Fungrim: The Mathematical Functions Grimoire

Fredrik Johansson

Inria Bordeaux

OPSFA 2019, Hagenberg, Austria
July 24, 2019

What is Fungrim?

`http://fungrim.org`

An attempt to make a better (at least for me)

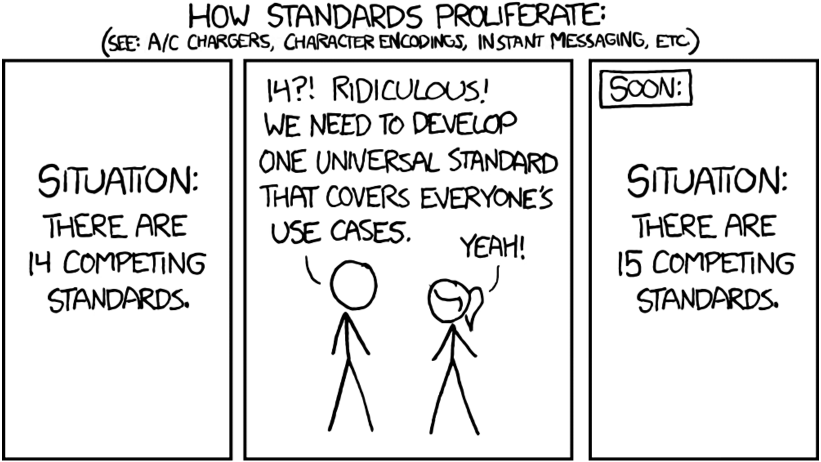
1. reference work
2. software library for symbolic computation

for special functions

---

grimoire = book of magic formulas

Relevant XKCD



`https://xkcd.com/927/`

Problems with existing reference works

Digital Library of Mathematical Functions

THE MATHEMATICAL FUNCTIONS SITE

WIKIPEDIA
The Free Encyclopedia

	`dlmf.nist.gov`	`functions.wolfram.com`	`wikipedia.org`
Open source?	×	×	✓
Symbolic?	×	✓	×
Good	Good presentation (edited by experts)	Well-structured, huge	Usually comprehensive
Bad	Terse, missing useful formulas, sometimes vague	*Mathematica* quirks and bugs, sometimes ugly formulas, missing some categories of info	Text-heavy, much trivia, often vague, inconsistent

Some reasons why the literature is frustrating to use

1. Vague or missing definitions
2. Conditions on variables not stated, ambiguous, or depend on non-local context
3. Implicit special cases, limits, analytic continuation, ...
4. The wanted formula can be derived by combining equation (43) with theorems 5 and 12... in a simple 10-page calculation, left as an exercise for the reader
5. The dreaded "$\approx$" sign
6. Errors (typos or more serious)
7. Text text text text text text text text text text

Some reasons why the literature is frustrating to use

1. Vague or missing definitions
2. Conditions on variables not stated, ambiguous, or depend on non-local context
3. Implicit special cases, limits, analytic continuation, . . .
4. The wanted formula can be derived by combining equation (43) with theorems 5 and 12... in a simple 10-page calculation, left as an exercise for the reader
5. The dreaded "$\approx$" sign
6. Errors (typos or more serious)
7. Text text text text text text text text text text

I'm personally as guilty as anyone, on all counts

Content goals for Fungrim

- Formulas as symbolic, machine-readable theorems
- Symbols have a globally consistent meaning
- Explicit assumptions for all free variables

$$|T_n(x)| \le 1, \quad n \in \mathbb{Z} \text{ and } x \in [-1, 1]$$

```
Entry(ID("15dd69"),
    Formula(LessEqual(Abs(ChebyshevT(n, x)), 1)),
    Variables(n, x),
    Assumptions(And(Element(n, ZZ),
        Element(x, ClosedInterval(-1, 1)))))
```

- Scope not limited by paper edition constraints
- Good coverage of inequalities, with explicit constants

Presentation goals for Fungrim

- Simple and fast to browse (including mobile!)
- Permanent ID and URL for each formula
- Beautiful formula rendering (symbolic expressions $\to$ TeX $\to$ KaTeX $\to$ HTML)
- Instant access to TeX code to copy and paste
- Instant access to symbolic representation
- Links to symbol definitions
- TODO: export to other languages, search functionality, browsing based on metadata

Non-goals (for now)

Formal proofs

- Randomized testing (to be done!) should be adequate to provide a high level of reliability

Fully computer-generated content

- Related: the Dynamic Dictionary of Mathematical Functions (`http://ddmf.msr-inria.inria.fr/1.9.1/ddmf`)

Covering all of mathematics

- Just special functions and elements of classical analysis and number theory

Long-term goal: symbolic computation

Applications of a library of formulas

- Automatic (or manual) term rewriting
- Code generation, testing mathematical software

What's missing in existing projects?

