Mathematical Functions

Relevant developments in the MathematicalFunctions project happened for Maple 2017, regarding both the addition of the four Appell functions, representing the first ever full implementation of these functions in computational environments, as well as the addition of a new package, Evalf, for performing numerical experimentation taking advantage of sophisticated symbolic computation functionality. The Evalf package and project aims to provide a user-friendly environment to develop and work with numerical algorithms for mathematical functions.

The Four Appell Functions

The four multi-parameter Appell functions, AppellF1, AppellF2, AppellF3 and AppellF4 are doubly hypergeometric functions that include as particular cases the 2F1 hypergeometric and some cases of the MeijerG function, and with them most of the known functions of mathematical physics. These Appell functions have been popping up with increasing frequency in applications in quantum mechanics, molecular physics, and general relativity.

As in the case of the hypergeometric function, a distinction is made between the four Appell *series*, with restricted domain of convergence, and the four Appell *functions*, that coincide with the series in their domain of convergence but also extend them analytically to the whole complex plane. The Maple implementation of the Appell functions includes a thorough set of their symbolic properties, all accessible using the FunctionAdvisor, as well *as numerical algorithms to evaluate the four functions over the whole complex plane*, representing the first ever complete computational implementation of these functions.

To display special functions and sequences using textbook notation as shown in this page, use extended typesetting:

```
> interface(typesetting = extended) : Typesetting:-EnableTypesetRule(Typesetting:-
      SpecialFunctionRules) :
```

Examples

The definition of the four Appell *series* and the corresponding domains of convergence can be seen through the FunctionAdvisor. For example,

> *FunctionAdvisor*(*definition*, *AppellF1*)

$$\left[F_1(a, b_1, b_2, c, z_1, z_2) = \sum_{\_k1=0}^{\infty} \sum_{\_k2=0}^{\infty} \frac{(a)_{\_k1+\_k2} (b_1)_{\_k1} (b_2)_{\_k2} z_1^{\_k1} z_2^{\_k2}}{(c)_{\_k1+\_k2} \_k1!\ \_k2!}, |z_1| < 1 \right. \quad \textbf{(1.1.1)}$$

$$\left. \wedge |z_2| < 1 \right]$$

> *FunctionAdvisor*(*definition*, *AppellF2*);

$$\left[F_2(a, b_1, b_2, c_1, c_2, z_1, z_2) = \sum_{\_k1=0}^{\infty} \sum_{\_k2=0}^{\infty} \frac{(a)_{\_k1+\_k2} (b_1)_{\_k1} (b_2)_{\_k2} z_1^{\_k1} z_2^{\_k2}}{(c_1)_{\_k1} (c_2)_{\_k2} \_k1!\ \_k2!}, |z_1| \right. \quad \textbf{(1.1.2)}$$

$$\left. + |z_2| < 1 \right]$$

> *FunctionAdvisor*(*definition*, *AppellF3*);

$$\left[F_3(a_1, a_2, b_1, b_2, c, z_1, z_2) = \sum_{\_k1=0}^{\infty} \sum_{\_k2=0}^{\infty} \frac{(a_1)_{\_k1} (a_2)_{\_k2} (b_1)_{\_k1} (b_2)_{\_k2} z_1^{\_k1} z_2^{\_k2}}{(c)_{\_k1+\_k2} \_k1!\ \_k2!}, \right. \quad \textbf{(1.1.3)}$$

$$\left. |z_1| < 1 \wedge |z_2| < 1 \right]$$

> *FunctionAdvisor*(*definition*, *AppellF4*);

$$\left[F_4(a, b, c_1, c_2, z_1, z_2) = \sum_{\_k1=0}^{\infty} \sum_{\_k2=0}^{\infty} \frac{(a)_{\_k1+\_k2} (b)_{\_k1+\_k2} z_1^{\_k1} z_2^{\_k2}}{(c_1)_{\_k1} (c_2)_{\_k2} \_k1!\ \_k2!}, \sqrt{|z_1|} + \sqrt{|z_2|} \right. \quad \textbf{(1.1.4)}$$

$$\left. < 1 \right]$$

From these definitions, these series and the corresponding analytic extensions (Appell functions) are singular (division by zero) when the c parameters entering the pochhammer functions in the denominators of these series are non-positive integers. For an analogous reason, when the a and/or b parameters entering the pochhammer functions in the numerators of the series are non-positive integers, the series will truncate and the Appell functions will be polynomial. Consult the FunctionAdvisor for comprehensive information on the combinations of all these conditions. For example, for AppellF1, the singular cases happen when any of the following conditions hold

> *FunctionAdvisor*(*singularities*, *AppellF1*)

$$\left[F_1(a, b_1, b_2, c, z_1, z_2), \left(c :: \mathbb{Z}^{(0,-)} \wedge a :: \mathbb{Z}^{(0,-)} \wedge b_1 :: (\neg \mathbb{Z}^{(0,-)}) \wedge a < c\right) \vee \left(c :: \mathbb{Z}^{(0,-)}\right. \right. \quad \textbf{(1.1.5)}$$

$$\wedge\, a::\mathbb{Z}^{(0,-)} \wedge b_2::(\neg\, \mathbb{Z}^{(0,-)}) \wedge a < c\Big) \vee \Big(c::\mathbb{Z}^{(0,-)} \wedge a::\mathbb{Z}^{(0,-)} \wedge b_1::\mathbb{Z}^{(0,-)}$$
$$\wedge\, b_2::\mathbb{Z}^{(0,-)} \wedge a < c \wedge b_1 + b_2 < c\Big) \vee \Big(c::\mathbb{Z}^{(0,-)} \wedge a::(\neg\, \mathbb{Z}^{(0,-)}) \wedge b_1::($$
$$\neg\, \mathbb{Z}^{(0,-)})\Big) \vee \Big(c::\mathbb{Z}^{(0,-)} \wedge a::(\neg\, \mathbb{Z}^{(0,-)}) \wedge b_2::(\neg\, \mathbb{Z}^{(0,-)})\Big) \vee \Big(c::\mathbb{Z}^{(0,-)} \wedge a::($$
$$\neg\, \mathbb{Z}^{(0,-)}) \wedge b_1::\mathbb{Z}^{(0,-)} \wedge b_2::\mathbb{Z}^{(0,-)} \wedge b_1 + b_2 < c\Big)\Big]$$

