Algorithms to Reduce the Instability of the HGM[*] and Tricks useful for the HGM[†]

Nobuki Takayama

November 8, 2019

The holonomic gradient method (HGM) consists of 3 steps. In the first step we derive a system of linear partial differential equations for a definite integral with parameters by algebraic methods. We need to find an initial condition for the differential equation in the second step. In the last step, the HGM suggests to solve the differential equation numerically and numerically evaluate the integral.

In some applications of the HGM for Fisher-Bingham distribution, Wishart matrices, Bingham distribution, orthant probabilities, the last step is not difficult because the integral to evaluate is a dominant solution of the system of differential equations. The last step requires special care of numerical analysis for some other applications. These things are not well explained in literatures as long as I know. The purpose of this expository paper is to figure out some difficulty of the last step and suppose some methods to make the last step to work well [1].

We provide an appendix of Risa/Asir programs. These are under `defusing_demo/` of OpenXM/Math[2]

1 The Runge-Kutta Method

This is an expository section to explain on the Runge-Kutta method and related topics such as the adaptive Runge-Kutta method, solving an ordinary differential equation numerically in the complex domain, the binary splitting method and other techniques for the matrix factorial.

We consider the linear ordinary differential equation (ODE)

$$\frac{dF}{dt} = P(t)F \tag{1}$$

[*] Holonomic Gradient Method

[†] This is a draft or a private note. Do not circulate it.

[1] This paper quotes several paragraphs and figures of author's preprints. Some parts of this paper are new results, which will be separated to a research paper soon with a title "Methods to reduce the instability of the holonomic gradient method".

[2] `http://www.math.kobe-u.ac.jp/OpenXM/Math`

where $P(t)$ is an $r \times r$ matrix and $F(t)$ is a column vector valued unknown function.

The 4th order Runge-Kutta method is given as

$$k_{i+1} = P(t_0 + c_{i+1}h)(F_0 + a_{i+1}k_i h), \quad k_0 = 0 \tag{2}$$

$$F_1 = F_0 + h(b_1 k_1 + b_2 k_2 + b_3 k_3 + b_4 k_4) \tag{3}$$

Determine the constants so that $F_1 - F(t_0 + h) = O(h^5)$[3] where $F(t)$ is the solution with the initial condition $F(t_0) = F_0$ and $a_1 = c_1 = 0$,

$$b_1 = 1/6, b_2 = 1/3, b_3 = 1/3, b_4 = 1/6, c_2 = c_3 = c_4 = 1/2, a_2 = a_3 = 1/2, a_4 = 1.$$

Define

$$F_1 = F_0 + hk_1, \quad k_1 = P(t_0)F_0.$$

Then, we have

$$F(t) - F_1 = F(t_0) + F'(t_0)h + O(h^2) - F_1 = F_0 + P(t_0)F_0 h + O(h^2) - F_1 = O(h^2)$$

This simple recursion may be called first order Runge-Kutta method.

Refer to standard text books, e.g., [6] on more details on the Runge-Kutta method.

1.1 Matrix Factorial

The Runge-Kutta method for linear equation is reduced to a matrix factorial evaluation. Let us explain what it is. We want to solve

$$\frac{dF}{dt} = \tilde{P}(t)F$$

Let $P(t)$ be the numerator matrix and $d(t)$ the denominator polynomial of $\tilde{P}$. Let h be a small number. We put

$$d_0 = \frac{1}{d(t)}, d_1 = \frac{1}{d(t+h)}, d_2 = \frac{1}{d(t+h)}$$

To reduce the computational cost of the matrix for Runge-Kutta method, we firstly express the denominator polynomials by the symbols d_i and utilize computer algebra systems.

We denote by $Q(t, h)$ the matrix for the 4th order Runge-Kutta method. It is expressed as

```
--> load("ak2.rr");
--> QQ=rk_mat2(newmat(2,2,[[0,1],[t,0]]))$
--> base_replace(QQ[0],QQ[1]);
[ 1/24*h^4*t^2+(1/48*h^5+1/2*h^2)*t+1/6*h^3+1 1/6*h^3*t+1/12*h^4+h ]
[ 1/6*h^3*t^2+(1/6*h^4+h)*t+1/24*h^5+1/2*h^2 1/24*h^4*t^2+(1/16*h^5+1/2*h^2)*t+1/48*h^6+1/3*h^3+1 ]
```

[3] vector $= O(h^m)$ means that $|\text{vector}| = O(h^m)$.

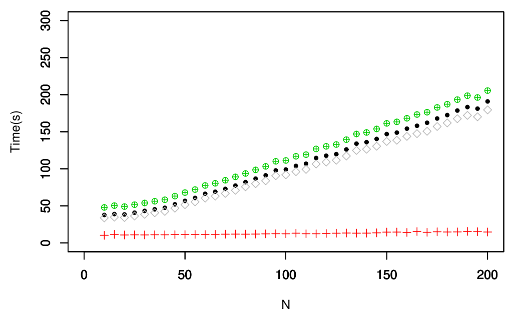


fig:graph-time2-crt-5-32

Figure 1: [8] 5×5 contingency table, a benchmark test of evaluating the normalizing constant (A-hypergeometric polynomial) with 32 processes from [8] . N is a parameter in the marginal sum.

The last output is the matrix $Q(t, h)$.

The function value $F(t_0 + kh)$ is approximated as

$$Q(t_0 + (k-1)h, h) \cdots Q(t_0 + 2h, h)Q(t_0 + h, h)Q(t_0, h)F_0 \tag{4}$$

We call the product $\prod_{i=0}^{k-1} Q(t_o + ih, h)$ the *matrix factorial* of $Q(t, h)$.

Let A be a $d \times n$ matrix with non-negative integral entries. For $\beta \in \mathbf{N}_0^d$, we put

$$Z(\beta; p) = \sum_{Au=\beta, u \in \mathbf{N}_0^n} \frac{p^u}{u!} \tag{5}$$

Fix β. For $u \in \mathbf{N}_0^n$ satisfying $Au = \beta$, the probability $\frac{p^u/u!}{Z(\beta;p)}$ is the conditional probability of the multinomial distribution. The polynomial is called A-hypergeometric polynomial and satisfies the A-hypergeometric system and contiguity relations (matrices of recurrence relations with respect to β). A fast and exact numerical evaluation of matrix factorials is used in [8] to solve the MLE problem of the distribution above theoretically studied in [9] by evaluating matrix factorial of contiguity relation. We suggest the binary splitting method and the modular methods and discuss on advantages of these methods.

The following is a description of the binary splitting of [8].

> It is well-known that the binary splitting method for the evaluation of the factorial $m!$ of a natural number m is faster method than a naive evaluation of the factorial by $m! = m \times (m-1)!$. The binary splitting method evaluates $m(m-1)\cdots(\lfloor m/2 \rfloor + 1)$ and $\lfloor m/2 \rfloor(\lfloor m/2 \rfloor - 1)\cdots 1$ and obtains $m!$. This procedure can be recursively executed. This binary splitting can be easily generalized to our generalized matrix factorial; we may evaluate, for example,

$M(a)M(a+1)\cdots M(\lfloor a/2\rfloor-1)$ and $M(\lfloor a/2\rfloor)\cdots M(-2)$ to obtain $M(a)M(a+1)\cdots M(-2)$, $a<-2$. This procedure can be recursively applied. However, what we want to evaluate is the application of the matrix to the vector $F(-1)$. The matrix multiplication is slower than the linear transformation. Then, we cannot expect that this method is efficient for our problem when the size of the matrix is not small and the length of multiplication is not very long. However, there are cases that the binary splitting method is faster. Here is an output by our package `gtt_ekn3.rr`.

```
[1828] import("gtt_ekn3.rr")$
[4014] cputime(1)$
0sec(1.001e-05sec)
[4015] gtt_ekn3.expectation(Marginal=[[1950,2550,5295],[1350,1785,6660]],
                   P=[[17/100,1,10],[7/50,1,33/10],[1,1,1]]|bs=1)$ //binary splitting
3.192sec(3.19sec)
[4016] gtt_ekn3.expectation(Marginal,P)$
4.156sec(4.157sec)
```

The manual page of `gtt_ekn` is
`http://www.math.kobe-u.ac.jp/OpenXM/Current/doc/asir-contrib/ja/gtt_ekn-html/gtt_ekn-ja.html`

Todo, complexity in p-adic.

1.2 Adaptive Runge-Kutta method

Let F_1 be the vector determined by the Runge-Kutta method (of the 4th order) of the step size $2h$ (not h). Let F_2 be the vector determined by the Runge-Kutta method two times with the step size h.

We have

$$|F(t_0+2h)-F_1| = \phi(2h)^5 + O(h^6) \qquad (6) \quad \boxed{\texttt{eq:2h}}$$

where ϕ depends only on the solution F and t_0, because F_1 is chosen so that the Taylor expansion of $F(t)$ at $t=t_0$ is eliminted by F_1 up to h^4. We assume the ODE is of rank 1 in the sequel and then we will omit $|\ |$ of order estimate. The case of a higher rank ODE can be studied analogously. We also have

$$|F(t_0+2h)-F_2| = \phi h^5 + \phi' h^5 + O(h^6) \qquad (7) \quad \boxed{\texttt{eq:h2}}$$

where ϕ depends only on the solution F and t_0. and ϕ' depends only on the solution F and t_0+h.

