# Boundary Layer Problems and Applications of Spectral Methods

Sumi Oldman
University of Massachusetts Dartmouth
CSUMS Summer 2011

August 4, 2011

**Abstract**

Boundary layer problems arise when thin boundary layers need to be approximated. When the width of the boundary layer is width $O(\epsilon_N)$, as the boundary layers get smaller ($\epsilon$ is decreasing), the number of collocation points, or nodes $(N)$ need to be increased so that the approximation to the solution can be found at the specific boundaries. This paper deals with different methods to approximate thin boundary layers. As the boundary layer gets thinner, the number of points must be concentrated at the boundaries, and depending on the method, there needs to be an increase in collocation points. As the three methods are explored, there is a higher concentration of collocation points at the boundaries, and therefore the accuracy of the approximations increase drastically.

## 1 Introduction

### 1.1 Proposed Research

Using spectral collocation, determine a more efficient way to execute boundary layer problems using a reduced basis method.

### 1.2 Background

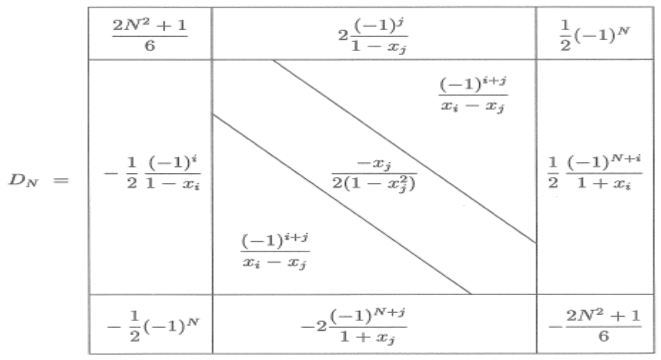
The collocation method chooses a finite-dimensional space of polynomials up to a certain degree and a number of points in the domain. These are referred to as collocation points. Chebyshev collocation tends to be more efficient than finite difference methods, since the points are concentrated at the boundaries. Because a large number of nodes $(N)$ is still needed to obtain an accurate solution when $\epsilon$ is significantly small, a reduced basis method can be implemented to try and predict how the boundary layer changes with several known parameters.

To understand the applications of spectral methods, different examples were done and error tables were duplicated to make sure that the results matched the given problem. Boundary layer problems involve a differentiation matrix. Multiplication by a first-order Chebyshev differentiation matrix can transform a vector of data at the collocation points into approximate derivations at those points. In order to form a differentiation matrix, the function is represented as a polynomial, and the derivative of each polynomial is calculated to form the differentiation matrix. For the first examples, a Chebyshev grid was implemented. The grid used the points $x_j = cos\frac{j\pi}{N} \quad (0 \leq j \leq N)$, and was projected onto the interval $[-1, 1]$ of equally spaced points along the unit circle.

For each $N \geq 1$, let the rows and columns of the $(N+1) \times (N+1)$ Chebyshev spectral differentiation matrix $D_N$ be indexed from 0 to $N$.
Theorem 1:

$$\begin{aligned}
(D_N)_{00} &= \frac{2N^2+1}{6}, \quad (D_N)_{NN} = -\frac{2N^2+1}{6} \\
(D_N)_{jj} &= \frac{-x_j}{2(1-x_j^2}, \quad j = 1, \ldots, N-1 \\
(D_N)_{ij} &= \frac{c_i}{c_j}\frac{(-1)^{i+j}}{(x_i - x_j)}, \quad i \neq j, \quad i, j = 0, \ldots, N, \\
c_i &= \begin{cases} 2, & \text{if } i = 0 \text{ or } N, \\ 1, & \text{otherwise} \end{cases}
\end{aligned}$$

An image of the differentiation matrix is another way of understand how the matrix is formed[1]:


$D_N =$
$\frac{2N^2+1}{6}$
$2\frac{(-1)^j}{1-x_j}$
$\frac{1}{2}(-1)^N$
$\frac{(-1)^{i+j}}{x_i - x_j}$
$-\frac{1}{2}\frac{(-1)^i}{1-x_i}$
$\frac{-x_j}{2(1-x_j^2)}$
$\frac{1}{2}\frac{(-1)^{N+i}}{1+x_i}$
$\frac{(-1)^{i+j}}{x_i - x_j}$
$-\frac{1}{2}(-1)^N$
$-2\frac{(-1)^{N+j}}{1+x_j}$
$-\frac{2N^2+1}{6}$

The $jth$ column of $D_N$ contains the derivative of the dgree $N$ polynomial interpolant $p_j(x)$ to the delta function supported at $x_j$ sampled at the grid points $x_i$.

