Interpolation methods

September 3, 2019

1 Executive Summary

In this report, several interpolation methods are implemented, tested and analyzed. These methods are based on Barycentric Form 2 for first and second kind of Chebyshev points, Newton divided difference form, Bernstein polynomial and piecewise polynomial. The specific functions used to interpolate are $f_1(x) = \|x\| + x/2 - x^2$ and $f_2(x) = (1 + x^2)^{-1}$. In task 6, we use $M_d(x) = \sum_{k=0}^{d+1} m_{d,k}(x)$ where $m_{d,k}(x)$ are computed using the recursive algorithm show in Algorithm 1.

Algorithm 1: $m_{d,k}(x)$

if $d = 0$ **then**
  Return $(-1)^k sign(x - k)/2$;
else if $d > 0$ *and* $k < d + 1$ **then**
  Return $\frac{(x-k)(d+1)}{d(d+1-k)} m_{d-1,k}(x)$;
else
  Return $-\frac{m_{d,d}(x-1)}{d+1}$;

2 Statement of the Problem

This report considers the methods that take N sample points $x = \Big(x_1, \cdots, x_n\Big)^T$ and their values $y = f(x) = \Big(f(x_1), \cdots, f(x_n)\Big)$ as inputs and return an interpolation function h of f.

The following table shows some notations used frequently.

Notation	
d	Degree of the method.
$N + 1$	Number of the sample points.
$C_{1,d}(x)$	Interpolation with barycentric Form with Chebyshev points of the first kind.
$C_{2,d}(x)$	Interpolation with barycentric Form with Chebyshev points of the second kind.
$N_d(x)$	Interpolation with Newton Form.
$B_d(x)$	Bernstein interpolation.
$g_s(x)$	Piecewise interpolation with degree s.
$\hat{h}$	Numerical result of the interpolation h.
(x_i, y_i)	Sample points.

3 Description of Mathematics

3.1 Preprocessing and Evaluation(Task 1)

The summary of this part is given in the following table.

Method	Complexity of evaluation	Complexity of preprocessing	Storage
$C_{1,d}$	$O(d)$	$O(d)$	$O(d)$
$C_{2,d}$	$O(d)$	$O(1)$	$O(d)$
N_d	$O(d)$	$O(d^2)$	$O(d)$
B_d	$O(d)$	$O(d^2)$	$O(d)$
g_s	$O(\log_2(d/s)+3s)$	$O(ds)$	$O(d)$

3.1.1 $C_{1,d}$ and $C_{2,d}$

The evaluation formula of barycentric form 2 is given by

$$\hat{h}(x)=\frac{\sum_{i=0}^{d}\frac{y_i c_i}{x-x_i}}{\sum_{i=0}^{d}\frac{c_i}{x-x_i}} \tag{3.1}$$

where c_i are coefficients that are determined by the sample x_i.

The evaluation involves $5d+6=O(d)$ operations.

For $C_{1,d}$, the sample points are

$$x_i=\cos\frac{(2i+1)\pi}{2d+2}, i=0,1,\cdots,d \tag{3.2}$$

with

$$c_i=\begin{cases}-\sin\dfrac{(2i+1)\pi}{2d+2}, & i \text{ is odd}\\ \sin\dfrac{(2i+1)\pi}{2d+2}, & i \text{ is even}\end{cases}, i=0,1,\cdots d.$$

Every c_i needs 7 operation which makes the total of preprocessing $7(d+1)=O(d)$.

For $C_{2,d}$, the sample points are given by

$$x_i=\cos\frac{i\pi}{d}, i=0,1,\cdots,d \tag{3.3}$$

with

$$c_i=\begin{cases}-1, & i \text{ is odd}\\ 1, & i \text{ is even}\end{cases}, i=1,\cdots d-1, c_0=\frac{1}{2}, c_d=\frac{(-1)^d}{2}.$$

In this case, the preprocessing is trivially $O(1)$.

Note that in my implementation, to reuse the code structure of $C_{1,d}$, the complexity of preprocessing is still $O(d)$.

3.1.2 N_d

The evaluation of Newton form is given by

$$\hat{h}(x) = \sum_{i=0}^{d} c_i \left(\prod_{j=0}^{i} (x - x_j) \right). \tag{3.4}$$

Use Horner rule from Program 1 to compute this polynomial, which gives us $3d = O(d)$ of evaluation.

To compute the coefficient c_i, the following routine is iterated $d + 1$ times.

Algorithm 2: i-th iterative step of computing coefficients for N_d

$c_0 \leftarrow y_i$;
$j \leftarrow 1$;
while $j \leq i$ **do**
  $c_j \leftarrow \frac{c_{j-1} - temp_{j-1}}{x_i - x_{i-j}}$;
  $temp_{j-1} \leftarrow c_{j-1}$;
$temp_i \leftarrow c_i$;

The i-th iteration requires $2i + 2$ storages and $3i + 3$ operations. After $d + 1$ iterations, by reusing the temperate vector *temp*, the total storage is $O(d)$ while the complexity is $3(d + 2)(d + 3)/2 = O(d^2)$.

