Изолинии — метод марширующих квадратов

Задача IS

Первый срок сдачи — 23 октября 2013 года.

Последний срок сдачи — 30 октября 2013 года (на семинаре).

Построить приложение, рисующее "портрет" однозначной функции 2-х переменных в виде цветной карты и карты изолиний.

1. Цветная карта. Итак, дана однозначная функция

$$z = f(x,y),\ (x,y) \in D = [a,b]*[c,d].$$

На клиентской области окна приложения выбираем прямоугольник **P** с углами **(u0,v0)** и **(u1,v1)** в экранных координатах (пикселях), ось **V** - сверху вниз. Пусть даны **n** чисел:

$$z1 < z2 < \ldots < zn,$$

и **n+1** цвет:

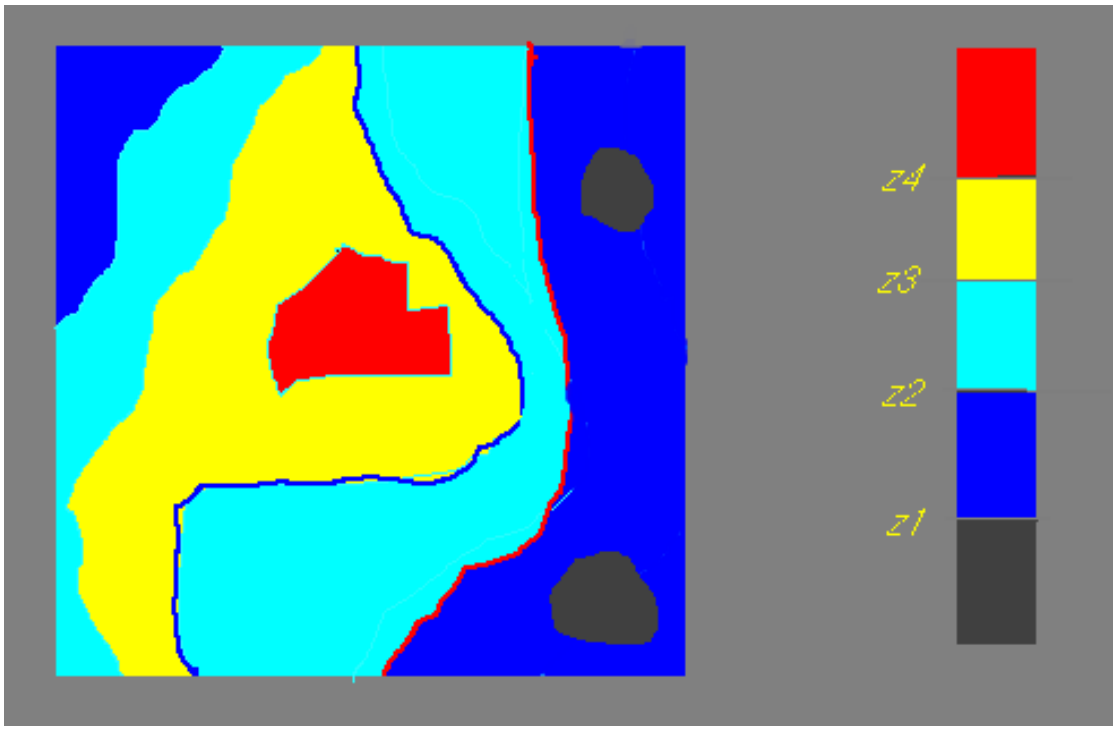
$$c0, c1, c2, \ldots, cn.$$

Будем говорить, что значению **z** соответствует цвет **c0**, если **z < z1**, иначе значению **z** соответствует цвет **ci (i = 1..n)**, где **i** определяется как

$$i = \min(j)\ \{z >= zj\}.$$

Рассмотрим довольно простое отображение **D** на **P**, когда точка **(a,d)** переходит в пиксель **(u0,v0)**, а точка **(b,c)** в **(u1,v1)**. Очевидное обратное преобразование позволяет по центру каждого пикселя **(u,v)** из **P** получать точку **(x,y)** из **D**. И, таким образом, пикселю присваивается цвет, соответствующий значению **f(x,y)**.

