Elementi di Informatica e Applicazioni Numeriche T

Polinomi e `fprintf`

Calore Molare

Il calore (specifico) molare di un sostanza:

- È la quantità di calore necessaria per variare la temperatura di 1K
- Dipende dalla temperatura, secondo una legge polinomiale:

$$c_p^* = a + b\,T + c\,T^2 + d\,T^3$$

- Il polinomio è al più di 4° grado
- I coefficienti sono stati determinati empiricamente per varie sostanze

Il valore di c_p^* è necessario per calcolare la differenza di Entalpia

$$\Delta H = \int_{T_0}^{T_1} c_p^*\, dT$$

- Avete visto questi argomenti nel corso di Termodinamica

Polinomi in Matlab

Matlab fornisce funzioni per operare su polinomi

Un polinomio viene rappresentato come un vettore di coefficienti:

$$\underline{c_n} x^n + \underline{c_{n-1}} x^{n-1} + \ldots + \underline{c_1} x + \underline{c_0} \longleftrightarrow p = (c_n, c_{n-1}, \ldots c_1, c_0)$$

- Il primo elemento del vettore corrisponde a c_n
- L'ultimo elemento del vettore corrisponde a c_0
- Il grado del polinomio è dato da `length(p)-1`

Quindi, per il calore molare:

- Il polinomio $a + b\,T + c\,T^2 + d\,T^3$
- Viene rappresentato come: `[d, c, b, a]` (invertito!)

Valutazione di Polinomi in Matlab

Per valutare un polinomio per un valore di x noto si usa:

```
function X = polyval(P, X)
```

- `P` è il polinomio da valutare
- `X` è il valore di x e può essere anche un vettore
- `Y` è il vettore con la valutazione di ogni elemento di `X`

Per esempio, per disegnare il calore molare in funzione di T:

```
p = [d, c, b, a];
T = linspace(25, 200, 300); % 300 punti
Y = polyval(p, T);
plot(T, Y);
```

Integrazione e Derivazione di Polinomi

Matlab fornisce anche funzioni per integrare e derivare polinomi

Dato un polinomio:

$$c_n\, x^n + c_{n-1}\, x^{n-1} + \ldots + c_1\, x + c_0$$

- La sua derivata è un altro polinomio, ossia:

$$n\, c_n\, x^{n-1} + (n-1) c_{n-1}\, x^{n-2} + \ldots + c_1$$

- Il suo integrale è un altro polinomio, ossia:

$$\frac{1}{n+1} c_n\, x^{n+1} + \frac{1}{n} c_n\, x^{n-1} + \ldots + \frac{1}{2} c_1\, x^2 + c_0\, x$$

Integrazione e Derivazione di Polinomi

In termini della rappresentazione utilizzata da Matlab:

Derivando otteniamo un polinomio inferiore di un grado:

$$(c_n, c_{n-1}, \dots c_1, c_0) \longrightarrow (nc_n, (n-1)c_{n-1}, \dots c_1)$$

Per l'integrale otteniamo un polinomio superiore di un grado:

$$(c_n, c_{n-1}, \dots c_1, c_0) \longrightarrow \left(\frac{1}{n+1}c_n, \frac{1}{n}c_{n-1}, \dots \frac{1}{2}c_1, c_0, 0\right)$$

Si possono ottenere con `polyint(p)` e `polyder(p)`

```
polyder([1, 2, 4]) % denota [2, 2]
polyint([1, 2, 4]) % denota [1/3, 1, 4, 0]
```

Differenza di Entalpia

A questo punto è facile (e.g.) calcolare la differenza di Entalpia

Per esempio, per l'n-ottano dell'esercitazione LADI 1:

$$\Delta H = \int_{T_0}^{T_1} c_p^* \, dT$$

```
p = [8.8551E-08, -4.1950E-04, 7.7121E-01, -6.096];
T0 = 25+273.15; % 25 °C in °K
T1 = 450+273.15; % 450 °C in °K
P = polyint(p); % Polinomio integrale
DH = polyval(P, T1) - polyval(P, T0);
fprintf('DH = %.3f', DH)
```

- La funzione **`fprintf`** serve a stampare con formattazione controllata

Funzione `fprintf`

La funzione `fprintg` stampa con formattazione controllata:

L'interfaccia della funzione è:

```
function fprintf(FORMAT, E1, E2, ...)
```

- **`FORMAT`** è una stringa da stampare
- **`E1`**, **`E2`** sono espressioni (e.g. nomi di variabile)
- **`FORMAT`** può contenere dei "segnaposto", con il carattere **`%`**
  - E.g. **`%f`** è un segnaposto per un valore reale
  - E.g. **`%d`** è un segnaposto per un valore intero
- Il primo segnaposto è sostituito con il valore di **`E1`**
- Il secondo segnaposto è sostituito con il valore di **`E2`**, etc.

Funzione `fprintf`

Vediamo un esempio:

```
A = 10.5;
B = 2;
fprintf('A = %f, B = %f, A*B = %f\n', A, B, A*B)
```

Stampa: `A = 10.500000, B = 2.000000, A*B = 21.000000`

- "`\n`" è un carattere speciale e serve ad andare a capo

I segnaposto cono configurabili

Ci interessa un caso solo: "`%.Nf`" stampa un reale con `N` valori decimali

- E.g. `%.3f` stampa con 3 cifre decimali
- E.g. `%.1f` stampa con 1 cifra decimale