3.1.3 B_d

The evaluation of Bernstein interpolation is given by

$$\hat{h}(x) = \sum_{i=0}^{d} y_i \binom{d}{i} x^i (1 - x)^{d-i} \tag{3.5}$$

The complexity is $O(d)$ if the power function is counted as 1 operation. The preprocessing of generating binomial coefficient $\binom{d}{i}$ is done in $O(d^2)$ complexity and stored in an $O(d)$ array.

3.1.4 g_s

The evaluating process of piecewise polynomial is different from the algorithms before.

To get a piecewise polynomial $g_s(x)$ generated from $d + 1$ samples, the algorithm will partition d intervals into d/s subintervals, each one has s samples, and perform Newton method on each subinterval. Note that this implies d must be a multiple of s.

While evaluating g_s, find out the subinterval x lays in first, then perform d/s times Newton interpolation. The searching requires $\log_2(d/s)$ complexity while the complexity of evaluating Newton form is $3s$, the overall complexity if $O(\log_2(d/s) + 3s)$.

The preprocessing is done in $O(s^2)$ with d/s times, i.e. $O(ds)$ in total.

3.2 Conditioning and Stability

In this part, we summarize the stability and conditioning of interpolation with barycentric form 2, which subsumes $C_{1,d}$, $C_{2,d}$ and N_d. Due to numerical difficulty of evaluating B_d

when d is large, the analysis of B_d is omitted. Since g_s is the composition of N_d in my implementation, the analysis is therefore the same.

For arbitrary sample points $x_i, i = 0, \cdots, d$, it defines the Lagrange basis functions $l_i(x), i = 0, \cdots, d$

$$l_i(x) = \frac{\omega_{d+1}(x)}{(x - x_i)\omega'_{d+1}(x)} \tag{3.6}$$

The basis induces two quantities, the Lebesgue constant Λ_d and the relative conditioning number $\kappa(x, d, y)$ as follow.

$$\Lambda_n = \left\| \sum_{i=0}^{d} |l_i(x)| \right\|_\infty, \tag{3.7}$$

$$\kappa(x, d, y) = \frac{\sum_{i=0}^{d} |l_i(x) y_i|}{|h(x)|}, \tag{3.8}$$

where exact arithmetic is assumed in both generating and evaluating the interpolant h. Note that Λ_n depends only on the sample mesh x_i while κ also depend on the interpolation method, the sample values y_i and evaluation x.

For conditioning analysis, we are interested in the difference between two exact interpolants h and $\tilde{h}$ from samples (x_i, y_i) and $(x, \tilde{y})$, which should be bounded by the Lebesgue constant.

$$\|h - \tilde{h}\|_\infty \leq \Lambda_d \|y - \tilde{y}\|_\infty \tag{3.9}$$

where $\|\cdot\|_\infty$ is the infinity norm of continuous functions and vectors respectively.

The Lebesgue constant of 3 different sample meshes are given below.

Mesh	Lebesgue constant
Chebyshev points of the first kind	$\Lambda_d^{C1} \approx \frac{2}{\pi} \log d$
Chebyshev points of the second kind	$\Lambda_d^{C2} \leq \Lambda_{d-1}^{C1} \approx \frac{2}{\pi} \log(d-1)$
Equally spaced points	$\Lambda_d^{E} \approx \frac{2^{d+1}}{ed \log d}$

For stability analysis, we are interested in the difference between the exact interpolants h and the numerically computed $\hat{h}$ at point x from the samples (x_i, y_i), where the samples are floating point numbers. In this case, the error comes from the finite precision operations used during the forming and evaluating process of the exact interpolants h.

Notice that these errors generated from the operation are closely related to the unit round-off of the floating point system we are using. Therefore, the final error bound for stability below is in terms of the unit round-off u and the associating $\mu_k = \frac{ku}{1-ku}$,

$$\frac{|h(x) - \hat{h}(x)|}{|h(x)|} \leq (3d+4)\kappa(x, n, y)u + (3d+2)\Lambda_d u + O(u^2) \tag{3.10}$$

or

$$|h(x) - \hat{h}(x)| \leq \left[(3d+4) \sum_{i=0}^{d} |l_i(x) y_i| + (3d+2)\Lambda_d |h(x)| \right] u + O(u^2). \tag{3.11}$$

3.3 Error Bounds of Piecewise Polynomial Interpolation(Task 4)

In this part, the error bounds of piecewise polynomial interpolations in term of particular functions f_1 and f_2 is founded explicitly. They are used to find the minimal numbers of sample points $d+1$ so that the interpolation error $\|h-f\|_\infty$ is bounded under the threshold 10^{-4}. The test results are straight-forward and therefore also included in this section.

