Компьютерная графика
Visual C++ & **OpenGL**

Визуализация результатов численных расчетов

1. Знакомство

- Open Graphics Library - стандартная библиотека для 32-разрядных ОС
- 3D-графика + рендернинг
- opengl32.dll, glu32.dll

1. Знакомство

Геометрические и растровые примитивы	B-сплайны	Альфа-канал	Сглаживание цвета
Буфер аккумулятора	Градиентная заливка	Двойная буферизация	Заливка и освещенность фактур
Пространственные преобразования	Текстуры	Атмосферные эффекты	

2. Основные типы данных OpenGL

- Команды OpenGL начинаются с префикса gl, константы – с префикса GL_.

Суффикс	Описание	Тип в C	Тип в OpenGL
b	8-битовое целое	char	GLbyte
s	16-битовое целое	short	GLshort
i	32-битовое целое	long	GLint GLsizei
f	32-битовое вещественное число	float	GLfloat, GLclampf
d	64-битовое вещественное число	double	GLdouble, GLclampd
ub	8-битовое беззнаковое целое	unsigned char	GLubyte, GLboolean
us	16-битовое беззнаковое целое	unsigned short	GLushort
ui	32-битовое беззнаковое целое	unsigned long	GLuint, GLenum, GLbitfield

2. Основные типы данных OpenGL

- Многие команды имеют как векторные, так и не векторные версии.

```
glColor3f(1.0, 1.0, 1.0);
```

=

```
GLfloat color[] = {1.0, 1.0, 1.0};
glColor3fv(color);
```

- Функции для определения значений переменных:

glGetBooleanv(), glGetDoublev(), glGetFloatv(), glGetIntegerv()

3. Рисование геометрических объектов. Буферы и цвет.

- Фрейм буфер, z-буфер, буфер трафарета, аккумулирующий буфер.
- **glClear(Glbitfieldmask):**

  GL_COLOR_BUFFER_BIT – очистить буфер изображения (фреймбуфер);

  GL_DEPTH_BUFFER_BIT – очистить z-буфер;

  GL_ACCUM_BUFFER_BIT – очистить аккумулирующий буфер;

  GL_STENCIL_BUFFER_BIT – очистить буфер трафарета.

3. Рисование геометрических объектов. Буферы и цвет.

- **glClearColor(GLfloat red, GLfloat green, GLfloat blue, GLfloat alpha) -** цвет, которым очищается буфер изображения
- **glClearDepth(GLfloatdepth) -** значение, записываемое в z-буфер
- **glClearStencil(GLint s) -** значение, записываемое в буфер трафарета
- **glClearAccum(GLfloat red, GLfloat green, GLfloat blue, GLfloat alpha) -** цвет, записываемый в аккумулирующий буфер

3. Рисование геометрических объектов. Буферы и цвет.

- **glColor{3 4}{b s i f d ub us ui}[v](TYPE red, …)** – задание цвета объекта
- **glFlush() -** немедленное рисование ранее переданных команд
- **glFinish() -** ожидание завершения всех ранее переданных команд
- **glEnable(GL_DEPTH_TEST)** – удаление невидимых поверхностей методом z-буфера

3. Рисование геометрических объектов. Графические примитивы.

- OpenGL работает с однородными координатами (x, y, z, w)
- **Линия** - отрезок, заданный своими начальной и конечной вершинами; **грань** - замкнутый выпуклый многоугольник
  с несамопересекающейся границей
- Все геометрические объекты задаются посредством вершин, которые задаются процедурой:

glVertex{2 3 4}{s i f d}[v](TYPE x, …)

Пример:

```
glVertex2s(1, 2);
glVertex3f(2.3, 1.5, 0.2);
GLdouble vect[] = {1.0, 2.0, 3.0, 4.0};
glVertex4dv(vect);
```

3. Рисование геометрических объектов. Графические примитивы.

- **glBegin(), glEnd()**

  GL_POINTS – набор отдельных точек;

  GL_LINES – пары вершин, задающих отдельные точки;

  GL_LINE_STRIP – незамкнутая ломаная;

  GL_LINE_LOOP – замкнутая ломаная;

  GL_POLYGON – простой выпуклый многоугольник;

  GL_TRIANGLES – тройки вершин, интерпретируемые как вершины отдельных треугольников;

  GL_TRIANGLE_STRIP – связанная полоса треугольников;

  GL_TRIANGLE_FAN – веер треугольников;

  GL_QUADS – четвёрки вершин, задающие выпуклые четырёхугольники;

  GL_QUAD_STRIP – полоса четырёхугольников.

3. Рисование геометрических объектов. Графические примитивы.

- Встречаются команды задания различных атрибутов вершин **glVertex(), glColor(), glNormal(), glCallList(), glCallLists(), glTexCoord(), glEdgeFlag(), glMaterial()**
- Пример – задание окружности:

```
glBegin(GL_LINE_LOOP);
for (int i = 0; I < N; i++)
{
    float angle = 2 * M_PI * i / N;
    glVertex2f(cos(angle), sin(angle));
}
glEnd();
```

3. Рисование геометрических объектов. Точки, линии, многоугольники.

- Размеры точки - **glPointSize(GLfloat size)**
- Задание ширины линии в пикселях - **gflineWidth(GLfloatwidth)**
- Шаблон для линии - **glLineStipple(Glint factor, GLushort pattern)** Шаблон задается переменной pattern и растягивается в factor раз.
- Использование шаблонов линии - **glEnable(GL_UNE_STIPPLE)**, запрет - **glDisable(GL_LINE_ST1PPLE)**

3. Рисование геометрических объектов. Точки, линии, многоугольники.

- Передняя и задняя стороны многоугольника – **glPolygonMode(GLenum face, GLenum mode)**
- Значения параметра face:
  GL_FRONT_AND_BACK (обе стороны)
  GL_FRONT (лицевая сторона)
  GL_BACK (нелицевая сторона)
- Значения параметра mode
  GL_POINT (многоугольник – набор граничных точек)
  GL_LINE (многоугольник – граничная ломаная линия)
  GL_FILL (многоугольник – заполненная область)

3. Рисование геометрических объектов. Точки, линии, многоугольники.

