

Kandidatprogrammet i elektronik och elektroteknik

Tillämpning av solenergi för frystorkning

En kostnadsanalys

Alexander Sippoin

Copyright ©2025 Alexander Sippoin

Författare Alexander Sippoin		
Titel Tillämpning av solenergi för frystorkning		
Utbildningsprogram Kandidatprogrammet i elektroteknik		
Huvudämne Elektronik och elektroteknik		
Övervakare Äldre universitetslektor Markus Turunen		
Handledare Äldre universitetslektor John Millar		
Datum 11.12.2025	Sidantal 45	Språk Svenska

Sammandrag

Solenergi har med årens lopp blivit mer och mer lönsamt och frystorkning är en effektiv metod för att konservera livsmedelsprodukter för att avsevärt förlänga deras hållbarhet. Men frystorkning är en mycket energikrävande process som kan ta över ett dygn per cykel, vilket leder till att en frystorkningsanläggning behöver en stadig försörjning av energi. Detta är en analys över solenergens lönsamhet och användning för frystorkningsanläggningar i Finland.

I arbetet utreddes vilka dimensioner för torkningsanläggningar, solceller och eventuellt batteriförvar är möjliga och vad för begränsningar frystorkning för med sig till planeringen och förverkligandet av en sådan anläggning. Tidsseriedata på solenergi i Finland analyserades och kombinerades med torkningsprocessens krav. I kostnadsanalysen beräknas optimala dimensioner på solceller och batteriförvar för att en frystorkningsanläggning i Helsingforsregionen i Finland.

Nyckelord solcell, sol-energi, förnybar energi, frystorkning, kostnadsanalys, MATLAB

Author	Alexander Sippoin	
Title of thesis	Utilising photovoltaics for freeze drying	
Programme	School of Electrical Engineering	
Major	Electrical Engineering	
Thesis supervisor	Senior University Lecturer Markus Turunen	
Thesis advisor(s)	Senior University Lecturer John Millar	
Date	Number of pages	Language
11.12.2025	45	Swedish

Abstract

Solar energy has over the years become increasingly profitable, and freeze-drying is an efficient method for preserving food products to significantly extend their shelf life. However, freeze-drying is a highly energy-intensive process that can take more than a day per cycle, which means that a freeze-drying facility requires a steady supply of energy. This work is an analysis of the profitability and use of solar energy for freeze-drying facilities in Finland.

The study investigated which sizes of drying facilities, solar panels, and possible battery storage solutions are feasible, and what limitations freeze-drying introduces to the planning and implementation of such a system. Time-series data on solar energy in Finland was analysed and combined with the requirements of the drying process. In the cost analysis, optimal dimensions for solar panels and battery storage were calculated for a freeze-drying facility in the Helsinki region of Finland.

Keywords Photovoltaic, freeze dry, renewable energy, solar power, cost analysis, MATLAB

Innehåll

Förord	6
Symboler och förkortningar	7
Symboler	7
Förkortningar	7
1 Inledning	8
2 Torkningsprocessens skeden och energikrav	10
3 Energimarknaden, energiproduktion och hushållsstorlek.....	11
3.1 Ändringar för stor skala med en tidsperiod på 10 år	12
4 Fem scenarier och antaganden	13
4.1 Antaganden, formler och använda siffror	13
4.2 Beräkningar efter basscenarier	14
4.3 Resultat	14
5 Slutsatser	21
5.1 Vidare utveckling och potentiella förbättringar	21
Källor.....	22
A. Informatörer och samlad data	23
B. Källhänvisningar	24
C. MATLAB-koden som bilaga	25

Förord

Ett stort tack till min handledare John Millar för handledning, uppmuntran och hjälpen med att hitta rätt riktning till kandidatarbetet. Även ett tack till Kaisa Lehtipuro från Nordic Freeze Dry för ork och tålamod att svara på alla frågor om frystorkning och frystorkningsanläggningar som jag ställde.

Alexander Sippoin

Symboler och förkortningar

Symboler

- A = årliga kostnader
- C = installeringskostnader
- D = diskonteringsränta eller diskonteringsfaktor (procentuell approximation på hur värdet går ner)

Förkortningar

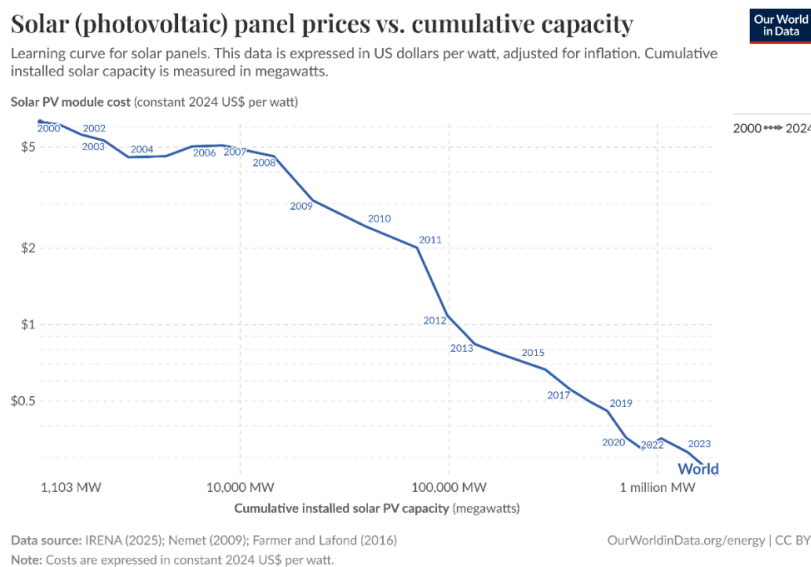
BESS	Batteri energi bevaringssystem (eng. "battery energy storage system")
CAPEX	Anläggningskostnader (eng. "capital expenditure"), mängden pengar som måste läggas i anläggningskostnader och utveckling.
DoD	Procentuell urladdning (eng. "depth of discharge")
kW	Kilowatt (eng. "kilo Watt") $10^3 \cdot W$, effekt
kWh	Kilowattimme (eng. "kilo Watt hour"), energi
kWp	Kilowatteffekt (eng. "kilo Watt power"), högsta effekt ut
NPC	Nettonuvärdeskostnad (eng. "net present cost"), totala kostnaden omräknad till dagens penningvärde
NPV	Nettonuvärde (eng. "net present value"), nuvärdet av alla intäkter minus alla kostnader – hur lönsam en investering är i dagens penningvärde
O&M	Drift och underhåll (eng. "operations and maintenance")
PV	Solceller (eng. "photo voltaic")
RTE	Procentuell kapacitet som kan användas (eng. "round-trip efficiency"), energi
SoC	Procentuell laddning (eng. "state of charge")

1 Inledning

Dagens energiläge och den gröna övergången för med sig nya utmaningar och behov av nya lösningar och teknologi. Arbetet forskar i hur frystorkning kan tillämpas i kombination med solenergi.

Kandidatarbetet utreder användningen av solkraftverk för att balansera och effektivisera energikostnaden för frystorkningsprocessen. Energiförbrukningen och energiproduktionen är inte perfekt synkroniserade och det finns tider då energipriserna reflekterar denna obalans, vilket går att använda till fördel. För att kunna ha mer frihet i energianvändningen, utreds det om batterier är en lönsam energiförvaringsmetod för energianvändningen då elen är dyr eller då solcellerna inte producerar el. Lönsamheten analyseras över en tidsperiod på 10 år.

Att utnyttja solenergi är billigare och lönsammare än förr. Solcellers pris har gått stadigt ner, som man ser från grafen i figur 1.



Figur 1. Prisutveckling av solceller åren 2000-2024 [1].

Solljusets irradians (W/m^2) varierar beroende på ställe, väder, dygnstid och årstid. Den simulerade data på Finland har använts till uträkningar på producerad energi. Hela årets energiproduktion är relevant eftersom olika populära produkter att torka finns tillgängliga året om. Fast man ofta hör talas om frystorkade bär – som har en säsongtid – finns det rikligt med produkter utan någon specifik säsongtid.

Forskningsfrågorna som besvaras i arbetet är: **är solkraft är kostnadseffektivt och gör batterier det mer kostnadseffektivt?** I utredningen

över batterier beaktas industriell skala samt en kort inblick i storleksklassen för hushåll.

I kapitel två går arbetet in på hur frystorkning och en torkningsanläggning opererar och kartlägger energikrav på en anläggning. I det tredje kapitlet behandlas energimarknadens och solcellsproduktionens samverkan och hur batterier kan användas till nytta på elmarknaden. Det fjärde kapitlet lägger upp fem olika scenarier med olika faktorer på vad som ska beaktas för att beräkna kostnader för en frystorkningsanläggning med solceller och jämför scenarierna sinsemellan med hjälp av MATLAB-mjukvara. Koden räknar också ut den ekonomiskt mest lönsamma sammansättningen av solceller, batteri-energiförvar och dimensionerna för en torkningsanläggning, På basen av spotpriser året 2024 och ett år från simulerade solcellsdata. Kapitel fem presenterar uträkningar, siffror och diskuterar slutsatser för vad de simulerade MATLAB-scenarierna resulterar i.

I slutet av kandidatarbetet finns information och förklaringar på data som använts, om varifrån information har fåtts och samlats samt en källförteckning för referenser. Helt i slutet har MATLAB-koden tillagts som en bilaga.

2 Torkningsprocessens skeden och energikrav

Frystorkningen har olika skeden: nedfrysning, undertryck, uppvärmning i undertryck och sublimering. Idén är att produkten fryses ner och värms upp sakta i vakuum så att vätskan i produkten sublimerar rakt från fast form till gas och pumpas ut ur systemet utan att förstöra produktens mikrostrukturer. Den mest energikrävande delen är att värma upp den nedfrysta produkten för att få så mycket vätska som möjligt bort (ca 80 % av energin går åt till att avdunsta vattnet - enligt information från Wave company).

Att torka en sats produkt med en typisk frystorkare för hushåll tar mellan 20 och 30 timmar, konsumerar 1-3 kW, eller 1,5-2 kW/kg av produkt, enligt information från Wave company [2] och Havion Freeze Dry Technology [3].

Batteriers livslängd varar kring 10 år, beroende på hur mycket de används, medan solcellernas livslängd är kring 20 år. Batterier ger möjlighet till att skära av topparna på den dyraste elen, på engelska "peak shaving", samt besparingar på elfördelningsavgifter.

Batterier har olika kemiska sammansättningar, men idag baserar de flesta sig på litium. Till näst jämförs hushållsbatterierna Tesla power wall 2 (Li-NCM, nickel-cobalt-manganese), Duracell Dura5 (LiFePO₄) och Huawei LUNA-2000-14 (LiFePO₄).

