МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ СИСТЕМ І ПРОЦЕСІВ

Лекція 8. Моделювання фрактального броунівського руху

Київ – 2015

Броуновское движение представляет собой пример естественного фрактала с фрактальной размерностью $d = 1.5$ (Мандельброт и Ван Несс, 1968). Впервые его наблюдал шотландский ботаник Роберт Броун[1] в 1827 году: он заметил непрерывное беспорядочное движение взвешенных в жидкости маленьких частиц (пыльцы), но ошибочно приписал причину движения самим частицам ([22]). Только в 1905 году Альберт Эйнштейн[2] и вслед за ним в 1906 году Мариан Смолуховский объяснили это движение хаотическими соударениями с молекулами окружающей среды ([21]). В 1908–1913 годах Жан Батист Перрен поставил ряд опытов ([23]), подтвердивших выводы Эйнштейна и Смолуховского. И, наконец, в 1923 год Норберт Винер построил первую математическую модель броуновского движения ([24, 25]). Альтернативные подходы были предложены Андреем Николаевичем Колмогоровым в 1933 году и Полем Леви в 1948 году ([26], [27], все три подхода см. в [28]).

Эту лекцию мы посвятим моделированию кривых классического броуновского движения, а также кривых и поверхностей фрактального броуновского движения (рассмотренные Б. Мандельбротом и В. Нессом [29]), пригодных для моделирования рельефа поверхности и других естественных процессов.

3.0.3 Классическое броуновское движение

Рассмотрим случайный процесс (случайную величину) $X(t)$, заданную на отрезке $[0, T]$.

Определение 3.1. *Случайный процесс $X(t)$ называется одномерным броуновским движением (или винеровским процессом) на интервале $[0, T]$, если он обладает следующими свойствами*[3]*:*

- $X(0) = 0$ *почти наверное и* $X(t)$ — *почти наверное непрерывная функция на* $[0, T]$
- $X(t)$ — *процесс с независимыми приращениями*[4]*;*
- $X(t)$ — *процесс с приращениями, распределёнными нормально*[5]*.*

Отметим следующие свойства броуновского движения:

- $X(t)$ почти наверное нигде не дифференцируем (см. [29]);
- $X(t)$ — марковский процесс (не обладает памятью), т.е. если известна величина $X(t)$, то при $t_1 < t < t_2$ величины $X(t_1)$ и $X(t_2)$ независимы;
- Фрактальная размерность графика $X(t)$ равна 1.5;
- Приращение $X(t)$ обладает свойством статистического самоподобия[6]: для любого $r > 0$

$$X(t + \triangle t) \triangleq \frac{1}{\sqrt{r}}(X(t + r\triangle t) - X(t)).$$

- Стационарность приращений: дисперсия приращения зависит только от разности моментов времени:

$$D(X(t_2) - X(t_1)) = \sigma^2|t_2 - t_1|. \tag{3.1}$$

- Математическое ожидание приращения равно

$$E(|X(t_2) - X(t_1)|) = \sqrt{2/\pi}\sigma\sqrt{|t_2 - t_1|}.$$

Для моделирования броуновского движения можно воспользоваться разными алгоритмами. Мы рассмотрим три из них.

Например, проще всего реализовать дискретную реализацию броуновского движения, рассмотрев последовательность $x_0 = 0$, $x_{n+1} = x_n + g_n$, где g_n — случайная величина, имеющая нормальное распределение (например, $N(0,1)$).

пример

```
load(distrib)$
N:1000$
X[0]:0$
for i:1 thru N do X[i]:X[i-1]+random_normal(0,1)$
plot2d([discrete,makelist(i,i,0,N),listarray(X)])$
```

3.0.4 Алгоритм срединных смещений

Метод *случайного срединного смещения* основан на работах Н. Винера ([24, 25]), он более сложен, чем метод из предыдущего параграфа, однако используется для конструктивного доказательства существования броуновского движения, а также для построения *фрактальной интерполяции* (когда необходимо, чтобы кривая проходила через заданные точки интерполяции). Метод также может быть обобщен на случай n-мерных броуновских движений.

Алгоритм случайного срединного смещения вычисляет значения $X(t)$ в диадических рациональных точках вида $\frac{k}{2^n} \in [0,1]$. Последовательно вычисляются значения в середине отрезка $[0,1]$, затем в серединах отрезков $[0,\frac{1}{2}]$ и $[\frac{1}{2},1]$ и т. д.. На каждом шаге итерации должен выполняться закон дисперсии для приращений (3.1) в вычисленных точках. Параметр σ определяет масштаб по вертикальной оси, не влияя на фрактальную размерность графика.

