FYS-4150 Project 2

Yevhenii Volkov

September 27, 2022

Link to github repository
https://github.com/YVol322/Comp-Phys/tree/Project-2/Project2

1 Problem

Show that $\gamma \frac{d^2u(x)}{dx^2} = -Fu(x)$ can be written as $A\vec{v} = \lambda\vec{v}$.

$\hat{x} \equiv \frac{x}{L} \Rightarrow x = \hat{x}L;$

$\frac{d^2u(\hat{x})}{d\hat{x}^2} = \frac{d^2u(\hat{x})}{d(\hat{x}^2L^2)} = \frac{1}{L^2}\frac{d^2u(\hat{x})}{d\hat{x}^2};$

$\frac{\gamma}{L^2}\frac{d^2u(\hat{x})}{d\hat{x}^2} = -Fu(\hat{x}),\ \lambda \equiv \frac{FL^2}{\gamma};$

$\frac{d^2u(\hat{x})}{d\hat{x}^2} = -\lambda u(\hat{x})$. Discretising this equation:

$\frac{d^2u(\hat{x}_i)}{d\hat{x}^2} = \frac{u(\hat{x}_{i+1})-2u(\hat{x}_i)+u(\hat{x}_{i-1})}{h^2} + O(h^2);$

$\frac{d^2v(\hat{x}_i)}{d\hat{x}^2} \approx \frac{v(\hat{x}_{i+1})-2v(\hat{x}_i)+v(\hat{x}_{i-1})}{h^2}.$

$i = 1 : \frac{v(\hat{x}_0)-2v(\hat{x}_1)+v(\hat{x}_2)}{h^2} = -\lambda v(\hat{x}_1);$

$v(\hat{x}_0) = 0;$

$i = 1 : \frac{2v(\hat{x}_1)-v(\hat{x}_2)}{h^2} = \lambda v(\hat{x}_1);$

$i = 2 : \frac{-v(\hat{x}_1)+2v(\hat{x}_2)-v(\hat{x}_3)}{h^2} = \lambda v(\hat{x}_2);$

...

$i = n : \frac{-v(\hat{x}_{n-1})+2v(\hat{x}_n))}{h^2} = \lambda v(\hat{x}_n);$

So we can write this equation in the matrix form $A\vec{v} = \lambda\vec{v}$, where A is tridiagonal matrix of size $(n-1) \times (n-1)$ and $\vec{v}$ is an eigenvector, λ is an eigenvalues.

2 Problem

C++ file to this problem pr2.cpp can be found om github repository.

3 Problem

C++ file to this problem pr3.cpp can be found on github repository.

4 Problem

C++ file to this problem pr4.cpp can be found on github repository.

5 Problem

Files to this problem can be found in github repository on folder pr5.

5.1 Problem

C++ file to this problem pr5.cpp can be found on github repo in folder pr5.
Data generated by this programme is saved to pr5table.csv.
Plot of this data and 2nd order polynomial fit pr5.png is generated by plotting script p5.py based on data from table pr5table.py.
From the plot pr5.png we can see that $N(j)$ is approximated by 2nd order polynomial with approximation coefficients: $[1.69961807, 10.31209, 14.86858518]$

5.2 Problem

C++ file to this problem pr5.cpp can be found on github repo in folder pr5.
Data generated by this programme is saved to pr5btable.csv.
Plot of this data and 2nd order polynomial fit pr5b.png is generated by plotting script p5.py based on data from table pr5btable.py.
From the plot pr5b.png we can see that $N(j)$ is approximated by 2nd order polynomial with approximation coefficients: $[1.7016317, 8.49277389, 13.52447552]$

We can see, that Jacobi's rotation method work approximately with the same amount of iterations for dense A matrix and for tridiagonal A matrix.
In my opinion, the reason is that this method does not really care how many zeros in matrix do we have. It cares only about largest off-diagonal elements, and this method transforming zeros in tridiagonal A matrix to non zeros.

