RdS
RICERCA DI SISTEMA

ENEA
AGENZIA NAZIONALE PER LE NUOVE TECNOLOGIE, L'ENERGIA E LO SVILUPPO ECONOMICO SOSTENIBILE

MINISTERO DELL'AMBIENTE E DELLA SICUREZZA ENERGETICA

Ricerca di Sistema elettrico

Definizione del modello per la valutazione dei sistemi di accumulo e delle FER per il supporto dell'adeguatezza di sistemi elettrici decarbonizzati

Roberto Ciavarella, Antonio Ricca, Maria Valenti

ENEA
AGENZIA NAZIONALE PER LE NUOVE TECNOLOGIE, L'ENERGIA E LO SVILUPPO ECONOMICO SOSTENIBILE

RdS_PTR 22-24_PR 2.3_LA1.4_156

Definizione del modello per la valutazione dei sistemi di accumulo e delle FER per il supporto dell'adeguatezza di sistemi elettrici decarbonizzati.

R. Ciavarella (ENEA), A. Ricca (ENEA), M. Valenti (ENEA)

Dicembre 2024

Report Ricerca di Sistema Elettrico

Accordo di Programma Ministero dell'Ambiente e della Sicurezza Energetica -ENEA Piano Triennale di Realizzazione 2022-2024

Obiettivo 2: Digitalizzazione ed evoluzione delle reti

Progetto 2.3: Evoluzione, pianificazione, gestione ed esercizio delle reti elettriche

Linea di attività: 1.4

Responsabile del Progetto: Maria Valenti

Responsabile Linea di Attività: Roberto Ciavarella

Mese inizio previsto: luglio 2023

Mese inizio effettivo: luglio 2023

Mese fine previsto: dicembre 2024

Mese fine effettivo: dicembre 2024

Sommario

Indice delle figure

Indice delle tabelle

1 Risultati attesi

I risultati attesi da capitolato sono di seguito elencati:

- Sintesi dei lavori di letteratura di riferimento (almeno 3).
- Descrizione degli scenari energetici (almeno 3).
- Descrizione della logica del modello sviluppato e del relativo pseudocodice.
- Rapporto tecnico descrittivo del modello e dello pseudocodice.

2 Risultati ottenuti

I risultati ottenuti sono in linea con quelli attesi. Ad integrazione di quanto atteso, nella LA1.4 si è provveduto non solo allo sviluppo della logica del modello proposto e del relativo pseudocodice ma anche del corrispondente codice Python.

Attualmente non sono disponibili software di tipo freeware che effettuano stime analoghe a quelle del modello proposto e ciò comporta un avanzamento rispetto allo stato dell'arte.

3 Prodotti attesi

- Modello e relativo pseudocodice

4 Prodotti ottenuti

- Modello e relativo pseudocodice
- Codice Python del modello

5 Analisi degli scostamenti su attività e risultati

Non si evidenziano scostamenti tecnici/economici sull'attività svolta né sui risultati previsti da capitolato.

6 Sintesi delle attività svolte

Le attività svolte nella presente LA hanno riguardato la definizione di un modello che, a partire degli input dell'utente (es. percentuale di FER, percentuale di elettrificazione, etc.), consente di valutare, per il sistema energetico analizzato, sia la quantità di energia da stoccare per rispettare i vincoli di sistema sia l'eventuale overgeneration da destinare ad altri utilizzi (es. generazione dell'idrogeno) per garantire il bilancio energetico del sistema stesso.

Il modello definito è stato "tradotto" in uno pseudocodice e, successivamente, in un codice Python fornito alla LA1.5. La LA1.5 ha, quindi, implementato il codice di integrazione da fornire alla LA1.15 per l'inclusione in ARS tool.

7 Dettaglio delle attività svolte

7.1 Selezione degli scenari e analisi preliminare

Per la costruzione del modello, è stata condotta preliminarmente un'analisi volta a selezionare:

- lavori di letteratura tecnico-scientifica che proponessero metodologie per la valutazione dell'accumulo necessario a garantire il bilancio energetico del sistema energetico in funzione di diversi valori di penetrazione delle fonti rinnovabili;
- scenari energetici di diffusione delle FER e dello storage in Italia da utilizzare come strumento di confronto per gli output del modello implementato.

In relazione al primo punto, si è selezionato il seguente lavoro:

- [1] Henrik Zsiborács, Nóra Hegedűsné Baranyai, András Vincze, László Zentkó, Zoltán Birkner, Kinga Máté and Gábor Pintér "Intermittent Renewable Energy Sources: The Role of Energy Storage in the European Power System of 2040"; Electronics 2019, 8(7), 729; https://doi.org/10.3390/electronics8070729.

Nell'articolo, gli autori propongono un modello per la stima della capacità di accumulo necessaria in Europa in funzione della percentuale di elettricità generata da fonti rinnovabili non programmabili (FRNP), assumendo condizioni ideali come una gestione efficiente della domanda, regolamentazioni di mercato efficienti, previsioni meteo avanzate e uno sviluppo continuo della rete elettrica.

L'articolo, in particolare, propone un modello di regressione polinomiale in MATLAB, basato sulla logica di sette studi precedenti, di seguito richiamati.