3.3.1 Piecewise linear interpolation of f_1

Under equally spaced mesh x where $x_{i+1}-x_i = h, \forall i = 0, \cdots, d-1$, $\forall f \in C^2$, we have

$$|f(x) - g_1(x)| \leq \frac{h^2 \|f''\|_\infty}{8} \tag{3.12}$$

derived from the Lagrange residue.

Notice that f_1' has one and only one discontinuity at $x = 0$. To use the error bound 3.12, we can include $x = 0$ into our sample mesh so that with in each subinterval $[x_i, x_{i+1}]$, the restriction $f_1|_{(x_i,x_{i+1})}$ is secondly differentiable. To include $x = 0$ in the sample mesh, we need d to be even number.

Let d be even number, then $\max\left\{\left\|\left(f_1|_{(x_i,x_{i+1})}\right)''\right\|\right\} = 2$ can be easily found. Therefore, we have

$$\begin{aligned} |f_1(x) - g_1(x)| &\leq \frac{h^2 \max\left\{\left\|\left(f_1|_{(x_i,x_{i+1})}\right)''\right\|\right\}}{8} \\ &= \frac{(2/d)^2 \times 2}{8} = \frac{1}{d^2}. \end{aligned} \tag{3.13}$$

Bound the error bound by $\frac{1}{d^2} \leq 10^{-4}$ and we have $d \geq 100$. The Figure 1 shows the error bound and the actual error achieved in g_1, which reaches up to 9.99996×10^{-5}, i.e. the bound we derived is very tight in this case.

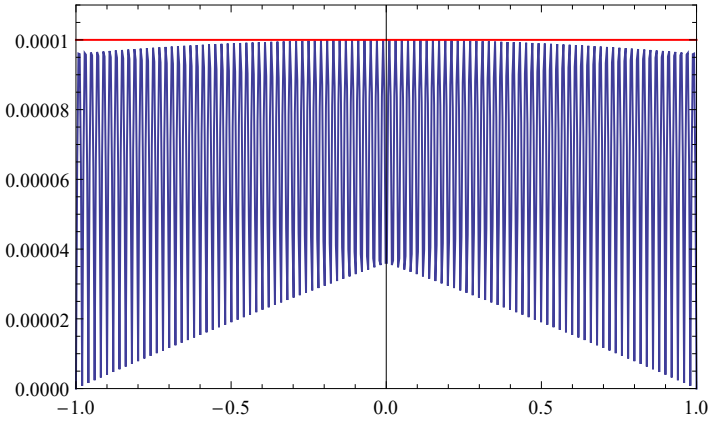

Figure 1: Interpolation Error $|f_1(x) - g_{1,d}(x)|$ with $d = 10$

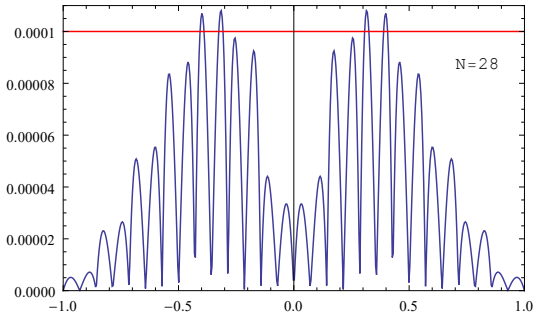

Figure 2: $|f_2(x) - g_{2,d}(x)|$ with $d = 28$

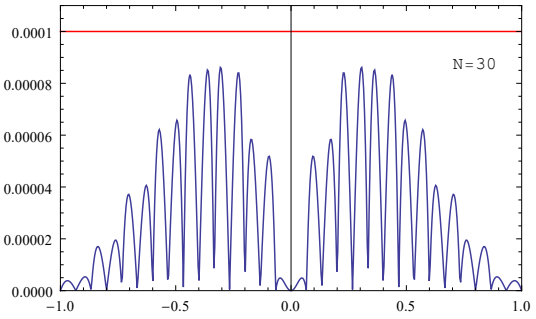

Figure 3: $|f_2(x) - g_{2,d}(x)|$ with $d = 30$

3.3.2 Piecewise quadratic interpolation of f_2

Similar error bound can be derived from the Lagrange residue. Under equally spaced mesh x where $x_{i+1} - x_i = h, \forall i = 0, \cdots, d-1$, $\forall f \in C^3$, we have

$$|f(x) - g_2(x)| \leq \frac{\sqrt{3}h^3\|f^{(3)}\|_\infty}{27} \tag{3.14}$$

Recall that d has to be a multiple of $s = 2$, therefore, d is again an even number but with different reason. After the algebra, we have

$$|f_1(x) - g_1(x)| \leq \frac{\sqrt{3}h^3\|f^{(3)}\|_\infty}{27} \leq 2.9948\cdots \times \left(\frac{2}{d}\right)^3 \leq \frac{2.4}{d^3}. \tag{3.15}$$

Bound the error bound by $\frac{2.4}{d^3} \leq 10^{-4}$, we can solve $d \geq 28.85$, i.e. $d = 30$. Figure 2 and 3 show the error bound and the actual error achieved under $d = 28, 30$. When $d = 28$, the error is unbounded by 3.14, i.e. the bound we derived is very tight in this case.

