Matrix Factorization

Geometric Algorithms

Lecture 14

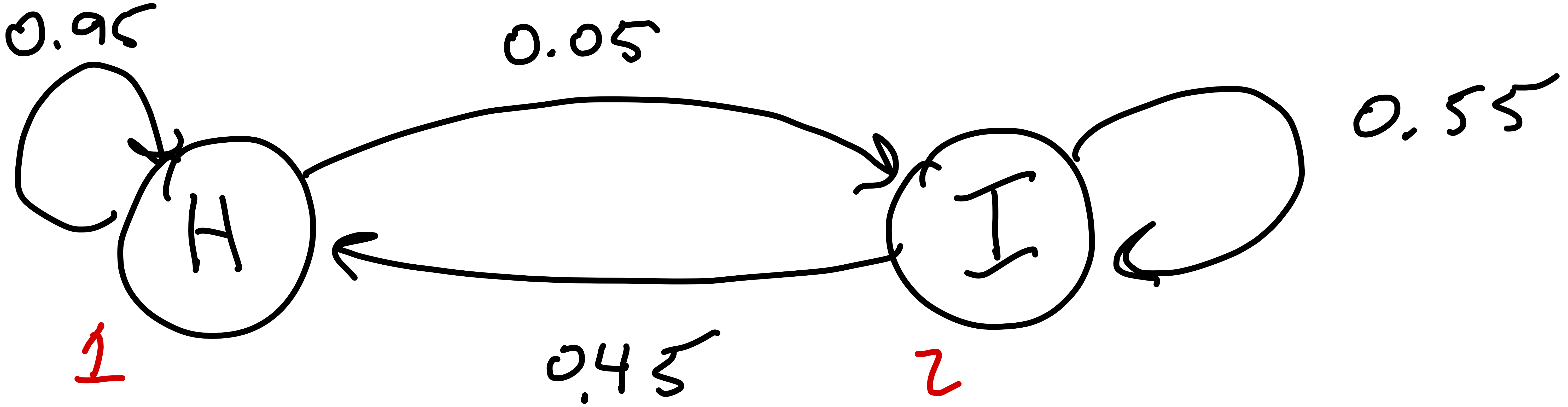
CAS CS 132

Practice Problem

$$\begin{bmatrix} 0.95 & 0.45 \\ 0.05 & 0.55 \end{bmatrix}$$

(LAA 4.9.3) On any given day a student is healthy or ill. Of the students healthy today, 5% will be ill tomorrow, and 55% of ill students will remain ill tomorrow.

Write down the stochastic matrix for this situation.



Objectives

1. Motivate matrix factorization in general, and the LU factorization in specific
2. Recall elementary row operations and connect them to matrices
3. Look at the LU factorization, how to find it, and how to use it

Keywords

elementary matrices

LU factorization

Catch up: State Diagrams

State Diagrams

$$\begin{bmatrix} 0.95 & 0.03 \\ 0.05 & 0.97 \end{bmatrix}$$

Definition. A **state diagram** is a directed weighted graph whose adjacency matrix is stochastic.

Example.

Naming Convention Clash

The nodes of a state diagram are often called states.

The vectors which are dynamically updated according to a linear dynamical system are called state vectors.