- [2] Schill, W.-P. Residual load, renewable surplus generation and storage requirements in Germany. Energy Policy 2014, 73, 65–79.
- [3] Bertsch, J.; Growitsch, C.; Lorenczik, S.; Nagl, S. Flexibility in Europe's power sector—An additional requirement or an automatic complement? Energy Econ. 2016, 53, 118–131
- [4] European Commission. Roadmap 2050: A practical Guide to a Prosperous, Low-Carbon Europe; European Climate Foundation (ECF): AM Den Haag, The Netherlands, 2010.
- [5] Kondziella, H.; Bruckner, T. Flexibility requirements of renewable energy based electricity systems—A review of research results and methodologies. Renew. Sustain. Energy Rev. 2016, 53, 10–22.
- [6] Agora Energiewende. Stromspeicher in der Energiewende; Agora Energiewende: Berlin, Germany, 2014.
- [7] Fraunhofer Institute for Solar Energy Systems. Roadmap Speicher—Speicherbedarf für Erneuerbare Energien Speicheralternativen—Speicheranreiz—Überwindung rechtlicher Hemmnisse; Endbericht: Freiburg, Germany, 2014.
- [8] Blanco, H.; Faaij, A. A review at the role of storage in energy systems with a focus on Power to Gas and long-term storage. Renew. Sustain. Energy Rev. 2018, 81, 1049–1086.

Dallo studio dei suddetti articoli [2-7] si evince che, per mantenere il bilanciamento del sistema energetico europeo, si rende necessaria una frazione di accumulo pari a:

- 0,0308–0,0372% per livelli di penetrazione FNRP del 16–45% [3];
- 1,02% per livelli di penetrazione FRNP dell'80% [4];
- 1,5% per livelli di penetrazione FRNP del 95% [8].

Dall'analisi dei dati in [1-7] emerge, inoltre, che una quota di produzione da FRNP tra il 40% e il 50% rappresenta un punto critico: oltre questa soglia, il bisogno di accumulo cresce rapidamente. Sotto il 50%, la crescita della frazione di accumulo è più lineare, ma i valori divergono molto tra gli studi per penetrazioni FRNP superiori. Lo scenario a penetrazione del 100% di FRNP, infatti, non è stato analizzato a causa dell'eccessiva variabilità nei dati disponibili.

Mettendo a sistema i vari dati, in [1] vengono ricostruite le curve "frazione di accumulo-penetrazione FRNP" e fornita la relativa correlazione matematica (Figura 1).

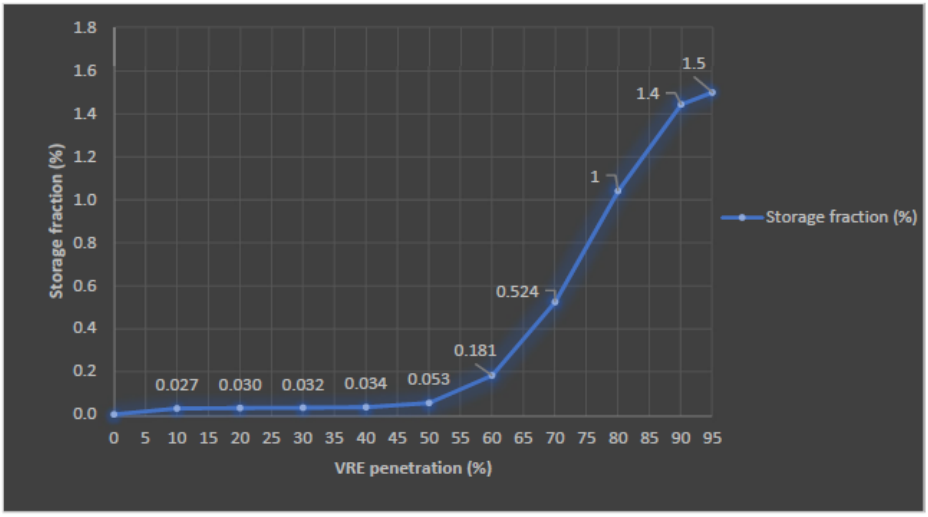

Figure 7. Result of the average European storage fraction analysis.

Table 6. Result of the European storage fraction analysis.

Description	Equation
Equation (1), storage fraction [%]	$Storage\ fraction = p_1VRE^8 + p_2VRE^7 + p_3VRE^6 + p_4VRE^5 + p_5VRE^4 + p_6VRE^3 + p_7VRE^2 + p_8VRE + p_9$
p_i parameter values	$p_1 = -3.758^{-14}$; $p_2 = -1.327^{-11}$; $p_3 = 1.818^{-09}$; $p_4 = -1.234^{-07}$; $p_5 = 4.443^{-06}$; $p_6 = -8.163^{-05}$; $p_7 = 5.844^{-04}$; $p_8 = 1.646^{-03}$; $p_8 = 3.687^{-04}$

Figura 1: curva "frazione di accumulo-penetrazione FRNP" e relativa correlazione matematica (Fonte [1])

Relativamente agli scenari benchmark per l'Italia, sono stati selezionali gli scenari Snam-Terna di cui al Documento di Descrizione degli Scenari 2022 (DDS22), elaborato da Terna e Snam. Il documento illustra 5 scenari, di cui 2 al 2030 e 3 al 2040, che esplorano diverse traiettorie di evoluzione del sistema energetico italiano e si pongono come obiettivo di supportare il raggiungimento dei target UE del pacchetto Fit for 55, che prevede una riduzione del 55% delle emissioni di gas serra entro il 2030 e la neutralità climatica entro il 2050.

