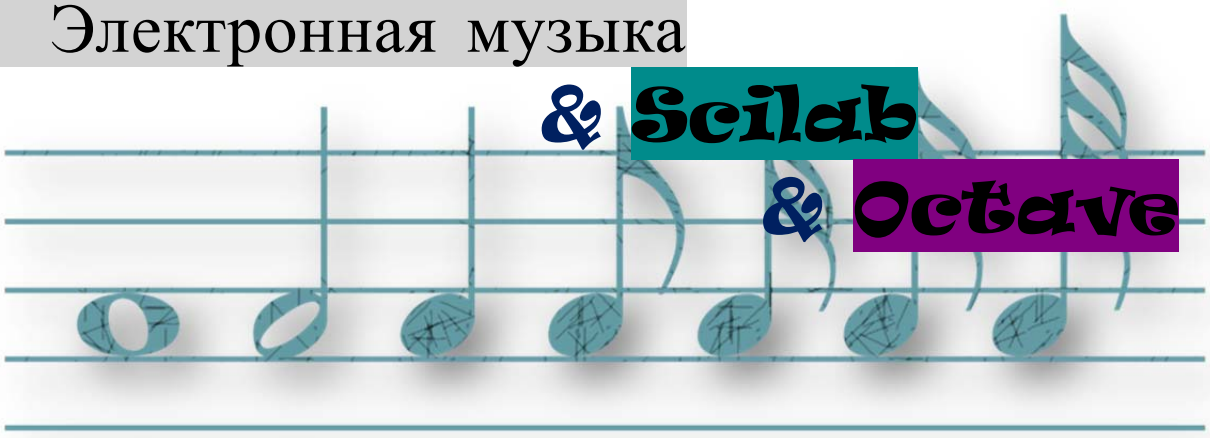
С. В. Ерин

Электронная музыка & Scilab & Octave



Москва 2019

Предисловие

В настоящее время большинство современной музыки создаётся с использованием компьютерных технологий. Если раньше для создания такой музыки требовались сложные электронные устройства, то теперь достаточно наличия компьютера. Компьютер видится идеальным инструментом из-за своей относительной дешевизны, технической лёгкости создания композиций, их записи и воспроизведения.

Важнейшей частью процесса создания электронной музыки на компьютере является наличие специализированного "software" –программного обеспечения. Но чтобы понять основы создания музыки на компьютере, достаточно использовать свободно распространяемые математические системы. На протяжении всего изложения будут использоваться такие пакеты, как : Scilab, Octave. Примеры, написанные на языке этих пакетов, легко могут быть адаптированы для исполнения в Matlab (коммерческий продукт) и наоборот.

Глава 1.

Синусоида, амплитуда, частота, частота выборки (сэмплинг).

Давайте определимся, что мы называем электронной музыкой. Википедия даёт следующее определение: "Электро́нная му́зыка — музыка, созданная с использованием электромузыкальных инструментов и электронных технологий (с последних десятилетий XX века — компьютерных технологий). Электронная музыка оперирует звуками, которые способны издавать электронные и электромеханические музыкальные инструменты, а также звуками, возникающими при помощи электрических / электронных устройств и различного рода преобразователей (магнитофоны, генераторы, компьютерные звуковые карты, звукосниматели и т.п.), которые в строгом смысле не являются музыкальными инструментами".

В настоящее время для создания электронной музыки (сокращённо - э.м.) повсеместно используется компьютер, с помощью которого происходит синтез и обработка цифровых аудиосигналов (так называемый- процессинг). Цифровой аудиосигнал –это последовательность чисел:

$$..., y[n-1], y[n], y[n+1], ... \qquad (1)$$

где: n- номер значения амплитуды сигнала, в выбранный момент времени, который может меняться в широком диапазоне целых чисел. Одиночное число в приведённой последовательности называется – отсчёт (часто называют – сэмпл, но надо иметь ввиду, что сэмплом называют также набор отсчётов (выборка) какого-нибудь аудиосигнала). Покажем, как из непрерывного аналогового сигнала можно получить цифровой аудиосигнал. Пусть сигналом является синусоида, которая генерирует звук нужной нам частоты. Синусоида играет ключевую роль в процессинге аудиосигнала, поскольку не изменяет форму из-за сдвига, и сложные звуковые сигналы (музыкальные эффекты) можно представить в виде комбинации синусоид.

$$y(t) = A \cdot sin(\Omega t + \Phi) \qquad (2)$$

где: A – амплитуда сигнала, Ω – угловая частота, t- время звучания в секундах [c], Φ – начальная фаза. Рассмотрим случай, когда начальная фаза равна нулю. Угловая частота Ω есть произведение : $\Omega=2\pi f$, где f – частота звука в герцах [$Гц$]. Надо отметить, что цифровой аудиосигнал не связан каким-либо внутренним соотношением со временем. И чтобы его слышать, нам необходимо выбрать частоту дискретизации аналогового сигнала (sample rate), её ещё называют частотой выборки, которую обозначим как R. Частота

дискретизации – это количество сэмплов в 1 секунду, т.е. значения амплитуды аналогового сигнала в моменты времени, которые определяются частотой дискретизации. Измеряется в герцах. Исходя из этого, запишем соотношения, связывающие общее количество отсчётов *n* (пропорционально длительности нашего сигнала) с частотой дискретизации f_s и временем длительности сигнала – *t*:

$$f_s t = n, \quad t = n/f_s \qquad (3)$$

И тогда цифровой аудио сигнал будет выглядеть следующим образом:

$$y[n] = A \cdot \sin(2\pi f \cdot n + \Phi) \qquad (4)$$

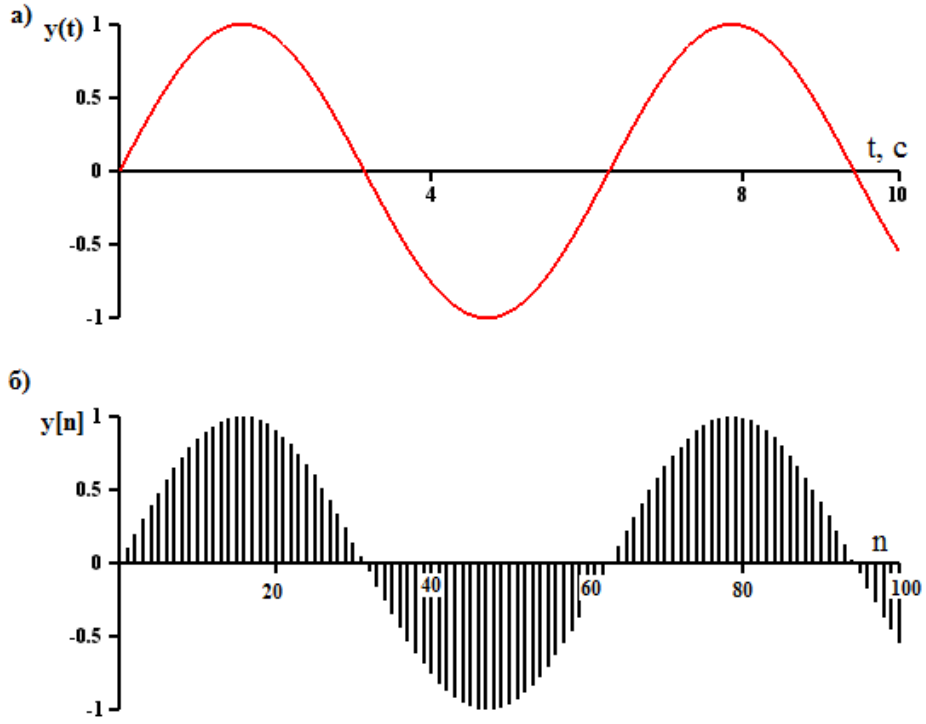


Рис.1 а) Аналоговый синусоидный аудио сигнал б) цифровой аудио сигнал, полученный из аналогового сигнала

Давайте смоделируем процесс цифровизации аудио сигнала с помощью таких бесплатных программ, как: Scilab и Octave [Ссылки].

Отметим, что в Scilab и в Octave можно работать со звуковой картой компьютера, записывать и воспроизводить звуковые файлы. Обсудим, как можно генерировать различные типы сигналов. Стоит отметить, что в Scilab и Octave очень просто оперировать с любыми элементарными (в том числе тригонометрическими) функциями. Будем использовать функцию *sin(x)* для генерации звука нужной нам частоты. Аргумент *x* означает какое-то

абстрактное значение угла, пока не установлена частота выборки (сэмплинг) значений функции в единицу времени (секунду) и частота звука. Учитывая это, запишем значение аргумента x, используя синтаксис Scilab:

```
x= 2*%pi*f*t ;
```

где- `%pi` – значение пи, `f`- частота звука [Гц], `t`-время звучания [сек]. Как известно, современные звуковые карты компьютеров поддерживают частоты дискретизации 8.0, 11.025, 22.05 и 44.1 кГц, а профессиональные и больше. Частота дискретизации (оцифровки) сигнала должна быть как минимум в два раза больше максимальной частоты входного сигнала (теорема Котельникова-Найквиста). Если человеческая речь, например, занимает полосу частот до 3-4 кГц, то для ее оцифровки потребуется частота 8 кГц и т.п. Таким образом, нужно связать длительность звучания t с частотой дискретизации звуковой карты f_s. Это мы проделали выше (3). Тогда аргумент x будет выглядеть следующим образом:

```
x= 2*%pi*(f/fs)*n;
```

Теперь получим для примера синусоидальный звуковой сигнал длительностью 2 секунды частотой 440 Гц. Пусть частота дискретизации звуковой карты 44100 Гц.

Программа в Scilab.

```
n=0:88200;// длительность сигнала в отсчётах
R=44100;// частота дискретизации
f=440;// частота сигнала
x =2*%pi *(f/fs)*n;// аргумент функции
y=sin(x);//синусоидальный сигнал
sound(y)// воспроизведение сигнала
plot(x(1:500),y(1:500))//график пятьсот первых точек сигнала
```

В Scilab для воспроизведения звука есть две команды: `playsnd` и `sound` (см. help). С помощью команды sound можно воспроизводить данные в виде вектора, тогда как командой `playsnd` возможно воспроизведение данных в виде матрицы, причём каждая строка является отдельным каналом (для звуковой карты с двумя аудио каналами число строк в матрице должно быть не больше 2, с большим числом строк не работает). Если мы зададим наш

сигнал в виде $y = A*sin(x)$, где A –число, то можем легко изменять амплитуду сигнала. Стоит отметить также, что в Scilab есть команда: `soundsec(t[,fs])`,с помощью которой можно сгенерировать вектор сэмплов для сигнала длительностью t сек. при частоте дискретизации f_s [Гц]. Т.е. если `t = soundsec(2,fs)`, то можно записать наш аргумент: `x=2*%pi*f*t`.

Наиболее часто используемая операция над электронными звуками - это изменение их амплитуды. Например, синтезировать звуки можно объединением синусоид. Но синусоида имеет постоянную номинальную амплитуду, а нам бы хотелось иметь возможность изменять её во времени. Тогда для умножения амплитуды сигнала $y[n]$ на коэффициент $x \geq 0$, можно просто умножить каждый сэмпл на x, и мы получим новый сигнал $x \cdot y[n]$.

В более общем случае, например, если нужно модулировать сигнал $y[n]$ другим сигналом $x[n]$, то каждый сэмпл конечного сигнала будет произведением y [n] · x [n] (в фиксированном окне от K до K+ N - 1).

Давайте посмотрим, как можно задать сигнал и записать их в файл с расширением .wav. Пусть частота для всех форм сигналов будет 440 Гц. Для синусоидального сигнала это будет выглядеть так:

```
f=440; // частота сигнала
sin_s = sin(2*%pi*f*t);
xx=pwd();//текущая директория
sinfile = xx+'\sin'+string(f)+'.wav';
wavwrite(sin_s, fs, bits, sinfile);
```

где bits может быть: 8, 16, 24, 32 бита – количество битов, используемых для записи информации для каждого сэмпла (16 – по умолчанию).

Давайте проделаем то же самое в программе Octave.

```
n=0:88200;# длительность сигнала в отсчётах
R=44100;#частота дискретизации
f=440;# частота сигнала
x =2*pi *(f/fs)*n;#аргумент функции
y=sin(x);#синусоидальный сигнал
plot(x(1:500),y(1:500))#график пятьсот первых точек сигнала
player = audioplayer (y, 44100, 8);
play (player);
```

Как можно видеть из приведённых примеров, отличия между Scilab и Octave незначительные и касаются функций для создания и использования объектов аудиоплеера. Эти объекты можно использовать для воспроизведения аудиоданных, хранящихся в матрицах и массивах Octave. Объект аудиоплеера поддерживает воспроизведение с различных устройств, доступных для системы, блокирующее и неблокирующее воспроизведение, удобную приостановку и возобновление, и многое другое.

```
player = audioplayer (y, fs)
player = audioplayer (y, fs, nbits)
player = audioplayer (y, fs, nbits, id)
player = audioplayer (recorder)
player = audioplayer (recorder, id)
```

Необязательные аргументы nbits и id определяют число бит при оцифровке амплитуды сигнала, и идентификатор устройства проигрывателя соответственно. Идентификаторы устройств можно найти с помощью функции `audiodevinfo`. Сигнал y может быть вектором или двумерным массивом.

