Математические структуры и моделирование
2002, вып. 9, с. 1–10

УДК 681.3

О ПАКЕТЕ ПРОГРАММ НЕДИФФЕРЕНЦИРУЕМОЙ ОПТИМИЗАЦИИ

Т.Б. Бигильдеева, В.В. Таркаев

In the article the complex of the nondifferential optimization programs including special means of maintenance of input-output of the data and visualization of optimization process is considered. These features of the software package make its convenient as by development both research of new algorithms and methods of optimization, and with use it in educational process.

Введение

В Челябинском государственном университете с начала девяностых годов под руководством профессора В.Д.Батухтина разрабатываются методы решения негладких и разрывных задач оптимизации, базирующиеся на понятии аппроксимационного градиента [1]. В этой связи возникла потребность в программировании разрабатываемых алгоритмов и проведении численных экспериментов. Однако почти сразу программистская задача была поставлена существенно шире - наряду с программированием численных методов была начата разработка набора средств, позволяющих использовать имеющиеся наработки сразу в нескольких тесно связанных областях научной и преподавательской деятельности.

А именно:

1) использование при разработке и тестировании новых алгоритмов;

2) использование при решении конкретных прикладных задач;

3) иллюстрирование выступлений, посвященных изложению полученных результатов;

4) разработка иллюстративно-обучающих программ по курсу «методы оптимизации»;

5) использование студентами в ходе подготовки курсовых и дипломных работ по родственной проблематике.

В результате был разработан описываемый программный комплекс : «Пакет программ недифференцируемой оптимизации». Неоднократно он был использован для иллюстрирования выступлений на международных семинарах по недифференцируемой оптимизации. В течение нескольких лет он установлен в локальной сети математического факультета ЧелГУ и был использован уже при подготовке более чем двадцати курсовых и дипломных работ. Но, разумеется, наиболее важное его использование - это разработка и тестирование новых вычислительных методов.

E-mail: tbig@cgu.chel.su

Челябинский государственный университет

1. Цели разработки и общая характеристика пакета

При разработке описываемого программного комплекса преследовались следующие цели:

1. Максимальное упрощение для исследователя самого процесса программирования алгоритмов и тестовых задач благодаря использованию специализированных библиотек.

2. Наличие удобных средств ввода/вывода данных.

3. Наличие средств визуализации хода вычислительного процесса.

4. Возможность сохранения информации о ходе и результатах вычислительного процесса.

5. Наличие библиотек задач оптимизации и методов оптимизации с максимально упрощенным доступом к ним.

6. Поддержка коллективной работы.

7. Наличие средств, облегчающих реализацию иллюстративных программ, представляющих собой заранее подготовленный набор вычислительных экспериментов, с возможностью интерактивного изменения ряда параметров и отображением поясняющих текстов.

8. Наличие нетрудоемкой технологии экспорта/импорта алгоритмов и задач оптимизации на уровне исходного текста.

Разработчики описываемого программного комплекса прежде всего ориентировались на потребности вынужденного программировать математика-исследователя и на родственные им потребности студента, программирующего в ходе выполнения курсовых и дипломных работ. В соответствии с этим при разработке структуры пакета и технологии его использования определяющим было стремление, во-первых, максимально упростить стадию программирования алгоритма, обеспечив, однако, во-вторых, максимум удобств на стадии тестирования программы и апробации алгоритма. Ясно, что одновременное достижение двух этих целей возможно лишь при условии достаточно жесткой стандартизации поддерживаемых типов задач. При этом обязательно должна быть оставлена возможность выхода за рамки этого стандарта, платой за это является повышение сложности программирования.

В случае пакета программ недифференцируемой оптимизации таким стандартом является вычислительный процесс, реализующий решение задачи безусловной оптимизации.

Причем он должен, во-первых, быть итерационным и, во-вторых, допускать деление на три блока: оптимизируемая функция, главный метод и вспомогательный метод (может отсутствовать). Как правило, вспомогательный метод - это одномерный метод оптимизации, осуществляющий выбор величины шага по определенному главным методом направлению.

Деление на логически независимые стандартные блоки обеспечивает возможность замены каждого из них на любой другой блок, относящийся к тому же типу, и, кроме того, упрощает их независимую разработку. Тем более, что каждый из таких блоков реализуется как отдельный exe-файл.

