Delrapport 2

Pythonmodell för simulering av laddningsmönster och energiflöden på stadsbussdepån

Kerstin Ljungemyr

Innehåll

1 Modellbeskrivning **2**

1.1 Bussflotta . . . 2

1.2 Lokal elproduktion via biogasmotor . . . 2

1.3 Fjärrvärmeproduktion från biogasmotorn . . . 3

1.4 Utveckling av syntetiska produktions- och kostnadsprofiler baserat på historisk data . 4

1.4.1 Elpriser . . . 4

1.4.2 Fjärrvärmepriser . . . 5

1.4.3 Lokal solelproduktion . . . 5

1.5 Laddmönster . . . 6

1.5.1 Beräkning av bussbatteriernas laddnivå . . . 7

1.5.2 Säsongsvariationer . . . 7

1.6 Optimeringsfaktorer . . . 8

1.7 Klimatpåverkan . . . 8

1.8 Antaganden och begränsningar . . . 9

2 Resultat **10**

3 Pythonkod **13**

Referenser **57**

1. Modellbeskrivning

För att analysera effektbehov, kostnader och utsläpp vid stadsbussdepån i en framtid där bussflottan är elektrifierad utvecklades en simuleringsmodell i Python där en biogasmotor implementeras. Modellen simulerar laddning av bussflottan timme för timme och beräknar depåns effektuttag, import/export från elnätet, fjärrvärmeproduktion, solelproduktion, samt kostnader baserat på gaskostnader och historiska el- och fjärrvärmepriser. Även klimatutsläpp beräknas utifrån givna emissionsfaktorer.

1.1 Bussflotta

Den simulerade bussflottan baseras på att Gamla Uppsala Buss (GUB) nuvarande stadsbussflotta elektrifieras. Enligt GUB består dagens stadsbusstrafik av totalt 168 bussar, varav 46 stycken är normalbussar och 122 stycken är ledbussar enligt tabellen nedan.

Tabell 1: Befintlig bussflotta för Uppsalas stadstrafik ägd av GUB.

Transportmedel	**Antal**	**Busstyp**	**Källa**
HVO	20	Normalbuss	[Gamla Uppsala Buss, u.å. d]
HVO	51	Ledbuss	[Gamla Uppsala Buss, u.å. d]
Biogas	26	Normalbuss	[Gamla Uppsala Buss, u.å. b]
Biogas	71	Ledbuss	[Gamla Uppsala Buss, u.å. b]
Summa	**168**		

I simuleringen antas dessa ersättas av eldrivna bussmodeller från MAN, vilket baseras på att det är regionens huvudleverantör av bussar och de visar deras modeller på sin webbplats [Gamla Uppsala Buss, u.å. c]. I tabellen nedan ges den info kring bussflottan som används i simuleringen. Mer ingående läsning om den framtida bussflottan finnes i huvudrapporten.

Tabell 2: Bussflotta och batterikapacitet i simuleringsmodellen.

Busstyp	**Antal (st)**	**Modell**	**Batterikapacitet (kWh)**	**Källa batteri**
Normalbuss	46	MAN LION'S CITY 12 E	534	[MAN Truck & Bus, u.å. a]
Ledbuss	122	MAN LION'S CITY 18 E	712	[MAN Truck & Bus, u.å. b]
Summa	**168**			

1.2 Lokal elproduktion via biogasmotor

I modellen representeras biogasmotorn som en effektproducerare med tidsvarierande effekt. Den producerade effekten jämförs timme för timme med den totala laddeffekten för bussarna för att beräkna depåns nettoeffekt mot elnätet. Om laddbehovet överstiger biogasmotorns produktion uppstår import från elnätet. Om biogasmotorns produktion i stället överstiger laddbehovet uppstår ett överskott som kan exporteras till elnätet.

Biogasmotorn antas inte producera samma effekt under hela dygnet. I stället används en förenklad driftprofil där motorn körs på 60 % av maximal eleffekt under dagtid, mellan kl. 08:00 och 18:00, och på 100 % av maximal eleffekt under övriga timmar. Detta antagande baseras på att det huvudsakliga laddbehovet för stadsbussarna uppstår under och nattetid, då bussarna återvänder till depån och laddas

inför kommande dygn [1].

Syftet med driftprofilen är att bättre anpassa den lokala elproduktionen till bussarnas laddmönster. Under dagtid, när majoriteten av bussarna antas vara i trafik och laddbehovet på depån är lägre, reduceras biogasmotorns produktion. Under kväll och natt ökas produktionen till full effekt för att möta det högre effektbehovet från nattladdningen. Modellen tar därmed hänsyn till en tidsberoende drift av biogasmotorn, även om styrningen fortfarande är förenklad och inte optimeras timme för timme utifrån elpris, fjärrvärmepris eller aktuellt laddbehov.

Modellen har testats med en biogasmotor med elektrisk effekt på 3, 4 och 5 MW. Den elenergi som produceras av biogasmotorn beräknas utifrån biogasmotorns timvisa eleffekt. Eftersom modellen använder ett tidssteg på en timme kan effekten i kW direkt omvandlas till energi i kWh genom multiplikation med tissteget, se ekvationen nedan.

$$E_{biogas,el} = P_{biogas,el} \cdot \Delta t \tag{1}$$

För att uppskatta hur mycket kemisk energi i rågas som krävs, används biogasmotorns elektriska verkningsgrad. Den sattes i modellen till 0,4.

$$E_{biogas,brnsle} = \frac{E_{biogas,el}}{\eta_{el}} \tag{2}$$

Den beräknade bränsleenergin kan sedan multipliceras med antaget inköpspris för rågasen. Antaget inköpspris är i modellen satt till 0,8 kr/kWh då detta enligt uppgift är produktionspriset för rågas[2] och Region Uppsala får köpa gasen för förmånligt pris[3].

Depåns nettoeffekt mot elnätet beräknas som bussarnas laddeffekt (se under rubriken Laddmönster) minus lokal elproduktion från biogasmotor och solceller (se under rubriken Lokal solelproduktion). Positiv nettoeffekt innebär import från elnätet, medan negativ netooeffekt innebär export. Genom att multiplicera effekten med tidssteget kan sedan nettoenergi i kWh fås.

$$P_{elnt} = P_{laddning} - P_{biogas} - P_{sol} \tag{3}$$

1.3 Fjärrvärmeproduktion från biogasmotorn

Förutom elproduktion antaas biogasmotorn även producera värme som kan ledas ut i Uppsalas fjärrvärmenät. I modellen antas värmeproduktionen vara proportionell mot biogasmotorns eleffekt och enligt uppgift motsvarar en biogasmotor med elektrik effekt på 5 MW ungefär 6 MW värme-effekt. Värmeproduktionen antas följa samma mönster som elproduktionen vilket innebär att även värmeproduktionen antas vara 60 % av maximal värmeeffekt under dagtid och 100 % under övriga timmar.

[1] Detta antagande skedde i samråd med Tobias Timander, teknikförvaltare och teknikspecialist biogas på Region Uppsala, och Jonas Eriksson, teknisk chef Fastighet och Service på Region Uppsala, via e-postkonversation 19/5-2026.

[2] Omräknat värde från e-postkonversation med Lennart Nordin på Uppsala Vatten, 12/5-2026.

[3] Tobias Timander, teknikförvaltare och teknikspe- cialist biogas på Region Uppsala, och Jonas Eriksson, teknisk chef Fastighet och Service på Region Uppsala, under ett möte 24/4-2026.

$$P_{vrme} = P_{el} \cdot \frac{6}{5} \tag{4}$$

För att få fram hur mycket värmeeffekt som produceras används ekvationen ovan. Den producerade värmeenergin kan därefter beräknas genom att multiplicera värmeeffekten med tidsstegets längd, vilket i modellen är 1 timme.

$$E_{vrme} = P_{vrme} \cdot \Delta t \tag{5}$$

1.4 Utveckling av syntetiska produktions- och kostnadsprofiler baserat på historisk data

För att kunna simulera bussdepån behövdes data för elpris, fjärrvärmepris och solelproduktion. Istället för att använda konstanta årsmedelvärden skapades syntetiska timprofiler baserade på hisotriska data. Syftet med detta var att få med tidsmässiga variationer över dygnet och året, men samtidigt inte bli beroende av ett enskilt historiskt år och dess tillfälliga händelser.

De syntetiska profilerna skapades först genom att gruppera historiska data efter månad, dag och timme. För varje sådan kombination beräknades sedan ett historiskt medelvärde och standardavvikelse. Vid simulering hämtades sedan detta medelvärde. För att få med variation adderades sedan en slumpmässig avvikelse baserad på den historiska standardavvikelsen.

1.4.1 Elpriser

Uppsala ligger i elområde SE3 [Svenska kraftnät, 2025]. Timvisa elspotpriser för detta område laddades ned från Energinet [Energinet, n.d.]. Data mellan september 2022 och 2025 laddades ned. Elpriserna hämtades i EUR/MWh. För att kunna göra om valutan från euro till svenska kronor laddades växlingskursen för motsvarande datum ned [Sveriges riksbank, n.d.]. I pythonkoden matchades sedan det timvisa elpriset med motsvarande dags valutakurs. Dessutom ändrades enheten från SEK/MWh till SEK/kWh.

Efter att ha bearbetas på detta sätt skapades sedan en syntetisk elprisprofil enligt ovan. Denna profil användes vid laddoptimering och vid beräkning av kostnader för inköpt el och intäkrer från exporterad el, se ekvationer nedan.

$$Kostnad_{el} = \sum_{t} E_{import,t} \cdot pris_{el,t} \tag{6}$$

$$Intkt_{el} = \sum_{t} E_{export,t} \cdot pris_{el,t} \tag{7}$$

1.4.2 Fjärrvärmepriser

För att uppsakatta intäker från producerad värme från biogasmotorn användes historiska fjärrvärmepriser [4]. Fjärrvärmepriserna hämtades som marginalkostnader för fjärrvärmeproduktion i Uppsala, eftersom biogasmotorn på bussdepån skulle kopplas på fjärrvärmenätet, men producera relativt lite energi i jämförelse med Uppsalas stora kraftvärmeverk.

På samma sätt som för elpriserna skapades sedan en syntetisk fjärrvärmeprofil baserad på dessa historiska värden. Denna profil användes för att beräkna intäkter från den värme som produceras av biogasmotorn. Att värmen nyttiggörs som fjärrvärme är en förenkling eftersom den faktiska värmeavsättningen beror på säsong, nätkapacitet och efterfrågan i fjärrvärmenätet, men enligt Michael Hägg på vattenfall visar en preliminär nätanalys på goda förutsättningar att ta emot de effekter biogasmotorn skulle producera [5].

De timvisa fjärrvärmepriserna i SEK/kWh multipliceras med värmeproduktionen för att få en uppskattad fjärrvärmeintäkt.

1.4.3 Lokal solelproduktion

Det finns sju solcellsanläggningar vid bussdepåområdet. Regionbussdepån har en total installerad effekt på ca 407 kWp [Region Uppsala, n.d.], medan stadsbussdepån har en installerad effekt på ca 300 kWp [6].

Historisk solelproduktion från solcellsanläggningarna på regionbussdepån användes för att uppskatta produktionen från solcellsanläggningarna vid stadsbussdepån för vilka data saknades. Den historiska produktionen skalades upp utifrån installerad effekt. Antagandet att den nya anläggningen har samma specifika produktionsprofil per installerad kWp sågs som rimlig eftersom den geografiska platsen, och därmed solinstrålningen, är i princip helt lika.

I soleldatan för regionbussdepån fanns orimliga negativa värden och mycket stora toppar som behövde rensas bort innan datan användes i simuleringen. Dessa värden togs bort och ersattes genom interpolation.

Eftersom produktionsdatan var angiven som energi per timme kunde den direkt tolkas som genomsnittlig effekt under respektive timme. Ett värde i kWh per timme motsvarar därmed ett medeleffektvärde i kW för samma timme. Den bearbetade soleldatan användes därefter för att skape en syntetisk solelprofil enligt samma princip som tidigare.

Den syntetiska solelprofilen användes i simuleringen som lokal elproduktion. Både regionbussdepåns och stadsbussdepåns solel antogs i denna modell gå till laddningen av stadsbussarna.

[4] Data för fjärrvärmepriser gavs från Michael Hägg på Vattenfall via e-post, 20/5-2026.
[5] Michael Hägg på Vattenfall via e-post, 20/5-2026.
[6] Stadsbussdepåns installerade effekt angavs av Daniel Svärdsjö, energicontroller på Region Uppsala, via e-post, 11/5-2026.

1.5 Laddmönster

I vilket mönster som bussarna laddas baserades på information från Keolis[7], vilka bedriver kollektivtrafik för bland annat SL och Västtrafik [Keolis, n.d.]. (Keolis har även ansvar för regiontrafiken i Uppsala, men i den verksamheten förekommer inte någon elbussladdning).

Den nattliga laddningen av elbussarna delades in i tre faser:

1. Initial laddning
2. Bulkladdning
3. Slutladdning

Då en buss anländer till depån genomförs en initial laddning upp till ca 30 % SoC (State of Charge). Denna laddning sker relativt snabbt för att säkerställa att bussarna kan flyttas inom depåområdet och för att undvika att batterierna står urladdade under längre perioder.

Efter den initiala laddningen genomförs en bulkladdning upp till ca 80 % SoC. Bulkladdningen optimeras utifrån elspotpriser och depåns effektbelastning. Syftet med optimeringen är att minska elkostnader och effekttoppar.

Inför att bussen ska avgå från bussdepån genomförs en slutladdning där batterierna laddas upp till önskad laddnivå för bussens planerade omlopp. Denna laddning sker under de sista timmarna före avgång för att undvika att batterierna står fulladdade under längre perioder.

Utöver den ordinarie nattladdningen inkluderas även lunchladdning i modellen. Detta baseras på att laddning under dagtid enligt uppgift kan förekomma i undantagsfall[8]. Lunchladdningen modelleras därför inte som en del av den normala laddstrategin, utan som ett extra laddtillfälle för en andel av bussarna. I modellen antas 10 % av bussarna genomföra lunchladdning under dagen. Dessa bussar antas anlända till depån någon gång mellan kl. 11:00 och 13:00 och stå tillgängliga för laddning under en till två timmar. Syftet är inte att detaljmodellera varje buss faktiska dagliga omlopp, utan att representera att ett visst begränsat laddbehov kan uppstå även under dagtid.