By requesting the sum form of the Appell functions, besides their double power series definition, we also see the particular form the four series take when one of the summations is performed and the result expressed in terms of 2F1 hypergeometric functions. For example, for AppellF3,

> $FunctionAdvisor(sum\_form, AppellF3)$

$$\Big[F_3(a_1, a_2, b_1, b_2, c, z_1, z_2) = \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} \frac{(a_1)_m\, (a_2)_n\, (b_1)_m\, (b_2)_n\, {z_1}^m\, {z_2}^n}{(c)_{m+n}\, m!\, n!}, |z_1| < 1 \wedge |z_2| < 1\Big], \Big[F_3(a_1, a_2, b_1, b_2, c, z_1, z_2) = \sum_{k=0}^{\infty} \frac{(a_1)_k\, (b_1)_k\, {}_2F_1(a_2, b_2; c+k; z_2)\, {z_1}^k}{(c)_k\, k!}, |z_1| < 1\Big], \Big[F_3(a_1, a_2, b_1, b_2, c, z_1, z_2) = \sum_{k=0}^{\infty} \frac{(a_2)_k\, (b_2)_k\, {}_2F_1(a_1, b_1; c+k; z_1)\, {z_2}^k}{(c)_k\, k!}, |z_2| < 1\Big] \quad \textbf{(1.1.6)}$$

So, for AppellF3 (and also for AppellF1, but not for AppellF2 nor AppellF4) the domain of convergence of the single sum with hypergeometric coefficients is larger than the domain of convergence of the double series, because the hypergeometric coefficient in the single sum - say the one in z_2 - analytically extends the series with regards to the other variable - say z_1 - entering the hypergeometric coefficient.

In the literature, the Appell series are analytically extended by integral representations in terms of Eulerian double integrals. With the exception of AppellF4, one of the two iterated integrals can always be computed resulting in a single integral with hypergeometric integrand. For example, for AppellF2

> $FunctionAdvisor(integral\_form, AppellF2)$;

$$\Big[F_2(a, b_1, b_2, c_1, c_2, z_1, z_2) = \frac{\Gamma(c_1)\left(\int_0^1 \frac{u^{b_1 - 1}\, {}_2F_1\left(a, b_2; c_2; -\frac{z_2}{u\, z_1 - 1}\right)}{(1-u)^{-c_1 + b_1 + 1}\, (-u\, z_1 + 1)^a}\, du\right)}{\Gamma(b_1)\, \Gamma(c_1 - b_1)}, z_1\Big] \quad \textbf{(1.1.7)}$$

$$\neq 1 \wedge 0 < \Re(b_1) \wedge 0 < -\Re(-c_1 + b_1)\Bigg], \Bigg[F_2(a, b_1, b_2, c_1, c_2, z_1, z_2)$$

$$= \frac{\Gamma(c_2)\left(\int_0^1 \frac{u^{b_2 - 1}\, {}_2F_1\left(a, b_1; c_1; -\frac{z_1}{u\, z_2 - 1}\right)}{(1-u)^{1 + b_2 - c_2}\,(-u\, z_2 + 1)^a}\, du\right)}{\Gamma(b_2)\,\Gamma(c_2 - b_2)}, z_2 \neq 1 \wedge 0 < \Re(b_2) \wedge 0$$

$$< -\Re(-c_2 + b_2)\Bigg], \Bigg[F_2(a, b_1, b_2, c_1, c_2, z_1, z_2)$$

$$= \frac{1}{\Gamma(b_1)\,\Gamma(b_2)\,\Gamma(c_1 - b_1)\,\Gamma(c_2 - b_2)}\left(\Gamma(c_1)\,\Gamma(c_2)\left(\int_0^1 \int_0^1 \frac{u^{b_1 - 1}\, v^{b_2 - 1}}{(1-u)^{-c_1 + b_1 + 1}\,(1-v)^{1 + b_2 - c_2}\,(-u\, z_1 - v\, z_2 + 1)^a}\, du\, dv\right)\right), 0 < \Re(b_1)$$

$$\wedge 0 < \Re(b_2) \wedge 0 < -\Re(-c_1 + b_1) \wedge 0 < -\Re(-c_2 + b_2)\Bigg], \Bigg[F_2(a, b_1, b_2, c_1, c_2,$$

$$z_1, z_2) = \frac{\int_0^\infty \frac{u^{a-1}\, {}_1F_1(b_1; c_1; u\, z_1)\, {}_1F_1(b_2; c_2; u\, z_2)}{e^u}\, du}{\Gamma(a)}, \Re(z_1 + z_2) < 1 \wedge 0$$

$$< \Re(a)\Bigg]$$

For the purpose of numerically evaluating the four Appell functions over the whole complex plane, instead of numerically evaluating the integral representations, it is simpler, when possible, to evaluate the function using identities. For example, with the exception of AppellF3, the Appell functions admit identities analogous to Euler identities for the hypergeometric function. These Euler-type identities, as well as contiguity identities for the four Appell functions, are visible using the FunctionAdvisor with the option identities, or directly from the function. For AppellF4, for instance, provided that none of $a, b, a-b, c_2-a$ is a non-positive integer,

> $F_4(a, b, c_1, c_2, z_1, z_2) = (AppellF4{:}\text{-}Transformations_{\text{"Euler"}})_1(a, b, c_1, c_2, z_1, z_2)$

$$AppellF4(a, b, c_1, c_2, z_1, z_2) \tag{1.1.8}$$
$$= \frac{\Gamma(c_2)\,\Gamma(b-a)\,(-z_2)^{-a}\,AppellF4\left(a, a-c_2+1, a-b+1, c_1, \frac{1}{z_2}, \frac{z_1}{z_2}\right)}{\Gamma(c_2-a)\,\Gamma(b)}$$
$$+ \frac{\Gamma(c_2)\,\Gamma(a-b)\,(-z_2)^{-b}\,AppellF4\left(b, 1+b-c_2, b-a+1, c_1, \frac{1}{z_2}, \frac{z_1}{z_2}\right)}{\Gamma(c_2-b)\,\Gamma(a)}$$

and this identity can be used to evaluate AppellF4 at $z_1 = 1$ over the whole complex plane since, in that case, the two variables of the Appell Functions on right-hand side become equal, and that is a special case of AppellF4, expressible in terms of hypergeometric 4F3 functions

