Разработка алгоритма и программного модуля для решения внешней задачи Лэмба на системах с массивным параллелизмом.

Устинов Иван, 4 курс

Научный руководитель: А. В. Вершинин

Цель работы:

- Разработать программу, моделирующую распространение упругих волн в упругих средах.
- Провести численное моделирование.
- Оценить качество полученного результата.

Постановка задачи Лэмба:

Исследуется плоская задача Лэмба о действии сосредоточенной нагрузки, меняющейся со временем, на поверхность однородной изотропной упругой среды.

Параметры Ламэ среды **λ** и **μ**, плотность **ρ**.

Так же могут присутствовать дополнительные условия, например среда может быть вязкой или неоднородной, а представляющей собой слои упругих тел.

Математическая постановка:

В общем случае малые возмущения в упругих средах являются решением уравнения Ламэ (частный случай уравнения движения):

$$\rho\frac{\partial^2\omega}{\partial^2 t} = (\lambda+\mu)graddiv\omega + \mu\Delta\omega + \rho F$$

- ω – перемещения
- F – массовая сила

Решениями этой задачи являются:

- Продольная волна $c_1 = \sqrt{\frac{\lambda + 2\mu}{\rho}}$,
- Поперечная волна $c_2 = \sqrt{\frac{\mu}{\rho}}$,
- Релеевская волна $C_R = \frac{0.87 + 1.12 \cdot \nu}{1 + \nu} \cdot C_2$

Последняя интересна тем, что она вносит основной вклад в смещение на свободной границе.

Результаты взяты из источников [1] и [2]

Численное моделирование

С точки зрения численных методов данная задача является *динамической*, апроксимация неизвестных перемещений выглядит соответствующим образом:

$$u(t) \approx \sum_{i=1}^{M} N_i * u_i(t)$$

- N_i – функция формы.
- M – число узлов в сетке.
- $u_i(t)$ — неизвестные узловые перемещения

Подсталяем в уравнение движения:

$$\nabla\sigma = \rho\frac{\partial^2 u}{\partial t^2}$$

Получаем соответсвтующую галёркинскую систему уравнений, используя МКЭ:

$$M \cdot \frac{\partial^2 \tilde{u}}{\partial t^2} + K(\tilde{u}) - f = 0$$

- M – матрица масс

$$M = \int_{\Omega} N^T \cdot \rho \cdot N \mathrm{d}\Omega$$

- K – матрица жесткости

$$K = \int_{\Omega} \nabla N^T \cdot D \cdot \nabla N \mathrm{d}\Omega$$

- f – вектор правоый части (вектор сил)

$$f = \oint_{\Gamma} N^T \cdot \rho \cdot f(t) \mathrm{d}\Gamma$$

Схемы Ньюмарка

Для решения полученной системы будем использовать дискретизацию по времени по одному из вариантов схем Ньюмарка, в частном случае которого имеем явную схему по времени.

- Ищем прогнозное значение перемещения:

$$\tilde{u}_{n+1} = \tilde{u}_n + \tilde{v}_n \cdot \tau + \alpha_n \cdot \frac{\tau^2}{2}$$

- Ускорение для следующего шага:

$$M \cdot \alpha_{n+1} = f - K(\tilde{u}_{n+1})$$

- Скорость на следующем шаге:

$$\tilde{v}_{n+1} = \tilde{v}_n + \frac{\alpha_n + \alpha_{n+1}}{2} \cdot \tau$$

Массивный параллелизм и CUDA

CUDA - программно-аппаратная архитектура параллельных вычислений, позволяющая существенно увеличить вычислительную производительность благодаря использованию графических процессоров.

- CPU созданы для исполнения одного потока последовательных инструкций с максимальной производительностью.
- GPU проектируются для быстрого исполнения большого числа параллельно выполняемых потоков инструкций.

Основыные понятия

При программировании на GPU используются понятия:

- «host» - всё что относится к CPU (переменные, память, функции)
- «device» - соответсвтвенно, всё что относится к GPU

Существует три спецификатора функций, определяющие кем и откуда будет вызываться функция(kernel):

- __host__ - выполняется на CPU, вызывается с CPU
- __global__ - выполняется на GPU, вызывается с CPU
- __device__ - выполняется на GPU, вызывается на GPU

Вызов програмного кода, выполняемого на GPU производится с помощью функции-ядро помеченного соответствующим спецификатором. При этом данная функция-ядро выполняется одновременного большим числом потоков.

