Билет 24

1. **Методы безусловной оптимизации нелинейного программирования**

Математическая постановка задачи оптимизации. Математическая постановка задачи безусловной оптимизации звучит следующим образом:

найти $X=(x_1, x_2, ..., x_n)$, при котором целевая функция $f_0(X)$ достигает экстремума.

Очевидно, что всякую задачу максимизации $f_0(X)$ можно заменить задачей минимизации функции $-f_0(X)$, поэтому в дальнейшем будем рассматривать оптимизационные задачи вида $f_0(X)\rightarrow\min$.

В зависимости от вида функций $f_0(X)$ выделяют частные случаи задачи оптимизации. Если $f_0(X)$ - линейная функция, то имеет место задача линейного программирования. Если функция $f_0(X)$ нелинейна, то задача есть задача нелинейного программирования.

Метод покоординатного спуска. Рассмотрим вначале метод покоординатного спуска Гаусса-Зейделя на примере функции двух переменных, линии равного уровня которой изображены на рис.2.6. Из некоторой начальной точки $x^0=(x_1^0, x_2^0)$ производится поиск минимума вдоль направления оси x_1 с получением точки X^0. В этой точке касательная к линии равного уровня параллельна оси x_1. Затем из точки X^0 производится поиск минимума вдоль направления оси x_2 с получением точки x^1. Следующие итерации выполняются аналогично. Для минимизации функции $f_0(x)$ вдоль осевых направлений может быть использован любой из методов одномерной минимизации.

Для улучшения метода в окрестностях экстремума можно применить процедуру деления шага.

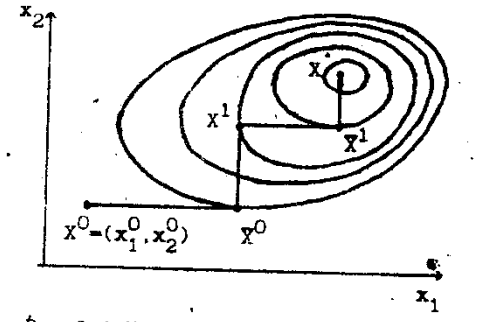

Рис.2.6 Метод покоординатного спуска Гаусса-Зейделя

Критерием окончания поиска является выполнение условия:

$$\left\|x^{k+1}-x^{k}\right\|\le\varepsilon,\quad \text{где}\quad \left\|x^{k+1}-x^{k}\right\|=\sqrt{\sum_{i=1}^{n}(x_i^{k+1}-x_i^{k})^2}$$

Метод Пауэлла. Для минимизации функций многих переменных М. Пауэлл предложил использовать следующую модификацию метода Гаусса-Зейделя. Покоординатным спуском из точки $\overline{X}^0$ осуществляется переход в точку $\overline{X}^1$. После выполняется поиск вдоль направления, соединяющего точки $\overline{X}^0$ и $\overline{X}^1$ (рис. 2.7), до тех пор, пока функция не начнет увеличиваться (точка X^1). Следующие итерации

выполняются аналогично.

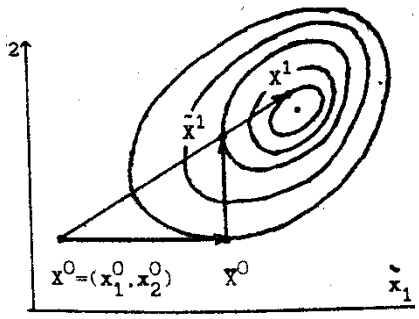

Рис.2.7. Метод Пауэлла

Симплексный метод. Множество (n+1)-й равноудаленной точки в n-мерном пространстве называется правильным симплексом. В случае n=2 правильным симплексом является равносторонний треугольник. Идея симплексного метода состоит в сравнении значений функции в (n+1) вершинах симплекса и перемещении его в направлении наилучшей точки.

