1D Lab Assignment Finite Elements

Wietse Vaes & Lori Trimpeneers

May 20, 2021

On the interval $(0,1)$, we consider a steady-state convection-diffsuion-reaction equation, with homogeneous Neumann boundary conditions:

$$\begin{aligned} -D\frac{d^2u}{dx^2} + \lambda u &= f(x), \\ -D\frac{du}{dx}(0) = 0, \qquad & D\frac{du}{dx}(1) = 0 \end{aligned} \tag{1}$$

Here D, and λ are positive real constants. Further, $f(x)$ is a given function.

The interval [0,1] is divided into $n-1$ elements (where n is a given positive integer), such that $e_i = [x_i, x_{i+1}]$, for $i \in \{1, \ldots, n-1\}$. So element e_i has end points (also called vertices) x_i and x_{i+1}, where we require $x_1 = 0$ and $x_n = 1$ and $h = 1/(n-1)$. Hence there are n gridpoints. In this lab assignment, the participant develops a finite-element code for $1D$ in MATLAB from scratch. The treatment is formal in terms of topology, element matrices and vectors such that the student gets the idea of how finite-element packages are constructed. Once the mesh and topology have been adapted to multidimensional problems, then it is relatively straightforward to adjust the code to higher dimensional problems.

Assignment 1 *Derive a weak form of the above problem (see equation* (1)*), where the order of the spatial derivative is minimised. Take care of the boundary conditions.*

Solution:
There holds that:

$$\begin{aligned} & -Du'' + \lambda u = f \\ & \Rightarrow \varphi f = -\varphi D u'' + \varphi \lambda u \text{ (with } \varphi \text{ test function)} \\ & \Rightarrow \int_0^1 \varphi f dx = \int_0^1 -\varphi D u'' + \varphi \lambda u dx \\ & \Rightarrow \int_0^1 \varphi f dx = \int_0^1 -(D(\varphi u')' - \varphi' u') + \varphi \lambda u dx \text{ (since } (\varphi u')' = \varphi' u' + \varphi u'') \\ & \Rightarrow \int_0^1 \varphi f dx = -\left[\varphi D u'\right]_0^1 + \int_0^1 D u' \varphi' + \lambda \varphi u dx \\ & \Rightarrow \int_0^1 \varphi f dx = \int_0^1 D u' \varphi' + \lambda \varphi u dx \text{ (since } -Du'(0) = 0 \text{ and } Du'(1) = 0). \end{aligned}$$

Therefore the weak formulation is:

$$\text{Find } u \in H^1(0,1) \text{ such that: } \int_0^1 D u' \varphi' + \lambda \varphi u dx = \int_0^1 \varphi f dx, \ \forall \varphi \in H^1(0,1).$$

Assignment 2 *Write the Galerkin Formulation of the weak form as derived in the previous assignment for a general number of elements given by n (hence $x_n = 1$). Give the Galerkin eqution, that is, the linear system in terms of*

$$S\underline{u} = \underline{f},$$

all exppressed in the basis-functions, $f(x)$, λ and D.

Solution:
If we were to approximate $u(x) \approx u^n(x) := \sum_{i=1}^n c_i\varphi_i(x)$ with $c_i \in \mathbb{R}$ and $\varphi_i \in H^1((0,1))$, then:

$$\int_0^1 D\left(\sum_{i=1}^n c_i\varphi_i'\right)\varphi' + \lambda\varphi\left(\sum_{i=1}^n c_i\varphi_i\right)dx = \sum_{i=1}^n c_i\int_0^1 D\varphi_i'\varphi' + \lambda\varphi_i\varphi dx = \int_0^1 \varphi f dx.$$

