`transform` — library for integral transforms

Table of contents

Introduction

The `transform` library provides some integral transformations.

The package functions are called using the package name `transform` and the name of the function. E.g., use

```
>> transform::fourier(exp(-t^2), t, s)
```

to compute the Fourier transform of e^{-t^2} with respect to `t` at the point `s`. This mechanism avoids naming conflicts with other library functions. If this is found to be inconvenient, then the routines of the `transform` package may be exported via `export`. E.g., after calling

```
>> export(transform, fourier)
```

the function `transform::fourier` may be called directly:

```
>> fourier(exp(-t^2), t, s)
```

All routines of the `transform` package are exported simultaneously by

```
>> export(transform)
```

The functions available in the `transform` library can be listed with:

```
>> info(transform)
```

`transform::fourier`, `transform::invfourier` – Fourier and inverse Fourier transform

`transform::fourier(f, t, s)` computes the Fourier transform $\int_{-\infty}^{\infty} f\, e^{i\, s\, t}\, dt$ of the expression $f = f(t)$ with respect to the variable t at the point s.

`transform::invfourier(F, S, T)` computes the inverse Fourier transform $\frac{1}{2\pi} \int_{-\infty}^{\infty} F\, e^{-i\, S\, T}\, dS$ of the expression $F = F(S)$ with respect to the variable S at the point T.

Call(s):

- `transform::fourier(f, t, s)`
- `transform::invfourier(F, S, T)`

Parameters:

`f, F` — arithmetical expressions
`t, S` — identifiers (the transformation variables)
`s, T` — arithmetical expressions (the evaluation points)

Return Value: an arithmetical expression

Overloadable by: `f, F`

Related Functions: `numeric::fft`, `numeric::invfft`

Details:

- An unevaluated function call is returned, if no explicit representation of the transform is found.
- `transform::invfourier(F, S, T)` is computed as

$$\mathtt{transform} :: \mathtt{fourier}(\mathtt{F},\ \mathtt{S},\ -\mathtt{T})/2/\mathtt{PI}.$$

  This result is returned, if no explicit representation of the transformation is found.
- The discrete Fourier transform is implemented by the functions `numeric::fft` and `numeric::invfft`.

Example 1. The following call produces the Fourier transform as an expression in the variable `s`:

```
>> transform::fourier(exp(-t^2), t, s)

                                 /    2 \
                          1/2    |   s  |
                        PI    exp| - -- |
                                 \   4  /

>> transform::invfourier(%, s, t)

                                   2
                            exp(- t )
```

Note that the Fourier transform can be evaluated directly at a specific point such as $s = 2a$ or $s = 5$:

```
>> transform::fourier(t*exp(-a*t^2), t, s),
   transform::fourier(t*exp(-a*t^2), t, 2*a),
   transform::fourier(t*exp(-a*t^2), t, 2)

                 /    2  \
          1/2    |   s   |                           1/2    /   1 \
 1/2 I s PI   exp| - --- |      1/2            I PI    exp| -
- |
                 \   4 a /  I PI    exp(-a)              \   a /
 ------------------------, --------------, ------------
------
            3/2                   1/2                3/2
           a                     a                  a
```

Example 2. An unevaluated call is returned, if no explicit representation of the transform is found:

```
>> transform::fourier(besselJ(0, 1/(1 + t^2)), t, s)

                         /        /      1    \       \
        transform::fourier| besselJ| 0, ------ |, t, s |
                         |        |     2     |       |
                         \        \    t  + 1 /       /

>> transform::invfourier(%, s, t)

                               /      1    \
                        besselJ| 0, ------ |
                               |     2     |
                               \    t  + 1 /
```

Note that the inverse transform is related to the direct transform:

```
>> transform::invfourier(unknown(s), s, t)

              transform::fourier(unknown(s), s, -t)
              -------------------------------------
                              2 PI
```

Example 3. The distribution `dirac` is handled:

```
>> transform::fourier(t^3, t, s)

                       2 I PI dirac(s, 3)

>> transform::invfourier(%, s, t)

                                3
                               t

>> transform::fourier(heaviside(t - t0), t, s)

                            /                I \
                 exp(I s t0) | PI dirac(s) + - |
                            \                s /
```

Example 4. The Fourier transform of a function is related to the Fourier transform of its derivative:

```
>> transform::fourier(diff(f(t), t), t, s)

               -I s transform::fourier(f(t), t, s)
```

Background:

- Reference: F. Oberhettinger, "Tables of Fourier Transforms and Fourier Transforms of Distributions", Springer, 1990.

