【レポート#2】スプライン補間

BV20036　大野 弘貴

令和 3 年 11 月 12 日

1　レポート内容

レポート内容【スプライン補間】

(1) 真の関数および区間を適当に設定し、そこから標本点を与え、スプライン補間を求める
(2) 標本点の数を適当に変え、スプライン補間で得られるグラフの概形の変化を調べる
(3) (1) において、真の関数値とスプライン補間によって得られた関数値のゴザをグラフに表示

- 上記の (1)，(2)，(3) を 3 種類以上の真の関数に対してそれぞれ行い，十分な考察を記入し，レポートにまとめること
  - 3 種類のうち 1 つは【レポート#1】のラグランジュ補間で使用した以下の真の関数を用いること．(区間は [-5, 5])

$$f(x) = \frac{1}{1+2x^2}$$

- 真の関数としては，多項式，三角関数，指数関数，対数関数，及びそれらの組み合わせ等，異なる種類 (形) を選ぶこと

2　スプライン補間とは

ラグランジュの補間は次数が高くなると振動してしまう．そこで，全区間を小区間に分割し，小区間ごとにラグランジュ補間を行うことで振動がなくなることを元に考えられたのがスプライン補間である．

スプライン補間

- スプライン補間

$$S(x) := S_k(x) = a_k(x-x_k)^3 + b_k(x-x_k)^2 + c_k(x-x_k) + d_k,$$
$$x_k \le x \le x_{k+1}, k = 0, 1, \cdots, n-1$$
$$S_k''(x_k) = u_k, k = 0, 1, \cdots, n$$

- 係数

$$a_k = \frac{u_{k+1}-u_k}{6(x_{k+1}-x_k)}, b_k = \frac{u_k}{2}, d_k = f_k$$
$$c_k = \frac{f_{k+1}-f_k}{x_{k+1}-x_k} - \frac{1}{6}(u_{k+1}+2u_k)(x_{k+1}-x_k)$$

3 実装したコード

ソースコード 1 splinesample.m

```
1  %
2  % スプライン補間のサンプルプログラム
3  %
4  clear
5  M = 99; %区間をM 等分し,M+1点に対するスプライン補間を計算
6  % x_in = [1, 2, 3, 4, 5]; %標本点のx 座標を格納
7  % f_in = [1, 2, 1, 1, 3]; %標本点のy 座標を格納
8   n = 9;%標本点の数 - 1
9
10  x_in = zeros(1, n+1);
11  f_in = zeros(1, n+1);
12
13 %関数
14  f =@(x) 1/(1 + x^2);
15 % f = @(x) cos(sin(x));
16 % f = @(x) exp(x);
17
18  for n_ = 0 : n
19  x_in(n_+1) = -5 + n_ * 10 / n;
20  f_in(n_+1) = f(x_in(n_+1));
21  end
22
23  n = length(x_in)-1;%標本点の数-1
24
25  h =x_in(2)-x_in(1); %標本点の間隔(等間隔の場合)
26  h_M = (x_in(n+1)-x_in(1))/M;%スプライン補間を行う際の分点の刻み幅
27  x_out = zeros(M+1,1); %結果のx 座標を格納
28  f_out = zeros(M+1,1); %結果のy 座標を格納
29  g = zeros(n-1,1);%この例では\tilde{g}のこと
30  a = zeros(n,1);%S(x)の係数a
31  b = zeros(n,1);%S(x)の係数b
32  c = zeros(n,1);%S(x)の係数c
33  d = zeros(n,1);%S(x)の係数d
34  %等間隔の場合の係数行列A の作成
35  A = diag(4*ones(1,n-1)) + diag(ones(1,n-2),-1) + diag(ones(1,n-2),1);
36
37  for k=1:n-1
38  g(k) = 6*(f_in(k)-2*f_in(k+1)+f_in(k+2))/(h*h);
39  end
40
41  %係数a_k, b_k, c_k, d_k の計算
```

```
42 for k = 1:n
43 a(k) = (u(k+1) - u(k)) / 6 * (x_in(k+1)-x_in(k));
44 b(k) = u(k)/2;
45 c(k) = (f_in(k+1) - f_in(k))/(x_in(k+1) - x_in(k))-1/6 * (u(k+1) + 2 * u(k
       )) * (x_in(k+1) - x_in(k));
46 d(k) = f_in(k);
47 end
48
49 % ---スプライン補間の計算--------------------------
50 %x(1)～x(n)までをM 等分して各分点に対するスプライン補間を計算する。
51 k = 1;%何番目の区間か
52 for i = 1:M+1
53    if x_in(1) + h_M * (i-1) > x_in(k+1)
54        k = k + 1;
55    end
56    x_out(i) = x_in(1) + h_M * (i-1);
57    f_out(i) = a(k) * (x_out(i) - x_in(k))^3 + b(k) * (x_out(i) - x_in(k))^2+
           c(k) * (x_out(i) - x_in(k)) + d(k);
58
59 end
60
61
62
63 % ---結果をグラフに表示-----------------------------
64 figure
65 plot(x_out,f_out,'*-'), grid on, hold on
66 set(gca,'FontSize',12), hold on
67 plot(x_in,f_in,'o', 'MarkerFaceColor', 'k', 'MarkerEdgeColor', 'k','MarkerSize
       ',10), hold on
68 fplot(f, [-5 5]), hold on
69
70 legend('スプライン補間','標本点','真の関数'), hold off
```

