ADR Features

Johannes Willkomm

March 10, 2020

Contents

1 ADR Features

2 Automatic Differentiation for R

2.1 Installing

As of yet, the ADR package is not on CRAN, so you have to download and install the tarball from `https://r-adr.de/download/` and install manually using either the console command

```
R CMD INSTALL adr_1.0.137.tar.gz
```

or, start an R session and type

```
install.packages('adr_1.0.137.tar.gz')
```

2.2 Running

After starting a new R session, you can load the ADR package with the `library` function:

```
library('adr')
```

```
f <- function(x) { if (x > 2) 2*x else x }
x <- 27
f(x)
```

```
[1] 54
```

```
df <- d(f)
```

```
ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 53 chars
ADR: HTTR response after 0.642 s, 1074 chars
```

```
df(1, x)
```

```
$df
[1] 2

$f
[1] 54
```

```
df(eye(2), 27 * eye(2))
```

```
$df
[1] 2

$f
[1] 54
```

```
d(function(x) sqrt(x))(1, x)
```

```
ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 20 chars
ADR: HTTR response after 0.873 s, 933 chars
$df
[1] 0.09622504

$f
[1] 5.196152
```

```
d(function(x) x^(1/17))(1, x)
```

```
ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 21 chars
ADR: HTTR response after 0.345 s, 1009 chars
$df
[1] 0.002644753

$f
[1] 1.213942
```

```
options = adrOptions(i = 1)
adrDiffFor(f, list(x), options=options)
```

```
ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 53 chars
ADR: HTTR response after 0.396 s, 1074 chars
$J
     [,1]
[1,]    2

$f
[1] 54
```

```
options = adrOptions(i = 1)
adrDiffFor(function(x) x^3, list(2), options=options)
```

```
ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 16 chars
ADR: HTTR response after 0.542 s, 903 chars
$J
     [,1]
[1,]   12

$f
[1] 8
```

```
options = adrOptions(i = 1)
adrDiffFor(function(x) x^3, list(2), options=options)
```

```
$J
     [,1]
[1,]   12

$f
[1] 8
```

2.3 Getting help

See the ADR help index with the R command `help`

```
help(package='adr')
```

Get help for individual ADR functions with the R command `help`

```
help(adrDiffFor)
```

See the ADR package vignettes with this R code:

```
vignette(package='adr')
```

This current ADR package vignette can be opened with this R code:

```
vignette('adr-features')
```

See the other ADR package vignettes with this R code:

```
vignette('adr')
vignette('adr-install')
```

3 ADR Usage

3.1 Basic features

You can differentiate w.r.t. one or more input parameters. By default ADR differentiates w.r.t. the first parameter only:

```
adrDiffFor(function(x, y) x * y, list(2, 3))
```

```
ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 21 chars
ADR: HTTR response after 0.469 s, 906 chars
$J
     [,1]
[1,]    3

$f
[1] 6
```

Using the `independents` field of the adrOptions structure you can choose another set of the independent variables:

```
adrDiffFor(function(x, y) x * y, list(2, 3), options=
   adrOptions(independents = 2))
```

```
ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 21 chars
ADR: HTTR response after 0.434 s, 906 chars
$J
```

```
     [,1]
[1,]    2

$f
[1] 6
```

Obviously you can also differentiate w.r.t. several inputs. The `independents` field can also be abbreviated with `i` when calling `adrOptions`:

```
adrDiffFor(function(x, y) x * y, list(2, 3), options=
    adrOptions(i = 1:2))
```

```
ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 21 chars
ADR: HTTR response after 0.462 s, 942 chars
$J
     [,1] [,2]
[1,]    3    2

$f
[1] 6
```

The order of the independents does not matter:

```
adrDiffFor(function(x, y) x * y, list(2, 3), options=
    adrOptions(i = c(2, 1)))
```

```
ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 21 chars
ADR: HTTR response after 0.482 s, 942 chars
$J
     [,1] [,2]
[1,]    3    2

$f
[1] 6
```

3.2 Source code differentiation

ADR works by *source transformation*, using a *transpiler* provided by a remote server. When a function `f` is differentiated, ADR will first try to find a symbol `dv_f`. Failing that it will try to obtain the source code of `f` and send it to the ADR server to differentiate the code.

