Chapter 7 Boundary Value Problem

Preliminary:

Theorem 1(Newton Approximation, see p224 , p213[1]): Assume that $P_n(x)$ is Lagrange polynomial (or Newton polynomial) to interpolate $f(x)$ such that $f(x)=P_n(x)+E_n(x)$. If $f\in C^{n+1}[a,b]$, then for each $x\in[a,b]$, there corresponds a number $c=c(x)\in(a,b)$ such that error term

(Eq. 1) $$E_n(x)=\frac{f^{(n+1)}(c)}{(n+1)!}\prod_{j=0}^{n}(x-x_j).$$

<proof> first consider $N=1$, define special function

$g(t)=f(t)-P_1(t)-E_1(x)\dfrac{(t-x_0)(t-x_1)}{(x-x_0)(x-x_1)}$, for fixed $x_0<x<x_1$, then we have

(1) $g(x)=f(x)-P_1(x)-E_1(x)\dfrac{(x-x_0)(x-x_1)}{(x-x_0)(x-x_1)}=f(x)-P_1(x)-E_1(x)=0$

(2) $g(x_0)=f(x_0)-P_1(x_0)=0$

(3) $g(x_1)=f(x_1)-P_1(x_1)=0$

Apply Rolle's theorem, we have $g'(d_0)=g'(d_1)=0$ for $x_0<d_0<x$, $x<d_1<x_1$.

Apply Rolle's theorem again, we have $g^{(2)}(c)=0$ for $d_0<c<d_1$.

$g^{(2)}(t)=f^{(2)}(t)-E_1(x)\dfrac{2}{(x-x_0)(x-x_1)}$, so $g^{(2)}(c)=0$ implies

$E_1(x)=\dfrac{f^{(2)}(c)}{2!}(x-x_0)(x-x_1)$. Note that such point $c=c(x)$.

The same argument holds for any N .

Let $p(x)$ be the unique polynomial of degree $\le N$ with $p(\pm1)=0$ and $p(x_j)=v_j$, $w_j=p''(x_j)$, $z_j=p'(x_j)$ for $1\le j\le N-1$.

Let D_N^2 be $(N+1)\times(N+1)$ matrix which maps $(v_0,v_1,\cdots,v_N)^T$ to $(w_0,w_1,\cdots,w_N)^T$, abbreviate as $w=D_N^2 v$, then $w_j=p''(x_j)$ for $0\le j\le N$.

However we have impose boundary condition $p(\pm1)=0$ (i.e $v_0=v_N=0$), in order to keep solvability of matrix D_N^2, we need to neglect w_0, w_N (that is, $\tilde{D}_N^2=D_N^2(1:N-1,1:N-1)$ is target transformation, see Figure 1)

The same reason holds for first derivative matrix $\tilde{D}_N=D_N(1:N-1,1:N-1)$.

Then we approximate differential equation $u_{xx}+au_x+cu=f$ by

$\left(\tilde{D}_N^2+a\tilde{D}_N+c\right)u=f$.

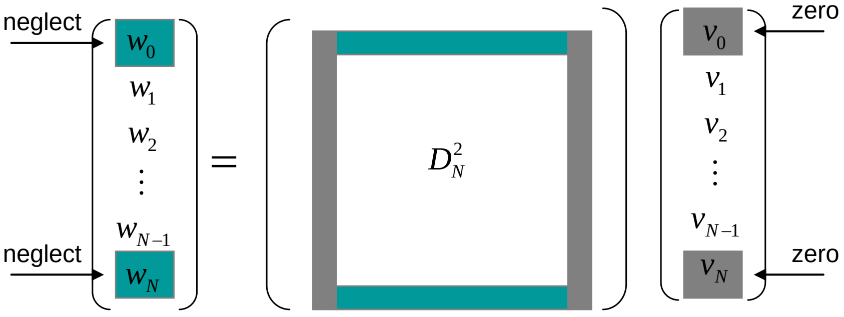


Figure 1: set $v_0 = v_N = 0$ and neglect equation on w_0, w_N.

Example 1(linear problem): $u_{xx} = e^{4x}$ on $-1 < x < 1$ and B.C. $u(\pm 1) = 0$. The exact solution is $u_{exact} = \frac{1}{16}\left(e^{4x} - x\sinh(4) - \cosh(4)\right)$. We try (1) chebyshev node and (2) uniform distribution. We plot the numerical result of polynomial interpolation (use polyval(polyfit(x,u,N),xx) to interpolate point value), one can find that chebyshev node has better result. However in this example uniform distribution also works.

Source code: F:\course\2008spring\spectral_method\matlab\p13.m

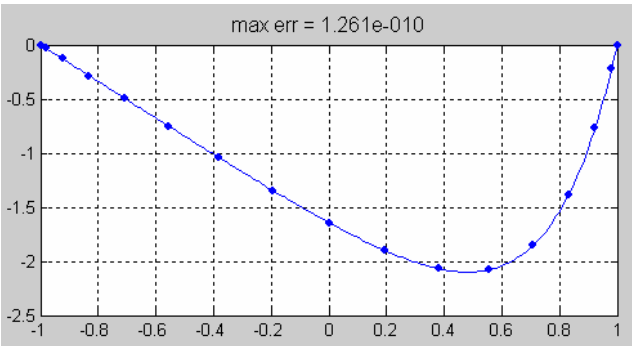


Figure 2: result of chebyshev node, error is 10 digits

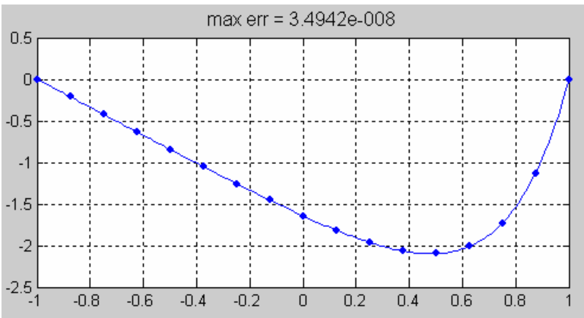


Figure 3: result of uniform distribution, error is 8 digits.

