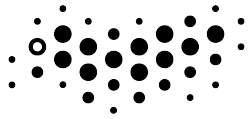
Федеральное государственное автономное
образовательное учреждение высшего образования
"Национальный исследовательский университет ИТМО"



УНИВЕРСИТЕТ ИТМО

Математический анализ
Расчетно-графическая работа №1
"Интеграл функции одной переменной"
Вариант 1

Выполнили:
Жуйков А. С.
Лопатенко Г. В.

Преподаватель:
Возианова А. В.

Апрель, 2022

Содержание

1 Интегральная сумма. 2
- 1.1 Ступенчатая фигура . . . 2
  - 1.1.1 Построение ступенчатой фигуры . . . 2
  - 1.1.2 Заключение по построению . . . 7
- 1.2 Последовательность интегральных сумм . . . 7
  - 1.2.1 Точное значение интеграла . . . 7
  - 1.2.2 Заключение . . . 9

2 Расчёт площади фигуры 10
- 2.1 Формулировка задачи . . . 10
- 2.2 Ход работы . . . 10

3 Несобственный интеграл 11
- 3.1 Формулировка задачи . . . 11
- 3.2 Первичный анализ . . . 11
- 3.3 Графики . . . 11
- 3.4 Признаки сравнения для определения сходимости несоюственных интегралов . . . 12
- 3.5 Анализ сходимости интеграла при разных значениях α . . . 13
  - 3.5.1 Случай 1: $\alpha < 0$ . . . 13
  - 3.5.2 Случай 2: $\alpha = 0$ . . . 13
  - 3.5.3 Случай 3: $0 < \alpha < 1$ . . . 14
  - 3.5.4 Случай 4: $\alpha = 1$ . . . 14
  - 3.5.5 Случай 5: $\alpha > 1$ . . . 14
- 3.6 Результаты . . . 15

4 Приложение определенного интеграла 16
- 4.1 Поле высот . . . 17
- 4.2 Освещение . . . 21

5 Подведение итогов 23
- 5.1 Полученные результаты . . . 23
- 5.2 Вывод . . . 23

1 Интегральная сумма.

Исследоваить интегральную сумму функции $f(x) = sinx,\ x \in \left[0; \frac{3\pi}{2}\right]$.

1) изобразить график функции, криволинейную трапецию, ограниченную графиком функции, вертикальными прямыми через концы отрезка и осью $0X$;
2) разбить отрезок на n элементарных отрезков, точками отметить их концы на рисунке;
3) выбрать по одной точке внутри каждого элементарного отрезка, отметить их на рисунке;
4) вычислить значения функции в выбранных точках, отметить их на рисунке;
5) изобразить ступенчатую фигуру на основе выбранного разбиения и точек внутри элементарных отрезков.

1.1 Ступенчатая фигура

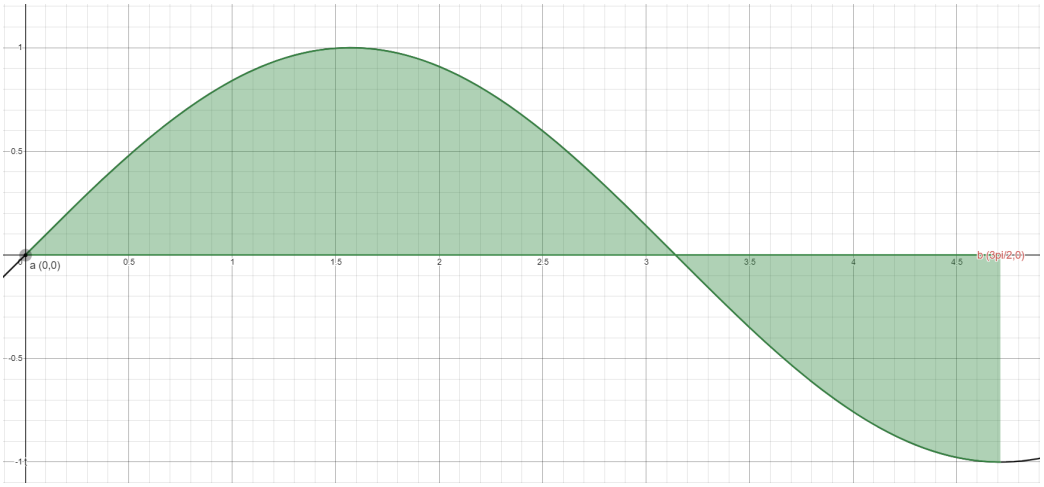


Рис. 1: График функции и криволиненая трапеция

1.1.1 Построение ступенчатой фигуры

При разбиении отрезка $[a; b] = \left[0; \frac{3\pi}{2}\right]$ на n равных частей, шаг разбиения находится по формуле:

$$h = \frac{b - a}{n}$$

Точки разбиения могут быть найдены соотношением:

$$x_k = a + (k-1) \cdot h,\ k \in [1; n]$$

Для выбора точек внутри отрезков введем параметр $t \in [0; 1]$:

$$\xi_k = a + (k-t) \cdot h$$

Рассмотрим конфигурации ξ_k от параметра t:
1) $t = 1 \Rightarrow \xi_k = x_k$ - левая сумма Дарбу
2) $t = 0 \Rightarrow \xi_k = x_{k+1}$ - правая сумма Дарбу
3) $t = 0.5 \Rightarrow \xi_k = \frac{x_k + x_{k+1}}{2}$

