Test of ALFAM2 closed-form solution

Sasha D. Hafner

09 March, 2024

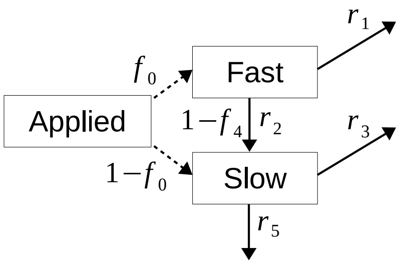


Figure 1: Structure of ALFAM2 model.

```
library(ALFAM2)
```

```
packageVersion('ALFAM2')
```

```
## [1] '3.72'
```

```
library(data.table)
library(ggplot2)
library(deSolve)
```

```
logistic <-
function (x) {
  exp(x)/(1 + exp(x))
}

logit <-
function (p) {
  log(p/(1 - p))
}

numalfam2 <- function(dat, pars = ALFAM2::alfam2pars03_alpha, app.name = 'TAN_app', time.name = 'time')

  pars <- pars[grepl('^int', names(pars))]
  pars[grepl('\\.f', names(pars))] <- logistic(pars[grepl('\\.f', names(pars))])
  pars[grepl('\\.r', names(pars))] <- 10^(pars[grepl('\\.r', names(pars))])
  names(pars) <- gsub('int\\.', '', names(pars))
  f0 <- pars['f0']
  r1 <- pars['r1']
  r2 <- pars['r2']
```

```r
  r3 <- pars['r3']
  r5 <- pars['r5']

  f <- dat[[app.name]][1] * pars['f0']
  s <- dat[[app.name]][1] * (1 - pars['f0'])
  e <- 0
  tt <- 0
  res <- data.table()

  for (j in unique(dat[dat[[group]] == i, ][[time.name]])) {
    dtt <- j - tt
    f2 <- f - r1 * f * dtt - r2 * f * dtt
    s2 <- s + r2 * f * dtt - r3 * s * dtt - r5 * s * dtt

    f <- (f + f2) / 2
    s <- (s + s2) / 2

    e <- e + r1 * f * dtt + r3 * s * dtt

    f <- f2
    s <- s2

    tt <- j

    res <- rbind(res, data.table(time = j, f = f, s = s, e = e))
  }

  return(res)

}
```

```r
tstart <- 0
tend <- 168
dat1 <- data.table(steps = '00002', time = seq(tstart, tend, length.out = 2), TAN_app = 100)
dat2 <- data.table(steps = '00010', time = seq(tstart, tend, length.out = 10), TAN_app = 100)
dat3 <- data.table(steps = '00100', time = seq(tstart, tend, length.out = 100), TAN_app = 100)
dat4 <- data.table(steps = '01000', time = seq(tstart, tend, length.out = 1E3), TAN_app = 100)
dat5 <- data.table(steps = '10000', time = seq(tstart, tend, length.out = 1E3), TAN_app = 100)
dat <- rbind(dat1, dat2, dat3, dat4, dat5, dat6)
```

```r
predscf <- data.table(alfam2(dat, pars = ALFAM2::alfam2pars03_alpha, app.name = 'TAN_app', time.name =
```

```
## User-supplied parameters are being used.

