1.

(a) the recurrence can be expressed in matrices as follows. for the $(k+1)$-th iteration:

$$\begin{bmatrix} x^{(k+1)} \\ y^{(k+1)} \end{bmatrix} = \begin{bmatrix} \frac{1}{3} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} \left(\begin{bmatrix} 4 \\ 3 \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x^{(k)} \\ y^{(k)} \end{bmatrix} \right)$$

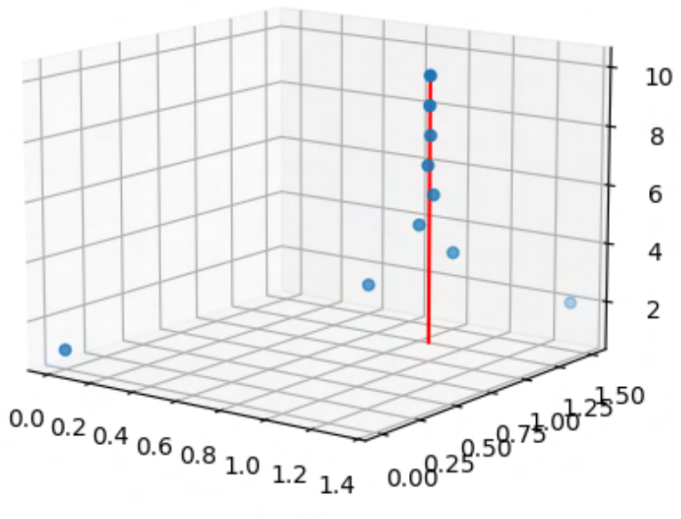
simplification

$$= \begin{bmatrix} \frac{1}{3} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} \cdot \begin{bmatrix} 4 - y^{(k)} \\ 3 - x^{(k)} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{4}{3} - \frac{y^{(k)}}{3} \\ \frac{3}{2} - \frac{x^{(k)}}{2} \end{bmatrix}$$

(b) the plot (below) shows that as iterations go on, our estimates approach the true solution $(x=1, y=1)$. The errors grew smaller and smaller - Jacobi was able to converge.

```
[0.   , 1.333, 0.833, 1.056, 0.972, 1.009, 0.995, 1.002, 0.999, 1.   ]
[0.   , 1.5  , 0.833, 1.083, 0.972, 1.014, 0.995, 1.002, 0.999, 1.   ]
```



(c) for the Gauss-Seidel method:

$$\begin{bmatrix} 3 & 0 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x^{(k+1)} \\ y^{(k+1)} \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x^{(k)} \\ y^{(k)} \end{bmatrix}$$

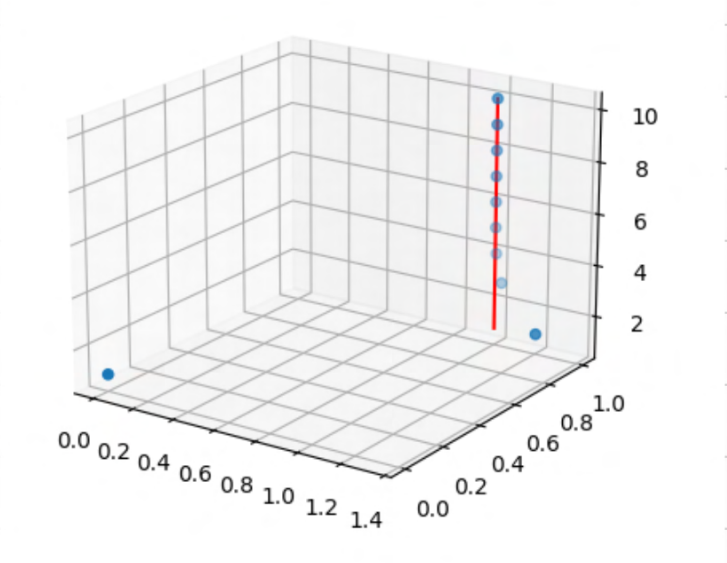
simplification:

$$\begin{bmatrix} x^{(k+1)} \\ y^{(k+1)} \end{bmatrix} = \begin{bmatrix} \frac{1}{3} & 0 \\ -\frac{1}{6} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} 4 - y^{(k)} \\ 3 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{4}{3} - \frac{y^{(k)}}{3} \\ \frac{5}{6} + \frac{1}{6} y^{(k)} \end{bmatrix}$$

