

Homework Sheet 3

Simon Etter, 2019/2020

Recommended deadline: 5 September 2019

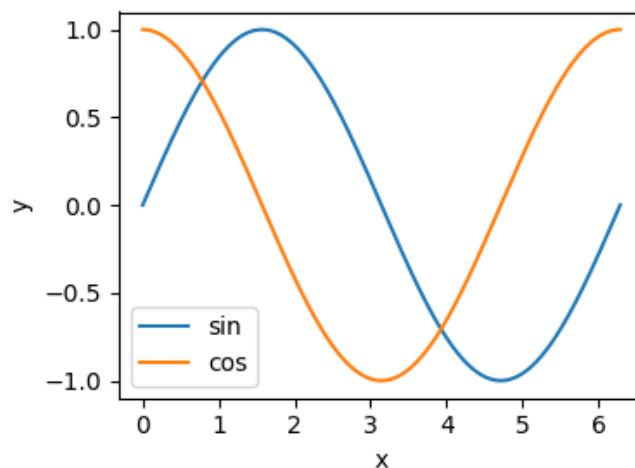
Final deadline: 12 September 2019, 7pm

1 Getting started with PyPlot (unmarked)

The second exercise of this sheet will ask you to create plots, which you can do using the PyPlot package. To install this package, open the Julia REPL, type `] add PyPlot` and press enter. After this, you can load the package by typing `using PyPlot` like any other package. Please contact me if you have any problems installing PyPlot.

The following example illustrates most of the commands that you will need for this assignment:

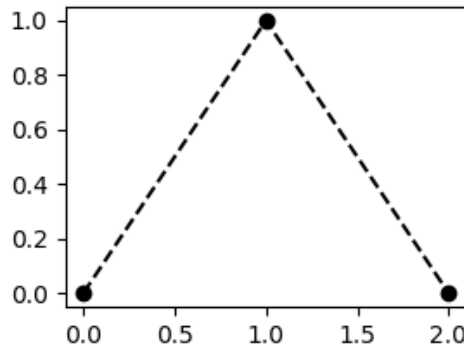
```
fig = figure(figsize=(4,3))
x = LinRange(0,2pi,1000)
plot(x, sin.(x), label="sin")
plot(x, cos.(x), label="cos")
legend(loc="best")
xlabel("x")
ylabel("y")
tight_layout() # resizes the plot so x and y label are not cut off
savefig("test.png")
close(fig)
```



Other commands which may prove useful are:

- `semilogx()`: use logarithmic scale for x axis
- `semilogy()`: use logarithmic scale for y axis.
- `loglog()`: use logarithmic scale for both axes.
- `plot(x,y,format,...)` with `format` being a combination of the following allows you to change the line format.
 - Line styles: `-` (solid), `--` (dashed), `-.` (dash-dotted), `:` (dotted).
 - Markers: see https://matplotlib.org/3.1.1/api/markers_api.html.
 - Colours: `C0` to `C9` (default colour palette), `k` (black)

Example: `plot([0,1,2], [0,1,0], "k--o")` yields the following.



2 Poisson equation with Dirichlet and Neumann boundary conditions

The aim of this exercise is to write code which solves the Poisson equation

$$\begin{cases} -u''(x) = f(x) & \text{for } x \in [0, 1] \\ u(0) = 0, & u'(1) = 0 \end{cases} \quad (1)$$

with Dirichlet boundary conditions at $x = 0$ and Neumann boundary conditions at $x = 1$.

As in class, we discretise this equation by replacing the continuous function $u : [0, 1] \rightarrow \mathbb{R}$ with the vector of point-values

$$u = \left(u\left(\frac{1}{n}\right) \quad u\left(\frac{2}{n}\right) \quad \dots \quad u\left(\frac{n-1}{n}\right) \quad u\left(\frac{n}{n}\right) \right)^T. \quad (2)$$