When a boundary layer is of width $\epsilon$, where $\epsilon \ll 1$, the number of grid points must be increased as $\epsilon \to 0$. Computationally, this can become extremely expensive, so eventually reduced basis method will be used to simplify this process.

The differentiation matrix was computed using the following Matlab code:

```
function [D,x]= chebsumi(N)
```

[1]Trefethen, Lloyd N. 2001. *Spectral Methods in Matlab.* Oxford, England. Oxford University. Society for Industrial and Applied Mathematics.

```
%creates differentiation matrix
if N==0, D=0; x=1; return, end
x = cos(pi*(0:N)/N)';
c = [2;ones(N-1,1);2].*(-1).^(0:N)';
X = repmat(x,1,N+1);
dX= X-X';
D=(c*(1./c)')./(dX+(eye(N+1)));
D= D - diag(sum(D'));
```

## 2 Chebyshev Collocation Method

### 2.1 Methods and Materials

A direct application of Chebyshev collocation method used a code (written with Matlab) to compute the numerical solution of the following problem with small parameter $\epsilon_N$. Using a small $\epsilon$, the following has variable coefficients and the solution develops two boundary layers near the boundaries. The $u's$ in the equation are the unknowns and we want the solution to satisfy the equation at the collocation points. The equation for this application follows:

$$\epsilon_N u''(x) - xu'(x) - u = (c_+(x) - 1)e^{-c_+(x)} - 2(c_-(x) + 1)e^{c_-(x)}$$

and boundary condition are $u(-1) = 1$, $u(+1) = 2$, where $c \pm (x) = (x \pm 1)/\epsilon_N$

We have the polynomial

$$\epsilon_N u''(x) - xu'(x) - u = (c_+(x) - 1)e^{-c_+(x)} - 2(c_-(x) + 1)e^{c_-(x)}$$

Where $u$ is unknown. With a set of polynomials, $u$ is rewritten as a vector $\vec{u}$. Using the differentiation matrix, we can solve for $\vec{u'}$ and $\vec{u''}$:

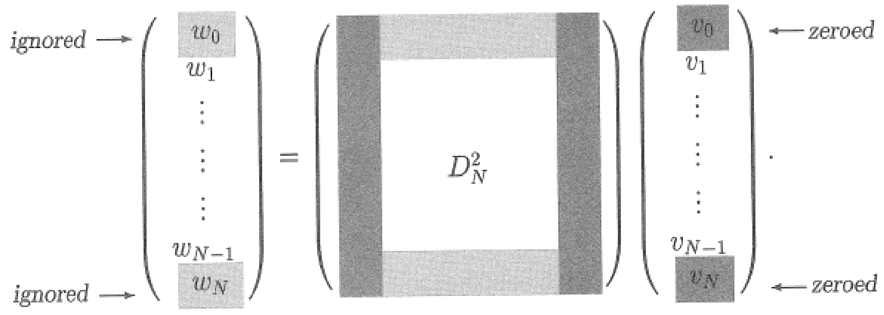
$$\begin{aligned} \vec{u'} &= D\vec{u} \\ \vec{u''} &= D^2 \end{aligned}$$

So the above equation can be written in terms of $D$ and we end up with a linear equation: $V\vec{u} = \vec{f}$

We set up the following array to solve for the values of $u(x)$

$$(LHS)\begin{pmatrix} 0 \\ u_1 \\ \vdots \\ u_N \\ 0 \end{pmatrix} = \begin{pmatrix} f(x_0) \\ \vdots \\ f(x_N) \end{pmatrix}$$

The LHS of the equation is originally a $(N + 1) \times (N + 1)$ matrix formed from the ODE and Differentiation Matrix. Given boundary conditions $u(-1) = u(1) = 0$, the first and last entries Because the first and last entries of the $\mathbf{X}$ are zero, the first and last columns, and the first and last rows become zero. Another image may help the reader to understand how the matrix is "trimmed"[2].



The LHS of the equation is originally a $(N+1)\times(N+1)$ matrix formed from the ODE and Differentiation Matrix. Where $-1 \le u \le 1$. Given the initial boundary conditions, $u(-1) = 1$, $u(1) = 2$ we have the following:

$$\begin{pmatrix} \vec{v_1} & \vec{v_2} & \dots & \vec{v_{N+1}} \end{pmatrix} \begin{pmatrix} 1 \\ u_2 \\ u_3 \\ \vdots \\ u_N \\ 2 \end{pmatrix} = \begin{pmatrix} \vec{f} \end{pmatrix}$$

$u_1$ and $u_{N+1}$ are known, so they can be moved to the RHS of the equation.