Алгоритм решения задачи симплексным методом может быть представлен в виде следующей последовательности шагов. Задаются точки X^1, X^2,X^3, являющиеся вершинами начального симплекса. Вычисляются величины $f_1 = f(X^1)$, $f_2 = f(X^2)$, $f_3 = f(X^3)$ и выбирается наибольшее значение. Пусть это точка X^1. Тогда находится середина противоположного отрезка X^2 X^3 (точка А) и делается шаг в направлении X^1–>А в точку X^4. Точку X^1 забываем. Процедура продолжается относительно точек X^1 , X^2 , X^3. В окрестностях экстремума может применяться процедура сжатия симплекса.

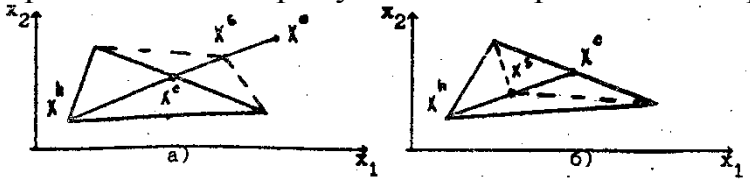

Рис.2.9. Операции сжатия симплекса

Метод наискорейшего спуска. В методе наискорейшего спуска используется свойство градиента функции в точке указывать направление наибыстрейшего возрастания (в направлении антиградиента - убывания) функции в точке.

Рассмотрим использование метода наискорейшего спуска для минимизации функции двух переменных $f_0(x_1, x_2)$. Пусть в качестве начального приближения выбрана некоторая точка $X^0 = (x_1^0, x_2^0)$. Каждая итерация при поиске минимума методом наискорейшего спуска состоит из двух этапов. На первом этапе в точке X^0 вычисляются значения частных производных по переменным x_1, x_2, которые определяют направление градиента функции $f_0(X)$ в этой точке. На втором этапе определяется следующая точка X^1 как $X^1 = X^0 + h_0 S^0$, где $S^0 = -\nabla f_0(X^0) = \left[-\frac{\partial f_0(X^0)}{\partial x_1} - \frac{\partial f_0(X^0)}{\partial X_2} \right]$ - антиградиент функции f_0 в точке X^0, h_0 - неизвестное значение коэффициента h.

Поиск заканчивается при выполнении следующего условия:

$$\sum_{i=1}^{n}\left[\frac{\partial f_0(X^k)}{\partial x_i}\right]^2 \le \varepsilon, \qquad (2.3)$$

где ε - заранее заданное положительное число.

Для приближенного вычисления частных производных целевой функции $f_0(X)$ по переменным x_i, i=l, ..., n, в точке X^k используется разностная схема:

$$\frac{\partial f_0(X^k)}{\partial x_i} = \frac{f_0(x_1, x_2, ..., x_i + \Delta x, ..., x_n) - f_0(x_1, x_2, ..., x_n)}{\Delta x}$$

где Δx - малое приращение по переменным x_i, i=1,..., n.

Геометрическая иллюстрация работы метода наискорейшего спуска приведена на рис.2.11.

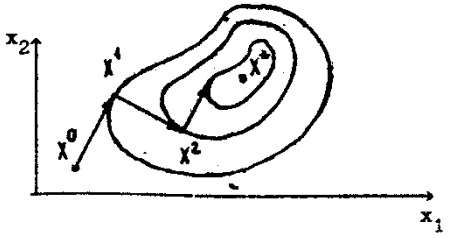

Рис.2.11 Метод наискорейшего спуска

2. Параллельное программирование. Примеры использования в промышленности.

Параллельное программирование - это техника программирования, которая использует преимущества многоядерных или многопроцессорных компьютеров и является подмножеством более широкого понятия многопоточности.

Параллельные вычисления - способ организации компьютерных вычислений, при котором программы разрабатываются, как набор взаимодействующих вычислительных процессов, работающих асинхронно и при этом одновременно.