3 Energimarknaden, energiproduktion och hushållsstorlek

Solenergi för med sig en problematik till energimarknaden; den har en tendens till självkannibalisering. Det betyder att produktionen bidrar till att dra ner elpriset så att ett stort utbud och en minskad efterfrågan uppstår. Till exempel solenergianläggningar orsakar elpriserna att gå ner när det är soligt. Eftersom anläggningarna säljer produktionen samtidigt, blir det en slags negativ feedbackeffekt. Så för små producenter blir det lönsammare att använda sin egen producerade energi när det är soligt, för att inte behöva sälja el oförmånligt. Genom att utnyttja batterier kan energins förvaringskapacitet utökas.

Kostnader för batterierna är relativt stora och batteriernas livslängd är dålig i jämförelse med solcellerna. Solceller anses ha en livslängd på 20 år medan batterierna vanligtvis har en garantiperiod på 10 år. På samma sätt som solcellerna, har batteripriserna också gått ner. För att utreda om batterier är lönsamma har batteripriser och egenskaper som kapaciteter och RTE jämförts. För kapacitet antas att tillverkaren har angett den användbara kapaciteten (mellan 20 och 80 procent laddning av batteriets fulla kapacitet).

Tabell 1 visar tre populära hushållsbatteriers specifikationer, pris och kostnadsjämförelse. Som källor har databladerna för Huawei LUNA2000-14 [4], Tesla Power Wall 2 [5] och Duracell Dura5 [6] använts. Priserna är uppskattningar gjorda med hjälp av priser på webb-butiker för kommersiella försäljare i EU. Priserna och siffrorna är grova uppskattningar och deras syfte är att visa trenden.

Tabell 1

Modell	Kapacitet (kWh)	RTE (datab-laden eller antagande), %	cykler i året (en cykel i dagen)	Årlig laddning efter RTE, kWh	pris (avrundade närmevärden), €	pris för kapacitet, €/kWh
Huawei LUNA2000-14	13,8	95 %	365	4785 kWh	6600 €	480
Tesla Power Wall 2	13,5	90 %	365	4434 kWh	6900 €	511
Duracell Dura5	4,6	95 %	365	1595 kWh	5000 €	1087

Formel för årlig laddning efter RTE är $Kapacitet * RTE * cykler$.

Enligt tabellen blir Huawei LUNA2000 billigast för en konsument. Med dessa tre datapunkter kan man extrapolera att med större kapacitet blir kWh-priset billigare. I tabellen ser man också att om LUNA2000 ger en kapacitet på 4785 kWh i året skulle det räcka till att torka ca 3190kg av produkt (med siffror från Wave company). I en finsk nätbutik kostar torkade blåbär 275 €/kg.

Färska eller frysta blåbär kostar i Finland mellan 8 €/kg och 20 €/kg och består till 85 % av vatten enligt USDA och livsmedelsdata.se. I frystorkningsprocessen avdunstar över 95 % av vattenhalten, vilket betyder att av 1kg färska blåbär får man 192,5 kg ($0,05 * 0,85 + 0,15 = 0,1925$ kg) torkad produkt. Den mängden skulle kosta 52,9 €. Blåbärens pris ökar med en faktor mellan 6,6 och 2,65 beroende på inköpspris (8–20 €). Med så höga vinster kan produktionskostnaderna vara ganska höga. Vilket betyder att energiprisernas roll kan vara liten.

3.1 Ändringar för stor skala med en tidsperiod på 10 år

För att vidare utreda om solceller och batterier är lönsamma investeringar för en anläggning, genererades simulationer med olika scenarier. När räkningar på större skala gjordes, användes följande konstanter för att tillåta skalandet av system; 105 €/kWh för pris på batteri-energiförvar [7], 548 €/kWp för pris på solceller i Tyskland (sidan 50 [8]). Ytterligare för att ta i beaktande CAPEX, dvs. kapitalkostnader, används ett tillägg på 0,8–1,05 €/W.

4 Fem scenarier och antaganden

För att reda ut kostnader har fem olika scenarier jämförts. Basfallen motsvarar en konstant förbrukning och behandlar kostnader över ett helt år. Frystorkningens konsumtion antas som en konstant belastning på 20 kW med 24 timmars torkningscykler. För industriella storlekens batteriers pris per kapacitet användes 105 €/kWh som användes i en studie på energireserv till elnätverk [7, tabell 3]. Priset är konverterat från dollarkurs till euro i november 2025. Scenarierna är:

1. Grundscenario, fast pris
 - Konstant energiförbrukning kombinerad med ett fast elavtal.
2. Grundscenario, spotpriser
 - Konstant energiförbrukning kombinerad med spotpriser timvis.
3. Solceller, fast pris
 - Fast elavtal där totala konsumtionen minskar med egenkonsumtion från solcellsdata för Helsingfors.
4. Solceller, spotpriser
 - Timvisa spotpriser kombinerade med egen solcellsproduktion. Överskott säljs till samma spotpris.
5. Solceller, batteri, spotpriser
 - Solcellsproduktion i kombination med batterilagring som tillåter förflyttning av egen produktion till dyrare timmar.

Scenarierna är simulerade med MATLAB-mjukvara. Koden utför räkningarna för alla scenarion, jämför dem och utför svepningar över storlek system både för solceller och batterier. Till slut ger den en rekommendation på optimal mängd solceller och batterikapacitet, om batterier blir billigare. Som spotprisdata har timpriser från webbplatsen Sähkömyyrä använts [9]. MATLAB-koden har utvecklats med hjälp av ChatGPT och den finns som bilaga i källorna.

Observera att inget scenario med fast elpris, solceller och batterier har analyserats, eftersom batterier är huvudsakligen ekonomiskt relevant med spotpriser.

4.1 Antaganden, formler och använda siffror

Från simulerade data användes endast ett år av simulerad PV-produktion, för att göra resten av uträkningarna lättare. Nettovärdeskostnaden NPC räknas med formeln:

$$NPC = C + A * DF$$
$$= 548 \frac{\text{€}}{\text{kWp}} + \sum(\text{energipris} \pm \text{försäljning} + \text{elöverföring} + O\&M) * 0,05$$

där:

- C = installationskostnader
- A = årliga driftskostnader (energiökop ± försäljning + elöverföring + O&M)
- D = diskonteringsfaktorn (för 10 år)

PV-installationskostnaderna beräknas med 548 €/kWp enligt Fraunhofer ISE [8, sidan 50].

Överföringsavgifter och elkostnader beräknas timvis för varje scenario.

Diskonteringsfaktorn är 5% för att representera hur värdet på pengar går ner och för att täcka uppehållskostnader.

En separat NPC-beräkning för frystorkningsutrustningen har också gjorts. Investeringen i frystorkare har uppskattats som 300 000 €, motsvarande tre IC550 frystorkare från Wave company á 100 000 € (effekt per enhet är 6,8 kW, sammanlagt ca 20 kW). Denna kapitalkostnad beräknas med en livslängd på 15 år och diskonteras över analysperioden, men påverkar inte återbetalningstiden för PV- och batterisystemet eftersom frystorkarna ingår i alla jämförda scenarier.

För de fem scenarierna räknar koden alltså ut:

- Årliga kostnader (energikostnader + O&M)
- Installationskostnader (PV, batterier, frystorkare ändast för total NPC)
- NPC på 10 år (med en diskontering på värde)
- Återbetalningstid för investeringarna i PV och batterier.

4.2 Beräkningar efter basscenarier

Utöver basscenierna utför koden:

- En svepning på PV 0–250 kW
- En svepning på batterikapaciteter 0–50 kWh
- En global optimering av NPC (värmekarta på NPC)
- Känslighetsanalys av elpriser med PV och PV med batteri
- Grafisk visualisering

4.3 Resultat

I figurerna nedan syns programmets utdata som kommer till konsolen. De printas ut om man kör den bifogade MATLAB-koden med PV data från första året i Helsingfors ("location 1") från de PV-simulerade data [10] och spotpris data från året 2024 [9].

```

--- SUMMARY OF ANNUAL COSTS & NPC ---

```

Case	AnnualCost_EUR	InstallCost_EUR	NPC_10yr_EUR	Payback_Years
{'Base: Const' }	18438	0	3.6555e+05	NaN
{'Base: Spot' }	14684	0	3.3656e+05	NaN
{'PV: Const' }	13260	54800	3.8037e+05	38.488
{'PV: Spot' }	10670	54800	3.6037e+05	13.654
{'PV + Battery' }	7506	57950	3.3909e+05	8.0738

Tabell 1. Årliga kostnader för samtliga scenarier. Årliga kostnader blir minst med kombinationen av solceller och batteri och högst med konstant elpris.

```

=====
OPTIMAL SYSTEM CONFIGURATION (NPC MIN)
=====

--- PV SYSTEM ---
Optimal PV capacity      : 25.0 kW
Annual cost              : 12589.49 €
10-year NPC              : 331661.23 €
Payback time             : 6.54 years

--- BATTERY SYSTEM ---
Optimal battery size     : 0.0 kWh
Annual cost              : 12570.34 €
10-year NPC              : 324237.24 €
Payback time             : NaN years

RECOMMENDED SYSTEM:
PV capacity = 25.00
Battery capacity = 0.0 kWh
Combined PV+Battery NPC must be evaluated using the heatmap.