## Warning in prepDat(dat, value = "dummy", warn = warn): Argument prep.dum = TRUE but there are no vari
##   Ignoring prep.dum = TRUE.

## Warning in alfam2(dat, pars = ALFAM2::alfam2pars03_alpha, app.name = "TAN_app", : Running with 5 para
## These secondary parameters have been dropped:
##   app.mthd.os.f0
##   app.rate.ni.f0
##   man.dm.f0
##   man.source.pig.f0
##   app.mthd.cs.f0
##   app.mthd.bc.r1
##   man.dm.r1
##   air.temp.r1
##   app.mthd.ts.r1
##   man.ph.r1
##   rain.rate.r2
##   app.mthd.bc.r3
##   app.mthd.cs.r3
##   man.ph.r3
##   incorp.shallow.f4
##   incorp.shallow.r3
##   incorp.deep.f4
##   incorp.deep.r3
##   rain.rate.r5
##   wind.sqrt.r1
```

```
predsnum <- data.table()
for (i in unique(dat[, steps])) {
  dd <- dat[steps == i, ]
  pr <- numalfam2(dd)
  pr[, steps := i]
  predsnum <- rbind(predsnum, pr)
}
```

```
pars <- ALFAM2::alfam2pars03_alpha
pars <- pars[grepl('^int', names(pars))]
pars[grepl('\\.f', names(pars))] <- logistic(pars[grepl('\\.f', names(pars))])
pars[grepl('\\.r', names(pars))] <- 10^(pars[grepl('\\.r', names(pars))])
names(pars) <- gsub('int\\.', '', names(pars))
pars
```

```
##          f0          r1          r2          r3          r5
## 0.741653831 0.056680798 0.071535526 0.002038241 0.015848932
```

```
y <- c(f = pars['f0'] * 100, s = (1 - pars['f0']) * 100, e = 0)
pars <- pars[-1]

rates <- function(t, x, parms) {
  r1 <- parms['r1']
  r2 <- parms['r2']
  r3 <- parms['r3']
  r5 <- parms['r5']

  f <- x[1]
  s <- x[2]

  dfdt <-  -r1 * f - r2 * f
  dsdt <- r2 * f - r3 * s - r5 * s
  dedt <- r1 * f + r3 * s
  return(list(c(dfdt, dsdt, dedt)))

}

lsoda(y = y, times = 1:168, func = rates, parms = pars)
```

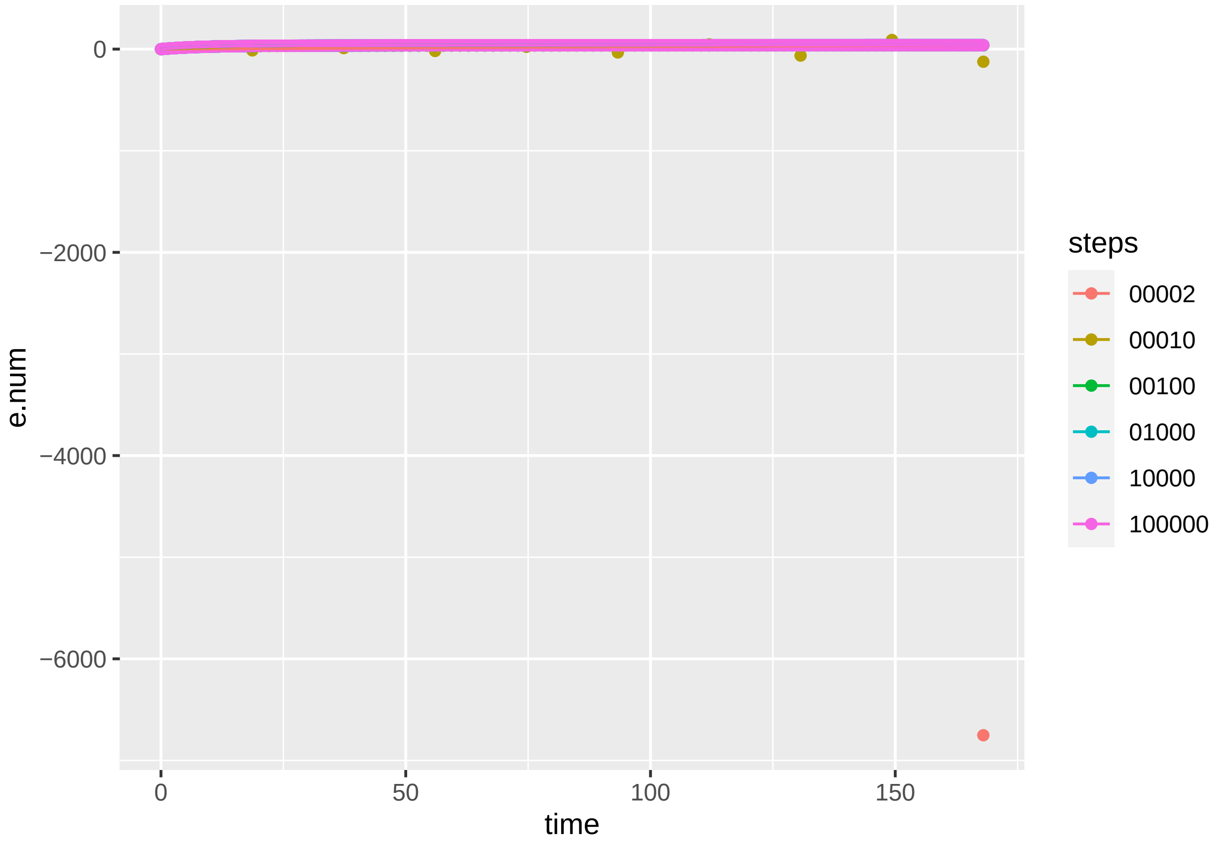