Example 2 (nonlinear problem): $u_{xx} = e^{u}$ on $-1 < x < 1$ and B.C. $u(\pm 1) = 0$. We use fixed point iteration $\tilde{D}_N^2 u_{k+1} = \exp(u_k)$ and try (1) Chebyshev node and (2) uniform distribution. We set $N = 16$ and convergence tolerance as $eps = 1.E-13$, then chebyshev needs 25 iteration but uniform distribution requires 26 iterations, moreover when $eps = 1.E-15$, uniform distribution cannot reach this tolerance (it only reach $eps = 1.E-13$) in Matlab.

Source code: F:\course\2008spring\spectral_method\matlab\p14.m

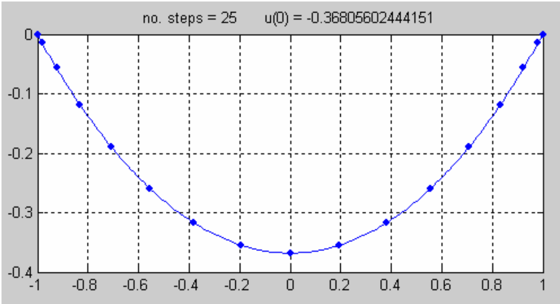


Figure 4: result of chebyshev node, with eps = 1.E-13

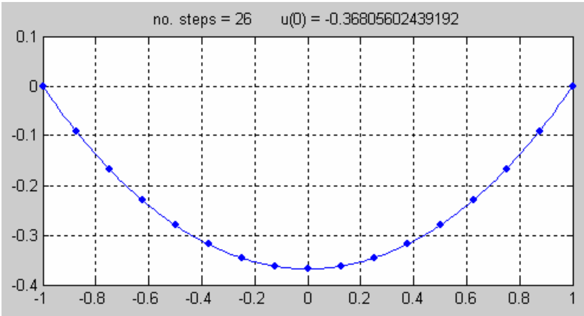


Figure 5: result of uniform distribution, with eps = 1.E-13.

Question 1 (exercise 7.3): Why fixed point iteration converges ?

<**ans**> we use iteration $u_{k+1} = \left(\tilde{D}_N^2\right)^{-1} \exp(u_k)$, then from standard argument of fixed point method, we estimate $u_{k+1} - u_k = \left(\tilde{D}_N^2\right)^{-1} \left(\exp(u_k) - \exp(u_{k-1})\right) = \left(\tilde{D}_N^2\right)^{-1} \exp(u^*)(u_k - u_{k-1})$ where $u^* = \left(u_1^* \mid u_2^* \mid \cdots \mid u_{N-1}^*\right)$. From convex property of u due to $u_{xx} = e^u > 0$, we know $u < 0$ on $-1 < x < 1$ since $u(\pm 1) = 0$. Hence $\exp(u_j^*) < 1$ for any j. Therefore

$$\|u_{k+1} - u_k\| \le \left\|\left(\tilde{D}_N^2\right)^{-1}\right\| \left\|\exp(u^*)(u_k - u_{k-1})\right\| \le \left\|\left(\tilde{D}_N^2\right)^{-1}\right\| \left\|(u_k - u_{k-1})\right\|$$

(1) all eigen-value of $\tilde{D}_N^2$ are negative and $\lambda_{\max}\left(\tilde{D}_N^2\right) = -3.17\text{E+3}$, $\lambda_{\min}\left(\tilde{D}_N^2\right) =$ 2.4674. This means $\rho_{\max}\left(\left(\tilde{D}_N^2\right)^{-1}\right) = 0.405 < 1$, hence such iteration converges.

(2) In fact, $\tilde{D}_N^2$ is not symmetry, since $norm\left(\tilde{D}_N^2 - \left(\tilde{D}_N^2\right)^T, \infty\right) = 1.1E3$, if we use 2-norm, then $\left\|\left(\tilde{D}_N^2\right)^{-1}\right\|_2 = \sqrt{\rho\left(\left(\tilde{D}_N^2\right)^{-1}\left(\tilde{D}_N^2\right)^{-T}\right)} = \sqrt{0.408} = 0.64 < 1$.

(3) We can measure convergence factor $r = \dfrac{\|u_{k+1} - u_k\|_\infty}{\|u_{k+2} - u_{k+1}\|_\infty}$

k	1	2	3	4	5	6
r	0.337	0.283	0.298	0.293	0.295	0.294

Question 2 (exercise 7.4): If we use Newton iteration rather than fixed point iteration, Can we obtain quadratic convergence?

<ans> define $f(\bar{u}) = \tilde{D}_N^2 u - \exp(u)$, $f \in R^{(N-1)\times(N-1)}$, write $f = (f_1, f_2, \cdots, f_{N-1})$, where $f_k(\bar{u}) = \left(\tilde{D}_N^2 u\right)_k - \exp(u_k) = a_{kj}u_j - \exp(u_k)$ for $\tilde{D}_N^2 = (a_{ij})$. Then $\frac{\partial}{\partial u_i} f_k(\bar{u}) = a_{ki} - \delta_k^i \exp(u_k)$, means $Df(u) = \tilde{D}_N^2 - diag(e^u)$.

Basic Newton iteration $f(u^{k+1}) = f(u^k) + Df(u^k)(u^{k+1} - u^k) + h.o.t$, then

(Eq. 2) $u^{k+1} = u^k - \left(Df(u^k)\right)^{-1} f(u^k)$

Table 1: convergence factor of Newton method, it is clear that Newton method is second order convergence

k	1	2	3	4	5	6
r	0.0457	0.0015	2.342E-6	8.559E-6		

Example 3 (eigenvalue problem): $u_{xx} = \lambda u$ on $-1 < x < 1$ and B.C. $u(\pm 1) = 0$.

We plot eigen-function $v_5, v_{10}, v_{15}, v_{20}, v_{25}$ in Figure 6 and Figure 7, also compare accuracy of eigen-value in **Table 2**. In the figure, we show a parameter, called **ppw** (number of points per wave length), we define **ppw** as follows.

