A Fast and Accurate Universal Kepler Solver without Stumpff Series

Jack Wisdom,[1⋆] and David M. Hernandez[1]
[1] *Massachusetts Institute of Technology, Cambridge, MA, 02139, USA*

Accepted XXX. Received YYY; in original form ZZZ

ABSTRACT
We derive and present a fast and accurate solution of the initial value problem for Keplerian motion in universal variables that does not use the Stumpff series. We find that it performs better than methods based on the Stumpff series.

Key words: celestial mechanics – methods: numerical

1 INTRODUCTION

Wisdom & Holman (1991) introduced a symplectic mapping method for the rapid simulation of the n-planet problem (n planets plus a massive central body). The method splits the Hamiltonian for the n-planet problem into Kepler Hamiltonians and an interaction Hamiltonian, each of which may be efficiently solved. The evolution of the n-planet problem is obtained by interleaving the elementary pieces. The rapid and accurate solution of the Kepler initial value problem is an essential part of the method. The Wisdom-Holman method and its variations has been widely adopted for solar system dynamics investigations.

The method of Wisdom & Holman (1991) evolved from the mapping method of Wisdom (1982); it relies on the averaging principle to introduce Dirac delta functions into the Hamiltonian. An alternate approach, which leads to the same algorithm, is that of symplectic integration, which uses the algebra of Lie series to approximate the local evolution to some order in the stepsize by interleaving the evolution of the pieces. An advantage of the mapping approach is that the stability of the method can be analyzed in terms of the overlap of "stepsize resonances," which can be read off the delta function Hamiltonian (Wisdom & Holman 1992). Another advantage of the mapping approach is that perturbation theory can be used to improve the method by eliminating the high-frequency terms introduced by the delta functions, leading to the "symplectic corrector" (Wisdom, Holman & Touma 1996). An advantage of the symplectic integration approach is its algebraic simplicity.

Jacobi coordinates were used in Wisdom & Holman (1991) to eliminate the center of mass freedom and to effect the separation of the Hamiltonian into Keplerian and interaction parts. Touma & Wisdom (1993, 1994) used a different splitting, making use of the canonical heliocentric coordinates. Levison & Duncan (1994) used the Wisdom-Holman method in Jacobi coordinates, patched together with a non-symplectic method for handling close encounters among the bodies. They distributed their program, called `RMVS3`, in their `SWIFT` package. Duncan, Levison & Lee (1998) used the canonical heliocentric variables with a slightly different splitting; they call their method the "democratic heliocentric" method. They handle close encounters by recursively subdividing the step in a symplectic manner. Their program is distributed as `SYMBA`. Chambers (1999) introduced another method based on the democratic heliocentric splitting, with a symplectic transition to an ordinary numerical integration method (the Bulirsch-Stoer method) for close encounters. His program is distributed as `MERCURY`. Levison & Duncan (2000) noticed that methods based on the democratic heliocentric splitting are unstable for large eccentricities, and modified their `SYMBA` program to numerically integrate "close encounters with the Sun." Their modified program is called "modified `SYMBA`." An essential element of all of these variations on the Wisdom-Holman method is that one must solve the Kepler initial value problem: given the position and velocity at one time, the task is to find the position and velocity at a different time (displaced by a timestep that can be either positive or negative).

A recent contribution using the Lie series approach is Hernandez & Bertschinger (2015). They developed a symplectic integrator for the collisional n-body problem. This method also relies on the rapid and accurate solution of the Kepler initial value problem.

We have found that the solution of the initial value problem may be efficiently and accurately carried out in universal variables. The resulting formulation and program work for all orbits, whether they are elliptic, parabolic, or hyperbolic. The traditional presentation of univeral variables (e.g. Danby (1992)) makes use of Stumpff functions and calculates them using their series representation. Argument four-folding is used to bring the function argument into an interval near zero so that the series converge rapidly enough. However, all of the functions can be represented in

⋆ E-mail: wisdom@mit.edu (JW)

terms of ordinary trigonometric functions. But, a straightforward computation using this representation has unpleasantly large error, so the series and argument four-folding appear to be necessary. We have been able to circumvent this problem by reexpressing the functions in a numerically well defined way. The builtin trigonometric functions are fast and accurate, and our solution of the initial value problem compares favorably with solutions based on the Stumpff series.

2 KEPLER PROBLEM

The Hamiltonian for the Kepler problem is:

$$H(t, \mathbf{x}, \mathbf{p}) = \frac{p^2}{2m} - \frac{\mu}{r}, \tag{1}$$

where p is the magnitude of $\mathbf{p}$ and r is the magnitude of $\mathbf{x}$. The constants μ and m depend on the context in which the Kepler problem is found. For example, if $\mathbf{x} = \mathbf{x}_2 - \mathbf{x}_1$ is the relative coordinate in the two-body problem, then we would be led to take m to be the reduced mass $(1/m_1 + 1/m_2)^{-1}$ and $\mu = Gm_1m_2$. In the n-body problem, the constants may be different, depending on the formulation (e.g. Wisdom & Holman (1991)).

Hamilton's equations may be written as a second order system:

$$\ddot{\mathbf{x}} = -\frac{k\mathbf{x}}{r^3}, \tag{2}$$

where the "Kepler constant" $k = \mu/m$. The evolution depends on the constants only through this combination. We use the dot notation for derivative with respect to time.

3 DERIVATION OF THE SOLUTION

A basic reference for the solution of the Kepler initial value problem is Danby (1992). We refer the reader to that presentation of universal variables and the Stumpff series for background on the series approach. We follow some aspects of that presentation here. Our derivation is self-contained.

