DOI: https://doi.org/10.15688/jvolsu1.2017.2.4
УДК 517.957+514.752
ББК 32.973.26-018.2

УНИВЕРСАЛЬНЫЙ ПРОГРАММНЫЙ КОМПЛЕКС ДЛЯ РЕШЕНИЯ МНОГОМЕРНЫХ ВАРИАЦИОННЫХ ЗАДАЧ [1]

Елена Геннадиевна Григорьева

Кандидат физико-математических наук,
доцент кафедры компьютерных наук и экспериментальной математики,
Волгоградский государственный университет
e_grigoreva@mail.ru, e_grigoreva@volsu.ru, kiem@volsu.ru
просп. Университетский, 100, 400062 г. Волгоград, Российская Федерация

Владимир Александрович Клячин

Доктор физико-математических наук,
заведующий кафедрой компьютерных наук и экспериментальной математики,
Волгоградский государственный университет
klchnv@mail.ru, klyachin.va@volsu.ru, kiem@volsu.ru
просп. Университетский, 100, 400062 г. Волгоград, Российская Федерация

Алексей Александрович Клячин

Доктор физико-математических наук,
заведующий кафедрой математического анализа и теории функций,
Волгоградский государственный университет
klyachin-aa@yandex.ru, matf@volsu.ru
просп. Университетский, 100, 400062 г. Волгоград, Российская Федерация

Аннотация. В статье рассматривается задача численного расчета равновесных поверхностей произвольной топологии в классе полигональных моделей. Такого рода поверхности являются решением вариационной задачи на минимум функционала типа площади при наличии ограничений интегрального вида. В статье представлено программное решение таких вариационных

задач в двух вариантах. В первом — в виде пакета на языке Python, который реализован на базе пакета NumPy и интегрирован в среду 3D-моделирования Blender. Второй вариант предназначен главным образом для решения непараметрических вариационных задач и выполнен стандартными средствами C++.

Ключевые слова: экстремальная поверхность, кусочно-линейная аппроксимация, триангуляция, пакет NumPy, краевая задача.

Введение

Одним из широко распространенных методов решения краевых задач математической физики является вариационный метод, основанный на минимизации некоторого интегрального функционала в том или ином классе функций. Например, при заданных краевых условиях можно свести интегрирование уравнений статической теории упругости к отысканию минимума потенциальной энергии деформации упругого тела. Другим примером может служить задача поиска поверхности, имеющей минимальную площадь и покрывающей фиксированный или максимальный объем. Данная задача возникает в практике проектирования современных сооружений в качестве покрытий или ограждений [7].

Известно, что метод конечных элементов (МКЭ), использующий построение триангуляции расчетной области, является одним из самых применяемых методов приближенного решения вычислительных задач математической физики, в том числе и при решении упомянутых вариационных задач. Стоит назвать такие популярные программные комплексы, как COMSOL, ANSYS, ИСПА, «Диана», COSMOS, MSC/NASTRAIN, SAMSAF и другие, повсеместно используемые в инженерных расчетах, в которых алгоритмы имитационного моделирования существенно используют МКЭ. В настоящей работе мы приводим свою реализацию вариационного метода, основанного на аппроксимации кусочно-линейными функциями и поверхностями. Мы исходим из идеи создания универсальной программной системы, основная часть которой не должна зависеть от используемой триангуляции, интегрального функционала, расчетной области и граничных условий.

1. Вычисление экстремальных поверхностей произвольной топологии

Пусть $\Phi : \mathbb{R}^3 \to \mathbb{R}$ — положительная, однородная степени 1, выпуклая функция такая, что $\Phi(\xi) = \Phi(-\xi)$. Рассмотрим ограниченную область $D \subset \mathbb{R}^3$ с кусочно-гладкой границей и пусть $\varphi(x) \in C(\overline{D})$ — положительная непрерывная функция. Для гладкой поверхности $M \subset \mathbb{R}^3$ такой, что $M \subset \partial D$ рассмотрим величину

$$F(M) = \int\limits_M (\Phi(\xi) + \alpha(x))\, dS,$$

где ξ — вектор единичной нормали к поверхности M, $\alpha(x)$ — положительная функция. Нас будет интересовать решение вариационной задачи на минимум этого значения при условии, что граница ∂M фиксирована, а интеграл

$$\int\limits_D \varphi(x)dx \tag{1}$$

остается неизменным.

Поверхность $\mathcal{M}$ будем аппроксимировать поверхностью $\mathcal{M}$, которая задается конечным множеством точек $P = \{P_i \in \mathbb{R}^3, i = 1, ..., N\}$ и совокупностью треугольников $T = \{T_j, j = 1, ..., K\}$. Каждый треугольник T_j однозначно определяется тремя целыми числами $k_j, l_j, m_j \in \{1, ..., N\}$, такими, что точки $P_{k_j}, P_{l_j}, P_{m_j}$ являются вершинами треугольника T_j.

Если $\mathcal{M}$ — кусочно-линейная аппроксимация поверхности M, то приближенное значение введенного функционала будет вычисляться согласно формуле

$$\tilde{F}(\mathcal{M}) = \sum_{j=1}^{K} \Phi(\xi_j)|T_j| + \alpha(p_j^*)|T_j|, \tag{2}$$