I modellen önskar bussarna olika effektnivåer under dessa olika faser. Under slut- och lunchladdning önskas snabbladdning på 150 kW, medan normalladdning på 100 kW önskas under bulkladdningen och initialladdningen. Dessa effekter är baserade på information från Keolis[9]. Om depåns tillgängliga effektkapacitet är begränsad reduceras dock den faktiska laddeffekten till den tillgängliga effekten. Mer om det under rubriken Optimeringsfaktorer.

För att representera att bussarna har olika rutter, ankomsttider och avgångstider används stokastiska parametrar i simuleringsmodellen. För varje buss slumpas:

- Ankomsttid till depån

[7] Informaion gällande nattlig laddning av elbussar lämnades via e-post från Hans-Petter Larsson, teknisk direktör på Keolis, den 12e maj 2026.

[8] Information om ungefärliga tider för laddningen angavs av Tobias Timander, teknikförvaltare och teknikspecialist biogas på Region Uppsala, och Jonas Eriksson, teknisk chef Fastighet och Service på Region Uppsala, under ett möte 24/4-26

[9] E-post från Hans-Petter Larsson, teknisk direktör på Keolis, den 12e maj 2026.

- SoC vid ankomst
- Avgångstid från depån
- Önskad SoC vid avgång

1.5.1 Beräkning av bussbatteriernas laddnivå

Varje bussbatteris laddnivå, State of Charge (SoC), uppdateras varje timme under simuleringen utifrån just den bussens tillförda effekt. Sambandet beskrivs under ekvationen nedan.

$$SoC_{ny} = SoC_{gammal} + \frac{P \cdot \Delta t \cdot \eta}{E_{batteri}} \cdot 100 \tag{8}$$

där:

- P är laddeffekten i kW
- Δt är tidsstegets längd, vilket är 1 h
- η är laddverkningsgraden, vilken har satts till 0,9
- $E_{batteri}$ är batteriets kapacitet i kWh

I modellens olika laddningsfaser beräknas hur mycket energi som krävs för att nå en viss SoC enligt samma samband. Exempelvis används följande ekvation för att beräkna energibehovet ör att få de önskade energibehoven i de olika laddningsfaserna:

$$E_{behov} = \frac{SoC_{nskad} - SoC_{aktuell}}{100} \cdot \frac{E_{batteri}}{\eta} \tag{9}$$

Ekvationen visar att skillnaden i SoC omvandlas till ett energibehov utifrån batteriets kapacitet och laddverkningsgraden.

Den totala laddenergin till bussarna beräknas sedan genom att summera den faktiska laddeffekt som bussarna får under varje timme i simuleringen. Som tidigare nämnt för effekten in på depån, kan även den timvisa laddeffekten i kW omvandlas till energi i kWh genom multiplikation med tidssteget som är en timme.

1.5.2 Säsongsvariationer

För att ta hänsyn till att elbussarnas energibehov och laddförutsättningar varierar över året inkluderas säsongsberoende antaganden i modellen. Vinterperioden definieras i modellen som november till mars. Under denna period antas laddverkningsgraden vara lägre än under övriga månader, samt att ett uppvärmningsbehov av bussarna finns.

I modellen används en laddverkningsgrad på 80 % under vintern jämfört med 90 % under övriga delar av året. Detta ska symbolisera att batteriets verkningsgrad minskar och att räckvidden för

bussen blir kortare på grund av ökade energibehov då det är kallare ute [Nilsson, 2023]. Den lägre verkningsgraden påverkar därmed både bussarnas totala laddbehov och därmed också depåns effekt- och energibalans.

Utöver den försämrade laddverkningsgraden inkluderas även ett uppvärmningsbehov under vinterperioden. Bussarna antas då behöva förvärmas innan de lämnar depån på morgonen[10]. I modellen representeras detta genom att varje buss under vinterperioden tilldelas ett extra effektuttag under timmen före avgång. Uppvärmningen behandlas som ett separat effektbehov från själva batteriladdningen och bidrar därför till depåns totala effektbelastning.

Syftet med dessa antaganden är inte att modellera exakt batteriprestanda eller varje buss faktiska värmebehov, utan att i allmänhet fånga de viktigaste säsongsvariationerna.

1.6 Optimeringsfaktorer

Eftersom många bussar antas ladda under ungefär samma timmar införs en begränsning av depåns maximala tillgängliga laddeffekt. Detta motiveras av tekniska begränsningar i elinfrastrukturen och av att höga effekttoppar kan medföra betydande kostnader. Den nuvarande reserverade effekten för depån är ca 5 MW [11]. För detta scenario sattes den maximala effekten för depån till 15 MW, vilket kan ses som en mycket optimistisk siffra då antalet elbussar kommer öka med mycket mer än en faktor 3.

I modellen antas därför att den totala laddeffekten inte får överstiga detta bestämda värde. Om flera bussar försöker ladda samtidigt fördelas den tillgängliga effekten mellan bussarna så att effektgränsen inte överskrisd. För att minska risken för höga effekttoppar reduceras även laddeffekten under perioder med hög samtidig belastning. När detta sker väljs en lägre laddeffekt än vad bussen egentligen önskar för att inte överskrida depåns maximala effektkapacitet.

1.7 Klimatpåverkan

Klimatpåverkan beräknades för både importerad el och använd rågas. Utsläppen från importerad el beräknas genom att multiplicera den importerade elenergin med emissionsfaktorn för nordisk elmix, vilken är 59 g CO2e/kWh [Chi Johansson and Sandgren, 2026].

Utsläppen från biogasmotorn beräknades genom att multiplicera den använda biogasenergin med emissionsfaktorn för rågasen. Snittfaktorn för emissioner från biogas är -2,52 g CO2e/kWh [Naturvårdsverket, 2026]. Värdet innefattar utvinning, transport, omvandling och förbränning av bränslet sett ur ett livscykelperspektiv. Där i ingår rötning av gödsel. Genom att ta tillvara på gödslet undviks utsläpp som annars skulle uppstått vid lagring och spridning av obehandlad gödsel på åkrar [Energigas Sverige, u.å.]. Enligt Nordin [2026] planeras en ökad andel gödsel i substratet efter utbyggnaden av biogasanläggningen i Uppsala. Därmed har denna siffra setts som relevant. Enligt uppgift är det också okej att använda denna emissionsfaktor trots att det är icke-uppgraderad rågas som ska förbrännas i

[10] Effekttoppar under vintern på grund av uppvärmningsbehov angavs av Tobias Timander, teknikförvaltare och tekniskspecialist biogas på Region Uppsala, och Jonas Eriksson, teknisk chef Fastighet och Service på Region Uppsala, under ett möte 24/4-26.

[11] Jonas Eriksson, teknisk chef Fastighet och Service på Region Uppsala, under ett möte 24/4-26

biogasmotorn[12].

Eftersom biogasmotorn inte bara producerar el som går till laddning av stadsbussarna, utan även genererar fjärrvärme och el för export till elnätet, har en allokering gjorts utifrån andel producerad energi per produkt. Se exemplet nedan.

$$Allokeringsfaktor_{fjrrvrme} = \frac{E_{fjrrvrme}}{E_{total}} \tag{10}$$

1.8 Antaganden och begränsningar

Simuleringsmodellen bygger åp ett antal förenklade antaganden. Modellens syfte är främst att analysera övergripande effektbalanser, laddmönster och elkostnader, samt att visa på hur dessa påverkas av installation av en biogasmotor, inte att ge en exakt verklig depådrift.

Bussarnas laddmönster är baserat på information från Keolis, vilket inte är den organisation med ansvar för stadsbussarna i Uppsala. Gamla Uppsala Buss (GUB) vilka ansvarar för stadsbussarna, har blivit tillfrågade kring laddmönster och effektuttag, men har inte gett svar i tid.

Bussarnas ankomst- och avgångstider modelleras slumpmässigt inom fördefinierade tidsintervall. Verkliga omlopp inkluderas inte i modellen.

För att få en laddningsstruktur som passar bättre med verkligheten skulle effektreduktion vid hög SoC, batteritemepratur och degradering av batterierna behöva inkluderas.

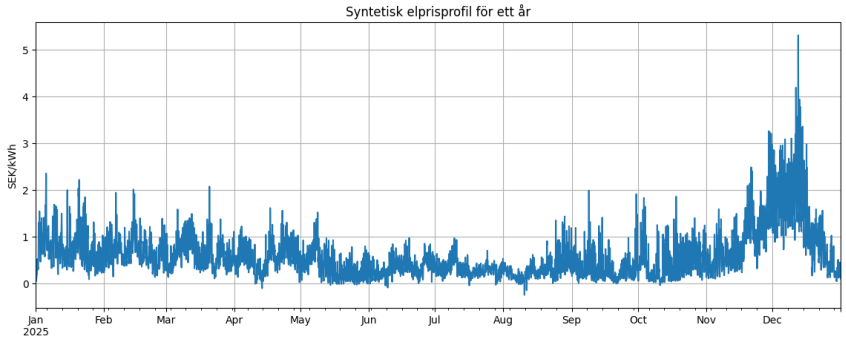
Depåns maximala effektkapacitet och minskning av effekttoppar representeras av en fast övre gräns. För att få ett bättre resultat hade dynamiska nätavgifter och verkliga begränsningar i elsystemet behövts.

I och med att den begränsade effektkapaciteten gör att bussarna inte alltid laddas med den effekt de egentligen önskar, händer det att bussar inte tillåts nå upp till sin önskade avgångs-SoC. För att minimera detta problem sorteras bussarna efter avgångstid och de bussar med en tidigare sådan prioriteras laddning först. För att detta inte ska orsaka för stora problem för bussarnas omlopp undersöks hur omfattande detta problem är i resultatet.

[12]Handledare tillika universitetslektor inom bioteknik Åke Nordin bekräftar detta i e-postkonversation 24/5-2026.

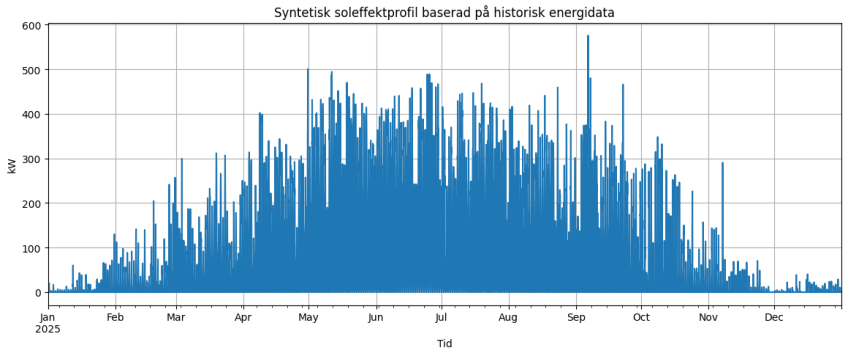
2. Resultat

Figur 1 visar resultatet av den syntetiska elprisprofilen som används för att beräkna kostnader för importerad och exporterad el, samt för optimering av bulkladdningen.



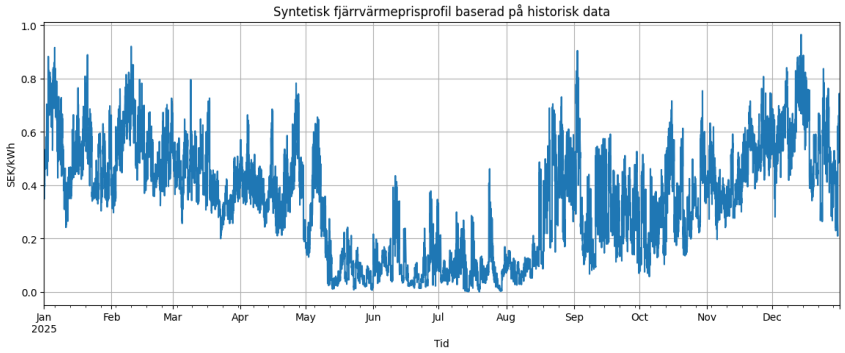
Figur 1: Syntetisk elprisprofil byggd på historisk elprisdata för elområde SE3.

Grafen i Figur 2 visar resultatet av den syntetiska profilen över producerad soleffekt på bussdepån. Den används i modellen för att estimera producerad solenergi per år.

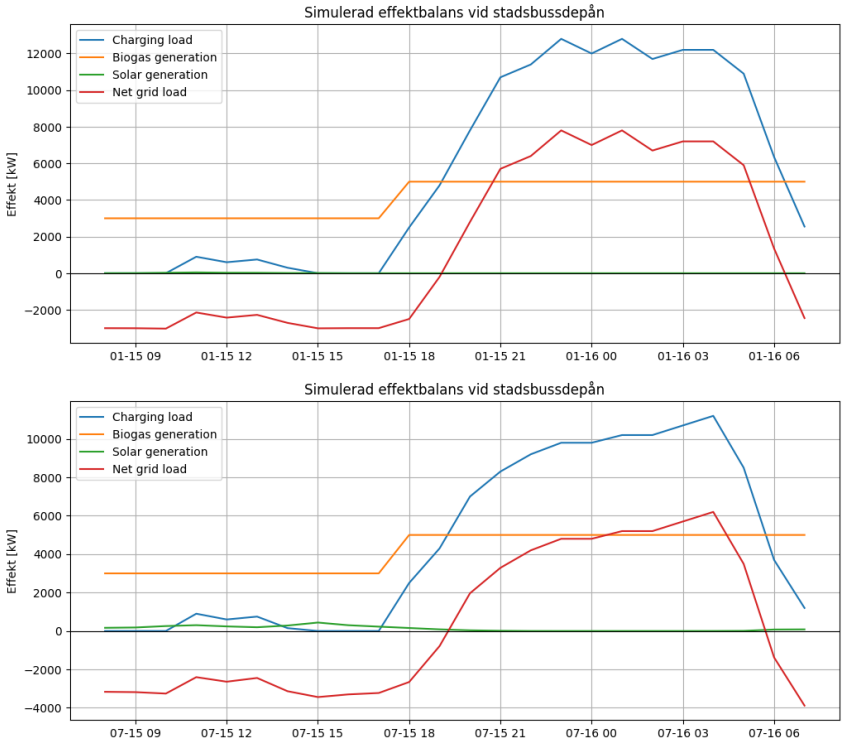


Figur 2: Syntetisk soleffektprofil byggd på historisk data från depåns solcellsanläggningar.

Figur 3 visar den syntetiska profilen för fjärrvärmepriserna som används för att få fram hur mycket intäkterna från fjärrvärmeexporten uppgår till.



Figur 3: Syntetisk fjärrvärmepris-profil byggd på historiska fjärrvärmepriser.



Figur 4: Dygnssimulering av effekt på bussdepån vid biogasmotor med elektrisk effekt på 5 MW. Överst visas ett dygn i januari månad. Undre grafen visar ett dygn i juli månad.

