Назад в будущее, вперед в прошлое

**Зачастую применению современных микроконтроллерных устройств препятствует ограниченность их ресурсов, в частности, оперативной памяти.
Интересно, что с аналогичными проблемами приходилось сталкиваться при решении многих задач в 60–70-х годах прошлого века программистам, работавшим на первых поколениях ЭВМ. Поэтому опыт и приемы, являющие собой пример нахождения наиболее оптимальных и эффективных алгоритмов решения задач, выработанные ими в то время, актуальны и сейчас, в частности, и для микроконтроллерных систем.
Итак, рассказывает член Ядерного общества России, один из создателей Комплекса графических программ на Фортране (ГРАФОР) С. А. Зверев. С результатами работы созданных им программ большинство из нас знакомы, ведь именно благодаря им мы могли видеть изотермы на карте Советского Союза в блоке погоды информационной программы «Время».**

Сергей Зверев

Возможности ЭВМ конца 70-х годов прошлого века уже позволяли решать достаточно большие и сложные задачи по расчету разного рода полей в пространстве или в исследуемом объекте. Например, полей значений температур, давления, скорости ветра приземных слоев воздуха (метеорология и навигация), плотности потока нейтронов и полей температур в выделенном сечении активной зоны ядерного реактора, магнитных полей и т. д.

Также к тому времени вырос объем соответствующих данных, получаемых в экспериментах и проведении практических измерений, например, значений высот поверхности земли (навигация, скрытые полеты в складках рельефа местности) и т. д.

Обработка результатов была чрезвычайно сложна из-за большого объема получаемых данных, возрастающего пропорционально квадрату частоты сетки, накладываемой на рассматриваемую область, в узлах которой задавались или вычислялись значения поля, а также самому размеру области. Так, например, при сетке 50×50 имеем 2500 узлов и, следовательно, значений интересующей нас величины, а при сетке 100×100 — уже 10000.

Для наглядного представления и обработки получаемых результатов наиболее приемлемыми являются так называемые карты изолиний, которые широко использовались еще в «домашинный» период для целей исследования функций двух переменных $Z=f(x, y)$.

В подавляющем большинстве случаев функция $f(x, y)$ задается не аналитически, а множеством значений в узлах, вообще говоря, неравномерной сетки в виде двумерного массива, либо достаточно просто может быть приведена к этому.

Найти решения уравнения $f(x, y) = C$ достаточно просто, например, организовав двойной цикл по i и j, с условиями:

$$f(x_i, y_j) \lesseqgtr C \lesseqgtr f(x_{i+1}, y_j),$$

либо

$$f(x_i, y_j) \lesseqgtr C \lesseqgtr f(x_i, y_j+1),$$

вычисляя координаты пересечения уровня C с поверхностью ¦(x, y), доопределяя ее на ребрах прямоугольных ячеек, соединяющих узлы сетки с помощью линейной, сплайн и прочей интерполяции.

Однако это не дает решения задачи, поскольку совершенно не понятно, каким образом (в какой последовательности) должны быть выстроены (соединены) полученные точки. Становится ясно, что после нахождения какой-либо точки изолиния в дальнейшем должна отслеживаться вплоть до ее замыкания на саму себя, то есть до выхода в исходную точку. (Следует иметь в виду, что изолинии могут быть как замкнутыми, лежащими целиком внутри области задания функции, так и не замкнутыми — начинающимися и заканчивающимися на границах области задания. Это легко преодолеваемое препятствие, на котором не стоит останавливаться. Так же, как и на том допущении, что на ребрах сетки между ее узлами функция полагается монотонной, то есть через ребро может проходить лишь одна изолиния данного уровня. Однако рассмотрение этих вопросов выходит за рамки статьи.)

