МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение  
высшего образования «Южно-Уральский государственный университет»  
(национальный исследовательский университет)

Высшая школа электроники и компьютерных наук

Кафедра «Информационно-измерительная техника»

Интерполирование функций с помощью нейронных сетей

ОТЧЁТ

по практической работе № 1

по дисциплине «Интеллектуальные средства измерений»

Выполнил:

студент группы КЭ–413

/ С.С. Ильин /

(подпись)

« » 2022 г.

Проверил: доцент

/ А.С. Волосников /

(подпись)

« » 2022 г.

ОГЛАВЛЕНИЕ

[1. ЗАДАНИЕ 3](#_Toc99993428)

[2. ОПИСАНИЕ МЕТОДОВ 3](#_Toc99993429)

[3. ИНТЕРПОЛИРОВАНИЕ ФУНКЦИЙ 14](#_Toc99993430)

[4. ИТОГОВАЯ ТАБЛИЦА СКО 41](#_Toc99993431)

[5. ВЫВОД 41](#_Toc99993432)

# ЗАДАНИЕ

1) Для заданных функций построить интерпеллянт на основе:

* полинома Лагранжа
* полиномов n-ой степени (polyfit)
* кубических сплайнов (spline)
* сплайнов Эрмита (Piecewise Cubic Hermite Interpolating Polynomial - pchip)
* нейронных сетей: прямого распространения (feedforward), радиальных базисных функций (rbf) и обобщенных регрессионных (grnn).

2) В отчете в виде одного файла .pdf привести:

* титульный лист
* краткое описание используемых методов интерполирования
* графики интерполируемых и интерполированных функций
* код программы

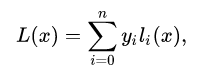
3) Заданные функции для интерполирования:

* линейная
* квадратичная
* экспоненциальная
* гиперболического тангенса
* гауссова
* гармоническая
* пилообразная
* меандр

# ОПИСАНИЕ МЕТОДОВ

1)Интерполяционный многочлен Лагранжа – многочлен минимальной степени, принимающий заданные значения в заданном наборе точек, то есть решающий задачу интерполяции.

Ж. Л. Лагранж предложил следующий способ вычисления таких многочленов:



где базисные полиномы *li* определяются по формуле



Для любого *i=0,…,n* многочлен *li* имеет степень n



Отсюда следует, что *L(x)*, является линейной комбинацией многочленов *li(x)*, имеет степень не больше n и *L(xi)=yi*.

Код полинома Лагранжа:

function Larg(x,y1,v,y1v,leg)

for z = 1:length(v)

for i = 1:length(x)

L(i)=1;

for j = 1:length(x)

if j~=i

S = (v(z)-x(j))/(x(i)-x(j));

L(i)=L(i)\*S;

end

L(i)= y1(i)\* L(i);

end

p1v(z) = sum(L); %Интерполированные значения

end

s1v = std(p1v-y1v); %СКО

s1v

figure; plot(x,y1,'--r', x,y1, 'bs', v, y1v, 'go',v,p1v,'m+'); grid;

title('Curve Fitting'); xlabel('x, s'); ylabel('Magnitude');

legend(leg, 'Узлы','Валидация', 'Полином');

Код вывода функций:

clear; clc; close all; % Очистка памяти и экрана выводы

%====================================================================================

x = 0:0.5:20; %Узлы интерполирования

v = 0.25:0.5:20; %Точки валидации

k = 0.5; b = -2; y1 = k\*x+b; leg = 'Прямая';

y1v = k\*v+b;

Larg(x,y1,v,y1v,leg);

f = 0.2; phi = -pi/7; y2 = sin(2\*pi\*x\*f+phi);

y2v = sin(2\*pi\*v\*f+phi);

leg = 'Гармоника';

Larg(x,y2,v,y2v,leg);

y3 = exp(x); leg = 'Экспоненциальная';

y3v = exp(v);

Larg(x,y3,v,y3v,leg);

y4 = tanh(2\*pi\*x\*f+phi); leg = 'Гиперболический тангенс';

y4v = tanh(2\*pi\*v\*f+phi);

Larg(x,y4,v,y4v,leg);

y5 = gauspuls(x,f); leg = 'Жордана-Гаусса';

y5v = gauspuls(v,f);

Larg(x,y5,v,y5v,leg);

y6 = sawtooth(2\*pi\*x\*f+phi); leg = 'Пилообразная функция';

y6v = sawtooth(2\*pi\*v\*f+phi);

Larg(x,y6,v,y6v,leg); ylim([-1 1]);

y7 = square(x); leg = 'меандр ';

y7v = square(v);

Larg(x,y7,v,y7v,leg);ylim([-1 1]);

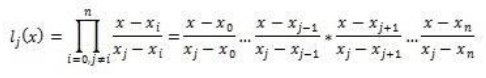
y8 = x.\*x; leg = 'Квадратичная';

y8v = v.\*v;

Larg(x,y8,v,y8v,leg);

2) Полиномы n-ой степени (polyfit)

Полиномиальная интерполяция является наиболее известным из методов одномерной интерполяции. Её достоинствами являются простота реализации и хорошее качество получаемых интерпеллянтов. Данный метод представляет полином n-ой степени P0, 1, …, n-1, n, проходящий через n точек (с 0-ой по n-ую), как функцию двух полиномов (n-1)-ой степени по формуле:



К полученным полиномам рекурсивно применяется та же формула, до тех пор, пока мы не дойдем до полиномов вида Pi, которые вычисляются по формуле Pi = yi. Достоинством данного метода является простота реализации, недостатком – сравнительно невысокое быстродействие.

Код полинома n-ой степени:

function poly(x,y1,v,y1v,leg,n)

p = polyval(polyfit(x,y1,n),v);

figure; plot(x,y1,'--r', x,y1, 'bs', v, y1v, 'go',v,p,'m+'); grid;

title('Curve Fitting'); xlabel('x, s'); ylabel('Magnitude');

legend(leg, 'Узлы','Валидация', 'Полином');

S = std(y1v-p)

end

Код вывода функций:

clear; clc; close all; % Очистка памяти и экрана выводы

%====================================================================================

x = 0:0.5:20; %Узлы интерполирования

v = 0.25:0.5:20; %Точки валидации

f = 0.2; phi = -pi/7;

k = 0.5; b = -2; y1 = k\*x+b; leg = 'Прямая';

y1v = k\*v+b;

n = 10;

poly(x,y1,v,y1v,leg,n);

f = 0.2; phi = -pi/7; y2 = sin(2\*pi\*x\*f+phi);

y2v = sin(2\*pi\*v\*f+phi);

leg = 'Гармоника';

n = 20;

poly(x,y2,v,y2v,leg,n);

y3 = exp(x); leg = 'Экспоненциальная';

y3v = exp(v);

n = 10;

poly(x,y3,v,y3v,leg,n);

y4 = tanh(2\*pi\*x\*f+phi); leg = 'Гиперболический тангенс';

y4v = tanh(2\*pi\*v\*f+phi);

n = 10;

poly(x,y4,v,y4v,leg,n);

y5 = gauspuls(x,f); leg = 'Жордана-Гаусса';

y5v = gauspuls(v,f);

n = 20;

poly(x,y5,v,y5v,leg,n);

y6 = sawtooth(2\*pi\*x\*f+phi); leg = 'Пилообразная функция';

y6v = sawtooth(2\*pi\*v\*f+phi);

n = 20;

poly(x,y6,v,y6v,leg,n);

y7 = square(x); leg = 'меандр ';

y7v = square(v);

n = 20;

poly(x,y7,v,y7v,leg,n);

k = 0.5; b = -2; y8 = x.\*x; leg = 'Квадратичная';

y8v = v.\*v;

n = 10;

poly(x,y8,v,y8v,leg,n);

3) Кубические сплайны (spline)

Сплайны позволяют эффективно решать задачи обработки экспериментальных зависимостей между параметрами, имеющих достаточно сложную структуру. Наиболее широкое практическое применение, в силу их простоты, нашли кубические сплайны. Основные идеи теории кубических сплайнов сформировались в результате попыток математически описать гибкие рейки из упругого материала (механические сплайны), которыми издавна пользовались чертежники в тех случаях, когда возникала необходимость проведения через 3 заданные точки достаточно гладкой кривой. Известно, что рейка из упругого материала, закрепленная в некоторых точках и находящаяся в положении равновесия, принимает форму, при которой ее энергия является минимальной. Это фундаментальное свойство позволяет эффективно использовать сплайны при решении практических задач обработки экспериментальной информации.

Код полинома:

function Spline(x,y1,v,y1v,leg)

p = spline(x,y1,v);

figure; plot(x,y1,'--r', x,y1, 'bs', v, y1v, 'go',v,p,'m+'); grid;

title('Curve Fitting'); xlabel('x, s'); ylabel('Magnitude');

legend(leg, 'Узлы','Валидация', 'Полином');

S = std(y1v-p)

end

Код вывода функций:

clear; clc; close all; % Очистка памяти и экрана выводы

%===============================================================================

x = 0:0.5:20; %Узлы интерполирования

v = 0.25:0.5:20; %Точки валидации

f = 0.2; phi = -pi/7;

k = 0.5; b = -2; y1 = k\*x+b; leg = 'Прямая';

y1v = k\*v+b;

Spline(x,y1,v,y1v,leg);

f = 0.2; phi = -pi/7; y2 = sin(2\*pi\*x\*f+phi);

y2v = sin(2\*pi\*v\*f+phi);

leg = 'Гармоника';

Spline(x,y2,v,y2v,leg);

y3 = exp(x); leg = 'Экспоненциальная';

y3v = exp(v);

Spline(x,y3,v,y3v,leg);

y4 = tanh(2\*pi\*x\*f+phi); leg = 'Гиперболический тангенс';

y4v = tanh(2\*pi\*v\*f+phi);

Spline(x,y4,v,y4v,leg);

y5 = gauspuls(x,f); leg = 'Жордана-Гаусса';

y5v = gauspuls(v,f);

Spline(x,y5,v,y5v,leg);

y6 = sawtooth(2\*pi\*x\*f+phi); leg = 'Пилообразная функция';

y6v = sawtooth(2\*pi\*v\*f+phi);

Larg(x,y6,v,y6v,leg); ylim([-1 1]);

y7 = square(x); leg = 'меандр ';

y7v = square(v);

Spline(x,y7,v,y7v,leg);

k = 0.5; b = -2; y8 = x.\*x; leg = 'Квадратичная';

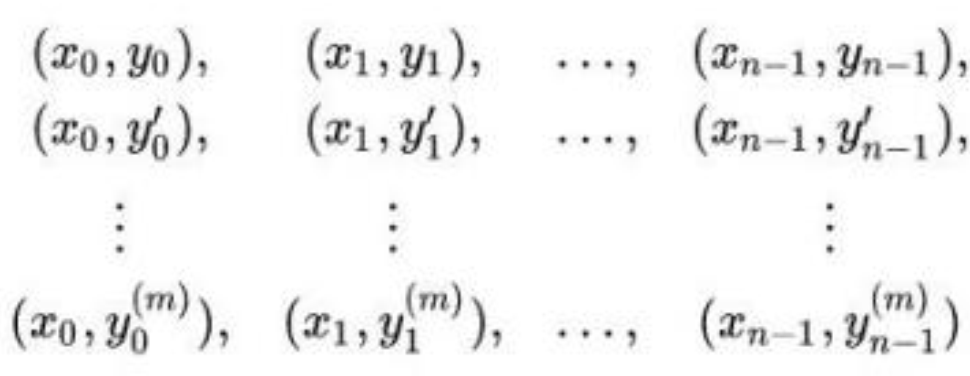
y8v = v.\*v;

Spline(x,y8,v,y8v,leg);

4) Сплайны Эрмита (pchip)

Эрмитова интерполяция – метод полиномиальной интерполяции, названный в честь французского математика Шарля Эрмита. Многочлены Эрмита тесно связаны с многочленами Ньютона.