Proof. Let $Q(t,h)$ be the Runge-Kutta matrix. $F_2 = Q(t_0+h,h)Q(t_0,h)F_0$. Then, we have

$$\begin{aligned}
& F(t_0+2h) - Q(t_0+h,h)Q(t_0,h)F_0 \\
= \ & F(t_0+2h) - Q(t_0+h,h)Q(t_0,h)F_0 \\
= \ & F(t_0+2h) - Q(t_0+h,h)F(t_0+h) + Q(t_0+h,h)F(t_0+h) - Q(t_0+h,h)Q(t_0,h)F_0 \\
\sim \ & (\phi' h^5 + O(h^6)) + (Q(t_0,h)(\phi h^5 + O(h^6)))
\end{aligned}$$

Since $Q(t_0,h) = E + O(h)$, we have the conclusion. //

Assume $\phi = \phi'$. Taking the difference of (eq:h2 7) and (eq:2h 6), we have

$$F_2 - F_1 = 30\phi h^5 + O(h^6) \tag{8}$$

The good point of this identity is that we can estimate ϕ without knowing the true solution $F(t)$ and estimate the coeficient of the error. We put $\Delta(h) = 30\phi h^5$. Let us assume

$$\Delta = \varepsilon h^5 |F_0| \tag{9}$$

Then, $\phi = |F_0|\varepsilon/30$. Then the relative error $|(F(t+h_0) - F_1)/F_0|$ is bounded by

$$\frac{|\phi| h^5}{|F_0|} + O(h^6) = \frac{\varepsilon}{30} + O(h^6) \tag{10}$$

When we want to make the relative error smaller than $\frac{\varepsilon}{30}$, we need to make $\Delta(h)$ (difference of $2h$ step and two times of h step) smaller than $\varepsilon h^5 |F_0|$.

In order to choose the next h, use the following relation

$$\frac{h_0}{h_1} = \left(\frac{\Delta_0}{\Delta_1}\right)^{1/5}$$

The adaptive Runge-Kutta method is implemented in most of the libraries of numerical solvers. A sample program for GSL[4] is `a26-y.c` for $H_n^k(x, y)$ (see Example ex:Hkn 6 on this function and its applications).

1.3 Solving ODE numerically in the complex domain

Let

$$\frac{d}{dz} F = P(z) F$$

be a differential equation in the complex domain where $P(z)$ is an $r \times r$ matrix and F is a column vector valued function of length r. We want to solve the differential equation along the path

$$z = z_0 + (z_1 - z_0)t, \quad 0 \le t \le 1, z_0, z_1 \in \mathbf{C}$$

with the initial value $F(z_0) = F_0$. Since $d/dz = (z_1 - z_0)^{-1} d/dt$, the differential equation is transformed into

$$\frac{dF}{dt} = (z_1 - z_0) P(z_0 + (z_1 - z_0)t) F \tag{11}$$ eq:ODEin_t

We decompose $(z_1 - z_0)P(z_0 + (z_1 - z_0)t)$ into the real part and the imaginary part as $P_1(t) + \sqrt{-1} P_2(t)$ where we assume t is a real number. Put $F = u + \sqrt{-1} v$ where u, v are column vector valued functions of length r. Since

$$\frac{du}{dt} + \sqrt{-1} \frac{dv}{dt} = P_1 u - P_2 v + \sqrt{-1}(P_1 v + P_2 u),$$

[4] `https://www.gnu.org/software/gsl/`

we obtain the rank $2r$ ODE of real valued unknown functions on $\mathbf{R}$

$$\frac{d}{dt}\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} P_1 & -P_2 \\ P_2 & P_1 \end{pmatrix}\begin{pmatrix} u \\ v \end{pmatrix} \tag{12}$$

eq:ODEonR

We can now use numerical solvers of ODE.
`c2rsys( (`eq:ODEin_t 11`) )` (complex to real system) in `ak2.rr` generates the coefficient matrix of (eq:ODEonR 12).
It utilizes the functions `rat_real_part` and `rat_imaginary_part`.
Example:

```
--> load("ak2.rr")$
--> c2rsys(base_replace((1-@i)*newmat(2,2,[[0,1],[1/z,0]]),[[z,(@i+(1-@i)*t)]]));
 [ 0 1 0 1 ]
 [ (2*t-1)/(2*t^2-2*t+1) 0 (1)/(2*t^2-2*t+1) 0 ]
 [ 0 -1 0 1 ]
 [ (-1)/(2*t^2-2*t+1) 0 (2*t-1)/(2*t^2-2*t+1) 0 ]
```

2 A heuristic method: correction of Initial Value Vector by Eigen Vectors

We have explained some well-known things of the Runge-Kutta method. We will propose a heuristic method to avoid a blow-up of a solution under some situations. This method might be well-known especially in hard efforts of solving stiff ODE's, but I do not find a relevant literature.

We want to find a numerical solution of the initial value problem of the ordinary differential equation

$$\frac{dF}{dt} = P(t)F \tag{13}$$

eq:ode1

$$F(t_0) = F_0^{\mathrm{true}} \in \mathbf{R}^n \tag{14}$$

eq:init1

where $P(t)$ is an $r \times r$ matrix, $F(t)$ is a column vector function of size r, and F_0^{true} is the initial value of F at $t = t_0$.

Solving this problem is the final step of the holonomic gradient method (HGM) [hgm 4]. We often encounter the following situation in the final step.

situation123

Situation 1

1. The initial value has at most 3 digits of accuracy. We denote this initial value F_0.
2. The property $|F| \to 0$ when $t \to +\infty$ is known, e.g., from a background of the statistics.
3. There exists a solution $\tilde{F}$ of (eq:ode1 13) such that $|\tilde{F}| \to +\infty$ or non-zero finite value when $t \to +\infty$.

Under this situation, the HGM works only for a very short interval of t because the error of the initial value vector makes the fake solution $\tilde{F}$ dominant and it hides the true solution $F(t)$. We call this bad behavior of the HGM *the instability of the HGM*.

Example 1

$$\frac{d}{dt}F = \begin{pmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & 0 \end{pmatrix} F$$

The solution space is spanned by $F^1 = (\exp(-t), 0, 0)^T$, $F^2 = (0, \exp(-t), 0)^T$, $F^3 = (1, 1, 1)^T$. The initial value $(1, 0, 0)^T$ at $t = 0$ yields the solution F_1. Add some errors $(1, 10^{-30}, 10^{-30})^T$ to the initial value. Then, we have

t	value F_1 by RK	difference $F_1 - F_1^1$
50	1.92827e-22	9.99959e-31
60	8.75556e-27	1.00000e-30
70	1.39737e-30	$\boxed{\text{1.00000e-30}}$
80	1.00002e-30	$\boxed{\text{1.00000e-30}}$

We can see the instability.

ex:airy1 **Example 2**

$$P(t) = \begin{pmatrix} 0 & 1 \\ t & 0 \end{pmatrix}.$$

This differential equation is obtained from the Airy differential equation

$$y''(t) - ty(t) = 0$$

by putting $F = (y(t), y'(t))^T$. It is well-known that the Airy function

$$\mathrm{Ai}(t) = \frac{1}{\pi} \lim_{b \to +\infty} \int_0^b \cos\left(\frac{s^3}{3} + ts\right) ds$$

is a solution of the Airy differential equation and

$$\begin{aligned} \mathrm{Ai}(0) &= \frac{1}{3^{2/3}\Gamma(2/3)} = 0.355028053887817\cdots \\ \mathrm{Ai}'(0) &= \frac{1}{3^{1/3}\Gamma(1/3)} = -0.258819403792807\cdots \\ \lim_{t \to +\infty} \mathrm{Ai}(t) &= 0 \\ \lim_{t \to +\infty} \mathrm{Ai}'(t) &= 0 \end{aligned}$$

Figure fig:airy 2 is a graph of Airy Ai function and Airy Bi function. The function $F(t) = (\mathrm{Ai}(t), \mathrm{Ai}'(t))^T$ satisfies the condition 2 of the Situation situation123 1 of the instability problem.

We can also see that the condition 3 of the Situation situation123 1 holds by applying the theory of singularity of ordinary differential equations (see, e.g., the manual `DEtools/formal_sol` of Maple and its references on the theory, which has a long history). In fact, the general solution of the Airy differential equation is expressed as

$$\begin{aligned} & C_1 t^{-1/4} \exp\left(-\frac{2}{3}t^{3/2}\right)(1 + O(t^{-3/2})) \\ + \quad & C_2 t^{-1/4} \exp\left(\frac{2}{3}t^{3/2}\right)(1 + O(t^{-3/2})) \end{aligned}$$

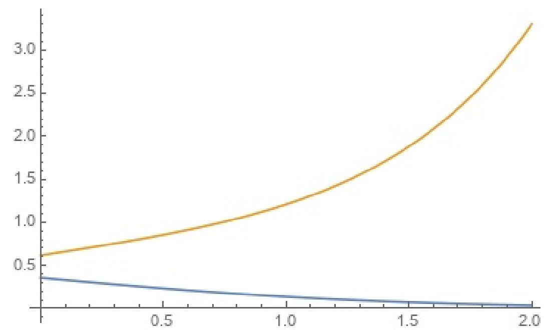


Figure 2: Airy $\mathrm{Ai}(x)$ and $\mathrm{Bi}(x)$ drawn by Mathematica fig:airy

when $t \to +\infty$ where C_i's are arbitrary constants.

We note that the high precision evaluation of the Airy function is studied by several methods (see, e.g., CM2013 [1] and its references). Some mathematical software systems have evaluation functions of the Airy function. For example, `N[AiryAi[10]]` gives the value of $\mathrm{Ai}(10)$ on Mathematica. By utilizing these advanced evaluation methods, we use the Airy differential equation for our test case to check the validity of our heuristic algorithm.