- Пример: параллелепипед с ребрами, параллельными координатным осям, по диапазонам изменения x, y и z

```
glBegin ( GL_POLYGON );
glNormal3f ( 0.0, 0.0, 1.0 );
glVertex3f ( x1, y1, z2 );
glVertex3f ( x2, y1, z2 );
glVertex3f ( x2, y2, z2 );
glVertex3f ( x1, y2, z2 );
glEnd ();
glBegin ( GL_POLYGON );
glNormal3f ( 0.0, 0.0, -1.0 );
glVertex3f ( x2, y1, z1 );
glVertex3f ( x1, y1, z1 );
glVertex3f ( x1, y2, z1 );
glVertex3f ( x2, y2, z1 );
glEnd ();
glBegin ( GL_POLYGON );
glNormal3f ( -1.0, 0.0, 0.0 );
glVertex3f ( x1, y1, z1 );
glVertex3f ( x1, y1, z2 );
glVertex3f ( x1, y2, z2 );
glVertex3f ( x1, y2, z1 );
glEnd ();
glBegin ( GL_POLYGON );
glNormal3f ( 1.0, 0.0, 0.0 );
glVertex3f ( x2, y1, z2 );
glVertex3f ( x2, y1, z1 );
glVertex3f ( x2, y2, z1 );
glVertex3f ( x2, y2, z2 );
glEnd ();
glBegin ( GL_POLYGON );
glNormal3f ( 0.0, 1.0, 0.0 );
```

3. Рисование геометрических объектов. Трехмерные фигуры (функции).

auxSolidSphere(R) // сфера

auxSolidCube(width) // куб

auxSolidBox(width, height, depth) // коробка

auxSolidTorus(r,R) // тор

auxSolidCylinder(r,height) // цилиндр

auxSolidCone(r,height) // конус

auxSolidIcosahedron(width) // многогранники

auxSolidOctahedron(width)

auxSolidTetrahedron(width)

auxSolidDodecahedron(width)

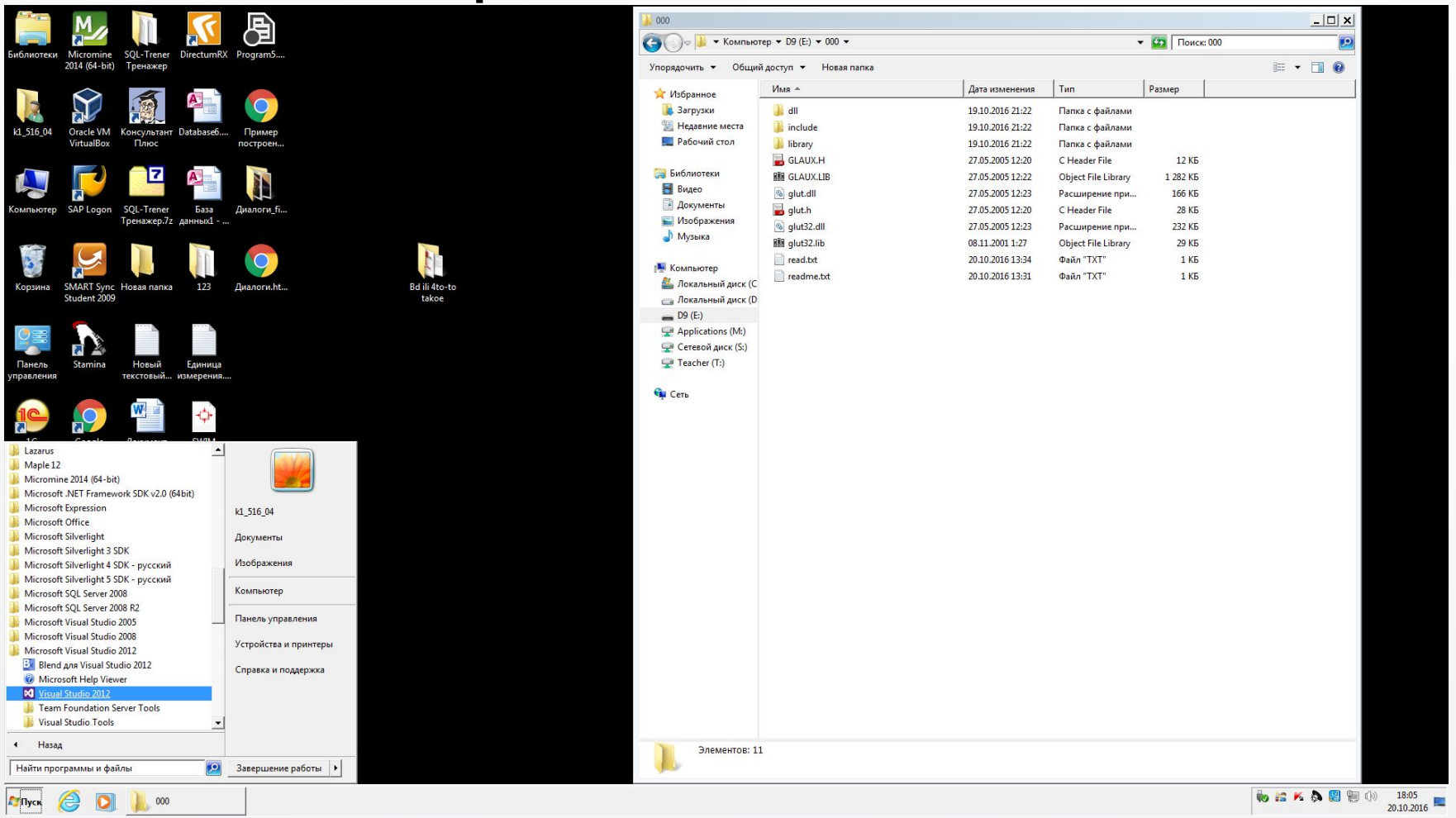
auxSolidTeapot(width) // рисует чайник

Для построения каркасных фигур вместо Solid необходимо использовать Wire.

