Содержание справки

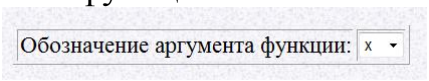
1. Описание построителя графиков

1.1.Предназначение

Построитель графиков осуществляет визуализацию графиков функций, заданных JavaScript – программным кодом. Для корректной работы построителя рекомендуется использовать обозреватель Internet Explorer.

1.2. Задание функции

Для задания функции следует выбрать из списка обозначение аргумента в коде задаваемой функции:



Выбор определяется исключительно соображениями удобства при задании кода функции.

Код функции задаётся на языке сценариев JavaScript в поле «Тело функции»:

При этом допускается:

1. Объявление и использование переменных. Для объявления переменной используется ключевое слово **`var`**. Пример **`var a = 0.0;`** - объявляется переменная «a» типа число («Number»), которой присваивается значение 0.
2. Использование операторов JavaScript:

2.1. Арифметических (сложение «**+**», вычитание «**-**», умножение «*****», деление «**/**»).

2.2. Операторов сравнения (равенство «**==**», неравенство «**!=**», больше «**>**», меньше «**<**», больше или равно «**>=**», меньше или равно «**<=**»), логических операторов (И «**&&**», Не «**!**», ИЛИ «**||**»).

2.3. Операторов условного перехода:

```
if(условие)
{блок операторов}
```

или

```
if(условие)
```

```
{блок операторов}
else
{блок операторов}
```

3. Задание функций внутри кода с использованием ключевого слова **`function`**.

Не рекомендуется использование циклов при задании функции.

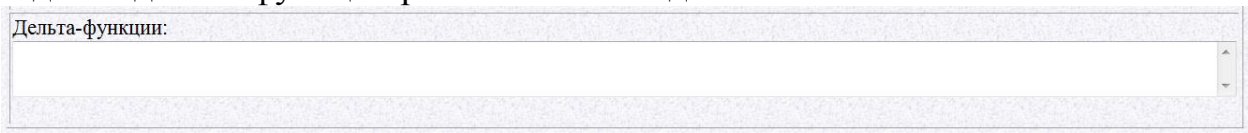
Записываемый в поле «тело функции» код обязательно должен генерировать возвращаемое значение и содержать оператор return.

1.3. Таблица математических функций и констант, предусмотренных в JavaScript:

Описание	Функция
Константы	
число π	**`Math.PI`**
число e	**`Math.E`**
Арифметические функции	
экспонента e^x	**`Math.exp(x)`**
натуральный логарифм ln (x)	**`Math.log(x)`**
возведение в степень x^y	**`Math.pow(x,y)`**
квадратный корень $\sqrt{x}$	**`Math.sqrt(x)`**
Тригонометрические функции (углы в радианах)	
синус sin (x)	**`Math.sin(x)`**
косинус cos (x)	**`Math.cos(x)`**
тангенс $tg(x)$	**`Math.tan(x)`**
арксинус arcsin (x)	**`Math.asin(x)`**
арккосинус arccos (x)	**`Math.acos(x)`**
арктангенс $arctg(x)$	**`Math.atan(x);`**
аргумент комплексного числа z=x+iy arg (x, y)	**`Math.atan2(y,x)`**

1.4. Изображение дельта-функций

На графике можно изображать дельта-функции (не более 100 штук). Задание дельта-функций реализовано в отдельной текстовой зоне:



Дельта-функции задаются в формате:

`Delta(x0,k,h);`

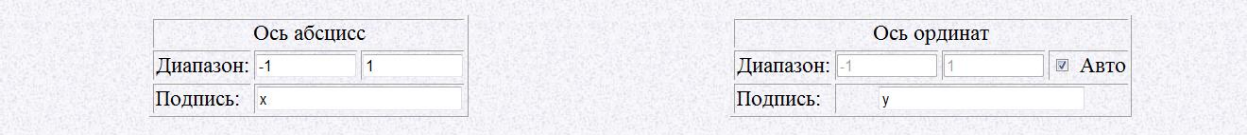
где **`x0`** – абсцисса, которой соответствует дельта-функция,

`k` – коэффициент при дельта-функции,

h – размер дельта функции. Этот параметр определяет размер изображения дельта-функции на графике, его выбор определяется соображениями наглядности получаемого рисунка. Дельта-функции перечисляются через точку с запятой.

1.5. Диапазоны по осям абсцисс и ординат и подписи осей

После описания функции следует задать диапазон изменения абсциссы при построении графика. Диапазон отображаемых ординат может быть задан пользователем или определён автоматически:



При задании подписей по оси абсцисс и ординат допускается использовать html – теги, позволяющие изменять цвет и размер текста. Отдельно отметим возможность задания нижних и верхних индексов. Нижний индекс задаётся тегом **_{*нижний индекс*}**, верхний – тегом **^{*верхний индекс*}**. Например строка «**R₁(т), B²c**» в поле «подпись» задаёт подпись оси ординат вида:

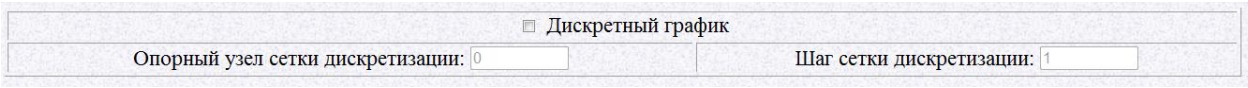
При задании подписей осей координат при выполнении курсовой работы или отчётов по лабораторным работам требуется использовать кодировку LaTeX. Подпись должна начинаться с управляющей комбинации символов «\(» заканчиваться комбинацией «\)». С правилами набора формул в LaTeX можно ознакомиться в книге Львовский С.М. Набор и верстка в системе LaTeX, или использовать онлайн редакторы формул. Полезные ссылки можно найти на форуме сайта. Некоторые примеры подписей осей координат приведены в таблице:

\(s(t),B\)	$s(t), B$
\(s_d(t),\frac{B}{\text{мс}}\)	$s_d(t), \frac{B}{\text{мс}}$
\(|S(\omega)|,\text{B}\cdot \text{мс}\)	$\|S(\omega)\|, \text{В} \cdot \text{мс}$
\(\varphi_S(\omega),\text{рад}\)	$\varphi_S(\omega), \text{рад}$
\(\{A_n\},B\)	$\{A_n\}, B$
\(\{\varphi_n\},\text{рад}\)	$\{\varphi_n\}, \text{рад}$
\(\psi(\omega),\text{рад}\)	$\psi(\omega), \text{рад}$
\(\{\psi_n\},\text{рад}\)	$\{\psi_n\}, \text{рад}$
\(u_{\text{вх}}(t),B\)	$u_{\text{вх}}(t), B$

`\(\omega \frac{\text{рад}}{\text{мс}}\)`	$\omega, \frac{\text{рад}}{\text{с}}$

1.6. Построение дискретных графиков

Реализована возможность построения дискретных графиков. Для этого следует установить флажок в разделе «Дискретный график» и задать опорный узел сетки дискретизации (это любой узел, принадлежащий сетке дискретизации) и шаг дискретизации. Дискретный график получается в результате дискретизации заданной функции на определённой таким образом сетке.