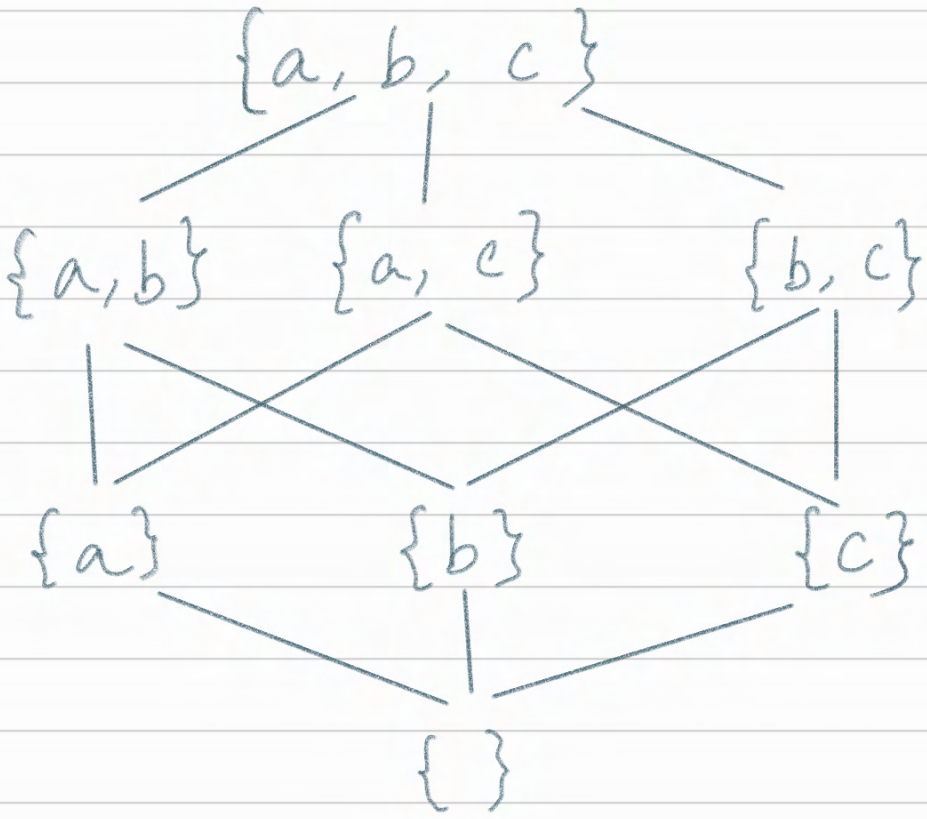
similarly as before for Jacobi, here Gauss-Seidel converged as iterations went on, albeit in a different pattern due to the difference in the underlying method.

```
[0.   , 1.333, 1.056, 1.009, 1.002, 1.   , 1.   , 1.   , 1.   , 1.   ]
[0.   , 0.833, 0.972, 0.995, 0.999, 1.   , 1.   , 1.   , 1.   , 1.   ]
```



2.

(a)



(b) monotonic but not extensive:
$X \to X - \{a\}$

(c) extensive but not monotonic:

$$X \to \begin{cases} \{a,b,c\} & \text{if } X = \{\} \\ X & \text{otherwise} \end{cases}$$

(d) extensive and monotonic:

$X \to X$

(e) function with no fixpoint:

$X \to \{a\} - X$

it's not monotonic because

$$f(\{\}) = \{a\}$$
$$f(\{a\}) = \{\}$$

($\{\} \leq \{a\}$, yet the respective output $\{a\} \not\leq \{\}$)

(f) function with multiple fixpoints:

$X \to X$

3. To show that $(L, \leq)$ is a domain we only need to show that it satisfies the following properties:

(1) finite

Since D was a domain, S must be finite, and thus $L \subseteq S$ must also be finite.

