Metodologia di modulazione della potenza del attuatore in funzione del costo dell’energia

1. Contesto e obiettivo

L’obiettivo di questa analisi è definire una **logica di regolazione della potenza (o velocità)** di un attuatore in funzione del **costo orario dell’energia elettrica**, utilizzando i dati di prezzo disponibili tramite ENTSO-E.

Nel nostro caso l’energia non viene venduta ma **acquistata**: di conseguenza la variabile economica principale è il **costo dell’energia**.
La logica di modulazione ha quindi lo scopo di **ridurre i consumi nelle fasce orarie più costose**, compatibilmente con i vincoli di processo, ottenendo una **riduzione percentuale della potenza** rispetto al valore massimo impostato nel attuatore.

---

2. Principio di funzionamento

L’attuatore dispone di un **valore massimo impostato** (es. velocità o potenza nominale) e può essere modulato tra 0% e 100% di questo valore.
Vogliamo definire una funzione che, dato il **prezzo orario dell’energia** `p(t)`, fornisca:

- una **riduzione percentuale** `Riduzione%(t)` compresa tra **0%** (nessuna riduzione) e **100%** (riduzione massima impostata),
- e, di conseguenza, un **setpoint di motore** espresso in percentuale del massimo:

$$Setpoint\_attuatore\%(t) = 100\% - Riduzione\%(t)$$

Setpoint_attuatore%(t)=100%−Riduzione%(t)

In pratica:

- quando il prezzo è **basso**, l’attuatore lavora vicino al 100% del valore massimo impostato;
- quando il prezzo è **alto**, l’attuatore viene **progressivamente “strozzato”**, fino ad arrivare a una riduzione massima del 100%.

---

3. Parametri di riferimento

Per la definizione della logica, sono stati scelti i seguenti parametri:

- **Prezzo minimo considerato (p_min)**: 75 €/MWh
  Sotto questo valore, il prezzo è ritenuto “accettabile” e **non viene applicata riduzione**.

- **Prezzo massimo considerato (p_max)**: 274,1 €/MWh
  Sopra questo valore, il prezzo è considerato “molto elevato” e viene applicata la **riduzione massima**.

- **Riduzione minima**: 0%
  → per prezzi ≤ 75 €/MWh l’attuatore lavora al **100%** della potenza impostata.

- **Riduzione massima**: 100%
  → per prezzi ≥ 274,1 €/MWh i l'attuatore lavora al **10%** della potenza impostata.

Questi valori sono configurabili in base alle esigenze operative, ai vincoli di processo e alla sensibilità economica dell'azienda.

---

4. Approccio non lineare

In una prima versione, la riduzione può essere definita come funzione **lineare** del prezzo tra `p_min` e `p_max`.
Tuttavia, si è scelto di adottare un **approccio non lineare** per ottenere un comportamento più aderente alle esigenze operative:

- **riduzione contenuta** per prezzi medi, in modo da non penalizzare eccessivamente il processo in condizioni "normali";
- **riduzione più aggressiva** solo quando i prezzi si avvicinano alla fascia alta.

Questo è ottenuto:

1. **Normalizzando** il prezzo tra `p_min` e `p_max` in un intervallo [0,1]:

$$\beta(t) = \frac{p(t) - p_{min}}{p_{max} - p_{min}}$$

2. Applicando una **trasformazione non lineare** (potenza):

$$\beta_{nl}(t) = (\beta(t))^n$$

con `n > 1` (ad esempio **n = 2**).
In questo modo, per valori intermedi di prezzo, `β_nl` cresce più lentamente di `β`, mentre per prezzi vicini a `p_max` cresce rapidamente.

3. Scalando il risultato sul range di riduzione 0%–100%:

$$\text{Riduzione\%}(t) = \beta_{nl}(t) \cdot 100$$

con saturazione ai limiti inferiori e superiori (`p ≤ p_min` → 0%, `p ≥ p_max` → 100%).

