Projektpraktikum Series 3 Documentation

Tom Lambert, Yuuma Odaka

15. December 2017

Contents

1 Introduction

Welcome to our documentation of Projektpraktikum Series 3. Our task was to write a program which creates certain matrices and runs experiments to test the accuracy of matrix operations. Due to physical limitations, computers can often cause problems when executing complex algorithms involving large matrices, especially when the sizes of individual matrix entries vary wildly.

2 Implementation

2.1 Matrix class

Our Matrix class holds the methods we use to create matrices and execute operations.

2.1.1 create_matrix_and_inv

The method `create\_matrix\_and\_inv` generates a matrix as well as its inverse, and returns them.

2.1.2 condition

`condition` calculates the condition of a given matrix by multiplying the infinity norm of the matrix with the infinity norm of the inverse matrix and returns the result.

2.1.3 lu

`lu` carries out a LU (Lower-Upper) decomposition on a given matrix and returns both resulting matrices.

2.1.4 solve

`solve` accepts a matrix A and a vector b and solves the equation Ax = b by first applying LU decomposition and then using Gaussian elimination to return the vector x.

2.1.5 Experiments: Start and Parameters in `start.py`

`start.py` has a `main` function to execute the experiments. In this function you can adjust multiple parameters:

- `experiment`
  Determines the experiment to execute. Valid values are `"3.1B"`, `"3.2B - A"` and `"3.2B - B"`
- `dtypes`
  The data types to use for the successive experiments. Valid values are `"float16"`, `"float32"` and `"float64"`
  Note: This parameter is ignored for the plotted graphics.

- `mtypes`
  `"hilbert"` or `"saite"` for Experiment 3.1A
- `dims` The dimensions to use for the Experiment 3.1A
- `n`
  Determines the n-value as described in Experiment 3.2B
- `i`
  Only $i < n$ will be evaluated in Experiment 3.2B - B.

2.2 Experiments: Implementation in `matrix.py`

2.2.1 `main`

`main` evaluates the parameter from `start.py` and executes one of the following functions.

2.2.2 `main_3.1b`

`main_3.1b` calculates and prints the analytical property
$||I - MM^1||$

as well as the condition of the matrix with the infinity norm

$\text{cond}_\infty(M) = ||M||_\infty ||M^-1||_\infty$

for Hilbert Matrices and tridiagonal matrices with the dimensions 5, 7 and 9 with the data types `float16`, `float32` and `float64`.

2.2.3 `main_32b_saite`

Executes `plot.py`, the Code for plotting the chart for task 3.2A.

2.2.4 `main_32b_hilbert`

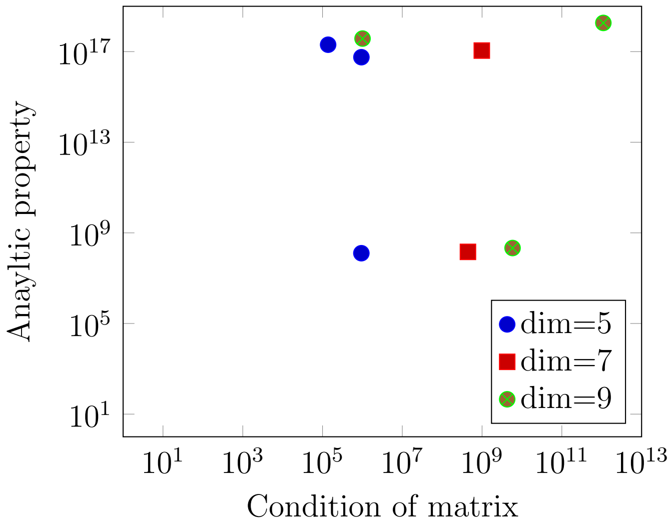
`main_32b_hilbert` executes experiments as described in 3.2B B (Hilbert)

3 Experimental results

3.1 Analytical Property

3.1.1 Hilbert Matrix

Analytic property as a function of matrix condition

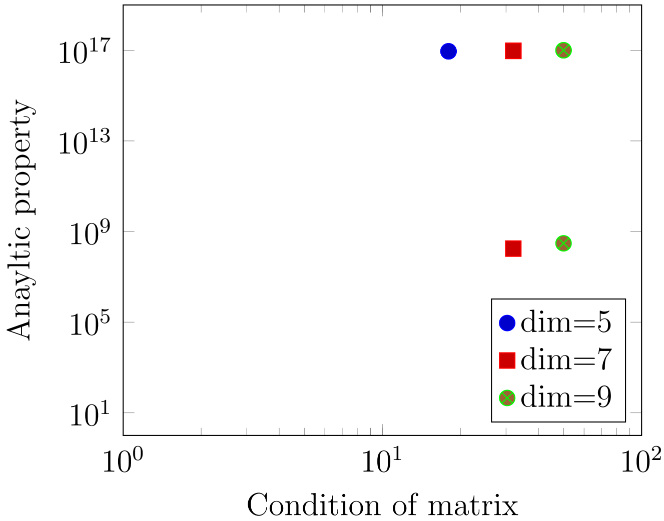


Note: for dim = 7 with `float16`, our result was 'Cannot calculate inverse of M: singular matrix'. This is likely due to a rounding error causing the rank of the matrix to fall below 7.