4 Description of the Algorithm and Implementation

Table 1 is the argument list of the main routines.

Chebyshev(T *x, T *y, int degree, T* xx, T* value, int pnum, int ctrl)
Newton(T* x, T* y, int degree, T* xx, T* value, int pnum)
Bernstein(T* y, int degree, T* xx, T* value, int pnum)
Piecewise(T* x, T *y, int n, int degree, T* xx, T* value, int pnum)

Table 1: Routine List

These routines use similar argument list. x and y are the arrays of sample points. They have integrated two parts together, forming the interpolant and evaluate the interpolant at xx and store in value. It is also possible to return the coefficients of the interpolant, which is essentially the interpolant itself.

As mentioned before, the preprocessing of barycentric form 2 of Chebyshev points of the second kind is $O(d)$ instead of $O(1)$, the searching algorithm for piecewise interpolation is $O(d/s)$ instead of $O(\log d/s)$.

Note that the pointer argument allow us to call Newton easily inside Piecewise, without creating a new sample vector and workspace.

Also note that for computing the binomial coefficient $\binom{i}{d}$, the recursive function structure is used but the workspace, $O(d)$, is created beforehand and it stays fix.

5 Description of Experimental Design and Result

5.1 Interpolants with $n = 6$, $n = 12$ and $n = 20$

The following figures display different interpolants that are done in double precision with small n. The Runge's phenomenon of Newton interpolation and convergence pattern for other interpolation is observed, which agrees with the analysis in class note.

Note that f_1 is a quadratic polynomial in $(-1, 0)$ and $(0, 1)$. Therefore, with proper d, g_2 should capture the exact f_1. When $d = 12, 20$, $x = 0$ happens to be the endpoint of the sub-interval where we apply N_2 within so that g_2 can capture f_2 in the whole domain. When $d = 6$, $x = 0$ is a inside the subinterval in the middle, $(-1/3, 1/3)$, and therefore blue line fails to capture f_1 in the middle. Since f_1 restricted on $(-1/3, 1/3)$ is not a quadratic function.

