PDEs with multiscale coefficients

Project Work Topic 2

Jiarong WU
wu08998@gtiit.edu.cn

Qiansheng HAN
han07024@gtiit.edu.cn

Yu MIAO
miao07685@gtiit.edu.cn

Guangdong Technion
Math with Computer Science

November 26, 2023

Abstract

The goal of this project is to investigate the use of finite element methods for solving partial differential equations (PDEs) with rapidly varying coefficients, which arise in numerous scientific fields. For instance, electrical machines frequently utilize laminated electrical steel sheets with rapidly changing properties. We develop a mathematical model of the differential equation that describes the behavior of a quantity affected by such coefficients. However, since the coefficient function does not have a limit, solving the differential equation with the coefficient set to its limiting value is not feasible. To address this issue, we use two numerical methods, the first-order and second-order finite element methods, to approximate the exact solution for each layer thickness ϵ. Both methods are implemented, but we discover that the second-order FE-Solver produces exact solutions when the domain is partitioned appropriately, which is surprising.

Using the second-order FE-Solver, we generate plots of the exact solutions as ϵ approaches zero, revealing that they converge to a parabolic shape over the domain. We also derive an explicit formula for the limiting solution through theoretical analysis. By contrast, the accuracy of the solutions obtained by the first-order FE-Solver depends on the mesh grid density h and ϵ. The finite element solution error only decreases when h is below a certain threshold, h_ϵ, that is dependent on ϵ.

Furthermore, we investigate the convergence of these exact solutions as ϵ tends to zero and explain the reason for the coincidence points between the exact solution and the limiting solution. We then extend the model to account for more general non-homogeneous conditions or coefficients, discuss ways to improve the model, and raise interesting questions related to the error of the finite element solution.

Overall, our study provides valuable insights into the behavior of PDEs with rapidly varying coefficients and demonstrates the effectiveness of finite element methods for solving such problems.

Contents

1 Introduction

The topic of this project is to study solutions to PDEs with rapidly varying coefficients using finite element method. Such PDEs arise in several fields of science. For example, electrical machines are often constructed by laminating electrical steel sheets together, resulting into material with rapidly varying properties.

We consider a sequence of problems: find u_ϵ such that

$$\begin{aligned} -(a(x/\epsilon)u(x))' &= f \quad \text{on } (0,1) \\ u_\epsilon(0) &= u_\epsilon(1) = 0 \end{aligned} \tag{1}$$

, where $f = 1$, a is some 1-periodic function and $\epsilon > 0$ is a parameter defining the period of the coefficient function $a(x/\epsilon)$. Fix,

$$a(x) = \begin{cases} a_1 & x \in [0, 1/2) \\ a_2 & x \in [1/2, 1) \end{cases},$$

where $a_1, a_2 > 0$. For example, use values $a_1 = 1$ and $a_2 = 10$.

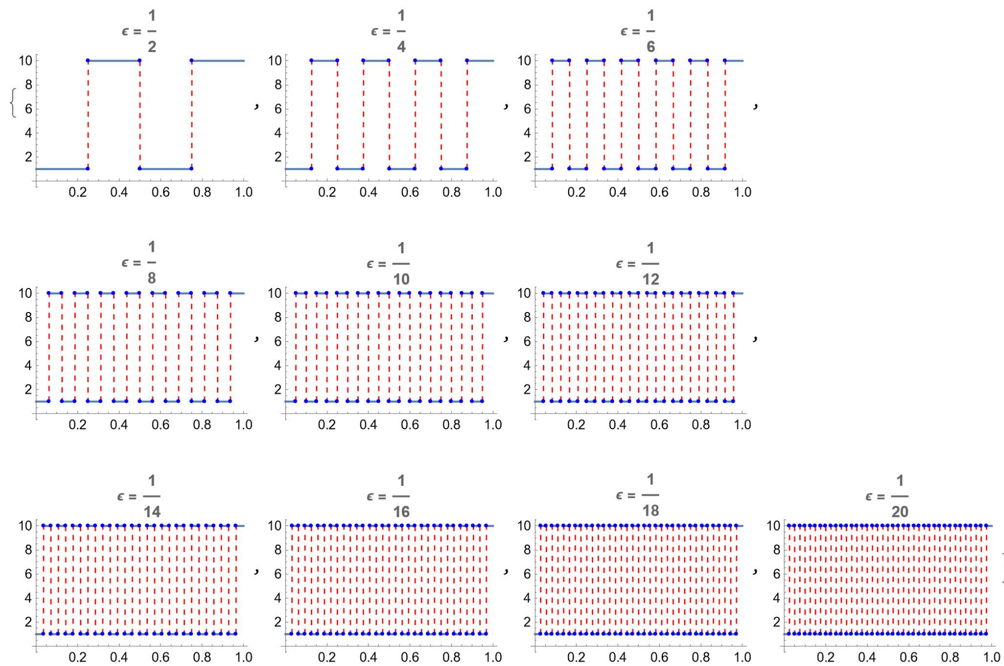


Figure 1.0.1: $a(x/\epsilon)$ for different ϵ values

2 Model

Explain your solution to the problem, also explain the process how you created this solution. Explain each equation or algorithm part. This is the most important part of your report, we are doing modelling after all.

Intuition To begin with, we introduce a brief intuition for the purpose of understanding the problem.

An immediate thought to estimate u_ϵ when $\epsilon \to 0$ is to replace $a(x/\epsilon)$ by its limit in the problem, if the limit of $a(x/\epsilon)$ exists. By a simple observation on the sequence of $a_\epsilon(1/3)$ for $\epsilon = 2^{-n}$ where $n \in \mathbb{N}$, we conclude that $a_\epsilon(x)$ does not converge as $\epsilon \to 0$. Thus, we seek on the following invariant in the process of $\epsilon \to 0$ for an intuition of u_ϵ.

$a(x/\epsilon)$ is piecewise constant in $[0,1]$, so $-(a(x/\epsilon)u_\epsilon'(x))' = -a(x/\epsilon)u_\epsilon''(x)$, the equation holds piecewisely on $[0,1]$.

$$\begin{cases} -a(x/\epsilon)u_\epsilon''(x) = 1 \\ u_\epsilon(0) = u_\epsilon(1) = 0 \end{cases}$$

For each interval $[(n-1)\epsilon, n\epsilon]$ where $n < 1/\epsilon$ is an positive integer, the change of the derivative

$$\begin{aligned} u'(n\epsilon) - u'((n-1)\epsilon) &= \int_{(n-1)\epsilon}^{(n-1/2)\epsilon} -\frac{1}{a(x/\epsilon)}dx + \int_{(n-1/2)\epsilon}^{n\epsilon} -\frac{1}{a(x/\epsilon)}dx \\ &= -\frac{\epsilon}{2a_1} - \frac{\epsilon}{2a_2} = -\frac{a_1+a_2}{2a_1a_2}\epsilon \end{aligned}$$

is fixed.

Thus, the following derivative-like fraction is a constant.

$$\frac{u'(n\epsilon) - u'((n-1)\epsilon)}{n\epsilon - (n-1)\epsilon} = -\frac{a_1+a_2}{2a_1a_2}$$

By intuition, as ϵ decreases, the intervals $[(n-1)\epsilon, n\epsilon]$ goes finer, we assume on the second order derivative

$$u''(x) = \lim_{h\to 0} \frac{u'(x+h) - u'(x)}{h} = -\frac{a_1+a_2}{2a_1a_2}$$

Combine it with the boundary conditions $u_\epsilon(0) = u_\epsilon(1) = 0$, we obtain our following conjecture

$$u(x) = \lim_{\epsilon\to 0} u_\epsilon(x) = -\frac{a_1+a_2}{4a_1a_2}\left((x-\frac{1}{2})^2 - \frac{1}{4}\right)$$

2.1 1-st order FE-solver

[1] Similar to the second-order FE-solver, the approximation of the solution is piecewise linear on each interval.

In the first-order FE-solver, the approximation of the solution is given by the formula:

$$\hat{u}(x) = \alpha_{k+1} * \frac{x_1 - x}{x_1 - x_2} + \alpha_k * \frac{x_2 - x}{x_2 - x_1}$$

where α_k and α_{k+1} correspond to the coefficients of the two basis functions on the k-th interval [x1, x2], and x is a point in the interval.

The approximate solution at each of these subintervals is plotted as shown below. The exact solution is also plotted for comparison.

```
% plot the solution
partition=11;
xdata=[];
ydata=[];
for k = 1:(N-1)
    % extract endpoints of the interval
    x1 = x(k);
    x2 = x(k+1);
    curxdata=linspace(x1,x2,partition);
    xdata = [xdata,curxdata];
    ydata =[ydata, (u(k+1)*(x1-curxdata)/(x1-x2)+u(k)*(x2-curxdata)/(x2-x1
        ))];
end
```

2.2 2-nd order FE-solver

[1] Consider for a fixed segment $[n\epsilon, (n+1)\epsilon)$, we have that a is constant, so

$$\begin{aligned}
&-\left(a(x/\epsilon)u_\epsilon'(x)\right)' = 1\\
\Rightarrow &-a(x/\epsilon)u_\epsilon''(x) = 1\\
\Rightarrow &\ u_\epsilon''(x) = -\frac{1}{a(x/\epsilon)} = Constant\\
\Rightarrow &\ u_\epsilon(x) = -\frac{1}{a(x/\epsilon)}x^2 + C_1x + C_2
\end{aligned}$$

Therefore, the solution is a piecewise second-order polynomial.

Let us implement a second-order FE-solver to compute the exact solution to (1). Following Section 5 of the given lecture note[2].

To construct a basis for V_h^2, where is the second-order finite element space, we use a hierarchical basis that includes the bubble functions, which are second-order polynomials with zero values at both endpoints. Including these bubble functions require modifications to the finite element solver, including indexing the basis functions, evaluating entries of the stiffness matrix using numerical integration, eliminating boundary basis functions, and plotting the solution. Then, the Ritz-Galerkin approximation can be applied to solve the problem.

First, the code starts by defining the parameters a1=1, a2=10, N=11 as shown in the example of the project description, and $eps = \frac{2}{N-1}$ as suggested at the end of section 2 of the project description.Then, the linearly spaced partition $\{0, \frac{1}{2N}, \frac{1}{N}, \frac{3}{2N}, ..., 1\}$ is created.

```
a1 = 1;
a2 = 10;
N = 11;
eps = 2/(N-1);
% Create uniform partition with N nodes for (0,1)
x = linspace(0,1,N);
```

The load function f(x) is defined as a function handle using the given formula:

$$a(x) = \begin{cases} a_1 & x \in \left[0,\ \frac{1}{2}\right) \\ a_2 & x \in \left[\frac{1}{2},\ 1\right) \end{cases}$$
$$a_\epsilon(x) = a(x/\epsilon)$$

```
% define the load function.
f = @(x) 1/( ((x/eps-floor(x/eps))<0.5)*a1 + ((x/eps-floor(x/eps))>=0.5)*
    a2 );
```

The code loops over the intervals defined by the partition. For each interval, the length of the interval is computed and the derivatives of the basis functions on the interval are evaluated. The midpoint quadrature points and corresponding weights are computed, and the values of the basis functions and the load function at these points are evaluated. Finally, the integrals related to matrix A and vector b are computed and added to Ahat and bhat, respectively.This part of code is similar to that given in the lecture notes[1].