The equation is then be rearranged

$$\vec{v_1}u_1 + \vec{v_2}u_2 + \cdots + \vec{v_N}u_N + \vec{v_{N+1}}u_{N+1} = \vec{f}$$

$$\vec{v_2}x_2 + \cdots + \vec{v_N}x_N = \vec{f} - 2\vec{v_1} - 1\vec{v_{N+1}}$$

$$V(\text{"}trimmed\text{"}) \begin{pmatrix} u_2 \\ u_3 \\ \vdots \\ u_N \end{pmatrix} = \vec{f} - 2\vec{v_1} - 1\vec{v_{N+1}}$$

Matlab will solve the equation for all possible values of $\vec{u}$ from $u_2$ to $u_N$

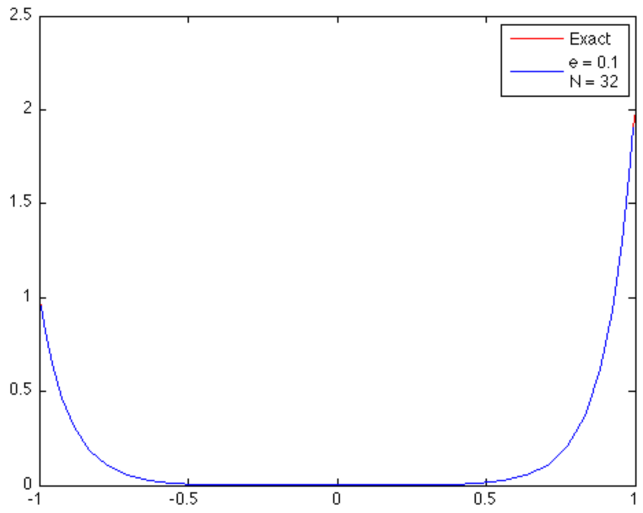
The code in Matlab involves the differentiation matrix $D$ and the following function:

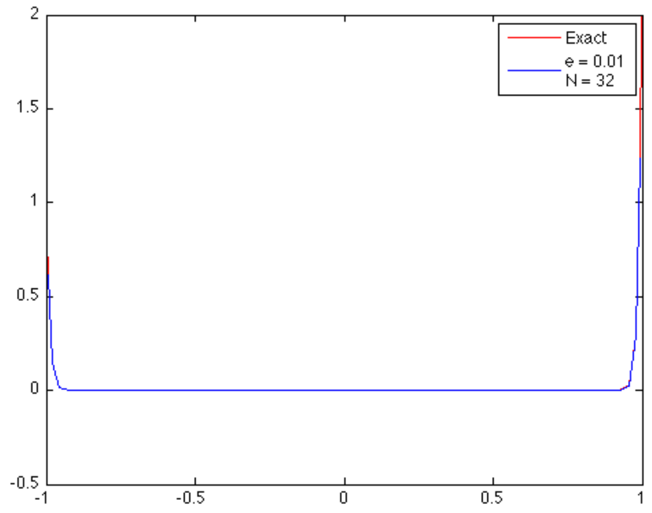
[2]Trefethen, Lloyd N. 2001. *Spectral Methods in Matlab*. Oxford, England. Oxford University. Society for Industrial and Applied Mathematics.

```
function [uapprox]=BLPsumi(e, N)
[D,x]=chebsumi(N);
D2=D*D;
OldLHS = e*D2-diag(x)*D-eye(N+1,N+1);
v1 = OldLHS(2:N,1);
v2 = OldLHS(2:N,N+1);
LHS = e*D2(2:N,2:N)-diag(x(2:N))*D(2:N,2:N)-eye(N-1,N-1);
cp=(x+1)/e;
cm = (x-1)/e;
RHS= (cp-1).*exp(-cp)-2*(cm+1).*exp(cm);
trueRhs = RHS(2:N) - 2*v1 - v2;
uapprox = LHS\trueRhs;
plot(x(2:N),uapprox)
```

## 2.2 Results

The following graphs show how the Chebyshev collocation method becomes more accurate as the number of nodes increase at the boundaries, and as epsilon gets small. The Chebyshev collocation method is compared with the exact solution that appears in the graph as the red line. The problem with this method is explained below in the table that follows.