```
##     time         f.f0      s.f0          e
## 1      1 7.416538e+01 25.834617  0.000000
## 2      2 6.524055e+01 30.310821  4.002755
## 3      3 5.738970e+01 34.113906  7.539157
## 4      4 5.048359e+01 37.327261 10.665053
## 5      5 4.440857e+01 40.024170 13.429562
## 6      6 3.906458e+01 42.269104 15.875925
## 7      7 3.436367e+01 44.118707 18.042166
## 8      8 3.022845e+01 45.622770 19.961740
## 9      9 2.659086e+01 46.825053 21.664082
## 10    10 2.339099e+01 47.764016 23.175090
## 11    11 2.057619e+01 48.473447 24.517548
## 12    12 1.810012e+01 48.983033 25.711502
## 13    13 1.592200e+01 49.318851 26.774593
## 14    14 1.400600e+01 49.503807 27.722341
## 15    15 1.232056e+01 49.558012 28.568402
## 16    16 1.083794e+01 49.499124 29.324794
## 17    17 9.533736e+00 49.342642 30.002093
## 18    18 8.386476e+00 49.102166 30.609605
## 19    19 7.377273e+00 48.789626 31.155520
## 20    20 6.489514e+00 48.415485 31.647046
## 21    21 5.708586e+00 47.988914 32.090528
## 22    22 5.021632e+00 47.517950 32.491551
## 23    23 4.417344e+00 47.009633 32.855030
## 24    24 3.885775e+00 46.470124 33.185294
## 25    25 3.418172e+00 45.904815 33.486153
## 26    26 3.006840e+00 45.318418 33.760962
## 27    27 2.645006e+00 44.715051 34.012676
## 28    28 2.326714e+00 44.098307 34.243897
## 29    29 2.046724e+00 43.471322 34.456918
## 30    30 1.800428e+00 42.836824 34.653758
## 31    31 1.583770e+00 42.197189 34.836197
## 32    32 1.393184e+00 41.554478 35.005803
## 33    33 1.225532e+00 40.910483 35.163959
## 34    34 1.078056e+00 40.266750 35.311883
## 35    35 9.483257e-01 39.624618 35.450652
## 36    36 8.342071e-01 38.985239 35.581213
## 37    37 7.338211e-01 38.349602 