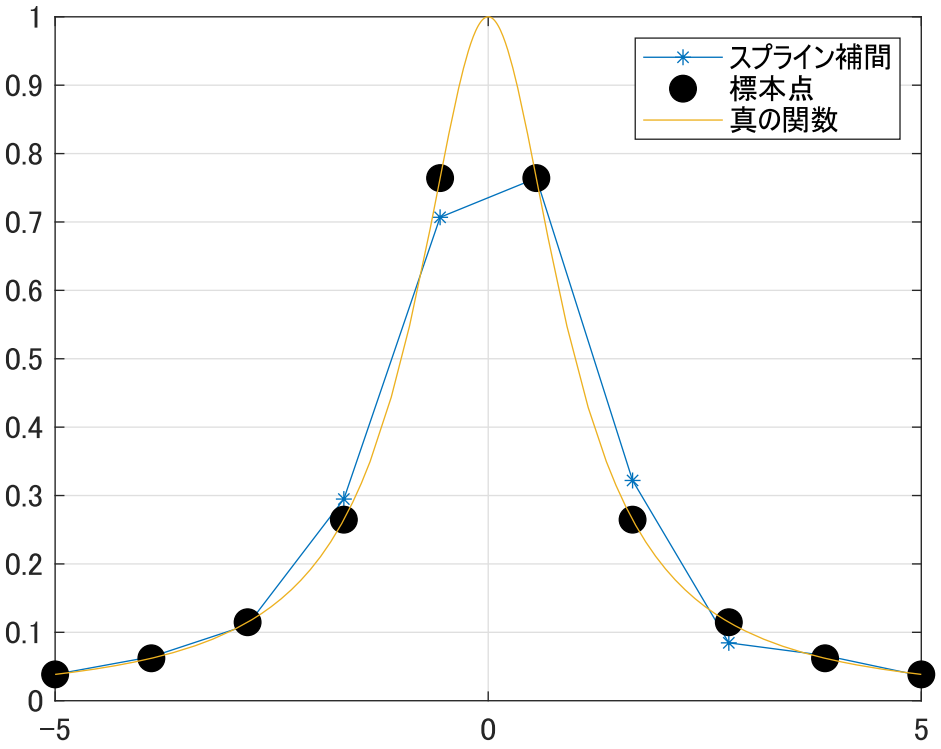

図 1　標本点 10 分割 10

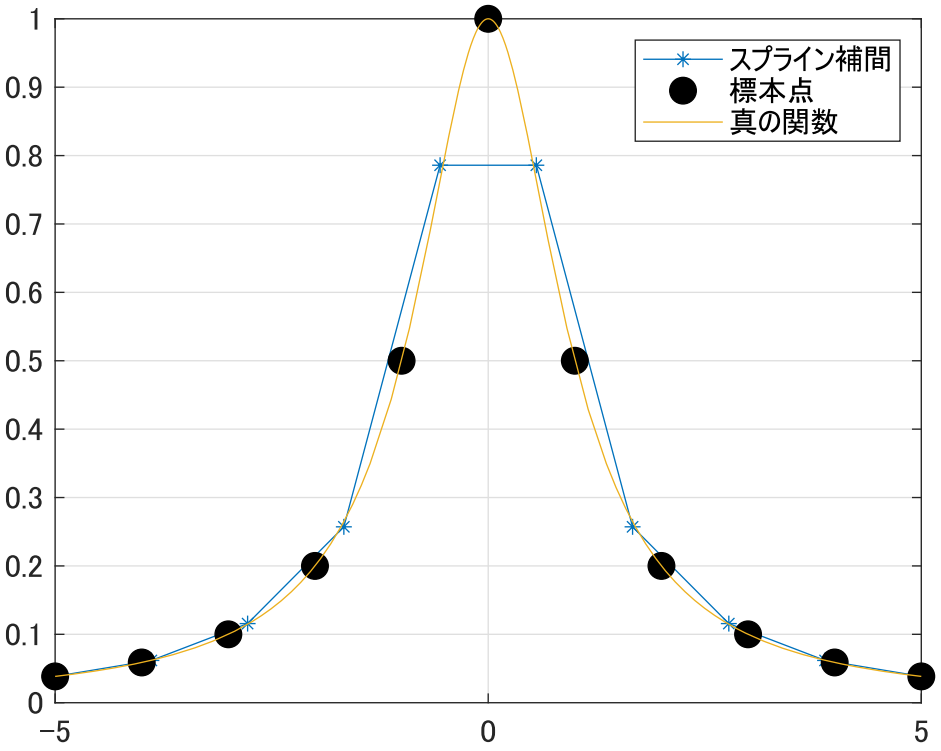

図 2　キャプション

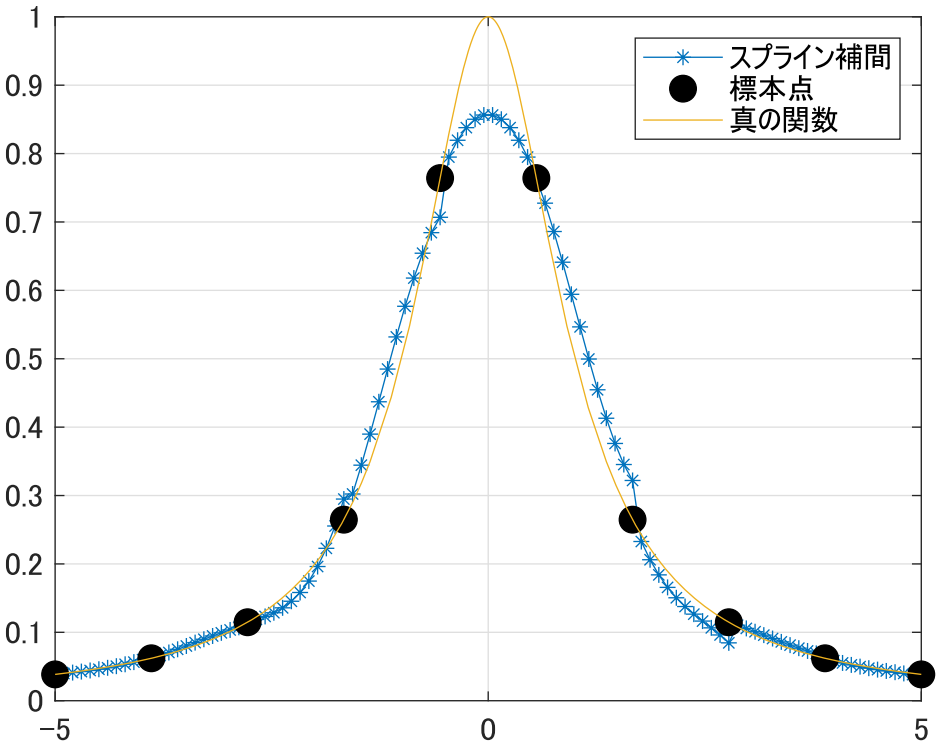

図 3　キャプション

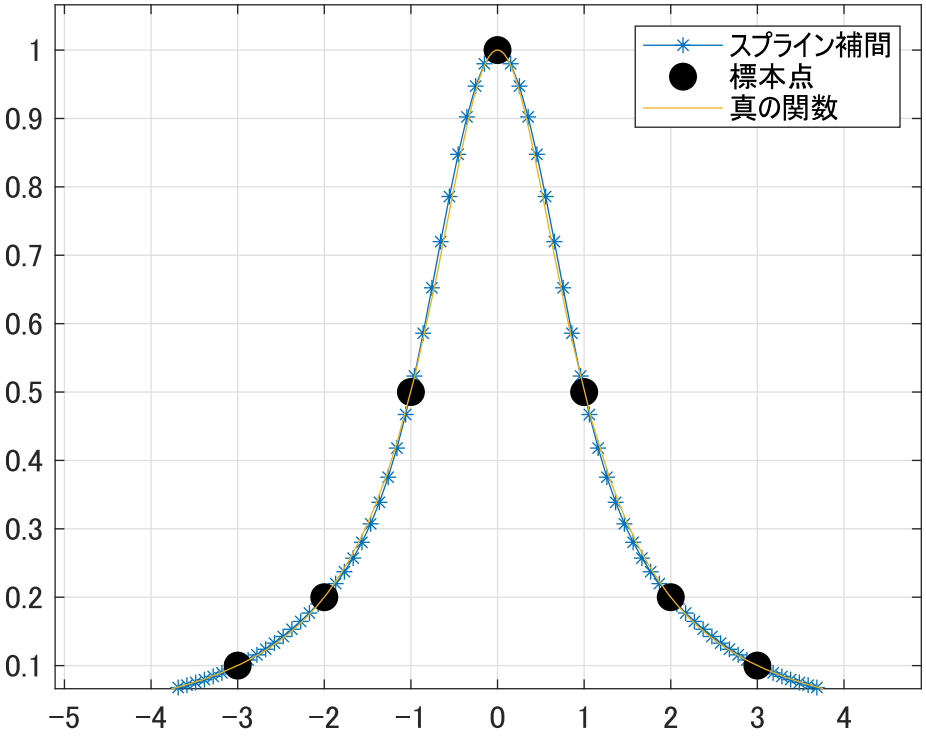

図 4　キャプション

