Team 77

- Tobias Pucher (S5751659)
- Matthias Heiden (S5751616)

Model

Initial performance

The results after implementing the retrieval were as follows:

```
? (zbrodoff-compare 5)
...

CORRELATION:  0.971
MEAN DEVIATION:  0.673

              2 (64)      3 (64)      4 (64)
Block  1  1.224 (64)  1.366 (64)  1.393 (64)
Block  2  1.041 (64)  1.009 (64)  1.022 (64)
Block  3  0.999 (64)  1.006 (64)  1.026 (64)
NIL
? (actr-load "ACT-R:tutorial;lisp;zbrodoff.lisp")
? (zbrodoff-compare 1)
...

CORRELATION:  0.933
MEAN DEVIATION:  0.666

              2 (64)      3 (64)      4 (64)
Block  1  1.347 (64)  1.318 (64)  1.429 (64)
Block  2  1.001 (64)  1.027 (64)  1.004 (64)
Block  3  0.991 (64)  0.985 (64)  0.980 (64)
NIL
```

The human experiment which should be approximated gave the following results:

```
Control Group (all problems equally frequently)
             Two    Three    Four
Block 1    1.840    2.460   2.820
Block 2    1.210    1.450   1.420
Block 3    1.140    1.210   1.170
```

Parameter tuning

- correlation is quite good already
- model is too fast overall -> high deviation

Parameters to change: latency factor, instantaneous activation noise, retrieval threshold

```
(sgp :v t :esc t :lf 0.4 :bll 0.5 :ans 0.5 :rt 0 :ncnar nil
```

Retrieval threshold

by adjusting the retreival threshold: `:rt 0.5`

```
CORRELATION:  0.957
MEAN DEVIATION:  0.434

              2 (64)      3 (64)      4 (64)
Block  1  1.360 (64)  1.558 (64)  2.232 (64)
Block  2  1.017 (64)  1.102 (64)  1.177 (64)
Block  3  1.026 (64)  0.996 (64)  0.991 (64)
```

```
 :rt 0.8

CORRELATION:  0.977
MEAN DEVIATION:  0.314

              2 (64)      3 (64)      4 (64)
Block  1  1.470 (64)  1.850 (64)  2.258 (64)
Block  2  1.150 (64)  1.422 (64)  1.375 (64)
Block  3  0.993 (64)  1.062 (64)  1.069 (64)
NIL
```

```
:rt 0.93

CORRELATION:  0.986
MEAN DEVIATION:  0.130

              2 (64)      3 (64)      4 (64)
Block  1  1.816 (64)  2.362 (64)  2.532 (64)
Block  2  1.118 (64)  1.320 (64)  1.544 (64)
Block  3  1.068 (64)  1.096 (64)  1.196 (64)
NIL
```

```
:rt 0.95

CORRELATION:  0.993
MEAN DEVIATION:  0.120
```

```
               2 (64)      3 (64)      4 (64)
Block  1  1.790 (64)  2.213 (64)  2.631 (64)
Block  2  1.207 (64)  1.296 (64)  1.479 (64)
Block  3  1.122 (64)  1.167 (64)  1.162 (64)
NIL
```

FINAL Tuning Result

The previously found rt:0.95 is the starting point for fine tuning, which was performed incrementally on the other parameters.

- a little higher response time in the first block but similar results in the last 2 blocks would be better.

After some more finetuning of the other parameters: (sgp :v nil :esc t :lf 0.35 :bll 0.5 :ans 0.5 :rt 0.9 :ncnar nil)

- A slightly lower retrieval threshold and a slightly reduced latency factor yielded the best results:

Final Parameter

```
(sgp :v t :esc t :lf 0.375 :bll 0.5 :ans 0.45 :rt 1.1 :ncnar nil)
```