где $\mathcal{M}$ представляет собой объединение треугольников T_j; ξ_j — нормаль к плоскости этих треугольников, а $|T_j|, p_j^*$ — площади и центры треугольников соответственно. Поверхность $\mathcal{M}$ можно рассматривать как точку P в пространстве LM размерности $3N$, где N — число вершин на $\mathcal{M}$. Если задать в каждой точке-вершине поверхности вектор h, то этим векторам будет соответствовать вектор в LM. Тогда оказывается, если функционал рассмотреть как числовую функцию : $LM \to \mathbb{R}$, то имеет место формула

$$\left(\frac{\partial \tilde{F}}{\partial h}(P)\right)^i = \frac{1}{2}\langle \sum_{j=1}^{k} (\nabla\Phi(2\xi_j|T_j|)) \times l_j, h\rangle + \frac{1}{2}\langle \sum_{j=1}^{k} \alpha(p_j^*)\xi_j \times l_j, h\rangle +$$

$$+ \frac{1}{3}\langle \sum_{j=1}^{k} \nabla\alpha(p_j^*)|T_j|, h\rangle,$$

где сумма идет по всем треугольникам, имеющим вершину $p_i \in \mathcal{M}$, ξ_j, как и выше, обозначает нормаль к соответствующему треугольнику и l_j — вектор, соединяющий вершины треугольника, не совпадающие с p_i таким образом, что эта вершина при обходе остается слева. Заметим, что если для j-го треугольника обозначить через a_j, b_j векторы, направленные вдоль сторон треугольника из вершины p_i, то эта формула может быть переписана в более простом виде, с использованием однородности функции $\Phi(\xi)$

$$\left(\frac{\partial \tilde{F}}{\partial h}(P)\right)^i = \frac{1}{2}\langle \sum_{j=1}^{k} \nabla\Phi(a_j \times b_j) \times l_j, h\rangle.$$

Пример 1. Рассмотрим случай

$$\Phi(\xi) = |\xi|, \quad \alpha(x) \equiv 0, \quad \varphi(x) \equiv 0.$$

Тогда, как нетрудно посчитать, получим

$$\left(\frac{\partial \tilde{F}}{\partial h}(P)\right)^i = \frac{1}{2}\langle \sum_{j=1}^{k} \xi_j \times l_j, h\rangle. \tag{3}$$

Этот случай соответствует вычислению минимальных кусочно-линейных поверхностей.

Пример 2. Рассмотрим случай

$$\Phi(\xi) = |\xi|, \quad \alpha(x) \equiv 0, \quad \varphi(x) \equiv 1.$$

Тогда, как нетрудно посчитать, получим

$$\left(\frac{\partial \tilde{F}}{\partial h}(P)\right)^i = \frac{1}{2}\langle \sum_{j=1}^{k} \xi_j \times l_j, h\rangle. \tag{4}$$

Этот случай соответствует моделированию кусочно-линейных поверхностей постоянной средней кривизны — равновесных поверхностей раздела сред с постоянным давлением без учета гравитационных сил [10; 12].

Пример 3. Рассмотрим случай

$$\Phi(\xi) = |\xi|, \quad \alpha(x) \equiv g \cdot x_3, \quad \varphi(x) \equiv 1.$$

Тогда, как нетрудно посчитать, получим

$$\left(\frac{\partial \tilde{F}}{\partial h}(P)\right)^i = \frac{1}{2}\langle \sum_{j=1}^{k} \xi_j \times l_j, h\rangle + \frac{1}{2}\langle \sum_{j=1}^{k} \langle p_j^*, e_3\rangle \xi_j \times l_j, h\rangle + \tag{5}$$

$$+ \frac{1}{3}\langle \sum_{j=1}^{k} e_3 |T_j|, h\rangle.$$

Этот случай соответствует моделированию капиллярных кусочно-линейных поверхностей — равновесных поверхностей раздела сред с постоянным давлением с учетом гравитационных сил [10; 12].

Для итерационного процесса градиентного спуска мы также должны учесть, что интеграл (1) остается неизменным. Ясно, что из постановки вариационной задачи следует, что это условие будет выполнено при малых деформациях $h(x)$ поверхности $\mathcal{M}$, если

$$\int_M \varphi(x) h(x) dS = 0.$$

Это условие для кусочно-линейной поверхности можно записать в виде

$$\sum_{i=j}^{K} h(p_j^*)|T_j| = 0,$$

где p_j^* — как и выше, центр треугольника T_j. Полученное соотношение представляет собой линейное уравнение для значений смещений h в вершинах кусочно-линейной поверхности $\mathcal{M}$. В пространстве LM множество решений этого уравнения задает гиперплоскость C, ортогональную проекцию на которую обозначим через $\pi_C : LM \to C$. Тогда формулы итерационного процесса можно записать в виде

$$P^{k+1} = P^k - \varepsilon_k \pi_C(\nabla \tilde{F}(P^k)),$$

где величины ε_k определяют смещения вдоль направления, обратного к направлению градиента $\nabla \tilde{F}(P^k)$.

2. Объектно-ориентированная модель вычислений

Реализацию вычислений было решено выполнить, преследуя две цели. С одной стороны, весь программный комплекс должен быть интегрирован в какую-либо систему визуализации, а с другой стороны, желательно было получить код вычислений, по максимуму не зависящий от такой интеграции. В связи с этим была выбрана объектно-ориентированная модель организации вычислений на базе шаблона проектирования классов «мост» [9; 11]. Этот шаблон используется в случаях, когда требуется отделить абстракцию от ее реализации так, чтобы то и другое можно было изменять независимо. В нашей задаче было решено реализацию интегрировать в пакет Blender и выполнить на языке программирования Python. Это, в частности, позволит задавать топологию искомой экстремальной поверхности, а также моделировать искомую поверхность в интерактивном режиме работы в окне 3D-вида программы Blender. Кроме этого, реализация основана на использовании числовых массивов модуля NumPy. Этот модуль представляет собой расширение языка программирования Python, реализующее множество различных операций при работе с многомерными массивами данных. Несмотря на то что этот пакет предназначен для использования его в контексте интерпретируемого языка, тем не менее массивы NumPy являются Си-подобными [8]. Так что эти массивы могут управляться стандартными инструкциями языка Python. Удобство использования массивов NumPy, в частности, состоит и в том, что для этих массивов перегружены арифметические операции и математические функции.

