Оглавление

1 Интерполяция функции двух переменных

Задача приближения функции одной переменной, которая рассматривалась до сих пор, естественным образом обобщается на случай функций нескольких переменных. Рассмотрим задачу интерполяции функций двух переменных на прямоугольной области. Дано: набор узлов $(x_i, y_j), i = \overline{0, n}, j = \overline{0, m}$ и значения неизвестной функции f в этих узлах: $f(x_i, y_j) = z_{ij}$. Требуется построить функцию удовлетворяющую условиям интерполяции: $(x_i, y_j) = z_{ij}$ Существует и более сложная версия интерполяции функции 2 переменных, в которой узлы задаются не в виде сетки, а произвольным образом. Но решение такой задачи в данной работе рассматриваться не будет.

1.1 Билинейная интерполяция

Ключевая идея заключается в том, чтобы провести обычную линейную интерполяцию сначала по одной переменной, затем по другой.
Пусть $x \in [x_i, x_{i+1}], y \in [y_j, y_{j+1}]$. Тогда

$$\begin{aligned} \varphi(x, y_j) &= z_{i,j} + (z_{i+1,j} - z_{i,j})\frac{x - x_i}{x_{i+1} - x_i} \\ \varphi(x, y_{j+1}) &= z_{i,j+1} + (z_{i+1,j+1} - z_{i,j+1})\frac{x - x_i}{x_{i+1} - x_i} \\ \varphi(x, y) &= \varphi(x, y_j) + (\varphi(x, y_{j+1}) - \varphi(x, y_j))\frac{y - y_j}{y_{j+1} - y_j} \end{aligned} \tag{1}$$

Временная сложность метода:

Построение φ: $O(nm)$ - требуется преобразовать входную сетку для возможности быстрого поиска нужного сегмента

Вычисление $\varphi(x, y)$: $O(\log n + \log m)$ - поиск сегмента

Пример:

```
; define grid
(def grid [[0 1 2]
           [3 4 5]
```

```
           [6 7 8]])

; build interpolation function
(interpolate-grid grid :bilinear)
```