В отличие от интерполяции Ньютона, Эрмитова интерполяция строит многочлен, значения которого в выбранных точках совпадают со значениями исходной функции в этих точках, и все производные многочлена вплоть до некоторого порядка m в данных точках совпадают со значениями производных функции. Это означает, что n(m + 1) величин должны быть известны, тогда как для ньютоновской интерполяции необходимы только первые n значений. Полученный многочлен может иметь степень не более, чем n(m + 1) − 1, максимальная степень многочлена Ньютона же равна n − 1. (В общем случае m не обязательно должно быть фиксировано, то есть в одних точках может быть известно значение большего количества производных, чем в других. В этом случае многочлен будет иметь степень N − 1, где N – число известных значений.)



Код полинома:

function Splinechip(x,y1,v,y1v,leg)

p = pchip(x,y1,v);

figure; plot(x,y1,'--r', x,y1, 'bs', v, y1v, 'go',v,p,'m+'); grid;

title('Curve Fitting'); xlabel('x, s'); ylabel('Magnitude');

legend(leg, 'Узлы','Валидация', 'Полином');

S = std(y1v-p)

end

Код вывода функций:

clear; clc; close all; % Очистка памяти и экрана выводы

%===============================================================================

x = 0:0.5:20; %Узлы интерполирования

v = 0.25:0.5:20; %Точки валидации

f = 0.2; phi = -pi/7;

k = 0.5; b = -2; y1 = k\*x+b; leg = 'Прямая';

y1v = k\*v+b;

Splinechip(x,y1,v,y1v,leg);

f = 0.2; phi = -pi/7; y2 = sin(2\*pi\*x\*f+phi);

y2v = sin(2\*pi\*v\*f+phi);

leg = 'Гармоника';

Splinechip(x,y2,v,y2v,leg);

y3 = exp(x); leg = 'Экспоненциальная';

y3v = exp(v);

Splinechip(x,y3,v,y3v,leg);

y4 = tanh(2\*pi\*x\*f+phi); leg = 'Гиперболический тангенс';

y4v = tanh(2\*pi\*v\*f+phi);

Splinechip(x,y4,v,y4v,leg);

y5 = gauspuls(x,f); leg = 'Жордана-Гаусса';

y5v = gauspuls(v,f);

Splinechip(x,y5,v,y5v,leg);

y6 = sawtooth(2\*pi\*x\*f+phi); leg = 'Пилообразная функция';

y6v = sawtooth(2\*pi\*v\*f+phi);

Splinechip(x,y6,v,y6v,leg);

y7 = square(x); leg = 'меандр ';

y7v = square(v);

Splinechip(x,y6,v,y6v,leg);

k = 0.5; b = -2; y8 = x.\*x; leg = 'Квадратичная';

y8v = v.\*v;

Splinechip(x,y8,v,y8v,leg);

5) Нейронные сети прямого распространения (feedforward)

Базовыми архитектурами нейронных сетей являются сети прямого распространения. Данные нейронные сети очень прямолинейны. Информация по ним передается от входа сразу к выходу. Клетки слоя данных сетей не связаны между собой, в отличии от соседних слоев, которые обычно полностью связаны.

Нейронные сети прямого распространения обычно обучаются по методу обратного распространения ошибки. При таком обучении нейронная сеть получает множество данных, как на вход, так и на выход. Этот процесс называется обучением с учителем.

clear; clc; close all; % Очистка памяти и экрана выводы

%====================================================================================

x = 0:0.5:20; %Узлы интерполирования

v = 0.25:0.5:20; %Точки валидации

f = 0.2; phi = -pi/7;

k = 0.5; b = -2; y1 = k\*x+b; leg = 'Прямая';

y1v = k\*v+b;

net = feedforwardnet(9);

net = train(net,x,y1);

view(net)

p2 = net(v);

perf = perform(net,p2,y1v)

s = std(y1v-p2)

plot(v, y1v, 'b', v, p2, 'r'); grid; legend(leg, 'nnet');

y2 = sin(2\*pi\*x\*f+phi);

y2v = sin(2\*pi\*v\*f+phi);

leg = 'Гармоника';

net = feedforwardnet(9);

net = train(net,x,y2);

view(net)

p2 = net(v);

perf = perform(net,p2,y2v)

s = std(y2v-p2)

plot(v, y2v, 'b', v, p2, 'r'); grid; legend(leg, 'nnet');

net = newrb(x,y2,0,8);

view(net)

p2 = net(v);

s = std(y2v-p2)

plot(v, y2v, 'b', v, p2, 'r'); grid; legend(leg, 'nnet');

net = newgrnn(x,y2,0.1);

view(net)

p2 = net(v);

s = std(y2v-p2)

plot(v, y2v, 'b', v, p2, 'r'); grid; legend(leg, 'nnet');

y3 = exp(x); leg = 'Экспоненциальная';

y3v = exp(v);

net = feedforwardnet(15);

net = train(net,x,y3);

view(net)

p2 = net(v);

perf = perform(net,p2,y3v)

s = std(y3v-p2)

plot(v, y3v, 'b', v, p2, 'r'); grid; legend(leg, 'nnet');

y4 = tanh(2\*pi\*x\*f+phi); leg = 'Гиперболический тангенс';

y4v = tanh(2\*pi\*v\*f+phi);

net = feedforwardnet(9);

net = train(net,x,y4);

view(net)

p2 = net(v);

perf = perform(net,p2,y4v)

s = std(y4v-p2)

plot(v, y4v, 'b', v, p2, 'r'); grid; legend(leg, 'nnet');

y5 = gauspuls(x,f); leg = 'Жордана-Гаусса';

y5v = gauspuls(v,f);

net = feedforwardnet(9);

net = train(net,x,y5);

view(net)

p2 = net(v);

perf = perform(net,p2,y5v)

s = std(y5v-p2)

plot(v, y5v, 'b', v, p2, 'r'); grid; legend(leg, 'nnet');

y6 = sawtooth(2\*pi\*x\*f+phi); leg = 'Пилообразная функция';

y6v = sawtooth(2\*pi\*v\*f+phi);

net = feedforwardnet(27);

net = train(net,x,y6);

view(net)

p2 = net(v);

perf = perform(net,p2,y6v)

s = std(y6v-p2)

plot(v, y6v, 'b', v, p2, 'r'); grid; legend(leg, 'nnet');

y7 = square(x); leg = 'меандр ';

y7v = square(v);

net = feedforwardnet(18);

net = train(net,x,y7);

view(net)

p2 = net(v);

perf = perform(net,p2,y7v)

s = std(y7v-p2)

plot(v, y7v, 'b', v, p2, 'r'); grid; legend(leg, 'nnet');

k = 0.5; b = -2; y8 = x.\*x; leg = 'Квадратичная';

y8v = v.\*v;

net = feedforwardnet(9);

net = train(net,x,y8);

view(net)

p2 = net(v);

perf = perform(net,p2,y8v)

s = std(y8v-p2)

plot(v, y8v, 'b', v, p2, 'r'); grid; legend(leg, 'nnet');

6) Нейронные сети радиальных базисных функций (rbf)

Сети РБФ имеют ряд преимуществ. Во-первых, они моделируют произвольную нелинейную функцию с помощью всего одного промежуточного слоя, тем самым избавляя разработчика от необходимости решать вопрос о числе слоев. Во-вторых, параметры линейной комбинации в выходном слое можно полностью оптимизировать с помощью хорошо известных методов линейной оптимизации, которые работают быстро и не испытывают трудностей с локальными минимумами, так мешающими при обучении с использованием алгоритма обратного распространения ошибки. Поэтому сеть РБФ обучается очень быстро. Недостатки сетей РБФ: данные сети обладают плохими экстраполирующими свойствами и получаются весьма громоздкими при большой размерности вектора входов.