Если каким-либо образом не регистрировать факт прохождения изолинии через ребра сетки, то программа, реализующая такой алгоритм, будет многократно находить и отслеживать уже найденные изолинии. Простейшим — в лоб — решением является организация служебного буферного массива, равного по объему массиву, содержащему данные об исследуемой функции (поверхности) $f(x, y)$. Однако применение таких программ (см., например, [1 с. 158]) ограничивала относительно малая память ЭВМ той эпохи. Выход из сложившейся ситуации пытались найти в написании на ассемблере специальных программ, организующих и обслуживающих буферный массив на байтовом и даже на битовом уровне! Что, вообще говоря, демонстрирует отчаянное положение и актуальность проблемы. Но применению таких программ препятствовала невозможность их переноса с одного типа ЭВМ на другие, не говоря уже о других их недостатках. Не секрет, что в ту пору более «закрытые» организации располагали CDC и «родными» IBM, БЭСМ-6 и специализированны-

ми ЭВМ, менее «закрытые» БЭСМ-6 и ЕС, совсем «открытые», как правило, первым рядом ЕС.

Поскольку ни одна из существующих в то время программ нахождения и построения линий уровня не удовлетворяла всем нашим потребностям, пришлось делать это самим. В процессе анализа задачи родилась идея регистрировать факт прохождения изолинии не в буферном массиве, а на самой поверхности, т. е. исключить даже намек на буферный массив. Действительно, если поверхность $f(x,y)$ всюду на области задания положительна (находится целиком в положительном полупространстве), то, присвоив знак «–» значениям функции в узлах сетки, соединяемых ребром, через которое прошла изолиния, мы тем самым регистрируем факт ее прохождения. Если поверхность $f(x, y)$ знакопеременна (лежит частью в положительном, частью в отрицательном полупространстве) или целиком в отрицательном полупространстве, то ничто не мешает нам «поднять» ее целиком в положительное полупространство, например, путем прибавления к каждому ее значению величины $min\{f(x, y)\} + 1$. После нахождения изолиний одного уровня и перехода к другому применить к ней процедуру $ABS\{f(x, y)\}$, вернув ее всю в положительное полупространство. А после нахождения изолиний всех искомых (заданных) уровней привести ее к исходному виду.

Внутренняя логика программы, по сравнению с аналогичной программой, использующей буферный массив [1, с. 158], надо полагать, несколько усложнилась, хотя с ее текстами ознакомиться не пришлось. Сравнение на тестах на БЭСМ-6, помнится, не показало никакого преимущества «буферного» варианта — ни по времени выполнения, ни по объему памяти, занимаемому самой программой. Экономия же памяти в два раза за счет отказа от буферного массива была налицо. Связано ли это с тщательностью написания, особенностями трансляторов с Фортрана, по-разному оптимизирующих программы, либо с тем, что логические операции более короткие и быстрые, даже по сравнению с операциями считывания из памяти и адресации, или к тому же с особенностями работы быстрых регистров БЭСМ-6, — судить не берусь. Интересно отметить, что объектный модуль ядра комплекса («безбуферного») IZOLIN [1, с. 174; 2], подпрограмма IZLIN, находящая изолинии подготовленной (поднятой в положительное полупространство) поверхности, занимала в памяти БЭСМ-6 всего 474 слова! Иначе — 2844 байта, так как слово БЭСМ-6 состояло из 6 байтов. Весь же комплекс, целиком загруженный в оперативную память, с программами подготовки и преобразования поверхности, сервисными программами преобразования координат, нахождения локальных экстремумов функции (поверхности), постановки берг-штрихов и т. д. занимал 1655 слов БЭСМ-6.