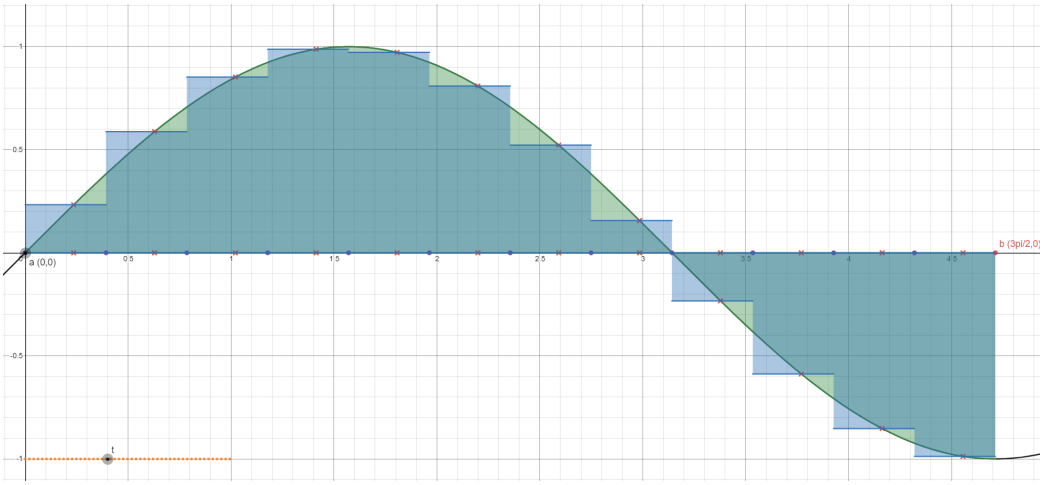
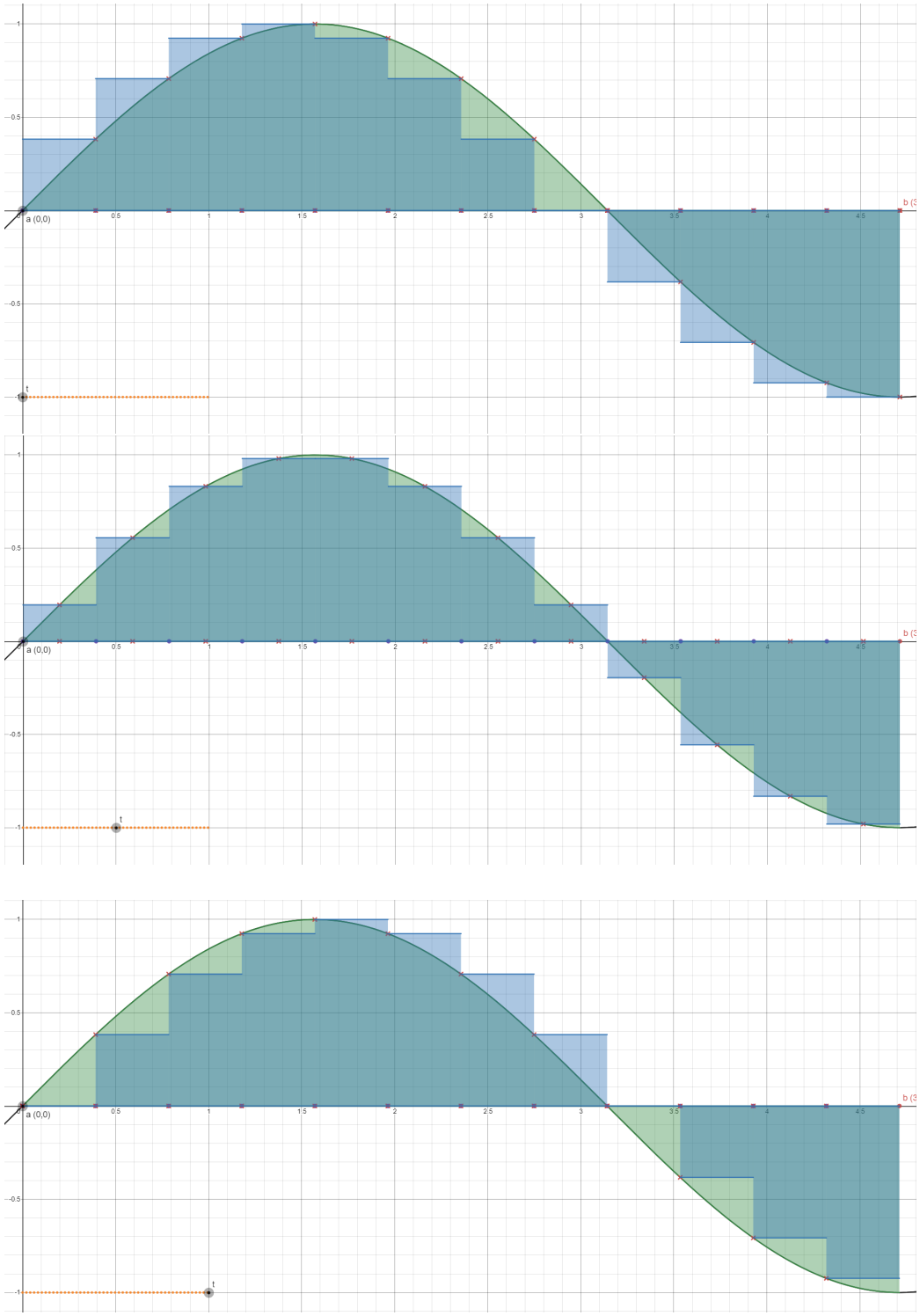