Алгоритм 3.1: **Броуновское движение методом срединного смещения (1)**

Вход: N; ✎ *число шагов алгоритма, при этом всего* $2^N + 1$ *точек интерполяции*

σ; ✎ *параметр вертикального масштаба, коэффициент дисперсии*

Выход: массив значений $\{X(\frac{k}{2^N})\}_{k=0}^{2^N}$. ✎ *реализация броуновского движения* $X(t)$ *на дискретном множестве точек вида* $t_k = \frac{k}{2^N}$, $k = \overline{0, 2^N}$

1: $X(0) = 0$;

2: $X(1) = \sigma g$ ✎ g — *случайная величина, распределенная нормально с параметрами* $N(0, 1)$

3: Шаг 1:

$$X\left(\tfrac{1}{2}\right) = \tfrac{1}{2}\left(X(0) + X(1)\right) + \tfrac{1}{2}\sigma g;$$

4: Шаг 2:

$$X\left(\tfrac{1}{4}\right) = \tfrac{1}{2}\left(X(0) + X\left(\tfrac{1}{2}\right)\right) + \tfrac{1}{2^{3/2}}\sigma g;$$

$$X\left(\tfrac{3}{4}\right) = \tfrac{1}{2}\left(X\left(\tfrac{1}{2}\right) + X(1)\right) + \tfrac{1}{2^{3/2}}\sigma g;$$

$\vdots$

5: Шаг N:

$$X\left(\tfrac{1}{2^N}\right) = \tfrac{1}{2}\left(X(0) + X\left(\tfrac{1}{2^{N-1}}\right)\right) + \tfrac{1}{2^{(N+1)/2}}\sigma g;$$

$\vdots$

$$X\left(\tfrac{2^N-1}{2^N}\right) = \tfrac{1}{2}\left(X\left(\tfrac{2^{N-1}-1}{2^{N-1}}\right) + X(1)\right) + \tfrac{1}{2^{(N+1)/2}}\sigma g;$$

Заметим, что точки $t_k = \frac{k}{2^N}$ можно последовательно занумеровать номерами k. При этом если точка имеет вид $\frac{a}{2^b}$, то её номер $k = a2^{N-b}$. Укажем алгоритм, в котором точки t_k пронумерованы эффективно.

Алгоритм 3.2: **Броуновское движение методом срединного смещения (2)**

Вход: N; ✎ *число шагов алгоритма, при этом всего $2^N + 1$ точек интерполяции*
σ; ✎ *параметр вертикального масштаба, коэффициент дисперсии*
Выход: массив значений $\{X(k)\}_{k=0}^{2^N}$. ✎ *реализация броуновского движения $X(t)$ на дискретном множестве точек вида $t_k = \frac{k}{2^N}$, $k = \overline{0, 2^N}$*

1: $X(0) = 0$;
2: $X(1) = \sigma g$ ✎ *g — случайная величина, распределенная нормально с параметрами $N(0,1)$*
3: **для** $j = 1, \ldots, N$
4: **для** $i = 1, \ldots, 2^{N-1}$
5: $X\left((2i-1)2^{N-j}\right) = X\left((i-1)2^{N-j+1}\right) + X\left(i2^{N-j+1}\right) + \frac{1}{2^{(j+1)/2}}\sigma g$; ✎ *это соответствует формуле $X\left(\frac{2i-1}{2^j}\right) = X\left(\frac{i-1}{2^{j-1}}\right) + X\left(\frac{i}{2^{j-1}}\right) + \frac{1}{2^{(j+1)/2}}\sigma g$*

Броуновское движение методом срединного смещения

```
1   load(distrib)$
2   N:10$
3   s:2$
4   [X[0],X[2^N]]:[0,s*random_normal(0,1)]$
5
6   for j:1 thru N do
7      for i:1 thru 2^(j-1) do
8         X[(2*i-1)*2^(N-j)]:float((X[(i-1)*2^(N-j+1)]+X[i*2^(N-j+1)])/2+
9                                  s*random_normal(0,1)/2^((j+1)/2))$
10
11  plot2d([discrete,makelist(k/2^N,k,0,2^N),listarray(X)]);
```

В строчке 8 используется команда `float`, чтобы хранить в памяти результат, а не последовательность действий (подумайте над этим).

3.0.5 Фрактальное броуновское движение

Фрактальное броуновское движение (ФБД) уже не является марковским процессом, а обладает некоторой «памятью» (Б. Мандельброт и В. Несс, [29]). Кроме того, вводя параметр $0 < H < 1$ можно получить одномерное ФБД размерности $d = 2 - H$ и двумерное ФБД размерности $d = 3 - H$. Заметим, что классическое броуновское движение получается как частный случай при $H = 0.5$. Для аппроксимации ФБД нет простого метода, вроде суммирования нормальных случайных величин, как в случае классического броуновского движения. Для аппроксимации ФБД наиболее удобно использовать преобразования Фурье.