The main function to perform such source code differentiation for R is simply called `d`.

```
f <- function(x, y) sin(x*y)*atan(x)
d(f)
```

```
ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 36 chars
ADR: HTTR response after 0.633 s, 1434 chars
function (f.p1, f.p0)
{
    x = f.p0[[1]]
    y = f.p0[[2]]
    dv_x = f.p1[[1]]
    dv_y = f.p1[[2]]
    f.ra5 = df_atan(list(dv_x), list(x))
    d_f.ca4 = f.ra5[[1]]
    f.ca4 = f.ra5[[2]]
    d_f.ca3 = dv_x * y + x * dv_y
    f.ca3 = x * y
    f.ra5 = df_sin(list(d_f.ca3), list(f.ca3))
    d_f.ca2 = f.ra5[[1]]
    f.ca2 = f.ra5[[2]]
    d_f.ca1 = d_f.ca2 * f.ca4 + f.ca2 * d_f.ca4
    f.ca1 = f.ca2 * f.ca4
    d_f.ridm16 <- d_f.ca1
    f.ridm16 <- f.ca1
    list(df = d_f.ridm16, f = f.ridm16)
}
```

When `f` is a simple function there is no problem to obtain the source code. When `f` is a method however, you must first use `getS3method` or `getMethod` to find the appropriate function given the class name.

```
library('mcmc')
d(getS3method('metrop', 'function'))
```

3.3 Derivative definitions and overrides

ADR will generally differentiate a function call `f` to a call to a function called `dv_f`. In some cases this differentiation of the call is done on the server

already. In most cases however, a call to `f` is generically differentiated to a call to the runtime function `dcall`. This function will in turn use `dgetfun` which will then dynamically

1. look for an existing derivative function *usually* called `dv_f`
2. otherwise, obtain the source code of `f` and send it to the ADR server and thus obtain `dv_f`

If step 1. `dgetfun` will construct a so called *code* for the differentiated function, depending on the current set of independents, i.e. the set of arguments that are active in terms of differentiation. When all formals of `f` are active the code is `dv_f`. When at least one argument is not active the code will be `dv_f.act1_act2_..._actn` when `act1`, `act2`, ..., `actn` etc are the indices of the active arguments. For example, `qr.qy` is differentiated to one of `qr.qy`, `qr.qy.1`, or `qr.qy.2`.

`dgetfun` then tries to find this code in the current environment. When the search succeeds, this symbol is returned. Otherwise the function `f` must be differentiated in source code by `d`.

This means that you can always override any existing derivative rule in ADR by providing you own versions of these derivative functions.

3.3.1 Provide derivatives of internals

Internal functions in R are usually part of the C runtime of R, or are otherwise functions written in native code that are interfaced to from R. These function obviously cannot be differentiated by ADR and hence you must provide a derivative function for each of these internals that your code uses. Ideally ADR will have rules for all internals of the R code packages one day, and hopefully also for many other packages functions written in native code. Currently this set is rather limited however. Please provide feedback on any internals from core packages lacking support by ADR that you encounter.

As an example consider the function `sqrt` and imagine for a moment that ADR does not support it yet.

```
f <- function(x) { sqrt(x) }
f(4)
```

```
[1] 2
```

ADR would in this case print an error message stating that `dv_sqrt` was not found. Hence you must provide that function by yourself.

```
dv_sqrt <- function(dv_x, x) {
    z <- sqrt(x)
    dz <- 0.5 * dv_x / z
    print('I know the derivative of sqrt!')
    list(dz, z)
}
adrDiffFor(f, list(4))
```

```
ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 26 chars
ADR: HTTR response after 0.505 s, 933 chars
$J
     [,1]
[1,] 0.25

$f
[1] 2
```

The important things to know about the derivative functions are the following:

- When the signature of `f` contains a triple-dot then `dv_f` must have the signature `dv_f(dx, x, indeps)` where
  - `dx` will receive a list of derivative arguments
  - `x` will receive the list of function arguments
  - `indeps` will receive an integer list of the active arguments
  - the names of the parameters of `dv_f` not matter
- When the signature of `f` does not have a triple-dot then for `f(a,b,c)` and all active arguments the derivative function is called `dv_f` and must have the signature `dv_f(dv_a, dv_b, dv_c, a, b, c)`, where
  - The derivative parameters `dv_a, dv_b, dv_c` come first in the list
    - The names must have the prefix `dv_` followed by the name of the parameters they stand for

