Лабораторная работа 1

Моделирование дискретных сигналов в Matlab и Simulink

Цель работы: освоение приемов моделирования дискретных сигналов в средах Matlab и Simulink

Рабочее задание

1. **Смоделировать в среде Matlab сигнал $s_2(t)$ на выходе дискретизатора** (частота дискретизации F_s), если на вход дискретизатора подается сигнал:

$$s_1(t) = A_1 \cos(2\pi f_{01} t + \varphi_{01}) + A_2 \cos(2\pi f_{02} t + \varphi_{02}), \quad 0 \le t \le T.$$

Варианты значений параметров сигнала $s_1(t)$ приведены в табл.1.

Парам. \ Вар.	1	2	3	4	5	6	7	8
A_1	1	1	2	1	1	1	1	2
A_2	1	2	1	1	1	2	2	1
f_{01}, Гц	100	100	100	100	100	100	100	100
f_{02}, Гц	200	200	200	200	200	200	200	200
φ_{01}, рад	0	0	0	0	π	0	π	0
φ_{02}, рад	0	0	0	π	0	π	0	π

Частоту дискретизации F_s задать двумя способами:

а) исходя из инженерной версии теоремы Найквиста-Котельникова;
б) увеличив выбранное по п.а) значение в 5 раз.

Длительность T выбрать так, чтобы на ней уложилось два периода.

В отчете представить:

- перечень команд Matlab, с помощью которых происходит вычисление сигнала $s_2(t)$;
- график сигнала $s_2(t)$ и составляющих его гармонических компонентов.

2. Смоделировать сигнал $s_2(t)$ в среде Simulink.

В отчете представить:

- блок-схему моделирования;
- график сигнала $s_2(t)$ и составляющих его гармонических компонентов.

3. Сделать общие выводы по работе, сравнив между собой моделирование в среде Matlab и моделирование в среде Simulink

Контрольные вопросы:

1. Что такое «инженерная версия теоремы Найквиста-Котельникова»?
2. Что собой представляет способ дискретизации, именуемый «выборка-хранение»?
3. Как реализовать способ дискретизации, именуемый «выборка-хранение», в среде Matlab и в среде Simulink?

Приложение

Теоретические сведения. Дискретные сигналы в Matlab и Simulink

Вводные замечания

Подавая электрический сигнал с выхода микрофона на вход звуковой платы компьютера, полезно представлять себе, как аналоговый сигнал преобразуется в дискретный, и как затем дискретный сигнал преобразуется в последовательность чисел. В данном разделе мы рассмотрим первый этап – преобразование аналогового сигнала в дискретный. Такое преобразование принято называть «дискретизацией».

Возможные варианты сигналов показаны на рис. 0. Сигнал, изображенный на рис. 0.а, будем называть *исходным аналоговым.* На рис. 0.б представлена дискретная версия исходного сигнала, обычно именуемая *данными, оцифрованными естественным способом,* или *данными с амплитудно-импульсной модуляцией* (pulse amplitude modulation — PAM). Данные на рис.0.б еще несовместимы с цифровой системой, поскольку амплитуда каждой естественной выборки все еще может принимать бесконечное множество возможных значений, а цифровая система работает с конечным набором значений. На рис. 0.в и рис.0.г показано представление исходного сигнала такими дискретными импульсами, вершина которых плоская. Если значения импульсов образуют несчетное множество, они называются *дискретными отсчетами.* Далее эти импульсы можно подать на устройство квантования, преобразующее импульсы так, что их значения образуют счетное множество - такие импульсы называются *квантованными отсчетами.* Данные в таком формате совместимы с цифровой системой.

Импульсы рис.0.г отличаются от импульсов рис.0.в тем, что полностью заполняют промежуток между моментами обновления значения сигнала. Такой способ дискретизации, именуемый «выборка-хранение» [3], наиболее эффективен с точки зрения помехоустойчивости.

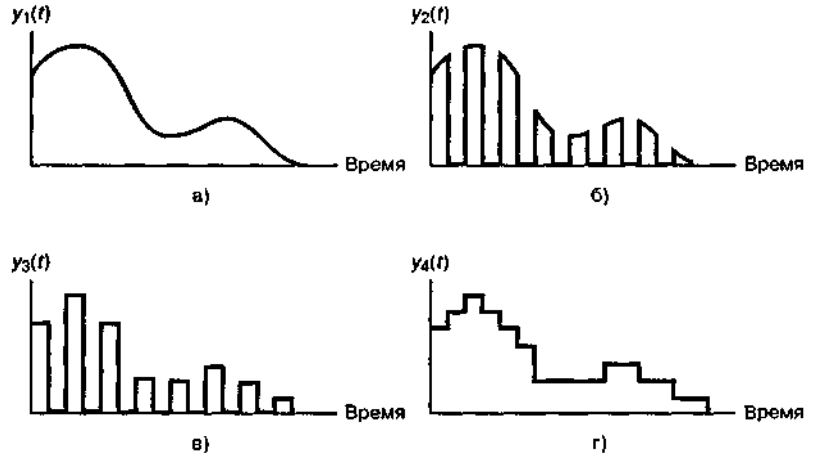


Рис. 0. Исходные данные в системе координат "время-амплитуда ":
а) исходный аналоговый сигнал; 6) данные в естественной дискретизации;
в) квантованные выборки; г) выборка-хранение

Моделирование дискретных сигналов в Matlab

Генерировать сигналы в **Matlab** можно тремя способами:

- в диалоговом режиме, с помощью последовательности команд в командном окне;
- в автоматическом режиме, путем создания и запуска на выполнение m-скрипта;
- в автоматическом режиме, путем создания и вызова m-функции.

