Aalto-yliopisto
Sähkötekniikan
korkeakoulu

# Dynaamiset rakennusvaipparatkaisut Suomen ilmasto-olosuhteissa

Jaakko Ketomäki, työelämäprofessori

Heikki Ihasalo, työelämäprofessori

# Sisällys

# 1 Johdanto

Rakennuksen vaippa, joka sisältää seinät, ikkunat ja ovet, katon sekä lattian on nähty perinteisesti staattiseksi rakennuksen ulkokuoreksi, joka ei muutu ympäristön tai käyttäjien vaatimuksien mukaan. Kuitenkin dynaamisia rakennusvaipparatkaisuja on jo olemassa ja niitä on myös sovellettu erilaisissa kohteissa. Tässä hankkeessa keskitytään erityisesti aktiivisiin varjostusratkaisuihin (esim. kaihtimet ja markiisit), sähköisiin ikkuna-avaajiin, aurinkoaktiivisiin julkisivuihin ja sähkökromaattisiin lasituksiin. Rakennusvaipan dynaamisuus mahdollistaa sen älykkään ohjaamisen. Suomessa nämä ratkaisut ovat tällä hetkellä harvinaisia ja koostettua tietoa ratkaisujen mahdollisuuksista Suomen ilmasto-olosuhteissa on vähän tarjolla. Monet ratkaisut ovat luonteeltaan sellaisia, joissa älykkäällä ulkovaipan ohjaamisella toivotaan säästöjä lämmityksen- tai viilennyksen energiakulutukseen tai parannusta sisäilma- ja valaistusolosuhteisiin. Energiatehokkuusvaatimuksien kasvaessa on tarve etsiä uusia keinoja tehostaa energiankäyttöä ja dynaamiset rakennusvaipparatkaisut voisivat olla tällainen keino. Vaipparatkaisuilla voidaan myös hallita sisäolosuhteita nykyistä paremmin ja esim. mahdollistaa tehokas päivänvalon käyttö.

Tässä hankkeessa luotiin katsaus käytössä oleviin sekä mahdollisiin dynaamisiin vaipparatkaisuihin ja alan toimijoihin sekä tehtiin simulointeja erilaisilla dynaamisilla vaipparatkaisuilla. Lisäksi hankkeen puitteissa on valmistunut kaksi aiheeseen liittyvää opinnäytetyötä:

- Kandidaatin tutkielma *Adaptiiviset julkisivuratkaisut* tarkasteli rakennusten suorituskyvyn parantamista adaptiivisten julkisivuratkaisujen avulla. Työn tavoitteena oli selvittää adaptiivisten julkisivujen määritelmä ja ominaispiirteet sekä kromogeenisten lasituksien soveltuvuus Helsingin ilmastoon. Työ rajattiin yleisimpiin julkisivuissa käytettäviin teknologioihin, jotka kykenevät muuttamaan julkisivujen ominaisuuksia adaptiivisesti. Työ toteutettiin kirjallisuustutkimuksena.
- Diplomityö, jonka teemana oli *Applying Machine Learning to Develop Black-box Control Model of Active Double-Skin Façade*. Siinä kehitettiin itseoppivaa ohjausalgoritmia kaksoisjulkisivuratkaisuun, joka sisältää sälekaihtimet ja tuloilmaikkunan.

Hankkeessa kartoitettiin myös Suomessa dynaamisten julkisivujen alueella toimivia tahoja. Suomessa on kaksi alaan liittyvää yhdistystä eli tässä tapauksessa yritysten etujärjestöä. Ne pyrkivät viemään eteenpäin alan kehitystä laajassa mittakaavassa edustaen samalla tietysti jäsenyrityksiään:

- **Aurinkosuojaus ry on** perustettu vuonna 1980 ja se toimii nimensä mukaisesti aurinkotuojauksen toimialalla. Tähän kuuluvat markiisit ja kaihtimet, mutta mukana on myös niiden ohjaamiseen keskittyneitä yrityksiä. Aurinkosuojaus ry:stä hankkeessa haastateltiin toiminnanjohtaja Petri Rokkasta.
- **Tasolasiyhdistys ry:n** tarkoituksena on toimia lasialalla ja niitä lähellä toimivien yritysten toimialajärjestönä. Jäsenistö kostuu pääosin perinteisiä ikkunalaseja valmistavista ja asentavista yrityksistä. Tasolasiyhdistyksestä hankkeessa haastateltiin toiminnanjohtajaa Jenni Heikkilää.

# 2 Dynaamisen rakennusvaipan teknologiota

Seuraavassa esitellään lyhyesti erilaiset dynaamiset ulkovaipan ratkaisut ja arvioidaan niiden käyttöpotentiaalia Suomen olosuhteissa. Dynaamisen ulkovaipan tarkoituksena on hallita rakennukseen pääsevää lämpösäteilyä ja päivänvaloa. Lämpimänä vuodenaikana liikaa lämpösäteilyä halutaan välttää ja näin säästää rakennuksen viilennyskustannuksissa. Kylmänä vuodenaikana taas auringosta tuleva lämpö on hyödyksi ja sen avulla toivotaan voitavan laskea rakennuksen lämmityskustannuksia. Tämä tietysti edellyttää sitä, että varsinainen lämmitysjärjestelmä kykenee reagoimaan ulkopuolelta tulevaan lämpöön riittävän nopeasti.

Päivänvalon osalta tavoitteena on usein maksimoida rakennukseen tulevan päivänvalon määrä mutta toisaalta eliminoida päivänvalon aiheuttama häikäisy. Joskus nämä vaatimukset ovat keskenään ristiriitaisia. Myös päivänvalon osalta valaistuksen energiansäästö riippuu siitä, voiko valaistusjärjestelmä reagoida päivänvalon vaihteluihin valaistustehoa muuttamalla.

Perinteisiä ratkaisuja päivänvalon ja lämpösäteilyn hallintaan ovat markiisit, rullaverhot, sälekaihtimet.

**Kaihtimet**

Puhuttaessa kaihtimista on aluksi täsmennettävä kaihtimen tyyppi ja sijainti ulkovaipassa. Perinteinen sälekaihdin on yleensä Suomessa sijoitettu ulomman ja sisemmän ikkunaelementin väliin. Tässä paikassa kaihdin on tehokas estämään auringon aiheuttamaa häikäisyä, mutta lämpösäteilyn osalta teho jää osin vajaaksi. Tämä siksi, että lämpösäteily on jo päässyt sisälle rakennukseen ulomman ikkunan läpi. Tässäkin tapauksessa suojaus auringon lämmöltä on vielä tehokasta verrattuna tilanteeseen ilman kaihtimia. Joka tapauksessa perinteinenkin kaihdin estää tehokkaasti lämpösäteilyn kohdistumisen vain yhdelle alueelle ja sen lämmittävä vaikutus tuntuu tasaisemmin koko tilassa. Perinteiset kaihtimet ovat harvoin automaattisesti ohjattavia.

Kaihtimet voidaan sijoittaa myös kokonaan ikkunan sisäpuolelle. Tällaisia ratkaisuja on toteutettu usein esimerkiksi julkisissa tiloissa, joissa ikkunoiden korkeus on suuri. Tällöin kaihtimet ovat usein pystysuuntaisia lamellikaihtimia. Näiden tehtävänä on usein paitsi päivänvalon hallinta myös näkösuojana toimiminen. Näitäkään kaihtimia ei yleensä ohjata automaattisesti.

Kuva 1. Lamellikaihtimia (kuva: Artickaihdin)

**Screen-kaihtimet**

Screen-kaihdin oikeastaan perinteisen sälekaihtimen ja markiisin yhdistelmä, jossa yhdistyvät niiden hyvät puolet. Screen-kaihdin on osin läpinäkyvä hieman rullaverhoa muistuttava tuote, joka voidaan asentaa sekä ikkunoiden sisä- että ulkopuolelle. Ohjaukseen liitetään yleensä automatiikka samoilla periaatteilla kuin markiisienkin tapauksessa. Ohjaus voi siis tapahtua kellon tai auringonsäteilyn tai yleensä molempien avustamana. Ohjauksessa on mahdollista käyttää erilaisia ennakoivia ohjaustapoja.

Aurinkosuojauksen rintamalla ja etenkin uusissa kohteissa screenkaihdin on vallitseva tapa toteuttaa aurinkosuojaus. Mikäli kaihtimien asennus otetaan huomioon jo suunnitteluvaiheessa, on niiden integroiminen ikkunarakenteisiin mahdollista.


Kuva 2. Screenkaihtimia (kuvat Sunsystems ja Kaihdinpukkila)

**Markiisit**

Markiisi on ikkunan ulkopuolinen yleensä kankaasta valmistettu katos. Markiisi voidaan rakentaa joko ikkunoissa tyypillisenä sivumarkiisina, jolloin se on tuettu myös alareunastaan tai liukumarkiisina, joka soveltuu parhaiten korkeisiin ikkunoihin. Lippamarkiisia käytetään lähinnä vain terassien aurinkosuojauksessa.


Kuva 3. Ikkunamarkiisiratkaisuja (kuvat: Markiisipalvelu, Tamar)

Markiisien ohjaus voi olla manuaalista tai automaattista. Dynaamisesta ulkovaipasta puhuttaessa ohjaus toteutetaan automaattisesti. Ohjaus voi perustua kelloon, auringon säteilyn määrään tai tuulen voimakuuteen tai kuten usein on tapana, näihin kaikkiin. Automaattisessa

ohjauksessa markiisit avataan ikkunoiden eteen, kun auringon säteily on riittävän voimakasta ja ne suljetaan, kun säteilytaso on taas riittävän matala. Tuulen voimakuuden mukaan ohjautuvuus on mukana lähinnä estämässä markiisin rikkoontumista.

Parhaimmillaan markiisi on varsin tehokas suoja auringon säteilyä vastaan. Lisäksi markiisi on hyvin suunniteltuna tyylikkään näköinen myös modernissa rakentamisessa. Markiisit ovat kuitenkin uusissa kohteissa antaneet tilaa uudemmille ratkaisuille, kuten edellä mainitut screen-kaihtimet.

**Aurinkoaktiiviset julkisivut**

Dynaamisiin ulkovaippoihin voidaan lukea myös aurinkoaktiiviset julkisivut. Tällaisia ovat esimerkiksi julkisivut, joihin on integroitu aurinkosähköpaneeleita. Tämänkaltaisia asennuksia on Suomessa tehty jo muutamia. Hyvänä puolena on se, että aurinkopaneeli toimii samalla julkisivupinnoitteena eikä erillisiä, katolle asennettavia paneeleita välttämättä tarvita. Paneelit antavat rakennukselle parhaassa tapauksessa myös persoonallisen ulkonäön. Julkisivuun asennettavat aurinkopaneelit on tietysti huomioitava jo rakennusta suunniteltaessa, sillä niiden suuntaaminen jälkikäteen on mahdotonta. Myös erilaiset varjostukset on huomioitava tavallista kattoasennusta tarkemmin.

Aurinkoaktiivisena julkisivuna voidaan pitää myös erilaisia viherratkaisuja, joissa kasvillisuuden avulla pienennetään auringon aiheuttamaa lämpökuormitusta.

Julkisivujen rakenteisiin on mahdollista liittää myös sopivia faasimuutosmateriaaleja kuten parafiinivahaa. Pehmentyessään se sitoo auringon lämpöä ja vastaavasti luovuttaa sen hitaasti taas kovettuessaan tasaten näin tehokkaasti sisätilojen lämpötilan vaihtelua.

**Kromogeeniset lasitukset**

Kromogeeniset lasitukset pyrkivät minimoimaan rakennuksien kokonaisenergiankulutusta säätämällä termo-optisia ominaisuuksiaan tummenemalla tai kirkastumalla. Kromogeeniset lasitukset perustuvat kromogeenisiin materiaaleihin, joiden termo-optisia ominaisuuksia voidaan säädellä ulkoisen ärsykkeen, kuten sähkökentän, avulla. Kromogeeniset lasitukset voidaan jakaa teknologioiden mukaan elektrokromiseen lasitukseen, nestekidelasitukseen, suspensiohiukkaslasitukseen ja termokromiseen lasitukseen.

Kromogeenisiä lasituksia voidaan käyttää ikkunoissa, kattolasituksessa ja ovissa. Ulkolasituksen lisäksi sitä käytetään sisätiloissa tarjoamaan yksityisyydensuojaa esimerkiksi neuvotteluhuoneisiin. Ulkoisia lasituksia voidaan ohjata sääolosuhteet ja auringonsäteily huomioiden ja sisätilojen lasituksia kytkimillä tai vaikkapa puheohjauksella.

Kuten on arvattavissakin, kromogeenisten lasituksien teknologia on varsin monimutkaista verrattuna tavallisiin ikkunalaseihin. Ne ovat kuitenkin maailman laajuisesti kasvussa olevaa liiketoimintaa. Alkuinvestoinnin hintaa näissä lasituksissa voi kompensoida niiden ohjaamisen helppous ja ainakin periaatteessa huoltovapaa toiminta.

Suomessa on jo joitain kromogeenisella tekniikalla toteutettuja lasituksia, joskin niiden merkitys on enemmänkin näkösuojana toimiminen kuin varsinainen julkisivun lämmönhallinta.

**Sähkölämmitteiset lasit**

Tietyissä tapauksissa sähkölämmitteisten lasien käyttö on perusteltua, vaikka rakennus muuten olisikin lämmitetty kaukolämmöllä tai esimerkiksi lämpöpumpuilla. Tällainen tilanne on esimerkiksi silloin, kun korkeassa aulatilassa on lattiasta kattoon asti ulottuvat ikkunat. Tällöin kylmään vuodenaikaan ikkunoiden yläpäästä virtaa kylmää ilmaa lasin pintaa pitkin ja tämä aiheuttaa tilassa vedon tunteen. Perinteisesti ongelma on ratkaistu asettamalla ikkunan alle lattiaan lämpimän ilman puhaltimet. Mikäli lasi kuitenkin lämmitetään, niin vältytään vedolta ja lisäksi siivousta hankaloittavia ilmanvaihtoaukkoja lattiassa ei tarvita.

Sähkölämmitteisen lasin vaikutus rakennuksen energialuokitukseen on tietysti negatiivinen etenkin sellaisissa kohteissa, joissa lämmitys tapahtuu esim. kaukolämmöllä tai lämpöpumpuilla. Kuitenkin yleensä tilojen, joissa sähkölämmitteiset lasit ovat tarpeen, on rakennuksissa varsin vähän ja lopullinen merkitys rakennuksen energialuokkaan on varsin pieni.

# 3 Dynaaminen ulkovaippa rakentamisessa

## 3.1 Suunnittelun merkitys aurinkosuojauksessa ja päivänvalon hyödyntämisessä

Hankkeessa tehtyjen haastattelujen tuloksena voidaan sanoa, että aurinkosuojaukseen on nykymaailmassa kiinnitettävä entistä enemmän huomiota ja onneksi niin enenevässä määrin tehdäänkin. Aurinkosuojaus voi rakennushankkeissa helposti tulla suunniteltavaksi liian myöhäisessä vaiheessa, jolloin päädytään lopulta päälle liimattuihin ratkaisuihin. Hyvällä ja ennakoivalla suunnittelulla aurinkosuojaus ja muutkin dynaamisen ulkovaipan ratkaisut saataisiin integroitua mukavasti rakenteisiin.

Hyvällä suunnittelulla on mahdollista myös tehostaa päivänvalon käyttöä osana tilojen valaistusta. Valaistusjärjestelmät on jo pitkän aikaa ollut mahdollista varustaa automatiikalla, joka säätää keinovalon määrää sen mukaan, paljonko tilaan saadaan luonnonvaloa. Mikäli aurinkosuojaus ja sen ohjaus on toteutettu asianmukaisesti, voidaan päivänvalon osuutta valaistuksessa lisätä. Tämä tietysti mahdollistaa energiansäästön mutta ennen kaikkea luonnonvalon riittävä määrä parantaa rakennuksen sisäolosuhteita ja tekee niistä miellyttävämmät.

Päivänvalon käyttö osana valaistusta vaatii sekin suunnittelua. Suunnitelmissa on mahdollista hyödyntää valaistussimulointeja. Niiden laatiminen on entistä helpompaa, koska rakennusten suunnitelmat ja mallinnukset ovat joka tapauksessa olemassa sähköisessä, simulointiohjelmien ymmärtämässä muodossa. Siitä huolimatta päivänvalon simuloinnit saattavat usein jäädä tekemättä kustannussyistä.

## 3.2 Dynaamiset ulkovaipan ratkaisut Suomessa

Suomalaisia tai Suomessa toimivia yrityksiä löytyy myös dynaamisten julkisivujen alueelta. Esimerkiksi moottoroituja kaihtimia ohjaavia laitteistoja tarjoavat useat kotiautomaatiota tarjoavat yritykset, kaihtimia ja markiiseja valmistavia yrityksiä on myös muutamia.

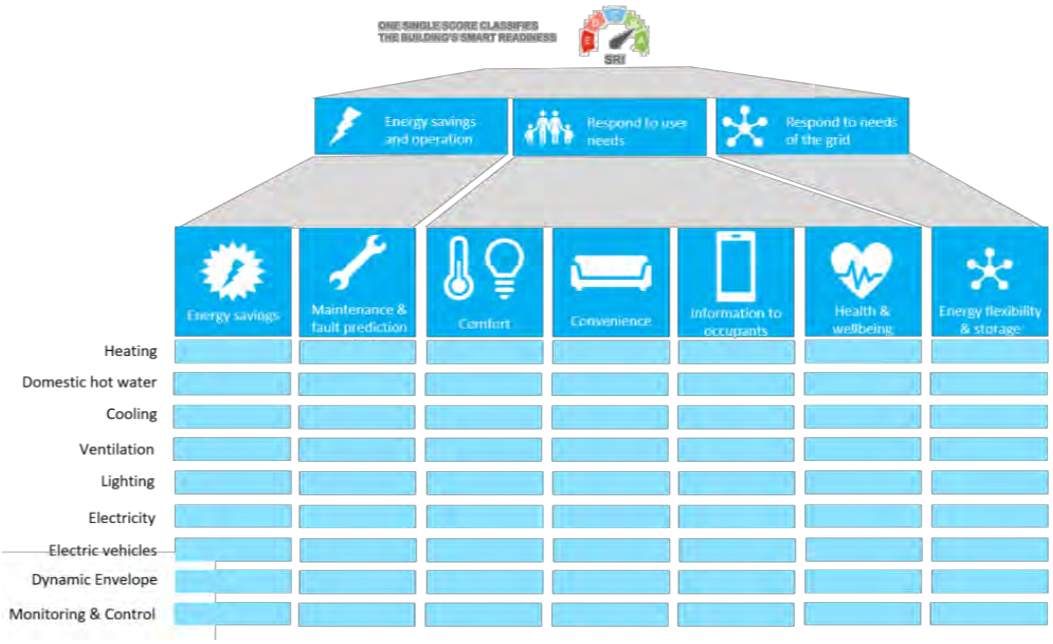
Varsinaisesti ikkunoihin integroitavia ratkaisuja kuten sähkökromaattisia laseja tai on/off -tyyppisiä laseja käytetään Suomessa varsin vähän. Neuvotteluhuoneissa ja vastaavissa sisätiloissa ok/off -laseja on käytössä ja niille löytyy myös toimittaja Suomesta. Ulkoikkunoissa tummuvien lasien kysyntä ja sitä mukaa myös käyttö on vähäistä.

Aurinkopaneelien integrointi ikkunoihin on mahdollista, mutta sitä käytetään varsin vähän. Paneeli kuitenkin vähentää ikkunan läpinäkyvyyttä ja se koetaan usein huonoksi asiaksi. Aurinkopaneelit ovat enemmänkin osa muuta julkisivua, mikäli niitä halutaan asentaa pystyasentoon.

## 3.3 Rakennusten älyindikaattori -Smart Readiness Indicator (SRI)

Dynaaminen ulkovaippa on tullut näkyville myös energiatehokuusdirektiivin (EPBD) määrittelemässä rakennusten älyindikaattorissa (SRI)

Rakennusten älyindikaattorilla arvioidaan rakennusten älyvalmiuksia yhtenevän menetelmän avulla energiatehokkuuden, rakennuksen käyttäjän sekä energian kysyntäjouston näkökulmasta. Älyindikaattorista on säädetty rakennusten energiatehokkuusdirektiivin (EPBD) artiklassa 8. Arvion kohteena ovat ennen kaikkea rakennuksen talotekniset järjestelmät ja indikaattorin avulla voidaan arvioida niiden tasoa energiatehokkuuden, rakennuksen käyttäjän sekä energian kysyntäjouston näkökulmasta.




*Kuva 3. Rakennusten älyindikaattorin (SRI) arviointimatriisi. Eräänä arvioitavana aihealueena on dynaaminen ulkovaippa*

Älyindikaattorin laskennassa käydään läpi rakennuksen tekniset järjestelmät yksi kerrallaan ja niiden toiminnallisuuksien perusteella annetaan yksittäiselle järjestelmälle arvosana. Yksittäisten järjestelmien arviosta muodostetaan lopullinen älyindikaattorin arvosana. Joidenkin järjestelmien kohdalla on mahdollista käyttää myös kansallisia painotuksia.

Yhtenä arvioitavana kohtana älyindikaattorin arvona laskettaessa on dynaaminen rakennusvaippa (dynamic envelope). Siitä arvioidaan kolmea tekijää:

**Ikkunoiden aurinkosuojaus**. Parhaan arvosanan saa, mikäli aurinkosuojaus toimii ennakoivasti esim. sääennusteisiin pohjautuen. Myös aurinkosuojaus yhdistettynä valaistuksen ja ilmanvaihdon ohjaukseen on arvioinnissa arvostettu toteutustapa. Moottoroitu ohjaus, joka

perustuu mittaustietoihin auringon säteilystä, on enemmänkin perustason teknologiaa – joskin se on tietysti pisteiltään parempi kuin pelkkä manuaalinen ohjaus

**Avattavien ikkunoiden ohjaus**. Tässä parhaan arvion saa järjestelmä, jossa ikkunoiden avautuminen on keskitetysti ohjattu sekä mahdollistaa yötuuletuksen. Ikkunoiden ohjauksen on myös oltava kytkettynä lämmitys- ja viilennysjärjestelmien ohjaukseen.

**Järjestelmän toiminnan raportointi.** Mikäli automaattisten ikkunoiden avautumisesta, virhetoiminnoista sekä olosuhteiden historiatiedoista on mahdollista saada raportit, on tästä kohdasta jaossa parhaat pisteet. Pelkkä vikatilanteiden raportointi on enemmänkin tavanomaista teknologiaa.

Älyindikaattori käsittää menetelmänä koko EU-alueen. Siksi menetelmässä on myös osioita, joiden merkitys on Suomen olosuhteissa (toistaiseksi) vähäinen. Esimerkki tällaisesta ovat mainitut automaattisesti avautuvat ikkunat. Sellaisia ei Suomessa juurikaan käytetä, mutta joissain Euroopan maissa ne ovat yleisiä.

Ilmaston muutos voi kuitenkin tuoda myös Suomen oloihin merkittävää lisätarvetta rakennusten viilennysratkaisuille. Siksi on hyvä, että asioita mietitään jo hieman ennakkoon kartoittaen Suomessa toimivia ratkaisuja.

# 4 Simulointituloksia

Hankkeen simuloinnit tehtiin Aalto-yliopiston LVI-laboratoriossa IDA-ICE -simulointiohjelmalla. Simulointien kohteena olivat uudehko 2012 rakennettu sekä vanhempi 1960-luvulta peräisin oleva asuinkerrostalo sekä uusi ja vanhempi toimistorakennus. Simuloinneissa huomioitiin dynaamisen ulkovaipan passiivisia menetelmiä (manuaaliset kaihtimet, aurinkosuojaikkunat) sekä aktiivisia ratkaisuja kuten automaattisesti säätyvät kaihtimet, automaattisesti avautuvat ikkunat ja markiisit sekä elektrokromaattiset ikkunat. Simuloinneissa kiinnitettiin huomiota rakennuksen energiankulutukseen sekä sisäolosuhteisiin.

Simulointien tuloksista voidaan tehdä joitain johtopäätöksiä, joita on esitetty seuraavassa.

**Kerrostalo**

Kerrostaloissa passiivisten ja aktiivisten ratkaisuvaihtoehtojen välillä ei ole suuria eroja energiankäytössä. Prosentuaaliset säästöt jäähdytyksen sähkön kulutuksessa ovat suuret (vanhassa kerrostalossa parhaalla ratkaisulla 67 % ja uudehkossa 83 %), mutta absoluuttisesti pienet (vanhassa 1,4 kWh/m$^2$ ja uudehkossa 1,9 kWh/m$^2$). Uudessa kerrostalossa lämmönkulutus hieman kasvaa johtuen aurinkosäteilyn lämmitysvaikutuksen vähentymisestä talviaikaan ja toisaalta vaikutus jäähdytystarpeeseen on vähäinen johtuen perustilanteen laadukkaista ikkunoista. Uudessa kerrostalossa jäähdytystarve häviää lähes kokonaan sähköisellä ikkuna-avauksella.

Jos jäähdytystä ei olisi käytettävissä uudessa kerrostalossa, ovat sähköllä avattavat ikkunat sisälämpötilan hallinnan kannalta selkeästi tehokkaampi ratkaisu verrattuna muihin passiivisiin ja aktiivisiin varjostusratkaisuihin. Ikkunoiden avauksella saadaan raikasta ilmaa sisätiloihin, mutta tähän menetelmään liittyy haasteita koskien ulkoilman laatua ja melua. Määräyksien mukaan olosuhdesimuloinneissa ikkuna-avauksia ei tämän takia voida hyödyntää.

Ilmastonmuutoksen myötä sisäilman kuumenemisen riski kasvaa ja jäähdytystarve lisääntyy. Siten kesäajan huoneilman lämpötilan hallinnan merkitys korostuu tulevaisuudessa.

Simuloinneissa verrattiin myös olosuhteita asunnoissa ja erityisesti havainnoitiin eroja aktiivisten ja passiivisten aurinkosuojausmenetelmien välillä. Vanhassa asuinkerrostalossa eroa aktiivisten ja passiivisten menetelmien välillä ei juurikaan löydetty. Sen sijaan uuden talon tapauksessa aktiiviset menetelmät olivat hieman tehokkaampia.

Asuinrakennuksessa erilaisilla dynaamisen vaipan ratkaisuilla ei ollut merkittävää vaikutusta lämmitysenergian kulutukseen. Vanhan rakennuksen tapauksessa ikkunoiden vaihtaminen hieman pudotti lämmityksen tarvetta, mutta uudessa rakennuksessa se taas nousi johtuen aurinkosuojaikkunan huonommasta lämmönläpäisystä.

**Toimistorakennus**
Vanhassa toimistorakennuksessa energiatehokkaimpia ratkaisuja ovat aurinkosuojatut ikkunat (kaihtimilla tai ilman) ja sähkökromaattiset ikkunat säästön ollessa n. 5% sähkössä ja 6,5-9,2 % lämmössä. Näiden ratkaisujen avulla vähennetään sekä lämmitys- (parempi ikkunan U-arvo) että jäähdytystarvetta (aurinkosuojaus). Muilla ratkaisuilla vähennetään jäähdytystarvetta, mutta lämmitystarve lisääntyy johtuen aurinkosäteilyn lämmitysvaikutuksen vähentymisestä talviaikaan.

Uudessa toimistorakennuksessa ikkunan laatu on lähtötilanteessa sen verran hyvä, että säästöjä ei samalla tavalla saada kuin vanhassa toimistorakennuksessa. Energiatehokkaimmaksi ratkaisuksi osoittautui avattava ikkuna sen säästäessä 6 % sähköenergiaa siten, että lämmitystarve pysyi ennallaan. Jäähdytysenergiassa säästetään myös aurinkosuojatuilla ikkunoilla (kaihtimilla tai ilman) ja sähkökromaattiset ikkunoilla, mutta näissä lämmönkulutus kasvoi verrattuna lähtötilanteeseen johtuen aurinkosäteilyn lämpövaikutuksen vähentymisestä. Säästöt sähkössä olivat n. 3 % aurinkosuojatuilla ja sähkökromaattisilla ikkunoilla.

Kaikilla vaihtoehdoilla voidaan parantaa sisäilman lämpötilatasoa. Tehokkaimmat ratkaisut vanhassa toimistorakennuksessa ovat aurinkosuojatut ikkunat (kaihtimilla tai ilman), sähkökromaattiset ikkunat ja ulkoiset automaattiset kaihtimet. Näiden lisäksi uudessa toimistossa tehokkaimmaksi ratkaisuksi osoittautui avattavat ikkunat.

# 5 Johtopäätökset

Julkisivujen tekniikka sähköistyy. Vielä jokin aika sitten auringon häikäisyä pyrittiin estämään vain käsin säädettävillä sälekaihtimilla. Ne ovat tietysti käypää tekniikka vielä tänäänkin, mutta rinnalle ovat tulleet jo sähköisesti ohjattavat ja automaattiset ratkaisut kuten screen-kaihtimet, joissain tapaukissa markiisien sähköinen ohjaus tai jopa tummuvien lasin käyttö. Tummuvat lasit voivat joissain tapauksissa lisätä myös yksityisyyden suojaa. Myös varsinaiset rakennusten julkisivut voidaan valjastaa tuottamaan sähköä aurinkopaneelien avulla.

Sähköiset ratkaisut tarjoavat automaattisella säädöllä parempia ja mukavampia sisäolosuhteita, joiden hallinta perustuu myös ennakoivaan ohjaukseen. Julkisivuelementtien automaattinen ohjaus tekee niiden käytöstä myös mukavampaa ja ennen kaikkea jokapäiväistä toimintaa.

Ilmaston lämpeneminen johtaa siihen, että myös Suomessa erilaiset aurinkosuojaukseen liittyvät teknologiat on otettava huomioon jo rakennuksia tai niiden korjauksia suunniteltaessa. Näin sisäolosuhteet saadaan pidettyä mukavina myös tulevaisuudessa ja vältytään myös mahdolliselta lisäjäähdytyksen mukanaan tuomilta investointi- ja energiakustannuksilta.

# Liite 1 Kandidaatintutkielma Adaptiiviset julksivut

# ADAPTIIVISET JULKISIVURATKAISUT 2020

Samuli Karvinen

Aalto-yliopisto
Sähkötekniikan korkeakoulu
Automaatio- ja systeemitekniikka

**Samuli Karvinen**

# Adaptiiviset julkisivuratkaisut

Kandidaatintyö

01.12.2020

Työn ohjaaja: Työelämäprofessori Heikki Ihasalo

AALTO-YLIOPISTO
SÄHKÖTEKNIIKAN KORKEAKOULU
PL 13000
00076 AALTO

TEKNIIKAN KANDIDAATINTYÖN TIIVISTELMÄ

Tekijä: Samuli Karvinen

Työn nimi: Adaptiiviset julkisivuratkaisut

Tutkinto-ohjelma: Automaatio- ja systeemitekniikka

Päiväys: 01.12.2020 Sivumäärä: 36

Vastuuopettaja: Yliopistonlehtori Pekka Forsman

Ohjaaja: Työelämäprofessori Heikki Ihasalo

Kieli: Suomi

Rakennuksen energiatehokkuuteen vaikuttaa erityisesti rakennuksen ja ympäristön rajapintana toimiva rakennuksen julkisivu, jonka kautta lämmön ja massavirran siirtymistä tapahtuu. Tässä kandidaatintyössä tarkastellaan rakennusten suorituskyvyn parantamista adaptiivisten julkisivuratkaisujen avulla, joista tarkemman tarkastelun alle on valittu kromogeeniset lasitukset, sillä perinteisissä staattisissa julkisivuissa läpinäkyvät elementit ovat tyypillisesti suurin potentiaalinen kehityksen kohde energian säästämiselle. Kandidaatintyön tavoitteena on selvittää adaptiivisten julkisivujen määritelmä ja ominaispiirteet sekä kromogeenisten lasituksien soveltuvuus Helsingin ilmastoon. Työ on rajattu yleisimpiin julkisivuissa käytettäviin teknologioihin, jotka kykenevät muuttamaan julkisivujen ominaisuuksia adaptiivisesti. Työ toteutetaan kirjallisuustutkimuksena ja sen aineiston etsimisessä on hyödynnetty muun muassa Web of Science- sekä Scopus-tietokantaa ja Google Patents-hakupalvelua.

Adaptiivisella julkisivulla tarkoitetaan mukautuvaa, monipuolisista järjestelmistä muodostuvaa rakennuksen vaippaa, joka pyrkii parantamaan rakennuksen suorituskykyä muuttamalla ajan myötä ominaisuuksiaan, toimintojaan tai käyttäytymistään säädelläkseen rakennukseen pääsevää luonnonvalon määrää, lämpöä ja massavirtaa. Adaptiiviset julkisivut voidaan jakaa neljään eri luokkaan; aurinkoaktiivisiin julkisivuihin, dynaamisiin varjostimiin, aktiivisesti tuulettuviin julkisivuihin sekä kromogeenisiin julkisivuihin.

Kromogeeniset lasitukset pyrkivät minimoimaan rakennuksien kokonais-energiankulutusta säätämällä termo-optisia ominaisuuksiaan tummenemalla tai kirkastumalla. Kromogeeniset lasitukset perustuvat kromogeenisiin materiaaleihin, joiden termo-optisia ominaisuuksia voidaan säädellä ulkoisen ärsykkeen, kuten sähkökentän, avulla. Kromogeenisistä lasituksista tarkastelun kohteena olivat elektrokrominen lasitus, nestekidelasitus, suspensiohiukkaslasitus ja termokrominen lasitus.

Markkinoilla olevien kromogeenisten lasitusten suorituskykyjä ja Helsingin ilmaston optimaalisia ominaisuuksia vertaillessa todettiin, että suspensiohiukkaslasitus soveltuu Helsingin ilmastoon parhaiten sen termo-optisten ominaisuuksien, kylmyyden sietokyvyn ja vähäisen virran-kulutuksen ansiosta. Suspensiohiukkaslasituksen suorituskyvyssä on kuitenkin vielä parantamisen varaa, varsinkin auringon säteilyn kokonaisläpäisevyyden moduloinnissa.

Lasitusten optimaalisia termo-optisia ominaisuuksia tarkasteltiin ainoastaan Helsingin ilmastossa, mikä poikkeaa paljon muusta Suomesta. Olisi siis hyvä, jos optimaalisia ominaisuuksia simuloitaisiin myös muissa Suomen ilmastoissa. Adaptoitumisvälejä tarkastellessa huomattiin, että mitä lyhyempi adaptoitumisväli oli, sitä enemmän säästettiin energiaa. Tarkastelussa oli kuitenkin ainoastaan päivittäin adaptoituvat lasitukset, joten suorituskyvyn optimointia voitaisiin tutkia lyhyemmillä adaptoitumisväleillä.

Avainsanat: adaptiiviset julkisivut, älykkäät rakennusvaipat, kromogeeniset lasitukset, älyikkunat, rakennuksen energiatehokkuus

# Sisällysluettelo

# Lyhenteet

| | |
|---|---|
| EC | Electrochromic (Elektrokrominen) |
| PDLC | Polymer-dispersed liquid crystal (Polymeerillä hajautettu nestekide) |
| PSLC | Polymer-stabilized liquid crystal (Polymeerillä stabiloitu nestekide) |
| SPD | Suspended Particle Device (Suspensiohiukkaslasitus) |
| TC | Thermochromic (Termokrominen) |

# Muuttujat

| | | |
|---|---|---|
| $n_o$ | Nestekiteiden tavallinen taitekerroin | [–] |
| $n_e$ | Nestekiteiden epätavallinen taitekerroin | [–] |
| $n_m$ | Matriisin taitekerroin | [–] |
| $V_{th}$ | Kynnysjännite | [V] |
| $V_{sat}$ | Saturaatiojännite | [V] |
| $\tau_c$ | Termokromisen lasituksen kriittinen lämpötila | [°C] |
| $T_{vis}$ | Näkyvän valon läpäisevyys | [–] |
| *g-arvo* | Auringon säteilyn kokonaisläpäisevyyskerroin | [–] |
| *U-arvo* | Lämmönläpäisykerroin | [W/(m$^2$·K)] |
| $K_e$ | Säteilyn valotehokkuus | [–] |

# 1 Johdanto

Maailman väkiluku ylittää vuonna 2020 arviolta yli 7,8 miljardin ihmisen rajan, mikä tulee kasvamaan samassa tahdissa pysyttäessä 8,6 miljardiin ihmiseen vuoteen 2030 mennessä [1]. Väkiluvun kasvaessa energiantarve lisääntyy, jolloin myös kasvihuonepäästöt lisääntyvät. Samalla kun maailman energiantarve ja ympäristön päästöjen määrä kasvaa, Euroopan unioni tavoittelee vuoteen 1990 verrattuna 40 % pienempiä kasvihuonepäästöjä ja 30 % energiansäästöä vuoteen 2030 mennessä. Jotta energiaa voidaan säästää, täytyy tiedostaa mihin sitä kuluu. Tällä hetkellä puolet Euroopan unionin energiatuotannosta kuluu pelkästään rakennusten lämmittämiseen ja viilentämiseen. [2] Euroopan unionin kunnianhimoisen tavoitteiden saavuttamiseksi on siis rakennusten energiatehokkuutta parannettava merkittävästi.

Rakennusten energiatehokkuutta voidaan parantaa minimoimalla niiden energian kokonaiskulutusta, jonka suurimpia tekijöitä ovat lämmitys, viilennys sekä valaistus. Rakennuksen energian kokonaiskulutukseen vaikuttaa erityisesti rakennuksen ja ympäristön rajapintana toimiva rakennuksen julkisivu (*façade*), jonka lämmön ja massavirran siirtymistä tapahtuu. [3] Nykypäivänä julkisivut ovat pääasiassa staattisia järjestelmiä, eli ne eivät kykene mukautumaan muuttuviin ympäristöolosuhteisiin tai käyttäjien tarpeisiin [4], [5]. Kylmissä ilmastoissa julkisivuihin valitaan yleensä materiaaleja, jotka eristävät lämpöä mahdollisimman hyvin vähentäen lämmityskustannuksia talvisin. Tämä voi kuitenkin olla ongelma kesäisin, jolloin helteillä rakennusten sisäilma voi saavuttaa sietämättömän kuumuuden ilman energiaa kuluttavaa ilmastointia. Staattisten julkisivujen energiansäästöt ovat myös hyvin rajalliset ja siksi tarvitaankin adaptiivisia julkisivuja (*adaptive façades*), sillä ne mahdollistavat sisätilan mukavuuden maksimoinnin vähentämällä samalla rakennuksien energiankulutusta [3]. Adaptiivisia julkisivuja voidaan ajatella ympäristön valvojina [6]: ne moduloivat lämpöä ja massavirtaa aktiivisesti tai passiivisesti muuttamalla palautuvasti termo-optisia ominaisuuksiaan tai operointiaan. Adaptiiviset julkisivut kykenevät mukautumaan muuttuviin sisä- ja ulkotilan ympäristöolosuhteisiin, kuten auringon säteilyyn, ilman lämpötilaan, tuulen nopeuteen ja suuntaan sekä sisäisiin kuormiin, joita ovat esimerkiksi sisällä olevien ihmisten määrä. [3]

Työn tavoitteena on selvittää adaptiivisten julkisivujen määritelmä ja ominaispiirteet sekä kromogeenisten lasituksien soveltuvuus Helsingin ilmastoon. Näihin tavoitteisiin päästään vastaamalla seuraaviin tutkimuskysymyksiin:

- Mitä tarkoittaa adaptiivinen julkisivu?
- Mitä erilaisia adaptiivisia julkisivuratkaisuja on olemassa?
- Miten kromogeeniset lasitukset toimivat?
- Kuinka suorituskykyisiä kromogeeniset lasitukset ovat?
- Soveltuvatko kromogeeniset lasitukset Helsingin ilmastoon?

Työ toteutetaan kirjallisuustutkimuksena ja sen aineiston etsimisessä on hyödynnetty muun muassa Web of Science- sekä Scopus-tietokantaa ja Google Patents-hakupalvelua. Työn pääpaino kohdistuu kromogeenisiin lasituksiin, sillä perinteisissä staattisissa julkisivuissa läpinäkyvät elementit ovat tyypillisesti suurin potentiaalinen kehityksen kohde energian säästämiselle [3]. Perinteisten lasituksien läpi siirtyy runsaasti energiaa, jota pyritään kompensoimaan sekä lämmityksellä että ilmastoinnilla. Tämä ei ole energiatehokasta. Ongelman voisi ratkaista pienemmillä ikkunoilla, mutta se ei ole viihtyvyyden kannalta järkevää. [7] Kromogeeniset lasitukset ovat suhteellisen helppoja asentaa jälkikäteen, jonka ansiosta myös vanhojen rakennusten energiatehokkuutta voidaan parantaa ilman suurempia rakenteellisia muutoksia. Kromogeenisten lasitusten markkina-arvo on myös eksponentiaalisesti kasvava: Kromogeenisten lasitusten maailman-markkinoiden arvioitiin olevan 4,03 miljardia dollaria vuonna 2019 ja niiden vuotuinen kasvuvauhti (*Compound Annual Growth Rate*) odotetaan olevan 14,7 % vuosina 2019–2027 saavuttaen 12,0 miljardia dollaria vuoteen 2027 mennessä [8].

Työ on rajattu yleisimpiin julkisivuissa käytettäviin teknologioihin, jotka kykenevät muuttamaan julkisivujen ominaisuuksia adaptiivisesti. Työssä ei huomioida laitteiden ohjausta, niiden hintakustannuksia eikä valmistusprosessia, vaan keskitytään esittämään julkisivuratkaisuja yleisellä tasolla paneutuen lopussa syvällisemmin kromogeenisiin lasituksiin ja niiden suorituskykyihin.

Luvussa 2 esitellään adaptiivisten julkisivujen määritelmä, ominaispiireet ja erilaisia adaptiivisia julkisivuratkaisuja. Luvussa 3 syvennytään tarkemmin kromogeenisten lasitusten rakenteisiin, toimintaan ja suorituskykyyn, jonka jälkeen luvussa 4 on yhteenveto ja luvussa 5 lähdeluettelo.