---

5. Vantaggi della soluzione proposta

La metodologia proposta presenta alcuni vantaggi pratici:

- **Semplicità di implementazione**
  La formula è direttamente implementabile in strumenti come **Excel**, sistemi di supervisione (SCADA) o logiche di controllo (PLC), senza richiedere modelli complessi di ottimizzazione.

- **Trasparenza**
  I parametri chiave (prezzo minimo, prezzo massimo, riduzione massima, grado di non linearità) sono **espliciti e facilmente regolabili**, consentendo di adattare la strategia alle esigenze aziendali.

- **Gradualità del controllo**
  La versione non lineare evita salti bruschi e permette una **transizione dolce** dalla piena potenza a valori ridotti, favorendo la stabilità del processo.

- **Orientamento al risparmio energetico**
  In condizioni di prezzi elevati, la riduzione della potenza permette di **contenere i costi energetici**, lasciando invece il motore libero di operare quasi al massimo quando l'energia è relativamente economica.

---

6. Utilizzi pratici del modello

Una volta definita la funzione di riduzione:

- può essere applicata in un **report storico**, utilizzando i prezzi ENTSO-E passati per stimare:
  - quanto si sarebbe ridotto il consumo dell'attuatore,
  - quale sarebbe stato il potenziale risparmio economico;
- può essere integrata in **logiche di controllo in tempo reale**, dove il prezzo dell'energia (o un segnale derivato, come un indice tariffario giornaliero) determina dinamicamente il setpoint del motore;
- può servire come base per una futura **ottimizzazione multi-variabile**, considerando non solo il prezzo dell'energia, ma anche:
  - priorità di produzione,
  - limiti minimi di processo,
  - vincoli di qualità del servizio.

---

7. Conclusioni

La relazione presenta una **metodologia strutturata ma semplice** per collegare in modo diretto il **costo dell'energia** alla **potenza effettiva di un motore**, espressa come riduzione percentuale rispetto al valore massimo impostato.

L'utilizzo di una funzione **non lineare** consente di ottenere una regolazione economicamente sensata e tecnicamente gestibile, che protegge il processo nelle condizioni standard e reagisce in modo deciso solo in presenza di prezzi eccezionalmente elevati.

Nelle sezioni successive del report potranno essere inseriti:

- gli **esempi numerici**,
- le **formule Excel** di dettaglio,
- e una **tabella comparativa** tra approccio lineare e non lineare, così da evidenziare in modo quantitativo il comportamento della logica proposta.

Qui le valutazioni in diversi punti sui calcoli, supponendo per praticità che l'attuatore sia un motore.

1. Idea di base: normalizzare il prezzo, poi tradurlo in riduzione

Il concetto chiave:

1. **Prendi i prezzi** `p(t)` (€/MWh, €/kWh, non importa l'unità, basta che sia coerente).
2. **Definisci un intervallo di prezzi "rilevante"**:
   - prezzi molto bassi → riduzione nulla o minima (puoi usare il motore a piena potenza),
   - prezzi molto alti → riduzione massima (limiti il motore per risparmiare).
3. **Scala il prezzo** in un valore tra 0 e 1, e poi lo converti in **percentuale di riduzione**.

Struttura generale:

- calcolo di un indice "caro/economico" del prezzo: `β(t) ∈ [0,1]`
- conversione in **riduzione percentuale**:

$$\text{Riduzione\%}(t) = \beta(t) \cdot 100\%$$

e quindi:

$$\text{Potenza\_effettiva}(t) = (1 - \beta(t)) \cdot P_{max}$$

(dove `P_max` è il valore massimo impostato del motore).