We introduce a new independent variable s satisfying

$$\frac{ds}{dt} = \frac{1}{r}. \tag{3}$$

The equation of motion, Eq. (2), becomes

$$\mathbf{x}'' - \frac{r'}{r}\mathbf{x}' + \frac{k}{r}\mathbf{x} = 0, \tag{4}$$

where prime indicates differentiation with respect to s.

The Hamiltonian is conserved, since there is no explicit time dependence. It is conventional to write the conserved quantity as

$$\beta = \frac{k}{a} = \frac{2k}{r} - \dot{\mathbf{x}} \cdot \dot{\mathbf{x}}. \tag{5}$$

The constant β is positive for elliptic motion, for which a is the semimajor axis, β is negative for hyperbolic motion, and β is zero for parabolic motion.

Expressing β in terms of derivatives with respect to s yields:

$$\beta = \frac{2k}{r} - \frac{1}{r^2}\mathbf{x}' \cdot \mathbf{x}'. \tag{6}$$

Differentiating this expression with respect to s and using the derivative of the equation of motion, Eq. (4), yields:

$$\mathbf{x}''' + \beta \mathbf{x}' = 0. \tag{7}$$

Next we introduce the f and g functions:

$$\begin{aligned} \mathbf{x} &= f\mathbf{x}_0 + g\mathbf{v}_0, \\ \mathbf{v} &= \dot{f}\mathbf{x}_0 + \dot{g}\mathbf{v}_0, \end{aligned} \tag{8}$$

where $\mathbf{x}_0$ is the initial position, and $\mathbf{v}_0$ is the initial time rate of change of position (the initial velocity). Substituting this into Eq. (7), we find

$$\begin{aligned} f''' + \beta f' &= 0 \\ g''' + \beta g' &= 0, \end{aligned} \tag{9}$$

using the independence of initial position and velocity. The equations are satisfied by an offset simple harmonic oscillation in s.

To determine the initial values for the derivatives, we start with Eq.(2), to find

$$\begin{aligned} \ddot{f} + \frac{k}{r^3}f &= 0 \\ \ddot{g} + \frac{k}{r^3}g &= 0. \end{aligned} \tag{10}$$

From Eq. (8), the initial values satisfy $f_0 = \dot{g}_0 = 1$ and $\dot{f}_0 = g_0 = 0$. Then

$$\begin{aligned} f_0' &= r_0\dot{f}_0 = 0 \\ g_0' &= r_0\dot{g}_0 = r_0 \\ f_0'' &= r_0^2\ddot{f}_0 + r_0\dot{r}_0\dot{f}_0 = -\frac{k}{r_0} \\ g_0'' &= r_0^2\ddot{f}_0 + r_0\dot{r}_0\dot{f}_0 = r_0\dot{r}_0. \end{aligned} \tag{11}$$

It is convenient to define solutions of Eq. (9) in terms of functions $G_i^\beta(s)$ satisfying

$$G_i^\beta(s) = \frac{d}{ds}G_{i+1}^\beta(s). \tag{12}$$

We start the ladder with

$$G_0^\beta(s) = \cos(\sqrt{\beta}s), \tag{13}$$

for $\beta > 0$, and

$$G_0^\beta(s) = \cosh(\sqrt{-\beta}s), \tag{14}$$

for $\beta < 0$. We can take

$$G_1^\beta(s) = \frac{\sin(\sqrt{\beta}s)}{\sqrt{\beta}}, \tag{15}$$

for $\beta > 0$, and

$$G_1^\beta(s) = \frac{\sinh(\sqrt{-\beta}s)}{\sqrt{-\beta}}, \tag{16}$$

for $\beta < 0$. We can then take

$$G_2^\beta(s) = (1 - \cos(\sqrt{\beta}s))/\beta, \tag{17}$$

for $\beta > 0$, and

$$G_2^\beta(s) = (1 - \cosh(\sqrt{-\beta}s))/\beta, \tag{18}$$

for $\beta < 0$. Onward,

$$\begin{aligned} G_3^\beta(s) &= (s - \sin(\sqrt{\beta}s)/\sqrt{\beta})/\beta \\ &= (s - G_1^\beta(s))/\beta. \end{aligned} \tag{19}$$

for $\beta > 0$, and

$$\begin{aligned} G_3^\beta(s) &= (s - \sinh(\sqrt{-\beta})/\sqrt{-\beta})/\beta \\ &= (s - G_1^\beta(s))/\beta, \end{aligned} \tag{20}$$

for $\beta < 0$. We could go on, but this is all we need here. We have chosen the constants so that $G_i^\beta(0) = 0$ for $i > 0$. Notice that for $i < 3$ these functions satisfy

$$(G_i^\beta(s))''' + \beta(G_i^\beta(s))' = 0. \tag{21}$$

Now we can use these functions to find solutions for f and g. Let

$$f(s) = A_f G_1^\beta(s) + B_f G_2^\beta(s) + C_f. \tag{22}$$

The condition that $f(0) = 1$ implies $C_f = 1$. Then we form the derivative

$$f'(s) = A_f G_0^\beta(s) + B_f G_1^\beta(s). \tag{23}$$

The condition that $f'(0) = 0$ implies $A_f = 0$. The next derivative is

$$f''(s) = B_f G_0^\beta(s), \tag{24}$$

using the fact that $A_f = 0$. The condition that $f''(0) = -k/r_0$ implies that $B_f = -k/r_0$. Putting it together we find

$$f(s) = 1 - (k/r_0)G_2^\beta(s). \tag{25}$$

Similarly, we let

$$g(s) = A_g G_1^\beta(s) + B_g G_2^\beta(s) + C_g. \tag{26}$$

We find

$$g(s) = r_0 G_1^\beta(s) + r_0 \dot{r}_0 G_2^\beta(s). \tag{27}$$

From these we find $\dot{f}$ and $\dot{g}$ by using the relation $ds/dt = 1/r$ to find

$$\begin{aligned} \dot{f}(s) &= -(k/(r r_0))G_1^\beta(s), \\ \dot{g}(s) &= (r_0/r)(G_0^\beta(s) + \dot{r}_0 G_1^\beta(s)). \end{aligned} \tag{28}$$

But we still have to find s!