Рассмотрим случайный процесс (случайную величину) $X(t)$, заданную на отрезке $[0, T]$.

Определение 3.2. *Случайный процесс* $X(t)$ *называется одномерным*[7] *фрактальным броуновским движением на интервале* $[0,T]$*, если он обладает следующими свойствами*[8]*:*

- $X(0)=0$ *почти наверное и* $X(t)$ — *почти наверное непрерывная функция на* $[0,T]$*;*
- $X(t)$ — *процесс с приращениями, распределенными нормально*[9]*.*

Отметим следующие свойства фрактального броуновского движения:

- $X(t)$ почти наверное нигде не дифференцируем (см. [29]);
- Фрактальная размерность графика $X(t)$ равна $2-H$;
- Процесс $x(t)$ не обладает свойством независимости приращений;
- Приращение $X(t)$ обладает свойством статистического самоподобия[10]: для любого $r>0$
$$X(t+\triangle t) \triangleq \frac{1}{r^H}(X(t+r\triangle t)-X(t)).$$
- Стационарность приращений: дисперсия приращения зависит только от разности моментов времени:
$$D(X(t_2)-X(t_1)) = \sigma^2|t_2-t_1|^{2H}. \tag{3.2}$$
- Математическое ожидание приращения равно
$$E(|X(t_2)-X(t_1)|) = \sqrt{2/\pi}\sigma|t_2-t_1|^H.$$

3.0.6 Метод Фурье-фильтрации для построения ФБД

Метод Фурье-фильтрации для аппроксимации ФБД заключается в следующем. Нам понадобится

Теорема 3.1 ([1]). *Если $X(t)$ — ФБД с параметром H, то его спектральная плотность*

$$S(f) \propto \frac{1}{f^{2H+1}}.$$

Идея метода состоит в следующем. Строится преобразование Фурье для искомого ФБД в частотной области, задавая случайные фазы и подбирая амплитуды, удовлетворяющие свойству из Теоремы 3.1. Затем получаем ФБД во временной области с помощью обратного преобразования Фурье.

Будем моделировать дискретный аналог ФБД, то есть наша цель — получить величины $\{X_n\}_{n=0}^{N-1}$, аппроксимирующие ФБД в точках n. Воспользуемся формулой дискретного преобразования Фурье

$$\hat{X}_n = \sum_{k=0}^{N-1} X_k e^{-2\pi k n/N}$$

и обратного дискретного преобразования Фурье

$$X_n = \sum_{k=0}^{N-1} \hat{X}_k e^{2\pi k n/N}.$$

Далее будем рассматривать только чётные значения N, а для применения метода *быстрого дискретного преобразования Фурье* нужно, чтобы $N = 2^M$, $M \in \mathbb{N}$. Метод быстрого дискретного преобразования Фурье реализован во многих системах компьютерной алгебры, в том числе и в Maxima. Он позволяет сократить вычисления в $\frac{2N}{\log_2 N}$ раз.

Для того, чтобы получающиеся величины X_n были вещественными, мы наложим условие сопряжённой симметрии:

$$\hat{X}_0, \hat{X}_{N/2} \in \mathbb{R}, \quad \hat{X}_n = \hat{X}_{N-n}, \quad n = 1, \ldots, N/2 - 1. \tag{3.3}$$

Фильтрация относится к той части моделирования, когда мы заставляем коэффициенты преобразования Фурье удовлетворять степенному закону из Теоремы 3.1:

$$|\hat{X}_n|^2 \propto \frac{1}{n^{2H+1}}, \quad n = 1, \ldots, N/2.$$

Для этого мы возьмём

$$\hat{X}_n = \frac{g e^{2\pi i u}}{n^{H+0.5}},$$

где g — независимые значения нормально распределённой случайной величины с параметрами $N(0,1)$, а u — независимые значения равномерно распределённой на отрезке $[0,1]$ случайной величины. Оставшиеся коэффициенты вычислим из соотношений (3.3).

Для вычисления искомой аппроксимации ФБД $\{X_n\}_{n=0}^{N-1}$ применим обратное дискретное преобразование Фурье к набору $\{\hat{X}_n\}_{n=0}^{N-1}$.