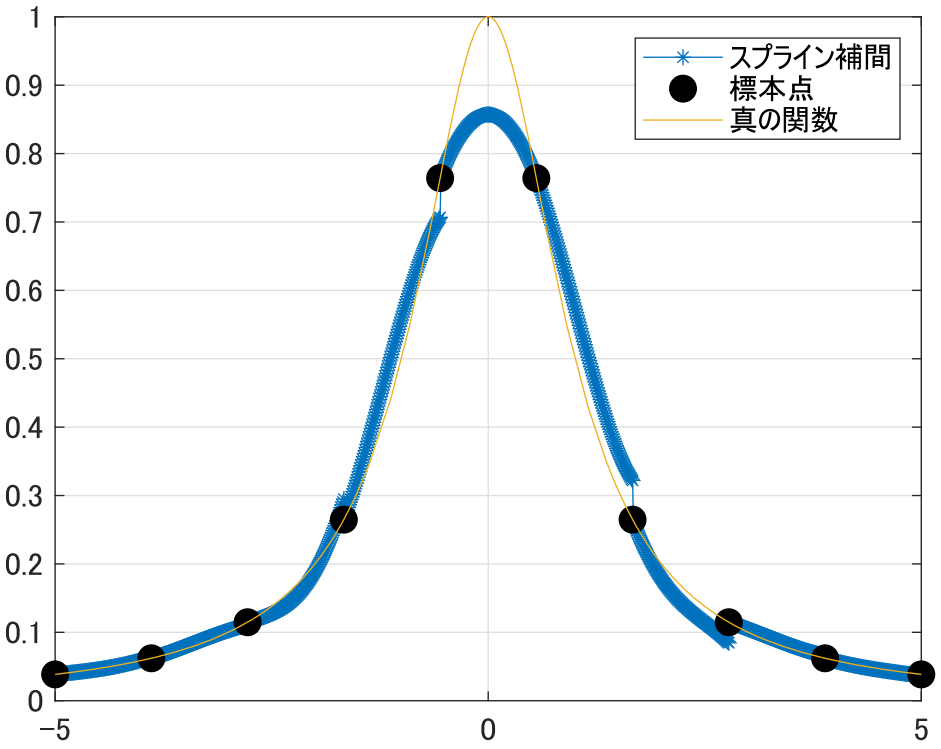

図 5　キャプション

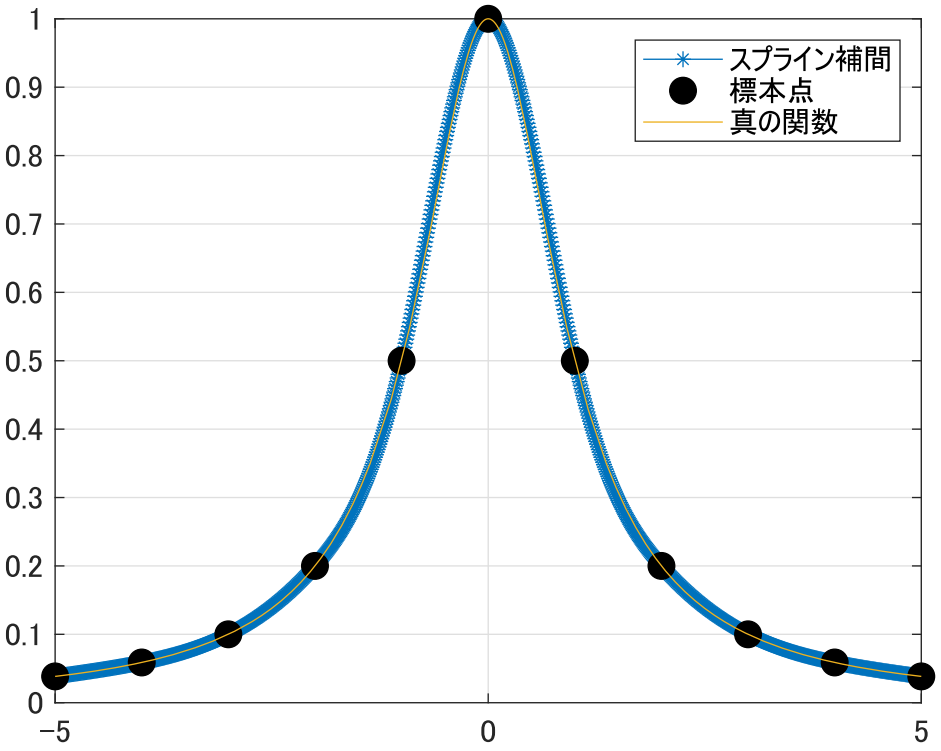

図 6　キャプション

分割数が同じでも標本が奇数のほうが正確なことである。また、分割数が図１のように分割数が少ない時はどちらも誤差も対して変わらなく大きくハズレている。また、分割数が多いほうが滑らかな１つの関数に見える。しかし標本点が奇数のときは、滑らかなことには変わりは無いが頂点の標本を通らずに少し潰れたグラフになっている。