Figure 1: Tridiagonal A matrix.

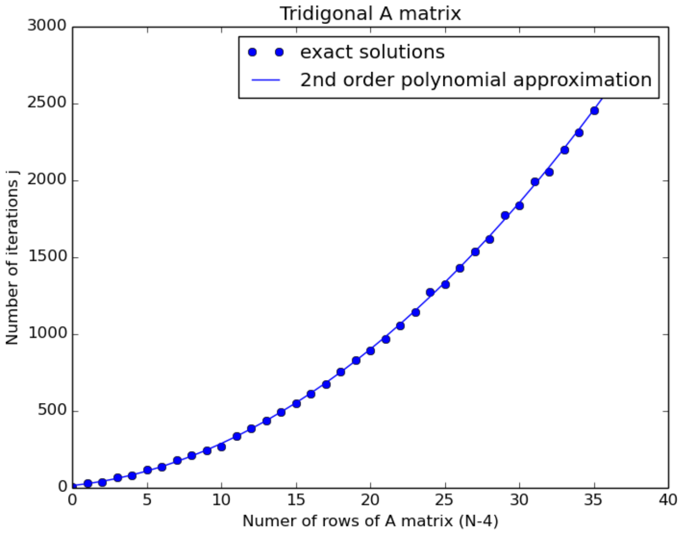
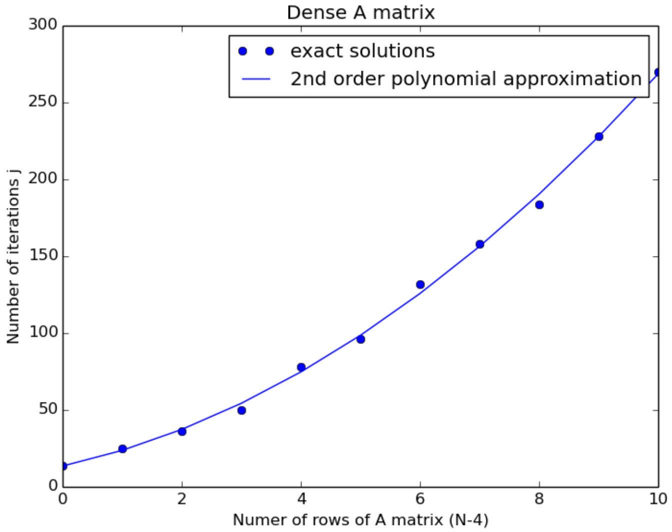


Figure 2: Dense A matrix



6 Problem

Files to this problem can be found in github repository on folder pr6.

6.1 Problem

C++ file to this problem pr6.cpp can be found on github repo in folder pr6.
Data generated by this programme is saved to pr6table.csv.
Plots of this data pr6i.png is generated by plotting script p6.py based on data from table pr6table.py.

6.2 Problem

C++ file to this problem pr6b.cpp can be found on github repo in folder pr6.
Data generated by this programme is saved to pr6btable.csv.
Plots of this data pr6bi.png is generated by plotting script p6b.py based on data from table pr6btable.py.

Figure 3: Solution of dif. eq. for the lowest eigval (antisymetric), $n = 10$.

