

Расширение транслятора
предметно-ориентированного
языка Landau арифметикой
сдвоенных чисел двойной
точности (double-double)

Цели работы

- Ознакомиться с принципами работы source-to-source компиляторов и ЯОП.
- Изучить основы Racket и написанного на нем языка Landau с системой автоматического дифференцирования.
- Добавить в транслятор Landau поддержку арифметики double-double.

Языково-ориентированное программирование

- Языково-ориентированное программирование – это подход к решению задач с узкой специализацией. Подход заключается в создании нового языка и написании программы на нём.
- Создаваемые языки называются **предметно-ориентированными языками** (DSL).
- Racket подходит для ЯОП из-за своей **системы макросов**. Они удобны для трансляции кода на DSL в Racket или другой язык.

Преимущества Racket

- Данный язык обладает проработанной системой макросов, что важно для определения синтаксиса языка.
- Racket позволяет компилировать текст в специальные *синтаксические объекты* при написании макросов.
- Синтаксические объекты хранят не только строку макроса, но и связанные с ней метаданные (местоположение в коде, синтаксический контекст и дополнительные свойства).
- Сохранение лексического контекста сильно упрощает написание предметно-ориентированных языков, например уменьшается возможность коллизий названий объектов.

Landau

Landau – язык для динамических систем с автоматическим дифференцированием (AD).

В языке есть функции, циклы, условия, вещественные числа, массивы.

Landau достаточно полон для того, чтобы выразить и продифференцировать уравнение любой сложности достаточно оптимально.

```
#lang landau

parameter[6] initial

real[6 + 36 + 6] xdot (real[6 + 36 + 6] x, real GM)
{
  real[36] state_derivatives_initial
  state_derivatives_initial[0 : 36] = x[6 : 6 + 36]

  real[6] state_derivatives_gm
  state_derivatives_gm[0 : 6] = x[6 + 36 : 6 + 36 + 6]

  real[6] state
  state[ : ] = x[0 : 6]

  state[ : ] ' initial[ : ] = state_derivatives_initial[0 : 36]
  state[ : ] ' GM = state_derivatives_gm[0 : 6]

  real[6] state_dot

  state_dot[ : 3] = state[3 : 6]
  xdot[ : 3] = state_dot[0 : 3]

  real dist2
  real dist3inv

  dist2 = sqr(state[0]) + sqr(state[1]) + sqr(state[2])
  dist3inv = 1 / (dist2 * sqrt(dist2))

  state_dot[3 : 6] = GM * (-state[0 : 3]) * dist3inv
  xdot[3 : 6] = state_dot[3 : 6]

  xdot[6 : 6 + 36] = state_dot[ : ] ' initial[ : ]
  xdot[6 + 36 : 6 + 36 + 6] = state_dot[ : ] ' GM
}
```

Символьное дифференцирование

- Символьное дифференцирование работает путем разделения сложного выражения на несколько меньших выражений используя правила дифференцирования.
- Пример:

$$f(x) = x^3 + 3 \quad \Rightarrow \quad \frac{d}{dx}f(x) = \frac{d}{dx}x^3 + \frac{d}{dx}3 \quad \Rightarrow \quad \frac{d}{dx}f(x) = 3x^2 + 0 = 3x^2$$

Упрощение выражения может привести к экспоненциальной сложности подсчета производной.

Численное дифференцирование

$$f'(x) = \lim_{\epsilon \rightarrow 0} \frac{f(x+\epsilon) - f(x)}{\epsilon}$$

- Если ϵ слишком маленькое, то появляются погрешности округления.
- Если ϵ слишком большое, то появляются ошибки аппроксимации.

Автоматическое дифференцирование. Двойные числа.

- Двойное число – пара чисел $\langle x, x' \rangle$.
- Арифметика двойных чисел:

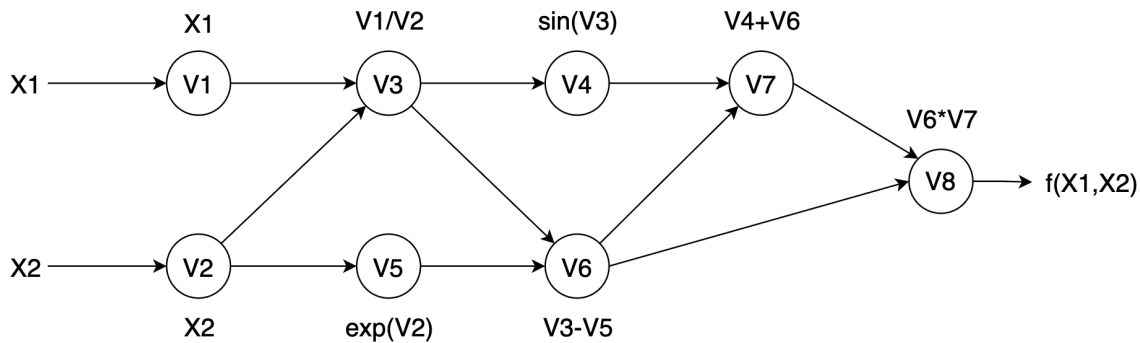
$$\langle u, u' \rangle \pm \langle v, v' \rangle = \langle u \pm v, u' \pm v' \rangle$$