2. Metodo pratico 1 – uso dei quantili (robusto e semplice)

Per non dipendere né dal minimo assoluto né dal massimo assoluto (che a volte sono “outlier”), il metodo robusto è:

1. Prendi tutta la serie di prezzi per il periodo che ti interessa (es. 1 mese, 1 anno).
2. Calcola due *valori caratteristici* del prezzo:
   - `p_low` = prezzo “basso tipico”
     → ad es. il **20° percentile** (il valore sotto il quale cade il 20% dei prezzi)
   - `p_high` = prezzo “alto tipico”
     → ad es. l’**80° percentile** (valore sopra il quale cade il 20% dei prezzi)

   Questi li ottieni facilmente con funzioni tipo:

   - in Excel: `=PERCENTILE.INC(intervallo_prezzi; 0,2)` e `=PERCENTILE.INC(...; 0,8)`
3. Definisci l’indice `β(t)` che misura quanto il prezzo è “alto” tra `p_low` e `p_high`:

$$\beta(t) = \begin{cases} 0 & p(t) \le p_{low} \\ \frac{p(t) - p_{low}}{p_{high} - p_{low}} & p_{low} < p(t) < p_{high} \\ 1 & p(t) \ge p_{high} \end{cases}$$

4. La **riduzione percentuale** da applicare al motore è:

$$\text{Riduzione\%}(t) = \beta(t) \cdot 100\%$$

5. La **potenza/velocità di setpoint** del motore (in percentuale del massimo) diventa:

$$\text{Setpoint\_motore\%}(t) = 100\% - \text{Riduzione\%}(t)$$

cioè:

- se `p(t)` è basso (≤ `p_low`) → `β = 0`, riduzione 0%, motore al 100%,
- se `p(t)` è medio → riduzione intermedia,
- se `p(t)` è alto (≥ `p_high`) → `β = 1`, riduzione 100% (o alla soglia minima che decidi tu).

Come evitare di fermare del tutto il motore

Se non si vuole mai scendere sotto, ad esempio, il **50%** di potenza (per ragioni di processo), basta introdurre un limite minimo `α_min` sulla potenza residua:

- fissa `α_min` = 0.5 (50%),
- potenza effettiva:

$$\text{Setpoint\_motore\%}(t) = \alpha_{min} \cdot 100\% + (1 - \alpha_{min}) \cdot (1 - \beta(t)) \cdot 100\%$$

e di conseguenza:

$$\text{Riduzione\%}(t) = 100\% - \text{Setpoint\_motore\%}(t)$$

È solo una piccola modifica alla scala.

3. Metodo pratico 2 – soglie “fisse” di prezzo

Qualcosa di ancora più intuitivo, si può scegliere **soglie di prezzo fisse** (non basate su quantili, ma sul personale senso di costo accettabile).

Esempio:

- `p_accettabile` = prezzo sotto cui l’energia è “ok” → nessuna riduzione
- `p_troppo_caro` = prezzo sopra cui vuoi la riduzione massima concepita (es. 70% di riduzione)

Allora:

1. Definisci un coefficiente normalizzato:

$$\beta(t) = \begin{cases} 0 & p(t) \le p_{accettabile} \\ \frac{p(t) - p_{accettabile}}{p_{troppo\_caro} - p_{accettabile}} & p_{accettabile} < p(t) < p_{troppo\_caro} \\ 1 & p(t) \ge p_{troppo\_caro} \end{cases}$$

2. Decidi una **riduzione massima consentita** `R_max` (es. 70%):

$$\text{Riduzione\%}(t) = \beta(t) \cdot R_{max}$$

3. Setpoint motore (in % del massimo):

$$\text{Setpoint\_motore\%}(t) = 100\% - \text{Riduzione\%}(t)$$

Esempio numerico:

- `p_accettabile` = 80 €/MWh
- `p_troppo_caro` = 200 €/MWh
- `R_max` = 60%

Se `p(t) = 140`:

$$\beta = \frac{140 - 80}{200 - 80} = \frac{60}{120} = 0{,}5$$

→ `Riduzione% = 0,5 × 60% = 30%`
→ `Setpoint_motore% = 70% del massimo`.

4. Collegare alla variazione rispetto al “valore massimo impostato”

Il dispositivo accetta il:

> valore di riduzione percentuale rispetto al valore massimo impostato sul motore

Nella pratica:

- supponiamo che il massimo impostato del motore corrisponda a un certo valore
  `P_set_max`
  (che può essere sia la potenza che la velocità nominale usata nel tuo processo).