Генерирование сигналов в диалоговом режиме. Этот способ наиболее трудоемок, поскольку требует каждую команду набирать с клавиатуры в командном окне. Чтобы

повысить производительность труда, можно всю последовательность команд предварительно набрать в любом текстовом редакторе (обычно это **Notebook** или **Word**), а затем, скопировав текст в буферную память (**Clipboard**), вставить его в командное окно. Недостаток этого способа в том, что необходимо одновременно держать активными две программы – **Matlab** и текстовый редактор. Достоинство данного способа проявляется тогда, когда работу в **Matlab** производят, следуя некоей инструкции, в которой теоретические сведения чередуются с практическими заданиями в виде фрагментов текстов m-скриптов. Такой стиль работы типичен, например, при проведении лабораторных работ [1].

Например, так выглядит в текстовом редакторе последовательность команд генерирования **N** отсчетов тонального сигнала амплитудой **A**, частотой **f0**, начальной фазой **Fi0**, с частотой дискретизации **fs**:

```
% гармонический сигнал
A=1; f0=100; Fi0=pi/2; fs=1000; N=20;  % параметры сигнала
t=(0:N-1)/fs;                          % моменты времени
s=A*sin(2*pi*f0*t+Fi0);        % вычисление отсчетов сигнала
plot(t,s)                              % вывод графика
title('Гармонический сигнал')     % заголовок
xlabel('ВремЯ, с'); ylabel('Уровень');   % надписи вдоль осей
grid on                                 % координатнаЯ сетка
```

Полученный график отображается в специальном окне с надписью Figure #1 (если это первый строящийся график). График удобно сохранять путем экспорта в экономном формате *.jpg (рис.1).

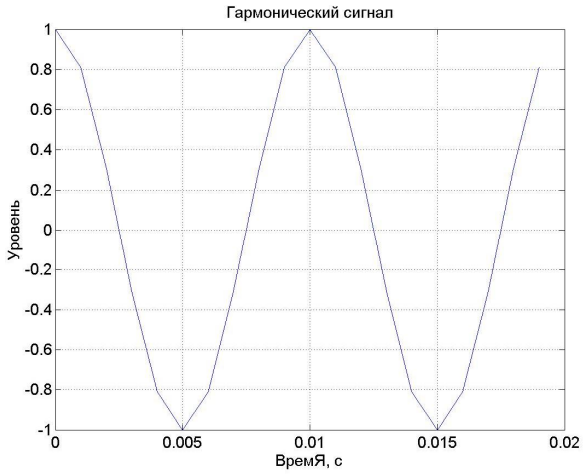


Рис.1

Примечание: при использовании символов кириллицы в тексте команд (комментарии, заголовки и т.п.) следует учитывать особенности «отношения» каждой конкретной версии Matlab к кириллице. Так, в Matlab версии 6.1 нельзя употреблять строчную букву «я» - вместо нее следует писать прописную букву «Я». Именно по этой причине в тексте на рис.1 вместо «Время» получилось «ВремЯ». Впрочем, эту надпись можно отредактировать (кнопка со стрелкой Edit Plot в графическом окне) перед тем, как сохранять рисунок на диске.

Генерирование сигналов путем создания m-скрипта. Данный способ отличается тем, что все команды набираются в специальном окне редактора m-файлов (рис.2).

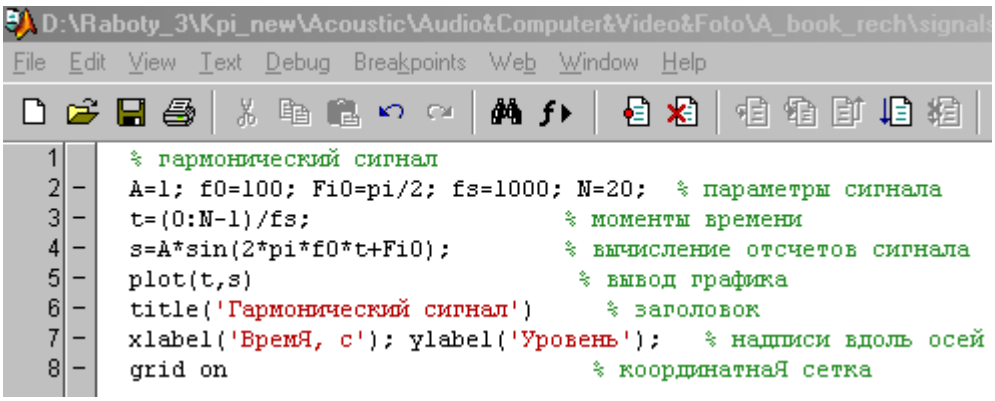


Рис.2

Данный способ хорош тем, что вместо сторонних программных продуктов используется собственный инструментарий Matlab, специализированный для написания и отладки m-скриптов.