Kernel «nBlk, nTid» (args);

- Kernel — название функции
- nBlk — количество блоков, на которых будет выполнятся функция-ядро
- nTid — количество потоков, входящих в один блок

Работа программы в самом общем случае выглядит следующим образом:

1. Получение данных для расчётов

2. Выделение графической памяти для данных
   CudaMalloc(void** ptr, size_t size);

3. Копирование данных на GPU
   CudaMemcpy(void* dst, void* src, size_t, size, kind);

4. Произвести вычисления в GPU с помощью функции-ядра

5. Скопировать данные с GPU в оперативную память

6. Высвободить используемые ресурсы
   CudaFree(void** ptr);

Результаты для задачи Даламбера

Для сравнения использовалось одномерное волновое уравнение под действием вынуждающей силы:

$$u_{tt} = a^2 u_{xx} + f \qquad f(t) = (1 - 2\pi^2 f^2 t^2) e^{-\pi^2 f^2 t^2}$$

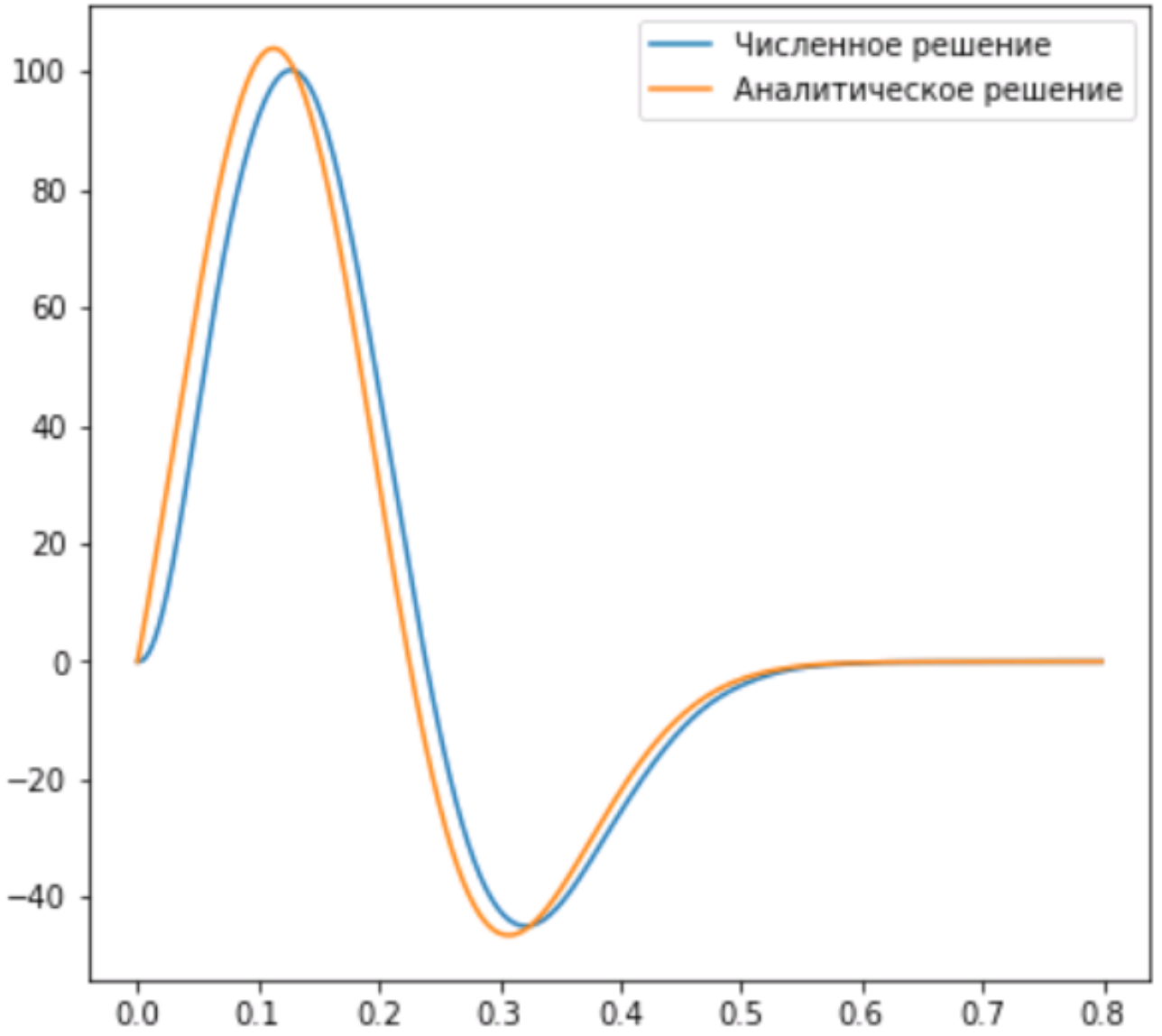
С начальными условиями:

$$u(x,0) = \varphi(x), \quad u_t(x,0) = \psi(x)$$

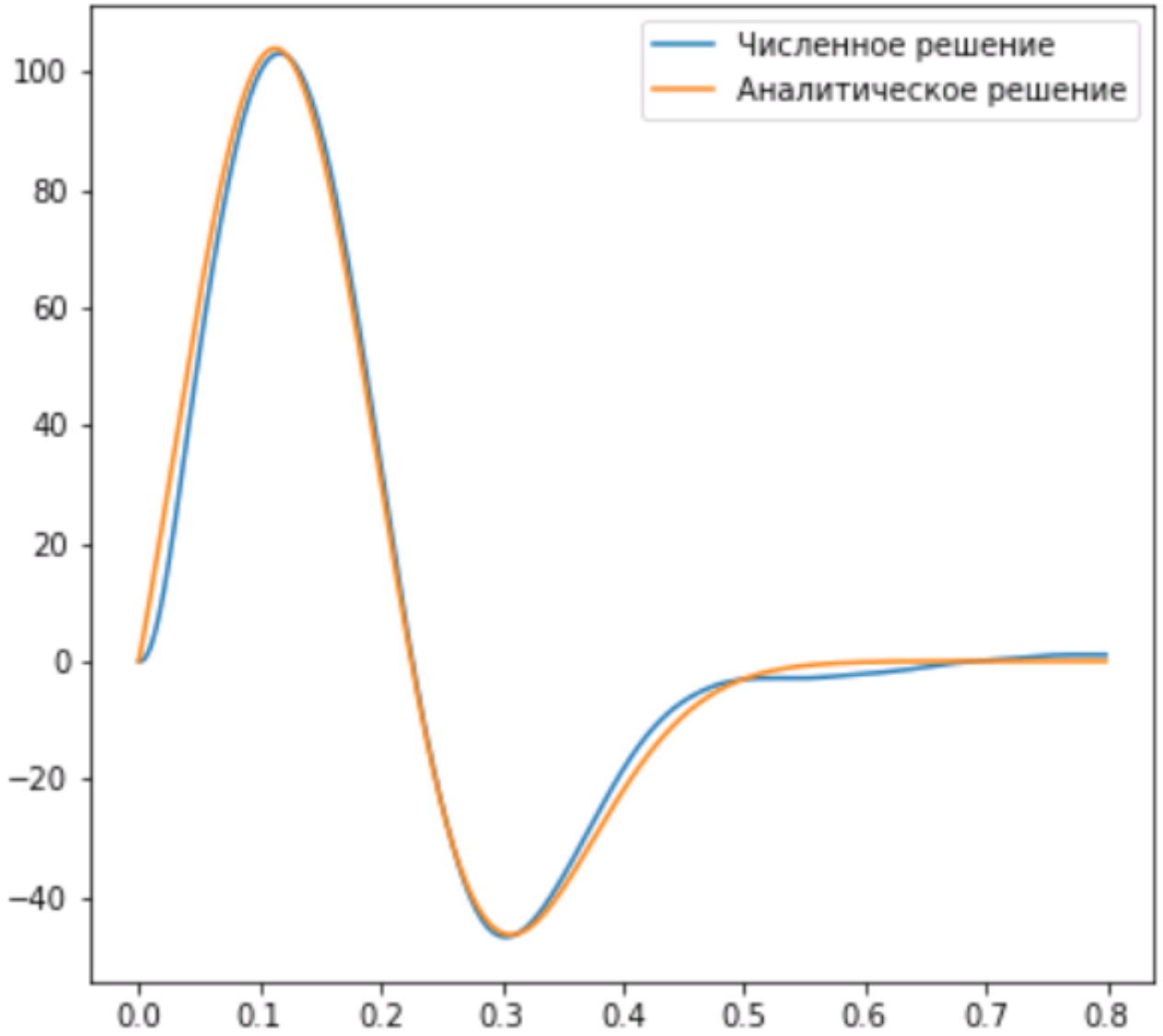
Его решение:

$$u(x,t) = \frac{\varphi(x+at) + \varphi(x-at)}{2} + \frac{1}{2a}\int\limits_{x-at}^{x+at} \psi(\alpha)d\alpha + \frac{1}{2a}\int\limits_{0}^{t}\int\limits_{x-a(t-\tau)}^{x+a(t-\tau)} f(s,\tau)dsd\tau$$