35.704403
## 38    38 6.455153e-01 37.718556 35.820963
## 39    39 5.678360e-01 37.092822 35.931543
## 40    40 4.995044e-01 36.473013 36.036722
## 41    41 4.393955e-01 35.859646 36.137009
## 42    42 3.865200e-01 35.253154 36.232855
## 43    43 3.400074e-01 34.653896 36.324660
## 44    44 2.990919e-01 34.062167 36.412776
## 45    45 2.631001e-01 33.478206 36.497517
## 46    46 2.314395e-01 32.902204 36.579162
## 47    47 2.035887e-01 32.334306 36.657956
## 48    48 1.790895e-01 31.774624 36.734120
## 49    49 1.575384e-01 31.223233 36.807848
## 50    50 1.385807e-01 30.680184 36.879314
## 51    51 1.219043e-01 30.145501 36.948674
## 52    52 1.072347e-01 29.619188 37.016065
## 53    53 9.433045e-02 29.101229 37.081611
## 54    54 8.297901e-02 28.591593 37.145424
## 55    55 7.299357e-02 28.090238 37.207602
## 56    56 6.420975e-02 27.597106 37.268236
## 57    57 5.648294e-02 27.112133 37.327406
## 58    58 4.968596e-02 26.635243 37.385184
## 59    59 4.370690e-02 26.166355 37.441637
## 60    60 3.844735e-02 25.705382 37.496825
## 61    61 3.382071e-02 25.252231 37.550801
## 62    62 2.975083e-02 24.806806 37.603615
## 63    63 2.617071e-02 24.369006 37.655312
## 64    64 2.302140e-02 23.938730 37.705935
## 65    65 2.025108e-02 23.515872 37.755520
## 66    66 1.781412e-02 23.100327 37.804103
## 67    67 1.567043e-02 22.691986 37.851718
## 68    68 1.378470e-02 22.290742 37.898393
## 69    69 1.212589e-02 21.896485 37.944157
## 70    70 1.066670e-02 21.509108 37.989037
## 71    71 9.383102e-03 21.128501 38.033056
## 72    72 8.253969e-03 20.754556 38.076238
## 73    73 7.260713e-03 20.387165 38.118604
## 74    74 6.386981e-03 20.026222 38.160175
## 75    75 5.618392e-03 19.671619 38.200971
## 