1.1 Измерения амплитуды

Как понятно из предыдущего параграфа, цифровой сигнал – это набор измеренных амплитуд аналогового сигнала, полученных в определённые моменты времени (т.е. все сэмплы цифрового аудио сигнала являются амплитудами). Возникает вопрос: Насколько точно амплитуда цифрового сигнала соответствует амплитуде аналогового сигнала?

Отметим, что процесс оцифровки сигнала приводит к дополнительным шумам в системе записи, которые называются ошибками квантования, или шумами квантования.

Их характер можно проследить наглядно, если сравнить отклонения графика оцифрованной волны от оригинальной синусоиды. На рисунке 2 показана разница между исходным и оцифрованным сигналом.

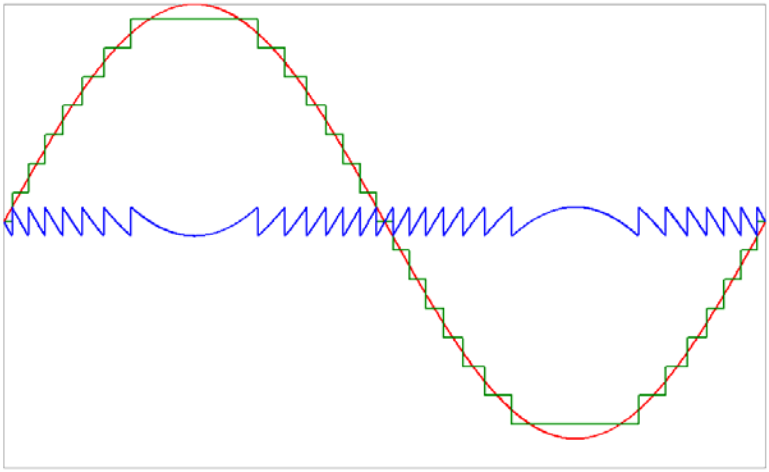


Рис.2 Исходный сигнал показан красным цветом, оцифрованный-зелёным, шумы квантования (разница) – синим [Wiki].

Для минимизации влияния этих шумов на звук в АЦП (амплитудно-цифровой преобразователь) включают специальные фильтры. Таким образом, у звуковой карты есть три параметра, на которые необходимо обращать внимание: разрядность АЦП (влияет на точность оцифровки амплитуды сигнала), частота дискретизации и наличие оптимального фильтра.

Чтобы оценить, как цифровой сигнал отличается от аналогового, необходимо иметь численные показатели, характеризующие его амплитуду. Для аналогового сигнала используют понятие пиковой амплитуды ($A_{пик}$) и среднеквадратичное значение амплитуды (RMS). В случае цифрового сигнала используются те же величины, но только в окне (под окном подразумевается фиксированный диапазон сэмплов сигнала). Например, окно начинается с сэмпла *K* и состоит из *N* сэмплов, т.е. аудио сигнал *y[n]* состоит из *y[K], y[K + 1], . . . , y[K + N − 1]* сэмплов. Тогда пиковая амплитуда для цифрового сигнала – сэмпл с наибольшей амплитудой (абсолютное значение) в окне.

$$A_{пик}\{y[n]\} = \max |y[n]|,\ n = K, \ldots, K + N - 1 \qquad (5)$$

Как видно из рис.3, пиковая амплитуда $A_{пик}$ зависит от ширины окна, т.е. для правильного определения пиковой амплитуды необходимо, чтобы хотя бы один период сигнала попадал в окно.

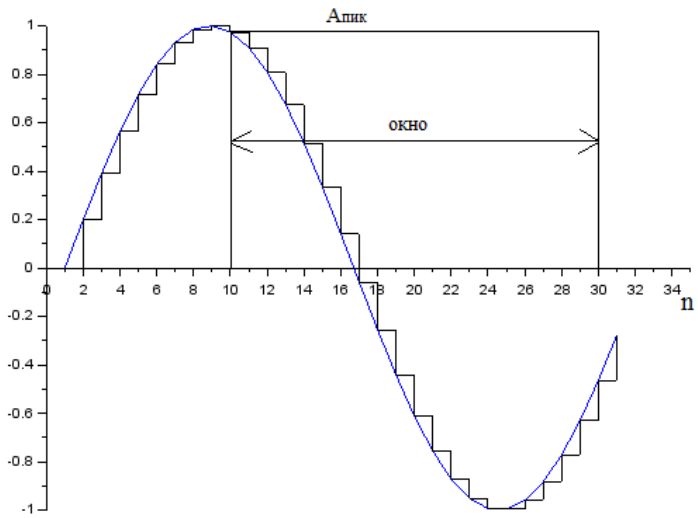


Рис.3 Графики сигнала (синим) и оцифрованного сигнала

В Scilab и Octave такие графики можно получить, используя следующие скрипты:

```
//Scilab
n=10:30;//ширина окна в отсчётах
x=[0:0.2:6];
y=sin(x);//сигнал
plot2d2(y)//график оцифрованного сигнала
plot(y)//график сигнала
Ap=max(y(n))//пиковое значение амплитуды в окне
a=gca();//управление текущим графическим окном
a.box ="off";//отключение рамки вокруг графиков
a.x_location="middle";//ось x перемещена в средину
графика
```

```
# Octave
x=[0:0.2:6];
y=sin(x);
plot(y,"k")# график сигнала чёрным цветом
n=10:30;
Ap=max(y(n)):
hold on# возможность наложить другой график в одном
графическом окне
stairs(y,"r")#график оцифрованного сигнала
box off#отключение рамки вокруг графиков
 set (gca, "xaxislocation", "origin")
 hold off
```

Чтобы команда `set (gca, "xaxislocation", "origin")` сработала (перемещение оси x в точку начала координат (0,0)) необходимо, чтобы набор графических элементов по умолчанию принадлежал gnuplot. Чтобы проверить это, введите `graphics_toolkit` в командной строке Octave.

