EUROPEAN JOURNAL OF PURE AND APPLIED MATHEMATICS
Vol. 11, No. 3, 2018, 1058-1099
ISSN 1307-5543 – www.ejpam.com
Published by New York Business Global

Can Fractional Calculus be Generalized? Problems and Efforts

Syamal K.Sen[1,2], J. Vasundhara Devi[1,3,*], R.V.G. Ravi Kumar[3]

[1] *GVP-Prof.V.Lakshmikantham Institute for Advanced Studies, GVP College of Engineering Campus, Visakhapatnam 530048, India*
[2] *Department of Computer Science and Engineering, GVP College of Engineering, Visakhapatnam, AP, India*
[3] *Department of Mathematics, GVP College of Engineering(A), Visakhapatnam, AP, India*

Abstract. Fractional order calculus always includes integer-order too. The question that crops up is: Can it be a widely accepted generalized version of classical calculus? We attempt to highlight the current problems that come in the way to define the fractional calculus that will be universally accepted as a perfect generalized version of integer-order calculus and to point out the efforts in this direction. Also, we discuss the question: Given a non-integer fractional order differential equation as a mathematical model can we readily write the corresponding physical model and vice versa in the same way as we traditionally do for classical differential equations? We demonstrate numerically computationally the pros and cons while addressing the questions keeping in the background the generalization of the inverse of a matrix.

2010 Mathematics Subject Classifications: 26A33, 26A36, 34A08, 44A45

Key Words and Phrases: Differintegrals, fractional calculus, fractional differential equations, generalization of calculus, integer order calculus

1. Introduction

Fractional calculus is the calculus investigating the properties of derivatives and integrals of both integer order and non-integer fractional order called fractional derivatives and fractional integrals, in short differintegrals. Specifically, this discipline introduces the notion and methods of solving differential equations both ordinary and partial consisting of fractional derivatives of the unknown function (named as fractional differential equations or FDEs in short).

The history of fractional calculus began almost at the same time (late 17th century A.D.) when conventional (classical) calculus viz. integer-order calculus was established. In conventional calculus, both derivatives and integrals (anti-derivatives a term not widely

*Corresponding author.
DOI: https://doi.org/10.29020/nybg.ejpam.v11i3.3338

Email address: jvdevi@gmail.com (J. Vasundhara Devi)

 © 2018 EJPAM All rights reserved.

used) are exactly defined with corresponding exact physical significance that is taught in high schools/colleges and is uniquely and readily understood.

The classical differential equations viz. the integer order differential equations both ordinary and partial also pervades not only almost the whole of science and engineering applications but also the whole of relatively easily understood mathematical modelling for physical problems involving derivatives and integrals. The concerned scientists/modellers attempted to model some of these physical problems in terms of non-integer fractional order differential equations and to demonstrate the scope of fractional differential equations against the classical ones. Their attempts mostly were limited to mathematical treatment rather than a serious computational one.

A rigorous computational treatment both in terms of quality of the numerical solution (computational relative error) and the cost of the solution (computational complexity) for fractional differential equations were practically not done and then compared with the corresponding integer order differential equations for a specified physical problem, the claim/statement was often in favour of fractional differential equations though in some sense.

In fractional calculus, both derivatives and integrals are also exactly defined but with no corresponding exact physical significance that is readily understood exactly in the same way as those in the conventional calculus.

For instance, Newtons 1st law of motion viz. Every object (a point mass, say) either remains at rest eternally or continues to move at a constant velocity eternally, unless acted upon by an external impressed force. is understood without any ambiguity.

In our physical environment, it is not possible to have external impressed force (from any of the infinity of possible directions) to have exact zero value, we can easily imagine (the exact zero value) though. There could be, for example, air resistance against the direction of movement of the object or air assistance/flow in the direction of the movement of the object as an external impressed force over a finite (not infinite) period of time since we have not been able to create exact vacuum in our environment nor possibly does it exist in the universe.

Or/and there could be any other friction (external impressed) force such as the one created by the surface on/through which the object is moving. Consequently the moving object cannot move eternally. It eventually comes to a halt based on the strength of the friction forces. If the object is stationary/static, then, however, such a friction has no effect on the object or, equivalently, a friction simply does not exist.

Mathematicians may consider the foregoing real-world situation as an ideal example of introducing non-integer (fractional) order calculus that necessarily brings (corresponds to) one or more friction forces and explains rather more conveniently the real-world situation for the object in motion.

The question is: Can we readily know/find out the value of the fraction that corresponds precisely to the friction forces in the same way as we do in the classical calculus/models?

An ansatz (An ansatz is the establishment of the starting equation(s), the theorem(s)/the value(s) describing a physical problem or mathematical model or solution.

It can take into consideration boundary conditions. Having established an ansatz constituting essentially an assumption, the equations are solved for the general function of interest constituting a confirmation of the assumption.) for friction forces F_R is a usually followed power law

$$F_R = -\mu \ sign(v)|v|^{\alpha}, \tag{1}$$

where μ is the friction coefficient measured in $[kg/s^{(2-\alpha)}]$ in MKS unit system, The value α is an arbitrarily chosen real exponent. In special cases, $\alpha \approx 0$ found for static and kinetic friction for solids, α=1 observed for Stokes friction in liquids with high velocity, $\alpha \approx 2$ used as a general trend for high velocity.

The nonlinear ordinary differential equation using Newtons 2nd law of motion with friction forces is

$$m(\ddot{x}) = F_R = -\mu \ sign(v)|\dot{x}|^{\alpha}, \tag{2}$$

where the initial conditions are

$$x(t=0) = x_0 \tag{3}$$

$$v(t=0) = \dot{x}(t=0) = v_0 \tag{4}$$

The point mass m is measured in [kg]. If $\dot{x} = v(t)$ is positive, the foregoing ordinary differential equation (2) reduces to

$$m(\ddot{x}) = -\mu \ \dot{x}^{\alpha} \tag{5}$$

When $\alpha = 1$, the classical force and the fractional friction force coincide. But for $\alpha \neq 1$ i.e. for all other real α except 1,

$$\dot{x}^{\alpha} = (\frac{dx}{dt})^{\alpha} \neq \frac{d^{\alpha}x}{dt^{\alpha}} \tag{6}$$

Thus we expect a dynamic behaviour different from the classical one. The fractional differential equation

$$m(\ddot{x}) = -\mu \ \frac{d^{\alpha}x}{dt^{\alpha}} \qquad \dot{x}(t) > 0 \tag{7}$$

with initial conditions (3) and (4) determines the dynamical behaviour of the classical object (point mass m) to a fractional friction force. Using the Liouvilles definition of the fractional derivative and using the ansatz

$$x(t) = e^{\omega t} \tag{8}$$

we obtain the general solution consisting of 3 different constants [1]. We thus need yet 1 more condition besides the foregoing 2 initial conditions. By choosing this 1 condition a reasonable numerical value, we get a solution. Since we can have many reasonable choices, there will be infinitely many (but a finite number of solutions in a digital computer since it is a finite precision machine) numerical solutions.

By appropriate choices of the condition, we could see that the solutions of the fractional differential equation match up to the 2nd order term in t with the solutions computed for

the Newtonian equation of motion with the classical friction force term (1) [1]. Hence the difference between the fractional solutions and the classical solutions manifests both numerically and graphically when t is sufficiently greater than 1[1].

It can be seen that there is an alternative choice for 3rd initial condition. Let the initial (maximum) velocity of the object (point mass) decrease through all stages of motion due to the fractional friction force. Hence, in addition to the initial conditions (3) and (4), we introduce the 3rd initial condition [1].

$$\dot{v}(t=0) = \ddot{x}(t=0) = 0 \tag{9}$$

For the general symbolic (mathematical) solution of the FDE (including a fractional friction force) valid for t=0 up to t_0 – the final time when the object halts permanently, see [1, Page 28]. t_0 is obtained as

$$t_0 = \frac{\pi}{(\bar{\alpha})\varphi sin(\frac{\pi}{\bar{\alpha}})} \tag{10}$$

where

$$\varphi = |\frac{\mu}{m}|^{\frac{1}{\bar{\alpha}}}, \quad \bar{\alpha} = 2 - \alpha \tag{11}$$

The foregoing example of an ever-existing friction force can be tackled always by the integer-order (i.e. classical) calculus without any recourse to fractional calculus, provided we know/determine the friction force by an experiment or otherwise. Hence it is not absolutely necessary to bring fractional calculus into picture for solving any physical problem.

The solution of a fractional differential equation with appropriate initial/ boundary conditions (BCs) will be an analytic function as is the case with that of classical differential equation. Numerically, the solution will be a (numerical) table consisting of the values of the independent variables and the corresponding values of the function. The function values can be graphically represented for the function of 1 or 2 independent variables since we can visualize 0, 1, 2, and 3 dimensional objects and not beyond, a mathematical extension beyond dimension 3 is always possibe and also is required for most real-word physical problems though.

The numerical table for the function of n independent variables (n is a finite integer) can be depicted, but cannot be graphically represented on a 2-dimensional paper when n is greater than 2. It can be seen that the graph is a less accurate form of the numerical solution, it is visually very useful though. The recreation of the numerical table from the graph will have accuracy even less than (or at best equal to) that of the graph.

Once the foregoing analytic function is known as the solution of the fractional differential equation, one can readily determine the corresponding classical differential equation with appropriate initial conditions. This implies that the fractional differential equations are not absolutely essential to solve any physical problem that can be tackled by the appropriate classical differential equations. In other words, there exists no physical problem in nature, which does not have a classical differential equation (CDE) model but has a fractional differential equation (FDE) model.

Thus the knowledge of fractional differential equations may be considered redundant from the engineering and scientific applications point of view. But the concerned quality

(error) and cost (complexity) of the solution of FDE need to be explored for computational competitiveness with that of classical differential equation (CDE) for engineering applications.

Are the FDE models better than CDE models in terms of quality and cost of the numerical solutions always (for solving a concerned physical problem)? If the answer is no implying not always, then which are the practical situations when the FDE models are better? This issue will be discussed elsewhere.

The interpretation of a fractional differential equation, more simply a non-integer fractional derivative and that of a classical differential equation, more simply an integer-order derivative are different more specifically from the exact comprehension of the corresponding physical problem point in view.

The later interpretation has been readily exactly comprehended and easily understood while the former one has not been.

However, the exact value of the fraction in the fractional order cannot be readily known from the physical problem except through a numerical experiment or by some other means (say trial and error).

Special cases of the foregoing general solution are the following solutions [1]:

$$x(t) = x_0 + \frac{v_0 sin(\varphi t)}{\varphi} \qquad (for\ \alpha\ =\ 0\ i.e.,\ for\ free\ oscillation) \tag{12}$$

$$x(t) = x_0 + v_0(2 - (2 + \varphi t)e^{-\varphi t}) \qquad (for\ \alpha\ =\ 1) \tag{13}$$

This is a totally new function type. The role of the initial velocity v_0 is not more than just a scaling factor. In terms of free oscillation ($\alpha = 0$) as well as in terms of damped oscillation ($0 < \alpha \leq 1$), the time variant solutions may be better understood when we consider the 1st quarter period to describe the process of a fractional friction force [1].

The presentation of the foregoing solutions is to reveal the effect of fractional friction forces on the solution. How do we integrate and merge these facts so that the classical calculus becomes a special case of fractional calculus?

The solutions of the differential equation

$$m\ddot{x}(t) + \gamma\dot{x}(t) + kx(t) = 0 \tag{14}$$

for the damped classical harmonic oscillator will allow us to investigate and understand this aspect of fractional friction, where m, γ, k are the mass, the damping coefficient, and the spring constant of the oscillator, respectively [1].

For every value of the fraction in a fractional mathematical model usually an FDE what will precisely be the corresponding physical model?

In a digital computer (finite precision machine), concerning differential equations, there will be a very large but finite number of physical models. Corresponding to each of these physical models there will also be an integer-order mathematical model.

Will each of the integer order models be equivalent numerically to the corresponding fractional order mathematical model over the complete interval (range) of integration?

Besides, quality of a numerical solution – a must for any engineering use as an engineer needs a solution in numbers not in mathematical symbols – implying computational relative error bounds needs to be determined. This quality (the error bounds) allows us to compare relative quality of 2 or more different algorithms – both fractional and corresponding integer order [2].

Cost (computational complexity) of the numerical solution also needs to be obtained. This cost permits us to choose that algorithm out of 2 or more algorithms, which is least expensive computationally [2].

Out of the foregoing 2 parameters viz. quality and cost, usually quality is more important than cost since most of the differential equation models are not too large. This is particularly so in the current (21st century) computational scenario where the computational power has reached/crossed 10^{18} flops (floating-point operations per second).