Elementi di Informatica e Applicazioni Numeriche T

Sistemi Tempo-Discreti Lineari

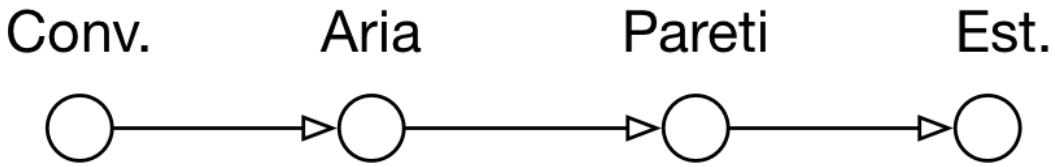
Riscaldare una Stanza

Supponiamo di volere riscaldare una stanza con un convettore

Indicativamente, quello che succede è:

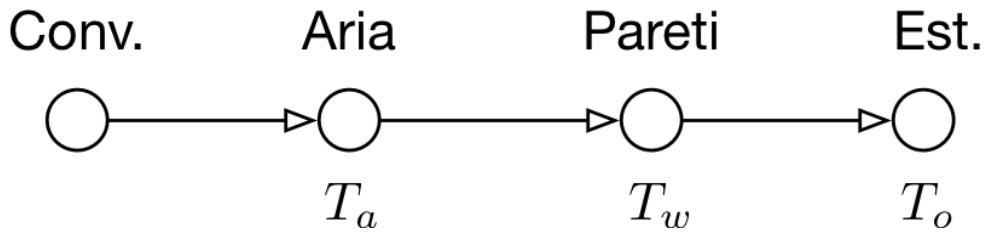
- Il convettore emette un flusso di calore costante
- Il flusso di calore scalda l'aria nella stanza
- Il calore passa dall'aria alle pareti
- Il calore passa dalle pareti all'esterno, che ha temperature nota

Schematicamente, abbiamo quattro "nodi":



Riscaldare una Stanza

Tre nodi sono caratterizzati da una temperatura



Il concetto di temperatura è generalizzabile!

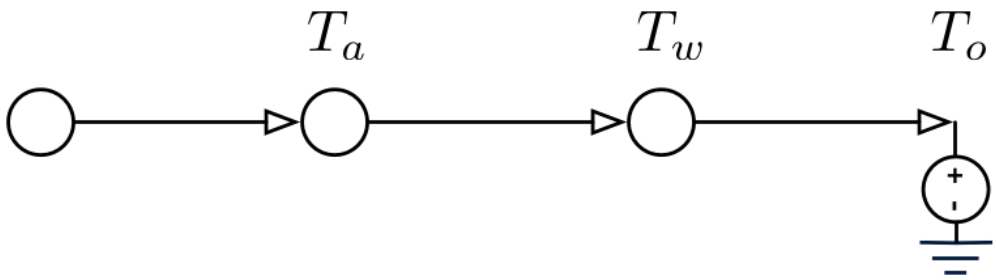
- Infatti, descriveremo il sistema...
- ...Utilizzando una notazione progettata per i circuiti elettrici

Non preoccupatevi se non avete ancora fatto elettrotecnica!

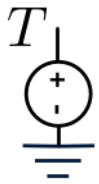
- Alla fine arriveremo ad un insieme di equazioni ben comprensibili

Riscaldare una Stanza

La temperatura T_O è approssimativamente costante



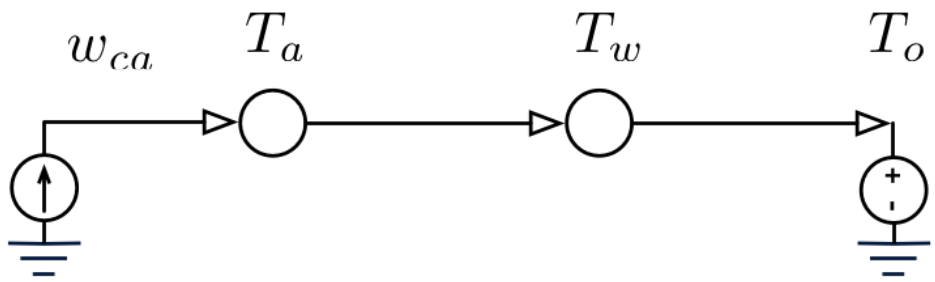
Per indicare una temperatura costante si usa:



- Curiosità: corrisponde ad generatore di tensione collegato a terra

Riscaldare una Stanza

Dal convettore arriva un flusso di calore w_{ca} costante

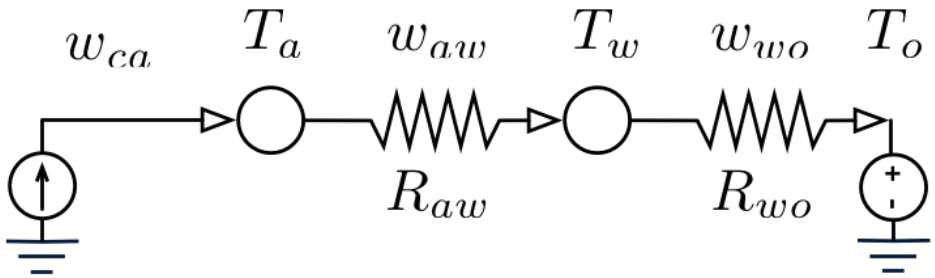


Per indicare una flusso costante si usa:

- Curiosità: corrisponde ad generatore di corrente collegato a terra

Riscaldare una Stanza

Le frecce rappresentano flussi di calore w attraverso un mezzo



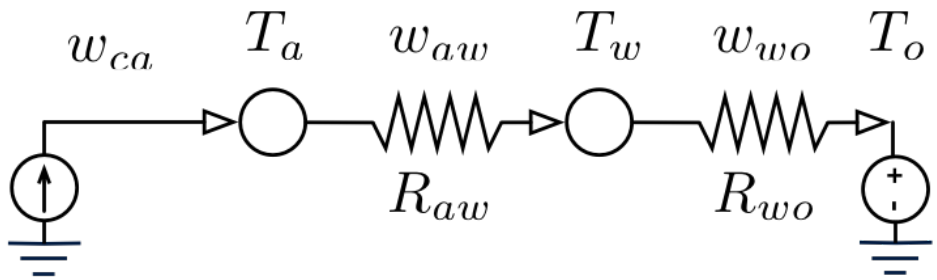
Il flusso di calore dipende dalla temperatura dei due estremi:

$$w = \frac{1}{R}(T_0 - T_1)$$

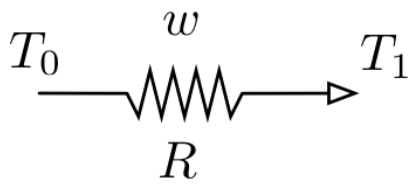
- 0 è l'estremo a monte e 1 quello a valle della freccia
- R rappresenta la resistenza termica del mezzo

Riscaldare una Stanza

Le frecce rappresentano direzioni di flussi di calore *w*



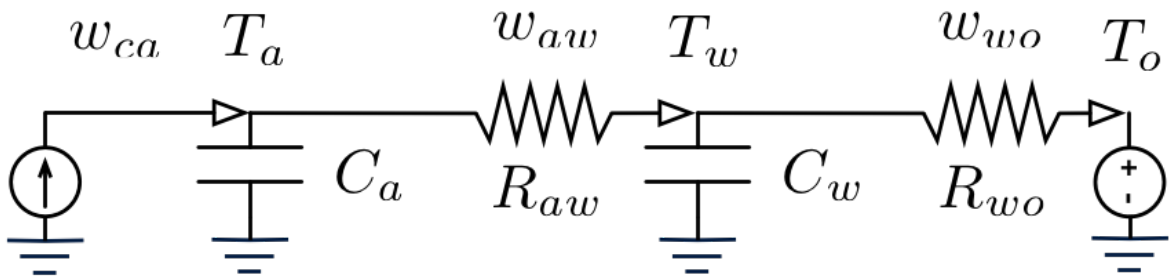
Un flusso che dipende linearmente dalla temp. si rappresenta con:



- Curiosità: corrisponde ad un resistore elettrico

Riscaldare una Stanza

L'aria ed le pareti sono masse, che possono accumulare calore



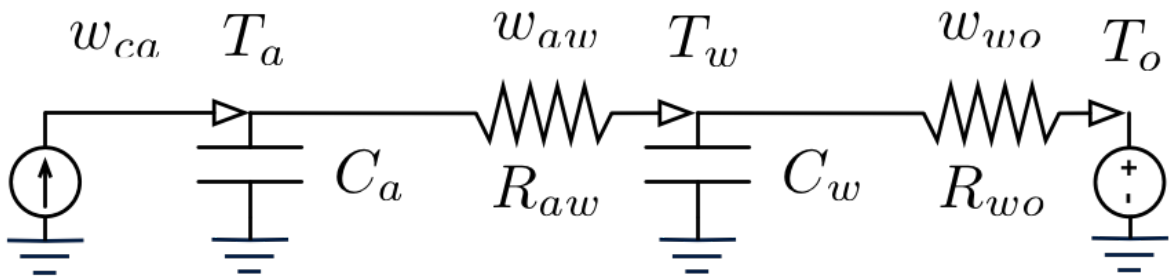
- La variazione della loro temperatura per unità di tempo (e.g 1 sec)...
- ...Dipende dal flusso di calore in ingresso/uscita

$$\Delta T = \frac{1}{C} w \longrightarrow T^{(k+1)} = T^{(k)} + \frac{1}{C} w^{(k)}$$

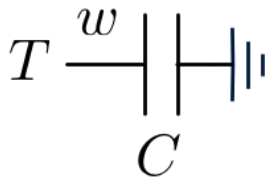
- C è la capacità termica della massa

Riscaldare una Stanza

L'aria ed le pareti sono masse, che possono accumulare calore



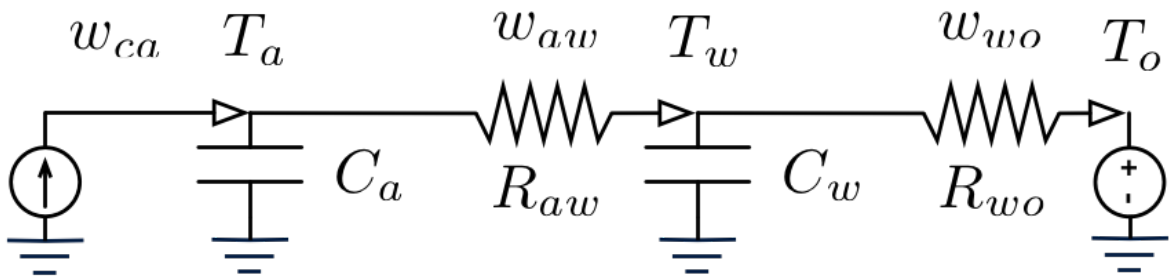
Un accumulatore di calore si rappresenta con:



- Curiosità: corrisponde ad un condensatore elettrico collegato a terra

Modellazione RC

Questo approccio si chiama modellazione RC

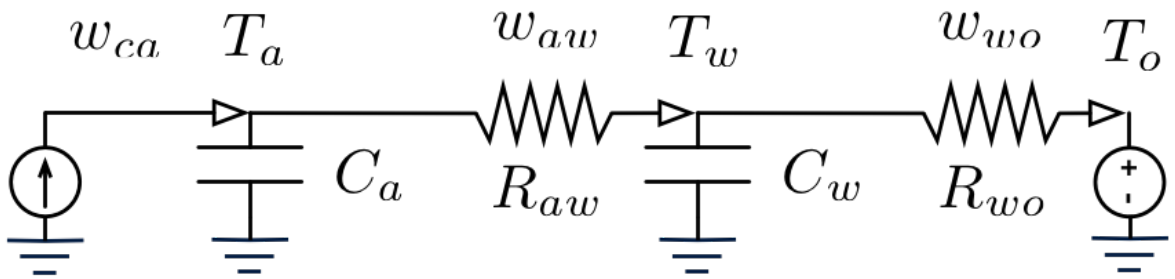


- Abbiamo una grandezza fisica che fluisce tra i nodi (e.g. calore)
- Le capacità C accumulano questa quantità
- Abbiamo una grandezza fisica che determina i flussi (e.g. temperatura)
- Le resistenze R permettono i flussi

È utile impararle perché è molto generalizzabile!

Modellazione RC

In termini di equazioni, per i flussi abbiamo:



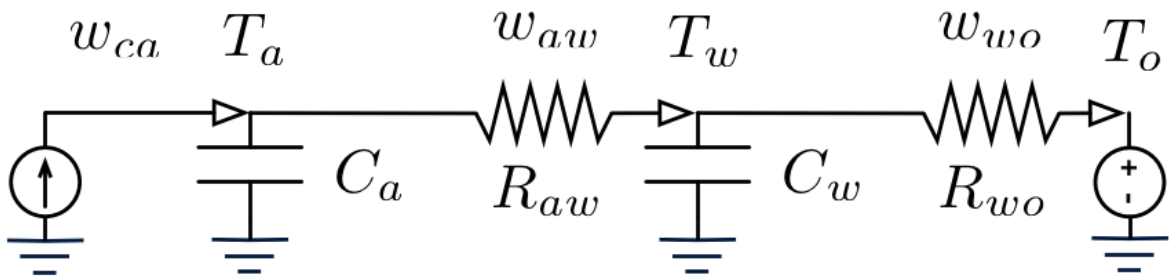
$$w_{ca}^{(k+1)} = w_{ca}$$ Calore tra convettore ed aria

$$w_{aw}^{(k+1)} = \frac{1}{R_{aw}}(T_a^{(k)} - T_w^{(k)})$$ Calore tra aria e pareti

$$w_{wo}^{(k+1)} = \frac{1}{R_{wo}}(T_w^{(k)} - T_o)$$ Calore tra pareti ed esterno

Modellazione RC

In termini di equazioni, per le temperature abbiamo:



$$T_a^{(k+1)} = T_a^{(k)} + \frac{1}{C_a}(w_{ca}^{(k)} - w_{aw}^{(k)})$$ Temperatura dell'aria

$$T_w^{(k+1)} = T_w^{(k)} + \frac{1}{C_w}(w_{aw}^{(k)} - w_{wo}^{(k)})$$ Temperatura delle pareti

Le capacità determinano come varia la temperatura

Modellazione RC

Complessivamente, il sistema è descritto da:

$$T_a^{(k+1)} = T_a^{(k)} + \frac{1}{C_a}(w_{ca}^{(k)} - w_{aw}^{(k)})$$ Temperatura dell'aria

$$T_w^{(k+1)} = T_w^{(k)} + \frac{1}{C_w}(w_{aw}^{(k)} - w_{wo}^{(k)})$$ Temperatura delle pareti

$$w_{ca}^{(k+1)} = w_{ca}$$ Calore tra convettore ed aria

$$w_{aw}^{(k+1)} = \frac{1}{R_{aw}}(T_a^{(k)} - T_w^{(k)})$$ Calore tra aria e pareti

$$w_{wo}^{(k+1)} = \frac{1}{R_{wo}}(T_w^{(k)} - T_o)$$ Calore tra pareti ed esterno

Modellazione RC

Le equazioni sono lineari, si può usare una notazione matriciale

$$\begin{pmatrix} T_a^{(k+1)} \\ T_w^{(k+1)} \\ w_{ca}^{(k+1)} \\ w_{aw}^{(k+1)} \\ w_{wo}^{(k+1)} \end{pmatrix} = \begin{pmatrix} 1 & 0 & +\frac{1}{C_a} & -\frac{1}{C_a} & 0 \\ 0 & 1 & 0 & +\frac{1}{C_w} & -\frac{1}{C_w} \\ 0 & 0 & 0 & 0 & 0 \\ \frac{1}{R_{aw}} & -\frac{1}{R_{aw}} & 0 & 0 & 0 \\ 0 & \frac{1}{R_{wo}} & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} T_a^{(k)} \\ T_w^{(k)} \\ w_{ca}^{(k)} \\ w_{aw}^{(k)} \\ w_{wo}^{(k)} \end{pmatrix} + \begin{pmatrix} 0 \\ \\ w_{ca} \\ 0 \\ -\frac{1}{R_{ow}} T_o \end{pmatrix}$$

In sintesi, **lo stato x evolve secondo una legge del tipo:**

$$x^{(k+1)} = Ax^{(k)} + b$$

Il sistema dinamico si dice tempo-discreto e lineare

Modellazione RC

Abbiamo visto un esempio di modellazione RC

- Si chiama anche "modellazione al circuito equivalente RC"
- È facilmente generalizzabile (e.g. calore = acqua, capacità = serbatoi)
- Porta sempre a sistemi dinamici lineari

Nel nostro caso, abbiamo ottenuto un sistema tempo-discreto

- Il sistema evolve secondo la legge:

$$x^{(k+1)} = Ax^{(k)} + b$$

- In realtà, la modellazione RC nasce per sistemi tempo-continui
- Nel caso discreto, il passo deve coincidere con una unità di tempo
- Nel nostro caso: un passo = 1 secondo

Elementi di Informatica e Applicazioni Numeriche T

Funzioni come Parametri
e Funzioni Anonime

Repetita Non Iuvant

In tutti gli esempi visti il codice di simulazione (di fatto) è:

```
xc = x0; % Inizializzazione dello stato
for t = T % T = vettore degli istanti di tempo
    X(t, :) = xc; % X(t, :) la t-ma riga di X
    xc = f(xc); % Avanzamento dello stato
end
X(end, :) = xc; % Salvataggio dello stato finale
```

Ripeterlo ogni volta non è una buona idea!

