Integral Transforms (revamped) and PDEs

Integral transforms, implemented in Maple as the inttrans package, are special integrals that appear frequently in mathematical-physics and that have remarkable properties. One of the main uses of integral transforms is for the computation of exact solutions to ordinary and partial differential equations with initial/boundary conditions. In Maple, that functionality is implemented in dsolve/inttrans and in pdsolve/boundary conditions.

During the last months, we have been working heavily on revamping several aspects of these valuable integral transform functions. This is work in progress, in collaboration with Katherina von Bulow. We have already achieved impactful results and this post is about that.

The integral transforms are represented by the commands of the inttrans package:

```
> with(inttrans)
```

$$[addtable, fourier, fouriercos, fouriersin, hankel, hilbert, invfourier, invhilbert, invlaplace, invmellin, laplace, mellin, savetable, setup]$$

Three of these commands, *addtable*, *savetable*, and *setup* (this one is new, only present after installing the Physics Updates) are "administrative" commands while the others are computational representations for integrals. For example,

```
> FunctionAdvisor(integral_form, fourier)
```

$$\left[fourier(a, b, z) = \int_{-\infty}^{\infty} \frac{a}{e^{I b z}} \, db, \text{ with no restrictions on } (a, b, z) \right]$$

```
> FunctionAdvisor(integral_form, mellin)
```

$$\left[mellin(a, b, z) = \int_{0}^{\infty} a\, b^{z-1} \, db, \text{ with no restrictions on } (a, b, z) \right]$$

For all the integral transform commands, the first argument is *the integrand*, the second one is *the dummy integration variable* of a definite integral and the third one is *the evaluation point*. (also called *transform variable*). The integral representation is also visible using the convert network

```
> laplace(f(t), t, s) :
  % = convert(%, Int)
```

$$laplace(f(t), t, s) = \int_{0}^{\infty} f(t)\, e^{-s t} \, dt$$

Having in mind the applications of these integral transforms to compute integrals and exact solutions to PDE with boundary conditions, five different aspects of these transforms received further development, as shown in the following sections:

- **Compute Derivatives: Yes or No**
- **Numerical Evaluation**
- **Two Hankel Transform Definitions**
- **More integral transform results**
- **Mellin and Hankel transform solutions for Partial Differential Equations with Boundary Conditions**

The project includes having all these tranforms available at user level (not ready), say as *FourierTransform* for *inttrans:-fourier*, so that we don't need to input $with(inttrans)$ anymore. Related to these changes we also intend to have Heaviside(0) not return *undefined* anymore, and return itself instead, unevaluated, so that one can set its value according to the problem/preferred convention (typically 0, 1/2 or 1) and have all the Maple library following that choice.

The material presented in the following sections is reproducible already in Maple 2019 by installing the latest Physics Updates (v.435 or higher),

Compute derivatives: Yes or No.

For historical reasons, previous implementations of these integral transform commands did not follow a standard paradigm of computer algebra: "*Given a function* $f(x)$, *the input* $\frac{\mathrm{d}}{\mathrm{d}x} f(x)$ *should return the derivative of* $f(x)$ ". The implementation instead worked in the opposite direction: if you were to input the result of the derivative, you would receive the derivative representation. For example, to the input $laplace(-t \cdot f(t), t, s)$ you would receive $\frac{d}{ds}\, laplace(f(t), t, s)$. This is particularly useful for the purpose of using integral transforms to solve differential equations but it is counter-intuitive and misleading; Maple knows the differentiation rule of these functions, but that rule was not evident anywhere. It was not clear how to compute the derivative (unless you knew the result in advance).

To solve this issue, a new command, *setup*, has been added to the package, so that you can set "*whether or not*" to compute derivatives, and the default has been changed to *computederivatives* = *true* while the old behavior is obtained only if you input $setup(computederivatives = false)$. For example, after having installed the Physics Updates,

> $Physics\text{:-}Version(\ )$

The "Physics Updates" version in the MapleCloud is 435 and is the same as the version installed in this computer, created 2019, October 1, 12:46 hours, found in the directory /Users/ecterrab/maple/toolbox/2019/Physics Updates/lib/

the current settings can be queried via

> $setup(computederivatives)$

$$computederivatives = true$$

and so differentiating returns the derivative computed

> $(\%diff = diff)\,(laplace(f(t), t, s), s)$

$$\frac{\mathrm{d}}{\mathrm{d}s}\, laplace(f(t), t, s) = -laplace(f(t)\ t, t, s)$$

while changing this setting to work as in previous releases you have this computation reversed: you input the output $\frac{\mathrm{d}}{\mathrm{d}s}\, laplace(f(t), t, s) = -laplace(f(t)\ t, t, s)$ and you get the corresponding input