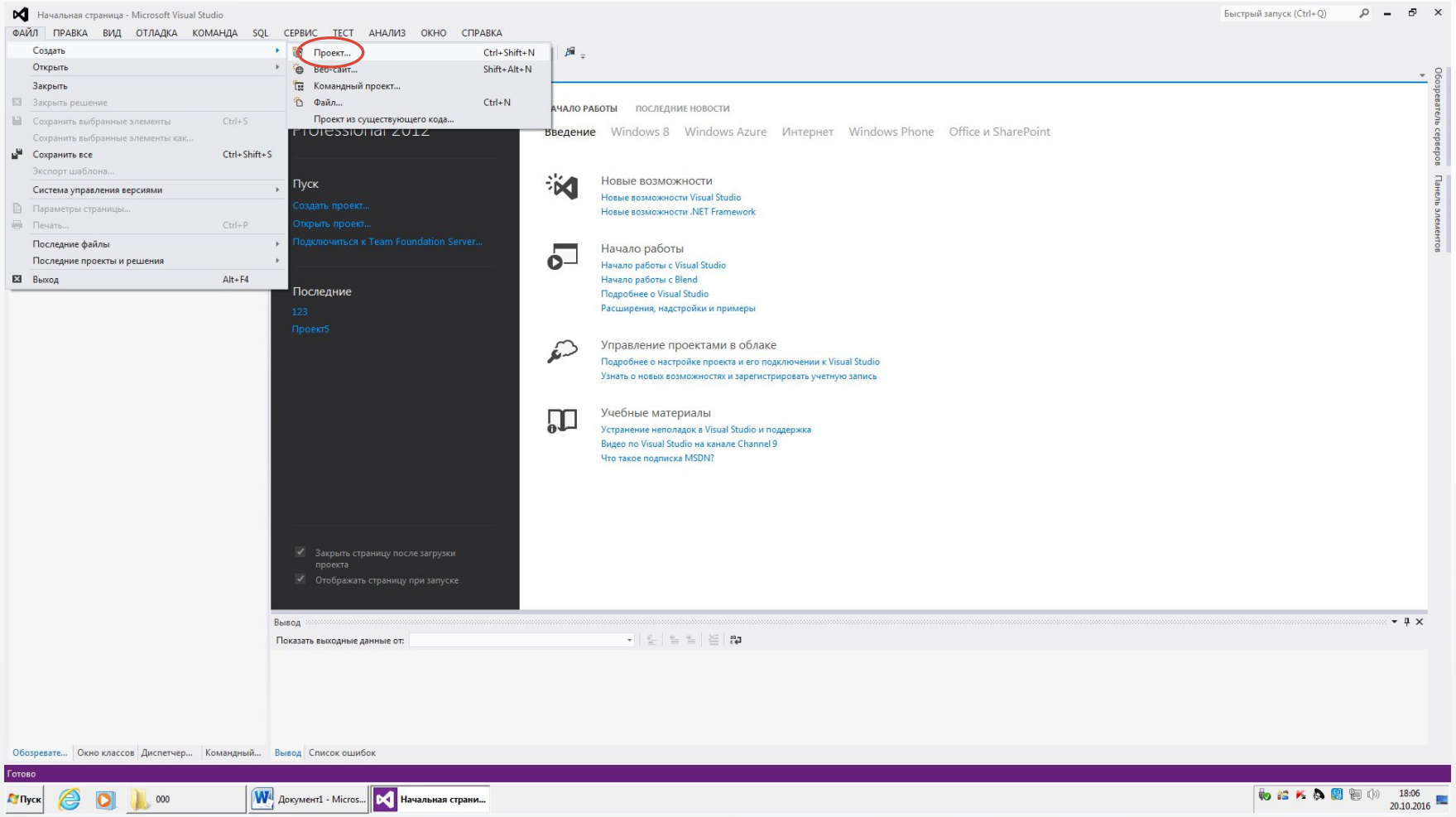
Пример: **auxWireCube(1)** // рисует каркасную модель куба.

4. Преобразование объектов в пространстве
5. Получение проекций
6. Задание моделей закрашивания
7. Освещение
8. Полупрозначность
9. Наложение текстуры