The following table contains the maximum error for $\epsilon = 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}$:

Table 1: Maximum error for changing $\epsilon$ and $N$

| $N$ | $\epsilon = 10^{-2}$ | $\epsilon = 10^{-3}$ | $\epsilon = 10^{-4}$ | $\epsilon = 10^{-5}$ |
|---|---|---|---|---|
| 32 | $2.900e-3$ | $3.354e+0$ | $6.181e+1$ | $6.954e+2$ |
| 64 | $3.517e-10$ | $2.540e-1$ | $2.599e+1$ | $3.237e+2$ |
| 128 | $3.352e-14$ | $1.593e-14$ | $6.440e+0$ | $1.516e+2$ |
| 256 | $6.460e-15$ | $1.643e-14$ | $1.291e-1$ | $5.833e+1$ |

As $\epsilon$ gets very small, and although there is a large number of nodes ($N$), the error is still very large. The next method implemented attempts to reduce that error even more, given a small $\epsilon$ and "not so large" $N$.

## 3 Transformation of Singularly Perturbed Linear BVP

### 3.1 Methods and Materials

Using a variable transformation of a singularly perturbed linear BVP, we can resolve very thin boundary layers using a "not too large" $N$. The variable transformation will be $x \to y(x)$ into the new BVP:

$$\epsilon v''(y) + P(y)v'(y) + Q(y)v(y) = F(y)$$

where $v$ is the transplant of $u$, $v(y) = u(x(y))$. The transformed coefficients are:

$$\begin{aligned} P(y) &= \frac{p(x)}{y'(x)} + \epsilon \frac{y''(x)}{y'(x)^2} \\ Q(y) &= \frac{q(x)}{y'(x)^2} \\ F(y) &= \frac{f(x)}{y'(x)^2} \end{aligned}$$

Keep in mind quantities $1/y'(x)$ and $y''(x)/y'(x)^2$.

Iterated SINE fcuntions are now introduced $x = g_m(y), \quad m = 0, 1, 2, ...$, where

$$g_0(y) := y, \quad g_m(y) = sin\left(\frac{\pi}{2} g_{m-1}(y)\right), \quad m \geq 1$$

Having the transformation $x(y) = g_m(y)$ the transformed coefficients can be computed

$$g_0'(y) = 1, \quad g_m'(y) = \frac{\pi}{2} cos\left(\frac{\pi}{2} g_{m-1}(y)\right) g_{m-1}'(y), \quad m \geq 1$$

Since $y'(x) = 1/g_m'(y)$ the following Lemma is introduced:

Lemma 1:

$$\frac{1}{y'(x)} = \prod_{k=0}^{m-1} \left(\frac{\pi}{2} cos\left(\frac{\pi}{2} g_{m-1}(y)\right)\right), \quad m \geq 1$$

Further, the function $h_m(x)$ is defined, mapping $[-1, 1]$ onto itself:

$$h_0(x) := x, \quad h_m(x) = \frac{2}{\pi} arcsin(h_{m-1}(x)), \quad m \geq 1$$

It is then showed that $h_m = g_m^{-1}$. For $m \geq 1$ we let $z = h_m(g_m(y))$ and through induction for $k = 0, \ldots, m$

$$g_k(z) = h_{m-k}(g_m(y))$$

For $k = m$ we obtain

$$g_m(z) = h_0(g_m(y)) = g_m(y)$$

Since $g_m$ is injective, it follows that $y = z$

Then, we find a recursion for the quantity $h_m''(x)/[h_m'(x)]^2$

From Equation 1, we obtain:

$$sin\left(\frac{\pi}{2} h_m(x)\right) = h_{m-1}(x), \quad m \geq 1$$

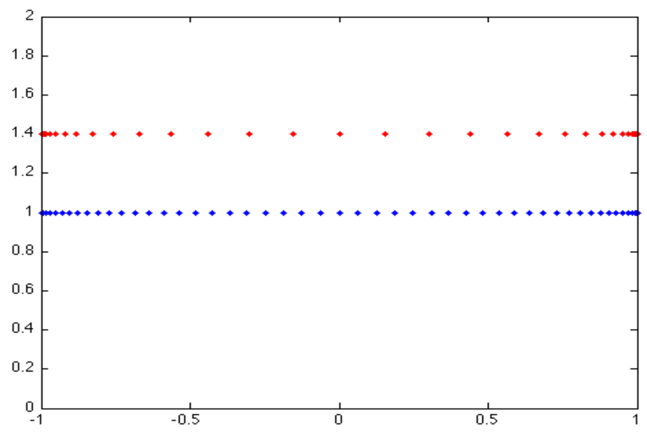
$$\begin{aligned} \frac{\pi}{2} cos\left(\frac{\pi}{2} h_m'(x)\right) h_m'(x) &= h_{m-1}'(x) \\ -\left(\frac{\pi}{2}\right)^2 sin\left(\frac{\pi}{2} h_m(x)\right) (h_m'(x))^2 + \left(\frac{\pi}{2}\right) cos\left(\frac{\pi}{2} h_m(x)\right) h_m''(x) &= h_{m-1}''(x) \\ \frac{\pi}{2} tan\left(\frac{\pi}{2} h_m(x)\right) + \frac{\pi}{2} cos\left(\frac{\pi}{2} h_m(x)\right) \frac{h_{m-1}''(x)}{(h_{m-1}'(x))^2} &= \frac{h_m''(x)}{(h_m'(x))^2} \end{aligned}$$