The fractional derivative for powers was systematically studied by Riemann:

$$\frac{d^\alpha}{dx^\alpha}x^k = \frac{\Gamma(1+k)}{\Gamma(1+k-\alpha)}x^{k-\alpha} \tag{15}$$

In the foregoing equation, we restrict $x \geq 0, k \geq 0$ to ensure the uniqueness of the fractional derivative definition. The fractional derivative of the exponential function was given by Liouville and that of trigonometric functions was proposed by Fourier. Around 100 years earlier Leibniz and Euler attempted to solve this problem of fractional derivatives. According to Riemanns definition, the fractional derivative of the constant x^0 is given by

$$\frac{d^\alpha}{dx^\alpha}x^0 = \frac{1}{\Gamma(1-\alpha)}x^{-\alpha} \tag{16}$$

which is not equal to 0, the known behaviour in the integer order derivative. If $\alpha = 0$ or 1, then the Riemann definition is compatible with the conventional calculus definition. Consequently as an additional postulate Caputo imposed

$$\frac{d^\alpha}{dx^\alpha}x^0 = 0 \tag{17}$$

We have thus 4 different definitions of a fractional derivative. The common aspects of all these definitions satisfy a correspondence principle:

$$lim_{\alpha\to n}\frac{d^\alpha}{dx^\alpha}f(x) = \frac{d^n}{dx^n}f(x), \quad n = 0, 1, 2, ... \tag{18}$$

and the rules

$$\frac{d^\alpha}{dx^\alpha}cf(x) = c\frac{d^\alpha}{dx^\alpha}f(x) \tag{19}$$

$$\frac{d^\alpha}{dx^\alpha}[f(x)+g(x)] = \frac{d^\alpha}{dx^\alpha}f(x) + \frac{d^\alpha}{dx^\alpha}g(x) \tag{20}$$

This principle and the rules need to be strictly adhered to by any other definition of fractional derivatives. If these are not adhered to by a definition, then such a definition

will not merit in the first place as a possible candidate for an attempt of fractional calculus to be a generalized version of conventional calculus.

Further, we can define, in analogy to the well-known Taylor series expansion, series on these 4 different function classes and specify the corresponding fractional derivatives according to Liouville, Fourier, Riemann, and Caputo:
Liouville: If $f(x) = \sum_{k=0}^{\infty} a_k e^{kx}$, then

$$\frac{d^\alpha}{dx^\alpha} f(x) = \sum_{k=0}^{\infty} a_k k^\alpha e^{kx} \tag{21}$$

Fourier: If $f(x) = a_0 + \sum_{k=1}^{\infty} a_k sin(kx) + \sum_{k=1}^{\infty} b_k cos(kx)$, then

$$\frac{d^\alpha}{dx^\alpha} f(x) = \sum_{k=1}^{\infty} a_k k^\alpha sin(kx + \frac{\pi\alpha}{2}) + \sum_{k=1}^{\infty} b_k cos(kx + \frac{\pi\alpha}{2}) \tag{22}$$

Riemann: If $f(x) = \sum_{k=0}^{\infty} a_k x^{k\alpha}$, then

$$\frac{d^\alpha}{dx^\alpha} f(x) = \sum_{k=0}^{\infty} a_{k+1} \frac{\Gamma((k+2)\alpha)}{\Gamma((k+1)\alpha)} x^{k\alpha} \tag{23}$$

Caputo: If $f(x) = \sum_{k=0}^{\infty} a_k x^{k\alpha}$, then

$$\frac{d^\alpha}{dx^\alpha} f(x) = \sum_{k=0}^{\infty} a_{k+1} \frac{\Gamma(1+(k+1)\alpha)}{\Gamma(1+k\alpha)} x^{k\alpha} \tag{24}$$

by considering the foregoing infinite series/exponential functions.

Are these 4 definitions equivalent? The answer is No. This can be demonstrated by considering the exponential function $f(x) = e^x$ as an example. According to Caputo

$$\frac{d^\alpha}{dx^\alpha} f(x) = \sum_{n=1}^{\infty} \frac{x^{n-\alpha}}{\Gamma(1+n+\alpha)} = x^{1-\alpha}\ E(1, 2-\alpha, x) \tag{25}$$

where $E(1, 2-\alpha, x)$ is a generalized Mittag-Leffler function and evidently not an exponential function any more. This is in contrast to Louvilles definition. The non-equivalence of these definitions is disturbing and an obstacle to the attempt of making fractional derivatives a generalized version of classical derivatives.

However, not only such an obstacle but also other obstacles that we could encounter for some function or the other need to be overcome by more appropriately defining the fractional derivatives. Once a generalization compatible with the concerned physical problems is achieved, the fractional calculus will be a part of the regular university curriculum for the students.

There could be many problems from physics which might be beneficially numerically solved using an appropriate mathematical model based on fractional differential equations. But this may not have direct connection to the generalized version of the calculus.

If a physical frictional model can be readily converted to a non-integer fractional differential equation model and vice versa exactly in the same way as we could do for integer-order differential equation model, then it would be a break-through in the process of achieving the desired generalization.

The unique generalized matrix inverse (viz. the Moore-Penrose matrix inverse also known as the minimum-norm least-squares inverse of the matrix or as the pseudo-inverse of the matrix) out of the possible infinity of generalized inverses of a matrix has been beautifully integrated and merged with the true matrix inverse during 1950-1970s. This integration is included in all university science and engineering curricula.

The pseudo-inverse [3] (the term is coined by Gene H. Golub of Stanford University and is used widely) of any matrix square, non-square rectangular, and singular –is unique and always exists unlike the true inverse that exists only for non-singular (necessarily square) matrices. The pseudo-inverse is meant for solving non-square rectangular as well as square linear systems and is extensively used in science and engineering applications.

The pseudo-inverse is non-redundant in the sense that the true inverse cannot take the place of the pseudo-inverse for solving linear equations including inconsistent equations (for inconsistent systems, the minimum-norm least-squares solution is obtained and is used and this solution is of paramount importance in all our concerned physical problems numerically). If the matrix is non-singular then the pseudo-inverse is the true inverse of the matrix.

In a linear consistent system if the matrix is near-singular then computational error creeps in the solution irrespective of whether the inverse is true (when it exists and computable within the available finite precision of the computer) or pseudo. The more pronounced the singularity is, the more will be the computational error. Depending on the context, this error may or may not be acceptable. Strictly speaking while a singular linear system like a non-singular one poses no extraordinary error problem for a true solution or the minimum norm least-squares solution, a near-singular system does. So a near-singular linear system may be termed ill-conditioned with respect to its solution while a strictly singular as well as a non-singular (sufficiently away from a near-singular (depending on the precision/context)) linear system may not be.

If the pseudo-inverse of the rectangular matrix A is B, then the pseudo inverse of the matrix B is the matrix A. Could the fractional calculus be made to have a similar status in future by appropriate definitions and procedures?

In other words, once the integer-order derivative/fractional derivative of a function is known, can we get back the original function using its anti-derivative (i.e. the integration/fractional integration of the derivative) by imposing the concerned condition(s)?

While computing the ever-existent pseudo-inverse or the non-ever existent true inverse (true matrix inverse does not exist for non-square rectangular matrices as well as square singular matrices) of a matrix, no fractional power of the matrix is involved in computing the pseudo-inverse. This is quite unlike the fractional calculus operations.

However, for a square singular or a non-singular matrix A, we can find its fractional power $B = A^{\alpha}$. Hence $B^{(1/\alpha)} = A$ for α positive real. If α is complex with the imaginary part $\neq 0$, then the preceding rule may not be valid for every matrix whose order is greater

then 1. A fractional positive real power α of the matrix A could have more than one matrix. Everyone of these matrices with power $\frac{1}{\alpha}$ will result in the original matrix A.

If the matrix A is rectangular real/complex, then appending 0 rows or 0 columns it can be made square. Replace the original rectangular matrix A by this new square matrix. That is, the most recent matrix A is square singular. The foregoing fractional power rule in the preceding paragraph is perfectly valid for this matrix A for α positive real. Incidentally, the identical rule is always valid for a real/complex matrix of order 1. A real/complex number may be viewed as a real/complex matrix of order 1 and vice versa.

The following numerical examples produced using Matlab demonstrate the foregoing facts.

$>> A = [1+1i\ \ 2-2i\ \ 3+3i; 4+4i\ \ 5+5i\ \ 6-6i; 7+7i\ \ 8-8i\ \ 9+9i], B = A\wedge(.4-.7i),$
$C = B\wedge(1/(.4-.7i))$
(The matrices A and C will be the same)

A =
1.0000 + 1.0000i 2.0000 - 2.0000i 3.0000 + 3.0000i
4.0000 + 4.0000i 5.0000 + 5.0000i 6.0000 - 6.0000i
7.0000 + 7.0000i 8.0000 - 8.0000i 9.0000 + 9.0000i

B =
0.0304 - 0.6891i -0.3652 + 0.8921i -0.1599 - 1.5749i
0.9850 - 0.7253i 2.0753 - 4.7575i -2.1730 + 2.2331i
-0.3766 - 1.8286i -1.9743 + 2.9792i i -0.1007 - 5.6070i

C =
1.0000 + 1.0000i 2.0000 - 2.0000i 3.0000 + 3.0000i
4.0000 + 4.0000i 5.0000 + 5.0000i 6.0000 - 6.0000i
7.0000 + 7.0000i 8.0000 - 8.0000i 9.0000 + 9.0000i

>> clear all; close all

$>> A = [1\ 2\ 3\ 4; 5\ 6\ 7\ 8], C = [A; zeros(2,4)], B = C \wedge (.4-.7i), D = B \wedge (1/(.4-.7i))$
(The matrix B and hence the matrix D do not exist.)

A =
1 2 3 4
5 6 7 8

C =
1 2 3 4
5 6 7 8
0 0 0 0
0 0 0 0

B =
NaN + NaNi NaN + NaNi NaN + NaNi NaN + NaNi

NaN + NaNi	NaN + NaNi	NaN + NaNi	NaN + NaNi
NaN + NaNi	NaN + NaNi	NaN + NaNi	NaN + NaNi
NaN + NaNi	NaN + NaNi	NaN + NaNi	NaN + NaNi

(NaN means Not a Number and implies usually division by 0)
??? Error using ==> mpower
Input to EIG must not contain NaN or Inf.

>> $A = [1\ 2\ 3\ 4; 5\ 6\ 7\ 8], C = [A; zeros(2,4)], B = C \wedge .4, D = B \wedge (1/.4)$
The matrices C and D are identical.)

A =
1	2	3	4
5	6	7	8

C =
1	2	3	4
5	6	7	8
0	0	0	0
0	0	0	0

B =
0.6202 + 0.5982i	0.4968 - 0.1832i	0.3733 - 0.9645i	0.2499 - 1.7458i
1.2420 - 0.4579i	1.8622 + 0.1402i	2.4825 + 0.7384i	3.1027 + 1.3366i
0	0	0	0
0	0	0	0

D =
1.0000 + 0.0000i	2.0000 - 0.0000i	3.0000 - 0.0000i	4.0000 - 0.0000i
5.0000 + 0.0000i	6.0000 - 0.0000i	7.0000 - 0.0000i	8.0000 - 0.0000i
0	0	0	0
0	0	0	0

>> $a = 5 + 7i, b = a \wedge (.4 - .7i), c = b \wedge (1/(.4 - .7i))$

(complex power of a complex number (1x1 matrix)

a = 5.0000 + 7.0000i
b =1.9787 - 4.1534i
c = 5.0000 + 7.0000i (c is the same as a as these should be.)

>> $A = [1+1i\ \ 2-2i\ \ 3+3i; 4+4i\ \ 5+5i\ \ 6-6i; 7+7i\ \ 8-8i\ \ 9+9i], B = A \wedge (.4 - .7i),$
$C = B \wedge (1/(.4 - .7i))$
(The matrix C is the same as the matrix A as is expected here.)

A =
1.0000 + 1.0000i	2.0000 - 2.0000i	3.0000 + 3.0000i

```
     4.0000 + 4.0000i   5.0000 + 5.0000i   6.0000 - 6.0000i
     7.0000 + 7.0000i   8.0000 - 8.0000i   9.0000 + 9.0000i

B =  0.0304 - 0.6891i   -0.3652 + 0.8921i   -0.1599 - 1.5749i
     0.9850 - 0.7253i   2.0753 - 4.7575i    -2.1730 + 2.2331i
    -0.3766 - 1.8286i   -1.9743 + 2.9792i i -0.1007 - 5.6070i

C =  1.0000 + 1.0000i   2.0000 - 2.0000i   3.0000 + 3.0000i
     4.0000 + 4.0000i   5.0000 + 5.0000i   6.0000 - 6.0000i
     7.0000 + 7.0000i   8.0000 - 8.0000i   9.0000 + 9.0000i
```

The Matlab provides only 1 of these possible matrices (as answers) and this serves our purpose for solving a physical problem well unless the context demands something different from the usual stuff.

We provide in section 2 the problems and efforts toward generalizing the calculus such that the integer order (classical) calculus is integrated/merged with/into the fractional one. One of the popular fractional derivative viz. Grunwald-Letnikov derivative demonstrates numerically and also graphically how the derivatives behave and how well they integrate/merge with the integer order ones. This is essentially an effort by the mathematicians/physicists toward making fractional calculus a generalized version that includes integer-order calculus. Also included in this section FDE and its integration with the classical one and the interpretation.

In section 3 we raise a few of questions on fractional ODEs (FODEs) regarding how we convert these to a system of α-th order FODEs or/and 1st order ODEs, how many initial conditions we would need to solve these ODEs, and what procedure(s) we could design and develop to numerically solve the system. Also talked about in this section are the partial as well as complete generalizations of calculus dealing with both fractional and integer orders. Conclusions are included in section 4.