We are going to propose some heuristic methods to avoid the instability problem of the HGM. Numerical schemes such as the Runge-Kutta method obtain a numerical solution by the recurrence

$$F_{k+1} = Q(k,h)F_k \tag{15}$$

from F_0 where $Q(k,h)$[5] is an $r \times r$ matrix determined by a numerical scheme and h is a small number The vector F_k is an approximate value of $F(t)$ at $t = t_k = t_0 + hk$.

Example 3 The Euler method assumes $dF/dt(t)$ is approximated by $(F(t+h) - F(t))/h$ and the scheme of this method is

$$F_{k+1} = (E + hP(t_k))F_k$$

where E is the $r \times r$ identity matrix.

In case that the initial value vector F_0 contains an error, the error may generate a blow-up solution $\tilde{F}$ under the Situation situation123 1 and we cannot obtain the true solution.

Let N be a suitable natural number and put

$$Q = Q(N-1,h)Q(N-2,h)\cdots Q(1,h)Q(0,h) \tag{16}$$

eq:bmatrix

[5] It was denoted by $Q(t_0 + kh, h)$ in the previous section. We denote $Q(t_0 + kh, h)$ by $Q(k,h)$ as long as no confusion arises.

We assume the eigenvalues of Q are positive real and distinct to simplify our presentation. The following heuristic algorithm avoids to get the blow-up solution.

alg:simple **Algorithm 1** *1. Obtain eigenvalues $\lambda_1 > \lambda_2 > \cdots > \lambda_r > 0$ of Q and the corresponding eigenvectors $v_1, \ldots, v_r$.*

2. Let λ_m be the eigenvalue which is almost equal to 0.

3. Express the initial value vector F_0 containing errors in terms of v_i's as

$$F_0 = f_1 v_1 + \cdots + f_r v_r, \quad f_i \in \mathbf{R} \tag{17}$$ eq:F0_by_vi

4. Choose a constant c such that $F_0' := c(f_m v_m + \cdots + f_r v_r)$ approximates F_0.

5. Determine F_N by $F_N = QF_0'$ with the new initial value vector F_0'.

We call this algorithm the *defusing method*. This is a heuristic algorithm and we cannot claim that F_0' gives a better approximation of the initial value vector than F_0 for now, but we can avoid the blow-up of the numerical solution with this method. However, it works well for the Airy differential equation as follows. We will see that this method also works well for the function $H_n^k(1, y)$ in Example ex:Hkn 6.

The function fit_init in ev_ak2.rr performs the steps 3 and 4 of the Algorithm alg:simple 1

```
--> Mat=newmat(2,2,[[0,1],[2,0]])$
--> EE=gsl.eigen_nonsymmv(Mat)$
// EE[0] is the set of the eigenvalues and EE[1] is the set
// of the corresponding eigenvectors

--> D=fit_init([1,3],[[1,0],[1,1]]);
--> D;
[[ 1 3 ],[[c_1,3],[c_0,-2]],[[ 1 0 ],[ 1 1 ]]]
// since [1,0],[1,1] spans the whole space, the answer agrees with the input [1,3].

--> base_replace(c_0*D[2][0]+c_1*D[2][1], Rule=D[1]);
[ 1 3 ]

--> fit_init(newvect(2,[1,2]),[[1.1,2.1+@i]]);
...
[[ 1 0.909090909090909*ii+1.90909090909091 ],[[c_0,10/11]],[[ 11/10 ii+21/10 ]]]  // ii = @i
```

Example 4 We set $t_0 = 0$, $h = 10^{-3}$, $N = 10 \times 10^3$ and use the 4-th order Runge-Kutta scheme. We have $\lambda_1 = 9.708 \times 10^9$, $v_1 = (-5.097, -159.919)^T$ and $\lambda_2 = 3.247 \times 10^{-7}$, $v_2 = (-5.09798, 37.1648136496805760375399714182094650 86)^T = (a, b)$ Then, $m = 2$. We assume the 3 digits accuracy of the value Ai(0) as 0.355 and set $F_0' = (0.355, 0.355b/a)$. Then, the obtained value F_{5000} at $t = 5$ is $(0.000108088745179140, -0.000246853220440734)$. We have the following accurate value by Mathematica

```
In[1]:= N[AiryAi[5]]
Out[1]= 0.000108344
In[2]:= N[D[AiryAi[x],{x}] /. {x->5}]
Out[2]= -0.000247414
```

Note that 3 digits accuracy has been kept for the value Ai(5). On the other hand, we appy the 4th order Runge-Kutta method with $h = 10^{-3}$ for $F_0 = (0.355, -0.259)^T$, which has the accuracy of 3 digits. It gives the value at $t = 5$ as $(-0.147395, -0.322215)$, which is a completely wrong value, and the value at $t = 10$ as $(-102173, -320491)$, which is a blow-up solution.

This heuristic algorithm avoids the blow-up of the numerical solution. Moreover, when the numerical scheme gives a good approximate solution for the exact initial value, we can give an error estimate of the solution by our algorithm. Let $|\cdot|$ be the Eucledian norm.

lem:error1

Lemma 1 *Let $F(t)$ be the solution. When $|QF_0^{\rm true} - F(Nh)| < \delta$ holds, we have*

$$|QF_0' - F(Nh)| < |QF_0'| + |F(Nh)| + 2\delta \tag{18}$$

for any $F_0' \in \mathbf{R}^n$.

Proof. It is a consequence of the triangular inequality. In fact, we have

$$\begin{aligned}
& |QF_0' - F(Nh)| \\
= \ & |QF_0' - QF_0^{\rm true} + QF_0^{\rm true} - F(Nh)| \\
\leq \ & |QF_0' - QF_0^{\rm true}| + |QF_0^{\rm true} - F(Nh)| \\
\leq \ & |QF_0'| + |QF_0^{\rm true}| + \delta \\
\leq \ & |QF_0'| + |F(Nh)| + 2\delta
\end{aligned}$$

//

Under the Situation situation123 1, $|F(Nh)|$ is small enough. Then, it follows from the Lemma that $|QF_0'|$ should be small. In this context, our algorithm is not heuristic and we can give an error estimate of our algorithm. However, our numerical experiments present that the algorithm shows a better behavior than this theoretical error estimate. Then, we would like to classify our defusing method as a heuristic method.

3 Variations of the Defusing Method

We have illustrated our heuristic algorithm in the simplest form in the Algorithm alg:simple 1. We will present some variations of the algorithm.

The first variation: We appy the algorithm to obtain the local solutions near a singularity before applying our heuristic defusing method.

Let us explain this method by the example of the Airy differential equation. We put $t = x^2$. Then the differential equation transformed into

$$xf'' - f' - 4xf = 0, \quad f(x) = y(x^2) \quad (y(t) \text{ is a solution of the Airy differential equation}).$$

We denote x by t in the sequel. The asymptotic series solutions of this differential equation at the infinity can be obtain by algorithmic was as

```
# Maple
--> with(DEtools);
--> formal_sol(t*Dt^2-Dt-4*t,[Dt,t],t=infinity);
```

and they are spanned by

$$t^{-1/2} \exp\left(-\frac{2}{3}t^3\right)(1 + O(t^{-3})), \quad t^{-1/2} \exp\left(\frac{2}{3}t^3\right)(1 + O(t^{-3}))$$

We replace the unknown function $f(t)$ by $g(t)\exp(-(2/3)t^3)$. Then, the function $g(t)$ satisfies

$$tg'' - (4t^3 + 1)g' - 2t^2 g = 0 \tag{19}$$

eq:modified_airy

We have $g(t) = f(t)\exp((2/3)t^3) = y(t^2)\exp((2/3)t^3)$.

Example 5 We set $t_0 = 1$, $h = 10^{-3}$, $N = 1.5 \times 10^3$ and use the 4-th order Runge-Kutta scheme. We have $\lambda_1 = 1.1290 \times 10^{10}$, $v_1 = (-0.040271, -0.99918)$ and $\lambda_2 = 0.66834$, $v_2 = (-0.94307, 0.33257)$. Then, we choose $m = 2$. We give 3 digit accurate value for $\mathrm{Ai}(1) \sim 0.135$.

t^2	$\mathrm{Ai}(t^2)$ by our Algorithm	Exact value
1	0.135	0.135292
4	0.000951564	0.00094928
4.9997	0.00010816	0.000108419
5.9976	0.0000100073	0.0000099654

There is a loss of accuracy, but we have no blowup.

```
--> load("2gauge.rr")$
--> rk_mairy(2000);   // mairy means "m"odified "airy"
```

In the Lemma lem:error1 1, the error is bounded by $|QF_0'|$ where F_0' is a given initial value. Let us see the shape of $\{v \in \mathbf{R}^2 \,|\, |Qv|^2 = c\}$ where $c > 0$ is a constant. Since $|Qv|^2 = v^T Q^T Qv$, this is a quadratic form with respect to v and the eigenvalues of $Q^T Q$ determines the shape of this. The eigenvalues and eigenvectors are $\lambda_1 = 1.3956 \times 10^{18}$, $v_1 = (-0.33257, -0.94307)$ and $\lambda_2 = 0.40798$ $v_2 = (-0.94307, 0.33257)$. Then, it is the ellipsoid of almost crushed shape. When F_0' belongs to the eigenspace of λ_2, $|QF_0'|^2 = \lambda_2 |F_0'|^2$ and this choice makes the error minimum. In our example, $|F_0'|^2 \lambda_2$ is equal to 0.031715.