> $setup(computederivatives = false)$

$$computederivatives = false$$

> $\frac{\mathrm{d}}{\mathrm{d}s}\, laplace(f(t), t, s) = -laplace(f(t)\ t, t, s)$

$$\frac{\mathrm{d}}{\mathrm{d}s}\, laplace(f(t), t, s) = \frac{\mathrm{d}}{\mathrm{d}s}\, laplace(f(t), t, s)$$

Reset the value of *computederivatives*

> $setup(computederivatives = true)$

$$computederivatives = true$$

> $\frac{d}{ds}\, laplace(f(t), t, s) = -laplace(f(t)\, t, t, s)$

$$\frac{d}{ds}\, laplace(f(t), t, s) = -laplace(f(t)\, t, t, s)$$

In summary: by *default*, derivatives of integral transforms *are now computed*; if you need to work with these derivatives as in previous releases, you can input $setup(computederivatives = false)$. This setting can be changed any time you want within one and the same Maple session, and *changing it does not have any impact on the performance of intsolve, dsolve and pdsolve to solve differential equations using integral transforms.*

>

Numerical Evaluation

In previous releases, integral transforms had no numerical evaluation implemented. This is in the process of changing. So, for example, to numerically evaluate the inverse laplace transform (invlaplace command), three different algorithms have been implemented: Gaver-Stehfest, Talbot and Euler, following the presentation by Abate and Whitt, "*Unified Framework for Numerically Inverting Laplace Transforms*", INFORMS Journal on Computing 18(4), pp. 408–421, 2006.

For example, consider the exact solution to this partial differential equation subject to initial and boundary conditions

> $pde := \frac{\partial}{\partial x}\, u(x, t) = 4 \left(\frac{\partial^2}{\partial t^2}\, u(x, t)\right):$

> $iv := u(x, 0) = 0, u(0, t) = 1 :$

Note that these two conditions are not entirely compatible: the solution returned cannot be valid for $x = 0$ and $t = 0$ simultaneously. However, a solution discarding that point does exist and is given by

> $sol := pdsolve([pde, iv])$

$$sol := u(x, t) = -invlaplace\left(\frac{e^{-\frac{\sqrt{s}\, t}{2}}}{s}, s, x\right) + 1$$

Verifying the solution, one condition remains to be tested

> $pdetest(sol, [pde, iv])$

$$\left[0, 0, -invlaplace\left(\frac{e^{-\frac{\sqrt{s}\, t}{2}}}{s}, s, 0\right)\right]$$