```

Tabell 2. Optimala PV, batterier och rekommendation på system. Systemet rekommenderar inga batterier och 25 kW av solceller.

```

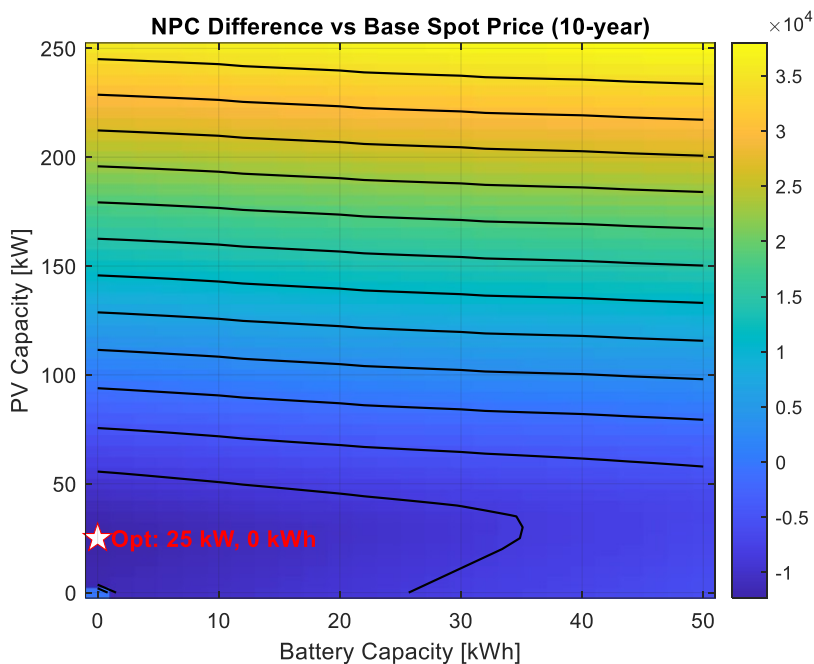
Computing global PV+Battery NPC heatmap...
Global optimum (heatmap): PV = 25.0 kW, Battery = 0.0 kWh, NPC diff = -12324.11 €
Running sensitivity analysis...
Sensitivity analysis done.
Done.

--- OPTIMUM SUMMARY ---

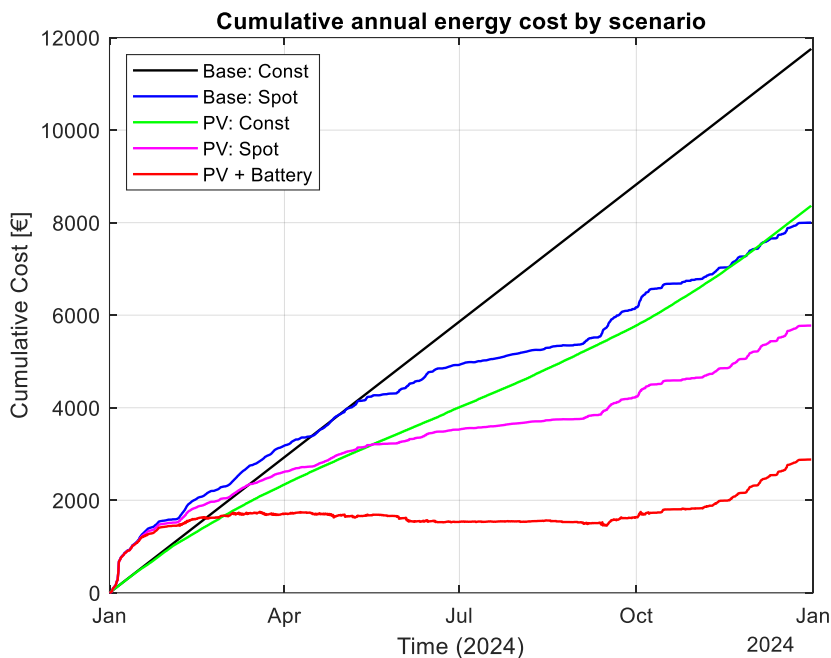
```

Case	PV_kW	Battery_kWh	AnnualCost	NPC_10yr	Payback_Years
"PV only"	25	0	12589	3.3166e+05	6.5423
"PV + Battery (sweep)"	25	0	12570	3.2424e+05	NaN
"Global Opt (heatmap)"	25	0	NaN	1.0106e+05	NaN

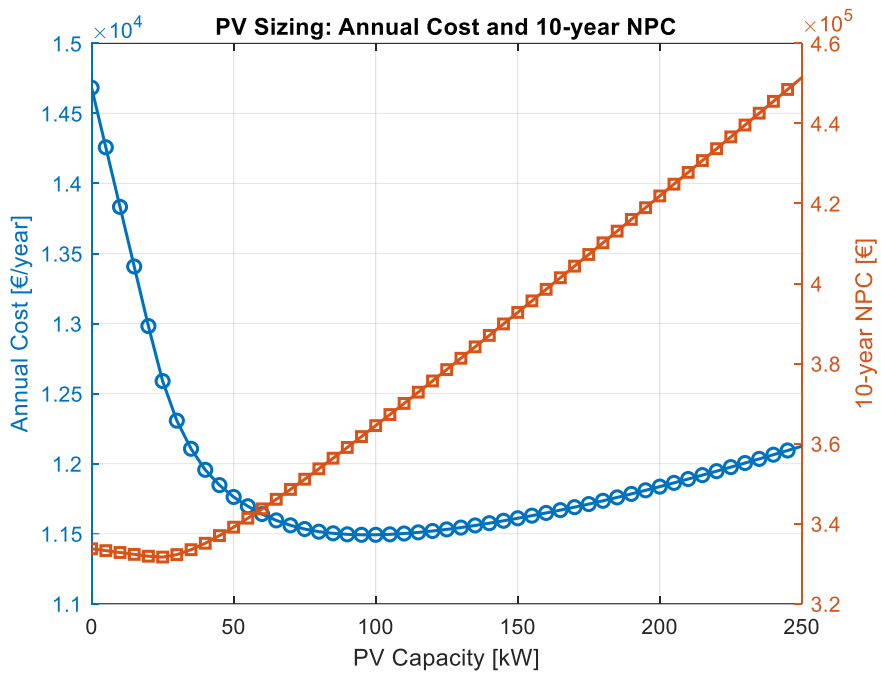
Tabell 3. Optimala storlekar på PV och batterier med beaktande på NPC. Swept för PV med batteri blir noll eftersom elpriserna inte fluktuerar tillräckligt och koden inte laddar batteriet från nätverket och säljer inte el vid höga elpriser.



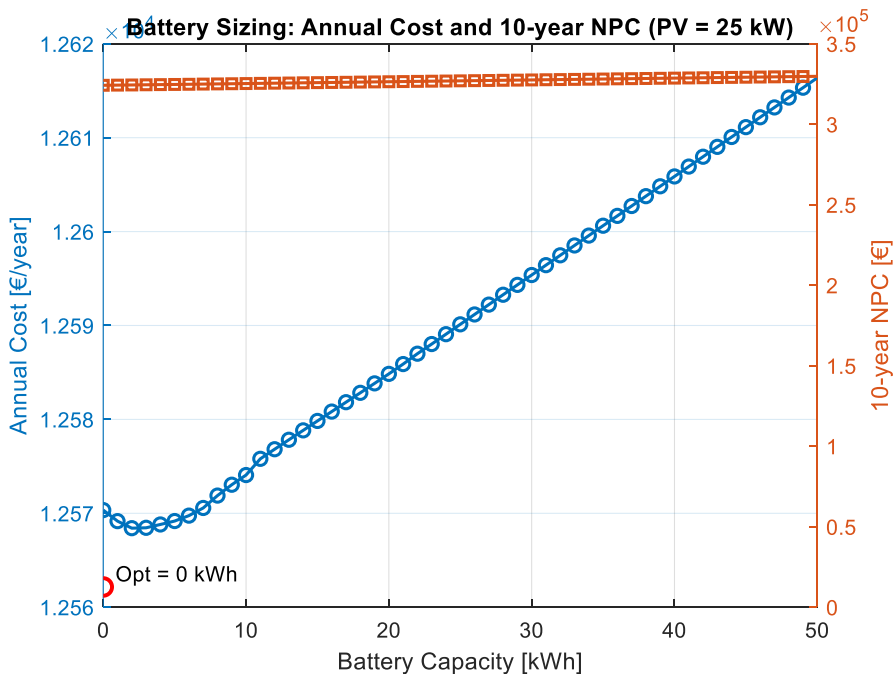
Figur 2. Värmekarta på NPC skillnad mot basscenario 1, stjärnan markerar optimal PV- och batteri-kapacitet. Blått indikerar lägre NPC och gult indikerar högre NPC.



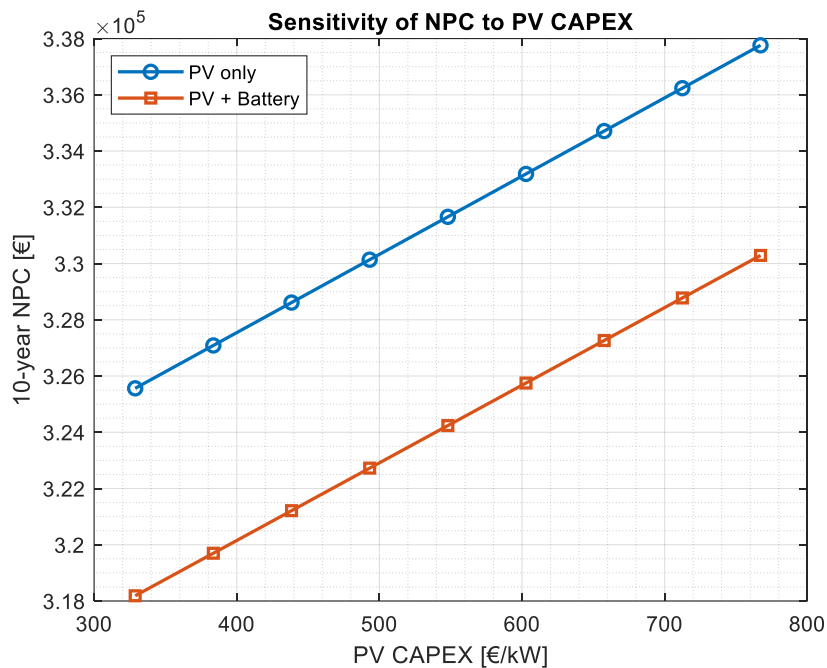
Figur 3. Kumulativa kostnader över ett år för alla fem scenarier. PV + batteri har tydligt mest energibesparingar, men grafen tar inte installationskostnader i beaktande.



Figur 4. PV-svep, illustrerar hur nuvärdeskostnaden går över årliga kostnader efter 60 kW kapacitet. Man ser också att det optimala NPC är vid 25 kWh (lägsta punkten på den röda kurvan).

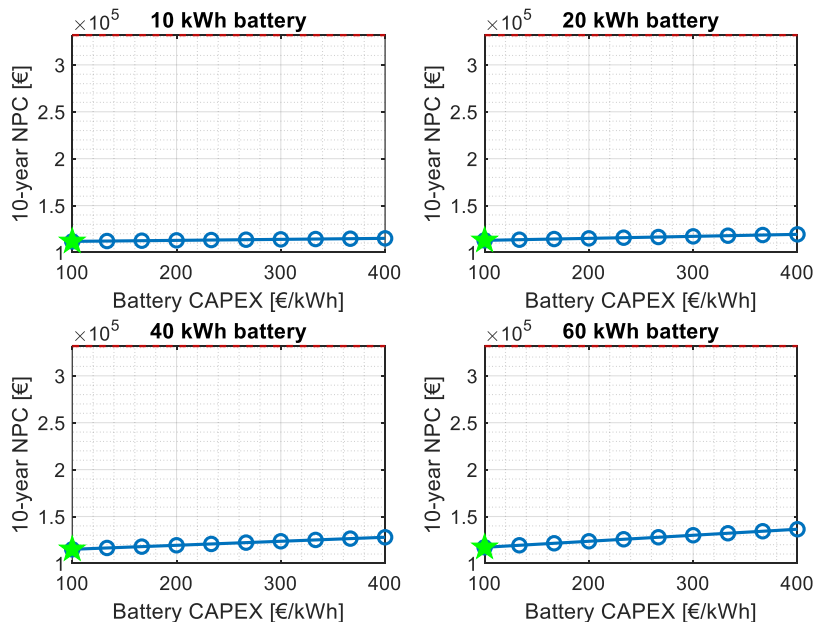


Figur 5. Illustrerar hur NPC stiger sakta (nästan konstant) och är lägst utan batterier. Optimum är markerat med röd halvcirkel.