2. Problems and Efforts

2.1. Leibnitz Product Rule

The Leibniz Product Rule (LPR) for integer order derivatives (IOD) is

$$\frac{d}{dx}(f_1(x)f_2(x)) = \frac{df_1}{dx}f_2 + f_1\frac{df_2}{dx} \tag{26}$$

But the LPR is not valid for fractional order derivatives:

$$\frac{d^\alpha}{dx}(f_1(x)f_2(x)) \neq \frac{d^\alpha f_1}{dx}f_2 + f_1\frac{d^\alpha f_2}{dx}, \quad if(\alpha \neq 1) \tag{27}$$

The proof follows from the counter-example where $f_1(x) = e^{k_1 x},\ f_2(x) = e^{k_2 x}$ The foregoing product rule will be valid for the given problem if $k_1^\alpha + k_2^\alpha = (k_1 + k_2)^\alpha$ i.e., if $k_1 = k_2$

and $\alpha =$ any nonnegative integer, or $\alpha = 1$. This is undesirable for a generalization of the rule. Is this problem not surmountable by designing an appropriate procedure? The answer is No. That is, the problem is surmountable.

The LPR for generalized derivative (valid for IOD as well as fractional order derivative (FOD) may be given:

Define $\frac{d^\alpha}{dx^\alpha} = D^\alpha$ When $\alpha = n \in N$ is an arbitrary integer (N is the set of natural numbers), then

$$D^n(f_1 f_2) = \sum_{j=0}^{n} {}^nC_j D^{n-j} f_1 D^j f_2 \tag{28}$$

For arbitrary α,

$$D^\alpha(f_1 f_2) = \sum_{j=0}^{\infty} {}^\alpha C_j D^{\alpha-j} f_1 D^j f_2 \tag{29}$$

where f_1, f_2 are functions of the independent variable x. The foregoing rules (28) and (29), though more expensive, in general, from quality (implying more computational relative error) and cost (implying more computational complexity) points of view, are still generalized LPR.

However, a direct transfer of any standard rule of classical calculus to fractional calculus could be hazardous. In every case the rule used in fractional calculus needs to be checked for its validity.

A useful strategy for a satisfactory generalization scheme is a 2-step procedure[1]:
Step 1. Derive a rule which is valid for all $n \in N$.
Step 2. Replace n by $\alpha \in C$ and check the validity,

where C is the set of complex numbers.

Computationally, the LPR is not an essentially required rule. Computation is a must in all engineering and science applications. Nobody can escape computation if he has some thing to do with a real world physical problem. Consequently, it is very much more important than the pure mathematical treatment of fractional calculus/differential equations. Let x be defined in a finite interval $[a, b]$, where a,b are numerical values. Suppose that $f_1(x), f_2(x), f_1(x)f_2(x)$ are functions defined by/computed as a numerical table having 4 columns viz.

x	$f_1(x)$	$f_2(x)$	$f_1(x)f_2(x)$
a	$f_1(a)$	$f_2(a)$	$f_1(a)f_2(a)$
.	.	.	

b $f_1(b)$ $f_2(b)$ $f_1(b)f_2(b)$

All the entries from the 2^{nd} row to the last row are all numerical elements. One may use the 1st and the 4^{th} columns of the foregoing table to compute a fractional numerical derivative based on 1 of the definitions such as the Grunwald-Letnikov definition.

In case of 3 or more functions, the product rule can be obviated similarly. Hence generalization problem for LPR is numerically computationally non-existent.

2.2. Fractional derivatives of general functions: Values and Graphs

We present some of the fractional derivatives for a general function. The presentation will help one to get a feel about the scope of each one of them. Consequently one can decide which of the definitions would be more desirable/suitable as an attempt to generalize the calculus.

It is also possible that a new definition for the fractional derivative can be proposed, which would allow a better generalization (with possibly less error and less computation).

Grunwald-Letnikov Derivative (Reverse)

$$D^{\alpha}f(x) = lim_{h\to 0}\frac{1}{h^{\alpha}}\sum_{(0\le m<\infty)}(-1)^m \ \ ({}^{\alpha}C_m)f(x+(\alpha-m)h)$$

$$= lim_{h\to 0}\frac{1}{h^{\alpha}}\sum_{(0\le m<\infty)}(-1)^m \ \ \frac{\Gamma(1+\alpha)}{\Gamma(1+m)\Gamma(1+\alpha-m)}f(x+(\alpha-m)h) \qquad (30)$$

Grunwald-Letnikov Derivative (Direct) Set $h = -h$

$$D^{\alpha}f(x) = lim_{h\to 0}\frac{1}{h^{\alpha}}\sum_{(0\le m<\infty)}(-1)^m \ \ ({}^{\alpha}C_m)f(x-(\alpha-m)h)$$

$$= lim_{h\to 0}\frac{1}{h^{\alpha}}\sum_{(0\le m<\infty)}(-1)^m \ \ \frac{\Gamma(1+\alpha)}{\Gamma(1+m)\Gamma(1+\alpha-m)} \ f(x+(\alpha-m)h) \qquad (31)$$

We have taken $h = .01,\ \alpha = .99,\ m = 0\ to 140$. Error in computation of the gamma function is significant when α is a non-integer fraction. If α is a positive integer, then we can use the factorial function (and also the gamma function) with many terms both in the numerator and the denominator cancelling. The accuracy will be significantly more. Observe that in Matlab, we can compute (up to argument 17) exactly $factorial(17) = gamma(18) = 355687428096000$. Beyond the argument 17 Matlab provides only approximate (erroneous) integer values.

The Matlab routine fgl.deriv.m [8, 17] is used to compute, in 3-D, the fractional derivatives (including integer ones) for different fractional orders of an equi-spaced sampled function using Grunwald-Letnikov formulation. The idea is to demonstrate how well the formulation integrates and merges the fractional orders with the integer orders. The 12-line Matlab program $fderiv\_sin1\_hp1$, where the sampling period h=0.1 is as follows.

```
% clear all; close all;
%h=0.1; t=0:h:4*pi;
%y=sin(t),
%order=0:0.1:1;
%for i=1:length(order)
%yd(i,:)=fgl _deriv(order(i),y,h);
%end; for i=1:length(order),
% 'Fractional order alpha=',disp(order(i)),
% 'Sampled derivatives (omitting 1st element 0) are ', disp(yd(i,:)), end;
%[X, Y]=meshgrid(t,order);
%mesh(X,Y,yd)
%xlabel( ' t '); ylabel( ' α ' ); zlabel( ' y ' )

% Usage We call the following 12-line Matlab program "fderiv _ sin1 _ hp1"
% without the 12 "%" symbols and use simply >> fderiv _ sin1_ hp1 (in the command
% window). Observe that the Matlab program uses the Matlab routine fgl _ deriv which
%needs to be available in the Matlab machine.
```

We have used the fractional order $\alpha = 0, .1, .2, .3, ..., 1$ and computed the fractional (including integer order derivatives for $x = 0, .1, .2, ..., 4\pi$. One can readily check how accurate the integer order derivatives are and how the fractional orders merge and integrate with the integer order derivatives. Also see [8].

The numerical results (only partially produced to conserve space) and the graph (Fig. 1a) depicting the results are as follows. The reader may view the complete numerical result along with the graph when he executes the foregoing Matlab program fderiv_sin1_hp1.

Fractional order alpha= 0 Sampled derivatives (omitting the 1st element 0) are

0(*omit*) 0.0998 0.1987 0.2955 0.3894 0.4794 0.5646
0.6442 0.7174 0.7833 0.8415 0.8912 0.9320 0.9636
0.9854 0.9975 0.9996 0.9917 0.9738 0.9463 0.9093
0.8632 0.8085 0.7457 0.6755 0.5985 0.5155 0.4274 0.3350
0.2392 0.1411 0.0416 -0.0584 -0.1577 -0.2555 -0.3508 -0.4425
-0.5298 -0.6119 -0.6878 -0.7568 -0.8183 -0.8716 -0.9162 -
0.9516 -0.9775 -0.9937 -0.9999 -0.9962 -0.9825 -0.9589 -
0.9258 -0.8835 -0.8323 -0.7728 -0.7055 -0.6313 -0.5507 -
0.4646 -0.3739 -0.2794 -0.1822 -0.0831 0.0168 0.1165 0.2151
0.3115 0.4048 0.4941 0.5784 0.6570 0.7290 0.7937 0.8504
0.8987 0.9380 0.9679 0.9882 0.9985 0.9989 0.9894 0.9699
0.9407 0.9022 0.8546 0.7985 0.7344 0.6630 0.5849 0.5010
0.4121 4 0.3191 0.2229 0.1245 0.0248 −0.0752 −0.1743 −0.2718
−0.3665 − 0.4575 − 0.5440 − 0.6251 − 0.6999 − 0.7677 − 0.8278
−0.8797 − 0.9228 − 0.9566 − 0.9809 − 0.9954 − 1.0000 − 0.9946

−0.9792 − 0.9540 − 0.9193 − 0.8755 − 0.8228 − 0.7620 − 0.6935
−0.6181 − 0.5366 − 0.4496 − 0.3582 − 0.2632 − 0.1656 − 0.0663

Fractional order alpha= 0.1000 Sampled derivatives are

0 (omit) 0.1257 0.2375 0.3414 0.4382 0.5281 0.6106 0.6854 0.7520 0.8098 0.8585 0.8977 0.9271 0.9464 0.9556 0.9546 0.9434 0.9223 0.8914 0.8511 0.8019 0.7442 0.6786 0.6059 0.5268 0.4420 0.3525 0.2591 0.1628 0.0646 -0.0345 -0.1336 -0.2316 -0.3276 -0.4205 -0.5095 -0.5936 -0.6720 -0.7440 -0.8087 -0.8655 -0.9139 -0.9534 -0.9835 -1.0040 -1.0146 -1.0153 -1.0060 -0.9868 -0.9580 -0.9197 -0.8724 -0.8165 -0.7527 -0.6814 -0.6035 -0.5197 -0.4309 -0.3379 -0.2416 -0.1431 -0.0433 0.0569 0.1563 0.2541 0.3492 0.4406 0.5276 0.6092 0.6846 0.7530 0.8138 0.8664 0.9102 0.9447 0.9698 0.9851 0.9904 0.9857 0.9711 0.9467 0.9127 0.8695 0.8175 0.7573 0.6894 0.6145 0.5334 0.4469 0.3559 0.2612 0.1638 0.0647 -0.0352 -0.1347 -0.2330 -0.3291 -0.4219 -0.5106 -0.5943 -0.6721 -0.7433 -0.8071 -0.8629 -0.9102 -0.9485 -0.9773 -0.9965 -1.0057 -1.0050 -0.9943 -0.9738 -0.9435 -0.9039 -0.8554 -0.7984 -0.7334 -0.6612 -0.5825 -0.4979 -0.4085 -0.3151 -0.2185 -0.1199 -0.0201 0.0799

Fractional order alpha= 0.2000 Sampled derivatives are

0 (omit) 0.1582 0.2832 0.3927 0.4907 0.5785 0.6565 0.7246 0.7829 0.8310 0.8689 0.8963 0.9133 0.9197 0.9157 0.9013 0.8770 0.8428 0.7993 0.7470 0.6864 0.6183 0.5432 0.4620 0.3756 0.2848 0.1906 0.0940 -0.0041 -0.1027 -0.2007 -0.2972 -0.3912 -0.4816 -0.5677 -0.6485 -0.7232 -0.7910 -0.8513 -0.9035 -0.9469 -0.9812 -1.0060 -1.0211 -1.0263 -1.0215 -1.0068 -0.9823 -0.9482 -0.9050 -0.8529 -0.7926 -0.7246 -0.6496 -0.5684 -0.4817 -0.3904 -0.2954 -0.1977 -0.0982 0.0021 0.1021 0.2009 0.2976 0.3910 0.4804 0.5648 0.6434 0.7153 0.7800 0.8366 0.8848 0.9239 0.9537 0.9738 0.9840 0.9842 0.9744 0.9547 0.9254 0.8866 0.8389 0.7826 0.7184 0.6469 0.5687 0.4848 0.3959 0.3029 0.2067 0.1084 0.0088 -0.0909 -0.1899 -0.2871 -0.3815 -0.4723 -0.5584 -0.6391 -0.7135 -0.7809 -0.8405 -0.8919 -0.9345 -0.9679 -0.9917 -1.0056 -1.0097 -1.0037 -0.9878 -0.9622 -0.9270 -0.8826 -0.8296 -0.7683 -0.6995 -0.6237 -0.5418 -0.4546 -0.3630 -0.2678 -0.1700 -0.0706 0.0295 0.1291 0.2274

Fractional order alpha= 0.9000 Sampled derivatives are

0 (omit) 0.7930 0.8644 0.8914 0.8965 0.8857 0.8619 0.8265 0.7807 0.7255 0.6618 0.5904 0.5123 0.4284 0.3395 0.2468 0.1511 0.0534 -0.0451 -0.1435 -0.2407 -0.3358 -0.4278 -0.5157 -0.5987 -0.6759 -0.7465 -0.8098 -0.8652 -0.9120 -0.9499 -0.9784 -0.9973 -1.0062 -1.0053 -0.9943 -0.9736 -0.9432 -0.9034 -0.8547 -0.7975 -0.7325 -0.6601 -0.5813 -0.4967 -0.4072 -0.3136 -0.2170 -0.1183 -0.0185 0.0815 0.1807 0.2779 0.3724 0.4631 0.5491 0.6297 0.7038 0.7710 0.8303 0.8814 0.9236 0.9565 0.9799 0.9935 0.9970 0.9907 0.9743 0.9482 0.9127 0.8679