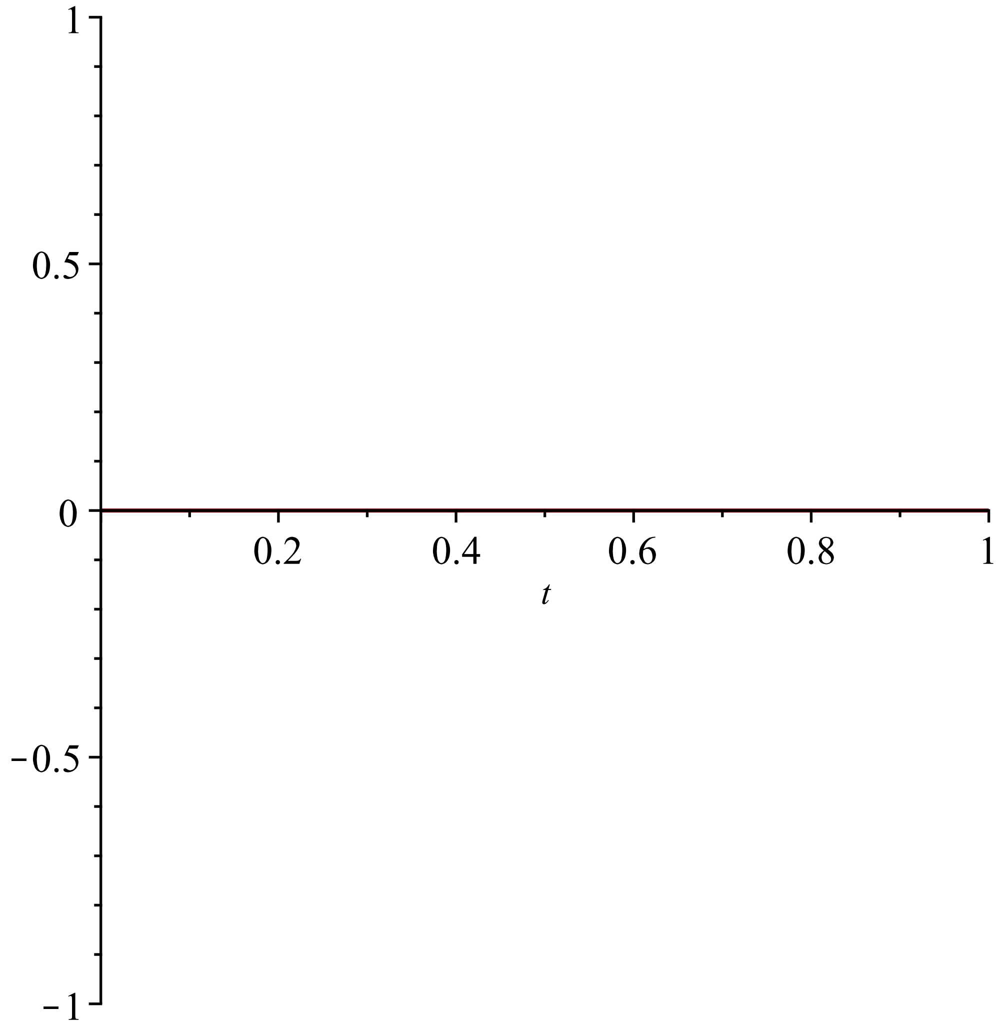
Since we now have numerical evaluation rules, we can test that what looks different from 0 in the above is actually 0.

> $zero := \left[0, 0, -invlaplace\left(\frac{e^{-\frac{\sqrt{s}\, t}{2}}}{s}, s, 0\right)\right][-1]$

$$zero := -invlaplace\left(\frac{e^{-\frac{\sqrt{s}\, t}{2}}}{s}, s, 0\right)$$

Add a small number to the initial value of t to skip the point $t = 0$:

> $plot(zero, t = 0 + 10^{-10}..1)$



The default method used is the method of *Euler sums* and the numerical evaluation is performed as usual using the evalf command. For example, consider

> $F := \sin\left(\sqrt{2\,t}\right)$:

The Laplace transform of F is given by

> $LT := laplace(F, t, s)$

$$LT := \frac{\sqrt{2}\,\sqrt{\pi}\,e^{-\frac{1}{2\,s}}}{2\,s^{3/2}}$$

and the inverse Laplace transform of LT in inert form is

> $ILT := \%invlaplace(LT, s, t)$

$$ILT := invlaplace\left(\frac{\sqrt{2}\,\sqrt{\pi}\,e^{-\frac{1}{2\,s}}}{2\,s^{3/2}}, s, t\right)$$

At $t = 1$ we have

> $eval(ILT, t = 1)$

$$\mathit{invlaplace}\left(\frac{\sqrt{2}\,\sqrt{\pi}\,e^{-\frac{1}{2s}}}{2\,s^{3/2}}, s, 1\right)$$

> $\mathit{evalf}\left(\mathit{invlaplace}\left(\frac{\sqrt{2}\,\sqrt{\pi}\,e^{-\frac{1}{2s}}}{2\,s^{3/2}}, s, 1\right)\right)$

$$0.9877659460$$

This result is consistent with the one we get if we first compute the exact form of the inverse Laplace transform at t = 1:

> $\mathit{invlaplace}\left(\frac{\sqrt{2}\,\sqrt{\pi}\,e^{-\frac{1}{2s}}}{2\,s^{3/2}}, s, 1\right) = \mathit{value}\left(\mathit{invlaplace}\left(\frac{\sqrt{2}\,\sqrt{\pi}\,e^{-\frac{1}{2s}}}{2\,s^{3/2}}, s, 1\right)\right)$

$$\mathit{invlaplace}\left(\frac{\sqrt{2}\,\sqrt{\pi}\,e^{-\frac{1}{2s}}}{2\,s^{3/2}}, s, 1\right) = \sin\left(\sqrt{2}\right)$$