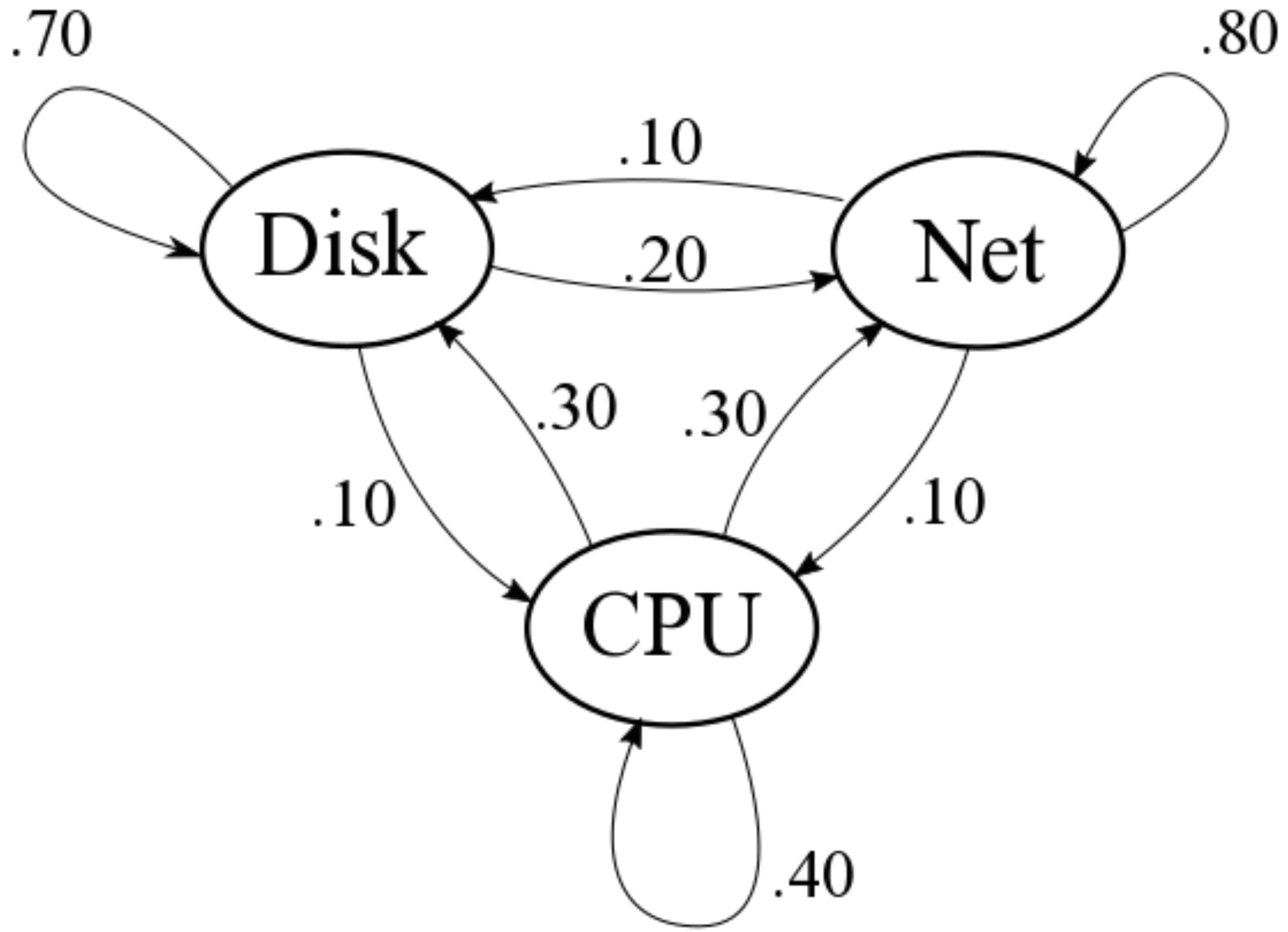
This is an unfortunate naming clash.

Example: Computer System

Imagine a computer system in which tasks request service from disk, network or CPU.

In the long term, which device is busiest?

This is about finding a stable state.