(2) partially ordered

Since $L \subseteq S$, any relation between elements in L can also be found in D (the relation "$\leq$" is the same). Thus all properties for partial order are trivially satisfied.

E.g. reflexivity must be true, because each element that needs to be related to itself is shown to be related to itself in D. The same logic follows for transitivity and anti-symmetry.

(3) has a least element

Given the fixpoint theorem, we know that a "least fixpoint" exist, and by definition, it is the least element in the set of all fixpoints, L.

4.)

Here we can prove by case analysis:

T	$f(T)$
$\{\}$	$\{\}$
$\{a\}$	$\{a,b\}$
$\{b\}$	$\{b\}$
$\{a,b\}$	$\{a,b\}$

we can see that

$f(\{\}) \subseteq f(\{a\}), f(\{b\}), f(\{a,b\})$

$f(\{a\}) f(\{b\}) \subseteq f(\{a,b\})$

More generally, for any $T_1, T_2 \in \mathcal{S}$, where $T_1 \subseteq T_2$:

$$\begin{cases} T_1 \cup \{b\} \subseteq T_2 \cup \{b\} & \text{(if } a \in T_1\text{)} \\ T_1 \subseteq T_2 & \text{(otherwise)} \end{cases}$$

$g(T) = T - \{a\}$ is also monotonic

for any $T_1, T_2 \in \mathcal{S}$, where $T_1 \subseteq T_2$:

$$T_1 - \{a\} \subseteq T_2 - \{a\}$$

if $a \in T_1$ then $a \in T_2$ $\because$ $T_1 \subseteq T_2$. Thus both sets lose a, and the subset relation persists.

if $a \notin T_1$, then whether T_2 loses "a" doesn't matter ($T_1 \subseteq T_2 - \{a\}$ since $a \notin T_1$)

5.) for any $x_1, x_2 \in D$ where $x_1 \leq x_2$

since $g(x)$ is monotonic,

$$g(x_1) \leq g(x_2)$$

since $f(x)$ is monotonic,

$$f(g(x_1)) \leq f(g(x_2))$$

(pretty trivial...)

(note that "$\leq$" is used to denote the relation in the domain D)

6. First, we show the "forward" direction:
given $\begin{cases} a = f(a) \\ a = g(a) \end{cases}$

(a) $f(g(a)) = f(a) = a$

(trivial)

Then, we show the "reverse":
given $x = a$ is a solution to $a = f(g(a))$

let $g(a) = b$,

since $g(\cdot)$ is extensive, $a \leq b$.

however, since $f(\cdot)$ is extensive, $b \leq f(b) = a$

$\therefore a \leq b$ and $b \leq a$
given the anti-symmetry property of a domain,
$a = b$

$\therefore g(a) = b = a$

$\therefore f(a) = f(b) = f(g(a)) = a$

QED.

(b) since $f(g(x))$ is monotonic with respect to $\sqsubseteq$ (per problem 5.), there must exist a least fixpoint "a" such that $a = f(g(a))$. Given part (a) above, we know that $x=a$ is also a solution to $\begin{cases} x = f(x) \\ x = g(x) \end{cases}$. We also know that $x=a$ is a least solution to that system, because it is the least solution to "$f(g(x)) = x$" and "$f(g(x)) = x$" has the exact same solution set as $\begin{cases} x = f(x) \\ x = g(x) \end{cases}$.

To compute "a", we can simply compute the solution to "$x = f(g(x))$" as we would for any least fixpoint problem.

(e.g. start with $\bot$, then $f(g(\bot))$, then $f(g(f(g(\bot))))$, etc. until convergence)

(C) no. For example:

Let $D = (P(\{a,b,c\}), \subseteq)$

$f(x) = x \cup \{a\}$
$g(x) = x - \{a\}$
} both obviously monotonic but $g(x)$ is not extensive

thus $f(g(x)) = x \cup \{a\}$

note $f(g(x)) = x$ has a solution $x = \{a\}$

but $x = \{a\}$ is NOT a solution to the system
$$\begin{cases} x = x \cup \{a\} \\ x = x - \{a\} \end{cases}$$

(in fact, that system has NO solution)

NOTE: that if the question mandates that BOTH $f(x)$ and $g(x)$ are not extensive, the property still fails to hold. For example:

$$\begin{cases} f(x) = (x \cup \{a\}) - \{b\} \\ g(x) = x - \{a\} \end{cases}$$ → still monotonic!