We have shown that the defusing method works for the Airy function. Does it work for HGM problems? We apply the defusing method (removing components belonging to some eigenspaces, a heuristic method) to the evaluation of

$$H_n^k(x,y) = \int_0^x t^k e^{-t} {}_0F_1(;n;yt)\mathrm{d}t.$$

by solving differential equation with respect to y. The equation is unstable as shown in [5]. The slides[6] contain numerical experiments and some analysis on this unstability. Although the accuracy is not very high, but it seems to work well as long as we do not need a very high accuracy results.

Kang-Alouini show that the outage probability of a communication system can be expressed by this function H_n^k.

Theorem 1 [2] *When the matrix $\Sigma^{-1}MM^{*}$*[7] *has the positive eigenvalues $0 < y_1 < y_2 < \cdots < y_s$, then the cummulative distribution function of the largest eigenvalue ϕ_s of S for the threshold x is*

$$\mathrm{P}\left(\phi_s \leq x\right) = \frac{\exp(-\sum_{i=1}^{s} \lambda_i)}{\Gamma(t-s+1)^s \prod_{1\leq i<j\leq s}(\lambda_j - \lambda_i)} \det \Psi(x)$$

where $\Psi(x)$ is a matrix valued function of which (i,j) element is

$$H_{t-s+1}^{t-i}(x, \lambda_j) = \int_0^x y^{t-i} \exp(-y)\, {}_0F_1(\,; t-s+1; y\lambda_j)\, dy$$

Proposition 1 ([5]) *The function $u = H_n^k(x,y)$ satisfies*

$$\begin{aligned} \{\theta_y(\theta_y + n - 1) + y(\theta_x - \theta_y - k - 1)\} \bullet u &= 0, \\ (\theta_x - \theta_y - k - 1 + x)\, \theta_x \bullet u &= 0. \end{aligned}$$

where $\theta_x = x\frac{\partial}{\partial x}, \theta_y = y\frac{\partial}{\partial y}$. The holonomic rank of this system is 4.

When $y \to +\infty$, solutions of the system has the following asymptotic behavior. It is shown by the `DEtools[formal_sol]` function of Maple.

$$\begin{aligned} h_1 &= (xy)^{-1/2(1/2+n)} \exp(-2(xy)^{1/2})(1 + O(1/y^{1/2})), \\ h_2 &= y^{-k-1}(1 + O(1/y)), \\ h_3 &= (xy)^{-1/2(1/2+n)} \exp(2(xy)^{1/2})(1 + O(1/y^{1/2})), \\ h_4 &= y^{1-n+k} \exp(y)(1 + O(1/y)), \end{aligned}$$

What is the asymptotic behavior of the function $H_n^k(x,y)$ where x is fixed. We compare the value of h_4 and the value by a numerical integration in Mathematica[8].

[6] `http://www.math.kobe-u.ac.jp/HOME/taka/2017/ohp-20171109-shinshu.pdf`

[7] channel matrix H is $N_T \times N_r$ complex valued random matrix. The column vector X satisfies $E[X] = M$ and the convariance is Σ^{-1}. $S = \Sigma^{-1}HH^*$.

[8] The method to evaluate hypergeometric functions in Mathematica is still a black box. It is not easy to give a numerical evaluator of hypergeometric functions which matches to Mathematica in all ranges of parameters and independent variables.

y	Ratio
1000	7.36595030875893e-452
2000	2.64621603289928e-881
3000	2.67723893601667e-1311

where Ratio $= (H_1^{10}(1/2, y))/(y^{1-n+k}\exp(y))$.

This computational experiments suggest that H_n^k is expressed by h_1, h_2, h_3 without the dominant component h_4.

ex:Hkn **Example 6** We apply the defusing method to $H_1^{10}(1, y)$ with $h = 10^{-3}$ and `setprec(30)` as

```
--> load("test3-ak2.rr");
--> setprec(30);
--> Ans=hkn_y_multi_defused( | to=1001, strategy=3)$ // long
--> Ans[0][0];
[1000.88499999999999999999998444,
 [ 15918931785190855105587.3578603 10759414054929503303.233084211 -9335569354914607056 1.700159933 -971947175204947
--> Ans[0][1];
[999.883999999999999999999984454,
  [ 15471988569396134006 33.6203503 10462858139591482973.498407182 -90774017565658103232.026934747 -94509067405993
```

The exact values compared are evaluated by the numerical integrator of Mathematica as

```
--> hh[k_,n_,x_,y_]:=NIntegrate[t^k*Exp[-t]*HypergeometricPFQ[{},{n},t*y],{t,0,x}];
--> hh[10,1,1,1000]
```

The Figure 3 shows that the adaptive Runge-Kutta method fails before y becomes 30. The Figure 4 presents the relative error of values by the defusing method and exact values. It shows that the defusing method works even when $y = 10^3$.

In [5], the cases of $N_T = 5, N_R = 7$, $y = [0.4, 6] \times 10^8$, $10^3? \leq x \leq 2 \times 10^8$. are studied. The differential equation with respect to x, which is obtained by a block diagonalization of the system of rank 4, is used. The initial value is evaluated by the numerical integration around small $x = 1000$ and the large y near 10^8. We have tried our defusing method only upto $y = 10^3$, because we have not yet made an efficient implementation of the method.

4 A method to obtain a stabile system

We gave a notion of a stabile linear ODE and it is announced in [5] that any linear ODE can be transformed into a stabile system for a target function in an algorithmic way.

Let us review the definition of a stabile ODE for a target function $f(x)$ following [5].

Consider a holonomic function $f(x)$ that satisfies a LODE of order m and denote with $f_1(x), \ldots, f_m(x)$ its linearly independent solutions. Then, let $f_i(x)$ be the dominant solution for $x \to \infty$, i.e., $|f_i(x)| \geq |f_j(x)|$, $\forall j$. We refer to the LODE as *stabile* for $f(x)$ if $\lim_{x\to\infty} \frac{|f_i(x)|}{|f(x)|} < \infty$. (Note that a LODE is

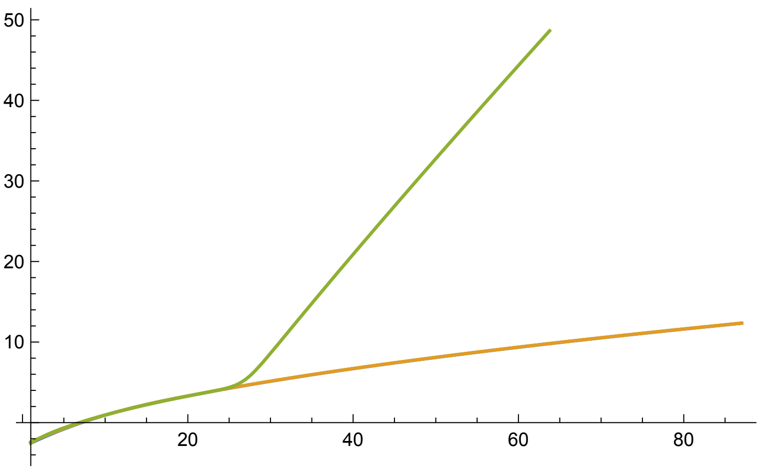


Figure 3: $\log H_1^{10}(1,y)$. Exact value (by numerical integration) and the value by our defusing method agree. The adaptive Runge-Kutta method with the initial relative error 10^{-20} (upper curve) does not agree with the exact value when y is larger than about 25. fig:gsl-RK

stabile or not regardless of the selected set of linearly independent solutions.) The notion of stabile LODE is defined analogously in the case of a vector-valued function, by replacing $|\cdot|$ with a vector norm $||\cdot||$.

em_stabile_LODE_from_gauge **Theorem 2** non-central [5] *From a given LODE system that is not stabile, a lower-dimensional stabile LODE system can be derived algorithmically by gauge transformations.*

This theorem only gives a general scheme and we need some ideas specialized to each problems to give an implementation which works well.

We explain this method and ideas in case of the Airy differential equation $y'' - xy = 0$. Put $F = (y, xy')^T$. Then, we have $x\frac{d}{dx}F = \begin{pmatrix} 0 & 1 \\ x^3 & 1 \end{pmatrix} F$. Changing the independent variable x to t with the relation $x = t^2$, we obtain the system

$$t\frac{d}{dt}F = 2\begin{pmatrix} 0 & 1 \\ t^6 & 1 \end{pmatrix} F. \quad (20)$$ eq:20191105a

When F stands for the Airy function $\mathrm{Ai}(t^2)$, the first component of F has the asymptotic behavir $y(t) = t^{-1/2}\exp(-(2/3)t^3)\cdot O(1)$. Devide the both hand sides of (eq:20191105a 20) by t. We want to transform the differential equation

$$\frac{dF}{dt} = \begin{pmatrix} 0 & 2/t \\ 2t^5 & 2/5 \end{pmatrix} F \quad (21)$$ eq:0930a

into a upper triangular form such that we can obtain the numerical value of the Airy function $\mathrm{Ai}(t^2)$ without the instability caused by the solution $\mathrm{Bi}(t^2)$ which

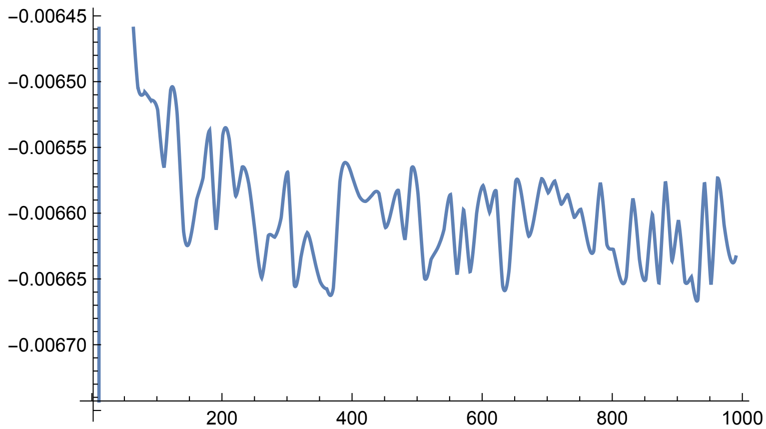


Figure 4: The relative error of $H_1^{10}(1,y)$ of our defusing method. The relative error is defined as $(H_d - H)/H$ where H_d is the value by the defusing method and H is the exact value. fig:relative

grows rapidly as $t^{-1/2}\exp((2/3)t^3)\cdot O(1)$. Note that possible asymptotic behaviors can be obtained by an algorithmic method based on the singularity theory of ODE (see, e.g., Hukuhara-Turrittin reduction, ..., `DEtools:formal_sol` or ISOLDE in Maple).