Если ответ fltk или qt, и ось графика замерзает, то дайте команду:
`graphics_toolkit('gnuplot').`
Сделайте изменение постоянным в файле octave.rc, расположенном в каталоге установки Octave. Нижеследующие действия используются также в ситуации, когда после команды `plot` не происходит отображение графика в графическом окне (обычно это происходит при первоначальной инсталляции Octave).
Шаг 1: Установите gnuplot на C:\Program Files (x86)\gnuplot:
https://sourceforge.net/projects/gnuplot/
Шаг 2: Перейти к приведенному каталогу :
C:\Octave\share\octave\site\m\startup\octaverc
открыть octaverc файл, присутствующий в этом каталоге в Блокноте.
Шаг 3:
Добавьте следующие строки в конец файла.

```
gnuplot_binary 'C:\Program
Files(x86)\gnuplot\bin\gnuplot.exe'
graphics_toolkit('gnuplot')
```

Шаг 4:
Сохраните файл и выйдите из Блокнота. Если Octave открыт, закройте его.
Шаг 5:
Откройте Octave и введите следующую команду для проверки:

```
plot([5,6,7],[8,9,10]);
```

Обратите внимание, что в первый раз может потребоваться gnuplot до 5 минут для отображения графика. Поэтому не надо закрывать приложение, последующее простые графики отображаются быстро.
Среднеквадратичное значение амплитуды:

$$A_{RMS} = \sqrt{P\{y[n]\}}$$

где *P{y[n]}* средняя мощность сигнала, которая определяется как:

$$P\{y[n]\} = \frac{1}{N}(|y[M]|^2 + \cdots + |y[M+N-1]|^2) \qquad (6)$$

Как известно, для непрерывного синусоидального сигнала теоретическое соотношение между пиковым и среднеквадратическим значением амплитуды равно $A_{RMS}=0.707 \cdot A_{пик.}$. В случае оцифрованного сигнала это отношение будет зависеть от частоты дискретизации.

```
#Octave
x=[0:0.2:6];
y=sin(x);
n=1:31;
p=1/30*sum(y(n).^2)
p =  0.52336
a=sqrt(p)
a =  0.72344
```

```
// Scilab
x=[0:0.2:6];
y=sin(x);
 n=1:31;
p=1/30*sum(y(n).^2)
 p  = 0.5233595
a=sqrt(p)
 a  =  0.7234359
```

Предлагается проверить, как от частоты дискретизации зависит точность соотношения между пиковой и среднеквадратичной амплитудой.

Амплитуды двух сигналов часто лучше сравнивать не по разнице, а по отношению. Поэтому широко используется внесистемная единица – децибел. Энергетические величины пропорциональны квадратам *величин поля*, таких как звуковое давление, электрическое напряжение, сила электрического тока и т. п., поэтому отношение двух значений этой величины , выраженное в децибелах (dB), определяется по формуле:

$$d = 20 \cdot \log_{10}(A/A_0) \qquad (7)$$

где A_0 - опорная амплитуда. Это определение настроено так, что, если мы увеличим мощность сигнала в десять раз (так что амплитуда возрастает в разы из √10), логарифм увеличится на 1/2, и поэтому значение в децибелах идет вверх на десять. Увеличение амплитуды в два раза соответствует увеличение примерно на 6,02 децибела; Удвоение мощности - увеличение на 3,01 дБ. График зависимости линейной амплитуды от амплитуды в децибелах на рисунке 4.

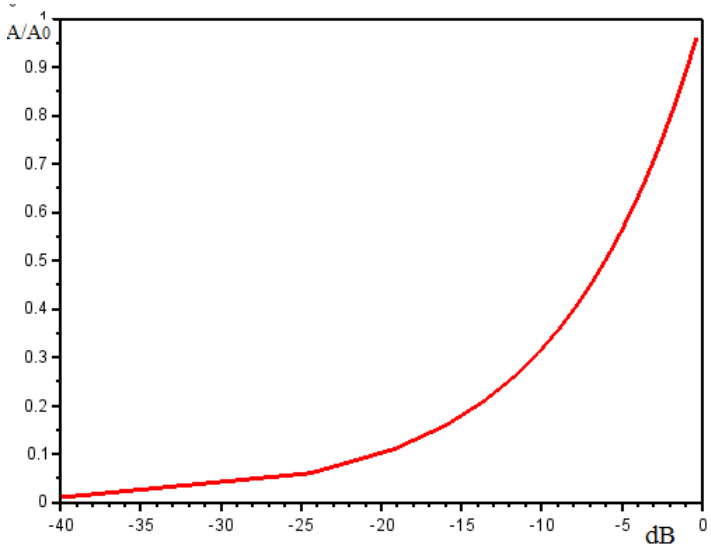


Рис.4 Зависимость линейной амплитуды от амплитуды в децибелах

Сигнал с линейной амплитудой *A* меньше, чем A_0 будет иметь отрицательную амплитуду в децибелах: для *A* меньше чем A_0 в 10 раз будет -20 дБ, в 100 раз -40дБ и так далее. Если линейная амплитуда равна нулю, то в дБ это будет $-\infty$.

В цифровом аудио удобно выбрать амплитуду опорного сигнала (при условии, что аппаратное обеспечение даёт максимальную амплитуду, равную единице), $A_0 = 10^{-5} = 0.00001$. Тогда максимально возможная амплитуда составит 100 дБ, а сигнал с амплитудой 0 дБ, практически, не будет слышен на любом разумном уровне прослушивания. Динамический диапазон человеческого слуха - соотношение между очень громким звуком

и неслышно тихим - около 100 дБ. Поэтому такой выбор опорного сигнала и максимального сигнала достаточно удобен.

Шум квантования, являющийся белым шумом, заметен на слух, при его интенсивности всего 4 дБ. Для бытовых помещений динамический диапазон музыки не должен превышать 101 - 108 дБ.

По закону, открытому немецкими учёными Э. Вебером (1795—1878) и Г. Фехнером (1801—1887), величина ощущений человека, и амплитуда, вызвавшего его раздражения, пропорционально связаны логарифмической формулой. Данный закон справедлив для всех видов ощущений человека: слуха, зрения, обоняния, осязания. Поэтому амплитуда неточным образом связана с воспринимаемой громкостью звука.

Как правило, два сигнала с одинаковым пиком или среднеквадратичной амплитудой не обязательно имеют одинаковую громкость для всех слушателей. Но усиление сигнала, скажем, на 3 дБ, будет довольно надежно отмечено всеми слушателями.

1.2 Частота

Частоты, также как и амплитуды, измеряются в логарифмическом масштабе. Соотношение частот между двумя музыкальными тонами определяет музыкальный интервал между ними.