Нематематическая часть программы, реализующей каждый из блоков алго-

ритма, удовлетворяющего двум выше указанным требованиям, в пакете недифференцируемой оптимизации сведена, по сути дела, к нетрудоемкому заполнению своеобразной «анкеты» метода или задачи. Требования же, предъявляемые к реализации математической части, сродни стилистическим. Состоят они в необходимости написания определенных функций с определенными типами и интерпретацией параметров и возвращаемых величин. Например, для задачи оптимизации это раздельная реализация функций, вычисляющих сам функционал и его градиент (если таковой определен) с предопределенными типами возвращаемых величин и параметров (см. пример в пункте 2). В реализации метода обязательно должна быть функция, выполняющая очередную итерацию. Кроме того, в каждом модуле должна быть реализована функция его инициализации.

Реализация методов, предназначенных для запуска в качестве главных и в качестве вспомогательных, почти ничем не различается, Во всяком случае, методы, написанные как вспомогательные, могут запускаться и как главные, что, в частности, упрощает их тестирование.

Простота интеграции в пакет блоков, стандартных с точки зрения описанной схемы, достигается, естественно, за счет стандартизации их интерфейсной части, то есть набора функций, реализованных в некотором блоке и предназначенных для использования другими блоками. Для задач безусловной оптимизации такая стандартизация является достаточно естественной и необременительной для программирующего математика. Но разработка чего-то подобного для более широких классов вычислительных процессов, включающих, например, решение задач условной оптимизации, оказалась практически невозможной, так как пришлось бы учесть слишком много вариантов, что свело бы удобства к минимуму. Поэтому было принято решение пойти по пути реализации простых в использовании средств конструирования интерфейсов самим пользователем пакета. Практически любая функция из одного блока может быть «объявлена» глобальной, что обеспечивает возможность ее вызова из другого блока. В этом случае возможность взаимной замены блоков обеспечивается уже не средствами пакета, а на уровне дисциплины программирования. Подчеркнем, что использование этой технологии не усложняет доступ к сервисным возможностям пакета, а лишь снимает ограничения на структуру реализуемого процесса.

2. Примеры программирования

Приведенные ниже примеры содержат полные тексты программ, реализующих простейшие метод и задачу оптимизации. Подчеркнем, что тексты эти являются именно полными, то есть exe-файлы, полученные в результате их компиляции и линковки, вполне могут быть запущены в составе пакета для проведения экспериментов.

2.1. Простейшая задача оптимизации (минимизация квадратичной функции)

Математическая часть программы включает функции, вычисляющие значение и градиент оптимизируемого функционала (запись x*x означает скалярное произведение векторов, а запись 2*x означает умножение вектора x на 2).

Нематематическая часть состоит из обязательной функции инициализации. В этой функции размерность задачи и координаты начальной точки считываются из файла данных. После этого заполняется «анкета» задачи. В данном примере в модуле содержится только одна задача, и поэтому значение параметра nproblem игнорируется.

```
/*Математическая часть */
double func ( vector & x ) { return ( x * x ); }
vector grad ( vector & x ) { return ( 2 * x ); }
/* Нематематическая часть */
int problem_init ( int nproblem )
{
int n;
n=input («размерность задачи»);
vector start_point(n);
start_point=input («начальная точка»);
set_problem_name («КВАДРАТИЧНАЯ ФУНКЦИЯ»);
set_problem_start_point (start_point);
set_function (func);
set_gradient (grad);
return (1);
}
```

2.2. Простейший градиентный метод оптимизации

Метод состоит в запуске одномерного подметода выбора шага по направлению антиградиента. Функция, вычисляющая градиент (grad), и сам подметод выбираются на этапе запуска метода. Процесс прекращается, если значение нормы градиента меньше заданного минимального значения градиента (min_grad).

Нематематическая часть состоит из функции инициализации и функции, осуществляющей графический вывод данных на каждой итерации. В данном примере выводится только текущая точка (gr_current_point) и норма градиента (gr_norma_grad).

В функции инициализации заполняется «анкета» метода (имя метода, имя функции, осуществляющей очередную итерацию, и функции, осуществляющей вывод). Инициализируются переменные метода, запрашиваются задача и вспомогательный метод, инициализируются переменные, обеспечивающие графический вывод.

```
/* Переменные метода */
method_type submethod; double min_grad; vector x;
array gr_norma_grad, gr_current_point;
```

```
/* Математическая часть */
int next_iteration (void)
{
submethod.run(x,-grad(x)); x=submethod.current_point();
if(norma(grad(x)) < min_grad) return(IR_STOP);
else return(IR_CONTINUE);
}
/* Нематематическая часть */
void func_output(void)
{
gr_norma_grad < < norma(grad(x));
gr_current_point < < x;
}
int method_init (int nmethod)
{
set_method_name(«Простейший градиентный метод»);
set_method_next_iteration(next_iteration);
set_method_output(func_output);
min_grad=input(«Минимальное значение градиента»);
submethod=load_submethod();
load_problem();
x=problem_start_point();
gr_norma_grad = output_array ( «grad» );
int dim = problem_dimension ();
gr_current_point = output_array («Траектория», dim);
return(1);
}
```

3. Проведение вычислительного эксперимента

Что же дает включение некоторого модуля в пакет? Или, иначе говоря, какие возможности предоставляются математику на этапе проведения вычислительного эксперимента при условии, что составляющие вычислительный процесс блоки реализованы в соответствии с предъявляемыми пакетом требованиями?

