Technical University of Munich
Department of Informatics
Dr. Tobias Neckel
Paul Cristian Sârbu
Sebastian Wolf
Hayden Liu Weng

WT 2020/2021

Lab Course
Scientific Computing

Worksheet 1

distributed: Thu., 12.11.2020
due: Sun., 22.11.2020, midnight (submission on the Moodle page)
oral examination: Tue., 24.11.2020 (exact time slots announced on the Moodle page)

In this worksheet, we study some numerical aspects of the classical two-player game *rock-paper-scissors*, using `MATLAB`. In particular, we focus on simple strategies that, in the long run, could benefit both the computer and the human player.
The provided files are:

- `game.fig` contains the graphical user interface and **should not be modified**
- `game.m` contains the initialization of the game and the rest of the logic (e.g. handling events) related to the GUI. It **may be modified**
- `mchoice.m` contains the "brain" of the game. It **may be modified**
- `gen_human_move.m` is a function that generates moves for the human player. It can be used to run a user-defined number of rounds automatically. For the purposes of this worksheet, it **should not be modified**

The GUI contains a button `Save data` that permits you to create a `.mat` file containing the *last transition matrix of the game* and *the results after each round*.

a) In `mchoice.m`, fill in the implementation of `predict1` such that

- **i)** it takes two input arguments, j - the previous move of the human player, and *transm* - the current predicted transition matrix for the human player
- **ii)** it prints the input parameter *transm*
- **iii)** it uses two global variables *param_a, param_b* satisfying $0 \leq param\_a \leq param\_b \leq 1$
- **iv)** it computes the variable *hnext*, representing the *predicted next move of the human player*, using the following algorithm:
  - **1)** Generate a sample from the standard *uniform* probability distribution using the built-in function `rand`.
  - **2)** Using the modulo function, compute *hnext* as

$$hnext = \begin{cases} \text{mod}(j, 3) + 1 & \text{with probability } param\_a \\ \text{mod}(j{+}1, 3) + 1 & \text{with probability } param\_b - \ param\_a \\ j & \text{with probability } 1 - param\_b \end{cases} \tag{1}$$

    using the generated uniform random variable to decide which branch to follow.
- **v)** it returns the variable $next = winchoice(hnext)$

b) Analyze the three prediction policy functions (the one implemented in **a**) and the two existing ones, found in `mchoice.m`) and fill in the entries of the following table with *yes/no*:

Method \ Property	Deterministic	With memory
predict1	No	
predict2	Yes	yes
predict3	No	Yes

c) Find (experimentally) the convergence limit of the transition matrix *transm* using the *predict2* policy for the computer player and the default strategy (implemented in `gen_human_move.m`) for the human player.

Hint: Perform simulations with an increasing number of rounds.

d) Using the functions `initmchoice`, `updatesamplem`, and `updatetransm`, implement the matrices `samplem2` and `transm2` corresponding to the sample matrix and transmition matrix of the computer player, respectively. Print the computed *transm2* matrix at the end of `mchoice`. Additionally, add *transm2* to the list of variables saved to file in `game.m`.

Hint: use the last two moves of the computer player saved in the *history* array as additional parameters of the `mchoice` function.

e) Similarly to **c)**, find (experimentally) the convergence limit of the transition matrix *transm2* using the *predict3* policy for the computer player and the default of the *User defined transition matrix* (check the box in the GUI and use the default values).

f) Fill in the matrices below with the results from **c)** and **e)**

$$transm \rightarrow \begin{bmatrix} 0.0030 & 0.9940 & 0.0030 \\ 0.0030 & 0.0030 & 0.9940 \\ 0.9940 & 0.0030 & 0.0030 \end{bmatrix} \qquad transm2 \rightarrow \begin{bmatrix} 0.0269 & 0.0507 & 0.9224 \\ 0.9326 & 0.0323 & 0.0352 \\ 0.0271 & 0.9398 & 0.0331 \end{bmatrix}$$

Questions:

Q1 What is the type of the transition matrix of the game? What properties does it possess?

Q2 Related to exercise **c)**: For which of the three prediction strategies of the computer player can it be (intuitively) guaranteed that, in the long run (i.e. an infinite number of rounds), a winning strategy for the human player can be found?

Q3 How can the modeling approach used in this worksheet be extended to study (slightly) more complex games or more complex player behaviors?

Q4 Where do similar scenarios to the one in this worksheet show up in real-world applications?