    * When some parameters do not have a derivative, then these can be omitted, e.g. you may provide `dv_f(dv_a, a, b, c)` when `b` and `c` do not have a derivative
  - Then follow the function parameters which must have the same names as the parameters of `f`

3.3.2 Hierarchical AD

Another powerful feature of this is that you can easily overrule ADR and provide an algebraic derivative for any function in your code.

```
babylonian <- function(x, y=1) {
    if (abs(y^2 - x) < 1e-14) {
        cat(';')
        y
    } else {
        cat('.')
        babylonian(x, (y + x / y) / 2)
    }
}
g <- function(x) { babylonian(x) }
f <- function(x, y) { g(x)*g(y) }
f(4, 9)
```

```
.....;......;[1] 6
```

For example, imagine that you know `babylonian` will take a long time to compute because it is an iterative alrogithm. In the output above each dot printed corresponds to one call of `babylonian`, or one iteration of the algorithm. You can differentiate through this algorithm with ADR, but that will be rather expensive:

```
adrDiffFor(f, list(4,9), options = adrOptions(i=1:2))
```

```
ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 33 chars
ADR: HTTR response after 0.469 s, 1298 chars
ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 32 chars
ADR: HTTR response after 0.519 s, 943 chars
ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 148 chars
ADR: HTTR response after 0.649 s, 1709 chars
```

```
.ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 148 chars
ADR: HTTR response after 0.734 s, 1729 chars
.....;.....;......;.....;$J
     [,1]      [,2]
[1,] 0.75 0.3333333

$f
[1] 6
```

Note that every dot printed here corresponds to one run of the differentiated code `dv_babylonian`, which will take at least three times as long as `babylonian` itself.

You also know however, that `g` will compute the square root $z = \sqrt{x}$, via the Babylonian algorithm, and that the derivative of that is just $1/(2z)$. Hence you can provide a differentiated version `dv_g` of `g`, which does not have to differentiate through the expensive algorithm.

```
dv_g <- function(dv_x, x) {
    z <- babylonian(x)
    dz <- dv_x / (2*z)
    print('I know the derivative of g!')
    list(dz, z)
}
adrDiffFor(f, list(4,9), options = adrOptions(i=1:2))
```

```
......;.....;......;.....;$J
     [,1]      [,2]
[1,] 0.75 0.3333333

$f
[1] 6
```

Here, every dot printed corresponds to one run of `babylonian` again, as it is run from your function `dv_g` to obtain the necessary function result, while the derivative is then computeted algebraically.

When you switch to vector mode then `babylonian` will even be run only as often as by the function `f` itself.

```
adrDiffFor(f, list(4,9), options = adrOptions(i=1:2,
    vectormode=TRUE))
```

```
......;.....;$J
     [,1]      [,2]
[1,] 0.75 0.3333333

$f
[1] 6
```

4 Supported language features

4.1 Control structures

4.1.1 If

4.1.2 While

4.1.3 For

1. Active loop variables

   Active loop variables are supported. This may occur when the iteration range is some numerical vector. The derivative iteration range is computed and the derivative loop variable provided with its values.

```
f <- function(x) {
    z <- 0
    s <- sqrt(x)
    for (k in s:100) {
        z <- z + k
    }
    z
}
d(f)
```

```
ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 91 chars
ADR: HTTR response after 0.681 s, 1534 chars
function (f.p1, f.p0)
{
    x = f.p0[[1]]
    dv_x = f.p1[[1]]
    z <- 0
    dv_z <- 0
```

```
        f.ra5 <- df_sqrt(list(dv_x), list(x))
        dv_s = f.ra5[[1]]
        s = f.ra5[[2]]
        f.reaidm48 = 100
        dv_range_idm124 = dop_colon_right(dv_s, s, f.reaidm48)
        range_idm124 = s:f.reaidm48
        for (adr_i_idm124 in 1:length(range_idm124)) {
            dv_k = dv_range_idm124[[adr_i_idm124]]
            k = range_idm124[[adr_i_idm124]]
            d_f.ca1 = dv_z
            f.ca1 = z
            dv_z <- d_f.ca1 + dv_k
            z <- f.ca1 + k
        }
        list(df = dv_z, f = z)
    }
```