Allora, in forma generale:

1. Calcoli `β(t)` (indice tra 0 e 1 che cresce con il prezzo).
2. Definisci la **percentuale di riduzione**:

$$\text{Riduzione\%}(t) = f(\beta(t)) \cdot 100\%$$

3. La **potenza/velocità effettiva** è:

$$P_{eff}(t) = P_{set\_max} \cdot (1 - f(\beta(t)))$$

Nei metodi di sopra, `f(β)` è spesso semplicemente `β` o `β × (R_max/100)`.

5. Esempio concreto “da Excel”

Diciamo che se hai una colonna:

- `A` = orario
- `B` = prezzo energia `p(t)`

Nells colonna `C` la **riduzione %** rispetto al massimo motore basata su quantili del prezzo (Metodo 1).

Passiamo:

1. Su un intervallo di dati (es. B2:B25 per un giorno, B2:B8737 per un anno):
   - in qualche cella (ad es. `F1`):
     `=PERCENTILE.INC($B$2:$B$25; 0,2)` → `p_low`
   - in `F2`:
     `=PERCENTILE.INC($B$2:$B$25; 0,8)` → `p_high`
2. In `C2` (riduzione fra 0% e 100%):

```
=IF(B2 <= $F$1;
     0;
     IF(B2 >= $F$2;
        100;
        (B2 - $F$1)/($F$2 - $F$1) * 100
     )
  )
```

3. In `D2` (setpoint motore in % del massimo):

```
=100 - C2
```

Possiamo aggiungere un limite minimo (es. non scendere mai sotto 50%):

```
=MAX(50; 100 - C2)
```

6. Implementazione più “morbida”: funzione non lineare

Invece di una funzione lineare (che cresce a scaletta), possiamo usare una funzione *convessa* o tipo potenza, per essere più aggressivo solo quando i prezzi sono davvero alti.

Esempio:

$$\text{Riduzione\%}(t) = (\beta(t))^n \cdot 100\%$$

- se `n > 1` (es. 2 o 3), la riduzione cresce lentamente per prezzi medi e fortemente per prezzi altissimi.

In Excel:

```
=POWER( beta; n ) * 100
```

dove `beta` è il valore calcolato come normalizzazione tra `p_low` e `p_high`.

7. Riepilogo operativo

- Acquistando energia, non producendola: non ci serve parlare di ricavi, ma solo di **costo energia**.
- Si definisce **un intervallo di prezzi** tra “ok” e “troppo caro” (via quantili o soglie).
- **Normalizza** il prezzo in un indice `β(t) ∈ [0,1]`.
- Trasforma `β(t)` in una **riduzione %** rispetto al massimo del motore:
    - lineare → semplice e intuitivo;
    - non lineare (potenza o sigmoide) → più flessibile.
- La **potenza/velocità del motore** è poi:
  `Setpoint% = 100% – Riduzione%`.

Esempi proposti due versioni:

1. **Formula Excel esatta** (lineare tra 75 e 274,1 €/MWh)
2. **Pseudo-codice chiaro** con gli stessi parametri

Parametri (come esempio richiesto):

- `p_min` (prezzo minimo) = **75 €/MWh**
- `p_max` (prezzo massimo) = **274,1 €/MWh**
- `Riduzione_min` = **0%**
- `Riduzione_max` = **90%**

---

1. Formula Excel esatta

Supponiamo che:

- nella cella **B2** ci sia il **prezzo orario** dell'energia (in €/MWh).