The relation between t and s is

$$h = t - t_0 = \int_0^s r(s)ds. \tag{29}$$

Let's express $r(s)$ in terms of $G_i^\beta(s)$. First,

$$r_0' = r_0\dot{r}_0 = \mathbf{x}_0 \cdot \dot{\mathbf{x}}_0. \tag{30}$$

After a small reduction, we find

$$r_0'' = k - r_0\beta, \tag{31}$$

and

$$r_0''' + \beta r_0' = 0. \tag{32}$$

So $r(s)$ may be expressed as

$$r(s) = r_0 + r_0' G_1^\beta(s) + r_0'' G_2^\beta(s). \tag{33}$$

Using Eqs. (30) and (31)

$$r(s) = r_0 G_0^\beta(s) + r_0\dot{r}_0 G_1^\beta(s) + k G_2^\beta(s). \tag{34}$$

The properties of the G_i^β allow an immediate integration to find

$$h = r_0 G_1^\beta(s) + r_0\dot{r}_0 G_2^\beta(s) + k G_3^\beta(s). \tag{35}$$

This implicit equation for s is our "Kepler equation." We can solve this by any of the standard methods: Newton, Halley, Laguerre, bisection, and so on. In practice, we first try Newton's method. If this fails, we try the Laguerre-Conway method. If this fails, we recursively subdivide the step.

For the hyperbolic case our solution is as follows. For small stepsizes the equation is well approximated by a cubic equation. We take the real solution of this cubic equation as our initial guess in Newton's method. If Newton's method does not converge, then we use the Laguerre-Conway method. If this fails we recursively subdivide the step.

Eq. (34) can be used to rewrite the equation for $\dot{g}$. We find

$$\dot{g}(s) = 1 - (k/r)G_2^\beta(s). \tag{36}$$

The expressions for the parabolic case can be found by taking a limit as β goes to zero. As it turns out the Kepler equation becomes a cubic equation in s and so can be solved without iteration.

4 NUMERICAL REFINEMENT

In order to have expressions that are well defined numerically we have to do a little more work.

Examining Eqs. (17) and (18), we see that for small arguments of the cosine and hyperbolic cosine functions there will be cancellation and loss of precision. But this problem can be fixed by using the half-angle formulas.

Assuming $\beta > 0$, let

$$\begin{aligned} s_2 &= \sin(\sqrt{\beta}s/2) \\ c_2 &= \cos(\sqrt{\beta}s/2), \end{aligned} \tag{37}$$

then we can write

$$\begin{aligned} G_1^\beta(s) &= 2 s_2 c_2/\sqrt{\beta} \\ G_2^\beta(s) &= 2 s_2 s_2/\beta \\ G_3^\beta(s) &= (s - G_1^\beta(s))/\beta \\ G_0^\beta(s) &= 1 - \beta G_2^\beta(s). \end{aligned} \tag{38}$$

The only expression that might be of concern is the expression for $G_3^\beta(s)$, but, in practice, it seems to not be a problem. Similar expressions can be derived for the $\beta < 0$ case. Such changes are made throughout the program.

5 NUMERICAL EXPLORATION

We compare here our method and program (`universal.c`), to two universal variable Kepler stepper programs that use Stumpff series. The program `drift_one.f` was written by Harold Levison and Martin Duncan. It is based on the presentation in Danby (1992). The program `drift_one.f` is available as part of the `SWIFT` package. This program is used in a number of programs derived from the Wisdom-Holman method (Wisdom & Holman 1991). These programs include `RMVS3` (Levison & Duncan 1994), also distributed in the `SWIFT` package. The same program is used to advance the Kepler problem in the program `SYMBA` (Duncan, Levison & Lee 1998), and in `MERCURY` (Chambers 1999). In order to compare to our program, written in the C programming language, we have translated, line by line, the program `drift_one.f` to a C version `drift_one.c`. We have *not* compared our code directly to the fortran program `drift_one.f`. We also compare our program to the Kepler stepper in `WHFast` (Rein & Tamayo 2015).

Our test is designed to work for both elliptic and hyperbolic orbits. Let $T = 2\pi/n$ where $n = \sqrt{k/|a|^3}$. For elliptic orbits, T is the orbital period, and n is the mean motion. We choose $G = (0.0172)^2$, which approximates the gravitational constant in units of AU, day, and solar mass. We take the mass factors to be one; thus the Kepler constant μ/m is numerically just G. We use the same Kepler constant in the `drift_one.c` calculations. We take the semimajor axis for the elliptic case to be 0.4AU. The eccentricity and stepsize are varied. The pericentric distance is $q = a(1 - e)$, and the velocity is determined from a through the value of the energy. We evolve the Kepler orbit back and forth through pericenter, adjusting the phase so that the pericenter is encountered with a wide range of phases. The chosen stepsize is h; we make use of the auxillary stepsize $h' = \gamma h$, where $\gamma = (\sqrt{5} - 1)/2 \approx 0.618$, is the irrational golden mean. We start at pericenter with $t = 0$, and evolve the orbit with stepsize h until $t > T/2$, i.e. until we have passed apocenter (for elliptic orbits). Then we adjust the phase with a positive step of h'. At this point we start collecting statistics. We reverse the timestep and evolve the orbit with timestep $-h$ until $t < -T/2$. Then we adjust the phase with a positive step of h'. We evolve with stepsize h until $t > T/2$; we adjust the phase with a postive step of h', and so on. We repeat this process for 100 pericenter passages. At the end we collect statistics again, and compare to the initial statistics. This procedure tests all parts of the evolution for a large variety of phases. It also tests both positive and negative timesteps.

