]^{\frac{1}{n_k}} + z_k. \tag{5}$$

Theorem 3.1. *If* $f(z) = \prod_{k=1}^{N}(z - z_k)^{n_k}$ *is a complex polynomial function, then the inverse of* $f(z)$ *relative to* y *is given by the function* HW *and the last equation of* (5), *whose Riemann surface has at most* $m = \sum_{k=1}^{N} n_k$ *branches, indexed by* m, *with* $k \in \mathbb{N}$.

Proof. The last expression of (5) is true for any $k \in \{1, 2, \ldots, N\}$, therefore the multiplicity is at least m because for each k the multiplicity is at least n_k and each n_k may give different branches. This means that the expression can indexfully all the branches of the corresponding HW using only an integral index k. □

Theorem 3.2. *If* $f(z)$ *is a complex polynomial function, the roots of* $f(z) = y$ *are given directly by a suitable* HW *function.*

Proof. Using Equation (1) of Definition 2.2, follows that for each HW, $\mathrm{HW}(\ldots; 0) = 0$, therefore calculating the corresponding HW of the last equations in (5) at $y = 0$, forces $z = z_k$ and these are the roots of $f(z) = y$. Therefore, we can extract all the roots of equation $f(z) = y$, manually. The first root, suppose z_1, is extracted as,

$$z_1 = \mathrm{HW}\left[\log\left(\frac{f(z)}{z}\right);\ y\right],$$

$$g_1(z) = \frac{f(z) - y}{z - z_1}.$$

Having the root z_1, the rest of the roots can be extracted recursively for $1 \leq k \leq N-1$ as,

$$z_{k+1} = \lim_{\epsilon \to 0^+} \mathrm{HW}\left[\log\left(\frac{g_k(z)}{z} \right); \epsilon \right],$$

$$g_{k+1}(z) = \frac{g_k(z)}{z - z_{k+1}},$$

and the Theorem follows. □

3.3. Rational functions

We suppose that $f(z) = P(z)/Q(z)$, with $P(z), Q(z)$ polynomial functions. We have similar results here.

Theorem 3.3. *If* $f(z) = P(z)/Q(z)$ *is a complex rational function such that* $N = \max\{\deg(P), \deg(Q)\}$, *then the inverse of* $f(z)$ *relative to* y *is given by*:

$$z = \mathrm{HW}\left[\log\left(\frac{P(z) - y \cdot Q(z)}{(z - z_k)^{n_k}} \right); y \right]^{\frac{1}{n_k}} + z_k,$$

whose Riemann surface has at most $m = \sum_{k=1}^{N} n_k$ *branches, indexed by* m, *with* $k \in \mathbb{N}$.

Proof. If $F(z) = P(z) - y \cdot Q(z)$, then $F(z)$ is a polynomial of degree N, in which case the Theorem follows similarly, with $f(z)$ replaced by $F(z)$. □

Theorem 3.4. *If* $f(z)$ *is a complex rational function, the roots of* $f(z) = y$ *can be given by a suitable* HW *function.*

Proof. Similarly, if $F(z) = P(z) - y \cdot Q(z)$, then $F(z)$ is polynomial map of degree N, therefore we can extract its roots as:

$$z_1 = \lim_{\epsilon \to 0^+} \mathrm{HW}\left[\log\left(\frac{F(z)}{z} \right); \epsilon \right],$$

$$g_1(z) = \frac{F(z)}{z - z_1}.$$

Having z_1, the rest of the roots can be extracted recursively for $1 \leq k \leq N - 1$ as,

$$z_{k+1} = \lim_{\epsilon \to 0^+} \mathrm{HW}\left[\log\left(\frac{g_k(z)}{z} \right); \epsilon \right],$$

$$g_{k+1}(z) = \frac{g_k(z)}{z - z_{k+1}},$$

and the Theorem follows. □

We observe that when $Q(z) = 1$, the case of a polynomial function arises.