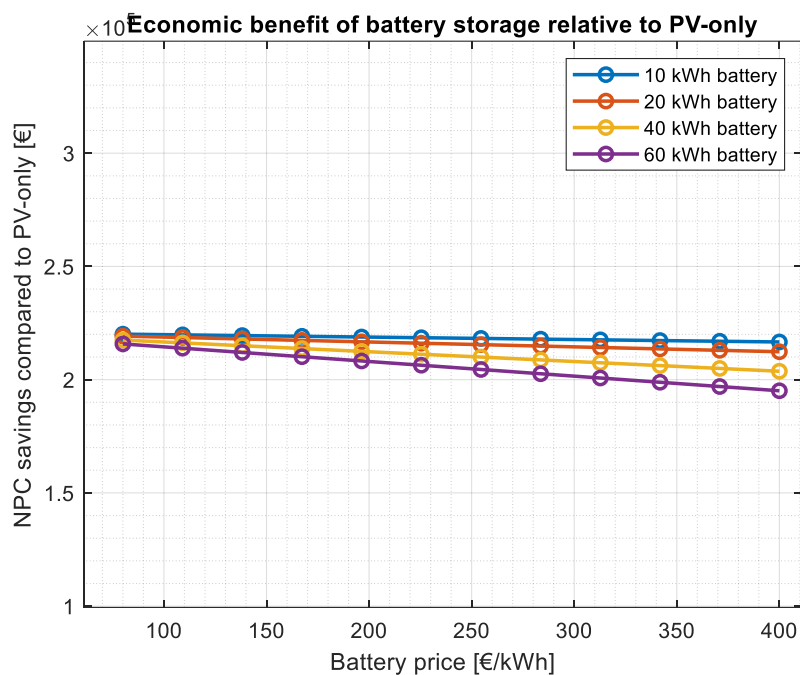
Figur 6. Sensitivitetsanalys på NPC jämfört med CAPEX. Pv med batterier har lägre NPC. Båda scenariernas NPC stiger linjärt med PV CAPEX.

NPC Sensitivity to Battery Price for Different Battery Sizes

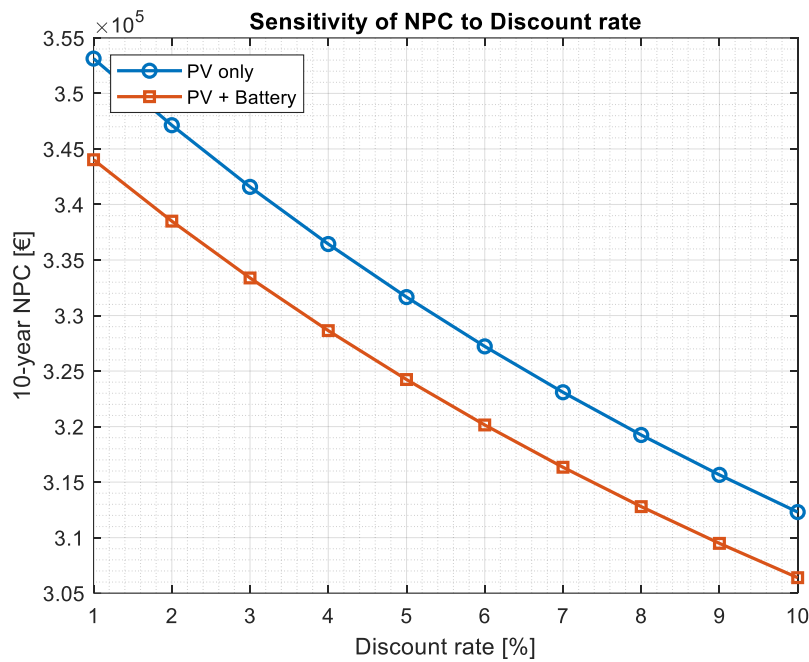


Figur 7. Sensitivitetsanalys över batteristorlekar från 10–60 kWh, vilka skulle motsvara drift utan elnätverket i 0,5-3 timmar den simulerade anläggningen (20 kW). NPC stiger tillsammans med batterikostnaden per kWh. Det bästa NPC nås med 0 kWh av batterier (igen). Den rödstreckade linjen är NPC för PV utan batterier, vilket tyder på att skillnader i batteripriserna inte

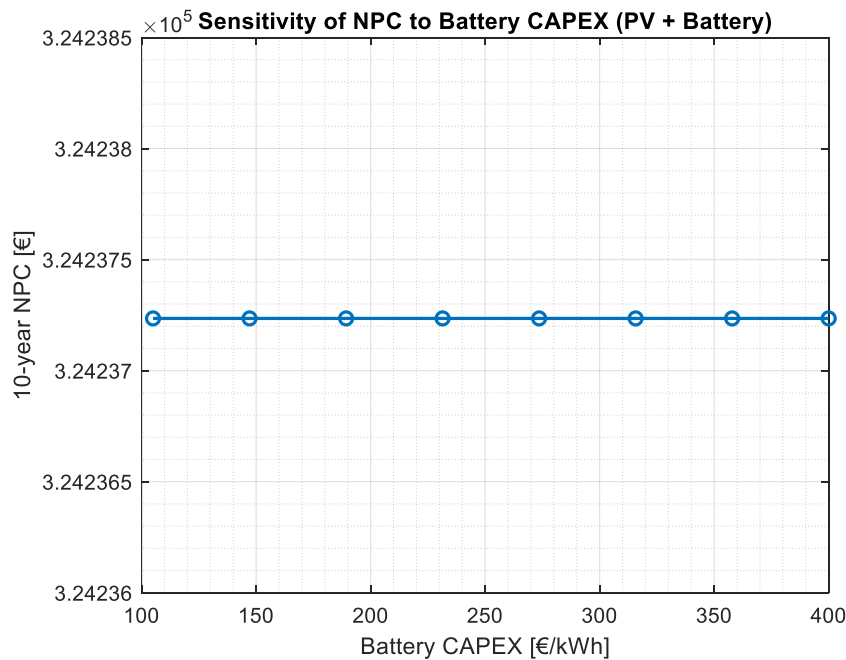
påverkar lönsamheten märkvärdigt. Den gröna stjärnan markera det aktuella priset 105 €/kWh. Graferna visar att de betydande vinsterna görs med PV, batterierna förbättrar den bara lite. Detta påverkas även av att simulationen inte laddar batterierna rakt ur elnätverket.



Figur 8. Illustrerar, tydligare än figur 7, hur mindre batteripris per kWh kan spara på kostnader uttryckt i skillnad mellan NPC med och utan batteri. För att batteri skulle vara lönsamt borde kurvorna nå y-axeln noll, men de gör det aldrig. Att tillsätta batterier ökar alltid på NPC.



Figur 9. Sensitivitet på 10-års-NPC och diskonträntan. Högre diskonträn­ta får nuvärdeskostnaderna för båda situationer att sjunka i en likadan kurva. Enligt bilden skulle PV med batteri vara lönsammare, men på grund av dy­rare initial investering blir tillbakabetalningstiden på PV med batteri längre, vilket gör PV med batteri till en mindre attraktiv investering på ett 10-årsper­spektiv.



Figur 10. Sensitivitet av NPC med batteri CAPEX. NPC hålls konstant fast batteri CPAEX stiger. Det beror på att koden tar 0 kWh optimum på batterier.

5 Slutsatser

Enligt data som programmet får, blir det för en anläggning med 25 kW belastning blir det lönsammaste systemet att investera i ett PV-system utan batteriförvaring med 25 kW av produktionskapacitet.

Man kan kortfattat säga att energiförbrukningen inte är en stor faktor för en frystorkningsanläggning på grund av billig el i Finland. Dessutom finns det många andra större kostnader och osäkerhetsvariabler som gör att investeringar lätt riktas annanstans än mot solceller och batterier.

Själva solcellerna tar ner på elpriset och energiförbrukningen och att installera solenergi anses för övrigt vara en grön gärning.

5.1 Vidare utveckling och potentiella förbättringar

För att göra koden mångsidigare och användbar för fler situationer, borde ett bättre utnyttjande av peak shaving implementeras, med förmåga att ladda batterierna från nätverket rakt och möjlighet att sälja ansamlad energi då elpriserna är höga. Koden har en färdig peak shave function inbyggd i "DISPATCH ENGINE"-blocket, men för att peak shaving skall vara lönsamt måste

Följande steg för att vidareutveckla arbetet skulle vara kodens anpassning till olika storlekar anläggningar, det vill säga ha en annan än 20 kW konstant belastning på systemet och möjlighet att svepa för olika belastningar. Andra utvecklingsområden skulle vara olika långa torkningscykler och varierande energikonsumtion. Koden går relativt lätt att köra med olika konstanta belastningar, användaren måste bara ändra på variablerna i variabel-blocket. Det är dock viktigt att man håller reda på installationspriser och andra kostnader som ändras med olika stora anläggningar.

Andra variabler som kunde svepas över mer skulle vara batteri-priser och storlekar. För att svara på frågor som: vid vilka priser blir batterier lönsamma att ha och vad händer om man har stor batterikapacitet.

Källor

Information om frystorkarnas energiförbruk har fåttts från datablad från företaget Wave samt från webbsidorna för företaget Havion Freeze Dry Technologies.

För industriella skalans batterier har priset 122 \$/kWh (ca 105€/kWh) tagits från forskning på elektriska bilars batteri-priser [7]. Beräkningen från dollarkurs till euro gjordes i november 2025.

Powerwall databladet är taget från energylibrary.tesla.com databasen, Duracell dura5 databladet är från duracell.hacdn.io databasen och Huawei LUNA2000-14 databladet är från webbsidan skesolar.com.

A. Informatörer och samlad data

Kaisa Lehtipuro från Nordic Freeze Dry har varit hjälpsam med att redogöra hur en frystorkningsanläggning opererar och hur torkningsprocessen och produktionen går till i praktiken.

Som hjälpredda med MATLAB-kodernas generering, rätt användning av syntax och klara kommentarer för att skilja på program-blocken har ChatGPT använts.

Spotpriserna baserar sig på data från The European Network of Transmission System Operators for Electricity (ENTSO-E). Sähkömyyrä webbsidans upprätthållare är företaget Nodesk. Ur data från Sähkömyyrä har problematiska icke-ASCII symboler så som å, ä och ö editerats bort för hand eller bytts ut för att minska problem i MATLAB-programvaran.