In Figura 2 si riporta uno schema dei 5 scenari e, a seguire, i principali dati relativi a ciascuno di essi.

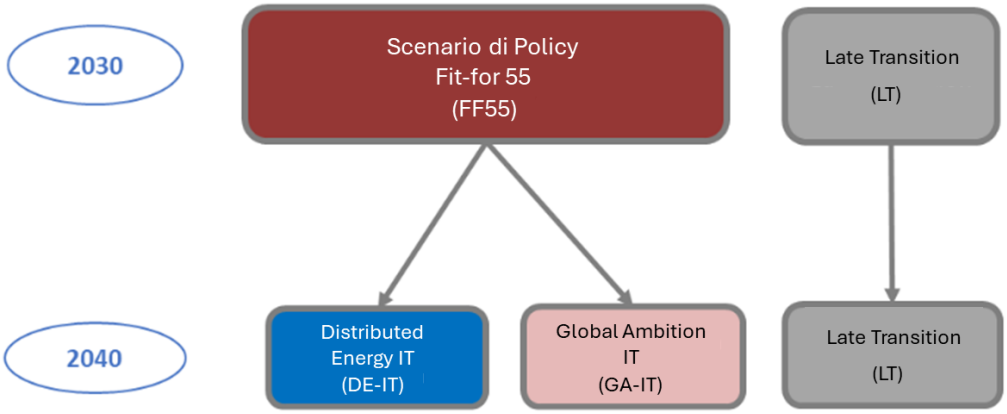

Figura 2: Scenari e anni orizzonte[1]

Scenari al 2030

1. **Fit-for-55 (FF55)**
    - Basato sugli obiettivi europei di riduzione delle emissioni del 55% rispetto ai livelli del 1990.
    - Prevede:
        - **+70 GW** di nuova capacità rinnovabile.
        - **71,5 GWh** di capacità di accumulo elettrico necessaria (esclusi gli impianti di pompaggio esistenti).
        - Sviluppo di **interconnessioni elettriche** per aumentare la capacità di scambio tra zone di mercato di circa **7 GW**.
        - Forte crescita del fotovoltaico, soprattutto al Sud e nelle isole.
        - Integrazione di **43 GW** di rinnovabili nella rete ad alta tensione.
2. **Late Transition 2030 (LT)**
    - Scenario alternativo che ipotizza un ritardo nell'adozione delle politiche climatiche.
    - Minore penetrazione delle rinnovabili e maggiore dipendenza da fonti fossili.
    - Rallentamento nello sviluppo delle infrastrutture e nella decarbonizzazione.

Scenari al 2040

1. **Distributed Energy Italy (DE-IT)**
    - Evoluzione del Fit-for-55 con forte decentralizzazione della produzione energetica.
    - Ampio uso di impianti rinnovabili distribuiti (es. fotovoltaico domestico).

[1] Figura tratta dal Documento di Descrizione degli Scenari 2022 Snam-Terna

- Maggiore coinvolgimento dei consumatori e delle comunità energetiche.

2. **Global Ambition Italy (GA-IT)**
   - Scenario con forte cooperazione internazionale e investimenti su larga scala.
   - Sviluppo massiccio di grandi impianti rinnovabili e infrastrutture di rete.
   - Maggiore penetrazione dell'idrogeno verde e delle tecnologie innovative.
3. **Late Transition 2040**
   - Prosegue lo scenario di transizione ritardata, con ritardi strutturali nella decarbonizzazione.
   - Maggiore rischio di non raggiungere gli obiettivi climatici al 2050.

	2019	2030		2040		
	Consuntivo	FF55	LT	DE-IT	GA-IT	LT
FABBISOGNO DI ELETTRICITA' (TWh)	320	366	331	418	396	389
di cui CONSUMI PER PRODUZIONE H2	-	*9*	-	*18*	*16*	*9*
GENERAZIONE FER (TWh)	*113*	*239*	*187*	*325*	*302*	*244*
di cui SOLARE	23	101	69	157	138	102
di cui EOLICO	20	68	46	108	99	71
GENERAZIONE TERMOELETTRICA NETTA (TWh)	*169*	*80*	*96*	*49*	*53*	*99*
di cui GAS	138	75	91	46	50	94
SALDO IMPORT/EXPORT (TWh)	*38*	*52*	*54*	*54*	*49*	*51*
CAPACITA' INST.FER (GW)	*55*	*122*	*91*	*175*	*160*	*123*
di cui SOLARE	21	75	52	114	102	75
di cui EOLICO	11	27	19	42	39	28
CAPACITA' INST. ACCUMULI (GWh)[4]	*1*	*95*	*51*	*175*	*144*	*71*
CAPACITA' INST. ELETTROLIZZATORI (GW)	-	*5*	-	*11*	*8*	*5*
DOMANDA DI METANO (TWh @ PCS 10,58 kWh/m3) [5]	*789*	*700*	*653*	*561*	*629*	*714*
di cui GAS NATURALE	788	620	641	375	393	599
di cui BIOMETANO	1	57	11	109	109	74
TERMOELETTRICO (INCLUSO CALORE DERIVATO E CALORE DIRETTO)	329	231	243	177	182	285
USI FINALI ENERGETICI	427	413	377	278	290	351
USI FINALI NON ENER, ALTRI USI, PERDITE E BUNKERAGGI	33	35	32	30	30	36
DOMANDA DI IDROGENO (TWh @ PCS 10,58 kWh/m3)	-	23	1	77	127	41
PICCO DI DOMANDA GAS (GWh/giorno)	4.169	4.762	4.373	3.830	3.947	4.582
di cui GAS NATURALE E BIOMETANO (GWh/giorno)	4.169	4.698	4.370	3.619	3.598	4.455
di cui IDROGENO(*) (GWh /giorno)	-	63	3	212	349	127

(*) valore annuale medio

Tabella 1: Principali valori degli scenari DDS22

Analizzati i lavori precedenti e gli scenari DDS22, sono stati definiti valori di frazione dello storage in funzione della penetrazione FNRP da utilizzare all'interno del modello descritto nella prossima sezione.