0.8145 0.7529 0.6838 0.6078 0.5257 0.4384 0.3466 0.2514 0.1536 0.0543 -0.0456 -0.1450 -0.2430 -0.3386 -0.4309 -0.5188 -0.6016 -0.6784 -0.7484 -0.8110 -0.8654 -0.9113 -0.9480 -0.9753 -0.9928 -1.0005 -0.9982 -0.9859 -0.9637 -0.9320 -0.8909 -0.8410 -0.7827 -0.7165 -0.6432 -0.5635 -0.4782 -0.3881 -0.2941 -0.1973 -0.0984 0.0014 0.1012 0.2000 0.2968 0.3906 0.4804 0.5655 0.6449 0.7179 0.7837 0.8416 0.8912 0.9318 0.9631

Fractional order alpha= 1 Sampled derivatives are

0 (omit) 0.9983 0.9884 0.9685 0.9390 0.9001 0.8522 0.7958 0.7314 0.6597 0.5814 0.4974 0.4083 0.3152 0.2189 0.1205 0.0208 -0.0791 -0.1782 -0.2755 -0.3700 -0.4609 -0.5471 -0.6279 -0.7024 -0.7699 -0.8297 -0.8812 -0.9239 -0.9574 -0.9813 -0.9954 -0.9995 -0.9937 -0.9780 -0.9524 -0.9174 -0.8732 -0.8202 -0.7591 -0.6904 -0.6147 -0.5330 -0.4459 -0.3544 -0.2593 -0.1616 -0.0623 0.0376 0.1371 0.2353 0.3311 0.4236 0.5119 0.5950 0.6722 0.7427 0.8058 0.8608 0.9073 0.9446 0.9725 0.9907 0.9990 0.9974 0.9857 0.9642 0.9331 0.8926 0.8433 0.7855 0.7198 0.6470 0.5677 0.4827 0.3929 0.2992 0.2025 0.1038 0.0040 -0.0958 -0.1947 -0.2916 -0.3856 -0.4757 -0.5611 -0.6409 -0.7143 -0.7805 -0.8390 -0.8890 -0.9302 -0.9621 -0.9844 -0.9968 -0.9993 -0.9918 -0.9743 -0.9472 -0.9106 -0.8649 -0.8105 -0.7480 -0.6781 -0.6014 -0.5187 -0.4308 -0.3386 -0.2430 -0.1450 -0.0455 0.0544 0.1537 0.2516 0.3469 0.4388 0.5262 0.6085 0.6846 0.7539 0.8156 0.8693 0.9142 0.9500 0.9763 0.9928

The foregoing results are correct up to 2 decimal places (same here as 2 significant digits since the sine and cosine functions are having order of magnitude 1).

For better accuracy, choose the sampling period h=.01 (10 times higher frequency). That is, replace in the foregoing 12-line program h=0.1 by h=0.01.

Consequently the numerical results (not depicted to conserve space) are correct up to 3 significant digits (enough for most engineering implementations). However, for the graph for h=0.01 see Fig. 1b.

It may be seen how the sampled fractional derivatives demonstrate the curvature (deviated from the uniform behaviour as it should be).

As another example we take cosine function and demonstrate how it behaves for $h = 0.1, \alpha = 0(.1)14$ and the cosine function sampled at points $\frac{-pi}{2}(h)6 * pi$. The Matlab program fderiv_cos8.m is as follows

```
clear all; close all;
h=0.1; t=-pi/2:h:6*pi;
y=cos(t),
order=0:0.1:1;
for i=1:length(order)
yd(i,:)=fgl_deriv(order(i),y,h);
end; for i=1:length(order),
```

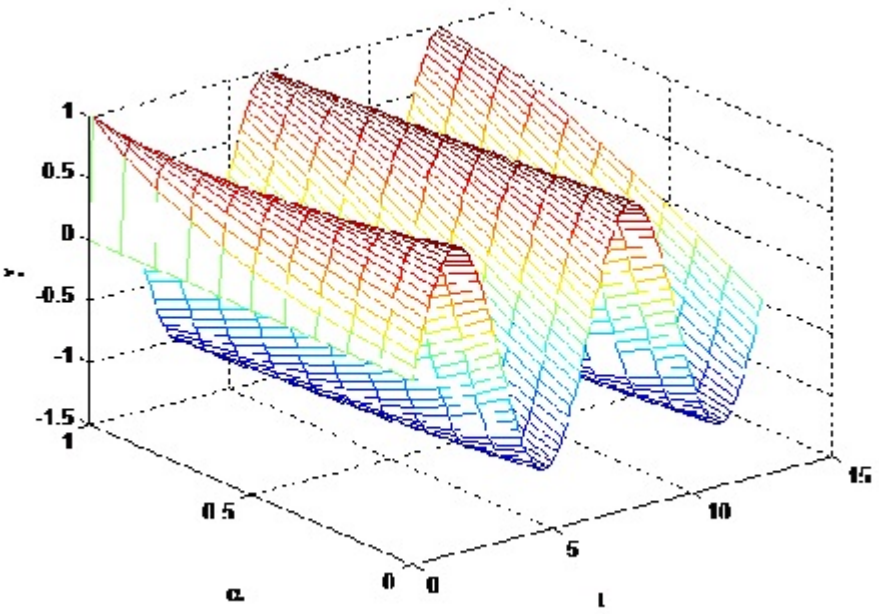

Fig. 1a Fractional derivative of $sin(x) for x = 0(h)4\pi$ for $\alpha = 0(0.1)1,\ h = 0.1$(The graph is slightly different in appearance from the original graph produced by Matlab due to copy (under Matlab Edit) and paste)

'Fractional order alpha=',disp(order(i)),
'Sampled derivatives (omitting 1st element 0) are', disp(yd(i,:)), end; [X, Y]=meshgrid(t,order); mesh(X,Y,yd)

xlabel('t'); ylabel($'\alpha'$); zlabel('y')

Observe that there is a great advantage in considering an equi-spaced sampled function in numerical computation. The program needs no change except in the 3rd line viz. $y = cos(t)$.

If we have to deal with different mathematical non-sampled functions inside the Matlab program, then there will be more changes in the program. However, ill-conditioned functions (such as violently fluctuating continuous functions) would always be needing very careful handling/programming so that we get reasonably good usable numerical results.

The results (produced partially here to save space) and the corresponding graph (Fig. 2) are as follows.

Fractional order alpha=0 Sampled derivatives are
0.0000 0.0998 0.1987 0.2955 0.3894 0.4794 0.5646 0.6442 0.7174 0.7833 0.8415 0.8912

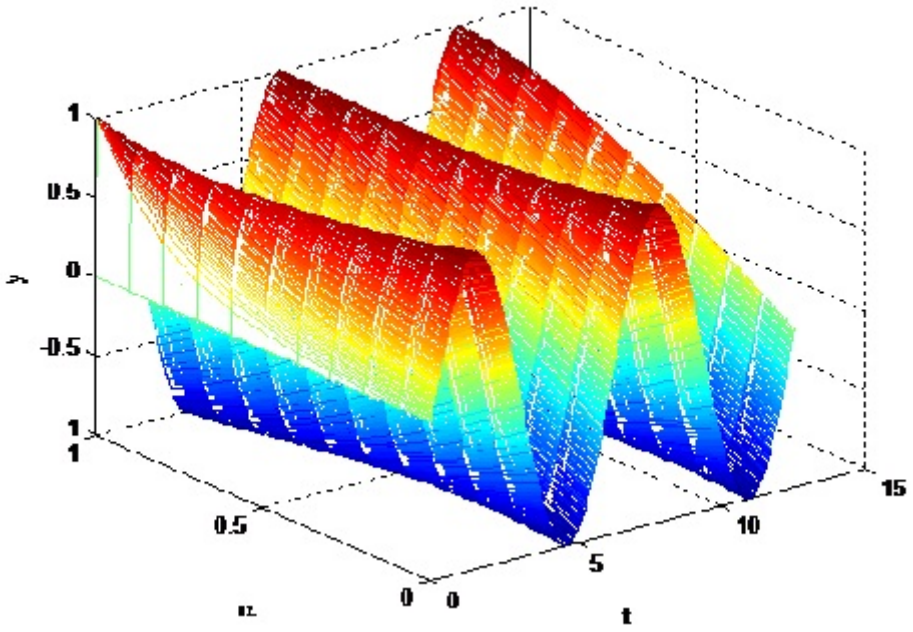

Fig. 1b Fractional derivative of $sin(x) for x = 0(h)4\pi$ for $\alpha = 0(0.1)1, h = 0.01$ (The graph is slightly different in appearance from the original graph produced by Matlab due to copy (under Matlab Edit) and paste)

0.9320 0.9636 0.9854 0.9975 0.9996 0.9917 0.9738 0.9463 0.9093 0.8632 0.8085 0.7457 0.6755 0.5985 0.5155 0.4274 0.3350 0.2392 0.1411 0.0416 -0.0584 -0.1577 -0.2555 -0.3508 -0.4425 -0.5298 -0.6119 -0.6878 -0.7568 -0.8183 -0.8716 -0.9162 -0.9516 -0.9775 -0.9937 -0.9999 -0.9962 -0.9825 -0.9589 -0.9258 -0.8835 -0.8323 -0.7728 -0.7055 -0.6313 -0.5507 -0.4646 -0.3739 -0.2794 -0.1822 -0.0831 0.0168 0.1165 0.2151 0.3115 0.4048 0.4941 0.5784 0.6570 0.7290 0.7937 0.8504 0.8987 0.9380 0.9679 0.9882 0.9985 0.9989 0.9894 0.9699 0.9407 0.9022 0.8546 0.7985 0.7344 0.6630 0.5849 0.5010 0.4121 0.3191 0.2229 0.1245 0.0248 -0.0752 -0.1743 -0.2718 -0.3665 -0.4575 -0.5440 -0.6251 -0.6999 -0.7677 -0.8278 -0.8797 -0.9228 -0.9566 -0.9809 -0.9954 -1.0000 -0.9946 -0.9792 -0.9540 -0.9193 -0.8755 -0.8228 -0.7620 -0.6935 -0.6181 -0.5366 -0.4496 -0.3582 -0.2632 -0.1656 -0.0663 0.0336 0.1332 0.2315 0.3275 0.4202 0.5087 0.5921 0.6696 0.7404 0.8038 0.8592 0.9060 0.9437 0.9720 0.9906 0.9993 0.9980 0.9868 0.9657 0.9349 0.8948 0.8457 0.7883 0.7229 0.6503 0.5712 0.4864 0.3967 0.3031 0.2065 0.1078 0.0080 -0.0919 -0.1909 -0.2879 -0.3821 -0.4724 -0.5581 -0.6381 -0.7118 -0.7784 -0.8371 -0.8876 -0.9291 -0.9614 -0.9841 -0.9969 -0.9998 -0.9927 -0.9756 -0.9488 -0.9126 -0.8672 -0.8132 -0.7510 -0.6813 -0.6048 -0.5223 -0.4346 -0.3425 -0.2470 -0.1490 -0.0495 0.0504 0.1499 0.2478 0.3433 0.4354 0.5231 0.6055 0.6820 0.7516 0.8137 0.8676 0.9129 0.9491 0.9758 0.9928 0.9998

Fractional order alpha= 0.1000 Sampled derivatives are

0.0000 (omit) 0.1257 0.2375 0.3414 0.4382 0.5281 0.6106 0.6854 0.7520 0.8098 0.8585 0.8977 0.9271 0.9464 0.9556 0.9546 0.9434 0.9223 0.8914 0.8511 0.8019 0.7442 0.6786 0.6059 0.5268 0.4420 0.3525 0.2591 0.1628 0.0646 -0.0345 -0.1336 -0.2316 -0.3276 -0.4205 -0.5095 -0.5936 -0.6720 -0.7440 -0.8087 -0.8655 -0.9139 -0.9534 -0.9835 -1.0040 -1.0146 -1.0153 -1.0060 -0.9868 -0.9580 -0.9197 -0.8724 -0.8165 -0.7527 -0.6814 -0.6035 -0.5197 -0.4309 -0.3379 -0.2416 -0.1431 -0.0433 0.0569 0.1563 0.2541 0.3492 0.4406 0.5276 0.6092 0.6846 0.7530 0.8138 0.8664 0.9102 0.9447 0.9698 0.9851 0.9904 0.9857 0.9711 0.9467 0.9127 0.8695 0.8175 0.7573 0.6894 0.6145 0.5334 0.4469 0.3559 0.2612 0.1638 0.0647 -0.0352 -0.1347 -0.2330 -0.3291 -0.4219 -0.5106 -0.5943 -0.6721 -0.7433 -0.8071 -0.8629 -0.9102 -0.9485 -0.9773 -0.9965 -1.0057 -1.0050 -0.9943 -0.9738 -0.9435 -0.9039 -0.8554 -0.7984 -0.7334 -0.6612 -0.5825 -0.4979 -0.4085 -0.3151 -0.2185 -0.1199 -0.0201 0.0799 0.1790 0.2762 0.3706 0.4613 0.5473 0.6277 0.7019 0.7690 0.8283 0.8793 0.9215 0.9544 0.9777 0.9912 0.9948 0.9883 0.9720 0.9458 0.9102 0.8654 0.8119 0.7503 0.6811 0.6051 0.5230 0.4356 0.3438 0.2485 0.1507 0.0513 -0.0486 -0.1481 -0.2461 -0.3418 -0.4340 -0.5220 -0.6048 -0.6816 -0.7516 -0.8142 -0.8687 -0.9145 -0.9512 -0.9785 -0.9961 -1.0037 -1.0013 -0.9889 -0.9666 -0.9347 -0.8934 -0.8432 -0.7845 -0.7181 -0.6444 -0.5644 -0.4786 -0.3881 -0.2937 -0.1964 -0.0971 0.0031 0.1033 0.2025 0.2997 0.3938 0.4841 0.5695 0.6492 0.7224 0.7884 0.8465 0.8962 0.9369 0.9683 0.9899 1.0017 1.0035 0.9953