The difference between the two codes is that our revised code also includes an additional term involves the value of the solution vector at the endpoints of the interval, multiplied by a quadratic polynomial that depends on the position within the interval.

```
    [X,W]=gaussint(2,x1,x2);
    Ahat(N+k,N+k) = Ahat(N+k,N+k) + W(1)*((x1+x2-2*X(1))^2)+W(2)*((x1+x2
        -2*X(2))^2);
```

```
    bhat(N+k,1) = bhat(N+k,1) +  f(t)*(W(1)*((X(1)-x1)*(x2-X(1)))+W(2)*((X
        (2)-x1)*(x2-X(2))));
```

Next, the code extracts the matrix A and vector b by removing the rows and columns corresponding to the boundary conditions. Then, the code solves the finite element problem by computing the solution vector u using the matrix A and vector.

```
idof = setdiff(1:(2*N-1),[1 N]);
A = Ahat(idof, idof );
b = bhat(idof,1);
u(1,1) = 0;
u(N,1) = 0;
u(idof) = A\b;
```

In the second-order FE-solver, the approximation of the solution is given by the formula:

$$\hat{u}(x) = \alpha_{k+1} * \frac{x_1 - x}{x_1 - x_2} + \alpha_k * \frac{x_2 - x}{x_2 - x_1} + \alpha_{k+N} * ((x - x_1) * (x_2 - x)$$

where α_k and α_{k+1} correspond to the coefficients of the two basis functions on the k-th interval [x1, x2], and α_{k+N} corresponds to the coefficient of the bubble function on the k-th interval [x1, x2], respectively, and x is a point in the interval.

The additional term $\alpha_{k+N} * ((x - x_1) * (x_2 - x)$ in the second-order FE-solver comes from using quadratic basis functions to approximate the solution. This term represents the curvature of the solution in the interval, which is not captured by the piecewise linear approximation used in the first-order FE-solver.

The approximate solution at each of these subintervals is plotted as shown below. The exact solution is also plotted for comparison.

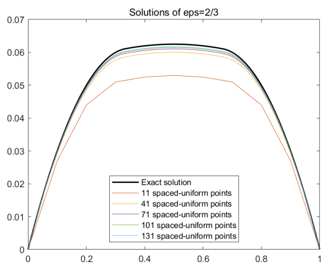
```
% plot the solution
partition=11;
xdata=[];
ydata=[];
for k = 1:(N-1)
    % extract endpoints of the interval
    x1 = x(k);
    x2 = x(k+1);
    curxdata=linspace(x1,x2,partition);
    xdata = [xdata,curxdata];
    ydata =[ydata, (u(k+1)*(x1-curxdata)/(x1-x2)+u(k)*(x2-curxdata)/(x2-x1
        )+u(k+N)*((curxdata-x1) .*(x2-curxdata)))];
end
```

3 Methods/Comparison

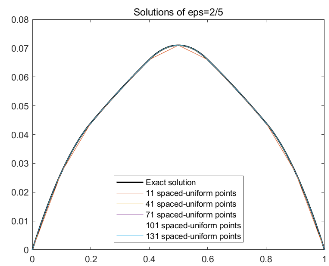
Compare different solutions with each other.

3.1 1-st order solution with exact solution

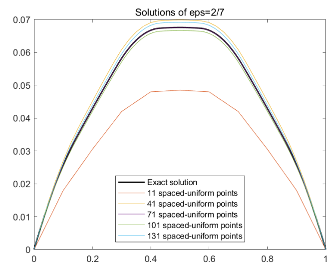
[1] The below figure shows some solutions provided by the first order FE-solver, where the red line represents the function $-\frac{a_1+a_2}{4a_1a_2}\left((x-\frac{1}{2})^2+\frac{1}{4}\right)$ and the blue line is the first order FE solution.



(a) $\epsilon = \frac{2}{3}$



(b) $\epsilon = \frac{2}{5}$

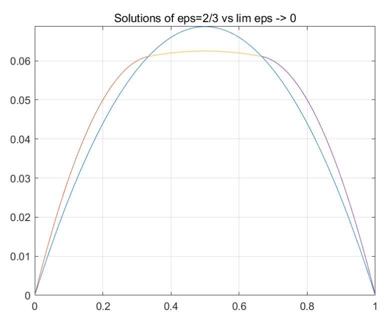


(c) $\epsilon = \frac{2}{7}$

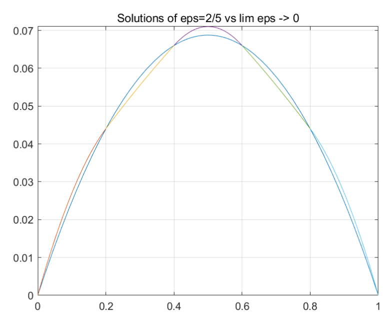
Figure 3.1.1: Some first order FE solutions for different ϵ values

3.2 Limit solution

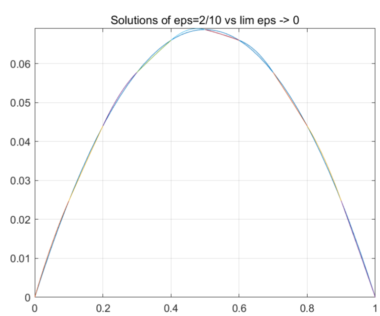
In the following graphs, the blue line represents the limit solution $-\frac{a_1+a_2}{4a_1a_2}\left((x-\frac{1}{2})^2-\frac{1}{4}\right)$ (i.e. u_ϵ when $\epsilon \to 0$), the other colors denote exact solution on the linearly spaced partition $\{0, \frac{1}{2N}, \frac{1}{N}, \frac{3}{2N}, ..., 1\}$. We can see from the graphs that as N increases, ϵ goes to zero, u_ϵ converges to $-\frac{a_1+a_2}{4a_1a_2}\left((x-\frac{1}{2})^2-\frac{1}{4}\right)$.



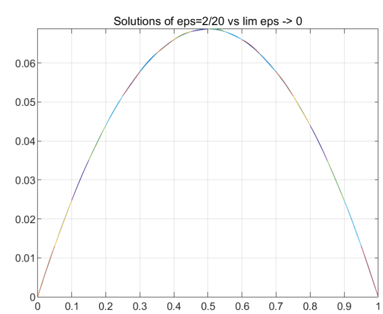
(a) $\epsilon = \frac{2}{3}$, $N = 4$



(b) $\epsilon = \frac{2}{5}$, $N = 6$



(e) $\epsilon = \frac{2}{10}$, $N = 11$



(f) $\epsilon = \frac{2}{20}$, $N = 21$

Figure 3.2.0: first order FE solutions

3.3 Error of the 1-st Order Solution

It is then nature to explore the error of the solution provided by the 1-st order FE-solver. As usual, we assume that $\epsilon = 1/N$ where $N \in \mathbb{N}$ to simplify the problem. Also, the points is uniformly spaced in $[0, 1]$ for the 1-st order FE-solver, denote the distance between two consecutive point to be h, thus, $1/h \in \mathbb{N}$. The error can be chosen in various ways, but for convenience if programming, we define the error $E = \sup\{\hat{u}_\epsilon - u_\epsilon\} = \max\{\hat{u}_\epsilon - u_\epsilon\}$, the maximum value between the 1-st order FE solution and the exact solution. In short, we consider for a fixed ϵ, what is the relation between h and E. In general, the $E \propto h$ with no surprise. But some interesting pattern is observed. Let $h = \epsilon$, there's a sudden decrease in the Error. Similar decrease appears for all $h = \epsilon/(2n)$ where $n \in \mathbb{N}$. See as the below figure as an example, more figures is attached at Some 1-st Order FE Solution Figures with Error.

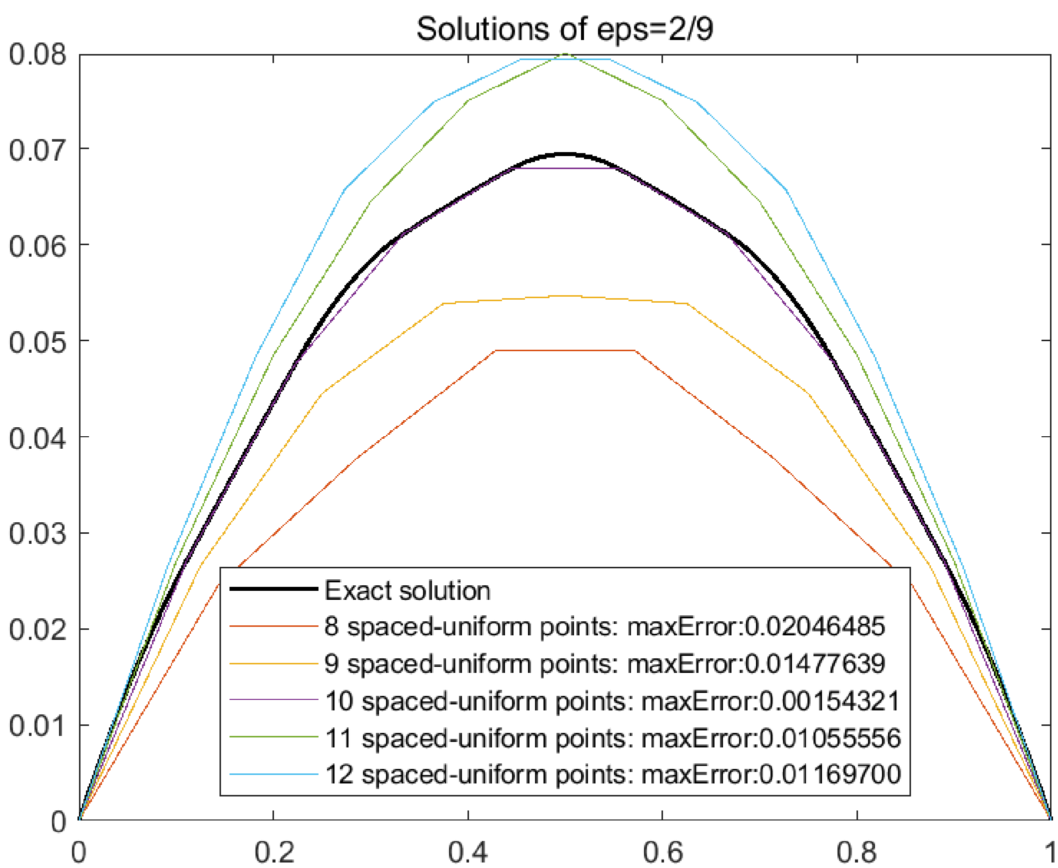


Figure 3.3.1: $\epsilon = 2/9$, $h = 1/9$

Sudden increase of the Error is also observed for $h = \epsilon/(2n + 1)$ where $n \in \mathbb{N}$. See as the below figure.

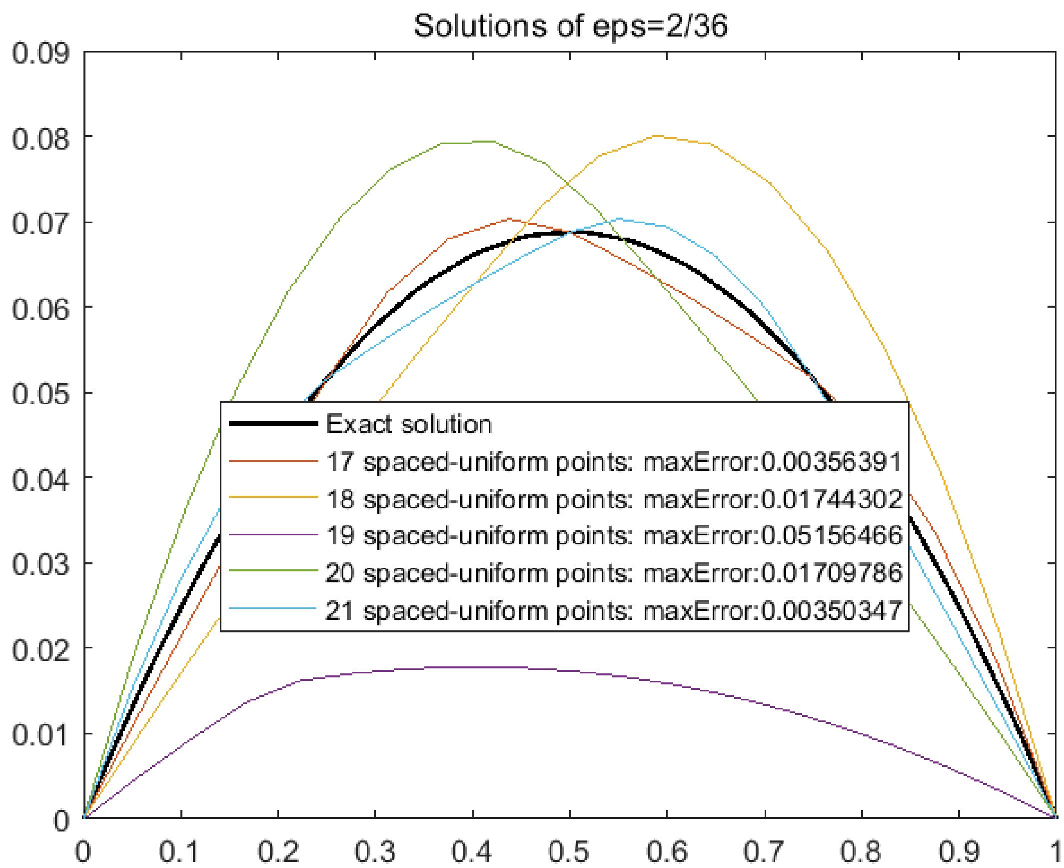


Figure 3.3.2: $\epsilon = 1/18$, $h = 1/18$

Not only for these special h occurs a sudden decrease or increase on the Error E. For h that $2/h$ and $1/\epsilon$ are not relatively prime, the sudden decrease or increase occurs with less amplitude. We provide our Matlab codes of the error calculation for the reader to verify this phenomenon, see in the Attachments at FirstOrderSolutionError.m. Note that in the program $\epsilon = 2/(N-1)$ is controlled by the symbol N and $h = 1/N0$ is controlled by the symbol $N0$.