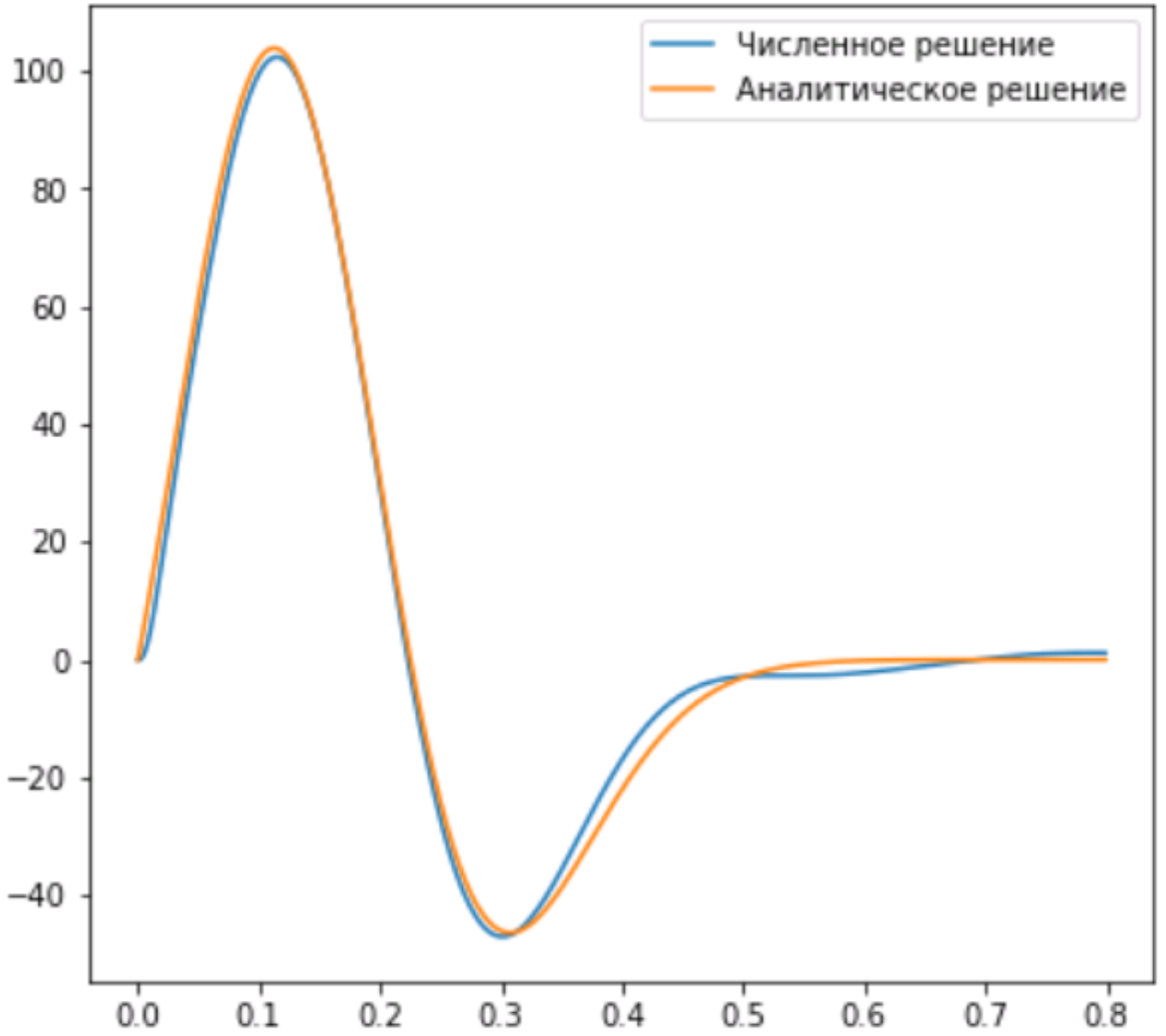
Размер сетки 144 узла



Размер сетки 288 узлов



Размер сетки 938 узлов



Анализ моделирования задачи Лемба

Было проведено сравнение решений для плоской внешней задачи Лэмба путем моделирования с помощью программы с использованием CUDA и путем моделирования с использованием CAE Fidesys.

Параметры задачи:

Модуль Юнга: E = 10000000

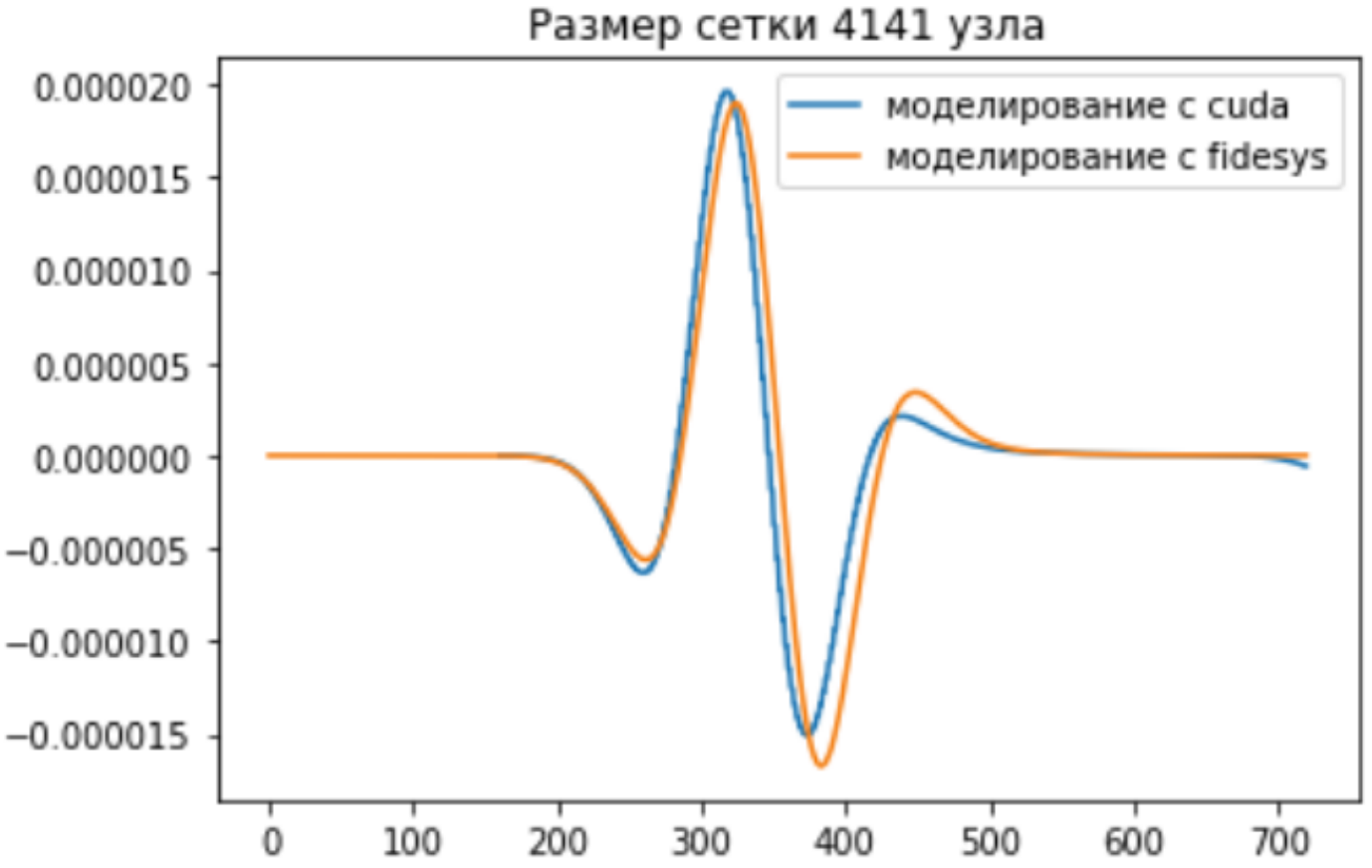
Коэфициент Пуассона: ν = 0

Плотность: ρ = 10000

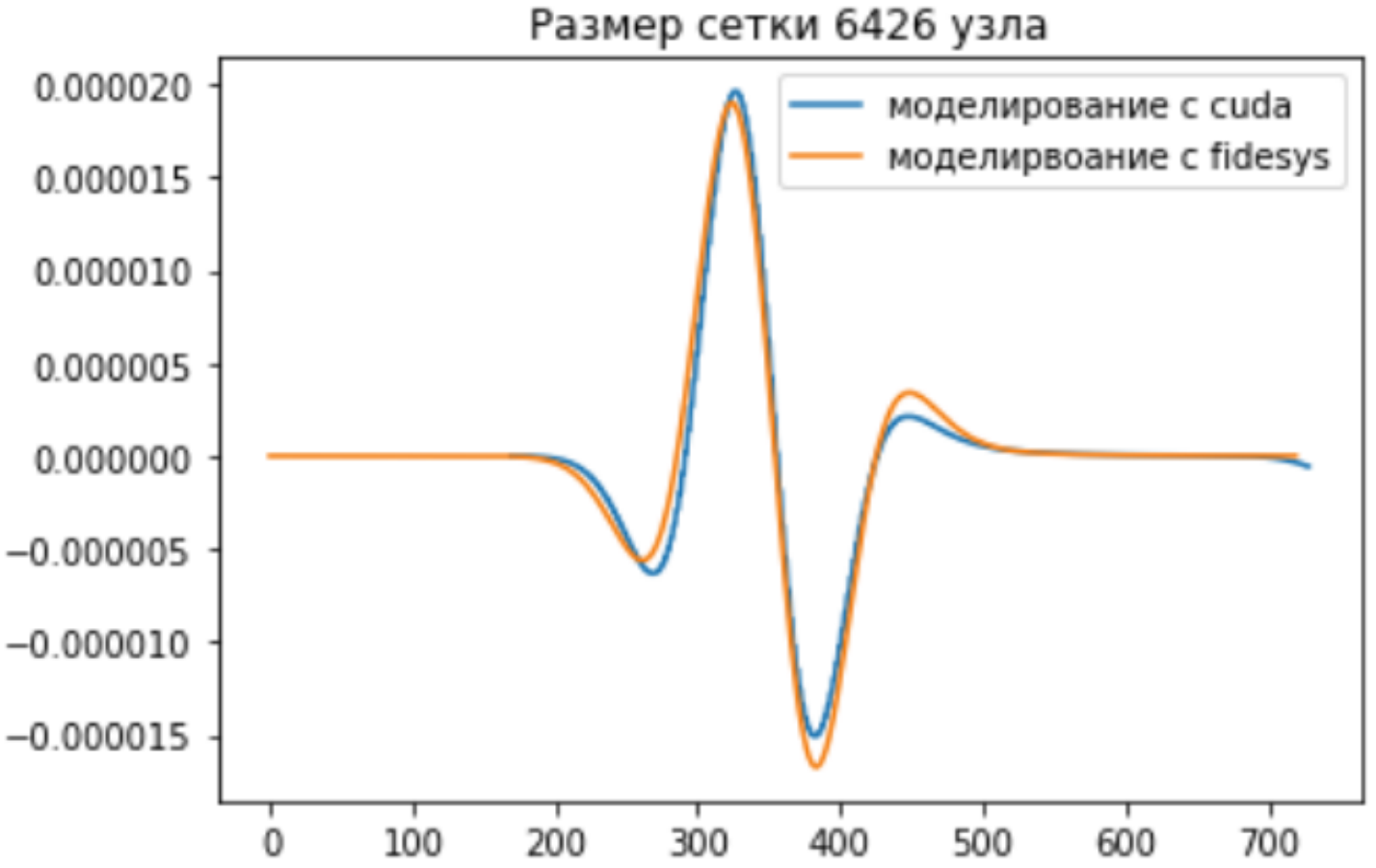
Источник: аплитуда Рикера A = 1, ω = 13, t_0 = 0;