Код функций:

clear; clc; close all; % Очистка памяти и экрана выводы

%====================================================================================

x = 0:0.5:20; %Узлы интерполирования

v = 0.25:0.5:20; %Точки валидации

f = 0.2; phi = -pi/7;

k = 0.5; b = -2; y1 = k\*x+b; leg = 'Прямая';

y1v = k\*v+b;

net = newrb(x,y1,0,8);

view(net)

p2 = net(v);

s = std(y1v-p2)

plot(v, y1v, 'b', v, p2, 'r'); grid; legend(leg, 'nnet');

y2 = sin(2\*pi\*x\*f+phi);

y2v = sin(2\*pi\*v\*f+phi);

leg = 'Гармоника';

net = newrb(x,y2,0,8);

view(net)

p2 = net(v);

s = std(y2v-p2)

plot(v, y2v, 'b', v, p2, 'r'); grid; legend(leg, 'nnet');

net = newgrnn(x,y2,0.1);

view(net)

p2 = net(v);

s = std(y2v-p2)

plot(v, y2v, 'b', v, p2, 'r'); grid; legend(leg, 'nnet');

y3 = exp(x); leg = 'Экспоненциальная';

y3v = exp(v);

net = newrb(x,y3,0,8);

view(net)

p2 = net(v);

s = std(y3v-p2)

plot(v, y3v, 'b', v, p2, 'r'); grid; legend(leg, 'nnet');

y4 = tanh(2\*pi\*x\*f+phi); leg = 'Гиперболический тангенс';

y4v = tanh(2\*pi\*v\*f+phi);

net = newrb(x,y4,0,8);

view(net)

p2 = net(v);

s = std(y4v-p2)

plot(v, y4v, 'b', v, p2, 'r'); grid; legend(leg, 'nnet');

y5 = gauspuls(x,f); leg = 'Жордана-Гаусса';

y5v = gauspuls(v,f);

net = newrb(x,y5,0,8);

view(net)

p2 = net(v);

s = std(y5v-p2)

plot(v, y5v, 'b', v, p2, 'r'); grid; legend(leg, 'nnet');

y6 = sawtooth(2\*pi\*x\*f+phi); leg = 'Пилообразная функция';

y6v = sawtooth(2\*pi\*v\*f+phi);

net = newrb(x,y6,0,8);

view(net)

p2 = net(v);

s = std(y6v-p2)

plot(v, y6v, 'b', v, p2, 'r'); grid; legend(leg, 'nnet');

y7 = square(x); leg = 'меандр ';

y7v = square(v);

net = newrb(x,y7,0,8);

view(net)

p2 = net(v);

s = std(y7v-p2)

plot(v, y7v, 'b', v, p2, 'r'); grid; legend(leg, 'nnet');

k = 0.5; b = -2; y8 = x.\*x; leg = 'Квадратичная';

y8v = v.\*v;

net = newrb(x,y8,0,8);

view(net)

p2 = net(v);

s = std(y8v-p2)

plot(v, y8v, 'b', v, p2, 'r'); grid; legend(leg, 'nnet');

7) Нейронные сети обобщенных регрессионных функций (grnn).

Обобщенно-регрессионная нейронная сеть (GRNN) предназначена для решения задач регрессии. В точку расположения каждого обучающего наблюдения помещается гауссова ядерная функция. Считается, что каждое наблюдение свидетельствует о некоторой уверенности в том, что поверхность отклика в данной точке имеет определённую высоту, и эта уверенность убывает при отходе в сторону от точки. GRNN-сеть копирует внутрь себя все обучающие наблюдения и использует их для оценки отклика в произвольной точке. Окончательная выходная оценка сети получается как взвешенное среднее выходов по всем обучающим наблюдениям, где величины весов отражают расстояние от этих наблюдений до той точки, в которой производится оценивание (и, таким образом, более близкие точки вносят больший вклад в оценку).

Код функций:

clear; clc; close all; % Очистка памяти и экрана выводы

%==================================================================================

x = 0:0.5:20; %Узлы интерполирования

v = 0.25:0.5:20; %Точки валидации

f = 0.2; phi = -pi/7;

k = 0.5; b = -2; y1 = k\*x+b; leg = 'Прямая';

y1v = k\*v+b;

net = newgrnn(x,y1,0.1);

view(net)

p2 = net(v);

s = std(y1v-p2)

plot(v, y1v, 'b', v, p2, 'r'); grid; legend(leg, 'nnet');

y2 = sin(2\*pi\*x\*f+phi);

y2v = sin(2\*pi\*v\*f+phi);

leg = 'Гармоника';

net = newgrnn(x,y2,0.1);

view(net)

p2 = net(v);

s = std(y2v-p2)

plot(v, y2v, 'b', v, p2, 'r'); grid; legend(leg, 'nnet');

y3 = exp(x); leg = 'Экспоненциальная';

y3v = exp(v);

net = newgrnn(x,y3,0.1);

view(net)

p2 = net(v);

s = std(y3v-p2)

plot(v, y3v, 'b', v, p2, 'r'); grid; legend(leg, 'nnet');

y4 = tanh(2\*pi\*x\*f+phi); leg = 'Гиперболический тангенс';

y4v = tanh(2\*pi\*v\*f+phi);

net = newgrnn(x,y4,0.1);

view(net)

p2 = net(v);

s = std(y4v-p2)

plot(v, y4v, 'b', v, p2, 'r'); grid; legend(leg, 'nnet');

y5 = gauspuls(x,f); leg = 'Жордана-Гаусса';

y5v = gauspuls(v,f);

net = newgrnn(x,y5,0.1);

view(net)

p2 = net(v);

s = std(y5v-p2)

plot(v, y5v, 'b', v, p2, 'r'); grid; legend(leg, 'nnet');

y6 = sawtooth(2\*pi\*x\*f+phi); leg = 'Пилообразная функция';

y6v = sawtooth(2\*pi\*v\*f+phi);

net = newgrnn(x,y6,0.1);

view(net)

p2 = net(v);

s = std(y6v-p2)

plot(v, y6v, 'b', v, p2, 'r'); grid; legend(leg, 'nnet');

y7 = square(x); leg = 'меандр ';

y7v = square(v);

net = newgrnn(x,y7,0.1);

view(net)

p2 = net(v);

s = std(y7v-p2)

plot(v, y7v, 'b', v, p2, 'r'); grid; legend(leg, 'nnet');

k = 0.5; b = -2; y8 = x.\*x; leg = 'Квадратичная';

y8v = v.\*v;

net = newgrnn(x,y8,0.1);

view(net)

p2 = net(v);

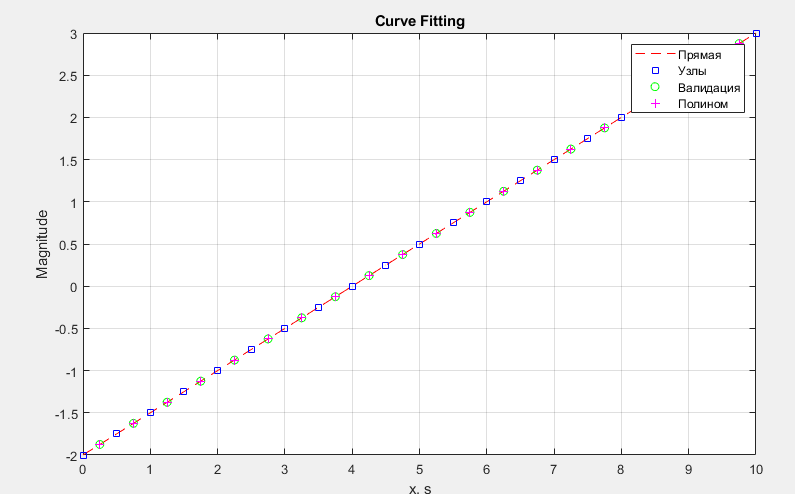
s = std(y8v-p2)

plot(v, y8v, 'b', v, p2, 'r'); grid; legend(leg, 'nnet');

# ИНТЕРПОЛИРОВАНИЕ ФУНКЦИЙ

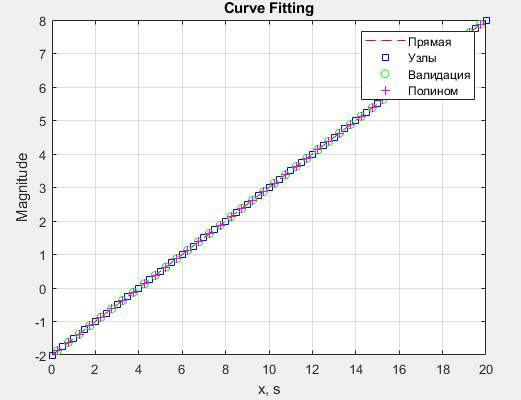
1. Прямая
   1. Линейная функция с интерпеллянтом на основе полинома Лагранжа

СКО: 8,84\*10-13



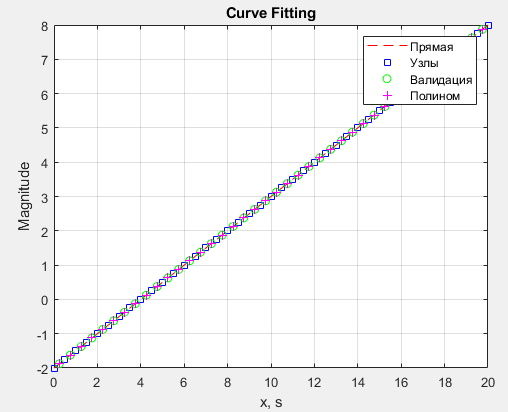
* 1. Линейная функция с интерпеллянтом на основе полинома n-ой степени.

СКО: 9.73\*10-16



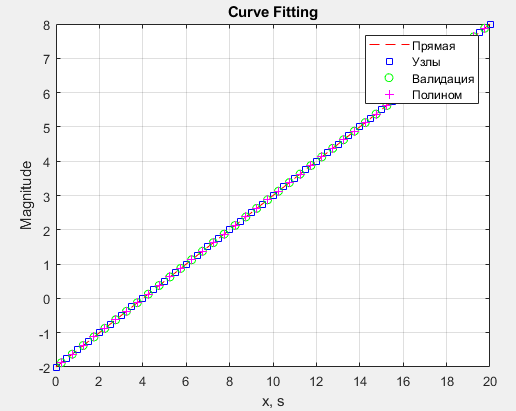
* 1. Линейная функция с интерпеллянтом на основе кубических сплайнов.

СКО: 1.6⋅10-16



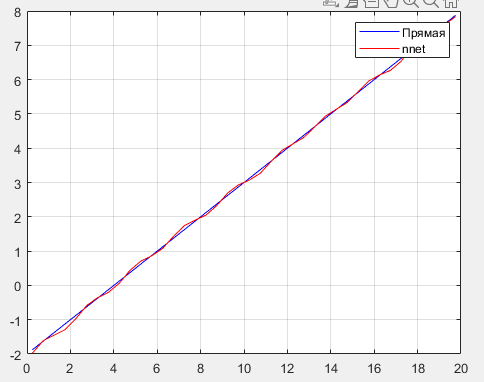
* 1. Линейная функция с интерполяцией Эрмита.

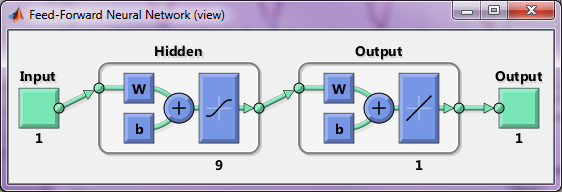
СКО: 1.6⋅10-16



* 1. Линейная функция с интерполяцией нейронных сетей прямого распространения.

