Aging:

I chose a good refresh period to be every 11 lines. This minimized the number of page faults in my aging algorithm, which makes sense, because it does not refresh so frequently that there is an extremely short access history, but also not so slowly that there is little difference between the aging values for each frame.

WSClock:

Varying Tau with a refresh rate of 11:

I selected 5 as a good value for tau because it resulted in the least number of page faults. Too large a value of tau will result in a large working set containing pages that may not be used, an too small of a tau value will result in pages that are actually part of the working set being determined as outside of the working set.

Varying the refresh rate with a Tau of 5:

The lowest possible refresh rate is best for WSClock. If The refresh rate was 1, the algorithm would essentially be least recently used, because each would store the actual virtual time at which it was last accessed rather than an estimate of that time. I will pick a refresh rate of 12, because the very low ones seem very unrealistic.

Varied Frame count:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Frame Count | 8 | 16 | 32 | 64 |
| Page Faults by Algorithm | Opt | 118480 | 80307 | 55802 | 38050 |
| Clock | 168290 | 115521 | 83859 | 57952 |
| Aging: r = 11 | 168775 | 116771 | 91667 | 87945 |
| WSClock: t = 8, r = 13 | 175219 | 125158 | 93490 | 67348 |

The results for each algorithm show that opt is clearly the best algorithm for page replacement of the four, as can be expected. However, this is not an implementable page replacement algorithm, so an operating system designer would have to choose between the other three. Based on the results shown, the Clock algorithm appears to have the best results of the three implementable algorithms, so I would choose to use that one for an actual operating system.