On page 176, in exercise 2.18

equations 2.144 and 2.144

The second 144 should probably be a 145 (unless there
is a mathematical joke behind the repetition).

In the exercise about symplectic integration many "2pi"'s ought to be
replaced by "1/2pi".

In equation 2.96 the middle entry in the bottom row of the 3x3 matrix
should have a minus sign.

Page 141, first paragraph, first sentence:

The two occurrences of the word "if" are incorrect. The clause
following each is a parenthetical.

Because of our ordering A <= B <= C the longest axis of this triaxial
energy ellipsoid coincides with the \hat{c} direction (All of the
angular momentum is along the axis of largest principal moment of
intertia.) and the shortest axis of the ellipsoid coincides with the
\hat{a} axis (All of the angular momentum is along the axis of
smallest principal moment of intertia.).

Equations 1.185 and 1.186 are a bit funny.

Lambda compose Gamma[q] = lambda

but in these equations Lambda has been replaced by lambda
even though there is still a composition with Gamma[q].

There is a hidden (non explicit) use of commutation in the action
derivation of Hamilton's equations.

Equation 5.92 on p. 348 has an error. The L_e^\prime on the second
line should just be L_e.

There is an extra "that" in the sentence just after equation 5.98 on
p. 349.

----

on p. 274 (paraphrased):

```
     Consider the system
     Dz(t) = F(t, z(t))
     or
     Dz^i(t) = F^i(t, z^1(t), ... , z^n(t))
     for i=1 to n
```

where n is the dimension of the state space.

If F depends on t then the "state" is not just z^1(t) to z^n(t), but must also include t. n is the dimension of z, not the dimension of the state.

Also, why is this 1-based instead of 0-based as the rest of the book?

----

on p. 278 I is called the "identity function" by it is really a multiplicative identity. (It is not a function of anything. Of course it will be different depending on what it multiplies.)

In the sentence just before equation 4.27 on p. 278, the word "show" should be "shows".

On p.279, last paragraph, there is a missing "a". The sentence should read: "If the reference trajectory is a fixed point ..."

This is not an error in the book, but it is a proof by assertion. On page 204 (Properties of the Poisson bracket) we blithely assert that {F, G} = - {G, F} -- not admitting that this is hairy if F and G have structured arguments. Thus the conservation of the Hamiltonian is also not obvious by this argument. The commutation of the partial derivatives depends on the commutability of tuples that are compatable for contraction.

On p.340 the sentence before 5.68 says "Abstracted to functions of phase-space states ..." but there is nothing to abstract. It is the same as eq 5.67. Apparently, 5.67 was improved, but the followup sentence leading to 5.68 was not adjusted.

On p.348 eq.5.92 second line... the "L'_e" should be just "L_e"

In the sentence just before eq.5.99 there are two "that"s. There should just be one.

We should think about whether "q_t" aught to be replaced by "t" everywhere in the extended phase space section. The advantage of "q_t" is that it is evidently one of the coordinates. But it is confusing to have two different names for the same thing.

Well, the "E" in eq.5.124 is funny. Isn't E the value of the Hamiltonian along solution paths, not the value of H on an arbitrary state.

P.244 Code does not work as written because of lack of (null) parameters for state-advancer.

The code as printed is:

```
        (define (HHmap E)
          (let ((advancer
                 (state-advancer
                  (phase-space-derivative HHHam))))
            (lambda (y py cont fail)
              (define (find-next-crossing s)
                (let lp ((s s))
                  (let ((ns (advancer s .1 1.e-12)))
                    (if (and (> (ref (coordinate ns) 0) 0)
                             (< (ref (coordinate s) 0) 0))
                        (refine-crossing s)
                        (lp ns)))))
              (define (refine-crossing s)
                (let ((dt (- (/ (ref (coordinate s) 0)
                                (ref (momentum s) 0)))))
                  (let ((ns (advancer s dt 1.e-12)))
                    (if (< (abs (ref (coordinate s) 0))
                           1.e-10)
                        ;; return new section point
                        (cont (ref (coordinate ns) 1)
                              (ref (momentum ns) 1))
                        ;; continue refining
                        (refine-crossing ns)))))
              (let ((initial-state (section->state E y py)))
                (if (not initial-state)
                    (fail)
                    (find-next-crossing initial-state))))))
```

It should have been

```
        (define (HHsysder)
          (phase-space-derivative HHHam))

        (define (HHmap E)
          (let ((advancer (state-advancer HHsysder)))
            (lambda (y py cont fail)
              (define (find-next-crossing s)
                (let lp ((s s))
                  (let ((ns (advancer s .1 1.e-12)))
                    (if (and (> (ref (coordinate ns) 0) 0)
                             (< (ref (coordinate s) 0) 0))
                        (refine-crossing s)
                        (lp ns)))))
              (define (refine-crossing s)
                (let ((dt (- (/ (ref (coordinate s) 0)
                                (ref (momentum s) 0)))))
                  (let ((ns (advancer s dt 1.e-12)))
                    (if (< (abs (ref (coordinate s) 0))
                           1.e-10)
                        ;; return new section point
                        (cont (ref (coordinate ns) 1)
                              (ref (momentum ns) 1))
                        ;; continue refining
                        (refine-crossing ns)))))
              (let ((initial-state (section->state E y py)))
                (if (not initial-state)
                    (fail)
                    (find-next-crossing initial-state))))))
```