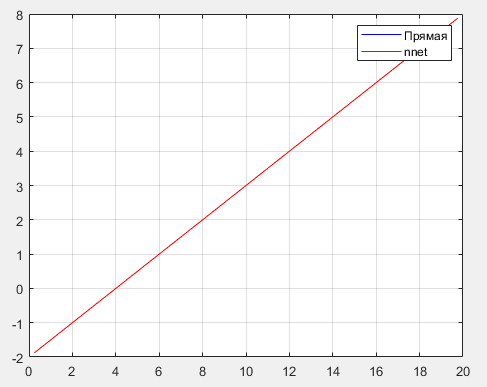
СКО: 0.0060

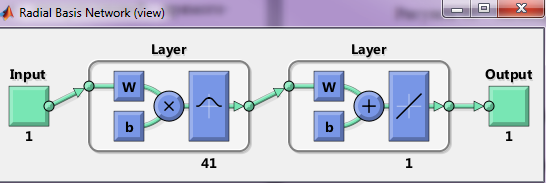




* 1. Линейная функция с интерполяцией радиальных базисных функций.

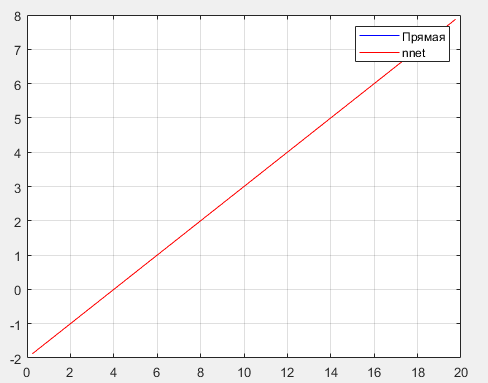
СКО: 5.07\*10-5

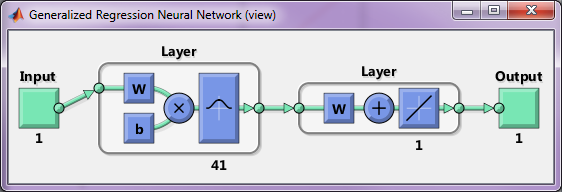




* 1. Линейная функция с интерполяцией обобщенных регрессионных функций.

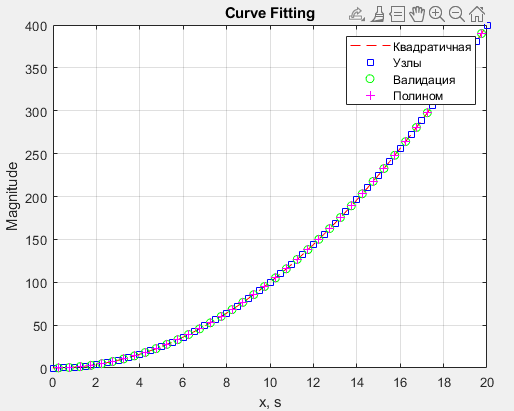
СКО: 3.51\*10-17





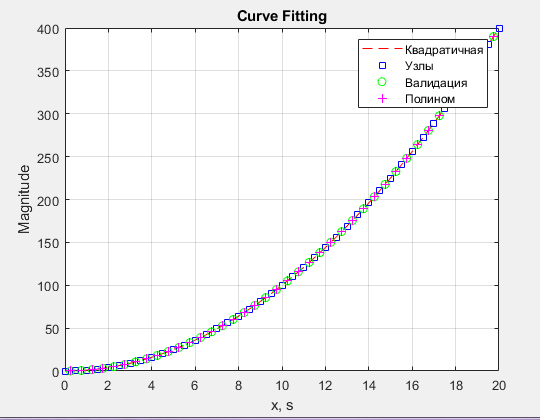
2) Квадратичная функция

2.1) Квадратичная функция с интерпеллянтом на основе полинома Лагранжа. СКО: 2.46\*10-6

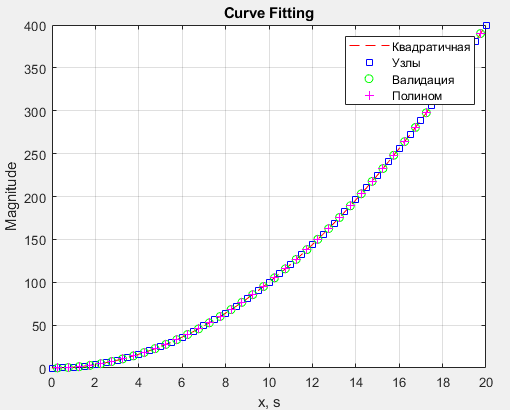


2.2) Квадратичная функция с интерпеллянтом на основе полинома n-ой степени.

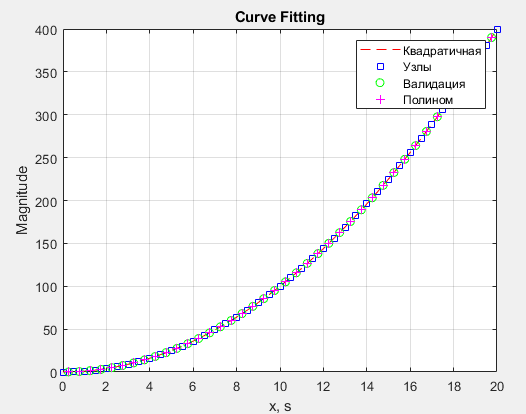
СКО: 4.18\*10-14



2.3) Квадратичная функция с интерпеллянтом на основе кубических сплайнов. СКО:0,00

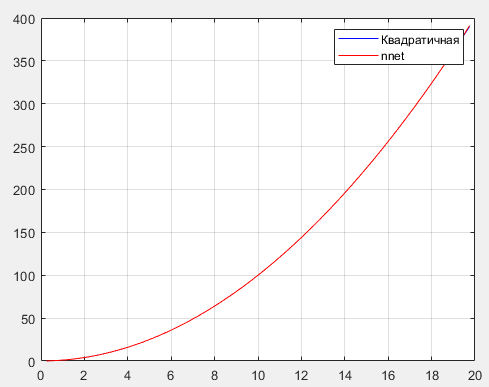


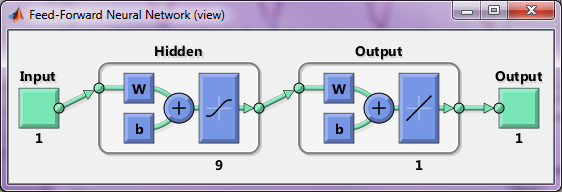
2.4) Квадратичная функция с интерполяцией Эрмита. СКО: 0.0028



2.5) Квадратичная функция с интерполяцией нейронных сетей прямого распространения.

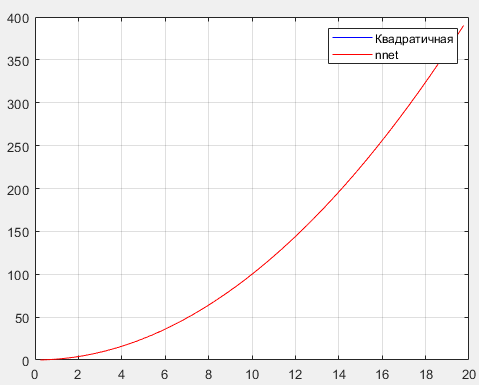
СКО: 0.035

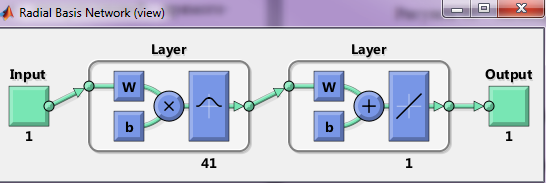




2.6) Квадратичная функция с интерполяцией радиальных базисных функций.

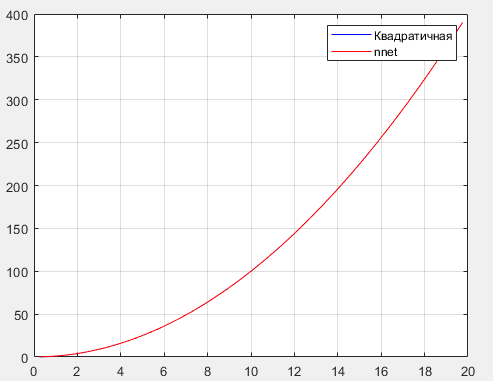
СКО: 2\*10-6

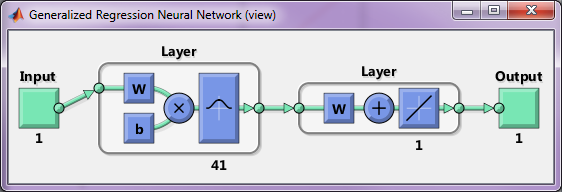




2.7) Квадратичная функция с интерполяцией обобщенных регрессионных функций.

СКО: 2.32\*10-16

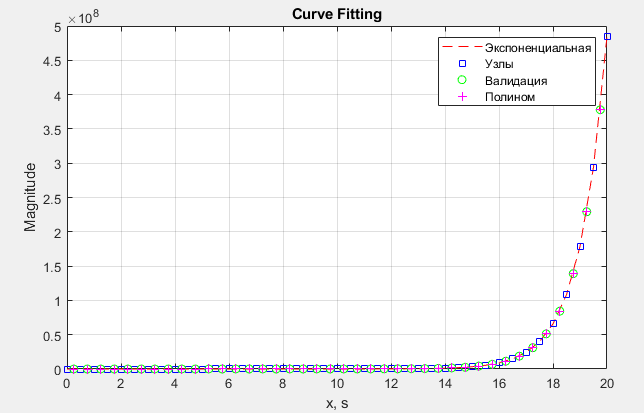




3) Экспоненциальная функция

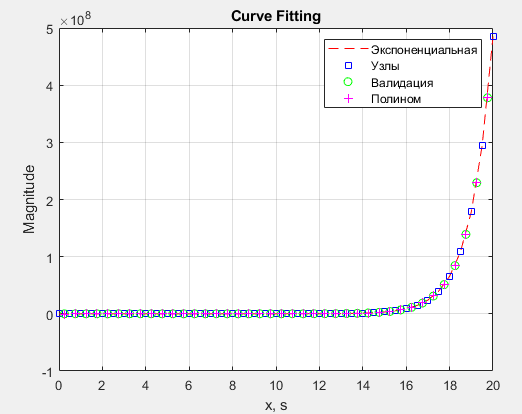
3.1) Экспоненциальная функция с интерпеллянтом на основе полинома Лагранжа.

