DTU Aqua: Department of Marine Fisheries

http://www.aqua.dtu.dk/

Two examples of adding functionality to AD Model Builder

Anders Nielsen & Casper Berg

an@aqua.dtu.dk

Example 1: Matrix exponential

- The exponential of a square matrix A can be defined (but not implemented) as:

$$e^A = \sum_{i=0}^{\infty} \frac{1}{i!} A^i$$

- It is often used to solve linear ordinary differential equation (ODE) systems:

$$\frac{d}{dt} y(t) = A y(t) \ , \ \text{where } y(0) = y_0$$

- where the solution becomes

$$y(t) = e^{At} y_0$$

Adding the code

- First tested the code by including it in the `GLOBALS_SECTION`, then when happy
- The code itself (next slide) was added to `src/linad99/expm.cpp`
- The following function header was added to `src/linad99/fvar.hpp`

  ```
  dvar_matrix expm(const dvar_matrix & A);
  ```

- The following lines was added to the list `src/linad99/objects.lst`

  ```
  expm.obj \
  ```

- We tried to put the additions near similar functions.
- The algorithm is taken from the paper:

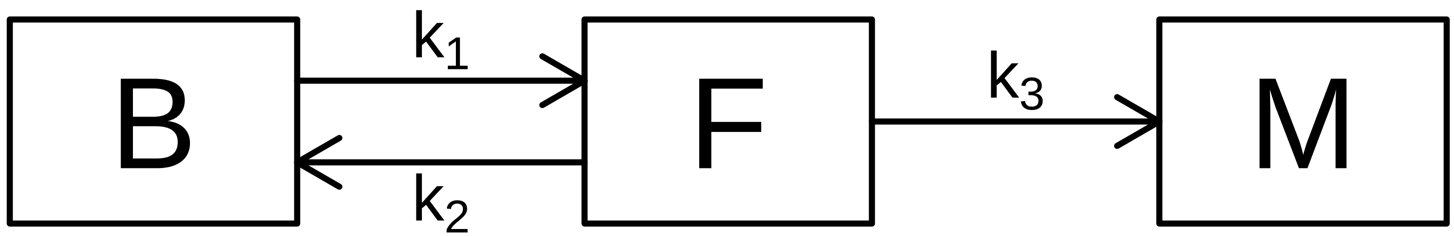
  Moler, Cleve; Van Loan, Charles F. (2003), "Nineteen Dubious Ways to Compute the Exponential of a Matrix, Twenty-Five Years Later"

```
  /**
  \ingroup PDF
   Matrix exponential using (6,6) Pade approximation adapted from Moler and van Loan
  \param A square dvar_matrix
  \returns The matrix exponentiel of A
  */
dvar_matrix expm(const dvar_matrix & A)
{
  RETURN_ARRAYS_INCREMENT();
  int rmin = A.rowmin();
  int rmax = A.rowmax();

  if(rmax != A.colmax()){cout<<"Error: Not square matrix in expm."<<endl; ad_exit(1);}
  if(rmin != A.colmin()){cout<<"Error: Not square matrix in expm."<<endl; ad_exit(1);}

  dvar_matrix I(rmin,rmax,rmin,rmax);
  dvar_matrix AA(rmin,rmax,rmin,rmax);
  dvar_matrix X(rmin,rmax,rmin,rmax);
  dvar_matrix E(rmin,rmax,rmin,rmax);
  dvar_matrix D(rmin,rmax,rmin,rmax);
  dvar_matrix cX(rmin,rmax,rmin,rmax);

  I.initialize();
  for(int i=rmin; i<=rmax; ++i){I(i,i)=1.0;}

  dvariable log2NormInf;
  log2NormInf=log(max(rowsum(fabs(value(A)))));
  log2NormInf/=log(2.0);
  int e = (int)value(log2NormInf) + 1;
```