# 2 Adaptiiviset julkisivuratkaisut

Vuosina 2014–2018 toimineen Eurooppalaisen COST-toimen (*European Cooperation in Science & Technology Action*) TU1403:n, toiselta nimeltään *Adaptive Façades Network*, tavoitteena oli yhtenäistää adaptiivisiin julkisivuihin liittyvää tietoa, asiantuntemusta, resursseja sekä taitoa ja luoda siten uusia ideoita ja konsepteja perusteellisella ja tuotekehityksen tasolla [9]. TU1403:n määritelmän mukaan adaptiivinen julkisivu on rakennuksen vaippa, joka koostuu monipuolisista ja ”erittäin” mukautuvista järjestelmistä. Järjestelmät tavoittelevat rakennusten yleisen suorituskyvyn parantamista muuttamalla ajan myötä ominaisuuksiaan, toimintojaan tai käyttäytymistään vastatakseen muuttuviin reunaehtoihin ja suorituskykyvaatimuksiin. [10] Reunaehtojen muutokset voivat johtua lyhytaikaisista sään vaihteluista, vuorokausisykleistä sekä kausivaihteluista. Adaptiivisilla julkisivuilla voidaan parantaa käyttäjien hyvinvointia ja ilmanlaatua (*Indoor Environmental Quality*) sekä vähentää rakennuksien energiankulutusta ja niihin kohdistuvia ympäristöhaittoja. [5]

Adaptiivisilla julkisivuilla voi olla termisiä, solaarisia sekä visuaalisia ominaisuuksia, jotka muuttuvat ajan myötä joko passiivisesti tai aktiivisen ohjauksen myötä. [5] Adaptiivisten julkisivujen toiminta voidaan jakaa kolmeen osaan: ympäristön mittaukseen, mittauksesta saadun tiedon käsittelyyn ja tämän pohjalta suoritettuun fyysiseen toimintaan, kuten liikkeeseen tai materiaalin ominaisuuden muutokseen, vastatakseen sisäisten tai ulkoisten olosuhteiden muutoksiin [10].

Adaptiiviset julkisivut ovat merkittäviä yhden tai useamman seuraavien teknologisten ominaisuuksien ansiosta:

- aurinkoenergian absorbointi ja varastointi
- luonnollisen ilmanvaihdon ohjaaminen
- auringon säteilyn määrän säätely
- rakennuksen sisätilan viihtyvyyden parantaminen ja hallinta
- rakennuksen vaipan mukautuvien järjestelmien ylläpito [6].

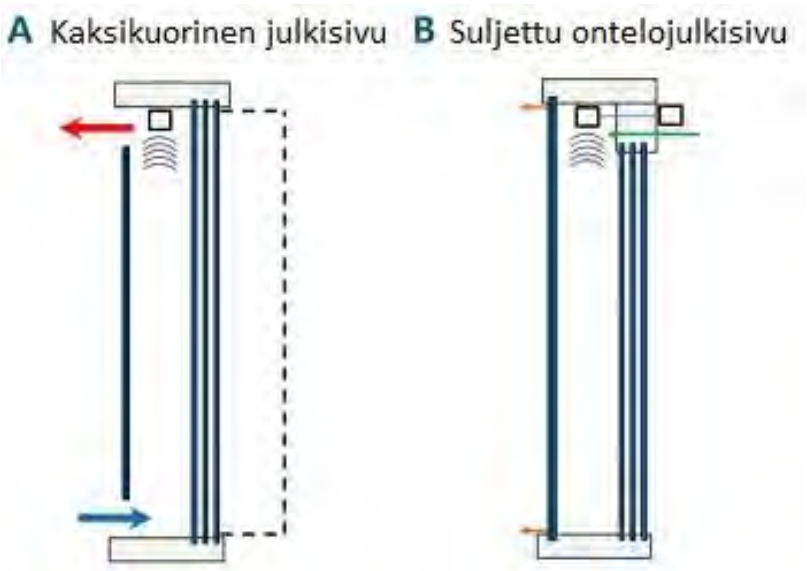
Attia ym. haastattelivat vuonna 2020 maailmanlaajuisesti 27:ää eri adaptiivisten julkisivujen asiantuntijaa, minkä tuloksena he päätyivät kategorisoimaan lupaavimmat adaptiiviset julkisivuratkaisut neljään eri luokkaan (kuva 1): aurinkoaktiivisiin julkisivuihin (*solar active façades*), dynaamisiin varjostimiin (*dynamic shadings*), aktiivisesti tuulettuviin julkisivuihin (*active ventilative façades*) sekä kromogeenisiin julkisivuihin (*chromogenic façades*) [5]. Esittelen seuraavaksi kolme ensimmäistä luokkaa hyvin yleisellä tasolla, jonka jälkeen seuraavassa luvussa syvennytään perusteellisemmin kromogeenisiin lasituksiin.


**Kuva 1:** Adaptiivisten julkisivujen teknologiat jaoteltuna neljään eri luokkaan (mukaillen [5]).

## 2.1 Aurinkoaktiiviset julkisivut

Aurinkoaktiiviset julkisivut hyödyntävät auringonenergiaa muun muassa lämmönsäätelyssä sekä energiantuotannossa. Aurinkoaktiivisiin julkisivuihin kuuluvat kaksikuoriset julkisivut (*double skin façades*), viherkatot ja -seinät (*green roofs and façades*), rakennuksiin integroidut aurinkokennot (*building integrated photovoltaics*) ja faasimuutosmateriaalit (*phase change materials*). [5]

Kaksikuorisilla julkisivuilla tarkoitetaan yhden tai useamman tason lasitettuja julkisivuja, jotka ovat erotettu toisistaan ilmaraoilla (kuva 2a). Tasojen välisissä onteloissa käytetään tyypillisesti dynaamisia varjostimia sekä luonnollista tai mekaanista tuuletusta. Kaksikuorisilla julkisivuilla mahdollistetaan täysin lasitettu julkisivu, joka parantaa rakennuksen ilmastointia, vähentää melusaasteita ja häikäisyä, lisää luonnollisen valon määrää sekä parantaa rakennuksen lämpöviihtyvyyttä ja energiatehokkuutta. [11]




**Kuva 2:** (a) Kaksikuorisen julkisivun ja (b) suljetun ontelojulkisivun rakenteet (mukaillen [12]).

Viherkatoilla ja -seinillä tarkoitetaan kasvualustoilla vuorattuja kattoja ja seiniä, joissa kasvaa viherkasvillisuutta. Viherkatot ja -seinät parantavat ilmanlaatua sitomalla kasvihuonekaasuja ja ilmansaasteita, estävät happosateiden syntyä, parantavat kaupunkiympäristöjen vedenlaatua, vähentävät melusaasteita, lisäävät kaupunkiympäristöjen biodiversiteettiä ja vähentävät kaupunkisaarekeilmiöitä. Ympäristölle positiivisten ominaisuuksien lisäksi viherkatot ja -seinät vähentävät rakennuksien energiankulutusta, parantavat lämmöneristystä, laskevat kattojen ja seinien lämpötiloja sekä suojelevat rakennuksia ulkoisilta haitoilta, kuten auringon säteilyltä ja happosateelta. [13]

Rakennuksiin integroidut aurinkokennot ovat aurinkokennoja, jotka integroidaan rakennuksien kattoihin tai julkisivuihin korvaamaan perinteisiä rakennus-materiaaleja, kuten tiiliä [14]. Aurinkokennoilla voidaan tuottaa sähköenergiaa muuttamalla auringon säteily tasavirtaiseksi sähköksi aurinkokennojen puolijohteissa ilmenevän valosähköisen ilmiön avulla [15]. Rakennuksiin integroiduissa aurinkokennojärjestelmissä ulkoilma kiertää järjestelmän läpi absorboiden aurinkokennojen lämpöä, mikä parantaa niiden suorituskykyä ja pidentää käyttöikää. Joissakin järjestelmissä lämmennyttä ilmaa hyödynnetään rakennuksien sisäilman lämmityksessä, mikä puolestaan vähentää lämmityskuluja talvisin. [14]

Faasimuutosmateriaalit ovat materiaaleja, jotka kykenevät varastoimaan faasimuutoksessa muodostuvan lämmön latenttilämpönä. Tiettyjen materiaalien kiinteän ja nesteen välisissä faasimuutoksissa (sulamisessa ja jähmettymisessä) voidaan varastoida runsas määrä lämpöenergiaa. Kun lämpöä siirretään

faasimuutosmateriaaleihin, niiden lämpötila pysyy samana koko faasimuutoksen ajan. Tätä lämpötilaa kutsutaan faasimuutoslämpötilaksi. Rakennuksissa käytettävien faasimuutosmateriaalien korkea lämpöhitaus (*thermo inertia*) ja hyvä lämpöeristys voivat vähentää rakennuksien energiankulutusta absorboimalla rakennuksien lämpösaantoa ja vähentämällä lämpövirtausta. Faasimuutos-materiaalit vähentävät rakennuksien lämpövirtausta absorboimalla osan lämmöstä päivisin sulaessa ja vapauttamalla varastoituneen lämmön öisin jähmettyessä. [16]

## 2.2 Dynaamiset varjostimet

Dynaamiset varjostimet ovat liikkuvista osista koostuva luokka, johon sisältyy muun muassa ikkunaluukut, rullaverhot, sälekaihtimet sekä sälekaihtimilla varustetut luonnollisesti tuulettuvat suljetut ontelojulkisivut (*natural ventilated closed cavity façades*). Dynaamiset varjostimet säätelevät sisälle pääsevää auringon säteilyn määrää ja ne voivat olla motorisoituja tai manuaalisesti ohjautuvia. Dynaamiset varjostimet voivat tarjota auringon säteilyn säätelyn lisäksi lämmöneristystä, kesäaikaista lämpöviihtyvyyttä, fyysistä suojaa, yksityisyyttä, häikäisysuojaa sekä viilennyskustannuksien pienentämistä. [5] Luonnollisesti tuulettuvat suljetut ontelojulkisivut eroavat luonnollisesti tuulettuvista kaksikuorisista julkisivuista siten, että niiden ontelot ovat suljettuja (kuva 2b). Onteloiden ja sisäilman välillä on kuiva ilmankierto, mikä estää nesteen kondensoitumisen onteloiden sisällä. Suljetun rakenteensa ansiosta suljetut ontelojulkisivut tarvitsevat kaksikuorisiin julkisivuihin verrattuna selvästi vähemmän pesemistä ja ylläpitoa. [17]

## 2.3 Aktiivisesti tuulettuvat julkisivut

Aktiivisesti tuulettuviin julkisivuihin kuuluvat aktiivisesti tuulettuvat suljetut ontelojulkisivut (*active ventilated closed cavity façades*) sekä automaattisesti operoitavat ikkunat (*automated operable windows*). Aktiivisesti tuulettuvat julkisivut keskittyvät ilmanvaihtoon; aktiivisesti tuulettuvat suljetut ontelojulkisivut keskittyvät säätelemään ontelon sisäistä ilmavirtaa ja automaattisesti operoitavat ikkunat ohjaavat rakennukseen pääsevää ilmaa. [5] Aktiivista tuuletusta hyödynnetään myös muissa adaptiivisten julkisivujen teknologioissa, kuten aiemmin mainituissa kaksikuorisissa julkisivuissa sekä rakennuksiin integroiduissa aurinkokennoissa.

# 3 Kromogeeniset julkisivut

Kromogeenisillä julkisivuilla (*chromogenic façades*) tarkoitetaan lasituksiin integroituneita kromisia materiaaleja (*chromic materials*) hyödyntäviä teknologioita, joista yleisimpiä ovat elektrokrominen lasitus (*electrochromic glazing*), nestekidelasitus (*liquid crystal glazing*), suspensiohiukkaslasitus (*suspended particle glazing*) ja termokrominen lasitus (*thermochromic glazing*). Kromiset materiaalit ovat materiaaleja, joiden termo-optisia ominaisuuksia voidaan muokata ulkoisen ärsykkeen avulla. Kromogeenisiä lasituksia (*chromogenic glazing*) kutsutaan myös muun muassa adaptiivisiksi lasituksiksi (*adaptive glazing*), älykkäiksi lasituksiksi (*smart glazing*) sekä dynaamisiksi lasituksiksi (*dynamic glazing*).

Kromogeeniset lasitukset pyrkivät minimoimaan rakennuksien kokonais-energiankulutusta moduloimalla dynaamisesti termo-optisia ominaisuuksiaan tummenemalla tai kirkastumalla, jotta ne voivat mukautua ulkoisiin ympäristön muutoksiin ja sisäisiin kuormiin, kuten ihmisten määrään, valaistukseen ja laitteiden käyttöön. Kromogeeniset lasitukset voidaan jakaa passiivisiin ja aktiivisiin teknologioihin: passiiviset teknologiat aktivoituvat itselaukaisevan (*self-trigger*) adaptiivisen mekanismin seurauksena ja aktiiviset aktivoituvat ulkoisen ärsykkeen, kuten sähkökentän, seurauksena. [3] Aiemmin mainituista teknologioista termokrominen lasitus on passiivinen ja aktiivisia ovat elektrokrominen lasitus, nestekidelasitus sekä suspensiohiukkaslasitus.

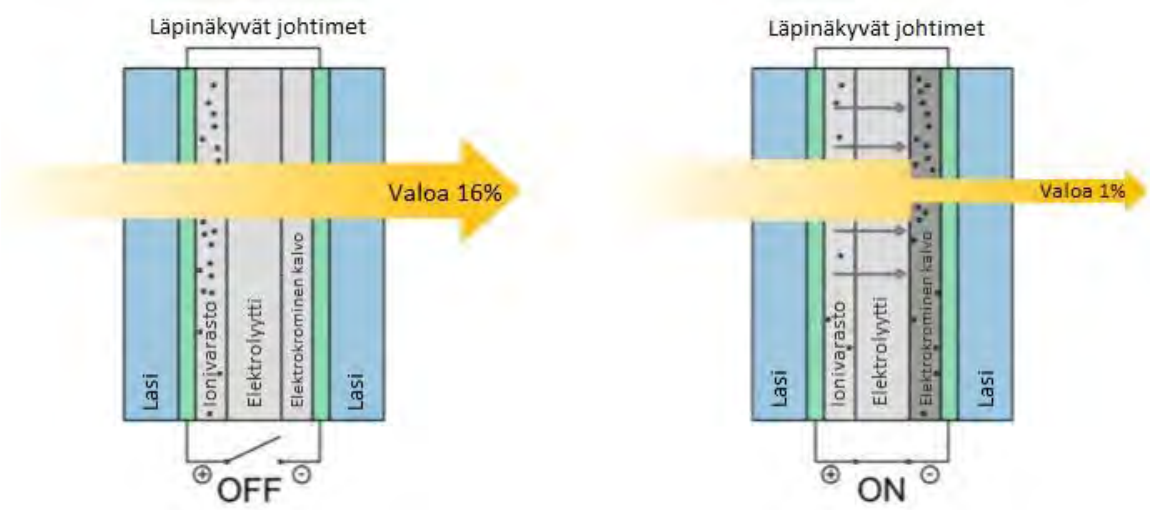
Luvuissa 3.1–3.4 käsitellään kromogeenisten teknologioiden toimintaperiaatteita, rakenteita sekä materiaaleja ja luvussa 3.5 vertaillaan markkinoilla olevien kromogeenisten lasitusten suorituskykyjä ja soveltuvuutta Helsingin ilmastoon.

## 3.1 Elektrokrominen lasitus

Elektrokromisen lasituksen toiminta perustuu elektrokromisiin materiaaleihin, jotka muuttuvat jännitteen avulla palautuvasti läpinäkyvästä tummaksi muuttaen lasituksen näkyvän valon ja lähi-infrapunan optista läpäisevyyttä [18]-[22]. Lasituksen tummuutta muuttamalla voidaan siis säätää rakennuksen sisälle pääsevän näkyvän valon ja lämpöenergian määrää, mikä mahdollistaa miellyttävän luonnollisen valaistuksen ja huoneenlämmön. Tämä säästää sähköä rakennuksen viilennyksessä sekä keinotekoisen valaistuksen käytössä.

Elektrokrominen laite koostuu yleisesti keskellä olevista elektrokromisesta kalvosta ja ioneja varastoivasta kalvosta, niiden välillä olevasta elektrolyytistä sekä niitä ympäröivistä läpinäkyvistä johtimista (kuva 3) [18], [19], [22]. Laite on kiinnittyneenä joko yhdelle pinnalle tai laminoituna kahden pinnan väliin [18], [22]. Pintana käytetään luonnollisesti lasia, mutta muovisia pintoja voidaan myös käyttää,

kuten polyetyleenitereftalaattia (PET) tai polykarbonaattia (PC) [18], [21].




**Kuva** 3**:** Elektrokromisen laitteen yleinen rakenne ja toimintaperiaate (mukaillen [23]).

Elektrokromiset kalvot ovat sekoitettuja johtimia, eli ne läpäisevät sekä elektroneja että ioneja [18], [21], [22], ja niiden materiaalina käytetään yleisesti epäorgaanisia metallioksideja, kuten eniten tutkittua volframitrioksidia $WO_3$. Elektrokromisissa kalvoissa voidaan käyttää myös muita metallioksideja, kuten volframiin (W), molybdeeniin (Mo), titaaniin (Ti), niobiumiin (Nb), iridiumiin (Ir) tai nikkeliin (Ni) perustuvia oksideja. [18], [21], [22] Orgaanisia materiaaleja on myös mahdollista käyttää, mutta niiden ongelmana on huono kestävyys [19], [22], jonka takia niitä ei usein käytetä rakennuksissa. Ioneja varastoiva kalvo on myös sekoitettu johdin ja se voi toimia joko pelkästään ionivarastona tai sillä voi olla myös elektrokromisia ominaisuuksia. Kalvojen välille saadaan paras yhteensopivuus, jos ioneja varastoivalla kalvolla on vastakkaisia elektrokromisia ominaisuuksia elektrokromisen kalvon kanssa. [22]

Elektrokromisia materiaaleja on kahdenlaisia; katodisia ja anodisia. Katodiset materiaalit tummuvat ioneja vastaanottaessa ja anodiset puolestaan luovuttaessaan ioneja. [18], [20], [22] Metallioksideista anodisia materiaaleja ovat esimerkiksi volframioksidit, molybdeenioksidit, titaanioksidit, niobiumoksidit ja katodisia materiaaleja ovat iridiumoksidit sekä nikkelioksidit. Vanadiinioksidit omaavat sekä anodisia että katodisia ominaisuuksia. Elektrokromisessa kalvossa käytetään anodisia materiaaleja ja ioneja varastoivassa kalvossa katodisia, jolloin molemmat tummenevat sekä vaalenevat samanaikaisesti sähkökentän läsnä ollessa. Kahden elektrokromisen kalvon käyttäminen on yleistä ja kannattavaa, sillä se mahdollistaa monipuolisia väritysintensiteettejä ja voi antaa tummennukselle luonnollisen värin [19], [21].

Kalvojen välillä oleva elektrolyytti on puhdas ionijohdin, eli se päästää lävitseen ioneja, muttei elektroneja [18], [21], [22]. Jos elektroneja pääsisi kalvon läpi, laitteessa syntyisi oikosulku, jolloin elektrokromisen kalvon tummennusta ei tapahtuisi tai tummennus vaalenisi ajan myötä, mikä johtuu ionien ”vuotamisesta” takaisin ioneja varastoivalle kalvolle. Elektrolyytti voi olla kiinteää, nestemäistä tai geeliä. Se koostuu yleensä yhdestä tai useammasta nestemäisestä liuottimesta sekä yhdestä tai useammasta suolasta. Elektrokromisissa laitteissa voidaan käyttää nestemäisiä, keraamisia, polymeerisiä ja kiinteitä epäorgaanisia elektrolyyttejä, joista polymeeriset elektrolyytit ovat räätälöity erityisesti elektrokemiallisille laitteille, kuten litiumakuille sekä elektrokromisille laitteille. Elektrokromisissa laitteissa yleisimpiä polymeerisiä elektrolyyttejä ovat boronaattiesterit, natrium-polystyreenisulfonaatti, polyakryylinitriili, polyesterit, poly(2-etoksietyyli-metakrylaatti), polyetyleeniglykoli, polyetyleenioksidi, polyetyylimetakrylaatti, polymetyylimetakrylaatti, polyvinyylialkoholi, polyvinyylideenifluoridi, poly-vinyylikloridi, polyvinyylisulfonit ja vesiohenteinen polyuretaani. [19] Elektrolyyttiin voidaan lisätä läpinäkyvissä johtimissa käytettyjä nanopartikkeleita, kuten $SnO_2$, joka estää lähi-infrapunan läpäisevyyttä, mutta sallii näkyvän valon läpäisyn elektrolyytin läpi, mikä on hyödyllistä varsinkin kuumassa ilmastossa [21].

Läpinäkyvien johtimien tehtävä on tasoittaa kalvojen välillä tapahtuvien ionivirtausten synnyttämiä sähkövarauksia [22]. Ne ovat puhtaita elektronijohtimia, joille on ominaista erinomainen sähkönjohtavuus sekä optinen läpinäkyvyys [21]. Pienissä laitteissa sähköä voidaan kohdistaa suoraan johtimiin, mutta isommissa laitteissa, kuten ikkunoissa, tarvitaan virtakiskoa, eli metallista kehystä, joka jakaa virran tasaisesti johtimiin mahdollistaen tasaisen ja nopean tummennuksen sekä vaalennuksen [18], [21], [22]. Johtimissa voidaan käyttää puolijohdepohjaisia, metallipohjaisia, hiilipohjaisia ja nanolankaan pohjautuvia kalvoja tai niiden sekoituksia. Puolijohdepohjaiset kalvot ovat raskaasti seostettuja (muutamien atomiprosenttien luokkaa) ja leveän energiaraon omaavia oksidikalvoja, joita käytetään yleisimmin elektrokromisissa laitteissa. Seostaminen (*doping*) tarkoittaa tässä yhteydessä epäpuhtauksien (metallien) lisäämistä oksidikalvoon, mikä parantaa kalvon sähkönjohtavuutta. Näitä ovat esimerkiksi:

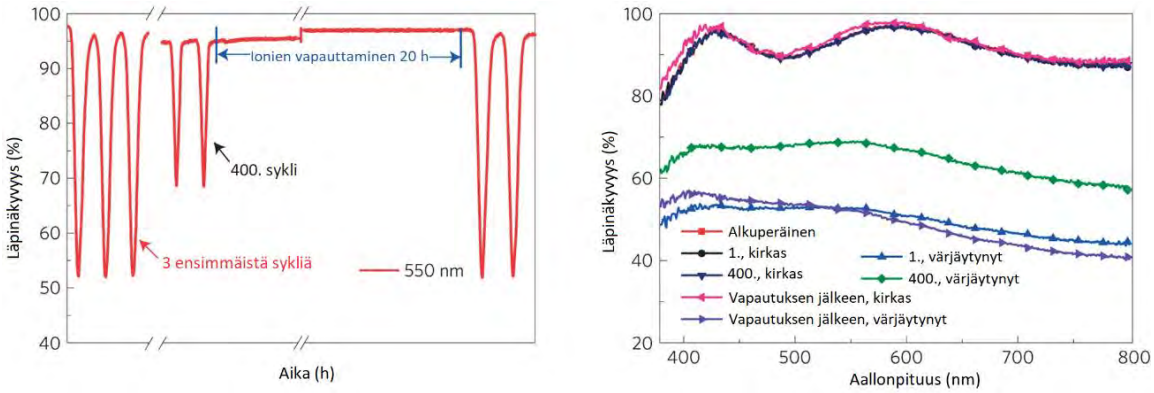
- Indiumoksidi, jota on seostettu tinalla ($In_2O_3$:Sn), sinkillä ($In_2O_3$:Zn) tai niobiumilla ($In_2O_3$:Nb).
- Tinaoksidi, jota on seostettu fluorilla ($SnO_2$:F), antimonilla ($SnO_2$:Sb), tantaalilla ($SnO_2$:Ta) tai volframilla ($SnO_2$:W).
- Sinkkioksidi, jota on seostettu alumiinilla (ZnO:Al), galliumilla (ZnO:Ga), indiumilla (ZnO:In), tinalla (ZnO:Si), boorilla (ZnO:B), fluorilla (ZnO:F), niobiumilla (ZnO:Nb) tai skandiumilla (ZnO:Sc). [18], [21]

Elektrokromista lasitusta voidaan ajatella akkuna, jonka varaustila ilmenee sen optisissa ominaisuuksissa [19], [22]. Kun läpinäkyviin johtimiin tuodaan tasavirtaista sähköä, syntyy johtimien välille sähkökenttä. Tämän seurauksena

ioneja varastoivasta kalvosta siirtyy ioneja elektrolyytin läpi (pieniä ioneja, yleensä vety- ($H^+$) tai litiumioneja ($Li^+$) [18], [21], [22]) elektrokromiseen kalvoon, jolloin kalvon optiset ominaisuudet muuttuvat tummentaen lasituksen ja siten estäen luonnonvalon sekä lämpöenergian siirtymisen lasituksen läpi (kuva 3). [18], [20]-[22] Tummeneminen tapahtuu yleensä sekunneissa, mutta isommissa laitteissa, kuten ikkunoissa, tummenemisessa voi kestää useampi minuutti [18], [19]. Laite voidaan palauttaa takaisin läpinäkyväksi kääntämällä virran suuntaa, tai joissakin tapauksissa oikosululla, jolloin ionit palaavat takaisin ioneja varastoivaan kalvoon [22].

Elektrokrominen laite kykenee säilyttämään ”muistinsa” pitkään avoimessa piirissä. Toisin sanoen laitteen optiset ominaisuudet säilyvät, vaikka virta ei enää kulje laitteessa. Tämän ansiosta laite tarvitsee sähköä ainoastaan, kun sen optisia ominaisuuksia halutaan muuttaa ja tummennus voidaan keskeyttää missä tahansa vaiheessa ääripäiden välillä mahdollistaen kalvolle lineaarisia tummuusasteita. Elektrolyytti eivät kuitenkaan ole täydellisiä ja ne vuotavat siksi hieman ioneja, minkä takia tilan ylläpitoon kuluu hieman sähköä. Vähäisen sähkön käytön takia (2,5 $W/m^2$ tilan muutokseen ja 0,4 $W/m^2$ tilan ylläpitoon [23]) elektrokromiset laitteet ovat energiatehokkaita. [18], [21], [22]

Elektrokromiset lasituksen kuluvat samalla tavalla kuin akut; niiden varaustaso huononee hiljalleen lataus-purkaus-syklien seurauksena, jolloin niiden tummuus-intensiteetti kärsii. Volframioksidiin ($WO_3$) perustuvan elektrokromisen lasituksen tummuusintensiteetti kärsii elektrokromiseen kalvoon loukkuun jääneiden litiumionien ($Li^+$) seurauksena. Tätä ongelmaa on pyritty korjaamaan galvanostaattisella hoidolla, jolloin lasituksen suorituskyky on saatu palautumaan ennalleen (kuva 4). Vasemmanpuoleisessa kuvaajassa on kuvattu kalvon läpinäkyvyyttä 550 nm aallonpituudella syklien aikana, jonka 400. syklin suorituskyky on huonontunut huomattavasti alkuperäiseen verrattuna. Huonontunut suorituskyky saatiin palautettua ionien vapauttamisella. Kuvan oikeanpuoleisessa kuvaajassa on esitelty sama tilanne koko näkyvän valon spektrillä, jossa on huomioitu ainoastaan alkuperäinen läpinäkyvyys, 1. sykli, 400. sykli sekä ionien vapauttamisen jälkeinen sykli. [24]

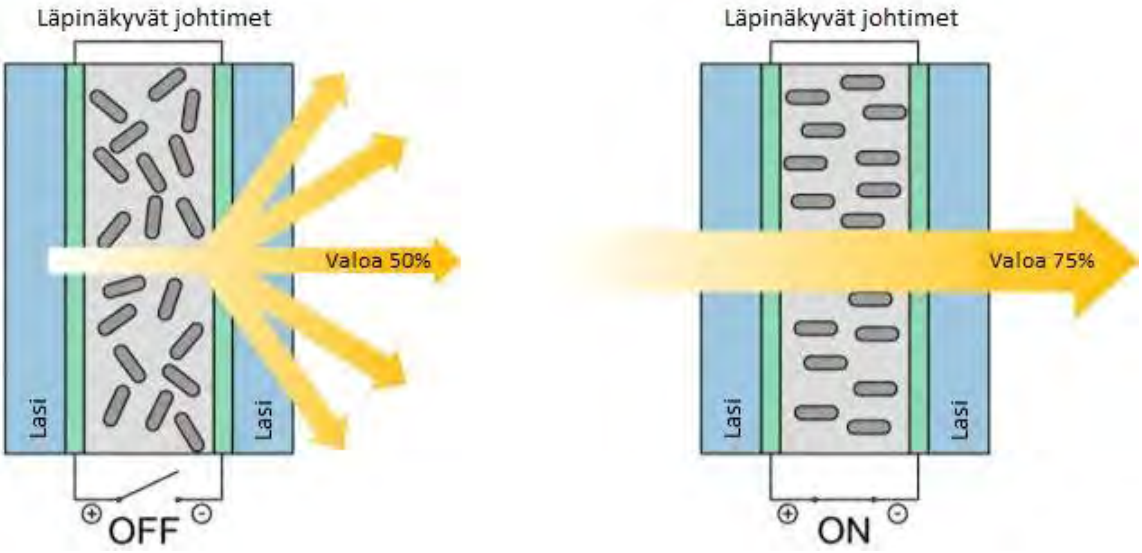



**Kuva 4:** Vasemmassa kuvaajassa näkyy kalvon läpinäkyvyys 550 nm aallonpituudella syklien aikana. Oikealla kuvataan samaa tilannetta koko näkyvän valon spektrillä (mukaillen [24]).

## 3.2 Nestekidelasitus

Nestekiteitä käytetään tunnetusti LCD-näytöissä, mutta niiden potentiaali on todettu myös adaptiivisissa lasituksissa. Nestekidelasitus perustuu nestekiteiden ja polymeerimatriisien muodostamiin komposiittimateriaaleihin, joissa valo hajautuu riippuen nestekidemolekyylien asennosta suhteessa polymeerimatriisiin [25], [26]. Lasitus toimii vaihtovirralla ja se saadaan läpinäkyväksi suljetussa piirissä (ON-tila) ja tummaksi avoimessa piirissä (OFF-tila) [25], [26]. Tyypillisiä lasituksia ovat PDLC (*polymer-dispersed liquid crystal*) ja PSLC (*polymer-stabilized liquid crystal*) [27].

Nestekidelasitus muodostuu nestekide-polymeeri-komposiitikalvosta ja sitä ympäröivistä läpinäkyvistä johtimista, joiden materiaalina käytetään yleensä tinalla seostettua indiumoksidia ($In_2O_3$:Sn) (kuva 5) [25], [26], [28]-[30]. Indiumoksidille vaihtoehtona on tutkittu hopeananolankaa, joka on halvempaa ja mahdollistaa 15 V pienemmän käyttöjännitteen sekä suuremman kontrastin [31]. Kalvo ja johtimet ovat luonnollisesti laminoituneena kahden lasipinnan väliin [25]-[28], [30]. PDLC-kalvossa nestekiteet ovat jakautuneet tasaisesti huokoiseen polymeerimatriisiin muutaman mikronin (1 mikroni = 1000 nm) kokoisiksi pisaroiksi [25], [26]. PSLC-kalvossa polymeerin osuus suhteessa nestekiteisiin on huomattavasti pienempi kuin PDLC-kalvossa (5 % vs. 30 %), ja ne muodostavat matriisin sijaan säikeisen polymeeriverkoston [26], [32].




**Kuva 5:** Nestekidelasituksen toimintaperiaate (mukaillen [23]).

PDLC-kalvon polymeerimatriisi voidaan korvata lasilla, jolloin puhutaan GDLC-kalvosta (*glass-dispersed liquid crystal*). Lasi eliminoi auringon UV-säteilystä aiheutuvan polymeerin keltaisuuden ja se on polymeeriä kirkkaampaa läpinäkyvässä tilassa. GDLC-kalvo on myös kestävämpi, mahdollistaa 10 kertaa lyhyemmän vasteen ja tarvitsee 85 % pienemmän käyttöjännitteen verrattuna samankaltaiseen PDLC-kalvoon. Ongelmia kuitenkin esiintyy sen valmistuksessa, jossa xerogeelin muodostamisessa kalvo saattaa halkeilla kapillaari-ilmiön takia. Tämä on pyritty estämään korvaamalla haihtuva liuos nestekiteillä. [28]

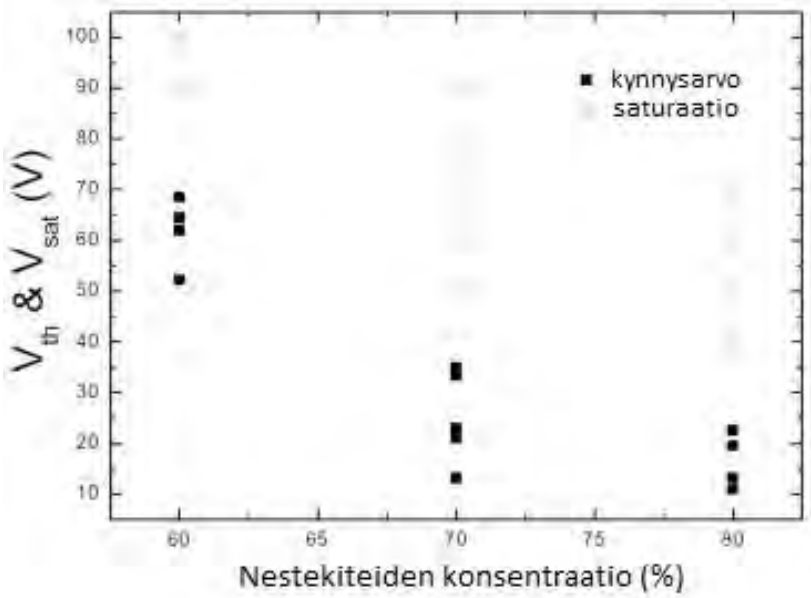
Nestekiteiden materiaalien selvittämisessä on ollut ongelmia, sillä vastaan on tullut pääasiassa tuotenimiä, kuten NOA65 [26], HNG30400 [33] ja SLC1717 [27], joiden kemiallisista rakenteista ei ole löytänyt tietoa. Vastaan on tullut myös muutama kemiallisen yhdisteen nimi, kuten 4-pentyylifenyylipropyylibentsoaatti ja 4-n-pentyylibifenyyli [28]. PDLC-kalvoissa käytettävät nestekidemolekyylit ovat yleensä nemaattisia molekyylejä [26]-[28], [31]-[33], joilla on ympäröivää polymeeriä pienempi moolimassa [25].

Nestekidelasituksessa käytetään yleensä positiivisesti dielektrisiä nestekiteitä, jotka kokevat dipolimomentin molekyyliakselin mukaisesti. Tällöin sähkökentän läsnä ollessa nestekidemolekyylit uudelleenorientoituvat sähkökentän mukaisesti. Negatiivisesti dielektrisissä nestekiteissä dipolimomentti ilmenee puolestaan kohtisuorassa suhteessa molekyyliakseliin, jolloin nestekidemolekyylit uudelleenorientoituvat kohtisuorasti suhteessa sähkökenttään. [34] Negatiivisesti dielektrisiä nestekiteitä käytetään käänteisissä nestekidelasituksissa, jotka ovat läpinäkyviä OFF-tilassa ja tummia ON-tilassa [32], [33]. Käänteiset nestekidelasitukset ovat normaalisti suo-

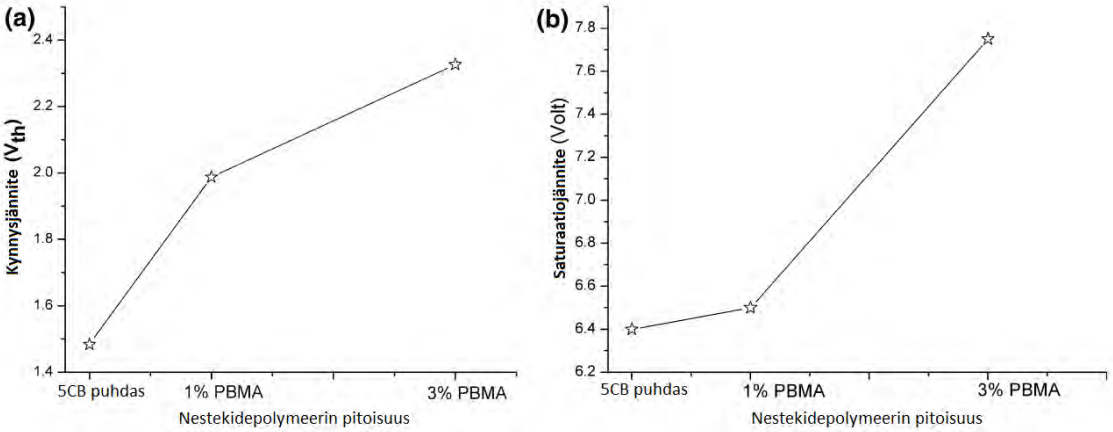
tavampia, sillä lasitusta pidetään ennemmin läpinäkyvänä kuin tummana, jolloin läpinäkyvyyden ylläpitämiseen ei kulu sähköä ja lasitus pysyy läpinäkyvänä, vaikka tulisi sähkökatkos [34].

Nestekiteet ovat kahtaistaitteisia eli ne omaavat kaksi eri taitekerrointa: tavallisen (*ordinary refractive index*) $n_o$ ja epätavallisen (*extraordinary refractive index*) $n_e$ [26]. Nestekidelasituksessa valon hajautuminen johtuu nestekidepisaroiden ja niitä ympäröivän matriisin välisistä valon taitekertoimista ja niiden eroavaisuudesta. Ilman ulkoista sähkökenttää, nestekidepisaroiden molekyylit ovat epäjärjestyksessä, johtuen nestekidemolekyylien ankkuroitumisesta polymeerimatriisiin, jolloin ne hajaannuttavat tulevaa valoa eri suuntiin. Tällöin lasitus näyttää tummalta tai maitomaisen samealta. Kun kalvoon kohdistetaan vaihtovirrasta syntyvä ulkoinen sähkökenttä, nestekidemolekyylit uudelleenorientoituvat homeotrooppisesti sähkökentän mukaisesti, jolloin niiden tavallinen taitekerroin $n_o$ vastaa matriisin taitekerrointa $n_m$. Tällöin valo pääsee lasituksen läpi tehden siitä läpinäkyvän (kuva 5). [25], [26], [28], [31] Nestekidepisaroiden koko määrittelee niiden kyvyn hajaannuttaa tulevaa valoa. Jos pisaroiden säde on tulevan valon aallonpituutta pienempi, valo pääsee kulkeutumaan läpi ilman minkäänlaista hajaantumista [25]. Jotta lasitus olisi tehokas, kalvon tummuuden ja läpinäkyvyyden välillä täytyy olla mahdollisimman suuri kontrasti. Paras kontrasti saadaan, kun nestekiteiden kaksoistaitteisuus (*birefringence*) on suuri ($\Delta n = n_e - n_o$) ja nestekidemolekyylin tavallinen taitekerroin on lähellä matriisin taitekerrointa ($n_o \approx n_m$). [26]

Lasituksen huonona puolena on suuri sähkönkulutus (5,00–10,00 W/m$^2$ tilan muutokseen sekä tilan ylläpitämiseen [23]), mikä johtuu korkeasta käyttöjännitteestä ja läpinäkyvyyden ylläpitämisestä [35]. PDLC-kalvon käyttöjännite on korkeampi kuin PSLC-kalvon johtuen sen suuremmasta polymeeripitoisuudesta (kuvat 6 ja 7). Kuvissa kynnysjännitteellä tarkoitetaan jännitettä, joka tarvitaan muuttamaan kalvon läpinäkyvyyttä 10 %. Saturaatiojännitteellä tarkoitetaan puolestaan jännitettä, joka tarvitaan ylläpitämään läpinäkyvyys 90 %:ssa (18). Epäilin kuvan 5 saturaatio- ja kynnysarvojen olevan artikkelissa merkitty väärinpäin, joten otin yhteyttä yhteen artikkelin tekijöistä, professori Yongho Seoon, varmistaakseni asian. Epäilykseni osoittautui todeksi, joten korjasin kyseisen virheen suomentaessani kuvaa. PDLC-kalvossa polymeerimatriisi ankkuroi voimakkaammin nestekidemolekyylejä, jolloin ne tarvitsevat suuremman jännitteen uudelleenorientoitumiseen. PSLC-kalvoissa ankkurointi on heikompaa, jolloin myös tarvittava jännite on pienempi. [33]




**Kuva 6:** PDLC-kalvon kynnysjännite ($V_{th}$) ja saturaatiojännite ($V_{sat}$) suhteessa nestekiteiden määrään (mukaillen [26]).