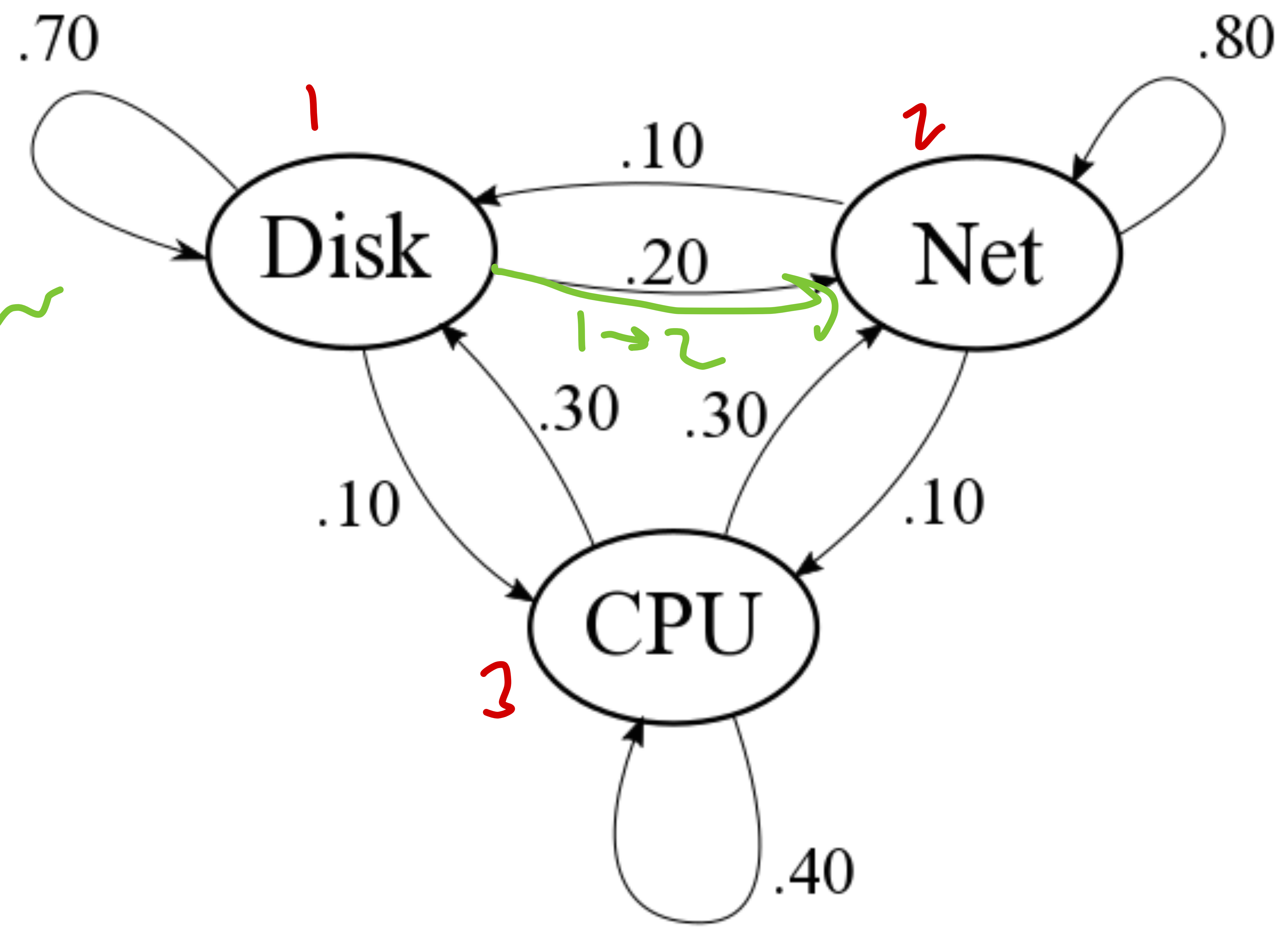


How To: State Diagram

Question. Given a state diagram, find the stable state for the corresponding linear dynamical system.

Solution. Find the adjacency matrix for the state diagram and go from there.

Example



$$\begin{bmatrix} 0.7 & 0.1 & 0.3 \\ 0.2 & 0.8 & 0.3 \\ 0.1 & 0.1 & 0.4 \end{bmatrix}$$

21

regular

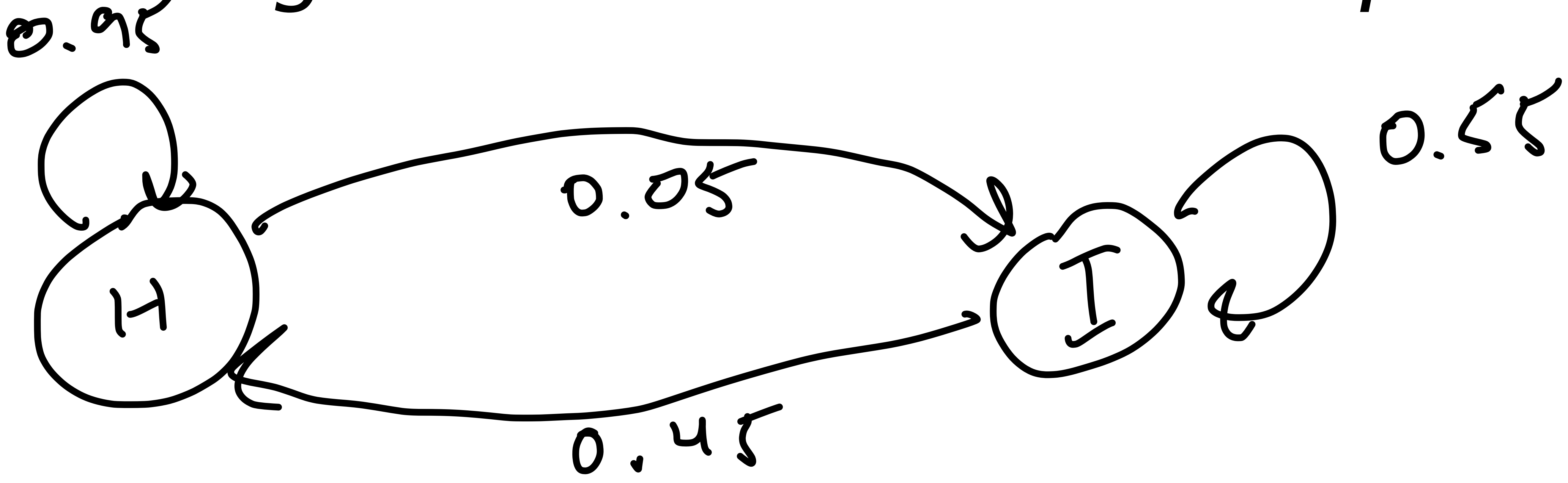
stochastic matrix

$$(A - I)\vec{x} = \vec{0} \Rightarrow \text{steady state}$$

Example

(LAA 4.9.3) On any given day a student is healthy or ill. Of the students healthy today, 5% will be ill tomorrow, and 55% of ill students will remain ill tomorrow.

Find the state diagram for the above problem.



Random Walks as Linear Dynamical Systems

Once we have a stochastic matrix, we can reason about random walks *as linear dynamical systems*.

What are its steady states?

How do we interpret these steady states?

Random Walks on State Diagrams

A **random walk** on a state diagram starting at v is the following process:

- choose a node v is connected to according to the *distribution* given by the edge weights
- go to that node
- repeat

Random Walks on State Diagrams

A **random walk** on a state diagram starting at v is the following process:

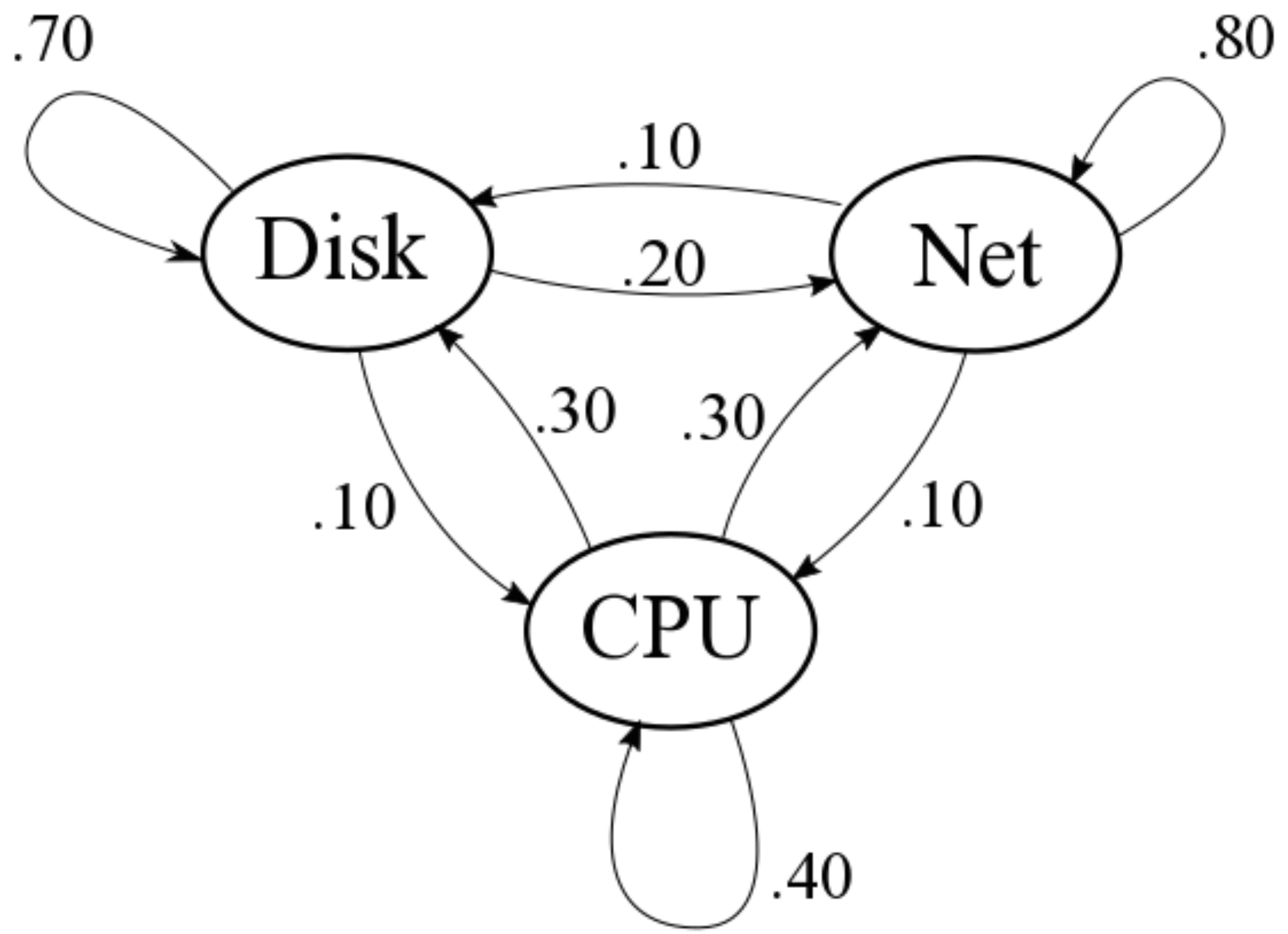
» c … o
the

Stable states of linear dynamical systems are stable states of random walks on state diagrams.

» g

» repeat

Example



Steady States of Random Walks

Theorem. Let A be the stochastic matrix for the graph G. The probability that a random walk starting at i of length k ends on node j is

$$(A^k)_{ji}$$

A transforms a distribution for length k walks to length $k+1$ walks.

Steady States of Random Walks

If a random walk goes on for a sufficiently long time, then the probability that we end up in a particular place becomes fixed.

If you wander for a sufficiently long time, it doesn't matter where you started.

moving on...

Motivation: Matrix Factorization

From Numbers to Matrices

From Numbers to Matrices

Much of linear algebra is about extending our intuitions about numbers to matrices.

From Numbers to Matrices

Much of linear algebra is about extending our intuitions about numbers to matrices.

For whole numbers, a **factor** of n is a number m such that m divides n.

From Numbers to Matrices

Much of linear algebra is about extending our intuitions about numbers to matrices.

For whole numbers, a **factor** of n is a number m such that m divides n.

2 is a factor of 10, 7 is a factor of 49,...

Polynomials

We've also likely seen this with polynomials, e.g.

$$x^3 + 6x^2 + 11x + 6 = (x + 1)(x + 2)(x + 3)$$

This is a **polynomial factorization.**

Matrix Factorization

Matrix Factorization

A **factorization** of a matrix A is an equation which expresses A as a product of one or more matrices, e.g.,

$$A = BC$$

Matrix Factorization

$A = AI$

A **factorization** of a matrix A is an equation which expresses A as a product of one or more matrices, e.g.,

$$A = BC$$

So far, we've been given two factors and asked to find their product.

Factorization is the harder direction.