Due to the fact that this holds $\forall\varphi \in H^1((0,1))$, it will also hold for a φ_j which is used as a basis when making u^n: $\sum_{i=1}^n c_i\int_0^1 D\varphi_i'\varphi_j' + \lambda\varphi_i\varphi_j dx = \int_0^1 \varphi_j f dx$, for $j = 1, 2, ..., n$. Thus a system can be made:

$$\begin{bmatrix} \int_0^1 D\varphi_1'\varphi_1' + \lambda\varphi_1\varphi_1 dx & \int_0^1 D\varphi_1'\varphi_2' + \lambda\varphi_1\varphi_2 dx & \dots & \int_0^1 D\varphi_1'\varphi_n' + \lambda\varphi_1\varphi_n dx \\ \int_0^1 D\varphi_2'\varphi_1' + \lambda\varphi_2\varphi_1 dx & \int_0^1 D\varphi_2'\varphi_2' + \lambda\varphi_2\varphi_2 dx & \dots & \int_0^1 D\varphi_2'\varphi_n' + \lambda\varphi_2\varphi_n dx \\ \vdots & \vdots & \ddots & \vdots \\ \int_0^1 D\varphi_n'\varphi_1' + \lambda\varphi_n\varphi_1 dx & \int_0^1 D\varphi_n'\varphi_2' + \lambda\varphi_n\varphi_2 dx & \dots & \int_0^1 D\varphi_n'\varphi_n' + \lambda\varphi_n\varphi_n dx \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} \int_0^1 \varphi_1 f dx \\ \int_0^1 \varphi_2 f dx \\ \vdots \\ \int_0^1 \varphi_n f dx \end{bmatrix}.$$

We now define the following matrices: $S_{ij} = \int_0^1 D\varphi_i'\varphi_j' + \lambda\varphi_i\varphi_j dx$, $\underline{u}_j = c_j$ and $\underline{f}_i = \int_0^1 \varphi_i f dx \quad \forall i, j \in \{1, \dots, n\}$. Note that $u(x_j) = c_j$.

Assignment 3 *Write a MATLAB routine, called* GenerateMesh.m *that generates an equidistant distribution of meshpoints over the interval [0,1], where $x_1 = 0$ and $x_n = 1$ and $h = \frac{1}{n-1}$.*

Solution:

```
function grid = GenerateMesh(a,b,n)
grid = linspace(a,b,n);
```

For $x_1 = 0$ and $x_n = 1$ use 'linspace(0,1,n)', but we wanted to make it more general.
Due to all gridpoints being equidistant, we will now define $\Delta x := x_{j+1} - x_j$.
We will use this in our theoretical answers. However, in our MATLAB code we will use $x_{j+1} - x_j$ as we want to make the code as general as possible

Assignment 4 *Write a routine, called* GenerateTopology.m*, that generates a two dimensional array, called* elmat*, which contains the indices of the vertices of each element*

Solution:

```
function elmat = GenerateTopology(n)
i = 1:(n-1);
elmat(i,1) = i; elmat(i, 2) = i+1;
```

Assignment 5 *Compute the element matrix, S_{elem} over a generic line element e_k.*

Solution:
Suppose that we use the basis seen in the course: $\varphi_j := \begin{cases} \frac{x-x_{j-1}}{x_j-x_{j-1}}, & x \in [x_{j-1}, x_j) \\ \frac{x_{j+1}-x}{x_{j+1}-x_j}, & x \in [x_j, x_{j+1}] \\ 0, & \text{elsewhere} \end{cases}$.

There holds:

$$S_{ij} = \int_0^1 D\varphi_i'\varphi_j' + \lambda\varphi_i\varphi_j dx = \sum_{k=1}^{n-1}\int_{e_i} \varphi_i'\varphi_j' + \lambda\varphi_i\varphi_j dx$$

If we were to now integrate over $e_k = e_i$ instead of $[0,1]$ (when calculating the matrix S) we would get the following. We first notice that:

For $i, j \notin \{k, k+1\}$:
$\int_{x_k}^{x_{k+1}} D\varphi_i'\varphi_j' + \lambda\varphi_i\varphi_j dx = \int_0^1 D \cdot 0 \cdot 0 + \lambda \cdot 0 \cdot 0 dx = 0.$