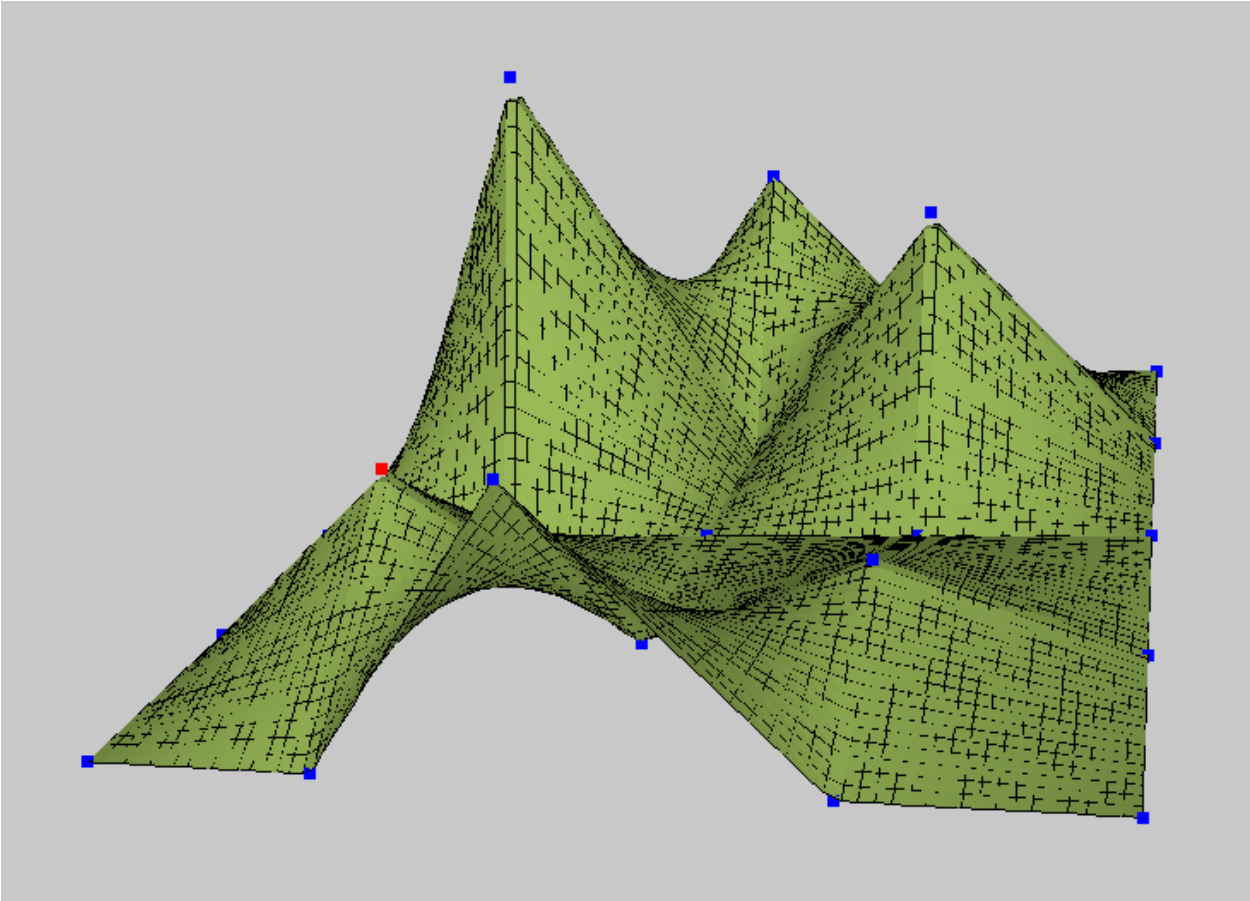

Рисунок 1 – Билинейная интерполяция

1.2 Полиномиальная интерполяция

Полиномиальная интерполяция функции 2 переменных была реализована по алгоритму, предложенному ТУТ. Матрица разделённых разностей P вычисляется по следующей формуле:

$$P_{i,j}^{(k)} = \begin{cases} f(x_i, y_j) & \text{if } k = 0 \\ \dfrac{P_{i,j}^{(k-1)} - P_{i-1,j}^{(k-1)}}{x_i - x_{i-k}} & \text{if } (j \leq k-1 \wedge i > k-1) \\ \dfrac{P_{i,j}^{(k-1)} - P_{i,j-1}^{(k-1)}}{y_j - y_{j-k}} & \text{if } (i \leq k-1 \wedge j > k-1) \\ \dfrac{P_{i,j}^{(k-1)} + P_{i-1,j-1}^{(k-1)} - P_{i,j-1}^{(k-1)} - P_{i-1,j}^{(k-1)}}{(x_i - x_{i-k})(y_j - y_{j-k})} & \text{if } (i > k-1 \wedge j > k-1) \end{cases} \tag{2}$$

Интерполяционный многочлен вычисляется по формуле:

$$\varphi(x, y) = Y^T P X \tag{3}$$

где

$$\begin{aligned} X &= \begin{pmatrix} 1 \\ x - x_0 \\ \vdots \\ (x - x_0)(x - x_1) \dots (x - x_{n-1}) \end{pmatrix} \\ Y &= \begin{pmatrix} 1 \\ y - y_0 \\ \vdots \\ (y - y_0)(y - y_1) \dots (y - y_{m-1}) \end{pmatrix} \end{aligned} \tag{4}$$

Временная сложность метода:

Построение φ: $O(nm \max(n, m))$ - вычисление разделённых разностей для двумерного случая.