4 Results

The restriction on the problem (1) that $f = 1$ is big assumption, with this assumption one could calculate the exact solution for any possible $\epsilon \in \mathbb{R}_+$, the only thing to do is just some complex calculation. So one may ask, could the methods be extended to a more general situation where f is some good behaved function? We try to use.

4.1 Limit solutions for non-constant source function

Consider for a general source function f, the problem is

$$\begin{cases} -a(x/\epsilon)u''_\epsilon(x) = f \\ u_\epsilon(0) = u_\epsilon(1) = 0 \end{cases}$$

For each interval $[(n-1)\epsilon, n\epsilon]$ where $n < 1/\epsilon$ is an positive integer, the change of the derivative

$$\begin{aligned} u'(n\epsilon) - u'((n-1)\epsilon) &= \int_{(n-1)\epsilon}^{(n-1/2)\epsilon} -\frac{f}{a(x/\epsilon)} dx + \int_{(n-1/2)\epsilon}^{n\epsilon} -\frac{f}{a(x/\epsilon)} dx \\ &= -\frac{1}{2a_1} \int_{(n-1)\epsilon}^{(n-1/2)\epsilon} f(x)dx - \frac{1}{2a_2} \int_{(n-1/2)\epsilon}^{n\epsilon} f(x)dx \\ &\approx -\frac{(a_1+a_2)\epsilon}{2a_1a_2} f((n-1/2)\epsilon) \end{aligned}$$

should exist if we assume the source function is good enough, for example, almost everywhere continuous.

As ϵ decreases, the intervals $[(n-1)\epsilon, n\epsilon]$ goes finer, we assume on the second order derivative

$$u''(x) = \lim_{h\to 0} \frac{u'(x+h) - u'(x)}{h} = -\frac{a_1+a_2}{2a_1a_2} f((n-1/2)\epsilon)$$

If there exists some function $F : [0,1] \to \mathbb{R}$, s.t. $F''(x) = f(x)$. Then the general solution for u should be

$$u(x) = F(x) + C_1x + C_2$$

Combine it with the boundary conditions $u_\epsilon(0) = u_\epsilon(1) = 0$, we obtain a more general conjecture

$$\lim_{\epsilon\to 0} u_\epsilon(x) = -\frac{a_1+a_2}{2a_1a_2} \left(F(x) - (F(1) - F(0))\, x - F(0)\right) \tag{2}$$

Especially, if we define it as $F(x) = \int_0^x \left(\int f\right) dx$, then $F(0) = 0$, we can simplify this as

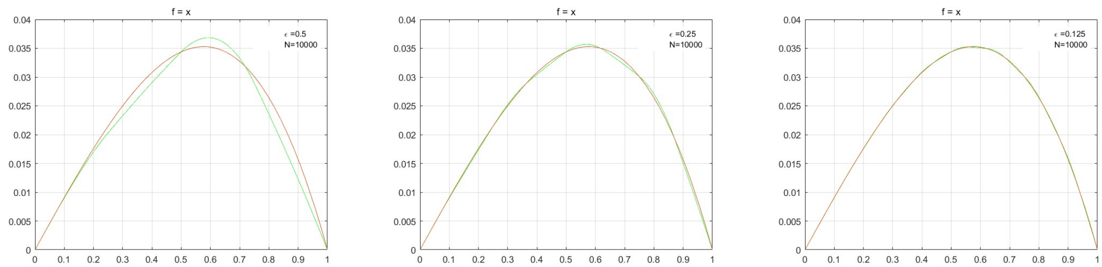
$$\lim_{\epsilon\to 0} u_\epsilon(x) = -\frac{a_1+a_2}{2a_1a_2} \left(F(x) - F(1)x\right)$$

The above limit solution (2) be verified numerically by a first order FE-solver. For example, use $a_1 = 1$ and $a_2 = 10$ as before.

The red line plots the limit solution $\lim_{\epsilon\to 0} u_\epsilon(x)$ and the green line plots the FE solution for different ϵ values in the following figures.

In the case of $f(x) = x$, take $F(x) = x^3/6$, then

$$\lim_{\epsilon\to 0} u_\epsilon(x) = -\frac{a_1+a_2}{2a_1a_2} \left(\frac{x^3}{6} - \frac{x}{6}\right)$$

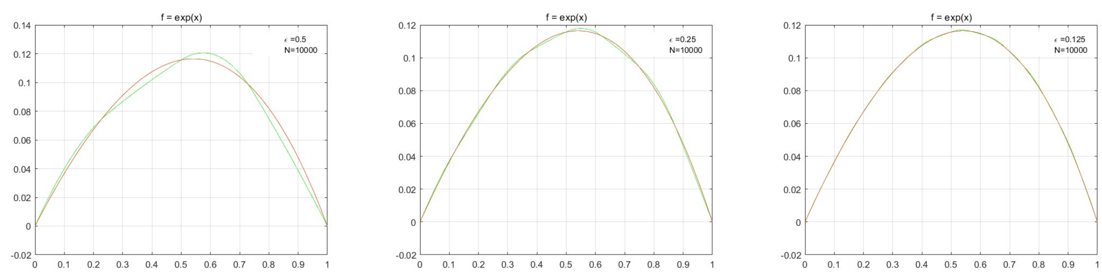


(a) $f(x) = x$, $\epsilon = 1/2$ (b) $f(x) = x$, $\epsilon = 1/4$ (c) $f(x) = x$, $\epsilon = 1/8$

Figure 4.1.1: 1-st order FE solutions for $f(x) = x$

In the case of $f(x) = e^x$, take $F(x) = e^x$, then

$$\lim_{\epsilon \to 0} u_\epsilon(x) = -\frac{a_1 + a_2}{2a_1 a_2}\left(e^x - (e-1)\,x - 1\right)$$

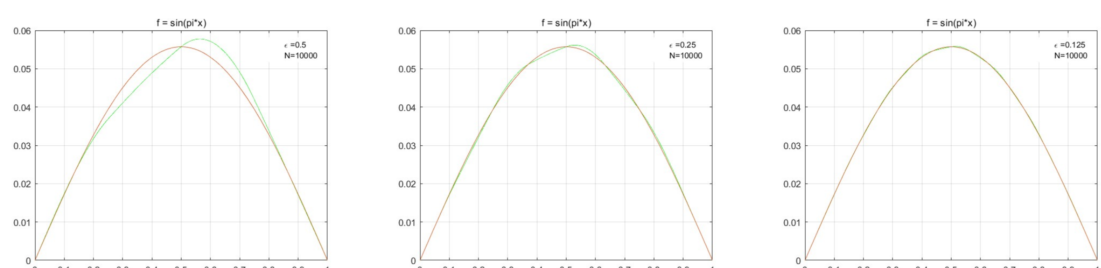


(a) $f(x) = e^x$, $\epsilon = 1/2$ (b) $f(x) = e^x$, $\epsilon = 1/4$ (c) $f(x) = e^x$, $\epsilon = 1/8$

Figure 4.1.2: 1-st order FE solutions for $f(x) = e^x$

In the case of $f(x) = \sin(\pi x)$, take $F(x) = \sin(\pi x)/\pi^2$, then

$$\lim_{\epsilon \to 0} u_\epsilon(x) = \frac{a_1 + a_2}{2a_1 a_2}\frac{\sin(\pi x)}{\pi^2}$$



(a) $f(x) = \sin(\pi x)$, $\epsilon = 1/2$ (b) $f(x) = \sin(\pi x)$, $\epsilon = 1/4$ (c) $f(x) = \sin(\pi x)$, $\epsilon = 1/8$

Figure 4.1.3: 1-st order FE solutions for $f(x) = \sin(\pi x)$

4.2 Coincidence points between exact solution and limit solution

We have introduced the limit function in (2) without rigorous proof of its convergence, let's denote it as $u(x) = \lim_{\epsilon\to 0} u_\epsilon(x)$. One possible way to prove the convergence is to prove that $\sup|u_\epsilon - u| \to 0$ as $\epsilon \to 0$. Thus, if we can obtain a sequence of partitions of the interval $[0,1]$ with the length of its subintervals decreasing, we can control the error between u_ϵ and u in each subinterval since $\sup|u'_\epsilon - u'|$ is bounded by $\sup|u''_\epsilon - u''|L_\epsilon$, where L_ϵ is the length of the subinterval.

It is then natural to consider the intersections between the image for u_ϵ and u, let's call it a coincidence point.

Definition. We say that x_0 is a coincidence point between the exact solution u_ϵ and the limit solution u if $u_\epsilon(x_0) = u(x_0)$. We will not mention the two functions and just call it a coincidence point for the purpose of convenience in this subsection.

To begin with, we consider the problem in a trivial situation where $f = 1$ and $\epsilon = 1$, but not to make it too trivial we assume that $a_1 \neq a_2$. Thus, the exact solution is a piecewise second order polynomial, we can write it as

$$u_1(x) = \begin{cases} -\dfrac{1}{a_1}\dfrac{x^2}{2} + Ax + B & x \in [0, 1/2] \\ -\dfrac{1}{a_2}\dfrac{x^2}{2} + Cx + D & x \in [1/2, 1] \end{cases}$$

By the boundary condition $u_1(0) = u_1(1) = 0$ and the continuity of u and u' at the point $x = 1/2$, we can solve for A, B, C, D, so the exact solution is

$$u_1(x) = \begin{cases} -\dfrac{1}{2a_1}x^2 + \left(\dfrac{3}{8a_1} + \dfrac{1}{8a_2}\right)x & x \in [0, 1/2] \\ -\dfrac{1}{2a_2}x^2 + \left(-\dfrac{1}{8a_1} + \dfrac{5}{8a_2}\right)x + \left(\dfrac{1}{8a_1} - \dfrac{1}{8a_2}\right) & x \in [1/2, 1] \end{cases}$$

Also, we have that the limit solution is

$$u(x) = \lim_{\epsilon\to 0} u_\epsilon(x) = -\frac{a_1 + a_2}{4a_1a_2}\left((x - \frac{1}{2})^2 - \frac{1}{4}\right)$$

Notice that u and u_1 coincides at the point $x = 1/2$,

$$u(1/2) = u_1(1/2) = \frac{a_1 + a_2}{16a_1a_2}$$

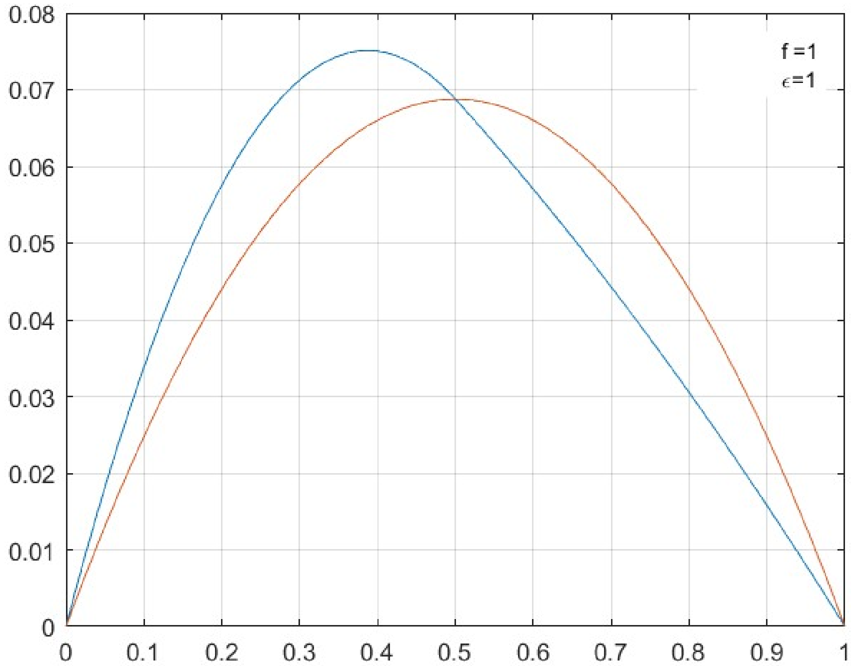


Figure 4.2.1: $f = 1$, $\epsilon = 1$

We can extend this result and find some pattern about the coincidence points.