СКО: 0.0048



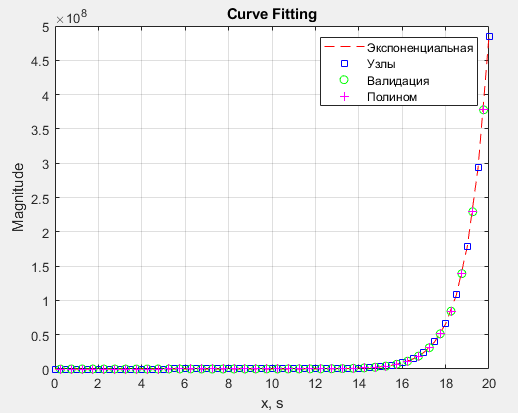
3.2) Экспоненциальная функция с интерпеллянтом на основе полинома n-ой степени.

СКО: 2.79\*10-5



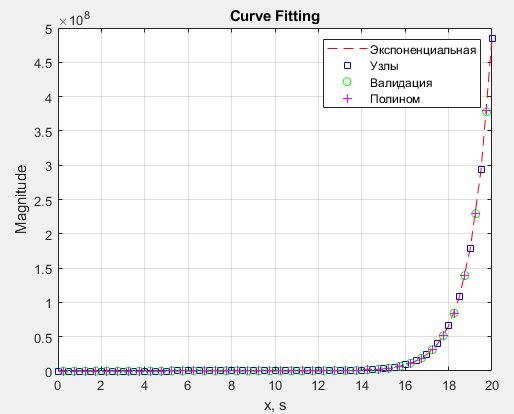
3.3) Экспоненциальная функция с интерпеллянтом на основе кубических сплайнов.

СКО: 7.95\*10-4



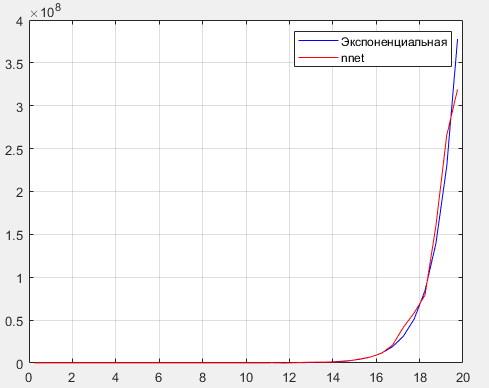
3.4) Экспоненциальная функция с интерполяцией Эрмита.

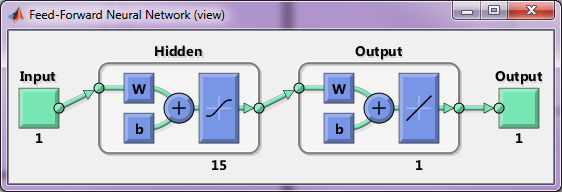
СКО: 2.102\*10-9



3.5) Экспоненциальная функция с интерполяцией нейронных сетей прямого распространения.

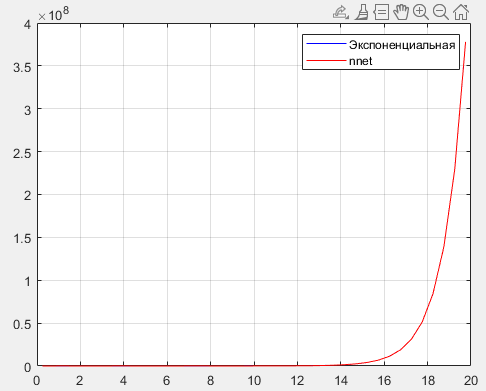
СКО: 1.34\*10-14

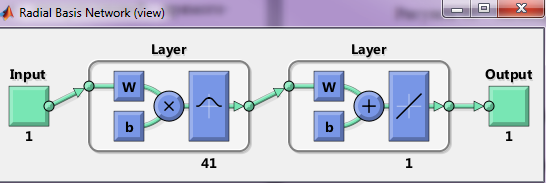




3.6) Экспоненциальная функция с интерполяцией радиальных базисных функций.

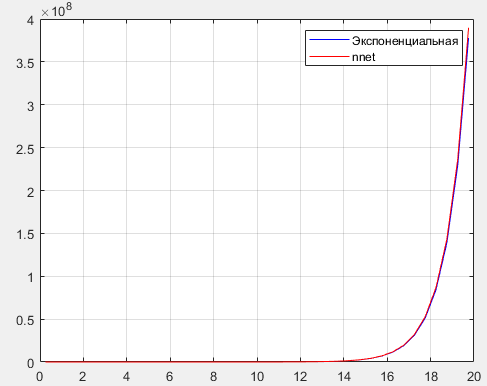
СКО: 2.83\*10-4

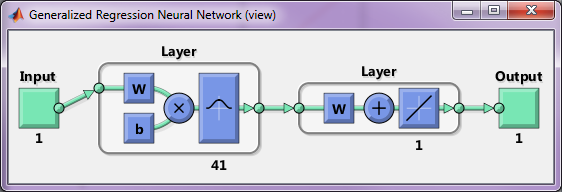




3.7) Экспоненциальная функция с интерполяцией обобщенных регрессионных функций.

СКО: 2.26\*10-6

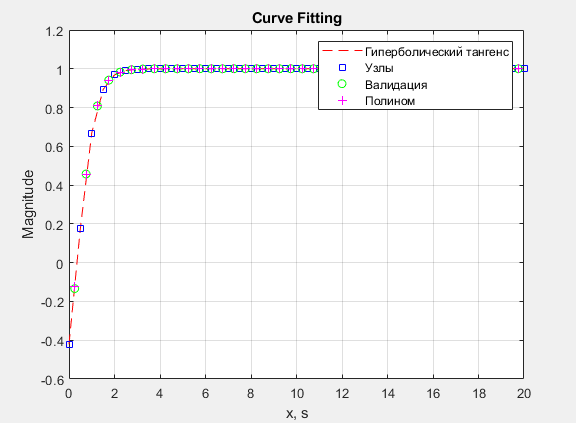




4) Функция гиперболического тангенса

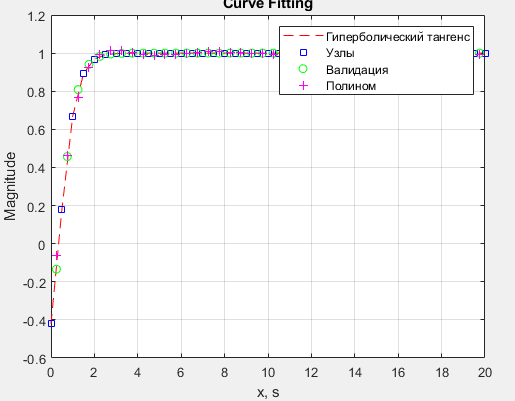
4.1) Функция гиперболического тангенса с интерпеллянтом на основе полинома Лагранжа.

СКО: 0.0016

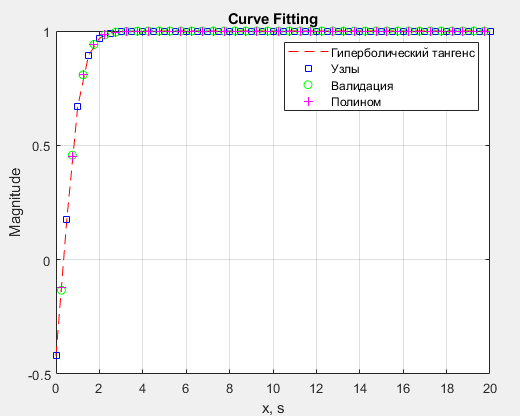


4.2) Функция гиперболического тангенса с интерпеллянтом на основе полинома n-ой степени.

СКО: 0.0146

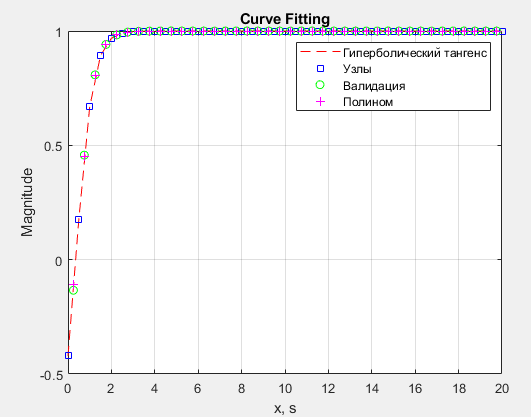


4.3) Функция гиперболического тангенса с интерпеллянтом на основе кубических сплайнов. СКО: 0.0020



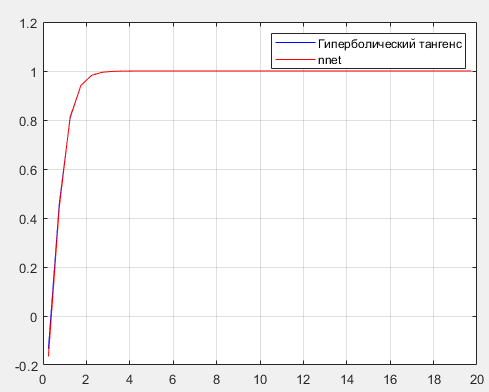
4.4) Функция гиперболического тангенса с интерполяцией Эрмита.

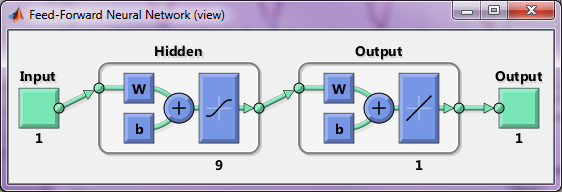
СКО: 0.0043



4.5) Функция гиперболического тангенса с интерполяцией нейронных сетей прямого распространения.

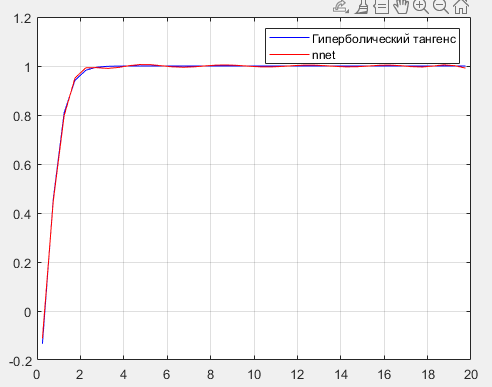
СКО: 2.96\*10-5

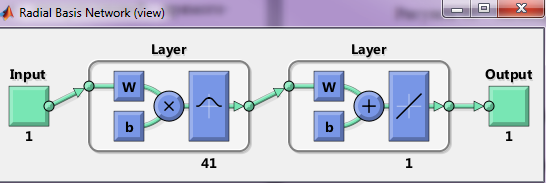




4.6) Функция гиперболического тангенса с интерполяцией радиальных базисных функций.