7.2 Definizione del modello e dello pseudocodice

Il modello proposto mira a valutare la quantità di accumulo e di overgeneration in funzione della penetrazione delle FRNP e dei valori di input dell'utente (Livello di elettrificazione, percentuale di crescita annua dei consumi di energia finale t.m.a, percentuale di generazione da FER, percentuale di solare su FRNP, consumi per produzione di H2).

Uno schema della logica proposta è riportato in Figura 3.

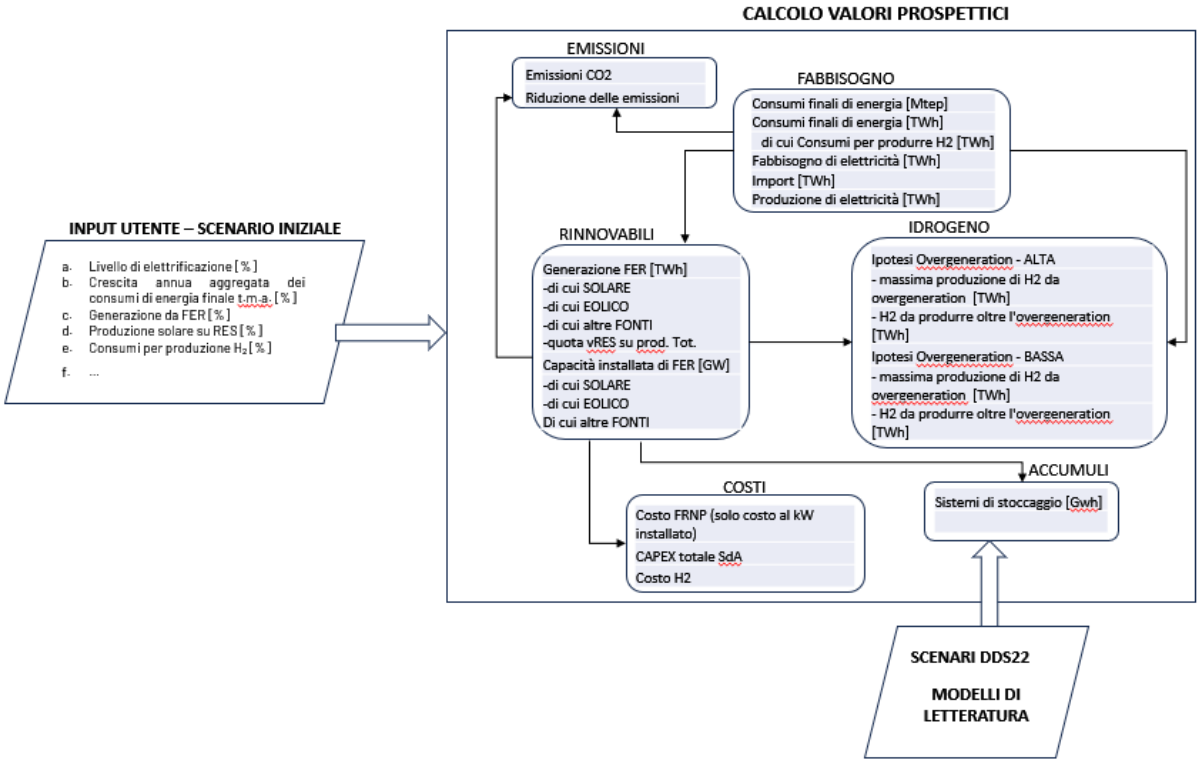

Figura 3: Logica del modello proposto

Più nello specifico, a partire dai valori input dell'utente (scenario iniziale), vengono calcolati (Figura 3):

- il "fabbisogno prospettico" (consumi finali di energia, fabbisogno e produzione di elettricità, import)
- le "rinnovabili" (generazione e capacità installata)
- l'accumulo
- l'overgeneration, eventualmente convertita in produzione di idrogeno
- costi al kW di FRNP installati
- emissioni

Per valutare l'accumulo, è stata, utilizzata la *frazione di storage in funzione della penetrazione delle FRNP* valutata utilizzando gli scenari DDS22 e i modelli di letteratura descritti in sezione 7.1.

La formulazione matematica del modello è di seguito riportata nello pseudocodice implementato.