For $S_{11}^{e_k}$:
$\int_{x_k}^{x_{k+1}} D{\varphi_k'}^2 + \lambda\varphi_k^2 dx = \int_{x_k}^{x_{k+1}} D\left(\frac{-1}{x_{k+1}-x_k}\right)^2 dx + \int_{x_k}^{x_{k+1}} \lambda\left(\frac{x_{k+1}-x}{x_{k+1}-x_k}\right)^2 dx = \frac{D}{x_{k+1}-x_k} - \lambda\left[\frac{(x_{k+1}-x)^3}{3(x_{k+1}-x_k)^2}\right]_{x_k}^{x_{k+1}}$
$= \frac{D}{x_{k+1}-x_k} + \lambda\frac{x_{k+1}-x_k}{3} = \frac{3D+\lambda\Delta x^2}{3\Delta x}$

For $S_{22}^{e_k}$:
$\int_{x_k}^{x_{k+1}} D{\varphi_{k+1}'}^2 + \lambda\varphi_{k+1}^2 dx = \int_{x_k}^{x_{k+1}} D\left(\frac{1}{x_{k+1}-x_k}\right)^2 dx + \int_{x_k}^{x_{k+1}} \lambda\left(\frac{x-x_k}{x_{k+1}-x_k}\right)^2 dx = \frac{D}{x_{k+1}-x_k} + \lambda\left[\frac{(x-x_k)^3}{3(x_{k+1}-x_k)^2}\right]_{x_k}^{x_{k+1}}$
$= \frac{D}{x_{k+1}-x_k} + \lambda\frac{x_{k+1}-x_k}{3} = \frac{3D+\lambda\Delta x^2}{3\Delta x}$

For $S_{12}^{e_k}$ or $S_{21}^{e_k}$:
$\int_{x_k}^{x_{k+1}} D\varphi_{k+1}'\varphi_k' + \lambda\varphi_{k+1}\varphi_k dx = \int_{x_k}^{x_{k+1}} D\frac{-1}{(x_{k+1}-x_k)^2} + \lambda\frac{(x-x_k)(x_{k+1}-x)}{(x_{k+1}-x_k)^2} dx = \frac{-D}{\Delta x} + \lambda\int_0^{\Delta x}\frac{u(\Delta x-u)}{\Delta x^2} = \frac{-D}{\Delta x} +$
$\frac{\lambda}{\Delta x^2}\left[\frac{\Delta x}{2}u^2 - \frac{u^3}{3}\right]_0^{\Delta x} = \frac{-D}{\Delta x} + \lambda\left(\frac{\Delta x}{2} - \frac{\Delta x}{3}\right) = \frac{-D}{\Delta x} + \frac{\lambda\Delta x}{6} = \frac{-6D+\lambda\Delta x^2}{6\Delta x}$

The 2×2 matrix that contains non-zero points will thus be:

$$S_{elem}^{e_k} = \frac{1}{3\Delta x}\begin{bmatrix} 3D+\lambda\Delta x^2 & \frac{-6D+\lambda\Delta x^2}{2} \\ \frac{-6D+\lambda\Delta x^2}{2} & 3D+\lambda\Delta x^2 \end{bmatrix} = \frac{D}{\Delta x}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} + \frac{\lambda\Delta x}{6}\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

Assignment 6 *Write a matlab routine, called* GenerateElementMatrix.m*, in which S_{elem} (2×2-matrix) is generated.*

Solution:

```
function S_elem = GenerateElementMatrix(karak, x_k, x_kk)
Dx = x_kk-x_k; D = karak(1); lambda = karak(2);
Rij = D*[1, -1]/Dx + lambda*Dx*[2, 1]/6;
S_elem(1,:) = Rij; S_elem(2,:) = fliplr(Rij);
```

Assignment 7 *Write a matlab routine, called* AssembleMatrix.m*, that performs this summation, such that S is first initialized as a zero $n-by-n$ matrix and subsequently:*

$$S(elmat(i,j), elmat(i,k)) = S(elmat(i,j), elmat(i,k)) + S_{elem}(j,k),$$

for $j, k \in \{1, 2\}$ over all elements $i \in \{1, \ldots, n-1\}$.

