Test of ALFAM2 closed-form solution

Sasha D. Hafner

09 March, 2024

Overview

The `alfam2()` function uses a closed-form solution to calculate emission over time. In this document that solution is checked by comparison to a simple numerical solution.

Model structure

With a sink for slow pool S, structure is given in Fig. 1.

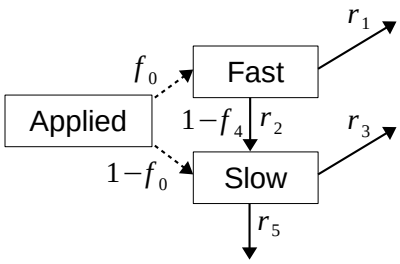


Figure 1: Structure of ALFAM2 model.

So derivatives are:

```
df/dt <-  -r1 * F - r2 * F
ds/dt <- r2 * F - r3 * S - r5 * S
de/dt <- r1 * F + r3 * S
```

Prep

Packages.

```
library(ALFAM2)
```

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

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

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

Predictions

Input data–no predictor variables so only intercept parameters are used.

```
dat <- data.table(time = 0:168, TAN_app = 100)
```

Closed-form solution with `alfam2()` function from ALFAM2 package.

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

```
## 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
```

Numerical solution uses `lsoda()`.

```
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']
```

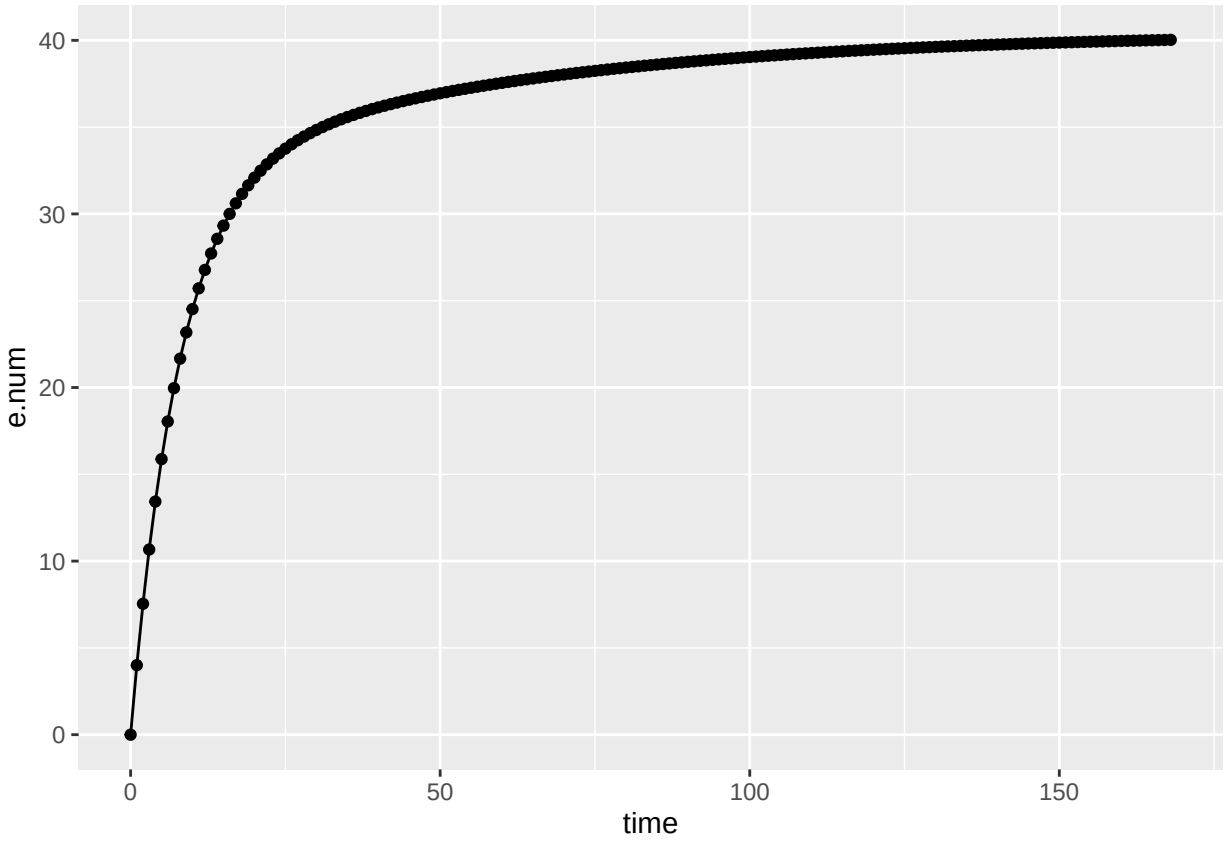
```
  r5 <- parms['r5']

  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)))

}

predsnum <- lsoda(y = y, times = dat[, time], func = rates, parms = pars)
```