A Warning: Intuitions only go so far

A Warning: Intuitions only go so far

`One nice feature of numbers is that they have a unique factorization into prime factors.`

A Warning: Intuitions only go so far

One nice feature of numbers is that they have a unique factorization into prime factors.

There is no such thing for matrices.

A Warning: Intuitions only go so far

One nice feature of numbers is that they have a unique factorization into prime factors.

There is no such thing for matrices.

This is a blessing and a curse:

> We have more than one kind of factorization but they tell us different things.

Reasons to Factorize

Reasons to Factorize

Writing A as the product of multiple matrices can

Reasons to Factorize

Writing A as the product of multiple matrices can

» make computing with A faster

Reasons to Factorize

Writing A as the product of multiple matrices can

» make computing with A faster

» make working with A easier

Reasons to Factorize

Writing A as the product of multiple matrices can

- make computing with A faster
- make working with A easier
- expose important information about A

Reasons to Factorize

Writing A as the product of multiple matrices can

» make computing with A faster LU Decomposition

» make working with A easier

» expose important information about A

The Problem

Question. For an matrix A, solve the equations

$$A\mathbf{x} = \mathbf{b}_1 \;\;, \;\; A\mathbf{x} = \mathbf{b}_2 \quad \ldots \quad A\mathbf{x} = \mathbf{b}_{k-1} \;\;, \;\; A\mathbf{x} = \mathbf{b}_k$$

In other words: we want to solve a bunch of matrix equations over the same matrix.

The Problem

Question. For a matrix A, solve (for X) in the equation

$$AX = B$$

where X and B are matrices of appropriate dimension.

This is (essentially) the same question.

The Problem

Question. Solve $AX = B$.

If A is *invertible*, then we have a solution:

Find A^{-1} and then $X = A^{-1}B$.

The Problem

Question. Solve $AX = B$.

If A is *invertible*, then we have a solution:

Find A^{-1} and then $X = A^{-1}B$.

What if A^{-1} is not invertible?
Even if it is, can we do it faster?

LU Factorization at a High Level

Given a $m \times n$ matrix A, we are going to factorize A as

echelon form of A

$$A = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ * & 1 & 0 & 0 \\ * & * & 1 & 0 \\ * & * & * & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} \blacksquare & * & * & * & * \\ 0 & \blacksquare & * & * & * \\ 0 & 0 & 0 & \blacksquare & * \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{U}$$

LU Factorization at a High Level

Given a $m \times n$ matrix A, we are going to factorize A as

echelon form of A

$$A = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ * & 1 & 0 & 0 \\ * & * & 1 & 0 \\ * & * & * & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} \blacksquare & * & * & * & * \\ 0 & \blacksquare & * & * & * \\ 0 & 0 & 0 & \blacksquare & * \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{U}$$

Note. This applies to non-square matrices

What are "L" and "U"?

`L stands for "lower" as in` *`lower triangular`*`.`

`U stands for "upper" as in` *`upper triangular`*`.`
`(This only happens when` A `is square.)`

$$A = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ * & 1 & 0 & 0 \\ * & * & 1 & 0 \\ * & * & * & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} \blacksquare & * & * & * \\ 0 & \blacksquare & * & * \\ 0 & 0 & \blacksquare & * \\ 0 & 0 & 0 & \blacksquare \end{bmatrix}}_{U}$$

Elementary Matrices

The Fundamental Question

$$A = LU$$

echelon form of A

The Fundamental Question

$$A = LU$$

echelon form of A

We know how to build U, that's just the forward phase of Gaussian elimination.

The Fundamental Question

$$A = LU$$ echelon form of A

We know how to build U, that's just the forward phase of Gaussian elimination.

How do we build L?

The Fundamental Question

$$A = LU$$ echelon form of A

We know how to build U, that's just the forward phase of Gaussian elimination.

How do we build L?

The idea. L "implements" the row operations of the forward phase.

Recall: Elementary Row Operations

scaling	multiply a row by a number
interchange	switch two rows
replacement	add a scaled equation to another

The First Key Observation

The First Key Observation

Elementary row operations are **linear transformations (viewed as transformation on columns)**

The First Key Observation

Elementary row operations are **linear transformations (viewed as transformation on columns)**

Example: Scale row 2 by 5

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \xrightarrow{R_2 \leftarrow 5R_2} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 5a_{21} & 5a_{22} & 5a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Example: Scaling

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \mapsto \begin{bmatrix} v_1 \\ 5v_2 \\ v_3 \end{bmatrix}$$

Restricted to one column, we see this is the above linear transformation.