B. Källhänvisningar

- [1] H. Ritchie, P. Rosado, and M. Roser. Data Page: Solar photovoltaic module price [Online] Tillgänglig: <https://archive.ourworldindata.org/20250909-093708/grapher/solar-pv-prices.html>
- [2] *Technical Description Wave FD440*, Wave company, 2025. [Online]. Tillgänglig: <https://wave.cc/wp-content/uploads/2025/09/Technical-Description-FD440.pdf>.
- [3] "Freeze Dryer Electricity Usage & Running Costs." Havion Freeze Dry Technology. <https://havionfreezedry.com/freeze-dryer-electricity-usage-running-costs/> (accessed 23.11, 2025).
- [4] *LUNA2000-14*, Huawei, 2025. [Online]. Tillgänglig: https://solar.com/productdata/LUNA2000-S1/01_Datasheets/DataSheet_LUNA2000-7-14-21-S1_V01_2024-06_EN.pdf.
- [5] *Power Wall 2*, Tesla Motors, 2025. [Online]. Tillgänglig: <https://energylibrary.tesla.com/docs/Public/EnergyStorage/Powerwall/2/Datasheet/en-us/Powerwall-2-Datasheet.pdf>.
- [6] *Dura5 Stack or Wall-mount Battery*, Duracell Energy, 2025. [Online]. Tillgänglig: https://duracell.hacdn.io/media/documents/Dura5_Datasheet_V5.5_UK_web_PD.pdf.
- [7] X. Li *et al.*, "A cost-benefit analysis of V2G electric vehicles supporting peak shaving in Shanghai," *Electric Power Systems Research*, vol. 179, Table 3, s. 106058, 01.02. 2020, doi: <https://doi.org/10.1016/j.epsr.2019.106058>.
- [8] "Photovoltaics Report," Fraunhofer ISE, 2025. Accessed: 23.11.2025. [Online]. Tillgänglig: <https://www.ise.fraunhofer.de/content/dam/ise/de/documents/publications/studies/Photovoltaics-Report.pdf>
- [9] *Spot elpriser i Finland året 2024*, The European Network of Transmission System Operators for Electricity (ENTSO-E), 09.11.2025. [Online]. Tillgänglig: <https://www.sahkomyyra.fi/historia>
- [10] J. Ekström, M. Koivisto, I. Mellin, R. J. Millar, and M. Lehtonen, "A Statistical Model for Hourly Large-Scale Wind and Photovoltaic Generation in New Locations," *IEEE Transactions on Sustainable Energy*, vol. 8, no. 4, s. 1383–1393, 2017, doi: 10.1109/TSTE.2017.2682338.

C. MATLAB-koden som bilaga

```
%% =====
% compare_all_final_full.m
% Fully integrated PV + battery + sweeps + NPC + payback
%% =====

clear; clc; close all;

%% -----
% USER PARAMETERS
%% -----
C = 20; % kWh per hour constant consumption
set_price = 0.0671; % €/kWh fixed electricity price
loc_idx = 1; % PV location index (1 = Helsinki)
year_idx = 1; % which simulated year to use from .mat
nHours = 8760; % use 8760 hours (trim leap year if
needed)

% Transfer (siirtomaksu)
transfer_var = 0.0354; % €/kWh variable transfer fee
transfer_fixed_month = 40; % €/month fixed fee assumption
transfer_fixed_annual = transfer_fixed_month * 12;

% Discount / lifetime
years = 10; % horizon for NPC
discount_rate = 0.05; % used for NPC and annualisation

% Freeze dryer CAPEX (only in NPC, not in payback)
freeze_dryer_capex = 300000; % € 3 IC550 units to be close to 20
kW load
freeze_dryer_lifetime = 15; % years (typical industrial freeze
dryer)

% Battery defaults (used for single-run scenarios)
battery_default.capacity = 30; % kWh
battery_default.eff = 0.90; % round-trip efficiency
battery_default.Pcharge = 10; % kW
battery_default.Pdischarge = 10; % kW

% Installation cost assumptions
pv_cost_per_kW = 548; % €/kW (548 €/kWp) as per Fraunho-
fer source
pv_reference_kW = 100; % PV power data is based on this
pv_reference_cost = pv_reference_kW * pv_cost_per_kW; % € for 100 kW
system

battery_cost_per_kwh = 105; % €/kWh
battery_default_cost = battery_default.capacity * battery_cost_per_kwh;

pv_scale_default = 1; % default scale 100 kW / 100 = 1
pv_cost = pv_reference_cost;

install_costs = [
```

```

0;
0;
pv_cost;
pv_cost;
pv_cost + battery_default_cost
];

% Case labels used in summary + plots
CaseNames = {
    'Base: Const'
    'Base: Spot'
    'PV: Const'
    'PV: Spot'
    'PV + Battery'
};

% PV system sizing (kW)
pv_sizes_kW = 0:5:250; % realistic system sizes (from 0-250 kW with 5kW
interwalls)

% Battery sizing (kWh)
battery_sizes = 0:1:50;

% Degradation assumptions
pv_degradation = 0.005; % 0.5% per year
battery_degradation = 0.02; % 2% per year

% O&M costs (operation and maintenance)
pv_om_rate = 0.015; % 1.5% of PV CAPEX per year
battery_om_rate = 0.01; % 1% of battery CAPEX per year

% System lifetime
pv_lifetime = 25;
battery_lifetime = 15; % a bit optimistic lifetimes

% Feature toggles
enable_heatmap = true; % Global PV+Battery NPC heatmap
enable_sensitivity = true; % Parameter sensitivity analysis

%% -----
% BATTERY CONTROL FEATURES
%% -----
battery_ctrl.grid_charge = true; % allow charging from grid
battery_ctrl.peak_shave = 20; % kW grid import limit (0 = disables)

% CSV file name
spot_csv_filename = 'chart2.csv';

%% -----
% LOAD SPOT PRICE CSV (robust)
%% -----
fprintf('Loading spot price CSV: %s\n', spot_csv_filename);
raw = readtable(spot_csv_filename, 'ReadVariableNames', false, 'Delimiter', ',');

```

```

priceStr = string(raw.Var2);
priceStr = strrep(priceStr,',','.');
priceNum = str2double(priceStr);
priceNum = priceNum(1:nHours); % trim to 8760 hours
priceNum(isnan(priceNum)) = 0;
price_spot = priceNum / 100; % €/kWh
t_hour = datetime(2024,1,1,0,0,0) + hours(0:nHours-1);

%% -----
% LOAD PV DATA
%% -----
matfile_name = 'simulated_solar_power_Helsinki+3locs_100years.mat';
fprintf('Loading PV data: %s\n', matfile_name);
S = load(matfile_name);
if ~isfield(S,'SOLAR_POWER_kW')
    error('Expected variable SOLAR_POWER_kW in %s', matfile_name);
end

SOLAR_POWER_kW = S.SOLAR_POWER_kW;
pv = SOLAR_POWER_kW(:, loc_idx, year_idx);
pv = pv(1:nHours);
pv_reference_output = pv; % kW for 100 kW plant

%% -----
% RUN BASE CASES & PV CASES
%% -----
fprintf('Computing base & PV scenarios...\n');

% Base constant price
energy_cost_base_set = C * set_price * nHours;
grid_import_base_set = C * nHours;
transfer_cost_base_set = grid_import_base_set * transfer_var + transfer_fixed_annual;
total_base_set = energy_cost_base_set + transfer_cost_base_set;

% Base spot price
energy_cost_base_spot = sum(C .* price_spot);
grid_import_base_spot = C * nHours;
transfer_cost_base_spot = grid_import_base_spot * transfer_var + transfer_fixed_annual;
total_base_spot = energy_cost_base_spot + transfer_cost_base_spot;

% PV constant price
pv_used_case3 = min(pv, C);
grid_import_case3 = sum(C - pv_used_case3);
energy_cost_pv_set = grid_import_case3 * set_price;
transfer_cost_pv_set = grid_import_case3 * transfer_var + transfer_fixed_annual;
total_pv_set = energy_cost_pv_set + transfer_cost_pv_set;

% PV spot price
pv_used_case4 = min(pv, C);
grid_import_case4 = sum(C - pv_used_case4);
energy_cost_pv_spot = sum((C - pv_used_case4) .* price_spot);

```

```

transfer_cost_pv_spot = grid_import_case4 * transfer_var + trans-
fer_fixed_annual;
total_pv_spot = energy_cost_pv_spot + transfer_cost_pv_spot;

% PV + battery
[energy_cost_pv_batt, grid_import_case5, grid_export_case5,
hourly_cost_batt, battery_energy] = ...
    pv_with_battery_full(C, price_spot, pv, ...
        struct('capacity', battery_default.capacity, ...
            'eff', battery_default.eff, ...
            'Pcharge', battery_default.Pcharge, ...
            'Pdischarge', battery_default.Pdischarge, ...
            'grid_charge', battery_ctrl.grid_charge, ...
            'peak_shave', battery_ctrl.peak_shave));

transfer_cost_pv_batt = grid_import_case5 * transfer_var + trans-
fer_fixed_annual;
total_pv_batt = energy_cost_pv_batt + transfer_cost_pv_batt;

%% -----
% COLLECT ANNUAL COSTS
%% -----
annual_costs = [total_base_set;
                total_base_spot;
                total_pv_set;
                total_pv_spot;
                total_pv_batt];

%% -----
% 10-YEAR NPC
%% -----
discount_factor = (1 - (1 + discount_rate)^(-years)) / discount_rate;

% Freeze dryer annualised CAPEX
r_fd = discount_rate;
n_fd = freeze_dryer_lifetime;
CRF_fd = (r_fd * (1 + r_fd)^n_fd) / ((1 + r_fd)^n_fd - 1);
freeze_dryer_annual_cost = freeze_dryer_capex * CRF_fd;

% Present value over 10 years
freeze_dryer_NPC = freeze_dryer_annual_cost * discount_factor;

% Total NPC per scenario
NPC_10yr = install_costs + annual_costs .* discount_factor +
freeze_dryer_NPC;

%% -----
% PAYBACK CALCULATION
%% -----
baseline_annual = total_base_spot;
payback_years = NaN(size(annual_costs));

for i = 1:length(annual_costs)
    if install_costs(i)==0 || i <= 2
        continue;
    end
end

```

```

end
annual_savings = baseline_annual - annual_costs(i);
if annual_savings > 0
    payback_years(i) = install_costs(i) / annual_savings;
end
end

%% -----
% SUMMARY TABLE (CLEAN)
%% -----
T = table( CaseNames, ...
          annual_costs(:), ...
          install_costs(:), ...
          NPC_10yr(:), ...
          payback_years(:), ...
          'VariableNames',{'Case','AnnualCost_EUR','In-
stallCost_EUR','NPC_10yr_EUR','Payback_Years'});

disp('--- SUMMARY OF ANNUAL COSTS & NPC ---');
disp(T);

%% -----
% PV SWEEP
%% -----
nPV = numel(pv_sizes_kW);
pv_sweep_annual = zeros(nPV,1);
pv_sweep_NPC10 = zeros(nPV,1);
pv_sweep_payback = zeros(nPV,1);

for i = 1:nPV

    pv_capacity_kW = pv_sizes_kW(i);
    pv_scale      = pv_capacity_kW / pv_reference_kW;
    pv_scaled     = pv_reference_output * pv_scale;

    pv_used      = min(pv_scaled, C);
    grid_import  = sum(C - pv_used);
    energy_cost  = sum((C - pv_used) .* price_spot);
    transfer_cost = grid_import * transfer_var + transfer_fixed_annual;

    % PV CAPEX + O&M
    pv_capex = pv_capacity_kW * pv_cost_per_kW;
    pv_om    = pv_capex * pv_om_rate;

    annual_cost = energy_cost + transfer_cost + pv_om;
    pv_sweep_annual(i) = annual_cost;

    % Degradation midpoint factor
    degraded_output_factor = 1 - pv_degradation * (years/2);

    % NPC INCLUDING FREEZE DRYER
    pv_sweep_NPC10(i) = pv_capex + ...
                        annual_cost * degraded_output_factor * dis-
count_factor + ...
                        freeze_dryer_NPC;

