УДК 519.688

ПРАКТИЧЕСКИЕ АСПЕКТЫ ОПРЕДЕЛЕНИЯ ЭКСТРЕМУМА ФУНКЦИИ НЕСКОЛЬКИХ ПЕРЕМЕННЫХ НА БАЗЕ МАТЕМАТИЧЕСКИХ МЕТОДОВ

Вотинов М.В.

Мурманский государственный технический университет

В процессе технического проектирования первостепенно ставится задача определения наилучших значений параметров объекта или его структуры. Такая задача называется оптимизационной.

В математике, задачей оптимизации называется задача определения экстремума вещественной функции в некоторой ее области.

В настоящее время существует большое разнообразие алгоритмов поиска экстремума функции. Задача выбора подходящего для решения конкретной задачи алгоритма весьма актуальна. При разработке программных алгоритмов определения экстремума функции выбор математического алгоритма представляет собой компромисс между точностью приближения к точке экстремума, временными затратами электронной вычислительной машины и простотой реализации.

На кафедре автоматики и вычислительной техники Мурманского государственного технического университета разработан автоматизированный программный модуль, позволяющий определять экстремум функции нескольких переменных, заданной пользователем.

Суть программной реализации модуля заключается в использовании метода градиентного спуска для определения экстремума функции. В техническом проектировании под экстремумом функции в большинстве случаев понимается ее минимум.

Основная идея метода градиентного спуска состоит в том, чтобы двигаться к минимуму в направлении наиболее быстрого убывания функции F, которое определяется антиградиентом ∇F. Например, для функции F(x,y,z):

$$\begin{cases} x^{[i+1]} = x^{[i]} - H * \nabla F(x^{[i]}) \\ y^{[i+1]} = y^{[i]} - H * \nabla F(y^{[i]}) \\ z^{[i+1]} = z^{[i]} - H * \nabla F(z^{[i]}) \end{cases} \tag{1}$$

В формулах H - шаг, с которым осуществляется градиентный спуск, на практике шаг может выбираться:

- постоянным;
- дробным;
- переменным.

Выбор постоянного и дробного шагов с точки зрения скорости и точности сходимости метода градиентного спуска не оптимален. Если постоянный шаг выбирается малым, то метод сходится медленно. Увеличение постоянного шага приводит к расходимости метода. Использование дробного шага, как и постоянного, приводит к увеличению итераций поиска экстремума функции.

Метод градиентного спуска с использование переменного шага позволяет избавиться от недостатков постоянного и дробного шагов, сократив количество итераций поиска и увеличив вероятность сходимости метода. Такой метод требует определения шага H на каждой итерации при помощи одномерной оптимизации.

Как показала практика работы с автоматизированным программным модулем, использование одного и того же значения шага H для каждой переменной функции – малоэффективно. Было принято решение о добавлении в программный алгоритм модуля расчета дифференцируемого по каждой переменной шага градиентного спуска, таким образом, формулы, определения экстремума функции преобразились:

$$\begin{cases} x^{[i+1]} = x^{[i]} - H[x] * \nabla F(x^{[i]}) \\ y^{[i+1]} = y^{[i]} - H[y] * \nabla F(y^{[i]}) \\ z^{[i+1]} = z^{[i]} - H[z] * \nabla F(z^{[i]}) \end{cases} \qquad (2)$$

Шаг по каждой переменной определяется исходя из условий:

$$\begin{cases} F(x^i - H[x]^i * \nabla F\left(x^{[i]}\right) \to min \\ F(y^i - H[y]^i * \nabla F\left(y^{[i]}\right) \to min \\ F(z^i - H[z]^i * \nabla F\left(z^{[i]}\right) \to min \end{cases} \qquad (3)$$

Определение дифференцируемого по каждой переменной шага градиентного спуска решается в автоматизированном программном модуле при помощи алгоритмов одномерной оптимизации, а именно метода золотого сечения.

Как известно, основным недостатком метода золотого сечения является то, что если в некоторой области функция, помимо ярко выраженного глобального экстремума, имеет несколько локальных минимумов (максимумов), то процесс, вполне вероятно, может сойтись к одному из них.

Для того чтобы устранить влияние локальных минимумов (максимумов) в автоматизированном программном модуле введено понятие «количества локальных областей поиска» - N. Величина N задается пользователем.

Область поиска переменной, соответствующей экстремуму функции, разбивается на N частей. На каждой части, из N-возможных, методом золотого сечения определяется свой локальный экстремум. Глобальный экстремум определяется на основании анализа локальных экстремумов на каждой из N частей. Соответственно, увеличивая значение N, увеличивается и точность работы программного алгоритма.

Алгоритм поиска экстремума функции производит останов, при соблюдении условия:

$$\sqrt{(\nabla F(x)^2 + \nabla F(y)^2 + \nabla F(z)^2)} \leq \mathrm{E}/2, \qquad (4)$$

где E - абсолютная ошибка, вводимая пользователем.

В теории оптимизации существуют тестовые задачи, позволяющие сравнивать эффективности различных алгоритмов поиска экстремума функции. Автоматизированный программный модуль был проверен на одной из них, а именно на функции Розенброка. Функция Розенброка, имеющая высокую степень нелинейности и сходящаяся чрезвычайно медленно при использовании излишне грубых методов оптимизации, определяется как:

F(x,y)=100*(y-x^2)^2+(1-x)^2 (5)

При вводе данной функции в автоматизированный программный модуль, экстремум, с точностью предложенного метода 0.001, определился за 12-15 шагов в зависимости от начальных условий.

Безусловно, дифференциация шага по всем переменным функции приводит к увеличению программного алгоритма, увеличению временных затрат на расчет, однако, даже при вводе сложных функций, позволяет максимально приблизиться к точке экстремума функции.

Практика показывает, что данный метод часто требует меньшего числа операций, чем метод градиентного спуска с постоянным шагом или дробным шагом.

Таким образом, автоматизированный программный модуль определения экстремума функции нескольких переменных на основе метода градиентного спуска с использованием диверсифицированного по каждой переменной шаге поиска, обеспечивает компромисс между точностью приближения к точке экстремума и временными затратами электронной вычислительной машины при существенной простоте реализации по сравнению с множеством аналогичных методов. Данный автоматизированный программный модуль также может применяться в качестве базисного алгоритмического элемента для построения глобальных систем оптимизации