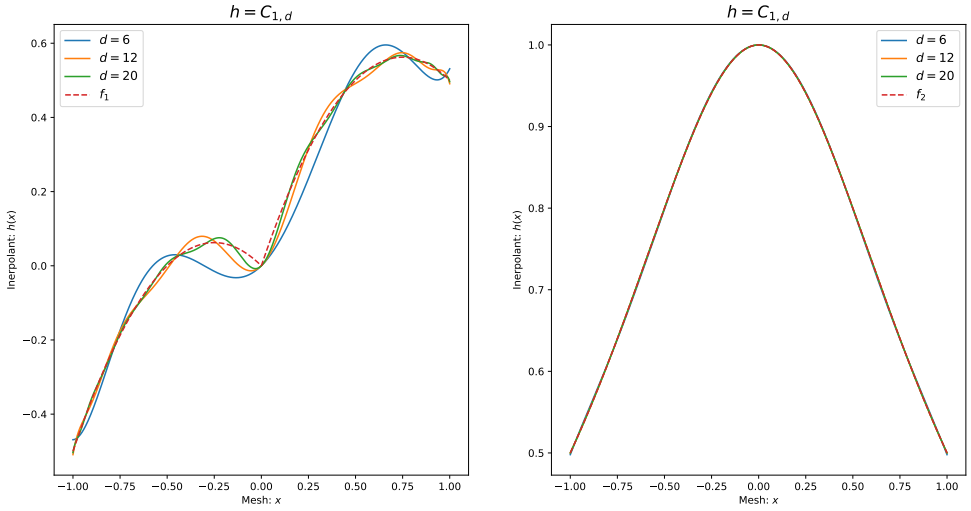

Figure 4: Interpolants of $C_{1,d}$ with Certain d

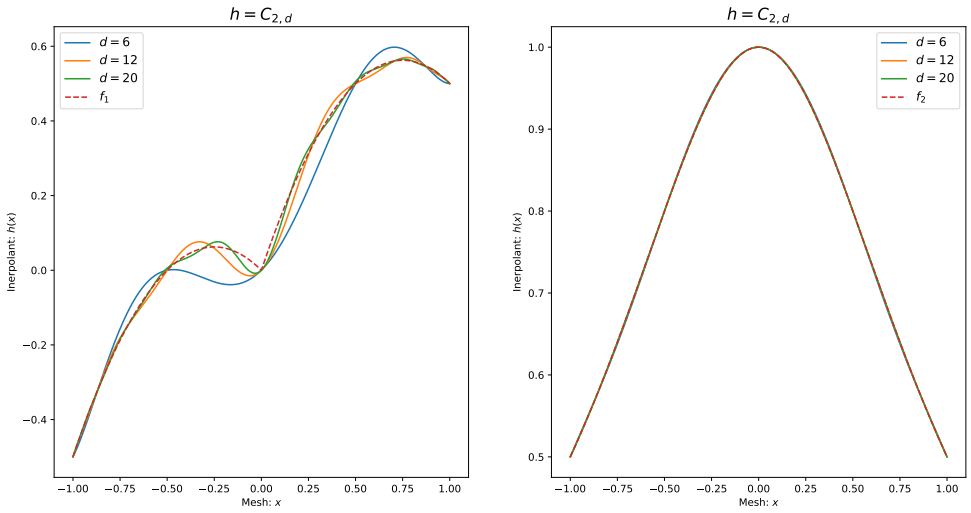

Figure 5: Interpolants of $C_{2,d}$ with Certain d

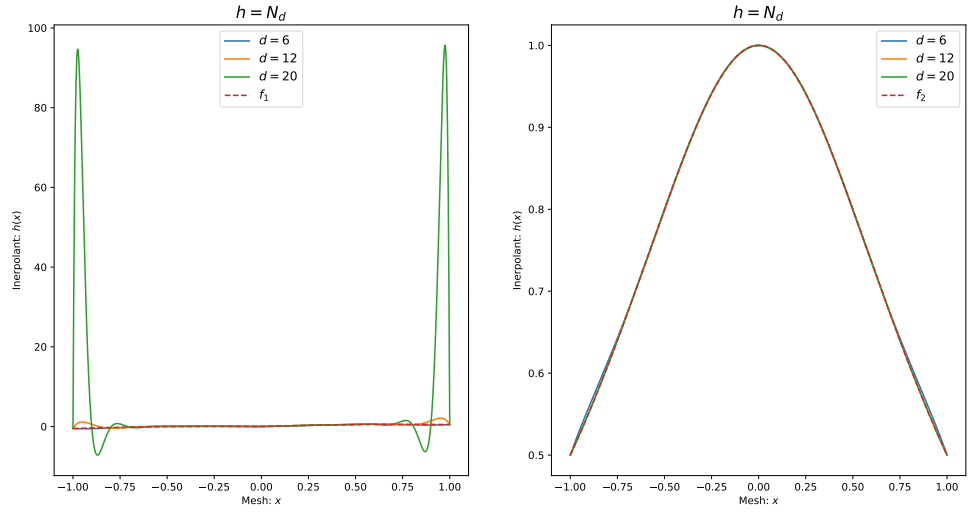

Figure 6: Interpolants of N_d with Certain d

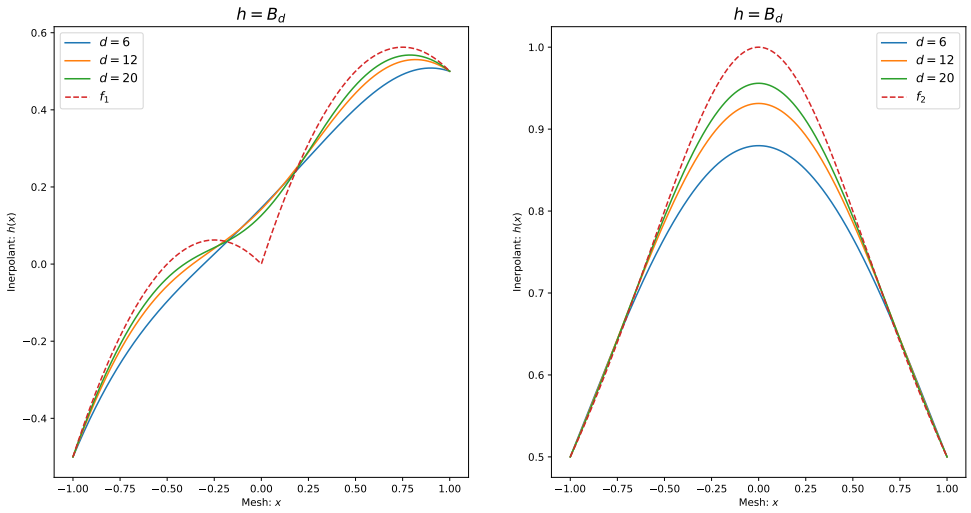

Figure 7: Interpolants of B_d with Certain d

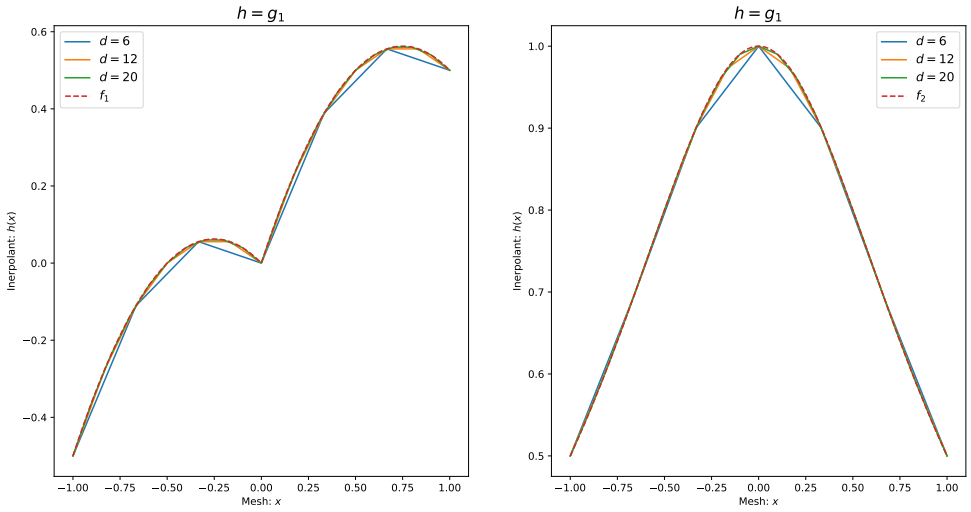

Figure 8: Interpolants of $g_{1,d}$ with Certain d

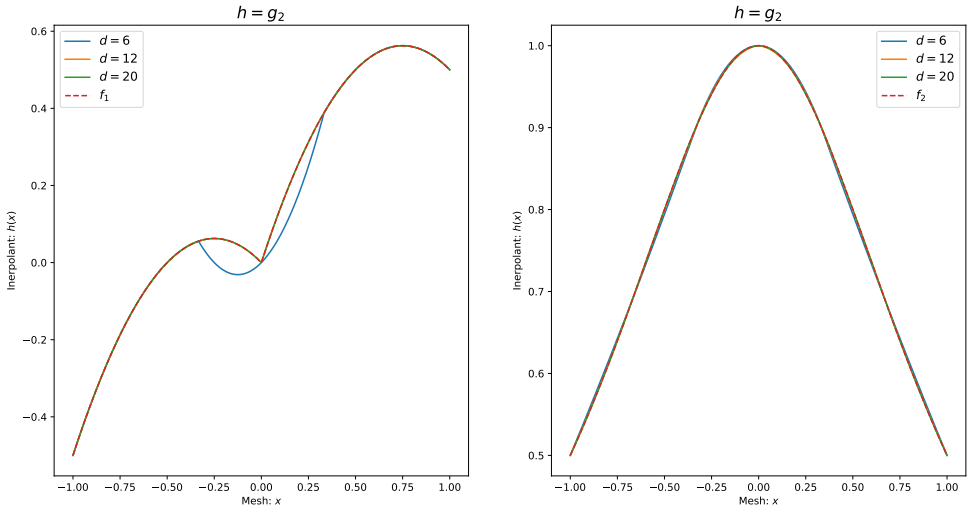

Figure 9: Interpolants of $g_{2,d}$ with Certain d

5.2 Conditioning Test(Task 2)

As discussed in previous section, we would like to verify the error bound 3.9, which says that the difference between the exact interpolants are bounded by the difference of the sampling point.

In this case, the error brought by numerical operation must be neglected, i.e. the difference brought by $\|y - \tilde{y}\|$ should dominant the error brought by numerical operation.

Therefore, we perform our algorithms in double precision with data (x_i, y_i) so that the numerical error is of $u^{double} = 1.11 \times 10^{-16}$ level. Then we perturb the sample data, $(x_i, y_i + p_i)$, where p_i is random noise of $u^{single} = 5.96 \times 10^{-8}$. Therefore, $\|h - \tilde{h}\| \gg \|\hat{h} - h\| \approx \|\hat{\tilde{h}} - \tilde{h}\|$, i.e. $\|h - \tilde{h}\| \approx \|\hat{h} - \hat{\tilde{h}}\|$.

Under this setup, the bound we want to test is

$$\|h - \tilde{h}\|\infty \leq \Lambda_d \|p\|_\infty,$$

where p is the noise we introduce. This bound is computed in Julia during the plotting. Figure 10 agrees with our expectations.

Generally speaking, the relative error stays bounded moderate increasing bounds, or even some constant, except for Newton method. In specific, the relative errors of barycentric form 2, upper 2 subplots, are bounded by 3.9. Newton form N_d, due to Runge's phenomenon, is ill-conditioned as expected. Bernstein interpolation is well-conditioned as stated in class notes. By restricting the degree of Newton interpolation, the piecewise interpolation methods trade smoothness of interpolant for well-conditioning.

