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) Padé 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

\[
\begin{align*}
\frac{dB_t}{dt} &= -k_1B_t + k_2F_t, & B_0 &= 0 \\
\frac{dF_t}{dt} &= k_1B_t - (k_2 + k_3)F_t, & F_0 &= 100 \\
\frac{dM_t}{dt} &= k_3F_t, & M_0 &= 0
\end{align*}
\]
Simplifying

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

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

- The system is linear, so it can 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.

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

\[ M_{t_i} \sim \mathcal{N}(100 - \sum X_{t_i}, \sigma^2), \quad \text{independent, and with } X_{t_i} = e^{A_{t_i}} X_0. \]
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

\[ M_{t_i} = 100 - \sum X_{t_i} + \eta_i, \]

where \( \eta \) follows a multivariate normal distribution with mean vector 0 and covariance matrix \( S \).

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

\[
\begin{align*}
    k &= \exp(\log K) \\
    \sigma_2 &= \exp(2.0 \cdot \log \Sigma) \\
    \tau_2 &= \exp(2.0 \cdot \log \ Tau) \\
    \rho_2 &= \exp(2.0 \cdot \log \ Rho) \\
    A(1,1) &= -k(1); A(1,2) = k(2); \\
    A(2,1) &= k(1); A(2,2) = -k(2) - k(3); \\
    S.\initialize(); \\
    \text{for (int } i = 1; i <= \text{noObs; ++i}) { \\
        \text{for (int } j = i; j <= \text{noObs; ++j}) { \\
            S(i,j) &= \tau_2 \cdot \exp(-\text{square}(\text{obs}(i,1) - \text{obs}(j,1)) / \rho_2); \\
            S(j,i) &= S(i,j); \\
        } \\
        S(i,i) &= S(i,i) + \sigma_2; \\
    } \\
    \text{for (int } i = 1; i <= \text{noObs; ++i}) { \\
        X(i) &= \expm(A \cdot \text{obs}(i,1)) \times X0; \\
        M(i) &= 100.0 - \text{sum}(X(i))); \\
        Mres(i) &= \text{obs}(i,2) - M(i); \\
    } \\
    nll &= 0.5 \cdot (\log(2.0 \cdot M_\PI) \cdot \text{noObs} + \log(\det(S)) + Mres \cdot \text{inv}(S) \cdot Mres); \\
\end{align*}
\]

REPORTSECTION

\[
\begin{align*}
    \text{for (int } i = 1; i <= \text{noObs; ++i}) { \\
        \text{report} \ll \text{obs}(i,1) \ll \text{obs}(i,2) \ll \text{X(i,1)} \ll \text{X(i,2)} \ll M(i) \ll \text{endl;} \\
    } \\
\end{align*}
\]
Fit of covariance model

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

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Estimate</th>
<th>Standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_1$</td>
<td>0.000710</td>
<td>0.00005023</td>
</tr>
<tr>
<td>$k_2$</td>
<td>0.20547</td>
<td>0.0055721</td>
</tr>
<tr>
<td>$k_3$</td>
<td>0.030929</td>
<td>0.00071481</td>
</tr>
<tr>
<td>$\sigma^2$</td>
<td>0.062936</td>
<td>0.017455</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Estimate</th>
<th>Standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_1$</td>
<td>0.000737</td>
<td>0.0000616</td>
</tr>
<tr>
<td>$k_2$</td>
<td>0.21474</td>
<td>0.0097399</td>
</tr>
<tr>
<td>$k_3$</td>
<td>0.032177</td>
<td>0.0013614</td>
</tr>
<tr>
<td>$\sigma^2$</td>
<td>0.000448</td>
<td>0.0003619</td>
</tr>
<tr>
<td>$\tau^2$</td>
<td>0.062581</td>
<td>0.020699</td>
</tr>
<tr>
<td>$\rho^2$</td>
<td>13.934</td>
<td>4.2844</td>
</tr>
</tbody>
</table>
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:
  
  ```r
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:

\begin{verbatim}
DATA_SECTION
    matrix A(0,-1,0,-1)
    ! dmatrix Atmp((adstring)"ex1.dat");
    ! A=Atmp;
\end{verbatim}

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

\begin{verbatim}
DATA_SECTION
    init_table A("ex1.dat")
\end{verbatim}

Or simply the following to read from the default input file

\begin{verbatim}
DATA_SECTION
    init_table A
\end{verbatim}

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 tpl2rem.c
  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);
  }