**Kuva 7:** PSLC-kalvon a) kynnysjännite ja b) saturaatiojännite suhteessa polymeerin määrään (mukaillen [30]).

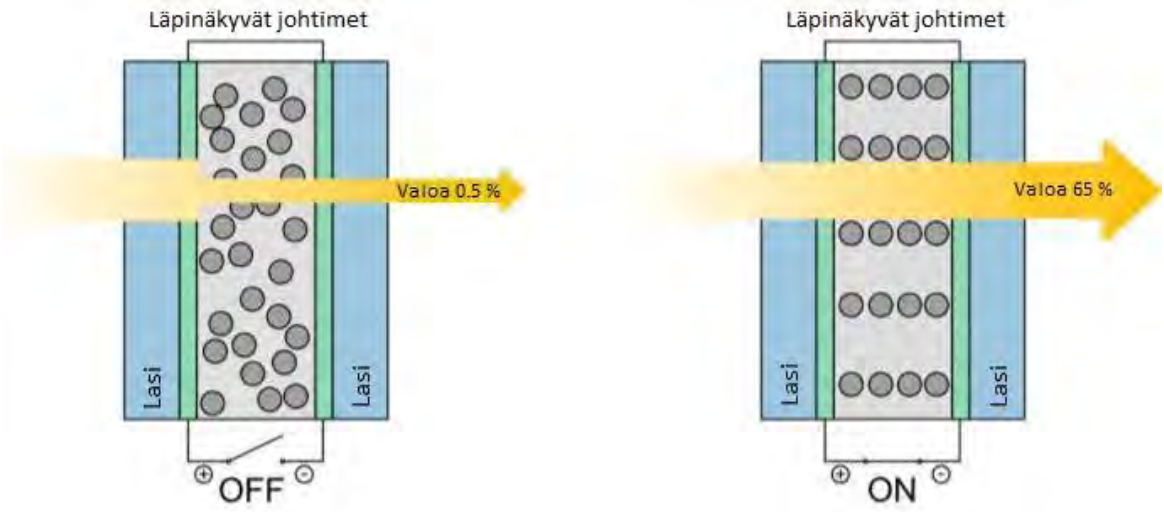
Toisaalta PSLC-kalvon valon hajaantuminen on heikompaa kuin PDLC-kalvossa, jolloin sen kontrasti kärsii. PSLC-kalvon vähäinen polymeeripitoisuus tekee kalvosta myös mekaanisesti heikon ja se ei siksi sovellu yhtä hyvin lasituksiin. Tähän on kehitelty materiaali, jossa yhdistetään PDLC- ja PSLC-kalvojen ominaisuudet muodostamalla PD&SChLC-kalvo. Muodostettu kalvo tarvitsee 50 % vähemmän jännitettä verrattuna PDLC-kalvoon ja se omaa hyvän mekaanisen kestävyyden. [27]

## 3.3 Suspensiohiukkaslasitus

Suspensiohiukkaslasitus perustuu dipolarisiin hiukkasiin, jotka leijuvat vapaasti nesteessä tai geelissä ja aiheuttavat lasituksen tummenemisen heijastamalla tai absorboimalla tulevaa valoa. Korkeajännitteisen vaihtovirran läsnä ollessa, hiukkaset orientoituvat sähkökentän mukaisesti, jolloin valo pääsee kalvon läpi tehden lasituksesta läpinäkyvän. Kuten nestekidelasitus, suspensiohiukkaslasitus on tumma avoimessa piirissä (OFF-tilassa) ja läpinäkyvä suljetussa piirissä (ON-tilassa). Suspensiohiukkaslasituksen ideana ei kuitenkaan ole valon hajauttaminen, vaan sen estäminen. [36], [37]

Suspensiohiukkaslasitus muodostuu yleisesti optisesti aktiivisesta kalvosta ja sitä ympäröivistä läpinäkyvistä johtimista, joiden materiaalina käytetään tinalla seostettua indiumoksidia ($In_2O_3$:Sn). Nämä ovat laminoituneena kahden substraatin väliin, joiden materiaaleina käytetään yleensä joko lasia tai muovia, kuten polyetyleenitereftalaattia (PET) (kuva 8). [36]-[39]

Optisesti aktiivinen kalvo koostuu yleisesti polymeerimatriisista, joka ympäröi hiukkasia sisältäviä heterogeenisiä suspensionestepisaroita [36]-[40]. Jos kalvo täytettäisiin kokonaan suspensionestellä polymeerimatriisin sijaan, nesteestä johtuva hydrostaattinen paine voisi saada aikaan kalvoon pullistumia ja aiheuttaa siten vuotoriskin [37], [40]-[42]. Polymeerimatriisissa suspensioneste on jakautunut kalvolle tasaisesti pieniksi pisaroiksi (patentissa ”US 6,416,827 B1” sanotaan pisaroiden olevan halkaisijaltaan alle 2 mikronia [42]), mikä eliminoi sekä hydrostaattisen paineen että vuotoriskin, mutta myös kalvon toistuvan aktivoitumisen seurauksena aiheutuvan hiukkasten agglomeroitumisen eli toisiinsa takertumisen [37], [40]-[42].




**Kuva 8:** Suspensiohiukkaslasituksen yleinen rakenne ja toimintaperiaate (mukaillen [23]).

Hiukkaset (*particles*) ovat muodoltaan sauvamaisia tai pyöreitä ja ne ovat yleisesti kolloidien kokoisia, eli niiden suurin ulottuvuus on mikronin (1000 nm) tai vähemmän [37], [39]-[41]. Niiden koko olisi kuitenkin suotavaa olla alle puolet sinisen valon aallonpituudesta, eli alle 200 nm, pitääkseen valon hajautumisen mahdollisimman alhaisena [36], [37], [39], [40]. Hiukkaset voivat koostua orgaanisista tai epäorgaanisista materiaaleista ja ne voivat absorboida tai heijastaa osittain tai kokonaan näkyvän valon spektriä [37], [40]-[42]. Hiukkasten materiaalina käytetään yleisesti polyhalideja, tarkemmin ottaen polyjodideja, joista tunnetuin on herafatiitti (*heraphatite*) [36].

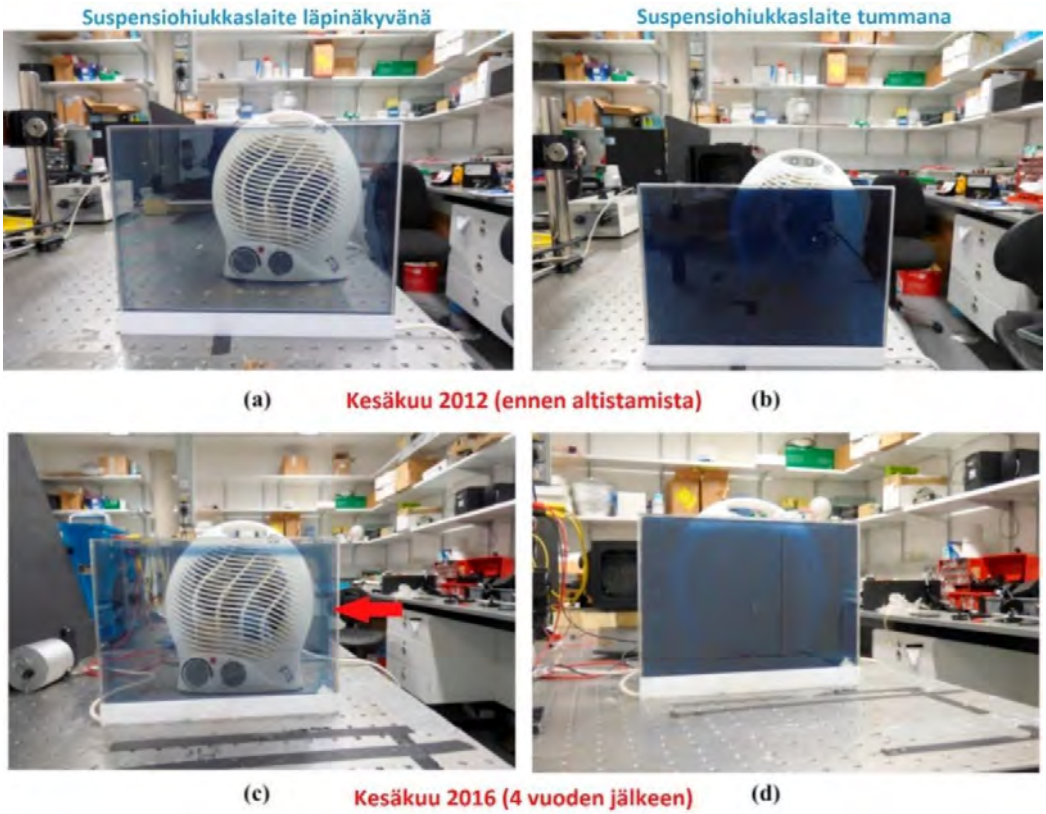
Suspensioneste (*liquid suspending media*) on optisesti aktiivisen kalvon väliaine, jossa liukenemattomat hiukkaset leijuvat. Suspensioneste koostuu yhdestä tai useammasta vedettömästä, sähköisesti resistiivisestä nesteestä ja siinä olisi suotavaa olla vähintään yksi polymeerinen stabilointiaine (*polymeric stabilizer*), joka estää hiukkasten agglomeroitumisen ja pitää hiukkaset suspensiossa eli samassa faasissa nesteen kanssa. Näiden väliaineiden avulla hiukkaset saadaan painovoiman tasapainotilaan (*gravitational equilibrium*). [37], [40]-[42] Väliaineen taittokertoimen täytyy vastata mahdollisimman tarkasti ympäröivän polymeerimatriisin taittokertoimen kanssa välttääkseen valon hajautumiselta [42].

Suspensionesteessä käytetään yleisesti moolimassaltaan suuria yhdisteitä tai suurten ja keskisuurten sekoituksia. Eri moolimassaisia yhdisteitä sekoittamalla voidaan muodostaa sopiva viskositeetti suspensionesteelle, mikä parantaa kalvon valmistuksessa olevan emulsion stabiilisuutta, joka puolestaan parantaa kalvon ominaisuuksia ja suorituskykyä. [37], [40]-[42] Mitä suurempi viskositeetti suspensionesteellä on, sitä hitaampi on kalvon vasteaika [38]. Suspensionesteinä käytetään muun muassa tri-isodekyylitrimellitaattia (*triisodecyl trimellitate*), dimetyylifluorisuberaattia (*dimethyl perfluorosuberate*) ja trietyylitrimellitaattia (*triethyl trimellitate*) [41], [42].

Kuten aiemmin mainittu, polymeerinen stabilointiaine auttaa pitämään hiukkaset erillään. Polymeerisenä stabilointiaineena voidaan käyttää yhdisteitä, jotka liukenevat suspensionesteeseen ja muodostavat sidoksia hiukkasten kanssa. Tällainen yhdiste on esimerkiksi kiinteä nitroselluloosa, joka suspensionesteeseen liuenneena antaa hiukkasille tasaisen pinnoitteen. Polymeerisiä stabilointiaineita voi olla yksi tai useampi. Useampi stabilointiaine muodostaa polymeerisen stabilointijärjestelmän, jossa ensimmäinen stabilointiaine muodostaa sidoksia hiukkasten kanssa ja yksi tai useampi stabilointiaine muodostaa sidoksia ensimmäisen stabilointiaineen kanssa hajauttaen hiukkasia erilleen ja antaen hiukkasille steeristä suojausta (*steric protection*). [37], [40]-[42]

Hiukkaset ajelehtivat satunnaisesti ympäriinsä suspensionesteessä jännitteen poissa ollessa (OFF-tilassa) Brownin liikkeen seurauksena. Tällöin hiukkaset absorboivat tai heijastavat saapuvaa valoa, jolloin lasitus näyttää tummalta. Syötettäessä johtimiin vaihtovirtaista jännitettä (ON-tilassa) saadaan suspensiohiukkaskalvolle aikaan sähkökenttä. Sähkökentän vaikutus saa hiukkaset orientoitumaan ja järjestäytymään sähkökentän mukaisesti päästäen valoa kalvon läpi, jolloin lasitus näyttää läpinäkyvältä. [36]-[40], [43] Jännitteen suuruutta muuttamalla voidaan säätää lasituksen tummuusintensiteettiä. Lasituksen jännite on oltava vaihtovirrassa, sillä tasavirta kuluttaisi kalvoa ajan myötä. [36]

Suspensiohiukkaslasitus tarvitsee tilanmuutokseen 5,00 W/m$^2$ ja tilan ylläpitämiseen 0,55 W/m$^2$, mikä tekee siitä energiatehokkaamman kuin PDLC-lasitus, mutta se kuluttaa tilanmuutokseen kaksi kertaa enemmän energiaa kuin elektrokrominen lasitus [23]. Suspensiohiukkaslasituksen ongelmana on muun muassa taipumus imeä kosteutta, joka näkyy ajan myötä värikehyksenä (kuva 9). Kyseistä ongelmaa on pyritty estämään teippaamalla lasituksen reunaa, mutta värjäytynyt kehys on mahdollista tulla myös lämmön vaikutuksesta. [37]

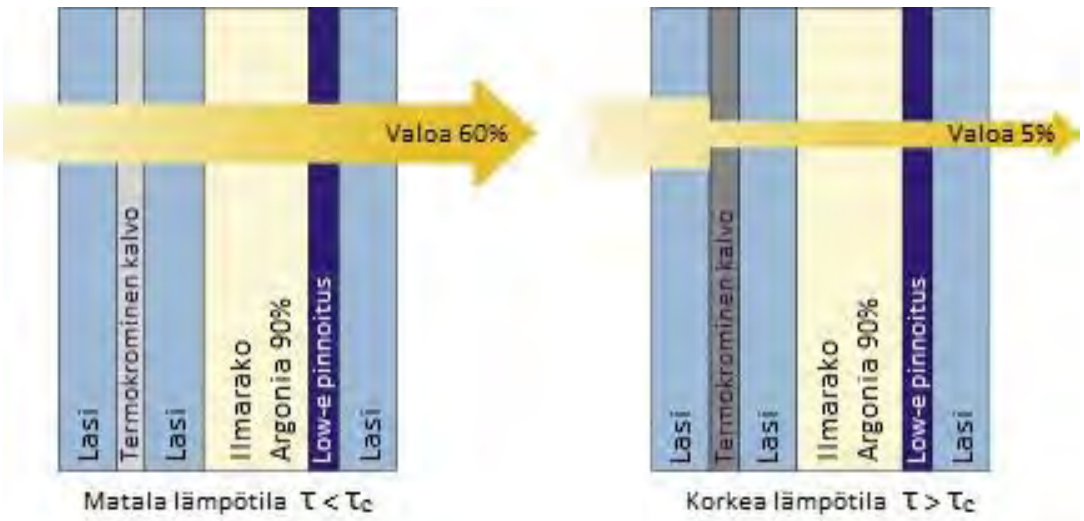



**Kuva 9:** Suspensiohiukkaslasituksen ympäristölle altistaminen neljäksi vuodeksi on aiheuttanut kalvolle värjäytyneen kehyksen, joka on merkattu punaisella nuolella kuvassa (c). Kyseistä lasitusta ei ole teipattu reunoista (mukaillen [43]).

## 3.4 Termokrominen lasitus

Termokrominen lasitus on passiivisesti toimiva laite, joka perustuu termokromisiin materiaaleihin. Termokromisilla materiaaleilla on ominaista muuttaa väritystään palautuvasti lämpötilan muutoksen seurauksena, mikä johtuu yleisesti materiaalin faasimuutoksesta tai kemiallisesta reaktiosta [44]. Termokromiset materiaalit ovat tyypillisesti läpinäkyviä matalissa lämpötiloissa ja tummenevat lämpötilan ylittäessä materiaalille ominaisen ”kriittisen” lämpötilan $\tau_c$ [7], [44], [45]. Termokromisen lasituksen tarkoituksena on estää liiallisen aurinkoenergian pääsyä kalvon läpi muuttamatta merkittävästi näkyvän valon läpäisevyyttä. Toisin sanoen, termokrominen lasitus säätelee auringon lämpösäteilyä muuttamalla laitteen läpäisevyyttä ja heijastuskykyä lähi-infrapuna-alueella.

Itsesäätelyn takia termokrominen lasitus ei tarvitse toimiakseen sähköä, ulkoisia johdotuksia tai ylipäätänsä minkäänlaista integraatiota rakennuksen kanssa [44], mikä tekee lasituksen rakenteesta yksinkertaisen. Tämän takia sitä ei voi myöskään ohjata käyttäjän toimesta. Termokrominen lasitus koostuu yleisesti joko termokromisella materiaalilla pinnoitetusta substraatista tai kahden substraatin väliin laminoituneesta termokromisesta kalvosta [46]. Ja kuten aiemmissakin laitteissa, nämä substraatit ovat yleisesti joko lasia tai muovia. Tällä hetkellä lupaavin teknillinen ratkaisu on sisällyttää termokromisia materiaaleja suoraan polyvinyylibutyraalista (PVB) valmistettuun muovikalvoon [23]. Tämän rakenteen lisäksi termokromisen lasituksen kanssa voidaan yhdistää muitakin rakenteita, kuten esimerkiksi kuvassa 10 olevan matalaemissiivisellä (*Low-e*) pinnoitteella vuoratun lasin, joka muodostaa argonilla (Ar) täytetyn ilmaraon termokromisen lasituksen kanssa. Nämä lisärakenteet parantavat laitteen lämpösuorituskykyä.

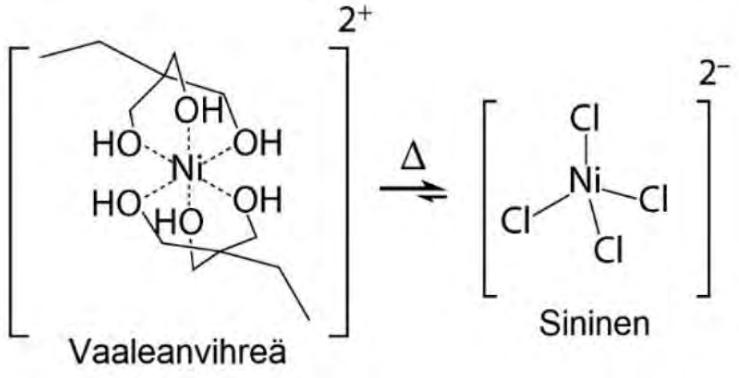



**Kuva 10:** Termokromisen lasituksen toimintaperiaate ja yleinen rakenne. Termokromisen kalvon lisäksi lasituksessa on argonilla täytetty ilmarako sekä matalaemissiivisellä (Low-e) pinnoitteella vuorattu lasi (mukaillen [23]).

Termokromisiin materiaaleihin kuuluu polymeerejä, pieniä orgaanisia molekyylejä, metalliligandikomplekseja sekä kiinteitä epäorgaanisia aineita. Useat konjugoituneet polymeerit, kuten polyasetyleenit (*polyacetylenes*), polysilaanit (*polysilanes*) ja polytiofeenit (*polythiophenes*) muuttuvat lämmetessään väritykseltään rakenteellisen muutoksen seurauksena. Konjugoituneet polymeerit ovat matalassa lämpötilassa tasaisia ja hyvin järjestäytyneitä kerrostumia, mutta lämpötilan noustessa yksittäiset polymeeriketjut kiertyvät, jolloin polymeerin aallonpituuksien absorptio muuttuu. Esimerkiksi polytiofeenit ovat purppuran värisiä huoneenlämmössä ja ne muuttuvat keltaisiksi lämpötilan noustessa. [44]

Orgaanisista molekyyleistä muun muassa biantrylideenit (*bianthrylidenes*), Schiffin emäkset (*Schiff bases*), spirooksatsiinit (*spirooxazines*) ja spiropyraanit (*spiropyrans*) omaavat termokromisia ominaisuuksia. Esimerkiksi bis-spiropyraanit muuttuvat läpinäkyvästä punaiseksi 60 °C ja punaisesta siniseksi 70 °C. Orgaanisia molekyylejä voidaan käyttää termokromisissa lasituksissa muodostamalla ohutkalvoja polymeerimatriisien kanssa. [44]

Termokromiset metalliligandikompleksit ovat molekyylejä, jotka muodostuvat keskellä olevasta metalli-ionista sekä siihen kiinnittyneistä ligandeista, jotka ovat yksittäisiä ioneja tai molekyylejä. Kyseisissä komplekseissa metalli-ionina toimii joko koboltti-, kupari-, rauta- tai nikkeli-ioni. Matalissa lämpötiloissa metalli-ionia koordinoi polydentaattiligandit ja korkeissa lämpötiloissa anioniset monodentaattiligandit (kuva 11) [44]. Polydentaattiligandeilla tarkoitetaan ligandeja, jotka muodostavat keskellä olevan metalli-ionin kanssa useita koordinaatiosidoksia (eli sidoselektronipareja) ja monodentaattiligandeilla tarkoitetaan ligandeja, jotka muodostavat metalli-ionin kanssa ainoastaan yhden koordinaatiosidoksen. Toisin sanoen, lämpötilan noustessa metalli-ionin ligandit vaihtuvat yksinkertaisempia koordinaatiosidoksia omaaviin ligandeihin, jolloin kompleksin geometrinen rakenne muuttuu johtaen tyypillisesti intensiivisempään väritykseen [44]. Termokromiset metalliligandikomplekseissa erikoisuutena on niiden lineaarinen tummeneminen eli ne tummenevat asteittain lämpötilan noustessa. Termokromisia metalliligandikomplekseja voidaan käyttää termokromisissa lasituksissa muodostamalla ohutkalvoja polymeerimatriisien kanssa, joissa polymeerit voivat olla metalliligandikompleksien kanssa osallisena ligandien vaihdossa. [44]




**Kuva 11:** $Ni^{2+}$-kompleksien ligandinvaihtotermokromismi trimetylolipropaani- ($C_6H_{14}O_3$) ja klooriligandien ($Cl^-$) kanssa (mukaillen [44]).

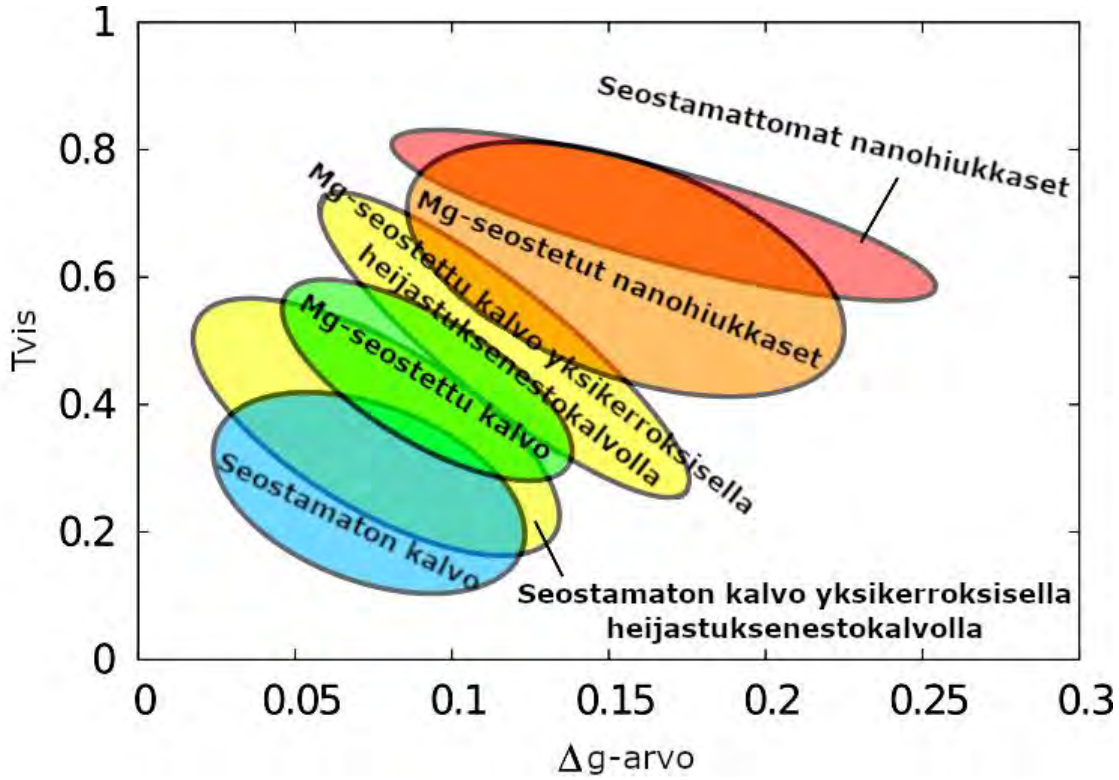
Termokromiselle lasitukselle soveltuvia kiinteitä epäorgaanisia aineita ovat useimmat alemman oksidin niobium-, rauta-, titaani- ja vanadiinioksidit. Ne omaavat ensimmäisen kertaluvun faasimuutoksen eli niillä on tarkka ”kriittinen” lämpötila $\tau_c$. [44] Toisaalta, vanadiinidioksidiin ($VO_2$) pohjautuvien materiaalien on sanottu olevan ainoita epäorgaanisia materiaaleja, jotka soveltuvat termokromisiin lasituksiin [7]. Vanadiinidioksidi on kiinteistä epäorgaanisista aineista eniten tutkittu [44] ja termokromisista materiaaleista lupaavin [47] sekä eniten käytetty [48] yhdiste. Sen etuna on muun muassa nopea vaste ympäristön lämpötilan muutoksille [48]. Vanadiinidioksidi kokee faasimuutoksen noin 68 °C asteessa, jossa sen kiderakenne muuttuu palautuvasti monokliinisestä faasista tetragonaaliseen rutiilifaasiin [44], [45], [47], [48]. Kiderakenteen palautumismekanismista on kiistelty vuosikymmeniä ja sitä ei vielä täysin ymmärretä. Vanadiinidioksidi käyttäytyy monokliinisessä faasissa kuin puolijohde, jolloin se läpäisee infrapunasäteilyä. Rutiilifaasissa vanadiinidioksidilla käyttäytyy puolestaan kuin metalli, jolloin se heijastaa infrapunasäteilyä. [47], [49]

Vanadiinidioksidin huonoina puolina ovat sen liian korkea ”kriittinen” lämpötila $\tau_c$ [7], [47], [49] matala näkyvän valon läpäisevyys $T_{vis}$ sekä matala auringon säteilyn kokonaisläpäisevyyden modulointialue eli $\Delta g$-*arvo* [7], [47]-[49]. Termokromisen lasituksen $\tau_c$ olisi suotavaa olla lähellä huoneenlämpöä, jotta lasitus aktivoituisi välittömästi lämpötilan noustessa yli huoneenlämmön, jolloin se estäisi liiallisen auringon säteilystä johtuvan lämpöenergian pääsyn rakennukseen [44]. Termokromista lasitusta voidaan siis ajatella eteisvahtimestarina: Jos tupa on ”täynnä” lämpöä, enempää ei saa päästää sisään. Mutta jos lämpöä on liian vähän, niin lisälämpö on kyllä tervetullutta. Vanadiinidioksidi on näiden ongelmien lisäksi myös termodynaamisesti epästabiili, sillä vanadiinidioksidi muuttuu ilman vaikutuksen alaisena divanadiinipentoksidiksi ($V_2O_5$) [7], [47]. Tämä voidaan estää pinnoittamalla kalvo alumiinioksidilla ($Al_2O_3$) [47] tai alumiininitridillä (AlN) [7].

Vanadiinidioksidin faasimuutokseen tarvittavaa lämpötilaa $\tau_c$ voidaan säädellä seostamisella: lisäämällä akseptoreita (*acceptor*), kuten alumiini- ($Al^{3+}$), kromi- ($Cr^{3+}$), rauta- ($Fe^{3+}$), gallium- ($Ga^{3+}$) tai germaniumioneja ($Ge^{3+}$) saadaan $\tau_c$ nousemaan ja puolestaan lisäämällä donoreita (*donor*), kuten molybdeeni- ($Mo^{6+}$), niobium- ($Nb^{5+}$), rutenium- ($Ru^{4+}$), tantaali- ($Ta^{5+}$) tai volframi-ioneja ($W^{6+}$) saadaan $\tau_c$ laskemaan [48]. Volframi-ioneilla seostaminen on tällä hetkellä yleisin vaihtoehto [45], [49]. Vain 2 m-% volframia riittää pudottamaan vanadiinidioksidin $\tau_c$:n noin 68 °C:sta noin 25 °C:een [44], [45], [49]. Kultananohiukkasilla (AuNPs) voidaan myös potentiaalisesti pienentää termokromisten materiaalien kriittistä lämpötilaa [44].

Vanadiinidioksidin huono läpinäkyvyys johtuu näkyvän valon absorptiosta, joka puolestaan johtuu vanadiinidioksidin kapeasta energiaraosta (*narrow band gap*) [7], [47], [48]. Läpinäkyvyyttä voitaisiin parantaa kalvoa ohentamalla, mutta silloin myös Δ*g-arvo* kärsisi. Yksi ratkaisu läpinäkyvyyden parantamiselle on lisätä heijastuksenestokalvoja (*antireflection layers*), joista titaanidioksidi ($TiO_2$) on todettu varsin toimivaksi. [7], [47] Käyttämällä useampaa titaanidioksidikalvoa $T_{vis}$ saavutti arvon 0,84. Toinen ratkaisu on pinnoittaa vanadiinidioksidi esimerkiksi piidioksidilla ($SiO_2$) tai zirkoniumdioksidilla ($ZrO_2$). Piidioksidipinnoitteella kalvon $T_{vis}$ parani 0,33:sta 0,75:een 700 nm aallonpituudella. [45] Alumiini-pii-pinnoitteella voidaan puolestaan parantaa sekä näkyvän valon läpäisevyyttä että auringon säteilyn kokonaisläpäisevyyden modulointialuetta [48]. Kolmas ratkaisu läpinäkyvyyden parantamiseksi on leventää energiarakoa fluoraamalla [7], [45], [47] tai seostamalla vanadiinidioksidia magnesiumilla (Mg) [7], [47], [48], sinkillä (Zn) [47] tai terbiumilla (Tb) [7]. Esimerkiksi magnesiumilla seostaessa 7,2 m-%:a $T_{vis}$ parani 0,39:stä 0,51:een [7]. Vanadiinidioksidia voidaan seostaa myös piidioksidi-nanohiukkasilla, jotka parantavat läpinäkyvyyden lisäksi auringon säteilyn kokonaisläpäisevyyden modulointialuetta [48]. Läpinäkyvyyttä voidaan parantaa vielä entisestään yhdistämällä seostamisen kanssa heijastuksenestokalvoja [7], [47].

Paras suorituskyky saadaan kuitenkin pallon muotoisilla vanadiinidioksidi-nanopartikkeleilla, jolloin $T_{vis}$ on noin 0,80 ja Δ*g-arvo* yli 0,20 [7], [47]. Vanadiinidioksidi-nanopartikkelit ovat myös pinnoitetta halutumpia, sillä ne voidaan sekoittaa kätevästi ohutkalvoihin [48]. Nanohiukkasten suorituskykyä ei yllättävästi voi parantaa magnesiumilla seostamalla, vaan se jopa huonontaa sitä, kuten kuvassa 12 näkyy.




**Kuva 12:** Vanadiinidioksidiin perustuvien termokromisten materiaalien arvioidut suorituskykyrajat monokliinisessä faasissa (mukaillen [7]).

Lupaavin tulevaisuuden termokrominen tekniikka on sisällyttää plasmonisia nanohiukkasia polymeeriin, joka palautuvassa lämpölaajentumisessa siirtäisi hiukkasia erilleen muuttaen nanohiukkasten eri aallonpituuksien absorbointikykyä [44].

## 3.5 Suorituskyky ja teknologinen valmiusaste

Jokaista kromogeenistä lasitusta voidaan kuvailla kyvyllä säädellä näkyvän valon läpäisevyyttä sekä auringon säteilyn kokonaisläpäisevyyttä riippumatta niiden aktivoitumismekanismista [3]. Työn lähteissä käytetään näkyvän valon läpäisevyyden arvona $T_{vis}$:ä, toiselta nimeltään V(L)T:tä (*Visible (Light) Transmittance*), ja auringon säteilyn kokonaisläpäisevyydessä käytetään arvona joko *g-arvoa*, $T_{sol}$:a (*Solar Transmittance*) tai SHGC:tä (*Solar Heat Gain Coefficient*), jotka tarkoittavat periaatteessa samaa asiaa. Työssä on käytetty lasitusten suorituskykyjen arvoina $T_{vis}$:ä sekä *g-arvoa*, jotka esitetään joko desimaaleissa (0,00–1,00) tai prosenteissa (0–100 %). 0,00 (0 %) tarkoittaa läpäisemätöntä lasitusta ja 1,00 (100 %) tarkoittaa täysin läpäisevää lasitusta. Lasituksien suorituskyvyssä käytetään lisäksi lämmönläpäisykerrointa eli *U-arvoa*, joka kertoo lasituksen läpäisevän lämpövuon pinta-alaa kohti yhden asteen lämpötilaerolla

(W/(m$^2$·K)): Mitä korkeampi *U-arvo*, sitä suurempi hukkalämpö ja mitä pienempi *U-arvo*, sitä pienempi hukkalämpö. Korkeammat *U-arvot* viittaavat tyypillisesti matalaan g-arvoon sekä korkeaan $T_{vis}$/*g-arvon* suhteeseen ja matalammat *U-arvot* viittaavat puolestaan korkeampiin $T_{vis}$:een ja *g-arvoon* [3].

Casini selvitti vuonna 2015 kirjallisuustutkimuksessaan [23] keskeisten markkinoilla olevien kromogeenisten lasitusten termo-optiset sekä sähköiset ominaisuudet. Casini valitsi markkinoilla olevien kromogeenisten lasitusten parhaimmat ominaisuudet ja esitti ne vertailutaulukossaan. Työhön on poimittu Casinin esittämästä vertailutaulukosta oleellisimmat ominaisuudet, jotka näkyvät taulukossa 1: EC viittaa elektrokromiseen lasitukseen, PDLC viittaa nestekide-lasitukseen, SPD viittaa suspensiohiukkaslasitukseen ja TC viittaa termokromiseen lasitukseen. Laskin taulukkoon lisäksi taulukon arvojen avulla säteilyn valotehokkuuden $K_e$, näkyvän valon modulointialueen $\Delta T_{vis}$ ja auringon säteilyn kokonaisläpäisevyyden modulointialueen $\Delta$*g-arvo*. Näistä lisää seuraavaksi.

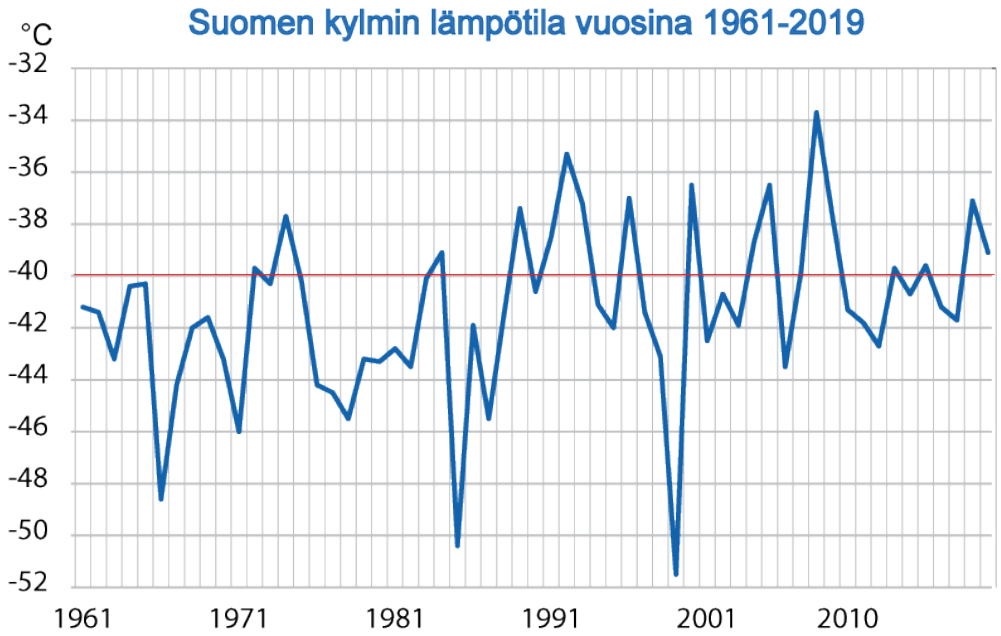
**Taulukko 1:** Keskeisten markkinoilla olevien kromogeenisten lasitusten vertailutaulukko (mukaillen [23]).

| **Ominaisuudet** | **Aktiiviset** | | | **Passiiviset** |
|---|---|---|---|---|
| | **EC** | **PDLC** | **SPD** | **TC** |
| **Termo-optiset ominaisuudet** | | | | |
| Näkyvän valon läpäisevyys, $T_{vis}$ (tumma–kirkas) | 0,01–0,60 | 0,50–0,75 | 0,005–0,65 | 0,05–0,60 |
| Näkyvän valon modulointialue, $\Delta T_{vis}$ | 0,59 | 0,25 | 0,645 | 0,55 |
| Auringon säteilyn kokonaisläpäisykerroin, *g-arvo* (tumma–kirkas) | 0,06–0,46 | 0,55–0,69 | 0,06–0,57 | 0,12–0,37 |
| Auringon säteilyn kokonaisläpäisevyyden modulointialue, $\Delta$*g-arvo* | 0,40 | 0,14 | 0,51 | 0,25 |
| Säteilyn valotehokkuus, $K_e$=$T_{vis}$/*g-arvo* (tumma–kirkas) | 0,17–1,30 | 0,91–1,09 | 0,08–1,14 | 0,42–1,62 |
| Käyttölämpötila (alaraja–yläraja) | -20–+70 °C | -20–+70 °C | -40–+120 °C | -20–+160 °C |
| **Sähköiset ominaisuudet** | | | | |
| Käyttöjännite | 12 V DC | 65–110 V AC | 65–110 V AC | — |
| Tilanmuutokseen tarvittava teho | 2,5 W/m$^2$ | 5,0–10,0 W/m$^2$ | 5,0 W/m$^2$ | — |
| Tilan ylläpitoon tarvittava teho | 0,4 W/m$^2$ | 5,0–10,0 W/m$^2$ | 0,55 W/m$^2$ | — |
| Kytkentänopeus | Tyypillisesti 3–5 min >90 % toimintasäteestä | Äkillinen (0,01 s) | Tyypillisesti 1–3 s | Muutama minuutti |

Kromogeenisten lasitusten suorituskykyä voidaan vertailla lasitusten näkyvän valon läpäisevyyden ja auringon säteilyn kokonaisläpäisevyyden modulointialueiden ($\Delta T_{vis}$ ja $\Delta g$-*arvo*) avulla [3]. Modulointialueella tarkoitetaan tässä yhteydessä läpäisevyyden vaihteluväliä; mitä suurempi vaihteluväli, sitä paremmin lasitus kykenee mukautumaan muuttuviin ilmaston olosuhteisiin. Taulukon 1 termo-optisia ominaisuuksia vertaamalla huomataan suspensiohiukkaslasituksella olevan suurimmat $\Delta T_{vis}$ ja $\Delta g$-*arvot* ja nestekidelasituksella pienimmät $\Delta T_{vis}$ ja $\Delta g$-*arvot*. Elektrokromisen lasituksen $\Delta T_{vis}$ ja $\Delta g$-*arvo* ovat hieman suspensiohiukkas-lasitusta huonompia. Termokromisen lasituksen $\Delta T_{vis}$ on myös lähellä suspensiohiukkaslasitusta, mutta sen $\Delta g$-*arvo* on selkeästi huonompi.

Säteilyn valotehokkuus $K_e$ kertoo lasituksen läpäisevän säteilyn näkyvän valon määrän verrattuna auringon säteilyn kokonaismäärään; jos $K_e$ on 0,10–0,58, säteily on pääsääntöisesti infrapunaa; jos $K_e$ on 1,365, säteilyssä on saman verran sekä näkyvää valoa että infrapunaa; jos $K_e$ on 2,41, säteily on pelkästään näkyvää valoa [3]. Taulukosta 1 nähdään kaikkien aktiivisten kromogeenisten lasitusten painottuvan suodattamaan ennemmin näkyvää valoa kuin infrapunaa. Termokrominen lasitus suodattaa myös tummana ennemmin näkyvää valoa kuin infrapunaa, mutta kirkkaana se suodattaa ennemmin infrapunaa näkyvän valon sijaan.

Käyttölämpötiloja vertaamalla havaitaan suspensiohiukkaslasituksen kestävän eniten kylmyyttä ja termokromisen lasituksen eniten lämpöä. Jotta kromogeeninen lasitus olisi luotettava, täytyy sen pystyä toimimaan myös vallitsevan ilmaston ääriolosuhteissa. Suomen alin lämpötila -51,5 °C mitattiin tammikuussa vuonna 1999 Kittilän Pokassa ja korkein lämpötila +37,2 °C Liperin Joensuun lentoasemalla heinäkuussa vuonna 2010 [50]. Taulukkoa 1 katsomalla jokaisen kromogeenisen lasituksen käyttölämpötila soveltuu Suomen kuumimpiin lämpötiloihin, mutta yksikään lasitus ei kykene toimimaan Suomen kylmimmissä lämpötiloissa (kuva 13). Toisaalta Suomen kylmimmät lämpötilat ovat aina Pohjois-Suomessa, enimmäkseen Lapissa [51], mikä ei vastaa Helsingin ilmastoa. Helsingissä lämpötila voi laskea alle -30 °C, mutta se ei ole vuodesta 1966 lähtien koskaan alittanut -40 °C [52]. Tämän perusteella kromogeenisistä lasituksista ainoastaan suspensiohiukkaslasitus soveltuisi Helsingin ilmastoon.