Генерирование сигналов путем создания m-функции. Данный способ отличается тем, что входные данные записывают как аргумент некоей функции $y = f(x)$, а выходные – как значение этой функции. Удобство в том, что символьные обозначения данных могут отличаться от обозначений, используемых в теле функции. Более того, числовые значения входных данных можно просто задавать в наименовании вызываемой функции. Последнее обстоятельство продемонстрируем на примере.

Создадим подпрограмму - m-скрипт **ton.m** вида:

```
% скрипт ton
s=A*sin(2*pi*f0*t+Fi0);        % вычисление отсчетов сигнала
```

Команду выполнения этого скрипта нужно «окружить» командами подготовки входных данных и вывода выходных данных:

```
A=1; f0=100; Fi0=pi/2; fs=1000; N=20;  % параметры сигнала
t=(0:N-1)/fs;                % моменты времени
ton;                         % вычисление отсчетов сигнала
plot(t,s)                    % вывод графика
title('Гармонический сигнал')     % заголовок
xlabel('ВремЯ, с'); ylabel('Уровень');   % надписи вдоль осей
grid on                      % координатнаЯ сетка
```

Очевидно, обозначения входных и выходных данных вызывающей программы должны совпадать с обозначениями соответствующих данных вызываемой подпрограммы.

Теперь поступим по-иному – напишем и сохраним m-функцию под именем **ton_sig.m**:

```
%-------функциЯ ton_sig.m -----------
% [s,t]=ton_sig(B,f1,Fi1,Fs,N1)
%---------------------------------------
% генерирование гармонического сигнала
% y = B * sin(2*pi*f1*x + Fi1),
% B - амплитуда;
% N1 - количество отсчетов сигнала;
% f1 - частота;
% Fs - частота дискретизации;
% Fi1 - начальнаЯ фаза сигнала
%---------------------------------------
function [y,x] = ton_sig( B, f1, Fi1, Fs, N1 )
%---------------------------------------
x = (0:N1-1)/Fs;    % моменты времени
```

```
y = B * sin( 2*pi*f1*x + Fi1 );
%-------- конец функции ton_sig.m ----------
```

Теперь m-скрипт генерирования того же отрезка косинусоиды будет выглядеть так:

```
% гармонический сигнал
[s,t]=ton_sig(1,100,pi/2,1000,20)  % вычисление отсчетов сигнала
plot(t,s)                  % вывод графика
title('Гармонический сигнал')     % заголовок
xlabel('ВремЯ, с'); ylabel('Уровень');   % надписи вдоль осей
grid on                    % координатнаЯ сетка
```

Как видим, теперь числовые значения входных данных задаются как аргументы m-функции **ton_sig.m**. Выходные данные функции используются для построения графика.

Очевидно, применение m-функций выгодно тогда, когда алгоритм формирования значений функции достаточно сложный: содержится много команд и обращений к разнообразным библиотечным функциям с непростым синтаксисом.

Очевиден и недостаток m-функций – необходимо помнить их синтаксис. Впрочем, получить нужную информацию можно, если в командном окне задать команду **help**:

```
>> help ton_sig
```

В результате на мониторе отобразится комментарий, с которого начинается m-функция. Для приведенного выше примера текст помощи имеет следующий вид:

```
%-------функциЯ ton_sig.m -----------
% [s,t]=ton_sig(B,f1,Fi1,Fs,N1)
%---------------------------------------
% генерирование гармонического сигнала
% y = B * sin(2*pi*f1*x + Fi1),
% B - амплитуда;
% N1 - количество отсчетов сигнала;
% f1 - частота;
% Fs - частота дискретизации;
% Fi1 - начальнаЯ фаза сигнала
%---------------------------------------
```

Таким образом, очевиден вывод: **очень важно при программировании m-функций снабжать их качественным и подробным комментарием.**

Моделирование дискретных сигналов в Simulink

Генерирование сигналов в **Simulink**, естественно, имеет свои особенности. Рассмотрим их.

Возьмем из библиотеки блоков **Simulink** два блока: **Sine Wave** (из раздела **Sources**) и **Scope** (из раздела **Sinks**). Соединив их, получим немудреную схему (рис.3).

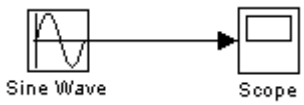


Рис.3

Затем двойным щелчком по блоку осциллоскопа активизируем окно, имитирующее экран осциллоскопа, и запустим модель (кнопка **Start simulation**). В результате получим изображение отрезка синусоиды (рис.4).