$f(g(x)) = (x \cup \{a\}) - \{b\}$
has solution $x = \{a\}$ but $x = \{a\}$ is not a solution of the system.

(d) if either $f(x)$ or $g(x)$ is not monotonic, then a least fixpoint simply isn't guaranteed.

E.g.
$$\begin{cases} f(x) = \{a\} - x \\ g(x) = \{a\} - x \end{cases}$$
→ not monotonic, as shown before

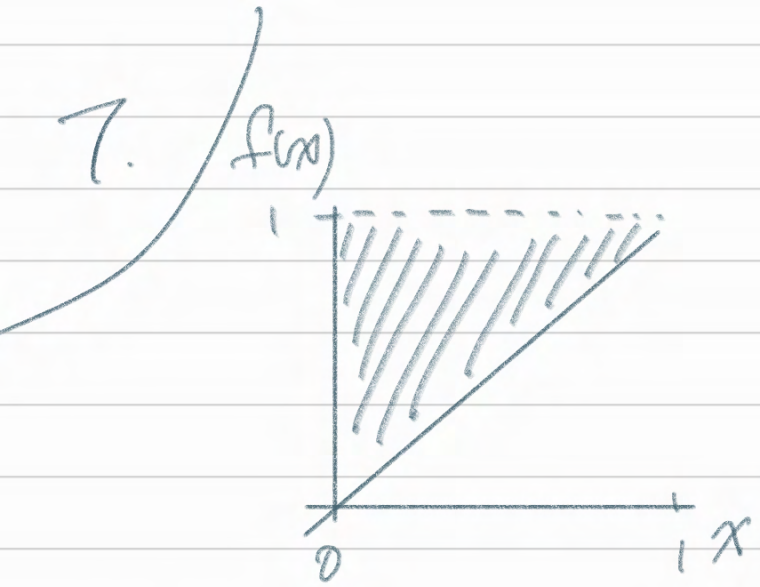
$$f(g(x)) = \{a\} - (\{a\} - x)$$

note that $\{\}$ is a solution of $f(g(x))$

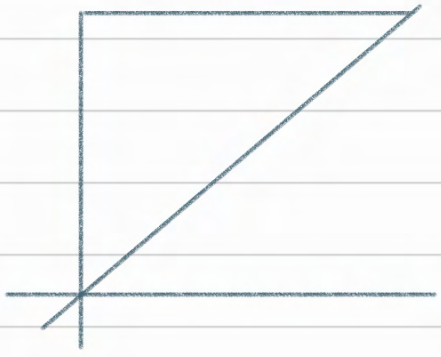
but not a solution of the system

$$\begin{cases} x = \{a\} - x \\ x = \{a\} - x \end{cases}$$

(the system just has no solution to begin with)

7. 

(a)
example: $f(x)=1$

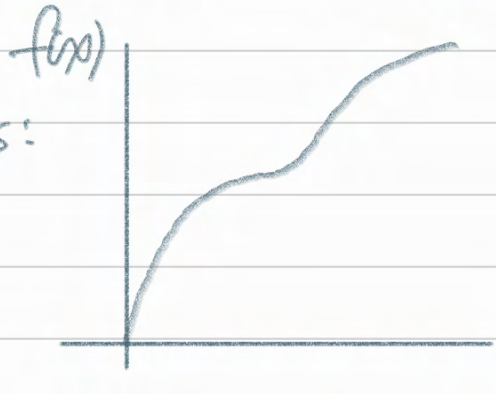


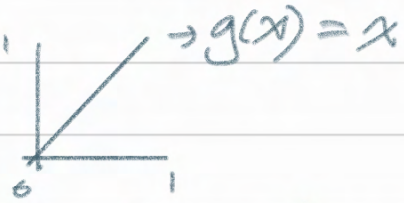
(a) if f is extensive, its graph must rest entirely within the shaded region, where $f(x) \geq x$. Example: $f(x)=1$ (for $x \in [0,1]$)

(b) if f is monotonic, then its graph must "never go down". It can stay flat or go up. The derivative must be always non-negative.

