Hand in 5, part 2 of 3 - Direct Methods

1 Getting acquainted with LU-factorization

1. Compute (with pen and paper) the LU factorization of the matrix, with partial pivoting

$$A = \begin{bmatrix} 2 & -1 & 1 \\ 1 & -2 & 1 \\ 2 & 1 & -4 \end{bmatrix}. \tag{1}$$

   Include the derivation in your hand in. Check if $PA = LU$ to ensure that you did it correctly.

2. Use `scipy.linalg.lu_factor` to perform the same calculation and check that it is the same. You don't need to include anything from this task in the hand in.

2 LU-factorization for an ODE-BVP

Consider the BVP that you worked in in the last lab, but with a time-dependent boundary condition

$$y''(x) + y(x) = 0 \tag{2}$$
$$y(x = 0) = 1 \tag{3}$$
$$y(x = \pi/2) = 2\sin(\pi t) \tag{4}$$

You will now solve this for $t = t_1, t_2, t_3, \ldots t_{M-1}, t_M = 1$, for M=100

1. Start by moving all code (from the previous lab) regarding the assembly of A, into its own subroutine. The subroutine should return A and have N as an input value. Also move all code regarding the assembly of the right hand side into another subroutine, which should take N and the boundary values as an input, and return F.

2. Create a for-loop looping over the time-values t_i. You can assemble A outside the for-loop, but you need to assemble the right hand side and solve the system inside the for-loop.

3. Set $N = 1000$ and measure the time it takes for the for-loop to run, e.g. by using `timeit` like below

```
import timeit
starttime=timeit.default_timer()
#your code...
print('solvetime is'+str(endtime-starttime))
```

4. Now utilize the power of LU-factorization. You can instead of using `numpy.linalg.solve` use `scipy.linalg.lu_factor` followed by `scipy.linalg.lu_solve`. The function `scipy.linalg.lu_solve` takes care of the forward and backward substitution step. Think about if you have to put `scipy.linalg.lu_factor` inside or outside the loop.

5. Time your lu-version of the code. How does it compare to the timing of `numpy.linalg.solve`? Explain your result.

6. What is `numpy.linalg.solve` actually doing under the hood? Google.

This should be included in this part of the hand in: The derivation of the LU-factorization of the $3x3$-matrix, your code for the BVP, your LU-timings, and answers to all questions.