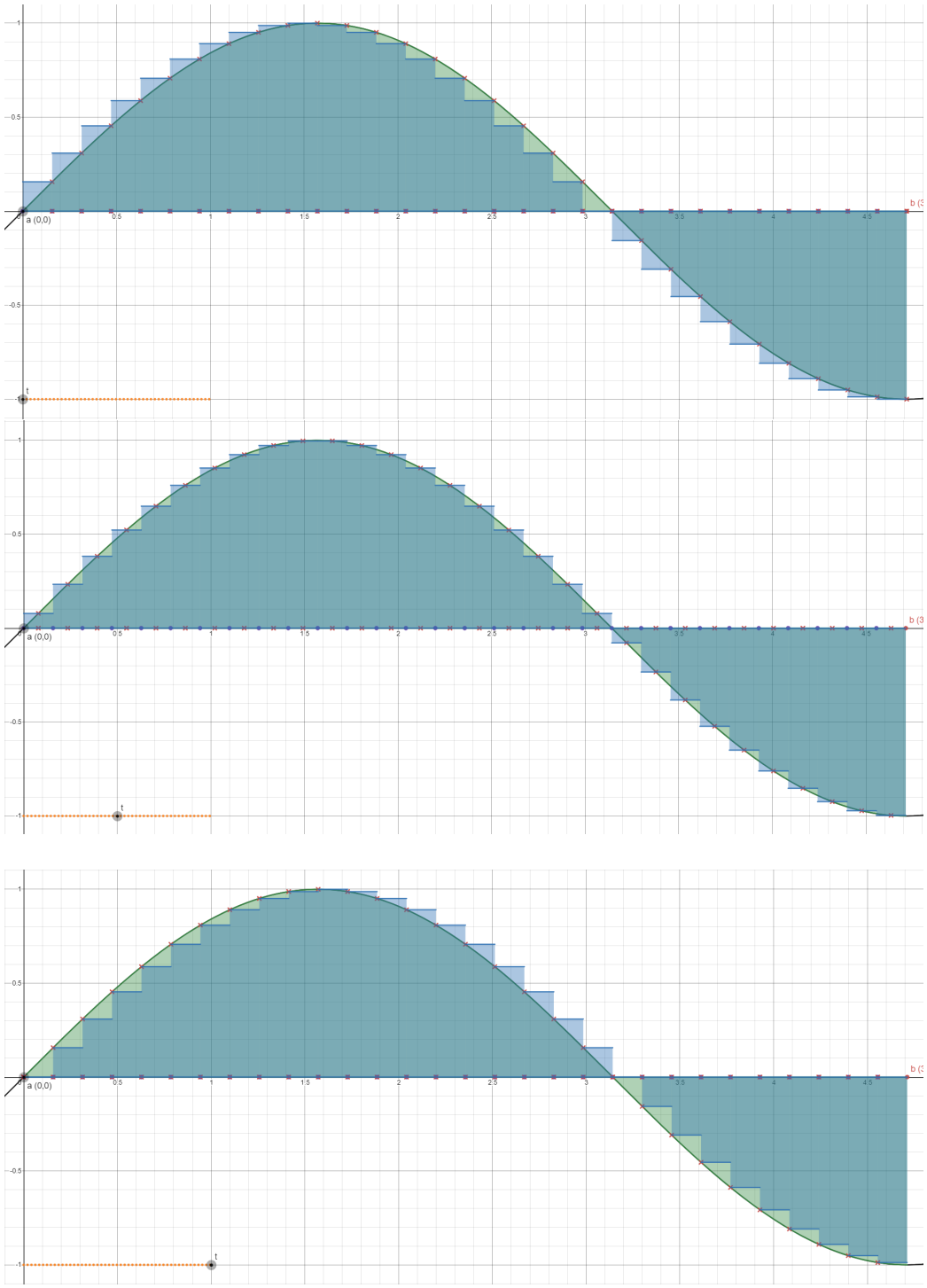
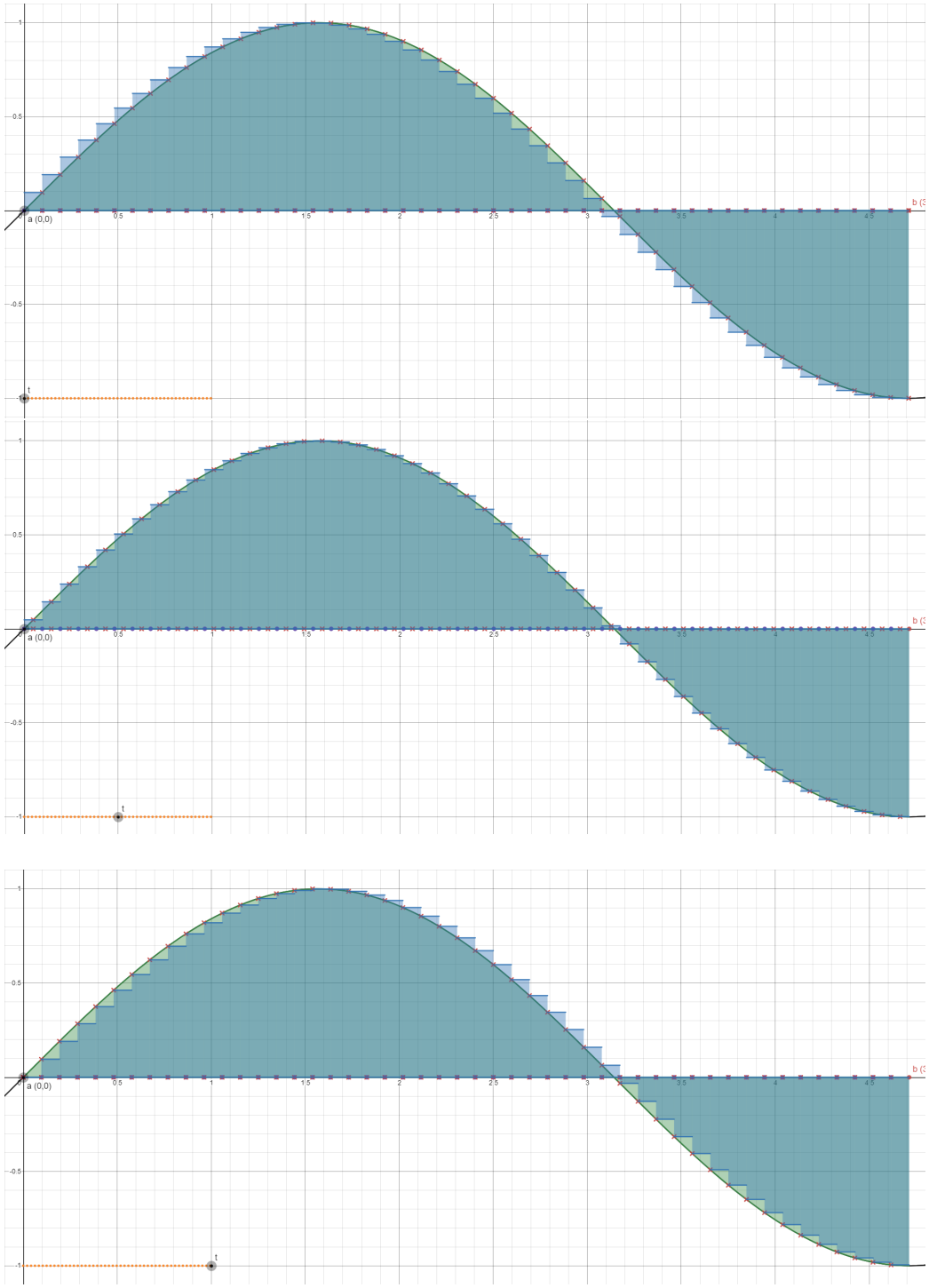


