1 Introduction

The \texttt{expm} package provides an \texttt{R} function \texttt{expm} to compute the matrix exponential of a real, square matrix. The matrix exponential of a matrix $A$ is defined as

$$e^A = I + A + \frac{A^2}{2!} + \ldots = \sum_{k=0}^{\infty} \frac{A^k}{k!}.$$ 

The actual computations are done in \texttt{C} by a function of the same name that is callable by other packages. Therefore, package authors can use these functions and avoid duplication of efforts.

2 Description of the functions

The \texttt{R} function \texttt{expm} takes as argument a real, square matrix and returns its exponential. Dimension names are preserved:

```r
> library(expm)
> m <- matrix(c(4, 1, 1, 2, 4, 1, 0, 1, 4), 3, 3)
> expm(m)
[,1]       [,2]       [,3]
[1,] 147.8666 183.7651  71.79703
[2,] 127.7811 183.7651  91.88257
[3,] 127.7811 163.6796 111.96811

> dimnames(m) <- list(letters[1:3], LETTERS[1:3])
> m
```

1
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>c</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

```
> expm(m)
```

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>A 147.8666</td>
<td>183.7651</td>
<td>71.79703</td>
</tr>
<tr>
<td>B 127.7811</td>
<td>183.7651</td>
<td>91.88257</td>
</tr>
<tr>
<td>C 127.7811</td>
<td>163.6796</td>
<td>111.96811</td>
</tr>
</tbody>
</table>

Note that the remainder of this text mainly relates to `expm(. , method = "Ward77"`, i.e., the method of Ward (1977) which is no longer the default method, as e.g., `method = "Higham08"` has found to be ("uniformly") superior, see Higham (2008).

The actual computational work is done in C by a routine defined as

```c
void expm(double *x, int n, double *z)
```

where `x` is the vector underlying the R matrix and `n` is the number of lines (or columns) of the matrix. The matrix exponential is returned in `z`. The routine uses the algorithm of Ward (1977) based on diagonal Padé table approximations in conjunction with three step preconditioning. The Padé approximation to $e^A$ is

$$e^A \approx R(A),$$

with

$$R_{pq}(A) = (D_{pq}(A))^{-1} N_{pq}(A)$$

where

$$D_{pq}(A) = \sum_{j=1}^{p} \frac{(p + q - j)jq!}{(p + q)!j!(p - j)!} A^j$$

and

$$N_{pq}(A) = \sum_{j=1}^{q} \frac{(p + q - j)jq!}{(p + q)!j!(q - j)!} A^j.$$ 

See Moler and Van Loan (1978) for an exhaustive treatment of the subject.

The C routine is based on a translation made by ?. of the implementation of the corresponding Octave function (Eaton, 2002).

2
3 Calling the functions from other packages

Package authors can use facilities from \texttt{expm} in two (possibly simultaneous) ways:

1. call the \texttt{R} level function \texttt{expm} in \texttt{R} code;
2. if matrix exponential calculations are needed in \texttt{C}, call the routine \texttt{expm}.

Using \texttt{R} level function \texttt{expm} in a package simply requires the following two import directives:

\texttt{Imports: expm}

in file \texttt{DESCRIPTION} and

\texttt{import(expm)}

in file \texttt{NAMESPACE}.

Accessing the \texttt{C} level routine further requires to prototype \texttt{expm} and to retrieve its pointer in the package initialization function \texttt{R\_init\_pkg}, where \texttt{pkg} is the name of the package:

\begin{verbatim}
void (*expm)(double *x, int n, double *z);

void R_init_pkg(DllInfo *dll)
{
    expm = (void (*)(double, int, double))
              R_GetCCallable("expm", "expm");
}
\end{verbatim}

The definitive reference for these matters remains the \textit{Writing R Extensions} manual.

References