4.2 Functions and calls

4.2.1 Nested functions

ADR will differentiate function definitions when it sees them.

```
f <- function(a,b,c,d,e) {
    s <- function(x,y,z) {
        7*x+11*y+13*z
    }
    1*a+2*b+3*s(c,d,e)
}
f(1,2,3,4,5)
```

```
[1] 395
```

```
d(f)
```

```
ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 125 chars
ADR: HTTR response after 0.828 s, 2144 chars
function (f.p1, f.p0)
{
```

```
    a = f.p0[[1]]
    b = f.p0[[2]]
    c = f.p0[[3]]
    d = f.p0[[4]]
    e = f.p0[[5]]
    dv_a = f.p1[[1]]
    dv_b = f.p1[[2]]
    dv_c = f.p1[[3]]
    dv_d = f.p1[[4]]
    dv_e = f.p1[[5]]
    dv_s <- function(f.pa1, f.pa0) {
        x = f.pa0[[1]]
        y = f.pa0[[2]]
        z = f.pa0[[3]]
        dv_x = f.pa1[[1]]
        dv_y = f.pa1[[2]]
        dv_z = f.pa1[[3]]
        d_f.cb1 = 7 * dv_x + 11 * dv_y + 13 * dv_z
        f.cb1 = 7 * x + 11 * y + 13 * z
        d_f.raidm33 <- d_f.cb1
        f.raidm33 <- f.cb1
        list(df = d_f.raidm33, f = f.raidm33)
    }
    s <- function(x, y, z) {
        f.cb1 = 7 * x + 11 * y + 13 * z
        f.raidm33 <- f.cb1
        f.raidm33
    }
    f.ra5 = dcall(s, list(dv_c, dv_d, dv_e), list(c, d, e))
    d_f.ca2 = f.ra5[[1]]
    f.ca2 = f.ra5[[2]]
    d_f.ca1 = 1 * dv_a + 2 * dv_b + 3 * d_f.ca2
    f.ca1 = 1 * a + 2 * b + 3 * f.ca2
    d_f.ridm16 <- d_f.ca1
    f.ridm16 <- f.ca1
    list(df = d_f.ridm16, f = f.ridm16)
}

adrDiffFor(f, list(1,2,3,4,5), options=adrOptions(i
    =1:5))
```

```
ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 125 chars
ADR: HTTR response after 0.932 s, 2144 chars
ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 91 chars
ADR: HTTR response after 0.449 s, 1030 chars
$J
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2   21   33   39

$f
[1] 395
```

4.2.2 Named function arguments

```
f <- function(a,b,c,d,e) {
    1*a+2*b+3*c+4*d+5*e
}
g1 <- function(a,b,c,d,e) { f( b, c, a, e, d) }
g2 <- function(a,b,c,d,e) { f(b=b,c=c,a=a,e=e,d=d) }
g1(1,2,3,4,5)
g2(1,2,3,4,5)
```

```
[1] 51

[1] 55
```

```
adrDiffFor(g1, list(1,2,3,4,5), options=adrOptions(i
   =1:5))
adrDiffFor(g2, list(1,2,3,4,5), options=adrOptions(i
   =1:5))
```

```
ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 47 chars
ADR: HTTR response after 0.653 s, 1142 chars
ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 68 chars
ADR: HTTR response after 0.57 s, 1148 chars
$J
     [,1] [,2] [,3] [,4] [,5]
[1,]    3    1    2    5    4

$f
```

```
[1] 51

ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 67 chars
ADR: HTTR response after 0.668 s, 1202 chars
$J
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5

$f
[1] 55
```