Опишем структуру программного комплекса (см. рис. 1). Весь код организован в пакет MinimalSurface, содержащий несколько модулей с требуемыми классами. Некоторые модули, ранее описанные в работе [2], были выбраны за основу и использованы для решения задачи моделирования экстремальных поверхностей:

```
ndimvar.py,
geometry.py,
triangulation.py.
```

Модуль ndimvar.py реализует абстрактный класс **NDimVar**, инкапсулирующий обобщенный итерационный процесс, в частности итерационный процесс метода градиентного спуска. Приведем отрывок кода, который реализует отдельную итерацию градиентного спуска.

```
def iteration(s,x,k):
        if s.sublin:          #  если задача на условный экстремум
            s.create_subLin() #  на каждом шаге
                              #  строится гиперплоскость C
            return x-s.epsilon(k)*s.proj(s.X(x))
                                      # вычисляем новую точку
                                      # с учетом проекции
                                      # на гиперплоскость C
        else:
            return x-s.epsilon(k)*s.X(x) # вычисляем новую точку
                                         # без учета проекции
                                         # на гиперплоскость C
```

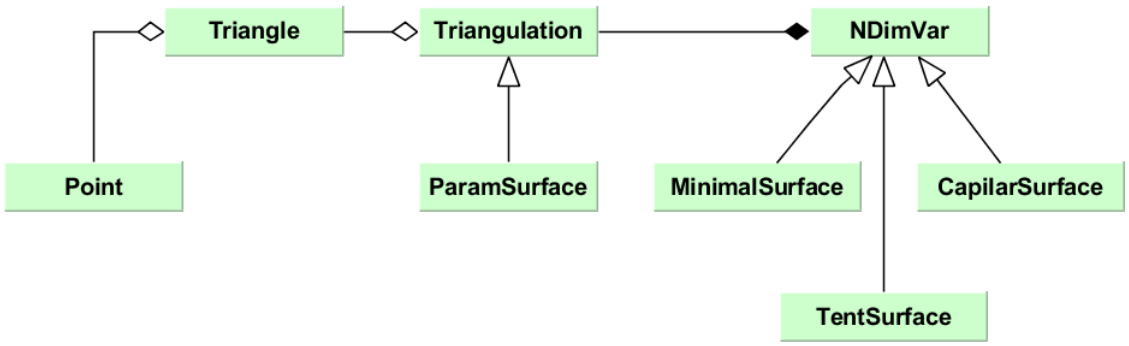

Рис. 1. Диаграмма вычислительных классов

В подклассах класса **NDimVar** требуется реализация метода вычисления градиента минимизируемой функции, который зависит от функционала $\tilde{F}(\mathcal{M})$. Модуль geometry.py реализует классы для вычисления геометрических характеристик граней кусочно-линейной поверхности: длины сторон, площади и углы треугольников. Модуль triangulation.py реализует базовый класс **Triangulation** для представления в системе вычислений кусочно-линейной поверхности, а также автоматизирует процесс вычисления граничных точек поверхности [1]. Дополнительно к перечисленным модулям реализованы модули minsquare.py, tentsurface.py. В модуле minsquare.py реализованы классы **MinimalSurface** и **CapillarSurface**, производные класса **NDimVar**, в которых переопределены виртуальные методы вычисления по формулам (3)–(5). В модуле tentsurface.py представлен класс **MinimalTent** для аналогичных вычислений так называемых тентовых поверхностей, как экстремалей функционала, соответствующего функциям

$$\Phi(\xi) = |\xi|, \quad \alpha(x) = \lambda_0 + \lambda \cdot x_3, \quad \varphi(x) \equiv 0.$$

Кроме перечисленных выше классов, непосредственно задействованных в вычислениях, реализованы классы интеграции в пакет Blender и предоставляющие взаимодействие среды Blender с указанными классами в соответствии с шаблоном «мост». Такая схема позволяет не только изменять и совершенствовать интерфейс пользователя, но и интегрировать классы вычислений в другие системы визуализации, например, в систему matplotlib, независимо от перечисленных выше классов. Соответствующие классы реализованы в модуле panel_addon.py. В этом модуле содержатся классы **MinimalSurfacePanel, FindBoundary, FindMinSurface, FindCMCSurface, FindCapillarSurface, FindTentSurface, Settings**.

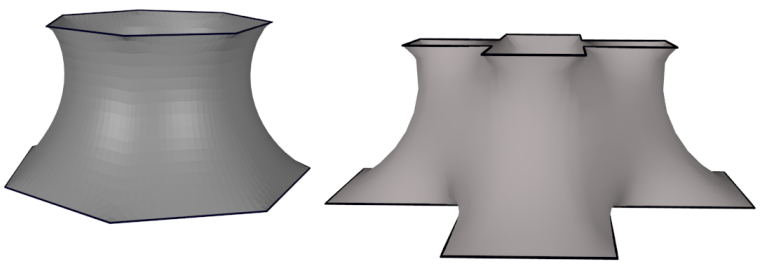

Рис. 2. Примеры результата расчета минимальной поверхности

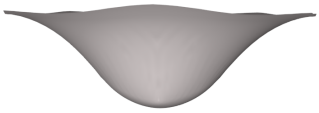

Рис. 3. Пример результата расчета капиллярной поверхности

Примеры визуализации результатов расчета равновесных поверхностей представлены на рисунках 2 и 3.

3. Решение непараметрических вариационных задач

Будем рассматривать следующую вариационную задачу

$$\int\limits_{\Omega} G(x,\vec{u},D\vec{u})dx + \int\limits_{\Gamma} F(x,\vec{u})dS \to \min \quad \vec{u}|_{\partial\Omega\setminus\Gamma} = \vec{\varphi}|_{\partial\Omega\setminus\Gamma},$$

где $G = G(x_1,...,x_n,u^1,...,u^p,\vec{\xi}^1,...,\vec{\xi}^p)$, $F = F(x_1,...,x_n,u^1,...,u^p)$, $\vec{\xi}^i = (\xi^i_1,...,\xi^i_n), i = 1,...,p$, $\vec{u} = (u^1,...,u^p) : \Omega \to \mathbf{R}^p$ и Ω — заданная область в $\mathbf{R}^n$. Приведем некоторые примеры.