Differentiating the previous recursive functions, a new set of points are formed. The Matlab code will compute the new coefficients $P(y), Q(y)$ and $F(y)$ to use in the transformation:

$$\epsilon v''(y) + P(y)v'(y) + Q(y)v(y) = F(y)$$

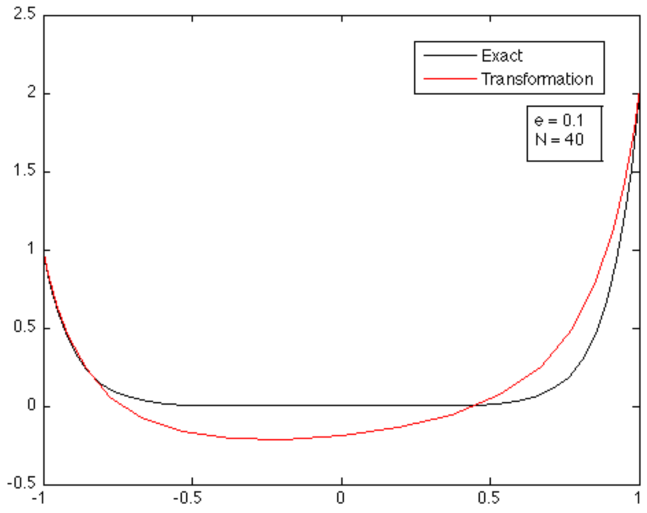
```
        P(i)=funp(x(i))*yprime(i)+e*hd(i);
        Q(i)=funq(x(i))*(yprime(i))^2;
        F(i)=funf(x(i), e)*(yprime(i))^2;
OldLHS2=e*D2+diag(P)*D;
V1=OldLHS2(2:N,1);
V2=OldLHS2(2:N,N+1);
LHS2=e*D2(2:N,2:N)+diag(P(2:N))*D(2:N,2:N)-diag(Q(2:N));
RHS2= F(2:N)'-2*V1-V2;
uapprox2=LHS2\RHS2;
```

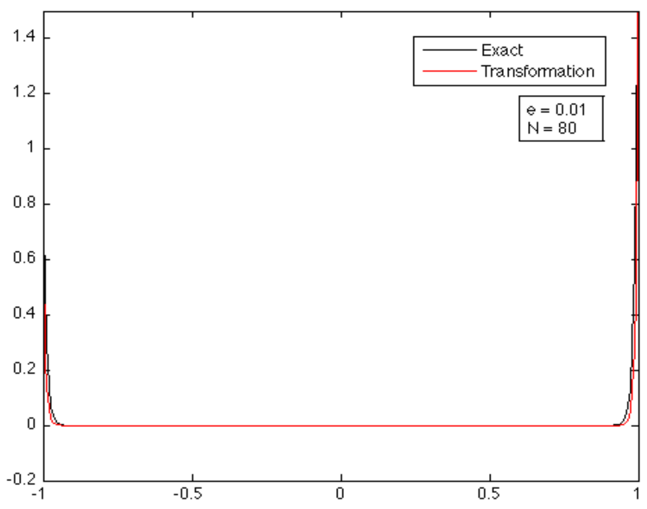
Transformation points (red) and Chebyshev points (blue). For the transformation, and iterated SINE function is introduced and the transformation is characterized based on the relative spacing of the transformed Chebyshev points. The new transformed points are even more closely spaced together at the boundaries. By using more points, more closely spaced at the boundaries, the new transformation should be able to deal with extremely small boundary layers with a relatively small number of collocation points.