- Not open source
- More narrow scope
- Mathematical knowledge encoded implicitly in text or code (and mixed with implementation details), not as symbolically readable data
- Missing or inconsistent assumptions

Inspiration 1: Rubi by Albert D. Rich

`https://rulebasedintegration.org`

> *[Rubi] uses pattern matching to uniquely determine which of its over 6600 integration rules to apply to a given integrand*

> *Rubi dramatically out-performs other symbolic integrators, including Maple and Mathematica*

> *Certainly much of analysis including equation solving, expression simplification, differentiation, summation, limits, etc. can be automated using this paradigm*

Inspiration 2: current computer algebra systems (and how broken they are)

- Too zealous “simplification”
- Mathematical inconsistencies or bugs

A simple symbolic integral: $\int_1^2 x^a dx$

Mathematica:

```
In[5]:= Integrate[x ^ (-1) , {x, 1, 2}]

Out[5]= Log[2]


In[7]:= Integrate[x ^ a, {x, 1, 2}]
```

Out[7]= $\frac{-1+2^{1+a}}{1+a}$

```
In[8]:= Integrate[x ^ a, {x, 1, 2}] /. (a → -1)
```

··· Power: Infinite expression $\frac{1}{0}$ encountered.

··· Infinity : Indeterminate expression 0 ComplexInfinity encountered.

```
Out[8]= Indeterminate
```

A simple symbolic integral: $\int_1^2 x^a dx$

SymPy does the right thing:

```
>>> integrate(x**a, (x, 1, 2))
Piecewise((2**(a + 1)/(a + 1) - 1/(a + 1),
    (a > -oo) & (a < oo) & Ne(a, -1)), (log(2), True))

>>> integrate(x**a, (x, 1, 2)).subs(a, -1)
log(2)
```

A simple symbolic integral: $\int_1^2 x^a dx$

SymPy does the right thing:

```
>>> integrate(x**a, (x, 1, 2))
Piecewise((2**(a + 1)/(a + 1) - 1/(a + 1),
    (a > -oo) & (a < oo) & Ne(a, -1)), (log(2), True))

>>> integrate(x**a, (x, 1, 2)).subs(a, -1)
log(2)
```

Well, almost:

```
>>> integrate(x**a, (x, 1, 2)).subs(a, I)
Traceback (most recent call last):
  ...
TypeError: Invalid comparison of complex I
```

What is ${}_1F_1(-1, -1, 1)$?

Mathematica:

```
In[•]:= Hypergeometric1F1[n, m, 1] /. {m → -1, n → -1, x → 1}
Out[•]= 2

In[•]:= (Hypergeometric1F1[n, m, x] /. {m → n}) /. {n → -1, x → 1}
Out[•]= e
```

SymPy:

```
>>> simplify(hyper([n],[m],x).subs({m:-1, n:-1, x:1}))
2
>>> simplify(hyper([n],[m],x).subs(m, n)).subs({n:-1, x:1})
E
```

What is ${}_1F_1(-1, -1, 1)$?

http://fungrim.org/entry/dec042/

$${}_1F_1(-n, b, z) = \sum_{k=0}^{n} \frac{(-n)_k}{(b)_k} \frac{z^k}{k!}$$

Assumptions:
$n \in \mathbb{Z}_{\ge 0}$ and $b \in \mathbb{C}$ and not $(b \in \{0, -1, \ldots\}$ and $b > -n)$ and $z \in \mathbb{C}$

http://fungrim.org/entry/be533c/

$${}_1F_1(a, b, z) = e^z \, {}_1F_1(b - a, b, -z)$$

Assumptions: $a \in \mathbb{C}$ and $b \in \mathbb{C} \setminus \{0, -1, \ldots\}$ and $z \in \mathbb{C}$

Example: derivative of the modular $\lambda(\tau)$ function

```
In[91]:= N[D[ModularLambda[tau], tau] /. tau → 1/2 + 3 I, 10]
         ND[ModularLambda[tau], tau, 1/2+ 3 I, Scale → 10 ^ -16,
          WorkingPrecision→30]

Out[91]= -0.004056396698+5.237591×10^-6 i

Out[92]= -0.004056396698+5.237591×10^-6 i

In[93]:= N[D[ModularLambda[tau], tau] /. tau → 1/2 + I/3, 10]
         ND[ModularLambda[tau], tau, 1/2+ I/3, Scale → 10 ^ -16,
          WorkingPrecision→ 30]

Out[93]= 0.332090725+4.583245973 i

Out[94]= -0.74720413-10.31230344 i
```

Example: derivative of the modular $\lambda(\tau)$ function

```
In[91]:= N[D[ModularLambda[tau], tau] /. tau → 1/2 + 3 I, 10]
         ND[ModularLambda[tau], tau, 1/2+ 3 I, Scale → 10 ^ -16,
          WorkingPrecision→30]

Out[91]= -0.004056396698+5.237591×10^-6 i

Out[92]= -0.004056396698+5.237591×10^-6 i

In[93]:= N[D[ModularLambda[tau], tau] /. tau → 1/2 + I/3, 10]
         ND[ModularLambda[tau], tau, 1/2+ I/3, Scale → 10 ^ -16,
          WorkingPrecision→ 30]

Out[93]= 0.332090725+4.583245973 i

Out[94]= -0.74720413-10.31230344 i
```

Mathematica uses an elliptic integral to express $\lambda'(\tau)$. This is not valid everywhere because of branch cuts!

Fast and simple versus correct

R. Corless and D. Jeffrey, “Well… It Isn’t Quite That Simple”, ACM SIGSAM Bulletin, 1992:

> *The automatic exploration of conditions or alternative results requires considerable computational resources, and for the sake of speed there is an attraction to picking one ’obvious’ answer. [...] The difficulty is to balance efficiency against correctness.*

27 years later, what is the right balance?