Eigenfunction $v_n = \sin(k_n(x+1)) = \sin\left(\frac{n\pi}{2}(x+1)\right)$, $\|v_n\|_{L^2} = 1$ where $k_n = \frac{2\pi}{\lambda_n}$ is wave vector, hence we have $\lambda_n = \frac{4}{n}$, so $\#$ of wave front $= \frac{length \text{ of domain}}{\lambda} = \frac{2}{4/n} = \frac{n}{2}$. Then $ppw = \frac{N}{\# \text{ of wave front}} = \frac{N}{n/2} = \frac{2N}{n}$ (in code, author use $ppw = \frac{4N}{\pi n} \approx 1.27\frac{N}{n}$).

Table 2: compare $4\lambda/\pi^2$, first row is analytic result n^2, second row is Chebyshev, third row is uniform node.

Eig5	Eig10	Eig15	Eig20	Eig25	Eig30
25	100	225	400	625	900
25	100	225	400	635.22	2375.33
25	100	$175 + 9.5i$	$211.57 + 138i$	$210.75 + 259i$	$156 + 480.76i$

It is clear that Chebyshev node has better result,

Question 3: In fact, eignevalue of Chebyshev Differentiation matrix is all real, but eigenvalue of matrix of uniform node may be complex ($\max(abs(imag(lam))) = 2183$), why ? In fact in

Matlab, $norm\left(D_2 - D_2^T, \infty\right) = 2.3E11$ for uniform node but $norm\left(D_2 - D_2^T, \infty\right) = 2.9E4$.

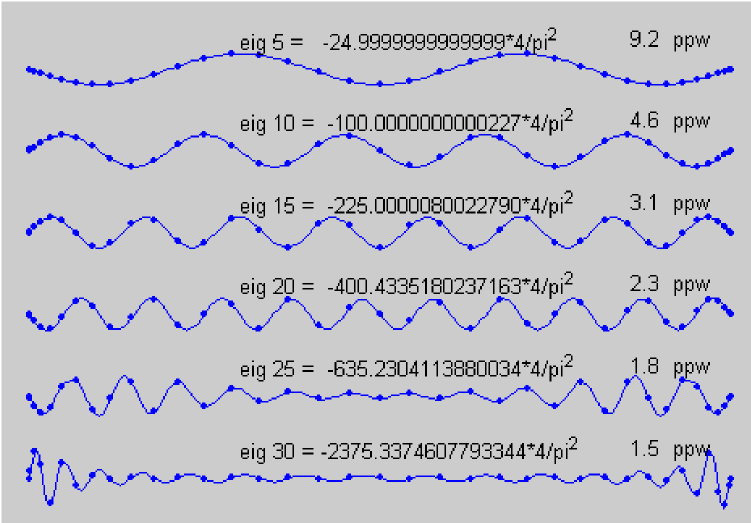


Figure 6: result of Chebyshev node.

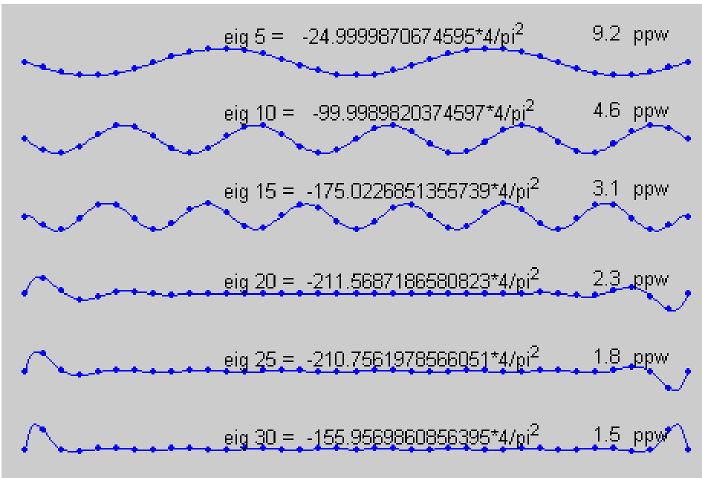


Figure 7: result of uniform distribution.

Example 4 (2D example):

(Eq. 3) $u_{xx} + u_{yy} = 10\sin\left(8x\left(y-1\right)\right), \ -1 < x, y < 1, \ u = 0$ on the boundary.

We try (1) Chebyshev node and (2) uniform distribution. Both distributions have the same sparse pattern of matrix

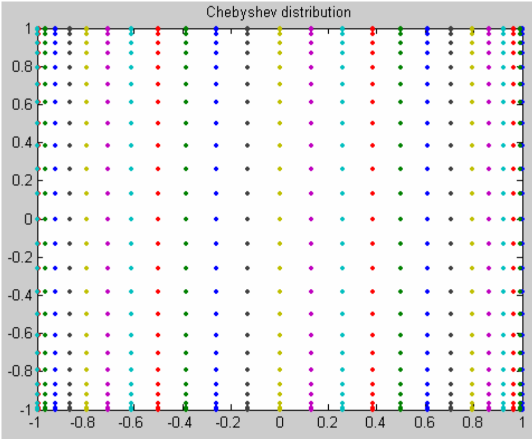


Figure 8: Chebyshev distribution on 2D, more grid points are clustered near boundary.

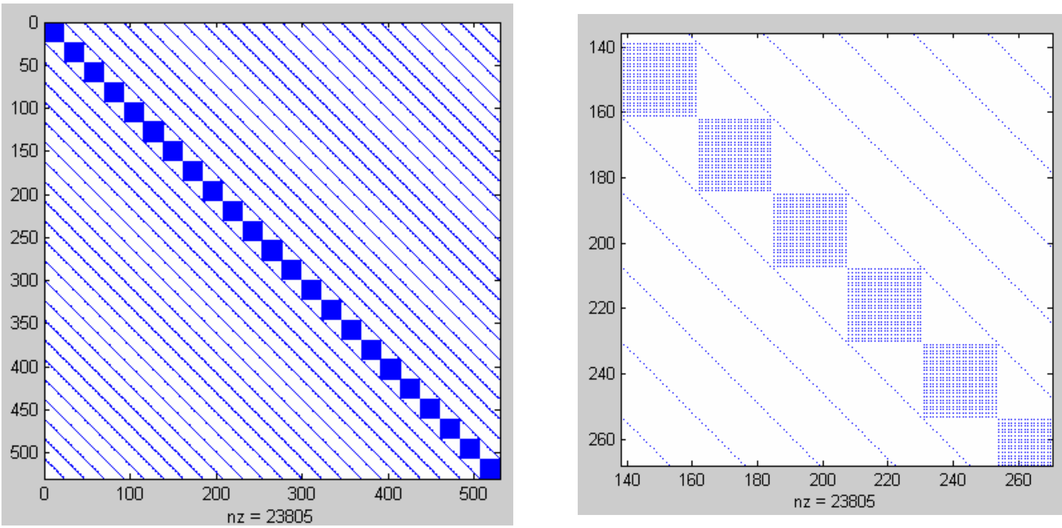


Figure 9: sparsity pattern of $L = I \otimes \tilde{D}_N^2 + \tilde{D}_N^2 \otimes I$