However, unlike what we have seen before this vector now includes the right end-point $u\left(\frac{n}{n}\right) = u(1)$ since we only know the value $u'(1)$ but not $u(1)$. It is because of this

extra unknown that we chose the denominators in (2) as n rather than $n+1$ in order to maintain the convention that n denotes the number of unknowns. Furthermore, we use the usual approximation

$$u''\left(\frac{k}{n}\right) \approx n^2 \left(u\left(\frac{k-1}{n}\right) - 2u\left(\frac{k}{n}\right) + u\left(\frac{k+1}{n}\right) \right)$$

for the second derivative, and in a first try we use the approximation

$$u'(1) \approx n \left(u\left(\frac{n}{n}\right) - u\left(\frac{n-1}{n}\right) \right) = 0$$

to discretise the Neumann boundary condition. The linear system for the unknown point-values u_n then becomes $A_n u_n = b_n$ where

$$A_n := \begin{pmatrix} 2n^2 & -n^2 & & & \\ -n^2 & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & -n^2 & 2n^2 & -n^2 \\ & & & -n & n \end{pmatrix}, \quad u_n \approx \begin{pmatrix} u\left(\frac{1}{n}\right) \\ u\left(\frac{2}{n}\right) \\ \dots \\ u\left(\frac{n-1}{n}\right) \\ u\left(\frac{n}{n}\right) \end{pmatrix}, \quad b_n := \begin{pmatrix} f\left(\frac{1}{n}\right) \\ f\left(\frac{2}{n}\right) \\ \dots \\ f\left(\frac{n-1}{n}\right) \\ 0 \end{pmatrix}. \quad (3)$$

1. Assume u is the solution to the continuous Poisson equation (1). Show that the consistency error decays as

$$\|A_n u - b_n\|_{2,n} = \mathcal{O}(n^{-1}),$$

where the norm $\|\cdot\|_{2,n}$ on \mathbb{R}^n is defined as

$$\|f\|_{2,n} := \frac{1}{\sqrt{n}} \sqrt{\sum_{k=1}^n f_k^2} = \frac{\|f\|_2}{\sqrt{n}}.$$

2. Derive a three-point approximation

$$u'(1) \approx D_n u := c_0 u\left(\frac{n}{n}\right) + c_1 u\left(\frac{n-1}{n}\right) + c_2 u\left(\frac{n-2}{n}\right)$$

which is accurate to second order, i.e. $D_n u - u'(1) = \mathcal{O}(n^{-2})$.

Hint. Taylor-expand $u\left(\frac{n-k}{n}\right)$ around $x = 1$ and determine the c_k by matching the coefficients in $D_n u - u'(1) = \mathcal{O}(n^{-2})$.

3. Propose modifications \hat{A}_n and \hat{b}_n to the A_n and b_n from (3) such that after modification, the consistency error satisfies

$$\|\hat{A}_n u - \hat{b}_n\|_{2,n} = \mathcal{O}(n^{-2}).$$

4. Let us assume without proof that both A_n and \hat{A}_n are stable, i.e. $\|A_n\|_{2,n}$, $\|\hat{A}_n\|_{2,n}$ remain bounded for $n \rightarrow \infty$. Then, the findings from [Tasks 1](#) and [3](#) imply that the solutions u_n and \hat{u}_n to $A_n u_n = b_n$ and $\hat{A}_n \hat{u}_n = \hat{b}_n$ converge as

$$\|u - u_n\|_{2,n} = \mathcal{O}(n^{-1}) \quad \text{and} \quad \|u - \hat{u}_n\|_{2,n} = \mathcal{O}(n^{-2}),$$

respectively. Verify this numerically by plotting $\|u - u_n\|_{2,n}$ and $\|u - \hat{u}_n\|_{2,n}$ for $n \in \{2^2, \dots, 2^{12}\}$ and some function $u(x)$ for which you can explicitly compute the correct right-hand side $f(x)$.

Your submission for this task should consist of the aforementioned plot in either PDF or PNG format and a Julia file containing all the code needed to reproduce your plot.

Hint. The matrix \tilde{A}_n from [Task 3](#) is no longer tridiagonal. I recommend that you assemble this matrix by first assembling the tridiagonal part using the `Tridiagonal()` function, then convert to a sparse matrix using `sparse()`, and finally add extra and/or modify entries using `A[i,j] = v`.

Example:

```
julia> A = sparse(Tridiagonal(
    fill(1,3),
    fill(2,4),
    fill(3,3)
))
A[4,2] = 4
A[4,3] = 5
Matrix{A}
4×4 Array{Int64,2}:
 2  3  0  0
 1  2  3  0
 0  1  2  3
 0  4  5  2
```