Fractional order alpha= 0.9000 Sampled derivatives are

0.0000 (omit) 0.7930 0.8644 0.8914 0.8965 0.8857 0.8619 0.8265 0.7807 0.7255 0.6618 0.5904 0.5123 0.4284 0.3395 0.2468 0.1511 0.0534 -0.0451 -0.1435 -0.2407 -0.3358 -0.4278 -0.5157 -0.5987 -0.6759 -0.7465 -0.8098 -0.8652 -0.9120 -0.9499 -0.9784 -0.9973 -1.0062 -1.0053 -0.9943 -0.9736 -0.9432 -0.9034 -0.8547 -0.7975 -0.7325 -0.6601 -0.5813 -0.4967 -0.4072 -0.3136 -0.2170 -0.1183 -0.0185 0.0815 0.1807 0.2779 0.3724 0.4631 0.5491 0.6297 0.7038 0.7710 0.8303 0.8814 0.9236 0.9565 0.9799 0.9935 0.9970 0.9907 0.9743 0.9482 0.9127 0.8679 0.8145 0.7529 0.6838 0.6078 0.5257 0.4384 0.3466 0.2514 0.1536 0.0543 -0.0456 -0.1450 -0.2430 -0.3386 -0.4309 -0.5188 -0.6016 -0.6784 -0.7484 -0.8110 -0.8654 -0.9113 -0.9480 -0.9753 -0.9928 -1.0005 -0.9982 -0.9859 -0.9637 -0.9320 -0.8909 -0.8410 -0.7827 -0.7165 -0.6432 -0.5635 -0.4782 -0.3881 -0.2941 -0.1973 -0.0984 0.0014 0.1012 0.2000 0.2968 0.3906 0.4804 0.5655 0.6449 0.7179 0.7837 0.8416 0.8912 0.9318 0.9631 0.9847 0.9965 0.9984 0.9903 0.9722 0.9445 0.9073 0.8610 0.8061 0.7432 0.6728 0.5957 0.5127 0.4245 0.3320 0.2363 0.1381 0.0386 -0.0613 -0.1606 -0.2583 -0.3534 -0.4450 -0.5322 -0.6140 -0.6897 -0.7586 -0.8198 -0.8729 -0.9173 -0.9525 -0.9782 -0.9941 -1.0001 -0.9961 -0.9821 -0.9584 -0.9250 -0.8825 -0.8311 -0.7714 -0.7041 -0.6296 -0.5490 -0.4628 -0.3720 -0.2775 -0.1802 -0.0811 0.0188 0.1185 0.2170 0.3134 0.4066 0.4958 0.5800 0.6584 0.7302 0.7947 0.8513 0.8994 0.9385 0.9683 0.9883 0.9985 0.9987 0.9889 0.9693 0.9399 0.9012 0.8534 0.7972 0.7329 0.6614 0.5832 0.4992 0.4103 0.3172 0.2209

Fractional order alpha= 1 Sampled derivatives are

0.0000 (omit) 0.9983 0.9884 0.9685 0.9390 0.9001 0.8522 0.7958 0.7314 0.6597 0.5814 0.4974 0.4083 0.3152 0.2189 0.1205 0.0208 -0.0791 -0.1782 -0.2755 -0.3700 -0.4609 -0.5471 -0.6279 -0.7024 -0.7699 -0.8297 -0.8812 -0.9239 -0.9574 -0.9813 -0.9954 -0.9995 -0.9937 -0.9780 -0.9524 -0.9174 -0.8732 -0.8202 -0.7591 -0.6904 -0.6147 -0.5330 -0.4459 -0.3544 -0.2593 -0.1616 -0.0623 0.0376 0.1371 0.2353 0.3311 0.4236 0.5119 0.5950 0.6722 0.7427 0.8058 0.8608 0.9073 0.9446 0.9725 0.9907 0.9990 0.9974 0.9857 0.9642 0.9331 0.8926 0.8433 0.7855 0.7198 0.6470 0.5677 0.4827 0.3929 0.2992 0.2025 0.1038 0.0040 -0.0958 -0.1947 -0.2916 -0.3856 -0.4757 -0.5611 -0.6409 -0.7143 -0.7805 -0.8390 -0.8890 -0.9302 -0.9621 -0.9844 -0.9968 -0.9993 -0.9918 -0.9743 -0.9472 -0.9106 -0.8649 -0.8105 -0.7480 -0.6781 -0.6014 -0.5187 -0.4308 -0.3386 -0.2430 -0.1450 -0.0455 0.0544 0.1537 0.2516 0.3469 0.4388 0.5262 0.6085 0.6846 0.7539 0.8156 0.8693 0.9142 0.9500 0.9763 0.9928 0.9994 0.9961 0.9828 0.9596 0.9269 0.8849 0.8341 0.7750 0.7081 0.6341 0.5538 0.4679 0.3774 0.2831 0.1860 0.0870 -0.0128 -0.1125 -0.2111 -0.3076 -0.4010 -0.4904 -0.5749 -0.6537 -0.7259 -0.7909 -0.8480 -0.8966 -0.9362 -0.9665 -0.9871 -0.9979 -0.9987 -0.9895 -0.9704 -0.9417 -0.9035 -0.8563 -0.8005 -0.7368 -0.6657 -0.5879 -0.5043 -0.4156 -0.3227 -0.2267 -0.1284 -0.0287 0.0712 0.1703 0.2678 0.3626 0.4538 0.5405 0.6217 0.6967 0.7648 0.8252 0.8774 0.9209 0.9551 0.9797 0.9946 0.9996 0.9945 0.9796 0.9548 0.9205 0.8770 0.8247 0.7642 0.6961 0.6210 0.5397 0.4530 0.3618 0.2670 0.1695 0.0703

As in the foregoing example, here too the accuracy improves when the function cos(t) is sampled with higher frequency. If $h = .01$ (implying 10 times higher frequency), then the result will be correct up to 3 significant digits. However, if the sampling is done with frequency increasingly too large, then the result will deteriorate since the computational error starts creeping in depending on the precision used. For most practical usage, too large a sampling frequency should not be used since instead of increasingly improving the result, it degenerates (worsens) the result.

Riemann-Liouville (RL) Derivative The 3 most frequently used definitions for the general fractional differintegral are GL, RL (Riemann-Liouville), and Caputo definitions. We have already discussed GL derivative in both reverse and direct forms. In terms of the fundamental continuous integrodifferential operator defined as

$$D_t^\alpha = \frac{d^\alpha}{dt^\alpha} : \alpha > 0; D_t^\alpha = 1 : \alpha = 0; D_t^\alpha = \int_a^t d\tau^\alpha : \alpha < 0 \tag{32}$$

where a,t are bounds of the operation and $\alpha \in R$ is a real fractional (including of course integer) order [6, 8].

The RL derivative for the general function f(t) is

$$D_t^\alpha f(t) = \frac{1}{\Gamma(n-\alpha)} \frac{d^n}{dt^n} \int_a^t \frac{f(\tau)}{(t-\tau)^{\alpha-n+1}} d\tau, \quad (n-1 < \alpha < n) \tag{33}$$

The RL definition for the fractional derivative of $f(t)$ involves both integration and differentiation (differintegral) unlike the integer order derivative which is mathematically

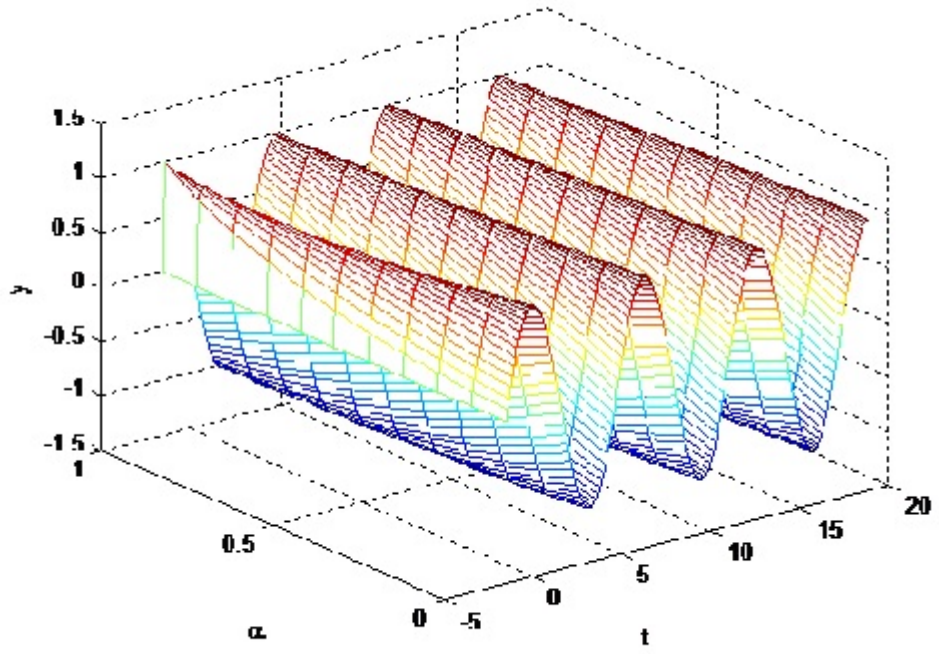

Fig. 2 Fractional derivative of cos(t) for $t = -\frac{\pi}{2}\ (h)6\pi\ for = 0(0.1)1, h = 0.1$

definable exactly for almost all analytic functions without any recourse to integration. Also, the restriction on the fractional order α viz. $(n-1 < \alpha < n)$ is a severe one in terms of generalization.

The mathematical integration in the equation (33) cannot be obtained in a closed form (unlike the fractional integration of functions such as $sin(x), e^x, x^k$) for most functions, a numerical integration is always possible for all computable mathematical analytical functions though.

However, we simply have to resort to mathematical/numerical differentiation only for derivatives of any integer order without any restriction on the integer order (unlike the fractional order α). Computational error and computational complexity in such definitions of fractional derivatives are usually more dominant and involved than those for integer order derivatives.

Can we devise/innovate a simple procedure (numerical or otherwise) that will obviate these problems in the process of achieving rather a straight-forward generalization for the derivative of any fractional order? There is a scope to explore this issue and come out either defining a straight-forward generalization or demonstrating/proving that such a generalization is impossible.

We leave the computational exploration of the RL fractional derivative definition for the reader so that he could find out the pros and cons of this definition compared with

those of Grunwald-Letnikov and Caputo definitions (described below).

Caputo derivative The Caputo derivative for the general function f(t) is

$$D_t^{\alpha} f(t) = \frac{1}{\Gamma(n-\alpha)} \int_a^t \frac{f^n(\tau)}{(t-\tau)^{\alpha-n+1}} d\tau, \quad (n-1 < \alpha < n) \tag{34}$$

The foregoing comments and question are more or less valid here too.

To illustrate mainly the numerical computation of the integration in equation (34), we consider the data $n = 1,\ f(t) = sin(t),\ \alpha = .99, a = 0, t = \pi$. The Caputo fractional derivative of sin(t) can be written as

$$D_t^{\alpha} sin(t) = \frac{1}{\Gamma(1-\alpha)} \int_a^t \frac{cos(\tau)}{(t-\tau)^{\alpha}} d\tau, \quad (0 < \alpha < 1)$$

$$= \frac{1}{\Gamma(0.01)} \int_0^{\pi} \frac{cos(\tau)}{(t-\tau)^{0.99}} d\tau.$$

The following 5-line Matlab program is named caputo_alp_sin.m

```
clear all; close all; global alp bet; alp=0:.01:1;
for i=1:101,
bet=alp(i); Q(i)=quadgk(@caputo_sind, 0, pi)./gamma(1 - bet);
end; disp( '    alpha derivative');
disp([alp' 'Q']); plot(alp, Q);

%Usage
% >> caputo_alp_sin
%uses function caputo_sind; computes fractional derivatives of sin(pi)
%Accuracy diminishes.

function y=caputo_sind(x)
%fractional (0 ≤ alp ≤ 1) derivative of sin (pi)
global alp bet; %alp=0:01:.9, bet=alp(i);
y=cos(x)./(pi-x). ^bet; % (0 <= alp <= 1)
```

The results (retaining only partially to conserve space) are as follows.

alpha derivative 0 0.0000 (omit this row)
0.0100 -0.0185
0.0200 -0.0368
0.0300 -0.0551
0.0400 -0.0732

. . .