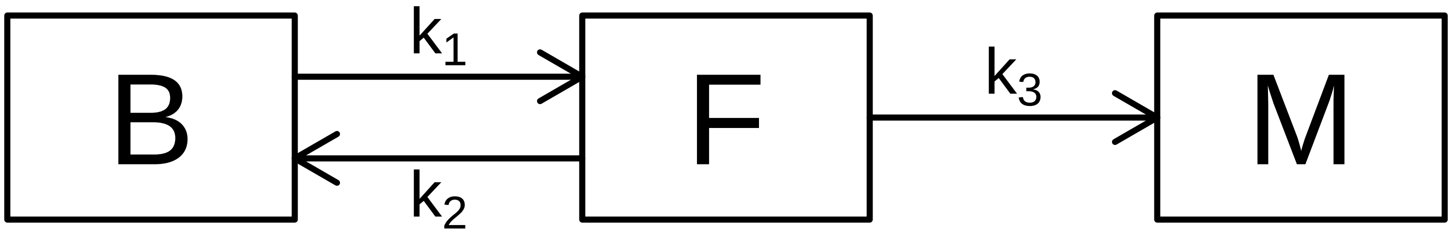
```
  int s = e+1;
  s = (s<0) ? 0 : s;
  AA=1.0/pow(2.0,s)*A;

  X=AA;
  dvariable c=0.5;

  E=I+c*AA;
  D=I-c*AA;
  int q=6, p=1, k;
  for(k=2;  k<=q; ++k){
    c*=((double)q-k+1.0)/((double)k*(2*q-k+1));
    X=AA*X;
    cX=c*X;
    E+=cX;
    if(p==1){D+=cX;}else{D-=cX;}
    p = (p==1) ? 0 : 1;
  }
  //E=inv(D)*E;
  E = solve(D,E);
  for(k=1; k<=s; ++k){
    E=E*E;
  }
  RETURN_ARRAYS_DECREMENT();
  return E;
}
```

Testing case: Terbuthylazine

- It is a herbicide
- Free terbuthylazine can be washed into the drinking water
- It can be bound to the soil
- Certain bacterias can mineralize it



The system



$$
\begin{aligned}
\frac{dB_t}{dt} &= -k_1 B_t + k_2 F_t, & B_0 &= 0\\
\frac{dF_t}{dt} &= k_1 B_t - (k_2 + k_3) F_t, & F_0 &= 100\\
\frac{dM_t}{dt} &= k_3 F_t, & M_0 &= 0
\end{aligned}
$$

Simplifying

- The system is closed, so $M_t = 100 - B_t - F_t$
- Define $X_t = \binom{B_t}{F_t}$
- The simplified system is:

$$\frac{dX_t}{dt} = \underbrace{\begin{pmatrix} -k_1 & k_2 \\ k_1 & -(k_2 + k_3) \end{pmatrix}}_{A} X_t, \qquad X_0 = \begin{pmatrix} 0 \\ 100 \end{pmatrix}$$