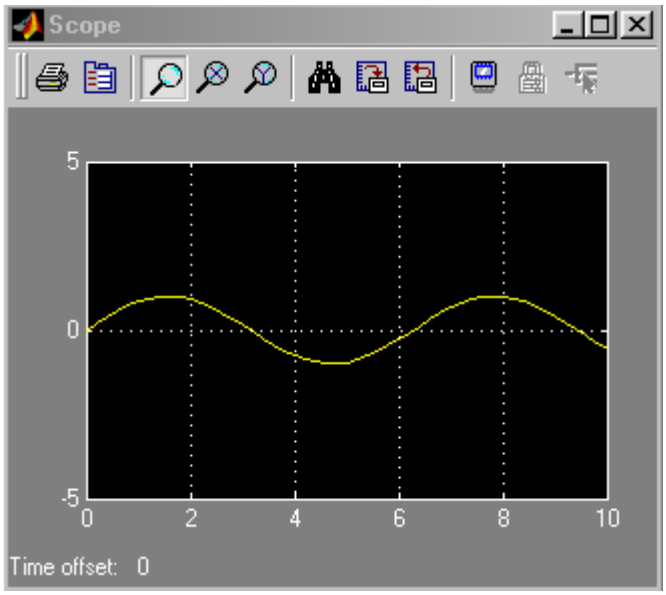


Рис.4

Как видим, генерировать гармонический сигнал в среде Simulink даже проще, чем в среде Matlab. Однако это первое впечатление весьма обманчиво. Действительно, ведь важно еще уметь управлять параметрами гармонического сигнала. То, что амплитуда гармонического сигнала оказалась равной единице – нам просто «повезло». Действительно, по умолчанию амплитуда генерируемого сигнала принята равной единице. Однако частотой, начальной фазой и длительностью сигнала мы пока не управляем.

Дважды щелкнем по блоку **Sine Wave** – в результате появится окно настроек параметров (рис.5). Щелкнув по кнопке **Help**, получим инструкцию по данному блоку, сущность которой сводится вкратце к тому, что в данном блоке выполняется операция

$$y = Amplitude \times \sin(frequency \times time + phase) + bias$$

Из приведенной формулы и надписей на рис.5 становится понятным смысл четырех переменных: амплитуды, угловой частоты, начальной фазы и постоянной составляющей. Остается пока зашифрованным смысл переменной “время”.

Останавливаясь на этом важном вопросе, отметим различие понятий “время” и “модельное время”. Так, генерирование отрезка сигнала длительностью 1 с (модельное время) может длиться значительно более короткий промежуток времени, например, 0.1 с (реальное время). Скорость генерирования зависит от объема вычислений, быстродействия компьютера, от выбранного “решателя”, т.е. алгоритма моделирования, и т.д. Кстати, вполне возможен обратный эффект - для сложного алгоритма процедура моделирования отрезка сигнала длительностью 0.1 с может растянуться на несколько секунд.

Сигнал может генерироваться двух типов: непрерывный **time-based** и дискретный **sample-based**. Для моделирования работы непрерывных систем рекомендуют использовать непрерывный тип **time-based**, а для моделирования работы дискретных систем – дискретный тип **sample-based** [2].

Если установлен тип **time-based**, тогда параметр **Sample time** может принимать значения:

- 0 (по умолчанию) – блок работает в непрерывном режиме;
- >0 - блок работает в дискретном режиме;
- -1 – блок наследует тот же режим, что и принимающий блок.

Как указывается в **Help**, работа в непрерывном режиме может приводить к большим погрешностям генерации на больших промежутках модельного времени.

Работа в дискретном режиме заставляет блок вести себя так, как если бы к выходу непрерывного генератора был присоединен блок **Zero-Order Hold**. Действительно, собрав две схемы (рис.6) и задав в обоих случаях значение параметра **Sample time**, равное 0.5 (окно настройки блока **Zero-Order Hold** показано на рис.7), получаем идентичные результаты (рис.8).