> $\mathit{evalf}\left(\mathit{invlaplace}\left(\frac{\sqrt{2}\,\sqrt{\pi}\,e^{-\frac{1}{2s}}}{2\,s^{3/2}}, s, 1\right) = \sin\left(\sqrt{2}\right)\right)$

$$0.9877659460 = 0.9877659459$$

In addition to the standard use of *evalf* to numerically evaluate inverse Laplace transforms, one can invoke each of the three different methods implemented using the MathematicalFunctions:-Evalf command

> $\mathit{with}(\mathit{MathematicalFunctions}, \mathit{Evalf})$

$$[\mathit{Evalf}]$$

> $\mathit{Evalf}\left(\mathit{invlaplace}\left(\frac{\sqrt{2}\,\sqrt{\pi}\,e^{-\frac{1}{2s}}}{2\,s^{3/2}}, s, 1\right), \mathit{method} = \mathit{Talbot}\right)$

$$0.9877659460$$

> $\mathit{MF}\text{:-}\mathit{Evalf}\left(\mathit{invlaplace}\left(\frac{\sqrt{2}\,\sqrt{\pi}\,e^{-\frac{1}{2s}}}{2\,s^{3/2}}, s, 1\right), \mathit{method} = \mathit{GaverStehfest}\right)$

$$0.9877659460$$

> $\mathit{MF}\text{:-}\mathit{Evalf}\left(\mathit{invlaplace}\left(\frac{\sqrt{2}\,\sqrt{\pi}\,e^{-\frac{1}{2s}}}{2\,s^{3/2}}, s, 1\right), \mathit{method} = \mathit{Euler}\right)$

$$0.9877659460$$

Regarding the method we use by default: from a numerical experiment with varied problems we have concluded that our implementation of the *Euler* (*sums*) method is faster and more accurate than the other two.

Two Hankel transform definitions

In previous Maple releases, the definition of the Hankel transform was given by

$$hankel(f(t), t, s, \nu) = \int_0^\infty f(t)\sqrt{s\,t}\,J_\nu(s\,t)\,\mathrm{d}t$$

where $J_\nu(s\,t)$ is the BesselJ$(\mathrm{nu}, s\cdot t)$ function. This definition, sometimes called *alternative definition* of the Hankel transform, has the inconvenience of the square root $\sqrt{s\,t}$ in the integrand, complicating the form of the hankel transform for the Laplacian in cylindrical coordinates. On the other hand, the *definition more frequently used in the literature* is

$$hankel(f(t), t, s, \nu) = \int_0^\infty f(t)\,t\,J_\nu(s\,t)\,\mathrm{d}t$$

With it, the Hankel transform of $\frac{\partial^2}{\partial r^2}u(r,t) + \frac{\frac{\partial}{\partial r}u(r,t)}{r} + \frac{\partial^2}{\partial t^2}u(r,t)$ is given by the simple ODE form $\frac{d^2}{dt^2}\mathbb{H}(k,t) - k^2\,\mathbb{H}(k,t)$. Not just this transform but several other ones acquire a simpler form with the definition that does not have a square root in the integrand.

So the idea is to align Maple with this simpler definition, while keeping the previous definition as an alternative. Hence, by default, when you load the inttrans package, the new definition in use for the Hankel transform is

```
> hankel(f(t), t, s, nu) :
  %=convert(%, Int)
```

$$hankel(f(t), t, s, \nu) = \int_0^\infty f(t)\,t\,J_\nu(s\,t)\,\mathrm{d}t$$

You can change this default so that Maple works with the *alternative definition* as in previous releases. For that purpose, use the new *inttrans:-setup* command (which you can also use to query about the definition in use at any moment):

```
> setup(alternativehankeldefinition)
```

$$alternativehankeldefinition = false$$

This change in definition is automatically taken into account by other parts of the Maple library using the Hankel transform. For example, the differentiation rule with the new definition is

```
> (%diff=diff)(hankel(f(t), t, z, nu), z)
```

$$\frac{\mathrm{d}}{\mathrm{d}z}hankel(f(t), t, z, \nu) = -hankel(t\,f(t), t, z, \nu+1) + \frac{\nu\,hankel(f(t), t, z, \nu)}{z}$$

This differentiation rule resembles (is connected to) the differentiation rule for BesselJ, and this is another advantage of the new definition.

```
> (%diff=diff)(BesselJ(nu, z), z)
```

$$\frac{\mathrm{d}}{\mathrm{d}z}J_\nu(z) = -J_{\nu+1}(z) + \frac{\nu\,J_\nu(z)}{z}$$