3.4. Analytic functions

For an analytic function $f(z) = \sum_{n=0}^{\infty} \alpha_n \cdot (z - z_0)^n$ in some region $D \subseteq \mathbb{C}$, with $z_0 \in D$, we have similar results.

Theorem 3.5. *If* $f(z) = \sum_{n=0}^{\infty} \alpha_n \cdot (z - z_0)^n$ *is a complex analytic function, then the inverse of* $f(z)$ *relative to y is given by a suitable* HW *function*:

$$z = \mathrm{HW}\left[\log\left(\frac{f(z)}{(z - z_k)^{n_k}} \right); y \right]^{\frac{1}{n_k}} + z_k,$$

whose Riemann surface has infinitely many branches given by $n \in \mathbb{N}$.

Proof. Suppose $T_N(z) = \sum_{n=0}^{N} \alpha_n \cdot (z - z_0)^n$, is the corresponding Taylor polynomial of degree N. Then $T_N(z)$ is obviously a polynomial function, therefore the inverse of $T_N(z)$ relative to y is given again by Theorem 3.1.

$$z = \mathrm{HW}\left[\log\left(\frac{T_N(z)}{(z - z_k)^{n_k}} \right);\ y \right]^{\frac{1}{n_k}} + z_k, \tag{6}$$

$T_N(z) \to f(z)$ uniformly in compact subsets and the HW are analytic ([3]), therefore (6) implies that the inverse is given by:

$$z = \lim_{N\to\infty} \mathrm{HW}\left[\log\left(\frac{T_N(z)}{(z - z_k)^{n_k}} \right);\ y \right]^{\frac{1}{n_k}} + z_k \Rightarrow$$

$$z = \mathrm{HW}\left[\log\left(\frac{\lim_{N\to\infty} T_N(z)}{(z - z_k)^{n_k}} \right);\ y \right]^{\frac{1}{n_k}} + z_k \Rightarrow$$

$$z = \mathrm{HW}\left[\log\left(\frac{f(z)}{(z - z_k)^{n_k}} \right);\ y \right]^{\frac{1}{n_k}} + z_k, \tag{7}$$

and the Theorem follows. □

We observe that in this case the inverse function has infinitely many branches, since N is not bounded.

Theorem 3.6. *If $f(z)$ is a complex analytic function, the roots of $f(z) = y$ are given again by a* HW *function.*

Proof. We can extract the roots as:

$$z_1 = \mathrm{HW}\left[\log\left(\frac{f(z)}{z}\right);\ y\right],$$

$$g_1(z) = \frac{f(z) - y}{z - z_1}. \tag{8}$$

The rest of the roots can be again extracted recursively for $1 \le k$ as,

$$z_{k+1} = \lim_{\epsilon \to 0^+} \mathrm{HW}\left[\log\left(\frac{g_k(z)}{z}\right);\ \epsilon\right],$$

$$g_{k+1}(z) = \frac{g_k(z)}{z - z_{k+1}}, \tag{9}$$

and the Theorem follows. □

4. HW Functional Index

An open problem set in ([2], p. 1114-1115) is whether there is a way to effectively index the numbering of the branches of the HW functions. With the following theorem we show that the answer is affirmative.

Theorem 4.1. *If* $f(z)$ *is a complex function and* $z_k \in \mathbb{C}$, $k \in \mathbb{N}$, *such that* $f(z_k) = y$ *and suppose* $g_k(z)$ *follows as in Equations* (8)-(9). *Then, if* HW *is the inverse of* $f(z)$ *relative to y, the following scheme covers all the branches of this inverse of* $f(z)$:

$$z_{k+1} = \begin{cases} \mathrm{HW}\left[(0,\ )\log\left(\frac{f(z)}{z}\right);\ y\right], \text{ if } k = 0, \\ \lim\limits_{\epsilon \to 0^+} \mathrm{HW}\left[k,\ \log\left(\frac{g_k(z)}{z}\right);\ \epsilon\right], \text{ if } k > 0. \end{cases}$$