Вычисление $\varphi(x, y)$: $O(nm)$

Пример:

```
; define grid
(def grid [[0 1 2]
```

```
            [3 4 5]
            [6 7 8]])

; build interpolation function
(interpolate-grid grid :polynomial)
```

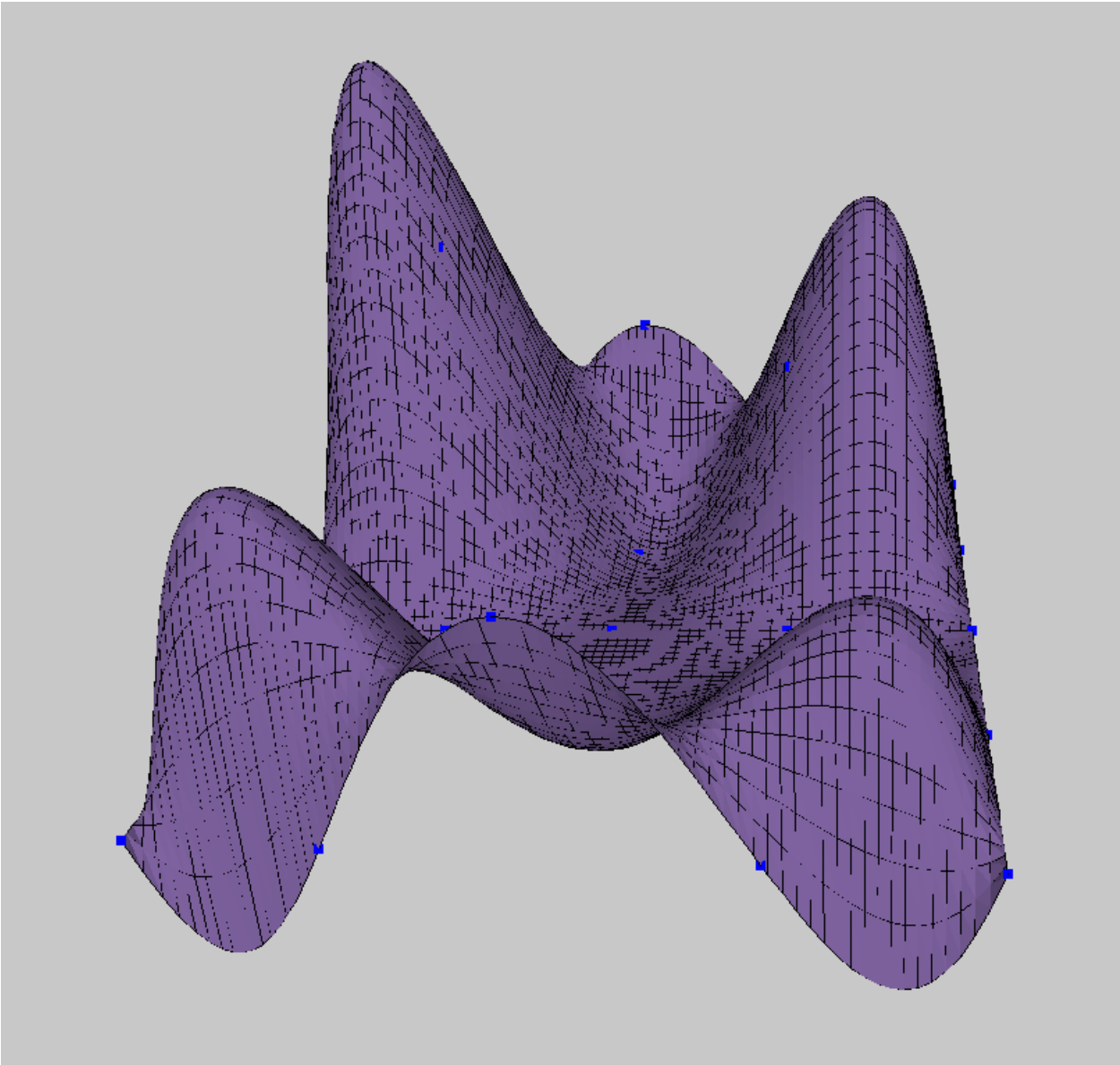

Рисунок 2 – Полиномиальная интерполяция

1.3 Бикубический сплайн

Бикубический сплайн является аналогом кубического сплайна. Будем искать функцию $\varphi(x, y)$ будем искать в виде:

$$\varphi(x,y) = \alpha_i(y)+\beta_i(y)(x-x_i)+\frac{\delta_i(y)}{y}(x-x_i)^2+\frac{\delta_i(y)}{6}(x-x_i)^3, x \in [x_{i-1}, x_i] \quad (5)$$