För att få en överblick över effektbalansen simulerades stadsbussdepåns energisystem över ett dygns tid. Figur 4 visar resultatet för två dygn i olika säsonger för fallet med en biogasmotor med elektrisk effekt på 5 MW. Den blå linjen visar elbussarnas effektbehov, den gula linjen visar biogasmotorns producerade effekt, den gröna linjen visar producerad effekt från solpaneler och den röda visar nettoeffekten från nätet. När den röda linjen är negativ sker elexport. Den blå linjen visar först på ett effektbehov för lunchladdning följt av ett stort effektbehov från nattladdningen.

En slutsats från graferna är att solel främst produceras under sommaren, men att solpanelernas bidrag är mycket litet.

Under vintern är effektuttaget för bussarna högre än på sommaren, vilket var förväntat. Att effektuttaget under vintern inte når upp till 15 MW, vilket sattes till depåns maxeffekt som optimeringen delvis sker utifrån, tyder på att optimeringsmetoden fungerar.

Tabell 3 till 5 visar totala energiflöden, driftkostnader och klimatutsläpp för stadsbussdepån från simulering av ett års tid beroende på vilken elektrisk effekt som sätts på biogasmotorn. Mer om detta kan läsas i Huvudrapporten.

Tabell 3: Års-sammanställning av energiflöden för olika eleffekter på biogasmotorn

Eleffekt biogasmotor (MW)	**Använd biogasenergi (GWh)**	**Producerad el från biogasmotor (GWh)**	**Importerad el (GWh)**	**Exporterad el (GWh)**	**Laddenergi till bussar (GWh)**	**Producerad värme från biogasmotor (GWh)**
3	55	22	28	6,4	44	26
4	73	29	23	9,3	44	35
5	91	37	19	12	44	44

Tabell 4: Ekonomisk års-sammanställning för olika eleffekter på biogasmotorn

Eleffekt biogasmotor (MW)	**Kostnad biogas (milj. kr)**	**Kostnad inköpt el (milj. kr)**	**Intäkt exporterad el (milj. kr)**	**Nettokostnad el (milj. kr)**	**Fjärrvärme-intäkt (milj. kr)**	**SUMMA (milj. kr)**
3	44	13	4,1	8,8	9,1	-44
4	58	11	6,1	4,6	12	-51
5	73	8,6	8,2	0,4	15	-58

Tabell 5: Års-sammanställning av utsläpp för olika eleffekter på biogasmotorn

Eleffekt biogasmotor (MW)	**Totala utsläpp biogasmotor (tusen kg CO_2e)**	**Allok. utsläpp fjärrvärme (tusen kg CO_2e)**	**Allok. utsläpp exp. el (tusen kg CO_2e)**	**Allok. utsläpp laddning (tusen kg CO_2e)**	**Utsläpp inköpt el (milj. kg CO_2e)**
3	-140	-75	-17	-46	+1,6
4	-180	-100	-25	-58	+1,4
5	-230	-130	-34	-71	+1,1

Under antaganden och begränsningar nämndes att den antagna begränsade effektkapaciteten på bussdepån ibland orsakar att bussar inte tillåts nå upp till sin önskade avgångs-SoC under natten. För årssimuleringen är den lägsta avgångs-SoC 63,71 %, den genomsnittliga avgångs-SoC för icke-fulladdade bussar är 83,3 % och den genomsnittliga saknade SoC för icke-fulladdade bussar är 10,46 %. Dessa värden bedöms inte orsaka problem för bussflottans omlopp och drift eftersom de icke-fulladdade bussarna fortfarande lämnar depån med en relativt hög laddnivå.

3. Pythonkod

```
# -*- coding: utf-8 -*-
"""Simulering.ipynb

Automatically␣generated␣by␣Colab.

Original␣file␣is␣located␣at
␣␣␣␣https://colab.research.google.com/drive/18
    lpLMQRr6zH6q2kgUdqSqlY98D52jWXs
"""

#SOLDATA

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from pathlib import Path
from google.colab import drive

#se till att hitta driven
drive.mount("/content/drive")

data_dir = Path("/content/drive/MyDrive/ES␣UU/ES3/Kandidatarbete/
    Scenarion␣+␣Framtidsbeskrivning/Kod␣f r␣optimering␣+␣data/Collab-
    milj ")

#Hitta filerna
df1= pd.read_csv(data_dir/ "Regionbussdep ␣Fyrislund␣Uppsala␣Lilla␣
    Bussrampen␣-␣Energi␣[kWh]-data-2026-04-22␣10_40_18.csv")
df2= pd.read_csv(data_dir/ "Regionbussdep ␣Fyrislund␣Uppsala␣Stora␣
    Bussrampen␣-␣Energi␣(kWh)-data-2026-04-22␣10_41_28.csv")
df3= pd.read_csv(data_dir/ "Regionbussdep ␣Fyrislund␣Uppsala␣
    Tv tthallen␣-␣Energi␣[kWh]-data-2026-04-22␣10_45_05.csv")
df4= pd.read_csv(data_dir/ "Regionbussdep ␣Fyrislund␣Uppsala␣Verkstad␣-␣
    Energi␣(kWh)-data-2026-04-22␣10_43_46.csv")

"""
#Se␣s ␣att␣den␣hittar␣csv-filerna
print("CSV-filer i mappen:")
for␣f␣in␣data_dir.glob("*.csv"):
␣␣␣␣print(f.name)
"""

#konvertera tiden i filerna
df1["Time"] = pd.to_datetime(df1["Time"])
df2["Time"] = pd.to_datetime(df2["Time"])
df3["Time"] = pd.to_datetime(df3["Time"])
```

```
df4["Time"] = pd.to_datetime(df4["Time"])

#s tt tid som index
df1.set_index("Time", inplace=True)
df2.set_index("Time", inplace=True)
df3.set_index("Time", inplace=True)
df4.set_index("Time", inplace=True)

#d p om kolumnerna med energi
df1.rename(
    columns={"Energi␣Regionbussdepa_lilla_rampen␣[kWh]":"
        solar_lilla_rampen"},inplace=True)
df2.rename(
    columns={"Energi␣Regionbussdepa_stora_rampen␣[kWh]":"
        solar_stora_rampen"},inplace=True)
df3.rename(
    columns={"Energi␣Regionbussdepa_tvatthall␣[kWh]":"solar_tvatthallen"
        },inplace=True)
df4.rename(
    columns={"Energi␣Regionbussdepa_verkstad␣[kWh]":"solar_verkstaden"},
        inplace=True)


#Addera filerna till en tabell
#F rsta kolumnen:tid, resterande kolumner: solar_namn
df=pd.concat([df1, df2, df3, df4], axis=1 )

    #Jag ser negativa v rden vid plottning, hitta minimumv rden:
for col in df.columns:
    print(col, df[col].min())

    #Hitta n r minimumv rdet intr ffat
#Hitta raden
idx = df4["solar_verkstaden"].idxmin()

print("Tid:", idx)
print(df4.loc[idx])

#Checka raderna runtomkring den
print("\nRader␣runt␣felet:")
print(df4.loc[idx - pd.Timedelta(hours=2): idx + pd.Timedelta(hours=2)])
#Resultat:
#Lilla rampen: 2023-10-19 10:00 -8446700.0, 11:00 8457362.0
#Stora rampen: 2023-10-19 10:00 -35312571.0, 11:00 35355470.0
#Tvatthallen: 2023-08-11 9:00 -14889979.0, 11:00 14900335.0
#Verkstaden: 2024-03-01 19:00 -382564053, 20:00 -191282027.0
#Alla anl ggningar har enorma negativa dalar.
#F ljt av enorma positiva peakar. (f rutom verkstaden)

```

```
    #Fixa datafelen, negativa v rden och f r stora v rden
def clean_solar(df, col, installed_kWp):
    #G r om fr n W till kW
    df[col] = df[col]/1000

    #ta bort negativa v rden (s tt till noll)
    df.loc[df[col] < 0, col] = pd.NA

    #ta bort fysiskt om jliga v rden f r timvis produktion
    #till t en liten marginal  ver  installerad effekt ex. kan det vara
        under 25 grader
    max_reasonably_kW=installed_kWp *1.1
    df.loc[df[col]>max_reasonably_kW, col]=pd.NA

    #interpolera  ver  gap/v rden som blir borttagna
    #g r detta bara  ver  korta gap p  max 3 timmar
    df[col] = df[col].interpolate(limit=3)

    return df

#Fixa "clean data"
df1 = clean_solar(df1, "solar_lilla_rampen", installed_kWp=35.75)
df2 = clean_solar(df2, "solar_stora_rampen", installed_kWp=145.2)
df3 = clean_solar(df3, "solar_tvatthallen", installed_kWp=36.3)
df4 = clean_solar(df4, "solar_verkstaden", installed_kWp=189.75)

    #Addera clean files till en tabell
#F rsta kolumnen: tid, f ljande kolumner: solar_name
df=pd.concat([df1, df2, df3, df4], axis=1 )
#Om ingen data ges, s tt 0:
df=df.fillna(0)

#N r detta plottas syns ingen produktion under 2022
#N r datafilerna checkas ytterligare syns det att v rden saknas mellan
    2022-04-06 och 2022-09-22
#B ttre med mindre, men mer tillf rlitlig data. B rjar d rf r fr n
    2023 ist llet f r 2022

    #Ta data fr.o.m. 2023-->
df = df.loc["2023-01-01":"2025-12-31"]

    #Summera total effekt
df["solar_total"] = df[
    ["solar_lilla_rampen", "solar_stora_rampen",
     "solar_tvatthallen", "solar_verkstaden" ]
    ].sum(axis=1, min_count=1)

df["solar_total"].plot(figsize=(12,5), label="Producerad␣energi␣RBD")
plt.ylabel("kWh␣varje␣timme")
```

```
plt.xlabel("Tid")
plt.title("Total soleleffekt vid Regionbussdep n")
plt.grid()
plt.show()


    #L GG TILL DE TRE SOLANL GGNINGARNA VID STADSBUSSDE N

#Total installerad effekt SBD: 300 kWp
increase_factor= (300 + 35.75 + 145.2 + 36.3 + 189.75) / (35.75 + 145.2 +
     36.3 + 189.75)

# ka  datan
df["solar_total_expanded"]=df["solar_total"] * increase_factor

#Eftersom datan  r  timvis energi kan kWh per timme tolkas som kW
    medeleffekt
df["solar_power_kW"] = df["solar_total_expanded"]


          #PLOT
df["solar_power_kW"].plot(figsize=(12,5), label="Estimerad producerad
    energi RBD + SBD")
plt.ylabel("kW varje timme")
plt.xlabel("Tid")
plt.title("Estimerad total soleleffekt vid Region- och stadsbussdep n")
plt.legend()
plt.grid()
plt.show()


    #FUNKTION  F  R  MEDELDATA  VER   R  + SLUMP

#skapa syntetisk medel+slump soleffektprofil kW
#byggs p  historiska medelvv rde  f r samma m nad , dag och timme
def create_synthetic_solar_power_profile(time_index, df_solar,
    random_std_factor=0.15, seed=42):

    #skapa slumpgenerator, seed g  r att slumpen blir reproducerbar
    rng = np.random.default_rng(seed)

    #kopierar datan s  att vi inte skriver  ver  den gamla
    solar_data = df_solar.copy()

    #extrahera m nad (1-12), dag (1-30/31), timme (0-23)
    solar_data["month"] = solar_data.index.month
    solar_data["day"] = solar_data.index.day
    solar_data["hour"] = solar_data.index.hour

```

```
    #ber kna medel och standardavvikelse
    profile = (
        solar_data
        .groupby(["month", "day", "hour"])["solar_power_kW"]
        .agg(["mean", "std"]))

    #skapa tom lista
    values = []

    #f r tiden:
    for t in time_index:
        #skapa en nyckel s  att datum fr n flera  r  kan kopplas ihop
        key = (t.month, t.day, t.hour)

        #om tiden finns, plocka ut medelv rde och standardavvikelse
        if key in profile.index:
            mean_value = profile.loc[key, "mean"]
            std_value = profile.loc[key, "std"]
        #annars: anv nd samma m nad och timme om exakt datum saknas
        else:
            fallback = solar_data[
                (solar_data["month"] == t.month) &
                 (solar_data["hour"] == t.hour)
            ]["solar_power_kW"]

            mean_value = fallback.mean()
            std_value = fallback.std()

        #om medelv rde saknas, s tt 0
        if pd.isna(mean_value):
            mean_value = 0

        #om standardavvikelse saknas elelr  r  0
        #"skapa" en standardavvikelse utfifr n medelv rdet och lite
            slump
        if pd.isna(std_value) or std_value == 0:
            std_value = abs(mean_value) * random_std_factor

        #ber kna soleffekten
        value = rng.normal(mean_value, std_value * random_std_factor)

        #soleffekt kan inte vara negativ
        value = max(value, 0)

        #l gg till effekten i listan
        values.append(value)

    df_solar_sim = pd.DataFrame(index=time_index)
    df_solar_sim["solar_power_kW"] = values
```

```
224
225     return df_solar_sim
226
227
228                 #PLOT
229
230 time_index_year = pd.date_range(
231     "2025-01-01␣00:00",
232     "2026-01-01␣00:00",
233     freq="1h",
234     inclusive="left")
235
236 df_solar_year = create_synthetic_solar_power_profile(
237     time_index_year,
238     df,
239     random_std_factor=0.15,
240     seed=42)
241
242 df_solar_year["solar_power_kW"].plot(figsize=(14,5))
243 plt.ylabel("kW")
244 plt.xlabel("Tid")
245 plt.title("Syntetisk␣soleffektprofil␣baserad␣p ␣historisk␣energidata")
246 plt.grid()
247 plt.show()
248
249 #ELPRISER
250
251 #se till att hitta filerna i google drive
252 from google.colab import drive
253 drive.mount('/content/drive')
254
255 from pathlib import Path
256 import pandas as pd
257 import matplotlib.pyplot as plt
258
259         #HITTA FILERNA
260 #Filen Elspotprices har datum mellan september 2022 och september 2025
261 data_dir=Path("/content/drive/MyDrive/ES␣UU/ES3/Kandidatarbete/Scenarion␣
        +␣Framtidsbeskrivning/Kod␣f r␣optimering␣+␣data/Collab-milj ")
262 price_file= data_dir/"Elspotprices.csv"
263 #L s ocks  in valutkurs-filen
264 currency_file=data_dir/"valutakurs_riksbanken.csv"
265
266
267       #HANTERA ELPRIS-filen
268 #L s in filen och byt fr n att separera med ; till att separera med ,
269 df_price =pd.read_csv(price_file, sep=";", decimal=",")
270
271 #V lj dansk tid, det st mmer med svensk
```

```
df_price["Time"] = pd.to_datetime(df_price["HourDK"])
#s tt index
df_price.set_index("Time", inplace=True)
#se till att tidsserien  r  i r tt ordning
df_price = df_price.sort_index()
#ta bort dubletter
df_price = df_price[~df_price.index.duplicated(keep="first")]

#v lj pris i EUR, inte DKK
df_price = df_price[["SpotPriceEUR"]]
#byt namn p  pris-kolumnen
df_price.rename(columns={"SpotPriceEUR": "price_EUR_MWh"}, inplace=True)

#G r fr n MWh till kWh
df_price["price_EUR_kWh"] = df_price["price_EUR_MWh"] /1000

#anvn nd EUR/kWh f r optimering
df_price["price_for_optimization"]=df_price["price_EUR_kWh"]


      #HANTERA VALUTAKURS-filenn
#se till att den l ses in r tt
df_currency=pd.read_csv(
    currency_file,  sep=";", decimal=",", encoding="utf-8-sig")

#v lj vilken koumn som anger datum
df_currency["Date"]=pd.to_datetime(df_currency["Datum"])

#V ljl vilken kolumn som  r  v xlingskurs, byt namn
df_currency=df_currency.rename(columns={"V rde":"EUR_to_SEK"})

#Beh ller bara de kolumner vi beh ver i dataframen
df_currency=df_currency[["Date", "EUR_to_SEK"]]

#S tt datumkolumnen till index och se till att den  r  sorterad
df_currency=df_currency.set_index("Date").sort_index()

#Riksbanken ger inte kurs f r alla dagar
#fyll i dessa tomma datum med v rdet f r datumet innnan
daily_currency=df_currency.resample("D").ffill()

#L gg till datumkolumn i elprisdatan
#s  att elpris och kurs kan matchas
df_price["Date"]=df_price.index.normalize()

#Matcha varje timmes elpris med dagens valutakurs
df_price=df_price.merge(
    daily_currency, left_on="Date", right_index=True, how="left")