> $(1.1.8)\Big|_{z_1=1}$

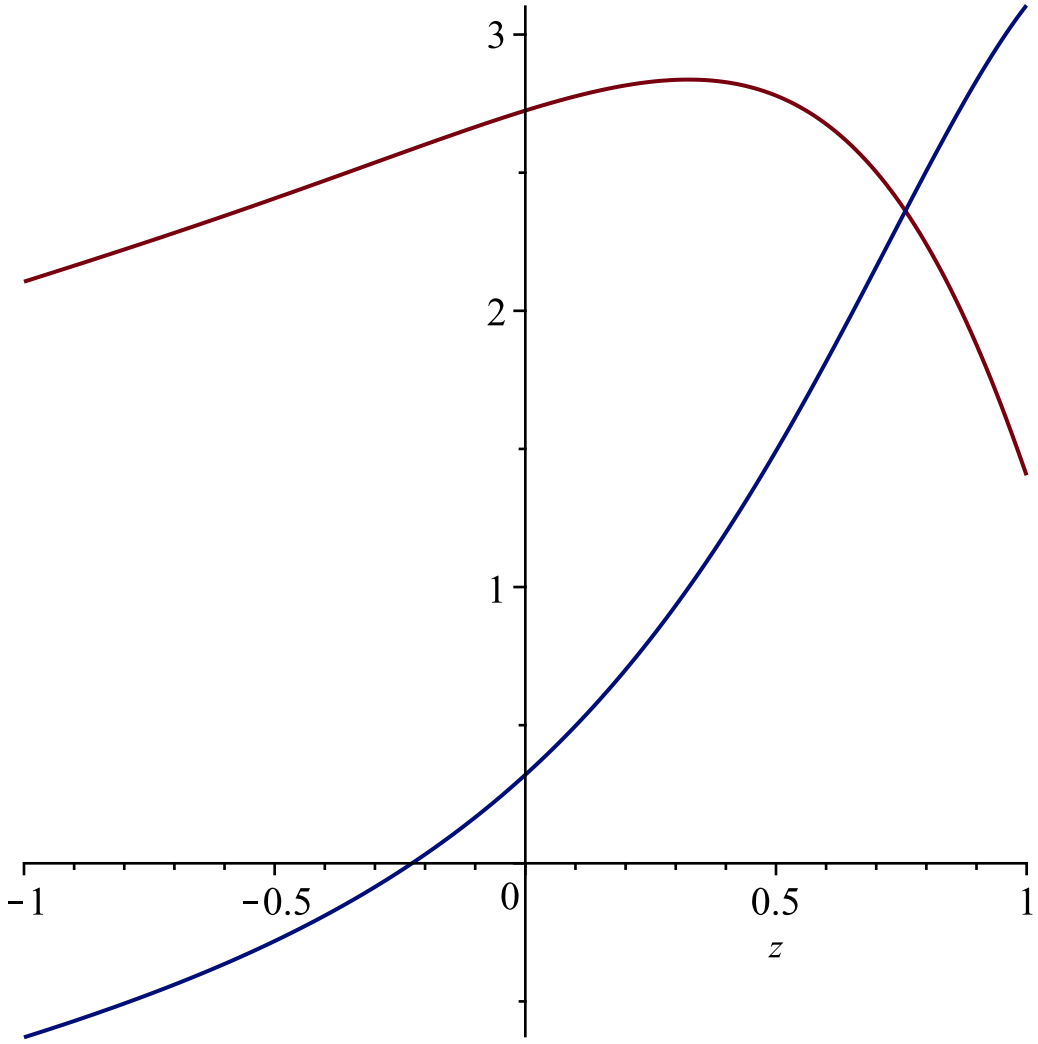
$$AppellF4(a, b, c_1, c_2, 1, z_2) = \frac{1}{\Gamma(c_2-a)\,\Gamma(b)}\left(\Gamma(c_2)\,\Gamma(b-a)\,(-z_2)^{-a}\,\text{hypergeom}\left(\left[a, a-c_2+1, \frac{1}{2}a-\frac{1}{2}b+\frac{1}{2}c_1, \frac{1}{2}a-\frac{1}{2}b+\frac{1}{2}+\frac{1}{2}c_1\right], \left[c_1, a-b+1, a-b+c_1\right], \frac{4}{z_2}\right)\right) + \frac{1}{\Gamma(c_2-b)\,\Gamma(a)}\left(\Gamma(c_2)\,\Gamma(a-b)\,(-z_2)^{-b}\,\text{hypergeom}\left(\left[b, 1+b-c_2, \frac{1}{2}b-\frac{1}{2}a+\frac{1}{2}c_1, \frac{1}{2}b-\frac{1}{2}a+\frac{1}{2}+\frac{1}{2}c_1\right], \left[c_1, b-a+1, b-a+c_1\right], \frac{4}{z_2}\right)\right) \tag{1.1.9}$$

A plot of the AppellF2 function for some values of its parameters

> $F2 := AppellF2\left(\exp(I\cdot z),\ \frac{1}{2}\cdot I,\ \frac{3}{7}-\frac{I}{4},\ 4,\ \frac{5}{7}+6\cdot I,\ 6,\ z\right)$

$$F2 := F_2\left(e^{Iz},\ \frac{I}{2},\ \frac{3}{7}-\frac{I}{4},\ 4,\ \frac{5}{7}+6\,I,\ 6,\ z\right) \qquad \textbf{(1.1.10)}$$

> $plot([\mathrm{Re}, \mathrm{Im}](F2), z=-1..1)$



A thorough set with the main symbolic properties of any of the four Appell functions, for instance for AppellF3, can be seen via

> $FunctionAdvisor(AppellF3)$

AppellF3

describe

definition

$$F_3(a_1, a_2, b_1, b_2, c, z_1, z_2) = \sum_{\_k1=0}^{\infty} \sum_{\_k2=0}^{\infty} \frac{(a_1)_{\_k1} (a_2)_{\_k2} (b_1)_{\_k1} (b_2)_{\_k2} z_1^{\_k1} z_2^{\_k2}}{(c)_{\_k1+\_k2} \_k1!\ \_k2!} \quad \left| |z_1| < 1 \wedge |z_2| < 1 \right.$$