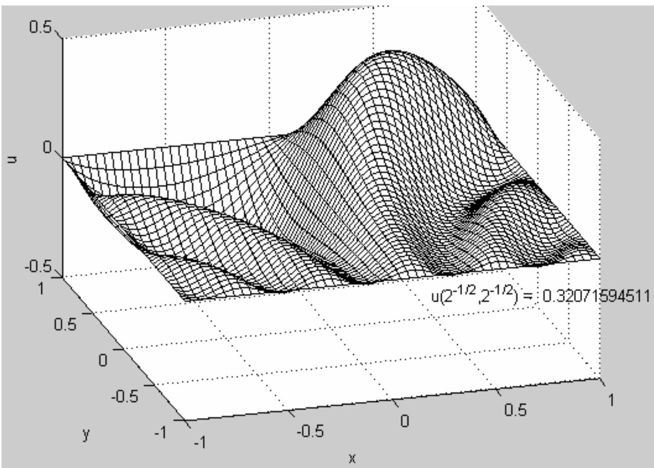


Figure 10: solution of Poisson equation (Eq. 3) under Chebyshev node.

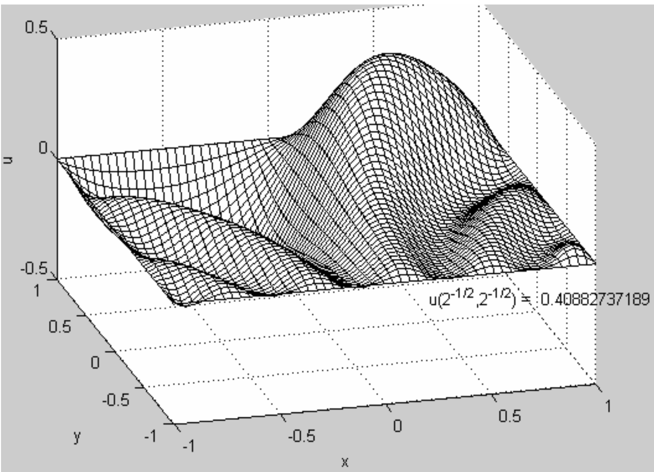


Figure 11: solution of Poisson equation (Eq. 3) under uniform node.

Question 4 (exercise 7.6): matrix $L = I \otimes \tilde{D}_N^2 + \tilde{D}_N^2 \otimes I$ is sparse, compare timing ($L \setminus f$) for dense and sparse matrix.

Table 3: black color, PC with MATLAB6, red color:quartet1 with MATLAB 7, green color:octet1. In quartet1 and octet1, MATLAB use parallel computing.

	$N = 24$	$N = 32$	$N = 64$	$N = 128$
Dimension of matrix	576	1024	4096	
nonzero	23805	58621	496125	
Dense	0.141 s 0.009253 s	0.641 s 0.043967 s	36.312 s 1.707721 s	68.573649 s
sparse	0.516 s 0.017739 s	2.718 s 0.060569 s	173.875 s 1.652521 s	72.938273 s
[L,U,P,Q] = lu(L)	0.019222 s	0.074058 s	0.487569 s	89.854222 s

Results show "dense matrix inversion is faster than sparse matrix inversion".

Example 5 (Helmhotz equation):

(Eq. 4) $u_{xx} + u_{yy} + w^2 u = f(x, y), \quad -1 < x, y < 1, \quad u = 0$ on the boundary.

Where $f(x, y) = \exp\left(-10(y-1)^2\right)\exp\left(-10(x-1/2)^2\right)$ and $w = 9$

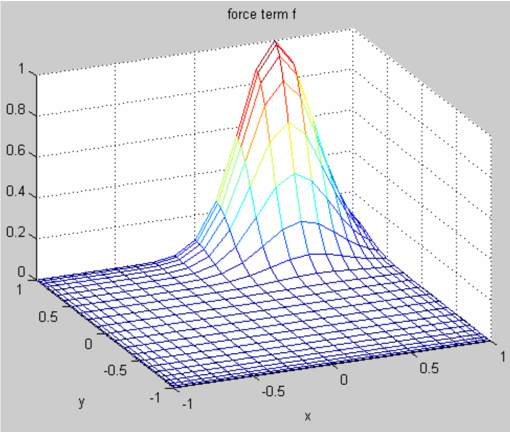


Figure 12: support of source term $f(x,y)$ is near boundary.

Note that for eigen-value problem

(Eq. 5) $u_{xx} + u_{yy} + w^2 u = \lambda u$, $-1 < x, y < 1$, $u = 0$ on the boundary.

We have $\psi_{k,m} = \sin\left(\frac{k\pi}{2}(x+1)\right)\sin\left(\frac{m\pi}{2}(y+1)\right)$, $\|\psi_{k,m}\|_{L^2} = 1$ for $k, m = 1, 2, \cdots$, and correspond eigenvalue $\lambda_{k,m} = w^2 - \frac{\pi^2}{4}\left(m^2 + k^2\right) \neq 0$ under $w = 9$.

If we write solution of (Eq. 4) as $u = \sum_{k,m=1}^{\infty} a_{k,m}\psi_{k,m}$ and $f = \sum_{k,m=1}^{\infty} \hat{f}_{k,m}\psi_{k,m}$, then we have

(Eq. 6) $\sum_{k,m=1}^{\infty} \lambda_{k,m} a_{k,m}\psi_{k,m} = \sum_{k,m=1}^{\infty} \hat{f}_{k,m}\psi_{k,m}$

Under Orthogonal property of basis $\psi_{k,m}$, we have $a_{k,m} = \frac{1}{\lambda_{k,m}}\hat{f}_{k,m}$ and $\|u\|_{L^2} = \sum_{k,m=1}^{\infty} |a_{k,m}|^2$, we can find dominant component $\psi_{k,m}$ (that is, largest $a_{k,m}$) to determine behavior of solution u.