1. Исходные данные вычислительного процесса считываются из текстового файла. В обязательной его части содержится указание на задачу, запускаемый метод и подметод. Подметод может отсутствовать. В остальном набор исходных данных, их имена и типы определяются при реализации конкретных модулей. Например, если где-то в тексте программы содержалась строка

start_point = input («начальная точка»);

где переменная start_point имеет тип vector, то в файле будет осуществляться поиск строки вида

начальная точка = {1.0, 2.0, 3.0 },

а если таковой найдено не будет, то на экране появится соответствующий запрос.

Такой способ задания исходной информации ориентирован на этап численного экспериментирования с разрабатываемым методом, когда нередко возникает потребность в смене состава параметров. Для «устоявшихся» модулей имеется другой способ. Он предполагает перечисление в специальном формате имен входных параметров, их рекомендуемых значений и кратких пояснений к ним. После чего все необходимое будет переноситься в файл данных средствами пакета.

2. Каждый из трех блоков – оптимизируемая функция, главный метод и вспомогательный метод – может быть заменен на любой другой ему подобный. Достигается это изменением соответствующего параметра в файле, содержащем исходные данные.

3. Выбор и/или изменение файла данных может осуществляться непосредственно перед запуском вычислительного процесса. Все это сводит к минимуму трудоемкость проведения серий вычислительных экспериментов вроде испытания одного метода на различных задачах. Такие серии могут готовиться и заранее для исполнения без непосредственного участия экспериментатора.

4. По ходу вычислительного процесса и по его окончании на экране могут отображаться последовательности, элементами которых являются либо вещественные числа, либо двумерные векторы, либо векторы произвольной размерности. Номер элемента последовательности есть номер итерации главного метода. Ломаные на плоскости, как правило, являющиеся траекториями метода, могут изображаться на линиях уровня оптимизируемого функционала. Реализованы и стандартные манипуляции с графиками, такие как сдвиги и изменение масштаба. Графические возможности пакета этим не исчерпываются, но для обеспечения выше описанного минимума в тексте программ достаточно иметь по две строки на каждый график: в первой он должен быть определен с заданием отображаемого на экране имени, например

array current_point=output_array(«текущая точка»,2);

(второй параметр - размерность элемента последовательности). Во второй строке осуществляется сам вывод очередного элемента:

current_point << x;

Об изображении линий уровня при программировании задачи вообще можно не заботиться. Если она была реализована в соответствии с описанной технологией, то все будет сделано средствами пакета.

5. По окончании расчета или после его прерывания данные (если они выводились описанным выше способом) могут быть сохранены в текстовом файле. Имеется также и возможность бинарного сохранения данных, что делает возможным последующий просмотр их с использованием выше перечисленных графических возможностей.

6. При необходимости могут быть включены счетчики вычислений оптимизируемого функционала и его градиента, причем возможен отдельный подсчет вычислений, осуществляемых из главного и вспомогательного методов. Если задача оптимизации была реализована в соответствии с выше описанными требованиями, то использование счетчиков не требует ничего от программы.

4. Структура и составные части пакета

4.1. Структура пакета

Структура пакета продиктована стремлением одновременно решить две сформулированные выше задачи: во-первых, максимально упростить программирование определенных классов оптимизационных задач, обеспечив при этом, во-вторых, максимум сервиса, специфичного для этого класса, на этапе проведения численных экспериментов.

И если для решения первой задачи достаточно разработать специализированные библиотеки, то более качественное решение второй может быть обеспечено благодаря использованию специализированных программ.

Именно этими соображениями и было продиктовано решение реализовать описываемый программный комплекс не как одну многофункциональную программу, а как набор специализированных программ. Решение это было в значительной степени предопределено выбором WINDOWS в качестве основной операционной системы.

Итак, пакет можно разделить на три части.

1. Служебная часть пакета, состоящая из нескольких .dll и .exe файлов. Они обеспечивают подготовку и запуск вычислительных процессов, управление ими, хранение, просмотр и сохранение информации, текстовый ввод/вывод данных.