1. Keep $f = 1$ and let $\epsilon = 1/n$ where $n \in \mathbb{N}$. By numerical observation we claim that $u(x) = u_\epsilon(x)$ if and only if $x = k\epsilon/2$ for $k \in \mathbb{N}$ and $0 \le k \le 2n$. That is to say, the coincidence points are exactly $\{k\epsilon/2\}_0^{2n}$. The rigorous proof might be done by solving a series of equations obtained from the continuity of u and u' at these points $\{k\epsilon/2\}_0^{2n}$, which is exactly the coincidence points.

2. Let $f = F''$ be some good behaved function where $F(0) = 0$ and keep $\epsilon = 1$. By boundary conditions $u_1(0) = u_1(1) = 0$ we can simplify the original problem to the following problem

$$u_1(x) = \begin{cases} -\dfrac{1}{a_1}F(x) + Ax & x \in [0, 1/2] \\ -\dfrac{1}{a_2}F(x) + Cx + \dfrac{1}{a_2} - C & x \in [1/2, 1] \end{cases}$$

   By numerical observation we claim that $u(1/2) = u_1(1/2)$ for every good behaved f. It is possible to do similar deduction as the trivial situation for $f = 1$ to obtain a more general proof for this situation.

What about combine this two situation for a more general situation? Let $f = F''$ be some good behaved function and let $\epsilon = 1/n$ where $n \in \mathbb{N}$. It is true that $x = 1/2$ will always be a coincidence point. However, there's no trivial pattern that is general among these coincidence points as $\epsilon \to 0$.

One special conjecture we can make is, if we restrict the source function f to have a symmetry on $[0, 1]$ s.t. $f(x) = f(1 - x)$, then $u(x_0) = u_\epsilon(x_0)$ implies $u(1 - x_0) = u_\epsilon(1 - x_0)$, or to say x_0 is a coincidence point implies that $1 - x_0$ will also be a coincidence point. Another way is to say that the coincidence points can share this symmetry from f, which should be somehow impressive since neither u_ϵ nor u'_ϵ have the symmetry.

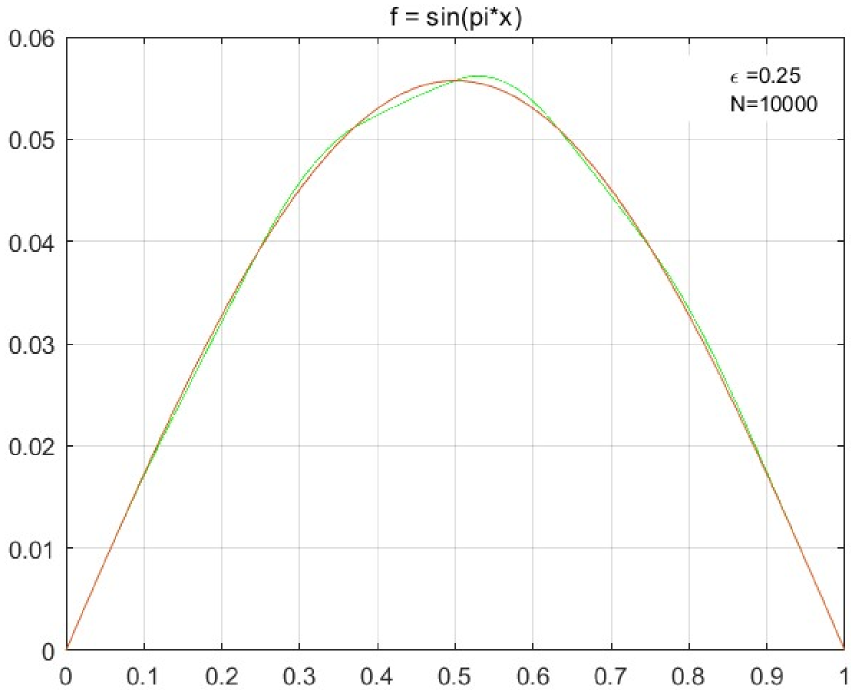


Figure 4.2.2: Symmetry of the coincidence points

5 Summary

We reached a conclusion for the limit function for the problem (1) in the special case when $f = 1$:

$$\lim_{\epsilon \to 0} u_\epsilon(x) = -\frac{a_1 + a_2}{4a_1a_2}\left((x - \frac{1}{2})^2 - \frac{1}{4}\right)$$

The result is verified numerically.
The good thing is that we generalized the result that for a more general f, if $f = F''$ and $F(0) = 0$, we can have a more general limit function (2):

$$\lim_{\epsilon \to 0} u_\epsilon(x) = -\frac{a_1 + a_2}{2a_1a_2}\left(F(x) - F(1)x\right)$$

But there's some rigor lacked in our modelling process, so it could be improved in the following ways:

1. The strict condition of f for the convergence of $u_\epsilon(x)$ and the equation (2) to hold.
2. The mathematical structure behind the coincidence points between the exact solution u_ϵ and the limit solution u.

Instead of having a general f, we can have a more general coefficient $a(x)$, so the problem can be done further by analyzing there topics:

3. The limit solution when a_1 and a_2 is not two fixed constant value but is constant only at each sub interval.
4. The limit solution when the piecewise constant coefficient $a(x)$ is varying at a non-constant frequency.

But not always people want the limit solution as $\epsilon \to 0$, it may happen that ϵ is some fixed constant in practical use. The exact solution u_ϵ may not be easy to calculate for a general f and a general a, so the FE-solver would help a lot. Therefore, some interesting question can be carried about the error of the FE solution:

5. The upper bound for the error of a 1-st/2-nd order FE solution with N points.
6. Possible refinements of the partition points for the FE-solver when $a(x)$ is varying at a non-constant frequency.

References

[1] Antti Hannukainen. "Finite element method in 1D". In: *Brief introduction to Finite Element method*. 2021, pp. 10–17. DOI: NULL. URL: https://moodle.tuni.fi/pluginfile.php/2972279/mod_resource/content/2/FEM2021.pdf.

[2] Antti Hannukainen. "Section 5". In: *Section 5: Brief introduction to Finite Element method*. NULL, 2023, pp. 1–28. DOI: NULL. URL: https://moodle.tuni.fi/pluginfile.php/2972279/mod_resource/content/2/FEM2021.pdf.

Attachments

FirstOrderFE.m

```
% matlab code blocks
clc,clearvars
a1 = 1;
a2 = 10;
N = 11;
eps = 2/(N-1);
% Create uniform partition with N nodes for (0,1)
x = linspace(0,1,N);
% define the load function .
f = @(x) 1/( ((x/eps-floor(x/eps))<0.5)*a1 + ((x/eps-floor(x/eps))>=0.5)*
    a2 );

% Initialise the matrix Ahat and vector bhat .
Ahat = sparse(2*N-1,2*N-1);
bhat = zeros(2*N-1,1);

% loop over the intervals
for k = 1:(N-1)
    % extract endpoints of the interval
    x1 = x(k);
    x2 = x(k+1);
    % evaluate length of interval k .
    len = x2-x1;
    % evaluate derivatives of basisfunctions on interval k .
    dphi(1) = 1/(x1-x2);
    dphi(2) = 1/(x2-x1);
    % midpoint quadrature points
    t = (x1+x2)/2; w = x2-x1;
    % evaluate values of basisfunctions
    % source term at integration points
    phi(:,1) = (t-x2) ./(x1-x2);
    phi(:,2) = (t-x1) ./(x2-x1);
    fval = f(t);
    % enumerate the basisfunctions on interval k .
    enum([1 2]) = [k k+1];
    for i=1:2
        % evaluate intergrals related to b .
        bhat(enum(i) ) = bhat(enum(i) ) + dot(fval .*phi(:,i),w);
        for j=1:2
            % evaluate integral related to A
            Ahat(enum(i),enum(j)) = Ahat(enum(i),enum(j)) + dphi(i)*dphi(j
                )*len;
        end
    end
    [X,W]=gaussint(2,x1,x2);
    Ahat(N+k,N+k) = Ahat(N+k,N+k) + W(1)*((x1+x2-2*X(1))^2)+W(2)*((x1+x2
        -2*X(2))^2);
```

```
    bhat(N+k,1) = bhat(N+k,1) +  f(t)*(W(1)*((X(1)-x1)*(x2-X(1)))+W(2)*((X
        (2)-x1)*(x2-X(2))));
end



idof = setdiff(1:(2*N-1),[1 N]);
A = Ahat(idof, idof );
b = bhat(idof,1);
u(1,1) = 0;
u(N,1) = 0;
u(idof) = A\b;

% plot the solution
partition=11;
xdata=[];
ydata=[];
for k = 1:(N-1)
    % extract endpoints of the interval
    x1 = x(k);
    x2 = x(k+1);
    curxdata=linspace(x1,x2,partition);
    xdata = [xdata,curxdata];
    ydata =[ydata, (u(k+1)*(x1-curxdata)/(x1-x2)+u(k)*(x2-curxdata)/(x2-x1
        ))];
end
plot(xdata,ydata);
hold on;
% fplot(f,[0,1])
fplot(@(x) limit_solution(a1,a2,x),[0,1])
legend('Approximation','Limit solution')
grid on; hold off;
```

SecondOrderFE.m

```
clc,clearvars
a1 = 1;
a2 = 10;
N = 11;
eps = 2/(N-1);
% Create uniform partition with N nodes for (0,1)
x = linspace(0,1,N);
% define the load function .
f = @(x) 1/( ((x/eps-floor(x/eps))<0.5)*a1 + ((x/eps-floor(x/eps))>=0.5)*
    a2 );

% Initialise the matrix Ahat and vector bhat .
Ahat = sparse(2*N-1,2*N-1);
bhat = zeros(2*N-1,1);