$$f(t) = A\left(1 - 2(\pi\omega(t - t_0) - \pi)^2)\right) e^{-(\pi\omega(t-t_0)-\pi)^2}$$

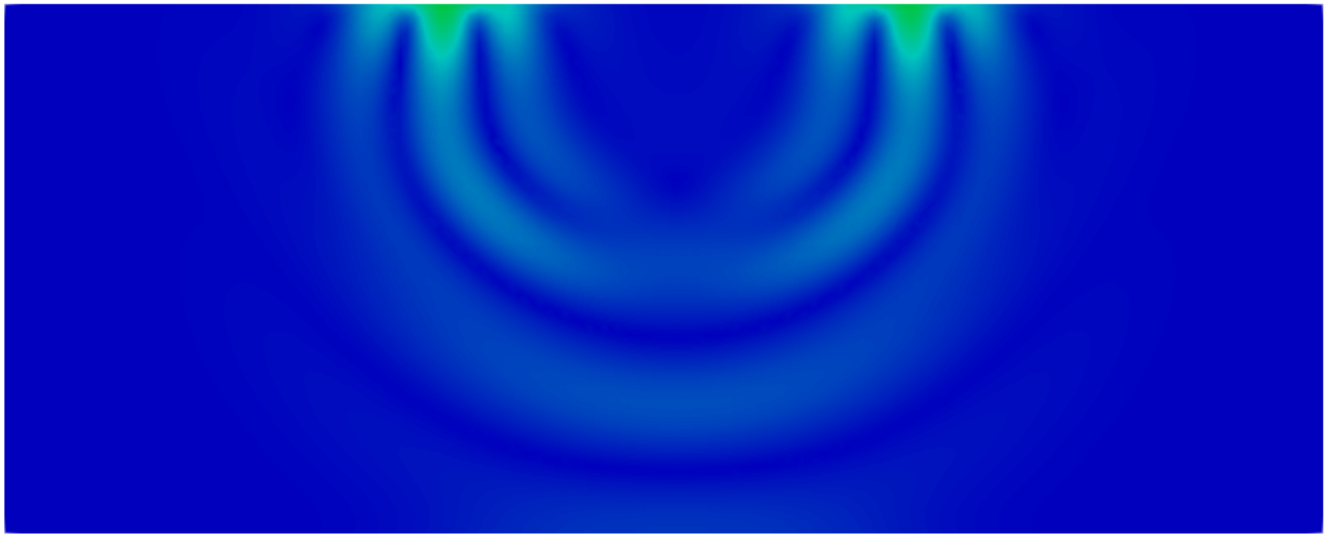
Результаты для скорости в конкретной точке