1.1. Riduzione percentuale (0%–90%)

```
=IF(B2 <= 75;
     0;
     IF(B2 >= 274,1;
        90;
        (B2 - 75) / (274,1 - 75) * 90
     )
  )
```

Questa formula fa esattamente:

- se `prezzo ≤ 75` → **Riduzione% = 0**
- se `prezzo ≥ 274,1` → **Riduzione% = 90**
- se `75 < prezzo < 274,1` → Riduzione% cresce **linearmente** da 0 a 90

1.2. Setpoint motore in % del massimo

Se vogliamo anche il **setpoint del motore** (quanta potenza/velocità usare rispetto al massimo), lo calcoliamo così:

```
=100 - (
    IF(B2 <= 75;
         0;
         IF(B2 >= 274,1;
            90;
            (B2 - 75) / (274,1 - 75) * 90
         )
      )
)
```

oppure, se la riduzione è già calcolata in `C2` con la formula di prima:

```
=100 - C2
```

- **Riduzione% = 0** → Setpoint = 100% (motore al massimo impostato)
- **Riduzione% = 90** → Setpoint = 10% (motore al 10% del massimo)

---

2. Pseudo-codice (stessa logica, stesso comportamento)

Usando nomi chiari:

- `price` = prezzo corrente [€/MWh]
- `p_min` = 75
- `p_max` = 274.1
- `R_min` = 0 (riduzione minima in %)
- `R_max` = 90 (riduzione massima in %)

2.1. Calcolo della riduzione percentuale

```
p_min = 75.0       // €/MWh
p_max = 274.1      // €/MWh
R_min = 0.0        // % riduzione minima
R_max = 90.0       // % riduzione massima

function calcola_riduzione_percentuale(price):

    if price <= p_min:
        riduzione = R_min          // 0%

    else if price >= p_max:
        riduzione = R_max          // 90%

    else:
        // normalizzazione lineare del prezzo tra p_min e p_max
        beta = (price - p_min) / (p_max - p_min)  // beta va da 0 a 1

        // riduzione lineare tra 0% e 90%
        riduzione = R_min + beta * (R_max - R_min)

    return riduzione   // in percentuale (%)
```

2.2. Calcolo del setpoint motore in % del massimo

```
function calcola_setpoint_motore(price):

    riduzione = calcola_riduzione_percentuale(price)

    // setpoint in percentuale rispetto al massimo
    setpoint_percent = 100.0 - riduzione

    return setpoint_percent
```

Se, ad esempio:

- `price = 75` → `riduzione = 0%`, `setpoint = 100%`
- `price = 274,1` → `riduzione = 90%`, `setpoint = 10%`
- `price` a metà tra 75 e 274,1 → riduzione ≈ 45%, setpoint ≈ 55%

Qui implementiamo una versione **non lineare** che:

- resta con:
  - prezzo minimo = **75 €/MWh**
  - prezzo massimo = **274,1 €/MWh**
  - riduzione minima = **0%**
  - riduzione massima = **90%**
- ma fa crescere la riduzione in modo **più dolce sui prezzi medi** e **più aggressivo sui prezzi alti**.

Useremo una funzione di tipo **potenza** (convessa), molto semplice da mettere in Excel e in codice.

---

1. Idea della versione non lineare

Passi logici:

1. **Normalizza** il prezzo nell'intervallo [0,1]:

$$\beta = \frac{\text{prezzo} - p_{\min}}{p_{\max} - p_{\min}}$$

2. "Pieghi" la curva con una potenza:

$$\beta_{nl} = \beta^{n}$$

- se `n > 1`, la curva è:
  - lenta all'inizio (poco effetto per prezzi medi),
  - molto rapida verso l'alto per prezzi alti.

3. Usi `β_nl` per scalare la riduzione tra 0% e 90%:

$$\text{Riduzione\%} = \beta_{nl} \cdot 90$$

con saturazione ai bordi (sotto p_min e sopra p_max).

Un valore tipico: **n = 2** (curva morbida ma non estrema).
Se vuoi più aggressività sui prezzi alti, puoi provare **n = 3**.

---

2. Formula Excel non lineare (con n = 2)

Supponiamo sempre:

- **B2** = prezzo corrente (€/MWh).

Parametro non linearità: `n = 2` (puoi cambiarlo).