1.7. Оформление графиков

Возможности по оформлению графика исчерпываются заданием названия графика, цвета линии графика, толщины линии и цвета линий сетки:

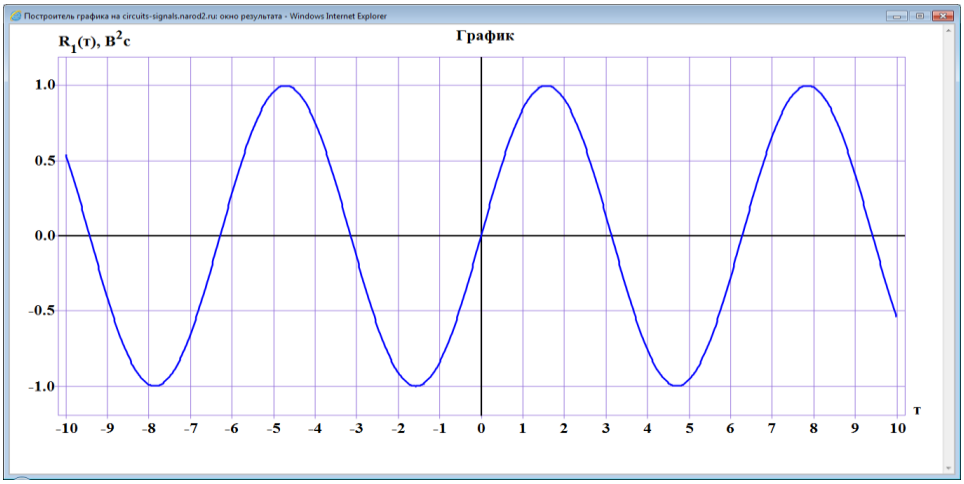
При задании названия графика допускается использование html – тегов форматирования текста.

1.8. Окно результата

После описания графика следует нажать кнопку «построить график»:

Построитель графиков является средством для построения графиков, а не средством поиска ошибок в JavaScript коде, допущенных пользователем при задании функции. Поэтому, если после нажатия кнопки «построить график» ничего не произошло и никакого сообщения с веб-страницы не последовало, то пользователю следует убедиться, что он использует браузер Internet Explorer, после чего заняться поиском допущенных при описании функции ошибок.

Если ошибки отсутствуют, то будет открыто отдельное окно обозревателя с графиком:



Для копирования содержимого окна следует нажать клавишу «PrintScreen» (она же «PrnScr», «PrtSc»), после чего, скопированное с экрана содержимое можно помесить в документ Word или вставить в рабочую зону графического редактора выполнением команды «Вставить из буфера» (обычно комбинация клавиш Shift+Ins или Ctrl+V). Полученные графики можно также выводить на печать непосредственно из обозревателя, с помощью команды «печать».

При закрытии основного окна построителя графиков отдельное окно с результатом тоже будет закрыто. Учитывайте это, чтобы не потерять результат.

2. Примеры построения графиков для курсовой работы

2.1. Построение графика непериодического сигнала

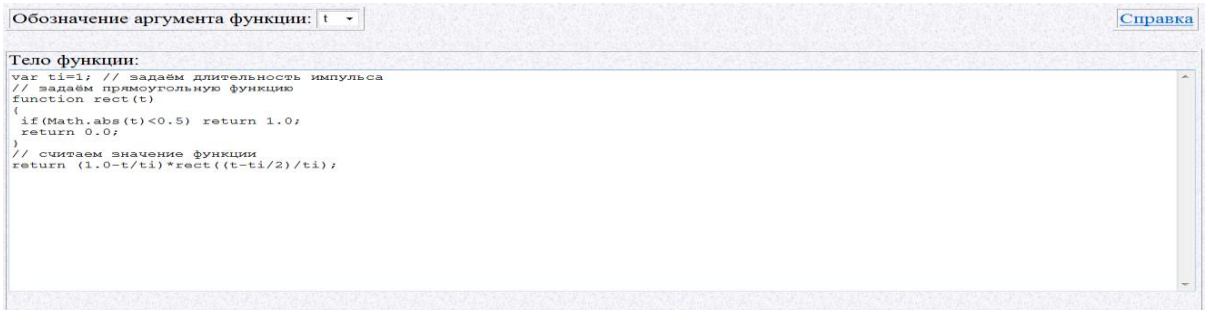
Сигнал описывается выражением

$$s(t) = \left(1 - \frac{t}{\tau_и}\right) rect\left(\frac{t-\frac{\tau_и}{2}}{\tau_и}\right) = \begin{cases} 1 - \frac{t}{\tau_и}, 0 \le t \le \tau_и \\ 0, t < 0, t > \tau_и \end{cases}.$$

Выбираем обозначение аргумента функции «t».

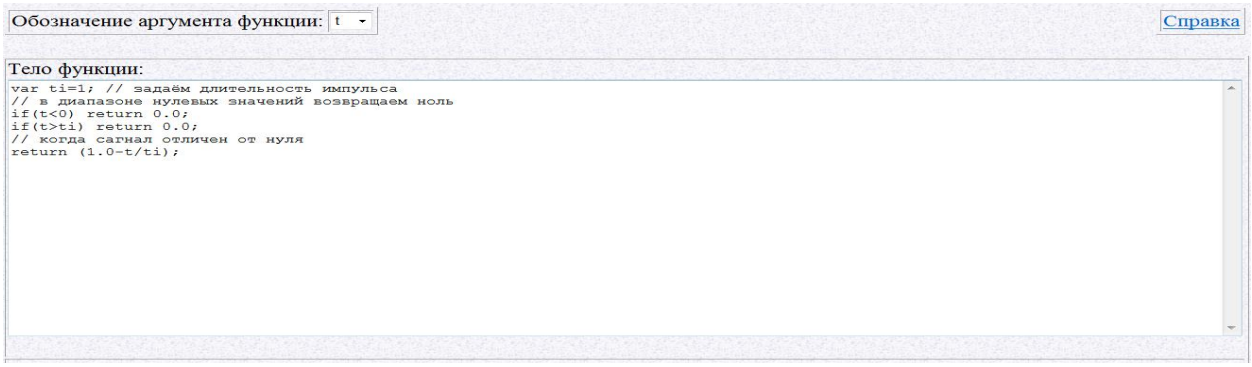
Первый вариант кода для тела функции:

```
var ti=1; // задаём длительность импульса
function rect(t)
{
 if(Math.abs(t)<0.5) return 1.0;
 return 0.0;
}
// считаем значение функции
return (1.0-t/ti)*rect((t-ti/2)/ti);
```