Здесь коэффициенты $\alpha_i(y), \beta_i(y), \delta_i(y), \gamma_i(y), i = \overline{1,n}$ являются одномерными кубическими сплайнами. Для построения функции φ необходимо построить данные одномерные сплайны. Но для их построения не хватает значений $\alpha_i(y), \beta_i(y), \delta_i(y), \gamma_i(y)$ в узлах $y_j, j = \overline{1,m}$.

Чтобы решить эту проблему зафиксируем $y = y_0$ в (5) и получаем одномерный кубический сплайн, вычисляя коэффициенты которого находим $\alpha_i(y_0), \beta_i(y_0), \delta_i(y_0), \gamma_i(y_0), i = \overline{1,n}$. Таким образом фиксируя y можно получить значения коэффициентов по всех остальных узлах. Далее по значениям коэффициентов в узлах строим сплайны $\alpha_i(y), \beta_i(y), \delta_i(y), \gamma_i(y)$.

Для вычисления значения φ в точке (x, y) необходимо по x определить отрезок $[x_{i-1}, x_i]$, которому принадлежит x. Далее требуется посчитать значения сплайнов $\alpha_i, \beta_i, \delta_i, \gamma_i$ в точке y. Полученные коэффициенты использовать в формуле (5) для получения конечного результата.

Временная сложность метода:

Построение φ: $O(nm)$
Вычисление $\varphi(x, y)$: $O(\log n + \log m)$

Пример:

```
; define grid
(def grid [[0 1 2]
           [3 4 5]
           [6 7 8]])

; build interpolation function
(interpolate-grid grid :bicubic)
```