We put

$$F = \begin{pmatrix} g_1 \\ g_2 \end{pmatrix} = \begin{pmatrix} \tilde{g}_1 t^{-1/2}\exp((2/3)t^3 \\ \tilde{g}_2 t^{5/2}\exp((2/3)t^3) \end{pmatrix} \tag{22}$$

The exponential part is chosen to be the solution $(g_1, g_2)^T$ is the dominant solution. The exponential part is obtained by the algorithmic method of obtaining the possible asymptotic behaviors. The function $(\tilde{g}_1, \tilde{g}_2)^T$ satisfies the following differential equation (try the function `s_airy()[1]` in our expository program `3stabile.rr`).

$$\frac{d}{dt}\begin{pmatrix} \tilde{g}_1 \\ \tilde{g}_2 \end{pmatrix} = \begin{pmatrix} -2t^2 + \frac{1}{2t} & 2t^2 \\ 2t^2 & -2t^2 - \frac{1}{2t} \end{pmatrix} \begin{pmatrix} \tilde{g}_1 \\ \tilde{g}_2 \end{pmatrix} \tag{23}$$

```
--> load("3satible.rr");
--> gen_g1g2(3000);  // generate a table of tilde g1, g2
                     // with the initial condition of Airy Bi
--> Table_g1g2_tilde[0];
  [1,[ 0.619912 0.478729 ]]  //value of tilde g1, g2 at t=1
--> Table_g1g2_tilde[3000];
  [4,[ 0.56512 0.56289 ]]   //value of tilde g1, g2 at t=1
```

Following the algorithm in non-central [5], we apply the Gauge transformation

$$F = \begin{pmatrix} g_1 & 0 \\ g_2 & 1 \end{pmatrix}$$

and obtain the differential equation for H

$$\frac{dH}{dt} = \begin{pmatrix} 0 & 2\tilde{g}_1^{-1} t^{-1/2} \exp(-(2/3)t^3) \\ 0 & -\tilde{g}_2\tilde{g}_1^{-1} 2t^2 + \frac{2}{t} \end{pmatrix} H \tag{24}$$

eq:0930b

Try `s_airy()[0]` in `3stabile.rr` to get this equation. Put $H = (h_1, h_2)^T$. Then, $F = (g_1 h_1, g_2 h_1 + h_2)$.

We solve the subsystem of (24)

$$\frac{d}{dt} h_2 = \left(-\frac{2\tilde{g}_2}{\tilde{g}_1} t^2 + \frac{2}{t} \right) h_2 \tag{25}$$

We expect that $h_2 = \exp(-(2/3)t^3) O(1)$ when $t \to +\infty$. The ODE (24) is stabile for this solution and we will show that this solution gives the second dominant solution of the original system. In this sense, we claim that a stabile system can be obtained in an algorithmic way in [5]. It follows from the Gauge transformation that we have $F = (g_1 h_1, g_2 h_1 + h_2)^T$.

In order to obain the second dominant solution from h_2, we will decompose F as

$$F = \begin{pmatrix} g_1 \\ g_2 \end{pmatrix} h_1(\infty) + \begin{pmatrix} -g_1 \bar{h}_1(t) \\ -g_2 \bar{h}_1(t) + h_2 \end{pmatrix} \tag{26}$$

where the first part of the sum is the first dominant solution and the second part of the sum is the second dominant solution. This decomposition is the key idea to make a practical numerical evaluation. Let us explain what are $\bar{h}_1$ and $h_1(\infty)$.

The function h_1 is determined from (24) by

$$\frac{dh_1}{dt} = h_3(t) \exp(-4t^3/3), \quad h_3(t) = \frac{2t^{-1/2} \tilde{h}_2(t)}{\tilde{g}_1}, \quad \tilde{h}_2(t) = h_2(t) \exp(2t^3/3)$$

```
// contined from the above input
--> gen_h2()$
--> Table_h2_tilde[0];
 [1,-0.513474647706052] // value of tilde h2
```

Fix a point $t = t_0$. Put

$$\bar{h}_1(s) = \int_s^\infty h_3(t) \exp(-4t^3/3) dt. \tag{27}$$

eq:0930c

Then, $h_1(s) = \bar{h}_1(t_0) - \bar{h}_1(s)$ is a solution of the differential equation. The numerical integration of the function $\bar{h}_1(s)$ can be done as follows.

$$\begin{aligned} & \bar{h}_1(s) \exp(4s^3/3) \\ = & \int_s^\infty h_3(t) \exp\left(-\frac{4}{3}(t^3 - s^3) \right) dt \end{aligned} \tag{28}$$

The argument of exp is always negative, and then the numerical integration is easy to perform.

```
--> check3(1000);  // it returns an approximate value of Airy Ai,
                   // which is the second dominant solution.
 ... snip ...
 Ai[3.98801]=0.000973637, Ai[3.992]=0.000965645,
 Ai[3.996]=0.000957712, Ai[4]=0.000949835,
```

We give a rough estimate of the growth order of $g_1\bar{h}_1$ and $g_2\bar{h}_1$. They are bounded by $O(\exp(-2t^3/3))$

A sketch of a formal proof. A formal paritial integration yields

$$\begin{aligned}
\bar{h}_1(s) &= \int_s^\infty h_3(t)\frac{1}{-4t^2}\left(\exp(-4t^3/3)\right)' dt \\
&= \left[h_3(t)\frac{1}{-4t^2}\left(\exp(-4t^3/3)\right)\right]_s^\infty - \int_s^\infty \left(h_3(t)\frac{1}{-4t^2}\right)' \exp(-4t^3/3)dt \\
&= -h_3(s)\frac{1}{-4s^2}\left(\exp(-4s^3/3)\right) + (\text{the integral above})
\end{aligned}$$

Repeating this partial integration, we obtain the estimate $\bar{h}_1(s) = O(\exp(-4t^3/3))$. Since $g_1(s) = O(\exp(2t^3/3))$, we have an estimate for $g_1\bar{h}_1$. $g_2\bar{h} = 1$ can be estimated analogously.

5 Other tips and tricks for HGM

5.1 Using HGM for a subprocedure of a numerical integration

In koyama2019 [7], a generalization of χ^2 distribution is studied motivated by the work of Marumo, Oaku, Takemura MOT2014 [3]. He obtains the following integral formua, which can be numerically evaluated by HGM. He defined the following function φ_3.

$$\varphi_3(s) = \int_0^\infty \exp(-st^r)\exp\left(-\frac{e^{2\pi\sqrt{-1}/r}}{2}t^2\right)dt \quad (29)$$

eq:phi3

for $s > 0$. We will also call this function $\mathrm{Akm}(r; s)$ (modified Ak). The real part and the imaginary part of φ_3 are

$$\begin{aligned}
\mathrm{Re}\,\varphi_3(s) &= \int_0^\infty \exp(-st^r - (\cos(2\pi/r))t^2/2)\cos(\sin(2\pi/r)t^2/2)dt \quad (30) \\
\mathrm{Im}\,\varphi_3(s) &= -\int_0^\infty \exp(-st^r - (\cos(2\pi/r))t^2/2)\sin(\sin(2\pi/r)t^2/2)dt \quad (31)
\end{aligned}$$

eq:phi3_re

eq:ph3_im

`phi3(R,S | diff=K);` in `ak2.rr` returns the real part of $\varphi_3^K(S)$, $r = R$. `phi3(R,S | im=1, diff=K);` returns the imaginary part of $\varphi_3^K(S)$, $r = R$. They are evaluated by the DE numerical integration formula.

Theorem 3 (koyama2019 [7]) *The probability density function* $f(x) = \frac{d}{dx}P(\sum_{k=1}^n X_k^r < x)$ *(X_k's are i.i.d random normal variables, $r \geq 3$) is expressed by the following* *integrals.*

$$f(x) = \frac{1}{\pi}\frac{1}{2\pi^{n/2}}\int_0^\infty \exp(-xs)\,\mathrm{Im}\,\left[\varphi_3(s)\exp(\sqrt{-1}\pi/r)+\varphi_0(s)\right]^n ds, \quad (r \textit{ is odd}) \quad (32)$$ eq:f_odd

$$f(x) = \frac{1}{\pi}\left(\frac{2}{\pi}\right)^{n/2}\int_0^\infty \exp(-xs)\,\mathrm{Im}\,\left[\varphi_3(s)\exp(\sqrt{-1}\pi/r)\right]^n ds, \quad (r \textit{ is even}) \quad (33)$$ eq:f_even

(34)

where

$$\varphi_0(s) = \int_0^\infty \exp(-st^r - t^2/2)dt \quad (35)$$ eq:phi0

These are derived from a Levy type formula of the characteristic function with changes of the path of integration in the complex domain. We evalute the function φ_3 by the HGM by a differential equation shown later. It seems that it is not good method to evaluate $f(x)$ itself by the HGM, because the rank of the holonomic system for the integrand becomes very high when n increases MOT2014 [3]. and it will be a good method to generate a table of φ_3 by the HGM and use a one dimensional numerical integration method to obtain the value of the PDF $f(x)$. Note that the HGM is a good method to generate a table of values.