Proof. The proof follows from (3) along with Theorems 3.1, 3.3, and 3.5. Note that for a specific analytic f expanded around z_k, we define $F(z) = \log\left(\frac{f(z)}{(z - z_k)^{n_k}}\right)$. The map F creates a Laurent series with residue

$\exp(a_{n_k})$, which is gotten from the HW through the Residue Theorem of Cauchy for *f*, with winding number a_{n_k} around z_k. Consequently, the repeated application of F(via g_k) for $z = z_k$, extracts recursively all the roots z_k of the inverse and as such it can be used as an index for the corresponding Riemann surface. □

5. Conclusion

The HW maps can solve any equation $f = 0$, provided it can be brought into a separable form with all *z*'s on the left and one *w* on the right. Further, the enumeration of the roots is origin consistent relative to the force of *f*.

References

[1] A. E. Dubinov and I. N. Galidakis, Explicit solution of the Kepler equation, Physics of Particles and Nuclei Letters 4(3) (2007), 213-216.

DOI: https://doi.org/10.1134/S1547477107030028

[2] I. N. Galidakis, On some applications of the generalized hyper-Lambert functions, Complex Variables and Elliptic Equations 52(12) (2007), 1101-1119.

DOI: https://doi.org/10.1080/17476930701589563

[3] I. N. Galidakis, On solving the *p*-th complex auxiliary equation $f^{(p)}(z) = z$, Complex Variables, Theory and Application: An International Journal 50(13) (2005), 977-997.

DOI: https://doi.org/10.1080/02781070500156827

[4] P. Nastou, Y. Stamatiou and A. Tsiakalos, Solving a class of odes arising in the analysis of a computer security process using generalized hyper-Lambert functions, International Journal of Applied Mathematics and Computation (IJAMC) 4(3) (2012), 67-76.

[5] S. Saks and A. Sygmund, Analytic Functions, Hafner Publishing Company, New York, 1952.

■

Appendix: Programming with the HW Maps

Code for the HW maps is given below. Arguments are HW (functional index, y, n)

```
restart;

Digits:=40;

HW := proc ()

local y, n, c, s, p, sol, i, aprx, dy, dist, r, newr, oldr,
fun, dfun, eps;

if nargs < 2 then ERROR("At least two arguments required") end
if;

n := args[-1]; y := args[-2]; c := [args[1 .. -3]];

if y = 0 then 0 else dist := infinity;

eps:=1e-10; fun := 1; for i from 1 to nargs-2 do fun :=
exp(c[-i]*fun) end do;

fun := z*fun-y; dfun := diff(fun, z);

s := series(fun, z, n); p := convert(s, polynom);

sol := {fsolve(p = 0, z, complex)};

for i from 1 to nops(sol) do aprx := evalf(subs(z = op(i,
sol), fun));

dy := evalf(abs(aprx)); if dy <= dist then r := op(i, sol);

dist := dy end if end do; oldr := r;

newr := r-evalf(subs(z = r, fun)/subs(z = r, dfun));

for i from 1 to 1000 while abs((oldr-newr)/oldr)>eps do oldr
:= newr;

newr := newr-evalf(subs(z = newr, fun)/subs(z = newr, dfun))
end do;

newr end if end proc:
```

Example 1. Using the program with five decimal digits accuracy to solve the equation $(z-2)(z-3)(z-5)=2$,

```
y:=2;
f:=z->(z-2)*(z-3) * (z-5);
z1:=HW(log(f(z)/z),y,10);
g1:=z->(f(z)-y)/(z-z1);
z2:=HW(log(g1(z)/z),1e-20,10);
g2:=z->g1(z)/(z-z2);
z3:=HW(log(g2(z)/z),1e-20,10);
```

gives:

$$z_1 \simeq 2.36523 - 0.69160i$$

$$z_2 \simeq 2.36523 + 0.69160i$$

$$z_3 \simeq 5.26953$$

Using the program for an approximate solution with Maple,

```
solve(f(z)=y,z);
evalf(%);
```

gives:

5.26953, 2.36523 + 0.69160i, 2.36523 – 0.69160i.