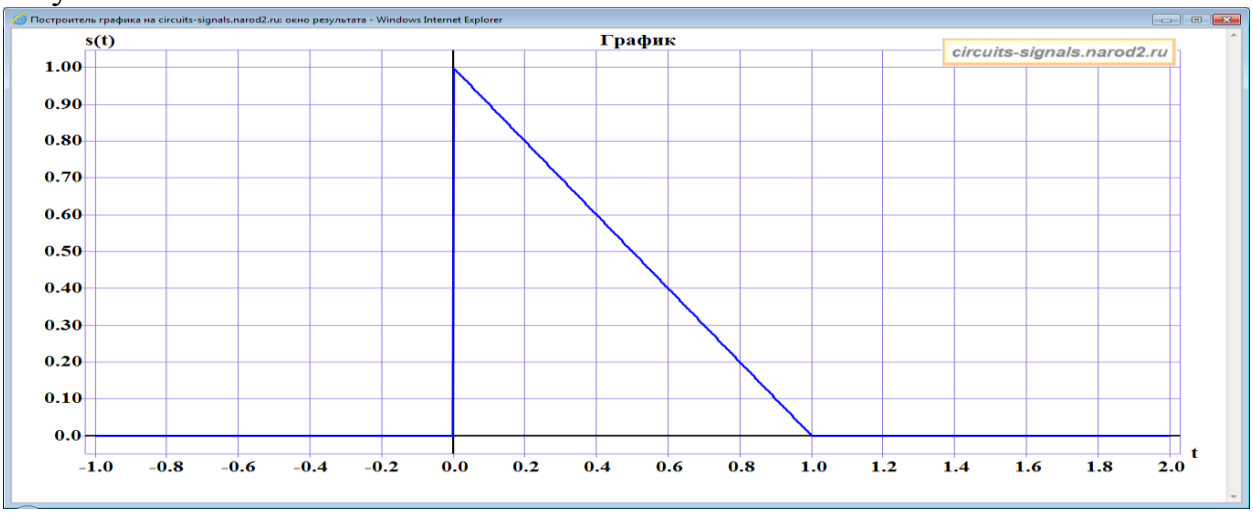


Второй вариант кода:

```
var ti=1;// задаём длительность импульса
// в диапазоне нулевых значений возвращаем ноль
if(t<0) return 0.0;
if(t>ti) return 0.0;
return (1.0-t/ti);
```



Результат:



При выполнении курсовой работы код, сделанный при построении графика заданного сигнала рекомендуется сохранить – он понадобится при построении графиков периодического сигнала и радиосигналов. Для сохранения текст следует выделить, копировать, вставить в любой текстовый редактор, например «Блокнот», и сохранить в файл.

2.2. График с дельта-функциями

Сигнал описывается выражением

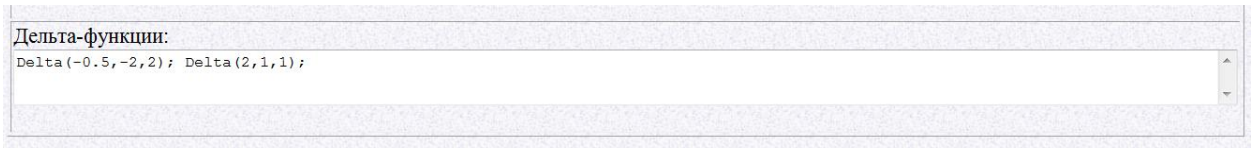
$$s(t) = -2\delta(t + 0.5) + rect(t) + \delta(t - 1).$$

Пример кода:

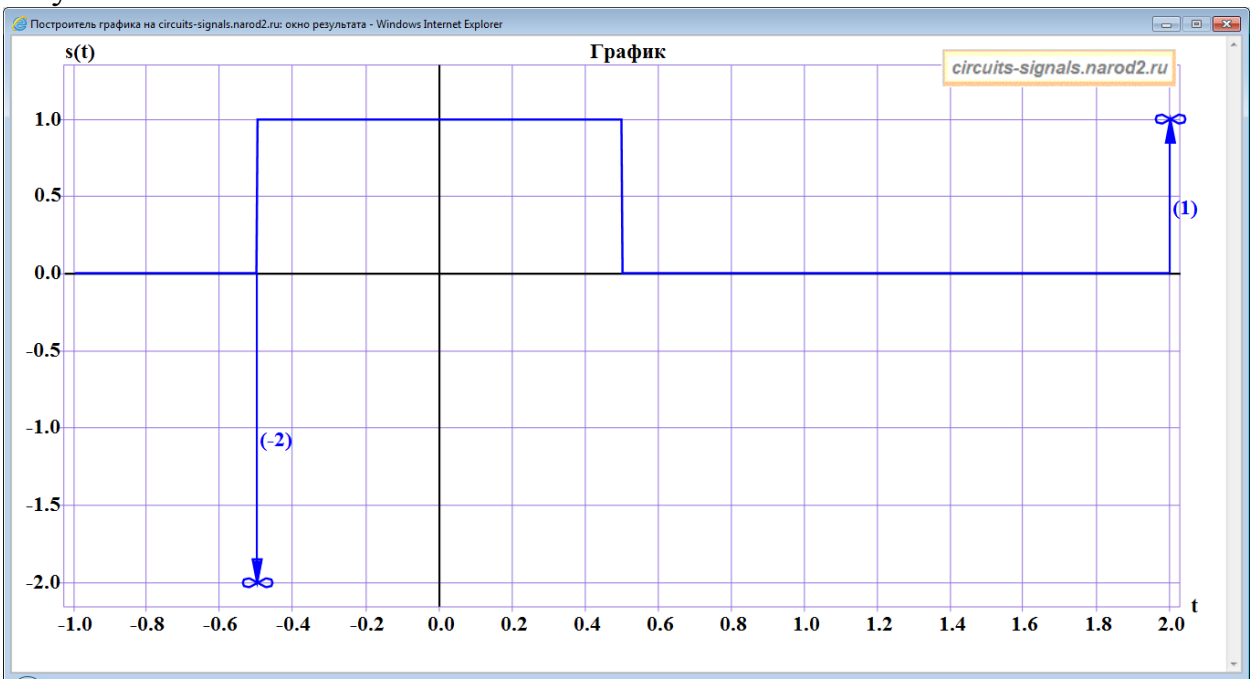
```
if(Math.abs(t)<0.5) return 1.0;
return 0.0;
```

Задание дельта-функций (для наглядности высота дельта-функций сделана равной их коэффициентам):

```
Delta(-0.5,-2,2); Delta(2,1,1);
```



Результат:



2.3. Построение графиков амплитудного и фазового спектра сигналов

В случае, когда для амплитудного и фазового спектров получены не очень громоздкие выражения, построение графиков осуществляется аналогично п.2.2. В большинстве случаев, выражение для спектральной плотности оказывается компактным, а для амплитудного и фазового спектров получаются громоздкие выражения. Рассмотрим подобный пример:

Спектральная плотность

$$S(\omega) = \frac{1}{2}\tau_{и1} sinc\left(\frac{\omega\tau_{и1}}{2}\right) e^{j\omega t_{01}} + \tau_{и2} sinc\left(\frac{\omega\tau_{и2}}{2}\right) e^{-j\omega t_{02}}.$$

Выделяем действительную и мнимую часть:

$$ReS(\omega) = \frac{1}{2}\tau_{и1} sinc\left(\frac{\omega\tau_{и1}}{2}\right) \cos(\omega t_{01}) + \tau_{и2} sinc\left(\frac{\omega\tau_{и2}}{2}\right) \cos(\omega t_{02})$$

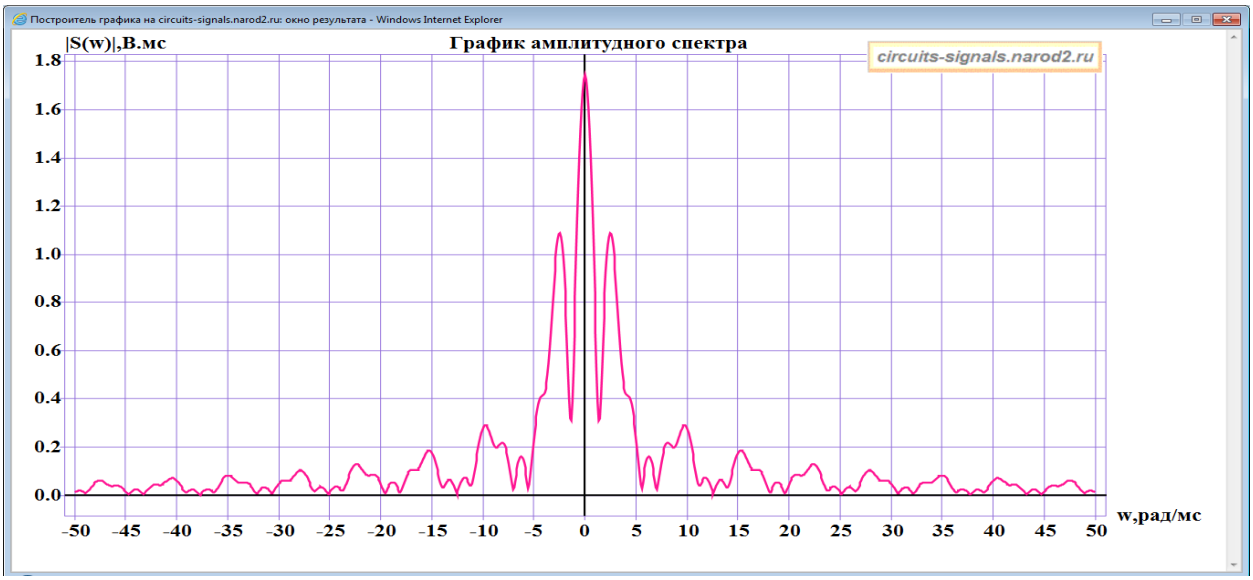
$$ImS(\omega) = \frac{1}{2}\tau_{и1} sinc\left(\frac{\omega\tau_{и1}}{2}\right) \sin(\omega t_{01}) - \tau_{и2} sinc\left(\frac{\omega\tau_{и2}}{2}\right) \sin(\omega t_{02})$$

Код для построения амплитудного спектра:

```
// параметр сигнала
var t1=1.0;
// задаём длительность импульса
var ti1=1.5*t1;
var ti2=1.0*t1;
// параметры сдвига
var t01=1.5*t1/2.0;
var t02=(t1+2.0*t1)/2.0;
```

```
// определяем sinc
function sinc(w)
{
 if(w==0) return 1.0;
 return Math.sin(w)/w;
}
// действительная часть спектральной плотности
var ReS =
0.5*ti1*sinc(0.5*w*ti1)*Math.cos(w*t01)+ti2*sinc(0.5*
w*ti2)*Math.cos(w*t02);
// мнимая часть спектральной плотности
var ImS = 0.5*ti1*sinc(0.5*w*ti1)*Math.sin(w*t01)-
ti2*sinc(0.5*w*ti2)*Math.sin(w*t02);
// амплитудный спектр
return Math.sqrt(ReS*ReS + ImS*ImS);
```

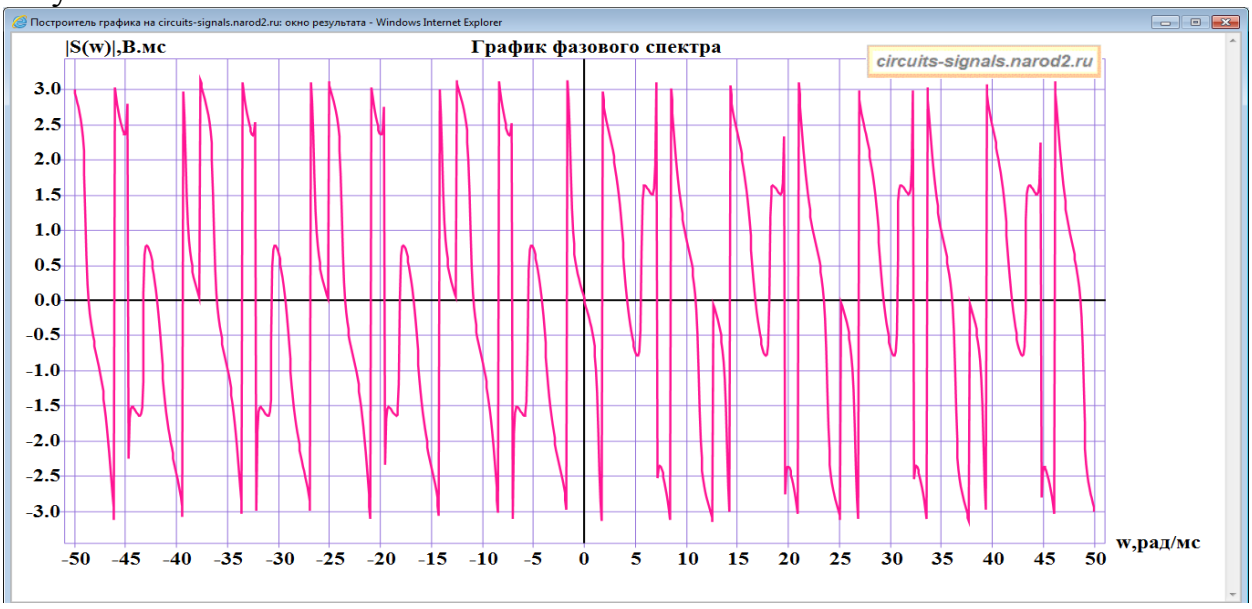
Результат:



Код рекомендуется сохранить, так как он понадобится при выполнении других пунктов курсовой работы.
Код для фазового спектра:

```
// параметр сигнала
var t1=1.0;
// задаём длительность импульса
var ti1=1.5*t1;
var ti2=1.0*t1;
// параметры сдвига
var t01=1.5*t1/2.0;
var t02=(t1+2.0*t1)/2.0;
// определяем sinc
function sinc(w)
{
 if(w==0) return 1.0;
 return Math.sin(w)/w;
}
// действительная часть спектральной плотности
var ReS =
0.5*ti1*sinc(0.5*w*ti1)*Math.cos(w*t01)+ti2*sinc(0.5*
w*ti2)*Math.cos(w*t02);
// мнимая часть спектральной плотности
var ImS = 0.5*ti1*sinc(0.5*w*ti1)*Math.sin(w*t01)-
ti2*sinc(0.5*w*ti2)*Math.sin(w*t02);
// фазовый спектр
return Math.atan2(ImS,ReS);
```

Результат:

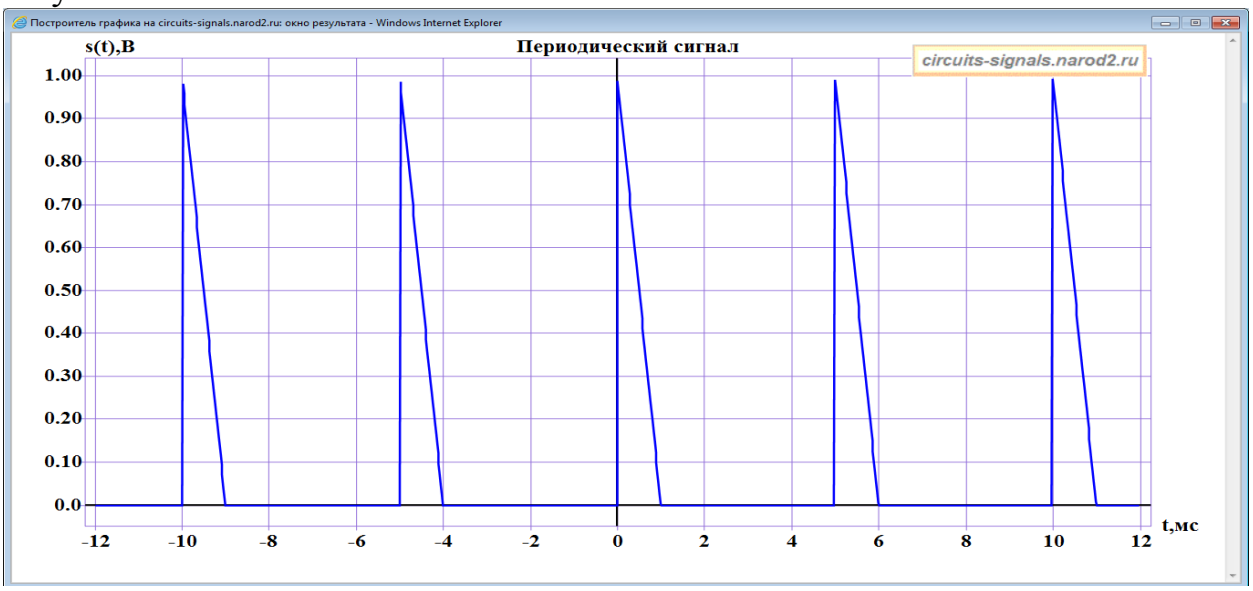


2.4. График периодического сигнала

Код:

```
var ti=1.0;// длительность сигнала
var T=5.0; // период сигнала
// задаём непериодический сигнал
function s(t)
{
 if(t<0.0) return 0.0;
 if(t>ti) return 0.0;
 return 1.0-t/ti;
}
// фрагмент периодического сигнала из 5-ти импульсов
return s(t+2.0*T)+s(t+T)+s(t)+s(t-T)+s(t-2.0*T);
```

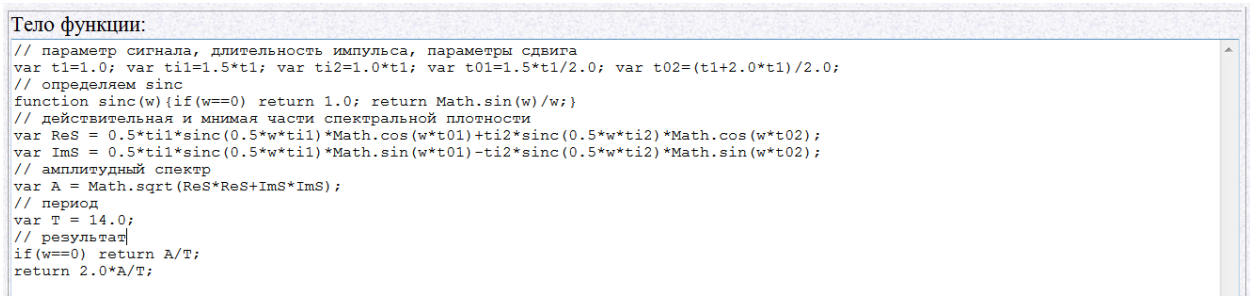
Результат:



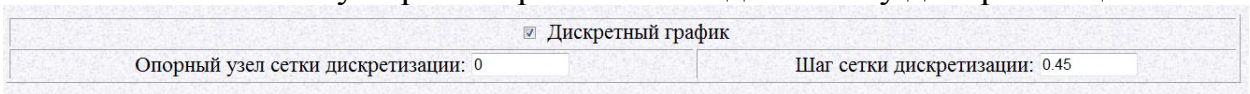
2.5. График спектра периодического сигнала

Восстанавливаем код из п.2.3, задаём период и добавляем множитель, который связывает спектры периодического и непериодического сигналов:

```
// параметр сигнала
var t1=1.0;
// задаём длительность импульса
var ti1=1.5*t1;
var ti2=1.0*t1;
// параметры сдвига
var t01=1.5*t1/2.0;
var t02=(t1+2.0*t1)/2.0;
// определяем sinc
function sinc(w)
{
 if(w==0) return 1.0;
 return Math.sin(w)/w;
}
// действительная часть спектральной плотности
var ReS =
0.5*ti1*sinc(0.5*w*ti1)*Math.cos(w*t01)+ti2*sinc(0.5*
w*ti2)*Math.cos(w*t02);
// мнимая часть спектральной плотности
var ImS = 0.5*ti1*sinc(0.5*w*ti1)*Math.sin(w*t01)-
ti2*sinc(0.5*w*ti2)*Math.sin(w*t02);
// амплитудный спектр
var A = Math.sqrt(ReS*ReS + ImS*ImS);
// период
var T=14.0;
// результат
if(w==0) return A/T; // постоянная составляющая
return 2.0*A/T;
```