```

```
% loop over the intervals
for k = 1:(N-1)
    % extract endpoints of the interval
    x1 = x(k);
    x2 = x(k+1);
    % evaluate length of interval k .
    len = x2-x1;
    % evaluate derivatives of basisfunctions on interval k .
    dphi(1) = 1/(x1-x2);
    dphi(2) = 1/(x2-x1);
    % midpoint quadrature points
    t = (x1+x2)/2; w = x2-x1;
    % evaluate values of basisfunctions
    % source term at integration points
    phi(:,1) = (t-x2) ./(x1-x2);
    phi(:,2) = (t-x1) ./(x2-x1);
    fval = f(t);
    % enumerate the basisfunctions on interval k .
    enum([1 2]) = [k k+1];
    for i=1:2
        % evaluate intergrals related to b .
        bhat(enum(i) ) = bhat(enum(i) ) + dot(fval .*phi(:,i),w);
        for j=1:2
            % evaluate integral related to A
            Ahat(enum(i),enum(j)) = Ahat(enum(i),enum(j)) + dphi(i)*dphi(j)*len;
        end
    end
    [X,W]=gaussint(2,x1,x2);
    Ahat(N+k,N+k) = Ahat(N+k,N+k) + W(1)*((x1+x2-2*X(1))^2)+W(2)*((x1+x2-2*X(2))^2);
    bhat(N+k,1) = bhat(N+k,1) +  f(t)*(W(1)*((X(1)-x1)*(x2-X(1)))+W(2)*((X(2)-x1)*(x2-X(2))));
end



idof = setdiff(1:(2*N-1),[1 N]);
A = Ahat(idof, idof );
b = bhat(idof,1);
u(1,1) = 0;
u(N,1) = 0;
u(idof) = A\b;

% plot the solution
partition=11;
xdata=[];
ydata=[];
for k = 1:(N-1)
    % extract endpoints of the interval
    x1 = x(k);
```

```
    x2 = x(k+1);
    curxdata=linspace(x1,x2,partition);
    xdata = [xdata,curxdata];
    ydata =[ydata, (u(k+1)*(x1-curxdata)/(x1-x2)+u(k)*(x2-curxdata)/(x2-x1
       )+u(k+N)*((curxdata-x1) .*(x2-curxdata)))];
end
plot(xdata,ydata);
hold on;
% fplot(f,[0,1])
fplot(@(x) limit_solution(a1,a2,x),[0,1])
legend('Approximation','Exact solution')
grid on; hold off;
```

limit_solution.m

```
function f = limit_solution(a1,a2,x)
    f = (a1+a2)/(4*a1*a2) * ( -(x-0.5).^2 + 0.25);
    % f = (a1+a2)/(2*a1*a2) * ( -x.^3/6 + x/6);
    % f = (a1+a2)/(2*a1*a2) * ( -exp(x) + (2.718-1).*x + 1);
    % f = (a1+a2)/(2*a1*a2) * ( sin(pi*x)/pi^2 );
end
```

FirstOrderSolutionError.m

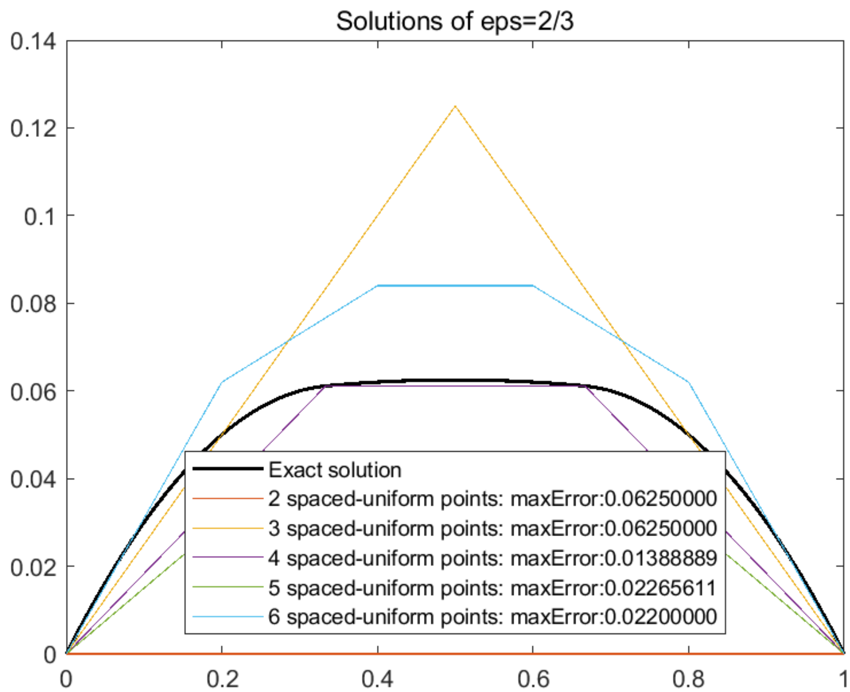
```
clc,clearvars,close all
a1 = 1;
a2 = 10;
for N=19
    eps = 2/(N-1);
    % Create uniform partition with N nodes for (0,1)
    x = linspace(0,1,N);
    % define the load function.
    f = @(x) 1/( ((x/eps-floor(x/eps))<0.5)*a1 + ((x/eps-floor(x/eps))
       >=0.5)*a2 );

    % Initialise the matrix Ahat and vector bhat.
    Ahat = sparse(2*N-1,2*N-1);
    bhat = zeros(2*N-1,1);

    % loop over the intervals
    for k = 1:(N-1)
        % extract endpoints of the interval
        x1 = x(k);
        x2 = x(k+1);
        % evaluate length of interval k.
        len = x2-x1;
        % evaluate derivatives of basisfunctions on interval k.
        dphi(1) = 1/(x1-x2);
```

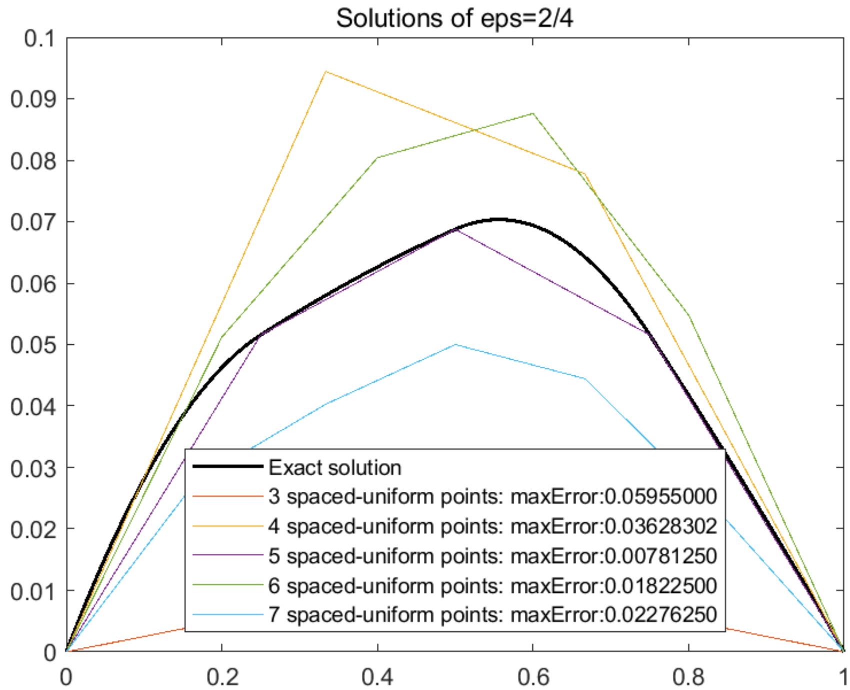
```
        dphi(2) = 1/(x2-x1);
        % midpoint quadrature points
        t = (x1+x2)/2; w = x2-x1;
        % evaluate values of basisfunctions
        % source term at integration points
        phi(:,1) = (t-x2)./(x1-x2);
        phi(:,2) = (t-x1)./(x2-x1);
        fval = f(t);
        % enumerate the basisfunctions on interval k.
        enum([1 2]) = [k k+1];
        for i=1:2
            % evaluate intergrals related to b.
            bhat(enum(i) ) = bhat(enum(i) ) + dot(fval.*phi(:,i),w);
            for j=1:2
                % evaluate integral related to A
                Ahat(enum(i),enum(j)) = Ahat(enum(i),enum(j)) + dphi(i)*dphi(j)*len;
            end
        end
        [X,W]=gaussint(2,x1,x2);
        Ahat(N+k,N+k) = Ahat(N+k,N+k) + W(1)*((x1+x2-2*X(1))^2)+W(2)*((x1+x2-2*X(2))^2);
        bhat(N+k,1) = bhat(N+k,1) +  f(t)*(W(1)*((X(1)-x1)*(x2-X(1)))+W(2)*((X(2)-x1)*(x2-X(2))));
    end



    idof = setdiff(1:(2*N-1),[1 N]);
    A = Ahat(idof, idof );
    b = bhat(idof,1);
    u(1,1) = 0;
    u(N,1) = 0;
    u(idof) = A\b;
    % plot the solution
    partition=101;

    xdata=[];
    ydata=[];
    for k = 1:(N-1)
        % extract endpoints of the interval
        x1 = x(k);
        x2 = x(k+1);
        curxdata=linspace(x1,x2,partition);
        xdata = [xdata,curxdata];
        %         ydata =[ydata, (u(k+1)*(x1-curxdata)/(x1-x2)+u(k)*(x2-curxdata)/(x2-x1))];
        ydata =[ydata, (u(k+1)*(x1-curxdata)/(x1-x2)+u(k)*(x2-curxdata)/(x2-x1)+u(k+N)*((curxdata-x1).*(x2-curxdata)))];
    end
    plot(xdata,ydata, 'LineWidth',1.5,'Color','k');
```

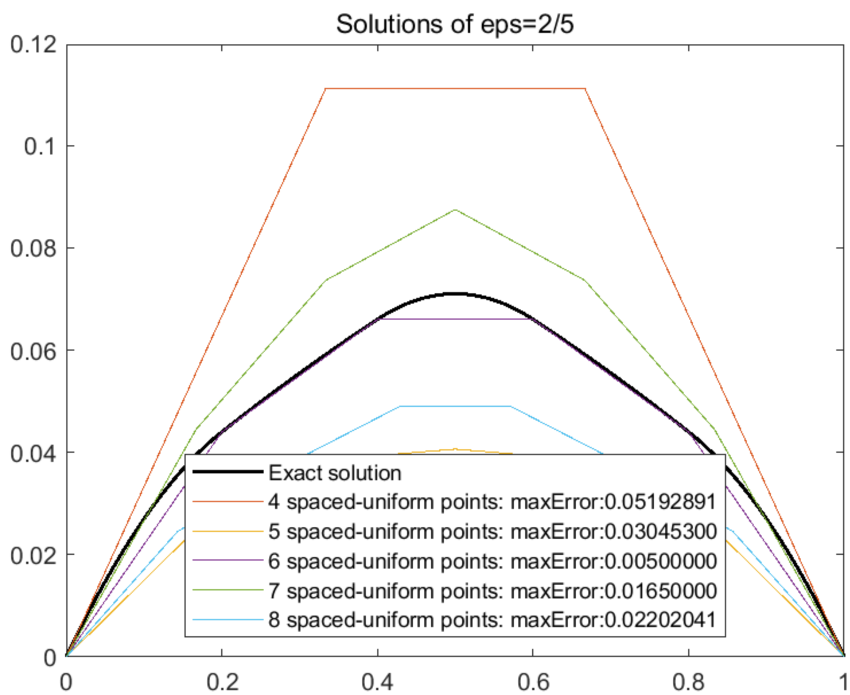
```
    hold on;
    legend('Exact solution','Location','south');
    title(sprintf('Solutions of eps=2/%d', (N-1)))

    % Create uniform partition with N nodes for (0,1)
    for N0 = N-2:N+2
        h=1/(N0-1);
        x0 = linspace(0,1,N0);
        % Initialise the matrix Ahat and vector bhat.
        Ahat0 = sparse(N0,N0);
        bhat0 = zeros(N0,1);
        % loop over the intervals
        for k = 1:(length(x0)-1)
            % extract endpoints of the interval
            x1 = x0(k);
            x2 = x0(k+1);
            % evaluate length of interval k.
            len = x2-x1;
            % evaluate derivatives of basisfunctions on interval k.
            dphi(1) = 1/(x1-x2);
            dphi(2) = 1/(x2-x1);
            % midpoint quadrature points
            t = (x1+x2)/2; w = x2-x1;
            % evaluate values of basisfunctions
            % source term at integration points
            phi(:,1) = (t-x2)./(x1-x2);
            phi(:,2) = (t-x1)./(x2-x1);
            fval = f(t);
            % enumerate the basisfunctions on interval k.
            enum([1 2]) = [k k+1];
            for i=1:2
                % evaluate intergrals related to b.
                bhat0(enum(i) ) = bhat0(enum(i) ) + dot(fval.*phi(:,i),w);
                for j=1:2
                    % evaluate integral related to A
                    Ahat0(enum(i),enum(j)) = Ahat0(enum(i),enum(j)) + dphi
                        (i)*dphi(j)*len;
                end
            end
        end
        % remove basisfunction 1 and N+1 from the system
        A0 = Ahat0(2:(N0-1), 2:(N0-1) );
        b0 = bhat0(2:(N0-1),1);
        clear u0;
        u0(1,1) = 0;
        u0(N0,1) = 0;
        u0(2:(N0-1),1) = A0\b0;

        ysample=[];
        for k = 1:(N-1)
            % extract endpoints of the interval
```

