Homework 1

2/3
DATE

1.7.4.

$$a \cdot b = \frac{(a+b)^2 - a^2 - b^2}{2}$$

bab. · to find Square root. S.

· Guess Pos. X_0

· Improve guess. $X_1 = (X_0 + S/X_0)/2$

· Iterate until convergence

$X_{n+1} = (X_n + S/X_n)/2$

& $X_n \approx X_{n+1}$

This is less optimal than the babylonian method. There are more computations needed (6) and the babylonian requires 4.

1.7.7.

.1	.9	1.7	2.5	3.3	4.1	4.8
.2	1.0	1.8	2.6	3.4	4.2	4.9
.3	1.1	1.9	2.7	3.5	4.3	5.0
.4	1.2	2.0	2.8	3.6	4.4	
.5	1.3	2.1	2.9	3.7	4.5	
.6	1.4	2.2	3.0	3.8	4.6	
.7	1.5	2.3	3.1	3.9	4.7	
.8	1.6	2.4	3.2	4.0		

1.7.7

lookup table is the same as the cumulative distribution function table. Z and Phi (Φ) tables.

Not writing it out but have it in front of me.

(It is probably the most famous lookup table in existence)

1. We know more accurate units (to the hundredth) than .1 in the z table.

Thus one strategy is taking a row of values (.01, .02, ..., .09), averaging them out and using this value for each tenth value.

↳ low accuracy at lower values that have higher range.

2. Taking average of prior and following responses.

↳ similar accuracy restriction as above.

1.7.18. a_1 and a_2 pos. #s

Prove Arithmetic mean ≥ geom. mean.

Arith. mean = $\frac{a_1 + a_2}{2}$

geom. mean = $\sqrt{a_1 a_2}$

to show: arith - geom ≥ 0

$$a - g = \frac{a_1 + a_2}{2} - \sqrt{a_1 a_2} = \frac{a_1 + a_2 - 2\sqrt{a_1 a_2}}{2}$$

$$= \frac{\left(\sqrt{a_1} - \sqrt{a_2}\right)^2}{2}$$

this will always be positive

So arith - geom > 0 showing that arith. mean is always greater or equal to geom mean.

When equal? when $a_1 = a_2$

arith - geom = 0 thus equal.

1.7.34. yes, we can multiply any # of square matrices using Strassens Algorithm.

This is possible through the splitting, ie

$ABC = (AB) \times C$

$ABCD = ((AB) \times C) \times D$

1 Brute force $n \times n = O(n^3)$

M Brute force $= (m-1)n^3 = O((m-1)n^3)$

Non brute force $= O((m-1)n^{\log_2 7})$

$\nearrow$ much faster

DATE

1.7.26. Prove # mults required by ~~fast~~ mult to compute X^n is ~~at most~~ $2\log_2(n)$

At the lowest level, the number of steps needed to multiply in fast multiplication is dependent on the # of 1's in the binary.

To reach X^n we continually square X until we approach X^n.
To compute this number we need $\log_2(n)$ steps.

Then multiplying these values by eachother we get $2\log_2(n)$ as we had expected.