О выборе языков программирования для модернизации системы эфемеридных расчетов в астрономии ЭРА

© В. А. Яковлев, К. А. Наумов, М. В. Васильев

ИПА РАН, г. Санкт-Петербург, Россия

Реферат

Цель работы — выбор языка или языков программирования для модернизации системы Эфемеридных Расчетов в Астрономии (ЭРА). Появление планов по модернизации связано с потенциальным отказом от поддержки 80-битных переменных в новых версиях базового для системы ЭРА языка Racket. Использование же 80-битных переменных является обязательным условием для обеспечения необходимой точности эфемеридных расчетов. Критериями выбора языков были распространенность и функционал для решения задач эфемеридной астрономии, возможности по ускорению и повышению точности эфемеридных расчетов, кроссплатформенность в рамках ОС Windows и Linux, сопровождение и модернизация системы ЭРА без привлечения высококвалифицированных программистов.

Исходя из перечисленных критериев для тестирования были выбраны языки программирования C++, Fortran и Python, а также библиотеки для вычислений с повышенной точностью MPFR, QD и mpmath. Язык Racket также участвовал в рассмотрении как основной инструмент, использовавшийся для программирования последней версии системы ЭРА. В силу очевидности выбора языков по всем критериям кроме быстродействия основные тесты были направлены на сравнение скорости выполнения базовых арифметических операций и операций ввода/вывода. Тестирование проводилось с применением двух компьютеров на базе процессоров Intel® Core™ i5 и i7 под управлением ОС Linux.

Тесты показали, что время выполнения базовых арифметических операций с использованием Fortran и C++ практически совпадает. Операции ввода/вывода быстрее всего работают на C++. Программы на Racket проигрывают в несколько раз как программам на Fortran, так и на C++. Библиотеки для вычислений с повышенной точностью также быстрее работают в тестах на C++ и Fortran. Python уступает по показателям быстродействия C++, Fortran и Racket.

По результатам тестирования для работ по модернизации системы ЭРА выбраны языки программирования Fortran и C++. Кроме сохранения необходимой точности эфемеридных расчетов использование этих языков позволит заметно сократить время вычислений, а также создаст задел для повышения их точности.

Ключевые слова: Эфемерида, программный комплекс, язык программирования, арифметические операции с повышенной точностью, операции ввода/вывода.

Контакты для связи: Яковлев Владимир Александрович (va.yakovlev@iaaras.ru).

Для цитирования: Яковлев В. А., Наумов К. А., Васильев М. В. О выборе языков программирования для модернизации системы эфемеридных расчетов в астрономии ЭРА // Труды ИПА РАН. 2026. Вып. 76. С. 32–38.
https://doi.org/10.32876/ApplAstron.76.32-38

On the Choice of Programming Languages for Modernizing the ERA System of Ephemeris Research in Astronomy

V. A. Yakovlev, K. A. Naumov, M. V. Vasilyev

Institute of Applied Astronomy of the Russian Academy of Sciences, Saint Petersburg, Russia

Abstract

The objective of this study was to select a programming language or languages for upgrading the Ephemeris Researh in Astronomy (ERA) system. The modernization plans were prompted by the removal of support for 80-bit variables in new versions of Racket, the ERA system's core language. The use of 80-bit variables is essential to ensure the required accuracy of ephemeris calculations. The criteria for selecting the languages included their widespread use and functionality for solving ephemeris astronomy problems, the ability to accelerate and improve the accuracy of ephemeris calculations, cross-platform compatibility, and the ability to maintain and upgrade the ERA system without the need for highly qualified programmers.

Based on these criteria, the programming languages C++, Fortran, and Python were selected for testing, as well as the high-precision calculation libraries MPFR, QD, and mpmath. Racket was also considered as the primary programming tool for the latest version of the ERA system. Since the choice of languages was obvious based on all criteria except performance, the main tests focused on comparing the speed of basic arithmetic operations and input/output operations. Testing was conducted using two computers with Intel® Core™ i5 and i7 processors running Linux.

Tests showed that the execution time for basic arithmetic operations using Fortran and C++ is virtually identical. Input/output operations are fastest in C++. Racket programs are several times slower than both Fortran and C++ programs. Libraries for high-precision calculations also perform faster in C++ and Fortran tests. Python tests perform worse than C++, Fortran, and Racket programs.