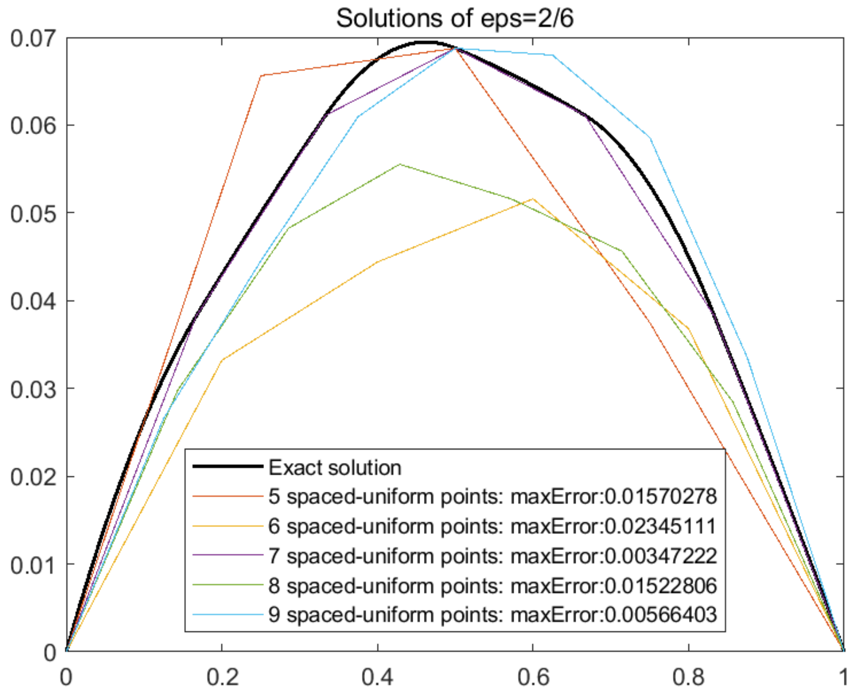
```
            x1 = x(k);
            x2 = x(k+1);
            curxdata=linspace(x1,x2,partition);
            curydata=[];
            for point = curxdata
                idx=0;
                for point2 = x0
                    if point<point2
                        break;
                    end
                    if point2==x0(end)
                        break;
                    end
                    idx = idx+1;
                end
                curydata=[curydata, u0(idx)*(x0(idx+1)-point)/(x0(idx+1)-
                    x0(idx))+u0(idx+1)*(x0(idx)-point)/(x0(idx)-x0(idx+1))
                    ];
            end
            ysample =[ysample, curydata];
        end

        plot(x0,u0,'DisplayName',sprintf('%d spaced-uniform points:
            maxError:%.8f',N0,max(abs(ydata-ysample))));
        hold on;

    end
    saveas(gcf,sprintf('1stOrderCompareExact_eps2over%d.png', N-1))
    hold off;
end
```

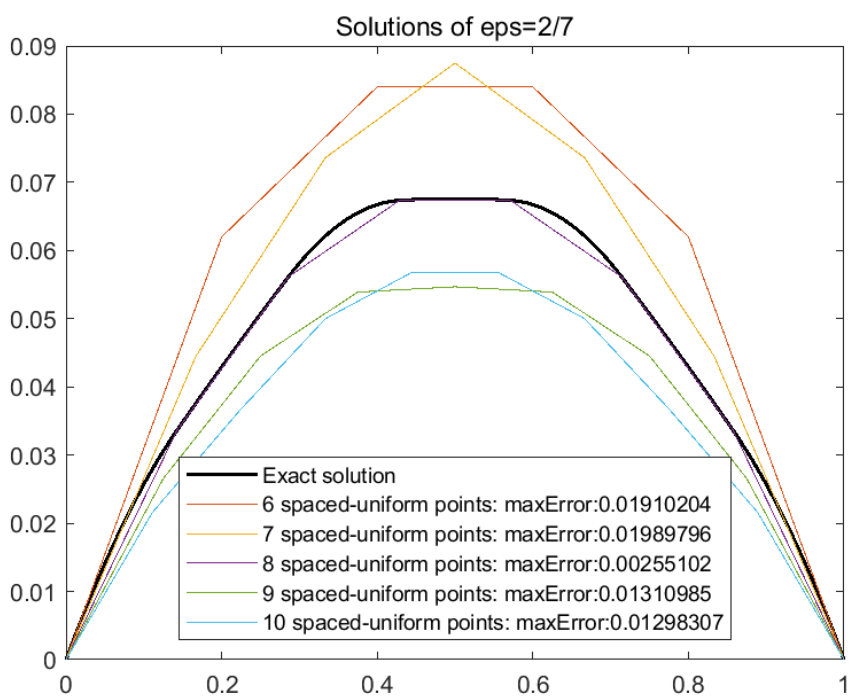
Some 1-st Order FE Solution Figures with Error

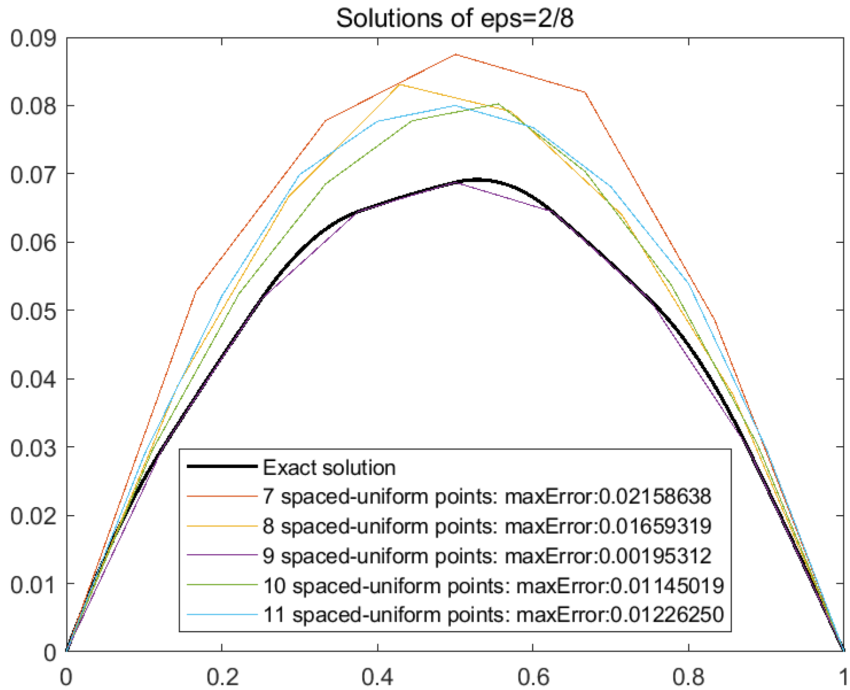


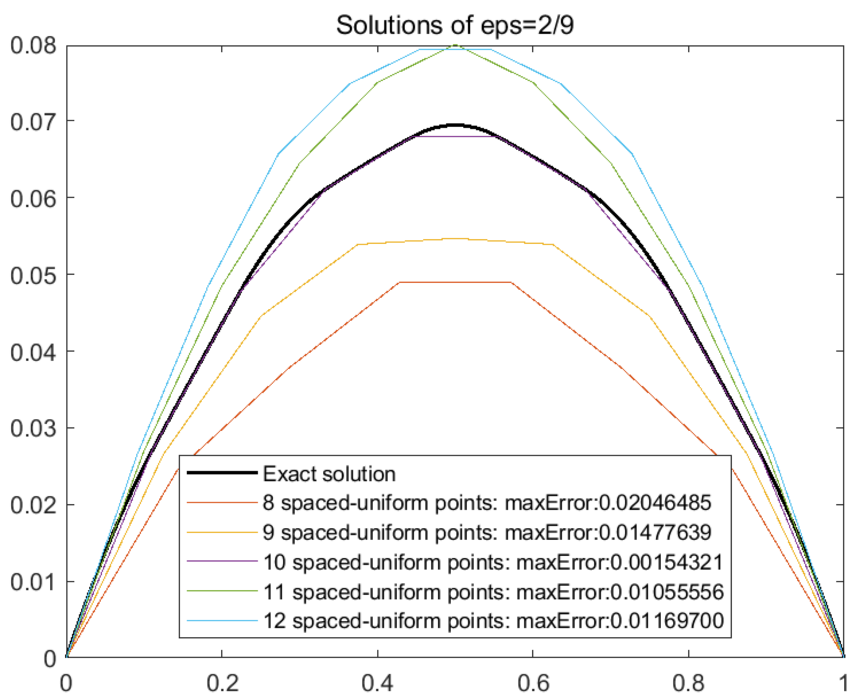

Solutions of eps=2/4
0.1
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0
0.2
0.4
0.6
0.8
1
Exact solution
3 spaced-uniform points: maxError:0.05955000
4 spaced-uniform points: maxError:0.03628302
5 spaced-uniform points: maxError:0.00781250
6 spaced-uniform points: maxError:0.01822500
7 spaced-uniform points: maxError:0.02276250

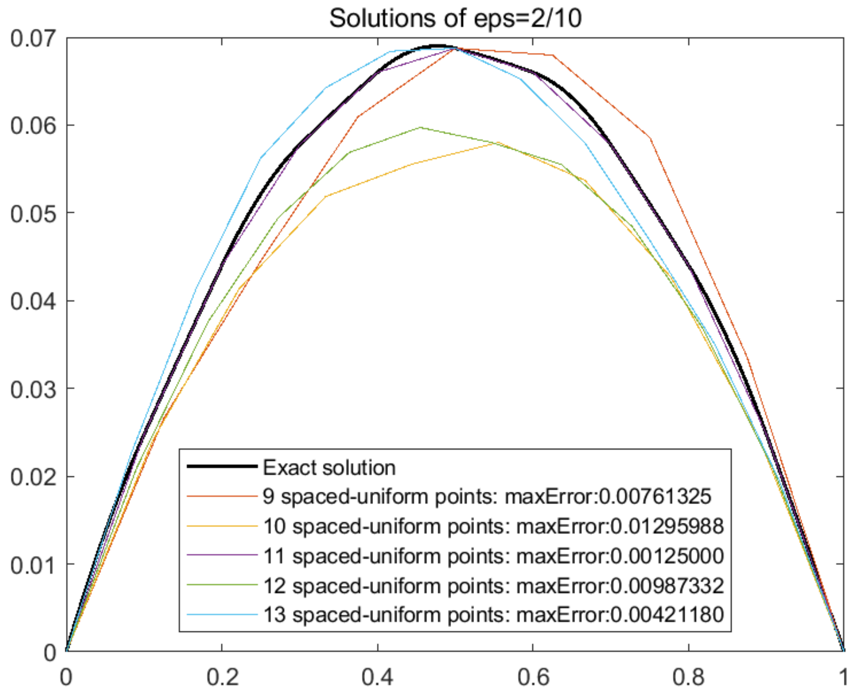

Solutions of eps=2/5
0.12
0.1
0.08
0.06
0.04
0.02
0
0
0.2
0.4
0.6
0.8
1
Exact solution
4 spaced-uniform points: maxError:0.05192891
5 spaced-uniform points: maxError:0.03045300
6 spaced-uniform points: maxError:0.00500000
7 spaced-uniform points: maxError:0.01650000
8 spaced-uniform points: maxError:0.02202041