Final Performace

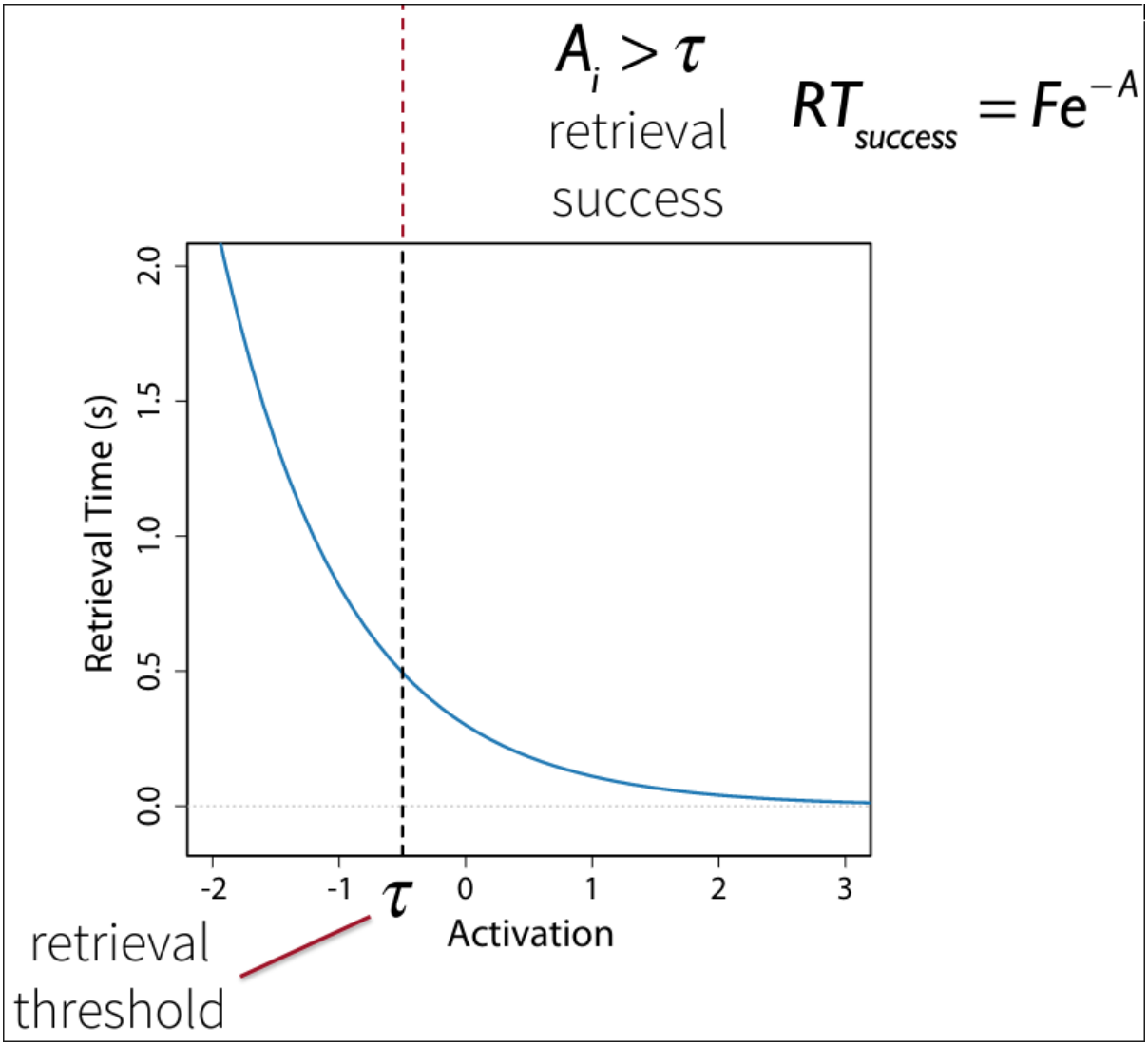
Provided consistently good results over many "compare 5" runs.

```
? (actr-load "ACT-R:tutorial;lisp;zbrodoff.lisp")
T
? (zbrodoff-compare 5)
CORRELATION:  0.994
MEAN DEVIATION:  0.065

               2 (64)      3 (64)      4 (64)
Block  1  1.899 (64)  2.459 (64)  2.692 (64)
Block  2  1.286 (64)  1.379 (64)  1.470 (64)
Block  3  1.122 (64)  1.141 (64)  1.191 (64)
NIL
? (zbrodoff-compare 5)
CORRELATION:  0.984
MEAN DEVIATION:  0.128

               2 (64)      3 (64)      4 (64)
Block  1  1.905 (64)  2.244 (64)  2.606 (64)
Block  2  1.316 (64)  1.328 (64)  1.563 (64)
Block  3  1.087 (64)  1.184 (64)  1.209 (64)
NIL
```