Музыкальная шкала делит октаву (музыкальный интервал, в котором соотношение частот между звуками составляет один к двум - то есть частота высокого звука в два раза больше низкого) на двенадцать равных интервалов (полутонов), каждый из которых соответствует соотношению $2^{1/12}$.

Отношение частот, соответствующее каждому интервалу, называется его интервальным коэффициентом. При этом должно выполняться следующее правило: чтобы сложить два интервала, надо умножить их интервальные коэффициенты, а чтобы вычесть один из другого, надо разделить их интервальные коэффициенты. Была создана шкала, которая получила название равномерно темперированной шкалы. Темперированной она называется потому, что содержит последовательность тонов с математически строго определенными частотными соотношениями. Разновидности этой шкалы, отличающиеся последовательностью расположения целых тонов и полутонов, называются ладами.

Если обозначить интервальный коэффициент (т. е. отношения частот) полутона как S, то для получения октавы надо умножить их друг на друга двенадцать раз: $S \cdot S \cdot S... = S^{12}$. Поскольку интервальный коэффициент октавы равен 2:1, то $S^{12}=2$ и

$$S = \sqrt[12]{2} \approx 1.0595$$

Таким образом, интервальный коэффициент полутона в темперированной шкале всегда равен 1.0595.

Например, если нота ля (А4) имеет частоту 440 Гц, то нота ля-диез имеет частоту 440 х 1,0595 = 466,18 Гц.[1] Чтобы в равномерно темперированной шкале получить значение частоты для любой ноты, надо, взяв за основу значение частоты какой-либо ноты, например до (С4) = 261,6 Гц, умножить его на S столько раз, на сколько полутонов отличается данная нота от С4. Значения частот для всех нот равномерно темперированной шкалы даны на рис. 5.

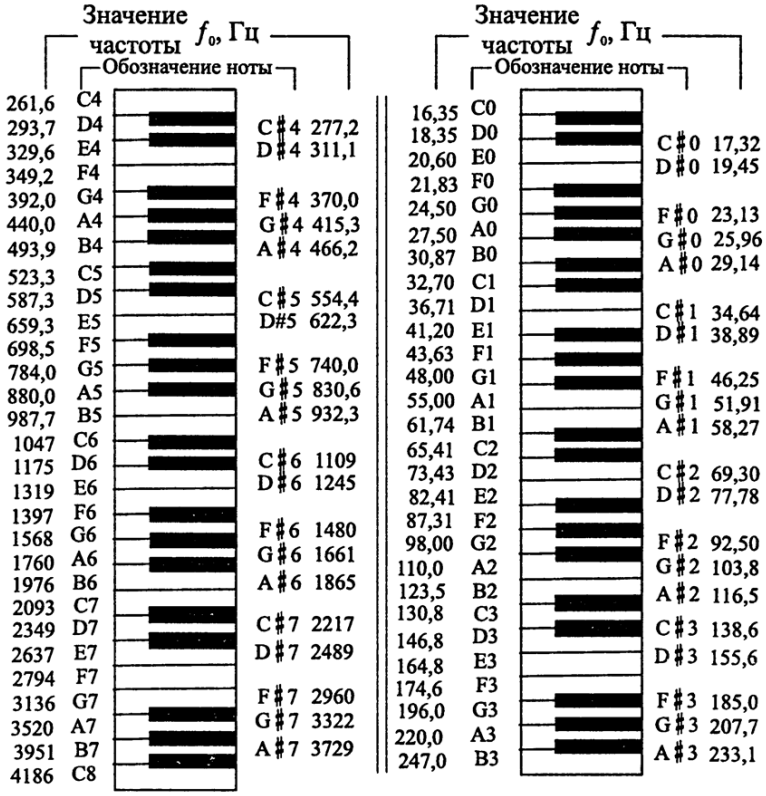


Рис.5 Значения частот для всех нот равномерно темперированной шкалы

Для более мелкого деления интервалов в темперированной шкале используется понятие цент; при этом каждый полутон делится на 100 центов. В этом случае интервальный коэффициент цента С определяется следующим образом: С·С·С· С...= C^{100} и С= $\sqrt[100]{1.0595}$ = 1.000578.

Определим, сколько центов содержится в интервале с заданным интервальным коэффициентом *r = fв(Гц)/fн(Гц)*. Умножим интервальные коэффициенты всех центов, из которых он состоит, т. е. С·С·С... = C^n, и приравняем их к заданному интервальному коэффициенту r = C^n. Взяв десятичный логарифм от обеих частей нашего уравнения:

$$lg\ (r) = n \cdot lg(C), \qquad (8)$$

получим число центов в любом интервале:

$$n = lg(r)/lg(C) = 3985\ lg(r). \qquad (9)$$

Например, у чистой квинты r = 3:2, отсюда число центов равно n = 702; у чистой кварты r = 4:3 и число центов равно 498; у октавы r = 2/1 и число центов равно 1200 и т. д.

[1] В 1939 году международная конференция в Лондоне приняла стандарт частоты для тона А4 равным 440 Гц.

Хорошая идея – пересчитать количество центов с помощью программ: Scilab, Octave.

Таблица 1

Интервальные коэффициенты и количество центов для основных музыкальных интервалов в трех музыкальных шкалах

Интервал	Равномерно темперированная шкала		Чистая шкала		Шкала Пифагора	
	Отношения частот	Центы	Отношения частот	Центы	Отношения частот	Центы
Октава	2,000	1200	2/1 = 2,000	1200	2,000	1200
Квинта	1,498	700	3/2 = 1,500	702	1,500	702
Кварта	1,335	500	4/3 = 1,333	498	1,333	498
Б. терция	1,260	400	5/4 =1,250	386	1,265	408
М. терция	1,189	300	6/5 = 1,200	316	1,184	294
Б. секста	1,682	900	5/3 = 1,667	884	1,687	906
М. секста	1,587	800	8/5 = 1,600	814	1,580	792