Based on the testing results, Fortran and C++ were selected for the ERA system upgrade. In addition to maintaining the required accuracy of ephemeris calculations, using these languages will significantly reduce computation time and create the foundation for improving their accuracy.

Keywords: Ephemeris, software package, programming language, high-precision arithmetic operations, input/output operations.

Contacts: Vladimir A. Yakovlev (va.yakovlev@iaaras.ru).

Для цитирования: Yakovlev V. A., Naumov K. A., Vasilyev M. V. On the choice of programming languages for modernizing the ERA system of ephemeris research in astronomy // Transactions of IAA RAS. 2026. Vol. 76. P. 32–38.
https://doi.org/10.32876/ApplAstron.76.32-38

Введение

В декабре 1985 г. был представлен первый экспериментальный вариант системы ЭРА и объектно-ориентированного языка СЛОН (Новиков, 1990; Агамирзян и др., 1986) на ЭВМ БЭСМ-6. Первая версия системы ЭРА (ERA-5) и языка СЛОН, которые были готовы к применению в практической научной работе, появились только через 10 лет (Krasinsky, Vasilyev, 1997). Система ERA-5 использовала для расчетов 32-, 64- и 80-битные типы переменных с плавающей точкой языка программирования Pascal. Этот набор типов вещественных чисел обеспечивал и обеспечивает до настоящего времени необходимую точность вычислений для решения задач эфемеридной астрономии.

Отметим, что для проведения большинства вычислений достаточно 64-битных переменных с плавающей точкой. Они обеспечивают 15–16 верных десятичных знаков для представления современных наблюдений и построения моделей, адекватных этим наблюдениям по точности. 80-битные типы переменных (19–20 верных десятичных знаков) используются для представления времени наблюдений и интегрирования дифференциальных уравнений орбитального и вращательного движения тел Солнечной системы (Pitjeva et al., 2022).

Точность фиксации времени высокоточных наблюдений, таких как светолокационные наблюдения ИСЗ или Луны, ГНСС данных, должна быть не хуже, а желательно лучше, чем 1 мкс. В случае представления момента наблюдений в принятом в астрономии формате юлианской даты применение 80-битных переменных не обеспечивает нужной субмикросекундной точности. Обычным способом решения данной проблемы является переход к модифицированной юлианской дате или другим ее производным.

Применение 80-битных переменных является обязательным при интегрировании дифференциальных уравнений орбитально-вращательного движения Луны. Это необходимо для уменьшения вычислительных ошибок при построении модели движения Луны до миллиметрового уровня, необходимого при обработке современных лунных светолокационных наблюдений.

В 2015 г. система ЭРА была переработана (Pavlov et al., 2015). В ходе переработки исходный программный код был переписан с Pascal на Racket (для запуска программ, написанных на языке СЛОН, управления данными и значительной части вычислений) и C (библиотеки для вычислений). До последнего времени все версии Racket поддерживали использовавшиеся до переработки системы ЭРА типы переменных с плавающей точкой, однако, в 2021 г. разработчики Racket поменяли стандартную среду выполнения на среду Chez Scheme, в которой нет поддержки 80-битных переменных (extflonum) (Racket, 2019; Racket, 2025). Чтобы вернуть поддержку 80-битных переменных в Racket версий 8.0 и старше необходимо собирать BC-версию языка из исходных кодов. Однако эта версия не будет поддерживать последние обновления Racket и, по-видимому, целесообразнее использовать старые версии языка, например 7.5.

Разработчики программного комплекса ЭРА были поставлены перед выбором: либо продолжать использовать одну из последних версий Racket с поддержкой 80-битных переменных с соответствующими рисками, либо перейти на языки программирования, которые будут поддерживать требуемую точность вычислений в актуальных версиях в обозримом будущем. С учетом опыта и длительности эксплуатации системы ЭРА, опыта эксплуатации других программных продуктов в области эфемеридной астрономии можно выделить лишь несколько подходящих языков: Python, Fortran, C/C++. Кроме того, для обеспечения в будущем возможности высокоточных вычислений с плавающей точкой (Bailey, Borwein 2015) и с большей чем 80 бит разрядностью, выбор языка программирования производился с учетом доступных библиотек для работы с 128-, 160- и 256-битными переменными.