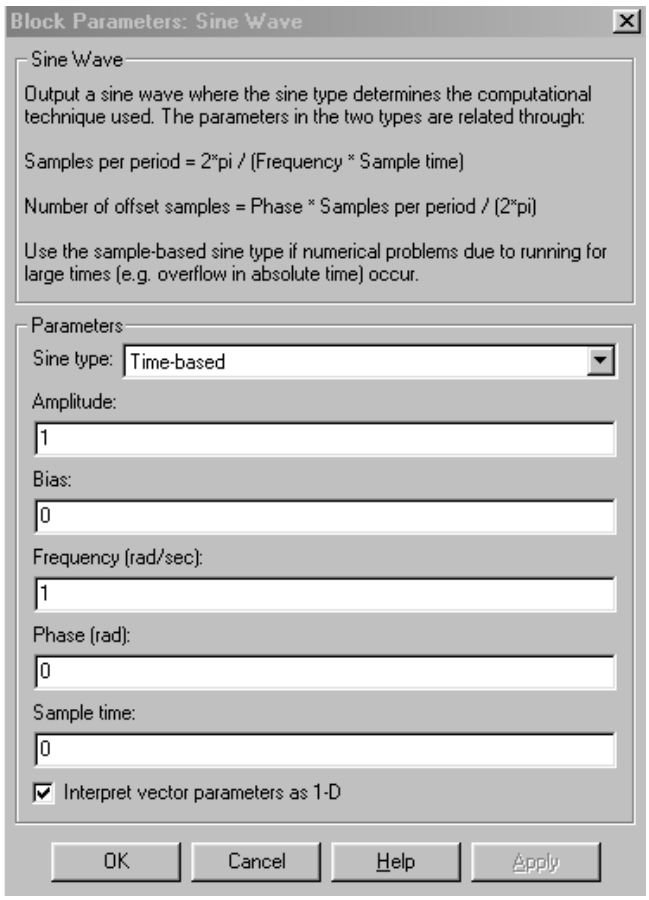


Рис.5

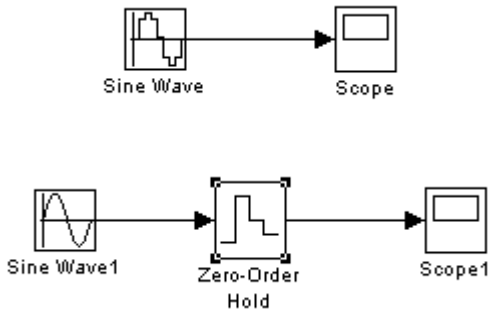


Рис.6

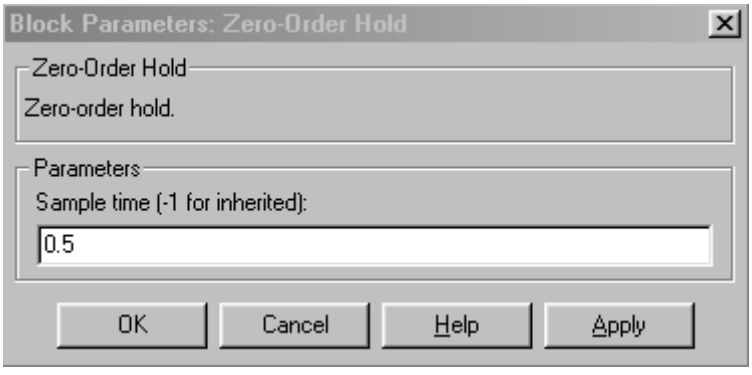


Рис.7

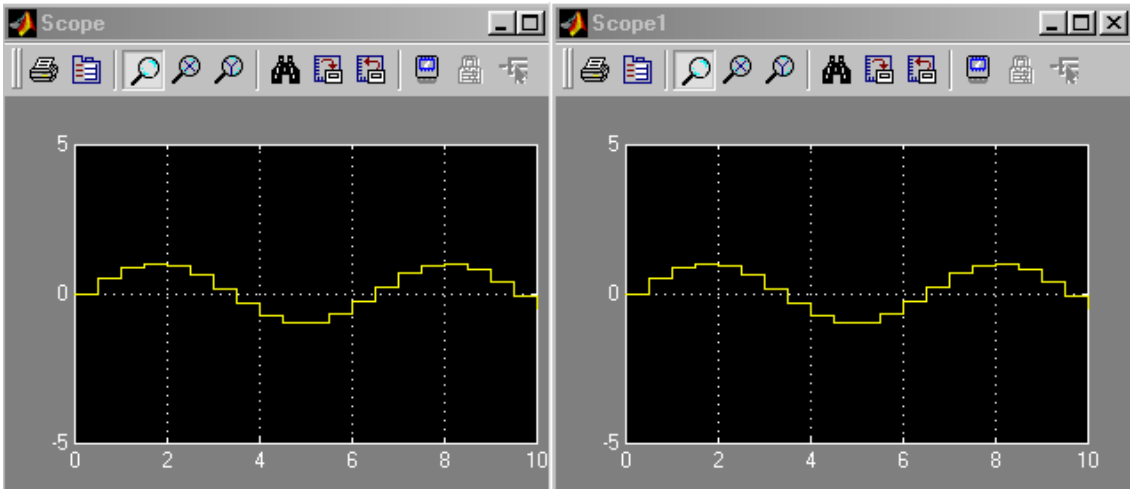


Рис.8

Таким образом, блок **Zero-Order Hold** можно трактовать как “дискретизатор”, т.е. часть АЦП, ответственную за дискретизацию сигнала. Иногда блок **Zero-Order Hold** именуют АЦП [4]. По нашему мнению, это не корректно, поскольку дискретизированный сигнал в “подлинном” АЦП подвергается еще и квантованию по уровню. В блоке **Zero-Order Hold,** однако, квантование не производится.

Несколько слов о построении графиков. Помимо блока **Scope**, график можно построить и с помощью блока **X-Y-Graf**, на верхний вход X которого нужно подать последовательность моментов времени с помощью блока **Clock** (часы), а на нижний вход Y – значения генерируемого сигнала (рис.9).