76    76 4.942292e-03 19.323251 38.241009
## 77    77 4.347552e-03 18.981015 38.280308
## 78    78 3.824382e-03 18.644806 38.318883
## 79    79 3.364168e-03 18.314522 38.356752
## 80    80 2.959334e-03 17.990063 38.393929
## 81    81 2.603217e-03 17.671330 38.430428
## 82    82 2.289954e-03 17.358223 38.466265
## 83    83 2.014388e-03 17.050646 38.501453
## 84    84 1.771983e-03 16.748504 38.536005
## 85    85 1.558748e-03 16.451702 38.569933
## 86    86 1.371173e-03 16.160147 38.603250
## 87    87 1.206171e-03 15.873749 38.635969
## 88    88 1.061024e-03 15.592417 38.668100
## 89    89 9.333436e-04 15.316063 38.699655
## 90    90 8.210278e-04 15.044600 38.730645
## 91    91 7.222272e-04 14.777942 38.761081
## 92    92 6.353160e-04 14.516004 38.790972
## 93    93 5.588635e-04 14.258705 38.820330
## 94    94 4.916112e-04 14.005961 38.849164
## 95    95 4.324518e-04 13.757694 38.877484
## 96    96 3.804115e-04 13.513825 38.905299
## 97    97 3.346337e-04 13.274275 38.932619
## 98    98 2.943646e-04 13.038969 38.959453
## 99    99 2.589415e-04 12.807832 38.985809
## 100  100 2.277810e-04 12.580790 39.011696
## 101  101 2.003704e-04 12.357771 39.037123
## 102  102 1.762583e-04 12.138704 39.062097
## 103  103 1.550477e-04 11.923519 39.086629
## 104  104 1.363896e-04 11.712147 39.110724
## 105  105 1.199768e-04 11.504522 39.134391
## 106  106 1.055391e-04 11.300576 39.157638
## 107  107 9.283875e-05 11.100245 39.180472
## 108  108 8.166675e-05 10.903464 39.202901
## 109  109 7.183916e-05 10.710171 39.224931
## 110  110 6.319420e-05 10.520304 39.246571
## 111  111 5.558955e-05 10.333803 39.267827
## 112  112 4.890003e-05 10.150608 39.288705
## 113  113 4.301551e-05  9.970659 39.309213
## 114  114 