Пример 4. Задача нахождения минимальной поверхности, натянутой на заданный контур, формулируется так

$$\iint\limits_{\Omega} \sqrt{1+u_x^2+u_y^2}\,dxdy \to \min, \quad u|_{\partial\Omega} = \varphi.$$

В этом случае мы имеем $G = \sqrt{1+(\xi^1_1)^2+(\xi^1_2)^2}$ для $p = 1$ и $n = 2$.

Пример 5. Рассмотрим следующую краевую задачу

$$\frac{\partial}{\partial x}\left(\frac{f_x}{\sqrt{1+f_x^2+f_y^2}}\right) + \frac{\partial}{\partial y}\left(\frac{f_y}{\sqrt{1+f_x^2+f_y^2}}\right) = \varkappa f \quad \text{для} \quad (x,y)\in\Omega, \tag{6}$$

$$\left.\frac{\langle\nabla f,\nu\rangle}{\sqrt{1+|\nabla f|^2}}\right|_{\partial\Omega} = \psi, \tag{7}$$

где Ω — область в $\mathbf{R}^2$ с кусочно-гладкой границей $\partial\Omega$, $\nu = (\nu_1,\nu_2)$ — вектор единичной внешней нормали в точках границы $\partial\Omega$, где он существует, $f \in C^2(\Omega)\cap C^1(\overline{\Omega})$, постоянная $\varkappa > 0$ и ψ — заданная непрерывная функция на $\partial\Omega$. Известно [12], что решение f описывает капиллярную поверхность, находящуюся в равновесии, над областью Ω с заданным углом контакта γ (в случае, когда $\psi = \cos\gamma$) между этой поверхностью и стенкой капиллярной трубки. Данная задача эквивалентна минимизации интеграла

$$E(g) = \iint\limits_{\Omega} \sqrt{1+g_x^2+g_y^2}\,dxdy + \frac{\varkappa}{2}\iint\limits_{\Omega} g^2(x,y)\,dxdy - \int\limits_{\partial\Omega} g\psi dS.$$

В этом случае $G = \sqrt{1+(\xi^1_1)^2+(\xi^1_2)^2} + \frac{\varkappa}{2}(u^1)^2$, $F = u^1\psi$, $p = 1$, $n = 2$ и $\Gamma = \partial\Omega$.

Пример 6. Задача определения тензора деформации упругого тела сводится к решению системы уравнений

$$(\lambda + H)\frac{\partial e}{\partial x_i} + H\Delta u^i + X_i = 0,\ i = 1, 2, 3,$$

где λ, H — постоянные, определяемые упругой средой, $e = \frac{\partial u^1}{\partial x_1} + \frac{\partial u^2}{\partial x_2} + \frac{\partial u^3}{\partial x_3}$, $X = (X_1, X_2, X_3)$ — вектор внешних сил. Решение различных краевых задач для этой системы может быть сведено к минимизации функционала

$$V(u^1, u^2, u^3) = \iint\limits_{\partial\Omega} \sum_{i=1}^{3} \psi_i u^i dS +$$

$$+ \frac{1}{2}\iiint\limits_{\Omega} \left(\lambda \left(\sum_{i=1}^{3} \frac{\partial u^i}{\partial x_i} \right)^2 + 2H \sum_{i=1}^{3} \left(\frac{\partial u^i}{\partial x_i} \right)^2 + H \sum_{i \neq j} \left(\frac{\partial u^i}{\partial x_j} + \frac{\partial u^j}{\partial x_i} \right)^2 + \sum_{i=1}^{3} u^i X_i \right) dx_1 dx_2 dx_3.$$

Так же как и в первых двух примерах, не сложно выписать функции G и F:

$$G = \frac{1}{2}\left(\lambda \left(\sum_{i=1}^{3} \xi_i^i \right)^2 + 2H \sum_{i=1}^{3} \left(\xi_i^i\right)^2 + H \sum_{i \neq j} \left(\xi_i^j + \xi_j^i \right)^2 + \sum_{i=1}^{3} u^i X_i \right),$$

$$F = \sum_{i=1}^{3} \psi_i u^i.$$

4. Описание входных данных

Мы предполагаем, что расчетная область представляет собой набор треугольников $\{T_k\}_{k=1}^N$, образующих триангуляцию. Обозначим через $P_1, ..., P_m$ все вершины этих треугольников. Будем рассматривать класс кусочно-линейных функций $u : \Omega \to \mathbf{R}$, которые являются непрерывными в Ω и в каждом треугольнике T_k представляют собой функцию вида $u(x, y) = a_k x + b_k y + c_k$. Каждая такая функция однозначно определяется набором ее значений $u_1, ..., u_m$ в соответствующих вершинах $P_1, ..., P_m$ триангуляции.

Теперь дадим описание входных файлов, содержащих вершины триангуляции, ее треугольники и функции, задающие краевые условия. Итак, узлы триангуляции хранятся в отдельном файле `points.pnt`, в котором первые 8 байт отводятся под количество точек, а дальше каждая точка занимает 20 байт — две координаты и информация о принадлежности границе:

Номер точки	Координата x **(8 байт)**	Координата y **(8 байт)**	Принадлежность границе (0 — внутренняя точка $\neq 0$ — граничная точка **(4 байта)**
0	x_0	y_0	b_0
1	x_1	y_1	b_1
2	x_2	y_2	b_2
...	...	...	...