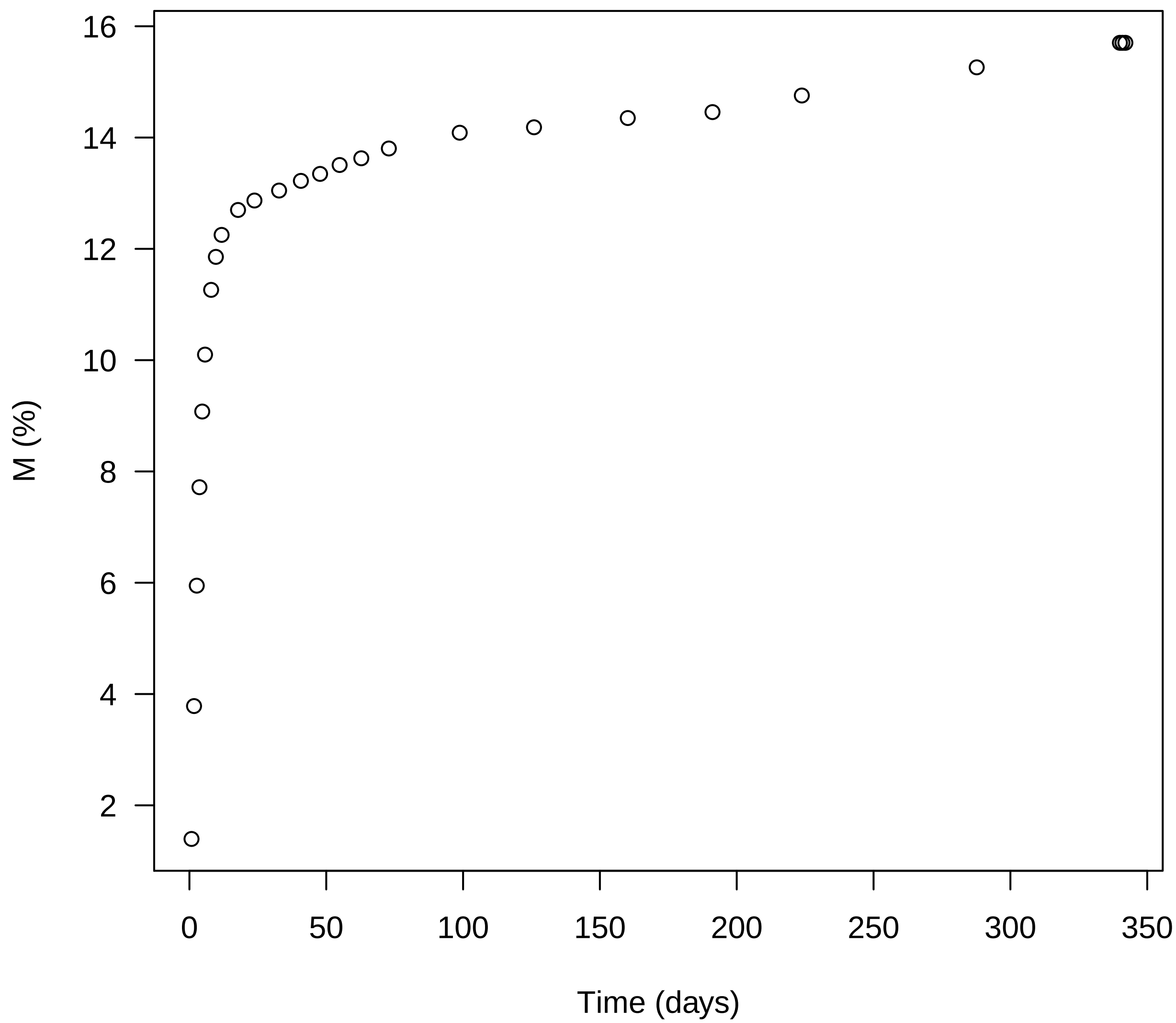
- The system is linear, so it can be be solved for instance via the matrix exponential

$$X_t = e^{At} X_0$$

Observations

- The amount of mineralized terbuthylazine was measured 26 times throughout a year

Time	M
0.77	1.396
1.69	3.784
2.69	5.948
3.67	7.717
4.69	9.077
5.71	10.100
7.94	11.263
9.67	11.856
11.77	12.251
17.77	12.699
23.77	12.869
32.77	13.048
40.73	13.222
47.75	13.347
54.90	13.507
62.81	13.628
72.88	13.804
98.77	14.087
125.92	14.185
160.19	14.351
191.15	14.458
223.78	14.756
287.70	15.262
340.01	15.703
340.95	15.703
342.01	15.703