Trick: use HGM as a subprocedure of a numerical integration.

```
  hgm_f_r4(N=2,X=1);  // From=1, To=2.  H=0.001, N=2, X=1
    0.0969109812000352  // Fast
  psi3_im(R=4,N=2,X=1 | from=1, to=2);  // double integral
    0.0968470258202232  // Slow
```

```
load("test-ak2.rr");
[2795] Ans=hgm_phi3(R=6,X=100)$   // evaluate by hgm.  every 0.1 H=0.001
 ...
Time=[ 41.2335 0 2313312788 41.2705 ]
[2796] Ans[0];
[100,[ (0.422986949995807-0.0123543330871498*@i) (-0.000678813968444877+6.03046590444843e-05*@i) (7.7840232366444
```

The figure fig:pdf-Xr 5 is a set of graphs of $f(x)$.

Let $F(y)$ be the cummulative distribution function (CDF). In other words,

$$F(y) = \int_0^y f(x)dx$$

When we need to specify the r (power) and n (freedom), we denote them by $F_n(r;y)$ and $f_n(r;y)$ respectively.

As an application of the result by T.Koyama koyama2019 [7] we have the following formula.

Proposition 2 *The cummulative distribution function (CDF) is approximately expressed as*

$$F(y) = P(\sum_{i=1}^n X_i^r < y)$$
$$\sim \int_0^b \frac{1-\exp(-ys)}{s}\xi(s)ds + c_\alpha\frac{b^{-\alpha}}{\alpha} - c_\alpha y^\alpha\int_{by}^\infty e^{-t}t^{-\alpha-1}dt \quad (36)$$

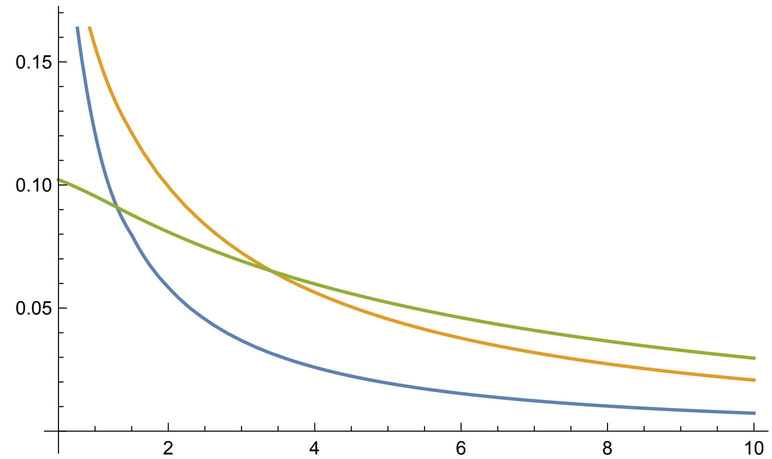


Figure 5: PDF $f(x)$ for $r = 4, n = 1, 3, 5$ fig:pdf-Xr

where b is a sufficiently large number, $\alpha = n/r$, and $\xi(s)$ is given in (37) and (38). eq:xi_odd eq:xi_even

Proof. We will give a method to evalute $F(y)$ with the HGM. We introduce the function $\xi(s)$ to save the space

$$\xi(s) = \frac{1}{\pi}\frac{1}{(2\pi)^{n/2}} \operatorname{Im}\left[\varphi_3(s)\exp(\sqrt{-1}\pi/r) + \varphi_0(s)\right]^n \quad r \text{ is odd} \tag{37}$$
eq:xi_odd

$$\xi(s) = \frac{1}{\pi}\left(\frac{2}{\pi}\right)^{n/2} \operatorname{Im}\left[\varphi_3(s)\exp(\sqrt{-1}\pi/r)\right]^n \quad r \text{ is even} \tag{38}$$
eq:xi_even

We firstly split the integral into two parts.

$$\begin{aligned} F(y) &= \int_0^y dx \int_0^\infty ds \exp(-xs)\xi(s) \\ &= \int_0^\infty ds\xi(s) \int_0^y dx \exp(-xs) \\ &= \int_0^\infty \frac{1-\exp(-ys)}{s}\xi(s)ds \end{aligned}$$

Let $b > 0$ be a number. Put

$$I_1 = = \int_0^b \frac{1-\exp(-ys)}{s}\xi(s)ds \tag{39}$$
eq:cdf_b

$$I_2 = = \int_b^\infty \frac{1-\exp(-ys)}{s}\xi(s)ds \tag{40}$$
eq:cdf_tail

$$\tag{41}$$

Then, $F(y) = I_1 + I_2$. When s is large $\varphi_3(s)$ is approximated by $c_r s^{-1/r}$ for a constant c_r by numerical experiments and the expression of local solutions of the ODE for φ_3. Let r be an even number. Put $\alpha = n/r$ and

$$c_\alpha = \frac{1}{\pi}\left(\frac{2}{\pi}\right)^{n/2} \mathrm{Im}\left(c_r^n \exp(\sqrt{-1}n\pi/r)\right)$$

We approximate I_2 when $y > 0$ in (eq:cdf_tail 40) as follows.

$$\begin{aligned} I_2 &\sim c_\alpha \int_b^\infty \frac{1-\exp(-ys)}{s} s^{-\alpha} ds \\ &= c_\alpha \int_b^\infty s^{-\alpha-1} - c_\alpha \int_b^\infty \exp(-ys) s^{-\alpha-1} ds \\ &= c_\alpha b^{-\alpha}/\alpha - c_\alpha y^\alpha \int_{by}^\infty e^{-t} t^{-\alpha-1} dt \end{aligned}$$

The last integral is the incomplete gamma function. When $y = 0$, we put $I_2 = 0$. //

Here is a method to obtain the CDF. (We have tried for $r = 3, 4, 5, 6$ for the step 1 and for $r = 4$ for the step 2.)

Step 1. Generate a table of values of $\varphi_3(s)$ We use the numerical integration for $s \in [0, 1/10]$ (the step size is 10^{-3}). We solve numerically the differential equation for $s \in [1/10, 10^4]$ with the starting point $s = 1$ (HGM).

Step 2. Evaluate (eq:cdf_b 39) with $b = 1000$ with the table and a numerical integration. Evaluate (eq:cdf_tail 40) by determining the constant c_r by the table. Return $I_1 + I_2$ as the value of $F(y)$.

We evaluate numerically some CDF's. The results are Figures fig:cdf-k-1 fig:cdf-k-2 6 and 7.

```
test7c(R=4,N=1, Y=20);
Pn=xi(0)=0, ds=1/1000, I=0
Pn=xi(1/1000)=5.30476211624276e-07, ds=1/1000, I=1
...
Pn=xi(694)=0.124065695044676, ds=0.99, I=2700
C_alpha=0.637460834571472, Alpha=1/4, Sb=990.01, G=0, myg=0.454572964768408
[0.971613600640028,0.856163639317813,0.115449961322216,0.843171434175572]

By triple integral on Mathematical
mak.m, cdfeven4n1[20]  slwcon warning  ==> 0.964
```

5.2 Exact ODE coefficients are necessary

Let us derive a differential equation for φ_3 in (eq:phi3 29). We consider a little more general integral.

Let $r \geq 3$ be a natural number and we assume $x_1 \in \mathbf{R}_{<0}$, $x_2 \in \sqrt{-1}\mathbf{R}$. Put

$$f(x_1, x_2) = \int_{-\infty}^\infty \exp(x_1 z^2 + x_2 z^r) dz \qquad (42) \quad \boxed{\texttt{eq:ak-f}}$$

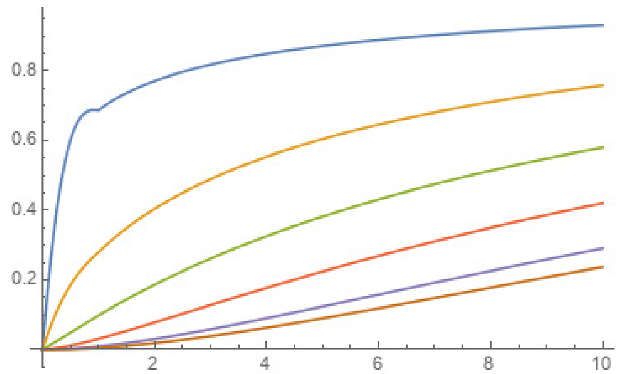


Figure 6: The CDF $F_n(y)$ for $y \in [0, 10]$, $r = 4, n = 1, 3, 5, 7, 9, 10$ (from the top to the bottom). fig:cdf-k-1

Lemma 2 *The function f satisfies the following A-hypergeometric system*

$$(2\theta_1 + r\theta_2 + 1) \bullet f = 0 \tag{43}$$
eq:euler

$$(\partial_1^{r_1} - \partial_2) \bullet f = 0, \quad (r = 2r_1 \textit{ is even}) \tag{44}$$
eq:box2

$$(\partial_1^{r} - \partial_2^2) \bullet f = 0, \quad (r \textit{ is odd}) \tag{45}$$
eq:box1

where $\theta_i = x_i \partial_i = x_i \frac{\partial}{\partial x_i}$.