Алгоритм 3.3: **Кривая ФБД методом Фурье-фильтрации**

Вход: $H \in (0,1)$; ✎ *параметр ФБД, размерность графика равна* $d = 2 - H$
$N = 2^M, M \in \mathbb{N}$; ✎ *параметр, определяющий количество точек дискретизации ФБД*
Выход: массив значений $\{X_n\}_{n=0}^{N-1}$. ✎ *дискретная аппроксимация ФБД в последовательные моменты времени* n

---

1: $\hat{X}_0 = g$;
2: **для** $j = 1, \ldots, N/2 - 1$
3: $\quad \hat{X}_j = \frac{g e^{2\pi i u}}{j^{H+0.5}}$;
4: $\hat{X}_{N/2} = \frac{g \cos(2\pi i u)}{(N/2)^{H+0.5}}$; ✎ *Здесь* cos — *вещественная часть комплексной экспоненты выше*
5: **для** $j = N/2 + 1, \ldots, N - 1$
6: $\quad \hat{X}_j = \overline{\hat{X}_{N-j}}$;
7: $X = \text{ОДПФ}(\hat{X})$. ✎ *Вектор* $X = \{X_0, \ldots, X_{N-1}\}$ *получается обратным дискретным преобразованием Фурье из вектора* $\hat{X} = \{\hat{X}_0, \ldots, \hat{X}_{N-1}\}$.

Кривая ФБД методом Фурье-фильтрации

```
load(distrib)$
load(fft)$
N:2^4$
H:1$

X[0]:random_normal(0,1)$
for j:1 thru N/2-1 do X[j]:random_normal(0,1)*
                          exp(2*%pi*%i*random_continuous_uniform(0,1))/j^(H+0.5)$
X[N/2]:random_normal(0,1)*cos(2*%pi*random_continuous_uniform(0,1))/(N/2)^(H+0.5)$
for j:N/2+1 thru N-1 do X[j]:conjugate(X[N-j])$

Y:inverse_fft(rectform(ev(listarray(X),numer)))$

plot2d([discrete,makelist(i,i,0,N-1),realpart(Y)])$
```

`fft` — пакет для работы с быстрым дискретным преобразованием Фурье. Для применения команд этого пакета длина вектора-аргумента должна быть степенью 2. Вектор-аргумент может быть комплекснозначным, но все числа должны быть записаны в алгебраической форме.

`fft(x)` — возвращает массив (или список), являющийся быстрым дискретным преобразованием Фурье от массива (или списка) `x`;

`inverse_fft(x)` — возвращает массив (или список), являющийся быстрым обратным дискретным преобразованием Фурье от массива (или списка) `x`.

`random_continuous_uniform(a,b)` — псевдослучайная величина, распределенная равномерно на отрезке [a, b], команда пакета `distrib`;

`conjugate(x)` — возвращает комплексно-сопряжённое к комплексному числу `x`;

Зачем нужны команды `rectform` и `ev`?

`rectform(x)` — приводит комплексное число `x` к алгебраической форме $a + ib$; без приведения числа к алгебраической форме нельзя пользоваться командой `fft`;

`ev(x,numer)` — в данном случае вычисляет значение `x` с флагом `numer` (численно).

Рассмотрим пример, чтобы понять, зачем нам нужны команды `rectform` и `ev`:

пример

```
fpprintprec:14$
a:-0.1234*%e^(0.1234*%e*i*%pi);
float(a);
ev(a,numer);
rectform(a);
rectform(ev(a,numer));
```

```
(%o2)  -0.1234 %e^(0.1234 %i π)
(%o3)  -0.1234 2.718281828459^(0.1234 %i π)
(%o4)  -0.1234 (0.378034692412 %i+0.925791429715)
(%o5)  -0.1234 %i sin(0.1234 π)-0.1234 cos(0.1234 π)
(%o6)  -0.0466494810436 %i-0.114242662427
```

`fpprintprec:x` — константа, ограничивающая количество выводимых на экран знаков в десятичном представлении всех чисел в программе; она не изменяет сами числа и не влияет на выполнение программы.

Зачем использована команда `realpart`, ведь теоретически должны получаться вещественные значения при соблюдении условия сопряжённой симметрии (3.3)? При подсчёте обратного преобразования Фурье с конечной точностью заданных значений получаются погрешности, в данном случае порядка $10^{-16}i$.

Для построения аппроксимации двумерного фрактального броуновского движения методом Фурье-фильтрации используются те же идеи, что и в одномерном случае[11]. Вместо $\hat{X}_n$ используются $\hat{X}_{k,j}$, $k,j=\overline{0,N-1}$, условие Теоремы 3.1 примет вид ([30]):

$$|\hat{X}_{k,j}|^2 \propto \frac{1}{(k^2+j^2)^{H+1}}, \quad n,k=1,\dots,N/2,$$

мы возьмем[12]

$$\hat{X}_{k,j} = \frac{ge^{2\pi iu}}{(k^2+j^2)^{H/2+0.5}}, \quad n,k=1,\dots,N/2.$$