Триангуляция этих точек хранится в другом файле `triangles.trg`, который имеет следующую структуру: вначале 8 байт отводится под число треугольников, а следующие данные можно изобразить в виде таблицы:

Номер треугольника	Номер точки, соответствующей 1-й вершине **(8 байт)**	Номер точки соответствующей 2-й вершине **(8 байт)**	Номер точки соответствующей 3-й вершине **(8 байт)**
0	P_0^1	P_0^2	P_0^3
1	P_1^1	P_1^2	P_1^3
2	P_2^1	P_2^2	P_2^3
...	...	...	...

Нужно заметить, что номера точек соответствуют их номерам из первого файла, содержащего все вершины триангуляции. Далее, так как кусочно-линейная функция однозначно определяется ее значениями в вершинах $P_1, ..., P_m$, то мы в отдельном файле `function.val` храним эти значения. В начале этого файла записывается количество точек, а затем идут значения функции в том же порядке, что и точки в первом файле `points.pnt`. В такого типа файлах мы записываем значения функций, задающих различные краевые условия (Дирихле, Неймана, смешанные условия и др.) и результаты вычислений.

Для случая трехмерных вариационных задач структура входных файлов аналогичная. Например, файл с вершинами триангуляции будет выглядеть так: первые 8 байт отводятся под количество точек, а дальше хранятся декартовы координаты вершин триангуляции.

Так как в случае пространственных вариационных задач триангуляция области состоит из тетраэдров, то соответствующим образом меняется и структура файла `triangles.trg`.

5. Задание функционала и его минимизация

Минимизация функционала осуществляется с помощью метода градиентного спуска и метода покоординатного спуска с применением алгоритма «золотого сечения» для одномерной минимизации. Отметим, что вершины триангуляции являются объектами класса `SQPoint`, а треугольники (тетраэдры) — объектами класса `SQTreug` (`SQTetr`). Для задания функционала достаточно определить функции G и F. Для функции G создан класс `SQIntegrand`, имеющий виртуальные функции:

```
IntG(SQPoint, SQPoint, double),
Gu(SQPoint, SQPoint, double),
gradG(SQPoint, SQPoint, double),
IntG2(SQPoint point,SQPoint px, SQPoint py, double u, double v),
gradGu2(SQPoint point, SQPoint px, SQPoint py, double u, double v),
gradGv2(SQPoint point, SQPoint px, SQPoint py, double u, double v),
Gu2(SQPoint point, SQPoint px, SQPoint py, double u, double v),
Gv2(SQPoint point, SQPoint px, SQPoint py, double u, double v).
```

Все функции этого класса реализуются в наследуемых классах, которые и определяют конкретный функционал. Например, если в качестве минимизируемого функционала взять площадь поверхности

$$\iint\limits_{\Omega} \sqrt{1+u_x^2+u_y^2}\,dxdy,$$

то мы определяем класс

```
class SQSurfArea : public SQIntegrand
{
public:
       double IntG(SQPoint point,SQPoint vect, double u)
      {
      double value;
      value=sqrt(1+vect.x*vect.x+vect.y*vect.y);
      return value;
      }
       double Gu(SQPoint point,SQPoint vect, double u)
      {
       return 0.0;
      }
  SQPoint gradG(SQPoint point, SQPoint vect, double u)
      {
       SQPoint grd(0.0,0.0);
       grd.x=vect.x/sqrt(1+vect.x*vect.x+vect.y*vect.y);
       grd.y=vect.y/sqrt(1+vect.x*vect.x+vect.y*vect.y);
       return grd;
      }
};
```

Функция `gradG()` является градиентом интегранда `IntG` по переменным ξ_i^j, а функция `Gu()` представляет собой производную интегранда по переменным u^i. Аналогично для случая, если поиск минимума осуществляется в классе отображений, задаваемых двумя функциями $u = u(x, y)$, $v = v(x, y)$. Для этого предназначены функции `IntG2()`, `gradGu2()`, `gradGv2()`, `Gu2()`, `Gv2()`.

Для задания функции F создан класс `SQIntegrandB`

```
class SQIntegrandB
{
      public:
      virtual double IntF(SQPoint point, double u)
      {
      return 0.0;
      }
};
```

Для минимизации функционала используются две функции `minimum_func_mix()` и `minimum_gold_sect()`, которые осуществляют один шаг в алгоритме градиентного спуска и метода «золотого сечения» соответственно. Каждая из этих функций имеет одинаковый набор аргументов. Например, рассмотрим функцию

```
minimum_func_mix(p_Func, p_Func2, p_Func_psi, p_Points, p_Treug,
p_Ind, p_PT, Numb, Ntr);
```

и поясним входящие в нее переменные. Итак, переменная `p_Func` является указателем на переменные типа `double`. Этот массив содержит значения нулевого приближения в методе градиентного спуска. Длина массива равна количеству точек триангуляции. Второй аргумент `p_Func2` — это указатель на массив, в который записываются результаты вычисления. Переменная `p_Func_psi` является указателем на массив данных, в которых хранятся значения функции ψ (краевое условие Неймана). Четвертая переменная `p_Points` является указателем на массив, состоящий из объектов класса `SQPoint_Mesh`. Этот класс является потомком класса `SQPoint`, в котором добавлено поле `bnd`, отвечающее за принадлежность точки к границе расчетной области и тип граничного условия. Следующая переменная `p_Treug` является указателем на массив объектов класса `SQTreug` треугольников используемой триангуляции. Шестой аргумент `p_Ind` — это указатель на объекты класса `SQIndexT`. Этот класс содержит поля `IndA`, `IndB`, `IndC`, которые являются номерами точек вершин треугольника. Индекс треугольника в массиве `p_Treug` совпадает с соответствующим индексом элемента массива `p_Ind`. Переменные `Numb`, `Ntr` содержат количество точек и треугольников триангуляции. Теперь поясним значение переменной `p_PT`. Для сравнения значений функционала или вычисления направления спуска мы используем для каждой вершины P_i только те треугольники, которые содержат эту точку в качестве своей вершины. Поэтому в отдельный массив `p_PT` мы помещаем соответствующую информацию. Выглядит этот массив так:

Номер точки	Количество треугольников у данной вершины (**4 байта**)	Номера треугольников
0	n_0	$T_1^0, T_2^0, ..., T_{n_0}^0$
1	n_1	$T_1^1, T_2^1, ..., T_{n_1}^1$
2	n_2	$T_1^2, T_2^2, ..., T_{n_2}^2$
...	...	...