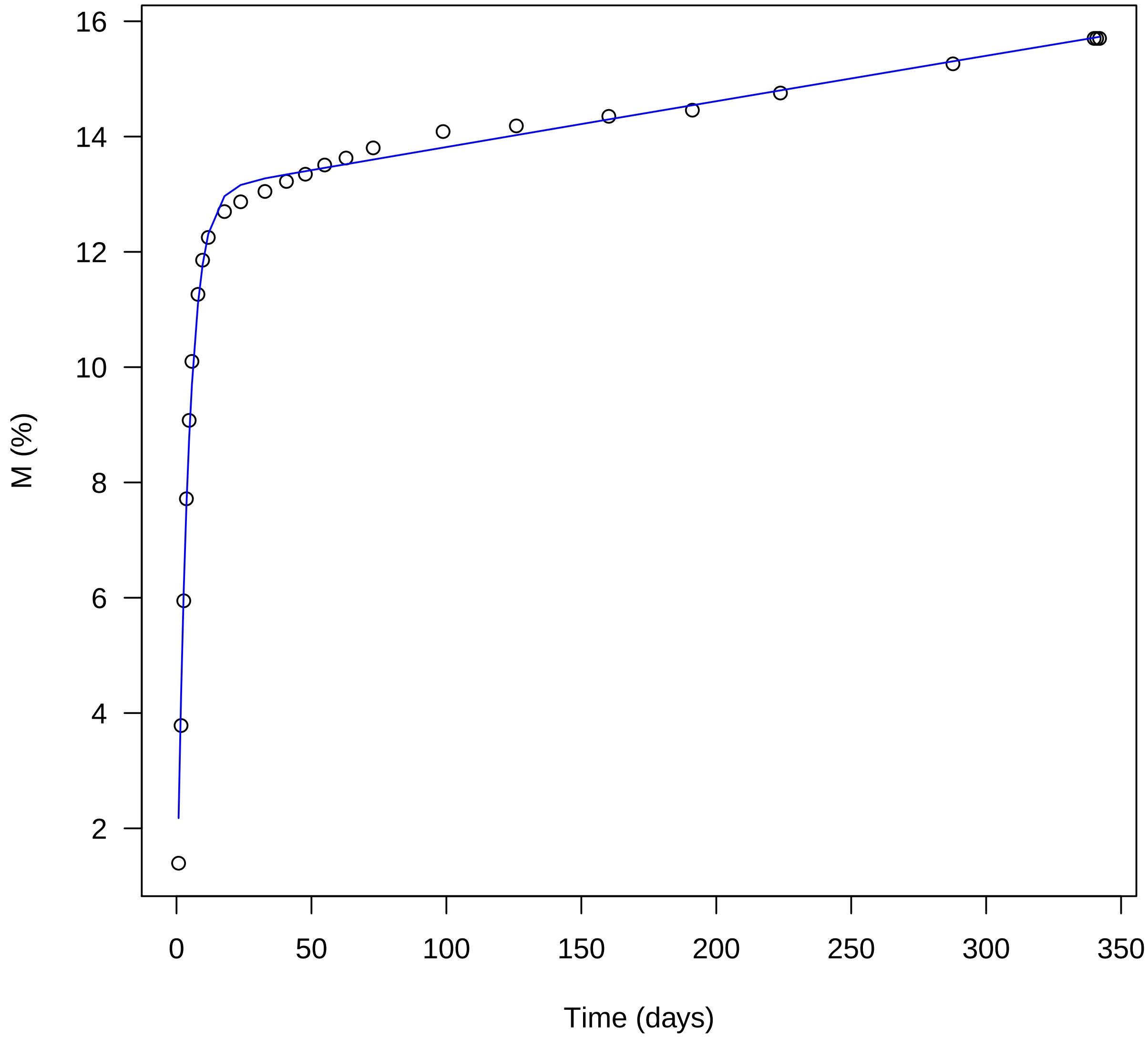
Simplest statistical model

$$M_{t_i} \sim \mathcal{N}(100 - \Sigma X_{t_i}, \sigma^2), \quad \text{independent, and with } X_{t_i} = e^{At_i} X_0.$$

AD Model Builder implementation

```
DATA_SECTION
  init_int noObs
  init_matrix obs(1,noObs,1,2)
  vector X0(1,2)

PARAMETER_SECTION
  init_vector logK(1,3);
  init_number logSigma;

  sdreport_vector k(1,3);
  sdreport_number sigma2;
  sdreport_vector M(1,noObs);

  matrix X(1,noObs,1,2);
  matrix A(1,2,1,2);
  objective_function_value nll;

PRELIMINARY_CALCS_SECTION
  X0(1)=0.0; X0(2)=100.0;
  logK=-2.0;
  logSigma=-2.0;

PROCEDURE_SECTION
  k=exp(logK);
  sigma2=exp(2.0*logSigma);

  A(1,1)= -k(1); A(1,2)=  k(2);
  A(2,1)=  k(1); A(2,2)= -k(2)-k(3);

  for(int i=1; i<=noObs; ++i){
    X(i)=expm(A*obs(i,1))*X0;
    M(i)=100.0-sum(X(i));
    nll+=0.5*(log(2.0*M_PI*sigma2)+square((obs(i,2)-M(i)))/sigma2);
  }
```

Simple model fit



Runtime was <0.8s on old laptop including standard deviation calculations.