- Si possono commettere piccoli errori
- Se facciamo una modifica, va ripetuta in tutti gli esercizi

Perché allora non incapsularlo in una funzione?

Definizione della Funzione `simulate`

Cominciamo definendo, nel file di funzione "`simulate.m`":

```
function X = simulate(...)
    xc = x0;
    for t = T
        X(t, :) = xc;
        xc = f(xc);
    end
    X(end, :) = xc;
end
```

Quali parametri dovrà avere la funzione?

Definizione della Funzione `simulate`

Cominciamo definendo, nel file di funzione "`simulate.m`":

```
function X = simulate(..., T, ...)
    xc = x0;
    for t = T
        X(t, :) = xc;
        xc = f(xc);
    end
    X(end, :) = xc;
end
```

Quali parametri dovrà avere la funzione?

- Il vettore `T` degli istanti di tempo

Definizione della Funzione `simulate`

Cominciamo definendo, nel file di funzione "`simulate.m`":

```
function X = simulate(..., T, x0)
    xc = x0;
    for t = T
        X(t, :) = xc;
        xc = f(xc);
    end
    X(end, :) = xc;
end
```

Quali parametri dovrà avere la funzione?

- Il vettore `T` degli istanti di tempo
- Lo stato iniziale `x0`

Definizione della Funzione `simulate`

Cominciamo definendo, nel file di funzione "`simulate.m`":

```
function X = simulate(f, T, x0)
    xc = x0;
    for t = T
        X(t, :) = xc;
        xc = f(xc);
    end
    X(end, :) = xc;
end
```

Quali parametri dovrà avere la funzione?

- Il vettore `T` degli istanti di tempo
- Lo stato iniziale `x0`
- E la funzione di transizione `f`!

Funzioni come Parametri

In Matlab, si può specificare una funzione come parametro

Si dichiara un normale parametro formale

```
function X = simulate(f, T, x0)
```

- Siamo noi a decidere che `f` dovrà contenere una funzione

Per utilizzarla, la si chiama utilizzando il nome del parametro:

```
...
xc = f(xc)
...
```

- Pensatela come: `f` è una variabile e denota un "oggetto funzione"...
- ...una volta ottenuto, "l'oggetto funzione" viene invocato

Funzioni come Parametri

Come utilizzare la nostra nuova funzione `simulate`?

Una prima ipotesi:

```
x0 = [1, 0]; % Definisco lo stato iniziale
T = 1:100; % Vettore degli istanti di tempo
X = simulate(ftr, T, xc);

function xf = ftr(xc) % Funzione di transizione
    ...
end
```

Questo approccio non funziona

- Matlab pensa che vogliamo invocare `ftr` senza parametri...
- ...E solleva un errore perché ci siamo "dimenticati" di passargli `xc`

Funzioni come Parametri

Come utilizzare la nostra nuova funzione `simulate`?

Una prima ipotesi:

```
x0 = [1, 0]; % Definisco lo stato iniziale
T = 1:100; % Vettore degli istanti di tempo
X = simulate(ftr, T, xc);

function xf = ftr(xc) % Funzione di transizione
    ...
end
```

Occorre un modo per dire a Matlab che:

- Non vogliamo invocare `ftr` senza parametri...
- ...Ma vogliamo passare `ftr` come "oggetto funzione"

Funzioni come Parametri

Come utilizzare la nostra nuova funzione `simulate`?

Una prima ipotesi:

```
x0 = [1, 0]; % Definisco lo stato iniziale
T = 1:100; % Vettore degli istanti di tempo
X = simulate(@ftr, T, xc); % NOTAZIONE CON @

function xf = ftr(xc) % Funzione di transizione
    ...
end
```

La notazione "@<nomefunzione>":

- Denota la funzione di nome **<nomefunzione>** come oggetto
- "L'oggetto funzione" viene passato come parametro a **simulate**

Meglio Generalizzare per Bene

È utile generalizzare la funzione `simulate` come segue:

```
function X = simulate(f, T, x0)
    xc = x0;
    for t = T
        X(t, :) = xc;
        xc = f(xc, t); % <---- ERA: f(xc)
    end
    X(end, :) = xc;
end
```

- Invochiamo la funzione di transizione passando lo stato…
- …Ma anche l'istante di tempo corrente
- È utile per ingressi variano nel tempo indipendentemente dallo stato
- E.g. dopo 1800 secondi spengo il convettore

Meglio Generalizzare per Bene

Come effetto collaterale, quando si usa `simulate`...

Occorre passare funzioni di transizione che accettino i due parametri:

```
X = simulate(@ftr, T, xc);

function xf = ftr(xc, t) % Stato corrente e tempo!
    ...
end
```

- Se non lo facciamo, quando `simulate` chiamerà `f(xc, t)`...
- ...Matlab si accorgerà del parametro di troppo e segnalerà un errore

Se `t` non serve, si può semplicemente non utilizzarlo nel codice di `ftr`

Ancora Qualche Problema

Come usare `simulate` per l'esempio del riscaldamento?