Рис. 2: График функции и криволиненая трапеция


a (0,0)
t
a (0,0)
t
a (0,0)
t


a (0,0)
t
a (0,0)
t
a (0,0)
t


a (0,0)
t
a (0,0)
t
a (0,0)
t

1.1.2 Заключение по построению

1. Без аналитически вычисленного значения интеграла нельзя однозначно определить, при каких значениях параметров (при каком положении точек внутри отрезков разбиения) фигура наиболее точно приближает криволинейную трапецию.

2. Можно предположить, что значения правой и левой сумм Дарбу не будут равны. Если бы график был симметричен относительно середины интервала интегрирования, то этот факт подтверждался бы аналитически.

3. Можно сделать вывод, что при уменьшении значения мелкоты разбиения (при увеличении n) точность приближения криволинейной трапеции увеличивается.

1.2 Последовательность интегральных сумм

Рассмотрим последовательность интегральных сумм, соответствующих различным значениям n, и установим ее поведение относительно истинного значения определенного интеграла исследуемой функции отрезке.

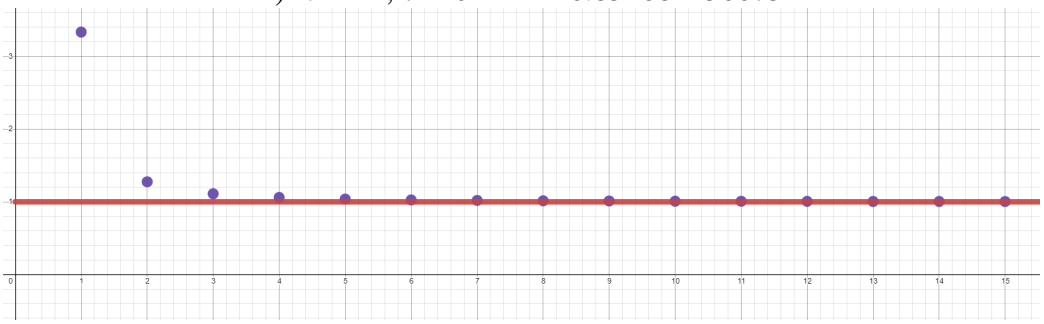
1.2.1 Точное значение интеграла

$$\int_0^{\frac{3\pi}{2}} sinx\,dx = -cosx\Big|_0^{\frac{3\pi}{2}} = 0 + 1 = 1$$

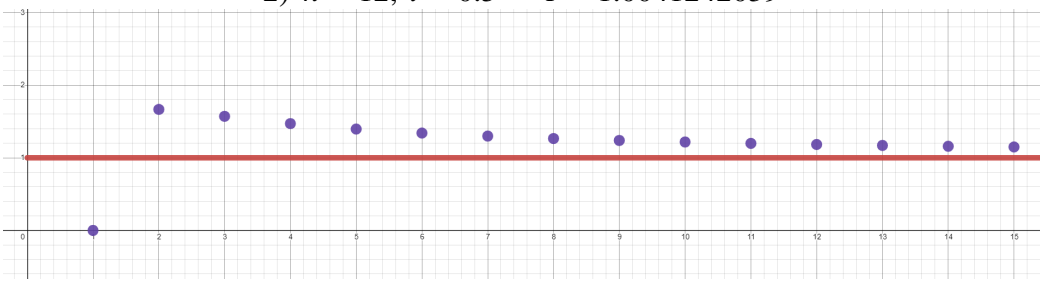
Значение интегральной суммы при разбиении:

$$\sum_{k=1}^{n} f(a + (k - t) \cdot h) \cdot h;\ h = \frac{b - a}{n}$$

1) $n = 12,\ t = 0 \rightarrow I = 0.834682136075$



2) $n = 12,\ t = 0.5 \rightarrow I = 1.0041242039$



3) $n = 12,\ t = 1 \rightarrow I = 1.1488414014$

1.2.2 Заключение

1. Вычислено точное значение определенного интеграла на заданном отрезке, что говорит об аналитическом подтверждении сходимости интеграла.

2. Элементы последовательностей левых, правых интегральных сумм, а также сумм, где значение ξ_k находится в серединах соответсвующих отрезков разбиения $[x_k; x_{k+1}]$ и прямая $y = 1$ изображены на координатной плоскости. При увеличении значения n элементы последовательности будут располагаться все ближе к искомой прямой со значением интеграла, посчитанного ранее.

Таким образом, сходимость ряда интегральных сумм показана аналитически и графически.

2 Расчёт площади фигуры

2.1 Формулировка задачи

Найти площадь фигуры, ограниченной функциями:

$$\begin{cases} \rho = 6\ sin(3x) \\ \rho = 3\ (\rho \geq 3) \end{cases} \tag{1}$$

Подзадачи:

1. Построить графики функций и тело, ограниченное ими
2. Предложить метод нахождения площади фигуры, ограниченной этими функциями
3. Подсчитать площадь и сверить со здравым смыслом

2.2 Ход работы

Решим уравнение для поиска точек пересечения графиков:

$$6 \sin 3x = 3$$

$$x_k = \frac{arcsin(\frac{1}{2})}{3} + 2\pi k,\ k \in \mathbb{Z}$$

$$x_l = \frac{\pi}{3} - \frac{arcsin(\frac{1}{2})}{3} + 2\pi l,\ l \in \mathbb{Z}$$

Тогда при $k = l = 0$ значения аргументов соответственно равны $x_{0k} = \frac{\pi}{18}$ и $x_{0l} = \frac{5\pi}{18}$.
Эти значения будут нижним и верхним пределами интегрирования.

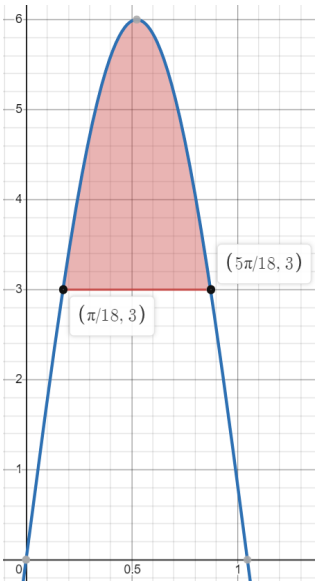


Рис. 3: Область между кривыми.

$$\int_{\frac{\pi}{18}}^{\frac{5\pi}{18}} 6 \sin 3x dx - \int_{\frac{\pi}{18}}^{\frac{5\pi}{18}} 3 dx =$$

$$\left(-2 \cos \frac{15\pi}{18} + 2 \cos \frac{3\pi}{18}\right) - 3\left(\frac{5\pi}{18} - \frac{\pi}{18}\right) =$$

$$2\sqrt{3} - \frac{2\pi}{3}$$

3 Несобственный интеграл

3.1 Формулировка задачи

Исследование несобственного интеграла на сходимость при всех параметрах α

$$\int_1^\infty \frac{\ln x}{x^\alpha} dx$$

3.2 Первичный анализ

1. Особые точки $x_1 = 1$ и $x_2 = \infty$
2. Тип: Несобственный интеграл первого рода
3. Функция неотрицательная на промежутке интегрирования