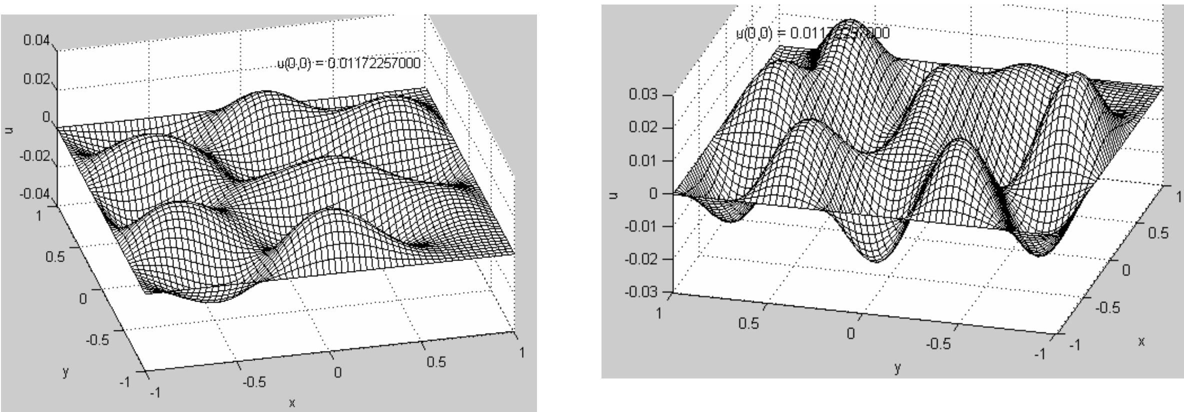


Figure 13: solution u, from left panel, one can see there are 3 half-wavelength in the x-direction and from right panel, one can see there are 5 half-wavelength in y-direction.

(1) $a_{k,m} = \frac{1}{\lambda_{k,m}}\hat{f}_{k,m}$, if $\lambda_{k,m} \approx 0$, then amplification is large, in fact, when $\lambda_{k,m} \approx 0$ we call the system is near resonant in physics. In our example,

$$\lambda_{k,m} = 9^2 - \frac{\pi^2}{4}\left(m^2 + k^2\right) = \frac{\pi^2}{4}\left(32.82 - \left(m^2 + k^2\right)\right).$$

In order to find (m,k) such that $\lambda_{k,m} \approx 0$, it suffices to find (m,k) such that $\left(m^2 + k^2\right) \approx 32.82$. $(m,k) = (3,5),(5,3)$ is what we want.

Table 4: first number is $\hat{f}_{k,m}$, second number is $\lambda_{k,m}$ and third number is $a_{k,m}$. We use Trapezoid rule to calculate $\hat{f}_{k,m}$ with grids 20000.

k\m	1	2	3	4	5	6
1	2.82E-02 76.07 3.70E-04	-4.99E-02 68.66 -7.27E-04	6.16E-02 56.33 1.09E-03	-6.36E-02 39.05 -1.63E-03	5.88E-02 16.85 3.49E-03	-5.09E-02 -10.29 4.95E-03
2	-3.31E-02 68.66 -4.83E-04	5.87E-02 61.26 9.59E-04	-7.26E-02 48.92 -1.48E-03	7.49E-02 31.65 2.37E-03	-6.92E-02 9.45 -7.33E-03	5.99E-02 -17.70 -3.39E-03
3	1.73E-02 56.33 3.08E-04	-3.07E-02 48.92 -6.28E-04	3.79E-02 36.59 1.04E-03	-3.91E-02 19.31 -2.03E-03	3.62E-02 -2.89 **-1.25E-02**	-3.13E-02 -30.03 1.04E-03
4	-2.19E-04 39.05 -5.61E-06	3.88E-04 31.65 1.23E-05	-4.79E-04 19.31 -2.48E-05	4.95E-04 2.04 2.42E-04	-4.57E-04 -20.16 2.27E-05	3.96E-04 -47.30 -8.37E-06
5	-6.14E-03 16.85 -3.65E-04	1.09E-02 9.45 1.15E-03	-1.34E-02 -2.89 4.65E-03	1.39E-02 -20.16 -6.88E-04	-1.28E-02 -42.37 3.03E-04	1.11E-02 -69.51 -1.60E-04
6	4.32E-03 -10.29 -4.19E-04	-7.65E-03 -17.70 4.32E-04	9.45E-03 -30.03 -3.15E-04	-9.75E-03 -47.30 2.06E-04	9.01E-03 -69.51 -1.30E-04	-7.81E-03 -96.65 8.08E-05

From above table, we know dominant mode is $(m,k) = (3,5)$. Moreover we sweep $k,m = 1,2,\cdots,100$ and $(m,k) = (3,5)$ is also dominant.

Question 5 (exercise 7.7): for each $w = 1:20$, check dominant mode for each w.

The dominant mode is shown in order.

k	1 (*)	2	3	4 (*)
$(k,m,a_{k,m})$	(1,1) 7.1557E-03 (2,1) 4.4019E-03 (2,2) 3.1346E-03	(1,1) 3.0120E-02 (2,1) 5.9860E-03 (1,2) 3.9751E-03	(2,1) 1.4955E-02 (1,2) 9.9313E-03 (1,1) 6.9261E-03	(2,2) 1.5709E-02 (2,1) 1.3624E-02 (1,2) 9.0474E-03