Changes:

- `transform::invfourier` used to be called `transform::ifourier`.

`transform::laplace`, `transform::invlaplace` – Laplace and inverse Laplace transform

`transform::laplace(f, t, s)` computes the Laplace transform $\int_0^\infty f\, e^{-s\,t}\, dt$ of the expression $f = f(t)$ with respect to the variable t at the point s.

`transform::invlaplace(F, S, T)` computes the inverse Laplace transform of the expression $F = F(S)$ with respect to the variable S at the point T.

Call(s):

- `transform::laplace(f, t, s)`
- `transform::invlaplace(F, S, T)`

Parameters:

`f, F` — arithmetical expressions
`t, S` — identifiers (the transformation variables)
`s, T` — arithmetical expressions (the evaluation points)

Return Value: an arithmetical expression or an unevaluated function call of domain type `transform::laplace` or `transform::invlaplace`, respectively.

Overloadable by: `f, F`

Details:

- An unevaluated function call is returned, if no explicit representation of the transform is found.

Example 1. The following call produces the Laplace transform as an expression in the variable `s`:

```
>> transform::laplace(exp(-a*t), t, s)

                                  1
                                -----
                                a + s
```

```
>> transform::invlaplace(%, s, t)

                               exp(-a t)
```

Note that the Laplace transform can be evaluated directly at a specific point such as $s = 2a$ or $s = 5$:

```
>> transform::laplace(t^10*exp(-a*t), t, s),
   transform::laplace(t^10*exp(-a*t), t, 2*a),
   transform::laplace(t^10*exp(-a*t), t, 1 + PI)
```

```
 3628800    44800       3628800
---------, --------, --------------
       11         11              11
(a + s)    2187 a    (a + PI + 1)
```

Some further examples:

```
>> transform::laplace(1 + exp(-a*t)*sin(b*t), t, s)

                         1         b
                         - + -------------
                         s    2          2
                             b  + (a + s)

>> transform::invlaplace(1/(s^3 + s^5), s, t)

                                  2
                                 t
                          cos(t) + -- - 1
                                 2

>> transform::invlaplace(exp(-2*s)/(s^2 + 1) + s/(s^3 + 1), s, t)

                                exp(-t)
 sin(t - 2) heaviside(t - 2) - ------- +
                                   3

           /    /    1/2 \           /    1/2 \ \
     / t \ |    | t 3    |    1/2    | t 3    | |
  exp| - | | cos| ------ | + 3    sin| ------ | |
     \ 2 / \    \   2    /           \   2    / /
  -----------------------------------------------
                          3
```

Example 2. An unevaluated call is returned, if no explicit representation of the transform is found:

```
>> transform::laplace(exp(-t^3), t, s)

                                           3
               transform::laplace(exp(- t ), t, s)
```

Note that this is not an ordinary expression, but a domain element of domain type `transform::laplace`:

```
>> domtype(%)

                       transform::laplace
```

The inverse of the formal transform yields the original expression:

```
>> transform::invlaplace(%2, s, t)

                                   3
                            exp(- t )
```

Example 3. The distribution `dirac` and the Heaviside function `heaviside` are handled:

```
>> transform::laplace(dirac(t - 3), t, s)

                            exp(-3 s)

>> transform::invlaplace(1, s, t)

                            dirac(t)

>> transform::laplace(heaviside(t - PI), t, s)

                           exp(-s PI)
                           ----------
                               s
```

Example 4. The Laplace transform of a function is related to the Laplace transform of its derivative:

```
>> transform::laplace(diff(f(t), t), t, s)

             s transform::laplace(f(t), t, s) - f(0)
```

Changes:

- `transform::invlaplace` used to be called `transform::ilaplace`.
- The Laplace transform is now overloadable by the first argument.