Запишем обратное дискретное преобразование Фурье: для $m,n=\overline{0,N-1}$

$$X_{m,n} = \sum_{k=0}^{N-1}\sum_{j=0}^{N-1} \hat{X}_{k,j}e^{-2\pi i\frac{kn+jm}{N}} = \hat{X}_{0,0} + \sum_{k=1}^{N-1}\hat{X}_{k,0}e^{-2\pi i\frac{kn}{N}} + \sum_{j=1}^{N-1}\hat{X}_{0,j}e^{-2\pi i\frac{jm}{N}} +$$
$$+ \sum_{k=1}^{N/2}\sum_{j=1}^{N/2}\hat{X}_{k,j}e^{-2\pi i\frac{kn+jm}{N}} + \sum_{k=\frac{N}{2}+1}^{N-1}\sum_{j=\frac{N}{2}+1}^{N-1}(\dots) + \sum_{k=1}^{N/2}\sum_{j=\frac{N}{2}+1}^{N-1}(\dots) + \sum_{k=\frac{N}{2}+1}^{N-1}\sum_{j=1}^{N/2}(\dots). \quad (3.4)$$

Из формулы (3.4) следует, что для вещественности всех величин $X_{m,n}$ достаточно выполнения следующих условий сопряжённой симметрии:

$$\hat{X}_{N-k,N-j} = \overline{\hat{X}_{k,j}}, \qquad k,j = \overline{1,N/2}; \qquad \hat{X}_{N/2,N/2} \in \mathbb{R}, \tag{3.5}$$

$$\hat{X}_{k,N-j} = \overline{\hat{X}_{N-k,j}}, \qquad k,j = \overline{1,N/2-1}; \qquad \hat{X}_{0,0} \in \mathbb{R}, \tag{3.6}$$

$$\hat{X}_{0,N-j} = \overline{\hat{X}_{0,j}}, \qquad j = \overline{1,N/2}; \qquad \hat{X}_{0,N/2} \in \mathbb{R}, \tag{3.7}$$

$$\hat{X}_{N-k,0} = \overline{\hat{X}_{k,0}}, \qquad k = \overline{1,N/2}; \qquad \hat{X}_{N/2,0} \in \mathbb{R}. \tag{3.8}$$

(очевидно, условия (3.7) (3.8) обеспечивают вещественность первых двух сумм, а условия (3.5) (3.6) — оставшихся четырёх сумм).

Алгоритм 3.4: **Поверхность ФБД методом Фурье-фильтрации**

Вход: $H \in (0,1)$; ✎ *параметр ФБД, размерность графика равна* $d = 3 - H$
$N = 2^M, M \in \mathbb{N}$; ✎ *параметр, определяющий количество точек ФБД по каждому из двух измерений*

Выход: массив значений $\{X_{n,k}\}_{n,k=0}^{N-1}$. ✎ *дискретная аппроксимация ФБД на решётке узлов*

для $j,k = \overline{1, N/2}$
  $\hat{X}_{j,k} = \frac{ge^{2\pi iu}}{(j^2+k^2)^{H/2+0.5}}$;
  $\hat{X}_{N-j,N-k} = \overline{\hat{X}_{j,k}}$;
для $k = \overline{1, N/2-1}$
  $\hat{X}_{0,k} = \frac{ge^{2\pi iu}}{(k^2)^{H/2+0.5}}$;
  $\hat{X}_{k,0} = \frac{ge^{2\pi iu}}{(k^2)^{H/2+0.5}}$;
  $\hat{X}_{0,N-k} = \overline{\hat{X}_{0,k}}$;
  $\hat{X}_{N-k,0} = \overline{\hat{X}_{k,0}}$;
для $j,k = \overline{1, N/2-1}$
  $\hat{X}_{N-j,k} = \frac{ge^{2\pi iu}}{((N-j)^2+k^2)^{H/2+0.5}}$;
  $\hat{X}_{j,N-k} = \overline{\hat{X}_{N-j,k}}$;
$\hat{X}_{0,0} = 0$;
$\hat{X}_{N/2,0} = \frac{g\cos(2\pi u)}{((N/2)^2)^{H/2+0.5}}$;
$\hat{X}_{0,N/2} = \frac{g\cos(2\pi u)}{((N/2)^2)^{H/2+0.5}}$;
$\hat{X}_{N/2,N/2} = \frac{g\cos(2\pi u)}{(2(N/2)^2)^{H/2+0.5}}$;
$X = \text{ОДПФ}(\hat{X})$; ✎ *Обратное дискретное преобразование Фурье матрицы* $\hat{X} = \{\hat{X}_{j,k}\}_{j,k=0}^{N-1}$