Pseudo codice categoria Fabbisogno energetico

LETTURA DATI DI INUT*: leggere i vincoli di sistema (vincoli: a,b,c,d,e) e lo scenario energetico scelti dall'utente*

CALCOLO VALORI PROSPETTICI FABBISOGNO ENERGETIO*:*

Consumi finali di Energia [Mtep] = valore_attuale_da_scenario_energetico(1+b)^8*

Consumi finali di Energia [TWh] = Consumi finali di Energia [Mtep]/0,086

*di cui Consumi per produrre H2 [TWh] = Consumi finali di Energia * e*

Fabbisogno di elettricità [TWh] = a Consumi finali di Energia [TWh]*

Import [TWh] = ipotizzare a partire dal valore attuale da scenario energetico scelto

Produzione di elettricità [TWh] = Fabbisogno di elettricità [TWh] - Import [TWh]

FINE

Pseudo codice categoria Rinnovabili

***LETTURA DATI DI INUT**: leggere i vincoli di sistema (vincoli: a,b,c,d,e) e lo scenario energetico scelti dall'utente*

***CALCOLO VALORI PROSPETTICI GENERAZIONE RINNOVABILE**:*

Generazione FER [TWh] = c Produzione di elettricità [TWh]*

*di cui SOLARE [TWh]_1 = (Generazione FER [TWh] - altre_fonti_da_scenario)*d*

di cui EOLICO [TWh]_1 = (Generazione FER [TWh] - altre_fonti_da_scenario)(1-d)*

di cui ALTRE FONTI_1 [TWh] = Generazione FER - di cui SOLARE - di cui EOLICO

Overgeneration BASSA-ALTA [TWh] = 3,60-6,66

capacità installata di FER [TWh] = di cui SOLARE + di cui SOLARE + di cui ALTRE FONTI

di cui SOLARE [TWh] = di cui EOLICO [TWh]_1 / 1,35

di cui EOLICO [TWh] = di cui EOLICO [TWh]_1 / 2,51

di cui ALTRE FONTI [TWh] = altre_fonti_da_scenario

FINE

Pseudo codice categoria Accumuli

***LETTURA DATI DI INUT**: leggere i vincoli di sistema (vincoli: a,b,c,d,e) e lo scenario energetico scelti dall'utente*

***CALCOLO VALORI PROSPETTICI ACCUMULI**:*

*sistemi di stoccaggio [GWh] **=** (di cui SOLARE _1 + di cui EOLICO_1) * β*

Pseudo codice categoria Idrogeno

***LETTURA DATI DI INUT**: leggere i vincoli di sistema (vincoli: a,b,c,d,e) e lo scenario energetico scelti dall'utente*

***CALCOLO VALORI PROSPETTICI IDROGENO**:*

*max produzione di H2 da overgeneration [TWh]= Overgeneration BASSA-ALTA * 0,7*

H2 da produrre oltre l'overgeneration [TWh] = di cui Consumi per produrre H2 - max produzione di H2 da overgeneration

FINE

Pseudo codice categoria Costi

***LETTURA DATI DI INUT**: leggere i vincoli di sistema (vincoli: a,b,c,d,e) e lo scenario energetico scelti dall'utente*

***CALCOLO VALORI PROSPETTICI COSTI**:*

Costo FRNP (solo costo al kW installato) = (800 di cui SOLARE + 900* di cui EOLICO)*1000000*

FINE

Pseudo codice categoria Emissioni

***LETTURA DATI DI INUT**: leggere i vincoli di sistema (vincoli: a,b,c,d,e) e lo scenario energetico scelti dall'utente*

***CALCOLO VALORI PROSPETTICI EMISSIONI**:*

Emissioni CO2 = 500(Produzione di elettricità - Generazione FER)*10^9 / 10^6*

Riduzione delle emissioni = Emissioni CO2 - Emissioni CO2 da scenario

FINE

7.3 Sviluppo del codice Python

Il modello sviluppato è stato implementato nel linguaggio Python al fine di essere integrato nel tool sviluppato nella LA1.15.

Di seguito, si riporta il codice sviluppato nella LA1.4 per l'esecuzione operativa del modello sopra riportato comprensiva delle righe di codice sviluppate nella LA1.5 per l'integrazione nella LA1.15. Per la modalità di utilizzo del modello si rimanda al manuale utente del software ARStool.