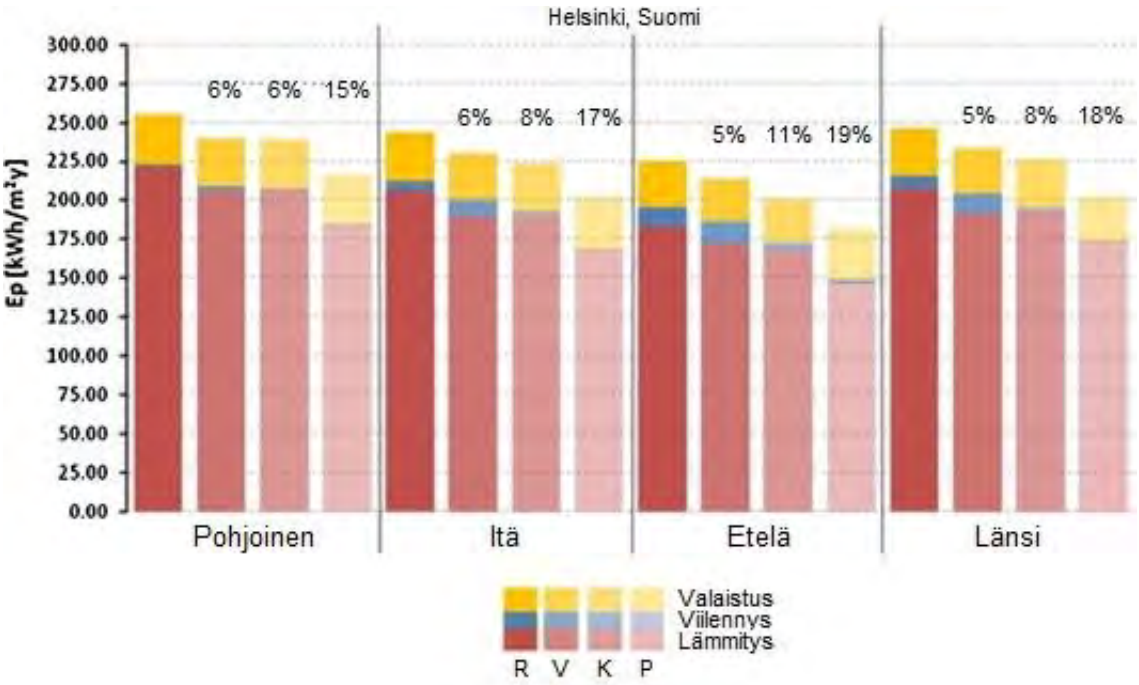



**Kuva 13:** Suomen kylmin lämpötila vuosina 1961–2019 (mukaillen [52]).

Sähköisiä ominaisuuksia vertaamalla huomataan, että nestekidelasitus kuluttaa kromogeenisistä lasituksista eniten ja elektrokrominen lasitus vähiten sähköä; termokrominen lasitus ei kuluta passiivisuutensa takia lainkaan sähköä. Nestekidelasitus tarvitsee tilanmuutokseen vähintään kaksi kertaa enemmän tehoa ja tilan ylläpitämiseen vähintään 12,5 kertaa enemmän tehoa verrattuna elektrokromiseen lasitukseen. Suspensiohiukkaslasitus tarvitsee tilanmuutokseen kaksi kertaa enemmän tehoa kuin elektrokrominen lasitus, mutta sen tilan ylläpito on miltei samaa luokkaa. Nestekidelasitus on markkinoilla olevista kromogeenisistä lasituksista nopein välittömällä (0,01 s) kytkentänopeudella ja elektrokrominen hitain 3–5 minuutin kytkentänopeudella. Jokainen lasitus kykenee siis adaptoitumaan noin viiden minuutin aikaikkunassa.

Kromogeenisen lasituksen optimaalinen suorituskyky riippuu lasituksen adaptoitumisen aikavälistä ja kyvystä moduloida termo-optisia ominaisuuksia. Lasituksen optimaaliset termo-optiset ominaisuudet riippuvat puolestaan ilmastosta sekä julkisivun suuntauksesta. Favoino ym. selvittivät vuonna 2015 adaptiiviselle lasitukselle optimaaliset parametrit käänteisellä simuloinnilla [3]. Heidän tapaustutkimuksessaan simuloitiin optimaalisia malleja Helsingin, Lontoon sekä Rooman ilmastoissa. Kuvassa 14 näkyy eri lasitusten vuosittainen primäärienergiankulutus eri pääilmansuunnissa. Referenssillä (R) tarkoitetaan staattista lasitettua julkisivua, joka täyttää viranomaisten asettamat minimivaatimukset. Vuosittaisella optimoinnilla (Y) tarkoitetaan adaptiivista lasitettua julkisivua, joka optimoidaan vuosittain minimoimaan primäärienergiankulutuksen ja se vastaa markkinoiden parasta staattista lasitettua julkisivua Helsingin ilmastossa. Kuukausittaisella adaptoitumisella (K) tarkoitetaan

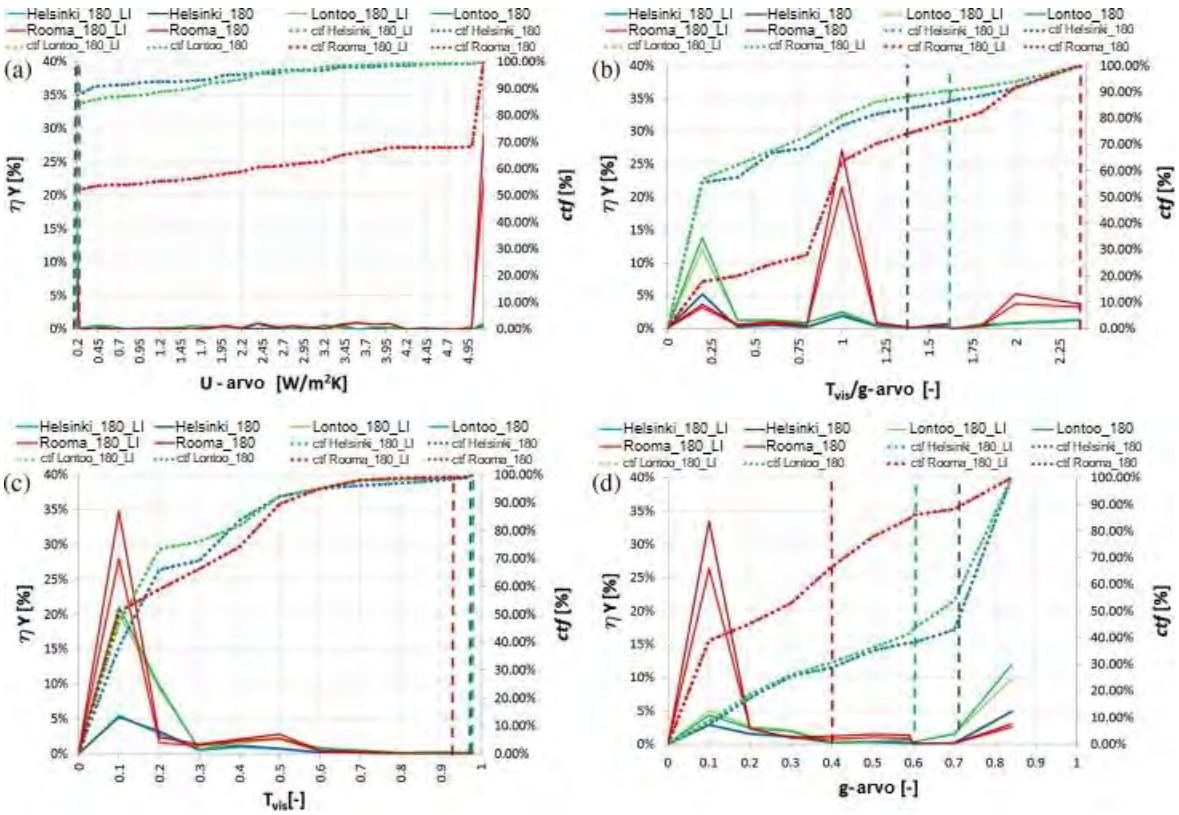
adaptiivista lasitettua julkisivua, joka adaptoituu kuukausittain ja päivittäisellä adaptoitumisella (P) tarkoitetaan adaptiivista lasitettua julkisivua, joka adaptoituu päivittäin. Kuten kuvasta 13 näkyy, tutkimuksen tuloksien mukaan suurimmat primäärienergian säästöt tulevat päivittäin adaptoituvien kromogeenisten lasituksien käytöstä, jolloin primäärienergian säästöt ovat referenssiin verrattuna 15–19 %. Taulukon 1 jokainen lasitus kykenee päivittäiseen adaptoitumiseen.




**Kuva 14:** Ideaalisen adaptiivisen lasituksen eri pääilmansuuntiin ja reaktioaikaan liittyvä spesifinen primäärienergiakäyttö Helsingin ilmastossa (R=referenssi, V=vuosittain optimoitu, K=kuukausittainen adaptoituminen, P=päivittäinen adaptoituminen). Prosenttiosuudet osoittavat energiansäästöt verrattuna Referenssi-julkisivuun (mukaillen [3]).

Jotta päivittäin adaptoituvien kromogeenisten lasitusten termo-optisten ominaisuuksien yleisimmät arvot ja intervallit saataisiin selville, Favoino ym. toteuttivat etelään suuntautuneille päivittäin adaptoituville lasitetuille julkisivuille taajuusanalyysin eri termo-optisille ominaisuuksille vuoden ajalta, jonka tulokset näkyvät kuvassa 15. Taajuusanalyysiin valittiin etelään suunnattu julkisivu sen suurimman primäärienergian säästön takia (kuva 14). Taajuusanalyysissä tarkastellaan lämmönläpäisykerrointa (*U-arvo*) välillä 0,2–5,14 W/(m$^2$·K), säteilyn valotehokkuutta ($K_e$=$T_{vis}$/*g-arvo*) välillä 0,00–2,41, näkyvän valon läpäisevyyttä ($T_{vis}$) välillä 0,01–0,84 ja auringon säteilyn kokonaisläpäisykerrointa (*g-arvo*) välillä 0,01–0,98. Favoino ym. valitsivat tarkasteluvälit kaksinkertaisten ikkunoiden termo-optisten ominaisuuksien pohjalta niiden suurimman vaihteluvälin takia [3]. Taajuusanalyysissä *ctf* tarkoittaa kumuloitunutta aikataajuutta (*cumulated time*

*frequency*), joka voi vaihdella 0 %:sta 100 %:iin ja se osoittaa adaptiivisten ominaisuuksien arvojen ajallisen osuuden vuoden aikana. Taajuusanalyysissä $\eta_Y$ tarkoittaa suorituskyvyn taajuutta (*performance frequency*), joka voi vaihdella 0 %:sta 100 %:iin ja se määrittää adaptiivisten ominaisuuksien arvojen energiansäästön verrattuna staattiseen vuosittain optimoituun ratkaisuun vuoden aikana. [3] *ctf*-kuvaajien kaltevuus osoittaa analysoitavan ominaisuuden arvon suurempaa esiintymistiheyttä, ja $\eta_Y$-kuvaajien piikit osoittavat säästetyn energian prosenttiosuuden. Katkonaiset pystyviivat edustavat analysoitavan ominaisuuden arvoja vuosittain optimoidussa tapauksessa. [3] Kuvan 15 mukaan päivittäin adaptoituvilla lasituksilla suurimmat energiansäästöt saadaan Rooman ja Lontoon ilmastoissa, mutta niillä on myös energiansäästöpotentiaalia Helsingin ilmastossa.




**Kuva 15:** Taajuusanalyysi etelään suuntautuneille ideaalisille adaptiivisten lasituksien ominaisuuksille Helsingin, Rooman ja Lontoon ilmastoissa: (a) U-arvo, (b) $K_e = T_{vis}$ / g-arvo, (c) $T_{vis}$ ja (d) g-arvo (mukaillen [3]).

Lämmönläpäisykertoimen taajuusanalyysiä tarkastelemalla (kuva 15a) nähdään, että Helsingin ilmastossa *U-arvo* on 90 % ajasta alimmassa mahdollisessa arvossaan (0,2 W/(m$^2$·K)) ja energiansäästö on *g-arvoa* muuttaessa olematon. Nämä viittaavat siihen, että *U-arvon* adaptiivisella muuttumisella ei ole merkittäviä hyötyjä Helsingin ilmastossa, kunhan se on vain mahdollisimman pieni. Casini esitti

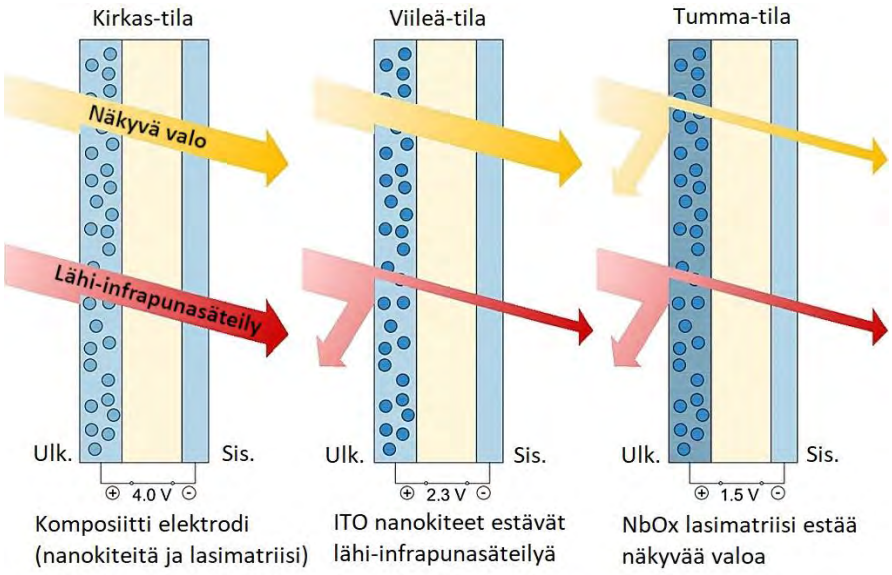
kirjallisuustutkimuksessaan ainoastaan markkinoilla olevien elektrokromisten lasitusten ja termokromisten lasituksien *U-arvot*, joten niitä ei ole lisätty taulukkoon 1. Markkinoilla olevista elektrokromisista lasituksista alin *U-arvo* on 1,10 ja termokromisista lasituksista alin on 1,36 [23], jotka ovat liian korkeita Helsingin ilmastoon.

Säteilyn valotehokkuuden taajuusanalyysiä (kuva 15b) tarkastelemalla nähdään, että Helsingin ilmastossa suurimmat energiasäästöt saadaan $K_e$:n ollessa noin 0,20 ja 1,00, jolloin energiansäästöt ovat 5 % ja 2 % luokkaa. *ctf*-kuvaajaa katsomalla havaitaan, että $K_e$ on välillä 0,00–0,25 yli puolet ajasta. Nämä viittaavat siihen, että Helsingissä optimaalisen kromogeenisten lasitusten spektrinen selektiivisyys painottuu auringon säteilyn infrapuna-alueelle. Toisaalta energiaa säästetään 1,7 % $K_e$:n ollessa 2,41, joten optimaalisimmat arvot ovat koko tarkasteluvälillä eli 0,00–2,41. Valotehokkuuden kannalta taulukon 1 elektrokrominen lasitus ja suspensiohiukkaslasitus soveltuisivat parhaiten Helsingin ilmastoon, mutta niiden $K_e$:n modulaatio ole lähelläkään optimaalista vaihteluväliä.

Näkyvän valon läpäisevyyden taajuusanalyysiä (kuva 15c) tarkastelemalla nähdään, että Helsingin ilmastossa suurin energiansäästö saadaan $T_{vis}$:n ollessa 0,10, jolloin energiaa säästyy noin 5,0 %. Eniten energiansäästöä saadaan $T_{vis}$:n ollessa 0,00–0,60 välillä, jossa se on noin 95 % ajasta. Tästä voidaan päätellä, että Helsingin ilmaston optimaalisimmat $T_{vis}$:n arvot liikkuvat 0,00–0,60 välissä. Taulukon 1 mukaan markkinoilla olevista kromogeenisistä lasituksista kaikkien paitsi nestekidelasituksen $T_{vis}$ soveltuu kyseiselle vaihteluvälille, mutta niistä paras on kuitenkin suspensiohiukkaslasitus matalimmalla näkyvän valon läpäisevyydellä (0,005).

Auringon säteilyn kokonaisläpäisykertoimen taajuusanalyysiä (kuva 15d) tarkastelemalla nähdään, että Helsingin ilmastossa suurimmat energiansäästöt saadaan *g-arvon* ollessa 0,10 ja 0,84, jolloin energiansäästö on noin 2,5 % ja 5,0 % luokkaa. Eniten energiansäästöä saadaan *g-arvon* ollessa 0,00–0,40 välillä sekä ylittäessä 0,70. *g-arvo* on noin 60 % ajasta yli 0,70, joka viittaa siihen, että Helsingin ilmastossa optimaalinen kromogeeninen lasitus pyrkii hyödyntämään mahdollisimman paljon auringon säteilyn lämpöenergiaa lämmityskauden aikana. Jos *g-arvo* saataisiin yli 0,84 maksimin, energiaa voitaisiin todennäköisesti säästää enemmän. Taajuusanalyysistä voidaan päätellä, että *g-arvon* optimaalisimmat arvot Helsingin ilmastossa ovat koko tarkasteluvälillä (0,00–0,84). Energiansäästön kannalta tärkeimmät arvot ovat kuitenkin yli 0,70, jota yksikään taulukon 1 kromogeeninen lasitus ei kykene saavuttamaan. Markkinoilla olevista kromogeenisistä lasituksista elektrokromisen lasituksen ja suspensiohiukkas-lasituksen *g-arvot* soveltuvat parhaiten Helsingin ilmastoon, sillä ne pystyvät moduloimaan auringon kokonaissäteilyä parhaiten välillä 0,00–0,40.

Taulukon 1 ja kuvan 15 perusteella nestekidelasitus soveltuu Helsingin ilmastoon huonoiten ja se soveltuukin tällä hetkellä ennemmin yksityisyyden lisäämiseen kuin ympäristöön adaptoitumiseen. Elektrokromisen lasituksen termo-optinen suorituskyky on lähellä suspensiohiukkaslasitusta, mutta se ei kestä Helsingin ilmaston kylmimpiä lämpötiloja, kuten ei myöskään termokrominen lasitus. Termokrominen lasitus on passiivisuutensa ansiosta energiatehokkain, mutta sen auringon säteilyn kokonaisenergian modulaatioalue on noin puolet kapeampi suspensiohiukkaslasitukseen ja elektrokromiseen lasitukseen verrattuna. Näiden perusteella voidaan todeta suspensiohiukkaslasituksen soveltuvan parhaiten Helsingin ilmastoon sen termo-optisten ominaisuuksien, kylmyyden sietokyvyn ja vähäisen virrankulutuksen ansiosta. Suspensiohiukkaslasituksen suorituskyvyssä on kuitenkin vielä parantamisen varaa, sillä $K_e$:n ja *g-arvon* modulointialueet eivät ole riittävän suuria. Favoinon ym. mukaan kromogeenisten lasituksien $K_e$:n modulointialue voitaisiin saada suuremmaksi, jos lasitukset kykenisivät säätämään samanaikaisesti sekä näkyvän valon että infrapunan spektriä [3]. Tämä ei kuitenkaan ole vielä markkinoilla olevilla kromogeenisillä lasituksilla mahdollista. Näkyvän valon ja infrapunan läpäisevyyttä voitaisiin säädellä erikseen yhdistämällä lasitukseen kaksi kromogeenistä kalvoa, joista toinen toimisi näkyvän valon alueella ja toinen infrapunan alueella [3]. Tutkijat ovat myös kehitelleet tinalla seostetuista indiumoksidi-nanokiteistä ($In_2O_3$:Sn) ja niobiumoksidimatriisista ($NbO_x$) muodostetun komposiittimateriaalin, joka kykenee selektiivisesti estämään lähi-infrapunasäteilyä ja näkyvää valoa jännitettä säätämällä (kuva 16): Nanokiteet ja $NbO_x$-lasimatriisi ovat läpinäkyviä 4,0 V:ssa. Pienentämällä jännitettä 2,3 V:iin nanokiteet estävät lähi-infrapunasäteilyä ja pienentämällä jännitettä 1,5 V:iin $NbO_x$-lasimatriisi estää tämän lisäksi näkyvää valoa [53].




**Kuva 16:** Indiumoksidi-nanokiteistä (In2O3:Sn; ITO) ja niobiumoksidimatriisista (NbOx) muodostetun komposiittimateriaalin toimintaperiaate (mukaillen [54]).

# 4 Yhteenveto

Tässä kandidaatintyössä tarkasteltiin rakennusten suorituskyvyn parantamista adaptiivisten julkisivuratkaisujen avulla, joista tarkemman tarkastelun alle valittiin kromogeeniset lasitukset. Tämän kandidaatintyön tavoitteena oli selvittää adaptiivisten julkisivujen määritelmä ja ominaispiirteet sekä kromogeenisten lasituksien soveltuvuus Helsingin ilmastoon.

Adaptiivisella julkisivulla tarkoitetaan mukautuvaa rakennuksen vaippaa, joka muodostuu monipuolisista järjestelmistä. Järjestelmät pyrkivät parantamaan rakennuksen suorituskykyä muuttamalla ajan myötä ominaisuuksiaan, toimintojaan tai käyttäytymistään, jotta ne voivat säädellä rakennukseen pääsevää luonnonvalon määrää, lämpöä ja massavirtaa. Adaptiiviset julkisivut voidaan jakaa neljään eri luokkaan; aurinkoaktiivisiin julkisivuihin, dynaamisiin varjostimiin, aktiivisesti tuulettuviin julkisivuihin sekä kromogeenisiin julkisivuihin.

Kromogeeniset lasitukset pyrkivät minimoimaan rakennuksien kokonais-energiankulutusta säätämällä termo-optisia ominaisuuksiaan tummenemalla tai kirkastumalla. Kromogeeniset lasitukset perustuvat kromogeenisiin materiaaleihin, joiden termo-optisia ominaisuuksia voidaan säädellä ulkoisen ärsykkeen, kuten sähkökentän, avulla. Kromogeenisten lasitusten tavoitteena on mukautua ulkoisiin ympäristön muutoksiin sekä sisäisiin kuormiin, kuten ihmisten määrään, valaistukseen ja laitteiden käyttöön. Kromogeenisistä lasituksista tarkastelun kohteena olivat elektrokrominen lasitus, nestekidelasitus, suspensiohiukkaslasitus ja termokrominen lasitus.

Markkinoilla olevien kromogeenisten lasitusten suorituskykyjä ja Helsingin ilmaston optimaalisia ominaisuuksia vertaillessa todettiin, että suspensiohiukkas-lasitus soveltuu Helsingin ilmastoon parhaiten sen termo-optisten ominaisuuksien, kylmyyden sietokyvyn ja vähäisen virran-kulutuksen ansiosta. Suspensio-hiukkaslasituksen suorituskyvyssä on kuitenkin vielä parantamisen varaa, varsinkin auringon säteilyn kokonaisläpäisevyyden moduloinnissa.

Favoinon ym. julkaiseman datan perusteella ei ollut mahdollista selvittää tarkkoja Helsingin ilmastolle optimaalisia termo-optisten ominaisuuksien arvoja, sillä ne olivat esitetty ainoastaan kuvaajissa eikä numeroina, minkä takia lasitusten soveltuvuudesta ei kyetty tekemään täsmällisiä johtopäätöksiä. Lasitusten optimaalisia termo-optisia ominaisuuksia tarkasteltiin ainoastaan Helsingin ilmastossa, mikä poikkeaa paljon muusta Suomesta. Olisi siis hyvä, jos optimaalisia ominaisuuksia simuloitaisiin myös muissa Suomen ilmastoissa.

Favoinon ym. tuloksista nähtiin, että mitä lyhyempi adaptoitumisväli oli, sitä enemmän säästettiin energiaa, mutta heidän tutkimuksessaan otettiin huomioon ainoastaan päivittäin adaptoituvat lasitukset. Voisiko energiaa säästää enemmän, jos lasitukset adaptoituisivat vaikka tunneittain tai jopa kerran viidessä minuutissa? Tämä olisi ainakin nykyisillä markkinoilla olevilla kromogeenisillä lasituksilla mahdollista, joten suorituskyvyn optimointia voitaisiin tutkia lyhyemmillä adaptoitumisväleillä.

# 5 Lähteet

[1] United Nations, "World Population Prospects - Population Division - United Nations," *United Nations Department of Economic and Social Affairs,* 2019. Saatavissa: https://population.un.org/wpp/Download/Files/1_Indicators%20(Standard)/EXCEL_FILES/1_Population/WPP2019_POP_F01_1_TOTAL_POPULATION_BOTH_SEXES.xlsx.

[2] J. Patronen, E. Kaura & C. Torvestad. (2017). *Nordic heating and cooling* s. 15-19. Saatavissa: doi: 10.6027/TN2017-532.

[3] F. Favoino, M. Overend & Q. Jin. (2015). The optimal thermo-optical properties and energy saving potential of adaptive glazing technologies. *Appl. Energy 156* s. 1-15. Saatavissa: doi: 10.1016/j.apenergy.2015.05.065.

[4] S. Attia *ym.* (2018). Current trends and future challenges in the performance assessment of adaptive façade systems. *Energy and Buildings 179* s. 165-182. Saatavissa: doi: 10.1016/j.enbuild.2018.09.017.

[5] S. Attia, R. Lioure & Q. Declaude. (2020). Future trends and main concepts of adaptive facade systems. *Energy Science & Engineering 8(9)* s. 3255-3272. Saatavissa: doi: 10.1002/ese3.725.

[6] R. Romano *ym.* (2018). What is an adaptive façade? Analysis of Recent Terms and definitions from an international perspective. *Journal of Facade Design and Engineering 6(3)*. Saatavissa: doi: 10.7480/jfde.2018.3.2478.

[7] C. G. Granqvist *ym.* (2016). Thermochromic vanadium-dioxide-based thin films and nanoparticles: Survey of some buildings-related advances. *Journal of Physics. Conference Series 764*. Saatavissa: doi: 10.1088/1742-6596/764/1/012002.

[8] Grand View Research. (2019). Smart Glass Market Size & Share | Global Industry Report, 2019-2025. Saatavissa: https://www.grandviewresearch.com/industry-analysis/smart-glass-market.

[9] *TU1403 - Adaptive Facades Network,* 2014. Saatavissa: https://www.cost.eu/actions/TU1403/#tabs|Name:overview.

[10] U. Desideri & F. Asdrubali, "Chapter 6 - Building Envelope," *Handbook of Energy Efficiency in Buildings,* Butterworth-Heinemann, 2019, s. 295-439. Saatavissa: doi: 10.1016/B978-0-12-812817-6.00039-5.

[11] A. Ghaffarianhoseini *ym.* (2016). Exploring the advantages and challenges of double-skin façades (DSFs). *Renewable and Sustainable Energy Reviews 60* s. 1052-1065. Saatavissa: doi: 10.1016/j.rser.2016.01.130.

[12] V. Balog. (2019). Innovative Facade Types – The Closed Cavity Facade. *Glassonweb.Com*. Saatavissa: https://www.glassonweb.com/article/innovative-facade-types-closed-cavity-facade.

[13] A. B. Besir & E. Cuce. (2018). Green roofs and facades: A comprehensive review. *Renewable and Sustainable Energy Reviews 82* s. 915-939. Saatavissa: doi: 10.1016/j.rser.2017.09.106.

[14] E. Biyik *ym.*, "A key review of building integrated photovoltaic (BIPV) systems," *Engineering Science and Technology, an International Journal,* vol. 20, *(3),* s. 833-858, 2017. Saatavissa: doi: 10.1016/j.jestch.2017.01.009.

[15] S. C. Bhatia, "1 - Energy resources and their utilisation," *Advanced Renewable Energy Systems,* Woodhead Publishing India, 2014, s. 1-31. Saatavissa: doi: 10.1016/B978-1-78242-269-3.50001-2.

[16] L. F. Cabeza, A. Castell and G. Pérez, "13 - Life cycle assessment (LCA) of phase change materials (PCMs) used in buildings," *Eco-Efficient Construction and Building Materials,* Elsevier Ltd, 2014, s. 287-310. Saatavissa: doi: 10.1533/9780857097729.2.287.

[17] J. Laverge, S. Schouwenaars, M. Steeman, A. Janssens & N. Van Den Bossche, "Condensation in a closed cavity double skin facade: a model for risk assessment," Building Envelope Systems and Technologies, International conference, Proceedings, Vancouver, BC, Canada, 2010, s. 125–134. Saatavissa: http://hdl.handle.net/1854/LU-1045855.

[18] C. Granqvist, "Electrochromic Metal Oxides: An Introduction to Materials and Devices," *Electrochromic Materials and Devices,* Weinheim, Germany: Wiley-VCH Verlag GmbH & Co. KGaA, 2013, s. 1-40. Saatavissa: doi: 10.1002/9783527679850.ch1.

[19] V. K. Thakur *ym.* (2012). Hybrid Materials and Polymer Electrolytes for Electrochromic Device Applications. *Advanced Materials (Weinheim) 24(30)* s. 4071-4096. Saatavissa: doi: 10.1002/adma.201200213.

[20] S. D. Rezaei, S. Shannigrahi & S. Ramakrishna. (2017). A review of conventional, advanced, and smart glazing technologies and materials for improving indoor environment. *Solar Energy Materials and Solar Cells 159* s. 26-51. Saatavissa: doi: 10.1016/j.solmat.2016.08.026.

[21] C. G. Granqvist. (2014). Electrochromics for smart windows: Oxide-based thin films and devices. *Thin Solid Films 564* s. 1-38. Saatavissa: doi: 10.1016/j.tsf.2014.02.002.

[22] C. G. Granqvist *ym.* (2018). Electrochromic materials and devices for energy efficiency and human comfort in buildings: A critical review. *Electrochimica Acta 259* s. 1170-1182. Saatavissa: doi: 10.1016/j.electacta.2017.11.169.

[23] M. Casini, "Smart windows for energy efficiency of buildings," *Proceedings of Second International Conference on Advances in Civil, Structural and Environmental Engineering–ACSEE,* 2014, s. 273-281. Saatavissa: doi: 10.15224/ 978-1-63248-030-9-56.

[24] R. Wen, C. G. Granqvist & G. A. Niklasson. (2015). Eliminating degradation and uncovering ion-trapping dynamics in electrochromic WO3 thin films. *Nature Materials 14(10)* s. 996-1001. Saatavissa: doi: 10.1038/nmat4368.

[25] A. Hemaida *ym.* (2020). Evaluation of thermal performance for a smart switchable adaptive polymer dispersed liquid crystal (PDLC) glazing. *Solar Energy 195* s. 185-193. Saatavissa: doi: 10.1016/j.solener.2019.11.024.

[26] Y. Kim *ym.* (2015). Optical properties and optimized conditions for polymer dispersed liquid crystal containing UV curable polymer and nematic liquid crystal. *Current Applied Physics 15(3)* s. 292-297. Saatavissa: doi: 10.1016/j.cap.2014.12.027.

[27] C. Li *ym.* (2019). Electro-Optical Properties of a Polymer Dispersed and Stabilized Cholesteric Liquid Crystals System Constructed by a Stepwise UV-Initiated Radical/Cationic Polymerization. *Crystals (Basel) 9(6)*. Saatavissa: doi: 10.3390/cryst9060282.

[28] D. Jung *ym.* (2017). Inorganic gel and liquid crystal based smart window using silica sol-gel process. *Solar Energy Materials and Solar Cells 159* s. 488-495. Saatavissa: doi: 10.1016/j.solmat.2016.10.001.

[29] A. Ghosh & B. Norton. (2018). Advances in switchable and highly insulating autonomous (self-powered) glazing systems for adaptive low energy buildings. *Renewable Energy 126* s. 1003-1031. Saatavissa: doi: 10.1016/j.renene.2018.04.038.

[30] M. Pande *ym.* (2016). Dielectric and electro-optical properties of polymer-stabilized liquid crystal system. *Appl. Phys. A 122(3)* s. 1-9. Saatavissa: doi: 10.1007/s00339-016-9749-8.

[31] H. Hosseinzadeh Khaligh *ym.* (2015). Silver nanowire transparent electrodes for liquid crystal-based smart windows. *Solar Energy Materials and Solar Cells 132* s. 337-341. Saatavissa: doi: 10.1016/j.solmat.2014.09.006.

[32] R. Yamaguchi, K. Inoue & R. Kurosawa. (2016). Effect of Liquid Crystal Material on Polymer Network Structure in Polymer Stabilized Liquid Crystal Cell. *Journal of Photopolymer Science and Technology 29(2)* s. 289-292. Saatavissa: doi: 10.2494/photopolymer.29.289.

[33] X. Hu *ym.* (2020). Stable and scalable smart window based on polymer stabilized liquid crystals. *Journal of Applied Polymer Science 137(30)*. Saatavissa: doi: 10.1002/app.48917.

[34] H. Khandelwal, Schenning, Albertus P. H. J & M. G. Debije. (2017). Infrared Regulating Smart Window Based on Organic Materials. *Advanced Energy Materials 7(14)*. Saatavissa: doi: 10.1002/aenm.201602209.

[35] M. Casini. (2018). Active dynamic windows for buildings: A review. *Renewable Energy 119* s. 923-934. Saatavissa: doi: 10.1016/j.renene.2017.12.049.

[36] D. Barrios *ym.* (2013). Toward a quantitative model for suspended particle devices: Optical scattering and absorption coefficients. *Solar Energy Materials and Solar Cells 111* s. 115-122. Saatavissa: doi: 10.1016/j.solmat.2012.12.012.

[37] SPD films and light valve laminates with improved durability, D. Wang *ym.* (Mar 14, 2014). *US 8,670,173 B2.* Saatavissa: https://patents.google.com/patent/US8670173B2/en.

[38] R. Vergaz *ym.* (2008). Modelling and electro-optical testing of suspended particle devices. *Solar Energy Materials and Solar Cells 92(11)* s. 1483-1487. Saatavissa: doi: 10.1016/j.solmat.2008.06.018.

[39] D. Barrios *ym.* (2015). Simulation of the thickness dependence of the optical properties of suspended particle devices. *Solar Energy Materials and Solar Cells 143* s. 613-622. Saatavissa: doi: 10.1016/j.solmat.2015.05.044.

[40] SPD films with darker off-state transmittances and lighter on-state transmittances, X. Chen, Steven M. Slovak & Seth Van Voorhees. (2014). *US 8,922,872 B2.* Saatavissa: https://patents.google.com/patent/US8922872B2/en.

[41] SPD light valves incorporating films comprising improved matrix polymers and methods for making such matrix polymers, S. M. Slovak, Xiao-Ping Chen & Robert L. Saxe. (2010). *US 7,791,788 B2.* Saatavissa: https://patents.google.com/patent/US7791788B2/en.

[42] SPD films and light valves comprising same, S. Chakrapani *ym.* (2002). *US 6,416,827 B1.* Saatavissa: https://patents.google.com/patent/US6416827/en/en22.

[43] A. Ghosh & B. Norton. (2017). Durability of switching behaviour after outdoor exposure for a suspended particle device switchable glazing. *Solar Energy Materials and Solar Cells 163* s. 178-184. Saatavissa: doi: 10.1016/j.solmat.2017.01.036.

[44] Y. Wang, E. L. Runnerstrom & D. J. Milliron. (2016). Switchable Materials for Smart Windows. *Annu. Rev. Chem. Biomol. Eng. 7(1)* s. 283-304. Saatavissa: doi: 10.1146/annurev-chembioeng-080615-034647.

[45] M. Kamalisarvestani *ym.* (2013). Performance, materials and coating technologies of thermochromic thin films on smart windows. *Renewable and Sustainable Energy Reviews 26* s. 353-364. Saatavissa: doi: 10.1016/j.rser.2013.05.038.

[46] W. Haase, M. Husser & W. Sobek, "Adaptive glazing systems - survey of systems," Jul 2017, s. 929-933. Saatavissa: doi: 10.1109/AIM.2017.8014137.

[47] C. G. Granqvist. (2016). Electrochromics and Thermochromics: Towards a New Paradigm for Energy Efficient Buildings. *Materials Today: Proceedings 3* s. 2-11. Saatavissa: doi: 10.1016/j.matpr.2016.01.002.

[48] G. Gorgolis & D. Karamanis. (2016). Solar energy materials for glazing technologies. *Solar Energy Mater. Solar Cells 144* s. 559-578. Saatavissa: doi: 10.1016/j.solmat.2015.09.040.

[49] M. Aburas *ym.* (2019). Thermochromic smart window technologies for building application: A review. *Appl. Energy 255*. Saatavissa: doi: 10.1016/j.apenergy.2019.113522.

[50] Ilmatieteen laitos. (2020). Lämpötilaennätyksiä. Viitattu: 15.11.2020. Saatavissa: https://www.ilmatieteenlaitos.fi/lampotilaennatyksia.

[51] Ilmatieteen laitos. (2020). Lämpimin ja kylmin paikka vuosittain. Viitattu: 15.11.2020. Saatavissa: https://www.ilmatieteenlaitos.fi/lampimin-ja-kylmin-paikka-vuosittain.

[52] Ilmatieteen laitos. (2019). Kovat pakkaset ja talven kylmimmät. Viitattu: 15.11.2020. Saatavissa: https://www.ilmatieteenlaitos.fi/kovat-pakkaset-ja-kylmimmat-talvet.

[53] A. Llordés *ym.* (2013). Tunable near-infrared and visible-light transmittance in nanocrystal-in-glass composites. *Nature 500(7462)* s. 323-326. Saatavissa: doi: 10.1038/nature12398.

[54] M. Casini. (2018). Active dynamic windows for buildings: A review. *Renewable Energy 119* s. 923-934. Saatavissa: doi: 10.1016/j.renene.2017.12.049.

# Liite 2 Diplomityö Applying Machine Learning to Develop Black-box Control Model of Active Double-Skin Façade

# Applying Machine Learning to Develop Black-box Control Model of Active Double-Skin Facade

Thao Nguyen

**School of Electrical Engineering**

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 31.12.2020

**Supervisor**

Prof. of Practice Heikki Ihasalo

**Advisor**

Prof. Alex Jung
Prof. Arto Visala

Aalto University
School of Electrical
Engineering

**Aalto University**
School of Electrical
Engineering

**Aalto University, P.O. BOX 11000, 00076 AALTO**
**www.aalto.fi**
**Abstract of the master's thesis**

**Author** Thao Nguyen

**Title** Applying Machine Learning to Develop Black-box Control Model of Active Double-Skin Facade

**Degree programme** Department of Electrical Engineering and Automation

**Major** Control, Robotics and Autonomous Systems **Code of major** ELEC3025

**Supervisor** Prof. of Practice Heikki Ihasalo

**Advisor** Prof. Alex Jung
Prof. Arto Visala

**Date** 31.12.2020 **Number of pages** 72+46 **Language** English

**Abstract**

The efficient energy performance of an active double-skin facade (DSF) has raised more attention to study and apply for developing the building control strategies and systems. Although DSFs can be actively operated in dynamic modes with controllable components such as shading slat angle, airflow path, and airflow rate, no autonomous control model has been deployed to utilize their maximum potential for building energy efficiency.

This thesis aims to apply Machine Learning (ML) to train predictive models for developing a black-box control model that estimates the deliverable operational modes of DSF for the desired energy performance parameters under the related environmental conditions. An autonomous control system of DSF to be developed based on applied ML algorithms as an advanced building control strategy is challenging to carry out for the first time. Data acquisition (DAQ) is also to be planned for future works.

The steady-state and dynamic simulations of a specific configuration of DSF were set up in EnergyPlus® and used as training data processed in Python® scripts. The simulations conducted thermal and visual performance against the possible variations of realistic boundary conditions and operational modes of DSF. After data analysis and identifying the ML problems, ANN, RF/ET, XGB predictive models were to learn for visual metric, airflow mode, and thermal metric in different airflow modes. Eventually, the black-box models were compared and selected according to several defined criteria of reliability. The DSF controller was designed by combining the selected models to control DSF operational modes and predict corresponding visual and thermal metrics. Python-programmed ML software libraries used are Scikit-learn, Keras based Tensorflow backend, and XGBoost.

In conclusion, the ML black-box models probably suffer from overfitting and instability due to noises in the real world. The proposed solution to reduce variance is to enlarge training data and retrain the black-box models by online transfer learning. Otherwise, it is still highly recommended to proceed with the reinforcement learning approach or hybrid models promising to overcome the limitations of the black-box models.

**Keywords** double-skin facade, machine learning, black-box models, artificial neural network, decision tree ensembles, XGBoost

# Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisor Dr. Heikki Ihasalo for always being a warm, patient, and sympathetic person who has ever best guided and encouraged me to fulfill this work. Next, I would also like to thank my two advisors Dr. Alex Jung and Dr. Arto Visala for being present on my journey and teaching me good lessons with their wisdom.

Particularly, I owe my deep gratitude to Dr. Francesco Goia who gave me the chance to join this amazing project as well as the invaluable experiences and supports with the best enthusiasm from him I have ever received. I also delightfully thank another assistant in the research group M.Sc Gaurav Chaudhary for providing me indirect but active support with the main background and material for my work.

Besides, I am grateful for my company and my colleagues who helped to facilitate the sharing of job tasks and gave me the good conditions to concentrate and finish my final college work. I also appreciate my friends who have always motivated me.

Finally, I give my honor to my parents who deserve more than I can give, and to my sweetheart and my dear sisters who have always encouraged and given me wise counsel.

Espoo, 31.12.2020

Thao Nguyen

# Contents

# Symbols and abbreviations

## Symbols

| | |
|---|---|
| `Airflow_ctrl` | Output control signal for Airflow mode |
| `Blinds_ctrl` | Output control signal for Blinds angle |
| `Speed_ctrl` | Output control signal for Airflow speed |
| `T_pred` | Predicted thermal metric |
| `V_pred` | Predicted visual metric |
| `Airflow` | Airflow mode |
| `Altitude` | Solar altitude angle |
| `Azimuth` | Solar azimuth angle |
| `Blinds` | Tilt angle of shading blinds |
| `SR_diffuse` | Diffused solar radiation |
| `SR_direct` | Direct solar radiation |
| `SR` | Solar radiation |
| `Speed` | Airflow speed |
| `DeltaT` | Temperature difference between outdoor air and indoor air |
| `T_in` | Indoor air temperature |
| `T_out` | Outdoor air temperature |
| `T_opt` | Optimal thermal metric |
| `V_opt` | Optimal visual metric |
| `T_real` | real-time thermal metric |
| `V_real` | real-time visual metric |
| $R^2$ | Coefficient of determination |

## Abbreviations

| | |
|---|---|
| AB | Adaptive Boosting (AdaBoost) |
| ANN | Artificial Neural Network |
| API | Application Programming Interface |
| AWS | Amazon Web Services |
| BMS | Building Management System |
| CART | Classification and Regression Tree |
| CNN | Convolutional Neural Network |
| CPU | Central Processing Unit |
| CUDA | Compute Unified Device Architecture |
| DAQ | Data Acquisition |
| DL | Deep Learning |
| DSF | Double Skin Facade |
| DT | Decision Tree |
| EN | Elastic Net |
| ET | Extra Trees |
| GPU | Graphic Processing Unit |
| HVAC | Heating, Ventilation, and Air conditioning |
| kNN/KNN | k-nearest neighbors |
| LASSO | l1-regularized Linear Regressor |
| LDA | Linear Aiscriminant Analysis |
| LOG | Logistic Regression |
| LOG | Linear Regression |
| MAE | Mean Absolute Error |
| ML | Machine Learning |
| MLP | Multi-layer Perceptron |
| MPC | Model Predictive Control |
| MSE | Mean Squared Error |
| NMSE | Negative Mean Squared Error |
| RBF(N) | Radial Basis Function (Network) |
| REINVENT | REsponsive, INtegrated and VENTilated Windows |
| RF | Random Forest |
| RFC/RFR | Random Forest Classifier/Regressor |
| RL | Reinforcement Learning |
| RNN | Recurrent Neural Network |
| RPi | Raspberry Pi |
| SGD | Stochastic Gradient Descent |
| SL | Supervised Learning |
| SSL | Semi-supervised Learning |
| SVM/SVC/SVR | Support Vector Machine/Classification/Regression |
| TDNN | Time-delayed Neural Network |
| TPU | Tensor Processing Unit |
| USL | Unsupervsied Learning |
| XGB | Extreme Gradient Boosting (XGBoost) |

# 1 Introduction

Building envelops techniques have been remarkably advancing all the time since the first glazing facades were adopted to replace buildings' enclosing walls in the late $19^{th}$ century [1]. The trend of using glazing facades increases with aesthetic purposes obviously while building envelope techniques with glazing facades of amazing and admirable state-of-the-art architectures are incredibly developing with today technology for energy efficiency and human comfort in this $21^{st}$ century. Indeed, the demand for passive solar energy-saving techniques and natural ventilation in buildings arose with the significance of energy consumption counted for up to forty percent by buildings back to the $20th$ century [2]. Being one of the most advanced glazing facades, the double-skin facade (DSF) has become an increasing architectural and important building-energy-integrated element in office buildings over the last 30 years.