Далее будут рассмотрены результаты сравнительного тестирования языков программирования с использованием встроенных типов веществен-

ных переменных и различных библиотек, реализующих расширенную точность вычислений с плавающей точкой.

Аппаратные и программные средства

Тестирование проводилось на двух компьютерах с ОС Linux:

1) компьютер 1 (процессор Intel® Core™ i5-660; 8 ГБ ОЗУ);

2) компьютер 2 (процессор: Intel® Core™ i7-12700K; 64 ГБ ОЗУ).

В тестировании применялись языки программирования Racket (версии 7.5 и 8.18 BC), C++ (gcc версий 11.4.0 и 15.2.1), Fortran (gfortran версий 11.4.0 и 15.2.1) и Python (версии 3.10.12 и 3.13.9). Версии компиляторов, указанные первыми, были установлены на компьютере 1, вторыми — на компьютере 2. Кроме того, программы на языках C++ и Fortran компилировались в двух версиях: с отключенной оптимизацией и с оптимизациями o1 или o2.

Для рассмотрения возможности высокоточных вычислений с плавающей точкой с разрядностью 80 бит и больше были использованы такие открытые решения, как:

1) библиотеки bigfloat — для языка Racket, MPFR C++ — для C++ и gmpy — для Python, которые базируются на библиотеке MPFR и обеспечивают контролируемую точность вычислений (MPFR);

2) библиотека QD — для языка C++, реализующая double-double (128 бит) и quad-double (256 бит) арифметику (Hida et al., 2001);

3) библиотека mpmath — для языка Python, которая позволяет работать с переменными контролируемой точности и содержит большой набор математических функций для работы с ними (MPMATH).

Для языка Fortran аналогичные библиотеки не использовались, тестировались только встроенные в язык переменные, включая 128-битные, и операции с ними.

Тестирование скорости выполнения арифметических операций и операций ввода/вывода проводилось на основе базовых для системы операций:

1) умножение;

2) деление;

3) чтение из текстового файла;

4) запись в базу данных (БД) и чтение из БД.

Скорости операций умножения, сложения и вычитания с некоторым допущением сопоставимы, поэтому эти действия рассмотрены как идентичные. Для тестирования были сгенерированы два массива из 10^8 равномерно распределенных в диапазоне [0;1] случайных вещественных чисел. Для генерации случайных чисел использовалась библиотека rband (Hamburg, 2012).

Для тестирования работы с БД был выбран SQlite, применяемый в системе ЭРА. Использование различных типов данных для разных вариантов переменных обусловлено отсутствием их прямой поддержки в SQlite. Например, значения, превышающие по размеру 64 бита не поддерживаются библиотекой SQlite, поэтому переменные размером больше 64 бита можно записать только в текстовом (тип TEXT) или бинарном (тип BLOB) форматах. Так же, поскольку библиотеки mpmath и MPFR базируются на своих собственных форматах переменных, их можно представить в SQlite только в текстовом формате (тип TEXT). 64-битные переменные по умолчанию поддерживаются в SQlite в качестве типа REAL, хотя их также можно представить в виде типов TEXT и BLOB. При тестировании для каждого из этих трех вариантов был выбран тип представления данных с наибольшей скоростью записи. В тестах с 64-битными переменными для записи в БД использован тип REAL, в тестах с подключением библиотек MPFR и mpmath использован тип TEXT, в остальных тестах — BLOB (бинарная запись). Выбранные типы обеспечивают высокую производительность при операциях с БД и гарантируют корректное представление исходных данных.

Тесты и средства их запуска были реализованы на языках программирования Python, Fortran, C/C++ и Racket. Исходный код всех тестов доступен по запросу на электронную почту, ниже приведен тест на умножение случайных чисел на языке Python:

```
bites = [64, 80, 128, 160, 256]
for bit in bites:
    gmpy2.get_context().precision = bit
    print("Calculations with", bit, "bits")
    data1 = load_data_2('data1.txt')
    data2 = load_data_2('data2.txt')
    data3 = load_data_2('data3.txt')
    def mult():
        for i in range(len(data1)):
            data3[i] = data1[i]*data2[i]

    timer_1 = Timer(partial(mult)).timeit(number=1)
```