Codice python della classe "OptStorWGT".

```
class OptStorWGT(QMainWindow):
  def __init__(self):
    super(OptStorWGT, self).__init__(None)
    self.ui = Ui_MainWindow()
    self.ui.setupUi(self)

    self.ui.resultsWgt.setVisible(False)

    # Popolazione della tendina "Anno"
    self.ui.yearScenCb.clear()
    self.ui.yearScenCb.addItems(list(s.keys()))

    self.year_selected()   # azioni conseguenti alla selezione dell'anno
    self.scen_selected()   # azioni conseguenti alla selezione dello scenario

    # — Azioni dei pulsanti e dei menu a tendina ——————————————
    self.ui.yearScenCb.currentTextChanged.connect(self.year_selected)
    self.ui.scenCb.currentTextChanged.connect(self.scen_selected)
    self.ui.calcPb.clicked.connect(self.calculation)
    # ————————————————————————————————————

  # Calcolo dei risultati
  def calculation(self):
```

```
    # lettura dei dati di input e scrittura nelle variabili
    for p in ['electrLev', 'enConsGrow', 'ferGen', 'h2Cons', 'solarFer']:
        self.__setattr__(p + 'Input', self.ui.__getattribute__(p + 'InputDsb').value())

    self.ui.resultsWgt.setVisible(True)    # Rendo visibile il Widget dei risultati

    # — Calcolo delle variabili dei risultati ————————————————————
    self.totEnConsMtep = self.totEnConsMtepScen * (1 + self.enConsGrowInput / 100) ** 8
    self.totEnCons = self.totEnConsMtep / 0.086
    self.h2Cons = self.h2ConsInput * self.totEnCons / 100
    self.enDuty = self.totEnCons * self.electrLevInput / 100
    self.enImport = 52
    self.enProd = self.enDuty - self.enImport
    self.ferGen = self.enProd * self.ferGenInput / 100
    self.ferGenPerc = self.ferGenInput
    self.solarGen = (self.ferGen - self.otherGenScen) * self.solarFerInput / 100
    self.windGen = (self.ferGen - self.otherGenScen) * (1 - self.solarFerInput / 100)
    self.otherGen = self.ferGen - self.solarGen - self.windGen
    self.solarCap = self.solarGen / 1.35
    self.windCap = self.windGen / 2.51
    self.otherCap = self.otherCapScen
    self.ferCap = self.solarCap + self.windCap + self.otherCap
    self.co2Emis = (self.enProd - self.ferGen) * 0.5
    self.co2EmisRed = self.co2EmisScen - self.co2Emis

    self.hiOvergen = (self.solarGen + self.windGen) * 0.05
    self.lowOvergen = (self.solarGen + self.windGen) * 0.027

    self.sysStor = self.sysStor_calc()  # calcolata con la funzione self.sysStor_calc

    self.maxHiH2 = self.lowOvergen * 0.7
    self.maxLowH2 = self.hiOvergen * 0.7
    self.toprodHiH2 = self.h2Cons - self.maxHiH2
    self.toprodLowH2 = self.h2Cons - self.maxLowH2

    self.frnpCosts = (800 * self.solarCap + 900 * self.windCap)
    self.capexCosts = 0
    self.h2Costs = 0
    # ————————————————————————————————————————————-

    # — Calcolo delle variabili delle percentuali dei risultati ——————————————
    for c in ['Gen', 'Cap']:
        for p in ['solar', 'wind', 'other']:
            self.__setattr__(p + c + 'Perc',
                             self.__getattribute__(p + c) / self.__getattribute__('fer' + c) * 100)
    # ————————————————————————————————————————————-

    # — Scrittura dei risultati nelle caselle ————————————————————
    # —- Scrittura dei risultati presenti come scenario che come risultato ——————-
    for p in ['totEnConsMtep', 'totEnCons', 'h2Cons', 'enDuty', 'enImport', 'enProd', 'ferGen', 'ferGenPerc',
              'solarGen', 'solarGenPerc', 'windGen', 'windGenPerc', 'otherGen', 'otherGenPerc', 'ferCap',
              'solarCap', 'solarCapPerc', 'windCap', 'windCapPerc', 'otherCap', 'otherCapPerc', 'co2Emis']:
        self.ui.__getattribute__(p + 'ResPerspDsb').setValue(self.__getattribute__(p))
        self.ui.__getattribute__(p + 'ResActDsb').setValue(self.__getattribute__(p + 'Scen'))
    # ————————————————————————————————————————————-

    # —- Scrittura dei risultati presenti solo come risultato ——————————————
    for p in ['co2EmisRed', 'hiOvergen', 'lowOvergen', 'sysStor', 'maxHiH2', 'maxLowH2', 'toprodHiH2',
              'toprodLowH2', 'frnpCosts', 'capexCosts', 'h2Costs']:
        self.ui.__getattribute__(p + 'ResPerspDsb').setValue(self.__getattribute__(p))
    # ————————————————————————————————————————————-
    # ————————————————————————————————————————————-

    self.barchart()    # creazione dei diagrammi

# Azioni conseguenti alla selezione dell'anno: popolazione della tendina dello scenario
```

```
def year_selected(self):
    self.ui.scenCb.clear()
    self.ui.scenCb.addItems(list(s[self.ui.yearScenCb.currentText()].keys()))

# Azioni conseguenti alla selezione delo scenario
def scen_selected(self):
    year = self.ui.yearScenCb.currentText()
    scen = self.ui.scenCb.currentText()

    if scen:    # evita azioni inutili durante la popolazione della tendina scenario
        for k in s[year][scen]:
            # Scrittura delle variabili di scenario
            self.__setattr__(k + 'Scen', s[year][scen][k])
            # Scrittura delle caselle di scenario
            self.ui.__getattribute__(k + 'ScenDsb').setValue(s[year][scen][k])