Furthermore, several transforms have acquired a simpler form, as for example:

```
> (%hankel=hankel)(exp(I a r)/r, r, k, 0) assuming a > 0, k < a
```

$$hankel\left(\frac{e^{I a r}}{r}, r, k, 0\right) = \frac{1}{\sqrt{-a^2+k^2}}$$

Let's compare: make the definition be as in previous releases.

```
> setup(alternativehankeldefinition = true)
```

$$alternativehankeldefinition = true$$

```
> hankel(f(t), t, s, nu) :
  % = convert(%, Int)
```

$$hankel(f(t), t, s, \nu) = \int_0^\infty f(t)\sqrt{s\,t}\,J_\nu(s\,t)\,\mathrm{d}t$$

The differentiation rule with the previous (alternative) definition was not as simple:

```
> (%diff = diff)(hankel(f(t), t, s, nu), s)
```

$$\frac{\mathrm{d}}{\mathrm{d}s} hankel(f(t), t, s, \nu) = -hankel(t f(t), t, s, \nu+1) + \frac{\nu\, hankel(f(t), t, s, \nu)}{s} + \frac{hankel(f(t), t, s, \nu)}{2\,s}$$

And the transform $hankel\left(\frac{e^{I a r}}{r}, r, k, 0\right) = \frac{1}{\sqrt{-a^2+k^2}}$ was also not so simple:

```
> (%hankel = hankel)(exp(I a r)/r, r, k, 0) assuming a > 0, k < a
```

$$hankel\left(\frac{e^{I a r}}{r}, r, k, 0\right) = \frac{I\,a\,{}_2F_1\left(\frac{3}{4}, \frac{3}{4}; \frac{3}{2}; \frac{a^2}{k^2}\right)\Gamma\left(\frac{3}{4}\right)^4 + \pi^2\,k\,{}_2F_1\left(\frac{1}{4}, \frac{1}{4}; \frac{1}{2}; \frac{a^2}{k^2}\right)}{k\,\pi\,\Gamma\left(\frac{3}{4}\right)^2}$$

Reset to the new default value of the definition.

```
> setup(alternativehankeldefinition = false)
```

$$alternativehankeldefinition = false$$

```
> hankel(f(t), t, s, nu) :
  % = convert(%, Int)
```

$$hankel(f(t), t, s, \nu) = \int_0^\infty f(t)\,t\,J_\nu(s\,t)\,\mathrm{d}t$$

More integral transform results

The revision of the integral transforms includes also filling gaps: previous transforms that were not being computed are now computed. Still with the Hankel transform, consider the operators

$$> \ `D/t` := u \to \frac{\frac{\partial}{\partial t} u}{t} :$$

$$formula\_plus := t^{-\nu}\,`D/t`^{(m)}\left(t^{m+\nu}\,u(t)\right) :$$

$$formula\_minus := t^{\nu}\,`D/t`^{(m)}\left(t^{m-\nu}\,u(t)\right) :$$

Being able to transform these operators into algebraic expressions or differential equations of lower order is key for solving PDE problems with Boundary Conditions.

Consider, for instance, this ODE

> $setup(computederivatives = false);$

$$computederivatives = false$$

> $simplify(eval(formula\_minus, [\mathrm{nu} = 6, m = 3]))$

$$\frac{\left(\frac{d^3}{dt^3} u(t)\right) t^3 - 12 \left(\frac{d^2}{dt^2} u(t)\right) t^2 + 57 \left(\frac{d}{dt} u(t)\right) t - 105 u(t)}{t^3}$$

Its Hankel transform is a simple algebraic expression

> $hankel\left(\frac{\left(\frac{d^3}{dt^3} u(t)\right) t^3 - 12 \left(\frac{d^2}{dt^2} u(t)\right) t^2 + 57 \left(\frac{d}{dt} u(t)\right) t - 105 u(t)}{t^3}, t, s, 6\right)$

$$-s^3\, hankel(u(t), t, s, 3)$$

An example with *formula_plus*

> $simplify(eval(formula\_plus, [\mathrm{nu} = 7, m = 4]))$

$$\frac{\left(\frac{d^4}{dt^4} u(t)\right) t^4 + 38 \left(\frac{d^3}{dt^3} u(t)\right) t^3 + 477 \left(\frac{d^2}{dt^2} u(t)\right) t^2 + 2295 \left(\frac{d}{dt} u(t)\right) t + 3465 u(t)}{t^4}$$