```

```
#Om f rsta dagen saknar kurs, fyll i bak t fr n k nd data
df_price["EUR_to_SEK"]=df_price["EUR_to_SEK"].ffill().bfill()

#G r fr n EUR/MWh till SEK/kWh med den dagliga valutakurs
df_price["price_SEK_kWh"] = (
    df_price["price_EUR_MWh"]*df_price["EUR_to_SEK"]/1000)



      #PLOT

#Plotta spotpriset varje timme
df_price["price_SEK_kWh"].plot(figsize=(12,5))

plt.ylabel("SEK/kWh")
plt.title("Elpris␣SE3␣(timvis)")
plt.grid()
plt.show()

#Plotta det h gsta priset varje dag
df_price["price_SEK_kWh"].resample("D").max().plot(figsize=(12,5))

plt.title("Daglig␣peak␣i␣elpris␣(SE3)")
plt.ylabel("SEK/kWh")
plt.grid()
plt.show()



      #FUNKTION F R ATT ANV NDA MEDELDATA FR N VARJE  R

#elpris-filen inneh ller data fr n september 2022 till september 2025
#f r varje timme p    ret  vill vi ha ett medelv rde p  elpris fr n
    tidigare  r
#vill ocks  slumpa elpriserna n got
#slumpen m ste vara rimlig s  anv nder standardavvikelse


def create_synthetic_price_profile(time_index, df_price,
    random_std_factor=0.15, seed=42):
    #time_index ger vilka timmar vi vill simulera
    #df_price  r  den historiska prisdatan
    #random_std_factor styr hur mycket slump som l ggs  p
    #seed g r slumpen reproducerbar, 42  r  standard

    #Skapa en slumpgenerator med (seed g r reproducerbar)
    rng= np.random.default_rng(seed)

```

```
368     #kopiera prisdatan s  att originalet inte  ndrass :
369     price_data= df_price.copy()
370     #extrahera m nad (1 till 12), dag (1-30/31), timme (0-23)
371     price_data["month"] = price_data.index.month
372     price_data["day"] = price_data.index.day
373     price_data["hour"] = price_data.index.hour
374
375     #R kna ut medelpriset och standardavvikelse
376     #g r detta f r varje kombination av m nad , dag och timme
377     profile = (
378         price_data
379         .groupby(["month", "day","hour"])["price_SEK_kWh"].agg(["mean", "
                std"]))
380
381     #skapa tom lista f r de syntetiska priserna
382     synthetic_prices = []
383
384     #f r tiden
385     for t in time_index:
386         #skapa en nyckel s  att datum fr n flera  r  kan kopplas ihop
387         key= (t.month, t.day, t.hour)
388
389         #om tiden finns, plocka ut medelv rde och standardavvikelse
390         mean_price= profile.loc[key, "mean"]
391         std_price= profile.loc[key, "std"]
392
393         #Om std saknas eller 0, skapa en rimlig variation baserad p
                medelpriset
394         if pd.isna(std_price) or std_price == 0:
395             std_price = abs(mean_price) * random_std_factor
396
397         #Slumpa pris kring historiskt medelv rde
398         price = rng.normal(mean_price, std_price * random_std_factor)
399
400         synthetic_prices.append(price)
401
402     df_price_sim = pd.DataFrame(index=time_index)
403     df_price_sim["price_SEK_kWh"] = synthetic_prices
404
405     return df_price_sim
406
407
408             #PLOT f r "medelpriset" f r ett  r
409 year_start = pd.Timestamp("2025-01-01␣00:00")
410 year_end = pd.Timestamp("2026-01-01␣00:00")
411
412 time_index_year = pd.date_range(
413     year_start,
414     year_end,
```

```
    freq="1h",
    inclusive="left")

df_price_year = create_synthetic_price_profile(
    time_index_year,
    df_price,
    random_std_factor=0.15,
    seed=42)

df_price_year["price_SEK_kWh"].plot(figsize=(14,5))
plt.ylabel("SEK/kWh")
plt.title("Syntetisk elprisprofil f r ett r ")
plt.grid()
plt.show()

#FJ RRV RMEPRISER

#se till att hitta filerna i google drive
from google.colab import drive
drive.mount('/content/drive')

from pathlib import Path
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

#hitta filerna
data_dir=Path("/content/drive/MyDrive/ES UU/ES3/Kandidatarbete/Scenarion
    + Framtidsbeskrivning/Kod f r optimering + data/Collab-milj ")
heat_price_file= data_dir/"SamEnergipriser Uppsala 2021-2025.xlsx"

#L s in fj rrv rmepriser
#det  r  en excel-fil
df_heat_price =pd.read_excel(heat_price_file)

#konvertera tid
df_heat_price["Time"]= pd.to_datetime(df_heat_price["Time"])
#s tt tid som index oc sortera
df_heat_price=df_heat_price.set_index("Time").sort_index()

#Byt namn p  kolumnen
df_heat_price=df_heat_price.rename(
    columns={"Submitted marginal cost": "heat_price_SEK_MWh"})

#G r om fr n SEK/MWh till SEK/kWh
df_heat_price["heat_price_SEK_kWh"]=df_heat_price["heat_price_SEK_MWh"
    ]/1000

#ta brot dubletter
```

```
462 df_heat_price=df_heat_price[~df_heat_price.index.duplicated(keep="first")
        ]
463
464
465
466       #FUNKTION F R ATT ANV NDA MEDELDATA FR N VARJE  R
467 # f r varje timme p    ret  vill vi ha ett medelv rde p   elpris fr n
        tidigare  r
468 #vill ocks  slumpa elpriserna n got
469 #slumpen m ste vara rimlig s   anv nder standardavvikelse
470 def create_synthetic_heat_price_profile(time_index,df_heat_price,
        random_std_factor=0.15,seed=42):
471     #skapa slumpgenerator
472     #seed g r att slumpen blir reproducerbar
473     rng=np.random.default_rng(seed)
474
475     #kopiera s   att vi inte  ndrar  originaldatan
476     heat_data=df_heat_price.copy()
477
478     #plockar ut m nad (1-12), dag (1-30/31), timme (0-23)
479     heat_data["month"]=heat_data.index.month
480     heat_data["day"]=heat_data.index.day
481     heat_data["hour"]=heat_data.index.hour
482
483     #R kna ut medelpriset och standardavvikelse
484     # g r detta f r varje kombination av m nad , dag och timme
485     profile=(
486         heat_data
487         .groupby(["month","day","hour"])["heat_price_SEK_kWh"]
488         .agg(["mean","std"]))
489
490     #skapa tom lista
491     values=[]
492
493     # f r tiden
494     for t in time_index:
495         #skapa en nyckel s   att datum fr n flera  r  kan kopplas ihop
496         key=(t.month, t.day, t.hour)
497
498         #om tiden finns, plocka ut medelv rde och standardavvikelse
499         if key in profile.index:
500           mean_price=profile.loc[key, "mean"]
501           std_price=profile.loc[key, "std"]
502
503         #annars: anv nd samma m nad och timme om exakt datum saknas
504         else:
505           fallback=heat_data[
506               (heat_data["month"]==t.month) &
507               (heat_data["hour"]==t.hour)
```

```
            ]["heat_price_SEK_kWh"]
            mean_price=fallback.mean()
            std_price=fallback.std()

        #om medelv rde saknas, s tt 0
        if pd.isna(mean_price):
            mean_price=0

        #om standardavvikelse saknas, skapa ett rimlgt mha medelv rde
        if pd.isna(std_price) or std_price==0:
            std_price=abs(mean_price)*random_std_factor

        #skapa v rde
        price=rng.normal(mean_price, std_price*random_std_factor)

        #finns inga negativa j rrv rmepriser
        price=max(price, 0)

        #l gg till i listan
        values.append(price)

    #skapa dataframe
    df_heat_price_sim = pd.DataFrame(index=time_index)
    df_heat_price_sim["heat_price_SEK_kWh"] = values

    return df_heat_price_sim

            #PLOT

  #skapa tidsindex f  ett helt  r
time_index_heat_year = pd.date_range(
    "2025-01-01 00:00",
    "2026-01-01 00:00",
    freq="1h",
    inclusive="left")

#skapa syntetisk fj rrv rmeprisprofil
df_heat_price_year = create_synthetic_heat_price_profile(
    time_index_heat_year,
    df_heat_price,
    random_std_factor=0.15,
    seed=42)

#Plotta fj rrv rmepriset
df_heat_price_year["heat_price_SEK_kWh"].plot(figsize=(14, 5))

plt.ylabel("SEK/kWh")
plt.xlabel("Tid")
plt.title("Syntetisk fj rrv rmeprisprofil baserad p  historisk data")
```

```
plt.grid()
plt.show()

#GEMENSAM KOD F R BUSSEP N-SIMULERINGEN

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

dt =1  #timmar

#Estimerad bussflotta och deras batterikapacitet
n_bus_12m=46  #st
n_bus_18m=122   #st

battery_12m =534  #kWh
battery_18m = 712  #kWh


        #BIOGASMOTORN

#Elektrisk effekt
biogas_power_max_kW =5000  #kW

#elektrisk verkningsgrad (f r att f  ut total m ngd r gas i slutet)
biogas_electrical_efficiency = 0.40 #ca

#V rmeeffekt
#Antag: 5 MW el ger 6 MW v rme
biogas_heat_to_electric_ratio = 6/5
#Skapar v rmeprofil fr n biogasmotorn
#V rmen antas f lja samma driftniv  som elproduktionen
def create_biogas_heat_profile(biogas_power_profile,
    biogas_heat_to_electric_ratio):
    return biogas_power_profile * biogas_heat_to_electric_ratio


#Produktionspris r gas
gas_production_cost = 0.8     #kr/kWh biogasenergi

#Emissionsfaktor r gas
ef_gas = -2.52    #g CO2e/kWh biogas


#biogasmotorn sk ag   p   60 % under dagen, 100 % under natten
def create_biogas_profile(
    time_index, biogas_power_max_kW):
    #Skapar en tidsserie f r biogasmotorns effekt.