3.3 Графики

Построим графики при различных значениях α

Рис. 4: $\alpha = 0$

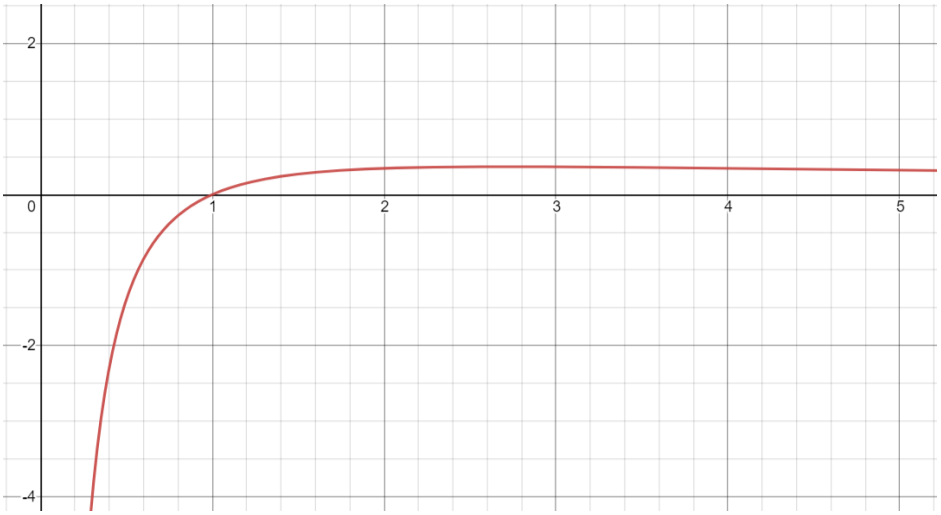


Рис. 5: $\alpha = 1$

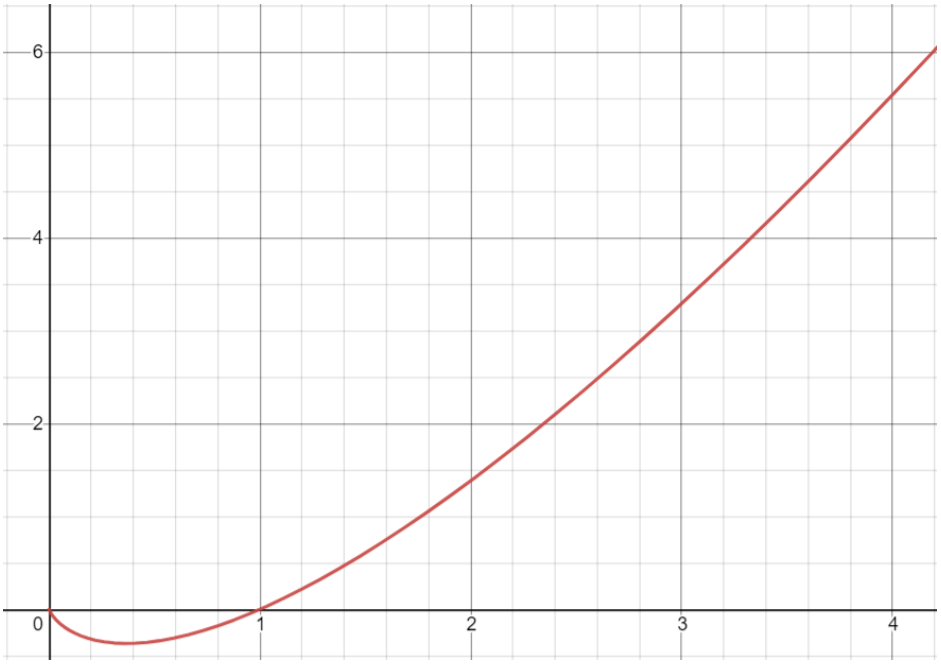


Рис. 6: $\alpha = -1$

3.4 Признаки сравнения для определения сходимости несобственных интегралов

Признак 1. Пусть функции $f(x)$ и $g(x)$ определены на промежутке (A, B) и удовлетворяют неравенству $f(x) \leq |g(x)|$. Тогда:

1. Из сходимости $\int\limits_A^B g(x)\,dx$ следует сходимость $\int\limits_A^B f(x)\,dx$.

2. Расходимость $\int\limits_A^B f(x)\,dx$ влечет расходимость $\int\limits_A^B g(x)\,dx$.

Признак 2. Если существует предел:

$$\lim_{x\to\infty}\left|\frac{f(x)}{g(x)}\right| = \lambda$$

1. При $0 < \lambda < \infty$ интегралы $\int\limits_A^\infty g(x)\,dx$ и $\int\limits_A^\infty f(x)\,dx$ имеют одинаковый характер сходимости.

2. При $\lambda = \infty$ из расходимости $\int\limits_A^\infty g(x)\,dx$ следует расходимость $\int\limits_A^\infty f(x)\,dx$.

3. При $\lambda = 0$ из сходимости $\int\limits_A^\infty g(x)\,dx$ следует сходимость $\int\limits_A^\infty f(x)\,dx$.

3.5 Анализ сходимости интеграла при разных значениях α

3.5.1 Случай 1: $\alpha < 0$

Если значение α отрицательное, то интеграл принимает вид:

$$\int\limits_1^\infty \frac{\ln x}{x^\alpha}dx = \int\limits_1^\infty x^\beta \ln x\,dx,\ \beta > 0$$

Так как $\lim\limits_{x\to\infty} x^\beta \ln x = \infty$, можно с уверенностью сказать, что при отрицательных значениях α интеграл расходится.

3.5.2 Случай 2: $\alpha = 0$

Интеграл принимает вид:

$$\int\limits_1^\infty \frac{\ln x}{x^0}dx = \int\limits_1^\infty \ln x\,dx$$

Применяя формулу интегрирования по частям, получаем следующее выражение:

$$\lim_{b\to\infty}(x\ln x - x)\,|_1^b = \lim_{b\to\infty}(x(\ln x - 1))\,|_1^b = \infty$$

Можно сделать вывод, что при $\alpha = 0$ интеграл расходится.

3.5.3 Случай 3: $0 < \alpha < 1$

В данном случае можно воспользоваться 2 признаком сравнения, где:

$$f(x) = \frac{\ln x}{x^\alpha}, \alpha \in (0, 1)$$
$$g(x) = \frac{1}{x}$$

И рассмотреть предел:

$$\lim_{x\to\infty} \left| \frac{\frac{\ln x}{x^\alpha}}{\frac{1}{x}} \right| = \lim_{x\to\infty} \left| \frac{\ln x}{x^{\alpha-1}} \right| = \lim_{x\to\infty} |x^\gamma \ln x| = \infty, \ \gamma \in (0, 1)$$

Так как $\int\limits_1^\infty \frac{1}{x}$ расходится, то и наш интеграл при $0 < \alpha < 1$ будет расходиться.