2. Библиотеки функций и классов в смысле языка C++, используемые при разработке программ, предназначенных для работы в составе пакета.

3. Реализованные на базе пакета модули, относящиеся к одному из четырех следующих типов :

А) задачи оптимизации,

Б) методы оптимизации,

В) иллюстративно-обучающие программы,

Г) модули без предопределенного набора интерфейсных функций.

Пользователи пакета занимаются разработкой приложений, относящихся к третьей категории.

Реализация пользовательских модулей как самостоятельных WINDOWS-приложений позволяет:

во-первых, избежать существенной части проблем, которые возникли бы при работе нескольких человек над одной программой;

во-вторых, пользуясь многозадачностью WINDOWS, формировать вычислительный процесс из нескольких независимо разработанных модулей (часть из которых может быть объединена в библиотеки);

в-третьих, обеспечивает определенную устойчивость пакета в целом к ошибкам, случившимся в одном из модулей.

4.2. Вспомогательные библиотеки

В состав пакета включены несколько файлов .lib. Объединенные в них функции и классы можно разделить на несколько групп :

1. Функции, позволяющие реализовать некоторый модуль как модуль определенного типа (метод, задача, иллюстративно-обучающая программа). Они, в

частности, позволяют начинающему пользователю разрабатывать WINDOWS-приложения, совершенно не зная особенностей программирования и тонкостей под WINDOWS.

2. Функции и классы, обеспечивающие стандартное взаимодействие стандартных блоков (методов, подметодов, задач), составляющих вычислительный процесс.

3. Функции и классы, обеспечивающие текстовый ввод/вывод данных различных типов.

4. Функции и классы, обеспечивающие доступ к графической подсистеме пакета.

5. Функции и классы, обеспечивающие программное управление подсистемой сохранения результатов вычислительного процесса.

6. Функции, используемые при разработке иллюстративно-обучающих программ, реализуемых на базе пакета.

7. Вспомогательные типы данных: вектор, матрица, динамический массив.

Возможности, предоставляемые пакетом при программировании, можно условно разделить на два уровня по сложности освоения. Возможности первого уровня ориентированы на пользователей, либо обладающих лишь начальными навыками программирования на C++ (например, студентов непрограммистских специализаций), либо на пользователей, не желающих тратить время и силы на освоение дополнительных возможностей. Таким пользователям предоставляется возможность реализовать задачи и методы оптимизации, почти не тратя сил на нематематическую часть программ. При этом практически все сервисные возможности пакета остаются доступными при проведении вычислительных экспериментов с таким образом написанными модулями.

С другой стороны, имеется возможность при разработке приложений, выходящих за пределы схемы «главный метод, вспомогательный метод, задача оптимизации», либо приложений, нуждающихся в пользовательском интерфейсе, учитывающем специфику конкретной задачи (прежде всего иллюстративно-обучающих программ), программировать на более низком уровне, что, разумеется, требует более высокой программистской квалификации и несколько больших затрат на освоение соответствующих средств.

Доступна также возможность программирования с использованием непосредственно программного интерфейса WINDOWS.

4.3. Библиотека задач оптимизации

Библиотека задач оптимизации в пакете недифференцируемой оптимизации представляет собой набор exe-файлов, реализующих задачи оптимизации и запрограммированных для работы в составе пакета.

Понятие задачи включает в себя две обязательные компоненты: оптимизируемый функционал и начальную точку процесса. Могут быть заданы еще и несколько необязательных: градиент оптимизируемого функционала, точка минимума, название задачи.

Библиотека задач оптимизации делится на три части:

1) задачи безусловной оптимизации дифференцируемых функций;

2) задачи безусловной оптимизации недифференцируемых, но непрерывных функций;

3) задачи безусловной оптимизации разрывных функций.

Библиотека задач оптимизации дифференцируемых функций включает в себя задачи оптимизации известных функций, таких, как расширенная функция Розенброка, функция Бокса, обобщенная функция Пауэлла, функция Вуда, функций Кларка и ряд других функций [3, 4, 6], каждая из этих задач обладает некоторой «изюминкой», создающей проблемы в процессе оптимизации, например плохой обусловленностью, вырожденностью точки минимума, многоэкстремальностью и так далее. Апробация численных методов и алгоритмов на этих задачах позволяет получить достаточно полную картину о возможностях методов, их достоинствах и недостатках.

Задачи оптимизации недифференцируемых функций состоят из задач выпуклого программирования (это задача аппроксимации полиномами в L1, минимизации негладкой функции Шора и другие) [4], а также из задач оптимизации невыпуклых функций.