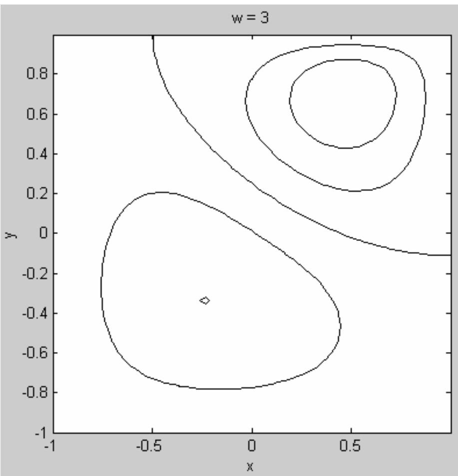
k	5	6 (*)	7	8

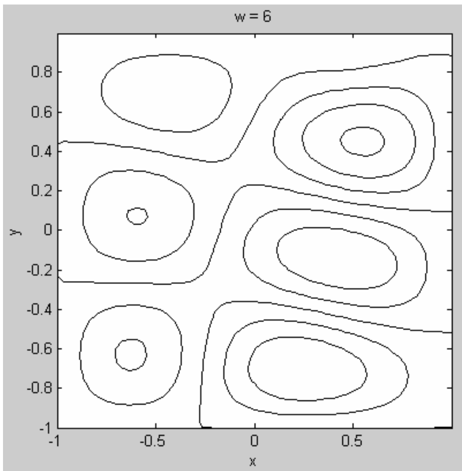
$(k, m, a_{k,m})$	(3,1) 1.8910E-01 (1,3) 5.3152E-02 (2,2) 1.1166E-02	(3,2) 1.8491E-02 (4,1) 1.0698E-02 (2,3) 7.8268E-03	(4,2) 2.1512E-01 (4.1) 9.0168E-03 (3,3) 8.2705E-03	(5,1) 3.8569E-01 (1,5) 4.0296E-02 (4,3) 1.6908E-02

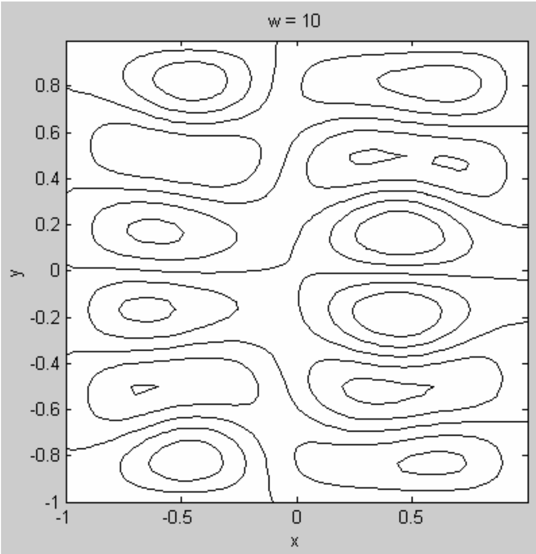
k	9	10	11	12
$(k, m, a_{k,m})$	(5,3) 1.2511E-02 (5,2) 7.3260E-03 (6,1) 4.9455E-03	(6,2) 4.5953E-02 (4,5) 1.1926E-02 (2,6) 5.8677E-03	(7,1) 1.8059E-02 (5,5) 5.4113E-03 (7,2) 5.1553E-03	(7,3) 2.9571E-02 (7,2) 3.8086E-03 (3,7) 2.8866E-03

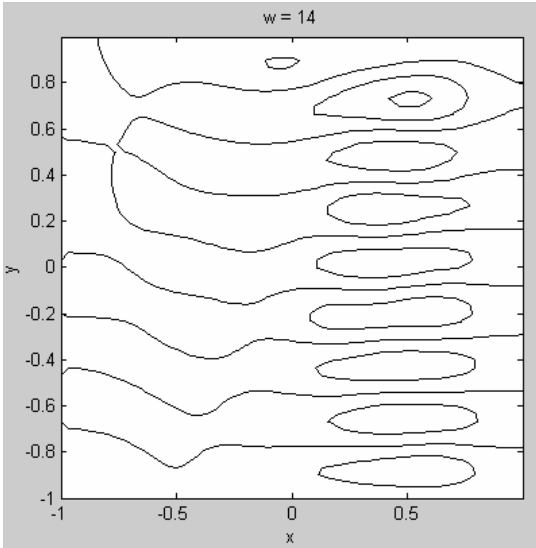
k	13	14 (*)	15	16
$(k, m, a_{k,m})$	(8,2) 3.4738E-02 (8,1) 4.1663E-03 (8,3) 1.9872E-03	(9,1) 4.8320E-03 (9,2) 2.6210E-03 (8,2) 1.4979E-03	(9,3) 6.4123E-03 (9,2) 2.3563E-03 (8,5) 1.4503E-03	(10,2) 5.1297E-02 (10,1) 3.9120E-03 (10,3) 1.2630E-03

k	17	18	19 (*)	20
$(k, m, a_{k,m})$	(9,6) 1.4924E-02 (6,9) 2.7783E-03 (11,1) 1.9593E-03	(11,3) 4.4771E-03 (11,2) 1.7802E-03 (11,1) 1.0252E-03	(11,5) 6.7666E-03 (12,1) 6.5753E-03 (12,2) 5.9812E-03	(9,9) 1.8646E-03 (13,1) 9.9413E-04 (13,2) 8.4766E-04









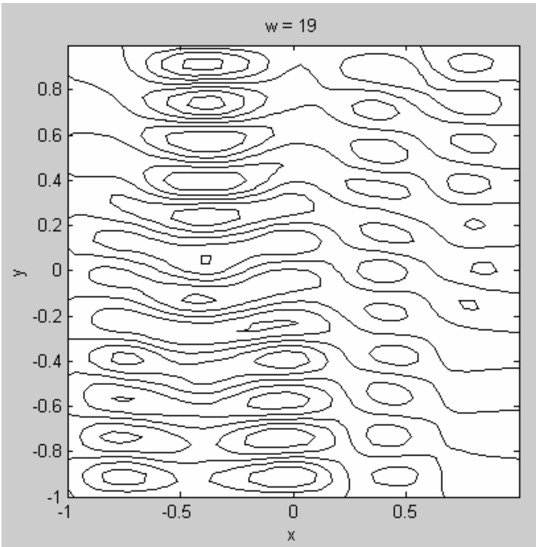


Figure 14: w = 19, it is close to (m,k) = (12,1)

Question 6 (exercise 7.1): use **polyval** and **polyfit** to do interpolation is unstable, we will show this in **exercise 7.2.** Here we provide another interpolation technique, called barycentric interpolation. Given sample points $x_0, x_1, \cdots, x_N$ and function value $u_0, u_1, \cdots, u_N$, we define interpolation function $p(x)$ with respect to u_j as