Полученное изображение прямоугольника **P** и будем называть цветной картой функции **f**. Справа от прямоугольника на свободном месте окна приложения рисуется вертикальная *легенда*, показывающая параметры цветной карты функции:



Границы, разделяющие зоны, раскрашенные в разные цвета, называются линиями уровня или изолиниями. Так, граница раздела областей, залитых цветами **i** и (**i - 1),** – это изолиния уровня **zi** – линия, являющаяся решением уравнения **f(x,y) = zi**.

Параметры задачи:

- **f(x,y) –** программируется внутри приложения. Выбор конкретной функции – за исполнителем. Локализовать в каком-либо методе (лучше — отдельный класс из одного метода);
- **a, b, c, d** – область определения функции;
- **n –** число ключевых значений, вводится из файла;
- Цвета **c0..cn** – вводятся из файла в виде 255 170 24. Одно значение цвета на строку файла;
- **z1..zn** – ключевые значения. Для определения этих значений: подсчитываются минимум и максимум функции (по узлам сетки), в полученном интервале равномерно распределяются остальные значения.

Примечание: полученную карту вместе с легендой вписать в клиентскую область окна. При изменении размеров окна картинка должна пересчитываться заново!

2. Изолинии сеточной функции. Имеем

$$a = x_1 < x_2 < \ldots < x_k = b,$$

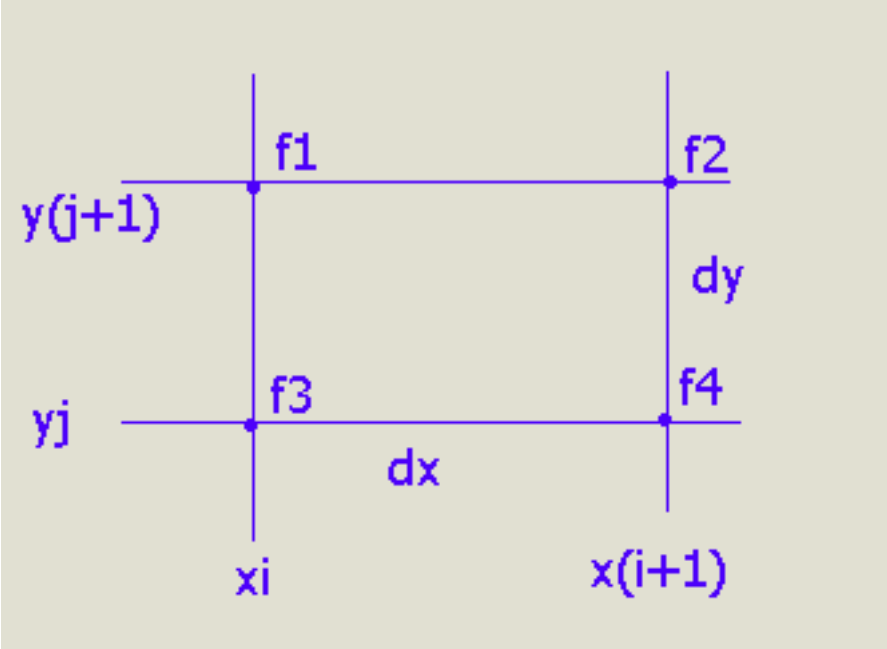
$$c = y_1 < y_2 < \ldots < y_m = d$$

некоторую сетку на прямоугольнике **D** (см. выше). И набор из **k*m** значений

$$z_{ij} = f(x_i, y_j)$$

некоторой неизвестной функции **f**, т.е. имеем функцию, заданную на сетке, аналог задания рельефа высотами на прямоугольной сетке. Предлагается реализовать следующий очень простой алгоритм марширующих кубиков (marching cubes).