3.783912e-05  9.793901 39.329357
## 115  115 3.328565e-05  9.620276 39.349144
## 116  116 2.928013e-05  9.449728 39.368580
## 117  117 2.575663e-05  9.282204 39.387671
## 118  118 2.265713e-05  9.117650 39.406424
## 119  119 1.993062e-05  8.956012 39.424844
## 120  120 1.753222e-05  8.797240 39.442937
## 121  121 1.542243e-05  8.641282 39.460710
## 122  122 1.356653e-05  8.488089 39.478167
## 123  123 1.193396e-05  8.337612 39.495314
## 124  124 1.049786e-05  8.189803 39.512158
## 125  125 9.234570e-06  8.044613 39.528703
## 126  126 8.123303e-06  7.901998 39.544955
## 127  127 7.145763e-06  7.761911 39.560918
## 128  128 6.285858e-06  7.624307 39.576598
## 129  129 5.529432e-06  7.489142 39.592001
## 130  130 4.864039e-06  7.356374 39.607130
## 131  131 4.278719e-06  7.225959 39.621991
## 132  132 3.763831e-06  7.097857 39.636589
## 133  133 3.310904e-06  6.972025 39.650927
## 134  134 2.912480e-06  6.848424 39.665012
## 135  135 2.562001e-06  6.727015 39.678847
## 136  136 2.253698e-06  6.607757 39.692436
## 137  137 1.982493e-06  6.490614 39.705785
## 138  138 1.743924e-06  6.375548 39.718897
## 139  139 1.534064e-06  6.262521 39.731776
## 140  140 1.349458e-06  6.151499 39.744427
## 141  141 1.187069e-06  6.042444 39.756854
## 142  142 1.044222e-06  5.935323 39.769061
## 143  143 9.185634e-07  5.830100 39.781051
## 144  144 8.080263e-07  5.726744 39.792828
## 145  145 7.107910e-07  5.625219 39.804397
## 146  146 6.252566e-07  5.525494 39.815761
## 147  147 5.500152e-07  5.427538 39.826923
## 148  148 4.838281e-07  5.331318 39.837887
## 149  149 4.256057e-07  5.236803 39.848657
## 150  150 3.743897e-07  5.143964 39.859236
## 151  151 3.293368e-07  5.052772 39.869627
## 152  152 2.897055e-07  