```

```
605     biogas_power_profile= pd.Series(index=time_index, dtype=float)
606
607     for t in time_index:
608         #mellan 08 och 18  r  det ingen nattladdning, d  kan den g  p
                60 %
609         if 8 <= t.hour < 18:
610             biogas_power_profile.loc[t] = 0.60 * biogas_power_max_kW
611         #annars  g  r  den  p  100 %
612         else:
613             biogas_power_profile.loc[t] = 1.00 * biogas_power_max_kW
614
615     return biogas_power_profile
616
617
618       #ANDRA PARAMETRAR  P  DEP N
619 #Max total laddeffekt  p  dep n , antagande
620 depot_charging_limit_kW = 15000
621
622
623 #Laddniv er SoC, state of charge
624 min_soc= 30
625 bulk_soc= 80
626
627
628 #M jliga laddeffekter
629 normal_charge_power = 100
630 slow_charge_power = 50
631 fast_charge_power = 150
632
633
634 #Effekt och tid  f  r  uppv rming av bussarna
635 preheating_power_kW = 50
636 preheating_duration_h = 1
637
638 #Emissionsfaktor nordisk elmix
639 ef_electricity_mix = 59   #g CO2e/kWh
640
641
642       #HJ LPFUNKTION  F  R  EFFEKTUTTAG
643 #Funktionen s ger att:
644 #En buss kanske vill ladda med 100kW, men om dep n redan anv nder
        n stan all
645 #tillg nglig effekt kanske bara 40kW finns kvar
646 #Isf  f  r bussen bara 40kW
647 #allts :
648 #L gger till laddning om det finns kapacitet kvar  p  dep n.
649 #Returnerar faktisk effekt som kunde l ggas till.
650 def add_charging(load, t, power, depot_limit):
651     #load  r  en tidsserie med laddeffekt som redan anv nds varje timme
```

```
652     #t  r  aktuella tidpunkten
653     #power  r  hur mycket denna buss vill ha
654     #depot_limit  r  maximal effektsom hela dep n klarar
655
656
657     #Den faktiskt tillg ngliga effekten  r  dep ns  gr ns minus det som
            anv nds:
658     #loc hittar v rdet vid tiden t
659     available_capacity = depot_limit - load.loc[t]
660
661
662     #V ljer den minsta av bussens  nskade  effekt och kvarvarande
            dep kapacitet:
663     actual_power = min(power, available_capacity)
664
665
666     #Om dep n redan  r   verbelastad  vill vi inte l gga till negativ
            effekt:
667     actual_power = max(actual_power, 0)
668
669
670     #l gg till bussens faktiska laddning:
671     load.loc[t] += actual_power
672
673
674     #returnera hur mycket effekt bussen faktiskt fick:
675     return actual_power
676
677
678
679       # NDRADE  F RUTS TTNINGAR UNDER VINTERN
680 #Under vintern s  m ste bussarna v rmas upp innan de l mnar dep n
681 #laddverkningsgraden blir ocks  s mre p  vintern
682
683 #definiera vad som  r  vinterperiod
684 def is_winter_time(t):
685     #vinterperiod: n o v e m b e r mars
686     return t.month in [11, 12, 1, 2, 3]
687
688
689 def get_eta_for_period(time_index):
690     #om simuleringen inneh ller vinterdatum anv nds l gre
            laddverkningsgrad
691     if any(is_winter_time(t) for t in time_index):
692         return 0.80
693     else:
694         return 0.90
695
696 #DYGNSSIMULERING
```

```
697
698
699             #INPUT
700
701
702 #G r  s  att resultatet blir reproducerbart trots slumpvariabler:
703 np.random.seed(42)
704
705 #V lj simuleringsdygn,
706 #senare v ljer vi kanske ett p  sommaren, ett p  vintern
707 start = pd.Timestamp("2025-01-15␣08:00")
708 end = pd.Timestamp("2025-01-16␣08:00")
709 time_index = pd.date_range(start, end, freq="1h", inclusive="left")
710
711 #skapar tidsprofil f r n r biogasmotorn ska vara p  60% och n r 100%
712 biogas_power_profile = create_biogas_profile(time_index,
        biogas_power_max_kW)
713
714 #skapar tidsprofil f r fj rrv rmeproduktion
715 biogas_heat_profile = create_biogas_heat_profile(
716     biogas_power_profile, biogas_heat_to_electric_ratio)
717
718
719
720               #ELPRISDATA
721
722 #Anv nd syntetiska-medel-slump-elpris-funktionen
723 df_price_sim = create_synthetic_price_profile(
724     time_index, df_price, random_std_factor=0.15, seed=42)
725
726
727 if df_price_sim["price_SEK_kWh"].isna().any():
728     raise ValueError("Saknar␣elpris␣f r␣n gon␣timme␣i␣simuleringsdygnet
            .")
729
730 """
731 #Kontroll␣s ␣att␣r tt␣priser␣anv nds
732 print(df_price_sim[["price_SEK_kWh"]])
733 """
734 """
735 #␣F r␣test␣utan␣riktig␣prisdata␣ n :
736 #slumpa␣elpriser
737 df_price␣=␣pd.DataFrame(index=time_index)
738 df_price["price_SEK_kWh"]␣=␣np.random.uniform(0.4,␣2.0,␣len(time_index))
739 """
740
741                 #SOLDATA
742
743 #Anv nd syntetiska-medel-slump-soleffekt-funktionen f r valda tiden
```

```
744 df_solar_sim = create_synthetic_solar_power_profile(
745     time_index,df,random_std_factor=0.15,seed=42)
746 
747 if df_solar_sim["solar_power_kW"].isna().any():
748     raise ValueError("Saknar␣soleffekt␣f r␣n gon␣timme␣i␣
        simuleringsdygnet.")
749 
750 """
751 #Kontroll
752 print(df_solar_sim[["solar_power_kW"]])
753 """
754 
755 
756             #FJ RRV RMEPRISER
757 
758 #Anv nd syntetiska-medel-slump-fj rrv rme -funktionen f r vald period
759 df_heat_price_sim = create_synthetic_heat_price_profile(
760     time_index,df_heat_price,random_std_factor=0.15,seed=42)
761 
762 if df_heat_price_sim["heat_price_SEK_kWh"].isna().any():
763     raise ValueError("Saknar␣fj rrv rmepris␣f r␣n gon␣timme␣i␣
        simuleringen.")
764 
765 
766             #BUSSGENERERING
767 
768 #G r en tom lista f r antalet bussar p   dep n
769 buses = []
770 
771 
772 #F r de 46 st normalbussarna som finns ger vi de ett namn och laddniv
773 for i in range(n_bus_12m):
774     buses.append({
775         "bus_id": f"12m_{i}",
776         "battery_kWh": battery_12m})
777 
778 
779 #F r de 122 st ledbussarna som finns ger vi de ett namn och laddniv
780 for i in range(n_bus_18m):
781     buses.append({
782         "bus_id": f"18m_{i}",
783         "battery_kWh": battery_18m})
784 
785 
786 #skapar 2dimensionel dataframe av detta
787 buses = pd.DataFrame(buses)
788 
789 #f ljande anv dns f r att slumpa ankomst och avg ng:
790 simulation_day = start.normalize()
```

```
791
792 #Slumpa ankomst mellan 18:00 och 23:00, vilket  r  6 timmar
793 buses["arrival_time"] = [
794     simulation_day + pd.Timedelta(hours=np.random.randint(18, 24))
795     for _ in range(len(buses))]
796
797
798 #Slumpa avg ng mellan 04:00 och 08:00
799 departure_base = simulation_day + pd.Timedelta(days=1, hours=4)
800 buses["departure_time"] = [
801     departure_base + pd.Timedelta(hours=np.random.randint(0, 5))
802     for _ in range(len(buses))]
803
804
805 #Slumpa SoC vid ankomst mellan 10-30 %
806 buses["arrival_soc"] = np.random.uniform(10, 30, len(buses))
807
808
809 #Slumpa SoC vid avg ng mellan 85-100 %
810 buses["target_soc"] = np.random.uniform(85, 100, len(buses))
811
812 #Nu har vi en "tabell" d r raderna  r  varje buss
813 #och kolumnerna  r :
814 #buss-id, batteri-kapacitet, ankomsttid, avg ngstid,
815 #SoC vid ankomstiid och Soc vid avg ngstid
816
817
818 #Lunchladdning sker endast i undantagsfall
819 #ntagande: en liten andel bussar kommer in till dep n mitt p  dagen
820 lunch_charge_probability = 0.10   #10 % av bussarna
821 buses["has_lunch_charge"]=np.random.random(len(buses))<
        lunch_charge_probability
822
823 #Lunchf nster, vi v ljer att de kan ankomma mellan 11 och 13
824 #Slumpar n r under dessa timmar som bussrna kommer fram
825 buses["lunch_arrival_time"] = [
826     simulation_day + pd.Timedelta(hours=np.random.randint(11, 14))
827     if has_lunch else pd.NaT
828     for has_lunch in buses["has_lunch_charge"]]
829
830 #Slumpar n r de ska avg  fr n lunchladdningen
831 #antag att de f r 1 eller 2 h p  sig att lunchladda
832 #totalt sett kan allts  bussar lunchladda mellan 11 och 15.
833 buses["lunch_departure_time"] = [
834     arrival + pd.Timedelta(hours=np.random.randint(1, 3))
835     if pd.notna(arrival) else pd.NaT
836     for arrival in buses["lunch_arrival_time"]]
837
838
```

```
839
840                     #SIMULERING
841
842 #Skapar en tom effekt-kolumn f r dep n
843 charging_load = pd.Series(0.0, index=time_index)
844
845 #skapar lista om den inte f r  nskad  effekt och inte n r sin  nskade
        laddning
846 unmet_charging = []
847
848 #sortera bussarna s  att de med tidigast avg ng laddas f rst
849 buses = buses.sort_values("departure_time").reset_index(drop=True)
850
851 #sortera s  att de med l gst arrival_soc och avg ngstid laddas f rst
852 buses = buses.sort_values(
853     ["departure_time", "arrival_soc"]).reset_index(drop=True)
854
855 #G  igenom vajre buss
856 for _, bus in buses.iterrows():
857
858
859     #H mta parametrarna f r denna buss, allt f rutom battery  r
            slumpat
860     battery= bus["battery_kWh"]
861     soc= bus["arrival_soc"]
862     target_soc= bus["target_soc"]
863     arrival= bus["arrival_time"]
864     departure= bus["departure_time"]
865
866
867     #V lj laddverkningsgrad f r bussen
868     eta_bus = 0.8 if is_winter_time(departure) else 0.9
869
870     #S ger att "f nstret" f r slutladdningen startar 2 timmar innan
            avg ng
871     final_window_start = departure - pd.Timedelta(hours=2)
872
873     #F r vilka timmar som bussen faktiskt st r p  dep n:
874     available_hours = time_index[
875         (time_index >= arrival) & (time_index < departure)]
876     #Om bussen inte har n gra timmar hoppar vi  ver , men l r inte
            h nda
877     if len(available_hours) == 0:
878         continue
879
880
881             #1. Snabb initial laddning till 30 %
882
883     #Hur m nga kWh beh vs f r att n  30 %:
```

```
884     #E_till30= (SoC_min-SoC_current)/100 * Batteri/verkningsgrad
885     #Om den redan  r  mer laddad  n  30 % s  blir differensen negativ,
886     #och  d  ska den v ljа 0
887     energy_to_30= (max(min_soc - soc, 0) /100) * (battery/eta_bus)
888
889     #skapa tom lista  f  r initial-timmarna
890     #ska sedan anv ndas  s  att de inte kan anv ndas av n gon anan fas
891     initial_used_hours=[]
892
893     # G  igenom alla timmar som bussen  st r  p  dep n
894     for t in available_hours:
895         if energy_to_30 <= 0: #Om SoC  n  tt  30 % ska vi breakea
896             break
897
898         #Under initialladdningen v ljer vi standardeffekt:
899         power = normal_charge_power
900
901         #Anv nder hj lpfunktionen  f  r att se om standardeffekten  r
                tillg nglig
902         #delivered  r  den effekt som  r  tillg nglg
903         delivered=add_charging(charging_load, t, power,
                depot_charging_limit_kW)
904
905         #om bussen initialladdar; l gg till tiden
906         if delivered > 0:
907           initial_used_hours.append(t)
908
909         #Uppdaterar hur mycket energi som nu  r  kvar tills SoC=30
910         energy_to_30 -= delivered * dt
911
912         #uppdatera faktisk SoC
913         soc += (delivered * dt *eta_bus /battery) *100
914
915         #se till att SoC inte g r  ver  30 % i denna fas
916         soc =min(soc, min_soc)
917
918
919
920
921                 #2. Bulkladdning 30  80  %, styrd av elpris
922
923     #Best m  m let  f  r bulkladdningen
924     bulk_target = min(bulk_soc, target_soc)
925     #Best mmer hur mycket energi som beh vs enligt samma formel som
            tidigare
926     energy_bulk = (max(bulk_target - soc, 0)/100) * (battery/eta_bus)
927
928
929     #V ljer vilka timmar som  r  m jliga  f  r bulk
```

```
930     #bulkladdningen f r ske fr n ankomst till slutladdningsf nstret
931     bulk_hours= time_index[
932         (time_index >= arrival) & (time_index < final_window_start)]
933
934     #se till att bulktimmarna inte kan ske samtidigt som initialtimmarna
935     bulk_hours = bulk_hours.difference(pd.DatetimeIndex(
            initial_used_hours))
936
937     #Om det finns tid f r bulk och bussen beh ver bulkenergi:
938     if len(bulk_hours) > 0 and energy_bulk > 0:
939
940
941         #Sortera timmar efter elpris och befintlig last
942         #Skapar en tabell d r;
943         #raderna  r  varje m jlig laddtimme
944         candidates = pd.DataFrame(index=bulk_hours)
945         #kolumnnerna  r  spotpris och effektbelastningen p  dep n
946         candidates["price"] = df_price_sim.loc[bulk_hours, "price_SEK_kWh
                "]
947         candidates["current_load"] = charging_load.loc[bulk_hours]
948
949
950         #L gger till en kolumn med score
951         #l gt score  r  en bra timme att ladda
952         #L gre pris prioriteras,  ven  l gre last
953         #VAD SKA VI V LJA  F  R  VARIABEL? Tog 0.0001 nu
954         candidates["score"]=candidates["price"]+0.0001*candidates["
                current_load"]
955
956
957         #Sortera timmarna fr n  b st till s mst enligt score
958         sorted_hours = candidates.sort_values("score").index
959
960
961         #Ladda p  prioriterade timmar
962         #G r igenom de "b sta" timmarna f rst
963         for t in sorted_hours:
964             if energy_bulk <= 0:  #ska stoppa om bulkenergin  r  uppn dd
965                 break
966
967
968             #V lj 100 kW normalt, 50 kW om dep n redan  r  h rt
                    belastad
969             #om dep n  r  ver  80% av sin kapacitet laddas bussen
                    l ngsammare
970             if charging_load.loc[t] > 0.8 * depot_charging_limit_kW:
971                 power = slow_charge_power
972             else:
973                 power = normal_charge_power
```

```
974
975
976             #Anv nder hj lpfunktionen  f r att se effekten bussen
                    faktiskt  f r
977             delivered = add_charging(charging_load, t, power,
978                                      depot_charging_limit_kW)
979
980
981             #Uppdaterar kvarvarande energi till bulkenergin
982             energy_bulk -= delivered * dt
983
984             #uppdatera faktisk SoC
985             soc += (delivered * dt * eta_bus / battery) * 100
986             #Se till att SoC inte g r  ver  bulkladdnings-gr nsne i
                    denna fas
987             soc = min(soc, bulk_target)
988
989
990
991               #3. Slutladdning inf r avg ng
992
993
994     #v ljer de timmar som ligger inom f nstret:
995     final_hours = time_index[
996         (time_index >= final_window_start) & (time_index < departure)]
997
998
999
1000    #r kna ut hur mycket energi som  r  kvar till  nskad  laddning
1001    #g rs med samma formel som tidigare
1002    energy_final = (max(target_soc - soc, 0)/100) * (battery/eta_bus)
1003
1004
1005    #G  igenom slutladdningstimmarna
1006    for t in final_hours:
1007        if energy_final <= 0:   #ska stopaa om vi n tt upp till  nskad
                laddning
1008            break
1009
1010
1011        #anv nder hj lpfunk. f r att se om normal laddeffekt  r
                tillg nglig
1012        delivered = add_charging(charging_load, t, normal_charge_power,
1013                                 depot_charging_limit_kW)
1014
1015
1016        #uppdaterar kvarvarande laddbehov
1017        energy_final -= delivered * dt
1018
```

```
        soc += (delivered * dt * eta_bus / battery) * 100
        #se till att SoC inte g r  ver  avg ngs-soc i denna fas
        soc = min(soc, target_soc)


            #OM BUSSEN INTE  N  R  UPP TILL  NSKAD  LADDNING
    if soc < target_soc:
      unmet_charging.append({
          "bus_id": bus["bus_id"],
          "departure_time": departure,
          "final_soc": soc,
          "target_soc": target_soc,
          "missing_soc": target_soc - soc,
          "missing_energy_kWh": ((target_soc - soc) / 100) * battery
      })



        #4. Eventuell lunchladdning

    if bus["has_lunch_charge"]:

        lunch_arrival = bus["lunch_arrival_time"]
        lunch_departure = bus["lunch_departure_time"]

        lunch_hours = time_index[
            (time_index >= lunch_arrival) & (time_index < lunch_departure
                )]

        #Antagande: lunchladdningen  r  en kort och snabb laddning
        #exempelvis fr n 40 % till 60 %
        #slumpar ankomst-SoC mellan 35 och 55 %
        #slumpar avg ngs-SoCC mellan 55 och 75 %
        lunch_start_soc = np.random.uniform(35, 55)
        lunch_target_soc = np.random.uniform(55, 75)

        #Anv nder samma formel som innan f r att ber kna
            energi tg ngen
        energy_lunch = (
            max(lunch_target_soc - lunch_start_soc, 0)/100 * battery/
                eta_bus)


        for t in lunch_hours:
            #Avbryt om  nskad  energi  r  uppn dd
            if energy_lunch<= 0:
                break

            #Anv nder hj lpfunktionen f r att se om  nskad  effekt
```

```
                finns tillg.
            #Antar att lunchladdningnen  r  med snabbladdning
            delivered = add_charging(
                charging_load, t, fast_charge_power,
                    depot_charging_limit_kW)

            #Uppdaterar hur mycket energi som kr vs efter varje timme
            energy_lunch -= delivered * dt




          #5. Uppv rmning vintertid inf r  avg ng
    if is_winter_time(departure):

        #startar v rmas upp en timme innan avg ng
        preheat_start = departure - pd.Timedelta(hours=
            preheating_duration_h)

        #uppv rmnings-f nstret
        preheat_hours = time_index[
            (time_index >= preheat_start) & (time_index < departure)]

        #anv nder hj lpfunktionen  f r att se hur mycket som faktiskt
            finns tillg
        for t in preheat_hours:
            delivered = add_charging(
                charging_load, t, preheating_power_kW,
                    depot_charging_limit_kW)


#skapa data frame  f r de bussar som inte upppn dde  nskade  laddningen
unmet_charging = pd.DataFrame(unmet_charging)


if len(unmet_charging) > 0:
    print("L gsta avg ngs-SoC:",
          round(unmet_charging["final_soc"].min(), 2), "%")

    print("Genomsnittlig avg ngs-SoC f r ej fulladdade bussar:",
          round(unmet_charging["final_soc"].mean(), 2), "%")

    print("Genomsnittlig saknad SoC:",
          round(unmet_charging["missing_soc"].mean(), 2), "procentenheter
              ")


                        #RESULTAT
```

```
#skapar en resultat-tabell d r tiden  r  raderna
result = pd.DataFrame(index=time_index)

          #Ladd-resultat
#L gger in totala laddlasten f r bussarna som kolumn
result["charging_load_kW"] = charging_load
#Laddenergi f r bussarna
result["charging_energy_kWh"] = result["charging_load_kW"] * dt
#Total laddenergi f r bussarna
total_charging_energy_kWh = result["charging_energy_kWh"].sum()


      #Biogas-resultat
#L gger in biogasmotorns effekt som kolumn
result["biogas_power_kW"] = biogas_power_profile
#L gger in producerad elenergi fr n biogasen
result["biogas_electricity_kWh"] = result["biogas_power_kW"] * dt
total_biogas_electricity_kWh = result["biogas_electricity_kWh"].sum()
#L gg till hur mycket biogas-energi som beh vts
result["biogas_fuel_energy_kWh"] = (
    result["biogas_electricity_kWh"] / biogas_electrical_efficiency)
total_biogas_fuel_energy_kWh = result["biogas_fuel_energy_kWh"].sum()
#Kostnad f r producerad/anv nd biogas
result["biogas_cost_SEK"] = (result["biogas_fuel_energy_kWh"] *
    gas_production_cost)
#Total gaskostnad
total_biogas_cost_SEK = result["biogas_cost_SEK"].sum()


    #Fj rrv rme -resultat
#v rmeeffekt fr n biogasmotorn
result["biogas_heat_kW"] = biogas_heat_profile
#producerad v rmeenergi
result["biogas_heat_kWh"] = result["biogas_heat_kW"] * dt
total_biogas_heat_kWh = result["biogas_heat_kWh"].sum()
#L gg till fj rrv rmepris
result["heat_price_SEK_kWh"] = (df_heat_price_sim["heat_price_SEK_kWh"])
#Int kt fr n producerad fj rrv rme
result["heat_revenue_SEK"] = (
    result["biogas_heat_kWh"]* result["heat_price_SEK_kWh"])
#Total fj rrv rmeint kt
total_heat_revenue_SEK = result["heat_revenue_SEK"].sum()


      #Sol-resultat
#L gger in soleffekten som en kolumn
```

```
result["solar_power_kW"] = df_solar_sim["solar_power_kW"]


      #Eln ts-resultat
#L gger till netto mot eln tet som kolumn:
#positivt= import fr n n tet
#negativt = export/ verskott  fr n biogasmotor och solceller
result["net_grid_kW"] =(
    result["charging_load_kW"]
    - result["biogas_power_kW"]
    - result["solar_power_kW"])

  #L gg importen i en egen kolummn:
#.clip(lower=0) betyder att allt under 0 s tts till 0 (d  exporterar vi
    ju)
result["grid_import_kW"] = result["net_grid_kW"].clip(lower=0)
#importerad elenergi fr n n tet
result["grid_import_kWh"] = result["grid_import_kW"] * dt
#total importerad elenergi
total_grid_import_kWh = result["grid_import_kWh"].sum()

#L gg till elpris f r varje timme
result["price_SEK_kWh"] = df_price_sim["price_SEK_kWh"]

#Kostnad f r ink pt el varje timme
#(kW * 1h = kWh eftersom dt = 1 timme)
result["electricity_cost_SEK"] = (
    result["grid_import_kWh"] * result["price_SEK_kWh"])

#Total elkostnad f r immporterad el
total_electricity_cost = result["electricity_cost_SEK"].sum()


  #L gg exporten i en egen kolumn:
#byter tecken och g ra likadant som ovan
result["grid_export_kW"] = (-result["net_grid_kW"]).clip(lower=0)
#exporterad elenergi fr n n tet
result["grid_export_kWh"] = result["grid_export_kW"] * dt
#total exporterad elenergi
total_grid_export_kWh = result["grid_export_kWh"].sum()

#Int kt (om negativa priser blir det en kostnad) fr n exporterad el
#antagnande: export s ljs till spotpris
result["electricity_revenue_SEK"]=(
    result["grid_export_kWh"] * result["price_SEK_kWh"])
#Total elint kt
total_electricity_revenue = result["electricity_revenue_SEK"].sum()

  #Nettokostnad = kostnad - int kt
```

```
result["net_electricity_cost_SEK"]= (
    result["electricity_cost_SEK"]- result["electricity_revenue_SEK"])

#Total nettokostnad f r hela perioden
total_net_cost = result["net_electricity_cost_SEK"].sum()


          #Vinst/F rlust
#Positivt: Tj nar pengar
#Negativt: F rluror pengar
total_operating_profit_SEK = (
    total_electricity_revenue
    + total_heat_revenue_SEK
    - total_electricity_cost
    - total_biogas_cost_SEK)



#UTSL PPs-resultat

#Importerad el fr n n tet
result["grid_import_emissions_kgCO2e"] = (
    result["grid_import_kWh"] * ef_electricity_mix / 1000)

total_grid_import_emissions_kgCO2e = (
    result["grid_import_emissions_kgCO2e"].sum())


#Totala utsl pp fr n anv nd r gas
result["biogas_total_emissions_kgCO2e"] = (
    result["biogas_fuel_energy_kWh"] * ef_gas / 1000)

total_biogas_emissions_kgCO2e = (
    result["biogas_total_emissions_kgCO2e"].sum())


#F rdela biogasutsl pp mellan el och v rme
result["biogas_total_useful_energy_kWh"] = (
    result["biogas_electricity_kWh"] + result["biogas_heat_kWh"])

result["share_to_electricity"] = (
    result["biogas_electricity_kWh"] / result["
        biogas_total_useful_energy_kWh"])

result["share_to_heat"] = (
    result["biogas_heat_kWh"] / result["biogas_total_useful_energy_kWh"])

result["biogas_emissions_to_electricity_kgCO2e"] = (
    result["biogas_total_emissions_kgCO2e"] * result["
```

```
        share_to_electricity"])

result["biogas_emissions_to_heat_kgCO2e"] = (
    result["biogas_total_emissions_kgCO2e"] * result["share_to_heat"])


#f rdela biogaselen mellan laddning och export
result["local_generation_kWh"] = (
    result["biogas_electricity_kWh"] +
    result["solar_power_kW"] * dt)

result["biogas_share_of_local_generation"] = (
    result["biogas_electricity_kWh"] / result["local_generation_kWh"]
).fillna(0)

#Lokal el som anv nds direkt till bussarna
result["local_electricity_to_charging_kWh"] = (
    result["charging_energy_kWh"] - result["grid_import_kWh"])

result["biogas_electricity_to_charging_kWh"] = (
    result["local_electricity_to_charging_kWh"]
    * result["biogas_share_of_local_generation"])

result["biogas_electricity_to_export_kWh"] = (
    result["grid_export_kWh"]
    * result["biogas_share_of_local_generation"])


#F rdela biogasens elutsl pp mellan laddning och export
result["biogas_electricity_total_allocated_kWh"] = (
    result["biogas_electricity_to_charging_kWh"]
    + result["biogas_electricity_to_export_kWh"])

result["biogas_emissions_to_charging_kgCO2e"] = (
    result["biogas_emissions_to_electricity_kgCO2e"]
    * result["biogas_electricity_to_charging_kWh"]
    / result["biogas_electricity_total_allocated_kWh"]
).fillna(0)

result["biogas_emissions_to_export_kgCO2e"] = (
    result["biogas_emissions_to_electricity_kgCO2e"]
    * result["biogas_electricity_to_export_kWh"]
    / result["biogas_electricity_total_allocated_kWh"]
).fillna(0)


#Summeringar
total_heat_emissions_kgCO2e = result["biogas_emissions_to_heat_kgCO2e"].
    sum()
```

```
total_biogas_charging_emissions_kgCO2e = result["
    biogas_emissions_to_charging_kgCO2e"].sum()
total_biogas_export_emissions_kgCO2e = result["
    biogas_emissions_to_export_kgCO2e"].sum()
```

```
            #Printa ut resultaten
print("\nMax laddeffekt:", result["charging_load_kW"].max(), "kW")
print("Max import fr n n t:", result["grid_import_kW"].max(), "kW")
print("Max export/ verskott :", result["grid_export_kW"].max(), "kW")
print("Total importerad el:",round(total_grid_import_kWh, 2), "kWh")
print("Total exporterad el:",round(total_grid_export_kWh, 2), "kWh")
print("Total elkostnad:", round(result["electricity_cost_SEK"].sum(),2),
    "SEK")
print("Total elint kt:", round(result["electricity_revenue_SEK"].sum()
    ,2), "SEK")
print("Nettokostnad:", round(total_net_cost,2), "SEK")
print("Max soleffekt:", round(result["solar_power_kW"].max(), 2), "kW")
print("Producerad el fr n biogas:", round(total_biogas_electricity_kWh,
    2), "kWh")
print("Anv nd biogasenergi:", round(total_biogas_fuel_energy_kWh, 2), "
    kWh")
print("Total gaskostnad:", round(total_biogas_cost_SEK, 2), "SEK")
print("Producerad v rme fr n biogas:",round(total_biogas_heat_kWh, 2),
    "kWh")
print("Total fj rrv rmeint kt:",round(total_heat_revenue_SEK, 2), "SEK
    ")
print("Total laddenergi till bussarna:",round(total_charging_energy_kWh,
    2), "kWh")
print("Totalt driftresultat:",
      round(total_operating_profit_SEK, 2), "SEK")
print("Utsl pp fr n importerad el:",
      round(total_grid_import_emissions_kgCO2e, 2), "kg CO2e")
print("Totala utsl pp fr n biogas:",
      round(total_biogas_emissions_kgCO2e, 2), "kg CO2e")
print("Biogasutsl pp allokerade till fj rrv rme:",
      round(total_heat_emissions_kgCO2e, 2), "kg CO2e")
print("Biogasutsl pp allokerade till bussladdning:",
      round(total_biogas_charging_emissions_kgCO2e, 2), "kg CO2e")
print("Biogasutsl pp allokerade till exporterad el:",
      round(total_biogas_export_emissions_kgCO2e, 2), "kg CO2e")
```

```
print("Antal bussar som inte n dde  nskad  SoC:", len(unmet_charging))

if len(unmet_charging) > 0:
    print("Total saknad batterienergi:",
```

```
            round(unmet_charging["missing_energy_kWh"].sum(), 2), "kWh")

                                #PLOT

plt.figure(figsize=(12, 5))
plt.plot(result.index, result["charging_load_kW"], label="Charging load")
plt.plot(result.index, result["biogas_power_kW"], label="Biogas
    generation")
plt.plot(result.index, result["solar_power_kW"], label="Solar generation"
    )
plt.plot(result.index, result["net_grid_kW"], label="Net grid load")
plt.axhline(0, color="black", linewidth=0.8)

plt.ylabel("Effekt [kW]")
plt.title("Simulerad effektbalans vid stadsbussdep n")
plt.legend()
plt.grid()
plt.show()

#Se till att dygns-resultaten inte skrivs  ver  av m nads-resultaten
result_day = result.copy()
buses_day = buses.copy()
df_price_day = df_price_sim.copy()
df_solar_day = df_solar_sim.copy()

#  rssimulering   BUSSDEP N


              #INPUT


#G r  s   att resultatet blir reproducerbart trots slumpvariabler:
np.random.seed(42)


#V lj simuleringsdygn,
#senare v ljer vi kanske ett p   sommaren, ett p   vintern
start = pd.Timestamp("2025-01-01 00:00")
end = pd.Timestamp("2026-01-01 09:00")
time_index = pd.date_range(start, end, freq="1h", inclusive="left")

#skapar tidsprofil f r n r biogasmotorn ska g   p   60 % och 100 %
biogas_power_profile = create_biogas_profile(time_index,
    biogas_power_max_kW)

#skapar fj rrv rmeprofil
biogas_heat_profile = create_biogas_heat_profile(
    biogas_power_profile, biogas_heat_to_electric_ratio)
```

```
                #ELPRISDATA

#Anv nd syntetiska-medel-slump-elpris-funktionen f r valda tiden
df_price_sim = create_synthetic_price_profile(
    time_index, df_price, random_std_factor=0.15, seed=42)


if df_price_sim["price_SEK_kWh"].isna().any():
    raise ValueError("Saknar elpris f r n gon timme i simuleringsdygnet
        .")

"""
#Kontroll s att r tt priser anv nds
print(df_price_sim[["price_SEK_kWh"]])
"""
"""
# F r test utan riktig prisdata n :
#slumpa elpriser
df_price = pd.DataFrame(index=time_index)
df_price["price_SEK_kWh"] = np.random.uniform(0.4, 2.0, len(time_index))
"""

            #SOLDATA

#Anv nd syntetiska-medel-slump-soleffekt-funktionen f r vald period
df_solar_sim = create_synthetic_solar_power_profile(
    time_index, df, random_std_factor=0.15, seed=42)

if df_solar_sim["solar_power_kW"].isna().any():
    raise ValueError("Saknar soleffekt f r n gon timme i simuleringen."
        )


            #FJ RRV RMEPRISER

#Anv nd syntetiska-medel-slump-fj rrv rme -funktionen f r vald period
df_heat_price_sim = create_synthetic_heat_price_profile(
    time_index,df_heat_price,random_std_factor=0.15,seed=42)

if df_heat_price_sim["heat_price_SEK_kWh"].isna().any():
    raise ValueError("Saknar fj rrv rmepris f r n gon timme i
        simuleringen.")



            #BUSSGENERERING