3.5.4 Случай 4: $\alpha = 1$

Интеграл принимает вид:

$$\int\limits_1^\infty \frac{\ln x}{x^1} dx = \int\limits_1^\infty \frac{\ln x}{x}\, dx$$

Занося $\frac{1}{x}$ под знак дифференциала, получаем следующее выражение:

$$\int\limits_1^\infty \ln x \, d(\ln x) = \lim_{b\to\infty} \frac{\ln^2 x}{2}\Big|_1^b = \infty$$

Можно сделать вывод, что при $\alpha = 1$ интеграл расходится.

3.5.5 Случай 5: $\alpha > 1$

Рассмотрим предел:

$$\lim_{x\to\infty} \left| \frac{\frac{\ln x}{x^\alpha}}{\frac{1}{x^p}} \right| = \lim_{x\to\infty} \left| \frac{\ln x}{x^{\alpha-p}} \right|, p \in (1, \alpha) \qquad (*)$$

Применяя правило лопиталя, получаем следующее выражение:

$$\frac{1}{\alpha - p} \cdot \lim_{x\to\infty} \frac{\frac{1}{x}}{x^{\alpha-p-1}} = \frac{1}{\alpha - p} \cdot \lim_{x\to\infty} \frac{1}{x^{\alpha-p}} = 0$$

Согласно 2 признаку сравнения, если предел * равен 0, то из сходимости знаменателя следует сходимость числителя. Так как $\int\limits_1^\infty \frac{1}{x^p}$ сходится при $\forall p > 1$, то и исходный интеграл будет сходиться.

3.6 Результаты

В результате применения признаков сравнения и различных методов вычисления несобетсвенных интегралов было установлено, что

$$\int_1^\infty \frac{\ln x}{x^\alpha} dx$$

расходится при $\alpha \leq 1$ и сходится при $\alpha > 1$.

4 Приложение определенного интеграла

Найти давление воды на поверхность цилиндра диаметром 4 м и высотой 6 м, если его верхнее основание находится на уровне свободной поверхности воды.

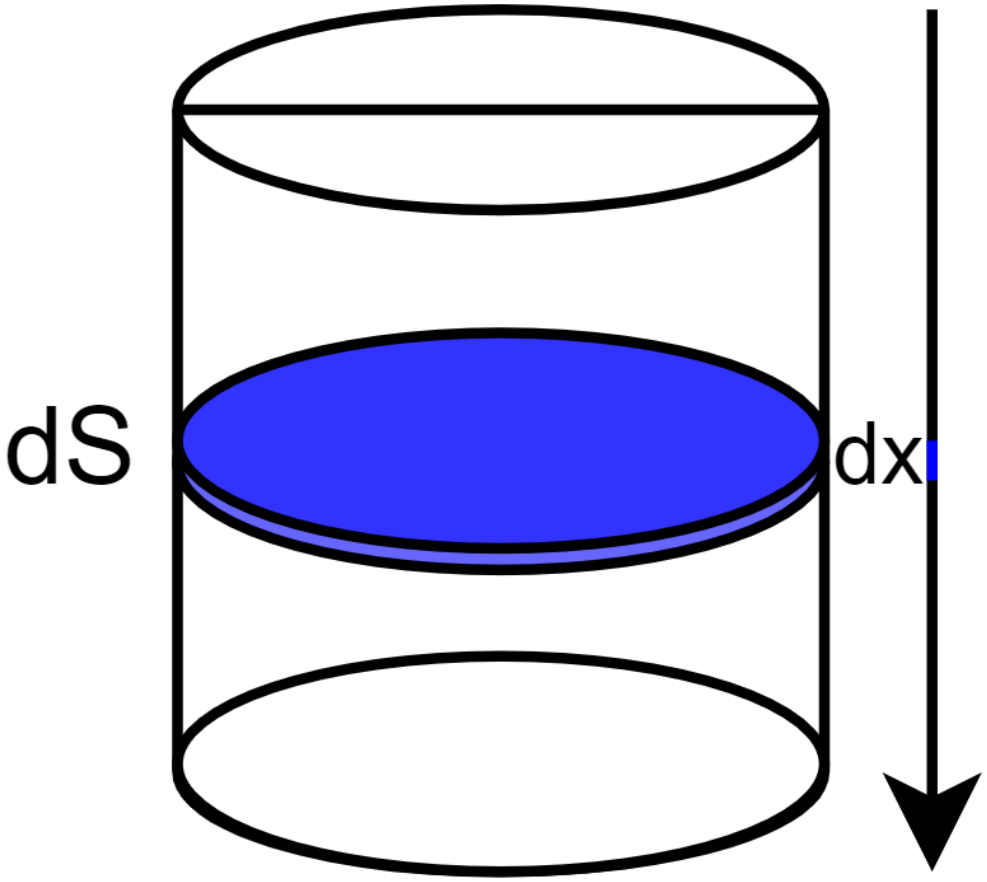


Рис. 7: Разбиение боковой площади цилиндра

При разбиении боковой площади цилиндра на элементарные компоненты

$$dS = 2\pi R\,dx$$

И если теперь найти интеграл $dp = \rho g x\,dS$ на промежутке интегрирования, получаем:

$$P = \int dp = \int_0^H \rho g x \cdot 2\pi R\,dx$$

$$P = \int_0^H \rho g x \cdot 2\pi R \, dx = \rho g x^2 \cdot \pi R \Big|_0^H = \rho g \pi R H^2$$

Таким образом, мы явно нашли зависимость значения давления на боковую поверхность цилиндра от геометрических характеристик тела, явно его задающих.

Однако заметим, что полное давление на поверхность цилиндрического тела еще не определено, потому что необходимо добавить компоненту силы Архимеда, действующую на нижнюю крышку цилиндра:

$$P_\Sigma = P + P_A = \rho g \pi R H^2 + \rho g V = \rho g \pi R H^2 + \rho g H \left(\pi R^2\right)$$

$$P_\Sigma = \rho g \pi R H \cdot (H + R)$$

$$P_\Sigma = 1000 * 10 * \pi * 4 * 6 * 10 = 24\pi \cdot 10^5 \; Pa$$

Ответ: $24\pi \cdot 10^5$ Па

4.1 Поле высот

Для того, чтобы эффективно реализовать симуляцию, используя распределенные вычисления на GPU, необходимо принимать во внимание следующие 2 факта:

1. Копирование данных со стороны CPU на сторону GPU является очень затратной операцией,

2. GPU имеет ограниченный объем памяти.

Вторая проблема решится на уровне постановки задачи – предполагается, что вся необходимая для симуляции информация, полностью помещается в памяти GPU.