Поверхность ФБД методом Фурье-фильтрации

```
load(distrib)$
load(fft)$
load(draw)$
N:2^4$
H:1$

for j:1 thru N/2 do
   for k:1 thru N/2 do
      (Y[j,k]:random_normal(0,1)*exp(2*%pi*%i*random_continuous_uniform(0,1))/
            (j^2+k^2)^(H/2+0.5),
      Y[N-j,N-k]:conjugate(Y[j,k])
      )$
for k:1 thru N/2-1 do
   (Y[0,k]:random_normal(0,1)*exp(2*%pi*%i*random_continuous_uniform(0,1))/
         (k^2)^(H/2+0.5),
```

```
      Y[k,0]:random_normal(0,1)*exp(2*%pi*%i*random_continuous_uniform(0,1))/
            (k^2)^(H/2+0.5),
      Y[0,N-k]:conjugate(Y[0,k]),
      Y[N-k,0]:conjugate(Y[k,0])
     )$
for j:1 thru N/2-1 do
   for k:1 thru N/2-1 do
      (Y[N-j,k]:random_normal(0,1)*exp(2*%pi*%i*random_continuous_uniform(0,1))/
               ((N-j)^2+k^2)^(H/2+0.5),
      Y[j,N-k]:conjugate(Y[N-j,k])
      )$
Y[0,0]:0$
Y[N/2,0]:random_normal(0,1)*cos(2*%pi*random_continuous_uniform(0,1))/
        ((N/2)^2)^(H/2+0.5)$
Y[0,N/2]:random_normal(0,1)*cos(2*%pi*random_continuous_uniform(0,1))/
        ((N/2)^2)^(H/2+0.5)$
Y[N/2,N/2]:random_normal(0,1)*cos(2*%pi*random_continuous_uniform(0,1))/
          (2*(N/2)^2)^(H/2+0.5)$

X:map(inverse_fft,rectform(genmatrix(Y,N-1,N-1,0,0)))$
Z:realpart(map(inverse_fft,transpose(X)))$

draw3d(elevation_grid(Z,0,0,1,1),surface_hide = true);
```

`genmatrix(A,`j_1`,`j_2`,`i_1`,`i_2`)` — возвращает матрицу, соответствующую массиву A, причём элемент `A[`i_1`,,`j_1`]` является левым верхним элементом матрицы, а элемент `A[`i_2`,,`j_2`]` — нижним правым (аналог команды `listarray` для списков); если пропустить аргументы i_1 и j_1, то по умолчанию считается $i_1 = j_1 = 1$;

`transpose(x)` — возвращает транспонированную к `x` матрицу;

Заметим, что в силу формулы

$$X_{m,n} = \sum_{k=0}^{N-1}\sum_{j=0}^{N-1} \hat{X}_{k,j} e^{-2\pi i \frac{kn+jm}{N}} = \sum_{k=0}^{N-1} e^{-2\pi i \frac{kn}{N}} \sum_{j=0}^{N-1} \hat{X}_{k,j} e^{-2\pi i \frac{jm}{N}} = \sum_{k=0}^{N-1} \widehat{\hat{X}_k} e^{-2\pi i \frac{kn}{N}}$$

двумерное дискретное преобразование Фурье и обратное к нему можно получить, используя одномерные преобразования Фурье следующим образом: сначала применяем преобразование Фурье к каждой строчке матрицы $\hat{X}$, а затем — к каждому столбцу полученной матрицы. Эта схема реализована в нашем алгоритме в строках 35-36;

`elevation_grid(Z,`$x_0, y_0, \triangle x, \triangle y$ — команда, используемая внутри команды `draw3d`, которая рисует график поверхности, аппликаты точек которой заданы элементами матрицы `Z` $_{m,n}$, а абсциссы и ординаты точек — индексами матрицы. А именно, индексы матрицы отображаются на плоскость следующим образом: $_{1,1} \leftrightarrow (x_0, y_0 + \triangle y)$, $_{1,n} \leftrightarrow (x_0 + \triangle x, y_0 + \triangle y)$, $_{m,1} \leftrightarrow (x_0, y_0)$ и $_{m,n} \leftrightarrow (x_0 + \triangle x, y_0)$, задавая отображение всех индексов на прямоугольную сетку узлов на плоскости.

Эта команда поддерживает следующие опции, указываемые в команде `draw3d`:

`surface_hide = true` — отрисовывает только видимые части поверхности;

`line_type = dots` — тип линии сетки на поверхности: `solid` (сплошная, по умолчанию) и `dots` (точечная);

`line_width = x` — ширина линии сетки на поверхности (`x`> 0, по умолчанию 1);

`color = red` — задаёт цвет графика;

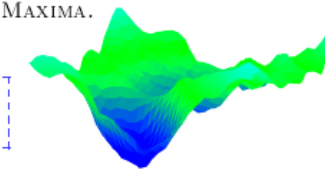
`enhanced3d(f(x,y,z),x,y,z)` — задает режим раскраски поверхности: `x,y,z` — явное указание переменных функции f, реализующей цвет точки (`x,y,z`) поверхности; кроме того, для этой опции можно (но не обязательно!) указать палитру цветов (по умолчанию она «марсианская»):

`palette=[`α, β, γ`]` — палитра, задаваемая тремя числами α, β, γ; заметим, её задание отличается для команды `plot3d`, описанной в справке по Maxima.