(Eq. 7) $$p(x) = \sum_{j=0}^{N} \frac{a_j^{-1} u_j}{x - x_j} / \sum_{j=0}^{N} \frac{a_j^{-1}}{x - x_j} \quad \text{for } a_j = \prod_{k=0, k\neq j}^{N} (x_j - x_k)$$

Note that $p(x_j) = u_j$ under such construction but when we do program, we must avoid $0/0$, hence we modify (Eq. 7) to $p(x) = \sum_{j=0}^{N} \frac{a_j^{-1} u_j}{x - x_j + \varepsilon} / \sum_{j=0}^{N} \frac{a_j^{-1}}{x - x_j + \varepsilon}$, see

F:\course\2008spring\spectral_method\matlab\barycentric_interp.m

Remark 1: we cannot choose $a_j = 1$, or highly oscillatory result appears,

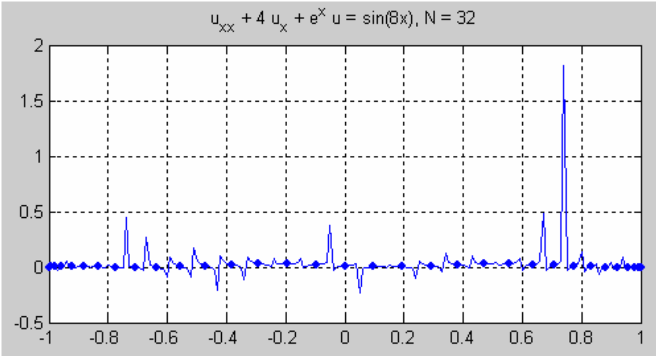


Figure 15: result of interpolation with $a_j = 1$.

Remark 2: if we choose $p(x) = \sum_{j=0}^{N} \frac{u_j}{|x - x_j|} / \sum_{j=0}^{N} \frac{1}{|x - x_j|}$, then result is also not good,

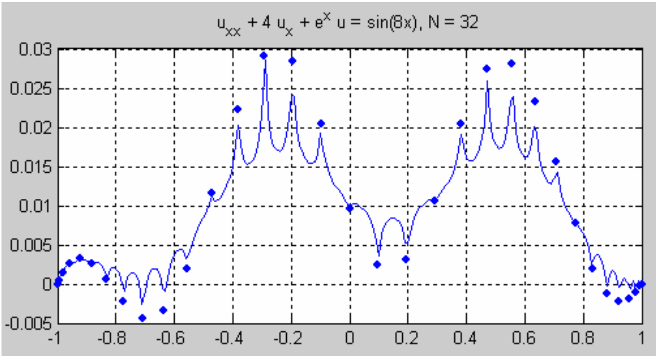


Figure 16: result of interpolation with weight $\frac{1}{|x - x_j|}$.

Question 7 (exercise 7.2): Solve $u_{xx} + 4u_x + e^x u = \sin(8x)$ numerically on $[-1,1]$ with B.C. $u(\pm 1) = 0$. To ten digits of accuracy, what is $u(0)$?

<sol> we modify program 13 to $\left(\tilde{D}_N^2 + 4\tilde{D}_N + diag\left(e^x\right)\right)u = \sin(8x)$, see F:\course\2008spring\spectral_method\matlab\chap7_ex2.m

Since we don't have analytic solution, so we refine $N = 8, 16, 32, 64$ and compare consecutive solution u, that is measure $\|u_{16} - u_8\|_\infty$, $\|u_{32} - u_{16}\|_\infty$, $\|u_{64} - u_{16}\|_\infty$. The same reason, we measure convergence of $u(0)$ by comparing consecutive solution.

Table 5: use chebyshev node, $u(0) = 0.0095978572295$

$\|u_{16} - u_8\|_\infty$, $\|u_{16}(0) - u_8(0)\|$	$\|u_{32} - u_{16}\|_\infty$, $\|u_{32}(0) - u_{16}(0)\|$	$\|u_{64} - u_{16}\|_\infty$, $\|u_{64}(0) - u_{32}(0)\|$
7.3554391255914488E-003 0.00735543912559	3.6934512882141890E-007 3.693451288214189e-007	4.0939474033052647E-016 1.960237527853792e-016

	$n = 8$	$n = 16$	$n = 32$	$n = 64$
$cond(L)$	6.6353E+001	9.7383E+002	1.5248E+004	2.4264E+005

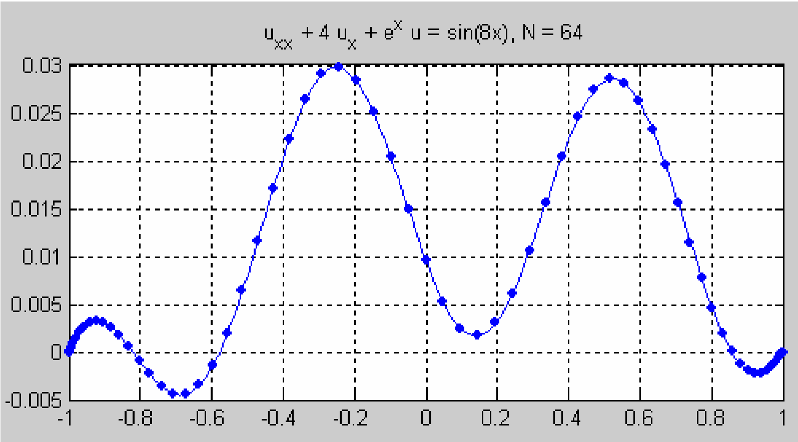


Figure 17: Chebyshev node

Remark 3: when $n=128$, we have $\|u_{128}-u_{64}\|_{\infty}=1.02\text{E-}015$ under Chebyshev node, but we we use **polyval** and **polyfit** to interpolation, MATLAB shows warning message

Warning: Polynomial is badly conditioned. Remove repeated data points
or try centering and scaling as described in HELP POLYFIT.

Moreover the interpolation is almost wrong, see Figure 18.