Example: Scaling

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \mapsto \begin{bmatrix} v_1 \\ 5v_2 \\ v_3 \end{bmatrix}$$

Let's build the matrix which implements it:

$$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \mapsto \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \mapsto \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \mapsto \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 5 & 5 & 5 \\ 1 & 1 & 1 \end{bmatrix}$$

Another Example: ~~Scaling +~~ Replacement

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \longrightarrow \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ (a_{31} - 2a_{11}) & (a_{32} - 2a_{12}) & (a_{33} - 2a_{13}) \end{bmatrix}$$

$$R_3 \leftarrow (R_3 - 2R_1)$$

Another Example: Scaling + Replacement

Let's build the transformation: $R_3 \leftarrow (R_3 - 2R_1)$

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \mapsto \begin{bmatrix} x_1 \\ x_2 \\ x_3 - 2x_1 \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \mapsto \begin{bmatrix} 1 \\ 0 \\ -2 \end{bmatrix} \qquad \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \mapsto \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \qquad \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \mapsto \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -2 & 0 & 1 \end{bmatrix}$$

Another Example: Scaling + Replacement

Let's build the matrix which implements it:

Elementary row operations are linear, so they are implemented by matrices

General Elementary Scaling Matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & k & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

General Elementary Scaling Matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & k & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

If we want to perform $R_3 \leftarrow kR_3$ then we need the identity matrix but with the entry $A_{33} = k$.

General Elementary Scaling Matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & k & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

If we want to perform $R_3 \leftarrow kR_3$ then we need the identity matrix but with the entry $A_{33} = k$.

If we want to perform $R_i \leftarrow kR_i$ then we need the identity matrix but with then entry $A_{ii} = k$.

General Replacement Matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ k & 0 & 0 & 1 \end{bmatrix}$$

General Replacement Matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ k & 0 & 0 & 1 \end{bmatrix}$$

If we want to perform $R_4 \leftarrow R_4 + kR_1$, then we need the identity matrix but with the entry $A_{41} = k$.

General Replacement Matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ k & 0 & 0 & 1 \end{bmatrix}$$

If we want to perform $R_4 \leftarrow R_4 + kR_1$, then we need the identity matrix but with the entry $A_{41} = k$.

If we want to perform $R_i \leftarrow R_i + kR_j$, then we need the identity matrix but with the entry $A_{ij} = k$.

General Swap Matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

If we want to swap R_2 and R_3, then we need the identity matrix, but with R_2 and R_3 swapped.

Elementary Matrices

Definition. An **elementary matrix** is a matrix obtained by applying a single row operation to the identity matrix I.

Example.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \xrightarrow{R_2 \leftarrow 4R_2} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \xrightarrow{R_2 \leftarrow R_2 + 2R_1} \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}$$

Elementary Matrices

Definition. An **elementary matrix** is a matrix obtained by applying a single row operation to the identity matrix I.

These are exactly the matrices we were just looking at.

How To: Finding Elementary Matrices

Question. Find the matrix implementing the elementary row operation `op`.

Solution. Apply `op` to the identity matrix of the appropriate size.

Products of Elementary Matrices

Products of Elementary Matrices

Taking stock:

Products of Elementary Matrices

Taking stock:

» Elementary matrices implement elementary row operations.

Products of Elementary Matrices

Taking stock:

» Elementary matrices implement elementary row operations.

» Remember that Matrix multiplication is transformation composition (i.e., do one then the other).

Products of Elementary Matrices

Taking stock:

» Elementary matrices implement elementary row operations.

» Remember that Matrix multiplication is transformation composition (i.e., do one then the other).

So we can implement any sequence of row operations as a product of elementary matrices.

How to: Matrices implementing Row Operations

Question. Find the matrix implementing a sequence of row operations $\mathtt{op}_1$, $\mathtt{op}_2$,...

Solution. Apply the row operations in sequence to the identity matrix of the appropriate size.

Question

Find the matrix implementing the following sequence of elementary row operations on a $3 \times n$ *matrix.*

$$R_2 \leftarrow 3R_2$$

$$R_1 \leftarrow R_1 + R_2$$

$$R_2 \leftrightarrow R_3$$

Then multiply it with the all-ones 3×3 *matrix.*

$$\begin{bmatrix} 1 & 3 & 0 \\ 0 & 0 & 1 \\ 0 & 3 & 0 \end{bmatrix}$$

Answer

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \xrightarrow{R_2 \leftarrow 3R_3} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\xrightarrow{R_1 \leftarrow R_1 + R_2} \begin{bmatrix} 1 & 3 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\xrightarrow{R_2 \leftrightarrow R_3} \begin{bmatrix} 1 & 3 & 0 \\ 0 & 0 & 1 \\ 0 & 3 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = ?$$

Second Key Observation

Second Key Observation

Elementary row operations are **invertible** linear transformations.

Second Key Observation

Elementary row operations are **invertible** linear transformations.

This also means the product of elementary matrices is invertible.

$$(E_1E_2E_3E_4)^{-1} = E_4^{-1}E_3^{-1}E_2^{-1}E_1^{-1}$$

!! the order reverses !!

Question (Conceptual)

Describe the inverse transformation for each elementary row operation.

Question (Conceptual)

Describe the inverse transformation for each elementary row operation.

The inverse of scaling by k is scaling by $1/k$.

Question (Conceptual)

Describe the inverse transformation for each elementary row operation.

The inverse of scaling by k is scaling by $1/k$.

The inverse of $R_i \leftarrow R_i + kR_j$ is $R_i \leftarrow R_i - kR_j$.

Question (Conceptual)

Describe the inverse transformation for each elementary row operation.

The inverse of scaling by k is scaling by $1/k$.