4.963195 39.879835
## 153  153 2.548432e-07  4.875207 39.889861
## 154  154 2.241761e-07  4.788779 39.899709
## 155  155 1.971994e-07  4.703883 39.909383
## 156  156 1.734691e-07  4.620491 39.918886
## 157  157 1.525943e-07  4.538579 39.928220
## 158  158 1.342316e-07  4.458118 39.937388
## 159  159 1.180786e-07  4.379084 39.946394
## 160  160 1.038694e-07  4.301451 39.955240
## 161  161 9.137004e-08  4.225194 39.963930
## 162  162 8.037485e-08  4.150289 39.972465
## 163  163 7.070279e-08  4.076712 39.980849
## 164  164 6.219463e-08  4.004440 39.989085
## 165  165 5.471032e-08  3.933449 39.997174
## 166  166 4.812664e-08  3.863716 40.005120
## 167  167 4.233523e-08  3.795219 40.012926
## 168  168 3.724074e-08  3.727937 40.020592
```

```
tail(dat)
```

```
##     steps     time TAN_app
## 1: 100000 167.1592     100
## 2: 100000 167.3273     100
## 3: 100000 167.4955     100
## 4: 100000 167.6637     100
## 5: 100000 167.8318     100
## 6: 100000 168.0000     100
```

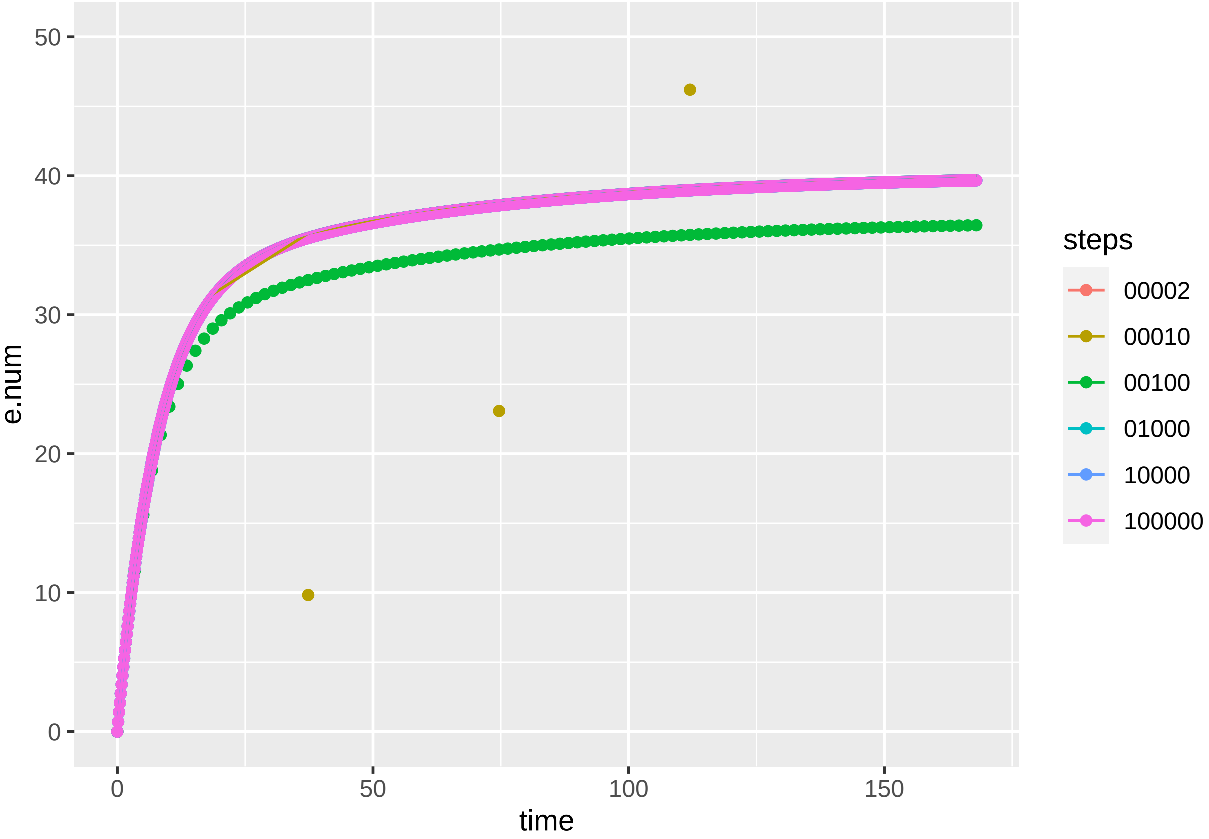
```
dat <- merge(dat, predscf, by = c('steps', 'time'))
dat <- merge(dat, predsnum, by = c('steps', 'time'), suffixes = c('.cf', '.num'))
```

```
ggplot(dat, aes(time, e.num, colour = steps)) +
  geom_point() +
  geom_line(aes(y = e.cf))
```



```
ggplot(dat, aes(time, e.num, colour = steps)) +
  geom_point() +
  geom_line(aes(y = e.cf)) +
  ylim(0, 50)
```

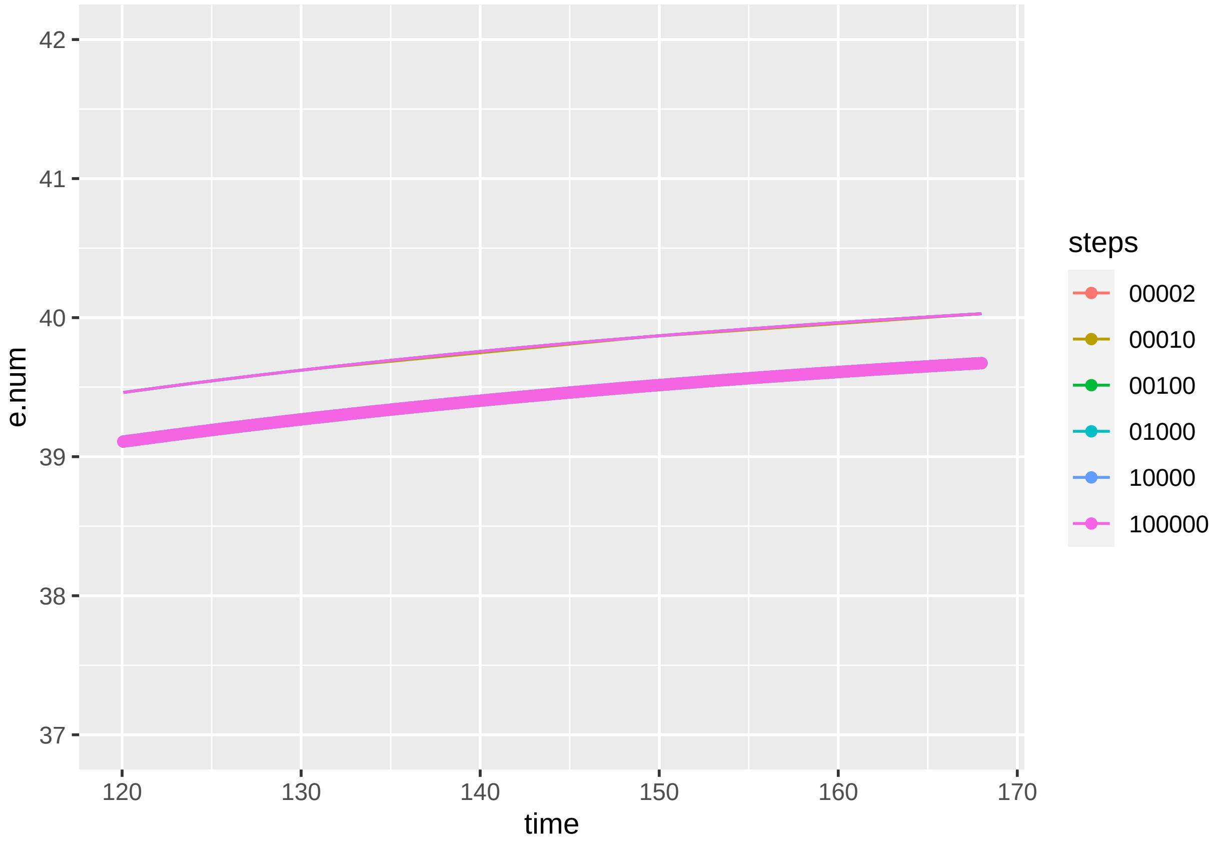
```
## Warning: Removed 7 rows containing missing values (`geom_point()`).
## Warning: Removed 6 rows containing missing values (`geom_line()`).
```



```
ggplot(dat, aes(time, e.num, colour = steps)) +
  geom_point() +
  geom_line(aes(y = e.cf)) +
  xlim(120, NA) +
  ylim(37, 42)
```