0.9600 -0.7603

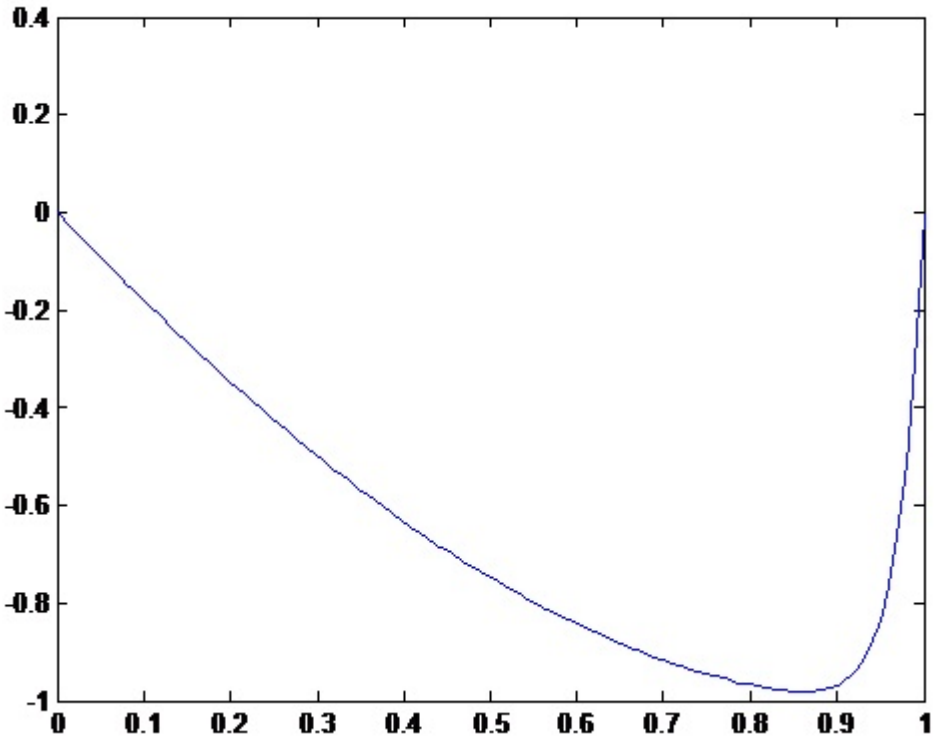

Fig. 3 Fractional order $\alpha = 0(.01)1$ versus the Caputo derivative of $sin(\pi)$

0.9700 -0.6569
0.9800 -0.5094
0.9900 -0.2992
1.0000 0

The graph (fractional order $\alpha = 0(.01)1$ versus the derivative of $sin(\pi)$ is as in (Fig. 3).

It can be seen both from the numerical results as well as from the graph that the computational error is considerable. For α close to 0.9 the derivative of $sin(\pi)$ is close to -1 while the derivative should be -1 for $\alpha = 1$, However, a better programming skill along with an increased precision would produce better accuracy.

Consider the Caputo fractional derivative of sin(pi/2) where the fractional order =.5,.9, .99,.999,.9999,.99999. The Matlab program is

```
function y=caputo_sin(x)
%fractional (0 ≤ alp ≤ 1) derivative of sin (pi)
global alp bet; %alp=.999; bet=alp;
```

y=cos(x)./(pi-x).^alp; % ($0 \leq alp \leq 1$)

%Usage for Caputo derivative ($0 < alp < 1$)

%clear all; close all; format long g; global alp bet;
%alp=.999; bet=alp; Q=quadgk(@caputo_sin, 0,pi/2)./gamma(1-bet);

%' alp alp-derivative of sin(pi/2)', disp([bet Q])

%Write foregoing 3 lines in 1 line in command window omitting %
% and replace "alp=.999" by "alp=.5" when we wish to compute
% the fractional derivative of sin(pi/2) at alp=0.5 .

%global alp bet; alp=.99, bet=alp; Q=quadgk(@ caputo_sin,
%0,pi/2)./gamma(1-bet); disp([bet Q]) OR

%>> clear all; close all; global alp bet; alp=.99; bet=alp;
%Q=quadgk(@ caputo_sin, 0,pi/2)./gamma(1.-bet); disp([bet Q])

%alp =

% 0.9900

% 0.9900 0.0040

% Caputo derivative viz. D^alp(sin x) for x= pi/2 is .0040
%when alp = .99 and 3.9892e-004 when alp = .999.
% quadgk(@ caputo_sin, 0, pi/2) when alp=.99
%x should not be pi because of division by 0
% and also it exceeds max no of divns. permitted.

We obtain the following output for the fractional derivative of sin(pi/2) at the fractional order α=.5,.9,.99,.999..9999,.99999

α	α order derivative of $sin(\pi/2)$
0.5	0.354970157186708
0.9	0.0458549655973837
0.99	0.00404259546470162
0.999	0.000398924379050003
0.9999	3.98391892857323e-005
0.99999	3.98338654632647e-006

Observe that the foregoing result provides some idea about the accuracy of Caputo fractional order derivative. Specifically one can see how well the fractional order α merges with the integer order 1.

Here the generalization of the fractional derivative through numerical integration is not a straight-forward approach. Can we have such an approach with improved accuracy without limiting ourselves among several definitions of fractional derivatives as mentioned above? As a matter of fact, one may have totally new definition for fractional derivatives and fractional integrals, that would eventually be widely accepted as a generalization of classical calculus.

On the other hand, can we prove that no more definition better than the existing ones is possible? We believe that we have not come to a dead end; there is still a scope of deeper exploration.

An important point is the readily understood precise physical interpretation of every fractional order derivative and every fractional order integral just like every integer order derivative and integer order integral. A generalization demands such an interpretation so that we need not worry about whether the order is an integer or a fraction.

Every integer/fractional order derivative/integral carries a message. Every derived step in a procedure also gives us a distinct message. It is a good habit to readily get/understand the message carried by a step/equation. This will immensely help us to grasp/to get a complete feel of what is going on in the process/procedure. One should never behave like a machine oblivious of what is going around us. Only then we will have the right frame of mind for the generalization or for showing mathematically that the desired generalization having similar applicable properties of generalized matrix inverses is impossible.

2.3. Fractional ordinary differential equations (FODEs)

Any system of integer order ordinary differential equations (IODEs) with adequate initial/2-point boundary conditions can always be written as a system of n 1^{st} order odes with n initial/2-point boundary conditions. The general form can be written follows.

$$\begin{aligned}
\frac{dy_1}{dt} &= f_1(t, y_1, y_2, ..., y_n) \\
\frac{dy_2}{dt} &= f_2(t, y_1, y_2, ..., y_n) \\
&\quad . \quad . \quad . \\
\frac{dy_n}{dt} &= f_n(t, y_1, y_2, ..., y_n)
\end{aligned} \tag{35}$$

with n initial conditions: At $t = T_0$,$y_1 = \alpha_1$,$y_2 = \alpha_2$,,$y_n = \alpha_n$ or with n boundary conditions where some of these n conditions are specified at $t = T_0$ while the rest at $t = T_final$. The functions $f_1, f_2, ..., f_n$ are written from a given actual system of IODEs with the initial/boundary conditions. Observe that $\alpha_i's$ are specified numerical values and are not connected with the fractional order α.

Similarly FODEs of fractional order $0 < \alpha < 1$ can be written as

$$\begin{aligned}
\frac{d^\alpha y_1}{dt} &= f_1(t, y_1, y_2, ..., y_n) \\
\frac{d^\alpha y_2}{dt} &= f_2(t, y_1, y_2, ..., y_n) \\
&\cdot \quad \cdot \quad \cdot \\
\frac{d^\alpha y_n}{dt} &= f_n(t, y_1, y_2, ..., y_n)
\end{aligned} \tag{36}$$

with n initial conditions: At $t = T_0$,$y_1 = \alpha_1$,$y_2 = \alpha_2$,,$y_n = \alpha_n$ or with n boundary conditions where some of these n conditions are specified at $t = T_0$ while the rest at $t = T_final$. The functions $f_1, f_2, ..., f_n$ are written from a given actual system of IODEs with the initial/boundary conditions.

Consider, as an example, the projectile motion problem. A projectile of mass m is launched with initial velocity v_0 at an angle θ at time $t = 0$. Atmosphere exerts a resistance force R on the mass, proportional to the velocity of the mass at time t, i.e.$R = \beta v(t)$, where β is a constant and is opposite to the direction of the velocity of the mass.

At time t, the mass is at location $(x(t), y(t))$; the velocity of the mass in the x-direction and that in the y-direction are $\dot{x}(t), \dot{y}(t)$, respectively. Thus the velocity of the mass is $v(t) = \sqrt{(\dot{x}^2(t) + \dot{y}^2(t))}$ at an angle $\theta(t) = tan^{-1}\frac{\dot{y}(t)}{\dot{x}(t)}$. The x-component of resistance force R is $R(t)cos\theta(t)$ and y-component of R is $R(t)sin\theta(t)$. Using Newtons 2nd law of motion, we can write

$$\begin{aligned}
x - direction: &\qquad m\ddot{x}(t) = -R(t)cos\theta(t) \\
y - direction: &\qquad m\ddot{y}(t) = -mg - R(t)sin\theta(t)
\end{aligned} \tag{37}$$

The equation of motion (mathematical model) becomes

$$\begin{aligned}
m\ddot{x}(t) = -\beta v cos\theta(t) = -\beta v\frac{\dot{x}}{\sqrt{\dot{x}^2 + \dot{y}^2}} \quad & i.e.,\ m\ddot{x} + \beta\dot{x} = 0 \\
m\ddot{y}(t) = -mg - \beta v\frac{\dot{y}}{\sqrt{\dot{x}^2 + \dot{y}^2}} \quad & i.e.,\ m\ddot{y} + \beta\dot{y} = -mg
\end{aligned} \tag{38}$$

$$IC:\ at\ t = 0, x(0) = 0, y(0) = 0, \dot{x}(0) = v_0 cos\theta_0, \dot{y}(0) = v_0 sin\theta_0$$

Hence the system of 4 1st order ODEs (computational mathematical model suitable for computer implementation for n (general) 1st order ODEs) can be written as follows. Let $x = x_1, y = x_2, \frac{dx}{dt} = x_3, \frac{dy}{dt} = x_4$. Then the equation of motion (mathematical model) becomes

$$\frac{dx_1}{dt} = f_1(t, x_1, x_2, x_3, x_4) = x_3$$
$$\frac{dx_2}{dt} = f_2(t, x_1, x_2, x_3, x_4) = x_4$$
$$\frac{dx_3}{dt} = f_3(t, x_1, x_2, x_3, x_4) = \frac{-\beta x_3}{m}$$
$$\frac{dx_4}{dt} = f_4(t, x_1, x_2, x_3, x_4) = \frac{(-mg - \beta x_4)}{m} \tag{39}$$

$IC : at\ t = 0, x_1 = 0, x_2 = 0, x_3 = v_0 cos\theta_0, x_4 = v_0 sin\theta_0.$
Compute: at $t = 0(0.1)5, x_1, x_2, x_3, x_4$ given $v_0 = 3000 meter/s, \beta = 0.05,$
$g = 9.8 meter/s^2.$

Introducing the fractional order α such that $0 < \alpha < 1$ in the foregoing system of 4 1st order ODEs and using the Matlab routine fde12s [9 − 13], we can solve a system of n fractional order ODEs (n is a finite positive integer).

Refer the systems (36). It can be seen that when α =0, the differential equations cease to exist. On the other hand, when α =1, the system ceases to be FODEs. It becomes the classical 1st order ODEs as in the system (39).

However, we consider first the classical (non-fractional) pendulum problem where no damping exists. The Matlab version (program) of this problem is named as pendulum. Then we present the Matlab program pendulumfde.m, where α =.97, which uses the Matlab routine fde12s. Both the programs are compatible with the computational mathematical notation and thus self-explanatory. Further both use the 3-line Matlab function sub-program pend shown below both the programs pendulum and pendulumfde.

The 2nd order ODE representing the motion of a simple gravity pendulum i.e. the simple harmonic motion is

$$\frac{d^2z}{dt^2} + \frac{g}{l} sinz = 0, \quad i.e., \ \frac{d^2z}{dt^2} = -kz \tag{40}$$

where $z = a$ small ($\leq$ 20 *degree or* 0.3491 radian) displacement, g= acceleration due to gravity, l= length of the pendulum, $k = g/l$. At time $t = 0, z = z_0, \dot{z} = v_0$. Expressing the foregoing 2nd order ODE into 2 1st order ODEs with 2 initial conditions, one can readily check the concerned pend Matlab function sub-program.
Pendulum non-fractional Matlab program (Main) uses Matlab routine ode23 and the Matlab function sub-program pend.m

```
%9-line Matlab program pendulum
[t,z]=ode23('pend',  [0,20], [1,0]);
disp(' t col 1 of z col 2 of z');
disp ([t z]);
plot (t,z(:,1), t, z(:, 2));
xlabel('t'); ylabel('col 1 of z & col 2 of z');
```

```
figure (2)
plot(z(:,1),z(:,2));
xlabel('Displacement');ylabel('Velocity');
title('Phase Plane Plot');
%Usage Type pendulum in command window

%pend.m Matlab function sub-program
function zdot=pend(t, z);
wsq=1.56;
zdot=[z(2); -wsq*sin(z(1))];
```

>> pendulum Integer-order alpha=1)

t	$col\ 1\ of\ z$	$col\ 2\ of\ z$
0	1.0000	0
0.0001	1.0000	− 0.0001
0.0004	1.0000	− 0.0005
0.0019	1.0000	− 0.0025
0.0095	0.9999	− 0.0125
19.3178	− 0.7918	0.6916
19.5401	− 0.6120	0.9163
19.7743	− 0.3755	1.0899
19.9454	− 0.1820	1.1634
20.0000	− 0.1181	1.17614

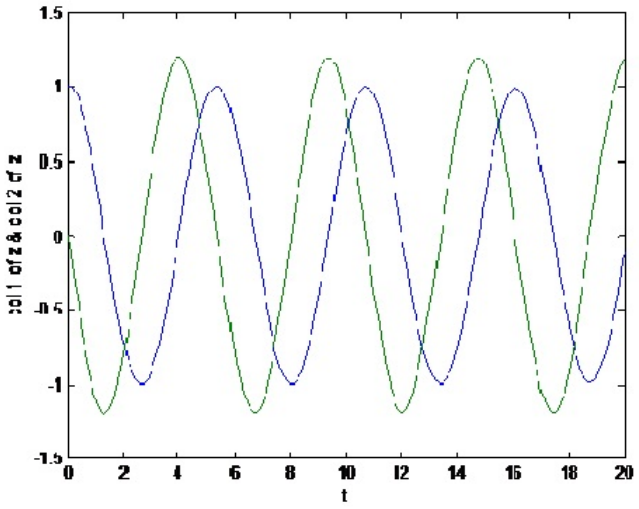

Fig. 4 The graph of z_1, z_2 for time t=0 to 20.