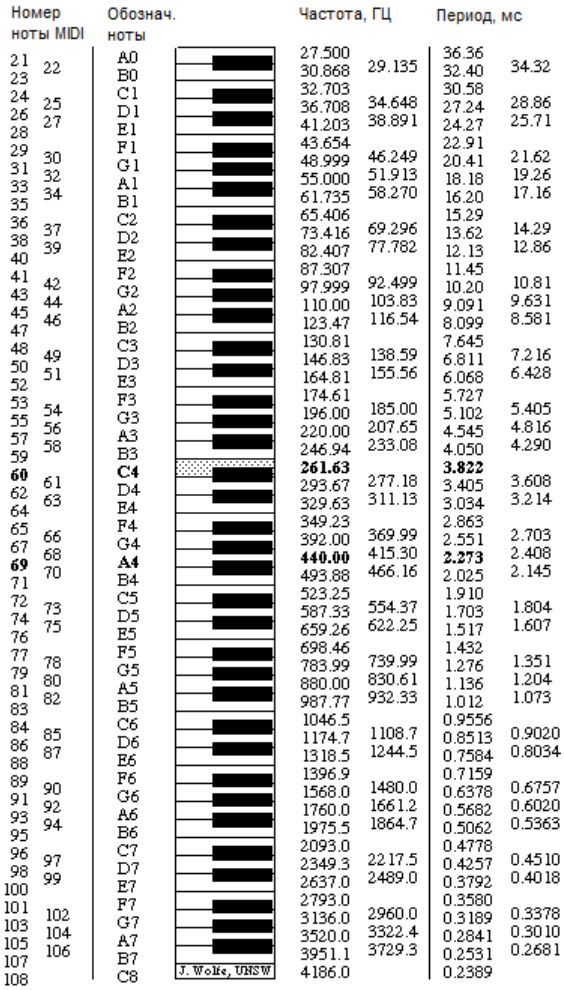


Рис.6 Соответствие между номером в MIDI и нотами

Давайте на примере рассмотрим алгоритм преобразования частоты в высоту звука (т. е. в ближайшую ноту) и узнаем, насколько она не настроена. Просто объединим то, что было изложено выше.

Как сделать расчет? Предположим, что две ноты имеют частоты f_1 и f_2 и отношение частот f_2/f_1. Октава имеет отношение 2: 1, поэтому число октав между f_2 и f_1 равно

$$n_0 = log2\ (f_2/f_1) \qquad (10)$$

Теперь разделим октаву на более мелкие единицы. Как мы уже знаем, при равномерно темперированной шкале все полутоны имеют одинаковое соотношение частот $2^{1/12}$. Для начала, нужна известная нота и её частота. Обычно это A4, частота которой принимается равной 440 Гц. Для ноты, которая лежит на n полутонов выше (или на –n полутонов ниже), частота равна:

$$f_n = 2^{n/12} \cdot 440\ [Гц] \qquad (11)$$

Число полутонов n от A4 можно получить из выражения:

$$n = 12 \cdot log2\ (f_n/440) \qquad (12)$$

Аналогичные уравнения дают: n_0- число октав от *A4*, а n_c- количество центов от *A4*:

$$n_o = log2\ (f_n/440) \quad \text{и} \quad n_c = 1200 \cdot log2\ (f_n/440) \qquad (13)$$

Для подсчёта центов в выражении (8) мы использовали десятичный логарифм, а в (13) по основанию 2. Здесь нет никакой разницы, вычисленные значения одинаковы. (*Проверьте*).

Для взаимодействия исполнителя (в конечном итоге) с электронными инструментами используются различные протоколы, например: OSC[2], MIDI[3]. Например, в MIDI полутон часто задается номером, обозначим его для наших целей *m*. Номер *m* для ноты *A4* равен 69 и увеличивается на единицу для каждого полутона, так что это дает нам простое преобразование между частотами и номерами MIDI (снова используя 440 Гц в качестве высоты A4):

$$m = 12 \cdot log2\ (f_m/440)+69 \text{ и } f_m = 2^{(m-69)/12} \cdot 440\ [Гц] \qquad (14)$$

---

[2] Open Sound Control — это новый, оптимизированный для современных сетевых технологий протокол для взаимодействия компьютеров, звуковых синтезаторов и других мультимедиа устройств» — так был представлен OSC на международной конференции по компьютерной музыке в 1997 году. OSC не является протоколом в том виде, каким является MIDI, так как он не описывает требований к аппаратному обеспечиванию — спецификации описывают лишь формат передачи данных. В этом плане OSC больше схож с XML или JSON, нежели с MIDI.

[3] MIDI (Musical Instrument Digital Interface) — это стандарт обмена данными между цифровыми музыкальными инструментам. Он позволяет обмениваться такой информацией, как номер ноты, скорость нажатия, таймкод и др. MIDI поддерживает большинство выпускаемых электронных музыкальных инструментов.

MIDI -этот стандарт, включает в себя требования к аппаратным средствам, например кабелям и разъемам, договоренности о способах кодирования данных. Является старым аппаратным протоколом и не отвечает современным требованиям, например, он допускает использовать только целое число шагов между 0 и 127. Несмотря на все недостатки, MIDI до сих пор успешно используется. Сфера применения протокола не ограничивается управлением только синтезаторами, MIDI используют для управления процессорами эффектов, микшерными пультами, осветительными, пиротехнические приборами и дымовыми машинами. Очевидно, что в персональных компьютерах и мультимедиа также используется данный протокол. Но несмотря на широчайшее использование, его постепенно вытесняет Open Sound Control — это новый, оптимизированный для современных сетевых технологий протокол для взаимодействия компьютеров, звуковых синтезаторов и других мультимедиа устройств.

1.3 Синтез сигналов с помощью Scilab и Octave

Основная идея, используемая для синтеза аудио сигналов, заключается в том, чтобы представить этот процесс, как манипуляции с набором управляемых генераторов. Эти генераторы, в свою очередь, способны выдавать звуки с заданными характеристиками по командам исполнителя-музыканта. Наиболее часто используются два метода синтеза звука: FM (Frequency modulation – частотная модуляция) и WT (Wave Table – таблично-волновой). При FM-синтезе любой сигнал является суммой простейших синусоид, и, наложив друг на друга сигналы от конечного числа генераторов синусоид, путем математических операций с их частотами и амплитудами получим звуки, почти идентичные, полученным физическими методами. При таблично-волновом WT-синтезе проводят преобразования заранее записанных (оцифрованных) образцов звуков реальных музыкальных инструментов. Эти образцы (сэмплы) хранятся в виде таблицы (sample table), из которой выбираются нужные звуки.

Практически у каждого понятие синтезатор ассоциируется с электронной музыкой и наоборот. Дадим определение, что такое синтезатор. Музыкальный синтезатор (секвенсор, англ. sequence) – это устройство, работающее с последовательностью команд или описаний. Таким образом, назначение синтезатора - считывание от управляющего устройства или программы последовательности команд, и выдача оцифрованного звука. Синтезаторы часто изготавливаются в виде отдельных электронных устройств с собственной клавиатурой и интерфейсами вывода звука. Обычный персональный компьютер также используется в роли синтезатора, причём двумя способами: аппаратным и программным.