Potremmo iniziare con qualcosa del genere:

```
function xf = f(xc, t)
    xc = xc'; % Stato come vettore riga
    xf = A * xc + b; % Stato futuro come colonna
    xf = xf'; % Stato futuro come riga
end
```

Le trasposizioni sono necessarie perché:

- La moltiplicazione a dx richiede che **`xc`** sia un vettore colonna
- Ma la nostra **`simulate`** tratta lo stato come una riga

Ancora non ci siamo, però! Le variabili **`A`** e **`b`** non sono definite!

Ancora Qualche Problema

Come ottenere `A` e `b`?

La prima soluzione consiste nel definirle dentro `f`

```
function xf = f(xc, t)
    A = ...; % Calcoli per cosruire A
    b = ...; % Calcoli per costruire b
    xf = A * xc' + b;
    xf = xf';
end
```

Questa soluzione però è inefficiente!

- Ogni volta che `f` viene chiamata...
- ...Ripetiamo i calcoli per costruire `A` e `b`

Ancora Qualche Problema

Come ottenere `A` e `b`?

Seconda soluzione: li passiamo come parametri

```
function xf = f(xc, t, A, b)
    xf = A * xc' + b;
    xf = xf';
end
```

In questo modo i calcoli non sono ripetuti!

- Però la funzione di transizione ha molti parametri...
- ...In particolare, non ha solo `xc` e `t`

Quindi è incompatibile con la nostra `simulate`!

Funzioni Anonime

Il problema è aggirabile utilizzando un nuovo costrutto

In Matlab si può costruire una funzione anonima con:

```
f = @(<parametri>) <expr.>
```

- `f` è una variabile, che contiene un "oggetto funzione"...
- I parametri della funzione costruita sono specificati in "`<parametri>`"
- Il corpo della funzione costruita è dato da "`<expr.>`"
- Quando è eseguita, la funzione restituisce il risultato di `<expr.>`

Sono utili per costruire rapidamente funzioni semplici

- E.g. `f = @(x) 2*x^2 -x +3` (una parabola con coefficienti noti)

Funzioni Anonime ed Ambienti

C'è un'altra ragione per cui le funzioni anonime sono utili:

Normalmente, una funzione può accedere solo:

- Ai propri parametri
- Alle variabili locali

Invece, una funzione anonima può accedere:

- Ai proprio parametri
- Alle variabili dell'ambiente in cui viene costruita

Questa proprietà le rende utilissime!

- Si capisce meglio con un esempio...

Funzioni Anonime: Utilizzo Tipico

```
function es_heating()
    A = ... % Definizione dei coefficienti
    b = ... % Definizione del termine noto

    x0 = ... % Stato iniziale
    T = 1:7200; % Orizzonte di simulazione
    ftr = @(xc, t) f(xc, t, A, b) % FUNZIONE ANONIMA
    X = simulate(ftr, T, x0); % Simulazione
    ... % Disegno
end

function xf = f(xc, t, A, b)
    xf = A * xc' + b;
    xf = xf';
end
```

Funzioni Anonime: Utilizzo Tipico

La funzione di transizione "vera" è:

```
function xf = f(xc, t, A, b)
    xf = A * xc' + b;
    xf = xf';
end
```

- Riceve **`A`** e **`b`** come parametri, quindi è incompatibile con **`simulate`**

La funzione anonima:

```
ftr = @(xc, t) f(xc, t, A, b)
```

- Espone solo due parametri **`xc`** e **`t`**: è compatibile con simulate
- Quando viene invocata, passa ad **`f`** anche **`A`** e **`b`**
- **`A`** e **`b`** sono recuperati dall'ambiente in cui è stata costruita!

Funzioni Anonime: Utilizzo Tipico

```
function es_heating()
    A = ... % La funzione anonima recupera questo A!
    b = ... % La funzione anonima recupera questo b!

    x0 = ...
    T = 1:7200;
    ftr = @(xc, t) f(xc, t, A, b)
    X = simulate(ftr, T, x0);
    ...
end

function xf = f(xc, t, A, b)
    xf = A * xc' + b;
    xf = xf';
end
```

Funzioni Anonime: Utilizzo Tipico

In questo modo abbiamo aggirato il problema originale!

- La funzione di transizione `f` ha anche `A` e `b` come parametri
- Usiamo una funzione anonima per "avvolgere" `f`...
- ...E nascondere i parametri incompatibili con `simulate`

La funzione anonima viene memorizzata in una variabile (i.e. `ftr`):

```
ftr = @(xc, t) f(xc, t, A, b)
```

- La variabile `ftr` contiene già un "oggetto funzione"...
- ...Quindi può essere passata a `simulate` senza la "`@`":

```
X = simulate(ftr, T, x0);
```

Funzioni Anonime: Utilizzo Tipico

La struttura tipica di un simulatore per sistemi dinamici lineari è quindi:

```
function es_heating()
    A = ... % Coefficienti della matrice
    b = ... % Termine noto (se c'è)
    x0 = ... % Stato iniziale
    T = ... % Istanti di tempo
    ftr = @(xc, t) f(xc, t, A, b) % Funzione anonima
    X = simulate(ftr, T, x0); % Simulazione
    ... % Disegno
end

function xf = f(xc, t, A, b)
    xf = A * xc' + b;
    xf = xf';
end
```

Elementi di Informatica e Applicazioni Numeriche T

Notazione Matriciale per Sistemi Tempo-Discreti Probabilistici

Esempio: Livello di Energia di un Elettrone

Un elettrone e può trovarsi a tre livelli di energia diversi

- e si trova inizialmente a livello 1 (il più basso)
- e è soggetto ad interazioni che possono cambiare il livello di energia
- Sappiamo che lo stato di e è ben descritto in termini di probabilità