Дадим несколько простых примеров решения различных краевых задач, используя описанные выше процедуры.

Пример 7. Предположим, нам нужно решить следующую задачу

$$\iint\limits_{\Omega} (u_x^2 + u_y^2)\, dxdy \to \min,$$

где $\Omega = [0,2] \times [0,2]$ и функции $u = u(x,y)$ удовлетворяют краевому условию

$$u(x,0) = x^2,\ u(x,2) = x^2 - 4,\ u(0,y) = -y^2,\ u(2,y) = 4 - y^2.$$

Понятно, что решением этой задачи является гармоническая функция $u = x^2 - y^2$. Для приближенного решения мы используем следующую триангуляцию нашей области. Разбиваем по каждой переменной 30-ю прямолинейными отрезками квадрат $[0,2] \times$ $\times [0,2]$ на 900 квадратов. Затем каждый из них делим диагональю пополам и получаем разбиение исходного квадрата на 1 800 треугольников. Результаты численного решения

поставленной задачи можно увидеть из следующей таблицы. В ней значения искомого и приближенного решений взяты для фиксированного значения x.

Номер точки	Приближенное значение	Точное значение	Погрешность, %
0	0,266674	0,266667	0,00272361
1	0,413341	0,413333	0,00176132
2	0,568896	0,568889	0,00126476
3	0,73334	0,733333	0,000964057
4	0,906674	0,906667	0,000761486
5	1,0889	1,08889	0,000600262
6	1,28001	1,28	0,000478745
7	1,48001	1,48	0,00037956
8	1,68889	1,68889	0,000303671
9	1,90667	1,90667	0,000243241
10	2,13334	2,13333	0,000187804
11	2,36889	2,36889	0,000133937
12	2,61334	2,61333	8,79675e-005
13	2,86667	2,86667	4,26486e-005
14	3,12889	3,12889	0
15	-1	-1	0
16	-0,995555	-0,995556	9,77761e-005
17	-0,98222	-0,982222	0,000197784
18	-0,959997	-0,96	0,000282251
19	-0,928886	-0,928889	0,000363266
20	-0,888885	-0,888889	0,000438596
21	-0,839996	-0,84	0,000521741
22	-0,782217	-0,782222	0,000629807
23	-0,71555	-0,715556	0,000758859
24	-0,639994	-0,64	0,000911162
25	-0,555549	-0,555556	0,00111466
26	-0,462216	-0,462222	0,00140913
27	-0,359993	-0,36	0,00188718
28	-0,248882	-0,248889	0,00282618
29	-0,128882	-0,128889	0,00556455

Пример 8. Рассмотрим теперь другой пример, в котором имеется краевое условие на нормальную производную. Пусть нужно найти решение уравнения

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0,\ 0 < x < 2,\ 0 < y < 2,$$

удовлетворяющее условиям

$$u(x,0) = x^2,\ u(x,2) = x^2 - 4,\ u(0,y) = -y^2,\ u_x(2,y) = 4.$$

Ясно, что решением этой краевой задачи является гармоническая функция $u = x^2 - y^2$,

на которой достигает своего минимума функционал

$$\frac{1}{2}\iint\limits_{\Omega}(u_x^2+u_y^2)\,dxdy-4\int\limits_0^2 u(2,y)dy.$$

Для поиска приближенного решения мы используем рассмотренную выше триангуляцию квадрата $[0,2]\times[0,2]$. Ниже приводится таблица полученных результатов с использованием метода «золотого сечения».

Номер точки	Приближенное значение	Точное значение	Погрешность, %
0	-1	-1	0
1	-0,995556	-0,995556	3,69844e-006
2	-0,982222	-0,982222	7,51804e-006
3	-0,96	-0,96	1,13207e-005
4	-0,928889	-0,928889	1,52227e-005
5	-0,888889	-0,888889	1,92366e-005
6	-0,84	-0,84	2,37347e-005
7	-0,782222	-0,782222	2,90668e-005
8	-0,715555	-0,715556	3,55389e-005
9	-0,64	-0,64	4,34191e-005
10	-0,555555	-0,555556	5,45779e-005
11	-0,462222	-0,462222	7,05253e-005
12	-0,36	-0,36	9,64248e-005
13	-0,248889	-0,248889	0,000147995
14	-0,128889	-0,128889	0,000300313
15	4,08992e-007	0	–
16	0,137778	0,137778	0,000312917
17	0,284445	0,284444	0,000158239
18	0,44	0,44	0,000106217
19	0,604445	0,604444	8,07618e-005
20	0,777778	0,777778	6,50983e-005
21	0,960001	0,96	5,45993e-005
22	1,15111	1,15111	4,72088e-005
23	1,35111	1,35111	4,14874e-005
24	1,56	1,56	3,70554e-005
25	1,77778	1,77778	3,33873e-005
26	2,00445	2,00444	3,05363e-005
27	2,24	2,24	2,80369e-005
28	2,48445	2,48444	2,60704e-005
29	2,73778	2,73778	2,44785e-005