В первом случае, аппаратный синтезатор является частью звуковой карты. И все действия, включая вывод звука в виде цифровой последовательности или WAVE-файла, выполняет собственный микропроцессор звуковой карты.

Во втором случае, программный синтезатор – это программа, эмулирующая работу аппаратного синтезатора и загружающая центральный процессор компьютера.
Команды или данные, передаваемые любому синтезатору, как уже было отмечено выше, должны соответствовать определённым протоколам: MIDI, OSC. Например, подключенные к компьютеру устройства управления, такие как: внешняя MIDI-клавиатура, или программа, например Sound Forge, отправляют синтезатору команды MIDI. MIDI-последовательность - это последовательность команд, которые определяют, какую ноту взять, её продолжительность, тональность звучания, инструмент и т. д. Последовательность таких команд записывается в виде файла с расширением (MID). Один и тот же MIDI-файл может звучать на разных синтезаторах по-разному, все зависит от умения исполнителя и качества инструмента, на котором он играет.
А, теперь давайте попробуем программы Scilab и Octave в качестве простейших синтезаторов. Для начала синтезируем синусоидальные сигналы с изменяющейся амплитудой.
В Scilab и Octave это выглядит следующим образом:

```
f=440;// частота сигнала
fs=44100;// частота дискретизации
t=soundsec(2,fs);// вектор сэмплов
x=2*%pi*f*t;
a0=1.e-5;//амплитуда опорного сигнала
n=1;//начальное значение множителя
while n<20 //начало цикла
   a=a0*10^(0.25.*n);//значение амплитуды сигнала
   y=a*sin(x);//сигнал
   sound(y)//звучание сигнала
   tic;sleep(2000);toc //задержка выполнения команд на 2 сек.
   n=n+1;//изменение значения множителя
end //конец цикла
```

```
                # Octave
f=440;# частота сигнала
fs=44100;# частота дискретизации
nn=0:88200;
t=nn/fs;# вектор сэмплов
x=2*pi*f*t;
a0=1.e-5;#амплитуда опорного сигнала
n=1;#начальное значение множителя
while n<=20 #начало цикла
   a=a0*10^(0.25.*n);#значение амплитуды сигнала
   y=a*sin(x);#сигнал
   player=audioplayer(y,fs,8);
   play(player)#звучание сигнала
   tic;pause(2);toc #задержка выполнения команд на 2 сек.
   n=n+1;#изменение значения множителя
endwhile #конец цикла
```

В данном скрипте через каждые 2 секунды происходит увеличение амплитуды сигнала на 5дБ в диапазоне от 5 до 100 дБ и его воспроизведение. Чтобы услышать сигналы, поставьте громкость на максимум (тем не менее, сигналы с маленькой амплитудой не будут слышны).

Задержка в выполнении команд на 2 секунды с помощью команд: `tic;sleep(2000);toc` позволяет услышать каждый сигнал с новой амплитудой. В Octave для задержки выполнения процесса используются команды: `tic;pause(2);toc,` где время задаётся в секундах, в отличие от Scilab (там в миллисекундах).

Следующая задача, которую решим – это изменение частоты сигнала, заодно и проверим полосу воспроизведения вашей акустической системы. Поскольку у меня обычные компьютерные колонки, то можно ожидать начало звучания при частоте 140-200 Гц, а конец при частоте около 18 кГц. Давайте проверим это. Напишем вот такой скрипт, немножко изменив предыдущий:

```
//Scilab
f0=20;//начальная частота в Гц
fs=44100;
t=soundsec(1,fs);// длительность 1 сек
x=2*%pi*f*t;
n=1;
while n<=20
  x=2*%pi*f0*n*t;
   y=sin(x);
   sound(y)
   tic;sleep(1000);toc
   s=f0*n;
   disp('Hz=',s)// значение частоты
   n=n+1;
end
```

При выполнении этого скрипта, в командном окне будет выводиться частота, и звучать тон этой частоты. При значении начальной частоты равной 1000 Гц, определим конец частотного диапазона нашей акустики. То же самое проделаем со скриптом для Octave:

```
#Octave
f0=20;# частота сигнала Гц
fs=44100;# частота дискретизации, Гц
nn=0:88200;
t=nn/fs;# вектор сэмплов
n=1;#начальное значение множителя
while n<=20 #начало цикла
   x=2*pi*f0*n*t;
   y=sin(x);#сигнал
   player=audioplayer(y,fs,8);
   play(player)#звучание сигнала
   tic;pause(1);toc; #задержка выполнения команд на 1 сек.
   s=f0*n;#частота, Гц
   disp('Hz='),disp(s)
   n=n+1;#изменение значения множителя
endwhile #конец цикла
```

А теперь сыграем гамму. Пусть наши частоты лежат в области первой октавы. В Scilab скрипте зададим эти частоты в виде вектора-строки f. И послушаем, что у нас получилось.

```
                    // Scilab
fs=44100;
t=soundsec(1,fs);
f=[261.6,293.7,329.6,349.2,392,440,493.9,523.3];// частоты 1 октавы
n=1;
while n<=8
  x=2*%pi*f(n).*t;
   y=sin(x);
   sound(y)
   tic;sleep(1000);toc
   s=f(n);
   disp('Hz=',s)
   n=n+1;
end
```

Аналогичный скрипт напишем в Octave. Только частоты найдём с помощью формул 10-13.

Частота ноты "до" первой октавы 261.63 Гц. Остальные частоты вычислим, зная частоту ноты "до" и количество полутонов между нотами.

```
                #Octave
fs=44100;# частота дискретизации
nn=0:88200;
t=nn/fs;# вектор сэмплов
k=[0,2,4,5,7,9,11];# разница количества полутонов между нотами
f= 2.^(k./12)*261.63;# частота ноты "до" 1 октава
n=1;#начальное значение множителя
while n<=7 #начало цикла
   x=2*pi*f(n).*t;
   y=sin(x);#сигнал
   player=audioplayer(y,fs,8);
   play(player)#звучание сигнала
   tic;pause(1);toc; #задержка выполнения команд на 1 сек.
   s=f(n);
   disp('Hz='),disp(s)
   n=n+1;#изменение значения множителя
endwhile #конец цикла
```