        # — Calcolo e scrittura nelle caselle di alcune variabili non presenti nello scenario —-
        self.otherGenScen = self.ferGenScen - self.solarGenScen - self.windGenScen
        self.otherCapScen = self.ferCapScen - self.solarCapScen - self.windCapScen
        self.co2EmisScen = (self.enProdScen - self.solarGenScen - self.windGenScen) * 0.33

        for p in ['otherGen', 'otherCap', 'co2Emis']:
            self.ui.__getattribute__(p + 'ScenDsb').setValue(self.__getattribute__(p + 'Scen'))
        # ——————————————————————————————————————————

        # — Calcolo delle variabili delle percentuali dei vari di scenario ——————————
        for c in ['Gen', 'Cap']:
            self.__setattr__('fer' + c + 'PercScen',
                             self.__getattribute__('fer' + c + 'Scen') / self.enDutyScen * 100)
            for p in ['solar', 'wind', 'other']:
                self.__setattr__(p + c + 'PercScen',
                                 self.__getattribute__(p + c + 'Scen') /
                                 self.__getattribute__('fer' + c + 'Scen') * 100)
        # ——————————————————————————————————————————

# Calcolo dello storage
def sysStor_calc(self):
    # Verifica dela linea corrispondente alla percentuale di input
    for i in range(len(storage[0])):
        x = storage[0][i]
        if self.ferGenInput / 100 <= x:
            break

    # Calcolo dell'interpolazione del valore corrispondente nell'intorno dell'input
    y = (storage[1][i - 1] + (self.ferGenInput / 100 - storage[0][i - 1]) *
         (storage[1][i] - storage[1][i - 1]) / (storage[0][i] - storage[0][i - 1]))

    return (self.solarGen + self.windGen) * y

# Creazione dei diagrammi dei risultati
def barchart(self):
    # Inizializzazione dei dizionari dei grafici
    self.graph = {
        '': {
            'title': 'Fabbisogno',
            'barSeries': QtCharts.QBarSeries(),
            'chart': QtCharts.QChart(),
            'axis_x': QtCharts.QBarCategoryAxis(),
            'axis_y': QtCharts.QValueAxis(),
            'y_title': 'Energia [TWh]',
            'hi': 0,
            'y_scale': 0,
            'series': {
                # 'totEnCons': 'Consumi finali di energia',
                'h2Cons': 'Consumi\nper la produzione\ndi H2',
                'enDuty': 'Fabbisogno\ndi energia\nelettrica',
```

```
            'enProd': 'Energia\nprodotta',
        }
    },
    'Gen': {
        'title': 'Generazione',
        'barSeries': QtCharts.QBarSeries(),
        'chart': QtCharts.QChart(),
        'axis_x': QtCharts.QBarCategoryAxis(),
        'axis_y': QtCharts.QValueAxis(),
        'y_title': 'Energia [TWh]',
        'hi': 0,
        'y_scale': 0,
        'series': {
            'solar': 'Solare',
            'wind': 'Eolico',
            'other': 'Altro',
        }
    },
    'Cap': {
        'title': 'CapacitÃ ',
        'barSeries': QtCharts.QBarSeries(),
        'chart': QtCharts.QChart(),
        'axis_x': QtCharts.QBarCategoryAxis(),
        'axis_y': QtCharts.QValueAxis(),
        'y_title': 'Potenza installata [GW]',
        'hi': 0,
        'y_scale': 0,
        'series': {
            'solar': 'Solare',
            'wind': 'Eolico',
            'other': 'Altro',
        }
    },
}

# Elimino il widget dei grafici
try:
    self.ui.graphWgt.deleteLater()
except:
    pass

self.ui.graphWgt = QtWidgets.QWidget()              # Ricreo il widget dei grafici
self.ui.optStorHL.insertWidget(2, self.ui.graphWgt)    # e lo inserisco nel layout

# Calcolo la dimensione del widget dei grafici sulla base della dimensione degli altri widget
graphsize = self.ui.optStorWgt.width() - self.ui.leftWgt.width() - self.ui.resultsWgt.width() - 40

# Creazione del layout verticale dei grafici
self.ui.graphVL = QtWidgets.QVBoxLayout(self.ui.graphWgt)
self.ui.graphVL.setContentsMargins(0, 0, 0, 0)

# I grafici dovranno avere un valore compreso tra 200 e 500, e pari al valore calcolato
self.ui.graphWgt.setFixedSize(max(200, min(graphsize, 500)), self.ui.optStorWgt.height())

# Creazione dei grafici mediante il dizionario
for c in self.graph:
    self.__setattr__('set' + c + 'Act', QtCharts.QBarSet('Valori attuali'))
    self.__setattr__('set' + c + 'Persp', QtCharts.QBarSet('Valori prospettici'))

    for p in self.graph[c]['series']:
        self.graph[c]['axis_x'].append([self.graph[c]['series'][p]])

        self.__getattribute__('set' + c + 'Act').append(self.__getattribute__(p + c + 'Scen'))
        self.__getattribute__('set' + c + 'Persp').append(self.__getattribute__(p + c))

        if self.__getattribute__(p + c + 'Scen') > self.graph[c]['hi']:
```

```
                self.graph[c]['hi'] = self.__getattribute__(p + c + 'Scen')
            if self.__getattribute__(p + c) > self.graph[c]['hi']:
                self.graph[c]['hi'] = self.__getattribute__(p + c)

            mag = math.log10(self.graph[c]['hi']) // 1
            self.graph[c]['y_scale'] = 10 ** mag * (1 + round(self.graph[c]['hi'] / (10 ** mag)))