The results for elliptic orbits for three methods are summarized in Figs. 3-5. The top plot in each set shows the energy error as a function of eccentricity e and stepsize h. The middle plot shows the sign of the energy error at the end of each evolution. We would like this plot to show an intimate mixture of positive and negative results, so that an evolution computed with the Kepler solver does not show any tendency to expand or contract. This "bias" plot was introduced by Rein & Tamayo (2015). The bottom plot shows an estimate of the computing time per Kepler step. The tests were run on a 4 GHz iMac, with an Intel i7 chip. All of the tests were compiled with C compiler optimizer option `-O3`. Comparing Figs. 3 and 4, we see that our code is more accurate, faster, and shows less bias than `drift_one.c`. Note the complicated structure in the bias plot for `drift_one.c`, and the intimate mixture of colors in the bias plots for `universal.c` and the Kepler stepper in `WHFast`.

We have computed the average ratio of the time per step for the interval $0.001 < h/T < 0.1$, which is the range of most interest for practical calculations, and find the average ratio for `drift_one.c` relative to `universal.c` is about 1.9. The code `universal.c` is about twice as fast as `drift_one.c`. We have also computed the average ratio of the time per step for the interval $0.001 < h/T < 0.1$ for the Kepler solver in `WHFast` and find that the time per step is about a factor of 0.79 smaller than our code—it is about 20% faster. In Fig. 1, histograms of the relative energy error in the elliptic case for the three methods are shown. We see that `drift_one.c` is less accurate than both `universal.c` and

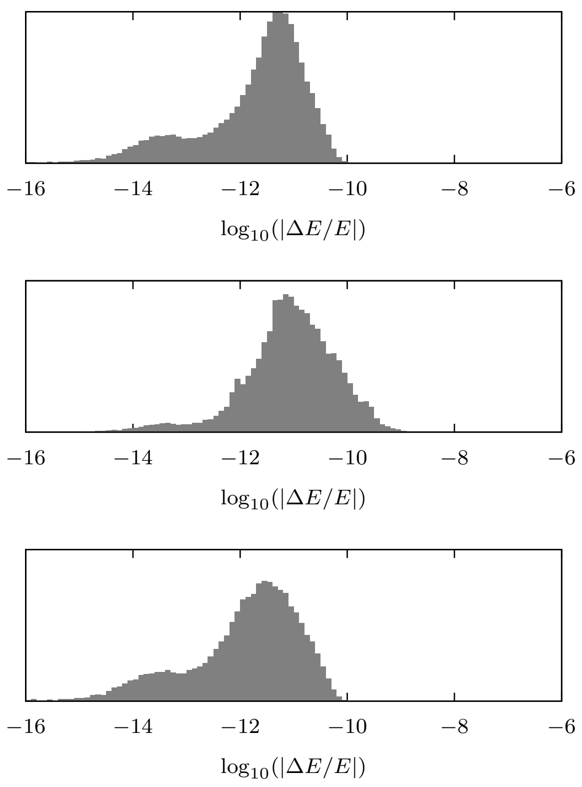

Figure 1. These plots histogram the relative error in the elliptic case for three methods. The bottom plot is for `universal.c`, the middle plot is for `drift_one.c`, and the top plot is for the Kepler solver in `WHFast`.

.

the Kepler solver in `WHFast`. Quantitatively, the averages of the common logarithm of the absolute value of the relative error are: -11.92, -11.74, and -11.09, for `universal.c`, the Kepler solver in `WHFast`, and `drift_one.c`, respectively. The error of `universal.c` is, on average, about 50% smaller than the Kepler solver in `WHFast`, and about a factor of 6.5 smaller than the error in `drift_one.c`.

For the hyperbolic case, we let $a = -0.4AU$. We start at pericenter. The pericentric distance is $q = a(1 - e)$; e is larger than 1. The initial velocity is determined from the energy. The back and forth procedure is the same as in the elliptic case.

The results for hyperbolic orbits for our code, `universal.c`, and for `drift_one.c` are shown in Figs. 6 and 7. Rein & Tamayo (2015) did not extensively test the hyperbolic case, and the Kepler solver in WHFast performs poorly for unbound orbits. Note again that the bias plot for our method `universal.c` shows an intimate mixture, whereas the bias plot for `drift_one.c` shows evidence of bias. We find that the average ratio of the time per step of `drift_one.c` to the time per step of `universal.c` for the interval $0.001 < h/T < 0.1$ is about 1.6; `drift_one.c` is about 60% slower than `universal.c`.

In Fig. 2, histograms of the relative energy error in the hyperbolic case for `drift_one.c` and `universal.c` are shown. We see that `drift_one.c` is typically less accurate than `universal.c`. Quantitatively, the averages of the com-

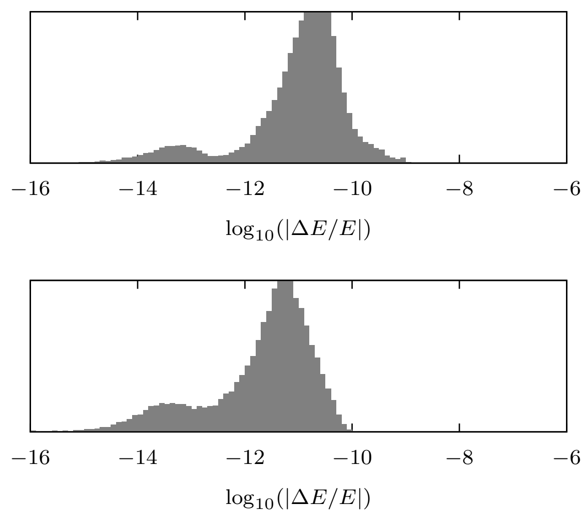

Figure 2. These plots histogram the relative error in the hyperbolic case for two methods. The bottom plot is for `universal.c`, and the top plot is for `drift_one.c`.

mon logarithm of the absolute value of the relative error are: -11.72 and -11.03, for `universal.c` and `drift_one.c`, respectively. The error of `universal.c` is, on average, about a factor of 5 smaller than the error in `drift_one.c`.