10. Практическая часть

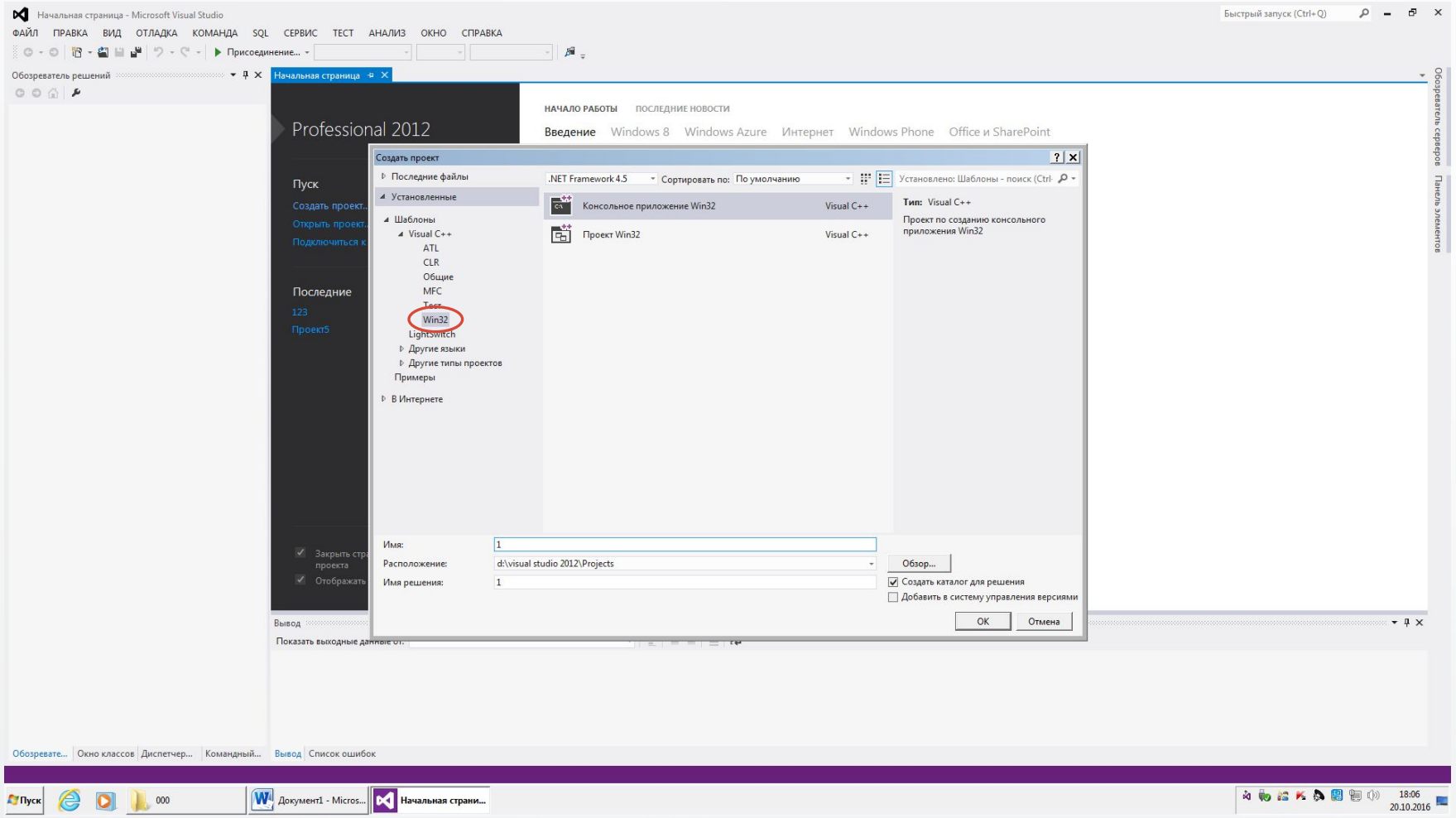


10. Практическая часть



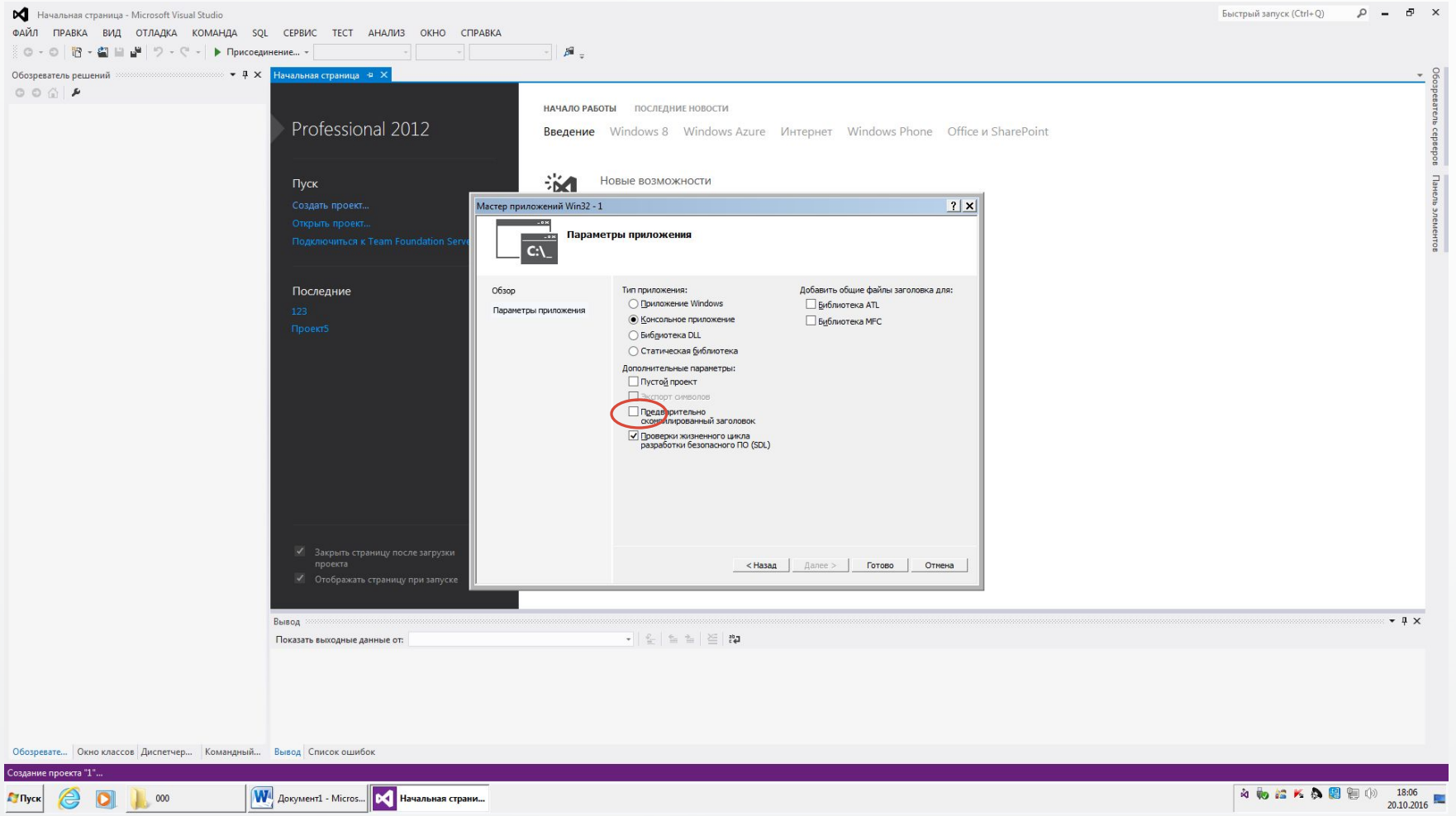
Визуализация результатов численных расчетов

10. Практическая часть



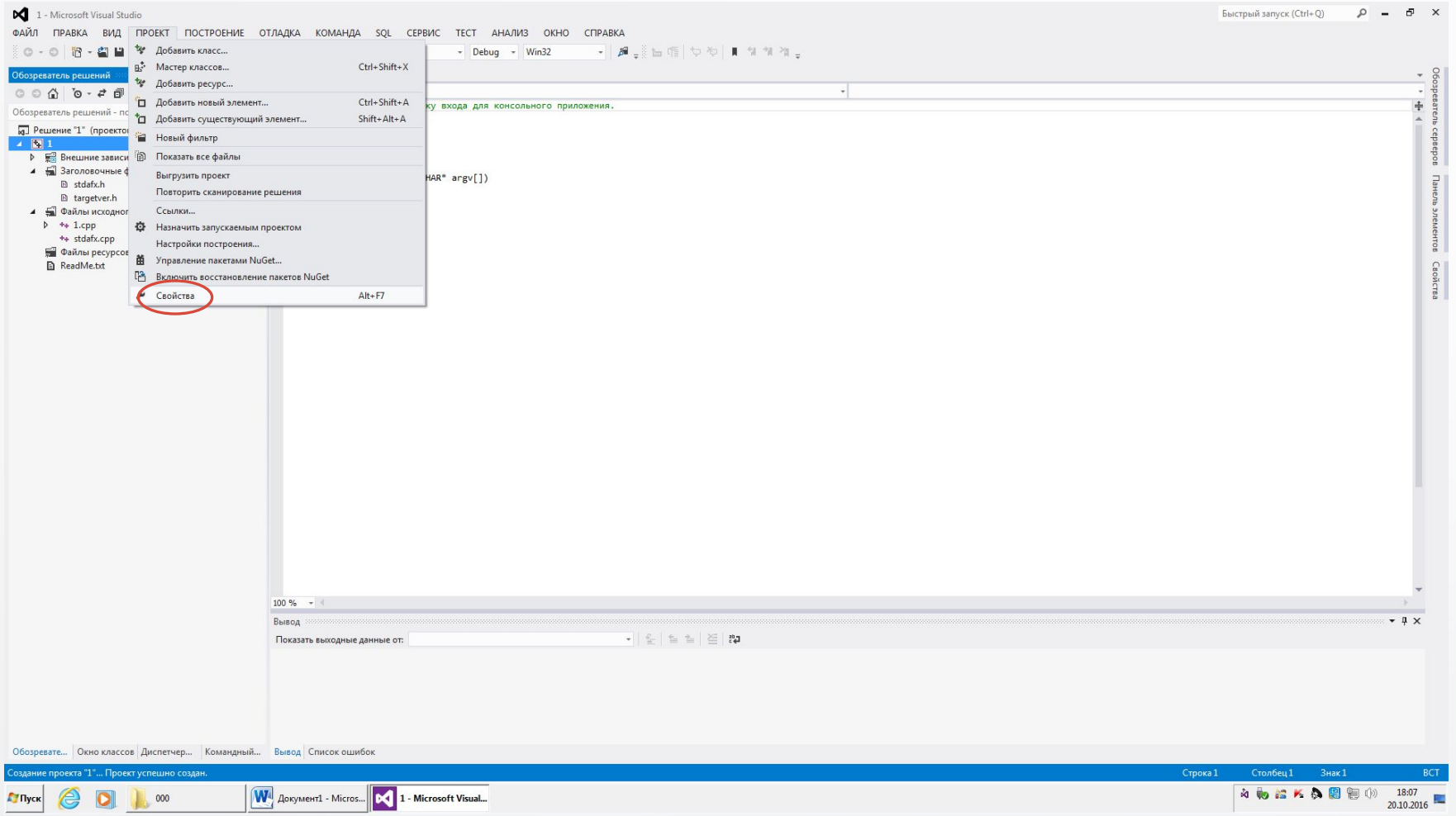
Визуализация результатов численных расчетов