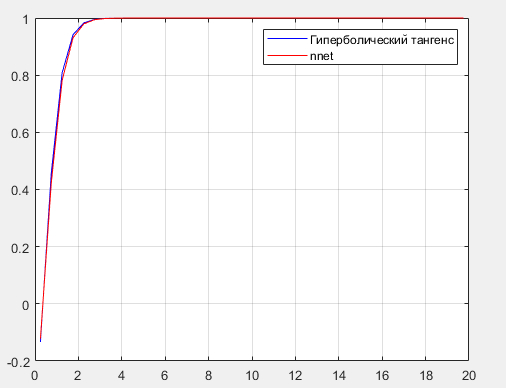
СКО: 0.0059

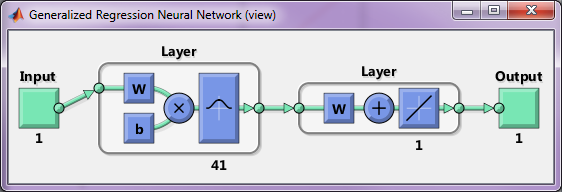




4.7) Функция гиперболического тангенса с интерполяцией обобщенных регрессионных функций.

СКО: 0.0073

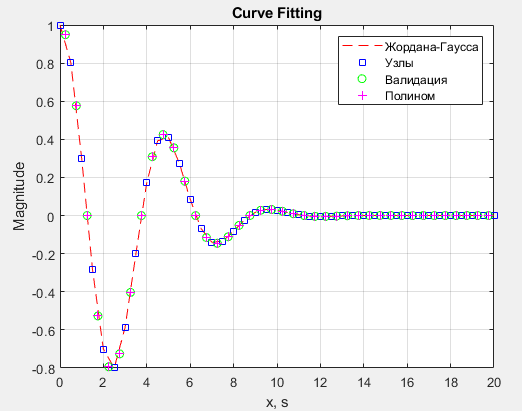




5) Гауссова функция

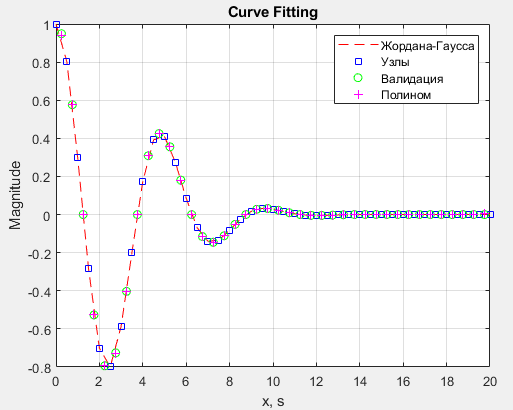
5.1) Гауссова функция с интерпеллянтом на основе полинома Лагранжа.

СКО: 2.876\*10-7



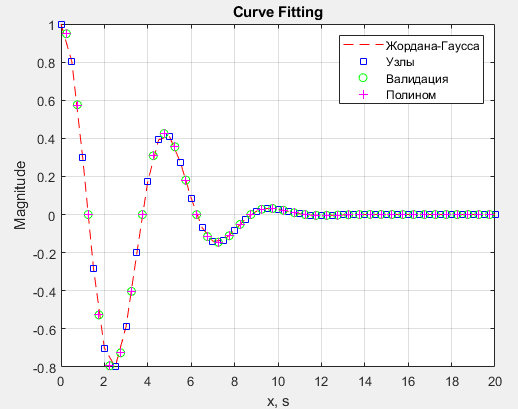
5.2) Гауссова функция с интерпеллянтом на основе полинома n-ой степени.

СКО: 0.0020

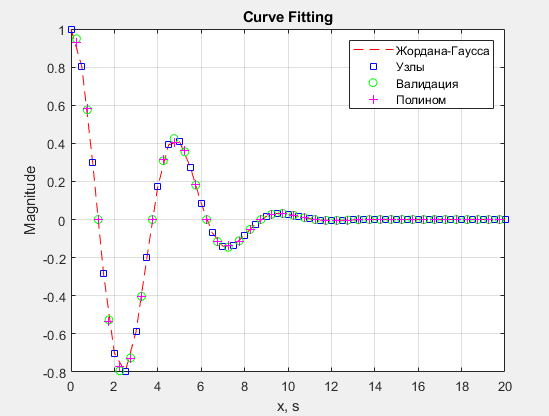


5.3) Гауссова функция с интерпеллянтом на основе кубических сплайнов.

СКО: 6.38\*10-4

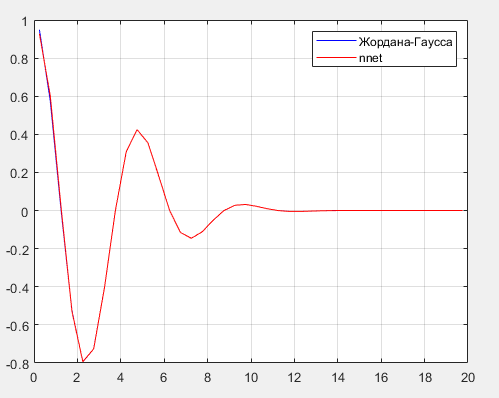


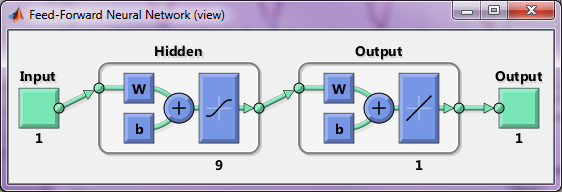
5.4) Гауссова функция с интерполяцией Эрмита. СКО: 0.0062



5.5) Гауссова функция с интерполяцией нейронных сетей прямого распространения.

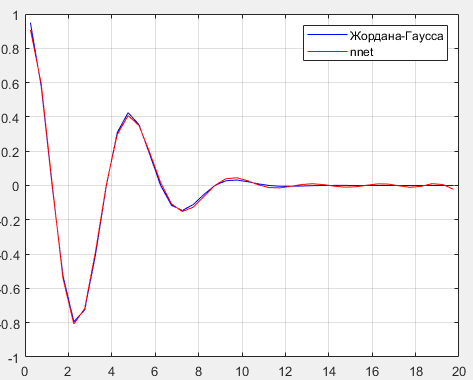
СКО: 3.26\*10-5

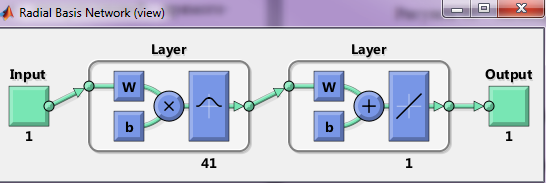




5.6) Гауссова функция с интерполяцией радиальных базисных функций.

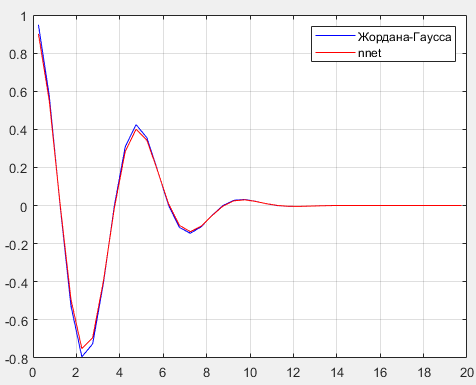
СКО: 0.0126

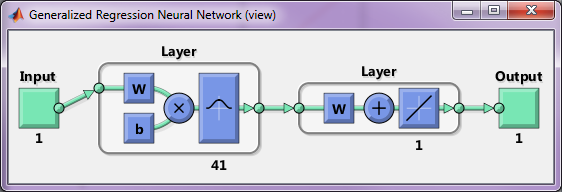




5.7) Гауссова функция с интерполяцией прямого распространения.

СКО: 0.0152

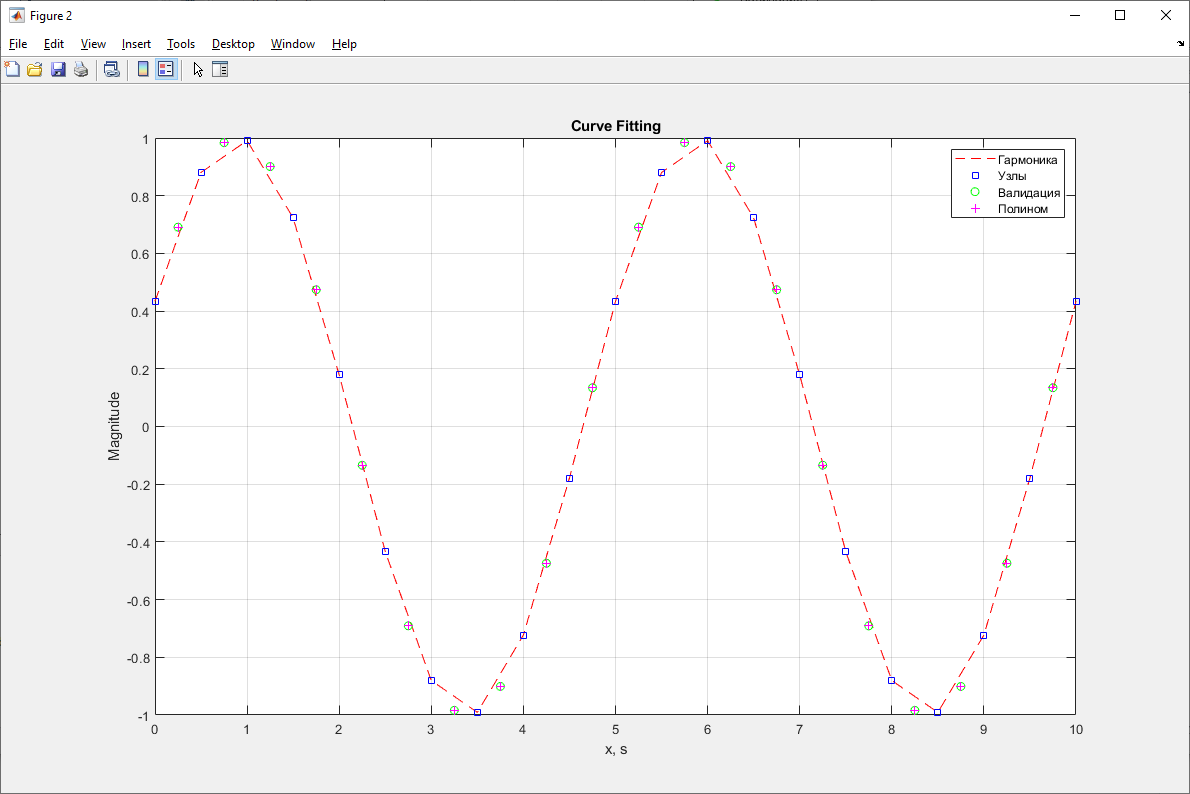




**6) Гармоническая функция**

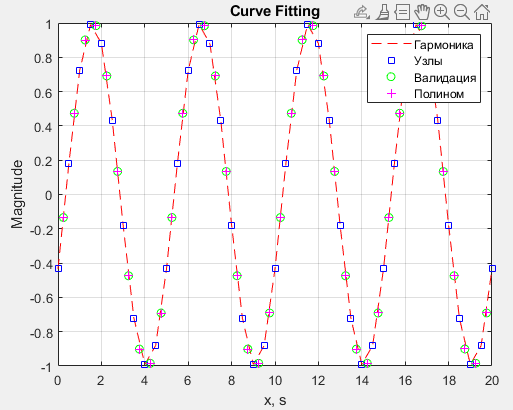
6.1) Гармоническая функция с интерпеллянтом на основе полинома Лагранжа.