6 SUMMARY

We have developed and tested a universal variable solver for the Kepler initial value problem. The method eschews the use of Stumpff series in favor of ordinary trigonometric functions. We have been careful to make sure that all expressions are numerically well defined. We find that our program performs better, in terms of accuracy, bias, and speed, than a C version of a widely used Kepler solver, which uses the Stumpff series. Our code is freely available; contact the authors.

ACKNOWLEDGEMENTS

We thank Hanno Rein, Daniel Tamayo, and Edmund Bertschinger for helpful discussions. DMH acknowledges support by a NSF Graduate Research Fellowship under Grant No. 1122374.

REFERENCES

Chambers J. E., 1999, MNRAS, 304, 793
Danby J. M. A., 1992, Fundamentals of celestial mechanics, 2nd edn. Willmann-Bell, Richmond, Va., U.S.A.
Duncan M. J., Levison H. F., Lee M. H., 1998, AJ, 116, 2067
Hernandez D. M., Bertschinger E., 2015, MNRAS, 452, 1934
Levison H. F., Duncan M. J., 1994, Icarus, 108, 18
Levison H. F., Duncan M. J., 2000, AJ, 120, 2117
Rein H., Tamayo D., 2015, MNRAS, 452, 376
Touma J., Wisdom J., 1993, Science, 259, 1294
Touma J., Wisdom J., 1994, AJ, 107, 1189
Wisdom J., 1982, AJ, 87, 577
Wisdom J., Holman M., 1991, AJ, 102, 1528
Wisdom J., Holman M., 1992, AJ, 104, 2022
Wisdom J., Holman M., Touma J., 1996, Fields Institute Communications, Vol. 10, p. 217, 10, 217