2.1. Riduzione% non lineare (0–90%)

```
=IF(B2 <= 75;
    0;
    IF(B2 >= 274,1;
       90;
```

```
        POWER( (B2 - 75) / (274,1 - 75) ; 2 ) * 90
     )
  )
```

Spiegazione:

- `((B2 - 75) / (274,1 - 75))` → `β` lineare ∈ (0,1)
- `POWER(β; 2)` → `β² = β_nl` (non lineare)
- `β² * 90` → riduzione in percentuale tra 0 e 90%

Comportamento qualitativo:

- vicino a 75 € → riduzione molto piccola (curva piatta)
- a metà tra 75 e 274,1 → riduzione **minore** rispetto al caso lineare
- vicino a 274,1 € → riduzione che sale rapidamente verso il 90%

2.2. Setpoint motore% (non lineare)

Se vuoi direttamente il **setpoint del motore** in % del massimo:

```
=100 -
 IF(B2 <= 75;
     0;
     IF(B2 >= 274,1;
        90;
        POWER( (B2 - 75) / (274,1 - 75) ; 2 ) * 90
     )
  )
```

oppure, se in `C2` hai già la riduzione%:

```
=100 - C2
```

---

3. Variante Excel con esponente personalizzabile (n come parametro)

Se vuoi poter cambiare facilmente il grado di non linearità, puoi mettere `n` in una cella, per esempio:

- `F1` = 2 (esponente n)

Allora la formula diventa:

```
=IF(B2 <= 75;
     0;
     IF(B2 >= 274,1;
        90;
        POWER( (B2 - 75) / (274,1 - 75) ; $F$1 ) * 90
     )
  )
```

- `F1 = 1` → comportamento **lineare**
- `F1 = 2` → curva **moderatamente non lineare**
- `F1 = 3` → ancora più “aggressiva” sui prezzi alti

4. Pseudo-codice versione non lineare

Parametri:

```
p_min = 75.0          // €/MWh
p_max = 274.1         // €/MWh
R_min = 0.0           // % riduzione minima
R_max = 90.0          // % riduzione massima
n    = 2.0            // esponente per la non linearità (puoi cambiare)
```

4.1. Riduzione% non lineare

```
function calcola_riduzione_percentuale_nonlineare(price):

    if price <= p_min:
        riduzione = R_min          // 0%

    else if price >= p_max:
        riduzione = R_max          // 90%

    else:
        // 1) normalizzazione lineare del prezzo tra p_min e p_max
        beta = (price - p_min) / (p_max - p_min)   // beta in (0,1)

        // 2) applica la non linearità (curva convessa)
        beta_nl = beta ^ n       // n > 1 → curva lenta per prezzi medi

        // 3) scala tra R_min e R_max
        riduzione = R_min + beta_nl * (R_max - R_min)

    return riduzione   // in percentuale (%)
```

4.2. Setpoint motore% non lineare

```
function calcola_setpoint_motore_nonlineare(price):

    riduzione = calcola_riduzione_percentuale_nonlineare(price)

    // setpoint in percentuale rispetto al massimo del motore
    setpoint_percent = 100.0 - riduzione

    return setpoint_percent
```

---

5. Suggerimenti pratici

- Parti con **n = 2**: è un buon compromesso tra semplicità e comportamento “intelligente”.
- Se vedi che il motore riduce troppo poco agli alti prezzi:
    - aumenta `n` (es. 2,5 o 3).
- Se vedi che è troppo “aggressivo” già a prezzi medi:
    - riduci `n` (es. 1,5).

Appendice:

Procedura selezionata:

1. **Non voglio fissare a mano p_min e p_max.**
   Voglio che siano “automaticamente” adattati ai prezzi reali del periodo →
   quindi uso i **quantili** per definire:
   - `p_low` = prezzo “basso tipico” (es. 20° percentile)
   - `p_high` = prezzo “alto tipico” (es. 80° percentile)
2. **Non voglio una risposta semplicemente lineare.**
   Voglio che la riduzione sia **più dolce sui prezzi medi** e **più forte solo sui prezzi davvero alti →**
   quindi applico l’**approccio non lineare** (es. elevare a potenza `n > 1`).