Попробуйте заменить строчку 38 на следующий код:

пример

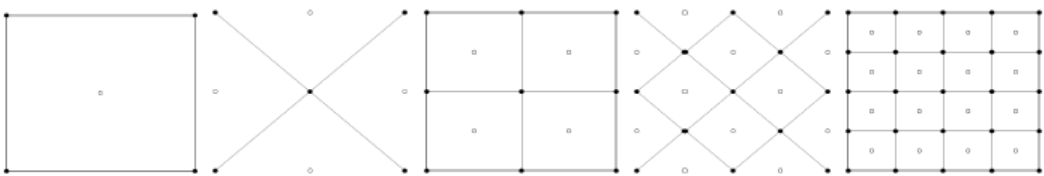
```
draw3d(palette=[0,30,25],enhanced3d=[z,x,y,z],
surface_hide=true, elevation_grid(m,0,0,1,1));
```



И напоследок приведем алгоритм, позволяющий строить аппроксимацию поверхности ФБД с помощью метода срединного смещения. Этот алгоритм работает сравнительно быстро и поэтому широко применяется, однако его реализация не полностью удовлетворяет определению ФБД.

Воспользуемся аналогией с одномерным методом срединных смещений для аппроксимации броуновского движения. Определим значения $X(t_1, t_2)$ в вершинах большого квадрата (см. схему ниже) и вычислим значение в его центре как среднее арифметическое значений его вершин плюс случайное смещение. Рассмотрим квадраты (треугольники, если их центры лежат на границе исходного квадрата), стороны которых в $\sqrt{2}$ раз меньше и параллельны диагоналям исходного квадрата, определим значение в них как среднее арифметическое значений окружающих их вершин плюс случайные смещения.

Повторим два этих шага необходимое число раз. Таким образом, на каждом шаге сначала вычисляются значения в центрах «обычных» квадратов, а затем — в центрах «диагональных» квадратов.



Алгоритм 3.5: **Поверхность ФБД методом срединного смещения**

Вход: $H \in (0,1)$; ✎ *параметр ФБД, размерность графика равна $d = 3 - H$*

σ; ✎ *параметр, определяющий вертикальный масштаб*

$N = 2^M, M \in \mathbb{N}$; ✎ *параметр, определяющий количество точек ФБД по каждому из двух измерений*

Выход: массив значений $\{X_{n,k}\}_{n,k=0}^{N}$. ✎ *дискретная аппроксимация ФБД на решётке узлов*

1: $X_{0,0} = X_{0,N} = X_{N,0} = X_{N,N} = 0$; ✎ *инициализация значений в вершинах большого квадрата*

2: $r = \sigma$;

3: $D = N$; ✎ *инициализация шагов решётки*

4: $d = N/2$;

5: **повторять** M **раз**

6: $r = r/2^{H/2}$; ✎ *меняем масштаб*

Считаем значения в центрах квадратов как среднее арифметическое их вершин плюс случайное приращение:

7: **для** $i, j = d, \ldots, N-2$ с шагом D

8: $X_{i,j} = (X_{i+d,j+d} + X_{i+d,j-d} + X_{i-d,j+d} + X_{i-d,j-d})/4 + rg$; ✎ *здесь и далее g — случайная величина, распределенная нормально с параметрами $N(0,1)$*

9: $r = r/2^{H/2}$; ✎ *меняем масштаб*

Считаем значения на границе большого квадрата как среднее значение трёх окружающих узлов плюс случайное приращение:

10: **для** $i = d, \ldots, N-d$ с шагом D

11: $X_{i,0} = (X_{i+d,0} + X_{i-d,0} + X_{i,d})/3 + rg$;

12: $X_{i,N} = (X_{i+d,N} + X_{i-d,N} + X_{i,N-d})/3 + rg$;

13: $X_{0,i} = (X_{0,i+d} + X_{0,i-d} + X_{d,i})/3 + rg$;

14: $X_{N,i} = (X_{N,i+d} + X_{N,i-d} + X_{N-d,i})/3 + rg$;

Считаем значения в центрах диагональных квадратов как среднее значение в вершинах этих квадратов плюс случайное приращение:

15: **для** $i = d, \ldots, N - d$ с шагом D
16: **для** $j = D, \ldots, N - D$ с шагом D
17: $X_{i,j} = (X_{i,j+d} + X_{i,j-d} + X_{i+d,j} + X_{i-d,j})/4 + rg;$
18: **для** $i = D, \ldots, N - D$ с шагом D
19: **для** $j = d, \ldots, N - d$ с шагом D
20: $X_{i,j} = (X_{i,j+d} + X_{i,j-d} + X_{i+d,j} + X_{i-d,j})/4 + rg;$
21: $D = D/2;$
22: $d = d/2;$

Поверхность ФБД методом срединного смещения

```
1   load(distrib)$
2   load(draw)$
3   H:1$                                    /* параметр размерности поверхности */
4   s:2$                                    /* параметр вертикального масштаба */
5   M:6$                                    /* число шагов алгоритма */
6
7   X[0,0]:0$              /* инициализация значений в точках первого квадрата */
8   X[0,N]:0$
9   X[N,0]:0$
10  X[N,N]:0$
11  N:2^M$
12  r:s$
13  D:N$                                    /* инициализация шага решётки */
14  d:N/2$
15
```

```
thru M do
   (
   r:r/2^(H/2),
   for i:d thru N-d step D do
      for j:d thru N-d step D do
         X[i,j]:float((X[i+d,j+d]+X[i+d,j-d]+X[i-d,j+d]+X[i-d,j-d])/4+
               r*random_normal(0,1)),
   r:r/2^(H/2),
   for i:d thru N-d step D do
      (
      X[i,0]:(X[i+d,0]+X[i-d,0]+X[i,d])/3+r*random_normal(0,1),
      X[i,N]:(X[i+d,N]+X[i-d,N]+X[i,N-d])/3+r*random_normal(0,1),
      X[0,i]:(X[0,i+d]+X[0,i-d]+X[d,i])/3+r*random_normal(0,1),
      X[N,i]:(X[N,i+d]+X[N,i-d]+X[N-d,i])/3+r*random_normal(0,1)
      ),
   for i:d thru N-d step D do
      for j:D thru N-D step D do
         X[i,j]:(X[i,j+d]+X[i,j-d]+X[i+d,j]+X[i-d,j])/4+r*random_normal(0,1),
   for i:D thru N-D step D do
      for j:d thru N-d step D do
         X[i,j]:(X[i,j+d]+X[i,j-d]+X[i+d,j]+X[i-d,j])/4+r*random_normal(0,1),

   D:D/2,
   d:d/2
   )$
Y:genmatrix(X,N,N,0,0)$
draw3d(elevation_grid(Y,0,0,1,1),surface_hide = true);
```

Здесь цикл `for` используется с дополнительной опцией `step`, означающей шаг инкремента цикла. Кроме того, подумайте, с какой целью используется команда `float` в строчке 22.

ДОМАШНЄ ЗАВДАННЯ

Реализуйте алгоритм построения аппроксимации ФБД с помощью ковариационной матрицы. Алгоритм состоит в следующем. Необходимо получить аппроксимацию ФБД $X(t)$ в точках $t_1, \ldots, t_n$. Для этого:

- Находим матрицу $\Gamma = (r(t_i, t_j))_{i,j=1}^n$, где $r(t,s) = \frac{1}{2}\left(s^{2H} + t^{2H} - |t-s|^{2H}\right)$;
- Находим $\Sigma = \sqrt{\Gamma}$ (т.е. $\Sigma^2 = \Gamma$) — соответствует матрице стандартного отклонения, ассоциированной с ковариационной матрицей Γ;
- Пусть $v = \{g_1, \ldots, g_n\}$ – вектор, составленный из n нормальных случайных величин с параметрами $N(0,1)$;
- Тогда $X = \Sigma v$.

Как вычислить матрицу Σ? Для этого можно использовать разложение Холецкого[13] (попробуйте!). Здесь же укажем другой способ (посредством собственных значений матрицы Γ):

- Поскольку Γ — симметричная положительно-определённая матрица, то её собственные значения λ_i вещественны и $\lambda_i \geq 0$, $i = \overline{1,n}$;
- Пусть $\Lambda = Diag(\lambda_i)$, $\Lambda_{i,j} = \lambda_i \delta_{ij}$, — диагональная матрица собственных значений, и пусть $\Lambda^{1/2} = Diag(\lambda_i^{1/2})$ — диагональная матрица с элементами $\Lambda_{ij}^{1/2} = \lambda_i^{1/2}\delta_{ij}$ (δ_{ij} — символ Кронекера);
- Пусть P — матрица, столбцами которой являются собственные векторы v_i, отвечающих собственных значениям λ_i. Заметим, что матрица P обратима ввиду линейной независимости собственных векторов.
- Тогда $\Sigma = P\Lambda^{1/2}P^{-1}$ (так как $\Gamma = P\Lambda P^{-1}$).