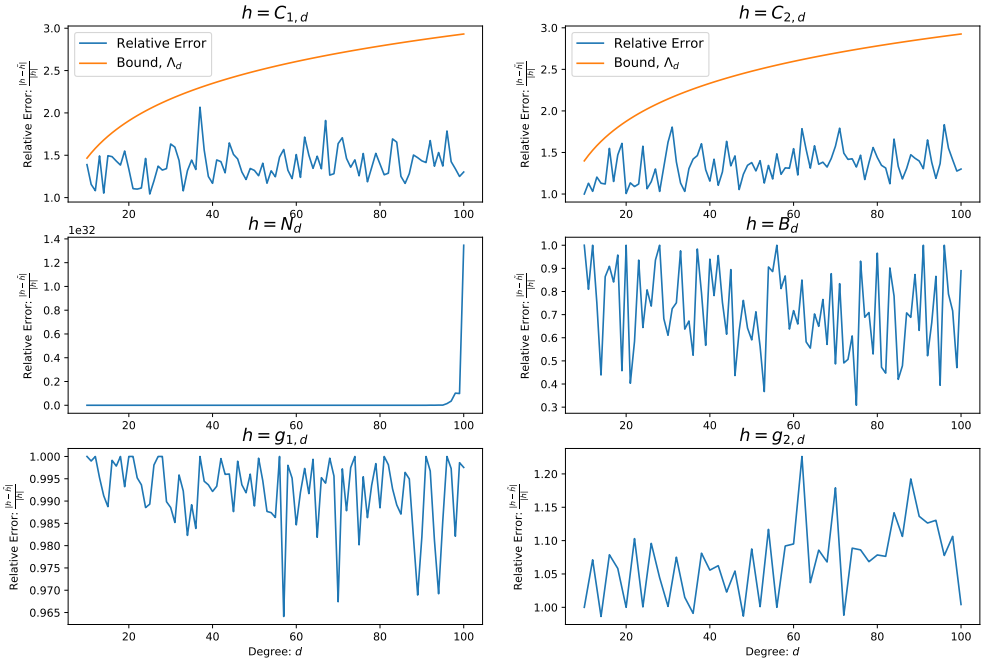

Figure 10: Relative Error under Perturbation at $5u^{single}$ Level

5.3 Stability Test(Task 3)

In this part, we would like to verify the error bound 3.11 with respect to the unit round-off u, i.e. $\|\hat{h} - h\|$ is of interest. However, we do not have access to the exact interpolant h. Therefore, similar to what was done in Conditioning Test, we will treat the computed interpolant $\hat{h}^{double}$ as our exact interpolant. This strategy is based on the assumption that the methods we implement are well posted, so that $\|\hat{h}^{single} - h\| \gg \|\hat{h}^{double} - h\|$, i.e. $\|\hat{h}^{single} - h\| \approx \|\hat{h}^{single} - \hat{h}^{double}\|$.

Note that to make sure we are working on the same problem, the sample (x_i, y_i) feed to the double precision routine are single precision floating numbers, i.e.

$$\hat{h}^{single} = Routine(x_i^{single}, y_i^{single})$$

and

$$\hat{h}^{double} = Rountin(double(x_i^{single}), double(y_i^{single}))$$

where $double(\cdot)$ cast the single precision floating point number to double precision.

Under this set up, we want to verify the bounds for $C_{1,d}$ and $C_{2,d}$ derived Equation 3.11

$$|\hat{h}^{double}(x) - \hat{h}^{single}(x)| \leq \left[(3d+4) \sum_{i=0}^{d} |l_i(x) y_i| + (3d+2) \Lambda_d |h(x)| \right] u^{single}.$$

and numerically verify the stability of Newton interpolation, Bernstein interpolation and piecewise polynomial interpolation. The bound for the former two are computed in Julia during the plotting. $d = 20$ is used here.

For Newton and Bernstein interpolation, we expected them to be numerically stable since the main operations involving repeatedly polynomial evaluations are done using Horner's rule, which has concluded to be stable in Program 1. For piecewise polynomial interpolation, it is trivial to see stability due to the low degree interpolant in each sub-interval. Figure 11 agrees with our expectation, where all methods has numerical error under around $5u^{single}$, $50u^{single}$ for Newton interpolation.

5.4 Convergence Rate of Piecewise Polynomial Interpolation(Task 5)

In this part, we verify the error bounds 3.12 and 3.14 we derived in Section 3.3. The log-log plot 12 shows that our error bounds for even d are very tight in terms of quantity and order to d. Interestingly, the odd d branch on the left is empirically bounded by d^{-1}, indicating the extra error brought by including the non-smooth $x = 0$ within the interpolant is of $O(d^{-1}) = O(h)$, where $h = 2/d$ is the length of each sub-intervals. This result can be proven using the fact that f_1 is Lipschitz continuous around $x = 0$, which is not the main concern in the report and therefore omitted.

6 Conclusion

In this report, several interpolation methods are implemented in efficient ways and tested in terms of conditioning, stability and convergence. We have shown that barycentric form 2 with two kinds of Chebyshev points are well-conditioned and stable with respect to finite precision operation under all circumstances. The disadvantage is that we have no freedom in choosing sample mesh (x_i).