Among dynamic transparent building envelope technologies, DSFs have been popularly utilized for efficient ventilation, occupant comfort, fire protection, acoustic insulation, good lighting while reducing building energy consumption and improving indoor air quality. The unique architecture of DSF benefits in many use cases, for example: (1) the air cavity with the width ranging from 0.2m to more than 2m separating the inner and external glazing facades of DSF and acting as a buffer space for controlled ventilation and solar protection, (2) the external facade made of heat-strengthened and/or laminated safety glass protecting against weather and noise, (3) cold air circulated between two glazing facades of a mechanically ventilated DSF in summer and hot air ventilated in winter to minimize the demand for extra cooling and heating energy loads, (4) a closed DSF increasing thermal insulation for the building to prevent the heat loss in winter season or cold climates but a ventilated DSF reducing the heat gain and the cooling demand in summer or hot climates [3].

However, a common issue of glazing facades is concerned with the thermal problem as passing high level of direct solar radiation entering the internal spaces, and thereby regions in hot climate particularly are facing significant high costs for their cooling energy consumption rate [2]. This problem can be significantly improved with efficient solar control and shading strategies for DSF. Besides, the source and destination of the cavity airflow should be optimized depending on many factors such as the climate, the location, the HVAC strategy, the occupational hours of the building, and the use case. Although DSFs can be actively operated in dynamic modes with controllable components such as shading slat angle, airflow path, and airflow rate, no autonomous control model has been deployed to utilize their maximum potential for energy efficiency of the building system.

## 1.1 Motivations

The topic was first introduced to the author when working as a research assistant in a part of the ongoing research project REINVENT (REsponsive, INtegrated and VENTilated Windows) which has been funded by the Research Council of Norway. The representatives of this research group are Prof. Francesco Goia from NTNU

University and Prof. Heikki Ihasalo from Aalto University in cooperation between these two international academic institutes for the subject of prototyping a new autonomous control valve of DSF. The thesis comes as another critical part of the project that is based on the previous research outcome of M.Sc Gaurav Chaudhary's Master thesis "Decoupling the thermal and visual performance in glazing systems: a novel methodology for the numerical investigation of the case of double-skin facade systems." - Politecnico di Torino, Italy, June 2019 [4]. Gaurav presented his research results on how the thermal and visual performance of DSF changes with boundary conditions and operational modes. The energy performance of an active DSF has been analyzed and applied to develop building control strategies and systems. It is meaningful to find a protocol of the control algorithm that applies to the autonomous control system of DSF for optimization of overall building energy performance. DSFs with integrated shading devices can be operated in various modes with their controllable components such as the shading slat angles, the airflow path, and the airflow rate in the air cavity. M.Sc Gaurav Chaudhary provided the approaches to understand how adjustments in different control components of a DSF affect its overall performance and how useful to control the DSF components in dynamic modes to utilize their maximum potential for building energy efficiency. The proposed control model can be applied to enhance the flexibility of DSF since operational modes can be automatically adjusted against variations of boundary conditions to maintain the required amount of thermal and visual gain.

The idea for this thesis work is to proceed with the implementation of real-time control strategies for autonomously operational modes of DSF such as the airflow path, the airflow rate, and the tilt angle of shading blinds according to data readings from the embedded sensors such as solar radiation, ambient indoor and outdoor temperature, or solar direction to DSF to gain the optimized thermal and visual performance. The outcome would be worth being applied for improving energy performance efficiency in building envelope systems with DSFs which are recognized as *"an integrated element of the building energy concept"* [5]. Such a control model could be possibly designed with the support of Machine Learning (ML) algorithms and made use for the further progression of the REINVENT research project in the practice of modeling a new control system of DSF which serves for the research purpose. However, it is greatly challenging and takes lots of dedicated work to design and apply an experimental plan for acquiring a sufficient quantity of realistic historical data for training the reliable ML models. Unfortunately, due to the severe influence of the COVID-19 pandemic, the research group couldn't access Torino University Laboratory where the experimental setup of the DSF is located in Italy, in the other words, no experiments could be carried out for the time being. Therefore, the thesis plan eventually came to be changed to work alternatively with simulation data only for studying the candidate ML models in preparation for the future applications and development of the research.

Despite the foreseen limitations, this thesis keeps aiming to exploit Gaurav's simulations and his findings for training Machine Learning (ML) models that predict the deliverable operational modes of DSF for the desired energy performance parameters under the known relevant environmental conditions. Although the control model was

designed with simulation data having not counted for the thermal inertial effect of variations in operational modes of DSF due to limitations of EneryPlus® building energy simulation tool as well as the time limit of the thesis, it could still serve to understand and preparing considerable approaches for collecting and training the historical data. Since developing an autonomous control system of a DSF with the application of ML as a building control strategy is challenging to carry out for the first time, a data acquisition (DAQ) plan for training ML models is also counted in this thesis author's efforts.

## 1.2 Research questions and objectives

This thesis focuses on giving the answers to the following two main research questions:

1. *"How can ML be applied to design the black-box control model of an active DSF?"* The question is also related to the concerns about the sufficient size of the training dataset to be collected.

2. *"What is reliability of the learned model of the DSF controller?"* This is targeted at the final outcome that the research group would like to figure out.

In order to find the solutions, the following objectives were determined for this thesis work:

1. Prepare and analyse correlations of the simulation datasets for training the ML models.

2. Define ML problems through which features and models will be chosen for training

3. Design and implement the pipeline of training the models including data processing, model tuning, visualization and evaluation of the results.

4. Define evaluation criteria for selection and application of the learned models.

5. Design the workflow of the DSF controller using the selected models.

6. Suppose possible methodologies for future development and applications of the designed control model.

## 1.3 Overview of research methodology

The thesis started with a literature review that was done by searching the electronic documents and resources from the internet, especially the Google Scholar search tool. Moreover, the Aalto library facilitates access to a database of selective good sources. The main keywords used for searching the relevant journals are, for example, *"double skin facade", "Machine Learning", "smart building", "building energy modeling and control solutions", and "black-box models"*. Besides, the empirical implementation of this thesis was conducted in two main phases. In the first phase, the steady-state

simulation data was ready at hand from Gaurav's thesis results for preparative data analysis and training the simplified ML models. In the later phase, the new big-size datasets of dynamic simulations were collected with the support from the research group using the building energy simulation software EnergyPlus® as also used for steady-state simulations. Dynamic simulations were reproduced with realistic boundary conditions of 4 different weather patterns in Cairo, Oslo, Rome, and Prague that vary hourly over a year. To design the controller for the operational modes of DSF with both steady-state and dynamic simulation data, the implementation steps of ML model training processes are summarized as follows.

- Step 1: Data preparation including data cleaning, correlation analysis, data visualization and data processing for training.
- Step 2: Define ML problems (regression or classification) with feature selection and possible hypothesis suggestions.
- Step 3: Model selection composed of spot checking, tuning and evaluating the good candidate models. The cross validation was used to compare the quality of prediction of the different models.
- Step 4: Design the control components for DSF operational modes with the selected models

In the practice of the project, for the purpose to make use of the trained models by the specific embedded controller of DSF such as Raspberry Pi (RPi) in the future experimental processes, data processing, and model training is executed in Python programming. The specific software library Scikit-learn was used to handle ML training tasks for medium-sized sets of simulations. The learned models are saved and later used by the RPi controller with ease and convenience. For dynamic simulation dataset in large size, the models were set up to be trained on multiple CPUs (central processing units) or GPUs (graphic processing units) on Aalto high-performance computing cluster named Triton, in which ANN models were trained with Keras - the library supported by Tensorflow backend for Deep Learning.

## 1.4 Structure of thesis report

The documentation of this thesis is organized as follows. Chapter 2 describes the research background of DSF in which an analytical review was concentrated on the effect of variations in boundary conditions and operational modes on the thermal and visual performance of DSF. Chapter 3 is a literature review on ML approaches for building energy modeling and control solutions. Chapter 4 presents the research methodology and materials for empirical implementation of this thesis including the design requirements for DSF controller, the preparation of simulation dataset, the data analysis for the ML problems to be solved, and the set-up for training processes. Chapter 5 demonstrates the implementation results and analyses the reliability of predictions as well as limitations of the application. Chapter 6 provides

suggestions for improving the model performance, the possible applications, and the future works of the project. Finally, conclusions on the achievements of this thesis are summarized in Chapter 7. All the Python scripts developed for the work are gathered in appendices.

# 2 Double Skin Facade

The research background of this thesis is studied primarily referring to M.Sc Gaurav Chaudhary's Master thesis report [4], especially to the simulation and the analysis of DSF thermal and visual performance against variations of boundary conditions and operational modes.

## 2.1 What is an active DSF?

Double-skin windows are quite popular and familiar in European buildings. Rather than general benefits such as noise reduction or fire projection, DSFs are preferable for supporting with heating/cooling and daylighting effectiveness that has a considerable impact on HVAC system. The clear definitions of DSF can be found in many technical and academic documents. The following are the favorite quotes for a good overview of DSF as an active envelope construction:

- DSF is *"a hybrid system made of an external glazed skin and the actual building façade, which constitutes the inner skin. The two layers are separated by an air cavity which has fixed or controllable inlets and outlets and may or may not incorporate fixed or controllable shading devices."* [6]
- *"In this kind of façade, the air cavity situated between the skins is naturally or mechanically ventilated. The air cavity ventilation strategy may vary with the time. Devices and systems are generally integrated in order to improve the indoor climate with active or passive techniques. Most of the time such systems are managed in semi-automatic way via control systems."* [4]
- *"Mechanically ventilated facades are usually strongly integrated with the HVAC system of the building (where the airflow is an imposed quantity set by the HVAC plant). In a naturally ventilated facade, the driving force for natural ventilation is either thermal buoyancy or wind pressure, or both. Therefore, the airflow is in this latter case not easy to control nor to predict, as it continuously changes depending on the weather conditions."* [5]

To utilize the effectiveness, DSFs can be designed as active facades that are integrated with the HVAC system of the building and equipped with ventilating fans, solar/thermal sensors, and shading devices. The air cavity of active DSFs should be mechanically ventilated or fans-supported. The direction and the velocity of the cavity airflow can be actively controlled by the fans. Figure 1 illustrates the possible airflow paths in the cavity and classification of DSF. The cavity airflow can be directed to control the amount of heat transfer between indoors and outdoors. The

reduction in heating loads can be achieved by using heat trapped in the air cavity, e.g. creating a barrier for heat loss (static air buffer mode) or supplying the trapped warmer air in the cavity into the indoor space (supply air mode or internal air curtain mode). The savings of cooling loads can be done by, for example, supplying fresh air from outdoors on cool days (supply air mode), or extracting the heat due to solar gain from the interior space on hot days (exhaust air mode), or cooling off the inner skin in external air curtain mode. Besides, the cavity airflow rate can be also adjusted to control the amount of heat gain/loss. Whereas, shading devices are mainly used to regulate the amount of visible light transmission with adjustable tilt angles of the shading blinds. Being placed inside the cavity, shading devices also help with protection and heat extraction during the cooling period. The shading devices and the cavity airflow are two components that can be theoretically controlled individually to control the thermal and visual gain independently. [4]

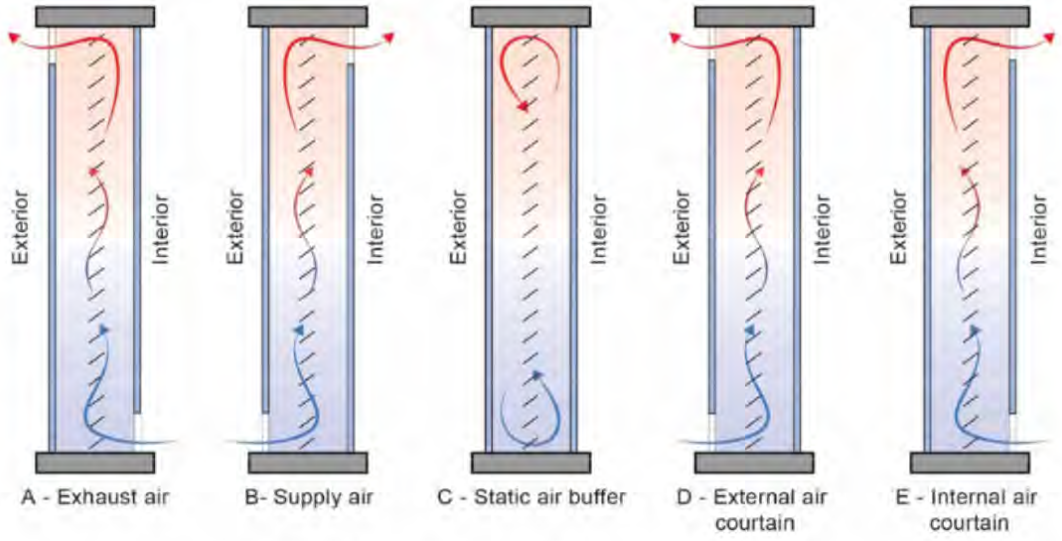



Figure 1: Possible airflow paths in DSFs (adapted from [5])

## 2.2 Thermal and visual performance of DSF

Figure 2 illustrates the thermal and visual performance of DSF that was quantified with thermal metric and visual metric. The thermal metric indicates the total heat gain/loss through the DSF. The visual metric gives the luminous flux into the indoor space through the interface of the DSF. The metrics can be normalized over the area

of the facade, which gives a unit of thermal metric as $W/m^2$ and a unit of visual metric as $Lm/m^2$. [4]

$$\textit{Thermal Metric} = \mathbf{Q_{sol,SW}} + \mathbf{Q_{sol,LW}} + \mathbf{Q_{conv}} + \mathbf{Q_{air}} + \mathbf{Q_{vent}} \tag{1}$$

$$\textit{Visual Metric} = \mathbf{KQ_{sol,SW}} \tag{2}$$

where,

- $Q_{sol,SW}$ = Directly Transmitted shortwave radiation
- $Q_{sol,LW}$ = Re-emitted longwave radiation from heat absorbed in glass
- $Q_{conv}$ = Convective heat exchange between glass and zone air
- $Q_{air}$ = Convective heat gain to the zone air due to the gap airflow
- $Q_{vent}$ = Extra energy to compensate for heating/cooling energy spent by HVAC
- Luminous Efficacy, K = 105 lm/W

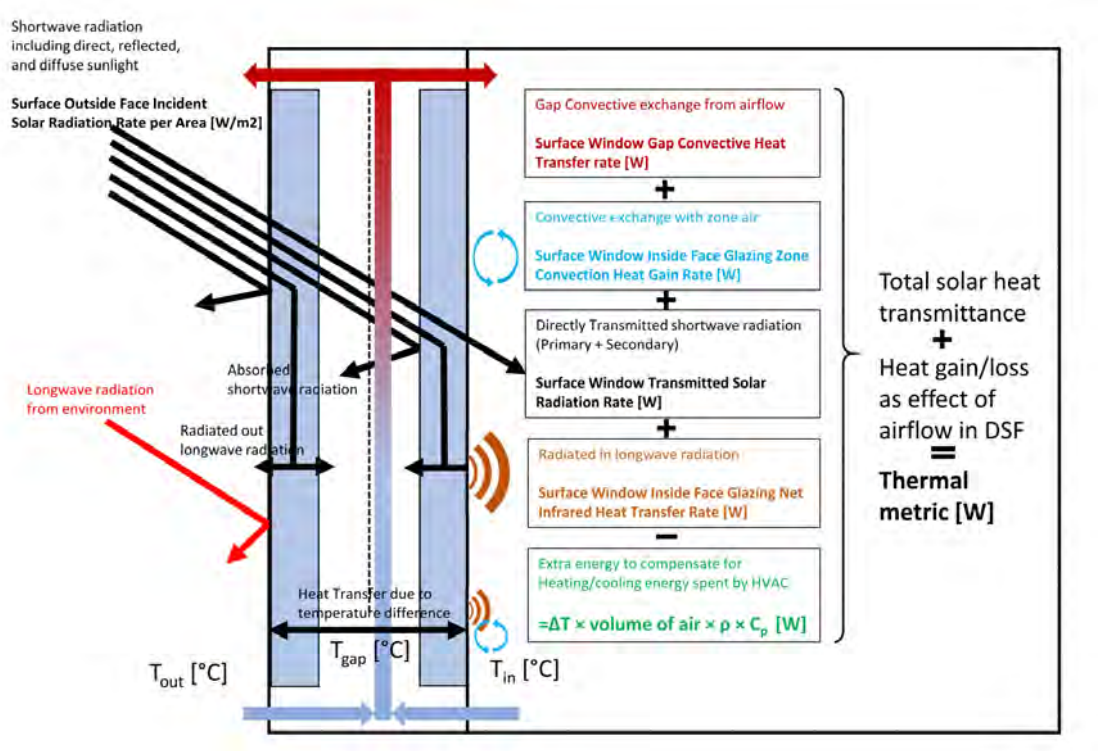



Figure 2: Heat flow schematic representation for DSF (reused from [4])

As can be seen from Equation 2, the visual metric depends only on the amount of visible light that is a fraction of solar radiation transmitted directly into the zone through the DSF. The sources of solar radiation include sky diffuse and direct

incident solar radiation that changes depending on the location where the DSF has situated as well as the climate conditions over the year. The amount of directly transmitted visible light through the DSF can be merely controlled by adjusting the different tilt angles of shading blinds and not affected by other control actions of the airflow path or the airspeed in the cavity. Meanwhile, Equation 7 confirms that the thermal metric depends on not only directly transmitted shortwave radiation but also other processes, i.e. invisible light transmission, convection due to temperature difference between glasses and zone air, or the gap airflow in the cavity as well as ventilation control activities by HVAC. Therefore, to control thermal metrics, combinations of all operational modes including different states of the blind angle, the airflow path, and the airflow speed in the cavity take different effects together with changes in temperature and solar radiation. For an intuitive example, when the outdoor temperature is higher than the indoor temperature, the thermal metric can be increased by setting the DSF in supply air mode in which the amount of heat gain is adjusted by the required amount of the cavity airspeed controlled by fans. The thermal metric is noted for 5 different airflow paths as follows [4].

- **Static Air Buffer**: There is no airflow in the closed cavity (airspeed = 0), no air exchange between the cavity and the zone ($Q_{vent} = 0$, $Q_{air} = 0$)

$$\textit{Thermal Metric} = \mathbf{Q_{sol,SW}} + \mathbf{Q_{sol,LW}} + \mathbf{Q_{conv}} \tag{3}$$

- **Supply Air**: $Q_{vent}$ is spent to bring the temperature of the amount of air entering the zone to the zone temperature, the same amount of air is then removed by HVAC system to maintain air balance.

$$\textit{Thermal Metric} = \mathbf{Q_{sol,SW}} + \mathbf{Q_{sol,LW}} + \mathbf{Q_{conv}} + \mathbf{Q_{air}} + \mathbf{Q_{vent}} \tag{4}$$

- **Exhaust Air**: The cavity air does not interact with inside air ($Q_{air} = 0$), $Q_{vent}$ compensates the heat transfer for bringing the temperature of the amount of air escaping the zone to the outdoor temperature, the same amount of air is then added by HVAC system to maintain air balance.

$$\textit{Thermal Metric} = \mathbf{Q_{sol,SW}} + \mathbf{Q_{sol,LW}} + \mathbf{Q_{conv}} + \mathbf{Q_{vent}} \tag{5}$$

- **Internal Air Curtain**: The cavity is ventilated by indoor air with no connection to the outdoor air ($Q_{vent} = 0$)

$$\textit{Thermal Metric} = \mathbf{Q_{sol,SW}} + \mathbf{Q_{sol,LW}} + \mathbf{Q_{conv}} + \mathbf{Q_{air}} \tag{6}$$

- **External Air Curtain**: The cavity is ventilated by outdoor air with no connection to the indoor air ($Q_{air} = 0$, $Q_{vent} = 0$)

$$\textit{Thermal Metric} = \mathbf{Q_{sol,SW}} + \mathbf{Q_{sol,LW}} + \mathbf{Q_{conv}} \tag{7}$$

Figure 3 describes how the thermal and visual performance range of DSF changes with varying boundary conditions, i.e. temperature difference (between indoors and outdoors) and outside incident solar radiation on the facade. The visual metric increases with a larger amount of solar radiation. Meanwhile, the thermal performance range decreases when the temperature difference goes down to zero.

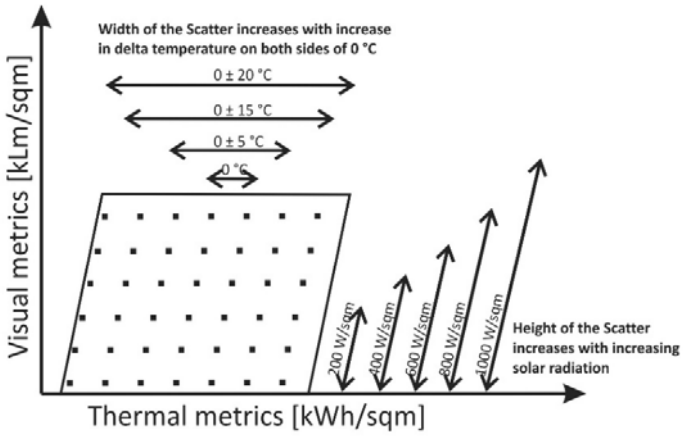



Figure 3: DSF performance range changes with varying boundary conditions (reused from [4])

The different effects of the different operational modes on DSF performance are visualized for illustration in Figure 4 with changes of the slat angle, the airflow path in the cavity, and the airflow speed respectively. Decreasing slat angles of shading blinds increasing both visual metric and thermal metric, except zero slat angle blocks visual gain completely. While the airflow paths do not give an effect on the visual metric, switching between the different airflow modes has a significant impact on the thermal range that is flexibly changing with large amplitude in exhaust and supply air modes but more narrow and less freedom in other modes. Airflow speed does also not affect Visual metric but the increasing airflow speed widens the Thermal range on both sides of zero. [4]

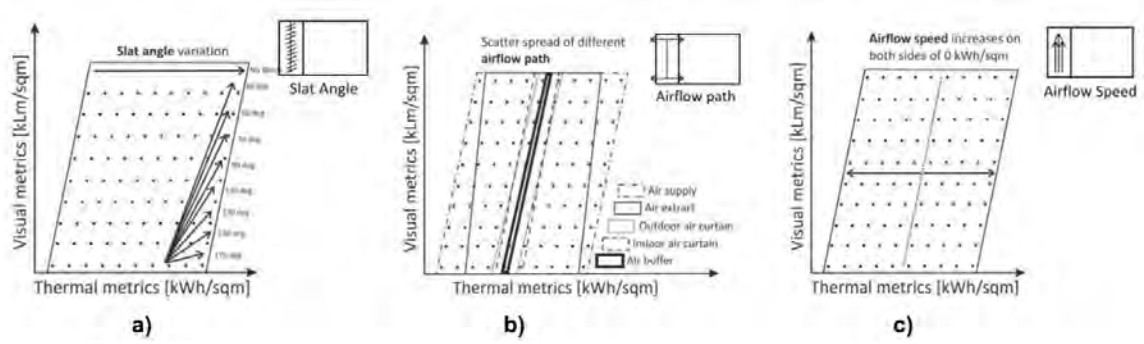



Figure 4: DSF performance is affected by changing a) slat angle of shading blinds, b) cavity airflow path, c) cavity airflow speed (collected from [4])

## 2.3 Advanced control strategies for DSF

From the analytical review of DSF performance, some methodology can be proposed for the advanced control strategies for DSF. A control scheme can be designed utilizing the knowledge learned for DSF thermal and visual performance. At the building

management system (BMS) level, an optimization process can be implemented for the optimal thermal and visual gains through the DSF based on the changes in environmental conditions that can be probably forecast. In turn, locally at the on-board real-time control level, the operational state of the DSF can be changed for the predicted thermal and visual metrics according to its senses to the ambient environment at the time of command from BMS. [4]. The system optimization and the local control process can be combined and addressed with the Model Predictive Control (MPC) approach [9] that requires lots of efforts and great work. However, MPC is unable to be implemented in this study due to unavailable data to proceed, and the work will have a broader scope to achieve than this thesis was required. Indeed, this work has concentrated on the implementation of an active controller of the DSF (see also requirements of DSF controller design in Section 4.1).

In comparison with other control algorithms designed by simulation tools like EnergyPlus, IDA ICE, or Matlab with a large degree of computational complexity and time consumption, implementing an onboard real-time DSF controller would require to use of a reduced order mathematical model of DSF energy performance so that it feasibly saves computational time and resources considering the minimum transition time between operational modes. The most favorable and powerful method known in many similar cases is Artificial Neural Network (ANN). The ANN-based numerical model can be adopted with simplicity to estimate the set of operation modes from the selected values of thermal and visual gain at given boundary conditions. To improve the prediction efficiency of the ANN model, a sufficiently large amount of samples (e.g. simulations) is required to relatively cover the whole domain of boundary conditions and operational mode combinations. Figure 5 demonstrates a possible approach suggested to apply ANN for the DSF control model in [4]. The structure, working principles, and more discussion in the practice of ANN are represented in Section 3.1.

However, depending on what kind of ML problems, the stochastic nature of ML algorithms, the characteristics of features, or statistic correlation between variables of interest, one ML algorithm may be able to perform better than the others on the same problem in practice. Learning a black-box model as ANN does, other outperforming ML algorithms that are known in much smart building (SB) applications are also worth being taken into account (Chapter 3). The training data set is essential to be able to resolve the minimum requirements of the data size and the sufficient diversity of the feature space. A too-large dataset implies a sign of massive consumption or waste of computing time and resources possibly resulting in impossibilities for the qualified outcome. Therefore, there is a prior need for careful data analysis and manual data pre-processing (Section 4.3) that helps to define the proper ML problems and evaluate the predictive modeling methods that can be employed for simplifying the tasks and the learning processes with the fast execution time and the satisfactory results. The target of this thesis is better to investigate an applicable ML approach to train the simplest models but still powerful for real-time processing.

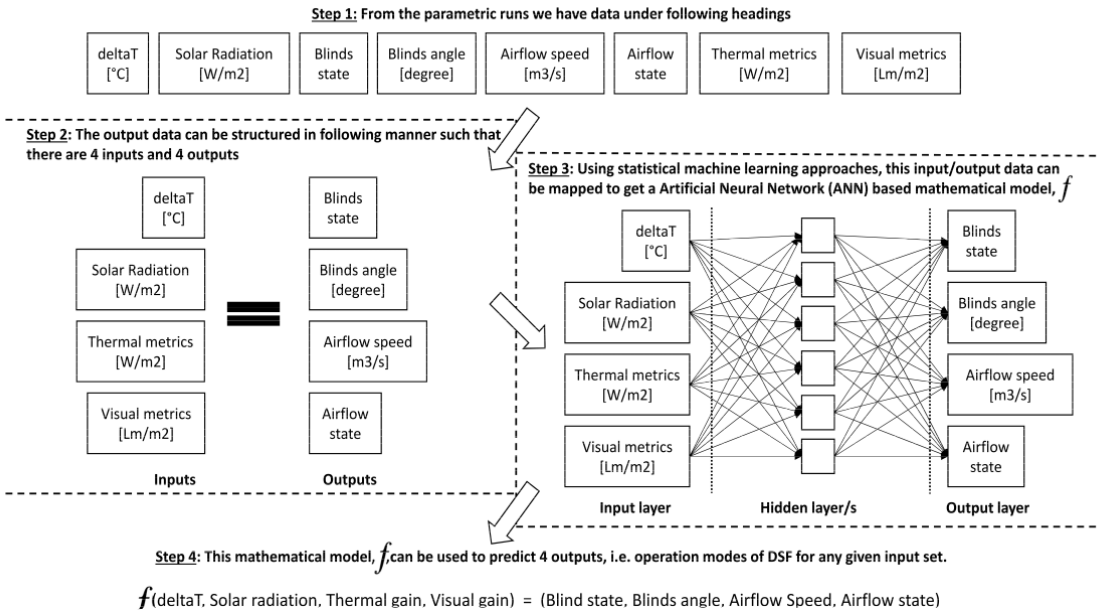



Figure 5: Workflow for predicting controls of DSF using ANN models (sourced from [4])

# 3 Applied Machine Learning for Building Energy Control Solutions

## 3.1 Building energy modeling and smart control applications

Predictive models as the representation of a system or a process such as building energy transfer have been analyzed and categorized as (1) physics-based/white-box models (interpretation of a physical structure with computational complexity), (2) data-driven/black-box models (no transparency of the physical relations between inputs and outputs), and (3) hybrid/gray-box models (combining both white- and black-box models) [8] [9]. An example of a white-box model is representing the building as a thermal resistance-capacitance (RC) network of connected nodes [9]. White-box models require less training data but computational complexity for practical implementation [9] such as in the case of real-time DSF control application. In contrast, black-box models are scalable numerical models using data-driven techniques that *"propose modeling and forecasting frameworks based on data analysis schemes"* [8], but lack of physical significance and possibly induce significant errors when operating outside the covering range of training data [9]. Among ML algorithms, ANN is commonly applied for black-box models that are developed based on empirical behaviors of the process and statistical methods for input-output relation without incorporation of prior physical knowledge. As a medium of white- and black-box models, gray-box modeling incorporates the physical processes with a

simplified system model and applies data-driven techniques to calculate the model parameters, which results in reducing model complexity and computational cost while still maintaining the physical significance of the parameters, and relatively well performance outside the training dataset. [9]

Among the aforementioned building energy modeling approaches, data-driven techniques applying recently achieved ML algorithms with simplicity and flexibility was considered most appropriate for bringing buildings in smart environments that rely on sensor networks collecting large quantities of energy-concerned data for efficient building energy management solutions [8]. ML is also applied for speedup of the parametric building energy simulations as studied in [10]. Many similar works are using neural networks in predicting the behavior of facades in a more reliable way than traditional methods such as Fourier theoretical model, for example, the prediction of thermal transmittance of windows, heat transfer coefficients, and moist porous materials [7]. In the research article *"modeling the thermal behavior of a building facade using deep learning"*, a thermal model of a building facade was successfully trained with deep learning for pretty good accuracy, using feed-forward time-delay neural networks (TDNN) that is a high-potential method for forecasting time series and other time-dependent problems [7]. For another example of ML application related to building energy management, M.Sc Joel Sarasti also applied ANN for compiling a numerical model of room thermal dynamics that approximates the room temperature that responses to the input heating power supplied [9]. There is a variety of smart building control applications using ML algorithms, for example, neural networks and reinforcement learning applied to control air heating, lighting, ventilation, and water heating in the smart home environment, or support vector machine (SVM) algorithm used to classify daily living activities based on the data from the different sensors in healthcare-focused smart home systems. [11].

Data-driven algorithms using ML were analyzed for the limitations of (1) relying on large quantities of data that must be representative of the different operating conditions of the building, and (2) only capturing specific patterns that lack generality. The mentioned problems can be addressed through more advanced or different ML techniques with unsupervised, reinforcement and transfer learning (1), and optimizing input data, using time-related parameters for example instead of physical variables (2). [8]

## 3.2 Machine learning approaches

### 3.2.1 Background

A simple and compact definition of ML is "the training of a model from data that generalizes a decision against a performance measure" - Jason Brownlee [12]. ML models are generally developed through a training process continuously on the input training data until the desired level of accuracy is reached [11]. ML is usually recalled and related in terms of "stochastic modeling" [13], "predictive modeling" [14], "data-driven techniques" or "black-box models" [8]. Generally, ML rely heavily on statistical algorithms [13] to apply for compiling numerical models [9]. Numerically

modeling of building energy performance have been using data-driven techniques that utilize ML algorithms to train black-box or gray-box models [8] [9] with advantages and limitations as contrasted previously in Section 3.1. Noteworthily, the concepts of white-, black- and gray-box models specified in ML may differ in use cases in comparison to the definitions of the model types named the same but related to the interpretability of underlying physical relations. ML models were categorized in [15] as follows:

- **White-box** is *"a model whose inner logic, workings and programming steps are transparent and therefore it's decision making process is interpretable"* [15]. For example, linear and monotonic models such as simple linear regression, decision trees, and Bayesian Networks are easy to explain as white-box models and more suitable for applications of e.g. medicine and finance with transparency in predictions required.

- **Black-box** is *"a more accurate ML model whose inner workings are not known and are hard to interpret"* [15] so that only the expected inputs and the corresponding outputs of this model could be seen. The example of black-box models are Artificial Neural Networks (ANN), Support Vector Machine (SVM), and ensemble methods such as Boosting and Random Forests that are often non-linear and non-monotonic models that fail to allow full understanding of their inner workings.

- **Gray-box** is an ensemble of black- and white-box models to exploit the benefits of both: black-box model's accuracy and the white-box model's intrinsic interpretability. The black-box model is firstly trained with a labeled dataset, and then the most confident predictions on an unlabeled dataset are collected to enlarge the dataset that is subsequently used for training the white-box model.

The sub-fields of ML that have been significantly developed over the past decades are namely Supervised learning , Unsupervised learning, Semi-supervised learning, Deep learning and Reinforcement learning with numerous applications in smart buildings (SB) were overall reviewed in [11]:

- **Supervised learning (SL)** is a traditional sub-field of ML that is extensively used to solve different problems in smart buildings due to its ability to generalize a learning model representation with the relations between the input, output, and system parameters [11]. The main goal of solving an SL problem is to learn from the labeled training dataset a mapping function from the input variables to the target. Based on the data variable type of the output, an SL problem can be categorized as *regression* for the real-valued target variable or *classification* for identifying the output variable of discrete values that can be labels or categories. [16]. SL is also selected as an application-focused approach of this study to apply in the empirical implementation. Regressive and categorical problems can be addressed with several common ML algorithms (Section 3.2.2) but different loss functions and metrics to measure performance and evaluate predictive skills (Section 3.2.3).

- **Unsupervised learning (USL)** of a model is applied when the labels (known outputs) for input data are unknown. Various USL algorithms including ones based on SL algorithms such as *k-Means* are used for *clustering* problems to classify the data by analyzing and grouping the input samples into clusters of similarity. Information extraction from the data by clustering exposes hidden correlations and allows to organize data by similarity. Other common USL problems are *dimensionality reduction* such as feature extraction using *principle component analysis* (PCA), and *association* rule learning. USL is probably a good solution to clustering problems of unlabeled data but might not learn any useful knowledge in the selected set of attributes. [11]

- **Semi-supervised learning (SSL)** is, as the name proposed, a combination of SL and USL when input data composed of labeled and unlabeled samples. SSL can address regression and classification problems while inheriting the strengths and mitigating the weaknesses of both SL and USL. Common SSL applications are, for example, label propagation using Gaussian Random Fields and Harmonic Functions [17], heuristic approaches, semi-supervised SVM, graph-based methods, self-training, and mixture models [11]

- **Deep learning (DL)** is a rapidly growing sub-field of ML that also covers areas of SL, USL, and SSL, and well-known with commonly used algorithms such as Artificial Neural Network (ANN), Convolutional Neural Network (CNN), and Recurrent Neural Network (RNN) that define a DL architecture of many hidden layers between input and output layers comprising multiple linear and non-linear transformations. Particularly, ANN algorithms such as Multi-layer Perceptron (MLP) or Radial Basis Function Network (RBFN) take advantages of less statistical training required and capability to detect complex nonlinear relationships but impose limitations of black-box nature, heavy computational burden, susceptibility to overfitting, the challenge in training with no local optima, complexity to interpret network workings, and exposure to data drift Generally, DL has gained successes in diverse applications of big data analytics, namely natural language processing, image identification, network security, medical diagnosis, and stock market trading trend prediction. DL is also promising to replace manual feature selection with efficient USL or SSL feature learning, and hierarchical feature extraction algorithms. [11]

- **Reinforcement learning (RL)** is a learning approach that takes care of optimization problems to control a system. A RL system so-called *agent* that repeatedly observes the *environment (x)*, performs an *action (a)*, and receives a *reward (r)* aims to maximize the future reward by continuously updating a control policy to select the actions from environmental feedback learned [17]. With no prior knowledge and a massive amount of training data, RL algorithms suffer the drawback of the high computational cost since all states need to be visited and checked to find the optimal solution. The well-known approaches of RL are Q-learning, Bruteforce, Monte Carlo methods, Temporal difference methods, and Value function. [11]

*Overfitting* is a common problem of ML techniques that requires evaluation on training, validation and testing data samples independently, and such constraint is often limited by manual *data pre-processing* [8], *regularization* [19], and *ensemble techniques* [18]. Overfitting is concerned with the model complexity that can be explained via the concept of *bias-variance trade-off* [16]. *Bias* is the difference in the model prediction and the actual value while *variance* is the difference in the predictions on the train and test data. Overfitting is the scenario that happens with high variance but low bias on the training data. To avoid overfitting, $l_1$ parameter regularization *(Lasso)* and $l_2$ parameter regularization*(weight decay)* [19] (see examples in Section 3.2.3) can be applied for regression handling numerical values , and *pruning* regularization is applied in cases of classification with Tree-based models [18].

### 3.2.2 Common ML algorithms

This section represents common ML algorithms usually applied in supervised learning and supervised deep learning (ANN) that were focused to study and apply in the empirical implementation of this thesis.

1. **Linear Regression (LR)** that is either single or multi-linear regression is the most simple and basic regression algorithm. LR fits a linear model with linear coefficients using ordinary least squares optimization to minimize the residual sum of squares between the labeled targets, and the predicted targets. The other brothers in the LR family are $l_1$−-regularized LR (LASSO), $l_2$-regularized LR (Ridge regression), and Elastic Net Regression (EN) that is LR applied with both $l_1$ and $l_2$-norm regularization of the coefficients. Regularization applied to LR will help sparse coefficients become more robust to collinearity. [18].

2. **Logistic Regression (LOG)** that is either binary or multi-class classification is the most simple and basic classification algorithm. Its name implies that its underlying technique being the same as LR is to train a linear model for classification, and the term "logistic" means *logistic* or *log function*, namely *sigmod function* that is used in this method of classification. LR is also known in the literature as logit regression, maximum-entropy classification or log-linear classifier. [18]

3. **Linear Discriminant Analysis (LDA)** is a classic classifier with a linear decision boundary by applying Bayes' rule to fit Gaussian density to each class, assuming that all classes share the same covariance matrix. LDA can be used to perform supervised dimension reduction by projecting the input data to the most discriminative directions maximizing the class separation. With easy-to-compute closed-form solutions, it has proven to work well with multiclass in practice without hyperparameters tuning required. [18]

4. **Classification and Regression Tree (CART)** is a widely used Decision Tree (DT) algorithm that is a non-parametric supervised learning method used for classification and regression. DT is well-known as a fundamental ML approach to build a model of decisions based on simple rules like if-then-else

from the data features. DT is structured by *leaves* representing class labels and *branches* representing conjunctions of attributes driving to those labels. The deeper the tree, the more complex the decision rules, and the fitter the model. Some advantages of DT are: (1) being interpretable and visualizable, (2) requiring little data preparation, (3) inference cost being logarithmic in the number of training samples, and (4) favorite for its speed and accuracy with the ability to handle both numerical and categorical data or multi-output problems. In contrast, DT imposes the following disadvantages: (1) probable over-complexity that can be constrained by *pruning* mechanism, (2) instability due to small variations in the data possibly generating a completely different tree but possibly mitigated by applying ensemble method, (3) practical DT algorithms based on heuristic algorithms such as the greedy algorithm making locally optimal decisions at each node but may not returning the globally optimal DT where ensemble method takes advantages again, (4) hard learning for XOR, parity or multiplexer problems, (5) creating biased trees if some classes dominate (imbalanced learning). The other DT algorithms such as ID3, C4.5 and C5.0, Chi-squared, M5, and conditional DTs are also most commonly used. [11] [18]

5. **Naive Bayes (NB)** a set of supervised learning algorithms for classification including Gaussian NB, Multinomial NB, Complement NB, Bernoulli NB, and Categorical NB based on applying Bayes' theorem. It assumes the conditional independence between every pair of features given the labeled class. NB can be used to tackle large-scale classification problems where there might not be enough memory for the full training set. Despite apparently "naive" i.e. over-simplified assumptions, NB algorithms are well done in many real-world applications, famously document classification and spam filtering. [18]

6. **k Nearest Neighbors (kNN)** is an ML algorithm for classification, clustering, and regression by calculating the similarity of two samples in terms of the metric distance between these samples in the feature space. The samples being close to each other are similar and grouped. Different distance calculation methods such as the Euclidean, absolute, and Minkowski metrics are in common use. [10]

7. **Support Vector Machine (SVM)** is a non-parametric regression method for the non-linear classification, regression, and outlier detection. Support vectors are defined as the closest points to the optimal hyperplane which separates two classes of training data. The goal is to find a kernel function minimizing the deviation from the outputs of training data. Alternative kernel functions are Gaussian, radial basis function (RBF) that are used to calculate the dot product of two vector variables or arguments in the design input space. The method of Support Vector Classification (SVC) can be extended to solve regression problems which are known as Support Vector Regression (SVR). [10]. Popular applications of SVM can be found for many statistical learning problems such as face and object recognition, text classification, and handwriting analysis.