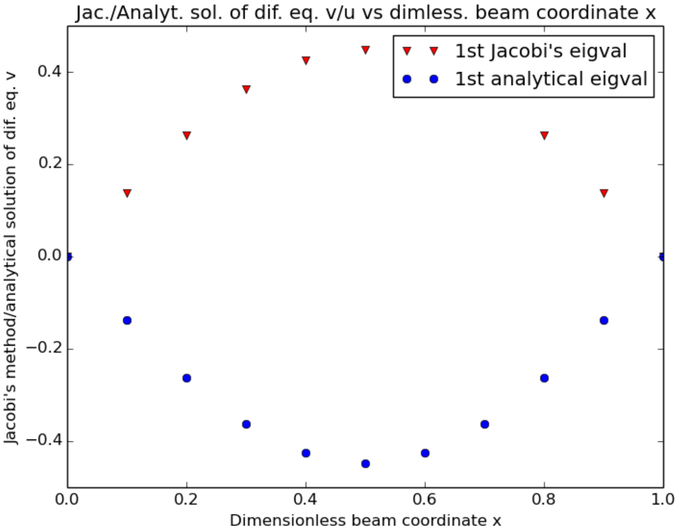


Figure 4: Solution of dif. eq. for the lowest eigval (symmetric)), $n = 10$.

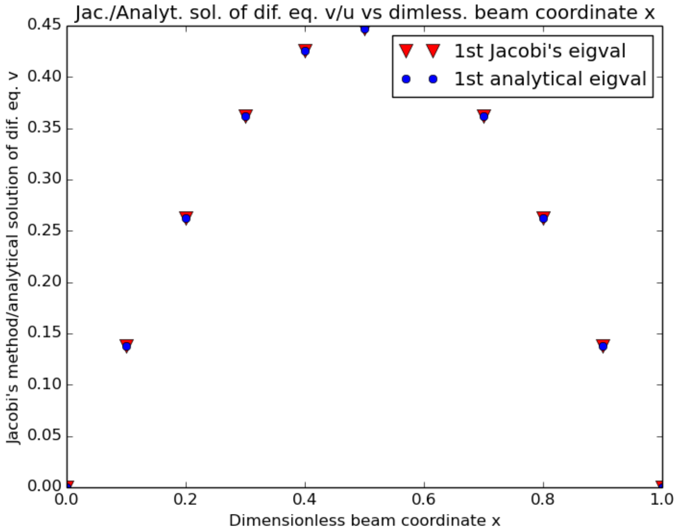


Figure 5: Solution of dif. eq. for the 2nd lowest eigval (antisymmetric)), $n = 10$.

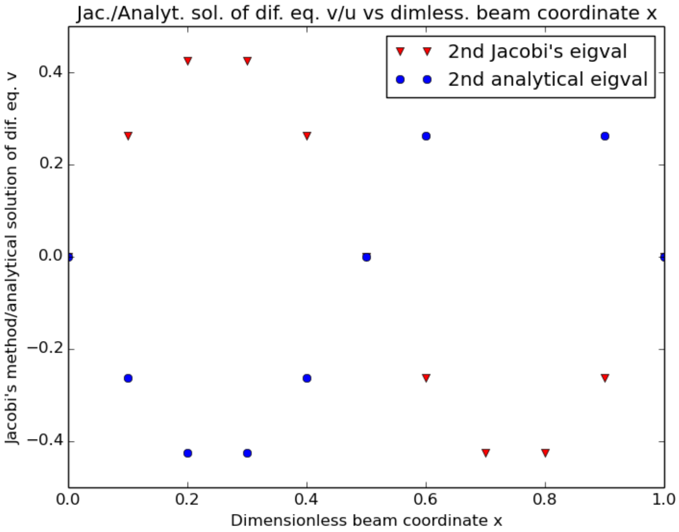


Figure 6: Solution of dif. eq. for the 2nd lowest eigval (symmetric)), $n = 10$.