Proof. Since the integrand is rapidly decaying function with respect z, we may exchange differentiations and the integral sign. The relation (45) or (44) can be obtained by a straightforward calculation. Let us show (43). Since

$$\begin{aligned} & (2\theta_1 + r\theta_2) \bullet \exp(x_1 z^2 + x_2 z^r) \\ = \; & (2x_1 z^2 + r x_2 z^r) \exp(x_1 z^2 + x_2 z^r) \\ = \; & z \frac{\partial}{\partial z} \exp(x_1 z^2 + x_2 z^r), \end{aligned}$$

the relation (43) can be obtained by the integration by parts. //

Note that, from this proof, the integral

$$\int_0^\infty \exp(x_1 z^2 + x_2 z^r) dz$$

also satisfies the same A-hypergeometric system.

We are going to eliminate ∂_1 from the A-hypergeometric system. Let us consider the case that $r = 2r_1$ is even. Multyplying $x_1^{r_1}$ to (44), we obtain

$$\begin{aligned} L &= x_1^{r_1} \partial_1^{r_1} - x_1^{r_1} \partial_2 \\ &= \theta_1(\theta_1 - 1) \cdots (\theta_1 - r_1 + 1) - x_1^{r_1} \partial_2 \end{aligned}$$

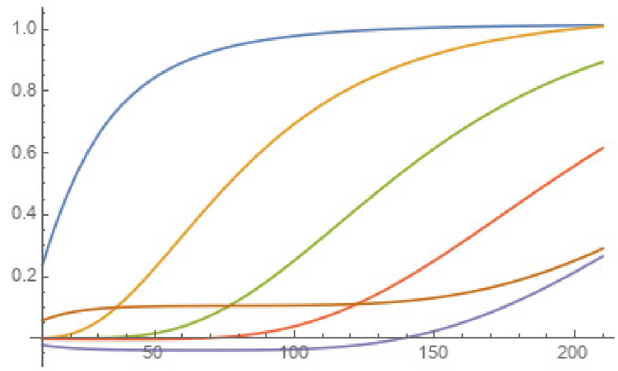


Figure 7: The CDF $F_n(y)$ for $y \in [10, 210]$, $n = 10, 30, 50, 70, 90, 100$. Note that $n = 90, 100$ cases (two lower curves) give wrong values because of numerical error of high powers n. fig:cdf-k-2

From (43) eq:euler, we have $\theta_1 = \frac{-1}{2}(r\theta_2 + 1)$ and substitute θ_1 in L by the righthand side. Then, we have

$$
\begin{aligned}
L &= \prod_{k=0}^{r_1-1}\left(\frac{-r}{2}\theta_2 - \frac{1}{2} - k\right) - x_1^{r_1}\partial_2 \\
&= \left(\frac{-r}{2}\right)^{r_1}\prod_{k=0}^{r_1-1}\left(\theta_2 + \frac{2k+1}{r}\right) - x_1^{r_1}\partial_2
\end{aligned}
$$

We can perform an analogous calculation for the case that r is odd. Thus, we have the following relations.

lem:ode2i **Lemma 3** *Fix x_1 to a number. The function $f(x_1, x_2)$ annihilated by the following ordinary differential operator*

$$\left(\frac{-r}{2}\right)^{r_1}\prod_{k=0}^{r_1-1}\left(\theta_2 + \frac{2k+1}{r}\right) - x_1^{r_1}\partial_2 \quad (r \textit{ is even}) \tag{46}$$ eq:ODEeven

$$\left(\frac{-r}{2}\right)^{r}\prod_{k=0}^{r-1}\left(\theta_2 + \frac{2k+1}{r}\right) - x_1^{r}\partial_2^2 \quad (r \textit{ is odd}) \tag{47}$$ eq:ODEodd

Multiplying $-x_2 x_1^{-r_1}$ to (46) eq:ODEeven, we have

$$\theta_2 - x_2 x_1^{-r_1}\left(\frac{-r}{2}\right)^{r_1}\prod_{k=0}^{r_1-1}\left(\theta_2 + \frac{2k+1}{r}\right)$$

It is the differential operator for the generalized hypergeometric function

$${}_{r_1}F_0\left(\frac{1}{r}, \frac{3}{r}, \ldots, \frac{2r_1-1}{r};; x_2 x_1^{-r_1}\left(\frac{-r}{2}\right)^{r_1}\right).$$

Multiplying $-x_2x_1^{-r}$ to (eq:ODEodd 47), we have

$$\theta_2(\theta_2-1)-x_2^2x_1^{-r}\left(\frac{-r}{2}\right)^r\prod_{k=0}^{r-1}\left(\theta_2+\frac{2k+1}{r}\right)$$

By putting $z=x_2^2x_2^2x_1^{-r}\left(\frac{-r}{2}\right)^r$, we obtain the differential equation for ${}_rF_1$. In fact,

$${}_rF_1\left(\frac{1}{2r},\frac{3}{2r},\ldots,\frac{2r-1}{2r};\frac{1}{2};\left(\frac{-r}{2x_1}\right)^r 2^{r-2}x_2^2\right)$$

is a solution of the ODE.

These discussions yields the following problem.

Problem: Study high precision and aribitrary precision evaluation of the generalized hypergeometric function ${}_rF_1$ globally.

We have

$$\varphi_3(s)=f\left(-\frac{e^{2\pi\sqrt{-1}/r}}{2},-s\right). \tag{48}$$

The differential equation contains the constant $\left(-\frac{e^{2\pi\sqrt{-1}/r}}{2}\right)^m$, $m=r$ or $m=r_1$. The author firstly use approximate value of this constant in the differential equation and obtained stupid values for φ_3. He realized this constant should be an exact value to get an exact matrix factorial for the Runge-Kutta method.

Trick: Exact ODE yields the exact matrix factorial $\prod_k Q(k,h)$.

5.3 Solving ODE in the complex domain

The characteristic function of r powered sum of the normal random variable X_i $\varphi(w)$ is $\frac{1}{\sqrt{2\pi}}f(-1/2,\sqrt{-1}w)$ where f is (eq:ak-f 42). In other words,

$$\varphi(w)=\int_{-\infty}^{\infty}\exp(\sqrt{-1}wx^r)\frac{1}{\sqrt{2\pi}}\exp(-x^2/2)dx \tag{49}$$ eq:airy-koyama-int

This integral is considered in [koyama2019 7] to study the powered sum of independent identically and normally distributed random variables. This function is a generalization of the Airy function or the Airy integral and we call it the Ak integral or the Ak function in this paper to avoid a confusion on the name "generalized Airy function". We use the notation

$$\mathrm{Ak}(r,w)=\varphi(w)=\int_{-\infty}^{\infty}\exp(\sqrt{-1}wx^r)\frac{1}{\sqrt{2\pi}}\exp(-x^2/2)dx \tag{50}$$ eq:Ak

We want to consider the problem of numerical evaluation of this function. As is shown in the example below, solving in the complex domain is useful.

When we change the independent variable, we need to translate the initial value for higher rank ODE. Let us note this fact by an example. Assume we evalute

$$\left(\frac{d}{dw_2}\right)^k\varphi(\sqrt{-1}w_2)=\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}(-t^r)^k\exp(-w_2t^r-t^2/2)dt$$

numerically. Since $w = \sqrt{-1}w_2$, we have $(d/dw)^k = \left(\frac{1}{\sqrt{-1}}\frac{d}{dw_2}\right)^k$. We need multipy $(1/\sqrt{-1})^k$ for the value of the integral above to get the k-th derivative of $\varphi^{(k)}(\sqrt{-1}w_2)$.

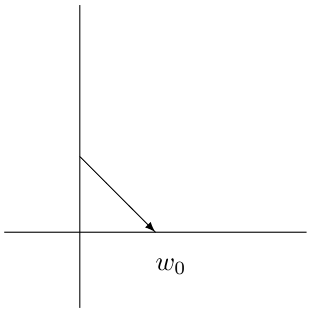


Figure 8: When w_0 is not too small, it works, `ak_even()`

```
ak_even_rec(4,1/100 | fixed_start=1);
      (7.24245451318315e+131+5.74558790493557e+131*@i)
ak_even(4,1/100)
      (0.99535327756843+0.0286105510980157*@i)
```

$\boxed{\text{Trick: Use a good path of integration in the complex domain.}}$

5.4 Using power series and bigfloat for inaccurate data

In [10] (Eeuler2019), the expected Euler characteristic for the largest eigenvalue of a real Wishart matrix is numerically evaluated for a small sized Wishart matrix by HGM. Let $A = (a_{ij})$ be a real $m \times n$ matrix valued random variable (random matrix) with the density

$$p(A)dA, \quad dA = \prod da_{ij}.$$

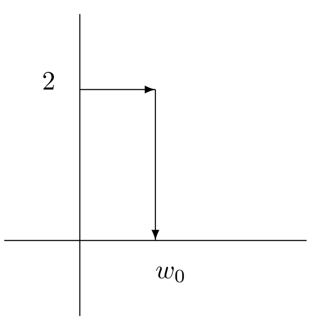


Figure 9: When w_0 is small, the numerical solution makes a blow-up, `ak_even_rec()`