$$\langle u, u' \rangle \langle v, v' \rangle = \langle uv, u'v + uv' \rangle$$

$$\frac{\langle u, u' \rangle}{\langle v, v' \rangle} = \left\langle \frac{u}{v}, \frac{u'v - uv'}{v^2} \right\rangle, v \neq 0$$

Автоматическое дифференцирование.

- $f(x_1, x_2) = \left[\sin\left(\frac{x_1}{x_2}\right) + \frac{x_1}{x_2} - e^{x_2} \right] * \left[\frac{x_1}{x_2} - e^{x_2} \right]$



$V_1 = X_1 = 1.5$	$V'_1 = 1$
$V_2 = X_2 = 0.5$	$V'_2 = 0$
$V_3 = V_1 / V_2 = 3$	$V'_3 = \frac{V_2 V'_1 - V_1 V'_2}{V_2^2} = 2.0$
$V_4 = \sin(V_3) = 0.141$	$V'_4 = \cos(V_3) * V'_3 = -1.98$
$V_5 = \exp(V_2) = 1.649$	$V'_5 = V_3 * V'_2 = 0$
$V_6 = V_3 - V_5 = 1.351$	$V'_6 = V'_3 - V'_5 = 2.0$
$V_7 = V_4 + V_6 = 1.492$	$V'_7 = V'_4 + V'_6 = 0.02$
$V_8 = V_6 * V_7 = 2.017$	$V'_8 = V'_7 V_6 + V'_6 V_7 = 3.012$
$f(x_1, x_2) = V_8 = 2.017$	$\frac{df}{dx_1} = V'_8 = 3.012$

Преимущества языка Landau

- Позволяет лёгкую генерацию эффективного C кода
- Освобождает пользователя от работы с C-указателями
- Позволят легко записывать математические уравнения
- Наличие синтаксиса для производных

Компоненты реализации Landau

- Транслятор Landau написан на Racket и состоит из двух основных частей:
 - Синтаксический анализатор
 - Разворачивание циклов в последовательности присваиваний.
 - Отслеживание зависимостей переменных и их производных.
 - Проверка программы на корректность.
 - Генератор кода
 - Генерация кода ANSI C и Racket.

Синтаксический анализатор

- Получает на вход список токенов (лексем) и генерирует из них дерево абстрактного синтаксиса – ориентированное дерево, в котором внутренние вершины сопоставлены с операторами языка программирования, а листья — с соответствующими операндами.
- На данном этапе обрабатывается большинство возможных ошибок в коде.

Генератор кода

- Получает на вход AST из синтаксических объектов
- Использует макросы Racket для трансформирования литералов исходного кода без его выполнения
- Узлы дерева преобразуются в соответствии с грамматикой целевого языка (завести необходимые переменные, выставить соответствующие идентификаторы и.т.д.)
- В итоге код Landau транслируется в код на соответствующем языке (Racket или ANSI C)

Double-double

- Число double-double выражается как сумма двух компонент double (большого и маленького числа)
$$x = x_l + x_h.$$
- Арифметика double-double имеет программную реализацию.

Algorithm 14.1 Dekker's algorithm for adding two double-word numbers (x_h, x_ℓ) and (y_h, y_ℓ) [108]. We assume radix 2.

```
if  $|x_h| \geq |y_h|$  then
   $(r_h, r_\ell) \leftarrow \text{Fast2Sum}(x_h, y_h)$ 
   $s \leftarrow \text{RN}(\text{RN}(r_\ell + y_\ell) + x_\ell)$ 
else
   $(r_h, r_\ell) \leftarrow \text{Fast2Sum}(y_h, x_h)$ 
   $s \leftarrow \text{RN}(\text{RN}(r_\ell + x_\ell) + y_\ell)$ 
end if
 $(t_h, t_\ell) \leftarrow \text{Fast2Sum}(r_h, s)$ 
return  $(t_h, t_\ell)$ 
```

Сложение double-double

Algorithm 4.3 The Fast2Sum algorithm [108].

```
 $s \leftarrow \text{RN}(a + b)$ 
 $z \leftarrow \text{RN}(s - a)$ 
 $t \leftarrow \text{RN}(b - z)$ 
```

Сумма двух double с
подсчетом ошибки ($a > b$)

Интеграция double-double.

