Hand in 5, part 3 of 3 - The Jacobi Method

1 Solving the ODE-BVP with the Jacobi method

Consider the ODE-BVP that we worked with during Lab 5, specifically with the time-dependent right hand side that we worked with during Lab 5, part 2. Include your derivation/reasoning in the hand in.

1. Implement a python function for the jacobi-method in your code from Lab 5, part 2. The function should take the matrix A, the right-hand side, an integer number being the number of iterations (note, we are not using a more advanced stopping criteria here), and an initial guess x_0, and return the solution x_k. *Hints:* You can create the diagonal matrix D like this `D = np.diagflat(np.diag(A))`. Other useful python functions are `numpy.tril()`, `numpy.triu()` and `numpy.matmul()`. Note that `numpy .tril()` and `numpy.triu()` includes the diagnoal of the original matrix, so for your purposes you need to subtract D. Include the code in your hand in.
2. Use your function for solving the ODE-BVP from Lab 5 part 2, for N=5 and a maximum number of iterations equal to 100. Make sure that the result is close to the analytical solution (e.g. by looking at your plot).
3. Decrease the number of Jacobi iterations until you can see in the plot that the solution is less accurate. How few iterations did you use? Include the plot in your hand in.
4. Set N=10 instead. Do you need more iterations for the solution to be acceptable? How many? Note: It is possible to prove that the number of iterations depend on N.

This should be included in this part of the hand in: Answers to all questions, as well as the proof that Jacobi converges for this problem, the Jacobi-method code, and the plot showing the unsatisfactory solution for few iterations with $N = 5$.