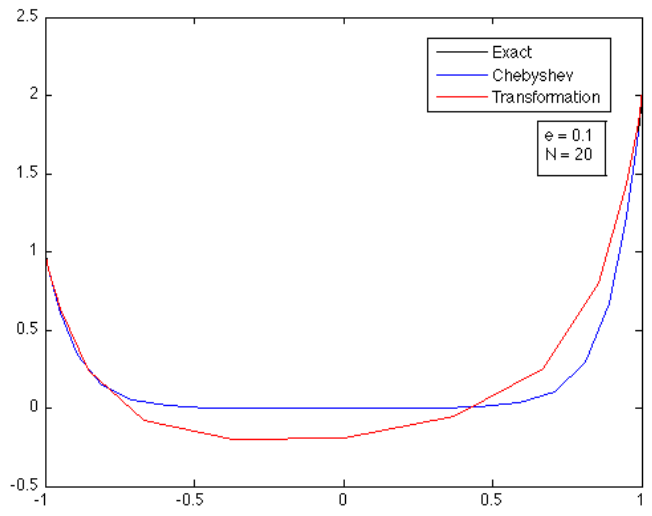


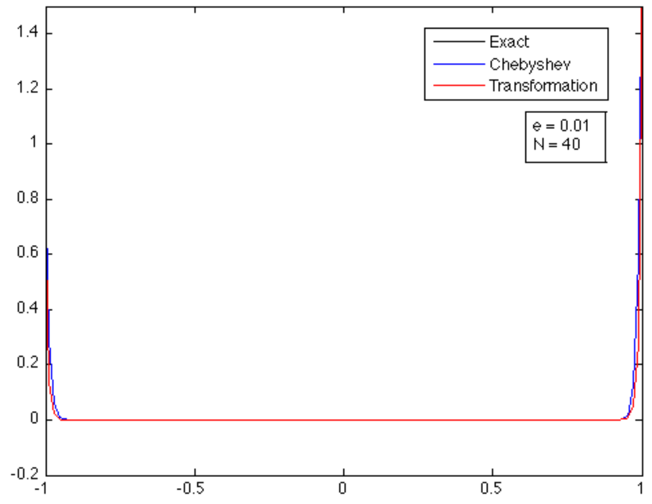
## 3.2 Results

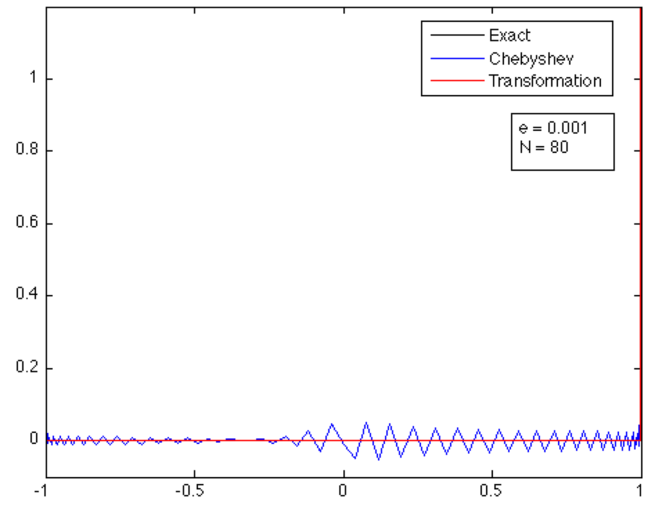
Five graphs that follow that show the difference between the exact solution and the transformation. The three last graphs show the Chebyshev solution, new transformation, and the exact solution for a given $\epsilon$ and $N$. For the last graph in the series, as $\epsilon$ get smaller than $10e-3$, it "blows up" in the center and can no longer be used in the results. The transformation still produces a somewhat accurate approximation, despite the small $\epsilon$ and fairly small number of nodes. The error table that follows will explain how well the transformation approximates the solution.


Exact
Transformation
e = 0.1
N = 40
2.5
2
1.5
1
0.5
0
-0.5
-1
-0.5
0
0.5
1


Exact
Transformation
e = 0.01
N = 80
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-1
-0.5
0
0.5
1


Exact
Chebyshev
Transformation
e = 0.1
N = 20


Exact
Chebyshev
Transformation
e = 0.01
N = 40



Results of the transformation: Suggests that with SINE iterations, very thin bound-