1. Делается цикл по всем ячейкам сетки, т.е. построение изолинии на каждой ячейке делается независимо.
2. Пусть мы в настоящее время строим изолинию уровня **z** в очередной ячейке:

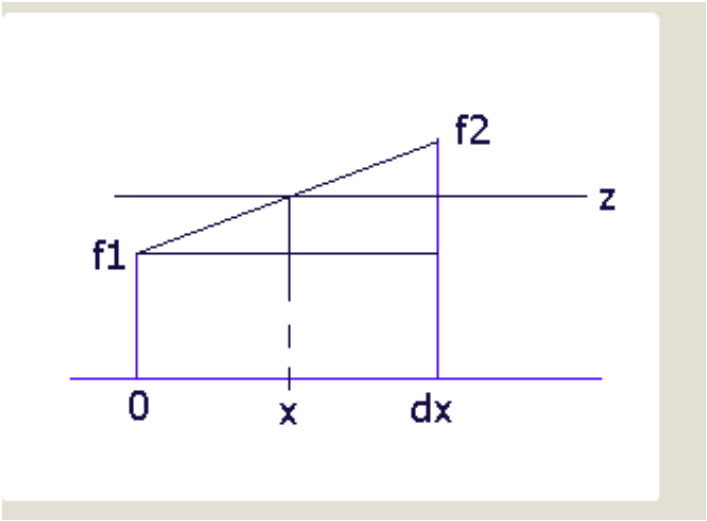


здесь **dx = x(i+1) - xi, dy = y(j+1) - yj**, а через **f1..f4** обозначены для краткости значения сеточной функции в соответствующих узлах.

3. Необходимо найти точки "входа/выхода" изолинии на границах ячейки (если таковые точки есть). Поскольку все 4 стороны рассматриваются независимо, то проведем поиск для верхней границы ячейки. В данном алгоритме считается, что неизвестная функция ведет себя между узлами линейно. Изолиния уровня **z** не пересекает верхнюю сторону ячейки, если

z < f1 & z < f2 или z > f1 & z > f2.

Иначе пересечение есть, найдем его, приняв для определенности, что **f1 < f2**.



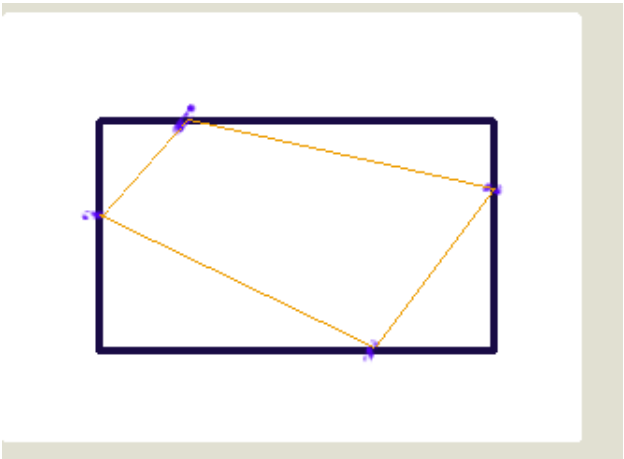
Получаем **x = dx * (z - f1) / (f2 - f1)** в пределах клетки, а на области определения функции точка входа изолинии в ячейку

(x[i] + (x[i+1] - x[i]) * (z - z[i, j+1]) / (z[i+1, j+1] - z[i, j+1]), y(j+1)).

Аналогично определяются и точки на других границах ячейки.

4. Рассмотрим возможные ситуации.

- Ни одной точки пересечения изолинии с границей ячейки. Это значит, что изолиния не проходит по ней.
- Ровно две точки, т.е. мы нашли точки входа на 2-х из 4-х сторон. Соединяем эти точки отрезком прямой линии.
- 4 точки, т.е. на каждой стороне имеется точка входа. Какие из этих точек соединять? Можно проверить значение функции в центре клетки.

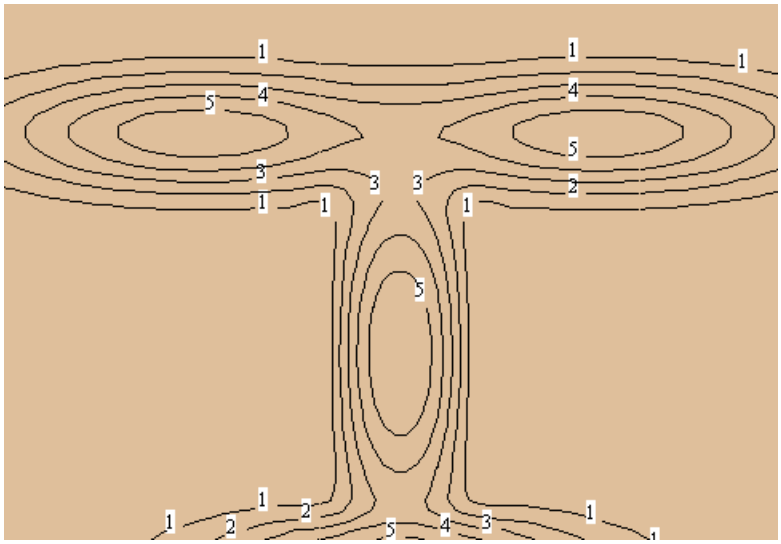


Задание.