Модуль генерации C кода

- Написаны новые функции и ветвления, которые генерируют ANSI C код.
- К примеру функция для вычитания. В случае double-double бэкенда генерирует C-функцию dd_sub, а в случае double и long double бэкендов генерирует "-".

```
(define (cr- x y) (match (target-real-implementation TARGET)
  ('double-double (format "dd_sub(~a , ~a)" x y))
  ('long-double (format "(~a - ~a)" x y))
  ('double (format "(~a - ~a)" x y))
  ))
```


Пример генерации кода

Landau код

```
#lang landau
```

```
real[3] test_function_1(real[3] a, real b)
{
    real c = b + 0.5 - a[0]
    test_function_1[:] = a[:] + b
}
```



Сгенерированный C код с double бэкендом

```
int test_function_1(double *restrict test_function_111, double *restrict a12, double b13) {
    double c14;
    c14 = ((b13 + 0.5e0) - a12[0]);
    {
        for (int slice_idx = 0; slice_idx < 3; slice_idx++) {
            test_function_111[(0 + slice_idx)] = (a12[(0 + slice_idx)] + b13);
        }
    }
    return 0;
}
```

Сгенерированный C код с double-double бэкендом

```
int test_function_1(dd *restrict test_function_111, dd *restrict a12, dd b13) {
    dd c14;
    c14 = dd_sub(dd_add_dd_d(b13 , 0.5e0) , a12[0]);
    {
        for (int slice_idx = 0; slice_idx < 3; slice_idx++) {
            test_function_111[(0 + slice_idx)] = dd_add(a12[(0 + slice_idx)] , b13);
        }
    }
    return 0;
}
```

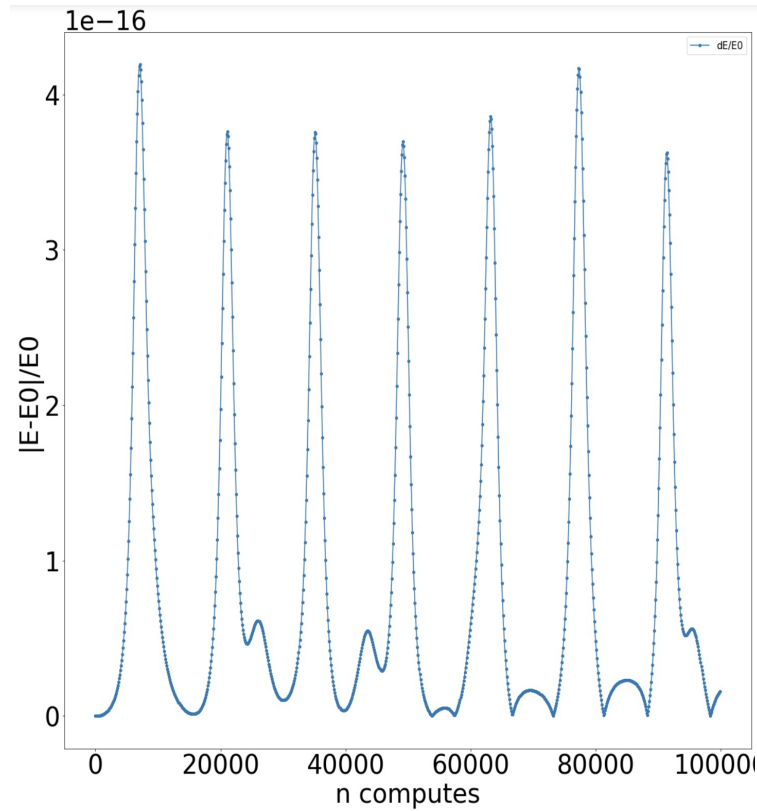
Проверка работоспособности сгенерированного ANSI C кода с помощью Landau на примере задачи N тел

- На языке Landau была написана функция для расчёта состояния системы N тел в следующий момент времени.
- Затем был сгенерирован ANSI C код.
- Из библиотеки QD были выбраны и переписаны на ANSI-C нужные функции для работы с double-double.
- Из этого кода была сгенерирована статическая библиотека и интегрирована в проект с задачей N тел.
- Результаты работы программы интегрирования с функциями из Landau сходятся с исходной программой.