```

```

    % Payback
    savings = total_base_spot - annual_cost;
    if savings > 0
        pv_sweep_payback(i) = pv_capex / savings;
    else
        pv_sweep_payback(i) = NaN;
    end
end

%% -----
% PV SWEEP TABLE + OPTIMUM
%% -----
PV_sweep = table( ...
    pv_sizes_kw(:), ...
    pv_sweep_annual(:), ...
    pv_sweep_NPC10(:), ...
    pv_sweep_payback(:), ...
    'VariableNames', {'PV_kW', 'AnnualCost', 'NPC10', 'Payback'});

% PV-only optimum (for fixed-PV battery sweep)
[~, optPV_idx] = min(pv_sweep_NPC10);

optimal_pv      = pv_sizes_kw(optPV_idx);
optimal_pv_npc  = pv_sweep_NPC10(optPV_idx);
optimal_pv_annual = pv_sweep_annual(optPV_idx);
optimal_pv_payback = pv_sweep_payback(optPV_idx);

%% =====
% BATTERY SWEEP USING OPTIMAL PV SIZE FROM PV SWEEP
%% =====

% Fixed PV size = optimal PV from PV sweep (in kW)
fixed_pv_kw      = optimal_pv; % [kW]
pv_fixed_scale   = fixed_pv_kw / pv_reference_kw;
pv_fixed_profile = pv_reference_output * pv_fixed_scale;

% PV CAPEX + O&M (fixed for this battery sweep)
pv_fixed_capex = fixed_pv_kw * pv_cost_per_kw;
pv_fixed_om    = pv_fixed_capex * pv_om_rate;

% Degradation factor for NPC (battery-driven, mid-point over 10 years)
deg_factor_batt = 1 - battery_degradation * (years / 2);

% --- Baseline: PV-only case with this fixed PV size, no battery ---
[energy_cost_pv_only, grid_import_pv_only, ~, ~, ~] = ...
    pv_with_battery_full(C, price_spot, pv_fixed_profile, ...
        struct('capacity', 0, ... % no battery
            'eff', battery_default.eff, ...
            'Pcharge', battery_default.Pcharge, ...
            'Pdischarge', battery_default.Pdischarge, ...
            'grid_charge', battery_ctrl.grid_charge, ...
            'peak_shave', battery_ctrl.peak_shave));

```

```

transfer_cost_pv_only = grid_import_pv_only * transfer_var + transfer_fixed_annual;
annual_cost_pv_only = energy_cost_pv_only + transfer_cost_pv_only + pv_fixed_om;
NPC10_pv_only_for_batts = freeze_dryer_NPC + pv_fixed_capex + ...
                        annual_cost_pv_only * deg_factor_batt * discount_factor;

% --- Battery sweep arrays ---
nBatt = numel(battery_sizes);
batt_sweep_annual = zeros(nBatt,1);
batt_sweep_NPC10 = zeros(nBatt,1);
batt_sweep_payback = zeros(nBatt,1);

for i = 1:nBatt

    batt_capacity = battery_sizes(i);           % [kWh]

    % Battery CAPEX & O&M
    battery_capex = batt_capacity * battery_cost_per_kwh;
    battery_om = battery_capex * battery_om_rate;

    % Dispatch with fixed PV + THIS battery size
    [energy_cost, grid_import, grid_export, hourly_cost, ~] = ...
        pv_with_battery_full(C, price_spot, pv_fixed_profile, ...
            struct('capacity', batt_capacity, ...
                  'eff', battery_default.eff, ...
                  'Pcharge', battery_default.Pcharge, ...
                  'Pdischarge', battery_default.Pdischarge, ...
                  'grid_charge', battery_ctrl.grid_charge, ...
                  'peak_shave', battery_ctrl.peak_shave));

    transfer_cost = grid_import * transfer_var + transfer_fixed_annual;

    % Annual operating cost = energy + transfer + PV O&M + battery O&M
    annual_cost = energy_cost + transfer_cost + pv_fixed_om + battery_om;
    batt_sweep_annual(i) = annual_cost;

    % 10-year NPC = freeze dryer + PV CAPEX + battery CAPEX + discounted
    annual costs
    batt_sweep_NPC10(i) = freeze_dryer_NPC + pv_fixed_capex + battery_capex + ...
                        annual_cost * deg_factor_batt * discount_factor;

    % Payback of battery *only* vs PV-only case
    annual_savings_vs_pv_only = annual_cost_pv_only - annual_cost;
    if annual_savings_vs_pv_only > 0 && batt_capacity > 0
        batt_sweep_payback(i) = battery_capex / annual_savings_vs_pv_only;
    else
        batt_sweep_payback(i) = NaN;
    end
end
end

```

```

%% -----
% REBUILD BATTERY SWEEP TABLE
%% -----
BATT_sweep = table( ...
    battery_sizes(:), ...
    batt_sweep_annual(:), ...
    batt_sweep_NPC10(:), ...
    batt_sweep_payback(:), ...
    'VariableNames', {'Battery_kWh', 'AnnualCost', 'NPC10', 'Payback'});

[~, optBatt_idx] = min(batt_sweep_NPC10);
optimal_batt     = battery_sizes(optBatt_idx);
optimal_batt_npc = batt_sweep_NPC10(optBatt_idx);
optimal_batt_annual = batt_sweep_annual(optBatt_idx);
optimal_batt_payback = batt_sweep_payback(optBatt_idx);

%% =====
% OPTIMAL SYSTEM SELECTOR (RESTORED & FIXED)
%% =====

disp('=====');
disp('OPTIMAL SYSTEM CONFIGURATION (NPC MIN)');
disp('=====');

fprintf('\n--- PV SYSTEM ---\n');
fprintf('Optimal PV capacity      : %.1f kW\n', optimal_pv);
fprintf('Annual cost                 : %.2f €\n', optimal_pv_annual);
fprintf('10-year NPC                  : %.2f €\n', optimal_pv_npc);
fprintf('Payback time                 : %.2f years\n', optimal_pv_payback);

fprintf('\n--- BATTERY SYSTEM ---\n');
fprintf('Optimal battery size       : %.1f kWh\n', optimal_batt);
fprintf('Annual cost                : %.2f €\n', optimal_batt_annual);
fprintf('10-year NPC                : %.2f €\n', optimal_batt_npc);
fprintf('Payback time               : %.2f years\n', optimal_batt_payback);

fprintf('\n RECOMMENDED SYSTEM:\n');
fprintf('PV capacity = %.2f\n', optimal_pv);
fprintf('Battery capacity = %.1f kWh\n', optimal_batt);
fprintf('Combined PV+Battery NPC must be evaluated using the heatmap.\n\n');

%% =====
% GLOBAL PV + BATTERY NPC HEATMAP (relative to base case)
%% =====
if enable_heatmap

    fprintf('Computing global PV+Battery NPC heatmap...\n');

    % High-resolution grids
    pv_grid_kw = 0:5:250;
    batt_grid_kWh = 0:2:50;

    nPVg = numel(pv_grid_kw);
    nBg = numel(batt_grid_kWh);

```

```

% NPC maps: ROWS = PV, COLS = BATTERY
NPC_map      = zeros(nPVg, nBg);
NPC_diff_map = zeros(nPVg, nBg);

% Base case NPC (spot, no PV, no battery)
NPC_base_spot = total_base_spot * discount_factor;

% Midpoint degradation factor
deg_factor_heat = 1 - battery_degradation * (years/2);

hwb = waitbar(0, 'Computing PV+Battery NPC map...');
counter = 0; total_runs = nPVg * nBg;

for ip = 1:nPVg

    pv_cap_kW = pv_grid_kW(ip);

    % Scale PV profile
    if pv_cap_kW == 0
        pv_profile = zeros(size(pv_reference_output));
    else
        pv_profile = pv_reference_output * (pv_cap_kW / pv_refer-
ence_kW);
    end

    capex_pv = pv_cap_kW * pv_cost_per_kW;
    om_pv     = capex_pv * pv_om_rate;

    for jb = 1:nBg
        counter = counter + 1;
        if mod(counter,50)==0
            waitbar(counter/total_runs, hwb);
        end

        batt_cap_kWh = batt_grid_kWh(jb);
        capex_batt    = batt_cap_kWh * battery_cost_per_kWh;
        om_batt       = capex_batt * battery_om_rate;

        if pv_cap_kW == 0 && batt_cap_kWh == 0
            % Base case NPC
            NPC_here = NPC_base_spot;
        else
            % Run dispatch with THIS PV + THIS battery
            [energy_cost, grid_import, grid_export, hourly_cost, ~] =
...
                pv_with_battery_full(C, price_spot, pv_profile, ...
                struct('capacity', batt_cap_kWh, ...
                    'eff', battery_default.eff, ...
                    'Pcharge', battery_default.Pcharge,
...
                    'Pdischarge', battery_default.Pdischarge,
...

```

```

...
        'grid_charge', battery_ctrl.grid_charge,
        'peak_shave', battery_ctrl.peak_shave));

    transfer_cost = grid_import * transfer_var + transfer_fixed_annual;
    annual_cost   = energy_cost + transfer_cost + om_pv + om_batt;

    % 10-year NPC (PV + Battery)
    NPC_here = capex_pv + capex_batt + annual_cost * deg_factor_heat * discount_factor;
end

    NPC_map(ip,jb)      = NPC_here;
    NPC_diff_map(ip,jb) = NPC_here - NPC_base_spot;
end
end
close(hwb);

% ---- GLOBAL OPTIMUM ----
[min_val, idx_flat] = min(NPC_diff_map(:));
[ip_min, jb_min] = ind2sub(size(NPC_diff_map), idx_flat);
pv_opt_global     = pv_grid_kw(ip_min);
batt_opt_global   = batt_grid_kWh(jb_min);

fprintf('Global optimum (heatmap): PV = %.1f kW, Battery = %.1f kWh, NPC diff = %.2f €\n',...
        pv_opt_global, batt_opt_global, min_val);

% ---- HEATMAP PLOT ----
figure('Name','NPC Difference Heatmap','NumberTitle','off','Color','w');

imagesc(batt_grid_kWh, pv_grid_kw, NPC_diff_map);
set(gca,'YDir','normal');
colormap(parula(256));
colorbar;

xlabel('Battery Capacity [kWh]');
ylabel('PV Capacity [kW]');
title('NPC Difference vs Base Spot Price (10-year)');
grid on;

hold on;

% Proper meshgrid for contour overlay
[BG, PVG] = meshgrid(batt_grid_kWh, pv_grid_kw);

contour(BG, PVG, NPC_diff_map, 12, 'k','LineWidth',1);

plot(batt_opt_global, pv_opt_global, 'rp', 'MarkerSize', 14, 'MarkerFaceColor','w');
text(batt_opt_global, pv_opt_global,...