Результаты тестирования

Результаты тестов на компьютере 1 приведены в табл. 1–5, на компьютере 2 — в табл. 6–10. Отметка «*» означает, что тест проводился с использованием библиотеки MPFR; «**» — библиотеки QD; «***» — библиотеки mpmath. В случае стандартных для языка программирования типов переменных тесты с использованием библиотек не проводились. Прочерк «–» означает, что тест не был завершен за разумное время. Пустое место в ячейке таблицы означает, что отсутствовала или не была найдена подходящая библиотека для проведения данного теста. Взаимодействие с БД на языке Fortran предполагает прямые вызовы методов на C, поэтому эти тесты неинформативны и убраны из рассмотрения. Поскольку большая оптимизация не всегда оказывается наилучшей по скорости выполнения базовых операций, то для языков Fortran и C++ представлены результаты как с выключенной (--o0), так и с наиболее производительной оптимизациями (из --o2 и --o1).

Таблица 1

Длительность арифметических операций умножения на компьютере 1 (в секундах)

Язык/тип переменных	64-битные	80-битные	128-битные	160-битные	256-битные
C++ --o0	0.098	0.138	1.61*/0.55**	1.72*	1.87*/4.13**
Fortran --o0	0.048	0.072	0.38		
C++ --o2	0.030	0.061	1.14*/0.084**	1.27*	1.41*/0.75**
Fortran --o2	0.030	0.062	0.35		
Racket	0.100	0.130	27.10*	26.70*	28.70*
Python	1.640	2.59*/–***	2.71*/–***	2.80*/–***	2.95*/–***

Таблица 2

Длительность арифметических операций деления на компьютере 1 (в секундах)

Язык/тип переменных	64-битные	80-битные	128-битные	160-битные	256-битные
C++ --o0	0.102	0.152	2.31*/1.154**	2.82*	3.05*/7.34**
Fortran --o0	0.068	0.085	0.872		
C++ --o2	0.065	0.078	1.72*/0.164**	2.29*	2.49*/3.52**
Fortran --o2	0.064	0.078	0.840		
Racket	0.105	0.132	27.33*	28.55*	29.22*
Python	1.646	2.907*/–***	3.364*/–***	4.01*/–***	4.17*/–***

Таблица 3

Длительность чтения из текстового файла на компьютере 1 (в секундах)

Язык/тип переменных	64-битные	80-битные	128-битные	160-битные	256-битные
C++ --o0	1.98	2.51	8.73*/2,59**	9.03*	9.47*/2.84**
Fortran --o0	12.5	12.9	13.5		
C++ --o2	1.78	2.34	7.58*/1.99**	7.73*	8.35*/2.22**
Fortran --o2	12.3	12.8	13.5		
Racket	57.0	57.9	96.1*	96.2*	98.6*
Python	7.6	15.0*/–***	15.7*/–***	15.9*/–***	16.3*/–***

Таблица 4

Длительность записи в БД на компьютере 1 (в секундах)

Язык/тип переменных	64-битные	80-битные	128-битные	160-битные	256-битные
C++ --o0	18.9	20.2	84.5*/21.2**	86.2*	91.2*/22.6**
C++ --o2	18.5	19.5	79.4*/ 21.4**	80.5*	85.2*/22.2**
Racket	354.4	368.6	607.5*	613.2*	634.4*
Python	41.1	167.4*/–***	169.5*/–***	171.8*/–***	179.3*/–***

Таблица 5

Длительность чтения из БД на компьютере 1 (в секундах)

Язык/тип переменных	64-битные	80-битные	128-битные	160-битные	256-битные
C++ --o0	1.65	2.11	14.4*/2.74**	15.3*	18.3*/3.05**
C++ --o2	1.46	1.87	11.7*/2.25**	12.7*	15.6*/2.47**
Racket	21.5	28.9	90.7*	93.6*	105.1*
Python	5.23	18.5*/–***	20.7*/–***	22.0*/–***	25.3*/–***

Таблица 6

Длительность арифметических операций умножения на компьютере 2 (в секундах)