Задачи оптимизации разрывных функций сконструированы из известных задач оптимизации гладких функций путем «введения» разрывов (к таким задачам, в частности, относится задача минимизации функции Розенброка, дополненной разрывом по гиперплоскости, проходящей через точку минимума вдоль оврага), а также путем сведения задач оптимизации с ограничениями к задачам безусловной минимизации разрывных функций (это возможно благодаря введению точных штрафов или специального продолжения минимизируемой функции вне области задания). При построении этих задач моделировались различные ситуации, усложняющие решение задачи, например прохождение разрыва через точку минимума, медленное изменение функции вдоль разрыва, что затрудняет движение к минимуму вдоль разрыва и так далее. В этом классе задач присутствуют задачи минимизации различных кусочно-линейных разрывных функций, а также задачи, возникающие на практике, например задача оптимизации распределения потока воды в ГРЭС [8], и другие.

Библиотека задач оптимизации постоянно пополняется, что расширяет возможость проведения численных экспериментов и позволяет глубже исследовать достоинства и недостатки создаваемых методов и алгоритмов.

4.4. Библиотека методов оптимизации

Библиотека методов оптимизации в пакете недифференцируемой оптимизации представляет собой набор exe-файлов, реализующих методы оптимизации и запрограммированных для работы в составе пакета. Этот набор, в частности, включает в себя методы одномерной оптимизации, предназначенные для использования в качестве вспомогательных для выбора величины шага по выбранному главным методом направлению.

Библиотека методов оптимизации делится условно на две части:

1) методы оптимизации дифференцируемых функций;

2) методы недифференцируемой оптимизации.

Методы оптимизации дифференцируемых функций включают в себя известные методы безусловной оптимизации: градиентные методы, квазиньютоновские методы, метод Ньютона и его модификации [4, 5, 7]. Эти методы могут работать с различными процедурами выбора шага по направлению (выбором шага путем одномерной минимизации методом золотого сечения, выбором шага по правилу Армихо, по правилу Голдстейна) [5, 7].

Методы оптимизации недифференцируемых функций включают в себя методы прямого поиска (метод покоординатного спуска, метод многогранника Нелдера-Мида), субградиентные методы (метод Шора, метод Поляка, субградиентный метод последовательной релаксации) [4, 5], а также метод Конна-Монго минимизации кусочно-линейных разрывных функций, основанный на декомпозиции функции [2].

Основное место в библиотеке методов занимают методы, созданные на базе понятия аппроксимационного градиента. К этим методам относятся метод аппроксимационного градиента и его модификации, аппроксимационный аналог метода Ньютона, адаптивные алгоритмы [1, 9]. Данные методы предназначены для решения широкого класса задач безусловной оптимизации разрывных функций. Эти методы постоянно совершенствуются, создаются их новые модификации.

В настоящее время пакет программ недифференцируемой оптимизации успешно используется как при выполнении научно-исследовательских работ, так и в учебном процессе. В частности, на базе пакета создано компьютерное пособие «Соревнование методов оптимизации» [10]. Лекции с использованием этого пособия вызывают большой интерес у студенческой аудитории.

Литература

1. Батухтин В.Д., Майборода Л.А. *Разрывные экстремальные задачи.* С.-П.: Гиппократ, 1995. С.358.
2. Conn A.R., Mongeau M. *Discontinuous piecewise linear optimization.* // Mathematical Programming. 1998. V.80. P.315–380.
3. Дэннис Дж., Шнабель Р. *Численные методы безусловной оптимизации и решения нелинейных уравнений.* М.: Мир, 1988.
4. Поляк Б.Т. *Введение в оптимизацию* М.: Наука, 1983. С.384.
5. Гилл Ф., Мюррей У., Райт М. *Практическая оптимизация.* М.: Мир, 1985. С.509.
6. Кларк Ф. *Оптимизация и негладкий анализ.* М.: Наука, 1988. С.279.
7. Бертсекас Д. *Условная оптимизация и методы множителей Лагранжа.* М.: Радио и связь, 1987. С.400.
8. Klein E.M. and Sim S.H. *Discharge allocation for hydro - electric generating stations* // European Journal of Operational Research. 1994. V.73. P.132–138.
9. Батухтин В.Д., Бигильдеев С.И., Бигильдеева Т.Б. *Численные методы решения разрывных экстремальных задач* // Изв.РАН. Теория и системы управления. 1997. N.3. С.113–120.
10. Бигильдеева Т.Б., Таркаева О.В. *Компьютерное пособие «Соревнование методов оптимизации»* // Тез.докл. ВВМШ «Понтрягинские чтения» 3-9 мая 2001. Воронеж, ВГУ. С.23.