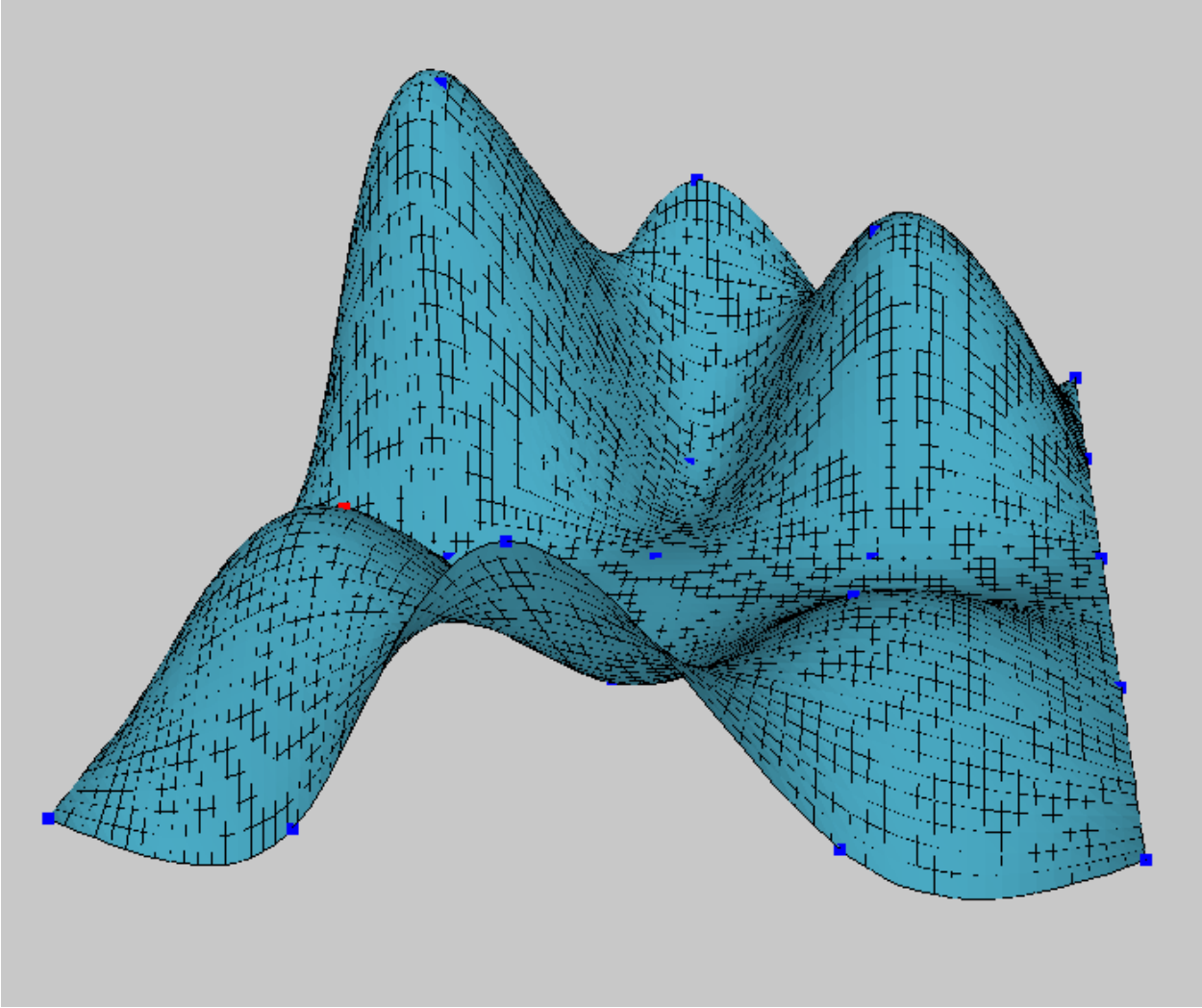

Рисунок 3 – Бикубический сплайн

1.4 Средневадратическое приближение

Среднеквадратическое приближение функции 2 переменных очень сильно похоже на среднеквадратическое приближения функции 1 переменной. Задаётся базис функций 2 переменных и вычисляются коэффициенты $\{\alpha_i\}$ при которых сумма расстояний от полученной поверхности до заданных точек минимальна. Положим $N \times M$ - размеры сетки, n - число базисных функций. Изменяются только формулы вычисления γ_{ij} и β_i:

$$\begin{aligned} \gamma_{ij} &= \sum_{k=0}^{N} \sum_{l=0}^{M} \varphi_i(x_k, y_l)\varphi_j(x_k, y_l) \\ \beta_i &= \sum_{k=0}^{N} \sum_{l=0}^{M} y_{kl}\varphi_i(x_k, y_l) \end{aligned} \tag{6}$$

Временная сложность метода:

Построение φ: $O(n^2NM + n^3)$ - построение матрицы коэффициентов γ_{ij} и решение СЛАУ.

Вычисление $\varphi(x)$: $O(n)$.

Пример:

```
; define grid
(def grid [[0 1 2]
           [3 4 5]
           [6 7 8]])

; basis consists of 7 funtions: 1, x, y, sin x, cos x, sin y, cos y
(defn basis [x y] [1 x y
                   (Math/sin x) (Math/cos x)
                   (Math/sin y) (Math/cos y)])

; build interpolation function
(interpolate-grid grid :linear-least-squares
                       :basis basis)
```

1.5 B-сплайновая поверхность

B-сплайновая поверхность для функции двух переменных строится используя тензорное произведение B-сплайнов для одномерного случая. Формула $\varphi(u, v)$ выглядит следующим образом:

$$\varphi(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} q_{ij} N_i^d(u) N_j^d(v) \tag{7}$$

Временная сложность метода:

Построение φ: $O(nm)$

Вычисление $\varphi(u, v)$: $O(d^2)$, где d - степень одномерных сплайнов.