Solution:

```
function S = AssembleMatrix(elmat, karakteristics, grid)
n = length(grid); S = zeros(n);
for i = 1:n-1
    S_elem = GenerateElementMatrix(karakteristics, grid(i), grid(i+1));
    for j = 1:2
        for k = 1:2
            S(elmat(i,j),elmat(i,k)) = S(elmat(i,j),elmat(i,k)) + S_elem(j,k);
        end
    end
end
```

Assignment 8 *Compute the element vector over a generic line-element.*

Solution:
From assignment 2, we know that: $\underline{f}_i = \int_0^1 \varphi_i f dx$. If we were to limit the integral to e_k, we get: $\int_{x_k}^{x_{k+1}} \varphi_k f dx$.
We first notice that: For $i \notin \{k, k+1\}$:
$\int_{x_k}^{x_{k+1}} \varphi_k f dx = \int_{x_k}^{x_{k+1}} 0 f dx = 0$.

For $i = k$:
Using Newton Cotes, we get: $\int_{x_k}^{x_{k+1}} \varphi_k f dx = \Delta x \frac{\varphi_k(x_k)f(x_k)+\varphi_k(x_{k+1})f(x_{k+1})}{2} = \Delta x \frac{1 \cdot f(x_k)}{2} = \frac{\Delta x}{2} f(x_k)$.

For $i = k+1$:
Using Newton Cotes, we get: $\int_{x_k}^{x_{k+1}} \varphi_{k+1} f dx = \Delta x \frac{\varphi_{k+1}(x_k)f(x_k)+\varphi_{k+1}(x_{k+1})f(x_{k+1})}{2} = \Delta x \frac{1 \cdot f(x_{k+1})}{2} = \frac{\Delta x}{2} f(x_{k+1})$.

Thus: $f_{elem}^{e_k} = \begin{bmatrix} \frac{\Delta x}{2} f(x_k) \\ \frac{\Delta x}{2} f(x_{k+1}) \end{bmatrix}$

Assignment 9 *Implementation of the right-hand vector:*

1. *Write a matlab routine, called GenerateElementVector.m, that gives the vector f_{elem} (column vector of length 2). in which $f_{elem}(1)$ and $f_{elem}(2)$ respectively provide information about node i and node $i+1$, which are the vertices of element e_i. This is needed for all elements. Use $f(x) = 1$ here.*

   Solution:
   We do it for a general function f, so the input for f needs to be $@(x)f(x)$.

```
function f_elem = GenerateElementVector(f , x_k, x_kk)
Dx = x_kk - x_k;
f_elem = [Dx/2*f(x_k) Dx/2*f(x_kk)];
```

2. *Write a matlab routine, called AssembleVector.m, that performs the following summation after setting $f = zeros(n, 1)$:*
$$f(elmat(i, j)) = f(elmat(i, j)) + f_{elem}(j),$$
for $j \in \{1, 2\}$ over all elements $i \in \{1, \ldots, n-1\}$.

   Solution:

```
function f = AssembleVector(elmat,func, grid)
n = length(grid); f = zeros(n,1);
for i = 1:(n-1)
    f_elem = GenerateElementVector(func , grid(i), grid(i+1));
    for j = 1:2
        f(elmat(i,j))=f(elmat(i,j))+f_elem(j);
    end
end
```

Assignment 10 *Run the assembly routines to get the matrix S and vector f for $n = 100$*