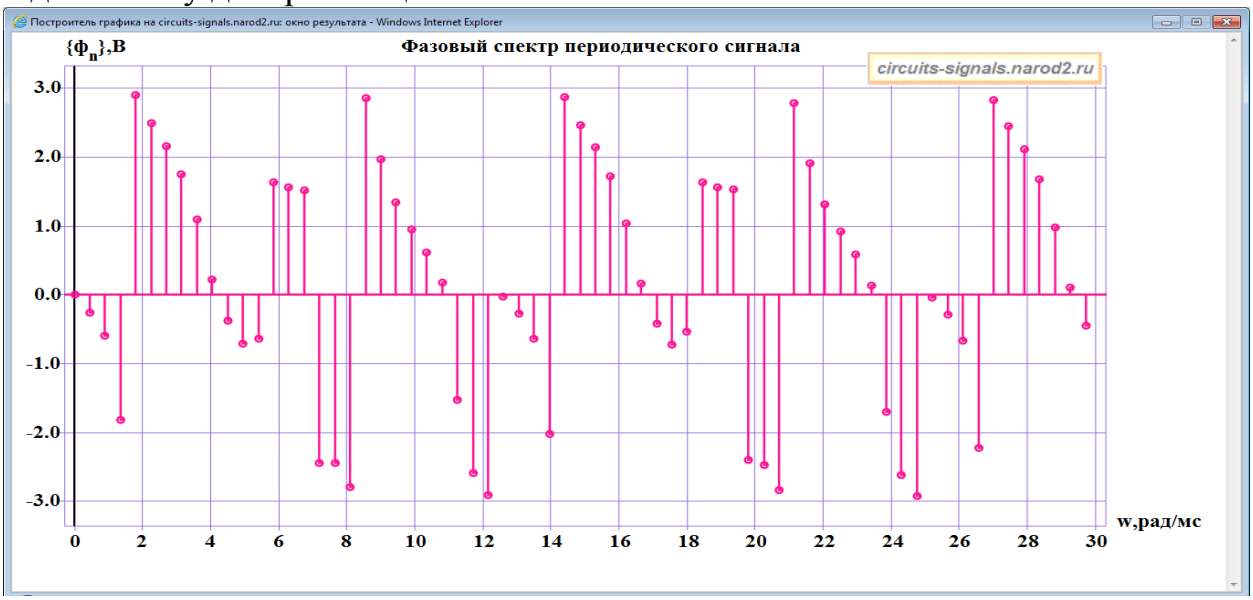


Рассчитываем частоту первой гармоники и задаём сетку дискретизации:



Результат:

Для графика фазового спектра просто восстанавливаем код из п.2.3 и задаём сетку дискретизации:

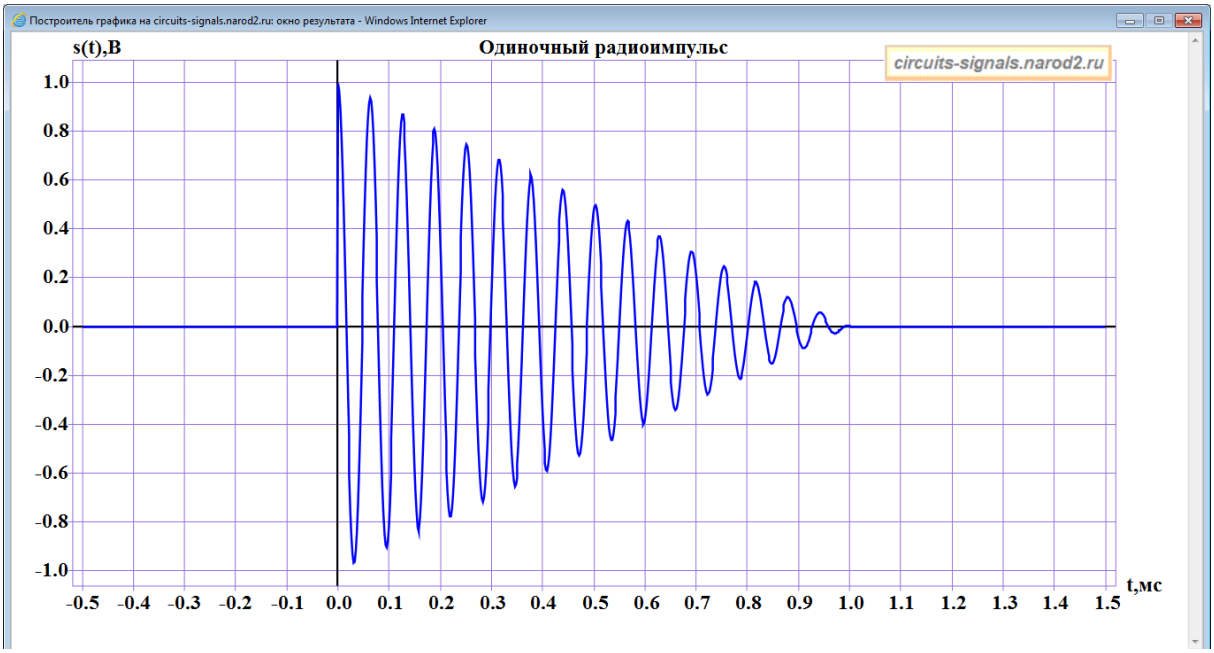


2.7. График радиоимпульса

```
var ti=1;
if(t<0) return 0.0;
if(t>ti) return 0.0;
return (1.0-t/ti)*Math.cos(100*t);
```

100 в множителе Math.cos(100*t) выбирается из соображений читаемости рисунка и, вообще говоря, будет не совпадать с несущей частотой, которая

будет фигурировать в курсовой работе.