Для того, чтобы решить первую проблему, необходимо написать программу таким образом, чтобы значения, вычисленные на GPU не передавались обратно на сторону CPU, а сразу копировались в OpenGL буффер. Для того, чтобы реализовать такое поведение, CUDA предоставляет 4 функции:

- cudaGraphicsGLRegisterBuffer – регистрирует вспомогательную CUDA структуру, которая может обращаться к OpenGL буферу,

- cudaGraphicsMapResources – соединяет вспомогательную структуру с OpenGL буфером,
- cudaGraphicsResourceGetMappedPointer – возвращает указатель, с помощью которого можно скопировать данные в OpenGL буфер напрямую,
- cudaGraphicsUnmapResources – закрывает соединение вспомогательной структуры с OpenGL буфером.

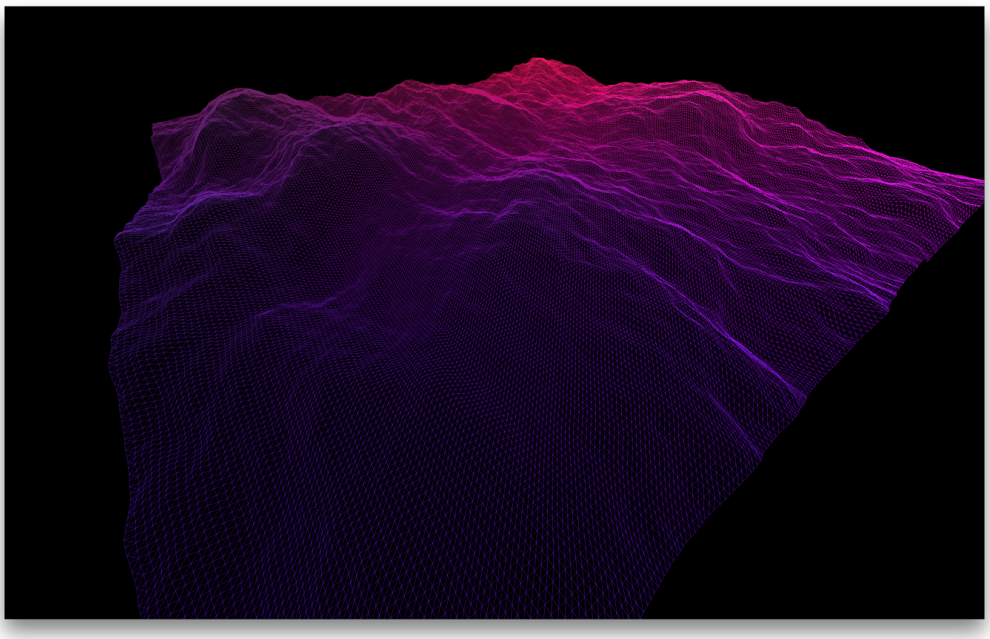


Рис. 8: Пример сгенерированного поля высот

Вычисление поля высот происходит каждый раз, когда рендерится изображение.

```
1 // generate wave spectrum in frequency domain
2 cudaGenerateSpectrumKernel(d_h0, d_ht, spectrum, meshSize,
      meshSize, curTime, patchSize);
3
4 // execute inverse FFT to convert to spatial domain
5 checkCudaErrors(cufftExecC2C(fftPlan, d_ht, d_ht,
      CUFFT_INVERSE));
6
7 // update heightmap values in vertex buffer
8 checkCudaErrors(cudaGraphicsMapResources(1, &
      cuda_heightVB_resource, 0));
```

```
9  checkCudaErrors(cudaGraphicsResourceGetMappedPointer((void **)
       &g_hptr, &num_bytes, cuda_heightVB_resource));
10 cudaUpdateHeightmapKernel(g_hptr, d_ht, meshSize, meshSize);
11 checkCudaErrors(cudaGraphicsUnmapResources(1, &
       cuda_heightVB_resource, 0));
```

В данном коде функции cudaGenerateSpectrumKernel, cufftExecC2C, cudaUpdateHeightmapKernel выполняются на GPU. Функция cufftExecC2C является библиотечной функцией, которая производит в данном случае обратное преобразование Фурье, а оставшиеся функции написаны вручную и имеют следующую реализацию:

```
1  // generate wave heightfield at time t based on initial
       heightfield and dispersion relationship
2  __global__ void generateSpectrumKernel(float2 *h0, float2 *ht,
3                  unsigned int in_width, unsigned int out_width,
4                  unsigned int out_height, float t, float
                       patchSize)
5  {
6    unsigned int x = blockIdx.x*blockDim.x + threadIdx.x;
7    unsigned int y = blockIdx.y*blockDim.y + threadIdx.y;
8    unsigned int in_index = y*in_width+x;
9    unsigned int in_mindex = (out_height - y)*in_width + (
         out_width - x); // mirrored
10   unsigned int out_index = y*out_width+x;
11
12   float2 k;
13   k.x = (-(int)out_width / 2.0f + x) * (2.0f * CUDART_PI_F /
         patchSize);
14   k.y = (-(int)out_width / 2.0f + y) * (2.0f * CUDART_PI_F /
         patchSize);
15
16   float k_len = sqrtf(k.x*k.x + k.y*k.y);
17   float w = sqrtf(9.81f * k_len);
18
19   if((x < out_width) && (y < out_height)) {
20     float2 h0_k = h0[in_index];
21     float2 h0_mk = h0[in_mindex];
22     ht[out_index] = complex_add(complex_mult(h0_k, complex_exp
           (w * t)),
23                         complex_mult(conjugate(h0_mk),
                               complex_exp(-w * t)));
24   }
25 }
26
27 // update height map values based on output of FFT
28 __global__ void updateHeightmapKernel(float  *heightMap,
29                  float2 *ht, unsigned int width)
30 {
```

```
31    unsigned int x = blockIdx.x * blockDim.x + threadIdx.x;
32    unsigned int y = blockIdx.y * blockDim.y + threadIdx.y;
33    unsigned int i = y * width  +x;
34
35    float sign_correction = ((x + y) & 0x01) ? -1.0f : 1.0f;
36
37    heightMap[i] = ht[i].x * sign_correction;
38  }
```

Функция generateSpectrumKernel генерирует поле высот из начального поля высот и пройденного времени, а updateHeightmapKernel – вспомогательная функция, которая реализует смещение точек после обратного преобразования Фурье.