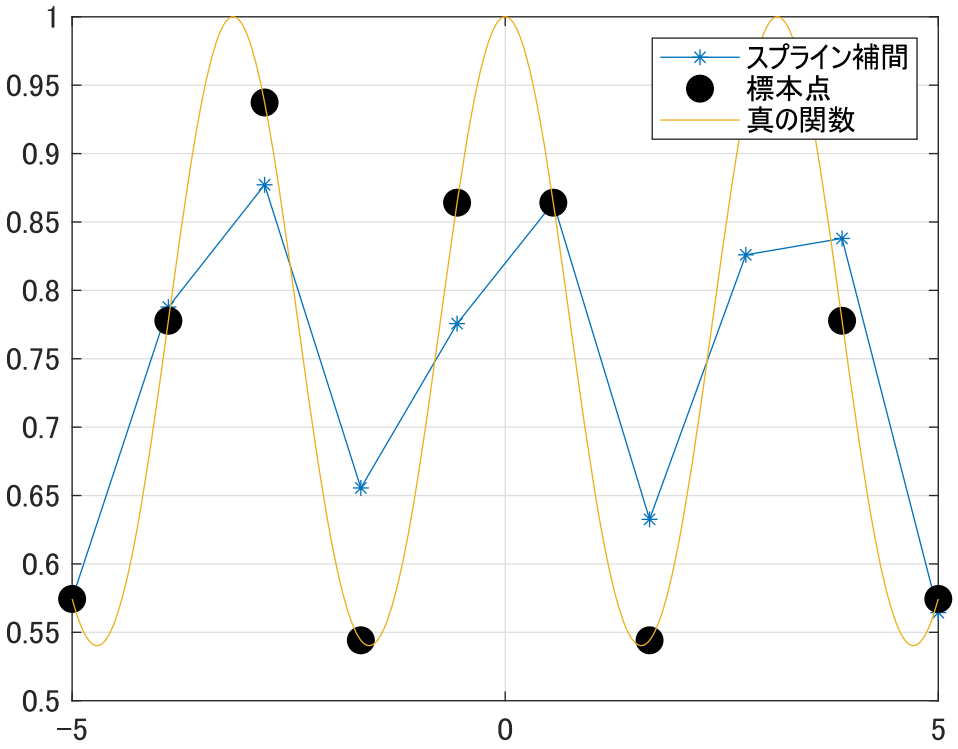

図 7　キャプション

大きくハズレている

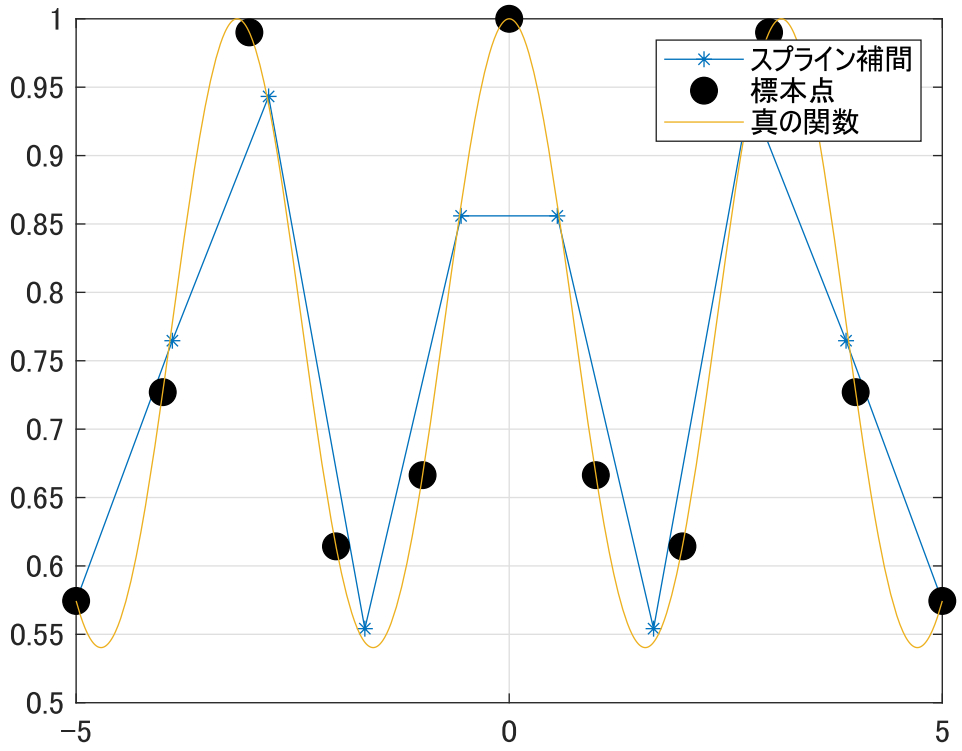

図 8　キャプション

大きくハズレている

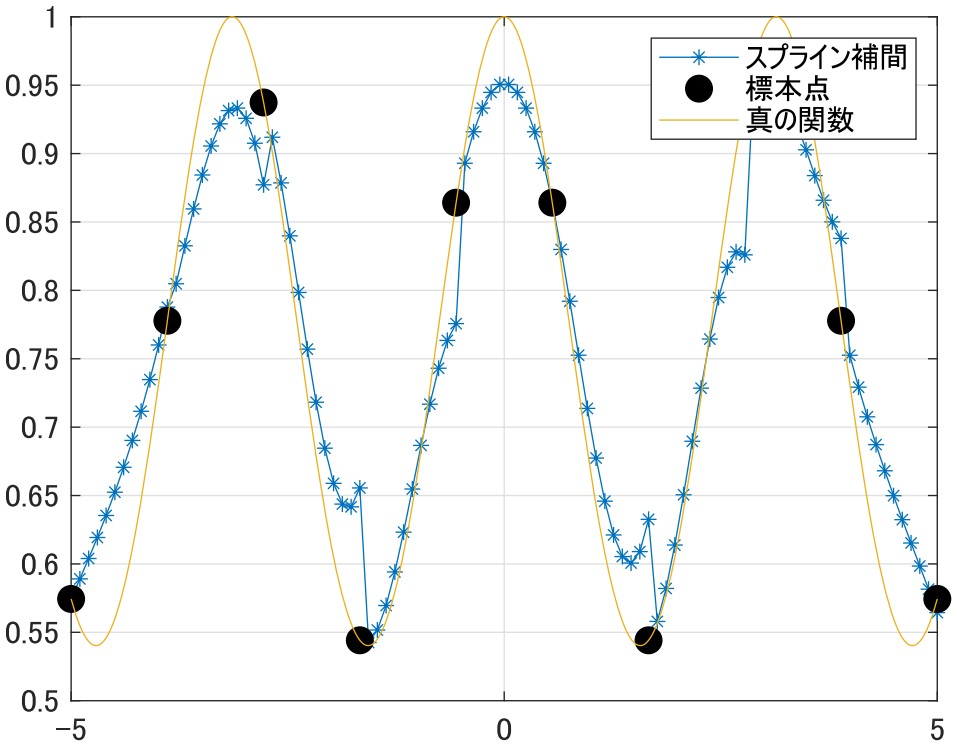

図 9　キャプション

頂点を通らず、ところどころカクカクになっている。

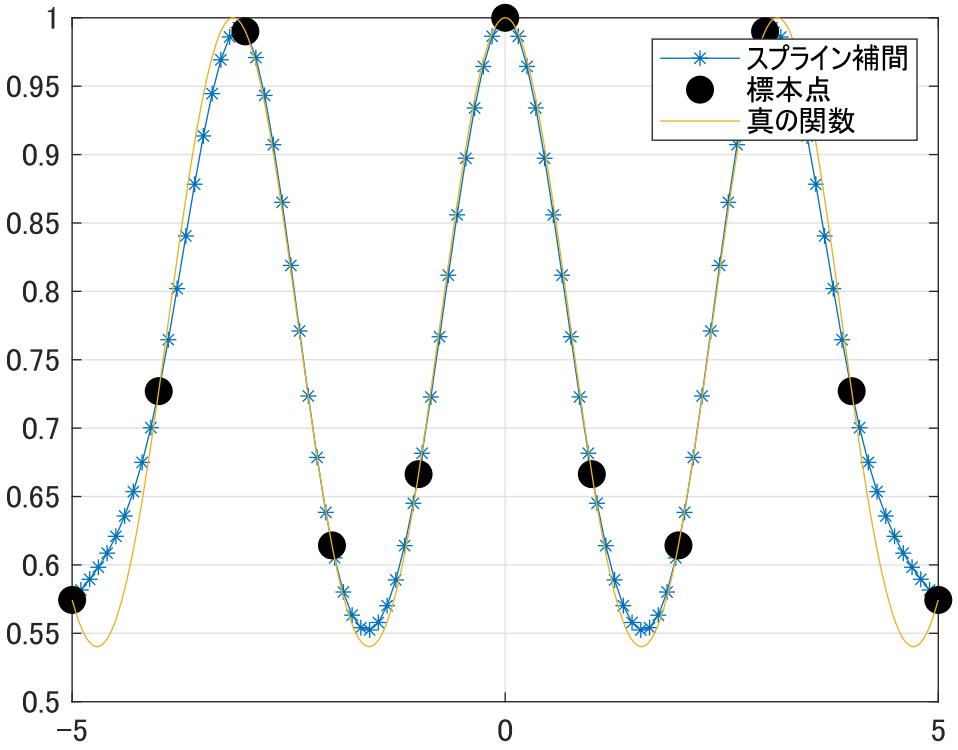

図 10　キャプション