10. Практическая часть



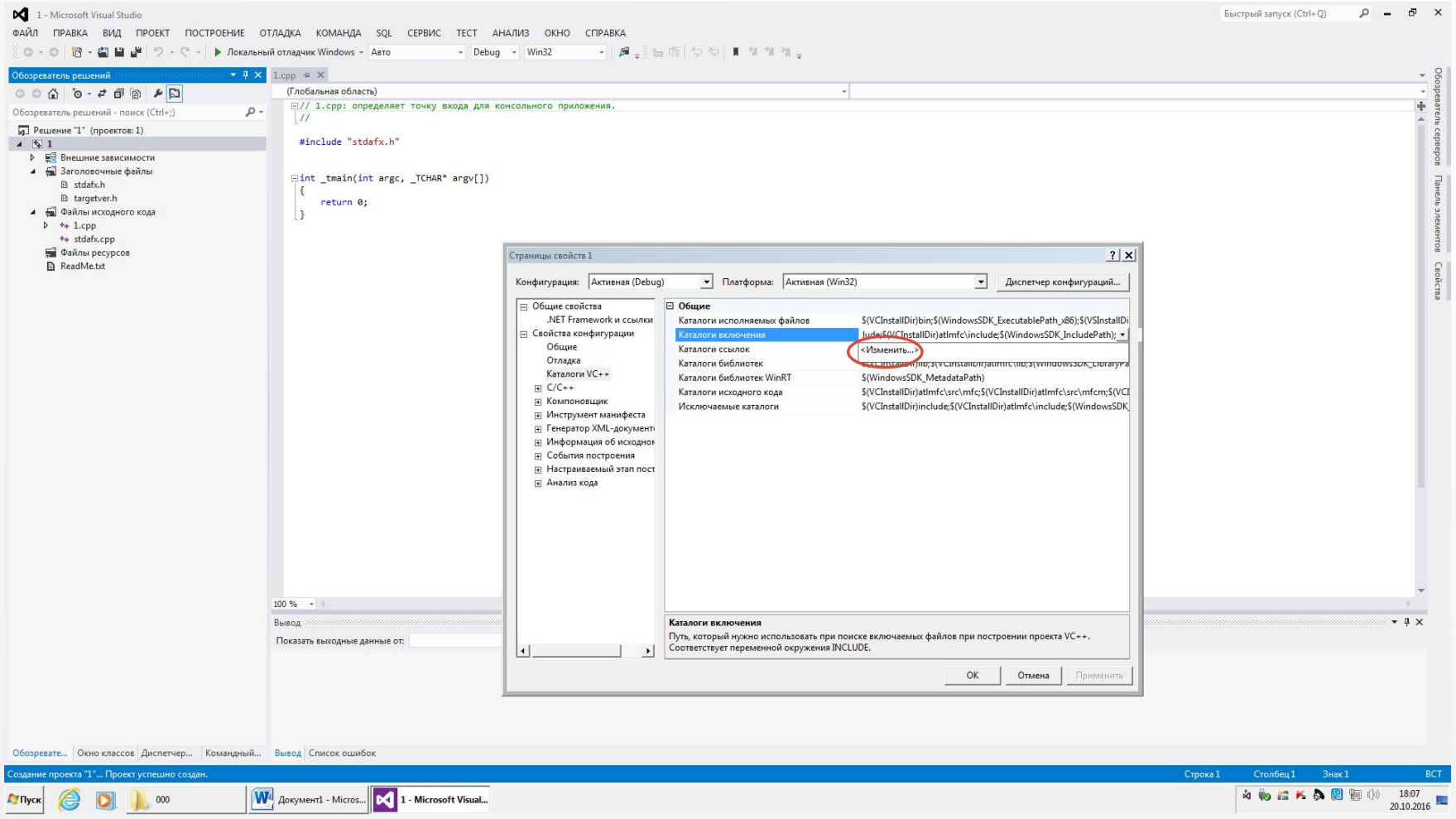
Визуализация результатов численных расчетов

10. Практическая часть



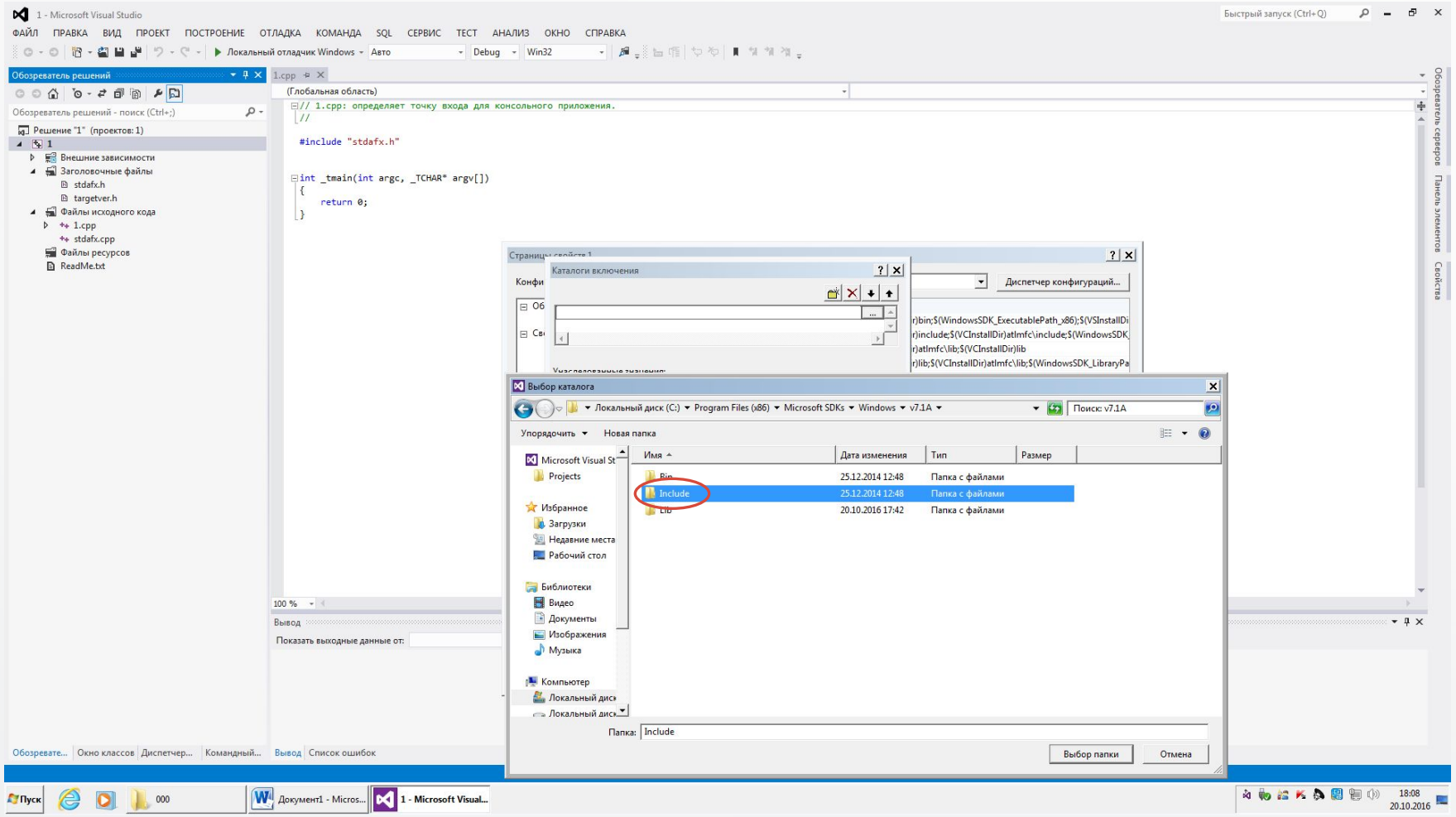
Визуализация результатов численных расчетов