We assume that $p(A)$ is smooth and $n \geq m \geq 2$. Define a manifold

$$M = \{hg^T \mid g \in S^{m-1}, h \in S \in S^{n-1}\} \simeq S^{m-1} \times S^{n-1} / \sim$$

where $(h, g) \sim (-h, -g)$ and h and g are regarded as column vectors and hg^T is a rank 1 $m \times n$ matrix. Put

$$f(U) = \mathrm{tr}(UA) = g^T A h, \quad U \in M$$

and

$$M_x = \{hg^T \in M \mid f(U) = g^T A h \geq x\}$$

Assume $m = n = 2$ and $p(A)$ is a Gaussian distribution

$$p(A)dA = \frac{1}{(2\pi)^{mn/2} \det(\Sigma)^{n/2}} \exp\Big\{-\frac{1}{2} \mathrm{Tr}\,(A-M)^T \Sigma^{-1} (A-M)\Big\} dA.$$

The mean is expressed by the variable $M = (m_{ij})$. We gave an integral representation of $E(\chi(M_x))$ in [10]. Moreover, we derived an ODE of rank 11 for (51) by the computer algebra package `HolonomicFunctions.m`.

$$\begin{aligned} & E[\chi(M_x)] \\ = \quad & \frac{1}{2\pi^2} \int_x^\infty d\sigma \int_{-\infty}^\infty db \int_{-\infty}^\infty ds \int_{-\infty}^\infty dt \frac{s_1 s_2 (\sigma^2 - b^2)}{(1+s^2)(1+t^2)} \exp\Big\{-\frac{1}{2}\tilde{R}\Big\}, \quad (51) \end{aligned}$$

EQ:secondeuler

where $\tilde{R}$ is a rational function in $\sigma, b, s, t, s_1, s_2, m_{11}, m_{21}, m_{22}$. More precisely, put

$$\begin{aligned} R \quad = \quad & s_1 \left(b \sin\theta \sin\phi + \sigma \cos\theta \cos\phi - m_{11}\right)^2 + s_2 \left(\sigma \sin\theta \cos\phi - b \cos\theta \sin\phi - m_{21}\right)^2 \\ & + s_1 \left(\sigma \cos\theta \sin\phi - b \sin\theta \cos\phi\right)^2 + s_2 \left(b \cos\theta \cos\phi + \sigma \sin\theta \sin\phi - m_{22}\right)^2, \end{aligned}$$

replace sin, cos in R by

$$\sin\theta = \frac{2s}{1+s^2}, \quad \cos\theta = \frac{1-s^2}{1+s^2}, \quad \sin\phi = \frac{2t}{1+t^2}, \quad \cos\phi = \frac{1-t^2}{1+t^2}.$$

and we set this $\tilde{R}$. We want to evaluate it when $m_{11} = 1, m_{21} = 2, m_{22} = 3$ (means) and $s_1 = 10^3, s_2 = 10^2$, See [10] as to details.
The following is a quotation from [10]:

> As far as we have tried, it is hard to evaluate (51) for these relatively large parameters s_i by numerical integration (even the Monte Carlo integration). Thus, we take a different approach. Using an algebraic method, we can compute a linear ODE for (51) of rank 11 with respect to the independent variable x. Then we construct series solutions for this differential equation and use them to extrapolate results by simulations.
>
> Although this extrapolation method is well-known, we explain it in a subtle form with application in our evaluation problem. Consider

an ODE with coefficients in $\mathbf{Q}(x)$ of rank r. Let $c \in \mathbf{Q}$ be a point in the x-space and we take r increasing numbers $y_j \in \mathbf{Q}$, where $j = 0, 1, \ldots, r-1$. We construct a series solution $f_i(x)$ as a series in $x - (c + y_i)$. We may further assume that $c + y_i$ is not a singular point of the ODE for each i. The initial value vector may be taken suitably so that the series is determined uniquely over $\mathbf{Q}$.

x	$f(x)$	simulation
3.8133	0.051146	0.051176
3.8166	0.047517	0.047695
3.82	0.044120	0.044515

Figure 10: Eeuler2019 [10] Numerical evaluation by extrapolation series fig:values2

The Figure fig:values2 10 and Figure fig:far 11 are respectively a table of values and a graph obtained by extrapolating simulation values by these power series solutions. We use bigfloat of size 380 to determine series solutions.
Trick: Do not hesitate to use the bigfloat and powerseries. We use series solutions as a basis of interpolation or extrapolation.

6 Computational Challenges and Questions

Computational Try 1 R.Vidunas and A.Takemura RT2016 [11] derived a system of linear partial differential equations for the outage probability $P(\phi_s \leq x)$. Try to make a numerical analysis of this system with Gröbner basis, the defusing method, or the method to obtain a stabile system.

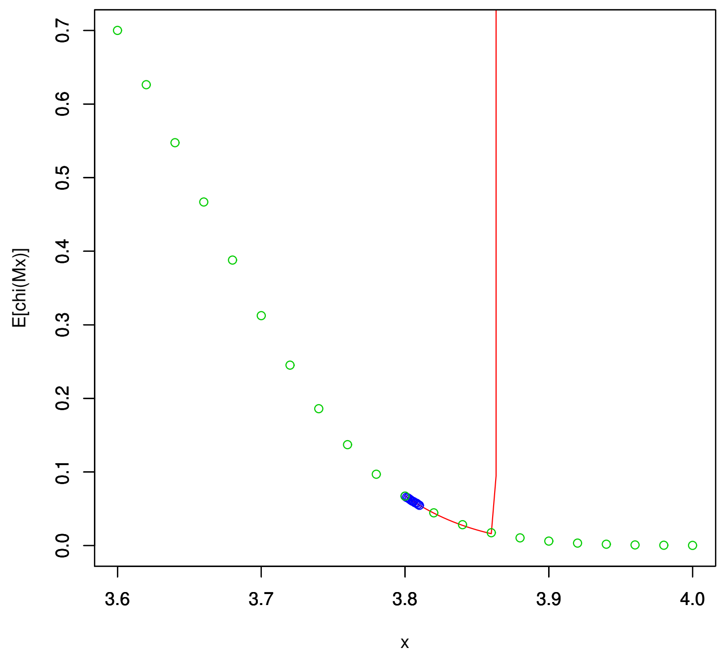
Problem 1 Derive a good system of non-linear equations satisfied by $\det \Psi(x)$. The theory of holonomic quantum field and Hirota bilinear equations might help to solve this problem. If we can find such system, try a numerical analysis of it.

Computational Try 2 The defusing method for non-linear equation needs to compute a composition of non-linear functions instead of the matrix factorial. What is the size of a problem feasible by current computer algebra systems?

Computational Try 3 Try the defusing method for $H_n^k(x, y)$ upto $y \sim 10^8$, which lies in a range to apply to practical problems.

Computational Try 4 Marumo, Oaku, Takemura gave a method to derive a linear ODE for φ^n. The function φ_3 for $r = 4$ satisfies a 2nd order linear ODE. Try to make a numerical analysis of the system for φ_3^n with the defusing method, or the method to obtain a stabile system.

Problem 2 Give a method for a high precision evalution of the hypergeometric function ${}_rF_1$ and ${}_rF_0$. Refer, e.g., to CM2013 [1].



Eeuler2019
Figure 11: [10] The extrapolation function with 20000 terms. Solid line is the extrapolation function, which diverges when $x > 3.8633$. Dots are values by simulations. fig:far

Computational Try 5 Try to make a numerical analysis of the ODE of rank 11 for $E[\chi(M_x)]$ with the defusing method, or the method to obtain a stabile system.

References

CM2013 [1] S. Chevillard, M. Mezzarobba, Multiple-precision evaluation of the Airy Ai function with reduced cancellation, arxiv:1212.4731.

KA [2] M. Kang, M. S. Alouini, Largest Eigenvalue of Complex Wishart Matrices and Performance Analysis of MIMO MRC Systems, IEEE Journal on Selected Areas in Communications 21 (2003), 418–426.

MOT2014 [3] N.Marumo, T.Oaku, A.Takemura, Properties of powers of functions satisfying second-order linear differential equations with applications to statistics, arxiv:1405.4451 hgm [4] References for HGM, `http://www.math.kobe-u.ac.jp/OpenXM/Math/hgm/ref-hgm.html`

non-central [5] F.H.Danufane, K.Ohara, N.Takayama, C.Siriteanu, Holonomic Gradient Method-Based CDF Evaluation for the Largest Eigenvalue of a Complex Noncentral Wishart Matrix, `https://arxiv.org/abs/1707.02564`.

hairer [6] , E.Hairer, S.P.Norsett, G.Wanner, Solving ordinary differential equations I, II, 1993, 1996, Springer `https://www.springer.com/gp/book/9783540566700`, `https://www.springer.com/gp/book/9783540604525`

koyama2019 [7] T. Koyama, An integral formula for the powered sum of the independent, identically and normally distributed random variables, preprint. Old version is at arxiv `https://arxiv.org/abs/1706.03989`

tgkt [8] Yoshihito Tachibana, Yoshiaki Goto, Tamio Koyama, Nobuki Takayama, Holonomic Gradient Method for Two Way Contingency Tables, arxiv:1803.04170

TKT [9] N. Takayama, S. Kuriki, A. Takemura, A-hypergeometric distributions and Newton polytopes, Advances in Applied Mathematics 99 (2018), 109 – 133.

Eeuler2019 [10] N.Takayama, L.Jiu, S.Kuriki, Y.Zhang, Computations of the Expected Euler Characteristic for the Largest Eigenvalue of a Real Wishart Matrix, arxiv:1903.10099

RT2016 [11] R.Vidunas, A.Takemura, Differential relations for the largest root distribution of complex non-central Wishart matrices, arxiv:1609.01799