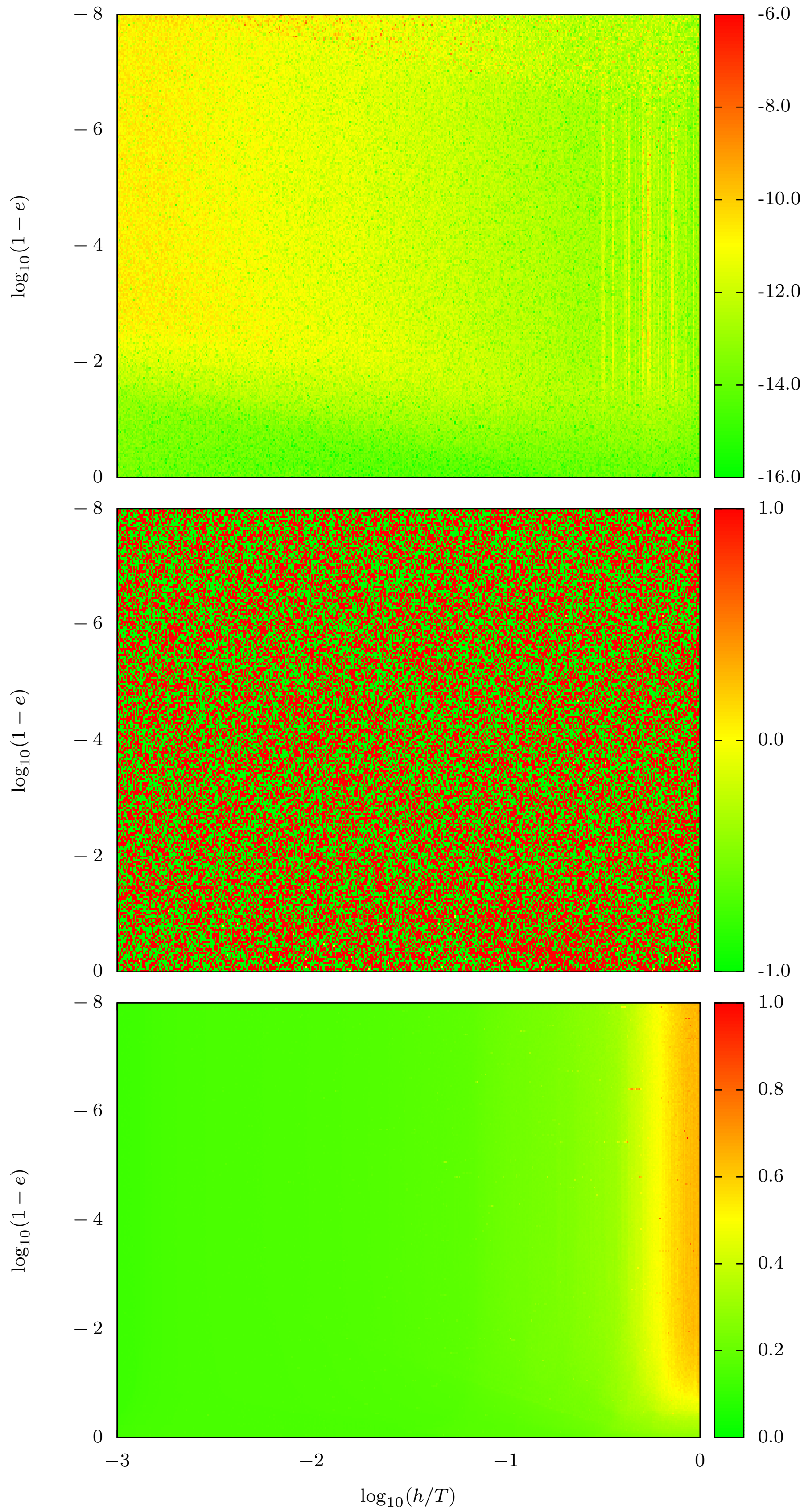

Figure 3. The summary diagrams in the elliptic case for `universal.c` are plotted. The top plot shows the relative energy error as a function of eccentricity e and stepsize h. The color bar shows the common logarithm of the magnitude. The middle plot illustrates bias in the calculation, plotting 1 if the energy error is positive, -1 if the energy error is negative, and 0 for zero energy error. The bottom plot shows the computing time in microseconds per call to the Kepler stepper.

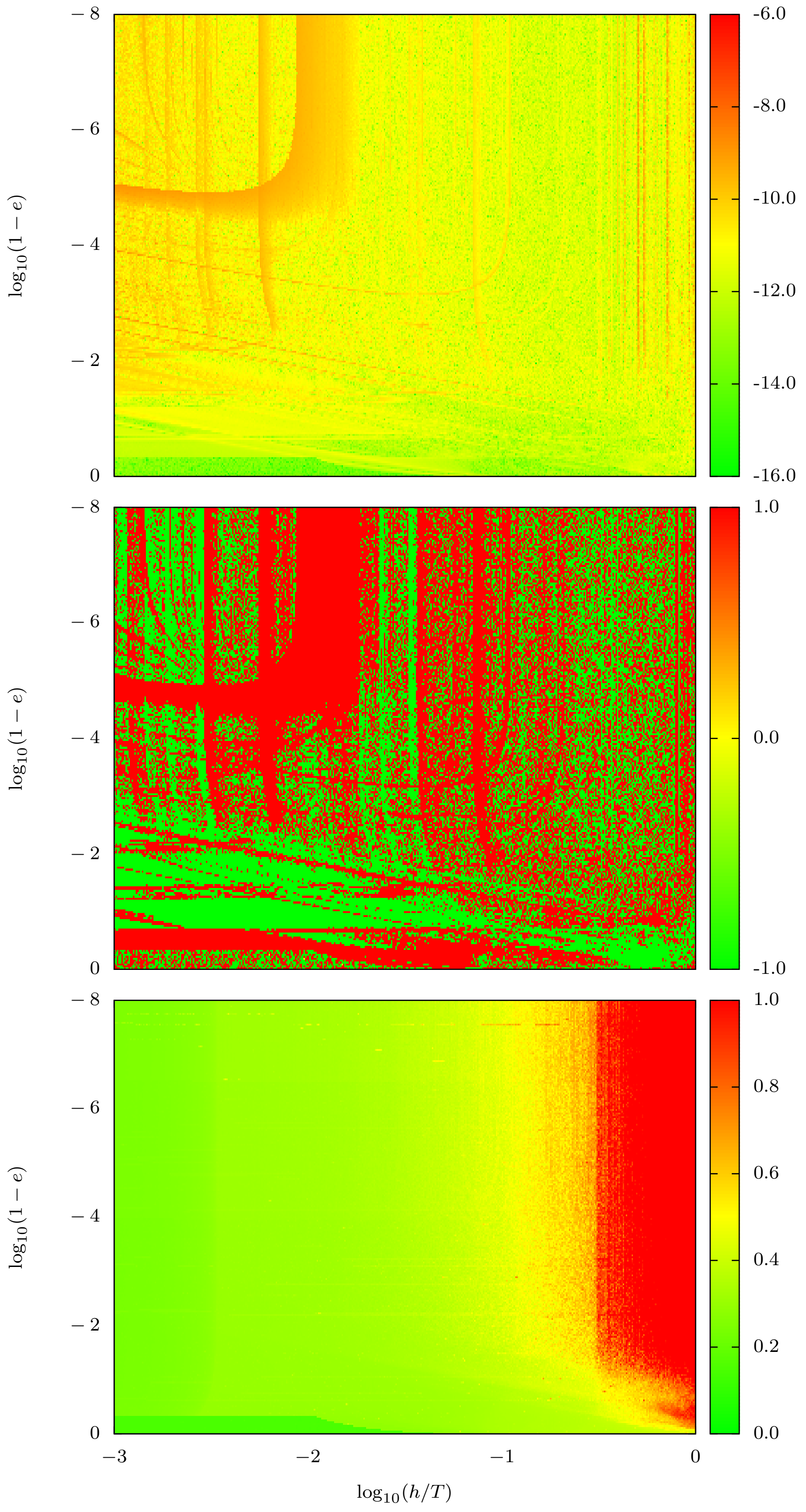

Figure 4. The summary diagrams in the elliptic case for `drift_one.c` are plotted. The details are the same as in Fig. 3.