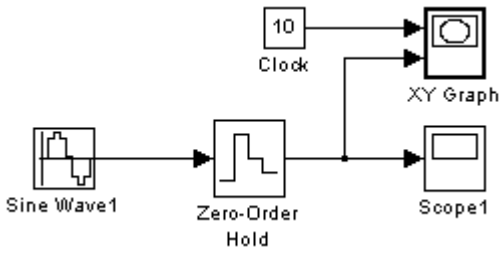


Рис.9

В результате предварительно настроенный (в соответствующем окне настройки задаются граничные значения аргумента и функции, а также указывается значение параметра **Sample time**) графопостроитель выдаст показанный на рис.10 график, если для блока **X-Y-Graf** задано **Sample time=-1** (т.е. период дискретизации наследуется).

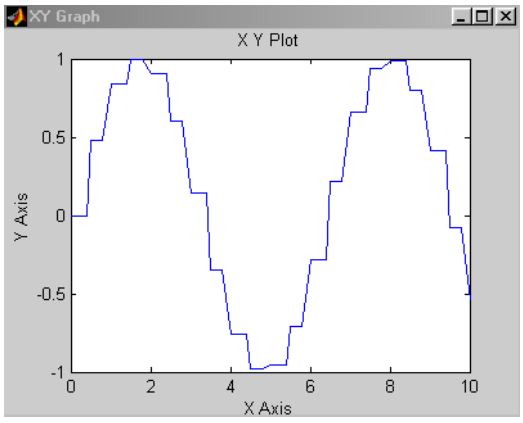


Рис.10

График будет несколько иным (рис.11), если для блока **X-Y-Graf** задано **Sample time=0.5**.

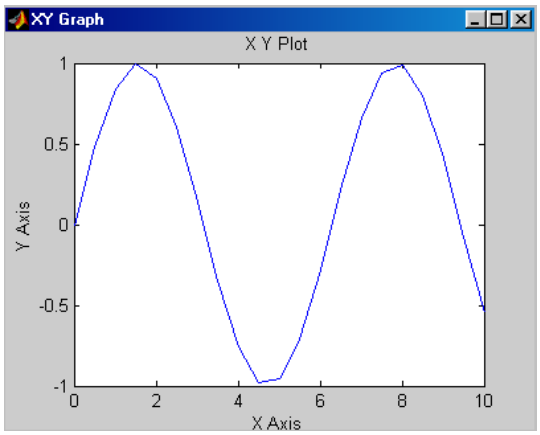


Рис.11

Еще об одном способе построения графиков. Массивы отсчетов моментов времени и соответствующих значений сигнала можно с помощью блока **To Workspace** экспортировать из среды **Simulink** в среду **Matlab** (рис.13).

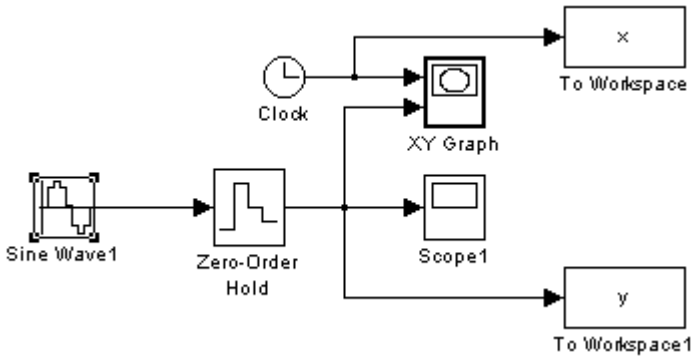


Рис.13

При этом, как показывает практика, лучше всего задать формат **array** для экспортируемых данных (рис.14).

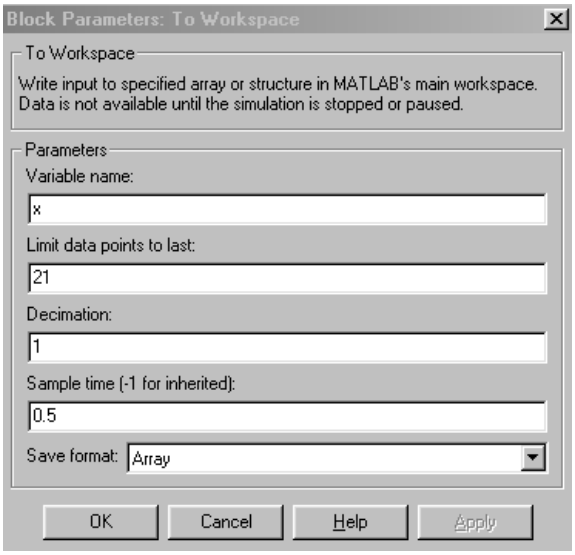


Рис.14

Дальнейшее построение графика в среде **Matlab** с помощью команды **plot(x,y)** не представляет никакого труда (рис.15).

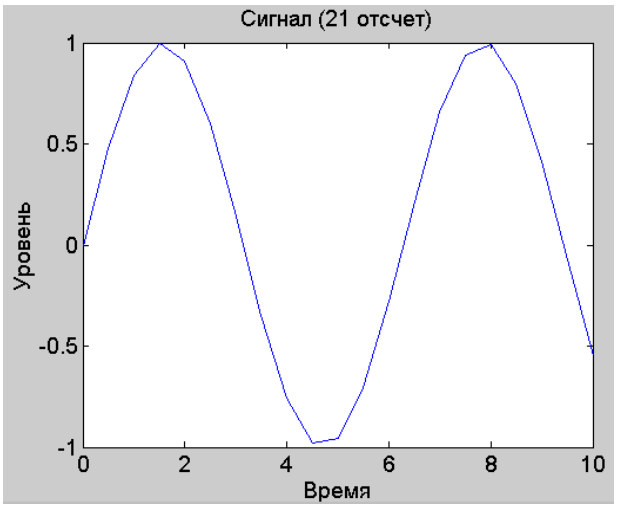


Рис.15

Подытожим результаты проведенных опытов.