```

```
#G r en tom lista f r antalet bussar p  dep n
buses = []


for day in pd.date_range(start.normalize(),
                         end.normalize() - pd.Timedelta(days=1),
                         freq="D"):
  #F r de 46 st normalbussarna som finns ger vi de ett namn och
      laddniv
  for i in range(n_bus_12m):
      buses.append({
          "bus_id": f"12m_{i}_{day.date()}",
          "battery_kWh": battery_12m,
          "arrival_time": day + pd.Timedelta(hours=np.random.randint(18,
              24)),
          "departure_time": day + pd.Timedelta(days=1, hours=np.random.
              randint(4, 9)),
          "arrival_soc": np.random.uniform(10, 30),
          "target_soc": np.random.uniform(85, 100)})


  #F r de 122 st ledbussarna som finns ger vi de ett namn och laddniv
  for i in range(n_bus_18m):
      buses.append({
          "bus_id": f"18m_{i}_{day.date()}",
          "battery_kWh": battery_18m,
          "arrival_time": day + pd.Timedelta(hours=np.random.randint(18,
              24)),
          "departure_time": day + pd.Timedelta(days=1, hours=np.random.
              randint(4, 9)),
          "arrival_soc": np.random.uniform(10, 30),
          "target_soc": np.random.uniform(85, 100)})


#skapar 2dimensionel dataframe av detta
buses = pd.DataFrame(buses)

#Nu har vi en "tabell" d r raderna  r  varje buss
#och kolumnerna  r :
#buss-id, batteri-kapacitet, ankomsttid, avg ngstid,
#SoC vid ankomstiid och Soc vid avg ngstid


#Lunchladdning sker endast i undantagsfall
#Antagande: en liten andel bussar kommer in till dep n mitt p  dagen
lunch_charge_probability=0.10   #10 % av bussarna lunchladdar
buses["has_lunch_charge"]=np.random.random(len(buses))<
    lunch_charge_probability