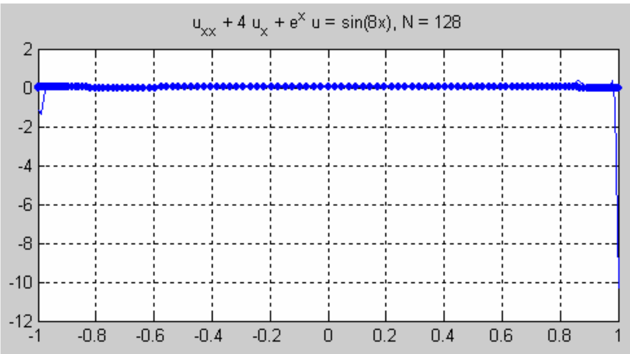


Figure 18: unstable of **polyval** and **polyfit**

We change to use **barycentric interpolation** techniques as in exercise 1, then the plot is good, see Figure 19.

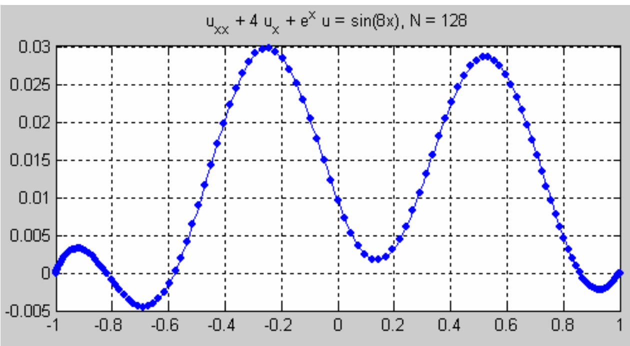


Figure 19: **barycentric interpolation** is stable.

Table 6: use uniform node, result is not good since matrix is close to singular.

$\|u_{16}-u_8\|_\infty$	$\|u_{32}-u_{16}\|_\infty$	$\|u_{64}-u_{16}\|_\infty$
8.6357502359060784E-002	3.9598660371845465E-004	3.5568806739495989E-001

$u_8(0)$	$u_{16}(0)$	$u_{32}(0)$	$u_{64}(0)$
-7.71556344E-002	9.20186792E-003	9.59785452E-003	-8.61064112E-003

	$n=8$	$n=16$	$n=32$	$n=64$
$cond(L)$	3.7221E+002	5.5751E+006	5.5171E+015	4.3762E+032

As you see in Figure 20, when $n=64$, large condition number cause wrong result, the disadvantage of uniform node is bad condition of differential matrix, except we do iterative refinement.

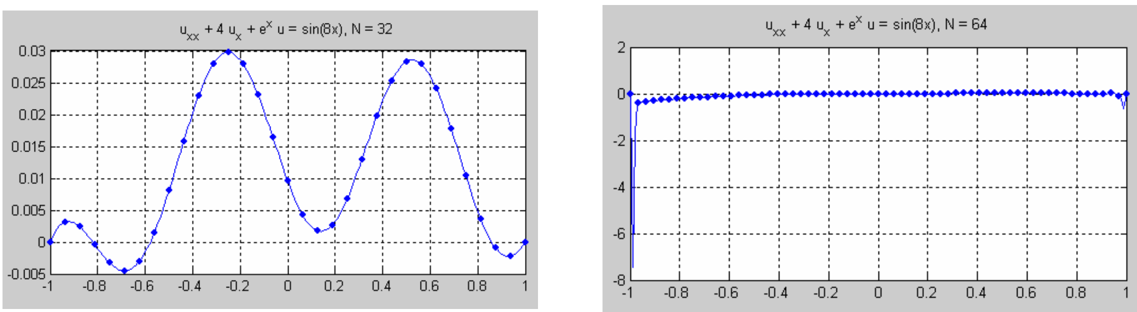


Figure 20: uniform node, plot by barycentric interpolation. Right panel is result of $n=64$, it is wrong since condition number is too large such that we have no accuracy.

Next we have a conjecture "chebyshev node results in small condition number", in order to check this assertion, we use change of variable, define $x=\cos t$, then $x\in[-1,1]$ implies $t\in[0,\pi]$.

$$u_x=\frac{du}{dt}\frac{dt}{dx}=\frac{-1}{\sin t}\frac{du}{dt}$$

$$u_{xx}=\frac{du_x}{dx}=\frac{du_x}{dt}\frac{dt}{dx}=\frac{-1}{\sin t}\frac{du_x}{dt}=\frac{-1}{\sin t}\frac{d}{dt}\left(\frac{-1}{\sin t}\frac{du}{dt}\right)=\frac{1}{\sin^2 t}\left(\frac{d^2u}{dt^2}-\frac{\cos t}{\sin t}\frac{du}{dt}\right)$$

Then $u_{xx}+4u_x+e^x u=\sin(8x)$ would leads to

(Eq. 8) $$u_{tt}-\left(\frac{\cos t}{\sin t}+4\sin t\right)u_t+\sin^2(t)e^{\cos t}u=\sin^2(t)\sin(8\cos t)$$

	$n=8$	$n=16$	$n=32$	$n=64$
$cond(L)$	5.1545E+002	7.3152E+006	6.8008E+015	

$\|u_{16} - u_8\|_\infty$	$\|u_{32} - u_{16}\|_\infty$	$\|u_{64} - u_{16}\|_\infty$
8.3872104019244165E-001	1.3347557971704758E+000	

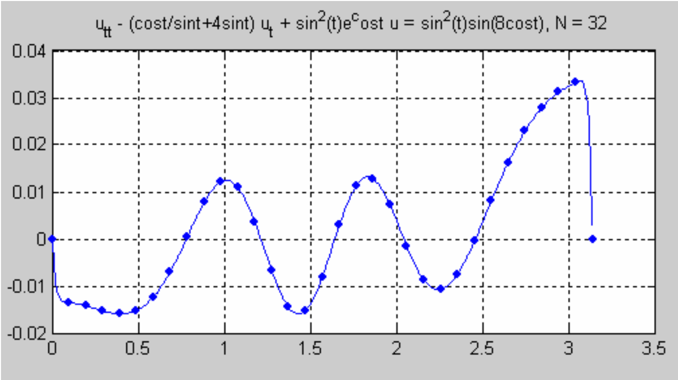


Figure 21: method of change of variable does not work.

Reference

[1] John H. Mathews, Numerical Methods Using Matlab, 3rd edition.