▶ classify function

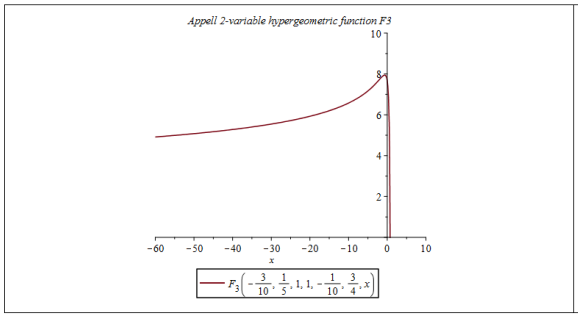
▼ symmetries

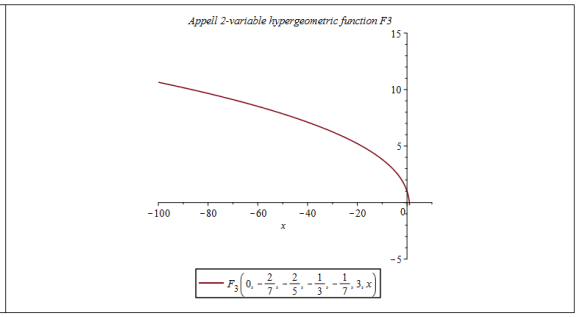
$$F_3(a_2, a_1, b_2, b_1, c, z_2, z_1) = F_3(a_1, a_2, b_1, b_2, c, z_1, z_2)$$

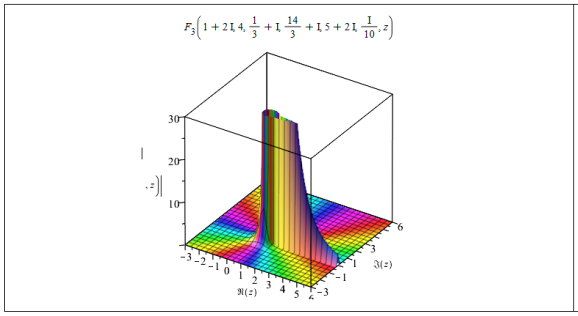
$$F_3(b_1, a_2, a_1, b_2, c, z_1, z_2) = F_3(a_1, a_2, b_1, b_2, c, z_1, z_2)$$

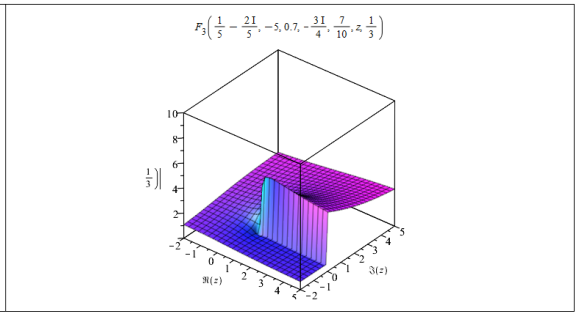
$$F_3(a_1, b_2, b_1, a_2, c, z_1, z_2) = F_3(a_1, a_2, b_1, b_2, c, z_1, z_2)$$

▼ plot









▶ singularities

▶ branch points

▶ branch cuts

▶ special values

▶ **identities**

▶ **sum form**

▶ **series**

▶ **integral form**

▼ **differentiation rule**

$$\frac{\partial}{\partial z_1} F_3\left(a_1, a_2, b_1, b_2, c, z_1, z_2\right) = \frac{a_1 b_1 F_3\left(a_1+1, a_2, b_1+1, b_2, c+1, z_1, z_2\right)}{c}$$

$$\frac{\partial^n}{\partial z_1{}^n} F_3\left(a_1, a_2, b_1, b_2, c, z_1, z_2\right) = \frac{\left(a_1\right)_n \left(b_1\right)_n F_3\left(n+a_1, a_2, n+b_1, b_2, n+c, z_1, z_2\right)}{\left(c\right)_n}$$

$$\frac{\partial}{\partial z_2} F_3\left(a_1, a_2, b_1, b_2, c, z_1, z_2\right) = \frac{a_2 b_2 F_3\left(a_1, a_2+1, b_1, b_2+1, c+1, z_1, z_2\right)}{c}$$

$$\frac{\partial^n}{\partial z_2{}^n} F_3\left(a_1, a_2, b_1, b_2, c, z_1, z_2\right) = \frac{\left(a_2\right)_n \left(b_2\right)_n F_3\left(a_1, n+a_2, b_1, n+b_2, n+c, z_1, z_2\right)}{\left(c\right)_n}$$

▶ **DE**

▼ The Evalf Package

Evalf is both a command and a package of commands for the numerical evaluation of mathematical expressions and functions, numerical experimentation, and fast development of numerical algorithms, taking advantage of the advanced symbolic capabilities of the Maple computer algebra system.

As an environment for working with special functions, Evalf helps developing/implementing the typical approaches used in the literature and comparing their performances. This kind of environment is increasingly relevant nowadays, when rather complicated mathematical expressions and advanced special functions, as for instance is the case of the Heun and Appell functions, appear more and more in the modeling of problems in science.

Examples

```
> with(MathematicalFunctions, Evalf) :
  with(Evalf);
```

$$\{Add, Evalb, Zoom, QuadrantNumbers, Singularities, GenerateRecurrence\} \quad \textbf{(2.1.1)}$$

Consider the following AppellF4 function

```
> F4 := AppellF4(1, 2, 3, 4, 5, z)
```

$$F4 := F_4(1, 2, 3, 4, 5, z) \quad \textbf{(2.1.2)}$$

This function satisfies a linear differential equation whose singularities, which depend on the function's parameters, are relevant in the context of numerically evaluating the function. To see the location of these singularities you can construct the linear ODE behind *F4* using PDEtools:-dpolyform) and use the DEtools:-singularities command, or directly use Evalf:-Singularities

```
> S := Singularities(F4)
```

$$S := \left[0, 6 - 2\sqrt{5}, \frac{20}{3}, 6 + 2\sqrt{5}, \infty + \infty\, I\right] \quad \textbf{(2.1.3)}$$