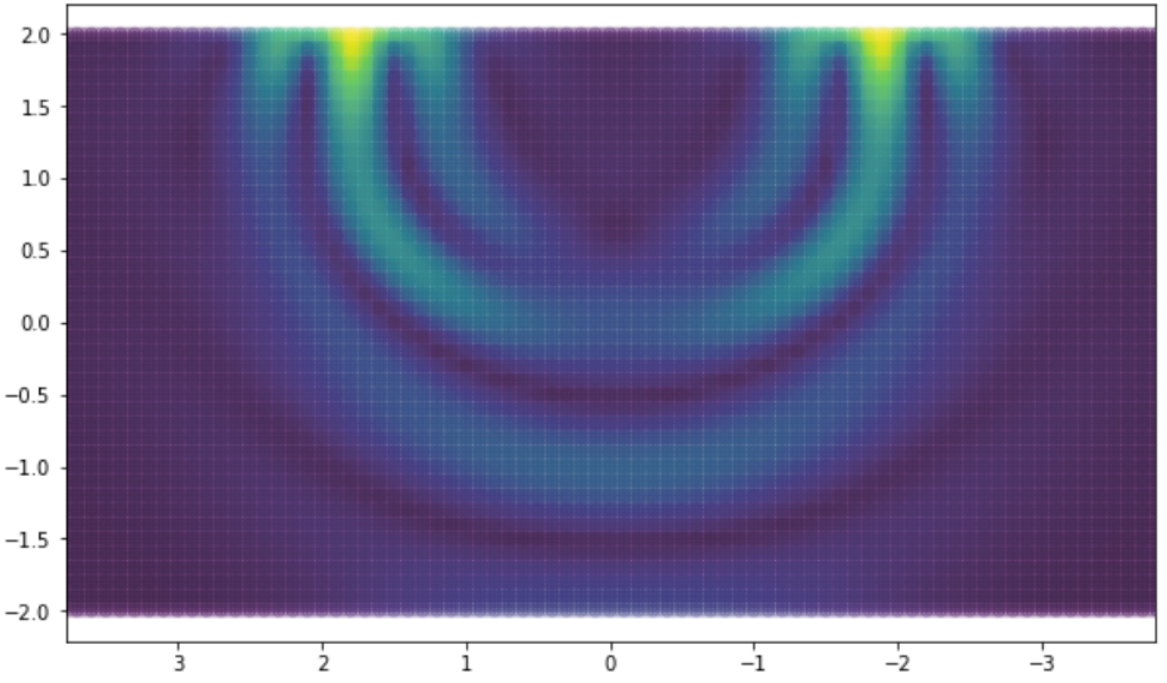


Результаты для скорости в конкретной точке

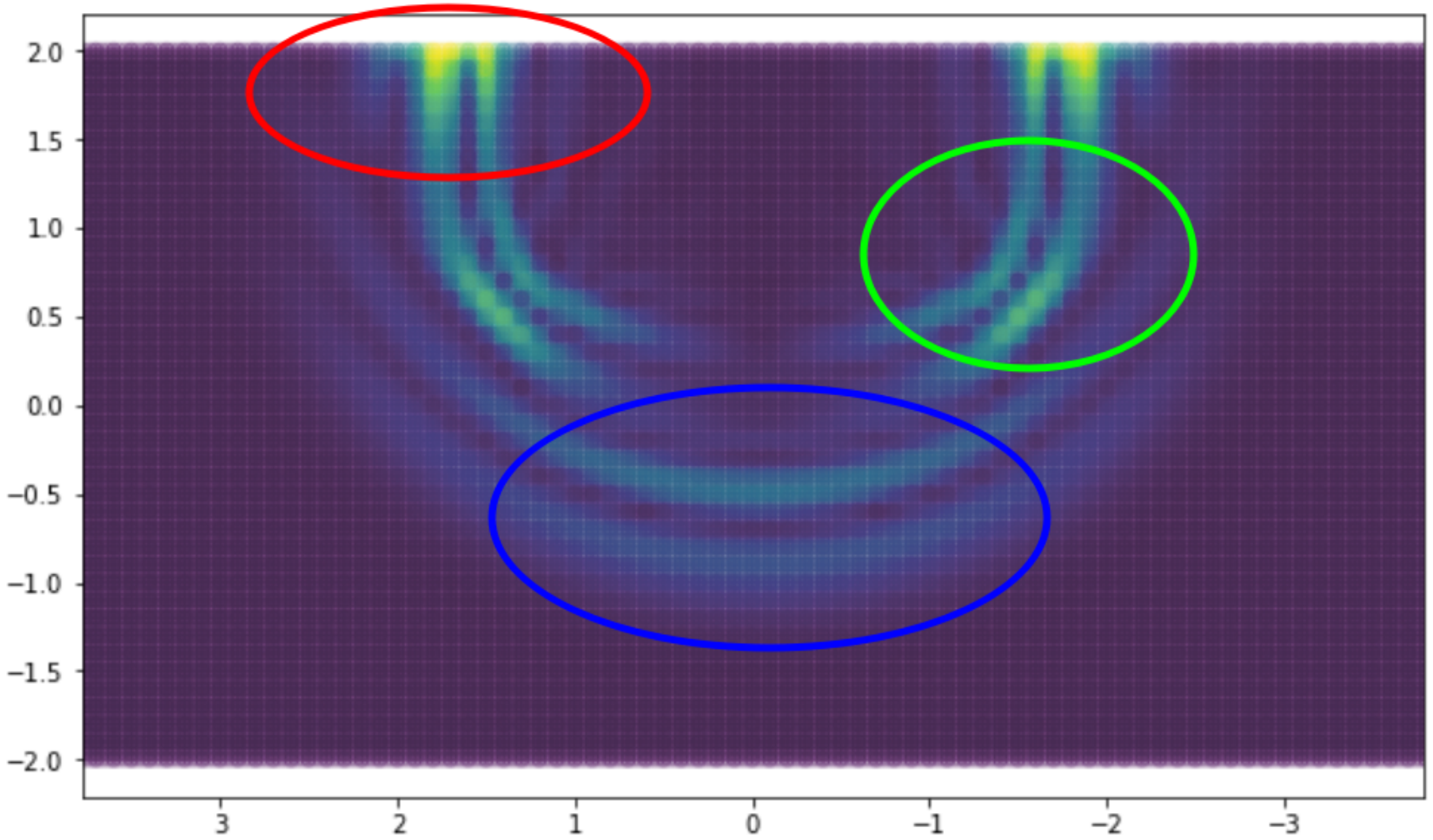


CAE Fidesys, программа с CUDA

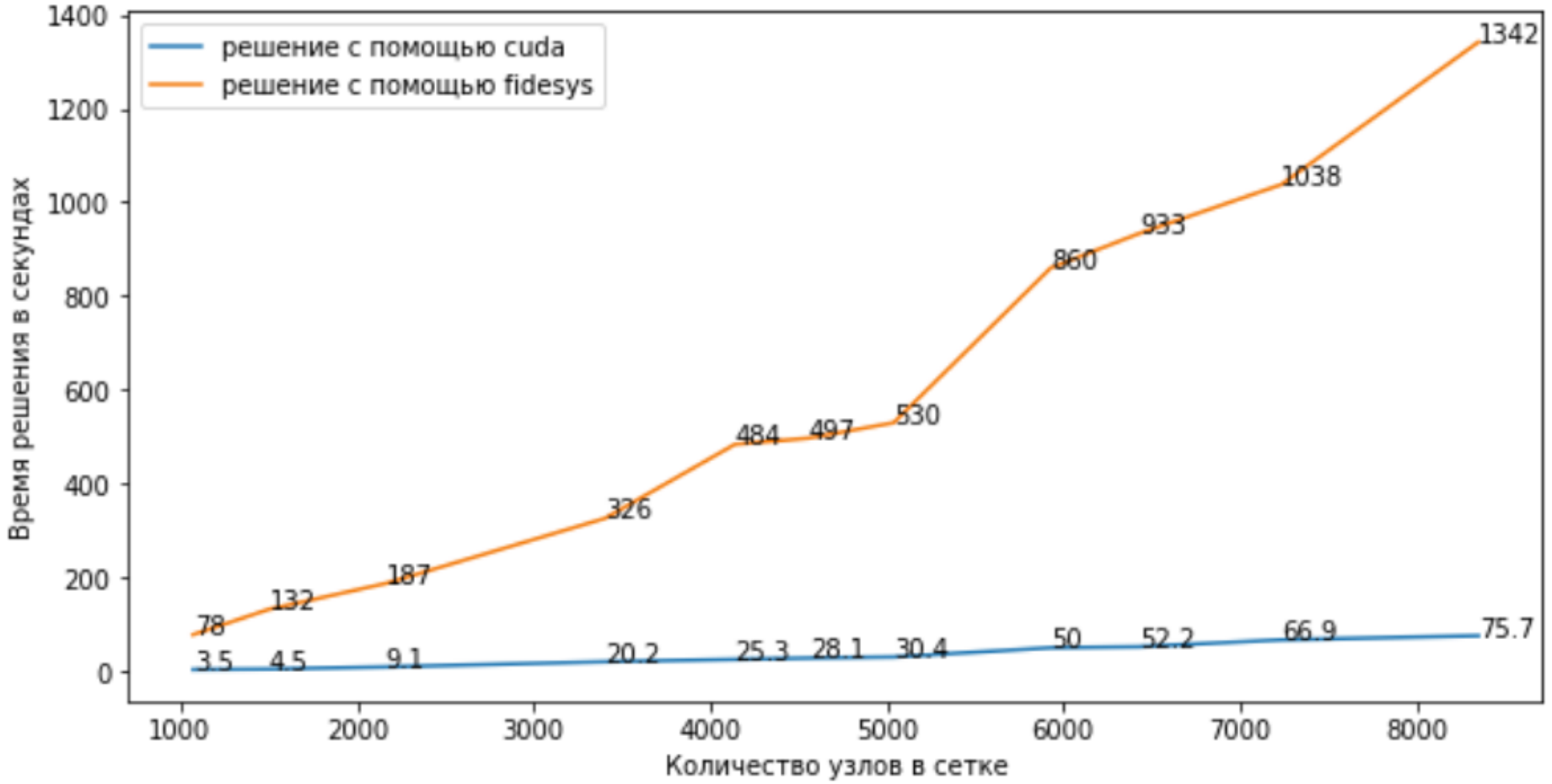




Красным — релеевская волна.
Зеленым — поперечная волна.
Синим — продольная волна.



Сравнение времени выполнения программы взависимости от числа узлов в сетке



Источники

[1] Механика сплошной среды. Том 2.
Л. И. Седов (1973)

[2] Е.О. Терентьева Задача Лэмба
[Электронный ресурс] // Строительство: наука образование. 2013. Вып. 3. Ст. 3.