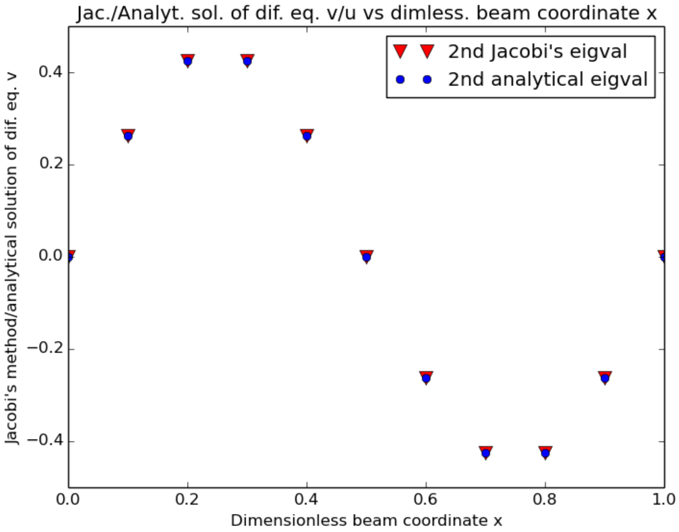


Figure 7: Solution of dif. eq. for the 3rd lowest eigval (antisymmetric)), $n = 10$.

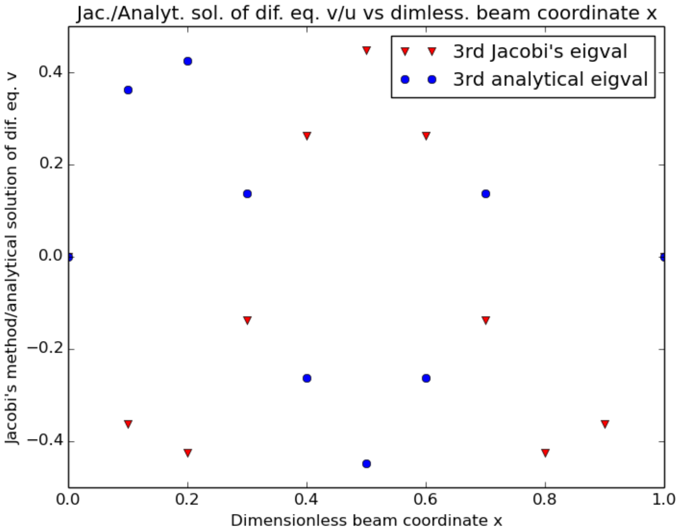


Figure 8: Solution of dif. eq. for the 3rd lowest eigval (symmetric)), $n = 10$.

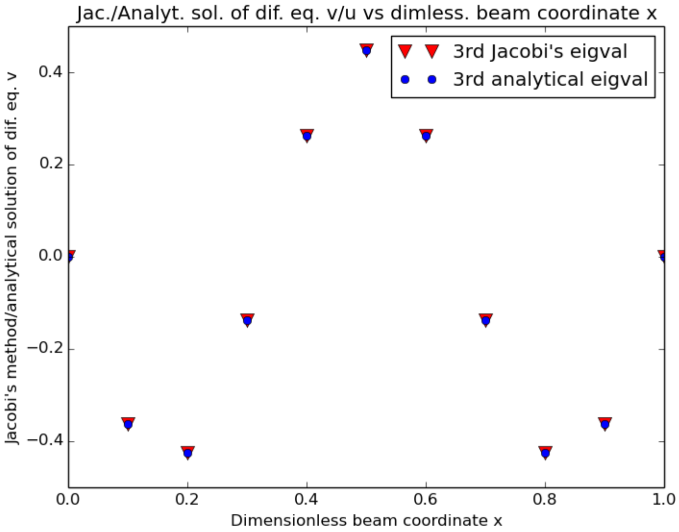


Figure 9: Solution of dif. eq. for the lowest eigval (antisymmetric).), $n = 100$