Пример 9. Точным решением задачи

$$\frac{\partial}{\partial x}\left(\frac{u_x}{\sqrt{1+u_x^2+u_y^2}}\right)+\frac{\partial}{\partial y}\left(\frac{u_y}{\sqrt{1+u_x^2+u_y^2}}\right)=0,\ 0<x<2,\ 1<y<10,$$

$$u(x,1) = \ln(\sqrt{x^2+1}+x),\ u(x,10) = \ln(\sqrt{x^2+100}+\sqrt{x^2+99}),\ 0 \le x \le 2,$$
$$u(0,y) = \ln(y+\sqrt{y^2-1}),\ u(2,y) = \ln(\sqrt{4+y^2}+\sqrt{3+y^2}),\ 1 \le y \le 10,$$

является функция $u(x,y) = \ln(\sqrt{x^2+y^2}+\sqrt{x^2+y^2-1})$, описывающая минимальную поверхность — катеноид. Приводим соответствующие результаты вычислений. Нужно отметить, что в этом примере мы разделили оба отрезка $[0,2]$ и $[1,10]$ на 20 равных частей. В таблице даны значения для фиксированного $y = 5,5$.

Номер точки	Приближенное значение	Точное значение	Погрешность, %
0	2,40606	2,40606	0
1	2,42379	2,42367	0,00478255
2	2,44704	2,4468	0,00996285
3	2,47504	2,47467	0,0146479
4	2,50695	2,50649	0,018244
5	2,54198	2,54146	0,0205082
6	2,57937	2,57882	0,0214784
7	2,61847	2,61791	0,0213603
8	2,65871	2,65816	0,0204247
9	2,6996	2,69909	0,0189375
10	2,74078	2,74031	0,0171238
11	2,78193	2,78151	0,0151552
12	2,82281	2,82243	0,013152
13	2,86323	2,86291	0,0111911
14	2,90307	2,9028	0,00931607
15	2,94223	2,94201	0,00754603
16	2,98062	2,98045	0,00588293
17	3,01821	3,01808	0,00431703
18	3,05497	3,05489	0,00283069
19	3,09089	3,09084	0,00140078

Обоснование используемого вариационного метода для уравнений минимальной поверхности и равновесной капиллярной поверхности дано в работах [5] и [6]. Результаты расчетов и их визуализация для этих уравнений приведены в работах [3] и [4].

ПРИМЕЧАНИЕ

[1] Работа выполнена при финансовой поддержке РФФИ и Администрации Волгоградской области (проект № 15-41-02517 р_поволжье_а).

СПИСОК ЛИТЕРАТУРЫ

1. Григорьева, Е. Г. Алгоритмы идентификации граничных точек полигональных 3D моделей / Е. Г. Григорьева, В. А. Клячин // Геометрический анализ и его приложения : материалы III Междунар. науч. шк.-конф. — Волгоград : Изд-во ВолГУ, 2016. — С. 114–116.

2. Григорьева, Е. Г. Численное исследование устойчивости равновесных поверхностей с использованием пакета NumPy / Е. Г. Григорьева, В. А. Клячин // Вестник Волгоградского

государственного университета. Серия 1, Математика. Физика. — 2015. — № 2 (27). — С. 17–30.

3. Клячин, А. А. Визуализация расчета формы поверхностей минимальной площади / А. А. Клячин, В. А. Клячин, Е. Г. Григорьева // Научная визуализация. Электронный журнал. — 2014. — Т. 6, № 2. — С. 34–42.

4. Клячин, А. А. Визуализация устойчивости и расчета формы равновесной капиллярной поверхности / А. А. Клячин, В. А. Клячин, Е. Г. Григорьева // Научная визуализация. Электронный журнал. — 2016. — Т. 8, № 2. — С. 37–52.

5. Клячин, А. А. О равномерной сходимости кусочно-линейных решений уравнения минимальной поверхности / А. А. Клячин, М. А. Гацунаев // Уфимский математический журнал. — 2014. — № 6 (3). — С. 3–16.

6. Клячин, А. А. О равномерной сходимости кусочно-линейных решений уравнения равновесной капиллярной поверхности / А. А. Клячин // Сибирский журнал индустриальной математики. — 2015. — Т. 18, № 2 (62). — С. 52–62.

7. Михайленко, В. Е. Конструирование форм современных архитектурных сооружений / В. Е. Михайленко, С. Н. Ковалев. — Киев : Будівельник, 1978. — 138 с.

8. Олифант, Т. Многомерные итераторы NumPy / Т. Олифант // Идеальный код. — СПб. : Питер, 2011. — С. 341–358.

9. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Д. Влиссидес, Г. Буч. — СПб. : Питер, 2007. — 366 с.

10. Саранин, В. А. Равновесие жидкостей и его устойчивость / В. А. Саранин. — М. : Институт компьютерных исследований, 2002. — 144 с.

11. Тепляков, С. Паттерны проектирования на платформе .NET / С. Тепляков. — СПб. : Питер, 2016. — 320 с.

12. Финн, Р. Равновесные капиллярные поверхности. Математическая теория / Р. Финн. — М. : Наука, 1989. — 312 с.

REFERENCES

1. Grigoryeva E.G., Klyachin V.A. Algoritmy identifikatsii granichnykh tochek poligonalnykh 3D modeley [Identifying Algorithm for Boundary Points of 3D Models]. *Geometricheskiy analiz i ego prilozheniya: materialy III Mezhdunar. nauch. shk.-konf.* Volgograd, Izd-vo VolGU Publ., 2016, pp. 114-116.

2. Grigoryeva E.G., Klyachin V.A. Chislennoe issledovanie ustoychivocti ravnovesnykh poverkhnostey s ispolzovaniem paketa NumPy [Numerical Study of the Stability of Equilibrium Surfaces Using NumPy Package]. *Vestnik Volgogradskogo gosudarstvennogo universiteta. Seriya 1, Matematika. Fizika* [Science Journal of Volgograd State University. Mathematics. Physics], 2015, no. 2 (27), pp. 17-30.