(note that here we consider "monotonic" in the context of domain theory, not in the context of calculus, which may include concept like "monotonically decreasing")

examples:



(c) $g(x) = x$ → 

(used "g(x)" to avoid confusion with "f(x)")

(d) no fixpoint:

$$f(x) = \begin{cases} 0 & \text{if } x = 1 \\ 1 & \text{if } 0 \leq x < 1 \end{cases}$$

some fixpoints:

$$f(x) = 1 \quad \text{if } 0 \leq x \leq 1$$

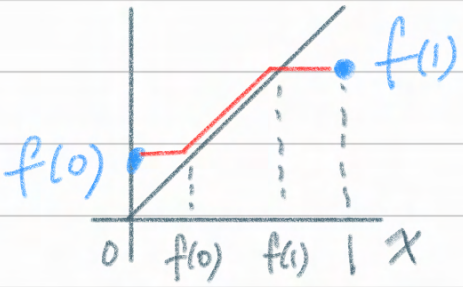
(e) the hairy ball theorem is more about topology which is closely related to set theory. which covers the fix-point theorems. It's supposedly about how you cannot comb a hairy ball so that all hairs lie flat on the ball. The hairy ball theorem is used to prove some famous fixpoint theorems such as "Brouwers

fixed-point theorem".

(f) [extra credit]

yes. The intuition can be found in the example for a no-fixpoint function I gave in part (d).

We can prove this by contradiction. Suppose that f is monotonic but has no fixpoint, then $f(0) > 0$, and $f(1) < 1$. (because $f[x] \in [0,1]$, we cannot go below 0 or above 1).



Now, since f is monotonic, for any $k \in (0,1)$ any $f(k)$ must be at least $f(0)$, and $f(f(0))$ must be greater than $f(0)$ (can't be equal because no fixpoint; can't be less because monotonicity). By the same logic,

the function can never go below the line "x". However, this means $f(f(1)) > f(1)$.

Since $f(1) \leq 1$, $f(f(1)) \leq f(1)$ by monotonicity so $f(f(1))$ is both greater than and no greater than $f(1)$, resulting in a contradiction.

NOTE that I informally claimed that "$f(x)$" must stay above the line "x". A formal proof might require proving that f has the Darboux property.

8.) A grammar is a set of rules that are used to generate a set of strings, and such string set is a language, specific to that grammar. Different grammars may

(a) generate the same language, but each grammar can only generate one language.

(b) an ambiguous grammar is a grammar in which there are more than one leftmost derivations for a given word.

For example, "$A \to A-A \mid x$" is ambiguous.

There are two leftmost derivations of "$x-x-x$":

① $A \to \underline{A}-A$
$\to x-\underline{A}$
$\to x-\underline{A}-A$
$\to x-x-\underline{A}$
$\to x-x-x$

② $A \to \underline{A}-A$
$\to \underline{A}-A-A$
$\to x-\underline{A}-A$
$\to x-x-\underline{A}$
$\to x-x-x$

This matters because the parse tree would be different, i.e. the order of operation is different.

① is essentially $x-(x-x)$ while
② is essentially $(x-x)-x$.

(c) depends on what's meant by "can be parsed". The ambiguous grammar in part (b) can be parsed in LL(2) but there are multiple possible parse trees. I.e., if we consider ambiguity a failure, then that's an example of Recursive Descent Parser failing over a context-free language.

Note that with backtracking and keep track of recursion depth, RDP should be able to terminate on any finite string, since it will enumerate all possible derivations one-depth at a time (the depth is finite because we will consume at least one token from the string at each depth).

c.d) As mentioned above, RDP with backtrack is a candidate, although it can have very bad performance due to exponential explosion.

"GLL" (Generalized LL) parsers can handle all context-free grammars (see paper by Scott in 2010 "GLL Parsing"). There are also other implementations that parse all context-free grammars.