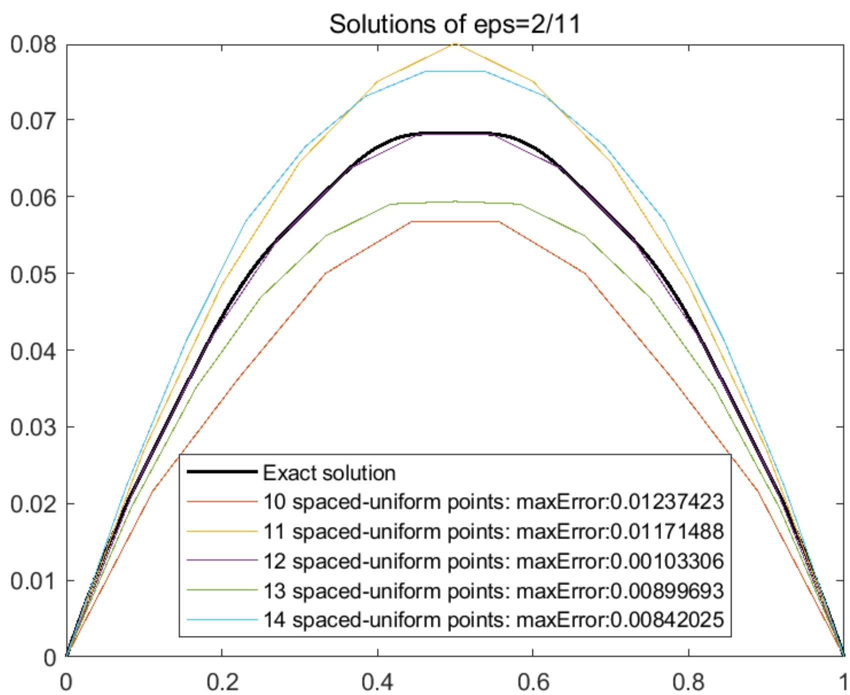

Solutions of eps=2/6
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0
0.2
0.4
0.6
0.8
1
Exact solution
5 spaced-uniform points: maxError:0.01570278
6 spaced-uniform points: maxError:0.02345111
7 spaced-uniform points: maxError:0.00347222
8 spaced-uniform points: maxError:0.01522806
9 spaced-uniform points: maxError:0.00566403

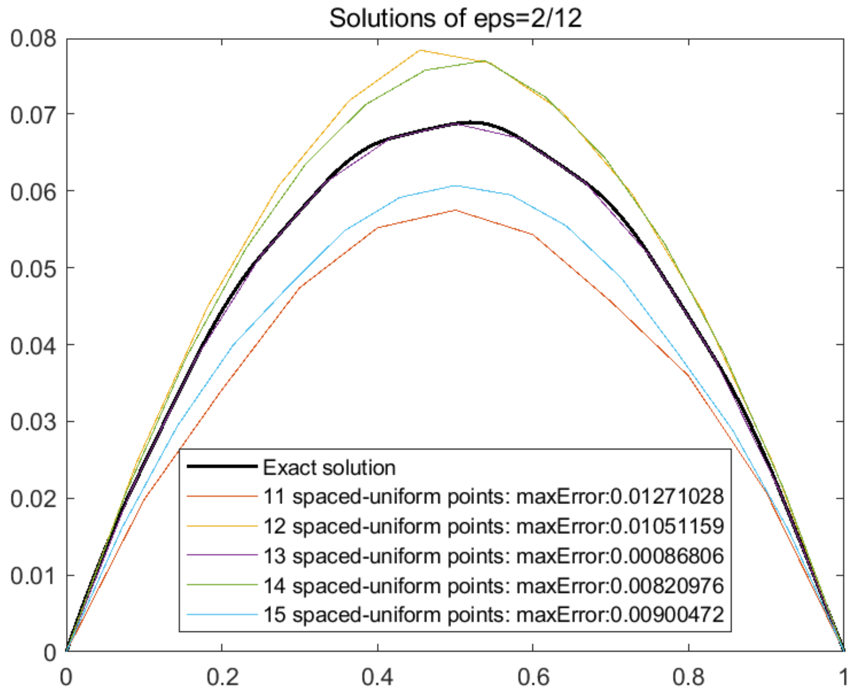

Solutions of eps=2/7
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0
0.2
0.4
0.6
0.8
1
Exact solution
6 spaced-uniform points: maxError:0.01910204
7 spaced-uniform points: maxError:0.01989796
8 spaced-uniform points: maxError:0.00255102
9 spaced-uniform points: maxError:0.01310985
10 spaced-uniform points: maxError:0.01298307

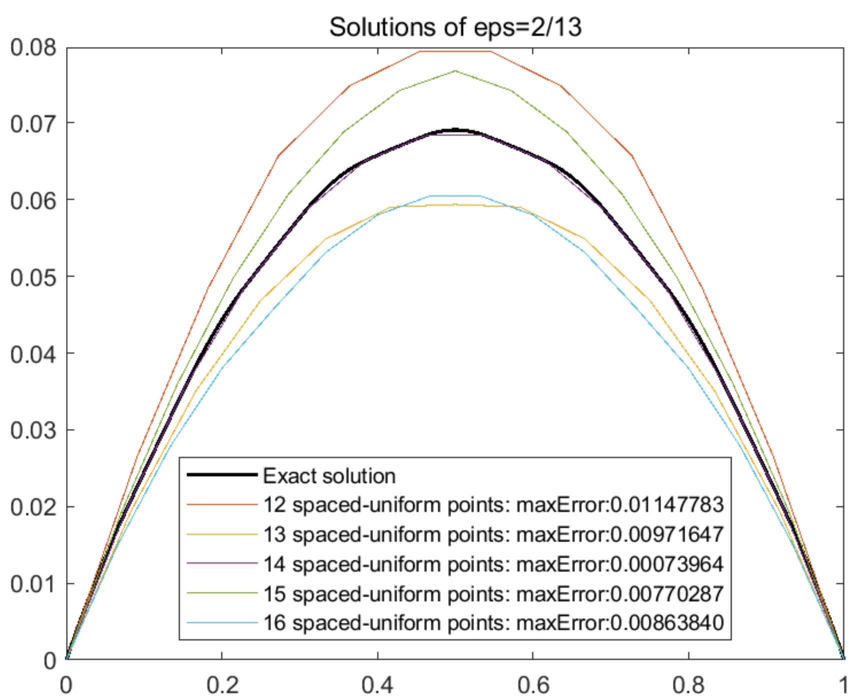

Solutions of eps=2/8
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0
0.2
0.4
0.6
0.8
1
Exact solution
7 spaced-uniform points: maxError:0.02158638
8 spaced-uniform points: maxError:0.01659319
9 spaced-uniform points: maxError:0.00195312
10 spaced-uniform points: maxError:0.01145019
11 spaced-uniform points: maxError:0.01226250

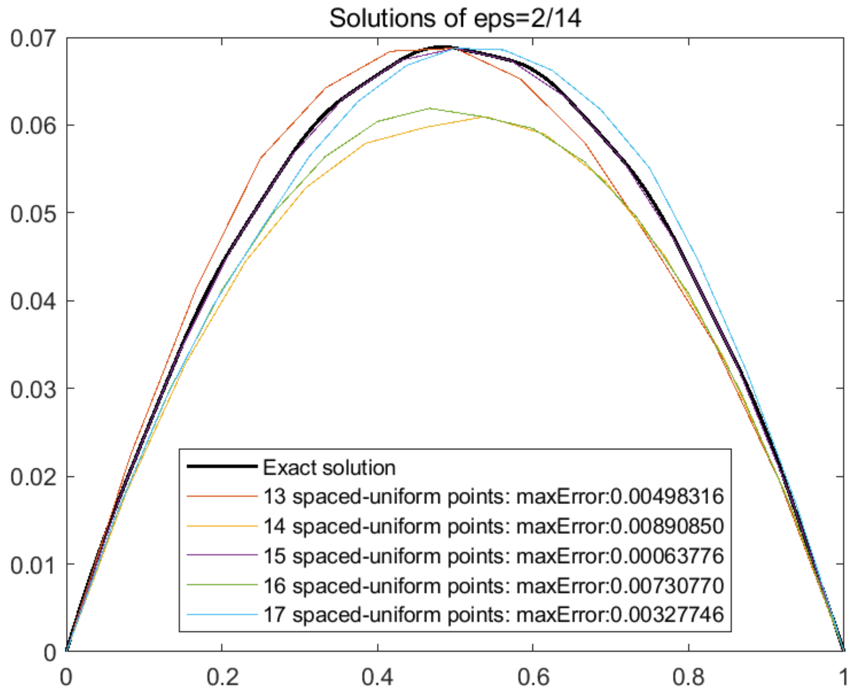

Solutions of eps=2/9
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0
0.2
0.4
0.6
0.8
1
Exact solution
8 spaced-uniform points: maxError:0.02046485
9 spaced-uniform points: maxError:0.01477639
10 spaced-uniform points: maxError:0.00154321
11 spaced-uniform points: maxError:0.01055556
12 spaced-uniform points: maxError:0.01169700

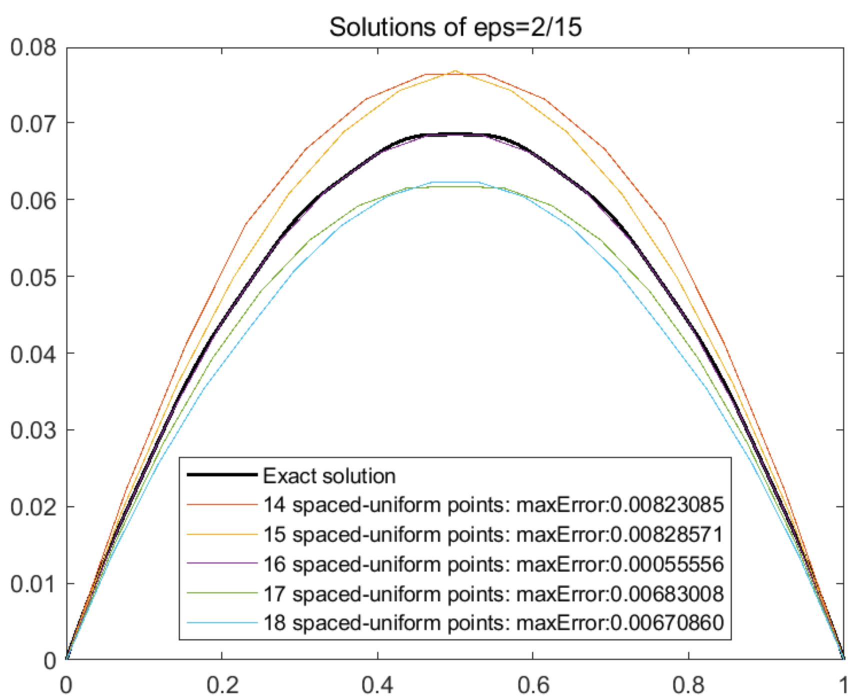

Solutions of eps=2/10
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0
0.2
0.4
0.6
0.8
1
Exact solution
9 spaced-uniform points: maxError:0.00761325
10 spaced-uniform points: maxError:0.01295988
11 spaced-uniform points: maxError:0.00125000
12 spaced-uniform points: maxError:0.00987332
13 spaced-uniform points: maxError:0.00421180