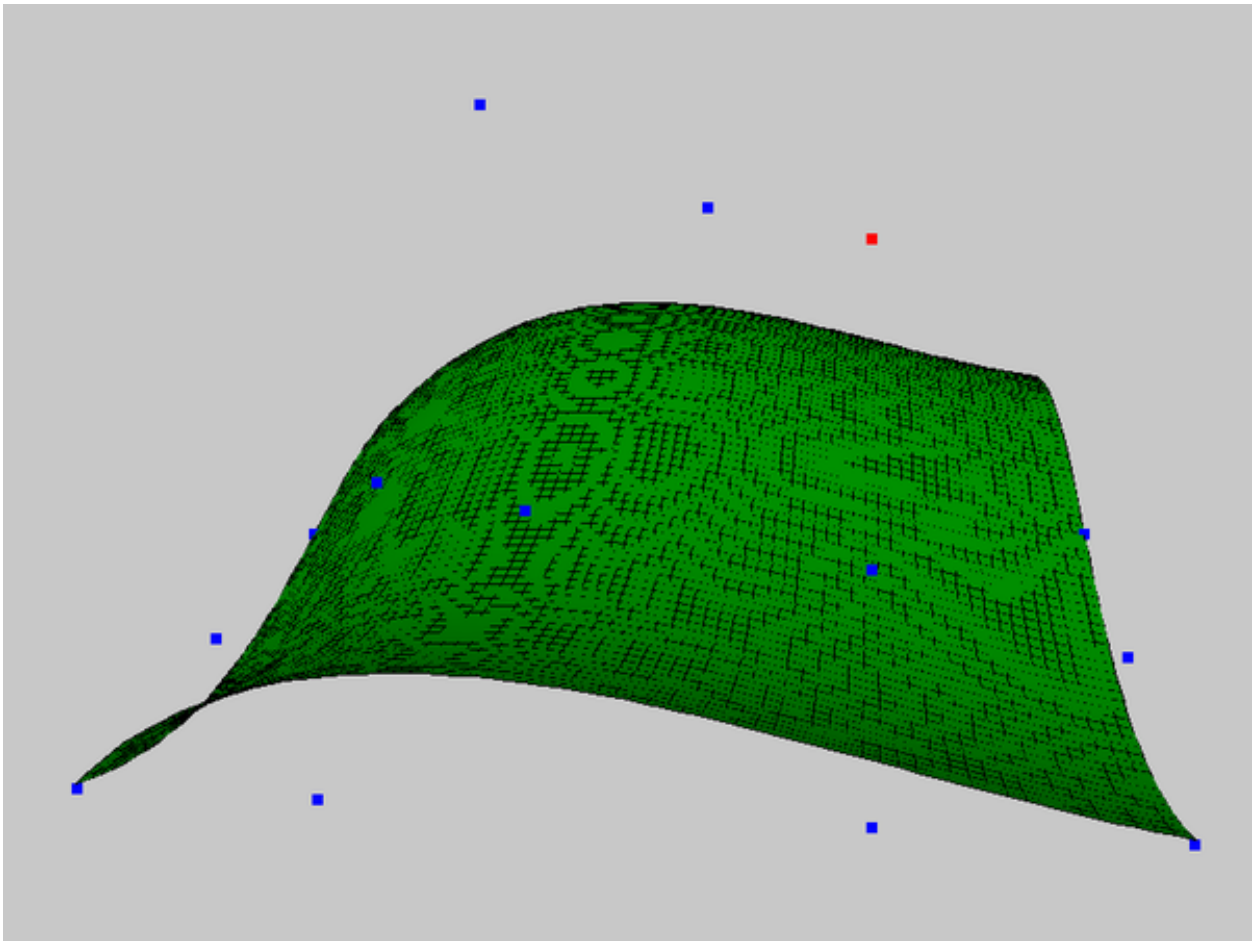

Рисунок 4 – Среднеквадратичное приближение по базису $1, x, y, sin(x), cos(x), sin(y), cos(y)$

Пример:

```
; define grid
(def grid [[0 1 2]
           [3 4 5]
           [6 7 8]])

; build interpolation function
(interpolate-grid grid :b-surface)
```