10. Практическая часть



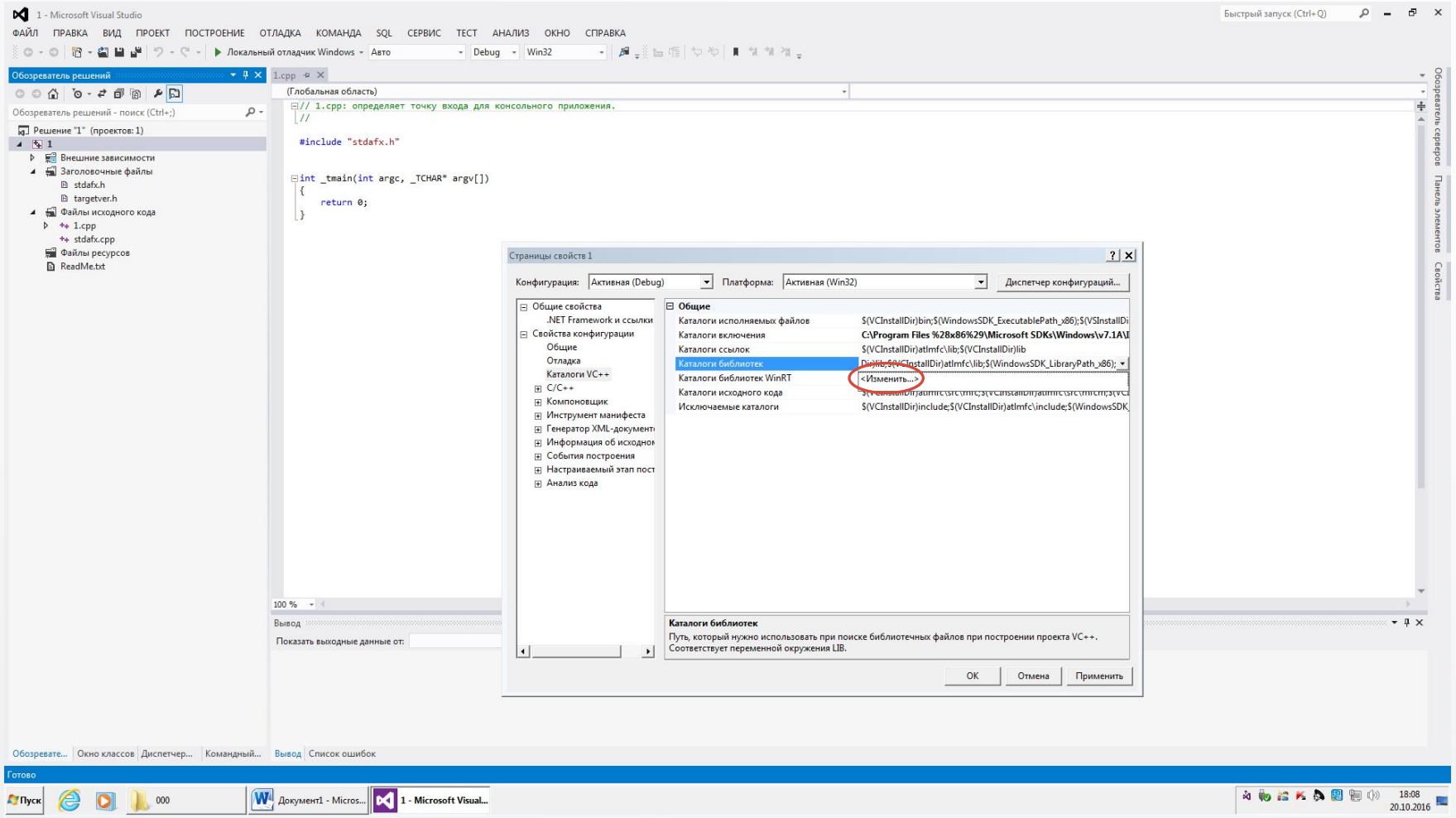
Визуализация результатов численных расчетов