Solution:
Here is $f = @(x)1$. First we use the GenerateMesh function with $a = 0, b = 1, n = 100$, then the GenerateTopology function with $n = 100$ and then the AssembleMatrix and AssembleVector functions. Then we get

$$S = \begin{bmatrix} 99.0034 & -98.9983 & 0 & 0 & & 0 \\ -98.9983 & 198.0067 & -98.9983 & 0 & \cdots & 0 \\ 0 & -98.9983 & 198.0067 & -98.9983 & & 0 \\ \vdots & & & \ddots & & \vdots \\ 0 & & -98.9983 & 198.0067 & -98.9983 & 0 \\ 0 & \cdots & 0 & -98.9983 & 198.0067 & -98.9983 \\ 0 & & 0 & 0 & -98.9983 & 99.0034 \end{bmatrix} \text{ and } \underline{f} = \begin{bmatrix} 0.0051 \\ 0.0101 \\ \vdots \\ 0.0101 \\ 0.0051 \end{bmatrix}.$$

Assignment 11 *Write the main program that gives the finite-element solution. Call the main program* femsolveld.m.

Solution:

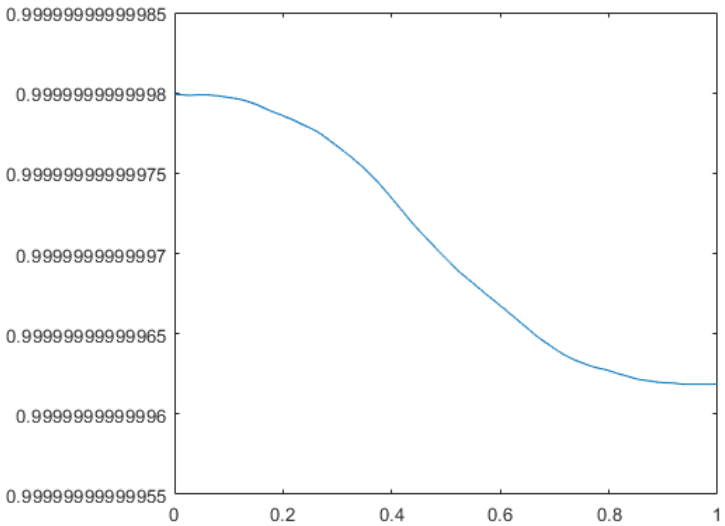
```
function u = femsolveld(n,D,lambda,func, a, b)
grid = GenerateMesh(a,b,n);
elmat = GenerateTopology(n);

S = AssembleMatrix(elmat, [D, lambda], grid);
f = AssembleVector(elmat,func, grid);

u = S\f;
plot(grid,u)
```

Assignment 12 *Compute the Finite Element solution u for $f(x) = 1$, $D = 1$, $\lambda = 1$ and $n = 100$ by using $u = S\backslash f$ in matlab. Plot the solution. Is this what you would expect?*

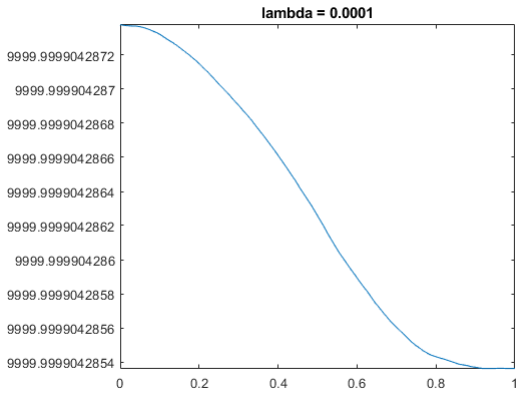
Solution:

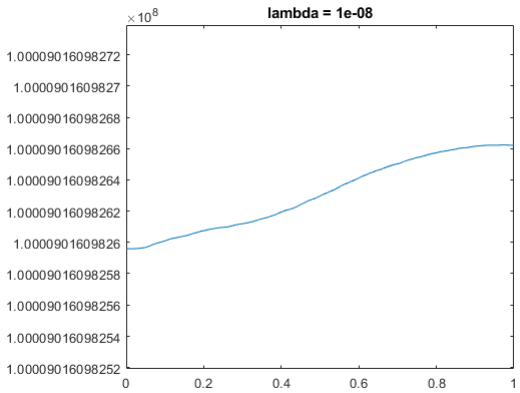


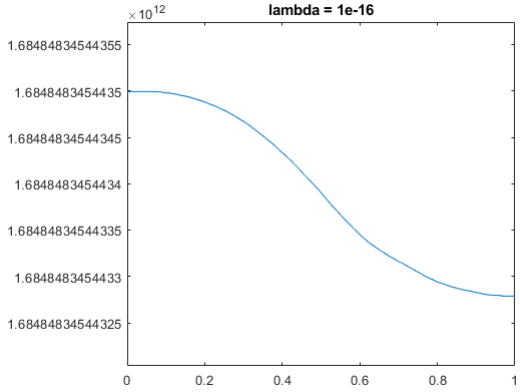
We notice that it flattens on the boundary, which means that the boundary condition is most likely fulfilled. We did not expect that the function would go from being concave to convex and that the transition would be at $u = 0.9...97$ and not $u = 1$. This because $u = 1 \Rightarrow u'' = 0$, $u > 1 \Rightarrow u'' > 0$ (convex) and $u < 1 \Rightarrow u'' < 0$ (concave). The sign of the second derivative of the approximated solution and the analytic solution aren't the same. What we really expected was for the curve to be a flat line at $u = 1$. u is near 1. The approximation however is close to it (it's off by a small amount). Thus, we blame everything on rounding errors.

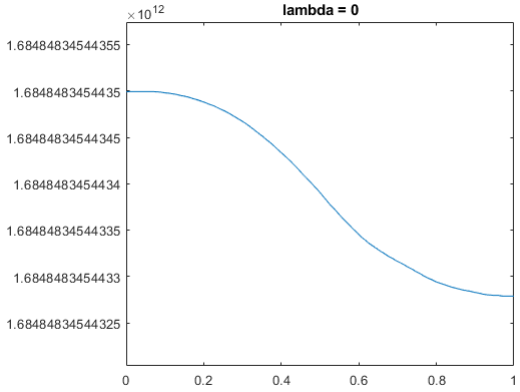
Assignment 13 *Take $\lambda = 10^{-4}, 10^{-8}, 10^{-12}$ and 0, what do you see? Explain your results.*

Solution:
If we were to use an n of 100 we would get:



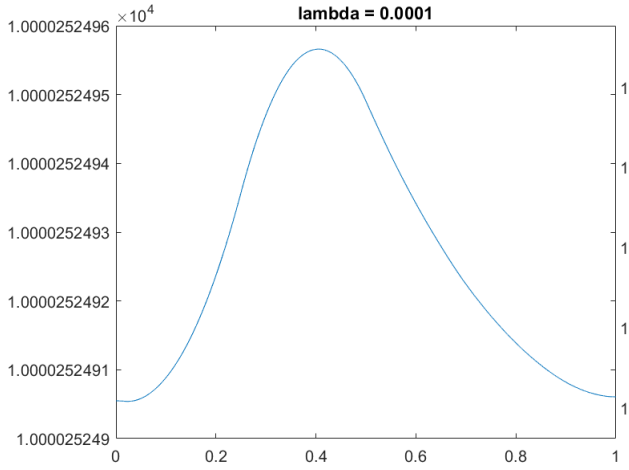


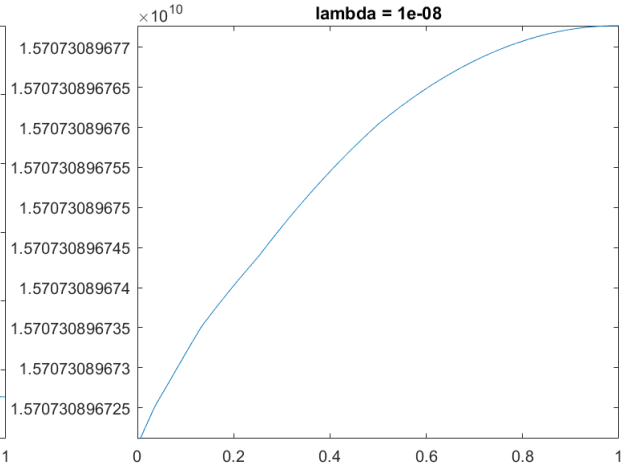


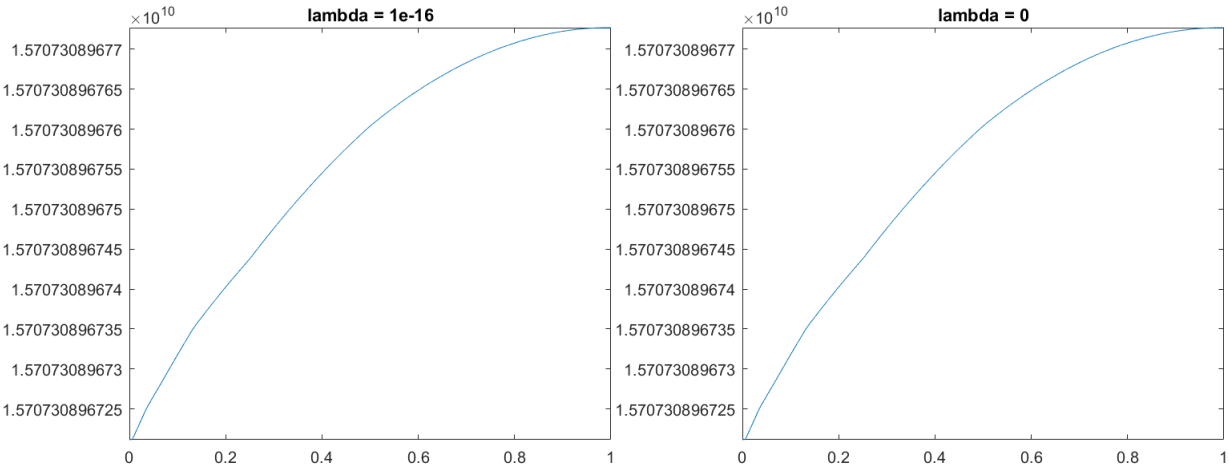


This doesn't seem right. We would geuss that, the lower λ would be, the less of an influence u would have on the ODE and it would slowely go to $\lambda = 0$. For $\lambda = 0$ we would suspect that the whole curve would be concave (u" $= -1 < 0$) except at a border as both $u'(0) = u'(1) = 0$, it should have been flat there (which it is). We also suspected the solution not to be smooth, which it is not.

However, if we were to take a larger n, for example $n = 10000$, we would get the following:







Here we see that the first and last seem to be as expected. We would expect, respectively, a straight line at $u = 10^4$, $u = 10^8$ and $u = 10^{16}$ and what we get for the last one (both ends flatten when zoomed in on). We can once again blame the first one on rounding errors, the following three however can be blamed on the matrix S. Here we notice a warning in MALTAB indicating that the matrices are almost singular, which indirectly indicates rounding errors.

Assignment 14 *Choose $f(x) = \sin(20x)$, do some experiments with several values of n ($n = 10, 20, 40, 80, 160$). Plot the solutions for the varius numbers of gridnodes in one plot. Explain what you see.*

Solution:

We first derive the exact analytical solution for: $\begin{cases} -\frac{d^2u}{dx^2} + u = \sin(20x), \\ -\frac{du}{dx}(0) = 0, \qquad \frac{du}{dx}(1) = 0 \end{cases}$

We first do the homogeneous solution:

$$\begin{aligned} &-u_h''(x) + u_h(x) = 0 \\ \Rightarrow &-\lambda^2 + 1 = 0 \Rightarrow \lambda = \pm 1 \\ \Rightarrow &u_h(x) = c_1 e^x + c_2 e^{-x} \end{aligned}$$