```

```
#Lunchankomst mellan 11 och 13 p  samma datum som nattankomsten
buses["lunch_arrival_time"] = [
    arrival.normalize() + pd.Timedelta(hours=np.random.randint(11, 14))
    if has_lunch else pd.NaT
    for arrival,has_lunch in zip(buses["arrival_time"],buses["
        has_lunch_charge"])]

#Lunchlanddning p g r i 1 2  timmar
buses["lunch_departure_time"] = [
    arrival + pd.Timedelta(hours=np.random.randint(1, 3))
    if pd.notna(arrival) else pd.NaT
    for arrival in buses["lunch_arrival_time"]]



                #SIMULERING


#Skapar en tom effekt-kolumn f r dep n
charging_load = pd.Series(0.0, index=time_index)


#g r tom lista f r de bussar som kanske inte f r  nskad  effekt och
#som d rf r kanske inte n r upp til  nskad  laddning
unmet_charging = []

#sortera bussarna s  att de med tidigast avg ng laddas f rst
buses = buses.sort_values("departure_time").reset_index(drop=True)

#sortera s  att de med l gst arrival_soc och avg ngstid laddas f rst
buses = buses.sort_values(
    ["departure_time", "arrival_soc"]).reset_index(drop=True)


#G  igenom vajre buss
for _, bus in buses.iterrows():


    #H mta parametrarna f r denna buss, allt f rutom battery  r
        slumpat
    battery= bus["battery_kWh"]
    soc= bus["arrival_soc"]
    target_soc= bus["target_soc"]
    arrival= bus["arrival_time"]
    departure= bus["departure_time"]


    #V lj laddverkningsgrad f r bussen
    eta_bus = 0.8 if is_winter_time(departure) else 0.9
```

```
1521
1522      #S ger att "f nstret" f r slutladdningen startar 2 timmar innan
              avg ng:
1523      final_window_start= departure - pd.Timedelta(hours=2)
1524
1525      #F r vilka timmar som bussen faktiskt st r p  dep n:
1526      available_hours = time_index[
1527          (time_index >= arrival) & (time_index < departure)]
1528      #Om bussen inte har n gra timmar hoppar vi  ver , men l r inte
              h nda
1529      if len(available_hours) == 0:
1530          continue
1531
1532
1533
1534              #1. Snabb initial laddning till 30 %
1535
1536
1537      #Hur m nga kWh beh vs f r att n  30 %:
1538      #E_till30= (SoC_min-SoC_current)/100 * Batteri/verkningsgrad
1539      #Om den redan  r  mer laddad  n  30 % s  blir differensen negativ,
1540      #och d  ska den v lja 0
1541      energy_to_30= (max(min_soc - soc, 0) /100) * (battery/eta_bus)
1542
1543      #skapa lista f r initialtimmarna
1544      #anv nds sedan s  att bulktimmarna inte kan ske samtidigt
1545      initial_used_hours =[]
1546
1547      #G  igenom alla timmar som bussen st r p  dep n
1548      for t in available_hours:
1549          if energy_to_30 <= 0: #Om SoC n tt 30 % ska vi breakea
1550              break
1551
1552
1553          #Under initialladdningen v ljer vi standardeffekt:
1554          power = normal_charge_power
1555
1556
1557          #Anv nder hj lpfunktionen f r att se om standardeffekten  r
                  tillg nglig
1558          #delivered  r  den effekt som  r  tillg nglg
1559          delivered=add_charging(charging_load, t, power,
                  depot_charging_limit_kW)
1560
1561          #om bussen laddar, l gg in tiden i listan
1562          if delivered > 0:
1563            initial_used_hours.append(t)
1564
1565          #Uppdaterar hur mycket energi som nu  r  kvar tills SoC=30
```

```
1566          energy_to_30 -= delivered * dt
1567
1568          #uppdatera faktisk soc
1569          soc += (delivered * dt * eta_bus /battery) *100
1570
1571          #se till att SoC inte g r  ver  30 % i denna fas
1572          soc = min(soc, min_soc)
1573
1574
1575
1576                #2. Bulkladdning 30  80  %, styrd av elpris
1577
1578
1579      #Best m m let  f r bulkladdningen
1580      bulk_target = min(bulk_soc, target_soc)
1581      #Best mmer hur mycket energi som beh vs enligt samma formel som
             tidigare
1582      energy_bulk = (max(bulk_target - soc, 0)/100) * (battery/eta_bus)
1583
1584
1585      #V ljer vilka timmar som  r  m jliga  f r bulk
1586      #bulkladdningen f r ske fr n ankomst till slutladdningsf nstret
1587      bulk_hours= time_index[
1588          (time_index >= arrival) & (time_index < final_window_start)]
1589
1590      bulk_hours = bulk_hours.difference(pd.DatetimeIndex(
             initial_used_hours))
1591
1592      #Om det finns tid f r bulk och bussen beh ver bulkenergi:
1593      if len(bulk_hours) > 0 and energy_bulk > 0:
1594
1595
1596          #Sortera timmar efter elpris och befintlig last
1597          #Skapar en tabell d r;
1598          #raderna  r  varje m jlig laddtimme
1599          candidates = pd.DataFrame(index=bulk_hours)
1600          #kolumnnerna  r  spotpris och effektbelastningen p  dep n
1601          candidates["price"] = df_price_sim.loc[bulk_hours, "price_SEK_kWh
                 "]
1602          candidates["current_load"] = charging_load.loc[bulk_hours]
1603
1604
1605          #L gger till en kolumn med score
1606          #l gt score  r  en bra timme att ladda
1607          #L gre pris prioriteras,  ven  l gre last
1608          #VAD SKA VI  V LJA  F  R  VARIABEL? Tog 0.0001 nu
1609          candidates["score"]=candidates["price"]+0.0001*candidates["
                 current_load"]
1610
```

```
1611
1612            #Sortera timmarna fr n b st till s mst enligt score
1613            sorted_hours = candidates.sort_values("score").index
1614
1615
1616            #Ladda p  prioriterade timmar
1617            #G r igenom de "b sta" timmarna f rst
1618            for t in sorted_hours:
1619                if energy_bulk <= 0:  #ska stoppa om bulkenergin  r  uppn dd
1620                    break
1621
1622
1623                #V lj 100 kW normalt, 50 kW om dep n redan  r  h rt
                        belastad
1624                #om dep n  r   ver  80% av sin kapacitet laddas bussen
                        l ngsammare
1625                if charging_load.loc[t] > 0.8 * depot_charging_limit_kW:
1626                    power = slow_charge_power
1627                else:
1628                    power = normal_charge_power
1629
1630
1631                #Anv nder hj lpfunktionen f r att se effekten bussen
                        faktiskt f r
1632                delivered = add_charging(charging_load, t, power,
1633                                         depot_charging_limit_kW)
1634
1635
1636                #Uppdaterar kvarvarande energi till bulkenergin
1637                energy_bulk -= delivered * dt
1638
1639                soc += (delivered * dt * eta_bus /battery) *100
1640
1641                #se till att SoC inte g r  ver   nskad  bulk i denna fas
1642                soc = min(soc, bulk_target)
1643
1644
1645
1646              #3. Slutladdning inf r avg ng
1647
1648
1649        #v ljer de timmar som ligger inom f nstret:
1650        final_hours = time_index[
1651            (time_index >= final_window_start) & (time_index < departure)]
1652
1653
1654        #r kna ut hur mycket energi som  r  kvar till  nskad  laddning
1655        #g rs med samma formel som tidigare
1656        energy_final = (max(target_soc - soc, 0)/100) *(battery/eta_bus)
```

```
1657
1658
1659      # G   igenom slutladdningstimmarna
1660      for t in final_hours:
1661          if energy_final <= 0:   #ska stopaa om vi  n  tt  upp till  nskad
                  laddning
1662              break
1663
1664
1665          #anv nder hj lpfunk. f r att se om normal laddeffekt  r
                  tillg nglig
1666          delivered = add_charging(charging_load, t, normal_charge_power,
1667                                   depot_charging_limit_kW)
1668
1669
1670          #uppdaterar kvarvarande laddbehov
1671          energy_final -= delivered * dt
1672
1673          soc += (delivered * dt *eta_bus /battery) * 100
1674
1675          soc=min(soc, target_soc)
1676
1677
1678            #OM BUSSEN INTE  N  R  UPP TILL   NSKAD   LADDNING
1679      if soc < target_soc:
1680          unmet_charging.append({
1681              "bus_id": bus["bus_id"],
1682              "departure_time": departure,
1683              "final_soc": soc,
1684              "target_soc": target_soc,
1685              "missing_soc": target_soc - soc,
1686              "missing_energy_kWh": ((target_soc - soc) / 100) * battery
1687          })
1688
1689                #4. Eventuell lunchladdning
1690
1691      if bus["has_lunch_charge"]:
1692
1693          #ankomst-och avg ngstider:
1694          lunch_arrival = bus["lunch_arrival_time"]
1695          lunch_departure = bus["lunch_departure_time"]
1696
1697          lunch_hours = time_index[
1698              (time_index >= lunch_arrival) & (time_index < lunch_departure
                      )]
1699
1700          #slumpar ankomst-SoC mellan 35 och 55 %
1701          #slumpar avg ngs-SoV mellan 55 och 75 %
1702          lunch_start_soc = np.random.uniform(35, 55)
```

```
1703          lunch_target_soc = np.random.uniform(55, 75)
1704
1705          #Ber knar hur mycket energi som beh vs med tidigare formel
1706          energy_lunch = (
1707              max(lunch_target_soc - lunch_start_soc, 0)/100 * battery/
                      eta_bus)
1708
1709          for t in lunch_hours:
1710              #om energin  r  uppn dd  s   avbryt
1711              if energy_lunch <= 0:
1712                  break
1713
1714              #anv nder hj lpfunktionen  f r  att se om  nskad  effekt  r
                       tillg.
1715              #antar att lnchladdning sker med snabbladdning-effekt
1716              delivered = add_charging(
1717                  charging_load,t,fast_charge_power,depot_charging_limit_kW
                          )
1718
1719              #uppdaterar hur mycket energi som kr vs varje timme
1720              energy_lunch -=delivered * dt
1721
1722
1723
1724
1725                  #5. Uppv rmning vintertid inf r  avg ng
1726
1727      if is_winter_time(departure):
1728
1729          #startar v rmas upp en timme innan avg ng
1730          preheat_start = departure - pd.Timedelta(hours=
                  preheating_duration_h)
1731
1732          #uppv rmnings-f nstret
1733          preheat_hours = time_index[
1734              (time_index >= preheat_start) & (time_index < departure)]
1735
1736          #anv nder hj lpfunktionen  f r  att se hur mycket som faktiskt
                  finns tillg
1737          for t in preheat_hours:
1738              delivered = add_charging(
1739                  charging_load, t, preheating_power_kW,
                          depot_charging_limit_kW)
1740
1741
1742  #skapa data frame  f r de bussar som inte  n r  nskad  laddning
1743  unmet_charging = pd.DataFrame(unmet_charging)
1744
1745  if len(unmet_charging) > 0:
```

```
    print("L gsta␣avg ngs-SoC:",
          round(unmet_charging["final_soc"].min(), 2), "%")

    print("Genomsnittlig␣avg ngs-SoC␣f r␣ej␣fulladdade␣bussar:",
          round(unmet_charging["final_soc"].mean(), 2), "%")

    print("Genomsnittlig␣saknad␣SoC:",
          round(unmet_charging["missing_soc"].mean(), 2), "procentenheter
              ")


                        #RESULTAT


#skapar en resultat-tabell d r tiden  r  raderna
result = pd.DataFrame(index=time_index)

    #Ladd-resultat
#L gger in totala laddlasten som kolumn
result["charging_load_kW"] = charging_load
#Laddenergi f r bussarna
result["charging_energy_kWh"] = result["charging_load_kW"] * dt
#Total laddenergi f r bussarna
total_charging_energy_kWh = result["charging_energy_kWh"].sum()


    #Biogas-resultat
#L gger in biogasmotorns effekt som kolumn
result["biogas_power_kW"] = biogas_power_profile
#L gger in producerad elenergi fr n biogasen
result["biogas_electricity_kWh"] = result["biogas_power_kW"] * dt
total_biogas_electricity_kWh = result["biogas_electricity_kWh"].sum()
#L gg till hur mycket biogas-energi som beh vts
result["biogas_fuel_energy_kWh"] = (
    result["biogas_electricity_kWh"] / biogas_electrical_efficiency)
total_biogas_fuel_energy_kWh = result["biogas_fuel_energy_kWh"].sum()
#Kostnad f r producerad/anv nd biogas
result["biogas_cost_SEK"] = (result["biogas_fuel_energy_kWh"] *
    gas_production_cost)
#Total gaskostnad
total_biogas_cost_SEK = result["biogas_cost_SEK"].sum()


    #Fj rrv rme-resultat
#v rmeeffekt fr n biogasmotorn
result["biogas_heat_kW"] = biogas_heat_profile
#producerad v rmeenergi
result["biogas_heat_kWh"] = result["biogas_heat_kW"] * dt
total_biogas_heat_kWh = result["biogas_heat_kWh"].sum()
```

```
#L gg till fj rrv rmepris
result["heat_price_SEK_kWh"] = (df_heat_price_sim["heat_price_SEK_kWh"])
#Int kt fr n producerad fj rrv rme
result["heat_revenue_SEK"] = (
    result["biogas_heat_kWh"]* result["heat_price_SEK_kWh"])
#Total fj rrv rmeint kt
total_heat_revenue_SEK = result["heat_revenue_SEK"].sum()


    #Sol-resultat
#L gg in soleffekten som kolumn
result["solar_power_kW"] = df_solar_sim["solar_power_kW"]


    #Eln ts-resultat
#L gger till netto mot eln tet som kolumn:
#positivt= import fr n n tet
#negativt = export/ verskott  fr n biogasmotor och solceller
result["net_grid_kW"] =(
    result["charging_load_kW"]
    - result["biogas_power_kW"]
    - result["solar_power_kW"])


#L gg importen i en egen kolummn:
#.clip(lower=0) betyder att allt under 0 s tts till 0 (d  exporterar vi
    ju)
result["grid_import_kW"] = result["net_grid_kW"].clip(lower=0)
#importerad elenergi fr n n tet
result["grid_import_kWh"] = result["grid_import_kW"] * dt
#total importerad elenergi
total_grid_import_kWh = result["grid_import_kWh"].sum()


#L gg till elpris f r varje timme
result["price_SEK_kWh"] = df_price_sim["price_SEK_kWh"]

#Kostnad f r ink pt el varje timme
result["electricity_cost_SEK"] = (
    result["grid_import_kWh"] * result["price_SEK_kWh"])

#Total elkostnad f r import f r hela simuleringen
total_electricity_cost = result["electricity_cost_SEK"].sum()


#L gg exporten i en egen kolumn:
#byter tecken och g ra likadant som ovan
result["grid_export_kW"] = (-result["net_grid_kW"]).clip(lower=0)
#exporterad elenergi till n tet
```

```
result["grid_export_kWh"] = result["grid_export_kW"] * dt
#total exporterad elenergi
total_grid_export_kWh = result["grid_export_kWh"].sum()

#Int kt (om negativa priser blir det en kostnad) fr n exporterad el
#antagande: export s ljs till spotpris
result["electricity_revenue_SEK"] = (
    result["grid_export_kWh"] * result["price_SEK_kWh"])

#Nettokostnad = kostnad - int kt
result["net_electricity_cost_SEK"] = (
    result["electricity_cost_SEK"] - result["electricity_revenue_SEK"])

#Totaler f r hela perioden
total_electricity_revenue = result["electricity_revenue_SEK"].sum()
total_net_cost = result["net_electricity_cost_SEK"].sum()


      #Vinst/F rlust
#Positivt: Tj nar pengar
#Negativt: F rluror pengar
total_operating_profit_SEK = (
    total_electricity_revenue
    + total_heat_revenue_SEK
    - total_electricity_cost
    - total_biogas_cost_SEK)


                    #UTSL PPs -resultat

#Importerad el fr n n tet
result["grid_import_emissions_kgCO2e"] = (
    result["grid_import_kWh"] * ef_electricity_mix / 1000)

total_grid_import_emissions_kgCO2e = (
    result["grid_import_emissions_kgCO2e"].sum())


#Totala utsl pp fr n anv nd r gas
result["biogas_total_emissions_kgCO2e"] = (
    result["biogas_fuel_energy_kWh"] * ef_gas / 1000)

total_biogas_emissions_kgCO2e = (
    result["biogas_total_emissions_kgCO2e"].sum())


#F rdela biogasutsl pp mellan el och v rme
result["biogas_total_useful_energy_kWh"] = (
    result["biogas_electricity_kWh"] + result["biogas_heat_kWh"])
```

```
1890
1891 result["share_to_electricity"] = (
1892     result["biogas_electricity_kWh"] / result["
         biogas_total_useful_energy_kWh"])
1893
1894 result["share_to_heat"] = (
1895     result["biogas_heat_kWh"] / result["biogas_total_useful_energy_kWh"])
1896
1897 result["biogas_emissions_to_electricity_kgCO2e"] = (
1898     result["biogas_total_emissions_kgCO2e"] * result["
         share_to_electricity"])
1899
1900 result["biogas_emissions_to_heat_kgCO2e"] = (
1901     result["biogas_total_emissions_kgCO2e"] * result["share_to_heat"])
1902
1903
1904 #f rdela biogaselen mellan laddning och export
1905 result["local_generation_kWh"] = (
1906     result["biogas_electricity_kWh"] +
1907     result["solar_power_kW"] * dt)
1908
1909 result["biogas_share_of_local_generation"] = (
1910     result["biogas_electricity_kWh"] / result["local_generation_kWh"]
1911 ).fillna(0)
1912
1913 #Lokal el som anv nds direkt till bussarna
1914 result["local_electricity_to_charging_kWh"] = (
1915     result["charging_energy_kWh"] - result["grid_import_kWh"])
1916
1917 result["biogas_electricity_to_charging_kWh"] = (
1918     result["local_electricity_to_charging_kWh"]
1919     * result["biogas_share_of_local_generation"])
1920
1921 result["biogas_electricity_to_export_kWh"] = (
1922     result["grid_export_kWh"]
1923     * result["biogas_share_of_local_generation"])
1924
1925
1926 #F rdela biogasens elutsl pp mellan laddning och export
1927 result["biogas_electricity_total_allocated_kWh"] = (
1928     result["biogas_electricity_to_charging_kWh"]
1929     + result["biogas_electricity_to_export_kWh"])
1930
1931 result["biogas_emissions_to_charging_kgCO2e"] = (
1932     result["biogas_emissions_to_electricity_kgCO2e"]
1933     * result["biogas_electricity_to_charging_kWh"]
1934     / result["biogas_electricity_total_allocated_kWh"]
1935 ).fillna(0)
1936
```

```
result["biogas_emissions_to_export_kgCO2e"] = (
    result["biogas_emissions_to_electricity_kgCO2e"]
    * result["biogas_electricity_to_export_kWh"]
    / result["biogas_electricity_total_allocated_kWh"]
).fillna(0)


#Summeringar
total_heat_emissions_kgCO2e = result["biogas_emissions_to_heat_kgCO2e"].sum()
total_biogas_charging_emissions_kgCO2e = result["biogas_emissions_to_charging_kgCO2e"].sum()
total_biogas_export_emissions_kgCO2e = result["biogas_emissions_to_export_kgCO2e"].sum()









    #Printa resultatet

print("\nMax laddeffekt:", result["charging_load_kW"].max(), "kW")
print("Max import fr n n t:", result["grid_import_kW"].max(), "kW")
print("Max export/ verskott :", result["grid_export_kW"].max(), "kW")
print("Total importerad el:",round(total_grid_import_kWh, 2), "kWh")
print("Total exporterad el:",round(total_grid_export_kWh, 2), "kWh")
print("Total kostnad f r ink pt el:", round(total_electricity_cost, 2), "SEK")
print("Total elint kt:", round(total_electricity_revenue, 2), "SEK")
print("Nettokostnad:", round(total_net_cost, 2), "SEK")
print("Max soleffekt:", round(result["solar_power_kW"].max(), 2), "kW")
print("Producerad el fr n biogas:", round(total_biogas_electricity_kWh, 2), "kWh")
print("Anv nd biogasenergi:", round(total_biogas_fuel_energy_kWh, 2), "kWh")
print("Total gaskostnad:", round(total_biogas_cost_SEK, 2), "SEK")
print("Producerad v rme fr n biogas:",round(total_biogas_heat_kWh, 2), "kWh")
print("Total fj rrv rmeint kt:",round(total_heat_revenue_SEK, 2), "SEK")
print("Total laddenergi till bussarna:",round(total_charging_energy_kWh, 2), "kWh")
print("Totalt driftresultat:",
      round(total_operating_profit_SEK, 2), "SEK")
print("Utsl pp fr n importerad el:",
      round(total_grid_import_emissions_kgCO2e, 2), "kg CO2e")

print("Totala utsl pp fr n biogas:",
```

```
      round(total_biogas_emissions_kgCO2e, 2), "kg CO2e")

print("Biogasutsl pp allokerade till fj rrv rme:",
      round(total_heat_emissions_kgCO2e, 2), "kg CO2e")

print("Biogasutsl pp allokerade till bussladdning:",
      round(total_biogas_charging_emissions_kgCO2e, 2), "kg CO2e")

print("Biogasutsl pp allokerade till exporterad el:",
      round(total_biogas_export_emissions_kgCO2e, 2), "kg CO2e")

print("Antal bussar som inte n dde  nskad  SoC:", len(unmet_charging))

if len(unmet_charging) > 0:
    print("Total saknad batterienergi:",
          round(unmet_charging["missing_energy_kWh"].sum(), 2), "kWh")

                            #PLOT

plt.figure(figsize=(12, 5))
plt.plot(result.index, result["charging_load_kW"], label="Charging load")
plt.plot(result.index, result["biogas_power_kW"], label="Biogas
    generation")
plt.plot(result.index, result["solar_power_kW"], label="Solar generation"
    )
plt.plot(result.index, result["net_grid_kW"], label="Net grid load")
plt.axhline(0, color="black", linewidth=0.8)

plt.ylabel("Effekt [kW]")
plt.title("Simulerad effektbalans vid stadsbussdep n")
plt.legend()
plt.grid()
plt.show()


#Se till att m nadssimuleringen och dygnssimulering  r  separerade
result_month = result.copy()
buses_month = buses.copy()
df_price_month = df_price_sim.copy()
df_solar_month = df_solar_sim.copy()
```