4.2.3 Function arguments with defaults

```
f <- function(a=5,b=4*a,c=3*b,d=2*c,e=1*d) {
    2*a+3*b+5*c+7*d+11*e
}
g1 <- function(a,b,c,d,e) { f(  b,  c) }
g2 <- function(a,b,c,d,e) { f(b=b,d=c) }
g1(1,2,3,4,5)
g2(1,2,3,4,5)
```

```
[1] 382

[1] 100
```

```
adrDiffFor(g1, list(1,2,3,4,5), options=adrOptions(i
   =1:5))
adrDiffFor(g2, list(1,2,3,4,5), options=adrOptions(i
   =1:5))
```

```
ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 38 chars
ADR: HTTR response after 0.465 s, 1046 chars
ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 105 chars
ADR: HTTR response after 0.628 s, 1363 chars
$J
     [,1] [,2] [,3] [,4] [,5]
[1,]    0    2  126    0    0

$f
```

```
[1] 382

ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 46 chars
ADR: HTTR response after 0.591 s, 1070 chars
ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 105 chars
ADR: HTTR response after 0.595 s, 1275 chars
$J
     [,1] [,2] [,3] [,4] [,5]
[1,]    0   18   18    0    0

$f
[1] 100
```

4.3 Arithmetic operators

All the arithmetic operators are supported. Addition and subtraction:

```
adrDiffFor(function(x, y) x + x + y - y, list(0.75,
   0.25), options=adrOptions(i = 1:2))

ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 29 chars
ADR: HTTR response after 0.516 s, 964 chars
$J
     [,1] [,2]
[1,]    2    0

$f
[1] 1.5
```

Multiplication and division:

```
adrDiffFor(function(x, y) x * y / y, list(0.75, 0.25),
    options=adrOptions(i = 1:2))

ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 23 chars
ADR: HTTR response after 0.472 s, 1089 chars
$J
     [,1] [,2]
[1,]    1    0

$f
[1] 0.75
```

Exponentiation:

```
adrDiffFor(function(x, y) (x ^ y) ^ (1 / y), list
   (0.75, 0.25), options=adrOptions(i = 1:2))
```

```
ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 27 chars
ADR: HTTR response after 0.575 s, 1239 chars
$J
     [,1]         [,2]
[1,]    1 7.771561e-16

$f
[1] 0.75
```

```
adrDiffFor(function(x, y) x ^ y * x ^ (-y), list(0.75,
    0.25), options=adrOptions(i = 1:2))
```

```
ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 28 chars
ADR: HTTR response after 0.582 s, 1358 chars
$J
             [,1] [,2]
[1,] 5.551115e-17    0

$f
[1] 1
```

4.4 Matrix Arithmetic

4.4.1 Matrix multiplication

```
A <- matrix(rnorm(1:4), 2)
adrDiffFor(function(x, y) x %*% y, list(A, eye(2)),
   options=adrOptions(i=1:2))
```

```
ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 23 chars
ADR: HTTR response after 0.393 s, 948 chars
$J
     [,1] [,2] [,3] [,4]       [,5]       [,6]       [,7]       [,8]
[1,]    1    0    0    0 -1.8652573  0.4701273  0.0000000  0.0000000
[2,]    0    1    0    0 -0.8590613 -0.5946854  0.0000000  0.0000000
```

```
[3,]    0    0    1    0  0.0000000  0.0000000 -1.8652573  0.4701273
[4,]    0    0    0    1  0.0000000  0.0000000 -0.8590613 -0.5946854

$f
           [,1]       [,2]
[1,] -1.8652573  0.4701273
[2,] -0.8590613 -0.5946854
```

4.4.2 Matrix division (solve)

Matrix division (solving) is supported

```
A <- matrix(rnorm(1:4), 2)
x <- matrix(rnorm(1:2), 2)
adrDiffFor(function(A, x) solve(A, A %*% x), list(A, x
    ), options=adrOptions(i=1:2))
```

```
ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 33 chars
ADR: HTTR response after 0.519 s, 1135 chars
$J
     [,1] [,2]          [,3]          [,4] [,5] [,6]
[1,]    0    0  1.487601e-17 -1.094163e-17    1    0
[2,]    0    0 -3.791052e-18  2.471316e-17    0    1

$f
           [,1]
[1,] -0.1472812
[2,]  0.1574937
```

In complex arithmetic:

```
A <- matrix(rnorm(1:4) + 1i*rnorm(1:4), 2)
x <- matrix(rnorm(1:2) + 1i*rnorm(1:2), 2)
adrDiffFor(function(A, x) solve(A, A %*% x), list(A, x
    ), options=adrOptions(i=1:2))
```

```
$J
                             [,1]                          [,2]
[1,] 2.585559e-16-9.462841e-17i -8.243019e-17-1.813046e-17i
```

```
[2,] 8.042019e-17-1.377575e-16i  3.574908e-17+6.160220e-17i
                           [,3]                          [,4]
[1,] 8.449668e-17-8.956254e-17i -3.659817e-17+9.233855e-18i
[2,] 4.616585e-18-7.118702e-17i  2.662007e-17+1.749106e-17i
                           [,5] [,6]
[1,]  1.000000e+00-0.00000e+00i 0+0i
[2,] -7.176409e-17+4.28179e-18i 1+0i

$f
                      [,1]
[1,] -1.4013404+0.6791557i
[2,] -0.2764495-0.1680218i
```