```
## Warning: Removed 2254 rows containing missing values (`geom_point()`).
```

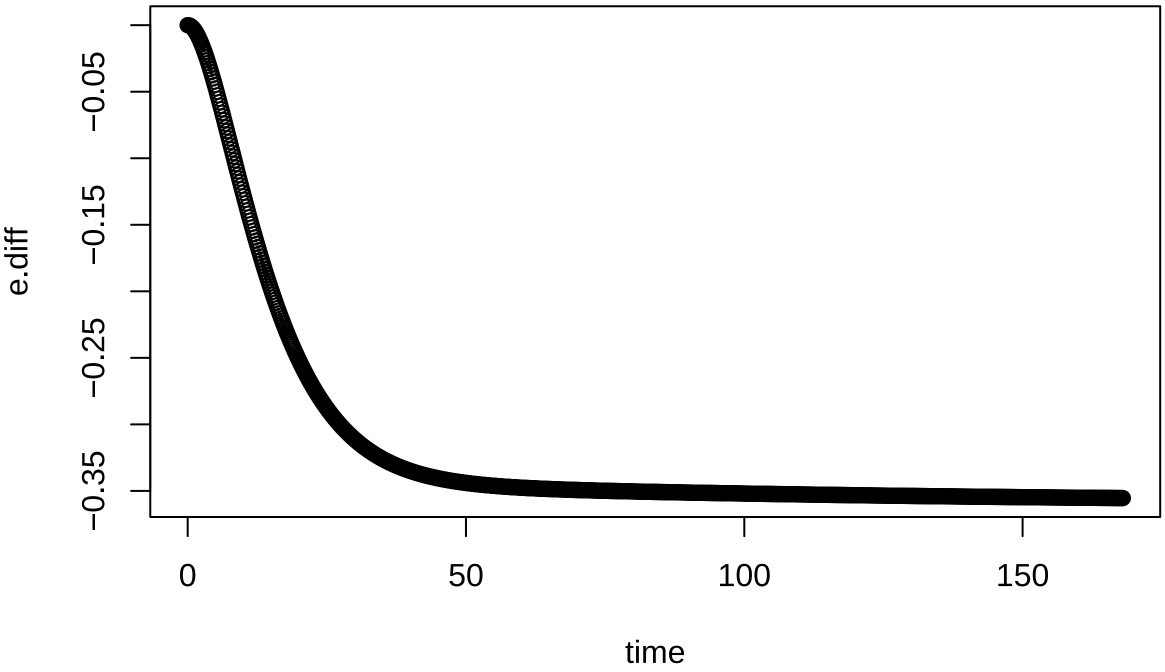
```
## Warning: Removed 2221 rows containing missing values (`geom_line()`).
```



```
dat[, e.diff := e.num - e.cf]
tail(dat)
```

```
##     steps     time TAN_app        dt          f.cf     s.cf     e.cf        e.int
## 1: 100000 167.1592     100 0.1681682 3.648972e-08 3.717339 40.02180 0.001276101
## 2: 100000 167.3273     100 0.1681682 3.571135e-08 3.706174 40.02307 0.001272268
## 3: 100000 167.4955     100 0.1681682 3.494959e-08 3.695042 40.02434 0.001268447
## 4: 100000 167.6637     100 0.1681682 3.420407e-08 3.683944 40.02561 0.001264637
## 5: 100000 167.8318     100 0.1681682 3.347446e-08 3.672879 40.02687 0.001260838
## 6: 100000 168.0000     100 0.1681682 3.276042e-08 3.661848 40.02812 0.001257051
##              j        er        f0        r1         r2          r3 f4
## 1: 0.007588242 0.4002180 0.7416538 0.0566808 0.07153553 0.002038241  1
## 2: 0.007565450 0.4002307 0.7416538 0.0566808 0.07153553 0.002038241  1
## 3: 0.007542727 0.4002434 0.7416538 0.0566808 0.07153553 0.002038241  1
## 4: 0.007520072 0.4002561 0.7416538 0.0566808 0.07153553 0.002038241  1
## 5: 0.007497485 0.4002687 0.7416538 0.0566808 0.07153553 0.002038241  1
## 6: 0.007474966 0.4002812 0.7416538 0.0566808 0.07153553 0.002038241  1
##            r5        f.num    s.num    e.num     e.diff
## 1: 0.01584893 2.886389e-08 3.700626 39.66644 -0.3553577
## 2: 0.01584893 2.824153e-08 3.689494 39.66771 -0.3553634
## 3: 0.01584893 2.763259e-08 3.678396 39.66897 -0.3553691
## 4: 0.01584893 2.703678e-08 3.667331 39.67023 -0.3553748
## 5: 0.01584893 2.645381e-08 3.656300 39.67149 -0.3553805
## 6: 0.01584893 2.588342e-08 3.645302 39.67274 -0.3553862
```

```
x <- dat[steps == '10000', .(steps, time, e.diff)]
plot(e.diff ~ time, data = x)
```


e.diff
−0.05
−0.15
−0.25
−0.35
0
50
100
150
time