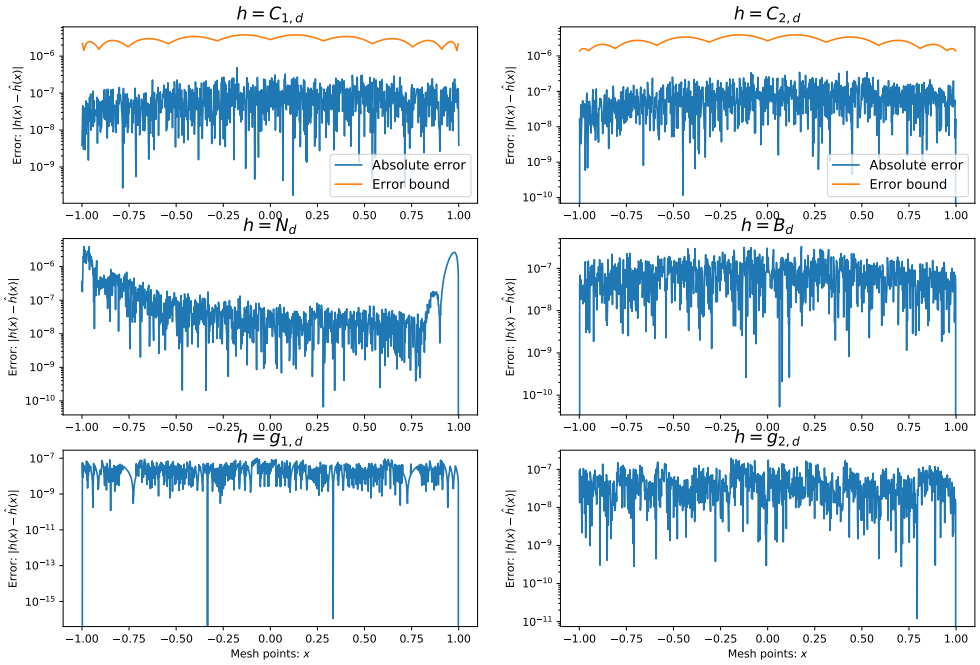

Figure 11: Absolute Error between Double and Single Precision Interpolants.

Newton interpolation diverges due to Runge's phenomenon but it is still stable under numerical operation.

Bernstein interpolation is convergent, well-conditioned and stable but it has numerical difficulty in evaluation. The complexity is $O(d^2)$ and there is potential overflow problem in evaluating binomial coefficient and $x^i(1-x)^{d-i}$ for large d.

Piecewise interpolation g_s is a good option if the smoothness of interpolant is not important. In this case, g_s works with any sample mesh and captures function nicely with a very tight error bound. By restricting the degree of Newton method, g_s is well-conditioned and numerically stable and the preprocessing is essentially $O(d)$ when s is fixed and small. Adaptive strategies can be used in choosing sample mesh in order to optimize efficiency.

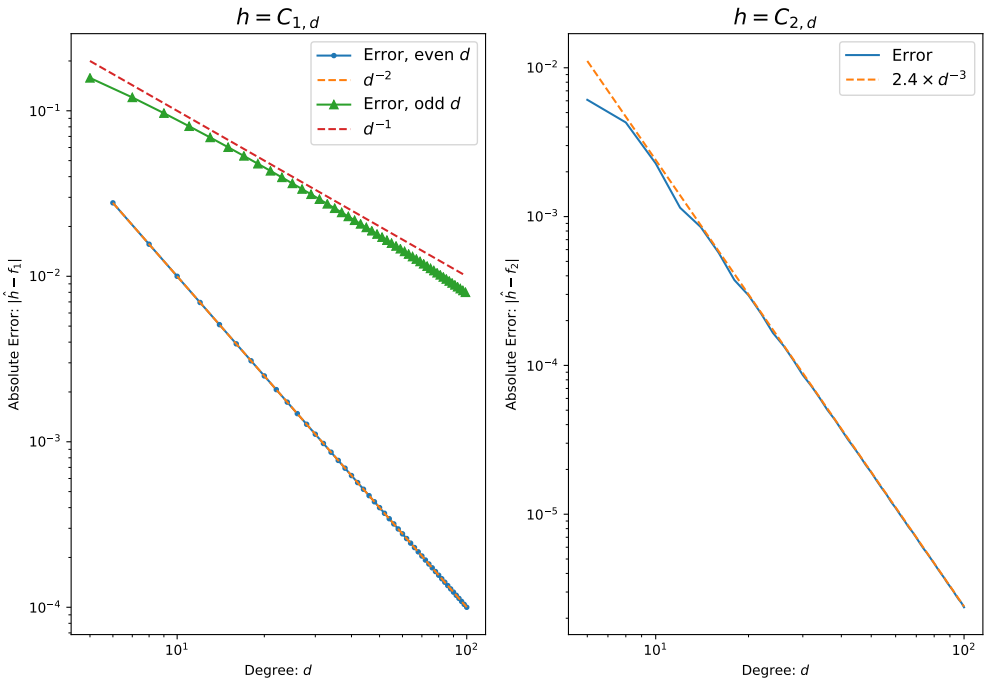

Figure 12: Absolute Error between Piecewise Interpolants and f_i in Double Precision.