10. Практическая часть



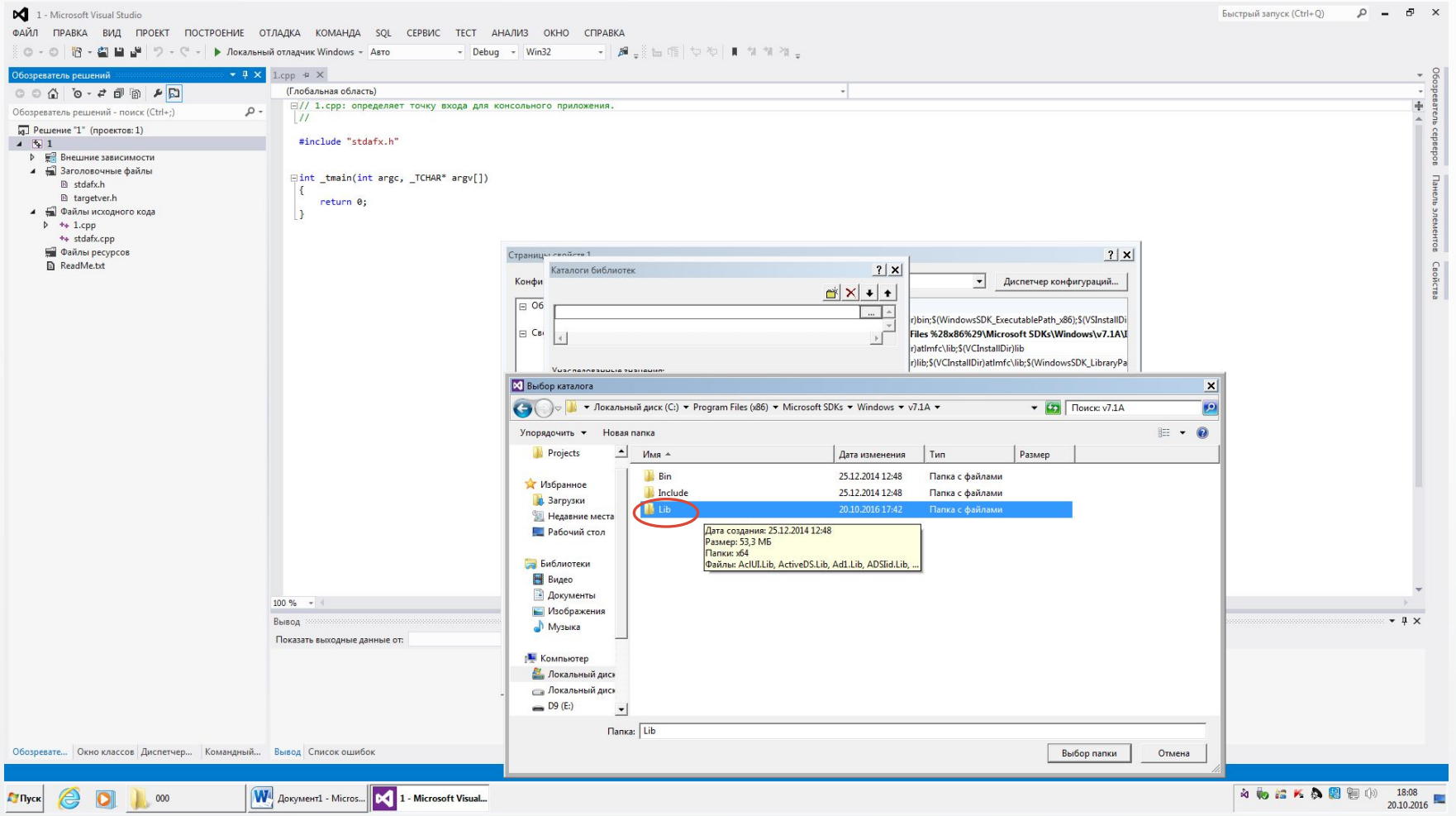
Визуализация результатов численных расчетов

10. Практическая часть



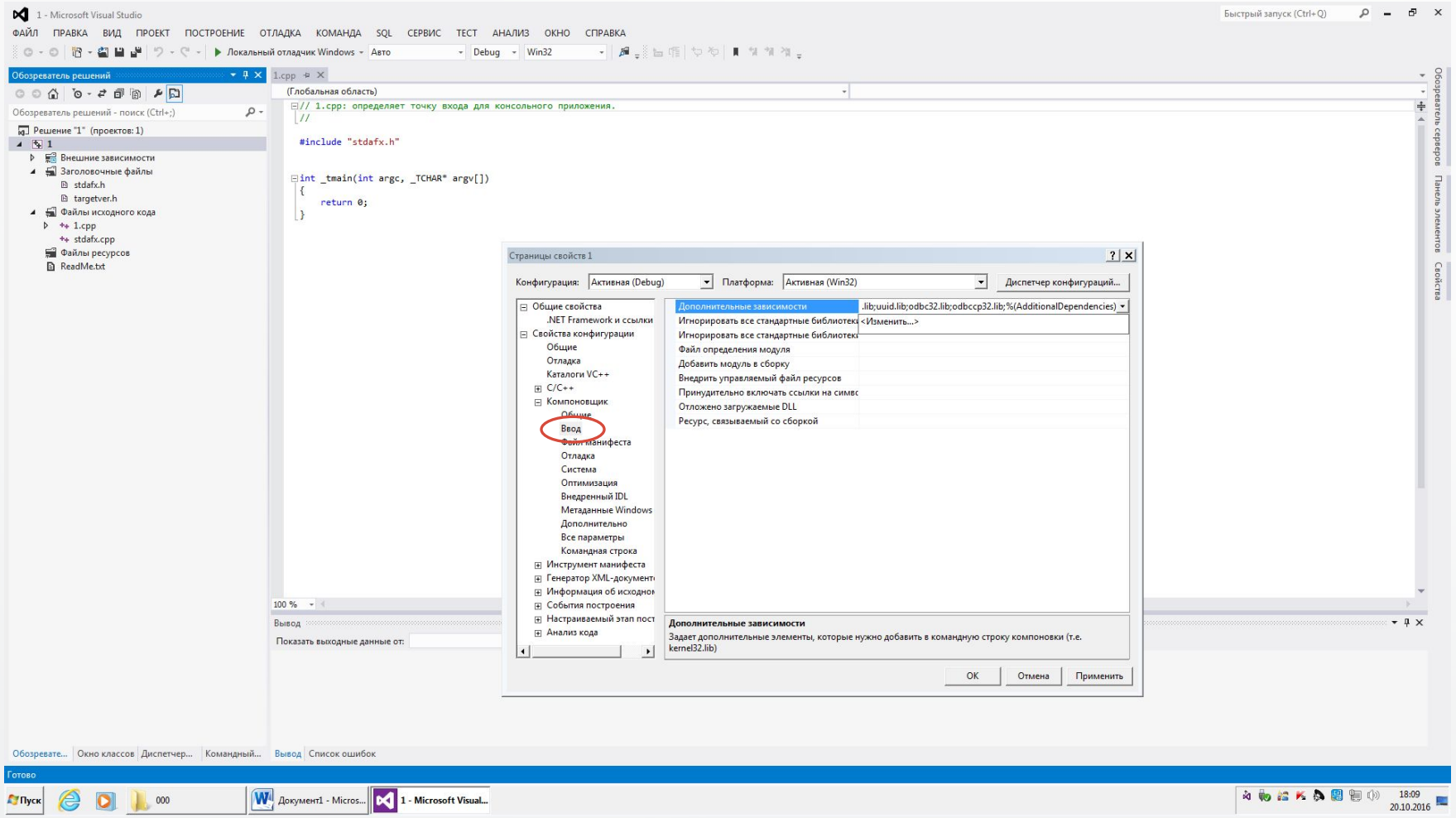
Визуализация результатов численных расчетов