2.8. Графики амплитудного и фазового спектров радиоимпульса

Будет полезным код, написанный в п.2.3, рассматриваем область положительных частот.

```
// задаём несущую частоту
var w0=1000.0;
// вводим смещение по частоте
w=w-w0;
// параметр сигнала
var t1=1.0;
// задаём длительность импульса
var ti1=1.5*t1;
var ti2=1.0*t1;
// параметры сдвига
var t01=1.5*t1/2.0;
var t02=(t1+2.0*t1)/2.0;
// определяем sinc
function sinc(w)
{
 if(w==0) return 1.0;
 return Math.sin(w)/w;
}
// действительная часть спектральной плотности
var ReS =
0.5*ti1*sinc(0.5*w*ti1)*Math.cos(w*t01)+ti2*sinc(0.5*
```

```
w*ti2)*Math.cos(w*t02);
// мнимая часть спектральной плотности
var ImS = 0.5*ti1*sinc(0.5*w*ti1)*Math.sin(w*t01)-
ti2*sinc(0.5*w*ti2)*Math.sin(w*t02);
// амплитудный спектр
return 0.5*Math.sqrt(ReS*ReS + ImS*ImS);
```

Тело функции:

```
// задаём несущую частоту
var w0=1000;
//вводим смещение по частоте
w = w-w0;
// задаём параметры импульса
var t1=1.0; var ti1=1.5*t1; var ti2=1.0*t1;
var t01=1.5*t1/2.0; var t02=(t1+2.0*t1)/2.0;
// определяем sinc
function sinc(w)
{
 if(w==0) return 1.0;
 return Math.sin(w)/w;
}
// действительная и мнимая часть спектральной плотности
var ReS = 0.5*ti1*sinc(0.5*w*ti1)*Math.cos(w*t01)+ti2*sinc(0.5*w*ti2)*Math.cos(w*t02);
var ImS = 0.5*ti1*sinc(0.5*w*ti1)*Math.sin(w*t01)-ti2*sinc(0.5*w*ti2)*Math.sin(w*t02);
// амплитудный спектр
return 0.5*Math.sqrt(ReS*ReS+ImS*ImS);
```

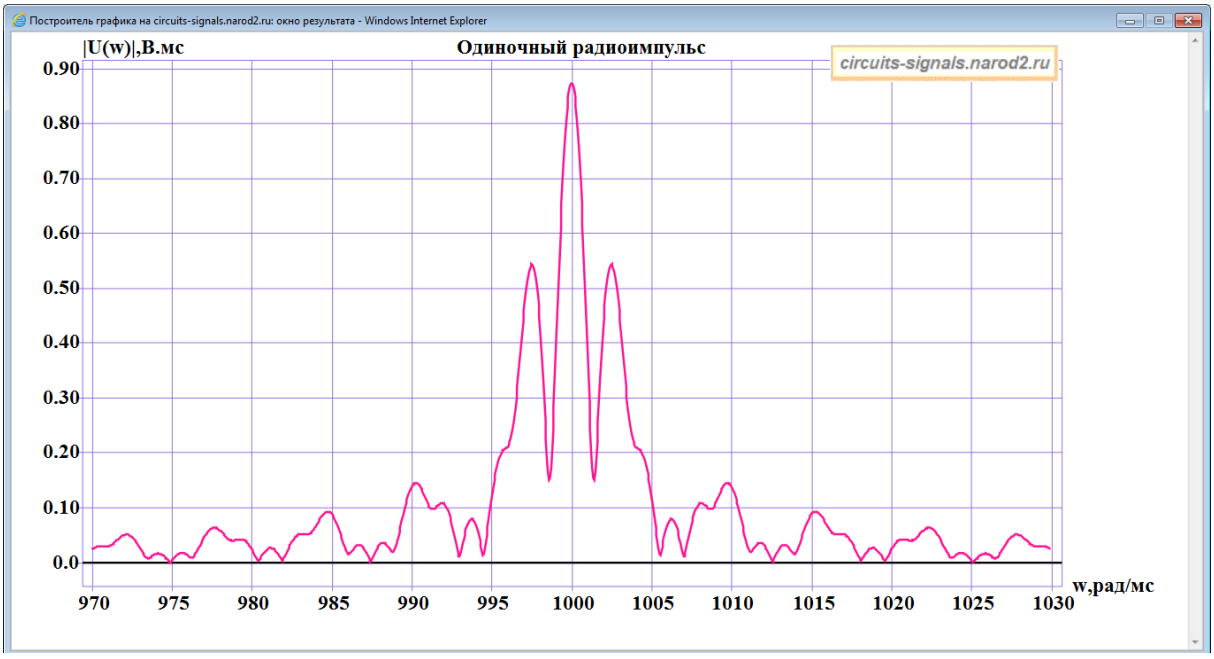


График фазового спектра строим аналогично.

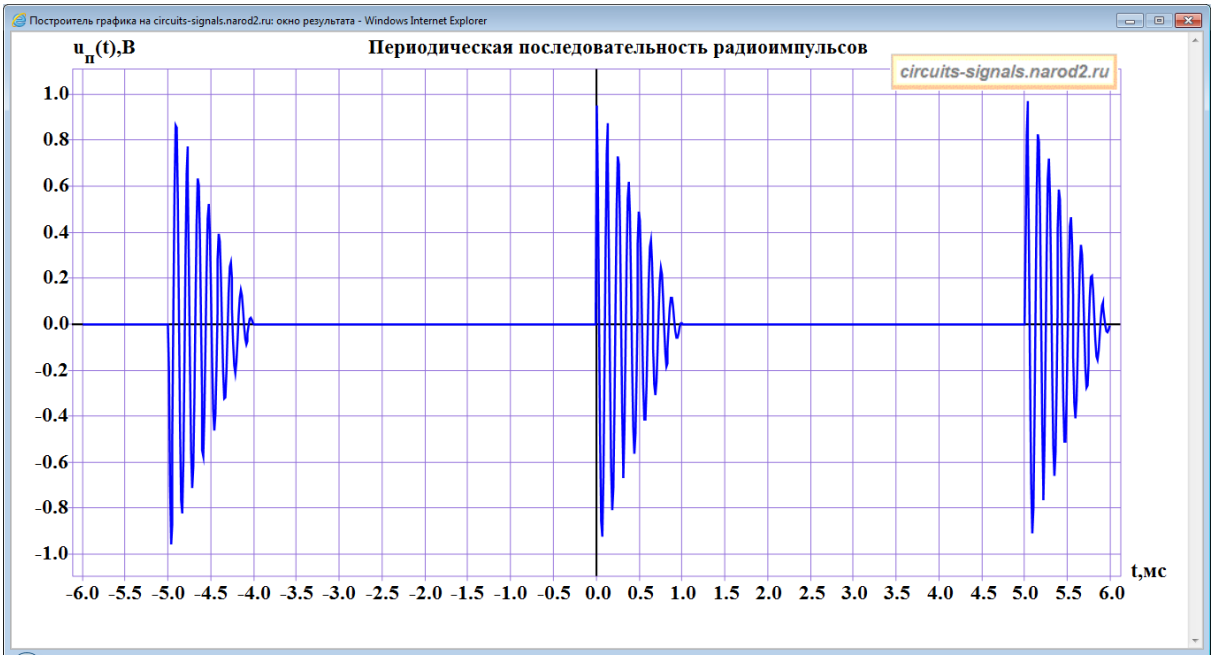
2.9. График периодической последовательности радиоимпульсов