Model with covariance

- Residuals show that observations are not independent
- A model accounting for this could be

$$M_{t_i} = 100 - \Sigma X_{t_i} + \eta_i,$$

where η followers a multivariate normal distribution with mean vector 0 and covariance matrix S.

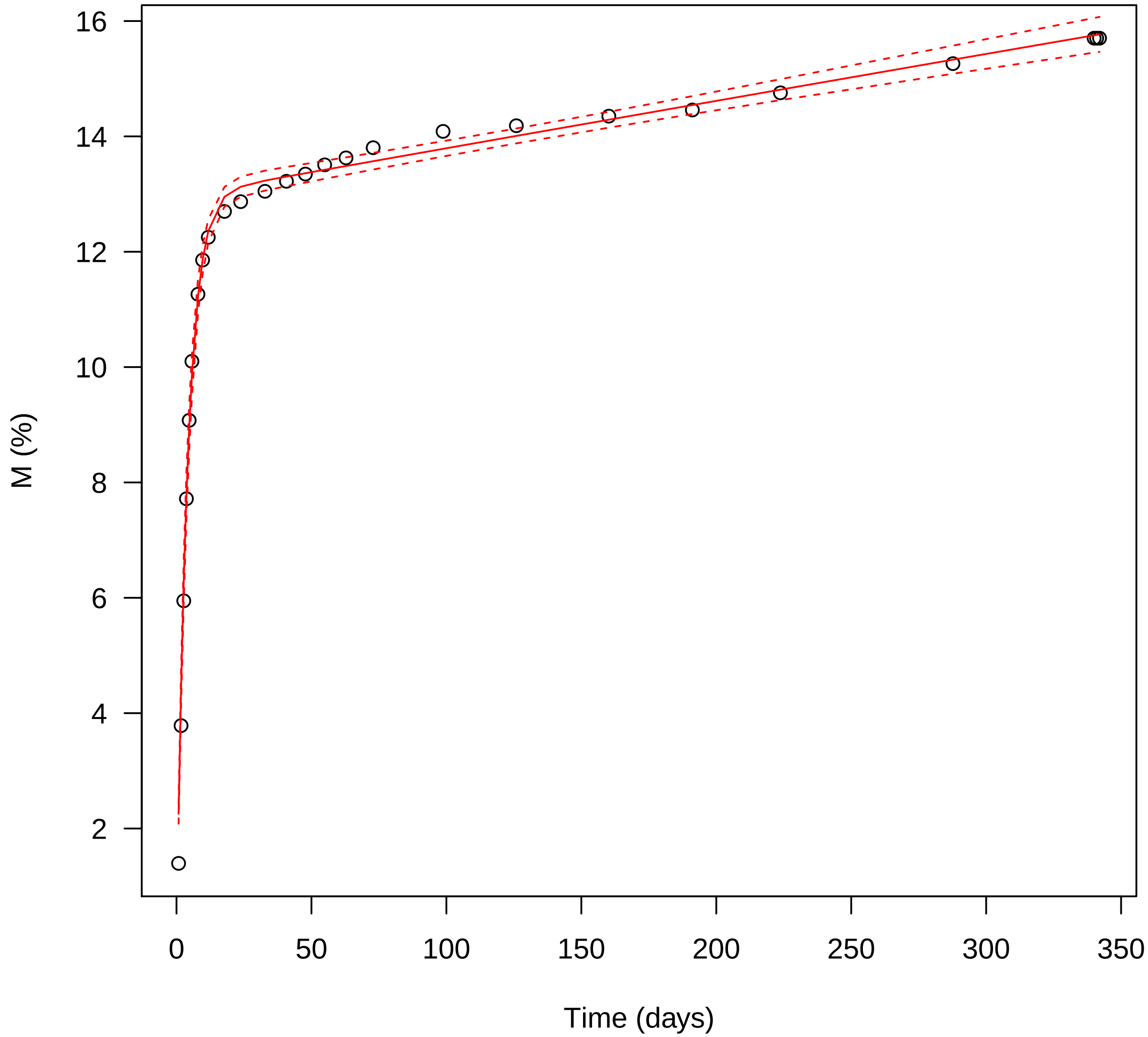
$$\eta \sim \mathcal{N}(0, S), \text{ where } S_{i,j} = \begin{cases} \tau^2 \exp(-(t_i - t_j)^2/\rho^2), & \text{if } i \neq j \\ \tau^2 \exp(-(t_i - t_j)^2/\rho^2) + \sigma^2, & \text{if } i = j \end{cases}$$