Questo mi dà un algoritmo:

- **adattivo** (si adatta ai dati di prezzo di ogni periodo),
- **robusto** (non dipende dai min/max estremi, ma dalla distribuzione reale dei prezzi),
- **morbido e controllabile** (tramite l’esponente `n` e la riduzione massima).

Perfetto per un’applicazione reale.

Struttura del metodo che stò scegliendo

Per chiarezza, riassumo il combinato “quantili + non lineare” in 4 step logici:

1. **Calcolo dei quantili sui prezzi storici / del periodo**
    - `p_low` = quantile basso (es. 20° percentile → 0,2)
    - `p_high` = quantile alto (es. 80° percentile → 0,8)
2. **Normalizzazione del prezzo sull’intervallo [0,1]**

$$\beta(t) = \begin{cases} 0 & p(t) \le p_{low} \\ \frac{p(t) - p_{low}}{p_{high} - p_{low}} & p_{low} < p(t) < p_{high} \\ 1 & p(t) \ge p_{high} \end{cases}$$

3. **Curva non lineare (esponente n)**

$$\beta_{nl}(t) = (\beta(t))^n \quad \text{con } n > 1$$

4. **Riduzione% e setpoint motore**

Se `R_max` è la riduzione massima (es. 90%):

$$\text{Riduzione\%}(t) = \beta_{nl}(t) \cdot R_{max}$$

$$\text{Setpoint\_motore\%}(t) = 100\% - \text{Riduzione\%}(t)$$

Riassumendo

1. I prezzi dell’energia non vengono confrontati con una soglia fissa, ma con due valori dinamici (`p_low`, `p_high`) determinati dai quantili (ad esempio 20° e 80° percentile) della serie prezzi ENTSO-E del periodo considerato.
2. In questo modo, la logica di controllo si adatta automaticamente al contesto di mercato (prezzi più o meno alti rispetto alla loro distribuzione tipica).
3. Tra `p_low` e `p_high` il prezzo viene normalizzato in un indice [0,1], che rappresenta “quanto il prezzo è alto” rispetto alla fascia tipica.
4. L’indice normalizzato viene poi trasformato tramite una funzione non lineare (esponente > 1), che rende la riduzione di potenza contenuta nelle fasce di prezzo medio, ma molto più incisiva quando il prezzo entra nella fascia alta.
5. Il risultato finale è una riduzione percentuale tra 0% e una riduzione massima configurabile (es. 90%), che viene sottratta al 100% per ottenere il setpoint effettivo del motore.

Per la definizione della strategia di modulazione della potenza del motore in funzione del costo dell'energia si è adottato un approccio combinato basato su **quantili di prezzo** e **funzione non lineare**.
In primo luogo, i prezzi orari dell'energia vengono analizzati sul periodo di riferimento per individuare due soglie dinamiche: il 20° percentile (`p_low`), che rappresenta una fascia di prezzo "bassa tipica", e l'80° percentile (`p_high`), che rappresenta una fascia di prezzo "alta tipica". I prezzi inferiori a `p_low` non generano riduzione (0%), mentre i prezzi superiori a `p_high` portano alla riduzione massima configurata (90%).
Per i prezzi compresi tra `p_low` e `p_high`, si calcola un indice normalizzato tra 0 e 1 e lo si trasforma tramite una funzione non lineare (potenza con esponente n = 2). Questo consente di ottenere una riduzione contenuta per prezzi medi e più aggressiva solo quando il prezzo dell'energia si avvicina alla fascia alta. Il risultato finale è una **riduzione percentuale variabile tra 0% e 90%**, da cui si ricava il **setpoint del motore** in percentuale del massimo (100% – Riduzione%).

Allora usiamo:

- `p_low` = 20° percentile dei prezzi
- `p_high` = 80° percentile dei prezzi
- `R_max` = 90% (riduzione massima)
- `n` = 2 (non linearità moderata)

1. Formule Excel complete (quantili + non lineare)

1.1. Dati di base

Supponiamo:

- Colonna **B** = prezzi orari dell'energia, da `B2` a `B8737` (un anno).
  (Adatta l'intervallo all'effettivo numero di righe che hai.)