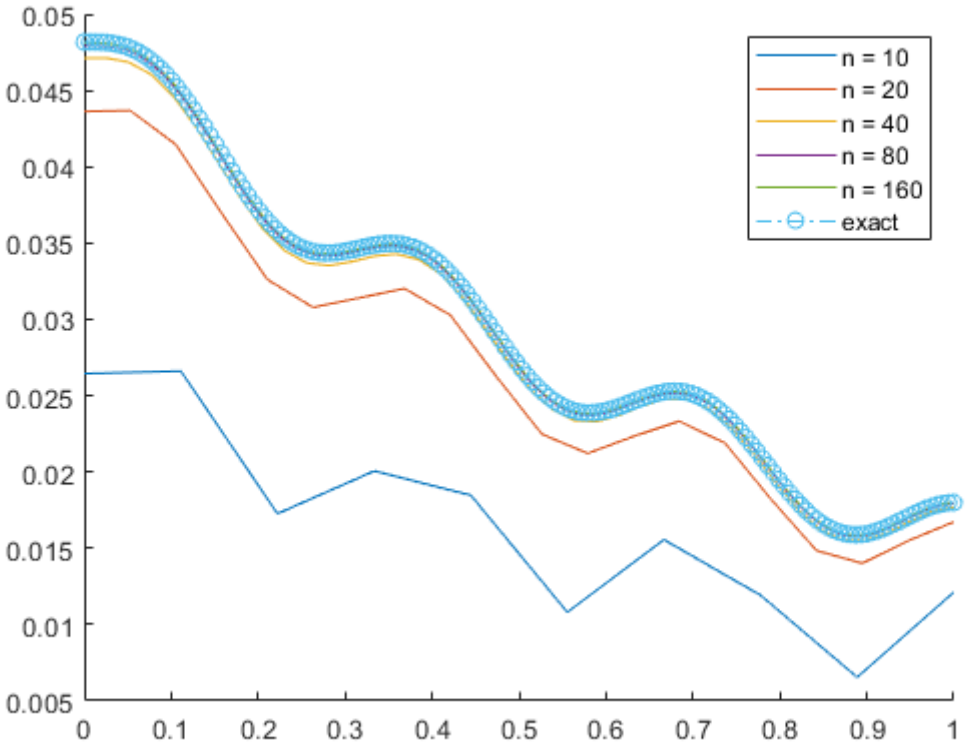
Now for the particular solution, we predict that u_p will have the form of: $u_p = c_3 \sin(20x) + c_4 \cos(20x)$

$$\begin{aligned} &400c_3 \sin(20x) + 400c_4 \cos(20x) + c_3 \sin(20x) + c_4 \cos(20x) = \sin(20x) \\ \Rightarrow &401c_3 \sin(20x) + 401c_4 \cos(20x) = \sin(20x) \\ \Rightarrow &c_3 = \frac{1}{401} \qquad \& \qquad c_4 = 0 \\ \Rightarrow &u_p = \frac{1}{401} \sin(20x) \end{aligned}$$

Thus: $u(x) = c_1 e^x + c_2 e^{-x} + \frac{1}{401} \sin(20x)$. Because $u'(0) = u'(1) = 0$, we know that:

$$\begin{cases} c_1 - c_2 + \frac{20}{401} = 0 \\ c_1 e - c_2 e^{-1} + \frac{20}{401} \cos(20) = 0 \end{cases} \Rightarrow \begin{bmatrix} 1 & -1 \\ e & -e^{-1} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} \frac{-20}{401} \\ \frac{-20}{401} \cos(20) \end{bmatrix} \Rightarrow \begin{cases} c_1 = -\frac{20}{401} \frac{e\cos(20)-1}{e^2-1} \\ c_2 = \frac{20}{401} \frac{e(e-\cos(20))}{e^2-1} \end{cases}$$

We suspect the solution to be: $u = -\frac{20(e\cos(20)-1)}{401(e^2-1)} e^x + \frac{20e(e-\cos(20))}{401(e^2-1)} e^{-x} + \frac{1}{401} \sin(20x)$
If we were to now plot everything, we would get:



When more gridpoints are used, the solution becomes smoother. We also see that, the larger the n, the better the approximation. This is evident from the errors ($\|\tilde{u} - u\|_2$) which are respectively: 0.0449, 0.0131, 0.0043, 0.0015 and 0.00051732.