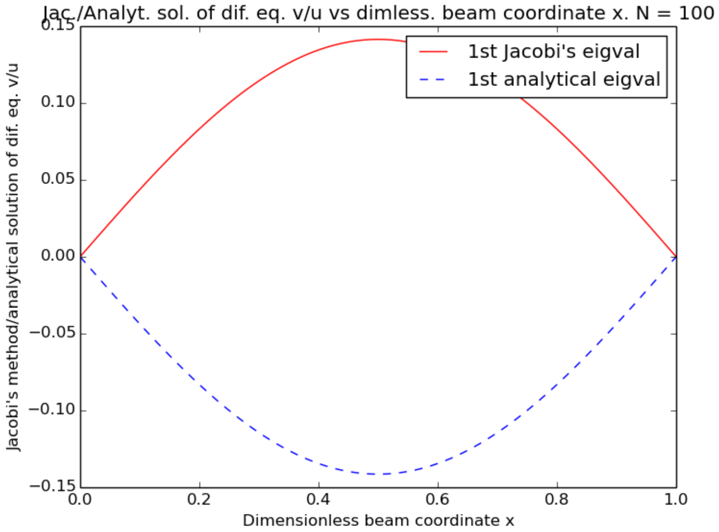


Figure 10: Solution of dif. eq. for the lowest eigval (symmetric).), $n = 100$

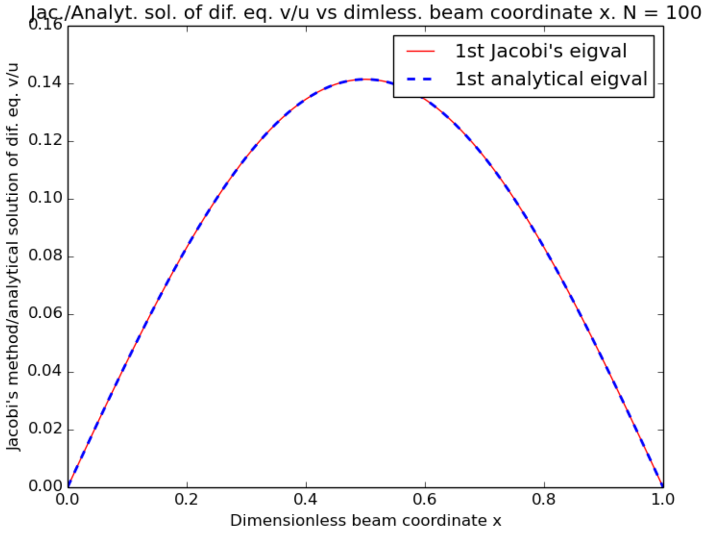


Figure 11: Solution of dif. eq. for the 2nd lowest eigval (antisymmetric).), $n = 100$

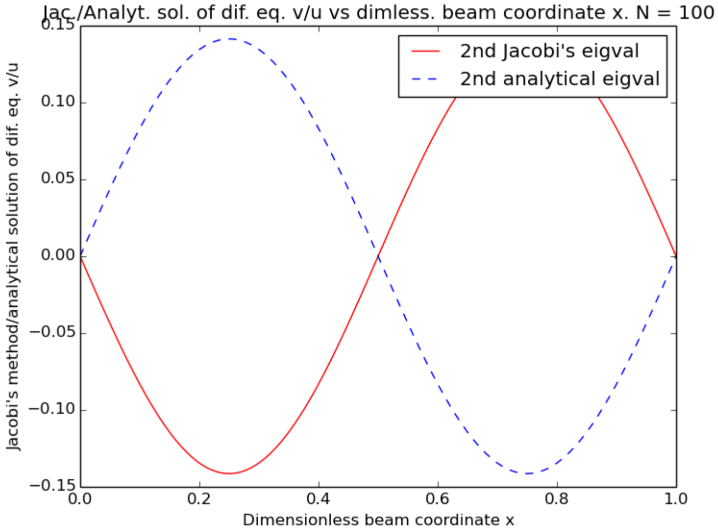


Figure 12: Solution of dif. eq. for the 2nd lowest eigval (symmetric).), $n = 100$