1.2. Calcolo dei quantili (20° e 80° percentile)

In una zona separata del foglio, ad esempio:

- in **F1** calcoliamo `p_low` = 20° percentile:

```
=QUANTILE.INC($B$2:$B$8737; 0,2)
```

(in Excel in inglese: `=PERCENTILE.INC($B$2:$B$8737, 0.2)`)

- in **F2** calcoliamo `p_high` = 80° percentile:

```
=QUANTILE.INC($B$2:$B$8737; 0,8)
```

(in inglese: `=PERCENTILE.INC($B$2:$B$8737, 0.8)`)

- in **F3** mettiamo l'esponente `n` (non linearità):

```
2
```

- in **F4** mettiamo la riduzione massima `R_max`:

```
90
```

Riassunto celle:

Cella	Significato	Contenuto
F1	p_low (20° perc.)	`=PERCENTILE.INC($B$2:$B$8737;0,2)`
F2	p_high (80° perc.)	`=PERCENTILE.INC($B$2:$B$8737;0,8)`
F3	n (esponente)	`2`
F4	R_max (%)	`90`

1.3. Riduzione percentuale (0%–R_max) con quantili + non lineare

Ora calcoliamo la **Riduzione%** per ogni ora sulla base del prezzo in `B2`.

In **C2**:

```
=LET(
    prezzo; B2;
    p_low;  $F$1;
    p_high; $F$2;
    n;      $F$3;
```

```
    R_max;  $F$4;

    beta; IF(
             prezzo <= p_low;
             0;
             IF(
                prezzo >= p_high;
                1;
                (prezzo - p_low) / (p_high - p_low)
             )
          );

    beta_nl; POWER(beta; n);

    riduzione; beta_nl * R_max;

    riduzione
)
```

Se non si dispone della funzione `LET`, usa una versione “classica”:

```
=IF(B2 <= $F$1;
     0;
     IF(B2 >= $F$2;
        $F$4;
        POWER( (B2 - $F$1) / ($F$2 - $F$1) ; $F$3 ) * $F$4
     )
  )
```

Questa formula fa:

- se `prezzo ≤ p_low (F1)` → Riduzione% = 0
- se `prezzo ≥ p_high (F2)` → Riduzione% = R_max (F4 = 90)
- se prezzo intermedio →
    1. normalizza tra `p_low` e `p_high`,
    2. applica potenza `n` (F3 = 2),
    3. scala tra 0 e `R_max`.

Copiamo la formula di `C2` verso il basso per tutte le righe.

1.4. Setpoint motore in % del massimo (quantili + non lineare)

Se vuoi anche il **setpoint del motore** in % del massimo (100% – Riduzione%), in **D2**:

```
=100 - C2
```

e trascini verso il basso.

---

2. Pseudo-codice (quantili + non lineare)

2.1. Parametri

```
// Input: array di prezzi price[1..N]

// Calcolo quantili sul periodo considerato (offline o a inizio giorno/mese)
p_low  = quantile(price, 0.20)   // 20° percentile
p_high = quantile(price, 0.80)   // 80° percentile

R_max = 90.0     // riduzione massima (%)
n     = 2.0      // esponente per la non linearità
```

2.2. Funzione per la riduzione% oraria

```
function calcola_riduzione_percentuale(price_t):

    if price_t <= p_low:
        return 0.0      // Riduzione% = 0%

    else if price_t >= p_high:
        return R_max    // Riduzione% = 90%

    else:
        // 1. normalizzazione lineare tra p_low e p_high
        beta = (price_t - p_low) / (p_high - p_low)     // beta in (0,1)

        // 2. curva non lineare (esponente n)
        beta_nl = beta ^ n

        // 3. scala tra 0% e R_max
        riduzione = beta_nl * R_max

        return riduzione
```

2.3. Funzione per il setpoint motore%

```
function calcola_setpoint_motore(price_t):

    riduzione = calcola_riduzione_percentuale(price_t)

    setpoint_percent = 100.0 - riduzione

    return setpoint_percent
```