Будет полезным код, написанный в п.2.4.

```
var ti=1.0;// длительность сигнала
var T=5.0; // период сигнала
// задаём непериодический сигнал
function s(t)
{
 if(t<0.0) return 0.0;
 if(t>ti) return 0.0;
 return 1.0-t/ti;
}
// фрагмент периодического сигнала из 5-ти импульсов
return (s(t+2.0*T)+s(t+T)+s(t)+s(t-T)+s(t-2.0*T))*Math.cos(50.0*t);
```

Тело функции:

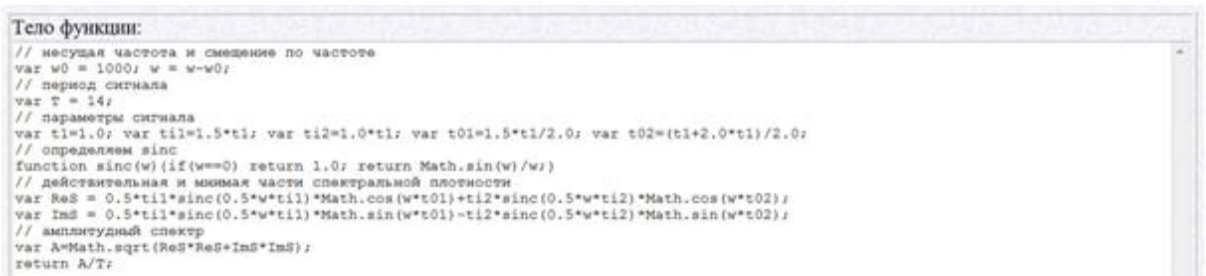
```
var ti = 1.0; // дительность сигнала
var T = 5.0;  // период сигнала
// задаём непериодический сигнал
function s(t)
{
 if(t<0.0) return 0.0;
 if(t>ti) return 0.0;
 return 1.0-t/ti;
}
// фрагмент периодического сигнала из 5-ти импульсов
return (s(t+2.0*T)+s(t+T)+s(t)+s(t-T)+s(t-2.0*T))*Math.cos(50*t);
```



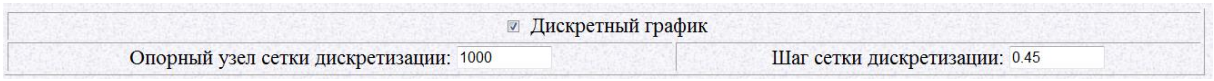
2.10. Графики амплитудного и фазового спектров периодической последовательности радиоимпульсов

Будет полезным код из п.2.3.

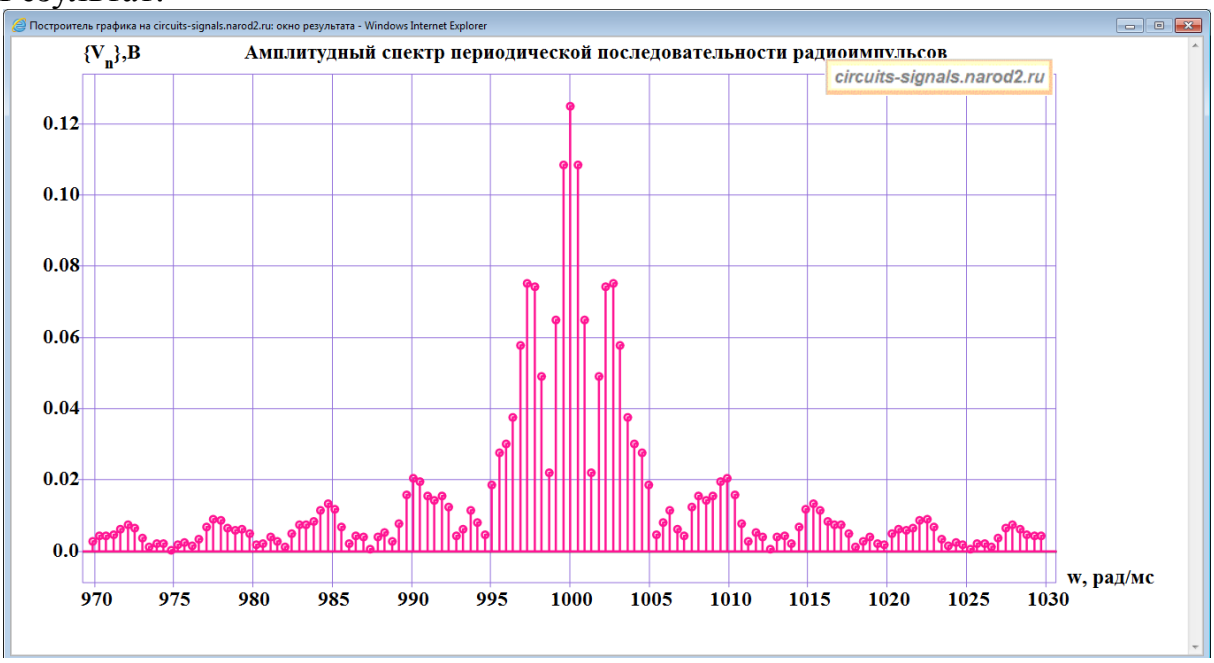
```
// задаём несущую частоту
var w0=1000.0;
// вводим смещение по частоте
w=w-w0;
// период сигнала
var T=14.0;
// параметр сигнала
var t1=1.0;
// задаём длительность импульса
var ti1=1.5*t1;
var ti2=1.0*t1;
// параметры сдвига
var t01=1.5*t1/2.0;
var t02=(t1+2.0*t1)/2.0;
// определяем sinc
function sinc(w)
{
 if(w==0) return 1.0;
 return Math.sin(w)/w;
}
// действительная часть спектральной плотности
var ReS =
0.5*ti1*sinc(0.5*w*ti1)*Math.cos(w*t01)+ti2*sinc(0.5*
w*ti2)*Math.cos(w*t02);
// мнимая часть спектральной плотности
var ImS = 0.5*ti1*sinc(0.5*w*ti1)*Math.sin(w*t01)-
ti2*sinc(0.5*w*ti2)*Math.sin(w*t02);
// амплитудный спектр
var A=Math.sqrt(ReS*ReS + ImS*ImS);
return A/T;
```



Обратите внимание на то, что опорный узел сетки дискретизации находится теперь на несущей частоте:



Результат:



3. Помощь

Если Вы не смогли найти свою ошибку в коде функции, не получилось разобраться с тем, как использовать построитель графиков, обнаружили факт некорректного функционирования построителя графиков, имеете интересную идею по развитию построителя графиков, нуждаетесь в иной помощи со стороны разработчика - пишите на форум или на почту сайта.

Для возможности разбора проблем использования построителя необходимо предъявить копию экрана.