Язык/тип переменных	64-битные	80-битные	128-битные	160-битные	256-битные
C++ --o0	0.065	0.095	0.70*/0.15**	0.68*	0.79*/1,23**
Fortran --o0	0.011	0.029	0.12		
C++ --o2	0.009	0.022	0.47*/0.024**	0.46*	0.54*/0.20**
Fortran --o1	0.009	0.022	0.11		
Racket	0.022	0.053	6.71*	6.61*	6.94*
Python	0.26	0.64*/3.68***	0.63*/2.95***	0.65*/2.97***	0.73*/2.96***

Таблица 7

Длительность арифметических операций деления на компьютере 2 (в секундах)

Язык / тип переменных	64-битные	80-битные	128-битные	160-битные	256-битные
C++ --o0	0.059	0.094	0.853*/0.318**	0.98*	1.03*/1.28**
Fortran --o0	0.012	0.029	0.186		
C++ --o2	0.010	0.022	0.618*/0.039**	0.74*	0.80*/1.23**
Fortran –o1	0.010	0.023	0.171		
Racket	0.022	0.052	6.950*	7.06*	7.18*
Python	0.278	0.731*/5.281***	0.852*/5.540***	0.93*/5.50***	0.99*/6.05***

Таблица 8

Длительность чтения из текстового файла на компьютере 2 (в секундах)

Язык/тип переменных	64-битные	80-битные	128-битные	160-битные	256-битные
C++ --o0	0.75	0.89	3.40*/0.88**	3.44*	3.60*/0.99**
Fortran --o0	4.50	4.57	4.64		
C++ --o2	0.72	0.84	2.52*/0.81**	2.55*	2.69*/–**
Fortran --o2	4.48	4.52	4.60		
Racket	14.8	15.2	21.9*	21.9*	22.6*
Python	2.40	4.29*/19.6***	4.46*/19.6***	4.38*/19.6***	4.56*/19.8***

Таблица 9

Длительность записи в БД на компьютере 2 (в секундах)

Язык/тип переменных	64-битные	80-битные	128-битные	160-битные	256-битные
C++ --o0	3.45	3.93	14.9*/4.18**	15.3*	16.6*/4.52**
C++ --o2	3.41	3.86	13.8*/4.17**	14.1*	15.2*/4.51**
Racket	55.8	56.1	83.9*	85.1*	89.4*
Python	7.47	21.6*/22.1***	21.8*/24.9***	22.2*/26.1***	23.6*/29.7***

Таблица10

Длительность чтения из БД на компьютере 2 (в секундах)

Язык/тип переменных	64-битные	80-битные	128-битные	160-битные	256-битные
C++ --o0	0.46	0.68	4.24*/0.81**	4.55*	5.57*/0.97**
C++ --o2	0.44	0.65	3.89*/0.78**	4.20*	5.25*/0.85**
Racket	6.5	8.1	21.4*	22.2*	25.9*
Python	1.96	5.68*/20.7***	6.18*/20.8***	6.46*/20.8***	7.67*/20.9***

Анализ результатов тестирования

Для стандартных 64- и 80-битных переменных тесты показывают, что арифметические операции выполняются быстрее всего при использовании языков программирования Fortran и C/C++. Оба языка показывают в этом случае практически одинаковые результаты. Racket проигрывает им примерно в два раза, а Python — более чем на порядок.

Для всех типов переменных, 64-, 80-, 128- 160- и 256-битных, операции ввода/вывода текстовых данных и операции с БД SQLite выполняются гораздо быстрее (см. табл. 4–5 и 9–10) при программировании на языке C/C++ по сравнению с остальными рассмотренными вариантами.

Для 160- и 256-битных переменных тесты показывают, что арифметические операции выполняются быстрее всего при использовании языка программирования C/C++. В случае 128-битных переменных быстрее всего работают тесты на Fortran и с использованием библиотеки QD в C/C++. С учетом потенциальных требований к точности эфемеридных расчетов в будущем, ориентация на библиотеку QD в C/C++, язык Fortran и 128-битные переменные — выбор разумный. Выбор конкретного варианта применения 128-битных переменных в системе ЭРА потребует дополнительного исследования.

Решение double-double (128 бит), реализованное в библиотеке QD, доказывает свою эффективность в тестах для языка C++ и может рассматриваться заменой 80-битных переменных при отсутствии их аппаратной поддержки.