[11]

8. **Random Forest (RF)** belonging to the family of Forest of Randomized Trees is one of the most popular and powerful ML algorithms for both classification and regression problems. RF not only applies a simple but very powerful *ensemble* method called *bootstrap aggregation* (*bagging*) that combines the predictions of multiple bagged greedy DTs and take average prediction for improving robustness over a single DT but also adapts such a way that the resulting predictions from all sub-trees are less correlated by preventing DTs from being greedy but limited to search through a *random* sample of features for selecting the most optimal split-point. The principle of bagging is to build several estimators independently and then to average their predictions. Bagging procedure is generally applied to reduce the variance in predictions of the high-variance algorithms such as CART and thereby efficiently overcome overfitting. [21] [18]. As a bagging method, RF classifies a new object based on attributes each tree in the forest (ensemble) gives so that classification is done by majority voting from the forest or regression is made by averaging predictions from DTs. RF is extremely flexible and well-known for its ability to accomplish high accuracy results, even without hyperparameter tuning, and to handle large data sets with higher dimensionality. The major disadvantage of RF is its complexity which makes it less intuitive, much harder, and much time-consuming to be built in comparison with DT. [22]

9. **Extra Trees (ET)** is shortly named for Extremely Randomized Trees. ET is another efficient *bagging* ensemble algorithm growing a large number of unpruned DTs from the training dataset as RF but adding extra randomness for more robustness. In comparison with RF, ET also randomly sample the features at each split-point but unlike RF using a greedy algorithm for an optimal split-point, ET selects a random split-point and fits each DT not from a bootstrap training sample but on the whole training dataset. There are three main hyperparameters to tune for ET that are the number of trees used, the number of features for random selection at each split-point, and the minimum number of samples in a node to create a new split-point. The randomness in split-point selection makes DTs in the ET ensemble less correlated and the increasing number of trees can reduce the variance of the algorithm.[23]

10. **AdaBoost (AB)**, fully Adaptive Boosting, is a popular *boosting* ensemble algorithm in which base estimators are sequentially built and trained using the weighted training data. Each estimator attempts to reduce the bias of the earlier combined one until perfectly fitting the entire training set or the final estimator is reached. Any ML algorithm can also be boosted to improve the performance with AB but weak learners prove the best to take the power of AdaBoost, for example, small DTs have been proven to work with AB best on binary classification problems. [24] [18]

11. **Gradient Boosting Machine (GBM)** that is also known as *gradient tree boosting* or *stochastic gradient boosting* is a powerful *boosting* ensemble algorithm

for both classification and regression using gradient boosted DTs. In addition to the *boosting* method applied as in AB, each model in the ensemble of GBM is fit using any arbitrary differentiable loss function and gradient descent optimization algorithm. *"gradient boosting"* means minimizing the loss gradient through training as in ANN. GBM is one of the main algorithms used to win ML competitions (like Kaggle) on tabular and similar structured datasets. [25] [18]

12. **XGBoost (XGB)** is short for *"Extreme Gradient Boosting"* that is a popular and increasingly dominating ML algorithm based on an ensemble of gradient boosted DTs [26]. XGB software library is designed for speed and performance with many support features that are to *"push the extreme of the computation limits of machines to provide a scalable, portable and accurate library"* [27]. XGB optimizes implementation of the *gradient boosting* algorithm for applied ML with the main benefits of high computational efficiency and often outperforming. While the gradient descent in GBM is the first-order optimization algorithm, XGB as a variant of GBM adds regularization to the objective [29]. Like GMB, XGB has recently been dominating applied ML and Kaggle competitions for structured or tabular data. [30]. Although XGB has been proven its outperformance in many applications in practice, it still has the common limitation of all Tree-based models being *"unable to extrapolate target values beyond the limits of the training data when making predictions "* [28].

13. **Multi-layer Preceptron (MLP)** is a classic approach and most useful type of ANN that dates back to the years of 1940s. MLP is commonly utilized to build black-box models for regression and classification problems. MLP has a hierarchical network structure of multiple layers of *neurons* including one or more *hidden layers* between the input and output layers. The neurons are only connected with the ones in the next layer on the way of computational propagation, without interacting with the others in the same layer. Each *neuron (perceptron)* is a computational unit that weights input signals coming from other neurons and uses an *activation function* to produce an output signal towards the neurons in the next layer. Two classical and widely used nonlinear activation functions are the *sigmoid (logistic)* and *hyperbolic tangent* activation functions with limitations of saturation and being not easy to train on evolved GPU platforms, hence the Rectified Linear Unit (ReLU) activation function shows advantages over the others [31]. Among stochastic optimization methods used in deep neural networks including Stochastic Gradient Descent (SGD), Adam, or L-BFGS, empirical results have shown that Adam works well in practice compared favorably to others [32]. The computational process starts with one training example exposed to the network at a time from the input layer to the out layer until the whole training data are computed for the final output is called *forward propagation* on the network *(feedforward neural network)*. A reverse process called *backpropagation* algorithm when the error between the network output and the expected output is calculated and then propagated back through the network and the weights are updated to reduce

the error. One round of updating the network for the entire training dataset is called an *epoch*. The process in which the weights are updated from the errors calculated for each training example is called *online learning* that is done fast but also chaotic. Alternatively, *batch learning* is often more stable by saving up the errors across all the training examples and updating the network at the end. The amount updated to weights for a given error is controlled by *learning rate*. [33] [18]. Like any other deep neural network (DNN), MLP suffers from overfitting that can be reduced using an ensemble of DNNs with different model configurations. However, instead of ensemble learning that requires the additional computational expense of training and maintaining multiple models, a single DNN model can be applied regularized by randomly dropping out nodes during training (*dropout*) with remarkably computational cheapness and effectiveness. [34]

### 3.2.3 Metrics

Statistical metrics used in this study for data processing and representing results include the mean/average (Equation 8) and the standard deviation (Equation 9) of the samples, Pearson's correlation coefficient (Equation 10), and standardization (Equation 12).

$$mean(\mathbf{y}) = \frac{1}{n}\sum_{i=1}^{n}\mathbf{y_i} \tag{8}$$

$$std(\mathbf{y}) = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{y_i} - mean)^2} \tag{9}$$

Pearson's correlation coefficient is used for computing pairwise correlation of the variables of interest given by Equation 10. The *"corr"* function from Pandas is used to calculate the correlation (see also Section 4.4.2):

$$corr(\mathbf{X}, \mathbf{Y}) = \frac{cov(\mathbf{X}, \mathbf{Y})}{std(\mathbf{X}) * std(\mathbf{Y})} \tag{10}$$

where $cov(\mathbf{X}, \mathbf{Y})$ is the sample covariance:

$$cov(\mathbf{X}, \mathbf{Y}) = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{X_i} - mean(\mathbf{X}))(\mathbf{Y_i} - mean(\mathbf{Y})) \tag{11}$$

The standard score from scikit-learn [18] calculates standardization of a sample as follows:

$$Z(\mathbf{y_i}) = \frac{\mathbf{y_i} - mean(\mathbf{y})}{std(\mathbf{y})} \tag{12}$$

Performance metrics and scoring for regression include mean square error (MSE) or negative mean square error (NMSE), mean absolute error (MAE) and coefficient of determination ($R^2$). For classification, *softmax function* and *accuracy* are used for evaluating performance and predictive quality. The error in regressive predictions (residual) used and named *"error"* coherently in this document is defined as:

$$error = \mathbf{y_{estimate}} - \mathbf{y_{actual}} \tag{13}$$

MSE is a metric used for evaluating regression performance as formulated in Equation 14.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{y_{estimate_i}} - \mathbf{y_{actual_i}})^2 \tag{14}$$

NMSE simply negate MSE (Equation 15) and can be applied as scoring for example in case of scikit-learn grid search of the tuned hyperparameters of the training model with cross validation. Since unified scoring application programming interface (API) always maximize the score, NMSE should be used instead of MSE that needs to be minimized for e.g. linear regression so that the scoring API works correctly. [18]

$$NMSE = -MSE \tag{15}$$

MAE given by Equation 16 is to calculate the average absolute difference between the predictions and the actual values.

$$MAE = \frac{\sum_{i=1}^{n} \mid \mathbf{y_{estimate_i}} - \mathbf{y_{actual_i}} \mid}{n} \tag{16}$$

Coefficient of determination ($R^2$) is another metric for accessing the accuracy and efficiency of the regression models in this study (see Equation 17) [18].

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(\mathbf{y_{actual_i}} - \mathbf{y_{estimate_i}})^2}{\sum_{i=1}^{n}(\mathbf{y_{actual_i}} - mean(\mathbf{y_{actual}}))^2} \tag{17}$$

For classification, the accuracy score metric is used to calculate the frequency at which predictions match the labels as represented in Equation 18 [18].

$$accuracy(\mathbf{y_{actual}}, \mathbf{y_{estimate}}) = \frac{1}{n}\sum_{i=1}^{n} 1(\mathbf{y_{estimate_i}} = \mathbf{y_{actual_i}}) \tag{18}$$

For multi-class classification, for example using MLP and XGB models, softmax function is often used to estimate the probabilities that a sample belongs to each class [18].

$$softmax(\mathbf{y})_{\mathbf{y}} = \frac{\mathbf{exp(y_i)}}{\sum_{j=1}^{k}\mathbf{exp(y_j)}} \tag{19}$$

Regression attempts to minimize MSE during training. For example, MLP uses the Square Error loss function applied with l2-regularization as in Equation 20 [18].

$$Loss(\mathbf{y_{estimate}}, \mathbf{y_{actual}}, \mathbf{W}) = \frac{1}{2}\left\|\mathbf{y_{estimate}} - \mathbf{y_{actual}}\right\|_2^2 + \frac{\alpha}{2}\left\|\mathbf{W}\right\|_2^2 \tag{20}$$

where $\frac{\alpha}{2}\left\|\mathbf{W}\right\|_2^2$ is an L2-regularization term to penalize complex models, and $\alpha > 0$ is a non-negative hyperparameter to control the magnitude of the penalty. Otherwise, multi-classification often uses Entropy (RF [35]) or Cross-Entropy (MLP [18], XGB [27]) as loss function to optimize the predictions.

### 3.2.4 Speedup of learning

Learning can be slow or fast for different ML models depending on the nature of the statistical algorithms used. Tuning for good hyperparameters and model parameters is not only to improve the accuracy but also reduce the complexity of the model that can be trained particularly with different learning rates being able to affect the training time that however majorly depends on the size of the training dataset.

To handle a large amount of training data, the best way to accelerate the data processing and learning process is to use parallel computing with multiprocessing on multiple CPU cores or GPU computing. Parallel computing requires powerful processing units of multi-cores, especially Nvidia GPU(s) working with compatible versions of CUDA for deep learning frameworks like Tensorflow or GPU supported models such as XGBoost. There are available powerful computing resources such as computing cluster or cloud computing. On the cloud or the cluster, the application is run in parallel nodes each of which is allocated adequate CPU(s)/GPU(s), memory, storage, and network capacity as required for the task. For example, CSC cluster [1] offers researchers free access to a high-performance computing platform, or cloud computing services widely support public or commercial use such as Amazon Web Services (AWS) [2] or Google Cloud TPU (tensor processing units) [3].

# 4 Research methodology and materials

As the overview of research methodology and materials was represented in Section 1.3, the thesis started with reviewing related works to prepare the methods for implementation. Google Scholar and Aalto library well facilitates the searching tools for the research work. The main keywords used for searching the relevant journals are, for example, *"double skin facade", "Machine Learning", "smart building", "building energy modeling and control solutions", and "black-box models"*. For the empirical implementation of this thesis, firstly the design requirements of the DSF controller were discussed by the research group and determined as the main objective. Hence, the specific configuration of DSF was also limited for the objective, and simulation data was decided to use for model training. The research group support with the materials of both steady-state and dynamic simulations using the building energy simulation software EnergyPlus®. Data processing and model training requires Python programming for the practice of model inference on the specific embedded (RPi) controller of DSF. Next, data analysis was needed for ML problem statements, followed by the work of preparation for the training environment and processes to produce the target predictive models. The specific software library Scikit-learn was used to handle ML training tasks for medium-sized sets of simulations. For dynamic simulation dataset in large size, the models were set up to be trained on multiple CPUs (central processing units) or GPUs (graphic processing units) on Aalto high-performance computing cluster named Triton, in which ANN models were trained with Keras - the library supported by Tensorflow backend for Deep Learning.

## 4.1 Design requirements

The minimum design requirements are illustrated with the concepts of the simplified black-box control model of DSF in Figure 6. The local controller and the sensors

[1] https://research.csc.fi/csc-s-servers
[2] https://aws.amazon.com/
[3] https://cloud.google.com/tpu

are embedded in the DSF. The local controller is the target device that employs the ML models. It receives the input signals from the embedded sensors and the central controller. The central controller can be a room or building level controller that should determine the optimal thermal metric `T_opt` and the optimal visual metric `V_opt` based on some optimization processes that can be applied with the MPC approach (see Section 2.3) but out of this thesis scope. The embedded sensors take the role of collecting boundary conditions, i.e. the indoor (`T_in`) and outdoor (`T_out`) temperature as well as solar radiation (`SR`) and feed them to the local controller that itself calculates the temperature difference `DeltaT = T_out - T_in`. The local controller is in turn loaded the trained models to compute the predicted values of thermal and visual metrics `T_pred, V_pred` from the inputs `DeltaT, SR, T_opt, V_opt` and correspondingly return the outputs for operational modes of the DSF, namely `Blinds_ctrl, Airflow_ctrl,` and `Speed_ctrl` that are subsequently translated by some logic as the real control signals to drive the tilt angle of the built-in shading blinds, the state of airflow path, and the fan speed that sets the airflow speed respectively. The predicted values `T_pred` and `V_pred` should be the best closed values to `T_opt and V_opt` in the limit that the boundary conditions `DeltaT` and `SR` do allow. The used ML algorithms can be evaluated for some requirements/specifications in the practice of real-time processing of the DSF controller.

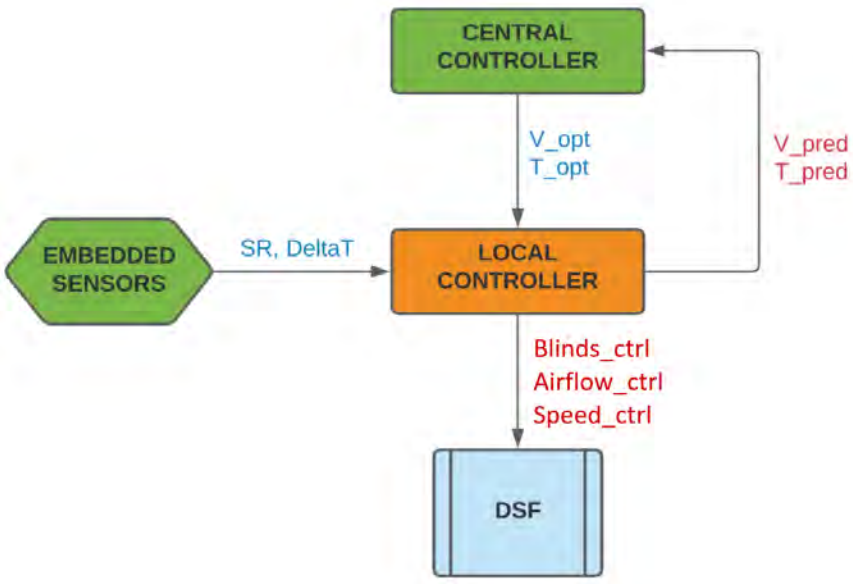



Figure 6: System control block diagram

## 4.2 DSF configuration and simulation settings

Figure 7 demonstrates a physical mock-up of DSF that is set up for experiments in the laboratory of Politecnico di Torino University, Italy. The DSF is instrumented with a Raspberry Pi 4 (RPi4) microcontroller for controlling the airflow path, the airflow rate through the fan's speed, and the activation/position of the shading.

There is a separate data acquisition (DAQ) system which is based on the National Instrument (NI) SbRio plus DAC with ModBus and all controlled through a LabView code. The RPi4 controller has the role to acquire data from the embedded sensors and to control the state of the facade according to different implementation strategies. The NI controller has the role to acquire all the data from around 50-60 laboratory sensors that are not embedded on the facade for recording the physical quantities giving the performance of the facade, for example, temperature values, heat flux values, irradiance values, air velocity values, The thermal and visual metrics can be calculated based on the quantities that are real data registered by the DAQ system and therefore containing the effect of the history on the performance.

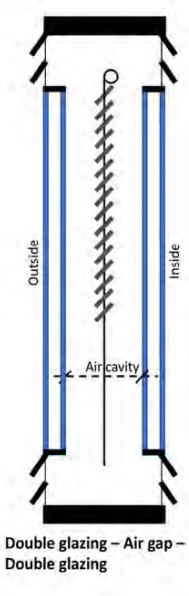



Figure 7: DSF configuration of the scale 1:1 (retrieved from [4])

To study for future applications and experiments, the ML models were trained with the simulations in both steady-state and dynamic cases that were set up close to the aforementioned specific configuration of DSF in experimental setup: double layers of glazing, 200mm gap width of the air cavity, and high reflectivity of blinds. The simulations were reproduced and collected with support from the research group. All Python scripts and simulation processes with EnergyPlus® building energy simulation tool can be retrieved from Gaurav's thesis [4]. Noteworthily, insightful analysis on the simulation speed of EnergyPlus was discussed in [10] that each energy simulation for a simple five-zone model takes around 150 seconds to complete on a regular personal computer, and therefore $10^5$ simulations for example require approximately 1666 CPU hours.

By steady-state simulations, the boundary conditions are kept the same for an extended period that allows an equilibrium state of dynamic systems, and only instant effect of changes on thermal and visual gain [4]. The steady-state simulation data contain the combined sets of the discrete values of the following variables as the simulation input settings in EnergyPlus. These simple cases help to analyze

the simplified correlation of the main features and get the intuition to define ML problems at the first stage.

- The temperature difference - `DeltaT` [$^o$C]: -45, -40, ..., 15, 20
- The incident sky diffuse solar radiation rate - `SR` [W/m$^2$]: 0, 1000, ..., 9000, 10000
- The blinds angle - `Blinds` [degree]: 0, 5, ..., 170, 175
- The cavity airflow rate - `Speed` [m$^3$/s.m]: 0.0, 0.0028, 0.0056, 0.0111, 0.0222, 0.0444, 0.0889, 0.1778 that simulate the scaled values of the fan rate in real-time control.
- The airflow path that was set by the inlet-outlet direction of the cavity airflow that is denoted as - `Airflow`: 'buffer mode', 'exterior air curtain', 'extract mode', 'interior air curtain', 'supply mode'.

The dynamic simulation data contain the combined sets of the discrete values of the controlled operational modes and the continuous values of the boundary conditions that change hourly over a year. For diversity in boundary conditions, 4 different weather patterns were combined in Cairo, Oslo, Prague, and Rome. The dynamic simulations were intended to investigate more features such as different kinds of incident solar radiation (direct or diffused) and solar position with respect to the DSF that impacts the solar radiation and energy performance of DSF, as well as look closer to challenges of the training data size, should be in real time.

- The temperature difference - `DeltaT` [$^o$C]
- The incident sky diffuse solar radiation rate - `SR_diffuse` [W/m$^2$]
- The incident direct solar radiation rate - `SR_direct` [W/m$^2$]
- The azimuth angle of the sun - `Azimuth` [degree]
- The altitude angle of the sun - `Altitude` [degree]
- The blinds angle - `Blinds` [degree]: 0, 10, ..., 80, 90
- The cavity airflow rate - `Speed` [m$^3$/s.m]: 0.0, 0.0028, 0.0056, 0.0111, 0.0222, 0.0444, 0.0889, 0.1778 that again simulate the scaled values of the fan rate in real-time control.
- The airflow path that was set by the inlet-outlet direction of the cavity airflow that is denoted as `Airflow`: 'Buffer mode', 'Exterior air curtain', 'Extract mode', 'Interior air curtain', 'Supply mode'.

The observed outputs in both cases are Visual metric and Thermal metric that are coherently denoted through the documentation as `Visual` [Lm/m$^2$] and `Thermal` [W/m$^2$] respectively.

## 4.3 Data analysis and problem definition

In this section, the datasets are analyzed to investigate and identify ML problems to be learned. This phase takes a critical role in simplifying the ML problems and selecting the alternative learning models properly, which means a lot to save time and effort for the effectiveness of our work. The following steps generalize the workflow through this phase.

1. **Data cleaning:** The data set was prepared as the first step required for any ML problem. Data cleaning was done by removing the fault data points, duplicates and missing values in the data set. The steady-state simulation data contain the fault values at solar radiation of 1000 that was removed for model training in the later implementation phase. Faults and outliers could be detected with aid of data visualization, for example, Figure 10 and other scatter plots of steady-state data shows the fault values at solar radiation of 1000 that is inconsistent with the rest of the dataset. Meanwhile, the dynamic data set has a size of 1.3 GB that is about 100 times larger than the size of the steady-state data set after cleaning up. Data cleaning was redone several times in every step of the data analyzing phase whenever there was a need to split the data set into smaller subsets of training data. For instance, feature extraction for modeling visual metrics possibly requires fewer features than for the thermal metric, and therefore, duplicates could be removed after deleting uninterested columns. Meanwhile, it is also helpful to handle the large data size of the dynamic data set by separating subsets of different airflow modes. After splitting the datasets into smaller subsets and applying data cleaning, the number of cases was counted for each cleaned dataset and listed in Table 1.

Table 1: Number of cases of the cleaned datasets

| **Data Set Name** | **Category** | **No. of Instances** | | **No. of Attributes** | |
|---|---|---|---|---|---|
| | | **steady-state** | **dynamic** | **steady-state** | **dynamic** |
| Airflow path | Classification | 160,155 | 8,067,960 | 5 | 8 |
| Visual metric | Regression | 396 | 350,555 | 3 | 6 |
| Thermal metric, Air Buffer | Regression | 5,544 | 350,400 | 5 | 8 |
| Thermal metric, Interior Air Curtain | Regression | 38,808 | 2,102,400 | 6 | 9 |
| Thermal metric, Exterior Air Curtain | Regression | 38,494 | 1,769,520 | 6 | 9 |
| Thermal metric, Exhaust Air | Regression | 38,808 | 2,102,400 | 6 | 9 |
| Thermal metric, Supply Air | Regression | 38,501 | 1,743,240 | 6 | 9 |

2. **Data description and visualization:** The used data sets are described with statistical summary, Pearson correlation, and visualized for detecting how

Visual metric and Thermal metric separately change depending on each of boundary conditions and operational modes. Subsequently, all information was collected and analyzed for identifying ML problems as well as evaluating feature importance for each predictive modeling problem.

3. **Evaluate features and identify ML problems:** It is vital to define the proper ML problems and evaluate features for each predictive modeling problem since they suggest the ideas for designing the working control model of DSF. On the other hand, the main challenge in designing the real-time control model of DSF is to collect a sufficient amount of data for modeling with ML. Training processes consume lots of time for doing the duty with datasets of large size. Started with the author's modest knowledge and experience in ML applications, the model training sometimes took place forever but met no end. Meanwhile, proper feature and model selection can save lots of work. This step reasons selecting alternative ML models that simplify the ML regression or classification problems as well as originate the ideas for designing the control workflow of DSF and the implementation of training predictive models that are presented in detail in Section 4.4.

### 4.3.1 Pearson Correlations

The metric used for feature evaluation is the Pearson correlation coefficient of each pair of variables in the dataset that was calculated and displayed in Figure 8. Overall, the visual metric mostly depends on the solar irradiance both directly or indirectly, the blinds angle, and the solar altitude angle (Figure 8.b). With a steady-state simulation dataset, the model was simplified to learn the relationship of the visual metric with the (diffuse) solar irradiance and the blinds angle first (Figure 8.a). Meanwhile, the cavity airflow state and speed contributes almost a zero effect on the visual performance of DSF. In contrast, the outdoor-indoor temperature difference, the direction, and the velocity of the airflow in the cavity dominate the thermal performance of DSF that results from multiple effects.

The scatter plots in Figure 9 visualize the correlations and illustrates clearly the different main factors that affect the visual metric and the thermal metric. Higher solar irradiance and wider open blinds angle from 0 to 90 degrees increase the amount of daylight. Meanwhile, the larger temperature difference and the higher airflow speed widen the thermal performance range of DSF. Supply air mode or exhaust air mode reserves a large thermal performance range that is limited in other modes, noted that airflow modes are encoded as 0 - air buffer mode, 1 - exterior air curtain, 2 - exhaust air mode, 3 - interior air curtain, and 4 - supply air mode. Since the thermal performance is different depending on the specific airflow mode in which DSF operates, it is good practice to analyze in reduced size of data for the correlations of the thermal metric vs the other factors in each single airflow mode.

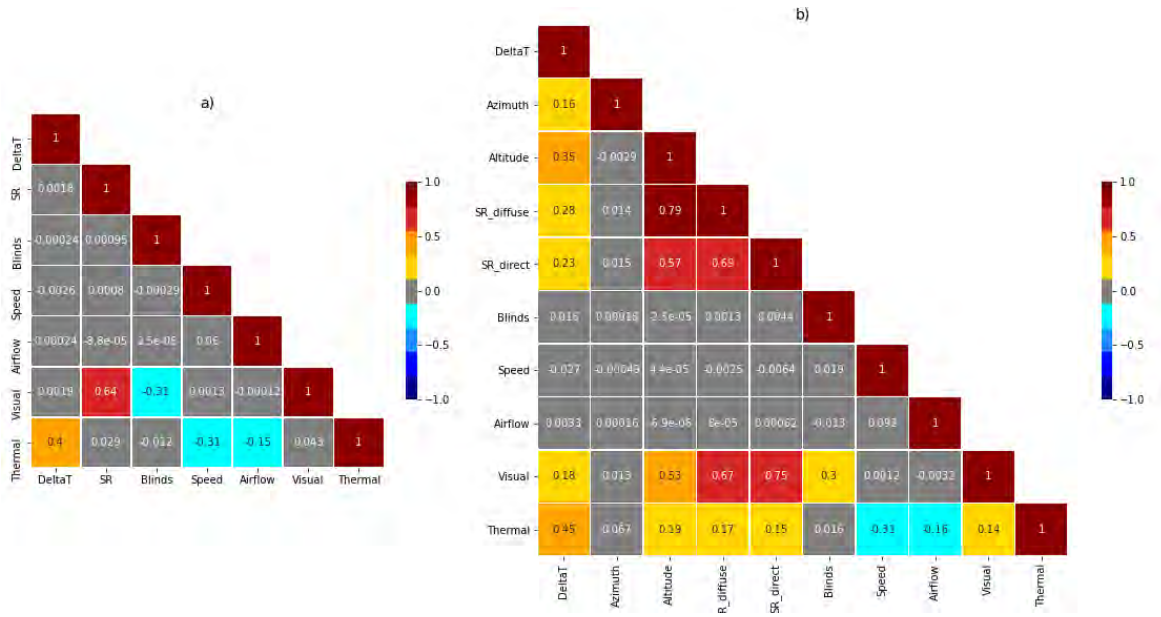



Figure 8: Pearson correlations of a) steady-state simulation dataset; b) dynamic simulation dataset

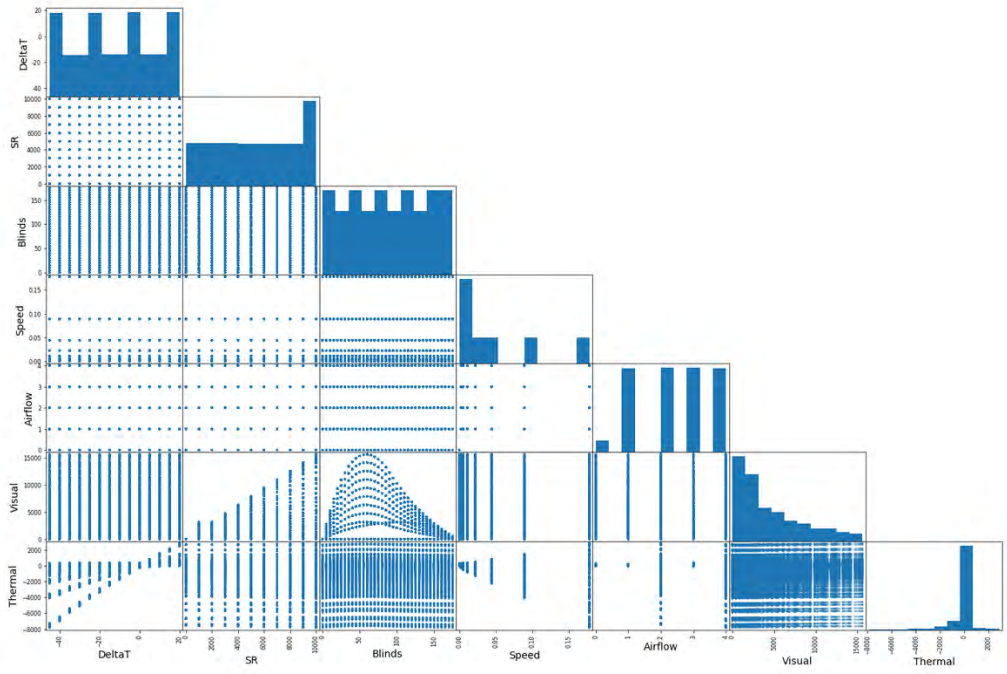



Figure 9: Scatter matrix plot of steady-state simulation dataset

### 4.3.2 Steady-state simulation dataset

Considering only the visual-driven factors, the steady-state simulation dataset for Visual metric was significantly reduced in size after data cleaning of duplicates which confirms the visual metric is independent of other features like airflow state (see Table

1). As can be seen from Figure 10, the visual metric reaches its peaks at the higher rate of solar radiation. There is an abnormal behavior of the steady-state simulation at solar radiation rate of 1000 W/m$^2$ due to a certain reason in the process of data reproduction but out of concerns. The default values were removed and treated as checking points for later result evaluation of the learned model.

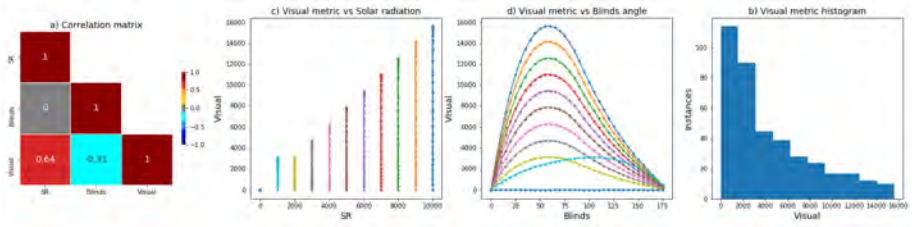



Figure 10: DSF visual performance (steady-state simulation)

Next, the thermal performance will be observed in different modes. In static air buffer mode (Figure 11), the airflow speed is set to zero and the thermal behavior is highly associated with the visual performance, i.e greatly depends on the solar radiation and the blinds angle as much as the delta temperature.

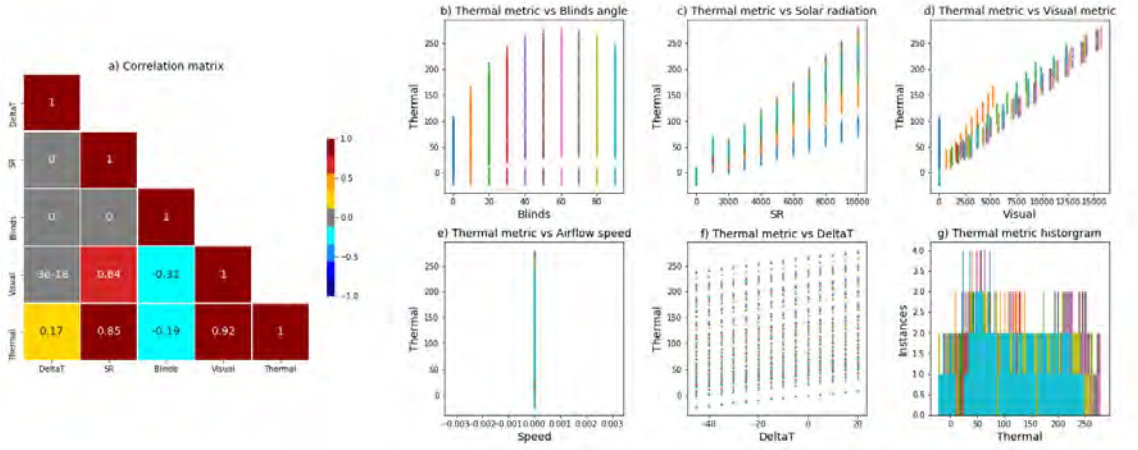



Figure 11: DSF thermal performance in buffer mode (steady-state simulation)

Similarly, due to isolation between indoor air and outdoor air as in static air buffer mode, the thermal performance does not much difference since the airflow speed has a less effect on thermal performance compared with solar radiation and blinds angle in interior air curtain (Figure 12) and exterior air curtain mode (Figure 13).

On the other hand, the thermal metric behaves different in supply air mode (Figure 15) and exhaust air mode (Figure 14). Exchanging indoor air and outdoor air through the cavity of DSF, the thermal behavior now depends primarily on the delta temperature and the cavity airflow speed in small association with the visual metric due to less being affected by the solar radiation as well as the blinds angle.

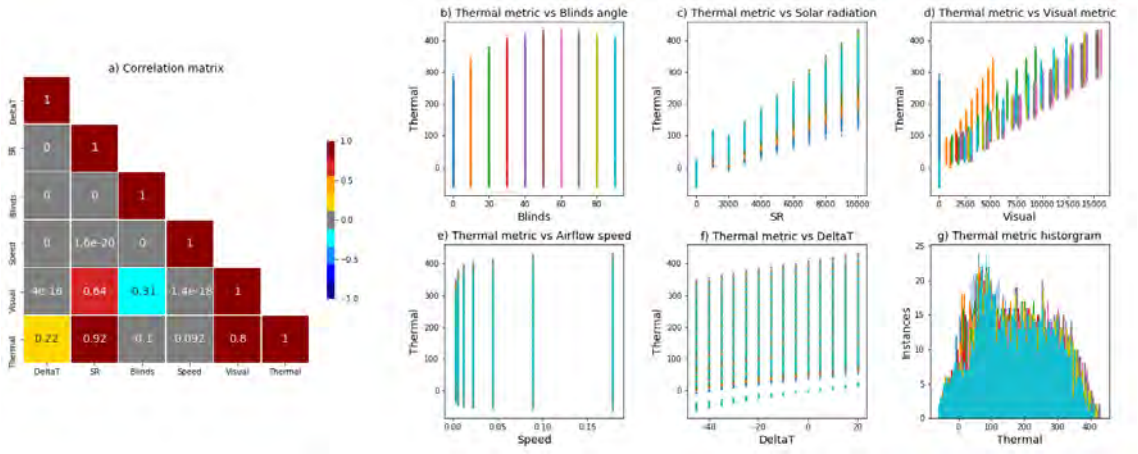



Figure 12: DSF thermal performance, Interior air curtain (steady-state simulation)

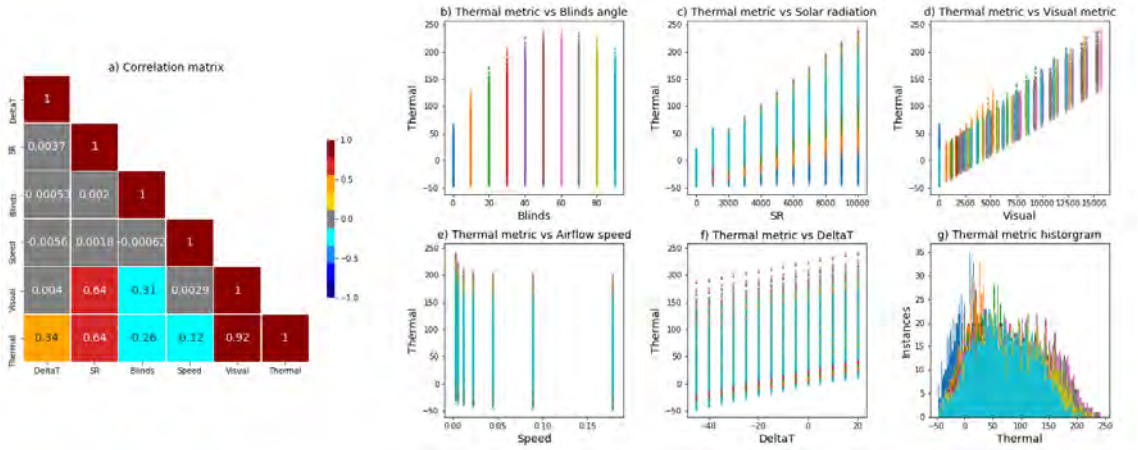



Figure 13: DSF thermal performance, Exterior air curtain (steady-state simulation)

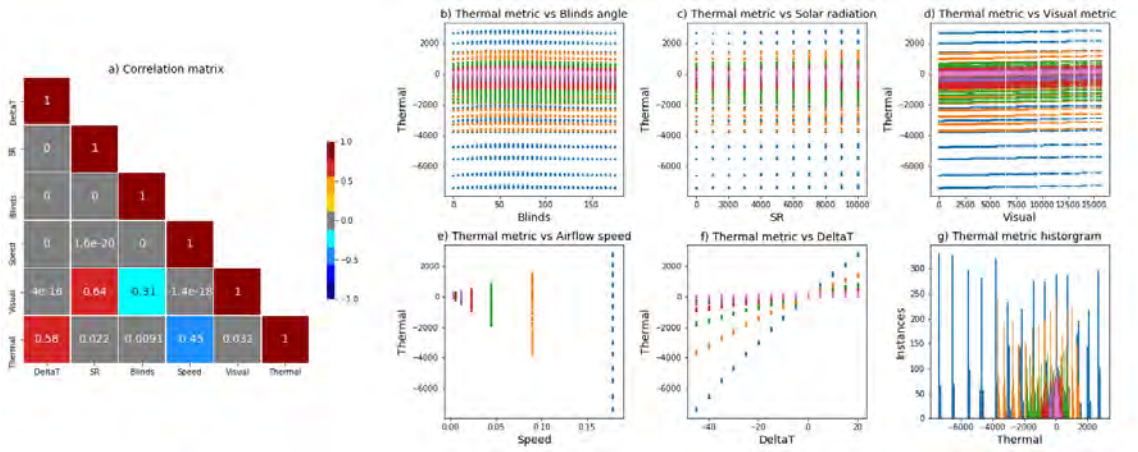



Figure 14: DSF thermal performance in exhaust mode (steady-state simulation)

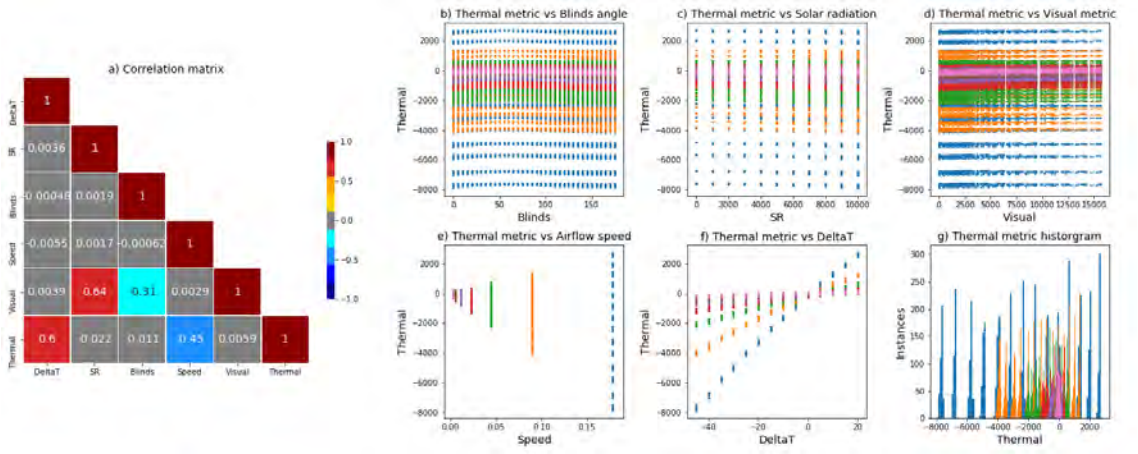



Figure 15: DSF thermal performance in supply mode (steady-state simulation)

### 4.3.3 Dynamic simulation dataset

With the large distribution of data in dynamic boundary conditions, the dynamic simulation dataset allows observing more realistic features and complicated relationships to analyze the ML problems. The trained models with the dynamic dataset were expected to be close to the behavior of real-time data. The following analysis will use the same approaches as done with the steady-state simulation dataset, i.e to discover the correlations between the variables standalone with time and their instant effects on the performance of DSF.