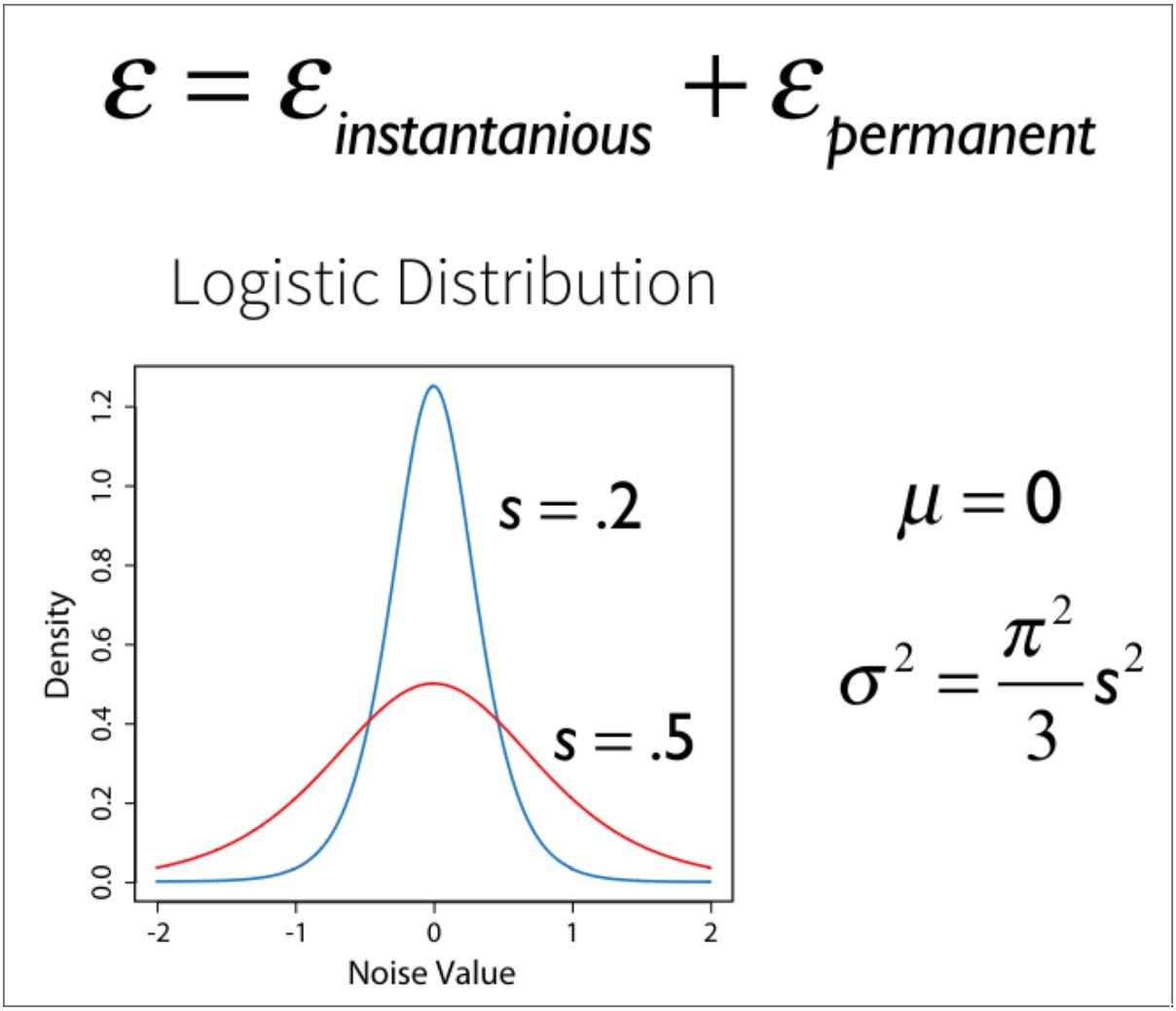
Theory

In a separate pdf file, show the output of your model, and answer the following questions:

What is the effect of manipulating the parameters below on the pattern of the results? Explain the reason behind it. It is useful to think in advance what the effect might be, instead of just trying it out. Of course, it is still a good idea to also try it out.

- Retrieval Threshold, τ (:rt, typical values -1.0 to 2.0)
    - **Effect**: Retrieval fails less/more often.
    - **Reason**: Determines if a chunk can be retrieved. The higher the threshold, the more likely it is that the retrieval fails (recall probability is decreases). When a retrieval request is made, then the chunk's activation is compared to the threshold. If the activation is higher than the threshold, then the retrieval is successful. The chunk with the highest activation is retrieved.
    - 

- Latency Factor, F (:lf, typical values range from .1 to 2)
    - **Effect**: Retrieval takes longer/shorter.
    - **Reason**: The activation of a chunk also determines how long it takes to retrieve it. It's a linear multiplier. If the latency factor is increased, then the retrieval takes longer and vice versa.
        - Retrieval time: `RT = Fe^{-A}`

- Activation noise, s (:ans range 0.1-0.8)
  - **Effect**: The higher the :ans value, the more noise will be added to the activation of a chunk.
  - **Reason**: The activation noise is modelled after a Logistic distribution. Increasing the s (scale) parameter, means that the distribution is wider. This means that the activation of a chunk is more likely to be higher or lower than the actual activation. This means that the retrieval is less accurate.
  - Also known as: instantaneous noise (s), recomputed at each retrieval attempt.
  - 

The model you constructed uses two strategies sequentially. First, it fires production rules that try to retrieve the answer from memory. If that retrieval fails, it fires production rules that calculate the answer through counting. This model is inspired by an older model from Gordon Logan. This original model executed both strategies (counting and retrieval) in parallel. That is, it attempted to retrieve the answer and, at the same time, it started to count. The final answer was provided by the strategy that finished first. Would it be possible to implement such a parallel strategy in ACT-R? If yes, what would such a model look like? If no, why is it not possible?

No, it's not possible to implement such a model.

The reasons for that are:

- ACT-R assumes that the central production system is serial

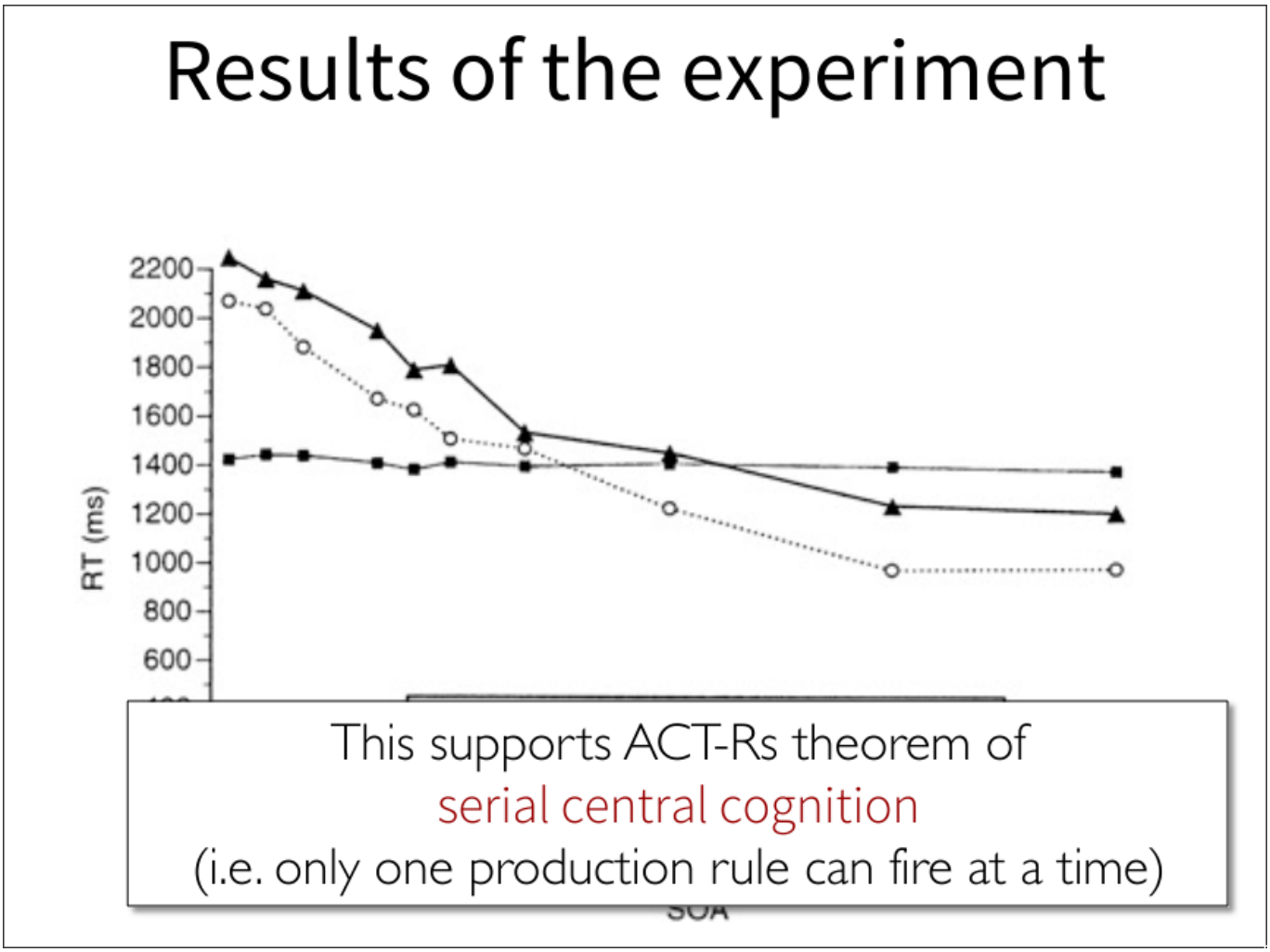
    - Only one production rule can fire at a time
- There's serial and parallel processing in ACT-R:
    - Serial:
        - Procedural module
            - One production at a time
        - Other modules
            - One request at a time
            - A single chunk in a buffer
    - Parallel:
        - The modules operate in parallel
        - Internal mechanisms of a module can be highly parallel
            - Conflict resolution
            - Declarative retrievals
            - Finding a visual-location
- **Perceptual and motor modules** work in parallel, but a single module can only work on one thing at a time

If you think about how the brain processes information, do you think it could do counting and retrieval in parallel? What would be requirement for this?

The brain's information processing is highly parallel, but has a bottleneck (see lecture 3). While it may be possible that the counting and retrieval is done in parallel, it's much more likely that it's not.

Both the counting and retrieval are done in the **central cognitive system**, which is responsible for coordinating perception/action and "thinking". Experiments suggest that the central cognitive system is **serial**.

Based on the Psychological Refractory Period experiments from Lecture 3, it's likely that the counting and retrieval is done in **serial**.



There's no way to test this theory, because retrieval is very fast. However, one could change the experiment to make the retrieval slower or replace it with an equivalent but more complex task. In fact, this was done the in the previously mentioned experiment.