Берг-штрихи — это штрихи, наносимые на изолинию поверхности нормально к ней в направлении «стока воды» с этой поверхности. Юрий Матвеевич Баяковский, патриарх машинной графики в Советском Союзе, поставил мне условием включения программ в ГРАФОР, который тогда уже был стандартом — развивался, распространялся и поддерживался во всем социалистическом лагере, возможность постановки этих самых берг-штрихов. Постановка задачи была весьма интересной: «Поскольку моим друзьям географам и метеорологам хотелось бы видеть на изолиниях берг-штрихи, то даже такие хорошие программы, но без берг-штрихов ГРАФОРУ не нужны». Все мои апелляции: что в других программах, уже включенных в ГРАФОР, берг-штрихов нет, что инициатива включения моего комплекса в ГРАФОР исходила изначально от Института прикладной математики (ИПМ), в котором он работал, а также то, что мы с его сотрудниками, с его же благословения провели уже большую работу по включению — никакого видимого эффекта не имели. Позднее я понял, что это было не так, но другого способа давления на меня у него не было, поскольку мы работали в разных «конторах». Скорее из сожаления за напрасно потраченные усилия и время на работу по включению программ в ГРАФОР, я готов был написать «монстра» по сравнению с тем, что уже было сделано. Однако попытки, если так можно выразиться, найти красивое, элегантное решение, к счастью, увенчались большим успехом.

Оказалось, что изолинии «выпуклостей» (холмы) поверхности программа отслеживает против часовой стрелки, а вогнутости (ямы, впадины) — по часовой стрелке. До определенного момента это было даже мне не интересно! Решение заключалось в том, что векторное произведение единичного орта k (ось z) на вектор (направленный отрезок), соединяющий две соседние точки найденной изолинии, и по определению векторного произведения нормальный к ней, и является искомым берг-штрихом! Параллельно с этим выяснилось, что можно решать не только задачи нахождения и построения изолиний, но и задачи нахождения и построения пространственных кривых, образованных пересечением поверхностей, если предварительно решить задачу нахождения и построения проекций линий пересечения поверхностей на область их задания. Все это также было осуществлено.

Действительно, если имеются две поверхности (функции) $A(x, y)$ и $B(x, y)$ на области задания P, то проекции линий пересечения этих поверхностей на P являются решением уравнения $A(x, y) = B(x, y)$. Если $\Phi(x, y) = A(x, y) - B(x, y)$, то задача их нахождения сводится к поиску изолиний $\Phi(x, y) = 0$.

И далее, найдя множество последовательных значений x_{iz}, y_{jz} удовлетворяющих $\Phi(x, y) = 0$ (то есть координаты изолинии x_{iz}, y_{jz} нулевого уровня $\Phi(x, y)$), пространственные линии пересечения поверхностей L находятся через $A(x, y)$ или $B(x, y)$ следующим образом:

$$L(x_{iz}, y_{jz}) = A(x_{iz}, y_{jz}) = B(x_{iz}, y_{jz})$$

Пространственные линии пересечения поверхностей — это уже задача систем автоматизированного проектирования (САПР). Например, сопряжения фюзеляжа самолета с крылом или обтекателем радара, корпуса реактора с трубопроводом, трубы с бачком (в унитазе) и т. д. Кто бы мог подумать, что все эти задачи можно решить с помощью программы объемом менее 3 кбайт, без привлечения буферных массивов, да еще написанной на языке высокого уровня Фортране? А уж про аппаратную реализацию и говорить не приходится.

К сожалению, в настоящее время, как мне кажется, отработке алгоритмов (уму) и оптимизации (жир) программ уделяется все меньше внимания. Программы от этого рождаются «тупыми» и страдающими «ожирением». Ознакомившись по случаю с характеристиками современных микроконтроллеров, я пришел к выводу, что при правильном, тщательном и грамотном подходе они способны на решение существенно большего круга задач, чем те, которые с их помощью решаются в настоящее время.

Литература

1. Баяковский Ю. М., Галактионов В. А., Михайлова Т. Н. Графор. Графическое расширение Фортрана // Наука. 1985.
2. Зверев С. А., Михайлова Т. Н. «ГРАФОР: Комплекс графических программ на ФОРТРАНЕ. Построение изолиний и линий пересечения поверхностей» // Москва. Препринт ИПМ № 95 за 1982.