Использование параллельного программирования становится наиболее необходимым, поскольку позволяет максимально эффективно использовать возможности многоядерных процессоров и многопроцессорных систем. По ряду причин, включая повышение потребления энергии и ограничения пропускной способности памяти, увеличивать тактовую частоту современных процессоров стало невозможно. Вместо этого производители процессоров стали увеличивать их производительность за счет размещения в одном чипе нескольких вычислительных ядер, не меняя или даже снижая тактовую частоту. Поэтому для увеличения скорости работы приложений теперь следует по-новому подходить к организации кода, а именно - оптимизировать программы под многоядерные системы.

Параллельное программирование включает в себя все черты более традиционного, последовательного программирования, но в параллельном программировании имеются три дополнительных, четко определенных ***этапа***.

- *Определение параллелизма*: анализ задачи с целью выделить подзадачи, которые могут выполняться одновременно
- *Выявление параллелизма*: изменение структуры задачи таким образом, чтобы можно было эффективно выполнять подзадачи. Для этого часто требуется найти зависимости между подзадачами и организовать исходный код так, чтобы ими можно было эффективно управлять

• *Выражение параллелизма*: реализация параллельного алгоритма в исходном коде с помощью системы обозначений параллельного программирования.

Основная сложность при проектировании параллельных программ — обеспечить правильную последовательность взаимодействий между различными вычислительными процессами, а также координацию ресурсов, разделяемых между процессами.

Способы синхронизации параллельного взаимодействия:

- *взаимодействие через разделяемую память*: на каждом процессоре мультипроцессорной системы запускается поток исполнения, который принадлежит одному процессу. Потоки обмениваются данными через общий для данного процесса участок памяти. Количество потоков соответствует количеству процессоров. Данный вид параллельного программирования обычно требует какой-то формы захвата управления (мьютексы, семафоры, мониторы) для координации потоков между собой;
- *взаимодействие с помощью передачи сообщений*: на каждом процессоре многопроцессорной системы запускается однопоточный процесс, который обменивается данными с другими процессами, работающими на других процессорах, с помощью сообщений.

3. Реалистическая графика. Обратная трассировка луча. Полная модель освещения Уиттеда.

С развитием вычислительной техники и ее возможностей все больший интерес вызывает задача построения по заданной модели сцены изображения, максимально приближенного к реальному. Построенное таким образом изображение должно восприниматься человеком как настоящее, реалистическое. Такие задачи возникают в самых разных областях человеческой деятельности - от рекламы и кинемотографа до архитектурных сооружений и создания реалистических изображений новых только проектируемых машин и т.п.

Рассмотрим самые простейшие методы создания реалистических изображений.

Прямой трассировкой лучей называется процесс расчета освещения сцены с испусканием от всех источников лучей во всех направлениях. При попадании на какой-либо объект сцены луч света может преломившись уйти внутрь тела или отразившись далее продолжить прямолинейное распространение до попадания на следующий объект и так далее. Следовательно, каждая точка сцены может освещаться либо напрямую источником, либо отраженным светом. Часть лучей в конце концов попадет в глаз наблюдателя и сформирует в нем изображение сцены.

Понятно, что вычисления, необходимые для трассировки всех лучей для всех источников и поверхностей слишком объемисты. Причем существенный вклад в полученное изображение внесет лишь небольшая часть оттрассированных лучей.

Для избавления от излишних вычислений используется ***обратная трассировка***, в которой вычисляются интенсивности только лучей, попавших в глаз наблюдателя. В простейшей реализации обратной трассировки отслеживаются лучи, проходящие из глаза наблюдателя через каждый пиксел экрана в сцену. На каждой поверхности сцены, на которую попадает луч, в общем случае формируются отраженный и преломленный лучи. Каждый из таких лучей отслеживается, чтобы определить пересекаемые поверхности. В результате для каждого пиксела строится дерево пересечений. Ветви такого дерева представляют распространение луча в сцене, а узлы

- пересечения с поверхностями в сцене. Окончательная закраска определяется прохождением по дереву и вычислением вклада каждой пересеченной поверхности в соответствии с используемыми моделями отражения. При этом различают и обычно по-разному рассчитывают первичную освещенность, непосредственно получаемую от источников света, и вторичную освещенность, получаемую от других объектов.