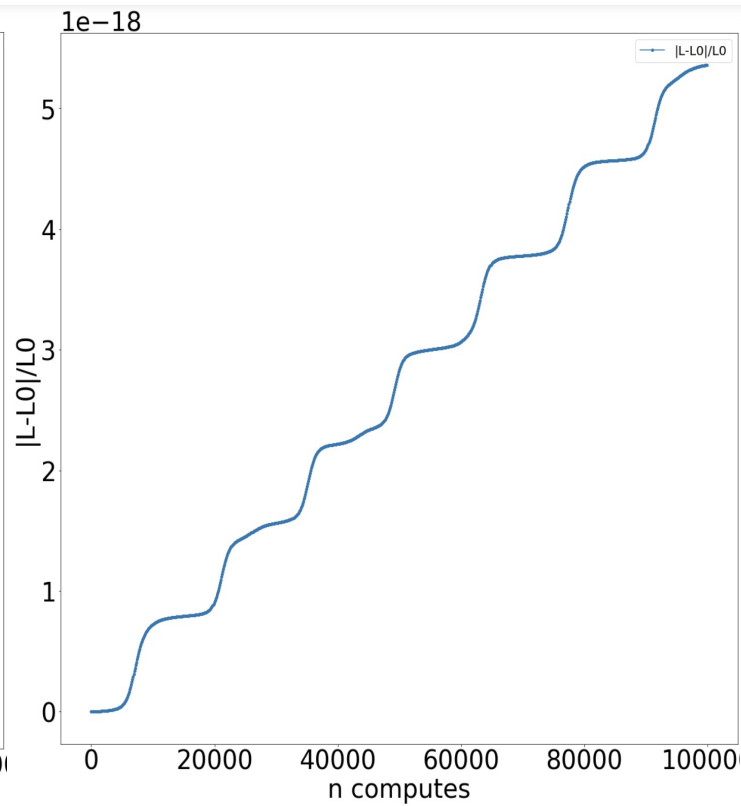
Выводы

- В Landaу был успешно интегрирован тип данных double-double.
- Была написана функция на Landaу и проверена ее корректность на примере задачи N тел.

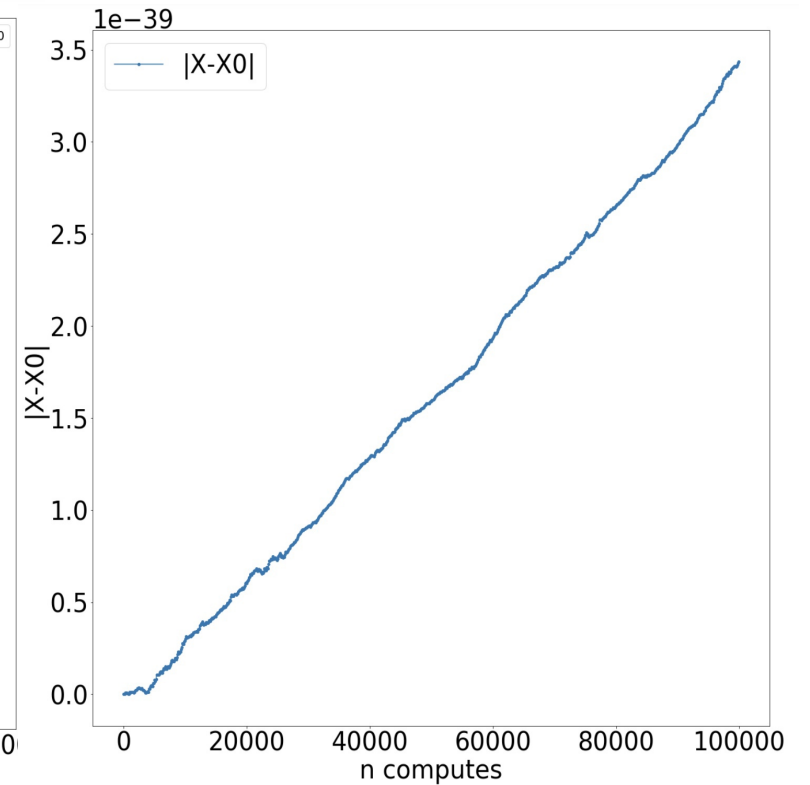
Результаты работы модифицированной программы



Относительное
изменение энергии



Относительное
изменение импульса



Изменение барицентра