```

```

        sprintf(' Opt: %.0f kW, %.0f kWh', pv_opt_global,
batt_opt_global),...
        'Color','r','FontSize',11,'FontWeight','bold');

end

%% =====
% SENSITIVITY ANALYSIS (PV-only vs PV+Battery)
%% =====
if enable_sensitivity

    fprintf('Running sensitivity analysis...\n');

    % -----
    % Base decomposition (current parameters)
    % -----

    % Optimal PV (from PV sweep)
    pv_opt_cap      = optimal_pv;                % [kW]
    pv_opt_capex    = pv_opt_cap * pv_cost_per_kw; % € CAPEX
    pv_opt_om       = pv_opt_capex * pv_om_rate;  % € / year O&M

    % annual cost = energy + transfer + pv_om (no freeze dryer here)
    % => energy + transfer part:
    energy_transfer_pv_opt = optimal_pv_annual - pv_opt_om;

    % Optimal battery (from battery sweep using fixed PV)
    batt_opt_cap    = optimal_batt;              % [kWh]
    batt_opt_capex  = batt_opt_cap * battery_cost_per_kwh; % € CAPEX
    batt_opt_om     = batt_opt_capex * battery_om_rate;  % € / year O&M

    % For PV+Battery: annual_cost = energy+transfer + pv_om + batt_om
    energy_transfer_pvbatt = optimal_batt_annual - pv_opt_om -
batt_opt_om;

    % Degradation factors (midpoint over 10 years)
    deg_pv   = 1 - pv_degradation * (years / 2);
    deg_batt = 1 - battery_degradation * (years / 2);

    % -----
    % Sensitivity A: Discount rate
    % -----
    dr_vec      = linspace(0.01, 0.10, 10);
    npc_pv_dr   = zeros(numel(dr_vec),1);
    npc_pvbatt_dr = zeros(numel(dr_vec),1);

    for k = 1:numel(dr_vec)
        r = dr_vec(k);
        df = (1 - (1+r)^(-years)) / r;

        % PV-only NPC(r)
        annual_pv_r = (energy_transfer_pv_opt + pv_opt_om); % same cost
structure, scaled via df & deg
        npc_pv_dr(k) = freeze_dryer_NPC + pv_opt_capex + ...
            annual_pv_r * deg_pv * df;
    end
end

```

```

    % PV + Battery NPC(r)
    annual_pvbatt_r = (energy_transfer_pvbatt + pv_opt_om +
batt_opt_om);
    npc_pvbatt_dr(k) = freeze_dryer_NPC + pv_opt_capex +
batt_opt_capex + ...
                                annual_pvbatt_r * deg_batt * df;
end

figure('Name','Sensitivity: Discount rate','NumberTitle','off','Col-
or','w');
plot(dr_vec*100, npc_pv_dr, '-o', 'LineWidth',1.5); hold on;
plot(dr_vec*100, npc_pvbatt_dr, '-s', 'LineWidth',1.5);
xlabel('Discount rate [%]');
ylabel('10-year NPC [€]');
legend({'PV only', 'PV + Battery'}, 'Location', 'northwest');
title('Sensitivity of NPC to Discount rate');
grid on; grid minor;

% -----
% Sensitivity B: PV CAPEX (€/kW)
% -----
pv_cost_vec      = linspace(0.6*pv_cost_per_kW, 1.4*pv_cost_per_kW,
9);
npc_pv_pvcost    = zeros(numel(pv_cost_vec),1);
npc_pvbatt_pvcost = zeros(numel(pv_cost_vec),1);

% Split annual costs cleanly:
% PV-only:    optimal_pv_annual = energy+transfer + pv_opt_om
% PV+Battery: optimal_batt_annual = energy+transfer + pv_opt_om +
batt_opt_om
et_pv      = energy_transfer_pv_opt;      % energy + transfer part
(PV only)
et_pvbatt = energy_transfer_pvbatt;      % energy + transfer part
(PV+Battery)

for k = 1:numel(pv_cost_vec)
    pv_cost_scan = pv_cost_vec(k);

    % --- PV-only ---
    capex_pv_scan = pv_opt_cap * pv_cost_scan;
    om_pv_scan    = capex_pv_scan * pv_om_rate;
    annual_pv_scan = et_pv + om_pv_scan;

    npc_pv_pvcost(k) = freeze_dryer_NPC + capex_pv_scan + ...
                                annual_pv_scan * deg_pv * discount_factor;

    % --- PV + Battery ---
    capex_pv_b_scan = pv_opt_cap * pv_cost_scan;
    om_pv_b_scan    = capex_pv_b_scan * pv_om_rate;
    annual_pvbatt_scan = et_pvbatt + om_pv_b_scan + batt_opt_om;

    npc_pvbatt_pvcost(k) = freeze_dryer_NPC + capex_pv_b_scan +
batt_opt_capex + ...

```

```

                                annual_pvbatt_scan * deg_batt * dis-
count_factor;
    end

    figure('Name','Sensitivity: PV cost','NumberTitle','off','Col-
or','w');
    plot(pv_cost_vec, npc_pv_pvcost, '-o', 'LineWidth',1.5); hold on;
    plot(pv_cost_vec, npc_pvbatt_pvcost, '-s', 'LineWidth',1.5);
    xlabel('PV CAPEX [€/kW]');
    ylabel('10-year NPC [€]');
    legend({'PV only', 'PV + Battery'}, 'Location', 'northwest');
    title('Sensitivity of NPC to PV CAPEX');
    grid on; grid minor;

% -----
% Sensitivity C: Battery CAPEX (€/kWh)
% -----
batt_cost_vec      = linspace(105, 400, 8);
npc_pvbatt_battcost = zeros(numel(batt_cost_vec),1);

% For battery CAPEX scan, PV part stays at base pv_cost_per_kW
for k = 1:numel(batt_cost_vec)
    bcost = batt_cost_vec(k);

    capex_batt_scan = batt_opt_cap * bcost;
    om_batt_scan    = capex_batt_scan * battery_om_rate;

    % Annual cost with scanned battery CAPEX (O&M depends on CAPEX)
    annual_pvbatt_scan = et_pvbatt + pv_opt_om + om_batt_scan;

    npc_pvbatt_battcost(k) = freeze_dryer_NPC + pv_opt_capex +
capex_batt_scan + ...
                                annual_pvbatt_scan * deg_batt * dis-
count_factor;
    end

    figure('Name','Sensitivity: Battery cost','NumberTitle','off','Col-
or','w');
    plot(batt_cost_vec, npc_pvbatt_battcost, '-o', 'LineWidth',1.5);
    xlabel('Battery CAPEX [€/kWh]');
    ylabel('10-year NPC [€]');
    title('Sensitivity of NPC to Battery CAPEX (PV + Battery)');
    grid on; grid minor;

%% ----- Sensitivity D: Battery CAPEX & Size Comparison -----
--
% Fixed battery sizes to evaluate (relative to 20 kW load)
batt_test_sizes = [10 20 40 60];           % kWh
batt_cost_vec   = linspace(100, 400, 10); % €/kWh range

npc_sweep_size = zeros(numel(batt_cost_vec), numel(batt_test_sizes));

for s = 1:numel(batt_test_sizes)
    B = batt_test_sizes(s);

```

```

% CAPEX + O&M baseline for battery size B
% degradation factor MUST be deg_batt (as defined previously)
if B > 0
    deg_batt = 1 - battery_degradation * (years / 2);
else
    deg_batt = 1;
end

for k = 1:numel(batt_cost_vec)
    bcost          = batt_cost_vec(k);
    capex_batt_scan = B * bcost;
    om_batt_scan   = capex_batt_scan * battery_om_rate;

    npc_sweep_size(k,s) = pv_opt_capex + capex_batt_scan + ...
        (energy_transfer_pvbatt + pv_opt_om) * discount_factor +
...
        (om_batt_scan * deg_batt * discount_factor);
end
end

% ----- 2x2 subplot visualisation -----
figure('Name','Battery Sensitivity - Size Comparison','NumberTitle','off','Color','w');

for s = 1:numel(batt_test_sizes)
    subplot(2,2,s);
    plot(batt_cost_vec, npc_sweep_size(:,s),'-o','LineWidth',1.4);
    title(sprintf('%d kWh battery', batt_test_sizes(s)));
    xlabel('Battery CAPEX [€/kWh]');
    ylabel('10-year NPC [€]');
    grid on; grid minor;

    % Optional baseline reference line
    hold on; yline(optimal_pv_npc,'r--','LineWidth',1.1);
    % Check if/where battery beats PV-only NPC
    idx = find(npc_sweep_size(:,s) < optimal_pv_npc, 1, 'first');

    if ~isempty(idx)
        % highlight breakeven point
        plot(batt_cost_vec(idx), npc_sweep_size(idx,s),'gp','MarkerSize',12,'MarkerFaceColor','g');
        %text(batt_cost_vec(idx), npc_sweep_size(idx,s), ' breakeven','Color','g','FontSize',10);
    end
end

sgtitle('NPC Sensitivity to Battery Price for Different Battery Sizes');

fprintf('Sensitivity analysis done.\n');
end

%% =====
%% NPC DIFFERENCE PLOT - Savings of PV+battery vs PV-only
%% For multiple battery sizes and varying battery CAPEX