А теперь послушаем, как звучит сложный сигнал вида:

$$f(x)=1/3\cdot(\sin(x) + 2\cdot\sin(3\cdot x-1.5) - \sin(x+0.5))$$

Множитель 1/3 взят, чтобы амплитуда сигнала не превышала 1. Если будет больше, то неизбежны искажения звука при воспроизведении. Впрочем, попробуйте без этого множителя. Слегка модифицируем предыдущий скрипт.

```
                #Octave
fs=44100;# частота дискретизации
nn=0:88200;
t=nn/fs;# вектор сэмплов
k=[0,2,4,5,7,9,11];# разница количества полутонов между нотами
f= 2.^(k./12)*261.63;# частота сигнала
n=1;#начальное значение множителя
   x=2*pi*f(n).*t;
   y=1/3*(sin(x)+2*sin(3*x-1.5)-sin(x+0.5));#сигнал
   player=audioplayer(y,fs,8);
   play(player)#звучание сигнала
   tic;pause(1);toc; #задержка выполнения команд на 1 сек.
   s=f(n);
   disp('Hz='),disp(s)
 plot(x(1:500),y(1:500),'k');# график функции y в первых 500x точках
```

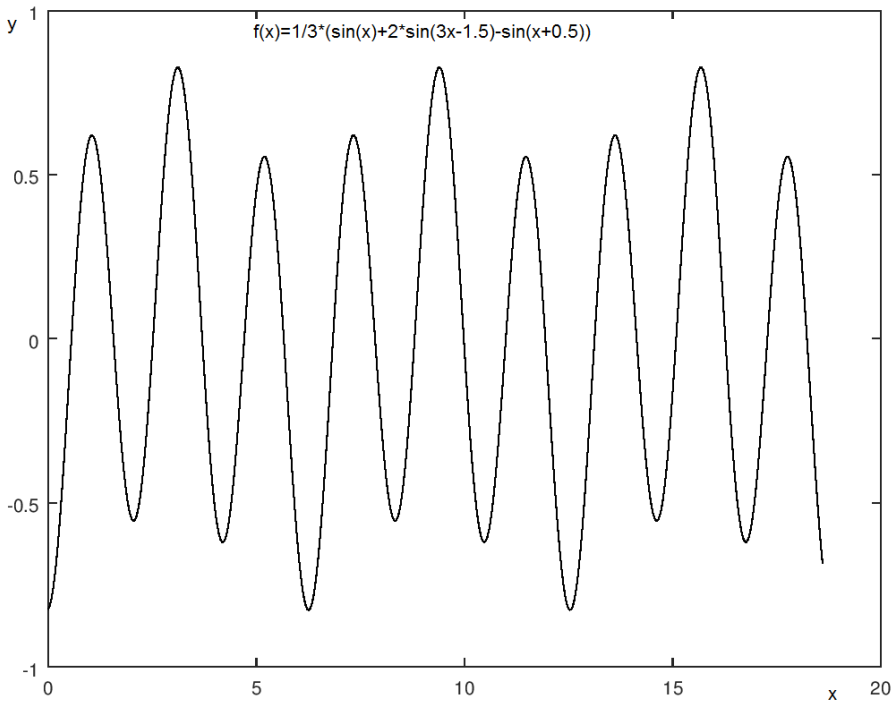


Рис.7 График функции (в первых пятистах точках)

Для упражнения получите с помощью Scilab и Octave таблицу соответствия нот частотам, взятую из Википедии и приведённую ниже.

Таблица соответствия нот частотам

Частоты в герцах (интервал от До первой октавы в полутонах)											
Октава › Нота ↓	Суб-контр	Контр	Большая	Малая	1	2	3	4	5	6	7
C	16,352 (−48)	32,703 (−36)	65,406 (−24)	130,81 (−12)	261,63 (±0)	523,25 (+12)	1046,5 (+24)	2093,0 (+36)	4186,0 (+48)	8372,0 (+60)	16744.0 (+72)
C♯ / D♭	17,324 (−47)	34,648 (−35)	69,296 (−23)	138,59 (−11)	277,18 (+1)	554,37 (+13)	1108,7 (+25)	2217,5 (+37)	4434,9 (+49)	8869,8 (+61)	17739.7 (+73)
D	18,354 (−46)	36,708 (−34)	73,416 (−22)	146,83 (−10)	293,66 (+2)	587,33 (+14)	1174,7 (+26)	2349,3 (+38)	4698,6 (+50)	9397,3 (+62)	18794.5 (+74)
D♯ / E♭	19,445 (−45)	38,891 (−33)	77,782 (−21)	155,56 (−9)	311,13 (+3)	622,25 (+15)	1244,5 (+27)	2489,0 (+39)	4978,0 (+51)	9956,1 (+63)	19912.1 (+75)
E	20,602 (−44)	41,203 (−32)	82,407 (−20)	164,81 (−8)	329,63 (+4)	659,26 (+16)	1318,5 (+28)	2637,0 (+40)	5274,0 (+52)	10548 (+64)	21096.2 (+76)
F	21,827 (−43)	43,654 (−31)	87,307 (−19)	174,61 (−7)	349,23 (+5)	698,46 (+17)	1396,9 (+29)	2793,8 (+41)	5587,7 (+53)	11175 (+65)	22350.6 (+77)
F♯ / G♭	23,125 (−42)	46,249 (−30)	92,499 (−18)	185,00 (−6)	369,99 (+6)	739,99 (+18)	1480,0 (+30)	2960,0 (+42)	5919,9 (+54)	11840 (+66)	23679.6 (+78)
G	24,500 (−41)	48,999 (−29)	97,999 (−17)	196,00 (−5)	392,00 (+7)	783,99 (+19)	1568,0 (+31)	3136,0 (+43)	6271,9 (+55)	12544 (+67)	25087.7 (+79)
G♯ / A♭	25,957 (−40)	51,913 (−28)	103,83 (−16)	207,65 (−4)	415,30 (+8)	830,61 (+20)	1661,2 (+32)	3322,4 (+44)	6644,9 (+56)	13290 (+68)	26579.5 (+80)
A	27,500 (−39)	55,000 (−27)	110,00 (−15)	220,00 (−3)	440,00 (+9)	880,00 (+21)	1760,0 (+33)	3520,0 (+45)	7040,0 (+57)	14080 (+69)	28160.0 (+81)
A♯ / H♭	29,135 (−38)	58,270 (−26)	116,54 (−14)	233,08 (−2)	466,16 (+10)	932,33 (+22)	1864,7 (+34)	3729,3 (+46)	7458,6 (+58)	14917 (+70)	29834.5 (+82)
H	30,868 (−37)	61,735 (−25)	123,47 (−13)	246,94 (−1)	493,88 (+11)	987,77 (+23)	1975,5 (+35)	3951,1 (+47)	7902,1 (+59)	15804 (+71)	31608.5 (+83)