Не менее интересной является реализация функции, которая генерирует начальное поле высот:

```
1   float Waves::phillips(float Kx, float Ky)
2   {
3     float k_squared = Kx * Kx + Ky * Ky;
4
5     if (k_squared == 0.0f) {
6       return 0.0f;
7     }
8
9     float L = windSpeed * windSpeed / g;
10    float k_x = Kx / sqrtf(k_squared);
11    float k_y = Ky / sqrtf(k_squared);
12    float w_dot_k = k_x * windDir.x + k_y * windDir.y;
13    float phillips = A * expf(-1.0f / (k_squared * L * L))
14                           / (k_squared * k_squared) * w_dot_k *
                                w_dot_k;
15
16    // filter out waves moving opposite to wind
17    if (w_dot_k < 0.0f) {
18      phillips *= dirDepend; // dir_depend;
19    }
20
21    return phillips;
22  }
23
24  void Waves::generateH0()
25  {
26    for (unsigned int y = 0; y < spectrum; ++y) {
27      for (unsigned int x = 0; x < spectrum; ++x) {
28        float kx = (-(int)meshSize / 2.0f + x) * (2.0f *
              CUDART_PI_F / patchSize);
29        float ky = (-(int)meshSize / 2.0f + y) * (2.0f *
              CUDART_PI_F / patchSize);
30
```

```
31          float P = sqrtf(phillips(kx, ky));
32
33          float Er = gauss();
34          float Ei = gauss();
35
36          float h0_re = Er * P * CUDART_SQRT_HALF_F;
37          float h0_im = Ei * P * CUDART_SQRT_HALF_F;
38
39          int i = y * spectrum + x;
40          h_h0[i].x = h0_re;
41          h_h0[i].y = h0_im;
42        }
43      }
44  }
```

4.2 Освещение

Реализация модели освещения Блинна-Фонга имеет стандартный вид:

```
1  /* vertex shader */
2  #version 410 core
3
4  layout(location = 0) in vec4 meshPos;
5  layout(location = 1) in float height;
6  layout(location = 2) in vec2 slope;
7
8  uniform mat4 PVM;
9  uniform vec3 lightPos;
10 uniform vec3 eyePos;
11
12 out vec3 l;
13 out vec3 h;
14 out vec3 n;
15 out vec3 r;
16
17 out vec4 pos;
18
19 void main() {
20   vec3 lp = abs(lightPos);
21   vec3 p = vec3(meshPos.x, 1e+2 * height, meshPos.z);
22   gl_Position = PVM * vec4(p, 1.0);
23   p.x = p.x - 1000; p.z = p.z - 1000;
24   p.y = p.y - 500;
25   pos = vec4(p, 1.0);
26   l = normalize(lp - p);
27   vec3 v = normalize(eyePos - p);
28   h = normalize((v + l) / length(v + l));
```

```
  n = normalize(cross( vec3(0.0, slope.y, 1.0 / 256), vec3(1.0
      / 256, slope.x, 0.0)));
  r = reflect(-l, n);
}
```

```
/* fragment shader */
#version 410 core

in vec4 pos;
out vec4 fColor;

uniform vec3 sourceColor;
uniform vec3 diffColor;
uniform vec3 specColor;
uniform vec3 lightPos;
uniform vec3 eyePos;

in vec3 l;
in vec3 h;
in vec3 n;
in vec3 r;

uniform vec3 Ka;
uniform vec3 Kd;
uniform vec3 Ks;
uniform float alpha;

vec3 BlinnPhongModel()
{
  return  Ka * sourceColor +
          Kd * max(dot(n, -l), 0.0) * diffColor +
          Ks * max(pow(dot(n, h), alpha), 0.0) * specColor;
}


void main (void) {
  vec3 BlinnPhong = exp(-0.8 + 1.2*abs(pos.x/3000+pos.z/3000))
      * BlinnPhongModel();
  fColor = vec4(BlinnPhong, 0.9);
}
```

В фрагментном шейдере используется α-канал не равный единице, в данной реализации $\alpha = 0.9$, чтобы была возможность сквозь воду просматривать дно.

5 Подведение итогов

5.1 Полученные результаты

Результат работы программы виден на следующем скриншоте:

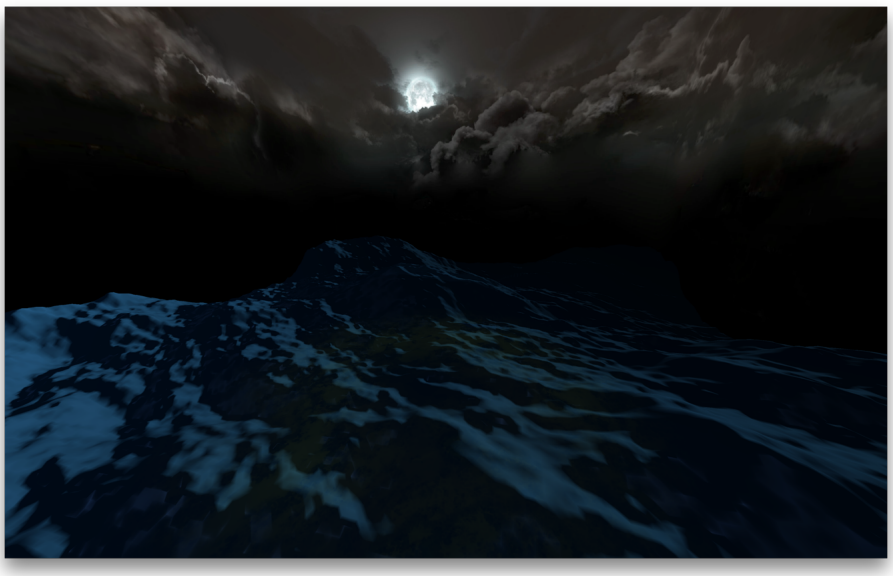


Рис. 9: Скриншот работы программы

Полученный результат напоминает жидкость, но океан имеет более сложную текстуру. Можно было бы продолжить исследовать проблему симуляции поверхности океана и добавить такие эффекты, как порывистые волны, брызги, пену, более подходящую для океана модель освещения, интерференцию волн, отражение мира на поверхности воды, каустический эффект и многое другое, которые бы улучшили внешний вид воды, но тема слишком сложная для любительского ознакомления.

5.2 Вывод

Симуляция поверхности океана - очень интересный, важный и активно развивающийся раздел моделирования. С помощью распределенных вычислений на GPU можно добиться вычисления очень большой площади поверхности в режиме реального времени с неплохой точностью. В данной работе была реализована самая простая статистическая модель волны, однако даже эта модель позволяет просчитать поведение волн с хорошей точностью.