For the purpose of experimentation with numerical methods, it is useful to have arrays of points located in different quadrants and different regions of the complex plane. For example, in connection with the singularities of *F4*, you can get an Array of floating-point complex numbers in the four quadrants around the absolute value of these singularities using Evalf:-QuadrantNumbers

```
> A := QuadrantNumbers(around = S)
```

$$\begin{aligned} A := {} & [[0.09096185863 + 1.205783116\, I,\ 1.255134238 + 0.8712098412\, I,\\ & 4.939576781 + 2.417014449\, I,\ 5.093003040 + 4.301832689\, I,\ 5.159673340\\ & + 8.714817536\, I,\ 10.24799305 + 2.155056773\, I,\ 56.48956305 + 76.36169567\, I\\ &],\\ & [-0.08511545637 + 1.206209908\, I,\ -1.081827151 + 1.078896918\, I,\\ & -2.959168243 + 4.635159213\, I,\ -2.189287060 + 6.296941052\, I,\\ & -10.08592081 + 0.9189532694\, I,\ -9.881182611 + 3.468120756\, I,\\ & -74.43594928 + 59.00397237\, I],\\ & [-0.2728483263 - 1.178024098\, I,\ -0.9255276779 - 1.215634427\, I,\\ & -1.758183662 - 5.210582293\, I,\ -2.156327649 - 6.308303696\, I,\\ & -0.8181035751 - 10.09460154\, I,\ -7.982590950 - 6.778190989\, I,\\ & -92.10357765 - 23.21874853\, I],\\ & [1.163589128 - 0.3290096109\, I,\ 1.506463679 - 0.2548248963\, I,\ 3.383276874\\ & - 4.335298746\, I,\ 3.923925098 - 5.389550656\, I,\ 5.180810882 - 8.702268225\, I,\\ & 0.3272774255 - 10.46702063\, I,\ 58.19478206 - 75.07027801\, I]] \end{aligned} \quad \textbf{(2.1.4)}$$

Check the quadrant location of the points in each of the four components of the Array A

> $map(u \rightarrow map(QuadrantNumbers, u), A)$

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 & 4 & 4 \end{bmatrix} \quad \textbf{(2.1.5)}$$

Use the Evalf:-GenerateRecurrence command to generate a recurrence with which one could numerically evaluate the following AppellF2 function and use it to compute the first four coefficients of its power series expansion around the origin

> $F2 := AppellF2(1, 2, 3, 4, 5, 6, z)$

$$F2 := F_2(1, 2, 3, 4, 5, 6, z) \quad \textbf{(2.1.6)}$$

> $U := GenerateRecurrence(F2)$

$$U := \mathbf{proc}(i) \quad \textbf{(2.1.7)}$$
$$\quad \mathbf{option}\ hfloat, cache;$$
$$\quad if(i::nonnegint, 4/5 * (i\text{^}2 - 4) * thisproc(i-1) / ((i+4) * i) + 1/5 * (i\text{^}3 - i\text{^}2 - 10 * i - 8) * thisproc(i-2) / (i * (i+4) * (i+3)), 'procname(args)')$$
$$\mathbf{end}$$

The procedure returned by Evalf:-GenerateRecurrence is optimized for numerical computations (option hfloat) but it can also compute with exact numbers as the arguments of *F2*

> $(\%sum = sum)\left(U(n) \cdot z^n, n = 0..3\right)$

$$\sum_{n=0}^{3} U(n)\, z^n = {}_2F_1(1, 2; 4; 6) + \frac{3\ {}_2F_1(2, 2; 4; 6)\, z}{5} + \frac{2\ {}_2F_1(2, 3; 4; 6)\, z^2}{5} + \frac{2\ {}_1F_0(2;; 6)\, z^3}{7} \quad \textbf{(2.1.8)}$$

Perform a *numerical double sum*, where both summation ranges are from 0 to infinity, using the Evalf:-Add command, so perform the double sum *until the summation converges up to the value of* Digits. For example, consider the definition of the AppellF2 double series

> $FunctionAdvisor\left(definition, F_2(a, b1, b2, c1, c2, z1, z2)\right)$

$$\left[F_2(a, b1, b2, c1, c2, z1, z2) = \sum_{\_k1=0}^{\infty} \sum_{\_k2=0}^{\infty} \frac{(a)_{\_k1+\_k2}\, (b1)_{\_k1}\, (b2)_{\_k2}\, z1^{\_k1}\, z2^{\_k2}}{(c1)_{\_k1}\, (c2)_{\_k2}\, \_k1!\ \_k2!}, |z1| + |z2| < 1\right] \quad \textbf{(2.1.9)}$$

Following the steps outlined in Evalf:-Add, get the summand and construct a formula-procedure depending on 2 parameters m and n

> $summand := op\left([1,1], rhs\left((2.1.9)_1\right)\right)$

$$summand := \frac{(a)_{\_k1+\_k2}\,(b1)_{\_k1}\,(b2)_{\_k2}\,z1^{-\_k1}\,z2^{-\_k2}}{(c1)_{\_k1}\,(c2)_{\_k2}\,\_k1!\,\_k2!} \tag{2.1.10}$$

> $general\_formula := unapply(summand, \_k1, \_k2)$

$$general\_formula := (\_k1, \_k2) \to (\text{pochhammer}(a, \_k1+\_k2)\ \text{pochhammer}(b1, \_k1)\ \text{pochhammer}(b2, \_k2)\ z1^{-\_k1}\,z2^{-\_k2}) \Big/ (\text{pochhammer}(c1, \_k1)\ \text{pochhammer}(c2, \_k2)\ \_k1!\,\_k2!) \tag{2.1.11}$$

Use this formula to create a routine for the numerical evaluation of AppellF2 valid when the condition $|z1| + |z2| < 1$, shown in the definition, is satisfied

> $`evalf/AF2` := (a, b1, b2, c1, c2, z1, z2) \to Add(subs('[:-a = a, :-b1 = b1, :-b2 = b2, :-c1 = c1, :-c2 = c2, :-z1 = z1, :-z2 = z2]', eval(general\_formula))) :$

where we are calling $AF2$ this representation of AppellF2. Consider now for instance some numerical complex values of the parameters $a, b1, b2, c1, c2, z1, z2$ satisfying $|z1| + |z2| < 1$ and numerically evaluate this $AF2$ representation at those values using the procedure $`evalf/AF2`$ just constructed

> $AF2\left(\frac{1}{2}, \frac{2}{7}, -\frac{1}{4}, 2, \frac{3}{2}+\mathrm{I}, \frac{1}{4}\,\mathrm{I}, \frac{1}{4}\right)$

$$AF2\left(\frac{1}{2}, \frac{2}{7}, -\frac{1}{4}, 2, \frac{3}{2}+\mathrm{I}, \frac{\mathrm{I}}{4}, \frac{1}{4}\right) \tag{2.1.12}$$