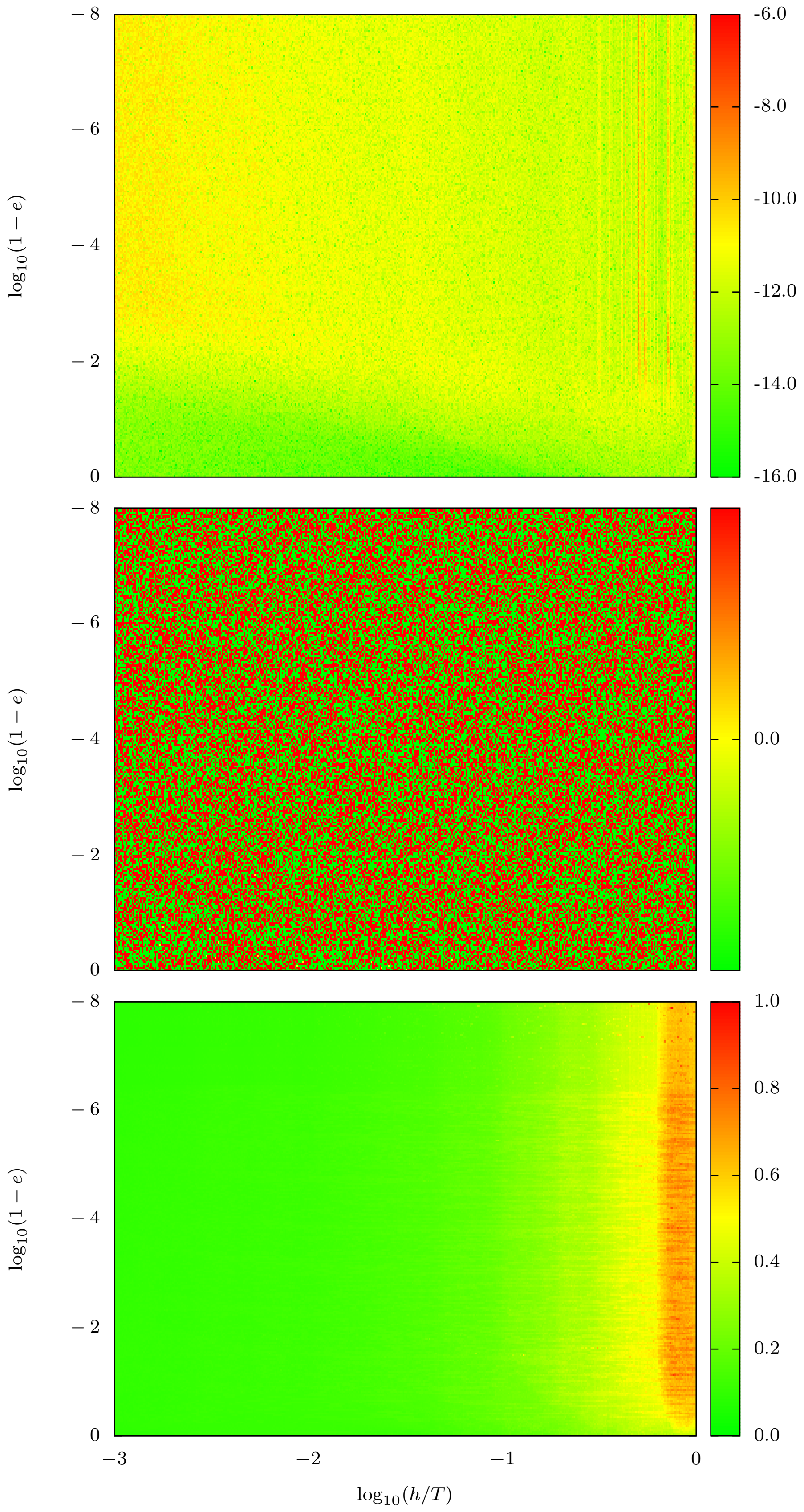

Figure 5. The summary diagrams in the elliptic case for the Kepler solver in WHFast are plotted. The details are the same as in Fig. 3.