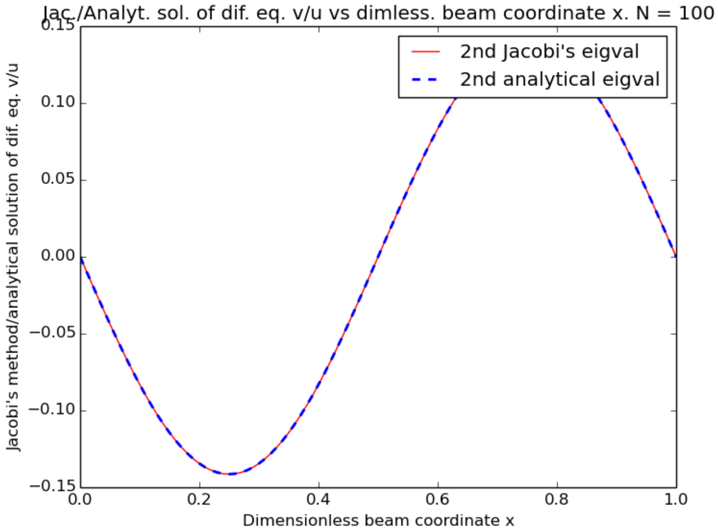


Figure 13: Solution of dif. eq. for the 3rd lowest eigval (antisymmetric).), $n = 10$

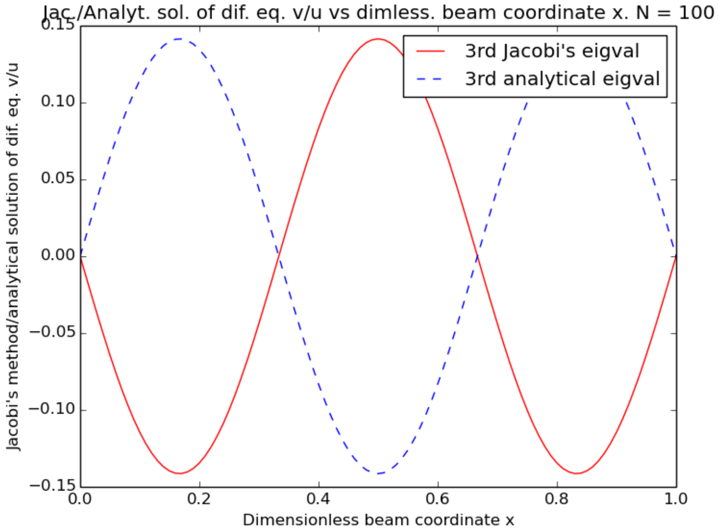
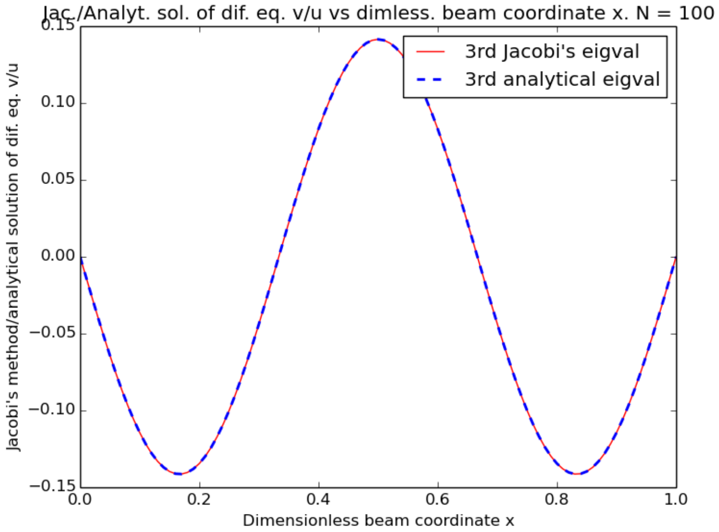


Figure 14: Solution of dif. eq. for the 3rd lowest eigval (symmetric).), $n = 10$


Jac./Analyt. sol. of dif. eq. v/u vs dimless. beam coordinate x. N = 100
Jacobi's method/analytical solution of dif. eq. v/u
Dimensionless beam coordinate x
3rd Jacobi's eigval
3rd analytical eigval