> $evalf\left((2.1.12)\right)$

$$0.9831522491 + 0.02726587338\,\mathrm{I} \tag{2.1.13}$$

Compare with the numerical value obtained using the standard Maple AppellF2 function

> $subs\left(AF2 = AppellF2, (2.1.12)\right)$

$$F_2\left(\frac{1}{2}, \frac{2}{7}, -\frac{1}{4}, 2, \frac{3}{2}+\mathrm{I}, \frac{\mathrm{I}}{4}, \frac{1}{4}\right) \tag{2.1.14}$$

> $evalf\left((2.1.14)\right)$

$$0.9831522491 + 0.02726587338\,\mathrm{I} \tag{2.1.15}$$

Consider the following confluent HeunC function

> $HC := \mathrm{HeunC}(1, 2, 3, 4, 5, 6)$

$$HC := HC(1, 2, 3, 4, 5, 6) \tag{2.1.16}$$

You can normally evaluate the function using evalf

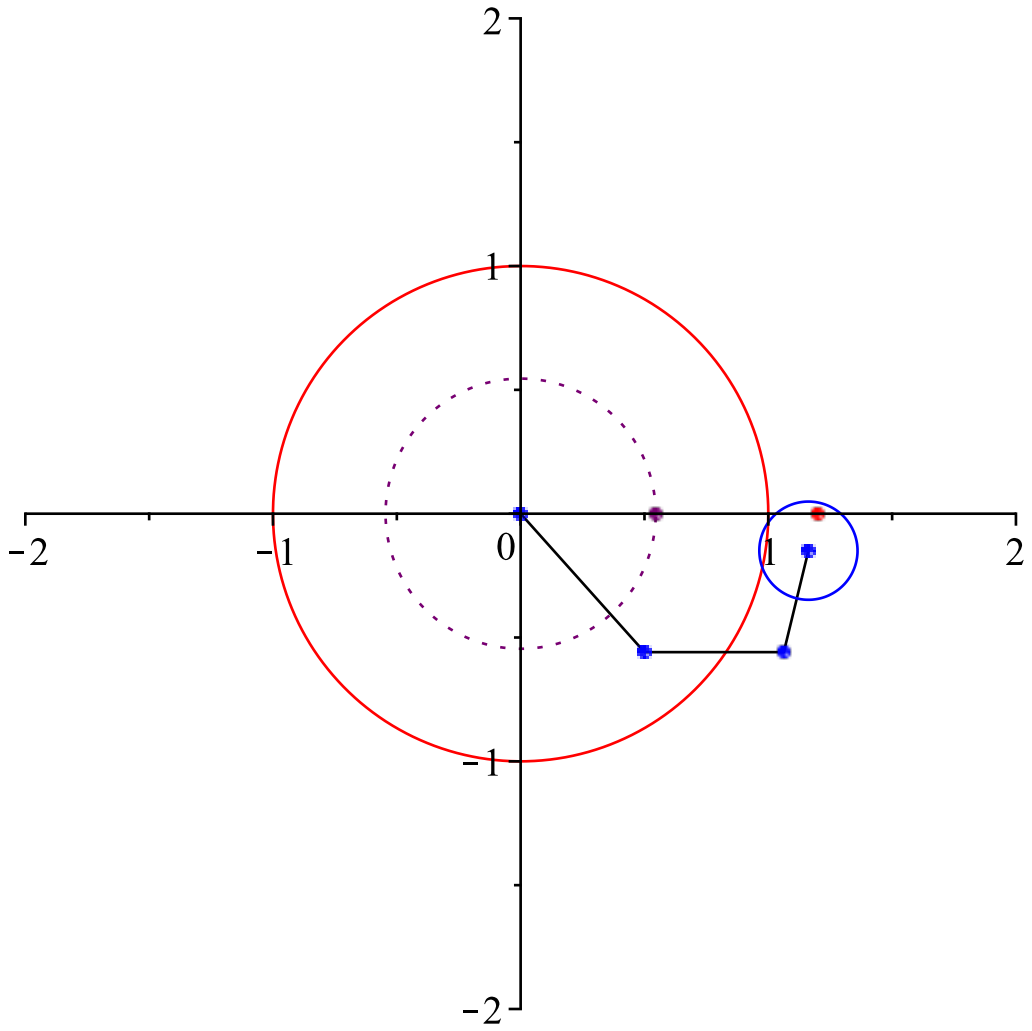
> $evalf(HC)$

$$-0.009083056536 - 0.004476814952\,\mathrm{I} \tag{2.1.17}$$

But how was this evaluation performed? Using Evalf, instead of a black box approach, you have access to information describing the approach used:

> $Evalf(HC)$

```
      HeunCZ and HeunCZPrime at Z = .500000000000
-.559016994375*I using a series expansion around Z = 0
      HeunCZ and HeunCZPrime at Z = 1.06250000000
-.559016994375*I using a series expansion around Z =
.500000000000-.559016994375*I
      HeunCZ and HeunCZPrime at Z = 1.16204916259
-.149352378815*I using a series expansion around Z =
1.06250000000-.559016994375*I
      HeunCZ at Z = 1.200000000 using a series expansion
around Z = 1.16204916259-.149352378815*I
```



CPU time elapsed during evaluation: .284 seconds

$$-0.009083056536 - 0.004476814952\,I \qquad \textbf{(2.1.18)}$$

From this information we see that the **Z** approach (see Evalf) was used to first map the original problem at $z = 6$ into a problem within the ring $1 \le Z \le 2$, then a

sequence of concatenated taylor series got used departing from the origin, working around the singularity at $z=1$, finally reaching $Z=1.200000000$ and from there getting the value at $z=6$.