左右の頂点は通っていないが概ね近似できている

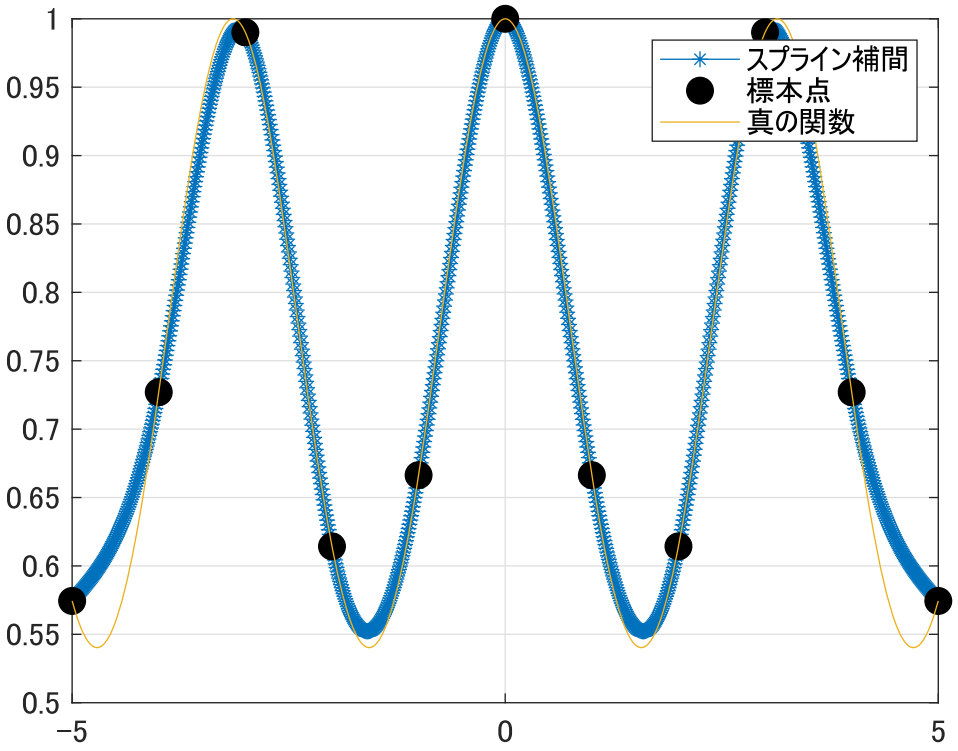

図 11　キャプション

左右の頂点は通っていないが概ね近似できている。

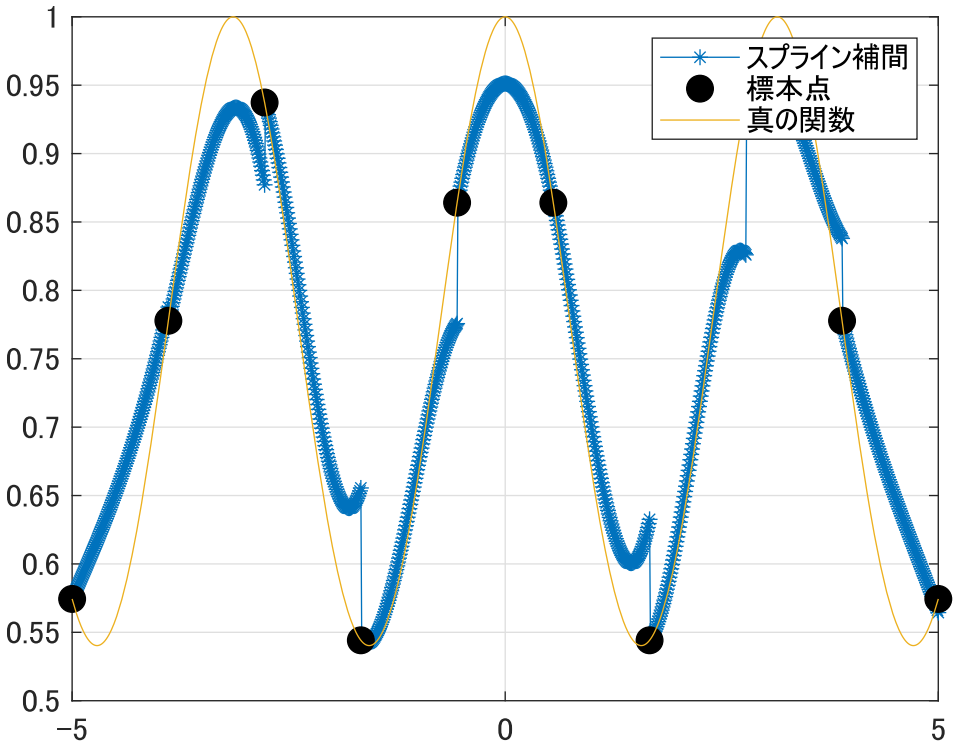

図 12　キャプション

頂点を通らず値が突然飛んでいる。

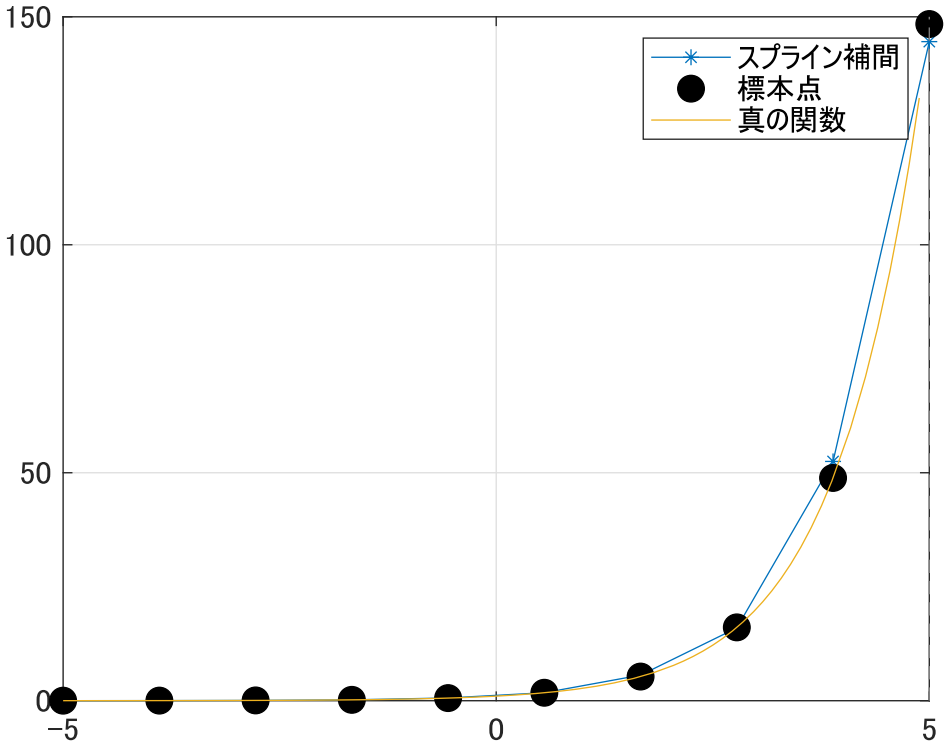

図 13　キャプション

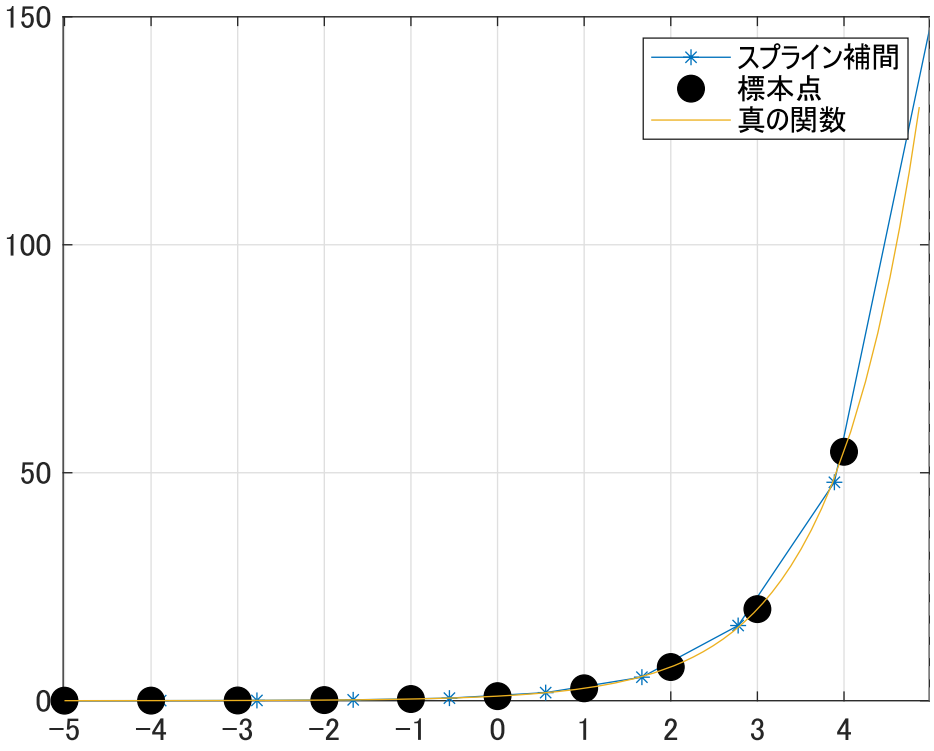