AD Model Builder implementation

```
DATA_SECTION
  init_int noObs
  init_matrix obs(1,noObs,1,2)
  vector X0(1,2)

PARAMETER_SECTION
  init_vector logK(1,3);
  init_number logSigma;
  init_number logTau(2);
  init_number logRho(2);

  sdreport_vector k(1,3);
  sdreport_number sigma2;
  sdreport_number tau2;
  sdreport_number rho2;
  sdreport_vector M(1,noObs);

  matrix X(1,noObs,1,2);
  matrix A(1,2,1,2);
  matrix S(1,noObs,1,noObs);
  sdreport_vector Mres(1,noObs);
  objective_function_value nll;

PRELIMINARY_CALCS_SECTION
  X0(1)=0.0; X0(2)=100.0;
  logK=-2.0;
  logSigma=-2;
  logRho=2;
  logTau=-1;

```

```
PROCEDURE_SECTION
  k=exp(logK);
  sigma2=exp(2.0*logSigma);
  tau2=exp(2.0*logTau);
  rho2=exp(2.0*logRho);

  A(1,1)= -k(1); A(1,2)=  k(2);
  A(2,1)=  k(1); A(2,2)= -k(2)-k(3);

  S.initialize();
  for(int i=1; i<=noObs; ++i){
    for(int j=i; j<=noObs; ++j){
      S(i,j)=tau2*exp(-square(obs(i,1)-obs(j,1))/rho2);
      S(j,i)=S(i,j);
    }
    S(i,i)+=sigma2;
  }

  for(int i=1; i<=noObs; ++i){
    X(i)=expm(A*obs(i,1))*X0;
    M(i)=100.0-sum(X(i));
    Mres(i)=obs(i,2)-M(i);
  }

  nll=0.5*(log(2.0*M_PI)*noObs+log(det(S))+Mres*inv(S)*Mres);

REPORT_SECTION
  for(int i=1; i<=noObs; ++i){
    report<<obs(i,1)<<" "<<obs(i,2)<<" "<<X(i,1)<<" "<<X(i,2)<<" "<<M(i)<<endl;
  }
```

Fit of covariance model



Runtime was <1.3s on old laptop including standard deviation calculations.

Estimates

- First for the simple model ($-\log L = 0.939214$):

Parameter	Estimate	Standard deviation
k_1	0.000710	0.00005023
k_2	0.20547	0.0055721
k_3	0.030929	0.00071481
σ^2	0.062936	0.017455

- The for the model with covariance ($-\log L = -10.1038$):

Parameter	Estimate	Standard deviation
k_1	0.000737	0.0000616
k_2	0.21474	0.0097399
k_3	0.032177	0.0013614
σ^2	0.000448	0.0003619
τ^2	0.062581	0.020699
ρ^2	13.934	4.2844