In the case of the Heun or Appell functions, the singularities behind the corresponding differential equations, and hence restricting the radius of convergence of power series expansions, can be viewed using the Evalf:-Singularities command

> $Singularities(HC)$

$$[0, 1] \quad \textbf{(2.1.19)}$$

Evalf can evaluate expressions in general, not just single function calls. Any optional arguments apply when Heun or Appell functions are present in the expression. Consider for instance one of the special values of AppellF1, a case where the function can be represented by a 2F1 hypergeometric function

> $F1 := \%AppellF1(4.0, 2.0, 0.3, 2.3, 1.12, 1.1)$

$$F1 := F_1(4.0, 2.0, 0.3, 2.3, 1.12, 1.1) \quad \textbf{(2.1.20)}$$

> $F1 = value(convert(F1, rational))$

$$F_1(4.0, 2.0, 0.3, 2.3, 1.12, 1.1) = 10000\ {}_2F_1\left(2, 4; \frac{23}{10}; -\frac{1}{5}\right) \quad \textbf{(2.1.21)}$$

The left-hand side is AppellF1 written in inert form, to avoid the automatic representation in terms of 2F1 functions, while the right-hand side involves only a hypergeometric 2F1 function in terms of rational numbers, also to avoid the automatic numerical evaluation of this function. Proceed now with the numerical evaluation of both sides of this equation

> $evalf(\textbf{(2.1.21)})$

$$5325.710910 = 5325.710910 \quad \textbf{(2.1.22)}$$

Compute now showing the strategy used

> $Evalf(\textbf{(2.1.21)})$

```
-> Numerical evaluation of AppellF1(4.0, 2.0, .3, 2.3,
1.12, 1.1)
   -> AppellF1, checking singular cases
   -> AppellF1, trying special values
Hypergeometric: case: c = b1 + b2
   <- special values of AppellF1 successful
```

CPU time elapsed during evaluation: .1e-2 seconds

$$5325.710910 = 5325.710910 \quad \textbf{(2.1.23)}$$

So during the numerical evaluation of AppellF1 the special value of hypergeometric form was identified and used. Numerically evaluate this equation now without using the special values information of AppellF1

> $Evalf(\textbf{(2.1.21)}, usespecialvalues = false)$

```
-> Numerical evaluation of AppellF1(4.0, 2.0, .3, 2.3,
1.12, 1.1)
   -> AppellF1, checking singular cases
   -> AppellF1, trying formulas
      case AKF page 36, after (13), maps into AppellF3
with abs(z2/(z2-1)) < 1
         case: abs(z1) > 1, abs(z2) > 1 and 1.1 <= abs
(z1) < abs(z2); swapping parameters and z1 <-> z2
         -> Numerical evaluation of AppellF3(-1.7, 2.0,
.3, 4.0, 2.3, 11.00000000, 1.12)
            -> AppellF3, checking singular cases
            -> AppellF3, trying special values
            -> AppellF3, trying formulas
```

CPU time elapsed during evaluation: .3e-2 seconds

```
      case AKF page 35, (9), maps into AppellF2 with abs
(1 - z1/z2) < 1
         case: abs(z1) < 1, abs(z2) > 1; swapping
parameters and z1 <-> z2
         -> Numerical evaluation of AppellF2(2.3, 4.0,
.3, 2.3, 2.3, 1.12, -.18181818e-1)
            -> AppellF2, checking singular cases
            -> AppellF2, trying special values
Hypergeometric: case: c1 = a, c2 = a and z1 <> 1 and z2
<> 1
            <- special values of AppellF2 successful
```

CPU time elapsed during evaluation: .2e-2 seconds

CPU time elapsed during evaluation: .15e-1 seconds

$$5325.710904 = 5325.710910 \qquad \textbf{(2.1.24)}$$

In the information above we see that first a mapping into AppellF3 was attempted, it did not succeed, then a mapping into AppellF2 was attempted and that resulted in the correct value. Perform the numerically evaluation once more, this time without mapping into other functions:

> *Evalf*(**(2.1.21)**, *usespecialvalues* = *false*, *allowmapping* = *false*)

```
-> Numerical evaluation of AppellF1(4.0, 2.0, .3, 2.3,
1.12, 1.1)
   -> AppellF1, checking singular cases
   -> AppellF1, trying formulas
      case z1 <> 0 and z1 <> 1 and abs(z1/(z1-1)) < 1;
swapping parameters and recursing
         case AKF page 30, (5.5), abs(z1) > 1, abs(z2) >
1, and abs((z2-z1)/(z2-1)) <= 1; using identity mapping
```

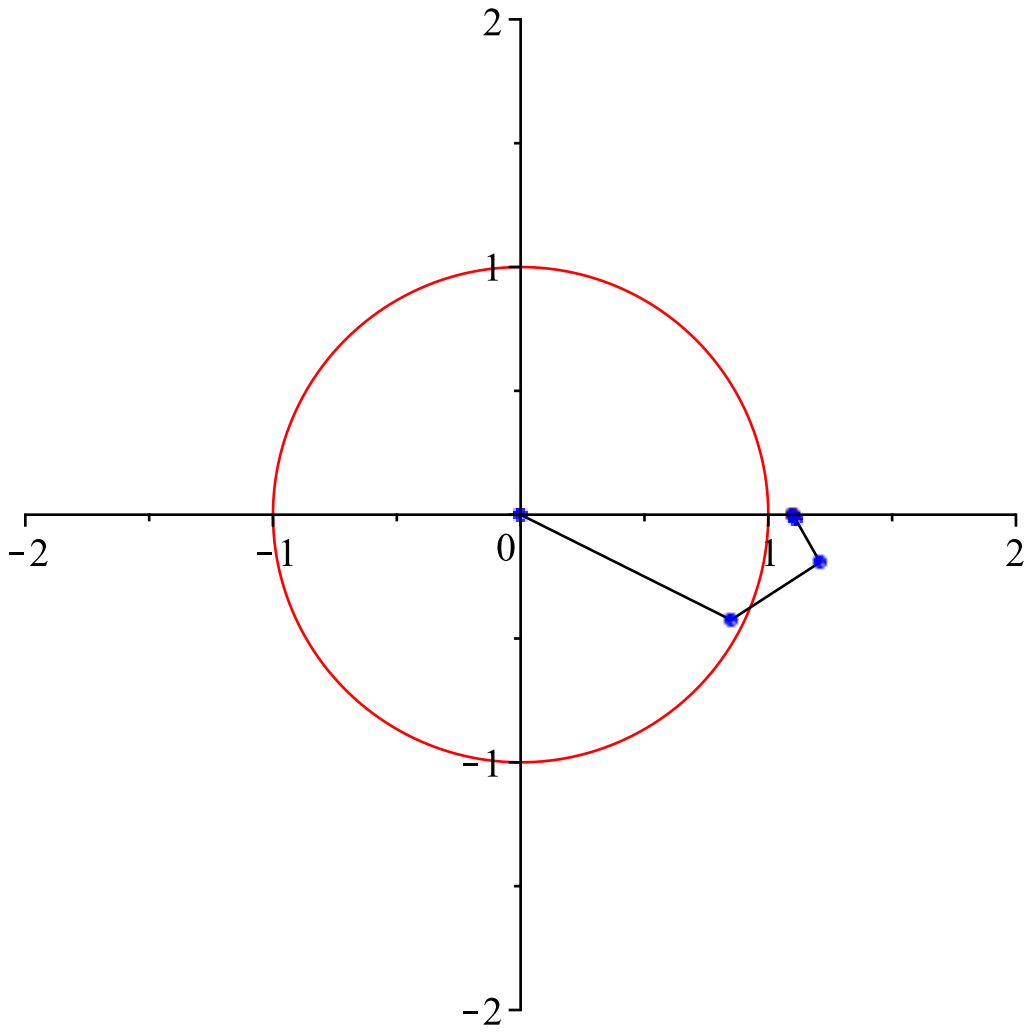
```
into F*AppellF1(..., z1, (z2-z1)/(z2-1))
            -> Numerical evaluation of AppellF1(-1.7, 0.,
2.0, 2.3, 1.1, .1666666667)
               -> AppellF1, checking singular cases
               -> AppellF1, trying special values
               <- special values of AppellF1 successful
```