> $hankel\left(\frac{\left(\frac{d^4}{dt^4} u(t)\right) t^4 + 38 \left(\frac{d^3}{dt^3} u(t)\right) t^3 + 477 \left(\frac{d^2}{dt^2} u(t)\right) t^2 + 2295 \left(\frac{d}{dt} u(t)\right) t + 3465 u(t)}{t^4}, t, s, 7\right)$

$$s^4\, hankel(u(t), t, s, 11)$$

In the case of *hankel*, not just differential operators but also several new transforms are now computable

> $hankel(1, r, k, \mathrm{nu})$

$$\begin{cases} \frac{\delta(k)}{k} & \nu = 0 \\ \frac{\nu}{k^2} & otherwise \end{cases}$$

> $hankel(r^m, r, k, \mathrm{nu})$

$$\begin{cases} \frac{\delta(k)}{k} & \nu=0 \wedge m=0 \\ \frac{2^{m+1} k^{-m-2} \Gamma\left(1+\frac{m}{2}+\frac{\nu}{2}\right)}{\Gamma\left(\frac{\nu}{2}-\frac{m}{2}\right)} & \mathit{otherwise} \end{cases}$$

>

Mellin and Hankel transform solutions for Partial Differential Equations with Boundary Conditions

In previous Maple releases, the Fourier and Laplace transforms were used to compute exact solutions to PDE problems with boundary conditions. Now, Mellin and Hankel transforms are also used for that same purpose.

Example:

> $pde := x^2 \left(\frac{\partial^2}{\partial x^2} u(x,y)\right) + x\left(\frac{\partial}{\partial x} u(x,y)\right) + \frac{\partial^2}{\partial y^2} u(x,y) = 0:$

$iv := u(x,0)=0, u(x,1) = \begin{cases} 1 & 0 \le x \textbf{ and } x<1 \\ 0 & 1<x \end{cases} :$

> $sol := pdsolve([pde, iv])$

$$sol := u(x,y) = invmellin\left(\frac{\sin(s\,y)}{\sin(s)\,s}, s, x\right)$$

As usual, you can let pdsolve choose the solving method, or indicate the method yourself:

> $pde := \frac{\partial^2}{\partial r^2} u(r,t) + \frac{\frac{\partial}{\partial r} u(r,t)}{r} + \frac{\partial^2}{\partial t^2} u(r,t) = -Q_0\, q(r):$

$iv := u(r,0)=0:$

> $pdsolve([pde, iv])$

$$u(r,t) = Q_0\left(-hankel\left(\frac{e^{-s\,t}\, hankel(q(r), r, s, 0)}{s^2}, s, r, 0\right) + hankel\left(\frac{hankel(q(r), r, s, 0)}{s^2}, s, r, 0\right)\right)$$

It is sometimes preferable to see these solutions in terms of more familiar integrals. For that purpose, use

> $convert(\textbf{Equality}, Int, only = hankel)$

$$u(r,t) = Q_0 \left(-\left(\int_0^{\infty} \frac{e^{-s t} \left(\int_0^{\infty} q(r)\, r J_0(r s)\, dr \right) J_0(r s)}{s} \, ds \right) + \int_0^{\infty} \frac{\left(\int_0^{\infty} q(r)\, r J_0(r s)\, dr \right) J_0(r s)}{s} \, ds \right)$$

An example where the hankel transform is computable to the end:

$$> \; pde := c^2 \left(\frac{\partial^2}{\partial r^2} u(r,t) + \frac{\frac{\partial}{\partial r} u(r,t)}{r} \right) = \frac{\partial^2}{\partial t^2} u(r,t) :$$

$$iv := u(r,0) = A\, a \left(a^2 + r^2\right)^{-\frac{1}{2}}, \mathrm{D}_2(u)(r,0) = 0 :$$

$> \;$ *pdsolve*([*pde*, *iv*], *method* = *Hankel*) assuming $r > 0$, $t > 0$, $a > 0$

$$u(r,t) = \frac{A\, a \left(\sqrt{-c^2 t^2 + 2\, \mathrm{I}\, a\, c\, t + a^2 + r^2} + \sqrt{-c^2 t^2 - 2\, \mathrm{I}\, a\, c\, t + a^2 + r^2} \right)}{2 \sqrt{-c^2 t^2 - 2\, \mathrm{I}\, a\, c\, t + a^2 + r^2}\, \sqrt{-c^2 t^2 + 2\, \mathrm{I}\, a\, c\, t + a^2 + r^2}}$$

$>$