10. Практическая часть



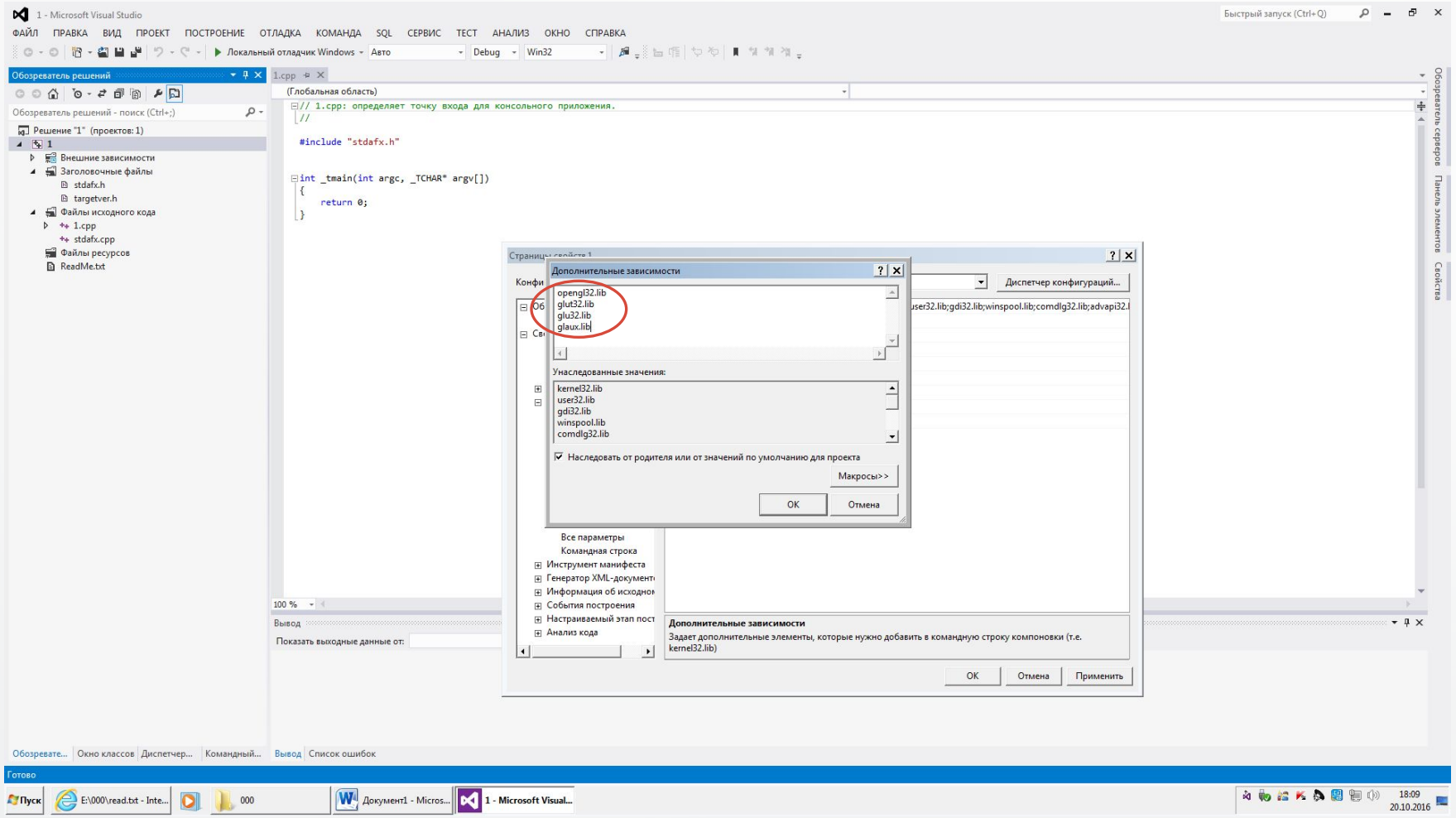
Визуализация результатов численных расчетов

10. Практическая часть

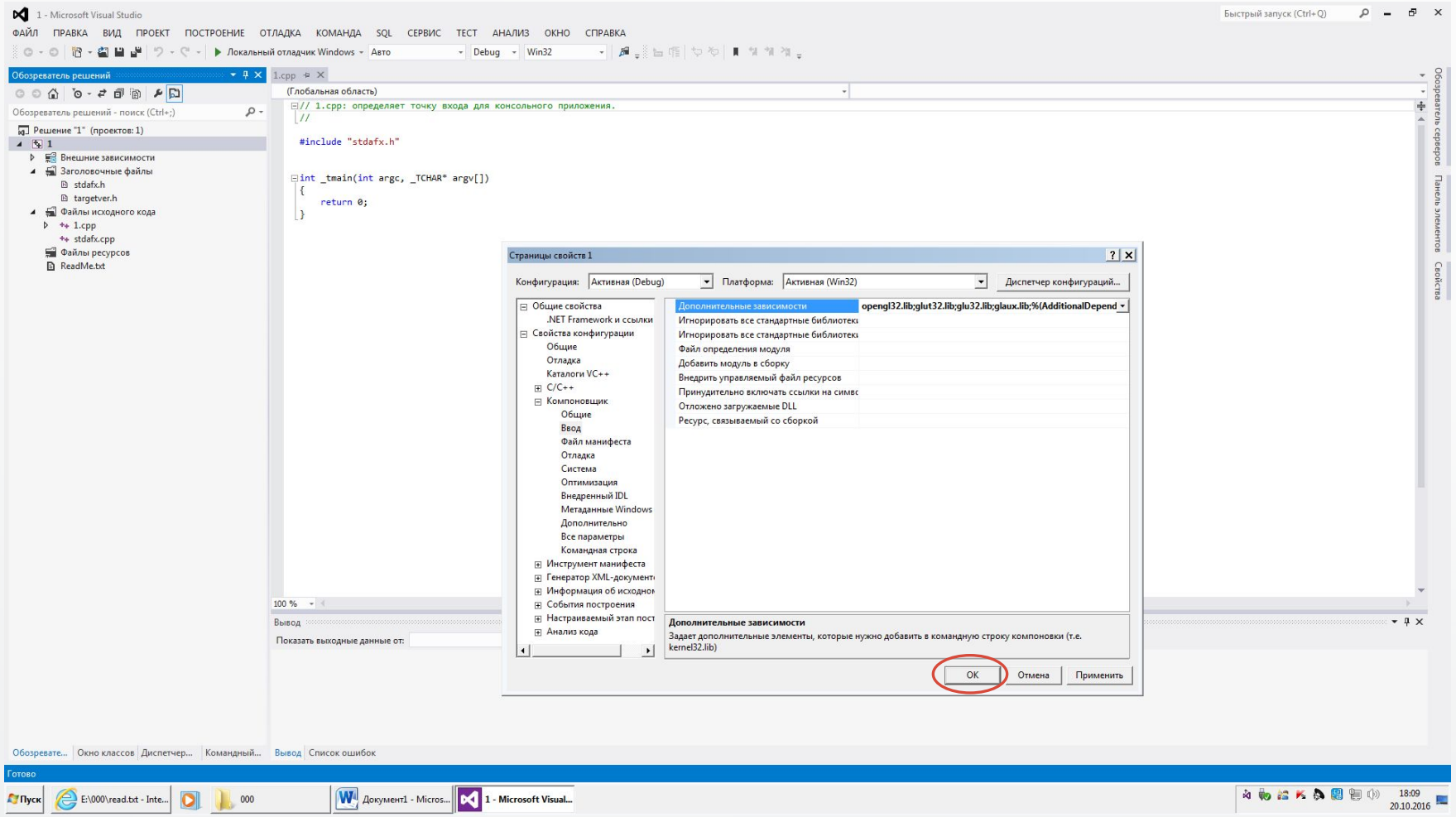


Визуализация результатов численных расчетов

10. Практическая часть



10. Практическая часть



Визуализация результатов численных расчетов

10. Практическая часть

Работаем!