3. Klyachin A.A., Klyachin V.A., Grigoryeva E.G. Vizualizatsiya rascheta formy poverkhnostey minimalnoy ploshchadi [Visualization of Calculation of Minimal Area Surfaces]. *Nauchnaya vizualizatsiya. Elektronnyy zhurnal*, 2014, vol. 6, no. 2, pp. 34-42.

4. Klyachin A.A., Klyachin V.A., Grigoryeva E.G. Vizualizatsiya ustoychivosti i rascheta formy ravnovesnoy kapillyarnoy poverkhnosti [Visualization of Stability and Calculation of the Shape of the Capillary Equilibrium Surface]. *Nauchnaya vizualizatsiya. Elektronnyy zhurnal*, 2016, vol. 8, no. 2, pp. 37-52.

5. Klyachin A.A., Gatsunaev M.A. O ravnomernoy skhodimosti kusochno-lineynykh resheniy uravneniya minimalnoy poverkhnosti [On Uniform Convergence of Piecewise Linear Solutions of the Minimal Surface Equation]. *Ufimskiy matematicheskiy zhurnal* [Ufa Mathematical Journal], 2014, no. 6 (3), pp. 3-16.

6. Klyachin A.A. O ravnomernoy skhodimosti kusochno-lineynykh resheniy uravneniya ravnovesnoy kapillyarnoy poverkhnosti [On the Uniform Convergence of Piecewise Linear Solutions of an Equilibrium Capillary Surface Equation]. *Sibirskiy zhurnal industrialnoy matematiki* [Journal of Applied and Industrial Mathematics], 2015, vol. 18, no. 2 (62), pp. 52-62.

7. Mikhaylenko V.E., Kovalev S.N. *Konstruirovanie form sovremennykh arkhitekturnykh sooruzheniy* [Designing Forms of Modern Architectural Structures]. Kiev, Budivelnik Publ., 1978. 138 p.

8. Oliphant T. Mnogomernye iteratory NumPy [Multidimensional Iterators in NumPy]. *Idealnyy kod* [Beautiful Code] Saint Petersburg, Piter Publ., 2011, pp. 341-358.

9. Gamma E., Helm R., Johnson R., Vlissides D., Booch G. *Priemy obyektno-orientirovannogo proektirovaniya. Patterny proektirovaniya* [Design Patterns: Elements of Reusable Object-Oriented Software]. Saint Petersburg, Piter Publ., 2007. 366 p.

10. Saranin V.A. *Ravnovesie zhidkostey i ego ustoychivost* [Equilibrium of Fluids and Its Stability]. Moscow, Institut kompyuternykh issledovaniy Publ., 2002. 144 p.

11. Teplyakov S. *Patterny proektirovaniya na platforme .NET* [Design Patterns for .NET Framework]. Saint Petersburg, Piter Publ., 2016. 320 p.

12. Finn R. *Ravnovesnye kapillyarnye poverkhnosti. Matematicheskaya teoriya* [Equilibrium Capillar Surfces. Mathematical Theory]. Moscow, Nauka Publ., 1989. 312 p.

UNIVERSAL SOFTWARE FOR SOLVING MULTIDIMENSIONAL VARIATIONAL PROBLEMS

Elena Gennadievna Grigoryeva

Candidate of Physical and Mathematical Sciences,
Associate Professor of Department of Computer Science and Experimental Mathematics,
Volgograd State University
e_grigoreva@mail.ru, e_grigoreva@volsu.ru, kiem@volsu.ru
Prosp. Universitetsky, 100, 400062 Volgograd, Russian Federation

Vladimir Aleksandrovich Klyachin

Doctor of Physical and Mathematical Sciences,
Head of Department of Computer Science and Experimental Mathematics,
Volgograd State University
klchnv@mail.ru, klyachin.va@volsu.ru, kiem@volsu.ru
Prosp. Universitetsky, 100, 400062 Volgograd, Russian Federation

Aleksey Aleksandrovich Klyachin

Doctor of Physical and Mathematical Sciences,
Head of Department of Mathematical Analysis and Function Theory,
Volgograd State University
klyachin-aa@yandex.ru, matf@volsu.ru
Prosp. Universitetsky, 100, 400062 Volgograd, Russian Federation

Abstract. The article considers the problem of numerical calculation of the piecewise linear surfaces M, wich is extremal for the functional type

$$F(M) = \int\limits_M (\Phi(\xi) + \alpha(x))\, dS.$$

To solve this problem, we obtain formulas for the calculation of the gradient of the functional in the space of polygonal surfaces M as a set P of its vertices

$$\left(\frac{\partial \tilde{F}}{\partial h}(P)\right)^i = \frac{1}{2}\langle \sum_{j=1}^{k} (\nabla\Phi(2\xi_j|T_j|)) \times l_j, h\rangle + \frac{1}{2}\langle \sum_{j=1}^{k} \alpha(p_j^*)\xi_j \times l_j, h\rangle +$$

$$+\frac{1}{3}\langle\sum_{j=1}^{k}\nabla\alpha(p_j^*)|T_j|,h\rangle.$$

For the organization of calculations on the basis of these formulas, we construct a system of classes in the Python programming language. The system is organized as a class package wich is integrated in the simulation environment 3D Blender. This solution allows the user for the program Blender in interactive mode to perform the desired surface topology modeling, to set the boundary conditions and to visualize the results of calculation.

In this paper, we present our implementation of a variational method based on approximation of piecewise-linear functions and surfaces. We proceed from the idea of creating a universal software system, most of which does not depend on the use of triangulation, integral functional, computational domain and boundary conditions. In this article we present the structure of input data, which is stored in the boundary conditions, and the vertices and the triangles of the triangulation and the description of the main program procedures. We are considering a number of illustrative examples of solving boundary value problems of mathematical physics.

Key words: extremal surface, piecewise linear approximation, triangulation, NumPy package, boundary problem.