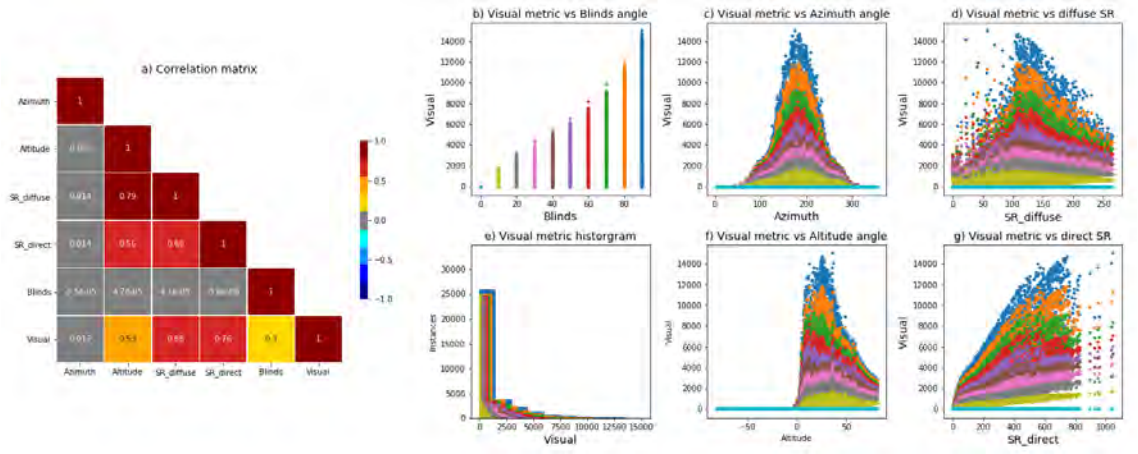



Figure 16: DSF visual performance (dynamic simulation)

Figure 16 exposes more relationships of the visual metric with both direct and diffuse solar irradiance as well as the azimuth angle and the altitude angle of the sun observed from DSF. The performance range of the Visual metric is widened with the larger open-angle of the blinds. Whereas, the visual metric increases with raising direct solar radiation but reversely with increasing diffuse solar irradiance that is

distributed with small values compared with direct solar irradiance though. Another feature on which the visual metric highly depends is the altitude of the sun with the attention of the blocking range of negative angles, i.e. not seen from DSF at all, and the peaks perfectly reached about 30 degrees. The visual metric range is also blocked when the azimuth angle goes below 50 degrees or above 300 degrees and peaks at about 180 degrees.

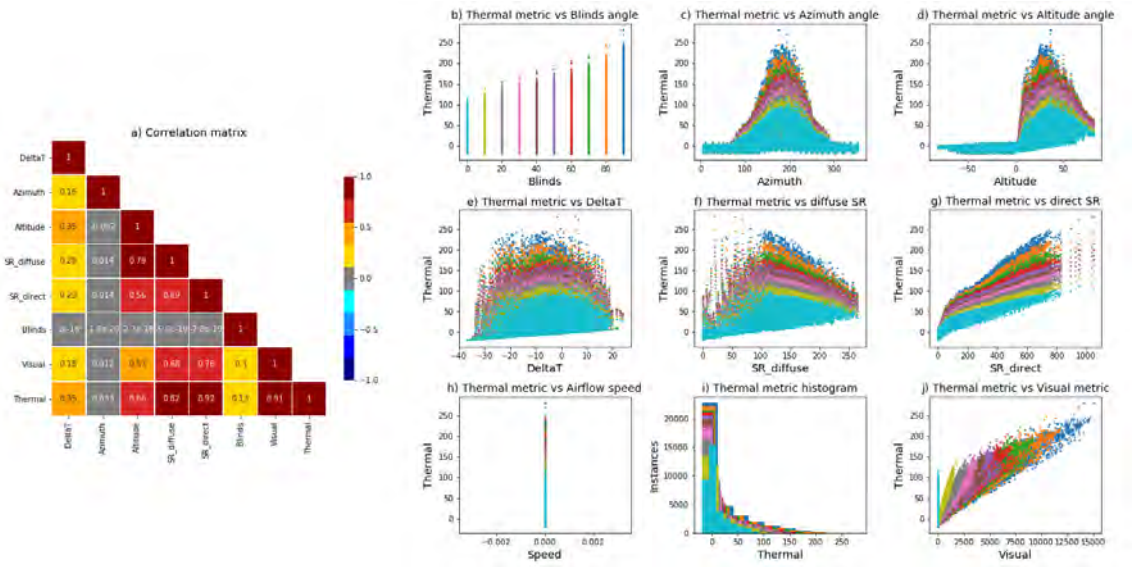



Figure 17: DSF thermal performance in buffer mode (dynamic simulation)

The same approach is applied to analyze the dependence of the thermal metric on the features of interest as in steady-state cases. Figure 17 explores the performance of thermal metric in air buffer mode set with zero airflow speed. The thermal behavior is in a similar shape but the decoupling between the thermal driving factors is not as clear as the visual performance. Figure 18 and Figure 19 again confirm on a larger scale that the thermal performance is less affected by the airflow speed compared with solar radiation and blinds angle in interior air curtain mode and exterior air curtain mode respectively, and isolation between indoor air and outdoor air makes similarity of these two modes and air buffer mode. In contrast, in supply air mode (Figure 21) and exhaust air mode (Figure 20), there is exchanging indoor air and outdoor air through the cavity of DSF. Once again the thermal behavior is expected to follow the analysis as with steady-state simulation that it now depends primarily on the delta temperature and the airflow speed that dominate the effects of the solar radiation, the solar altitude, and the blinds angle.

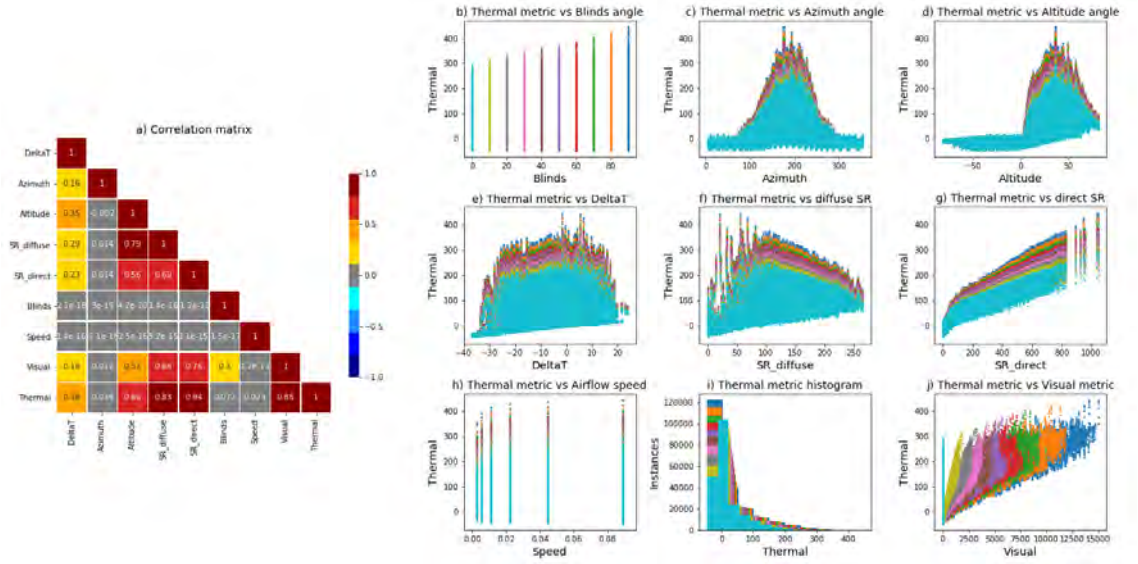



Figure 18: DSF thermal performance in interior air curtain mode (dynamic simulation)

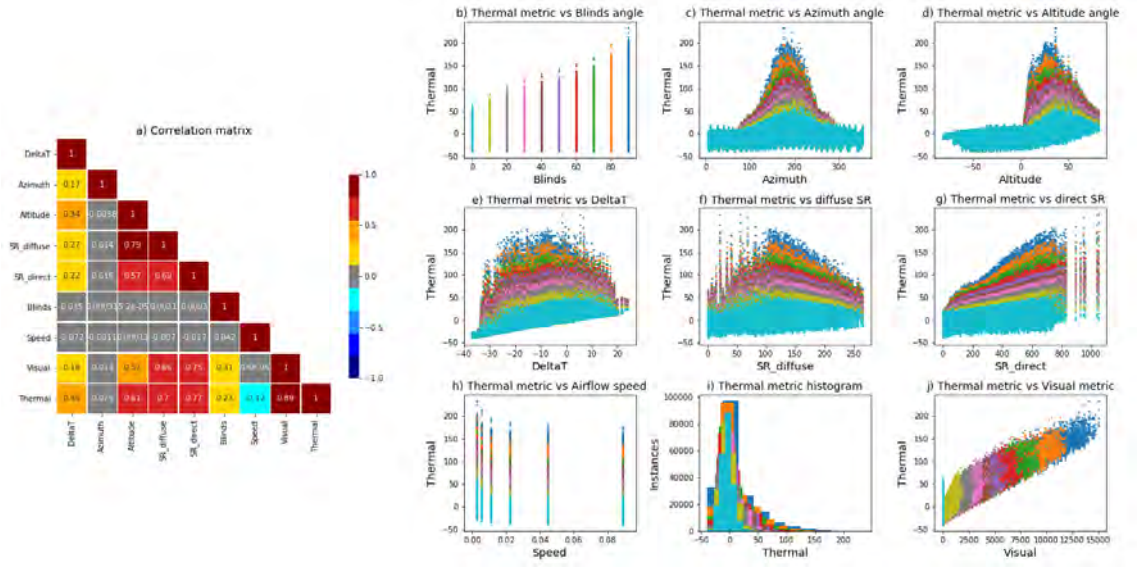



Figure 19: DSF thermal performance in exterior air curtain mode (dynamic simulation)

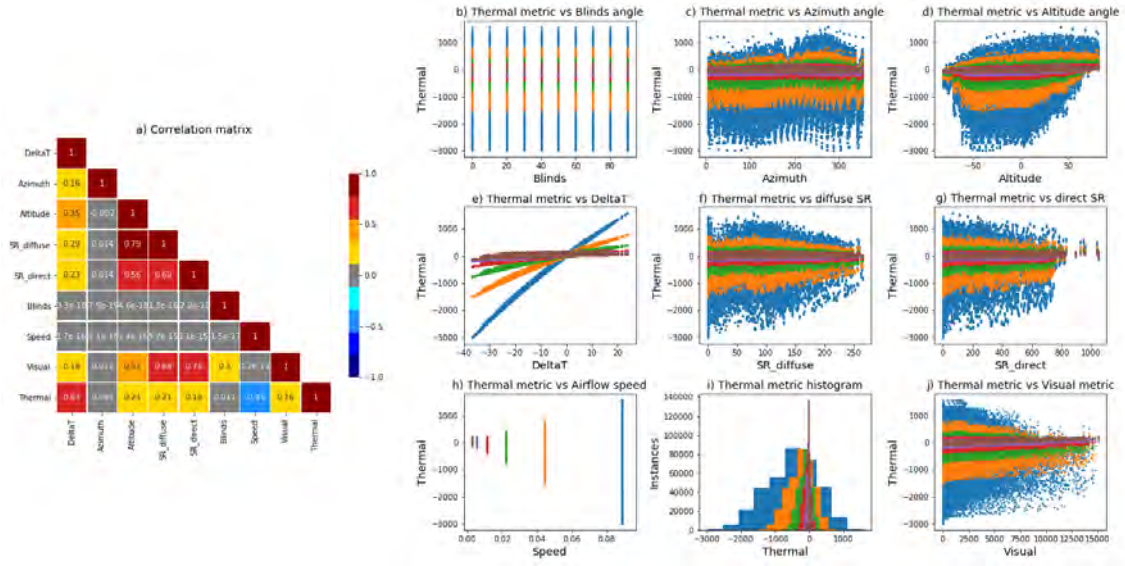



Figure 20: DSF thermal performance in exhaust mode (dynamic simulation)

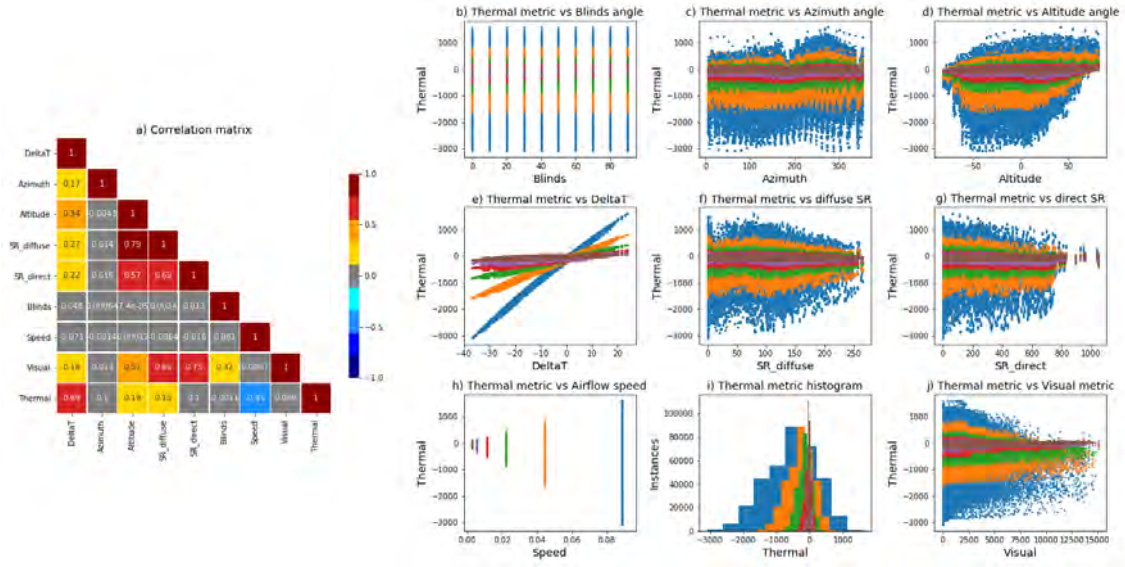



Figure 21: DSF thermal performance in supply mode (dynamic simulation)

### 4.3.4 Definition of ML problems

With observations obtained from the dataset, supervised learning can be applied to train black-box models. Indeed, the dataset is labeled which means we can always assign a known output to a set of inputs. Moreover, supervised learning as well as supervised neural networks are the most popular approach for black-box models to map the inputs to the output. There are two types of ML problems for supervised learning or supervised deep learning: regression and classification, which

are determined based on the data type of the target of prediction. Each type of ML problem imposes different learning skills and different metrics to evaluate the learning. Based on the targets of the DSF controller to be designed, the numerical and categorical outputs together with their notations are listed as follows:

- **Numerical outputs** : Thermal metric (`Thermal`), Visual metric (`Visual`), Blinds angle (`Blinds`), Airflow speed (`Speed`)
- **Categorical output** : Airflow path (`Airflow`)

Since Thermal metric and Visual metric are two main targets of the control problem and based on physical characteristics as well as correlations analysed earlier, the ML problems to be solved are defined as follows:

1. **Regression for Visual metric**: With `Visual` as the output variable, the analysis suggests the important features that are `SR` (Solar radiation) and `Blinds` in case of steady-state simulation dataset, or `SR_diffuse` (diffuse incident solar radiation), `SR_dirrect` (direct incident solar radiation), `Azimuth` (solar azimuth angle), `Altitude` (solar altitude angle), and `Blinds` in case of dynamic simulation dataset.
2. **Classification for Airflow path**: `Airflow` is the categorical output and it takes as inputs `DeltaT`, `SR`, `Visual`, and `Thermal` in case of steady-state simulation dataset, or `DeltaT`, `SR_diffuse`, `SR_dirrect`, `Azimuth`, `Altitude`, `Visual`, and `Thermal` in case of dynamic simulation dataset.
3. **Regression for Thermal metric in different modes**: For each of airflow modes, `Thermal` is the output variable and it takes as inputs `DeltaT`, `SR`, `Blinds`, `Speed`, and `Visual` in case of steady-state simulation dataset, or `DeltaT`, `SR_diffuse`, `SR_dirrect`, `Azimuth`, `Altitude`, `Blinds`, `Speed`, and `Visual` in case of dynamic simulation dataset.

The different categories of the datasets that are split for the corresponding ML problems above are listed in Table 1. Recalling from Section 3.2.2, different hypothesis suggestions can be considered to apply for either regression or classification such as CART or ANN. The possible hypotheses were checked out and used to solve the defined ML problems during training processes reported in Section 4.4. Thereafter, several best-learned models were cherry-picked for predicting Visual metric, Airflow path, and Thermal metric. The candidate models were subsequently compared and selected according to the evaluation criteria defined in Section 5.4. The finalized models were manipulated to design the predictors for the target operational modes of DSF the implementation of which is represented in Section 5.5.

## 4.4 Predictive modeling

### 4.4.1 Workflow of training processes

To solve each of the defined ML problems in Section 4.3.4, the implementation steps of ML model training processes are applied in the following order.

**Step 1: Prepare the training environment**. The software environment for training needs to be prepared with the installation of the required Python packages and dependencies such as Scikit-learn, Keras, Tensorflow, Cuda with compatible versions. Python programs were coded using multi-core processing and/or GPU computing (Section 4.4.3).

**Step 2: Prepare Problem**. Corresponding to what kind of ML problem to be solved at a time, this step was to load the different learning skills and evaluation metrics from the ML packages to be used. The dataset for learning, visualization tools, and the other relevant skills for data processing (normalization/standardization scalers or transformers, grid search, and cross-validation tools) were also to be loaded ready for use.

**Step 3: Prepare Data**. It was already started with cleaning the dataset for the specific ML problem that was done during the data analysis phase in Section 4.3. After selecting and separating the features (inputs) and the target (output), the data was split into a train set and a test set with a ratio of 0.33 for testing. Subsequently, standardization was needed to apply on both the features and the target of the training set.

**Step 4: Check Algorithms**. There are several options with different ML algorithms to solve the problem of regression or classification. This step was to run a spot check on the alternative algorithms being applied on the same training dataset of steady-state simulations with test options and evaluation metrics set up correspondingly and to compare for the best performance model(s) on a validation dataset that was also split out with a validation size of 0.33 from the training dataset. The cross-validation of 10 folds was used to evaluate the prediction quality of the different models.

**Step 5: Improve Performance**. The good candidate algorithms that were cherry-picked from Step 4 were to be improved for optimal performance by tuning with different parameters using grid search and 5-fold cross-validation skills.

**Step 6: Finalize Cherry-picked Models**. The best models with corresponding tuned parameters that result from Step 5 were applied to train the standalone models on the whole training set. The trained models were saved in readable format for later deployment. The quality of prediction on the test dataset with these models was evaluated and analyzed in Section 5.4.

### 4.4.2 Software libraries

The following are the software libraries or packages in Python that were set up to use in this work:

- **SciPy** is a Python-based ecosystem of open source libraries for mathematics, science, and engineering, being comprised of the core modules relevant to machine learning such as Pandas for tools and data structures to organize and analyze data, NumPy for efficiently working with data in arrays, and Matplotlib allowing charts and plots from data.

- **Scikit-learn** provides a collection of wrappers of ML algorithms for regression,

classification, or clustering, as well as tools for related tasks such as model evaluation, parameter tuning, and data, pre-processing. It is an open source that is built upon the SciPy ecosystem as "scikit" implying a SciPy plug-in or toolkit, and usable commercially under the BSD license. The ML wrappers for regression or classification that were used less or more in this thesis work include Linear Regressor (LR), Lasso Regressor (LASSO), Elastic Net Regressor (EN), Logistic Regression (LOG), Linear Discriminant Analysis (LDA), Gaussian Naive Bayes Classifier (NB), k-Nearest Neighbors Regressor/Classifier (KNN), Classification and Rregression Trees so-called Decision Tree Regressor/Classifier (CART/DT), Support Vector Machine Regressor/Classifier (SVM/SVR/SVC), Ada Boosting Regressor/Classifier (AB), Gradient Boosting Regressor/Classifier (GBM), Random Forest Regressor/Classifier (RF), Extra Trees Regressor/Classifier (ET), and Multi-layer Perceptron Regressor/Classifier (MLP). [18]

- **Keras** is a powerful wrapper library that allows minimalist code snippets for deep learning. It can run on top of Theano and TensorFlow backends - two of the top numerical computation platforms in Python that provide the basis for deep learning research and development. Keras is an open-source that is released under the permission of the MIT license. In this work, Keras was set up to use Tensorflow backend.

- **Tensorflow** is a Python library for fast numerical computing created and maintained by Google, and released under the Apache 2.0 open source license. It can run on systems of single or multiple CPU(s)/GPU(s) as well as mobile devices and large scale distributed systems of hundreds of computers (cluster).

### 4.4.3 Multi-core processing and GPU computing

To train the ANN model with Keras using Tensorflow backend on a local computer, it requires to have Nvidia GPU and install CUDA to run on it. CUDA (Compute Unified Device Architecture) is a parallel computing platform and application programming interface (API) model created by Nvidia. All software versions must match each other. However, the training work was divided into multiple tasks with a significant number of repeated runs per task during training or tuning models, which results in lots of time consumption applying parallel computing on a single computer though.

Another solution to handle a large amount of dynamic simulation data was to set up the training work or data processing to run on the Triton cluster. Triton is the Aalto high-performance computing cluster. With free access for researchers, the Triton cluster that is integrated into the Aalto environment is similar to the larger CSC cluster (https://research.csc.fi/csc-s-servers). Being allocated a number of enough CPUs/GPUs and memory, each run of a training/tuning task could be performed by parallel computing within 1-2 hours on average. The system of fast processing units including multi CPU cores and/or GPUs for parallel computing is powerful to accelerate compute-intensive applications and help a lot to release the pressure of the work.

# 5 Results and analysis

## 5.1 Visual metric predictive models

It was good to get started with the small-sized dataset of steady-state simulations for spot-checking and overall evaluation of the different learning algorithms for the same ML problem. There was no need for multiprocessing or GPU computing in these cases as well. Model tuning or cross-validation of 10 folds can be executed pretty fast. The checking models were evaluated with comparable validation errors through negative mean squared error with the application of standardization on the dataset.

For regression of Visual metric in steady-state simulation cases, the options of learning skills borrowed from Scikit-learn for model checking are shown in Figure 22. The checking used the default settings of training parameters for each algorithm except the MLP model that was adjusted with two hidden layers of the size (120, 60) for better comparison. The models best performed in spot-checking for Visual metric are the models using Tree methods such as ET, RF, GBM, or DT. The three former models are ensembles of multiple DTs that better improve the performance in terms of both training speed and scoring. Especially, the MLP model needs to be evaluated separately for deep learning skills that the checking result was not enough to conclude since the neural network method requires a sufficiently large data size to be processed for training. For a closer view, Figure 23 visualize the effectiveness of Visual metric prediction on the standardized dataset using the ETR model that owns the best score on checking without limit of maximum tree depth nor pruning by default settings. It fits the data perfectly so that the predicted visual metric at missing solar radiation of 1000 W/m$^2$ looks pretty much consistent with the rest of the known data.

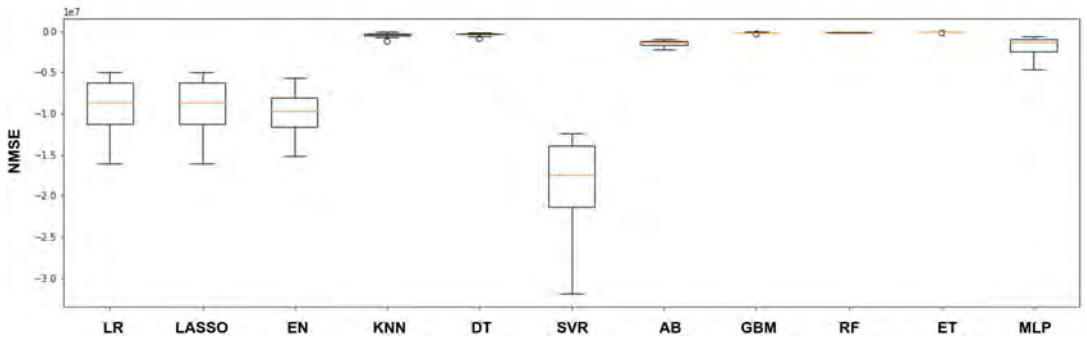



Figure 22: Spot checking of different predictive models for Visual metric

It was interesting to apply the ETR model and see how it was fitted for Visual metric on the dynamic simulation dataset. It was also worth paying attention to the GBM model or the MLP model. The dynamic data size is large enough for training the MLP model but with Keras libraries instead of Scikit-learn since it was limited to train the MLP model from Scikit-learn which offers only forward propagation algorithm without the support of multi-core processing nor GPU computing. More luckily, the ETR model or RFR model can utilize multi CPU cores for processing. On

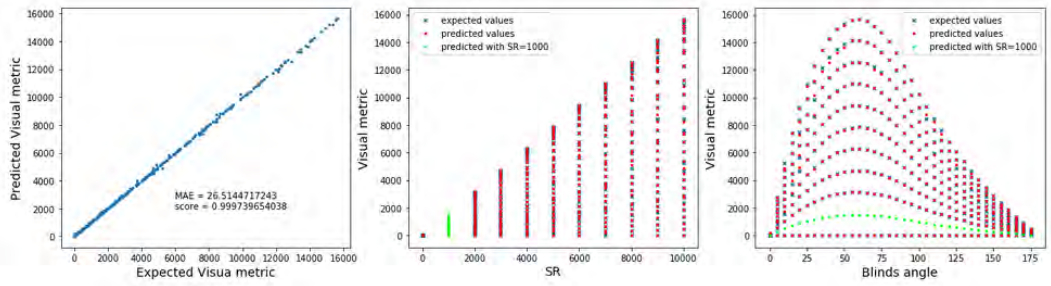



Figure 23: Visual metric prediction with ETR model (steady-state simulation)

the other hand, XGB could be used instead of the GBM algorithm from Scikit-learn. XGB is more robust in performance, supports GPU computing, and produces the trained models of compact size compared with other Tree Ensembles. ANN model and XGB model proved their advantages on the model size with the growing size of the training dataset which will be discussed in more details in Section 5.4. Consequently, ETR, XGB, and ANN were three candidates that were selected for training and evaluated for the prediction of Visual metric in dynamic simulation cases.

With experiments, the longer training time of the ANN model with increasing sizes of epochs and batches reduced the loss and raised better scores. The number of batches was chosen to be as close as to the sample size, resulting in better scores. For quick tuning hyperparameters with k-fold cross-validation, small batch size and small epoch size can be used but the final ANN model was trained longer with 10000 epochs and 300000 batches. The activation 'relu', the optimizer 'adam', normalization used for kernel initialization also proved well performance. The structure of hidden layers was the noticeable hyperparameter for tuning the ANN model (Figure 24). To avoid overfitting, regularization with an L2 value of 0.0001 and a dropout ratio of 0.5 was applied. Besides, Figure 25 and 26 shows tuning the ETR model and the XGB model respectively with different amounts of the maximum depth of trees and the number of trees. The XGB model was trained on a GPU with a learning rate set at a high ratio of 0.3. Training the ETR model was also sped up using multi CPU scores by setting parameter `n_jobs`.

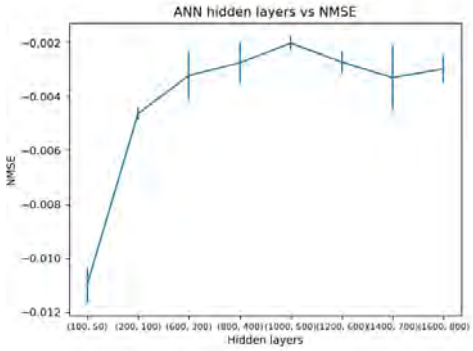



Figure 24: Tuning Visual metric ANN model with different sizes of two hidden layers

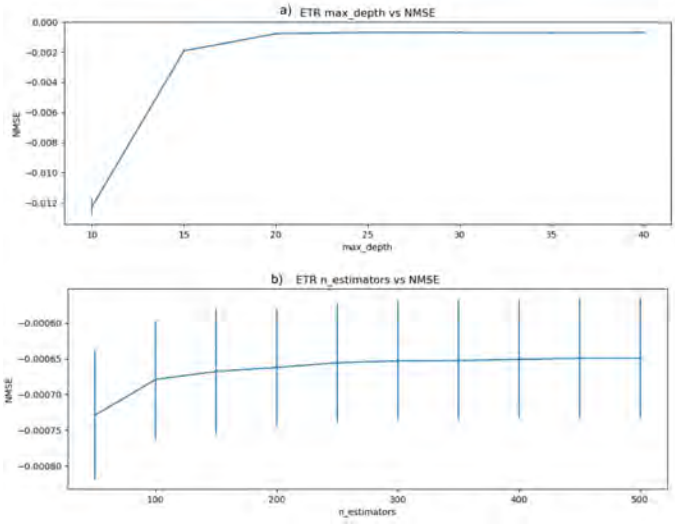



Figure 25: Tuning Visual metric ETR model with different sizes and numbers of trees

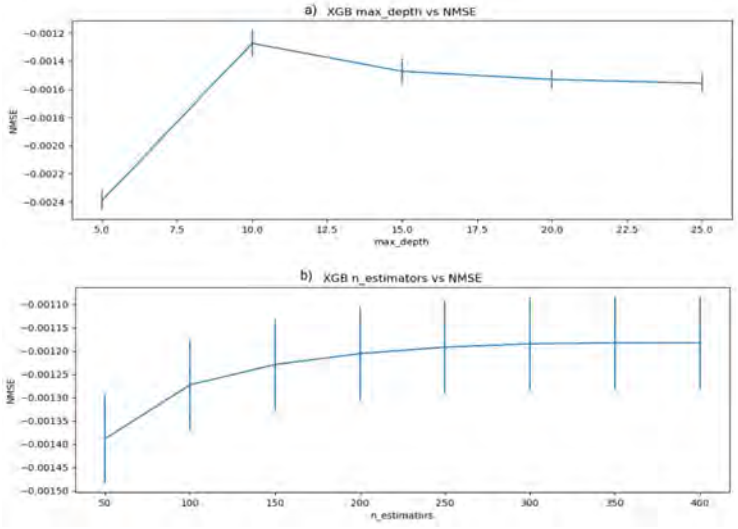
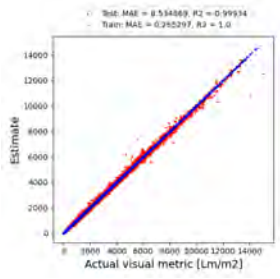
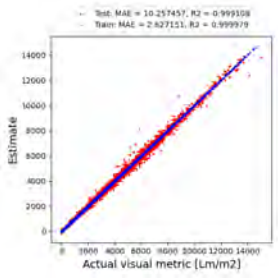
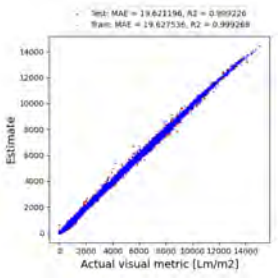



Figure 26: Tuning Visual metric XGB model with different sizes and numbers of trees

Table 2 summarizes the key parameters that were set for training the models of Visual metric and evaluation of predictive errors on both the train set and the test set. The ANN model induced higher bias but nearly zero variance. It was taken into account that the ETR model was not applied regularization by decision tree pruning. By adjusting `ccp_alpha` parameter with a small value of one millionth, lower variance can be induced but also higher bias to the error level of ANN network. The XGB model was set with L2 regularization of 1 as default.

Table 2: Parameter settings of training models for Visual metric

| Models | ETR | XGB | ANN |
|---|---|---|---|
| **Parameters** | n_jobs = 16,<br>criterion = 'mse',<br>max_depth = 25,<br>n_estimators = 400,<br>random_state = 7 | tree_method = 'gpu_hist',<br>predictor = 'gpu_predictor',<br>objective = 'reg:squarederror',<br>max_depth = 10,<br>n_estimators = 300,<br>learning_rate = 0.3,<br>random_state = 7 | hidden_layers = (1000,500),<br>epochs = 10000,<br>batch_size = 300000,<br>L2=0.0001,<br>dropout = 0.5,<br>kernel_initializer = 'normal',<br>activation = 'relu',<br>optimizer = 'adam',<br>loss = 'mse',<br>metrics = 'mae' |
| **Train vs Test** |  |  |  |

## 5.2 Airflow path classifying models

The training processes for the Airflow path were done with similar steps as for the Visual metric. Figure 27 shows a spot check on different models for classification of Airflow path using steady-state simulation dataset. Once again the Tree models dominate the other methods. The best one was the RFC model. With the same reasons that were previously analyzed for training the predictive models of Visual metric, XGB, and ANN together with RFC were the selected models for training and improving the estimation of the Airflow path.

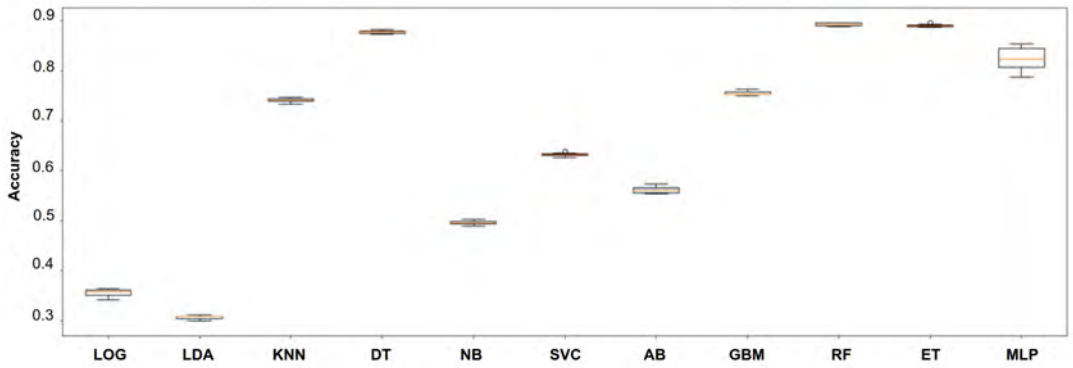
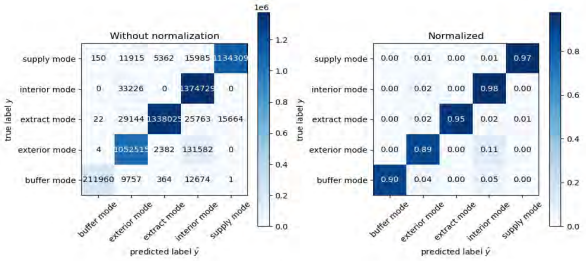
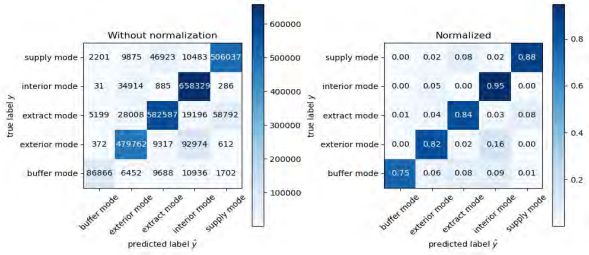
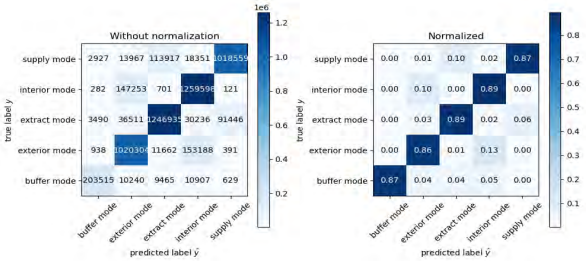
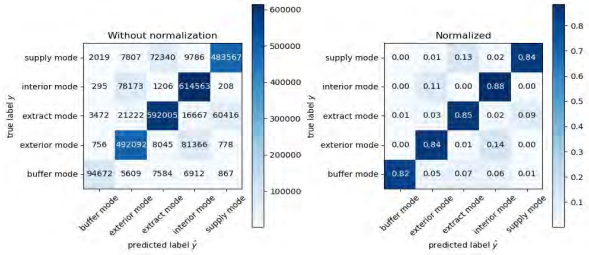
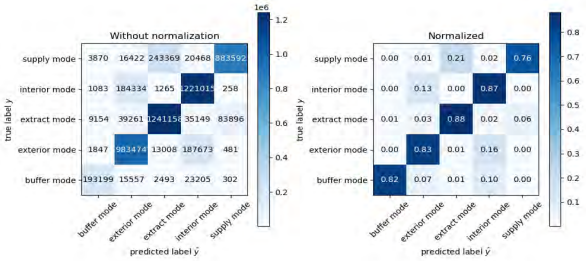
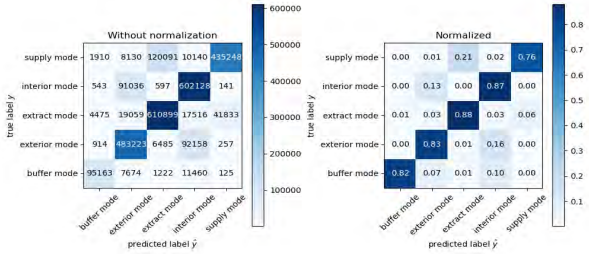



Figure 27: Spot checking of different predictive models for Airflow path using steady-state simulations

Table 3 shows settings for training the models of Airflow path with dynamic simulation dataset, and compare Table 3 shows settings for training the models of Airflow path with dynamic simulation dataset and compare the predictive accuracy on the train set and the test set. Since the whole dynamic dataset is large and due to the time limit of the work, model tuning was restricted by constructing the training models with similar structures as used for Visual metric models but setting the corresponding evaluation metrics for classification problems. The results are pretty promising. In common sense, the RF model gained the highest accuracy average of 94.5% on the training set but also the most variance in comparison with about 86,9% accuracy on the testing set. Otherwise, the XGB model has average performance with train accuracy of 87,9% and less variance with a test accuracy of 85,5%. Although, being the last in the order, the ANN model won less success in estimating the supply air mode but inducing almost no variance on the unseen data with approximately 83.6% accuracy.

Table 3: Parameter settings of training models for Airflow Path

| Models | Parameters | Train Accuracy | Test Accuracy |
|---|---|---|---|
| **RFC** | n_jobs=24,<br>'criterion': 'entropy',<br>max_depth = 25,<br>n_estimators = 400,<br>random_state=7 |  |  |
| **XGB** | tree_method='gpu_hist',<br>predictor='gpu_predictor',<br>objective='multi:softprob',<br>max_depth = 10,<br>n_estimators = 300,<br>learning_rate=0.3,<br>random_state=7 |  |  |
| **ANN** | hidden_layers = (1000,500),<br>epochs = 1000,<br>batch_size = 500000,<br>L2 = 0.0001,<br>dropout = 0.5,<br>kernel_initializer='normal',<br>activation = 'relu' (hidden layers)/<br>'softmax'(output layer),<br>optimizer = 'adam',<br>loss = 'categorical_crossentropy',<br>metrics='acc' |  |  |

## 5.3 Thermal metric predictive models

For training models of Thermal metric in each airflow mode, model checking on the steady-state simulation dataset was run with the same regression models used for Visual metric (Figure 28). There were more good models for selection in Buffer mode cases. The MLP neural network of one hidden layer of 100 neurons was checked out as the most outstanding learning skill over the others whereas Trees still keep good performance as usual. Otherwise, normal or Lasso linear multi-regression, as well as k-nearest neighbors, also came out significantly well-done in air buffer mode. In the other modes, the spot-checking turned out the similar results as checking models for Visual metric so that the ETR was the most excellent in performance, followed by RF, DT, MLP, GBM, and KNN. Figure 29 represents how Thermal metric was fitted on the steady-state simulation dataset with MLP model for Buffer mode and ETR models for the other modes. The prediction gave a consistent shape of missing samples at solar radiation of 1000 $W/m^2$.

The dynamic or real data can be expected to treat differently so that the linear regression algorithms may be too simple to apply. With dynamic data, ANN, ETR, and XGB were again selected to be improved and compared performance on fitting Thermal metric in any airflow mode. The models were tuned with a small size of samples to show working well with the same settings as used for the models of Visual metric. All training parameters and results are listed in Table 4 for the models forecasting Thermal metric in 5 different modes. XGB and ANN were trained on a GPU while ETR was done by parallel working CPUs with the same structures of trees or neural networks as used for Visual metric predictive modeling. Tuning the models, especially cost complexity pruning for ET consumed a lot of time, and therefore the settings that have been found may not optimal yet and no pruning was applied on ETR eventually. However, the results show the pretty promising and comparable performance of the learning models. For overall ranking, the ETR models were best in the estimation of the Thermal metric considering the lowest bias, followed by the XGB model and the ANN model which were well regularized for lower variance. For instance, the thermal performance in Buffer mode is estimated differently by the three models: ETR and XGB has done with an average MAE of Thermal metric less than 0.4 $W/m^2$ on the whole data set, noticed that the test set weights only 33%, while ANN gives the MAE of Thermal metric of about 0.6 $W/m^2$. However, the difference between the train and the test MAE of the Thermal metric estimated by ANN is negligible but most significant with ETR. Comparing also the performance in other modes, the variance of the XGB model is quite balanced in the medium between these two models. For ranking the three models, ETR holds the highest average $R^2$ score, followed by XGB and ANN the lowest. Nevertheless, if applying regularization with DT pruning on ETR, the performance would be brought down to the level of the XGB or worse. Moreover, the large model size of ETR is a disadvantage for designing the real-time DSF controller as discussed more general in Section 5.4.