The inverse of $R_i \leftarrow R_i + kR_j$ is $R_i \leftarrow R_i - kR_j$.

The inverse of swapping is swapping again.

LU Factorization

Recall: Elementary Row Operations

scaling	multiply a row by a number
interchange	switch two rows
replacement	add a scaled equation to another

Recall: Elementary Row Operations

We only need these two for the forward phase

interchange	switch two rows
replacement	add a scaled equation to another

Recall: Elementary Row Operations

We'll assume we only need this

replacement   add a scaled equation to another

Reminder: LU Factorization at a High Level

Given a $m \times n$ matrix A, we are going to factorize A as

Echelon form of A

$$A = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ * & 1 & 0 & 0 \\ * & * & 1 & 0 \\ * & * & * & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} \blacksquare & * & * & * & * \\ 0 & \blacksquare & * & * & * \\ 0 & 0 & 0 & \blacksquare & * \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{U}$$

LU Factorization Algorithm

LU Factorization Algorithm

```
FUNCTION LU_Factorization(A):
```

LU Factorization Algorithm

```
FUNCTION LU_Factorization(A):
    L ← identity matrix
```

LU Factorization Algorithm

```
FUNCTION LU_Factorization(A):
    L ← identity matrix
    U ← A
```

LU Factorization Algorithm

```
FUNCTION LU_Factorization(A):
    L ← identity matrix
    U ← A
    convert U to an echelon form by GE forward step # without swaps
```

LU Factorization Algorithm

```
FUNCTION LU_Factorization(A):
    L ← identity matrix
    U ← A
    convert U to an echelon form by GE forward step # without swaps
    FOR each row operation OP in the prev step:
```

LU Factorization Algorithm

```
FUNCTION LU_Factorization(A):
    L ← identity matrix
    U ← A
    convert U to an echelon form by GE forward step # without swaps
    FOR each row operation OP in the prev step:
        E ← the matrix implementing OP
```

LU Factorization Algorithm

```
FUNCTION LU_Factorization(A):
    L ← identity matrix
    U ← A
    convert U to an echelon form by GE forward step # without swaps
    FOR each row operation OP in the prev step:
        E ← the matrix implementing OP
        L ← L @ E^(-1)     # note the multiplication on the right
```

LU Factorization Algorithm

```
FUNCTION LU_Factorization(A):
    L ← identity matrix
    U ← A
    convert U to an echelon form by GE forward step # without swaps
    FOR each row operation OP in the prev step:
        E ← the matrix implementing OP
        L ← L @ E^{-1}    # note the multiplication on the right
    RETURN (L, U)
```

LU Factorization Algorithm

```
FUNCTION LU_Factorization(A):
    L ← identity matrix
    U ← A
    convert U to an echelon form by GE forward step # without swaps
    FOR each row operation OP in the prev step:
        E ← the matrix implementing OP
        L ← L @ E^-1     # note the multiplication on the right
    RETURN (L, U)   we'll see how to do this part smarter
```

Gaussian Elimination and Elementary Matrices

$$A \sim A_1 \sim A_2 \sim \dots \sim A_k$$

Consider a sequence of elementary row operations from A to an echelon form.

Each step can be represent as a **product with an elementary matrix.**

Gaussian Elimination and Elementary Matrices

$$A \sim E_1 A \sim E_2 E_1 A \sim \ldots \sim E_k E_{k-1} \ldots E_2 E_1 A$$

Gaussian Elimination and Elementary Matrices

$$A \sim E_1 A \sim E_2 E_1 A \sim \ldots \sim E_k E_{k-1} \ldots E_2 E_1 A$$

This exactly tells us that if B is the final echelon form we get then

$$B = (E_k E_{k-1} \ldots E_2 E_1) A = EA$$

where E implements a <u>sequence</u> of row operations.

Gaussian Elimination and Elementary Matrices

$$A \sim E_1 A \sim E_2 E_1 A \sim \ldots \sim E_k E_{k-1} \ldots E_2 E_1 A$$

This exactly tells us that if B is the final echelon form we get then

Invertible

$$B = (E_k E_{k-1} \ldots E_2 E_1) A = EA$$

where E implements a <u>sequence</u> of row operations.

Gaussian Elimination and Elementary Matrices

$$A \sim E_1 A \sim E_2 E_1 A \sim \ldots \sim E_k E_{k-1} \ldots E_2 E_1 A$$

This exactly tells us that if B is the final echelon form we get then

Invertible

$$B = (E_k E_{k-1} \ldots E_2 E_1) A = EA$$

where E implements a <u>sequence</u> of row operations.

So
$$A = E^{-1} B = (E_1^{-1} E_2^{-1} \ldots E_{k-1}^{-1} E_k^{-1}) B$$

echelon form

A New Perspective on Gaussian Elimination

The forward part of Gaussian elimination is matrix factorization

$$A = E_1^{-1} E_1 A = \underbrace{E_1^{-1} E_2^{-1}}_{L} \underbrace{E_2 E_1 A}_{U} \quad \cdots$$

The "L" Part

$$E = E_k E_{k-1} \cdots E_2 E_1$$

This a product of elementary matrices

So $L = E^{-1} = E_1^{-1} E_2^{-1} \ldots E_{k-1}^{-1} E_k^{-1}$ **!! the order reverses !!**

We won't prove this, but it's worth thinking about: **why is this lower triangular?**

And can we build this in a more efficient way?

demo

How To: LU Factorization by hand

Question. Find a LU Factorization for the matrix A (assuming no swaps).