Pendulumfde.m matlab program and the Matlab function subprogram pend are as follows. One can change/vary the value of alpha (α) in the program and observe for himself the damped motion of the pendulum for each alpha.

```
global h y0;
%alpha=1
%alpha=.9
%alpha=.99
%alpha=.98
%alpha=.97
h=.1739; y0=[1;0]; alpha=.97,
[t,z]=fde12s(alpha,'pend',0,20);
%global h y0;
%h=1; y0=[1;0]
disp(alpha');
disp(' t col 1 of z col 2 of z');
disp ([t' z']);
plot (t',z(1,:)', t', z(2, :)');

xlabel('t'); ylabel('col 1 of z & col 2 of z');
figure (2)
plot(z(1,:),z(2,:));
xlabel('Displacement');ylabel('Velocity');
title('Phase Plane Plot');
```

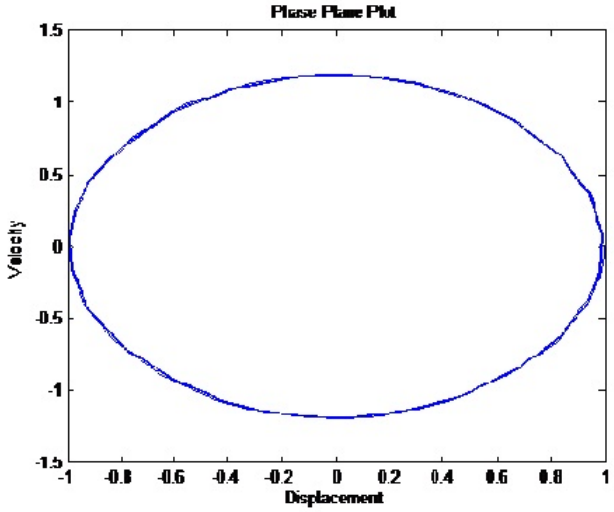

Fig. 5 The Phase-Plane plot Displacement vs. Velocity (Observe that there are rotations (non-spiral).)

%Usage

function zdot=pend(t, z);
wsq=1.56;
zdot=[z(2); -wsq*sin(z(1))];

Data and Results

>> pendulumfde

alpha = 1

t	$col\ 1\ of\ z$	$col\ 2\ of\ z$
0	1.0000	0
0.1739	0.9802	− 0.2283
0.3478	0.9209	− 0.4506
0.5217	0.8237	− 0.6602
0.6956	0.6916	− 0.8486
. .	.	
19.4768	− 0.6150	0.9634
19.6507	− 0.4338	1.1011
19.8246	− 0.2324	1.1915
19.9985	− 0.0198	1.2266

20.0000 − 0.0180 1.2264

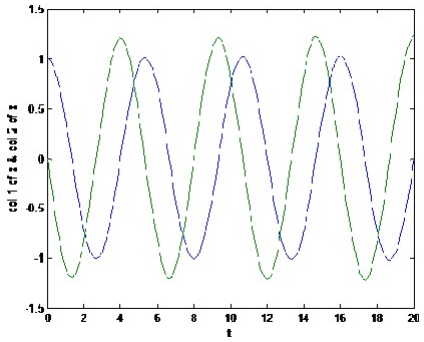

Fig. 6 z_1, z_2 for t=[0,20]. $\alpha = 1$ (non-fractional no damping as in the foregoing simple pendulum)

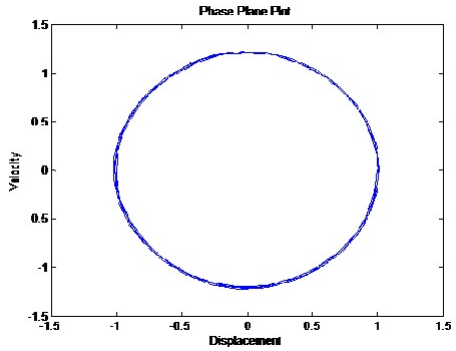

Fig. 7 Phase-plane plot for alpha = 1 (The graph goes round and round on the same closed curve no spiral)

alpha = 0.9900

t	$col\ 1\ of\ z$	$col\ 2\ of\ z$
0	1.0000	0
0.1739	0.9792	− 0.2333
0.3478	0.9178	− 0.4570
0.5217	0.8184	− 0.6661
0.6956	0.6842	− 0.8525
. .	.	
19.4768	− 0.0901	0.8556
19.6507	0.0610	0.8569
19.8246	0.2083	0.8173
19.9985	0.3447	0.7395
20.0000	0.3458	0.7386

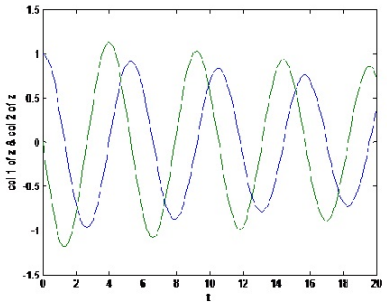

Fig. 8 z_1, z_2 for alpha=.99 (Observe that due to slight damping (since alpha slightly reduced) the amplitudes decrease gradually)

>> pendulumfde

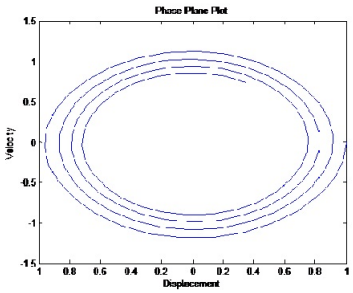

Fig.9 Phase-plane plot (Observe that the graph (line goes round and round with successive decrease in radius due to slight damping spiral formation)

α= 0.9800

t	$col\ 1\ of\ z$	$col\ 2\ of\ z$
0	1.0000	0
0.1739	0.9781	− 0.2384
0.3478	0.9147	− 0.4634
0.5217	0.8129	− 0.6720
0.6956	0.6767	− 0.8563
0.8695	0.5115	− 1.0069
. .	.	
19.4768	0.1044	0.5781
19.6507	0.2017	0.5322
19.8246	0.2881	0.4619
19.9985	0.3597	0.3713
20.0000	0.3602	0.3704

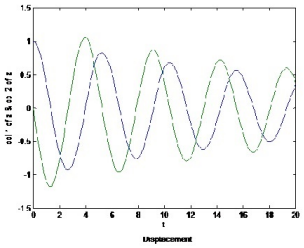

Fig.10 z_1, z_2 for alpha=.98 (Observe that due to further reduced value of alpha the amplitudes decrease still further)

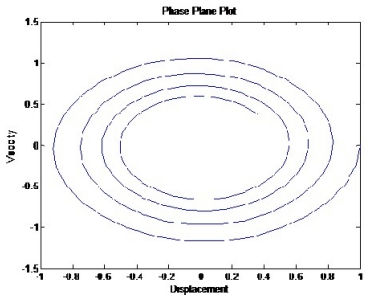

Fig. 11 Phase-plane plot for alpha=.98; a spiral converging gradually.

alpha = 0.9700

t	$col\ 1\ of\ z$	$col\ 2\ of\ z$

0	1.0000	0
0.1739	0.9771	− 0.2436
0.1739	0.9771	− 0.2436
0.3478	0.9114	− 0.4699
0.5217	0.8073	− 0.6778
0.6956	0.6691	− 0.8599
. .	.	
19.4768	0.1437	0.3571
19.6507	0.2009	0.3062
19.8246	0.2474	0.2418
19.9985	0.2812	0.1675
20.0000	0.2813	0.1668

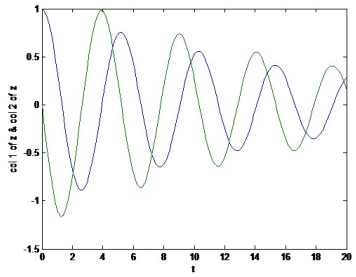

Fig. 12 z_1, z_2 for alpha=.97 (Observe that the amplitudes decrease more sharply since alpha is further reduced)

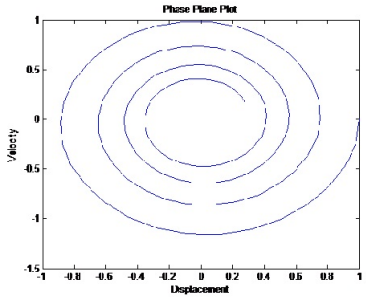

Fig. 13 Phase-plane plot for alpha=.97 (The spiral becomes shorter as alpha becomes smaller.)

alpha = 0.9000

t	$col\ 1\ of\ z$	$col\ 2\ of\ z$
0	1.0000	0

0.1739	0.9680	− 0.2827
0.3478	0.8854	− 0.5161
0.5217	0.7644	− 0.7167
0.6956	0.6134	− 0.8799
0.8695	0.4411	− 0.9986
. .	.	
19.4768	0.0121	0.0014
19.6507	0.0128	− 0.0024
19.8246	0.0128	− 0.0062
19.9985	0.0121	− 0.0097
20.0000	0.0121	− 0.0097

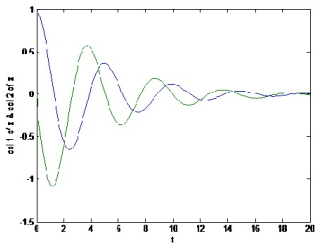

Fig. 14 z_1,z_2 for alpha=.90 (Observe that due to increased damping (i.e. reduced alpha) the amplitudes almost merge as these should be.)

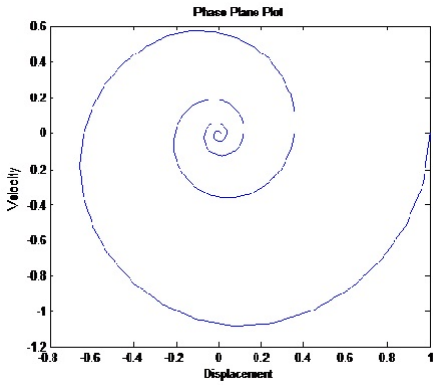

Fig. 15 Phase-plane plot looks like a snail's back i.e. the spiral rapidly converges to a point for alpha =0.9.

Remarks Zero Friction force (Ideal case): Non-existent in Nature If alpha (fractional order, in general) is 1, then the pendulum will continue simple harmonic oscillations

without any damping. The oscillation will have constant amplitude, constant time period, and will continue the oscillatory motion for ever. The Matlab program for pendulum depicts the motion (Figs. 4 and 5).

(Positive) Friction force. If alpha<1 is fractional and assumes successively the values .99, .98, and .97, .90, then the pendulum will continue simple harmonic oscillations/motion with progressively increased damping. While the time period remains unaltered, the amplitude will progressively decrease.

The Matlab program pendulumfde depicts the motion (Figs. 8-15). It can be seen that for alpha=1 both the programs pendulum and pendulumfde depict beautifully same/similar graphs.

Figs. 4 and 5 output by pendulum correspond to Figs. 6 and 7 produced by pendulumfde, respectively. For such a specific type of FODEs, both the fractional order and the integer order merge and integrate well. Consequently the efforts toward limited generalization are appreciated. But the physical implication of alpha in terms of friction forces and that of friction forces in terms of alpha are not readily known (as we do for IODEs).

Implication of friction forces. In other words, friction forces s uch as resistance due to air/wind and that due to various parts of the pendulum touching one another should determine the value of alpha viz. the value of the fractional order approximately. Can we still have precise magnitude of all the friction forces combined together from the knowledge of alpha and vice versa? A meaningful generalization needs to sort out this question.

Is it possible that throughout the interval of the independent (time) variable t, the combination of friction forces and the value of alpha match/are numerically the same at all equi-spaced (or non-equi-spaced) points in the time interval? On the other hand, if we know the magnitude and direction of friction forces, then the mathematical model can be written as integer-order differential equations and then solved without any need of fractional order model.

3. FODEs: Converting to $\alpha - th$ or/and 1st order ODEs with generalizations

Consider, for instance, the following mathematical problem involving FODEs. Solve numerically the following system of 2 FODEs

$$\begin{aligned} \frac{d^{3.71}y}{dx^{3.71}} + \frac{d^{2.81}y}{dx^{2.81}} + \frac{dy}{dx} + y^2 &= sint \\ y\frac{d^{2.11}y}{dx^{2.11}} - \frac{d^{1.9}y}{dx^{1.9}} + y^{0.5} &= cost \end{aligned} \tag{41}$$

under the requisite number of initial conditions for t=0(.1)10.