Combine results.

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

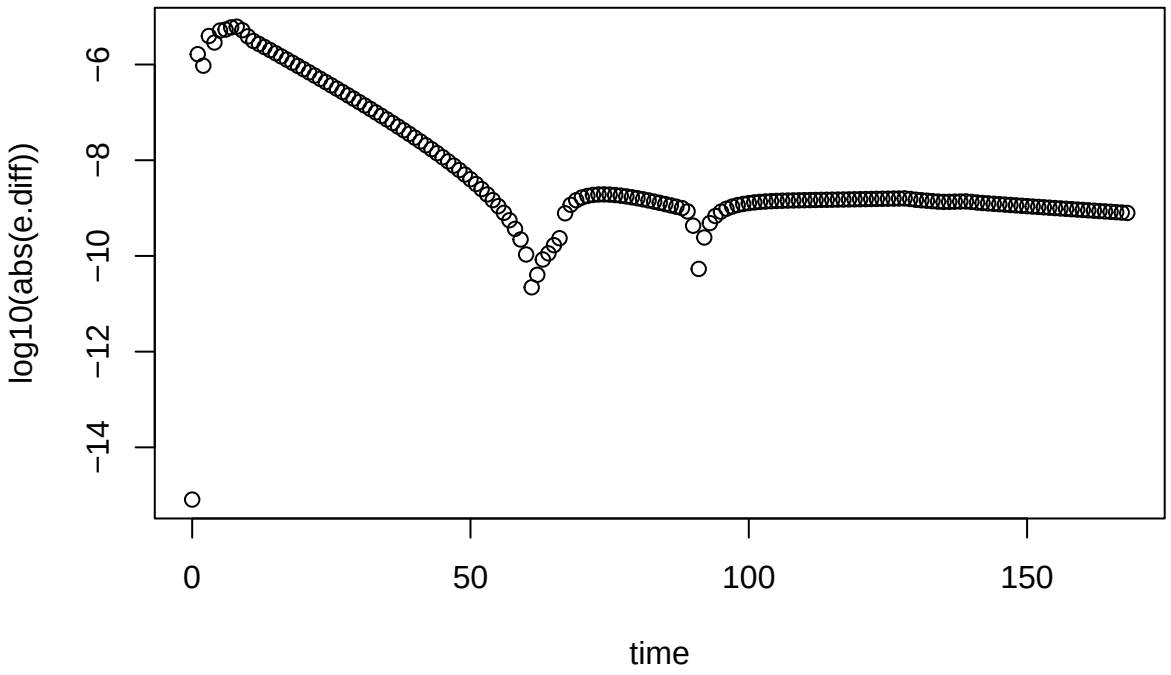
Compare.

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



Check difference–very small.

```
dat[, e.diff := e.num - e.cf]
plot(log10(abs(e.diff)) ~ time, data = dat)
```



Conclusion

The closed-form solution seems to be accurate.