図 14　キャプション

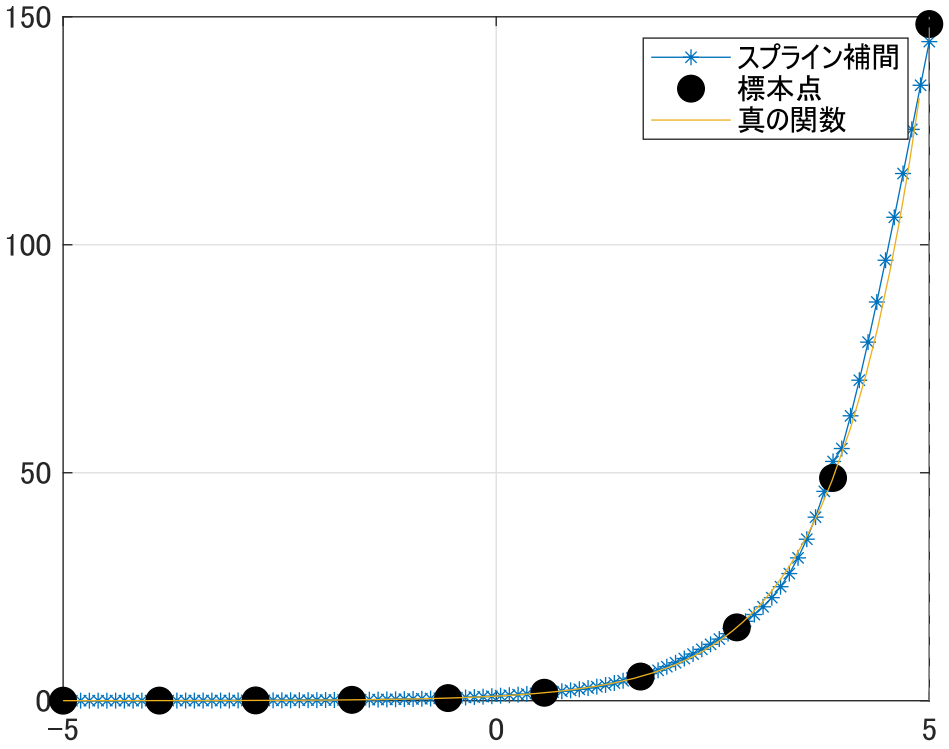

図 15　キャプション

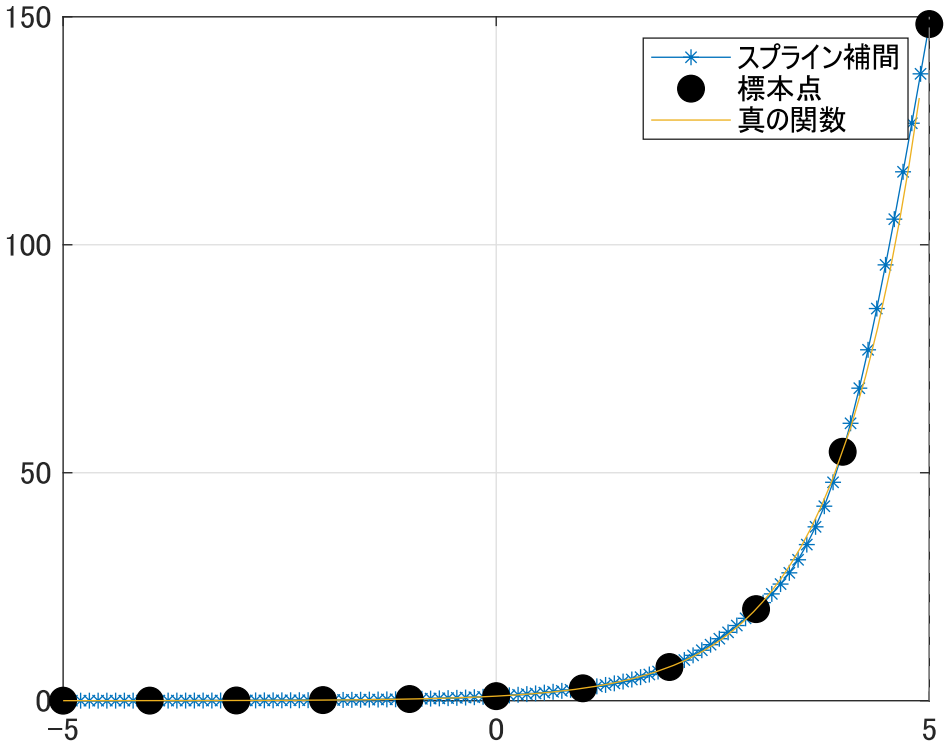

図 16　キャプション

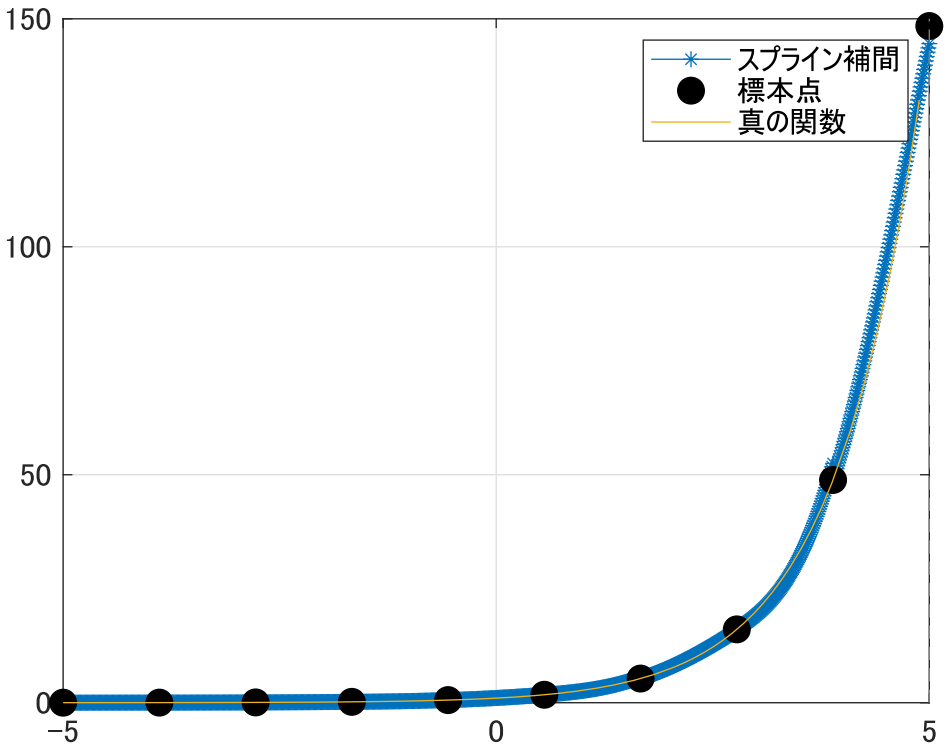

図 17　キャプション

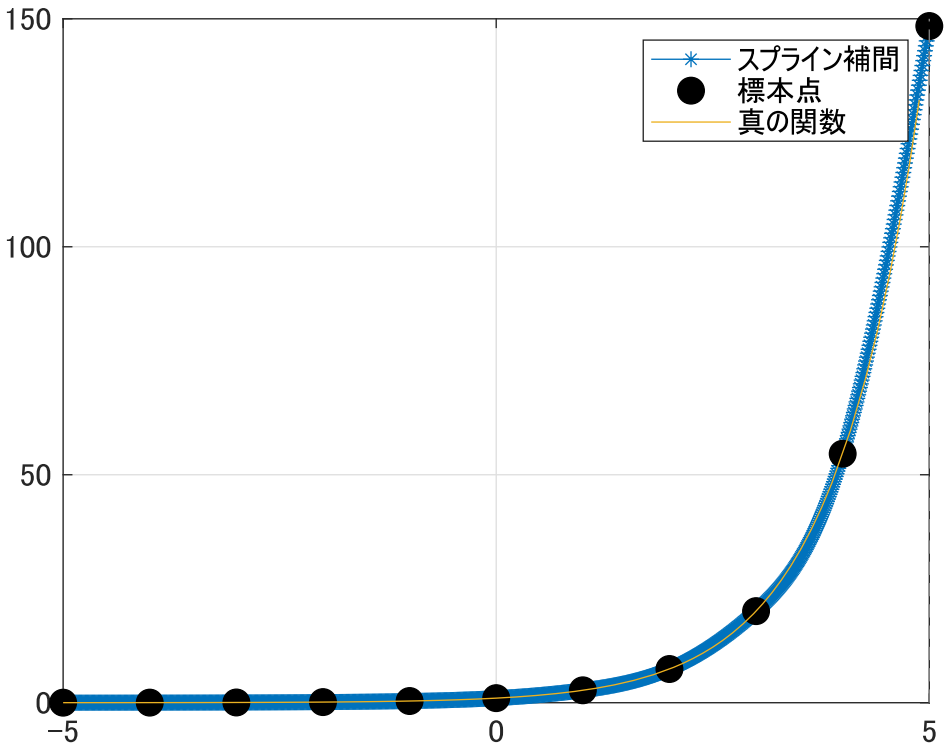

図 18　キャプション

誤差

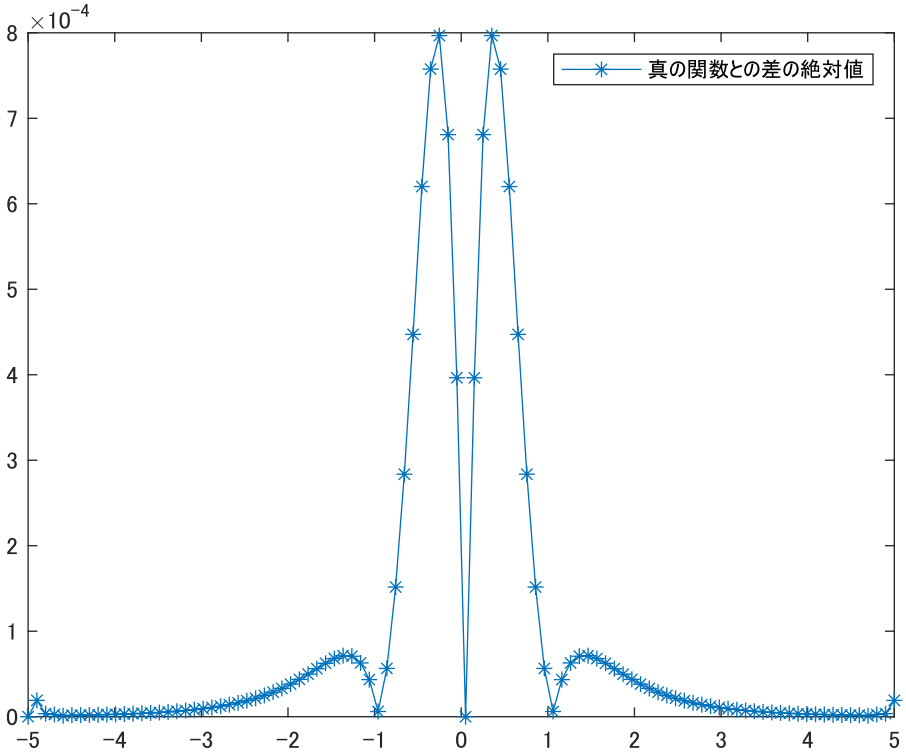