CPU time elapsed during evaluation: .3e-2 seconds

$$5325.710909 = 5325.710910 \qquad \textbf{(2.1.25)}$$

We see now that there was still another formula that could be used, it is a transformation of Euler type mapping AppellF1 into an equivalent form of AppellF1 that has the advantage of having as evaluation point $\left|\frac{z1}{z1-1}\right| < 1$ so after swapping parameters the function got evaluated again as expected. The same problem using *a sequence of concatenated Taylor expansions*, avoid displaying extra information, just the time consumed and a plot showing the path used and expansion points

> *Evalf*(**(2.1.21)**, *usetaylor*, *time*, *plot*, *quiet*)

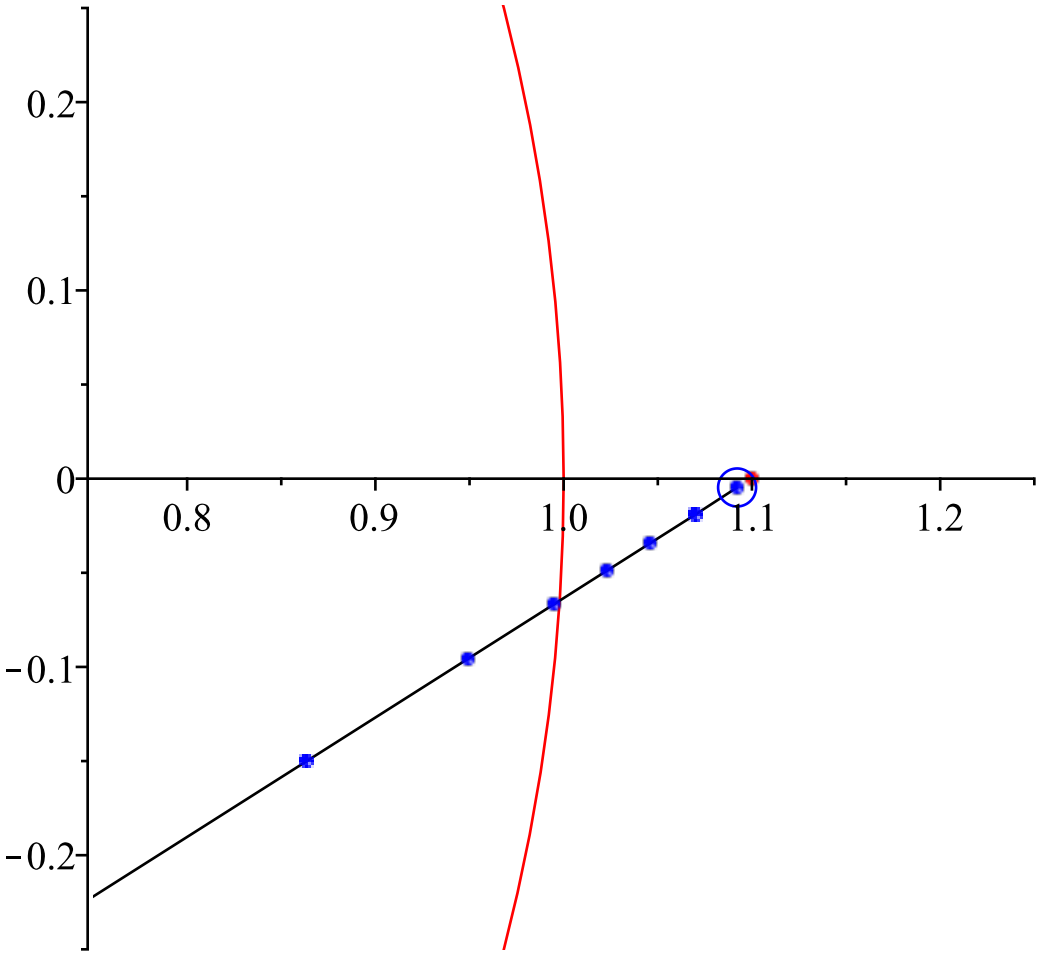


CPU time elapsed during evaluation: .56e-1 seconds

$$5325.710910 + 1.\ 10^{-13}\ \mathrm{I} = 5325.710910 \qquad \textbf{(2.1.26)}$$

In this Taylor approach, each expansion around a point is used to reach up to 95/100 of the radius of convergence before starting another expansion. Reduce that to 1/2, compute internally at Digits = 50 (but return as if computing with Digits = 10) and zoom the plot around $z = 1$ extending 1/4 to either side of the singularity located at $z = 1$

> $Evalf\left[50\right]\left(\textbf{(2.1.21)}, usetaylor, R = \frac{1}{2}, time, plot, zoom = \left[1, \frac{1}{4}\right], quiet\right)$



CPU time elapsed during evaluation: .206 seconds

$$5325.710910 + 1.410240821\ 10^{-52}\ \mathrm{I} = 5325.710910 \qquad \textbf{(2.1.27)}$$