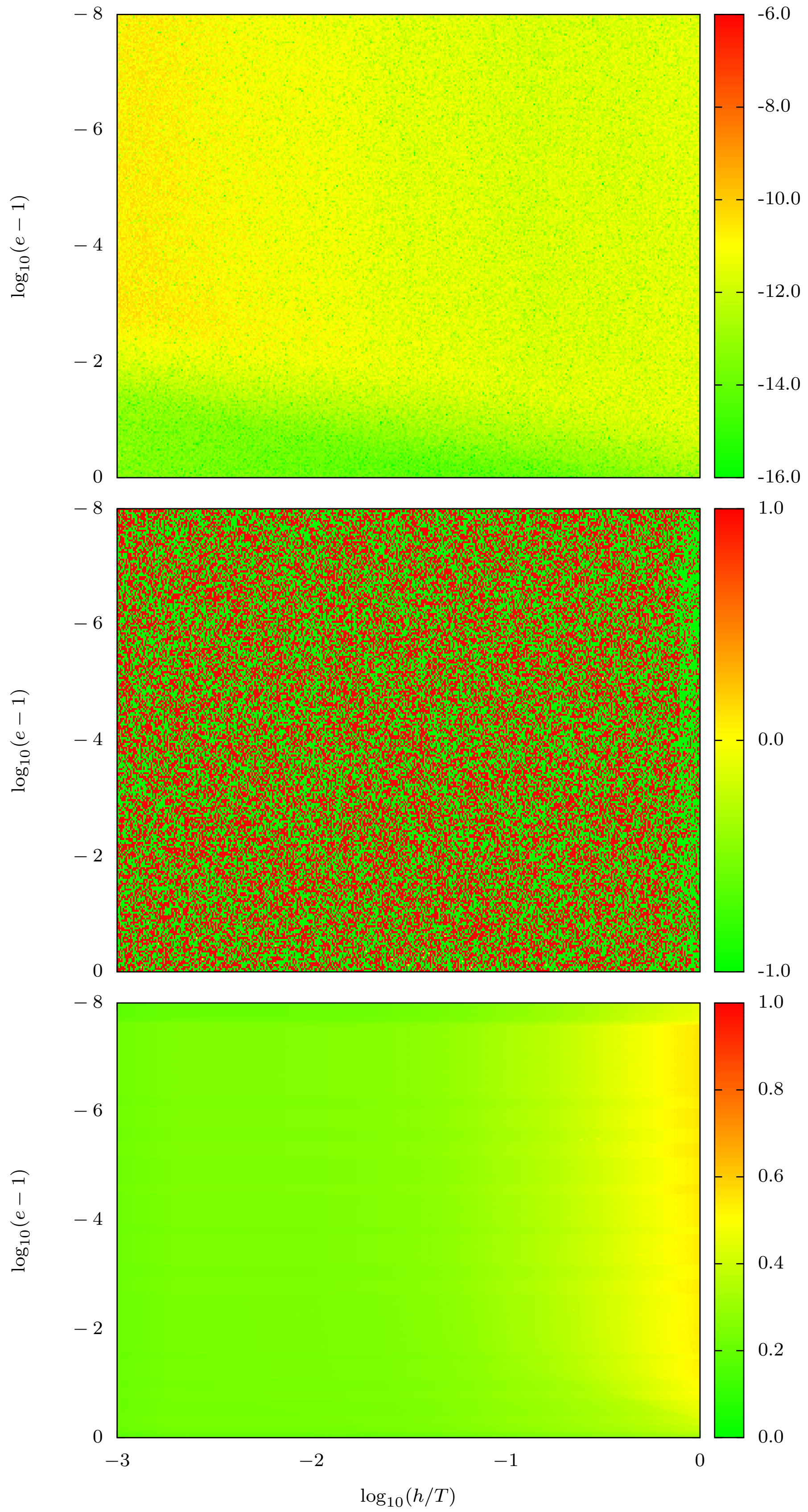

Figure 6. The summary diagrams in the hyperbolic case for `universal.c` are plotted. The details are the same as in Fig. 3.

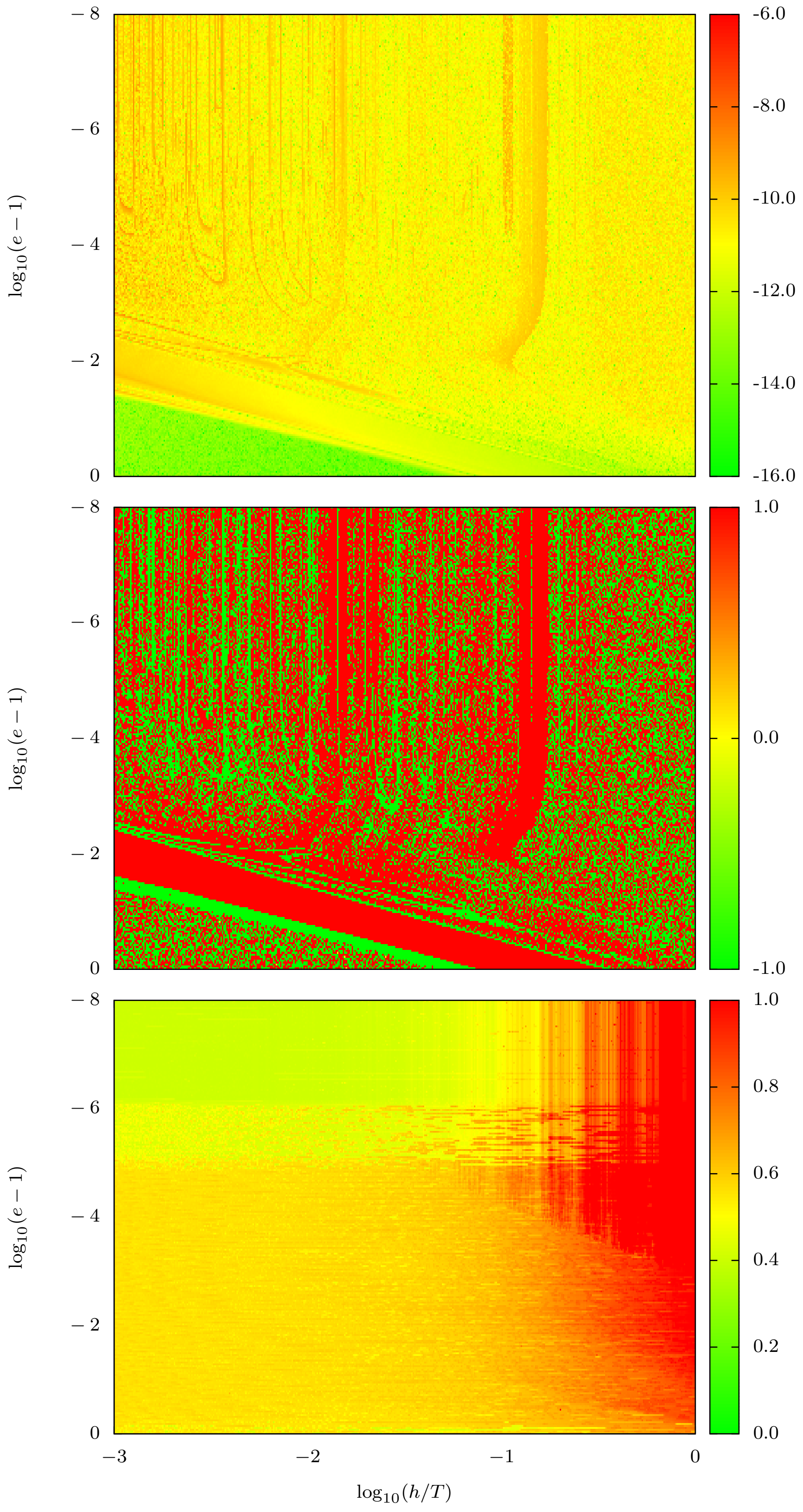

Figure 7. The summary diagrams in the hyperbolic case for `drift_one.c` are plotted. The details are the same as in Fig. 3.