The simple one in R

```
library(Matrix)

dat<-read.table('min.dat', skip=3, header=FALSE)

nlogL<-function(theta){
  k<-exp(theta[1:3])
  sigma<-exp(theta[4])
  A<-rbind(
       c(-k[1], k[2]),
       c( k[1], -(k[2]+k[3]))
     )
  x0<-c(0,100)
  sol<-function(t)100-sum(expm(A*t)%*%x0)
  pred<-sapply(dat[,1],sol)
  -sum(dnorm(dat[,2],mean=pred,sd=sigma, log=TRUE))
}

fit<-optim(c(-2,-2,-2,-2),nlogL,hessian=TRUE)

#> system.time(fit<-optim(c(-2,-2,-2,-2),nlogL,hessian=TRUE))
#   user  system elapsed
# 30.334   0.008  30.452
#> fit$value
#[1] 19.26905
#> fit$convergence
#[1] 0

#> est<-c(-7.24961, -1.58246, -3.47606, -1.38281491705)
#> system.time(fit<-optim(est,nlogL,hessian=TRUE))
#   user  system elapsed
# 22.582   0.000  22.650
#> fit$par
#[1] -7.249613 -1.582464 -3.476065 -1.382820
#> fit$value
#[1] 0.9392142
#> fit$convergence
#[1] 0
```

What is Expm still missing

- Version for random effects (we have the code)
- Test case included in automatic testing procedure.
- Description in manual
- ...

Example 2: init_table (modifying the flex scripts)

- Very often (outside fisheries) all data can be organized as a simple matrix
- To read a data matrix into R, we would write:

```
dat<-read.table('ex.dat')
```

- To read a data matrix into AD Model Builder, we would write:

```
DATA_SECTION
  init_int N
  init_matrix dat(1,N,1,2)
  vector x(1,N)
  !! x=column(dat,1);
  vector y(1,N)
  !! y=column(dat,2);
```

  (after we counted the number of lines and put that in the beginning of the file)

- We are all used to that, but new users might find that strange
- And when you think of this there is actually quite a bit to explain here

- Some years ago I noticed (in the autodif manual) that autodif can actually read a matrix without knowing the number of lines.
- So I have sometimes been reading data matrices with:

```
DATA_SECTION
  matrix A(0,-1,0,-1)
  !! dmatrix Atmp((adstring)"ex1.dat");
  !! A=Atmp;
```

- Here we will try to use that to get something like:

```
DATA_SECTION
  init_table A("ex1.dat")
```

- Or simply the following to read from the default input file

```
DATA_SECTION
  init_table A
```

- Unfortunately we have to mess with the flex scripts to do that (look at `src/nh99/tpl2cpp.lex`)
- Thanks to Dave, John, and Johnoel for answering many questions!

Compiling with flex changes

- In the long run it should just be

```
make
```

- Currently it is not, but more something like:

```
cd /home/an/ADMB/admb-trunk/trunk
rm src/nh99/tpl2cpp.c
make --directory=src/nh99 --file=optg32-rh8-laplace.mak tpl2cpp.c
rm src/df1b2-separable/tpl2rem.c
make --directory=src/df1b2-separable --file=optg32-rh8-laplace.mak tpl2r
make clean
make
```

- (look at the test file)

What is init_table still missing

- Would be nice with files with headers
- Instead of using dmatrix we could set up a class more like R's data.frame
- In any case it was interesting for me to look at the flex scripts.

Mineralization example with init_table

```
DATA_SECTION
  init_table obs
  vector X0(1,2)

PARAMETER_SECTION
  init_vector logK(1,3);
  init_number logSigma;

  sdreport_vector k(1,3);
  sdreport_number sigma2;
  number pred;
  matrix A(1,2,1,2);
  objective_function_value nll;

PRELIMINARY_CALCS_SECTION
  X0(1)=0.0; X0(2)=100.0;
  logK=-2.0;
  logSigma=-2.0;

PROCEDURE_SECTION
  k=exp(logK);
  sigma2=exp(2.0*logSigma);

  A(1,1)= -k(1); A(1,2)=  k(2);
  A(2,1)=  k(1); A(2,2)= -k(2)-k(3);

  for(int i=1; i<=obs.rowmax(); ++i){
    pred=100.0-sum(expm(A*obs(i,1))*X0);
    nll+=0.5*(log(2.0*M_PI*sigma2)+square(obs(i,2)-pred)/sigma2);
  }
```