Итак, идея обратной трассировки лучей. Для определения цвета пиксела экрана через него из камеры проводится луч, ищется его ближайшее пересечение со сценой и определяется освещенность точки пересечения. Эта освещенность складывается из отраженной и преломленной энергий, непосредственно полученных от источников света, а также отраженной и преломленной энергий, идущих от других объектов сцены. После определения освещенности этой точки учитывается ослабление света при прохождении через прозрачный материал и в результате получается цвет точки экрана.

Для определения освещенности, привносимой непосредственным освещением, из точки пересечения выпускаются лучи ко всем источникам света и определяется вклад всех источников, которые не заслонены другими объектами сцены. Для определения отраженной и преломленной освещенности из точки выпускаются отраженный и преломленный лучи и определяется привносимая ими освещенность.

Алгоритм работы функции трассировки луча с началом o = (ox,oy,oz) и направлением d = (dx,dy,dz), возвращающей, кстати, освещенность по всем трем цветовым компонентам, будет, таким образом, выглядеть примерно так:

1. ищем ближайшее пересечение луча со сценой (определяем точку p, где произошло пересечение, выясняем, с каким конкретно объектом оно произошло)
2. если не нашли, возвращаем 0
3. если пересечение есть, текущую освещенность полагаем равной фоновой освещенности (ambient)
4. определяем вектор n, нормаль к объекту в точке пересечения для каждого (точечного) источника света
5. считаем вектор l, соединяющий источник и точку p (начало в p)
6. ищем пересечение луча с началом в точке p и направлением d со сценой
7. если нашли, прекращаем обработку этого источника, так как его не видно
8. считаем симметричный l относительно n вектор r
9. считаем по уравнению Фонга (оно использует n, l, r, d) освещенность в точке за счет этого источника света, добавляем ее к текущей
10. считаем симметричный d относительно n вектор refl, то есть отраженный вектор
11. трассируем отраженный луч (с началом p и направлением refl), добавляем полученную освещенность к текущей
12. считаем направление преломленного луча (вектор refr) по d, n и коэффициентам преломления
13. трассируем преломленный луч (с началом p и направлением refr), добавляем полученную освещенность к текущей
14. покомпонентно умножаем текущую освещенность на цвет объекта
15. если для материала, заполняющего пройденную лучом от o до p трассу beta != 0, умножаем текущую освещенность на коэффициент ослабления
16. возвращаем текущую освещенность

Простейшей моделью освещенности является ***модель Уиттеда***, согласно которой световая энергия, покидающая точку в направлении вектора **r**, определяется формулой

$$I = k_a I_a + k_d C \sum_j I_j (n, l_j) + k_s \sum_j I_j (s, r_j)^p + k_r I_r e^{-\beta_r d_r} + k_t I_t e^{-\beta_t d_t}$$

где k_a,k_d,k_s- веса фонового, диффузного и зеркального освещения;
k_r, k_t - веса отраженного и преломленного лучей;
C - цвет объекта в точке P;
n - вектор нормали в точке P;
l_j - направление на j-е источник света;
I_j - цвет j-го источника света;
b_r , b_t - коэффициенты ослабления для сред, содержащих отраженный и преломленный лучи;
d_r , d_t - расстояния от точки P до ближайших точек пересечения отраженного и преломленного лучей с объектами сцены;
s – вектор наблюдения
r_j – вектор отражения
h_j - вектор, задаваемый формулой
p - коэффициент Фонга.