1. Пусть сетка равномерная по каждой из осей. Насчитать сеточную функцию по функции **f**, запрограммированной в предыдущей части задания. Значения **k** и **m** взять из файла.

2. Цвет линии задать в файле.

Пример карты изолиний.



Преобразование координат D -> P.

(x,y) -> (u,v), на примере горизонтальной координаты

$$u = [(u1 - u0) * (x - a) / (b - a) + u0 + 0.5]$$

$$x = (b - a) * (u - u0) / (u1 - u0) + a$$

[..] - целая часть. Почему мы прибавляем **0.5**?

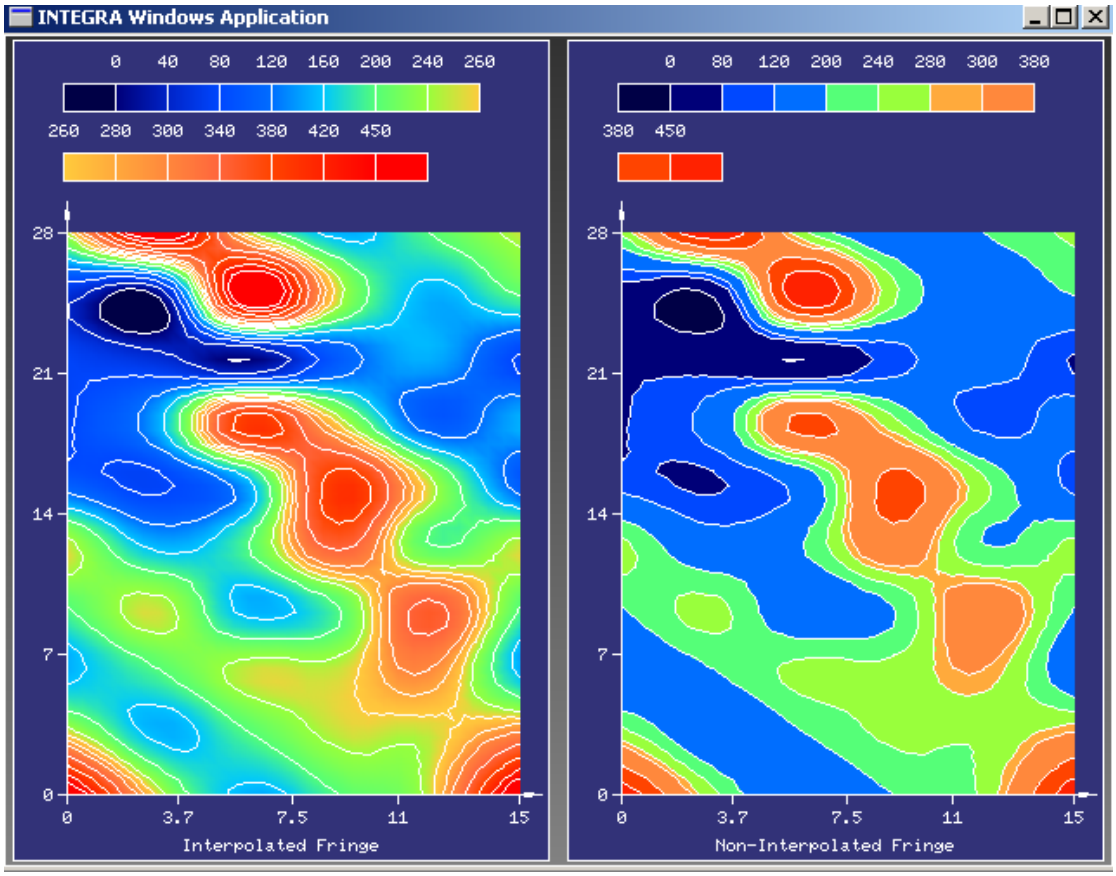
3. Интерполяция цвета. Режим "плавной" закраски. Легенда тоже должна иметь интерполированный вид в этом случае.