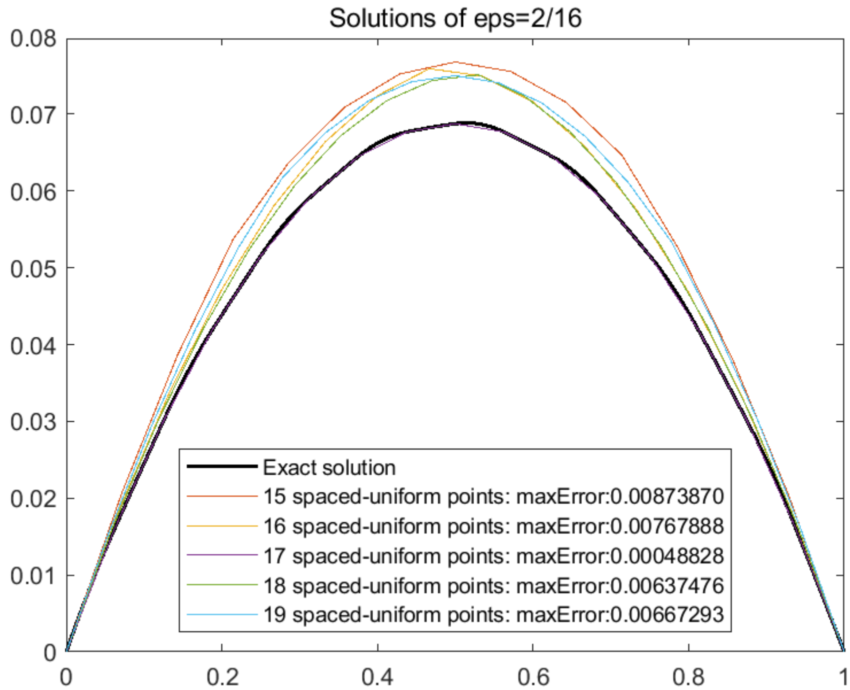

Solutions of eps=2/11
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0
0.2
0.4
0.6
0.8
1
Exact solution
10 spaced-uniform points: maxError:0.01237423
11 spaced-uniform points: maxError:0.01171488
12 spaced-uniform points: maxError:0.00103306
13 spaced-uniform points: maxError:0.00899693
14 spaced-uniform points: maxError:0.00842025

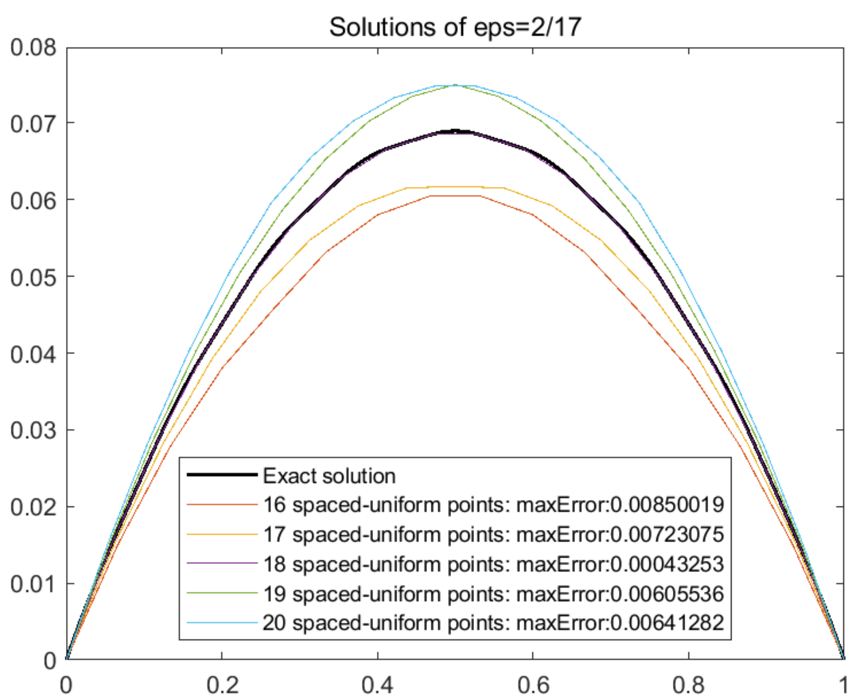

Solutions of eps=2/12
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0
0.2
0.4
0.6
0.8
1
Exact solution
11 spaced-uniform points: maxError:0.01271028
12 spaced-uniform points: maxError:0.01051159
13 spaced-uniform points: maxError:0.00086806
14 spaced-uniform points: maxError:0.00820976
15 spaced-uniform points: maxError:0.00900472

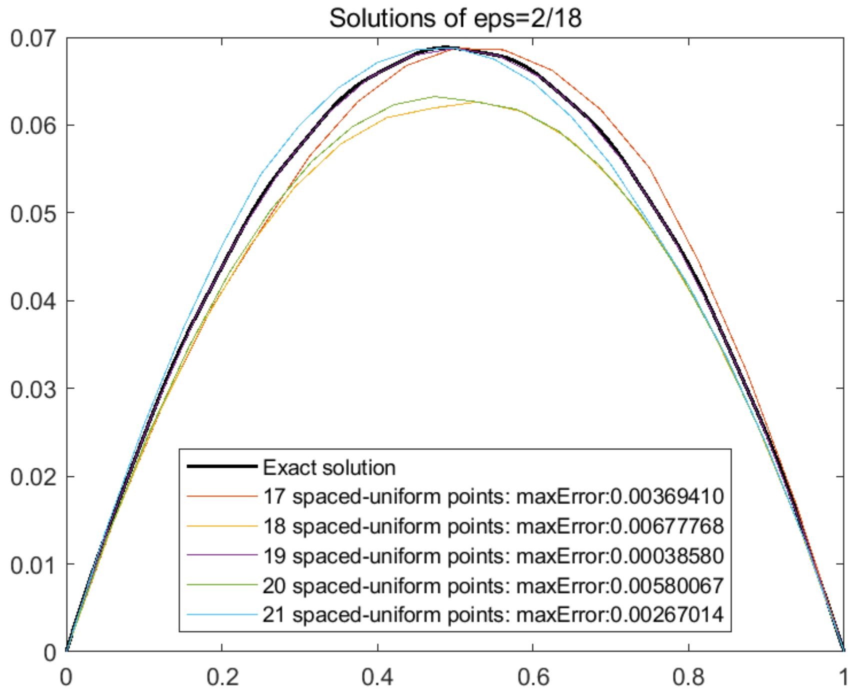

Solutions of eps=2/13
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0
0.2
0.4
0.6
0.8
1
Exact solution
12 spaced-uniform points: maxError:0.01147783
13 spaced-uniform points: maxError:0.00971647
14 spaced-uniform points: maxError:0.00073964
15 spaced-uniform points: maxError:0.00770287
16 spaced-uniform points: maxError:0.00863840

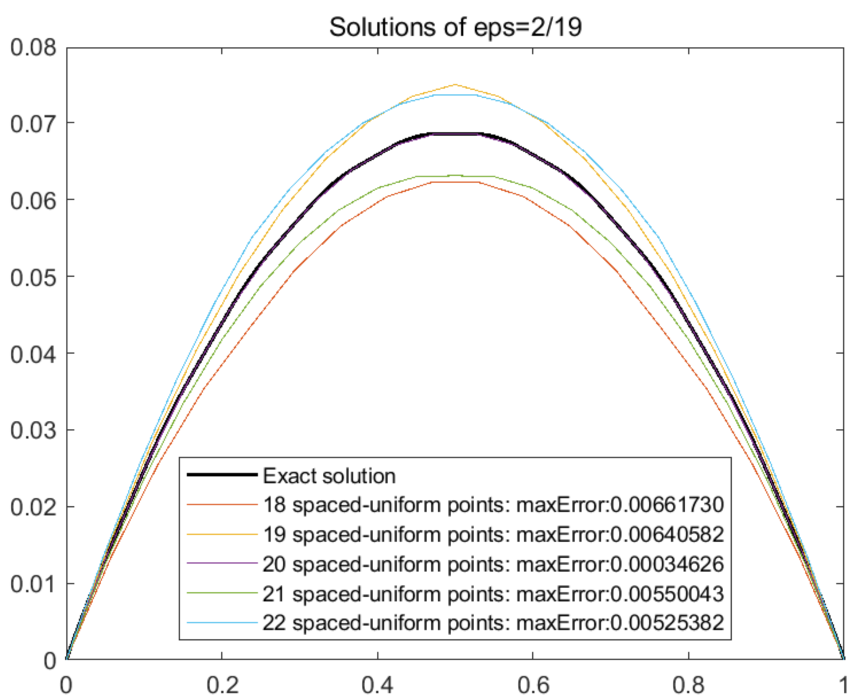

Solutions of eps=2/14
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0
0.2
0.4
0.6
0.8
1
Exact solution
13 spaced-uniform points: maxError:0.00498316
14 spaced-uniform points: maxError:0.00890850
15 spaced-uniform points: maxError:0.00063776
16 spaced-uniform points: maxError:0.00730770
17 spaced-uniform points: maxError:0.00327746

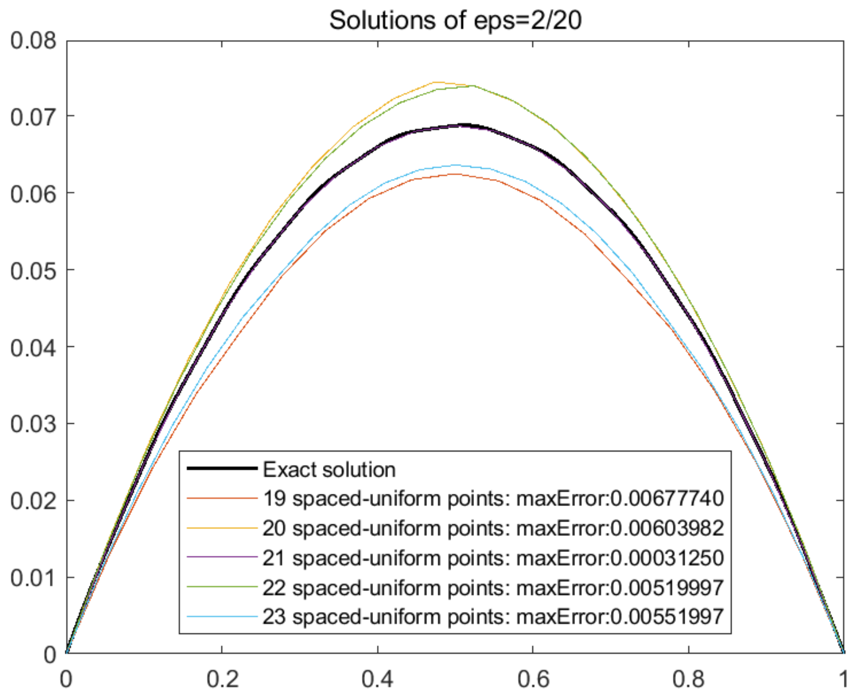

Solutions of eps=2/15
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0
0.2
0.4
0.6
0.8
1
Exact solution
14 spaced-uniform points: maxError:0.00823085
15 spaced-uniform points: maxError:0.00828571
16 spaced-uniform points: maxError:0.00055556
17 spaced-uniform points: maxError:0.00683008
18 spaced-uniform points: maxError:0.00670860


Solutions of eps=2/16
Exact solution
15 spaced-uniform points: maxError:0.00873870
16 spaced-uniform points: maxError:0.00767888
17 spaced-uniform points: maxError:0.00048828
18 spaced-uniform points: maxError:0.00637476
19 spaced-uniform points: maxError:0.00667293


Solutions of eps=2/17
Exact solution
16 spaced-uniform points: maxError:0.00850019
17 spaced-uniform points: maxError:0.00723075
18 spaced-uniform points: maxError:0.00043253
19 spaced-uniform points: maxError:0.00605536
20 spaced-uniform points: maxError:0.00641282


Solutions of eps=2/18
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0
0.2
0.4
0.6
0.8
1
Exact solution
17 spaced-uniform points: maxError:0.00369410
18 spaced-uniform points: maxError:0.00677768
19 spaced-uniform points: maxError:0.00038580
20 spaced-uniform points: maxError:0.00580067
21 spaced-uniform points: maxError:0.00267014


Solutions of eps=2/19
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0
0.2
0.4
0.6
0.8
1
Exact solution
18 spaced-uniform points: maxError:0.00661730
19 spaced-uniform points: maxError:0.00640582
20 spaced-uniform points: maxError:0.00034626
21 spaced-uniform points: maxError:0.00550043
22 spaced-uniform points: maxError:0.00525382


Solutions of eps=2/20
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0
0.2
0.4
0.6
0.8
1
Exact solution
19 spaced-uniform points: maxError:0.00677740
20 spaced-uniform points: maxError:0.00603982
21 spaced-uniform points: maxError:0.00031250
22 spaced-uniform points: maxError:0.00519997
23 spaced-uniform points: maxError:0.00551997