図 19　キャプション

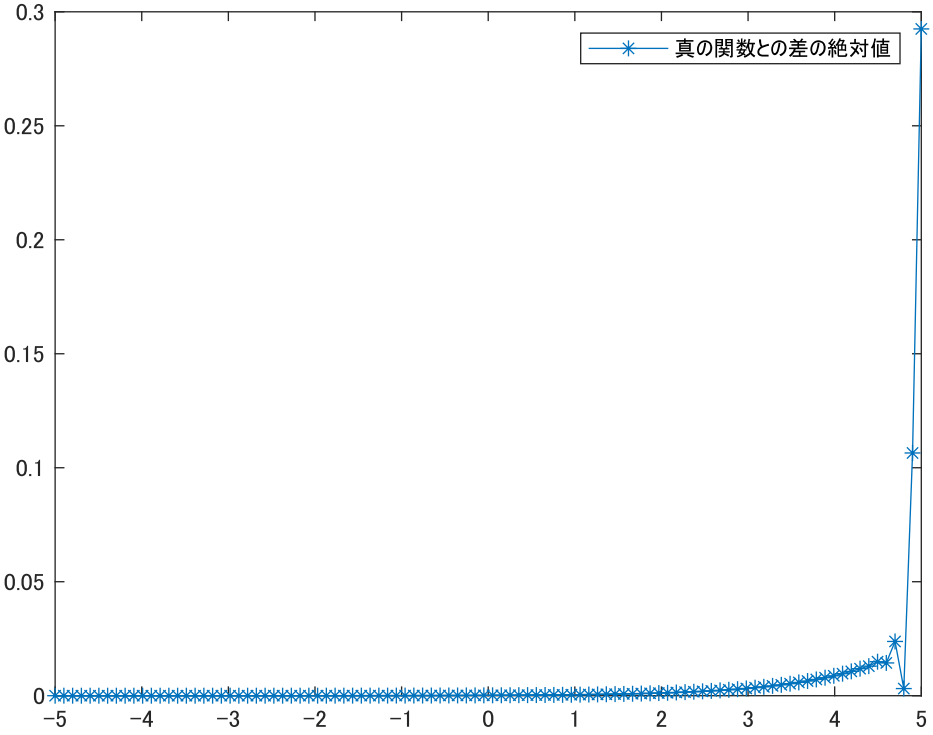

図 20　キャプション

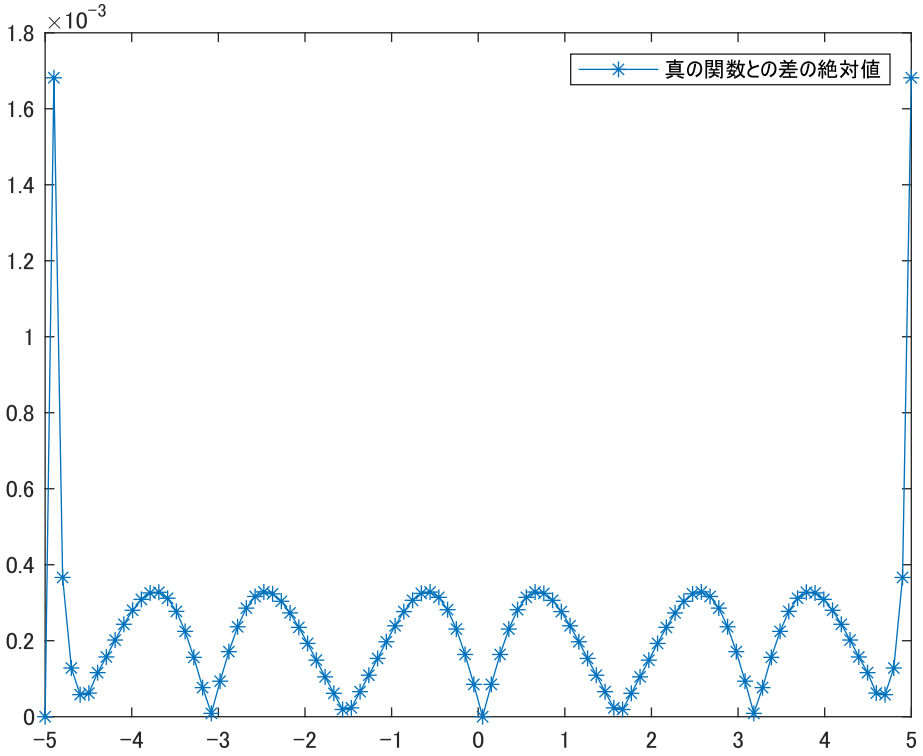

図 21　キャプション

4　考察

最初の関数まず、標本点が奇数の場合、標本点は真の関数の極大値を通る。そのため、標本点が奇数のときは近似式は極大値を通りうまく近似できている。しかし分割数が極端に少ない場合は各頂点を通るだけの分割数がないためうまく近似できていないことが分かる。２個目の関数まず、標本点が奇数の場合、標本点は端の極大値を除き真の関数の極大値を通る。これも最初の関数と同様のことが言える。３個目無理数である e を用いた指数関数なら突飛なが起きるかもしれないと考えたが、e はたかが定数であるからグラフ自体も簡単であるから何も起きずに近似された。この式には頂点が無いので、奇数偶数のときに関わらず同じように分割数を増やせばより良い近似式が得られた。有効桁を指定すれば無理数も有理数と同じなので変なことが起きなかったのかもしれない。

誤差に関しては自然スプランにしたせいか、両端が大きくハズレる形になっている。