Example 2. Using the program to five digits of accuracy to solve the equation $(z-2)/(z-3)/(z-5)/(z-1)=2$,

```
y:=2;
f:=z->(z-2)*(z-3)/(z-5)/(z-1);
P:=unapply(numer(f(z)),z);
Q:=unapply(denom(f(z)),z);
```

```
F:=unapply(P(z)-y*Q(z),z);

z1:=HW(log(F(z)/z),1e-10,10);

g1:=z->F(z)/(z-z1);

z2:=HW(log(g1(z)/z),1e-10,10);
```

gives:

$$z_1 \simeq 6.37228$$

$$z_2 \simeq 0.62771$$

Using Maple approximation code,

```
solve(F(z)=y,z);

evalf(%);
```

gives:

$$6.37228,\ 0.62771.$$

Example 3. Using the program to five decimals of accuracy to solve the equation $\sin(z) = 1/2$,

```
y:=1/2;

f:=z->sin(z);

z1:=HW(log(f(z)/z),y,10);

g1:=z->(f(z)-y)/(z-z1);

z2:=HW(log(g1(z)/z),1e-20,10);

g2:=z->g1(z)/(z-z2);

z3:=HW(log(g2(z)/z),1e-20,10);
```

gives:

$$z_1 \simeq 0.52359$$

$$z_2 \simeq 2.61799$$

$$z_3 \simeq -3.66519$$

$$z_4 \simeq \ldots$$

The results are approximations of the numbers $\pi/6,\ 5\pi/6,\ -7\pi/6,\ \ldots,$ which are the roots of $\sin(z) = 1/2.$ Many more complex equations can be solved here, provided they are separable and the terms are analytic, like.

Example 4. Using the code with five decimal accuracy to solve the equation $\sin(z) + \exp(\sin(z))/\sqrt{1+\tanh(z)} = 1/2,$

```
y=1/2;
f:=sin(z)+exp(sin(z))/sqrt(1+tanh(z));
z1:=HW(log(f(z)/z), y, 10);
g1:= z->(f(z)-y)/(z-z1);
z2:=HW(log(f1(z)/z),1e-20,10);
g2:=z->g1(z)/(z-z2);
z3:=HW(log(g2(z)/z),1e-20,10);
g3:=z->g2(z)/(z-z3);
z4:=HW(log(g3(z)/z), 0.1e-19, 10);
```

gives:

$$z_1 \simeq -0.37435$$

$$z_2 \simeq -1.71811$$

$$z_3 \simeq 3.26659$$

$$z_4 \simeq 6.15846$$

While using Maple approximation code,

```
solve(f(z)=y,z);
```

gives an open answer in terms of "RootOf", i.e., it cannot relay the roots directly.

Example 5. Using the code with five decimal accuracy to solve the equation $z^3 - 4z^2 + 5z = 2$,

```
y:=2;
f:=z->z^3-4*z^2+5*z;
z1:=HW(log(f(z)/z),y,10);
g1:=z->(f(z)-y)/(z-z1);
z2:=HW(log(f1(z)/z),1e-20,10);
g2:=z->g1(z)/(z-z2);
z3:=HW(log(g2(z)/z),1e-20,10);
```

gives:

$$z_1 \simeq 2$$

$$z_2 \simeq 1.00005$$

$$z_3 \simeq 1.00000$$

The description calculates correctly roots with multiplicity greater than 1.

The example is $f(z) - y = (z-1)^2(z-2)$, therefore the multiplicity of the root 1 is indeed 2.