Vogliamo determinare la probabilità che e sia in ciascun livello

Un possibile approccio:

- Modelliamo la particella come un sistema dinamico tempo-discreto
- Definiamo un modello probabilistico (stato = probabilità)
- Simuliamo ed analizziamo lo stato

Così otteniamo in modo esatto la distribuzione di probabilità

Esercizio: Livello di Energia di un Elettrone

Al livello 1:

- Con il 75% di probabilità, l'elettrone resta stabile
- Con il 25% di probabilità, l'elettrone si sposta al livello 2

Al livello 2:

- Con il 25% di probabilità, l'elettrone resta stabile
- Con il 50% di probabilità, l'elettrone si sposta al livello 1
- Con il 25% di probabilità, l'elettrone si sposta al livello 3

Al livello 3:

- Con il 25% di probabilità, l'elettrone resta stabile
- Con il 75% di probabilità, l'elettrone si sposta al livello 2

Stato e Funzione di Transizione

Lo stato è una distribuzione di probabilità:

Una scelta naturale è usare: $x = (x_1, x_2, x_3)$

- Dove x_i è la probabilità che e sia al livello i

Per quanto riguarda la funzione di transizione:

- Dobbiamo calcolare la probabilità di ciascun livello al passo $k + 1$ ...
- ...A partire dalle probabilità al passo k

$$
\begin{aligned}
x_1^{(k+1)} &= 0.75\, x_1^{(k)} + 0.5\, x_2^{(k)} \\
x_2^{(k+1)} &= 0.25\, x_1^{(k)} + 0.25\, x_2^{(k)} + 0.75\, x_3^{(k)} \\
x_3^{(k+1)} &= 0.25\, x_2^{(k)} + 0.25\, x_3^{(k)}
\end{aligned}
$$

Funzione di Transizione: Notazione Matriciale

Le equazioni sono tutte lineari!

Quindi possiamo usare la notazione matriciale:

$$\begin{pmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \\ x_3^{(k+1)} \end{pmatrix} = \begin{pmatrix} 0.75 & 0.5 & 0 \\ 0.25 & 0.25 & 0.75 \\ 0 & 0.25 & 0.25 \end{pmatrix} \begin{pmatrix} x_1^{(k)} \\ x_2^{(k)} \\ x_3^{(k)} \end{pmatrix}$$

La nuova rappresentazione è equivalente, ma molto più intuitiva:

- Infatti, l'elemento in posizione (i, j) nella matrice...
- ...È la probabilità di passare dallo stato j allo stato i

Risultato: una regola semplice per costruire modelli probabilistici

Modelli Probabilistici

Il sistema risultante è lineare, ed evolve secondo la legge:

$$x^{(k+1)} = Ax^{(k)}$$

- In questo caso A è detta anche matrice di transizione
- Il termine noto è tipicamente assente

Qualche osservazione:

- L'approccio funziona solo per stati discreti e finiti
- Non tutti i sistemi ammettono un modello probabilistico
- Non tutti i modelli probabilistici sono lineari... ma molti sì!
- A volte il comportamento può essere più complesso, ma lineare...
- ...Vediamo un paio di esempi nella prossima slide

Modelli Probabilistici

Configurazione di default (e.g. pagerank):

- Il sistema assume una configurazione di default con probabilità p...
- ...Ed effettua una transizione di stato con probabilità $1 - p$

$$x^{(k+1)} = (1 - p)\, A\, x^{(k)} + p\, b$$

- Il vettore b definisce le probabilità di default

Matrici di transizione alternative:

- Con probabilità p la transizione è guidata da una matrice A...
- ...E con probabilità $1 - p$ da un'altra matrice B

$$x^{(k+1)} = (p\, A + (1 - p)\, B)\, x^{(k)}$$

Elementi di Informatica e Applicazioni Numeriche T

Esercizio: Previsioni del Tempo

Esercizio: Previsioni del Tempo

Supponiamo che (sommariamente) valgano le regole seguenti:

- Se un giorno c'è bel tempo
  - Il giorno successivo sarà bello con il 90% di probabilità
  - Il giorno successivo sarà brutto con il 10% di probabilità
- Se un giorno c'è brutto tempo
  - Il giorno successivo sarà bello con il 50% di probabilità
  - Il giorno successivo sarà brutto con il 50% di probabilità

Si sviluppi un modello per prevedere il tempo i giorni successivi

- Si utilizzi un approccio probabilistico
- Si assume il giorno iniziale il tempo sia brutto
- Come evolve la probabilità di avere bel/brutto tempo?

Esercizio: Previsioni del Tempo

Si parta dal file `es_weather.m` nello start-kit

Si definisca la funzione di transizione:

```
function xf = f(xc, t)
```

- Dove **`xc`** è lo stato corrente e **`t`** il tempo corrente

Si sviluppi il codice di simulazione nella funzione principale:

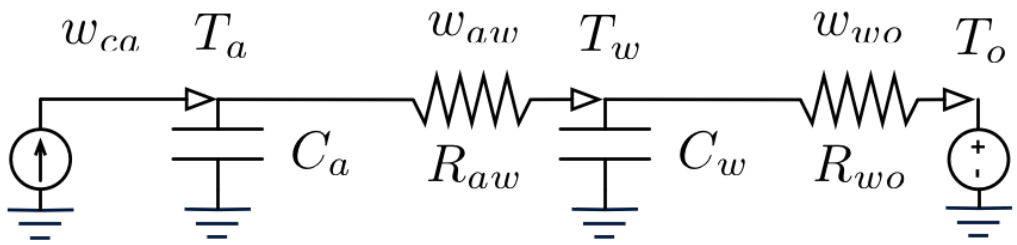
- Si sfrutti la funzione **`simulate`**, disponibile nello start-kit
- Si disegni l'andamento delle due probabilità (bel/brutto tempo)
- Si osservi come variano se il primo giorno il tempo è bello