Сигнал типа **time-based** при работе блока генерации в режиме непрерывного времени имеет вид гладкой функции времени, а в режиме дискретного времени - вид ступенчатого сигнала, такого, как если бы к выходу генератора плавного сигнала был подсоединен блок **Zero-Order Hold,** являющийся дискретизатором типа “отсчет-хранение”.

Иными словами, задавая режим дискретного времени, мы уходим от необходимости в использовании блока **Zero-Order Hold**.

А теперь сгенерируем в **Simulink** отрезок дискретного гармонического сигнала с темы же параметрами, что были заданы в **Matlab**: амплитуда 1, частота 100 Гц, частота дискретизации 1000 Гц, начальная фаза $\pi/2$, количество отсчетов 20.

Собираем снова схему из генератора и осциллоскопа. В окне-маске настройки генератора производим указание нужных числовых значений параметров, задаем тип **time-based** и присваиваем значение Sample time = 0.001 (рис.16).

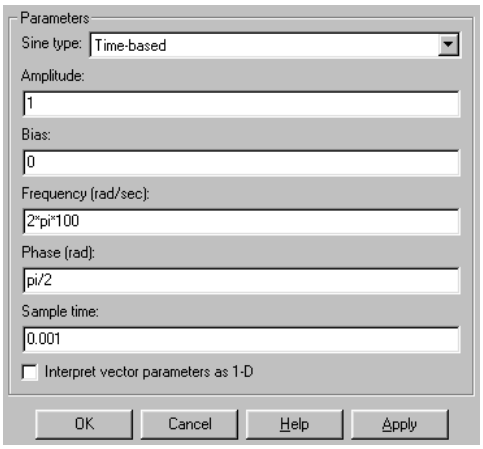


Рис.16

После запуска модели получаем на экране осциллоскопа совсем не ту картину, которую ожидали (рис.17).

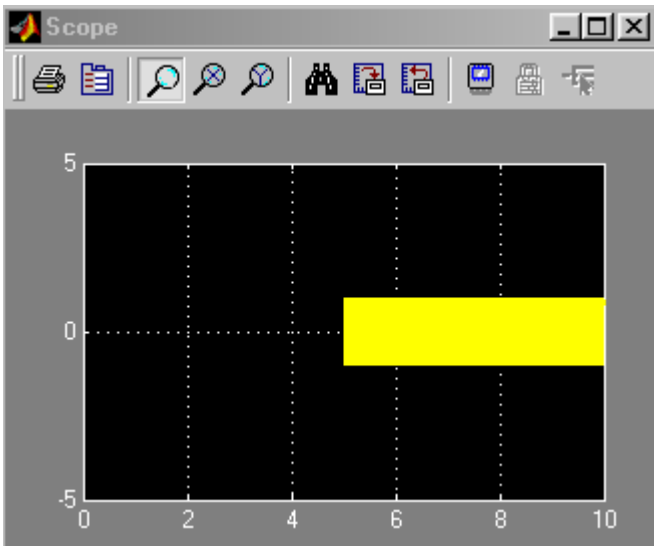


Рис.17

Причина проста – нужно еще настроить параметры моделирования: задать начало и конец модельного времени (в нашем случае это 0 и 0.02 с, соответственно), а также выбрать алгоритм моделирования (тип «решателя»). На рис.18 показано окно настроек параметров моделирования, активизирующееся при выборе позиции меню **Simulation/Simulation parameters**.

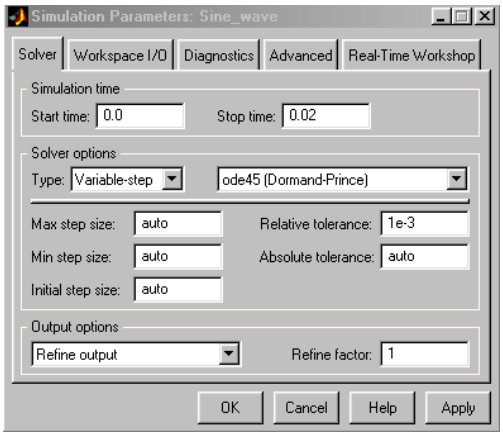
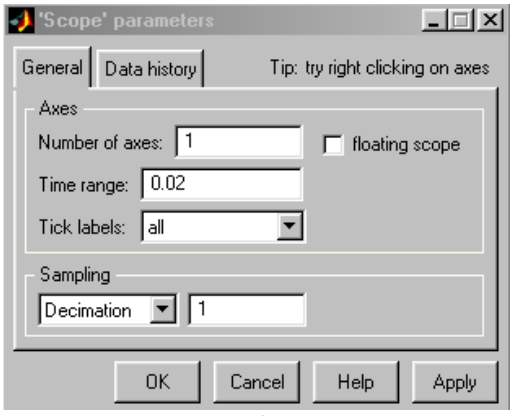
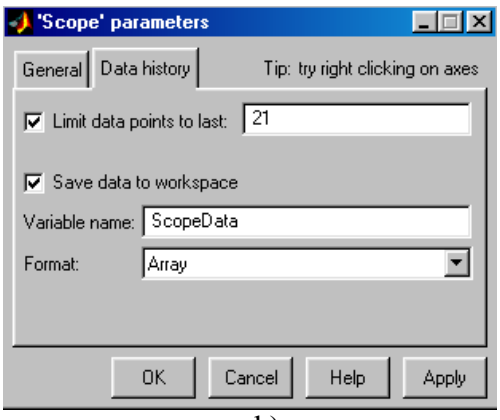