Table 2: Maximum error for changing $\epsilon$ and $N$

| $N$ | $\epsilon = 10^{-2}$ | $\epsilon = 10^{-3}$ | $\epsilon = 10^{-4}$ | $\epsilon = 10^{-5}$ |
|---|---|---|---|---|
| 32 | $9.192e-3$ | $8.022e-1$ | $8.698e-1$ | $9.328e-3$ |
| 64 | $34.346e-8$ | $6.214e-3$ | $8.552-1$ | $2.629e-1$ |
| 128 | $4.34e-13$ | $1.822e-10$ | $3.446e-5$ | $1.039e-1$ |
| 256 | $2.037e-10$ | $5.515e-11$ | $2.132e-11$ | $1.952e-6$ |

aries can be resolved with $N$ of the order of several hundreds

## 4 Nonlinear Burgers' Equation

### 4.1 Methods and Materials

Another example is using a nonlinear Burgers' equation:

$$\epsilon u''(x) + u(x)u'(x) = 0, \quad x \in [-1, 1]$$

This equation is transformed with the new variable $y = y(x)$ in the same way we transformed the previous equation. The transformed Burgers' equation is:

$$\epsilon v''(y) + \left[\frac{1}{y'(x)}v(y) + \epsilon\frac{y''(x)}{y'(x)^2}\right] v'(y) = 0$$

The solution to the Burgers' equation is computed with Newton's method (using $v \equiv 1$ as the initial guess, for small values of $\epsilon$.

We want $f(\vec{x_N})$ to be as close to zero as possible and we use the equation:

$$\vec{v_{N+1}} = \vec{v_N} - [F'(v_N)]^{-1} \cdot \vec{F}(v_N)$$

Where $[F'(v_N)]^{-1}$ is the Jacobian and $\vec{F}(v_N)$ is nonlinear.
Multiple steps were taken to compute the final results:

- Compute the Jacobian of $(v_N)$
- Compute nonlinear $\vec{v_N}$
- Set up Newton's method with Matlab

Step 1:

$$[F'(v_N)]^{-1}$$

is transformed into the Jacobian:

$$A = \epsilon D^2 + \left[\epsilon \frac{y''}{(y')^2}\right] \cdot D + \left[\frac{1}{y'}\right] \cdot I \cdot [D\vec{v}] + \left[\frac{1}{y'}\right] \cdot [\vec{v}] \cdot D$$

Step 2**:

$$\vec{F}(v_N)$$

is nonlinear and is transformed into

$$\epsilon D^2 v_N + \left(\frac{1}{y'} .* \vec{v_N}\right) .* (D\vec{v_N}) + \left(\epsilon \cdot \frac{y''(x)}{y'(x)^2}\right) .* (D\vec{v_N})$$

Step 2**: For the boundary conditions, we need to find $f(x)$ such that $\epsilon u'' + u \cdot u' = f$ has the exact solution $tan\left(\frac{x+1}{2\epsilon}\right)$. After solving for $f$ we get the following:

$$u' = \frac{1}{2\epsilon} sech^2\left(\frac{x+1}{2\epsilon}\right)$$

$$u'' = -\frac{sech^2\left(\frac{x+1}{2\epsilon}\right) tanh\left(\frac{x+1}{2\epsilon}\right)}{2\epsilon^2}$$

So our f equals:

$$f = \epsilon\left(-\frac{sech^2\left(\frac{x+1}{2\epsilon}\right) tanh\left(\frac{x+1}{2\epsilon}\right)}{2\epsilon^2}\right) + tan\left(\frac{x+1}{2\epsilon}\right) \cdot \left(\frac{1}{2\epsilon} sech^2\left(\frac{x+1}{2\epsilon}\right)\right)$$

To get the correct nonlinear solution for $\vec{F}(v_N)$, we need to subtract $f$ from the right hand side of the equation

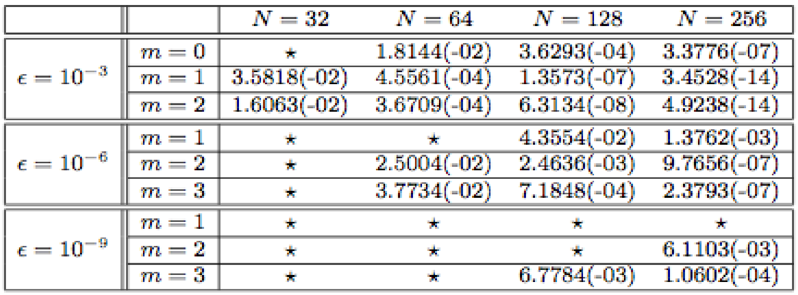
$$\vec{F}(v_N) = \epsilon D^2 v_N + \left(\frac{1}{y'} .* \vec{v_N}\right) .* (D\vec{v_N}) + \left(\epsilon \cdot \frac{y''(x)}{y'(x)^2}\right) .* (D\vec{v_N}) - \mathbf{f}$$

Step 3: Set up Newton's Method with Matlab
. . . To be continued . . .