The three marks at approximately 10^9 are all for `float16`.

3.1.2 Tridiagonal 'String' matrix

Analytic property as a function of matrix condition



Note: for dim = 5 with `float32`, our result was 'Cannot calculate inverse of M: singular matrix'. It is unclear why this is the case only for `float32`; it functions for both `float16` and `float64`

We see a similar pattern here. For both matrices, we see that the condition and size of the matrices have very little effect on the analytic property, while the float site radically alters the result.

3.2 LU-Decomposition

3.2.1 Tridiagonal matrix

Figure 1: Approximation for n = 5

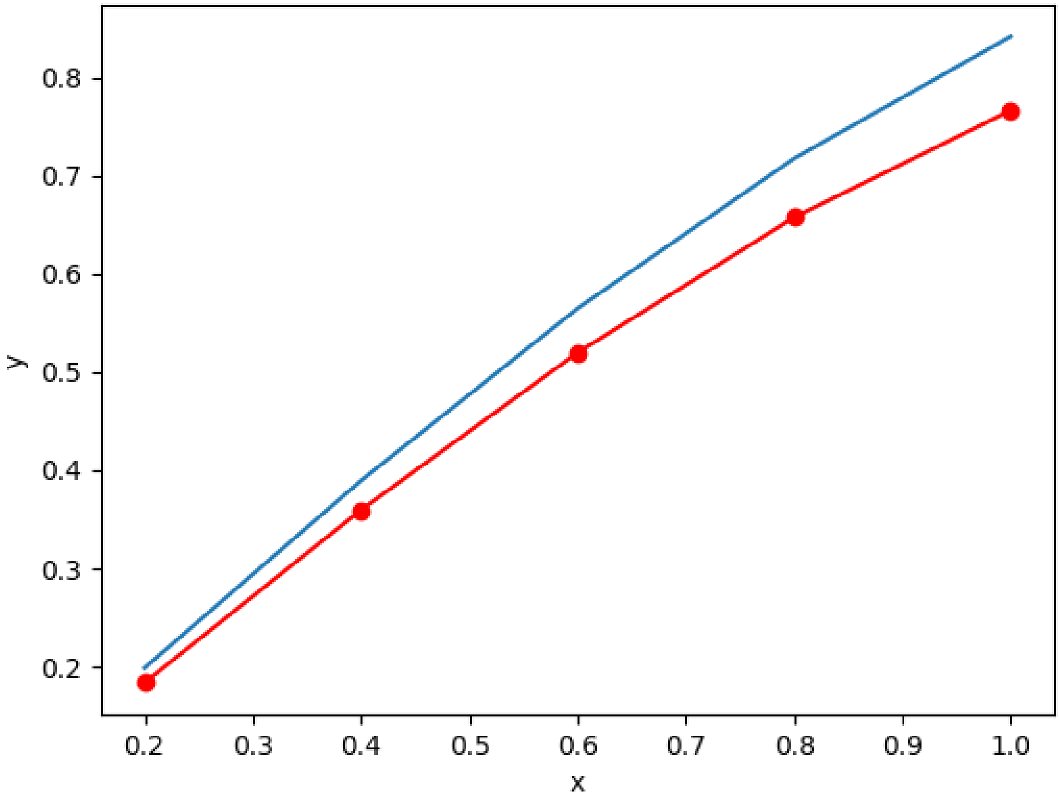


Figure 2: Approximation for n = 8

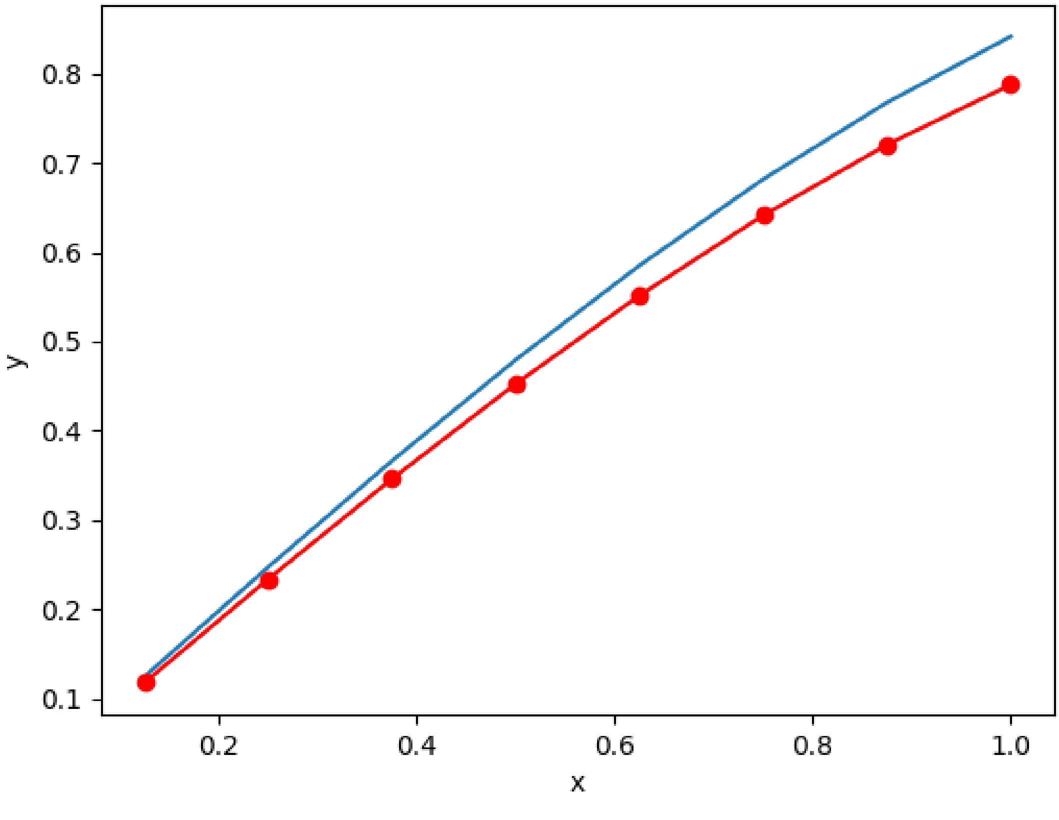
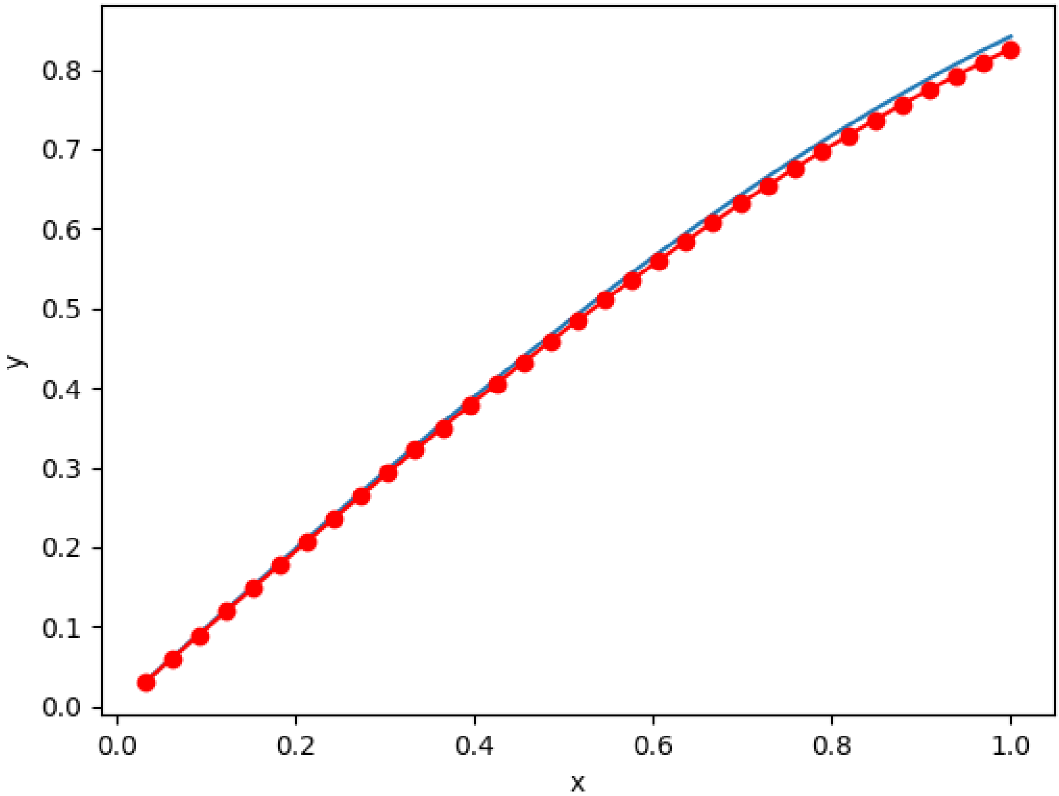


Figure 3: Approximation for n = 33



We can see that increasing n significantly improves the accuracy of the approximation. Since the computing power necessary is in $\mathcal{O}(n^3)$, however, there is a fairly low cap on achievable accuracy.