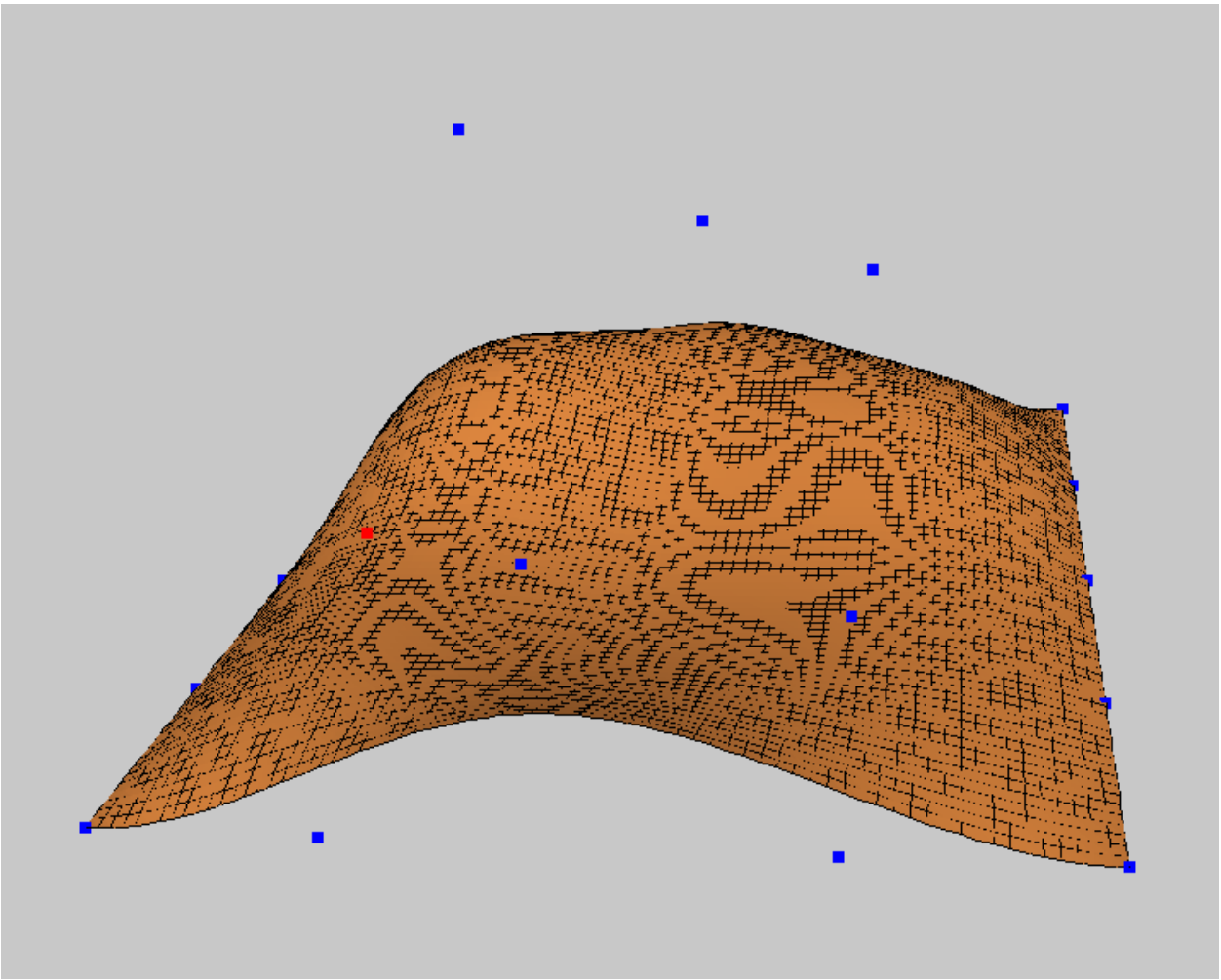

Рисунок 5 – B-сплайновая поверхность