# Common Lisp's Predilection for Mathematical Programming

Robert Smith
Clozure Associates
1368 Beacon Street, Suite 112
Brookline, MA 02446
rsmith@clozure.com

## ABSTRACT

Common Lisp is a towering language that supports a plethora of functionality useful for both scientific and mathematical programming. However—except for a few notable systems such as Axiom, Macsyma/Maxima, and ACL2—Lisp has not taken center stage for such kinds of programming tasks. We will analyze existing systems, including computer algebra systems, technical computing systems, and other programming languages, and their utility in scientific and mathematical programming. Such a discussion will form a foundation for comparative study.

Following that, we will expound on some features of Lisp that augment the expressiveness, simplicity, and utility of programs written in the language. In particular, we do so by way of three carefully selected pragmatic examples arising in fields ranging from the theory of special functions to numerical simulation.

First, we will show how a mixture of Common Lisp features allows for concise implementations of state-of-the-art algorithms for computing hypergeometric series to high precision. We will use this to evaluate Chudnovsky's formula,

$$\frac{C}{\pi} = \sum_{k=0}^{\infty} \frac{(6k)!(545140134k + 13591409)}{(3k)!(k!)^3} \left(-\frac{1}{640320^3}\right)^k$$

where

$$C = \frac{\sqrt{640320^3}}{12},$$

which is one of the most practical and fastest ways to compute the digits of $\pi$.

Next, we will show how one can easily implement algorithms for doing experimental mathematics, including so-called *number identification* algorithms. We will present an implementation of the PSLQ algorithm, a descendant of lattice reduction algorithms, which can be used to find a "short" integer vector $\vec{a}$ which solves

$$\vec{a} \cdot \vec{x} = 0$$

for a given real vector $\vec{x}$, or provides a lower bound on the length of $\vec{a}$ if it cannot be solved within given computational constraints. As an application, we show that if $\alpha$ is an algebraic number, then letting $x_i = \alpha^{i-1}$ will allow us to find the polynomial of which $\alpha$ is a root.

Finally, we will sketch a method for simulating arbitrary mechanical systems using the Lagrangian formalism of classical mechanics. We will focus on a step requiring us to solve a system of first-order differential equations derived from the Euler–Lagrange equations. To solve the system, we present Common Lisp code to *generate* efficient Runge–Kutta methods from declaratively specified Butcher tableaux. Different Butcher tableaux allow for the creation of different solvers with varying characteristics. An example demonstration of a generated Runge–Kutta method will be the display of a trajectory of a double pendulum, a widely known chaotic system.

## Categories and Subject Descriptors

B.2.4 [**Arithmetic and Logic Structures**]: High-Speed Arithmetic—*Algorithms*; D.3.4 [**Programming Languages**]: Processors—*Code Generation*; G.1.0 [**Mathematics of Computing**]: Numerical Analysis—*General*; G.1.10 [**Mathematics of Computing**]: Numerical Analysis—*Applications*

## General Terms

Algorithms

## Keywords

Common Lisp, mathematics, computer algebra, arbitrary precision arithmetic, hypergeometric series, pi, number identification, lattice reduction, experimental mathematics, numerical simulation, Lagrangian mechanics, numerical differential equations



*ILC '14* August 14–17 2014, Montreal, QC, Canada
ACM 978-1-4503-2931-6/14/08.
http://dx.doi.org/10.1145/2635648.2639484