СКО: 3,5\*10-8



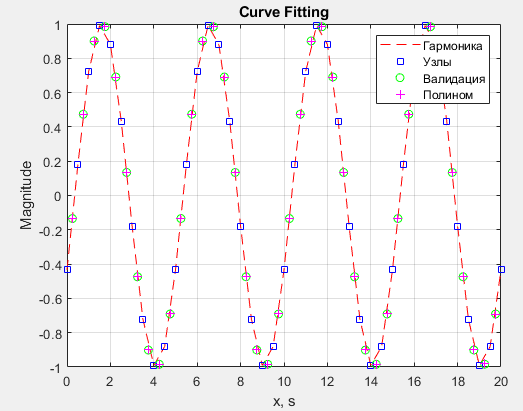
6.2) Гармоническая функция с интерпеллянтом на основе полинома n-ой степени.

СКО: 6.2\*10-4



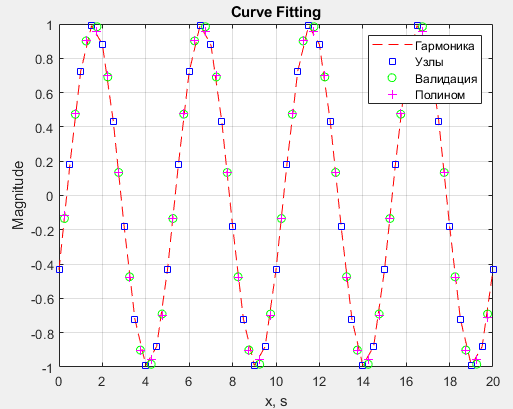
6.3) Гармоническая функция с интерпеллянтом на основе кубических сплайнов.

СКО: 7.055\*10-4



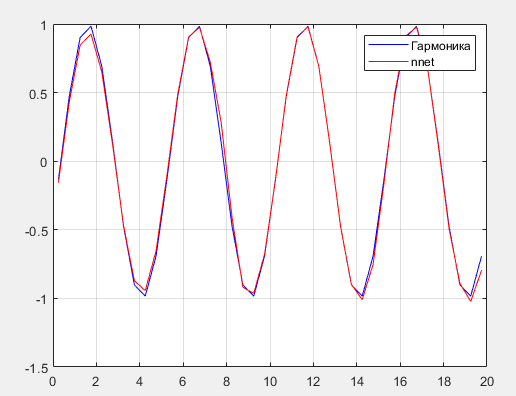
6.4) Гармоническая функция с интерполяцией Эрмита.

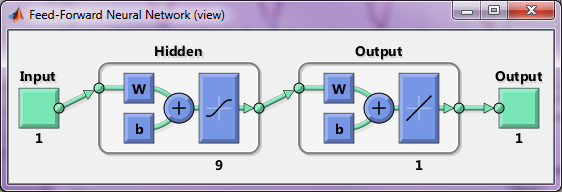
СКО: 0.0132



6.5) Гармоническая функция с интерполяцией нейронных сетей прямого распространения.

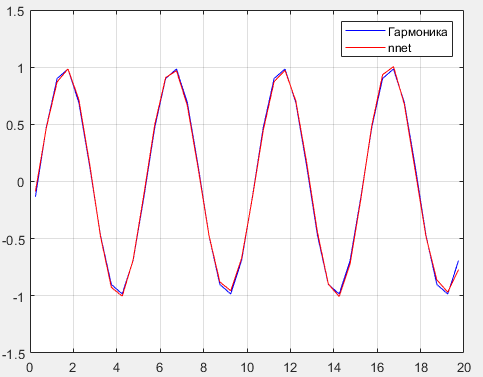
СКО: 0.0016

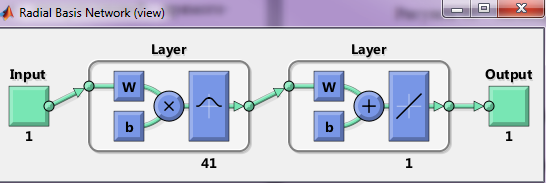




6.6) Гармоническая функция с интерполяцией радиальных базисных функций.

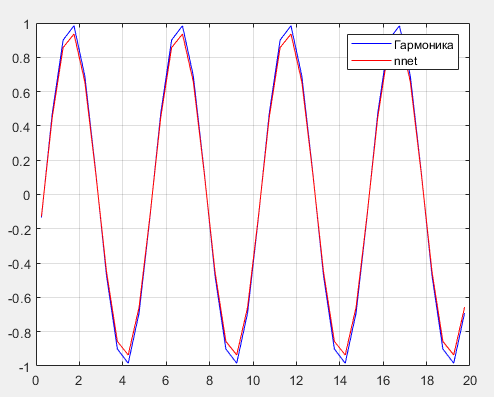
СКО: 0.025

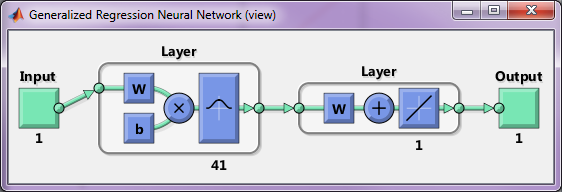




6.7) Гармоническая функция с интерполяцией прямого распространения.

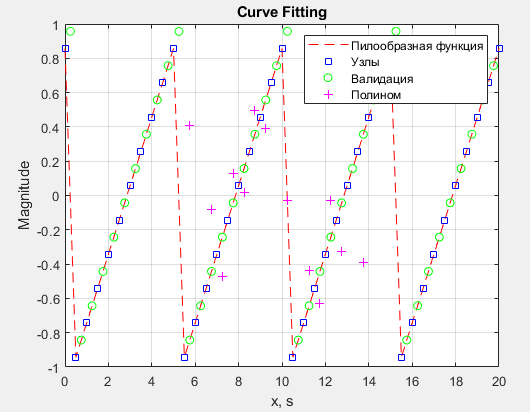
СКО: 0.0350





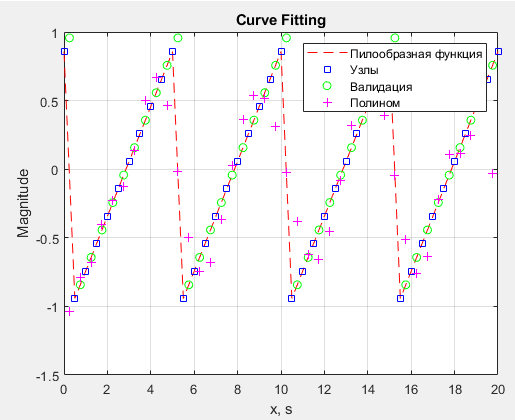
7) Пилообразная функция

7.1) Пилообразная функция с интерпеллянтом на основе полинома Лагранжа. СКО: 7.07\*10-7



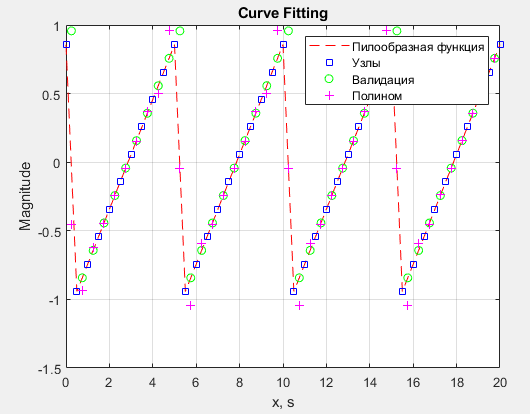
7.2) Пилообразная функция с интерпеллянтом на основе полинома n-ой степени.

СКО: 0.4569



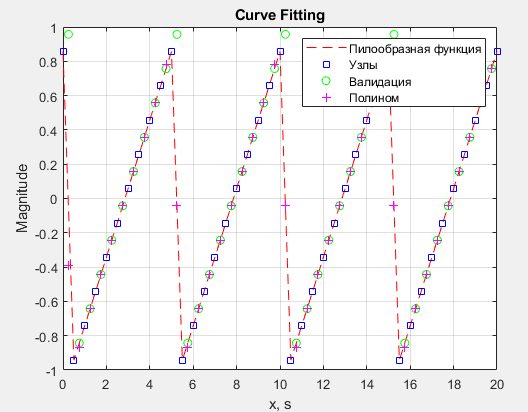
7.3) Пилообразная функция с интерпеллянтом на основе кубических сплайнов.

СКО: 0.3491



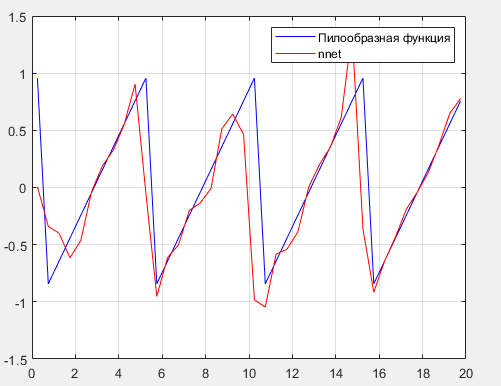
7.4) Пилообразная функция с интерполяцией Эрмита.

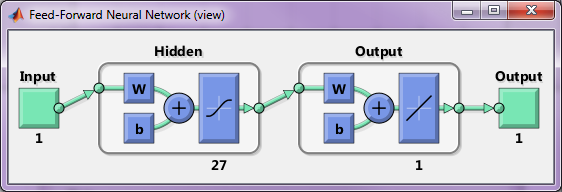
СКО: 0.3339



7.5) Пилообразная функция с интерполяцией нейронных сетей прямого распространения.