9. (i)

$FIRST_k(\varepsilon) = \varepsilon$
$FIRST_k(t \in T) = t$
} trivially make sense

$FIRST_k(u_1 \dots u_n) = FIRST_k(u_1) +_k \dots +_k FIRST_k(u_n)$

makes sense because we are trying to pad out each $FIRST_k(u_1)$ expression with $FIRST_k(u_2) \dots$ so on so they have length k.

$\forall A \in N,\ FIRST_k(A) = \bigcup_{A \to \alpha} FIRST_k(\alpha)$

makes sense because all possible rewrite rules $A \to \alpha$ have their own k-prefixes parallel to each other, so a union is a natural choice

---

$FOLLOW_k(S) = \{\$^k\}$ makes sense trivially

$FOLLOW_k(B) = \bigcup_{A \to \alpha B \gamma} FIRST_k(\gamma) +_k FOLLOW_k(A)$

$\forall B \in (N' - \{S, S'\})$

makes sense because any k-prefixes of γ must follow B, and then we pad them out to max-k-length with the "$+_k$" operation on what follows A (those would follow γ).

Note that we don't have to worry about NULLABLE non-terminals because the definitions of $FIRST_k$ $FOLLOW_k$, and $\oplus_k$ already account for them. The emphasis lies on the "$\oplus_k$" operator. We essentially used it to combine rule 2 and rule 3 on slide 4 in the lecture slides:

previous rule 2: $$\frac{(A \to \alpha B Y_1 \dots Y_n) \in P}{FOLLOW(B) = FOLLOW(B) \cup (FIRST(Y_1 \dots Y_n) - \{\varepsilon\})}$$

previous rule 3: $$\frac{(A \to \alpha B Y_1 \dots Y_n) \in P \quad Y_1, \dots Y_n \in NULLABLE}{FOLLOW(B) = FOLLOW(B) \cup FOLLOW(A)}$$

(iii)

$$\begin{cases} FIRST_2(S) = FIRST_2(yLab) \cup FIRST_2(yLbc) \cup FIRST_2(M) \\ FIRST_2(L) = FIRST_2(a) \cup FIRST_2(\varepsilon) \\ FIRST_2(M) = FIRST_2(MM) \cup FIRST_2(x) \end{cases}$$

$$\Downarrow$$

$FIRST_2(L) = \{a, \varepsilon\}$

$FIRST_2(M) = (FIRST_2(M) \oplus_2 FIRST_2(M)) \cup \{x\}$

$FIRST_2(M)^{(1)} = (\{\} \oplus_2 \{\}) \cup \{x\} = \{x\}$

$FIRST_2(M)^{(2)} = (\{x\} \oplus_2 \{x\}) \cup \{x\} = \{xx, x\}$

$FIRST_2(M)^{(3)} = (\{xx, x\} \oplus_2 \{xx, x\}) \cup \{x\} = \{xx, x\}$

$FIRST_2(S) = \{ya\} \cup \{ya, yb\} \cup \{xx, x\} = \{ya, yb, xx, x\}$

for $FOLLOW_2$ (using the augmented grammar)

$FOLLOW_2(S) = \{\$\$\}$

$FOLLOW_2(L) = [FIRST_2(ab) +_2 FOLLOW_2(S)] \cup$
$[FIRST_2(bc) +_2 FOLLOW_2(S)]$

$= \{ab, bc\}$

$FOLLOW_2(M) = [FIRST_2(M) +_2 FOLLOW_2(M)] \cup$
$[FIRST_2(\varepsilon) +_2 FOLLOW_2(S)]$

$= [\{xx, x\} +_2 FOLLOW_2(M)] \cup \{\$\$\}$

$FOLLOW_2(M)^{(1)} = [\{xx, x\} +_2 \{\$\$\}] \cup \{\$\$\}$

$= \{xx, x\$, \$\$\}$

$FOLLOW_2(M)^{(2)} = [\{xx, x\} +_2 \{xx, x\$, \$\$\}] \cup \{\$\$\}$

$= \{xx, x\$, \$\$\}$