Заключение

Тестирование арифметических операций и операций ввода/вывода показало, что вычислительные модули системы ЭРА, написанные на языке Racket, целесообразнее переписать с помощью языков Fortran и C/C++. Для программирования операций ввода/вывода разумнее использовать язык C/C++. Модернизация системы ЭРА

в этом направлении может уменьшить время эфемеридных расчетов в несколько раз. Отметим, что в настоящее время одна итерация при вычислении эфемерид EPM длится около 7 часов.

Отказ от использования Racket и переход на языки программирования Fortran и C/C++ упростит сопровождение и модернизацию системы ЭРА, а также сохранит кроссплатформенность программного комплекса в рамках ОС Windows и Linux.

Возможность использования 128-битных вещественных переменных языка Fortran или переменных типа «double-double» библиотеки QD в C/C++ представляется полезным для будущих версий системы ЭРА при появлении задач, требующих повышения точности эфемеридных расчетов.

Таким образом, по результатам тестирования для работ по модернизации вычислительных модулей системы ЭРА выбраны языки программирования Fortran и C++. Кроме сохранения необходимой точности эфемеридных расчетов использование этих языков позволит заметно сократить время вычислений, а также создаст задел для реализации вычислений с арифметикой, расширенной вплоть до 160 бит.

Еще одним направлением будущих работ по ускорению эфемеридных расчетов в системе ЭРА является применение параллельных вычислений с использованием языков программирования Fortran и C++, в том числе на базе графических процессоров GPU (graphics processing unit). Реализация параллельных вычислений при численном интегрировании уравнений движения представляется непростой задачей и требует особого рассмотрения на следующем этапе модернизации системы ЭРА.

Благодарность

Авторы благодарны Сергею Леонидовичу Курдубову за полезные замечания, которые позволили улучшить содержание работы, а также глубже раскрыть тему исследования.

Литература

Агамирзян И. Р., Красинский Г. А., Новиков Ф. А., Скрипниченко В. И. Автоматизированная система эфемеридных расчетов астрономии // Сборник «Всесоюзная школа — семинар «Динамика механических систем». Тезисы докладов». Томск: 1986.

Krasinsky G. A., Vasilyev M. V. Era: Knowledge base for ephemeris and dynamical astronomy // Proceedings of IAU Colloquium 165, In: I. M. Wytrzyszczak, J. H. Lieske, R. A. Feldman, Dynamics and Astrometry of Natural and Artificial Celestial Bodies, Kluwer Academic Publishers, Dordrecht. 1997. P. 239–244.

Новиков Ф. А. Архитектура системы ЭРА – Табличный подход к обработке данных // Препринт ИПА РАН 1990. № 16. С. 1–31.

Pavlov D. A., Skripnichenko V. I. Rework of the ERA software system: ERA-8 // Proceedings of the Journées 2014 “Systèmes de Référence Spatio-Temporels”, ed. by Z. Malkin and N. Capitaine, Pulkovo observatory. 2015. P. 243–246.

Pitjeva E. V., Pavlov D. A., Aksim D., Kan M. Planetary and lunar ephemeris EPM2021 and its significance for Solar system research // Proceedings of the International Astronomical Union. Vol. 15. Symposium S364. 2022. P. 220–225.

Bailey D. H., Borwein J. M. High-precision arithmetic in mathematical physics // Mathematics 2015. Vol. 3. P. 337–367.

Hamburg M. Understanding Intel's ivy bridge random number generator. 2012 [Электронный ресурс]. URL: https://www.electronicdesign.com/resources/article/21796238/understanding-intels-ivy-bridge-random-number-generator (accessed: 19.01.2026).

Hida Y., Li X. S., Bailey D. H. Algorithms for quad-double precision floating point arithmetic. Proceedings 15th IEEE Symposium on Computer Arithmetic. ARITH-15 2001, Vail, CO, USA, 2001. P. 155–162, doi: 10.1109/ARITH.2001.930115.

MPFR. 2025 [Электронный ресурс]. URL: https://www.mpfr.org/ (дата обращения 02.02.2026).

MPMATH. 2025 [Электронный ресурс]. URL: https://mpmath.org/ (дата обращения 02.02.2026).

Racket. 2019 [Электронный ресурс]. URL:https://blog.racket-lang.org/2019/01/racket-on-chez-status.html

Racket. 2025 [Электронный ресурс]. URL: https://github.com/racket/racket?path=racket/src (дата обращения 02.02.2026).