Solution.

» Start with L as the identity matrix.
» Find U by the forward part of GE.
» For each operation $R_i \leftarrow R_i + kR_j$, set L_{ij} to $-k$.

Solving Systems using the LU Factorization

Analyzing Linear Algebra Algorithms

Analyzing Linear Algebra Algorithms

We will not use $O(\cdot)$ notation!

Analyzing Linear Algebra Algorithms

We will not use $O(\cdot)$ notation!

For numerics, we care about number of **FL**oating–oint **OP**erations (FLOPs):

>> addition
>> subtraction
>> multiplication
>> division
>> square root

Analyzing Linear Algebra Algorithms

We will not use $O(\cdot)$ notation!

For numerics, we care about number of **FL**oating-oint **OP**erations (FLOPs):

>> addition
>> subtraction
>> multiplication
>> division
>> square root

$2n$ vs. n is very different when $n \sim 10^{20}$

Dominant Terms

Dominant Terms

`that said, we don't care about` *`exact`* `bounds`

Dominant Terms

that said, we don't care about *exact* bounds

A function $f(n)$ is ***asymptotically equivalent*** to $g(n)$ if

$$\lim_{i \to \infty} \frac{f(i)}{g(i)} = 1$$

Dominant Terms

that said, we don't care about *exact* bounds

A function $f(n)$ is ***asymptotically equivalent*** to $g(n)$ if

$$\lim_{i \to \infty} \frac{f(i)}{g(i)} = 1$$

for polynomials, they are equivalent to their dominant term

Dominant Terms

the dominant term of a polynomial is the monomial with the highest degree

$$\lim_{i \to \infty} \frac{3x^3 + 100000x^2}{3x^3} = 1$$

$3x^3$ dominates the function even though the coefficient for x^2 is so large

How To: Solving systems with the LU

Question. Solve the equation $A\mathbf{x} = \mathbf{b}$ given that $A = LU$ is a LU factorization.

Solution. First solve $L\mathbf{x} = \mathbf{b}$ to get a solution $\mathbf{c}$, then solve $U\mathbf{x} = \mathbf{c}$ to get a solution $\mathbf{d}$.

Verify: $A\vec{x} = b$ $\quad L\vec{c} = \vec{b}$ $\quad U\vec{d} = \vec{c}$

$A\vec{d} = LU\vec{d} = L\vec{c} = \vec{b}$

How To: Solving systems with the LU

Question. Solve the equation $A\mathbf{x} = \mathbf{b}$ given that $A = LU$ is a LU factorization.

Solution. First solve $L\mathbf{x} = \mathbf{b}$ to get a solution $\mathbf{c}$, then solve $U\mathbf{x} = \mathbf{c}$ to get a solution $\mathbf{d}$.

Why is this better than just solving $A\mathbf{x} = \mathbf{b}$?

FLOPs for Solving General Systems

The following FLOP estimates are based on $n \times n$ matrices

Gaussian Elimination: $\sim \frac{2n^3}{3}$ FLOPS

GE Forward: $\sim \frac{2n^3}{3}$ FLOPS

GE Backward: $\sim 2n^2$ FLOPS

Matrix Inversion: $\sim 2n^3$ FLOPS

Matrix–Vector Multiplication: $\sim 2n^2$ FLOPS

Solving by matrix inversion: $\sim 2n^3$ FLOPS

Solving by Gaussian elimination: $\sim \frac{2n^3}{3}$ FLOPS

FLOPS for solving LU systems

LU Factorization: $\sim \frac{2n^3}{3}$ FLOPS   forward part GE

Solving $L\mathbf{x} = \mathbf{b}$: $\sim 2n^2$ FLOPS (by "forward" elimination)

Solving $U\mathbf{x} = \mathbf{c}$: $\sim 2n^2$ FLOPS (already in echelon form)

Solving by LU Factorization: $\sim \frac{2n^3}{3}$ FLOPS

If you solve several matrix equations for the same matrix, ***LU factorization*** *is* <u>*faster*</u> *than* ***matrix inversion*** *on the first equation, and the same (asymptotically) in later equations (and it works for rectangular matrices).*

Other Considerations: Density

Other Considerations: Density

If A doesn't have to many entries (A is **sparse)**, then its likely that L and U won't either.

Other Considerations: Density

If A doesn't have to many entries (A is **sparse)**, then its likely that L and U won't either.

But A^{-1} may have ***many*** entries (A^{-1} is **dense**)

Other Considerations: Density

If A doesn't have to many entries (A is **sparse**), then its likely that L and U won't either.

But A^{-1} may have ***many*** entries (A^{-1} is **dense**)

Sparse matrices are faster to compute with and better with respect to storage.

Summary

We can factorize matrices to make them easier to work with, or get more information about them

LU Factorizations allow us to solve multiple matrix equations, with one forward step and multiple backwards steps.