The following questions arise: Can we express the system of equations (41) as a system of k 1^{st} order and/or m αth order ODEs? What will be the value of k and/or the value of m? Can we write a real-world physical problem that corresponds to the foregoing system (41) as the mathematical model? Or, can we have a real-world physical problem whose

mathematical model will be one similar to the system (41)? Can we have a procedure to solve a corresponding system of 1st order and/or αth order ODEs?

A generalization deserves the answer of the foregoing questions. Only after obtaining the answer, we may determine if a complete generalization (i.e. a generalization that does not at all differentiate between an integer-order say, 3 or 4 and a fractional order say 2.21 and 3.78 and get the desired numerical solution) is achievable. Or, to which extent a generalization is feasible.

What are the specific procedures that need to be designed/ innovated to achieve a partial generalization or a complete one? Then, of course, comes the efforts to solve numerically computationally the problems for scientific/engineering implementation following the designed new procedures.

4. Conclusions

Generalizing matrix inverse and calculus: Analogy Generalized version of a matrix inverse and that of the calculus viz. fractional calculus do not have one-to-one correspondence so far as an analogy is concerned. One is fractional while the other is non-fractional. Till early/mid- 20^{th} century matrix algebra was in a formative stage. We knew how to solve linear consistent square full-rank system using the true matrix inverse. Or, without recourse to matrix algebra, we could solve the system using methods such as the Gauss reduction method using partial/complete pivoting (during 19^{th} century).

However, the essence of the concept of generalizing the true inverse so that we can solve any linear system written/given in the matrix-vector form consistent or not, square or rectangular without any botheration about pruning redundant (linearly dependent) rows of the system was developed during around mid-20^{th} century.

For the inconsistent (contradictory linear equations) system, we do not have any solution that satisfies all the equations. But a specific generalized inverse of the coefficient matrix of the equation in matrix form, called the pseudo-inverse, provides the minimum-norm least-squares inverse (i.e. $||x||$ = minimum implying minimum norm as well as $||Ax - b||$ =minimum implying least squares, where x is the computed solution of the linear system Ax=b, and ||.|| is the Euclidian matrix norm).

Consequently, we obtain the minimum-norm least-squares solution of the system. This solution is of paramount importance and extensively used in whole of engineering and science as long as the degree of inconsistency (contradiction among the linear equations) is within acceptable limit depending on the context.

If there is unacceptable inconsistency in the linear system, then one should fall back to the physical problem, find out the cause for abnormal inconsistency and correct the error/mistake. It can be seen that Nature knows no inconsistency. The inconsistency in a linear system (viz. the mathematical model corresponding to a physical problem of Nature) could result only due to human error/mistake.

Since error (error (not mistake) in measurement and/or in computation) is ever impossible to be removed, it could often (not always) result in inconsistent/near-consistent system. We need to deal with such a system almost always. So is the need for the

minimum-norm least-squares inverse (not the true inverse as it does not exist for most of the problems or is too erroneous due to near-singularity/non-square rectangularity of the coefficient matrix A) almost always in the physical world.

One can devise a row-pruning algorithm for a given linear system Ax=b to remove all redundant rows (i.e. linearly dependent rows) of the augmented matrix (A, b) [14 , 15]. Such an algorithm will act as a pre-processor for solving a linear system using the pseudo-inverse. The pruning reduces the computational error and also storage complexity whenever it occurs. However, this is not widely used mainly because it is an additional task and also, for most of the real-world problems, the current (2018) computing precision and also storage is not an issue to obviate the foregoing 2 problems. Thus the generalized inverse viz. the pseudo inverse of the matrix A is good enough for practically all physical problems.

With the foregoing essence of generalizing the matrix inverse in mind, can we have the similar essence of generalizing the classical calculus so that the generalized calculus (viz. fractional calculus) universally serves the purpose for all calculus (including differential equations) problems always? If we can, then the advantage of generalizing the calculus, of course including the physical interpretation of fractional order derivative/integral, will be enormous in terms of solving numerous problems in the physical world readily and easily.

Classical ode corresponding to any fractional ode A 1-independent variable FDE can be written as 1 or more distinct classical ODEs in so far as its solution is concerned. Consider, for example, 1 FDE where the fraction $\alpha < 1$. Its solution will correspond to a continuous function of 1 variable. Graphically, this will be a curve (a continuous line) in the concerned interval of the independent variable.

Corresponding to this graph, we can readily produce a classical ode of order 1 or more by successive differentiation. On the other hand, we may have a classical ODE (based, say, on the 2^{nd} law of Newton) appended with a friction force term obtained from a given physical problem.

We can easily interpret the corresponding physical problem for each of the foregoing classical ODEs. These interpretations are distinct in the sense that one physical interpretation cannot be obtained from other and vice versa, although their solutions are the same.

Best way to compute integer order derivatives While computing the derivative of an analytical function is not only easier but also is possible for a much larger set of functions by a concerned person. This is unlike analytical (not computational) integration of an analytical function.

Most of the functions cannot be analytically integrated in a closed form. A famous example is the normal distribution function

$$\Phi(z) = \frac{1}{\sqrt{2}} \int_0^z e^{-x^2} dx = erf(\frac{z}{\sqrt{2}}) \tag{42}$$

where erf= error function, gives the probability that a standard normal variate assumes a value in the interval $[0, z]$. Neither $\Phi(z)$ nor erf can be expressed exactly in terms of a finite number of add, subtract, multiply, square-root operations. So both must be either

computed numerically or approximated (to a finite number of terms). In either case we get only erroneous (acceptable or not depending on the context) values (numerical or mathematical).

However, numerical computation of a derivative and of an integral with specified numerical bounds (limits of integration) is not only always possible but also absolutely required in every engineering/science application.

The numerical computation of a derivative of higher integer order (order 2 or more) does involve increasingly more pronounced error. Successive numerical derivatives are successively more erroneous since a higher order numerical derivative is computed based on already computed lower order derivative having inherent computational error. Like entropy, errors go on increasing successively the higher the derivatives we compute.

Under these circumstances, a best way to compute integer order derivatives would be to use (successive) symbolic derivative computations followed by the numerical derivative computation at the final step. The successive symbolic computations can be carried out exactly in Matlab for most (not all) of the analytic functions and/or for a combination of this functions. The foregoing best way will provide least computational error (for a given precision of the computer) which is most desired in all forms of computations.

Sometimes the successive integer-order derivative of a non-polynomial function or a combination of non-polynomial and polynomial functions could result in a much larger analytical function implying much more numerical computation. In such a case, one may follow one of the 2 ways viz. the foregoing "a best way" and the straight-forward (usual) successive numerical computation way (from the very 1st step) for the given function and decide one of the ways, that produces less computational error. The computational complexity (cost of computation) issue is mostly a minor/trivial issue in this hyper-computing (over 10^{18} floating-point operations per sec) era.

The difficulty of determining symbolic derivative for a long involved function (not being able to write/automate the symbolic derivative computation for all highly involved/lengthy analytic functions) is due to the programmers limitations in his comprehension (A common human being can comprehend 5 to 9 things at a time.) and his programming skill involving differentiation of a general function. Such limitations may not come into picture for a person/programmer when he uses the rules of symbolic derivation with sufficient care and concentration for a specific function or a combination of functions.

However, a mistake, though rare, cannot be ruled out for any living being including a mathematical/computational prodigy (as to err (mistake) is human (living being) is eternally true without any exception. In this context, Not to err is computer (non-living being) may be considered equally true).

For computing a fractional order derivative of a function where the order $\alpha = n.\text{f}$, n being a positive integer and (.f) the fractional part, successive integer order derivatives can be obtained as in the foregoing case and then the fractional part (.f) could be computed according to one of the most appropriate definitions such as the Grunwald-Letnikov definition. Here a generalization procedure that could be considered a straight-forward extension of the integer order derivatives deserves to be devised/designed for a widely accepted integration/merger.

When the function is known as equi-spaced sampled numbers or non-equi-spaced sampled numbers, then the problem of computing derivative is usually completely numerical. The symbolic derivation successive or not that either physically inflates the function or deflates the function usually will not figure here.

Numerical input functions. It is possible that we may deal with functions which are known through tables of numbers. After all, everything has to be in numbers for every practical/real-world application. The problem of generalization seems to be comparatively easy (if it is possible) since we may not have to worry too much about a large array of mathematical functions and their varying properties over the concerned domain. All we need to worry is about computational error and occasionally computational complexity subject, however, to continuity/analyticity of the functions.

For the differential equations integer-order as well as fractional order we would get the solution viz. the function not in terms of a mathematical function but in terms of a numerical table containing a number of rows, every row consists of values of independent variables and the corresponding function value. Here too the problem of generalization appears to be relatively simple. However, the background mathematical knowledge in this subject area is extremely (sometimes critically) helpful to completely and confidently rely on the numerical results and the concerned graphs. Blindly/mechanically doing computations without getting a feel/an understanding about what the numbers are saying (or giving us message) is unacceptable.

Problems and efforts toward generalization. We have mentioned the various problems and innovative efforts toward generalization of fractional calculus not only in section 2 (Problems and Efforts), but also in section 1 (Introduction). Section 3 that deals with fractional ordinary differential equation may be considered as a good effort toward generalization. The pain-staking effort of the author in writing a detailed Matlab program that works well for ODEs with, of course, restricted fractional order is remarkable. However, the restriction on the fractional order is an important issue that needs to be sorted out for a possible usable generalization.

Any system of ODEs can always be written as a system of n 1st order ODEs with n initial conditions or n 2-point boundary conditions. A general computer (Matlab, say) program for the system of n 1st order ODEs can always be written [16]. The widely used Matlab provides such a program for n 1st order ODEs with n initial conditions. Such a general computer program cannot be written for a system of integer order partial differential equations (PDEs). Fractional partial differential equations are no exceptions. However, we are not much concerned with this issue in this context.

Quality and cost of the solution. How do the quality (relative computational error) and the cost (computational complexity) of the solution of a system of FDEs differ from that of the equivalent system of integer order differential equations? Which solution will be better? These questions, though appear to be out of context for generalization, are still important to support the scope of fractional order systems in calculus and to make it an integral part of the current calculus that we are taught globally.

Acknowledgements

The authors thank the Science and Engineering Research Board (SERB) of the Department of Science and Technology (DST), Government of India for their support of the reported work under the project DST SERB EMR/2016/003572 dated Feb 06, 2017.

References

[1] R. Herrmann, Fractional Calculus: *An Introduction for Physicists, World Scientific,* New Jersey, 2010.

[2] V. Lakshmikantham and S.K. Sen, *Computational Error and Complexity in Sience and Engineering,* Elsevier, Amsterdam, 2005.

[3] C. R. Rao and S. K. Mitra, *Generatized Inverse of Matrices and Its Applications,* Wiley, 1972.

[4] G. H. Golub, van Loan, and F. Charles, it Matrix Computations (3rd ed.), Johns Hopkins University Press, 1996.

[5] K. B. Oldham and J. Spanier, *The Fractional Calculus,* Academic Press, 1974.

[6] K. S. Miller and B. Ross, *An Introduction to the Fractional Calculus and Fractional Differential Equations,* Wiley, New York, 1993.

[7] I. Podlubny, *Fractioal Differential Equations,* Academic Press, San Diago, 1999.

[8] I. Petras, *Fractional derivatives, fractional integrals, and fractional differential equations in Matlab, 239-264, in: Engineering Education and Research Using Matlab, ed.:* Ali Assi, InTech, 480 Pages, 2011.

[9] K. Diethelm, A.D. Freed, *The Frac PECE subroutine for the Numerical solution of differential equations of fractional order,* in: S. Heinzel, T. Plesser (Eds.), Forschung und Wissenschaftliches Rechnen 1998, Gessellschaft fur Wissenschaftliche Datenverarbeitung, Gottingen, 1999, pp. 57-71.

[10] K. Diethelm, N.J. Ford, A.D. Freed, *Detailed error analysis for a fractional Adams method, Numer. Algorithms,* 36 (1) (2004) 31-52.

[11] K. Diethelm, *Efficient solution of multi-term fractional differential equations using P(EC)mE methods,* Computing 71 (2003), pp. 305-319.

[12] E. Hairer, C. Lubich, M. Schlichte, *Fast numerical solution of nonlinear Volterra convolution equations, SIAM,* J. Sci. Statist. Comput. 6 (3) (1985) 532-541.

[13] R. Garrappa, *On linear stability of predictor-corrector algorithms for fractional differential equations,* Internat. J. Comput. Math. 87 (10) (2010) 2281-2290. Copyright

(c) 2011-2012, Roberto Garrappa, University of Bari, Italy, garrappa at dm dot uniba dot it, Revision: 1.2 - Date: July, 6 2012.

[14] S.K. Sen, *A row-pruning algorithm to compute a generalized inverse, 37th Congress of ISTAM and Symp. On Rheology of Biological Materials,* January 14-17, 1993, G.B. Pant Univ. of Agriculture and Technology, Pantnagar, India, 99-100 (abstract).

[15] V. Lakshmikantham, S.K. Sen, and S. Sivasundaram, *Concise row-pruning algorithm to invert a matrix, Applied Mathematics and Computation,* (Elsevier Science Pub. Co., New York), 60, 1994, 17-24.

[16] S.K. Sen and M. Chanda, *A general scheme for solving ordinary differential equations under two-point boundary conditions,* J. Ind. Inst. Sci., 54, 3, 1972, 139 145.

[17] F.M. Bayat, *Fractional Differentiator, MathWorks, Inc. Matlab Central File Exchange,* 2007, URL: www.mathworks.com/matlabcentral/fileexchange/13858