Рис.18

Кроме того, настроим параметры осциллоскопа, щелкнув по кнопке **Parameters** на окне **Scope** (рис.19a,b).



a)



b)

Рис.19

После запустка модели на экране осциллоскопа появится изображение (рис.20).

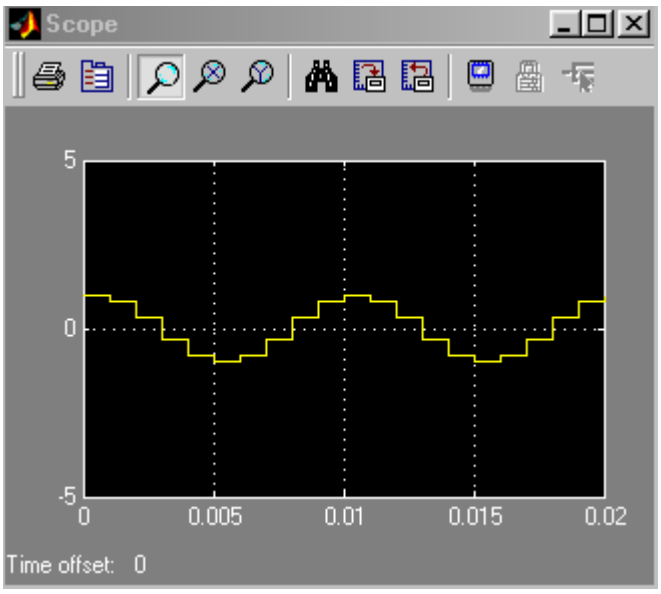


Рис.20

Поскольку параметры осциллоскопа были заданы так, чтобы в рабочее пространство выводился двумерный массив **ScopeData** значений аргумента и функции, с помощью команд

```
>> y1=ScopeData(:,1);
>> y2=ScopeData(:,2);
>> plot(y1,y2)
```

можно построить график сгенерированной функции средствами **Matlab** (рис.21).

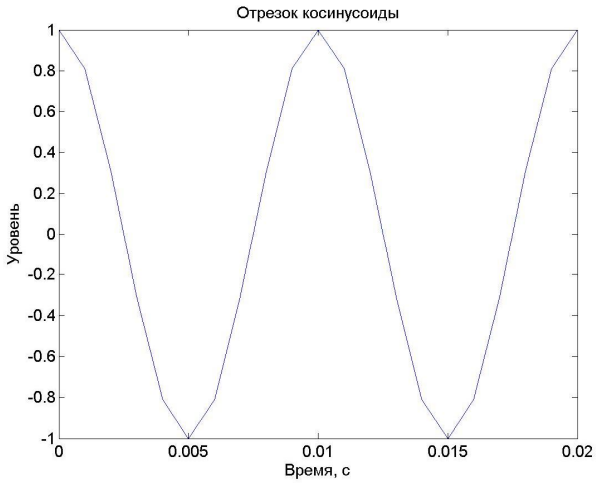


Рис.21

Сравнивая рис.21 и рис.1, замечаем лишь одно отличие – при моделировании в **Simulink** сгенерирована 21 точка, тогда как в **Matlab** генерировалось 20 точек. Причина различия проста: на интервале модельного времени T при частоте дискретизации F_s находится TF_s+1 моментов времени, для которых будет сгенерирован сигнал. Очевидно, это обстоятельство легко учесть, добившись полного совпадения результатов моделирования в средах **Matlab** и **Simulink**.

Литература

1. Продеус А.Н., Родінова М.В. Безпаперова технологія проведення практикумів із статистичної обробки сигналів. – Электроника и связь, №20, 2003, pp.117-120

2. Гультяев А. Имитационное моделирование в среде Windows. – С-Пб, КОРОНА принт, 1999. – 287 с.
3. Скляр Б. Цифровая связь. Теоретические основы и практическое применение. – М.,С-Пб,К., изд.дом «Вильямс», 2003. – 1092 с.
4. Калюжний О.Я. Моделювання систем передачі сигналів в обчислювальному середовищі MATLAB-Simulink. – К., "Політехніка", 2004. – 135 с.