```

```

%% =====

battery_test_sizes = [10 20 40 60];      % kWh to compare visually
batt_cost_vec = linspace(80,400,12);     % €/kWh range to test
nb = numel(battery_test_sizes);
nc = numel(batt_cost_vec);

NPC_diff_matrix = zeros(nb,nc); % (NPC_PVonly - NPC_PV+batt)

for bi = 1:nb
    batt_size = battery_test_sizes(bi);

    for ci = 1:nc
        bcost = batt_cost_vec(ci);

        % System CAPEX & O&M
        capex_batt = batt_size * bcost;
        om_batt    = capex_batt * battery_om_rate;

        % Run dispatch using optimal PV
        [energy_cost_b, grid_imp_b, ~, ~, ~] = pv_with_battery_full( ...
            C, price_spot, pv_reference_output*(optimal_pv/pv_refer-
ence_kw), ...
            struct('capacity', batt_size, 'eff', battery_default.eff, ...
                'Pcharge', battery_default.Pcharge, 'Pdischarge', bat-
tery_default.Pdischarge, ...
                'grid_charge', bat-
tery_ctrl.grid_charge, 'peak_shave', battery_ctrl.peak_shave));

        transfer_cost_b = grid_imp_b*transfer_var + transfer_fixed_an-
nual;
        annual_cost_b = energy_cost_b + transfer_cost_b + ...
            (optimal_pv*pv_cost_per_kw*pv_om_rate) +
om_batt;

        NPC_batt = optimal_pv*pv_cost_per_kw + capex_batt + ...
            (annual_cost_b*discount_factor);

        % Difference vs PV-only NPC
        NPC_diff_matrix(bi,ci) = optimal_pv_npc - NPC_batt;
    end
end

% Final visual: Savings relative to PV-only
figure('Name','Battery value vs price','NumberTitle','off','Color','w');
for bi = 1:nb
    plot(batt_cost_vec, NPC_diff_matrix(bi,:), '-o', 'LineWidth', 1.6); hold
on;
end
yline(0, 'k--', 'Break-even', 'LineWidth', 1.4);

xlabel('Battery price [€/kWh]');
ylabel('NPC savings compared to PV-only [€]');

```

```

legend(arrayfun(@(x) sprintf('%d kWh battery',x),battery_test_sizes,'Uni-
formOutput',false));
title('Economic benefit of battery storage relative to PV-only');
grid on; grid minor;

%% =====
% PLOTS
%% =====

% Cumulative cost (use your existing hourly_cost_batt etc.)
pv_used = min(pv, C);
cum_base_set = cumsum((C*set_price)*ones(nHours,1));
cum_base_spot = cumsum(C.*price_spot);
cum_pv_set = cumsum((C - pv_used)*set_price);
cum_pv_spot = cumsum((C - pv_used).*price_spot);
cum_pv_batt = cumsum(hourly_cost_batt);

figure('Name','Cumulative Cost','NumberTitle','off','Color','w');
plot(t_hour, cum_base_set,'k', ...
      t_hour, cum_base_spot,'b', ...
      t_hour, cum_pv_set,'g', ...
      t_hour, cum_pv_spot,'m', ...
      t_hour, cum_pv_batt,'r','LineWidth',1.1);
legend(CaseNames,'Location','northwest');
xlabel('Time (2024)');
ylabel('Cumulative Cost [€]');
title('Cumulative annual energy cost by scenario');
grid on;

% PV sweep
figure('Name','PV Sweep','NumberTitle','off','Color','w');
yyaxis left;
plot(pv_sizes_kW, pv_sweep_annual,'-o','LineWidth',1.4);
ylabel('Annual Cost [€/year]');
yyaxis right;
plot(pv_sizes_kW, pv_sweep_NPC10,'-s','LineWidth',1.4);
ylabel('10-year NPC [€]');
xlabel('PV Capacity [kW]');
title('PV Sizing: Annual Cost and 10-year NPC');
grid on;

% Battery sweep (fixed PV = optimal_pv)
figure('Name','Battery Sweep','NumberTitle','off','Color','w');
yyaxis left;
plot(battery_sizes, batt_sweep_annual,'-o','LineWidth',1.4);
ylabel('Annual Cost [€/year]');
yyaxis right;
plot(battery_sizes, batt_sweep_NPC10,'-s','LineWidth',1.4);
ylabel('10-year NPC [€]');
xlabel('Battery Capacity [kWh]');
title(sprintf('Battery Sizing: Annual Cost and 10-year NPC (PV = %.0f
kW)', optimal_pv));
grid on;

```

```

% Highlight optimal battery
hold on;
plot(optimal_batt, optimal_batt_annual, 'ro', 'MarkerSize', 8, 'Lin-
ewidth', 1.8);
text(optimal_batt, optimal_batt_annual, sprintf(' Opt = %.0f kWh', opti-
mal_batt), ...
    'VerticalAlignment', 'bottom');

fprintf('Done.\n');

%% =====
% OPTIMAL SYSTEM SUMMARY TABLE
%% =====

fprintf('\n--- OPTIMUM SUMMARY ---\n');

% PV-only optimum (from PV sweep)
PVonly_PV      = optimal_pv;
PVonly_Batt    = 0;
PVonly_Annual  = optimal_pv_annual;
PVonly_NPC10   = optimal_pv_npc;
PVonly_Payback = optimal_pv_payback;

% Battery sweep optimum (with fixed optimal PV)
BattOpt_PV     = optimal_pv;           % same fixed PV used in battery
sweep
BattOpt_Batt   = optimal_batt;
BattOpt_Annual = optimal_batt_annual;
BattOpt_NPC10  = optimal_batt_npc;
BattOpt_Payback = optimal_batt_payback;

% Global optimum (from high-resolution heatmap)
Global_PV      = pv_opt_global;
Global_Batt    = batt_opt_global;
Global_Annual  = NaN;                 % optional: can compute if needed
Global_NPC10   = NPC_base_spot + min_val; % absolute NPC
Global_Payback = NaN;                 % global optimum not tied to simple pay-
back

% Build summary table
OptSummary = table( ...
    ["PV only"; "PV + Battery (sweep)"; "Global Opt (heatmap)"], ...
    [PVonly_PV; BattOpt_PV; Global_PV], ...
    [PVonly_Batt; BattOpt_Batt; Global_Batt], ...
    [PVonly_Annual; BattOpt_Annual; Global_Annual], ...
    [PVonly_NPC10; BattOpt_NPC10; Global_NPC10], ...
    [PVonly_Payback; BattOpt_Payback; Global_Payback], ...
    'VariableNames', {'Case', 'PV_kw', 'Battery_kWh', 'AnnualC-
ost', 'NPC_10yr', 'Payback_Years'});

disp(OptSummary);

%% =====
% DISPATCH ENGINE: PV + Battery
% Full upgraded version:

```

```

% - Correct grid charging
% - Correct peak shaving
% - Clean & consistent logic for all cases
% =====
function [energy_cost, grid_import_kWh, grid_export_kWh, hourly_cost,
battery_energy] = ...
    pv_with_battery_full(C_load, spot, pv_gen, params)
%
% INPUTS:
% C_load      - constant load (kW or kWh per hour)
% spot        - spot prices [€/kWh]
% pv_gen      - PV generation timeseries [kWh]
%
% params.capacity      - battery capacity [kWh]
% params.Pcharge       - max charge power [kW]
% params.Pdischarge    - max discharge power [kW]
% params.eff           - round-trip efficiency (0-1)
% params.grid_charge   - allow battery charging from grid? (true/false)
% params.peak_shave    - optional grid import limit [kW], 0 = disabled
%
% OUTPUTS:
% energy_cost          - total cost = sum(hourly_cost)
% grid_import_kWh     - yearly grid imports
% grid_export_kWh     - yearly grid exports
% hourly_cost         - vector of import/export costs
% battery_energy      - SOC timeseries
%
%% -----
% Ensure column vectors
%% -----
pv_gen = pv_gen(:);
spot   = spot(:);
nH     = numel(pv_gen);

%% -----
% Handle missing parameters
%% -----
if ~isfield(params, 'capacity'), params.capacity = 0; end
if ~isfield(params, 'Pcharge'),  params.Pcharge   = 0; end
if ~isfield(params, 'Pdischarge'), params.Pdischarge = 0; end
if ~isfield(params, 'eff'),      params.eff       = 1.0; end
if ~isfield(params, 'grid_charge'), params.grid_charge = false; end
if ~isfield(params, 'peak_shave'), params.peak_shave = 0; end

Cap    = params.capacity;
Pch    = params.Pcharge;
Pds    = params.Pdischarge;

eff_rt = params.eff;
eff_ch = sqrt(eff_rt);
eff_ds = sqrt(eff_rt);

%% =====
% NO BATTERY → fast return path

```

```

%% =====
if Cap <= 0
    grid_import_kWh = sum(max(C_load - pv_gen, 0));
    grid_export_kWh = sum(max(pv_gen - C_load, 0));
    hourly_cost    = (max(C_load - pv_gen, 0) - max(pv_gen - C_load, 0))
.* spot;
    energy_cost     = sum(hourly_cost);
    battery_energy  = zeros(nH,1);
    return;
end

%% =====
% INITIALIZE STORAGE
%% =====
soc          = 0;
grid_import  = zeros(nH,1);
grid_export  = zeros(nH,1);
battery_energy = zeros(nH,1);
hourly_cost  = zeros(nH,1);

%% =====
% MAIN DISPATCH LOOP
%% =====
for t = 1:nH

    load_t = C_load;
    pv_t   = pv_gen(t);

    net = pv_t - load_t; % positive = surplus, negative = deficit

    discharge_used = 0; % track for peak shaving

    %% =====
    % CASE A: PV surplus
    %% =====
    if net >= 0

        % Charge battery first
        charge_from_pv = min([net, Pch, Cap - soc]);
        soc = soc + charge_from_pv * eff_ch;

        % Any leftover PV → grid export
        grid_export(t) = net - charge_from_pv;

    %% =====
    % CASE B: PV deficit
    %% =====
    else
        deficit = -net;

        % Battery discharge
        discharge_possible = min([deficit, Pds, soc]);
        discharge_used = discharge_possible;

        soc = soc - discharge_possible / eff_ds;
    end
end

```

```

remaining_deficit = deficit - discharge_possible;

% -----
% Optional grid charging logic
% -----
if params.grid_charge && remaining_deficit == 0
    % If price is low → charge from grid
    % (simple heuristic: charge if SOC < 100% and price < median)
    % User can refine
    if soc < Cap
        cheap = spot(t) < median(spot);
        if cheap
            grid_charge_power = min([Pch, Cap - soc]);
            soc = soc + grid_charge_power * eff_ch;
            grid_import(t) = grid_import(t) + grid_charge_power;
        end
    end
end

% Remaining deficit supplied by grid
grid_import(t) = grid_import(t) + remaining_deficit;
end

%% =====
% OPTIONAL PEAK SHAVING
%% =====
if params.peak_shave > 0
    limit = params.peak_shave;

    if grid_import(t) > limit
        overshoot = grid_import(t) - limit;

        % Can the battery shave?
        extra_discharge = min([overshoot, Pds - discharge_used,
soc]);

        soc = soc - extra_discharge / eff_ds;
        grid_import(t) = grid_import(t) - extra_discharge;
    end
end

%% =====
% ENFORCE SOC and COST
%% =====
soc = max(0, min(soc, Cap));

battery_energy(t) = soc;
hourly_cost(t) = (grid_import(t) - grid_export(t)) * spot(t);

end

%% =====
% OUTPUT TOTALS
%% =====

```

```
grid_import_kWh = sum(grid_import);  
grid_export_kWh = sum(grid_export);  
energy_cost     = sum(hourly_cost);
```

```
end
```