Elementi di Informatica e Applicazioni Numeriche T

Esercizio: Riscaldamento di una Stanza

Esercizio: Riscaldamento di una Stanza

Si consideri il sistema discreto dell'esempio sul riscaldamento



La soluzione è disponibile nello start-kit, nel file **`es_heating.m`**

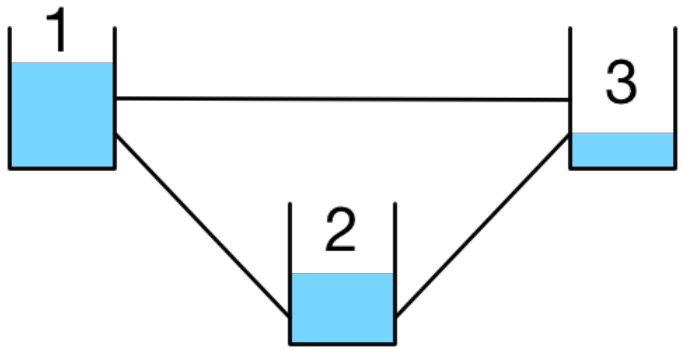
- Si copi il codice in un nuovo file **`es_heating2.m`**
- Si assuma che il convettore venga acceso solo per la prima ora
- Si disegni l'andamento di T_a e T_w per due ore
- Cosa succede cambiando il tempo di simulazione/accensione?
- Cosa succede cambiando il flusso di potenza in ingresso?

Elementi di Informatica e Applicazioni Numeriche T

Esercizio: Serbatoi Comunicanti

Esercizio: Serbatoi Comunicanti

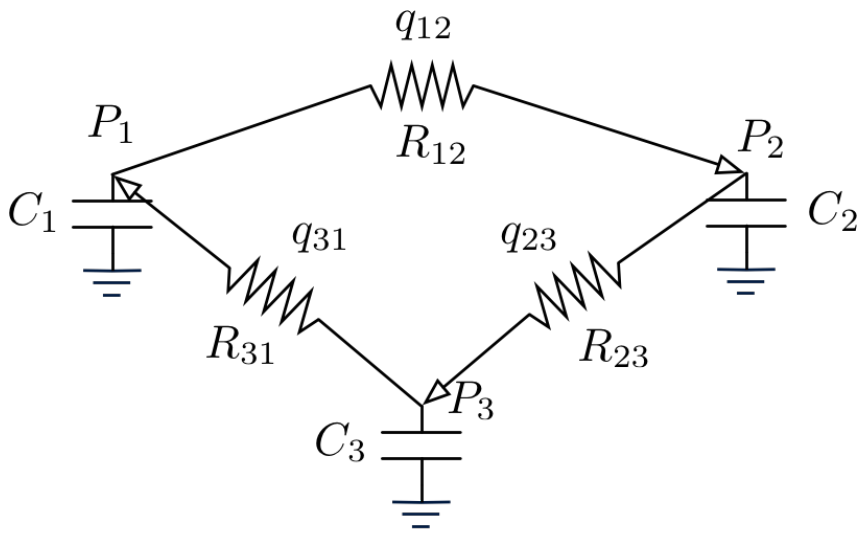
Tre serbatoi contenenti acqua comunicano attraverso condotte



- Conosciamo il livello e la superficie di ogni serbatoio
- Ciò è sufficiente per determinarne la pressione...
- ...Che è analoga alla temperatura nell'esempio sul riscaldamento
- I serbatoi si comportano come capacità
- Le condotte come resistenze

Esercizio: Serbatoi Comunicanti

Quindi possiamo modellare il sistema come un circuito RC



- Alle condotte viene assegnata una direzione convenzionale
- Tutti i valori di C ed R sono noti
- NOTA: il modello è approssimato!
- In realtà le cose sono più complesse (lo vedrete in fluidodinamica)

Esercizio: Serbatoi Comunicanti

Partite dal file `es_tubes.m` nello start-kit

Definite la funzione di transizione:

```
function xf = f(xc, t)
```

Definite il codice di simulazione nella funzione principale:

- Sfruttate la funzione **`simulate`** dello start-kit
- I valori iniziali della pressioni sono noti
- Assumete che le portate iniziali delle condotte siano 0
- Visualizzate l'andamento delle tre pressioni nel tempo
- Cosa succede? Sapete renderne ragione intuitivamente?
- Notate qualcosa di strano all'inizio? Sapete darne ragione?

Elementi di Informatica e Applicazioni Numeriche T

Funzioni Polinomiali

Funzioni Polinomiali

Nel file di funzione `es_polynomials.m` (dallo start-kit), definire:

```
function y = my_polyval(p, x)
```

- Che replichi il comportamento di **`polyval`**...
- ...Per **`x`** scalare (si trascuri il caso con **`x`** vettore)

Si verifichi la correttezza nella funzione principale

- Si utilizzi la funzione **`polyval`** per avere un riscontro

Funzioni Polinomiali

Nel file di funzione `es_polynomials.m` (dallo start-kit), definire:

```
function y = my_polyder(p)
```

- Che replichi il comportamento di **`polyder`**

Si verifichi la correttezza nella funzione principale

- Si utilizzi la funzione **`polyder`** per avere un riscontro

Funzioni Polinomiali

Nel file di funzione `es_polynomials.m` (dallo start-kit), definire:

```
function y = my_polyint(p)
```

- Che replichi il comportamento di **`polyint`**

Si verifichi la correttezza nella funzione principale

- Si utilizzi la funzione **`polyint`** per avere un riscontro