### 4.2 Results

Although writing the full code in Matlab was not able to be completed before the end of the program, the results from evaluating the nonlinear Burger's method are as follows[3]:

TABLE 9.1.1. Maximum errors for Example 2 ('$\star$' indicates an error $> 1$ or convergence difficulties in the Newton process)



| | | $N = 32$ | $N = 64$ | $N = 128$ | $N = 256$ |
|---|---|---|---|---|---|
| $\epsilon = 10^{-3}$ | $m = 0$ | $\star$ | 1.8144(-02) | 3.6293(-04) | 3.3776(-07) |
| | $m = 1$ | 3.5818(-02) | 4.5561(-04) | 1.3573(-07) | 3.4528(-14) |
| | $m = 2$ | 1.6063(-02) | 3.6709(-04) | 6.3134(-08) | 4.9238(-14) |
| $\epsilon = 10^{-6}$ | $m = 1$ | $\star$ | $\star$ | 4.3554(-02) | 1.3762(-03) |
| | $m = 2$ | $\star$ | 2.5004(-02) | 2.4636(-03) | 9.7656(-07) |
| | $m = 3$ | $\star$ | 3.7734(-02) | 7.1848(-04) | 2.3793(-07) |
| $\epsilon = 10^{-9}$ | $m = 1$ | $\star$ | $\star$ | $\star$ | $\star$ |
| | $m = 2$ | $\star$ | $\star$ | $\star$ | 6.1103(-03) |
| | $m = 3$ | $\star$ | $\star$ | 6.7784(-03) | 1.0602(-04) |

Even with a significantly smaller $\epsilon$, the number of nodes ($N$) does not have to be drastically increased. The error remains relatively small, so the approximation using Newton's method for a nonlinear Burger's equation is the most accurate way to approximate thin boundary layers.

Newton's method tends to converge quickly. There is a free parameter $m$ (calculated during the transformation of Chebyshev) and the value of $m$ should be increased as the boundary layer width decreases. If different curves obtained by two different values of $m$ are nearly indistinguishable, then the corresponding numerical solution are good approximations to the solution. If different curves obtained by two different values of $m$ are different, especially near the boundaries, then the value of $N$ must be increased until a more accurate solution is obtained.

## 5 Conclusions and Future Work

This summer at CSUMS, I worked on understanding boundary layer problems, and learning about their applications, rather than trying to come up with any original research. It is clear now, that as the boundary layer becomes thinner in width, the approximation of the solution becomes more and more difficult to find with a given set of nodes. The three methods explored within this paper (Chebyshev collocation, the transformation of Chebyshev through the introduction of an iterated SINE function, and the nonlinear Burger's equation approximating with Newton's method) are all different applications of the spectral method. Through the exploration of each different method, the significance of the placement of the nodes (collocation points) became a key aspect to finding the most accurate solution. As the boundary layer get increasingly

[3]Shen, Jie, Tang, Tao and Wang, Li-Lian. 2011. *Spectral Methods Algorithms, Analyses and Applications*. West Lafayette, IN. Perdue University. Springer.

smaller, the collocation points must be more closely concentrated at the boundaries, to find a decent approximation of a thin boundary layer.

The next step that I would like to take with this problem is to ”invent” a function that is changing abruptly close to the boundary. The position where the function exhibits the ”sudden” change needs to move closer and closer to the boundary layer as $\epsilon$ gets smaller; thus the width of the layer is shrinking and getting steeper. Then I need to compute the exact solution to the equation using the initial boundary value problem:

$$\epsilon u'' - xu' - u$$

to obtain a new right hand side. Then, pretending that I don’t know the solution the the equation, use the collocation codes that I obtained from the three applications to solve the problem. I want to compare the exact solution with the approximations solved using the collocation codes written in Matlab. Eventually, I would like to implement the reduced basis method observe and predict how the boundary layers behave and change given a set of known parameters.

## Works Cited

Shen, Jie, Tang, Tao and Wang, Li-Lian. 2011. *Spectral Methods Algorithms, Analyses and Applications*. West Lafayette, IN. Perdue University. Springer.

Trefethen, Lloyd N. 2001. *Spectral Methods in Matlab*. Oxford, England. Oxford University. Society for Industrial and Applied Mathematics.

Trench, William F. 2003. *Introduction to Real Analysis*. San Antonio, TX. Trinity University. Prentice Hall, Inc.