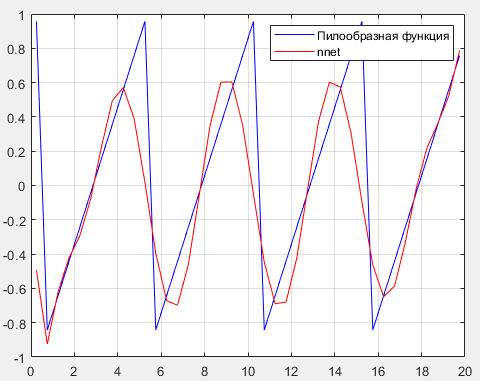
СКО: 0.212

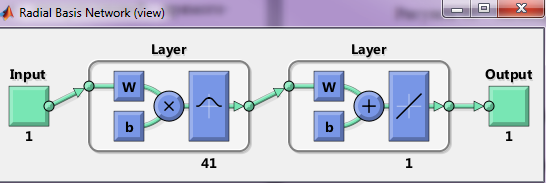




7.6) Пилообразная функция с интерполяцией радиальных базисных функций.

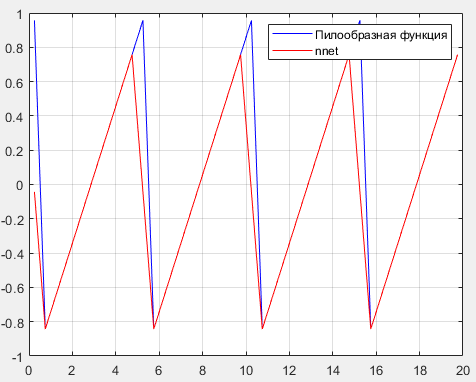
СКО: 0.395

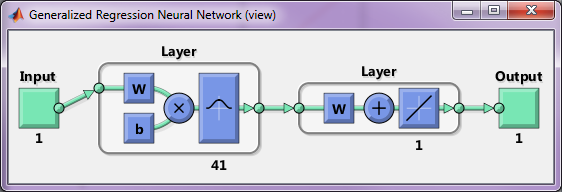




7.7) Пилообразная функция с интерполяцией прямого распространения.

СКО: 0.3038

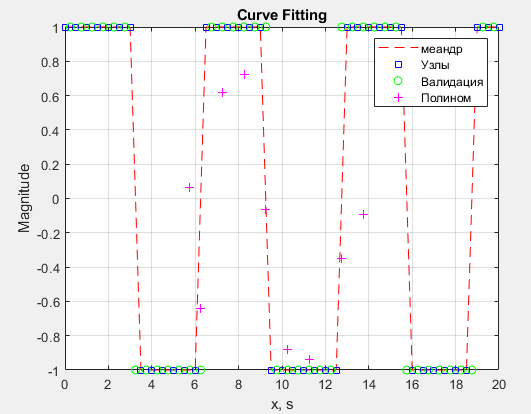




8) Меандр функция

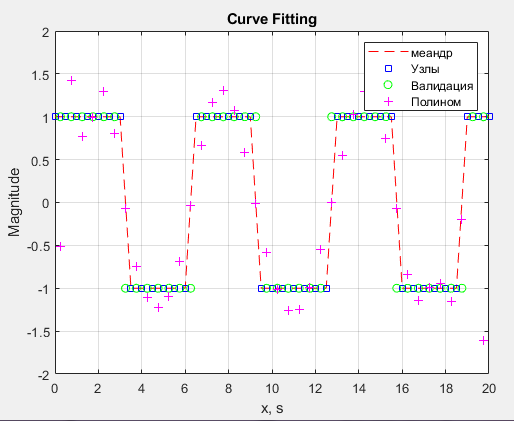
8.1) Меандр функция с интерпеллянтом на основе полинома Лагранжа.

СКО: 7.51\*10-7



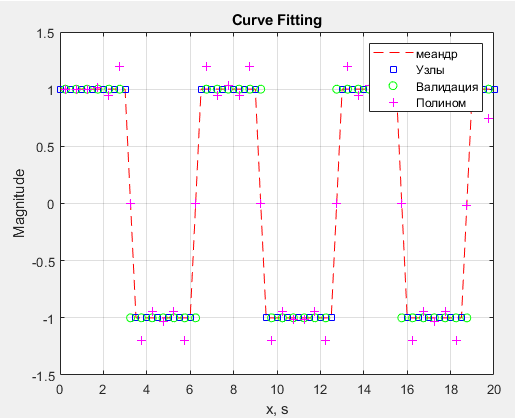
8.2) Меандр функция с интерпеллянтом на основе полинома n-ой степени.

СКО: 0.6611



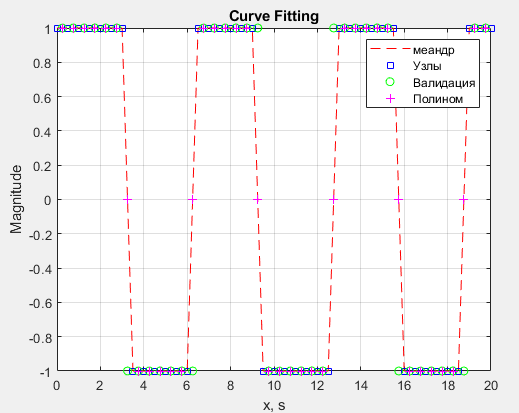
8.3) Меандр функция с интерпеллянтом на основе кубических сплайнов.

СКО: 0.4084



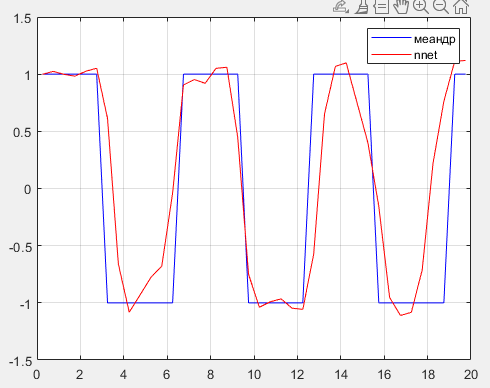
8.4) Меандр функция с интерполяцией Эрмита.

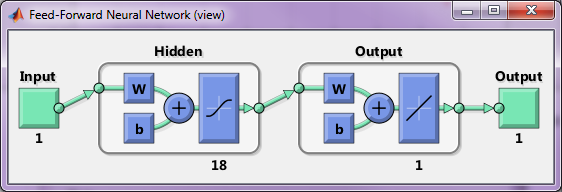
СКО: 0.3889



8.5) Меандр функция с интерполяцией нейронных сетей прямого распространения.

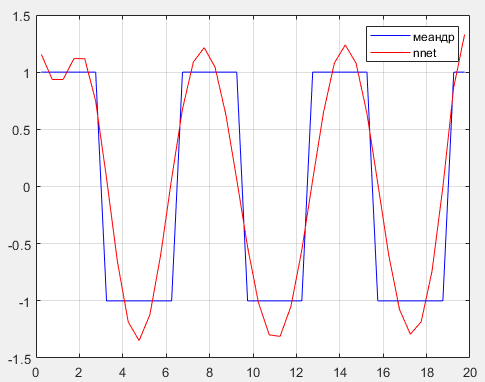
СКО: 0.317

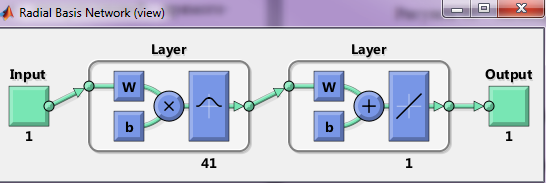




8.6) Меандр функция с интерполяцией радиальных базисных функций.

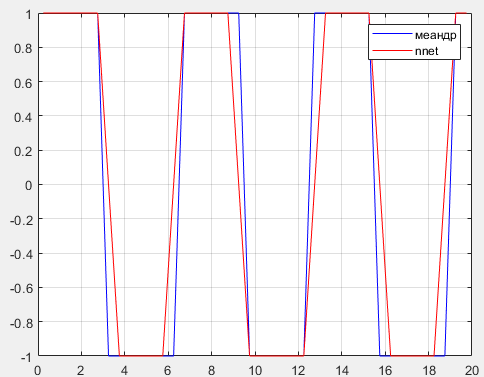
СКО: 0.4605

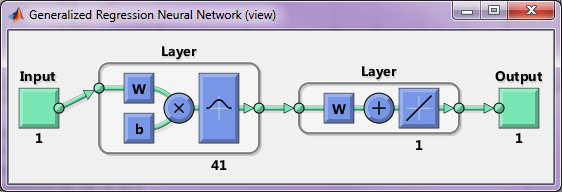




8.7) Меандр функция с интерполяцией прямого распространения.

СКО: 0.3889





# ИТОГОВАЯ ТАБЛИЦА СКО

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | полином Лагранжа | polyfit | spline | pchip | feedforwar | rbf | grnn |
| линейная | 8.84 \*10-8 | 9.73\*10-16 | 1.6⋅10-16 | 1.6⋅10-16 | 0.0060 | 5.07\*10-8 | 3.51\*10-17 |
| квадратичная | 2.46\*10-6 | 4.18\*10-14 | 0,00 | 0.0028 | 0.035 | 2\*10-6 | 2.32\*10-16 |
| экспоненциальная | 0.0048 | 2.79\*10-5 | 7.95\*10-4 | 2.102\*10-9 | 1.34\*10-14 | 2.83\*10-4 | 2.26\*10-6 |
| гиперболического тангенса | 0.0016 | 0.0146 | 0.0020 | 0.0043 | 2.96\*10-5 | 0.0059 | 0.0073 |
| гауссова | 2.876\*10-7 | 0.0020 | 6.38\*10-4 | 0.0062 | 3.26\*10-5 | 0.0126 | 0.0152 |
| гармоническая | 3,5\*10-8 | 6.2\*10-4 | 7.06\*10-4 | 0.0132 | 0.0016 | 0.025 | 0.0350 |
| пилообразная | 7.07\*10-7 | 0.4569 | 0.3491 | 0.3339 | 0.212 | 0.395 | 0.3038 |
| меандр | 7.51\*10-7 | 0.6611 | 0.4084 | 0.3889 | 0.317 | 0.4605 | 0.3889 |

# ВЫВОД

Таким образом, наиболее точными методами интерполяции можно считать метод кубического сплайна и полинома *n* степени.

Малое СКО по сравнению с другими нейронными сетями имеет нейронная сеть обобщенных регрессионных функций.