4.4.3 QR decomposition

Matrix QR decomposition is supported:

```
A <- matrix(rnorm(1:4), 2)
f <- function(A) {
    q <- qr(A)
    qr.Q(q) %*% qr.R(q)
}
adrDiffFor(f, list(A))
```

```
ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 52 chars
ADR: HTTR response after 0.538 s, 1425 chars
ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 352 chars
ADR: HTTR response after 1.95 s, 3468 chars
ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 582 chars
ADR: HTTR response after 2.06 s, 4480 chars
ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 226 chars
ADR: HTTR response after 0.983 s, 2168 chars
ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 41 chars
ADR: HTTR response after 0.642 s, 1167 chars
$J
             [,1]          [,2]          [,3]         [,4]
[1,]  1.000000e+00  2.775558e-17  0.000000e+00 0.000000e+00
[2,] -3.747003e-16  1.000000e+00  0.000000e+00 0.000000e+00
[3,] -1.332268e-15  5.273559e-16  1.000000e+00 3.400058e-16
```

```
[4,] -2.220446e-15 -5.689893e-16 -1.179612e-16 1.000000e+00

$f
            [,1]      [,2]
[1,]  0.01461015 0.8729599
[2,] -0.38804832 1.4409187
```

4.5 Trigonometric functions

The three trigonometric functions and their inverses are supported:

```
adrDiffFor(function(x) asin(acos(atan(tan(cos(sin(x))))
    )))), list(0.75))
```

```
ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 47 chars
ADR: HTTR response after 0.776 s, 1795 chars
$J
     [,1]
[1,]    1

$f
[1] 0.75
```

5 Use cases

5.1 Fractal function

```
f <- function(x) {
    if (x > 0) {
        f(sqrt(x) - (1 + 1.21111e-6))
    } else {
        x
    }
}
discr <- function(x, f) {
    r <- array(0, length(x));
    for (k in 1:length(x)) {
        r[[k]] <- f(x[[k]])
    }
    r
```

```
}
plot(function(x) discr(x, f), 0, 20, type='p', ylab='f
    (x)', main='Function')

d(f)
```

```
ADR: Post HTTR request for "r-ad-fm" to 'https://r-adr.de/adr/', 98 chars
ADR: HTTR response after 0.666 s, 1461 chars
function (f.p1, f.p0)
{
    x = f.p0[[1]]
    dv_x = f.p1[[1]]
    if (x > 0) {
        f.ra8 = df_sqrt(list(dv_x), list(x))
        d_f.ca3 = f.ra8[[1]]
        f.ca3 = f.ra8[[2]]
        d_f.ca2 = d_f.ca3
        f.ca2 = f.ca3 - (1 + 1.21111e-06)
        f.ra8 = dcall(f, list(d_f.ca2), list(f.ca2))
        d_f.ca1 = f.ra8[[1]]
        f.ca1 = f.ra8[[2]]
        d_f.raidm22 <- d_f.ca1
        f.raidm22 <- f.ca1
    }
    else {
        d_f.raidm22 <- dv_x
        f.raidm22 <- x
    }
    list(df = d_f.raidm22, f = f.raidm22)
}
```

```
plot(function(x) discr(x, function(x) adrDiffFor(f,
    list(x))$J), 0, 20, type='p', ylab='d f(x)', main='
    AD derivatives')

plot(function(x) discr(x, function(x) adrDiffFD(f,
    list(x))$J), 1, 20, type='p', ylab='d f(x)', main='
    FD derivatives')

plot(function(x) discr(x, function(x) abs(adrDiffFor(f
    , list(x))$J - adrDiffFD(f, list(x))$J)), 1, 20,
```

```
type='p', ylab='AD - FD', main='AD - FD derivatives
')
```