Referenser

Referenser

N. Chi Johansson and A. Sandgren. Emissionsfaktor för nordisk elmix år 2021–2023. https://www.ivl.se/press/nyheter/2025-09-25-klimatpaverkan-for-kopt-el-har-minskat-visar-nya-berakningar.html, 2026. Hämtad: 2026-14-04.

Energigas Sverige. Biogas och miljön, u.å. URL https://www.energigas.se/fakta-om-gas/biogas/biogas-och-miljon/. Hämtad 2026-05-22.

Energinet. Energi data service, n.d. URL https://www.energidataservice.dk/tso-electricity/elspotprices#metadata-info. Hämtad 27 april 2026.

Gamla Uppsala Buss. Biogas, u.å. b. URL https://www.gub.se/gub/miljo/fornybara-drivmedel/biogas/. Hämtad 2026-04-22.

Gamla Uppsala Buss. El, u.å. c. URL https://www.gub.se/gub/miljo/fornybara-drivmedel/el/. Hämtad 28 april 2026.

Gamla Uppsala Buss. Hvo, u.å. d. URL https://www.gub.se/gub/miljo/fornybara-drivmedel/hvo/. Hämtad 2026-05-04.

Keolis. Keolis sverige, n.d. URL https://www.keolis.se/om-keolis.html. Hämtad 19 maj 2026.

MAN Truck & Bus. Technical data: MAN lion's city 12 e. [Datablad], u.å. a. URL https://www.man.eu/content/dam/man/countries/doc/bw-master/bus/datenblaetter/man-lions-city/en/man_lions_city_datenblatt_12-e_EN.pdf/_jcr_content/renditions/original./man_lions_city_datenblatt_12-e_EN.pdf. Hämtad 2026-04-28.

MAN Truck & Bus. Technical data: MAN lion's city 18 e. [Datablad], u.å. b. URL https://www.man.eu/content/dam/man/countries/doc/bw-master/bus/datenblaetter/man-lions-city/en/man_lions_city_datenblatt_18-E_EN.pdf. Hämtad 2026-04-28.

Naturvårdsverket. Värden för att beräkna utsläppsminskning. https://www.naturvardsverket.se/496762/globalassets/amnen/klimatomstallning/klimatklivet/dokument/klimatklivet-varden-for-att-berakna-utslappsminskning.pdf, 2026. Hämtad: 2026-04-17.

M. Nilsson. Forskning: Därför fungerar elbilar sämre i vinterkyla. https://carup.se/forskning-darfor-fungerar-elbilar-samre-i-vinterkyla/, Jan. 2023. Hämtad 12 maj 2026.

L. Nordin. Personlig korrespondens, 2026. Sektionschef Biogas, Uppsala Vatten och Avfall AB.

Region Uppsala. Energiportalen, n.d. URL https://energiportalregionuppsala.se/. Hämtad 25 april 2026.

Svenska kraftnät. Elområden, 2025. URL https://www.svk.se/om-kraftsystemet/om-elmarknaden/elomraden/. Hämtad 19 maj 2026.

Sveriges riksbank. Sök räntor och valutakurser, n.d. URL https://www.riksbank.se/sv/statistik/rantor-och-valutakurser/sok-rantor-och-valutakurser/?s=g130-SEKEURPMI&a=D&from=2022-09-29&to=2025-09-30&fs=3#result-section. Hämtad 18 maj 2026.