Надо проинтерполировать цвета от пикселя с координатой **u1** и цветом **(r1,g1,b1)** до пикселя с координатой **u2** и цветом **(r2,g2,b2)**. По каждой компоненте цвета интерполяция выполняется отдельно. Пусть нам надо получить красный **r** для пикселя с координатой **u**. Применяем формулу линейной интерполяции:

$$r = r1 * (u2 - u) / (u2 - u1) + r2 * (u - u1) / (u2 - u1).$$

Примеры карт с интерполяцией.

1) Слева применена интерполяция, справа нет.



4. Интерполяция с дизерингом Флойда-Стайнберга

Высчитывать промежуточный цвет, как в режиме интерполяции, но отображать только основными цветами **c0, c1, c2, …, cn,** используя алгоритм Флойда-Стайнберга (ошибки надо считать по каждой компоненте цвета по отдельности)

Требования к программе.

1. Читать параметры задачи из файла (должен быть приготовлен свой файл данных).
2. Изменять параметры задачи в диалоге (JDialog): k, m, a, b, c, d (обязательно), остальные (цвета) по желанию. В диалоге обязательно должны быть проставлены tab-stops. Диалог вызывается по кнопке на тулбаре.
3. Отображать функцию в одном из трёх видов: цветовая карта, интерполяция, интерполяция с дизерингом. Переключать виды кнопками на тулбаре. Перерисовывать легенду для каждого вида. Легенда — это как бы отдельная функция, равномерно возрастающая по вертикали, плюс подписи; будет красиво, если вы отобразите легенду с помощью того же кода, что и основную картинку, просто подсунув в этот код другую функцию.
4. Отображать поверх функции изолинии для уровней z1…zn; возможность включить и выключить изолинии кнопкой на тулбаре
5. Отображать поверх функции сетку k*m; возможность включить и выключить кнопкой на тулбаре; возможность сочетать с изолиниями

6. Добавить интерактивное построение изолиний по клику мыши: определяете по координатам пикселя, куда кликнули, точку в области определения функции f, берёте её значение и строите изолинию. Можно проводить изолинию через курсор, пока кнопка мыши прижата (тогда при движении мыши изолиния будет двигаться вместе с ней).

7. при движении мыши над полем значения **x**, **y** и **f(x,y)**, соответствующее точке экрана, выводится в определенное место, напр., в правом нижнем углу, под легендой. Или в статусбаре.

Замечание: легенда состоит не только из палитры используемых цветов, но и цифровых значений уровней.

Что будет отмечаться.

1. Имя ЗИПа, имя директории проекта, файл About, подсказки у кнопок и т.п. – это причина того, что программа не принята.
2. В описании точно указать, где в программе задается функция. Это должен быть отдельный метод.
3. Все действия осуществляются по кнопкам: режим отображения функции, вкл/выкл карты из изолиний, вкл/выкл сетки. Сетка строится либо при помощи линий (толщины 1), либо в виде узловых точек (отдельным цветом). Возможны варианты личных решений.
4. Отображение изолиний (Нарушение – максимум 2).
5. Поле изолиний и легенда должны ВПИСЫВАТЬСЯ в клиентскую область окна, т.е. необходимо производить расчет в программе. (Нарушение – максимум 3).
6. Отсутствие разрывов в изолиниях. (Нарушение – максимум 3).
7. Наличие анализа случаев 4-х и 3-х точек пересечения изолинии с клеткой (Нет – максимум 3).
8. Диалога для задания и смены параметров (Нарушение – минус балл).
9. Наличие слежения за мышью, т.е. отображение координат точки (не пиксели) в области определения функции **f** и самого значения функции (Нарушение – максимум 4).
10. Построение отдельных изолиний с помощью мыши (Нарушение – максимум 4).
11. Наличие режима интерполяции цветов (Нарушение – максимум 4).
12. Наличие дизеринга (Нарушение – максимум 4).

Формат файла:

```
k m       // число значений сетки по X и по Y
n         // число уровней
r0 g0 b0 // цвета легенды
r1 g1 b1
```

```
...
rn gn bn
ris gis bis // цвет построения изолиний
```