        self.graph[c]['barSeries'].append(self.__getattribute__('set' + c + 'Act'))
        self.graph[c]['barSeries'].append(self.__getattribute__('set' + c + 'Persp'))

        self.graph[c]['chart'].addSeries(self.graph[c]['barSeries'])

        self.graph[c]['chart'].setTitle(self.graph[c]['title'])
        self.graph[c]['chart'].setAnimationOptions(QtCharts.QChart.SeriesAnimations)

        self.graph[c]['chart'].addAxis(self.graph[c]['axis_x'], Qt.AlignBottom)
        self.graph[c]['barSeries'].attachAxis(self.graph[c]['axis_x'])

        self.graph[c]['axis_y'].setRange(0, self.graph[c]['y_scale'])
        self.graph[c]['axis_y'].setTitleText(self.graph[c]['y_title'])

        self.graph[c]['chart'].addAxis(self.graph[c]['axis_y'], Qt.AlignLeft)
        self.graph[c]['barSeries'].attachAxis(self.graph[c]['axis_y'])

        self.graph[c]['chart'].legend().setVisible(True)
        self.graph[c]['chart'].legend().setAlignment(Qt.AlignBottom)

        self.graph[c]['barGraph'] = QtCharts.QChartView(self.graph[c]['chart'])
        # L'altezza del grafico deve essere tale da permettere 3 gradici nel layout
        self.graph[c]['barGraph'].setMaximumHeight((self.ui.leftWgt.height() - 20) / 3)

        self.graph[c]['chart'].setBackgroundBrush(QtGui.QBrush(QtGui.QColor('transparent')))
        self.graph[c]['chart'].legend().setLabelColor(QtGui.QColor('white'))
        self.graph[c]['chart'].setTitleBrush(QtGui.QColor(255, 255, 255))
        self.graph[c]['chart'].setTitleFont(QtGui.QFont("Arial", 12))
        self.graph[c]['chart'].axisX().setLabelsColor(QtGui.QColor(255, 255, 255))
        self.graph[c]['chart'].axisY().setLabelsColor(QtGui.QColor(255, 255, 255))
        self.graph[c]['chart'].axisY().setTitleBrush(QtGui.QColor(255, 255, 255))
        self.graph[c]['chart'].axisY().setLabelFormat("%.0f")
        self.graph[c]['chart'].legend().setFont(QtGui.QFont("Arial", 10))

        self.ui.graphVL.addWidget(self.graph[c]['barGraph'])  # Aggiunta dei grafici nel layout

    # Inserimentdo ti duno spaziatore sotto i grafici
    bm_spacer = QSpacerItem(20, 10)
    self.ui.graphVL.addItem(bm_spacer)


s = {
  '2019': {
    'Consuntivo': {
      'totEnConsMtep': 112,
      'totEnCons': 1302.3,
      'h2Cons': 0,
      'enDuty': 320,
      'enImport': 51,
      'enProd': 269,
      'ferGen': 113,
      'solarGen': 23,
      'windGen': 20,
      'ferCap': 55,
      'solarCap': 21,
      'windCap': 11,
    },
  },
  '2030': {
```

```
    'FF55': {
      'totEnConsMtep': 112,
      'totEnCons': 1302.3,
      'h2Cons': 9,
      'enDuty': 366,
      'enImport': 52,
      'enProd': 314,
      'ferGen': 239,
      'solarGen': 101,
      'windGen': 68,
      'ferCap': 122,
      'solarCap': 75,
      'windCap': 27,
    },
    'LT': {
      'totEnConsMtep': 112,
      'totEnCons': 1302.3,
      'h2Cons': 0,
      'enDuty': 331,
      'enImport': 91,
      'enProd': 260,
      'ferGen': 187,
      'solarGen': 69,
      'windGen': 46,
      'ferCap': 91,
      'solarCap': 52,
      'windCap': 19,
    },
  },
  '2040': {
    'DE-IT': {
      'totEnConsMtep': 112,
      'totEnCons': 1302.3,
      'h2Cons': 18,
      'enDuty': 418,
      'enImport': 54,
      'enProd': 364,
      'ferGen': 325,
      'solarGen': 157,
      'windGen': 108,
      'ferCap': 175,
      'solarCap': 114,
      'windCap': 42,
    },
    'GA-IT': {
      'totEnConsMtep': 112,
      'totEnCons': 1302.3,
      'h2Cons': 16,
      'enDuty': 396,
      'enImport': 49,
      'enProd': 347,
      'ferGen': 302,
      'solarGen': 138,
      'windGen': 99,
      'ferCap': 160,
      'solarCap': 102,
      'windCap': 39,
    },
    'LT': {
      'totEnConsMtep': 112,
      'totEnCons': 1302.3,
      'h2Cons': 9,
      'enDuty': 389,
      'enImport': 52,
      'enProd': 337,
      'ferGen': 244,
```

```
        'solarGen': 102,
        'windGen': 71,
        'ferCap': 123,
        'solarCap': 75,
        'windCap': 28,
      },
    },
}


# Lettura del dizionario della curva di riferimento dello storage
storage = [[ ], [ ]]
with open('./Functionalities/OptimalStorage/StorageTerna.csv', mode='r') as csv_file:
    for line in csv_file:
        storage[0].append(float(line.rstrip().split(';')[0]))
        storage[1].append(float(line.rstrip().split(';')[1]))
    csv_file.close()
```

8 Contributo delle eventuali consulenze alle attività sopra descritte

Nella LA1.4 non sono state attivate consulenze.

9 Pubblicazioni scientifiche

G. Ferruzzi, M. Valenti, G. Graditi, C. Delcea, A. Domenteanu, "Social Variables in Energy System Optimization: Models, Tools, and Current Approaches", 2024 IEEE International Humanitarian Technologies Conference (IHTC), DOI: 10.1109/IHTC61819.2024.10855090

10 Eventi di disseminazione

L'attività è stata presentata nel corso dell'evento finale di disseminazione del progetto 2.3 (3 dicembre 2024, Centro Ricerche di Portici) e nel corso della presentazione dei risultati del progetto all'EERA Joint Programme Smart Grid presso il Centro Ricerche DTU (Danimarca, 23-24 aprile 2024).