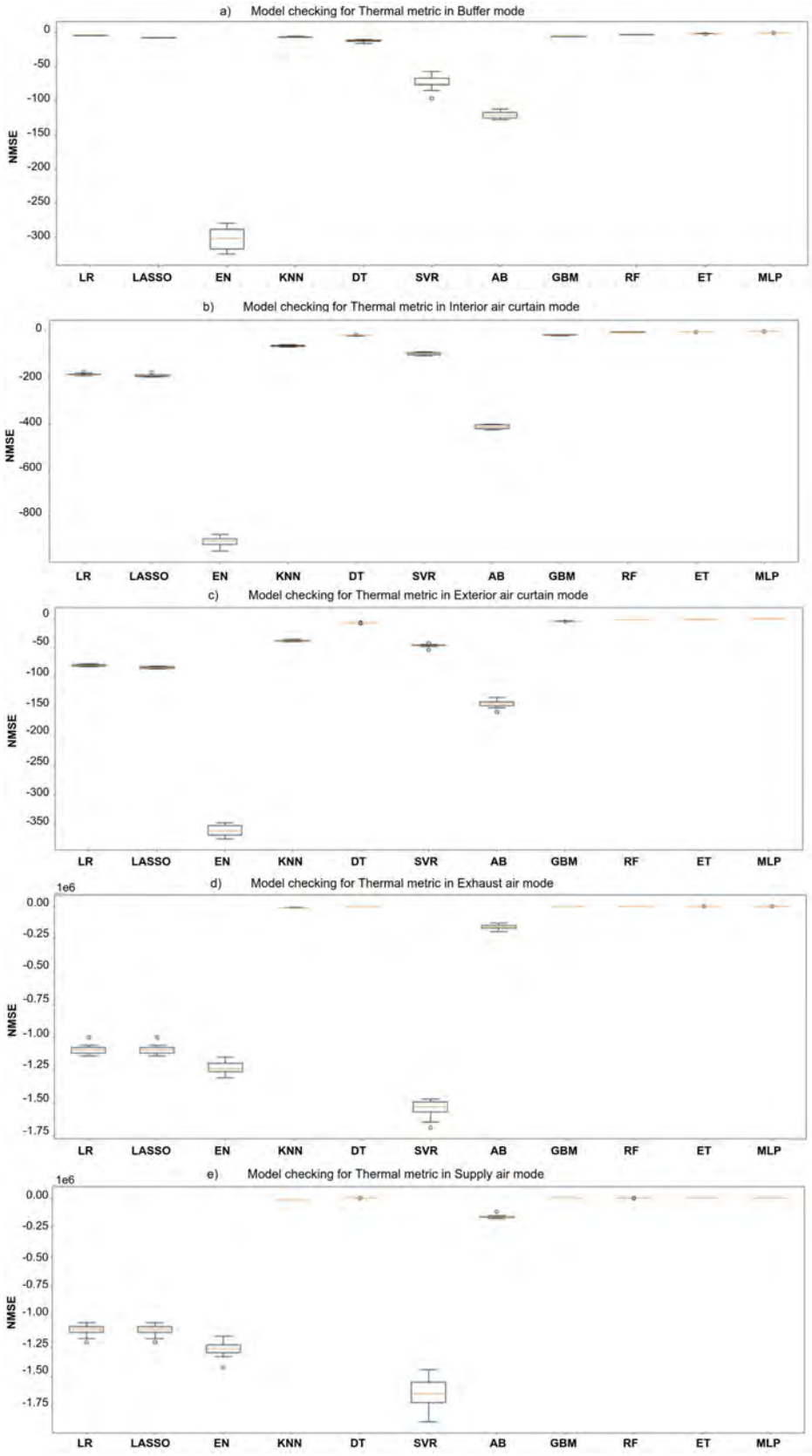



Figure 28: Model spot-checking for Thermal metric in different modes using steady-state simulations

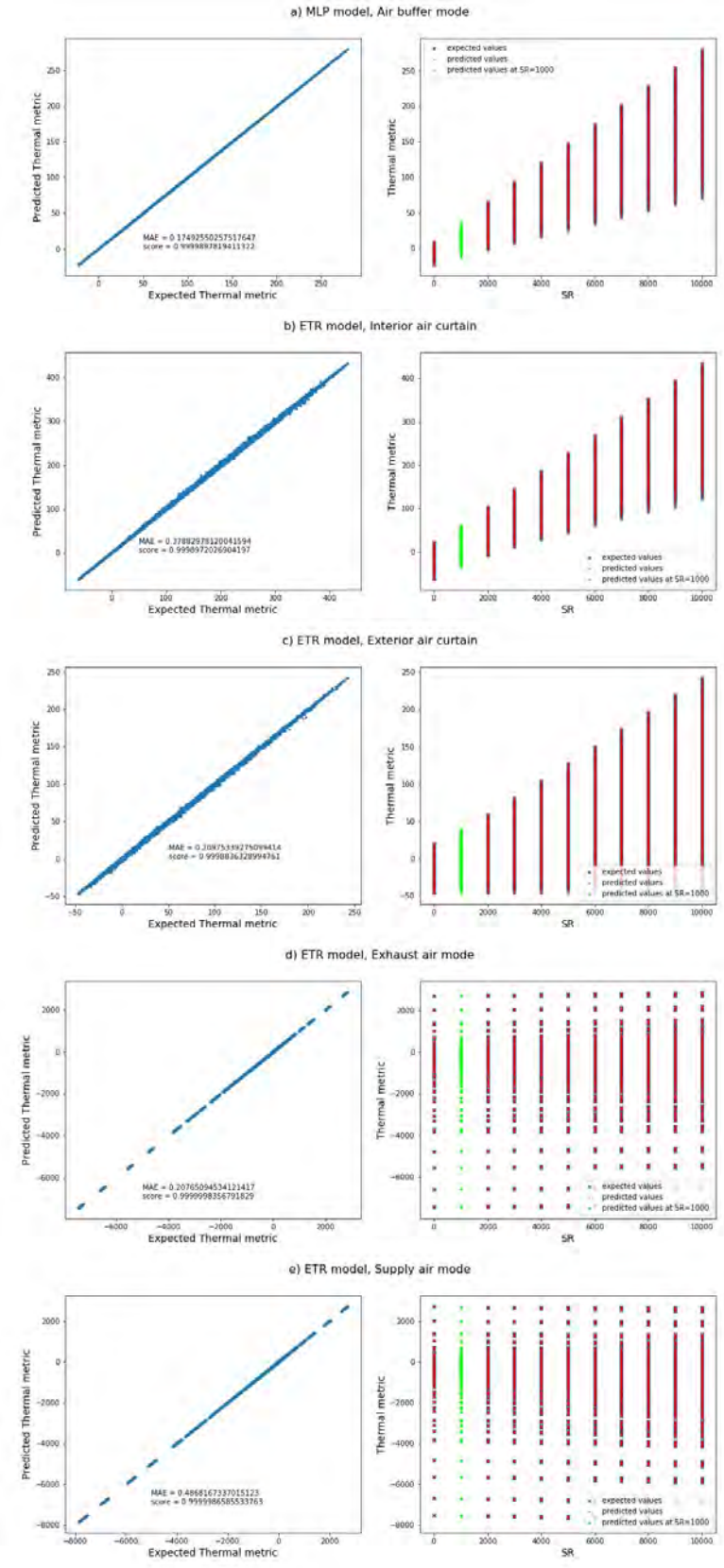
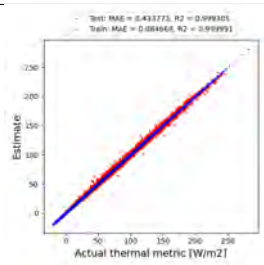
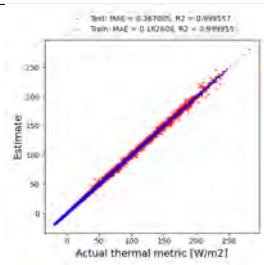
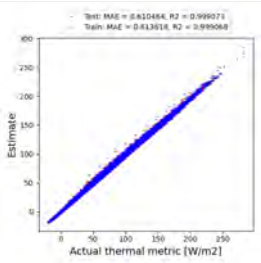
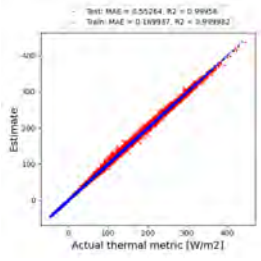
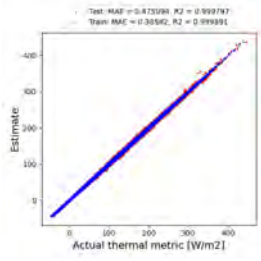
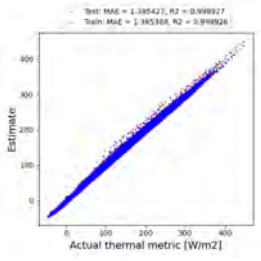
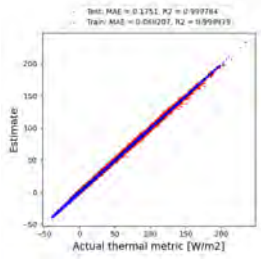
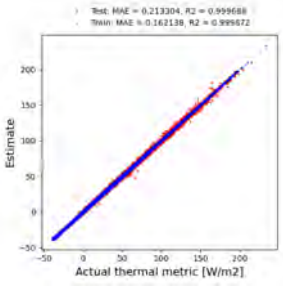
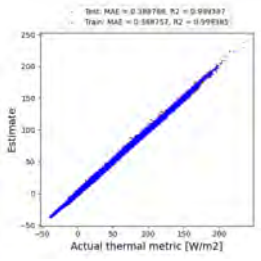
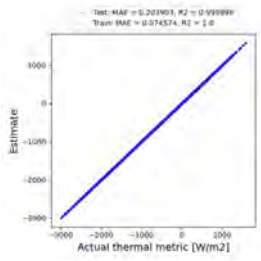
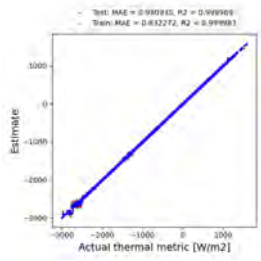
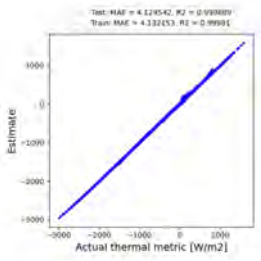
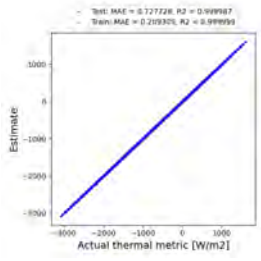
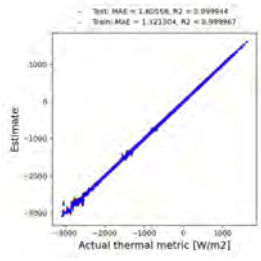
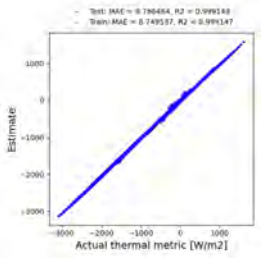



Figure 29: Thermal metric prediction with different airflow modes (steady-state simulation)

Table 4: Parameter settings of training models for Thermal metric

| Models | ETR | XGB | ANN |
|---|---|---|---|
| **Parameters** | n_jobs = 24,<br>criterion = 'mse',<br>max_depth = 25,<br>n_estimators = 400,<br>random_state = 7 | tree_method = 'gpu_hist',<br>predictor = 'gpu_predictor',<br>objective = 'reg:squarederror',<br>max_depth = 10,<br>n_estimators = 300,<br>learning_rate = 0.3,<br>random_state = 7 | hidden_layers = (1000,500),<br>epochs = 10000,<br>batch_size = 300000,<br>L2=0.0001,<br>dropout = 0.5,<br>kernel_initializer = 'normal',<br>activation='relu',<br>optimizer = 'adam',<br>loss = 'mse',<br>metrics='mae' |
| **Air Buffer** |  |  |  |
| **Interior Air** |  |  |  |
| **Exterior Air** |  |  |  |
| **Exhaust Air** |  |  |  |
| **Supply Air** |  |  |  |

## 5.4 Model Evaluation and Selection

The training results generally showed that RF, ET, or XGB outperformed both the regression and classification problems. However, the reliability of each learned model is judged by analyzing the bias-variance trade-off [16] between the train errors and the test errors. The compromise between bias (systematic error in prediction from actual values) and variance (systematic error in prediction on the train and test data) is a criterion to evaluate the model complexity, i.e whether the trained model is overfitting (high variance), underfitting (high bias), or appropriate fitting. With appropriate fitting, the model can be considered relatively reliable for use if the errors have low bias and low variance within an acceptable limit that meets the specific requirements in the practice of the application. Comparing the learned models (Table 5) and selecting the final one for designing the DSF controller was based on the compromise of the criteria defined as follows:

Table 5: Model comparisons

| | **Bias-variance trade-off** | **$R^2$ score or accuracy** | **Model size** |
|---|---|---|---|
| **ET/RF** | lower bias but higher variance | highest | >30 GB |
| **XGB** | average balance between bias and variance | average | ∼20 MB |
| **ANN** | lower variance but higher bias | lowest | ∼2 MB |

- **Bias-variance trade-off**: The level of bias-variance balance between the train errors and the test errors. The scatter plots of the train and test errors that are collected into Table 6 interpret comparable bias-variance compromise of XGB, ETR, and ANN models for Visual metric, and Thermal metric in different modes. As can be seen, the variance of regression errors is not constant likely due to the presence of outliers or extreme leverage values that were caught by the model but not generalized well from training to be applied on the unseen data. It needs to take into account the confidence range of sample prediction, e.g. +/- 50 W/m$^2$ or W/m$^2$.

- **$R^2$ score or accuracy**: $R^2$ score for regression models and accuracy for classification models are the metrics used to evaluate the predictive quality of the trained models. In comparison, ETR modeling Visual metric or Thermal metric gives the highest overall score or RFC for Airflow Path the highest accuracy, followed by XGB the second and ANN the third, recalled from Table 2, Table 3, and Table 4.

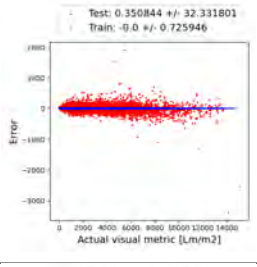
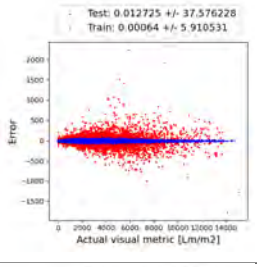
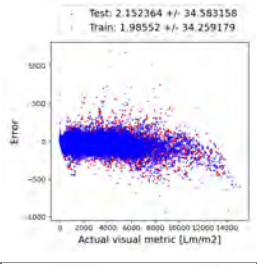
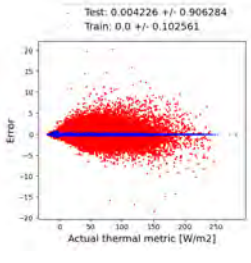
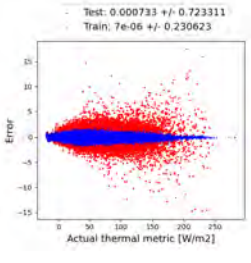
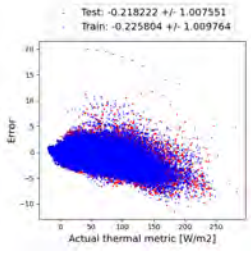
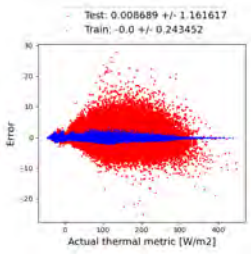
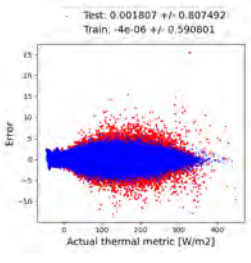
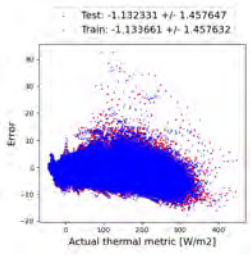
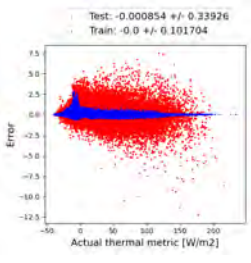
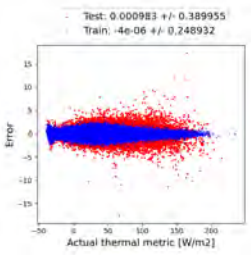
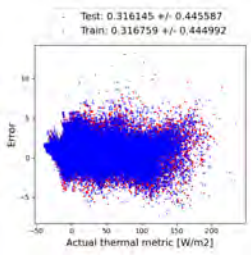
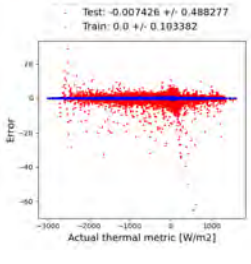
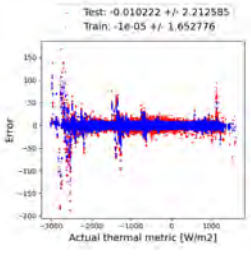
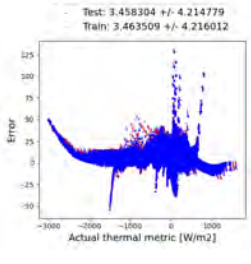
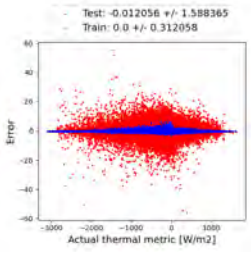
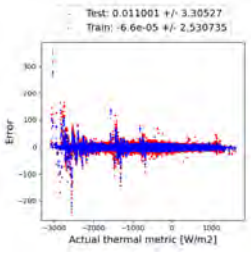
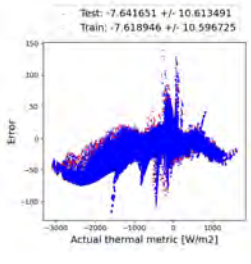
- **Model size**: The size of the used model affects memory storage and usage, real-time latency, and inference speed. Moreover, GPU can take full usage of the RAM of a few GB total, which probably results in running out of memory at runtime and the program being terminated by the operating system. Whereas, full CPU usage quickly drains the battery or causes the overheating problems of

the control device. Although this thesis work has not yet proceeded to deploy a trained model on a real-time controller, it is useful to have some insight into how fast the model will be running on the target device in the real world.

ANN model was well regularized for low variance but comparatively, high bias, leading to lower overall $R^2$ score or accuracy which can be improved by, e.g. feeding more training data to be exposed to the neural network. Moreover, ANN is a light model to be deployed with small model size and fast inference speed. In contrast, ETR/RFC holds not only the highest score/accuracy and the lowest bias but also the highest variance. To make bias and variance more balanced for the ETR model for Thermal metric or RFC for Airflow Path, regularization can be applied to lower the variance but also increase the bias. Provided that regularization adds a penalty to reduce the model complexity, the performance of ETR/RFC will be just as good as XGB but the model size does not benefit, e.g. 33.57GB of ETR model for Thermal metric in Exhaust air mode, or 21.68GB of RFC for Airflow path compared with tens of MB of XGB model or a few MB of ANN model. Although, the model size of ETR/RFC can be much reduced by using fewer estimators. For example, given the same `max_depth` of 25 and allowed a little drop in the predictive score, the ETR model for Visual metric has the size (about 7 times less than the size of the ETR models for Thermal metric in any mode except Air buffer mode) of approximately 5,05GB, 3.78GB, 2.52GB, and 1.26GB using the number of trees of 400, 300, 200, and 100 respectively. Finally, XGB comes out as the average performance model as allowed with some compromise considering all criteria in terms of bias, variance, score/accuracy, size, and speed. In fact, XGBoost algorithm takes advantages of both the DT ensemble method and the SGD algorithm as in ANN, giving the good reasons to explain its average performance. Therefore, the models learned with XGBoost were the chosen ones for the purpose to design the DSF control model and evaluate the outcomes of the design.

Nevertheless, there is no conclusion on which model is the best since each of them has its own opportunities to be improved for better quality with hyperparameter tuning and/or better regularization that was limited by the time span of this thesis. The evaluation criteria for model selection will be applied depending on the circumstances and the applications. More discussion about limitations and available solutions will be presented in Chapter 6.

Table 6: Comparison of bias-variance balance between XGB, ETR, and ANN models

| Mode vs Model | ETR | XGB | ANN |
|---|---|---|---|
| **Visual metric** |  |  |  |
| **Thermal metric, Air Buffer** |  |  |  |
| **Thermal metric, Interior Air** |  |  |  |
| **Thermal metric, Exterior Air** |  |  |  |
| **Thermal metric, Exhaust Air** |  |  |  |
| **Thermal metric, Supply Air** |  |  |  |

## 5.5 Design of DSF control model

### 5.5.1 Control workflow

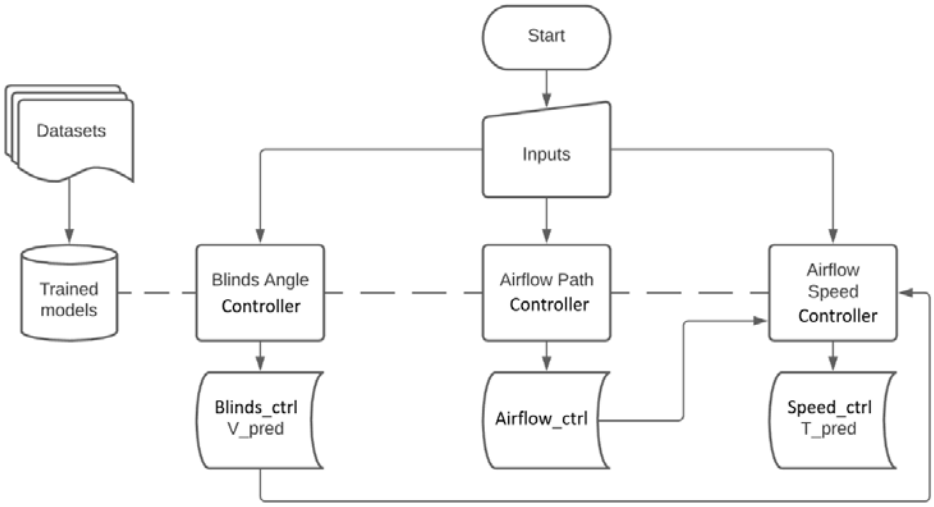



Figure 30: The workflow of the control model of DSF

Figure 30 demonstrates the workflow of the designed DSF controller using the learned models from solving the ML problems defined in Section 4.3.4. The control model consists three main components that are Blinds Angle Controller, Airflow Path Controller, and Airflow Speed Controller. The model evaluating criteria were discussed in details in Section 5.4 for selecting which predictive models to be applied as the components of the designed DSF controller.

- **Blinds Angle Controller** reads in the inputs for the learned model of Visual metric which is then manipulated with some simple calculation to output the Blinds angle (`Blinds_ctrl`), and return the predicted Visual metric (`V_pred`)
- **Airflow Path Controller** takes the inputs for the learned model of Airflow path, and estimate Airflow path (`Airflow_ctrl`)
- **Airflow Speed Controller** requires the inputs for the learned model of Thermal metric in the specified Airflow mode. Rather than the inputs of boundary conditions and `T_opt`, it waits for `Airflow_ctrl` previously delivered by Airflow Path Controller as well as `Blinds_ctrl` and `V_pred` from Blinds Angle Controller as the remain inputs. The Airflow speed `Speed_ctrl` is then calculated and returned together with predicted Thermal metric `T_pred`.

### 5.5.2 Implementation of Blinds Angle Controller

The XGB model for Visual metric was used to operate the Blinds Angle Controller the workflow of which is illustrated in Figure 31. The inputs are the optimal value of Visual metric `V_opt` sent from the central controller, and the measures converted

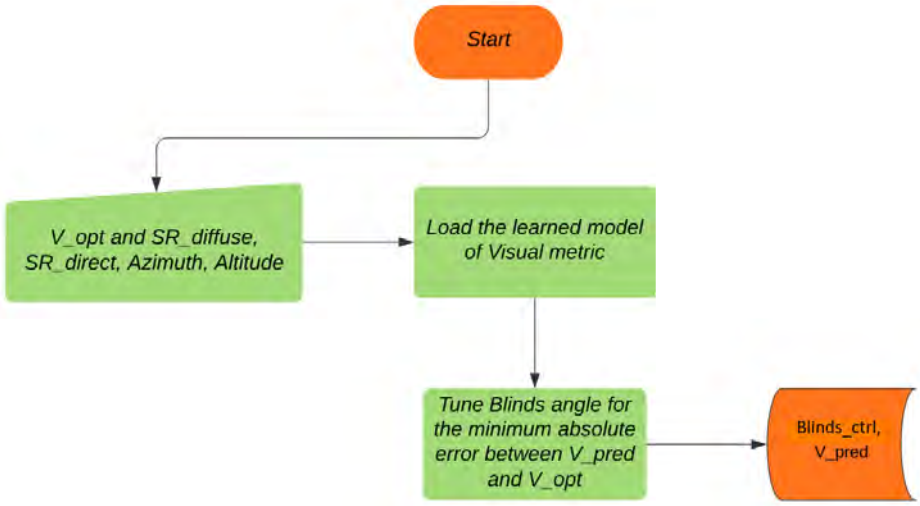



Figure 31: The workflow of Blinds Angle Controller

from the DSF embedded sensors of both direct and diffuse solar irradiance, the azimuth and altitude angle locating the sun's position with respect to the DSF. The XGB model is then loaded and ready for prediction. The Blinds angle is another required input that is internally ranged from 0 to 90 degrees with an interval of 10 degrees that is adjustable with control. The XGB model collects all those inputs together and does its magic to output the corresponding results of prediction for Visual metric that is subsequently filtered for the only one with the least absolute error to `V_opt`. The corresponding Blinds angle and the selected prediction of Visual metric will be returned to the mother program as `Blinds_ctrl` and `V_pred`. By this simple "filtering" algorithm, there is a chance for some predictive values of Visual metric with less accuracy to be replaced with better estimates of other operational Blinds angles.

### 5.5.3 Implementation of Airflow Path Controller

The simple workflow of Airflow Path Controller as seen in Figure 32 is to directly use the XGB classifier for the predicted airflow path `Airflow_ctrl`, recalled with average 86% predictive accuracy. The predictor reads in the optimal values `T_opt, V_opt` of Thermal metric and Visual metric that the central controller demands the DSF controller to estimate the delivery, and the measures of the direct solar radiation `SR_direct`, the diffuse solar radiation `SR_diffuse`, the azimuth angle `Azimuth`, and the altitude angle `Altitude` that are sensed by the DSF embedded sensors. The XGB model classifies and returns which Airflow path is approximated to be. Although there is the probability of the wrong estimation of the Airflow path, the estimate of the Thermal metric and Visual metric likely falls into the mutual area of values shared by the different Airflow path modes. Consequently, outputting expected `T_pred` is still doable by applying regression in the specific mode with Airflow Speed Controller in a later phase of the process.

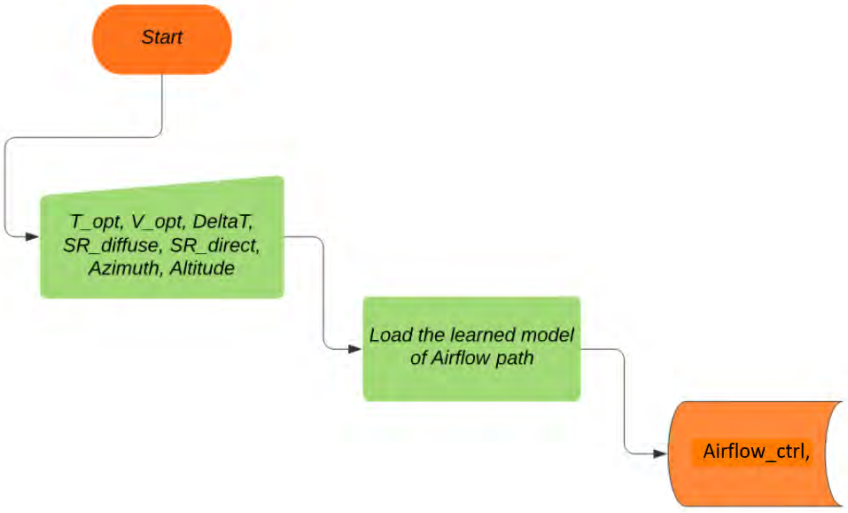



Figure 32: The workflow of Airflow Path Controller

### 5.5.4 Implementation of Airflow Speed Controller

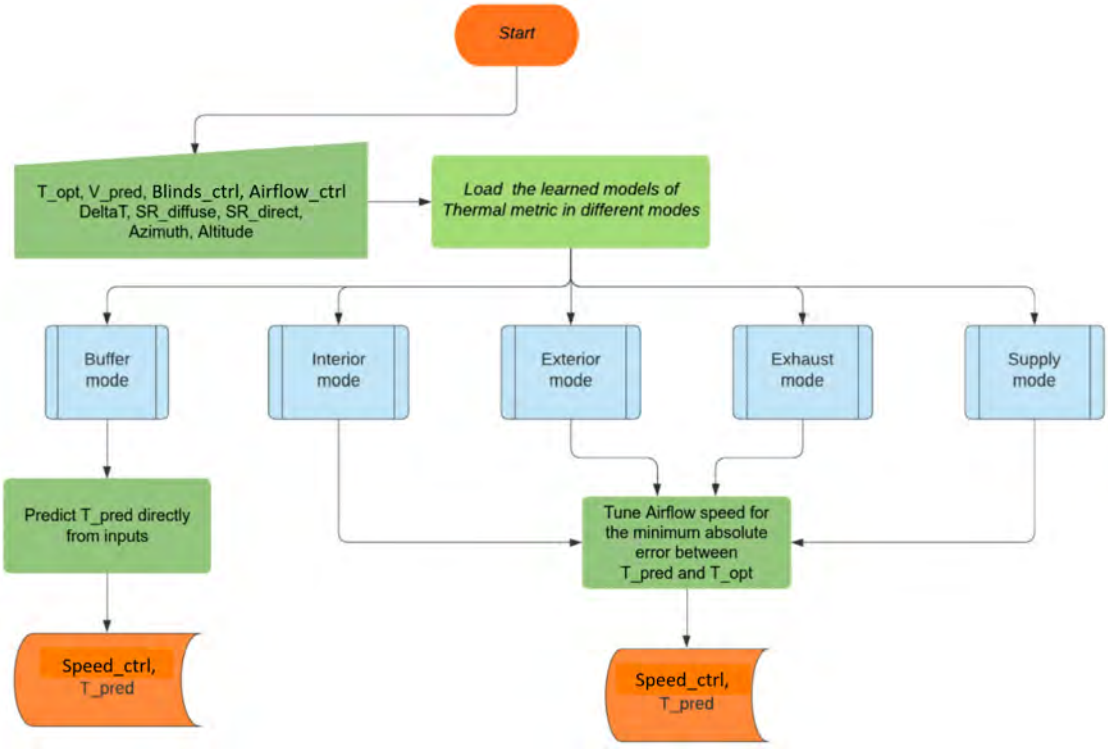



Figure 33: The workflow of Airflow Speed Controller

Figure 33 represents the working principles of the designed Airflow Speed Controller with the XGB models for Thermal metric in different modes as the operators. The predictor receives `T_opt` from the central controller, the earlier estimates `V_pred`

and `Blinds_ctrl` from the Blinds Angle Controller that is assumed to deliver the best `V_pred` as it did, and `Airflow_ctrl` from the Airflow Path Controller. The other inputs are readings from the embedded sensors of the direct solar radiation `SR_direct`, the diffuse solar radiation `SR_diffuse`, the azimuth angle `Azimuth`, and the altitude angle `Altitude`. The sensors also collect the indoor and outdoor ambient temperatures as (`T_in`) and (`T_out`)that is preprocessed for the final required input (`DeltaT = T_out - T_in`). The specific model is loaded for calculation of `T_pred` depending on which airflow path is estimated earlier. If the airflow path is expected as Buffer air mode, the XGB model for Thermal metric will be activated for prediction of Thermal metric. `T_pred` is returned together with a zero airflow speed to be set. Otherwise, with the airflow path predicted other than Buffer mode, the program loads the corresponding XGB model for Thermal metric in that specific mode and applies the same trick with the "filtering" algorithm as done in the Blinds Angle Controller. The airflow speed is tuned with the different values as wanted. The XGB model predicts and returns the corresponding values of Thermal metric. The program proceeds to pick the best one and output the target thermal metric `T_pred` and the estimate of airflow speed `Speed_ctrl`. In practice, the airflow speed is intended to be controlled at the scaled values between the defined minimum and maximum values with iteration by e.g. every 10% that can be translated into the power operates the fan speed.

## 5.6 Performance of the designed DSF control model with XGBoost

Figure 34 represents the results of prediction confused with actual categories of Airflow path, Airflow speed, and Blinds angle using the designed DSF control model. The Blinds Angle Controller delivered the tilt angles of the blinds close to the actual ones except returning 47% of the positive angles back to zero since the Blinds Angle Controller was programmed to set a zero angle for closing the blinds whenever Visual metric was estimated close to zero. To illustrate, the visual metric blocking ranges of the azimuth angle or the altitude cause almost no visual gain regardless of blinds angles. In addition, different blinds angles induce low thermal gain/loss of similar values as long as the azimuth and altitude angles are within the blocking ranges. Overall, the majority of prediction matches the actual classes. The air buffer mode was estimated by the accuracy of 85% to which the zero airflow speed corresponds. Otherwise, the airflow speed was adjusted depending on the predicted airflow mode that is different from the actual mode.

Table 7 summarizes the overall regression errors in the prediction of Visual metric and Thermal metric using the designed DSF control model. There are statistically about 40,000 samples out of 8 million was estimated with absolute errors more than 100 W/m$^2$ for Visual metric, and approximately 4,000/8,000,000 with absolute errors more than 50 W/m$^2$ for Thermal metric. The regression was gained with a high $R^2$ score of more than 0.999 for both Visual metric and Thermal metric. However, applying the prediction of Visual metric on unseen data raises the probability of extremely large errors that are unacceptable, leading to unrealistic range out of

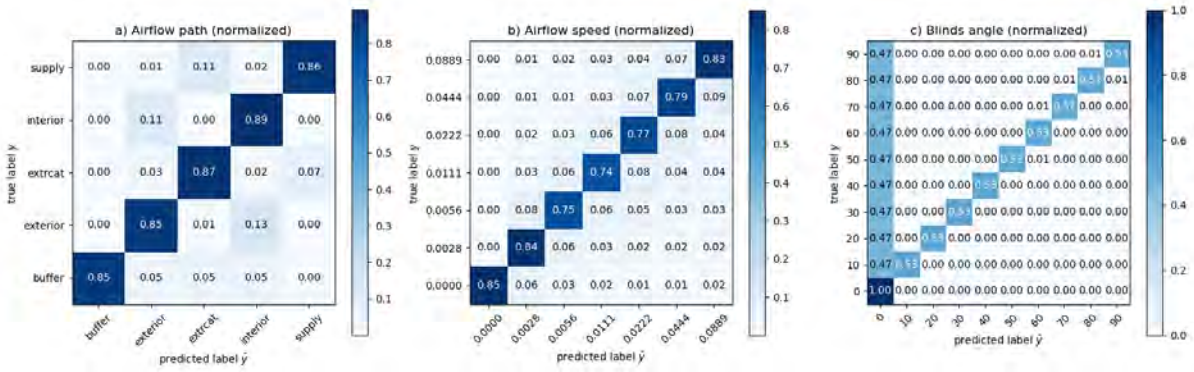



Figure 34: Confusion matrix of predicting Airflow path, Airflow speed, and Blinds angle by the designed DSF contol model

sample prediction. If the Visual metric errors are produced by the predictive (XGB) model itself, the estimation of the Thermal metric is subject to the correlated errors in the prediction of Airflow path, Visual metric, and Blinds angle. For example, about 5% of the estimates as Buffer mode are actual other modes. The data of other actual modes that were sorted as Buffer mode are unseen to the predictive (XGB) model of Thermal metric in air buffer mode, shortly Buffer model. Therefore, the error by Airflow path prediction was counted together with the estimate errors of Visual metric and Blinds angle. In the worst case where about 10% of estimates as Supply/Exhaust air mode is actual Exhaust/Supply air mode, the prediction error of Thermal metric reaches extremes of +/- 300 W/m$^2$ as listed and visualized in Table 8.

The regression errors are diverse on the unseen data, leading to significant instability or failure of producing reasonable predictions in the visual and thermal metrics returned by the controller when operating out of the range covered by training data. In addition to greatly possible overfitting, the cause for the poor performance of the controller is probably the black-box nature of the trained models without physical interpretation. However, ML can be improved with experience, which implies the more data seen by the model during training, the more improvement in model performance. As a solution, the controller can return a warning to ignore the inputs that do not allow an available choice of Blinds angle, Airflow path, and Airflow speed for a good prediction of the visual and thermal metrics within the confidence range of sample prediction or an acceptable limit of errors. Since the learned ML black-box models always deliver output in each prediction for any given values of the inputs, it is necessary to define a limit of acceptance for prediction errors between the input optimal values (`V_opt`, `T_opt`) and the estimated values (`V_pred`, `T_pred`). By the limit, the controller should simply return, for example, "NULL" for `V_pred`, `T_pred` signals, make no change to the current operational modes, and raise a warning to inform the central controller about `V_opt` and/or `T_opt` being unreachable. In that way, the model is also limited to the operating range of the training data. The control model can be constantly updated during the operational stage by collecting the ignored inputs to produce new data to retrain the model parameters. However,

the learning models were not taught with the labeled dataset to recognize the cases that `V_opt` and/or `T_opt` cannot be reached in the real world, and therefore extra information would be required to promote the model thinking.

Table 7: Prediction errors of Visual metric and Thermal metric by the designed DSF control model using XGBoost models

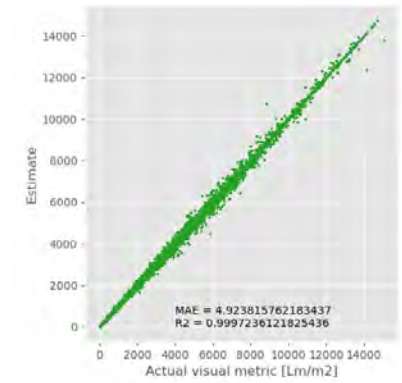
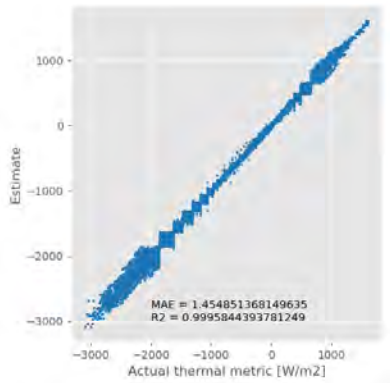
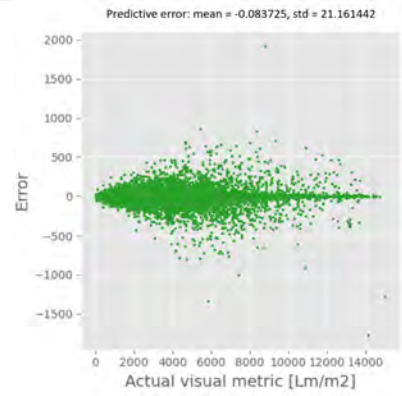
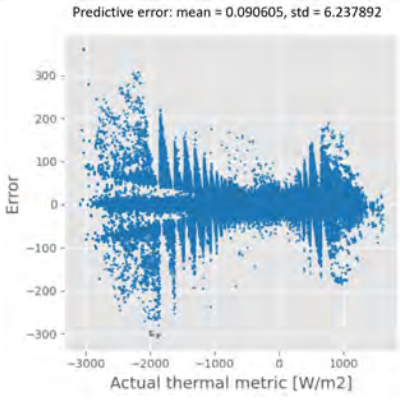
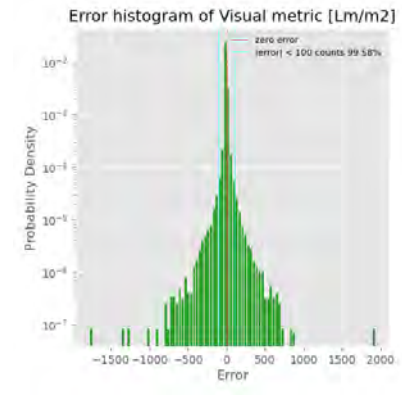
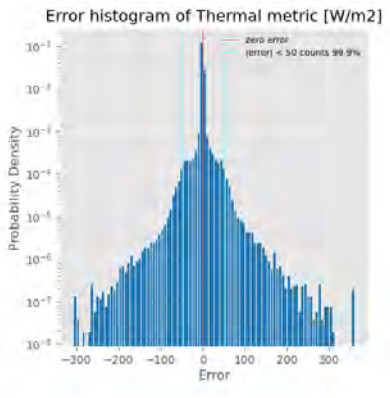
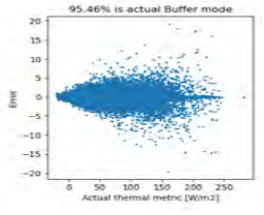
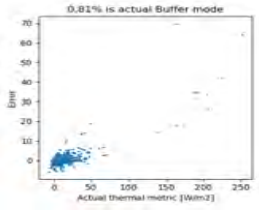
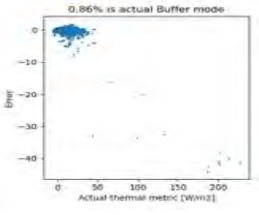
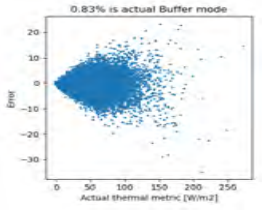
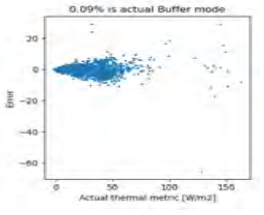
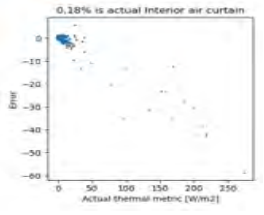
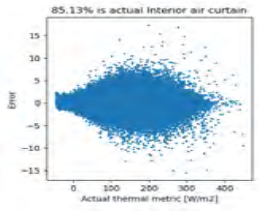
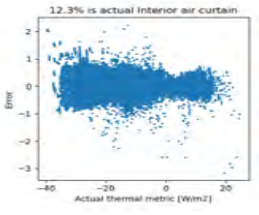
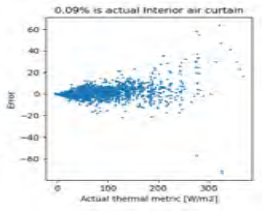
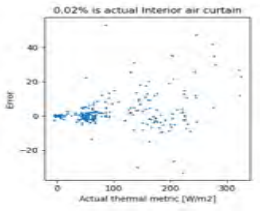
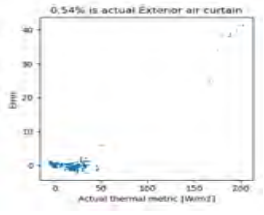
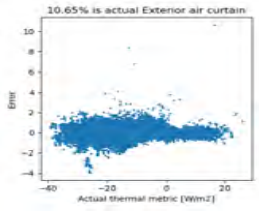
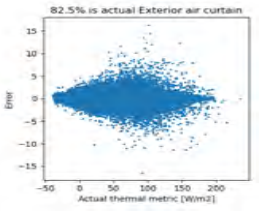
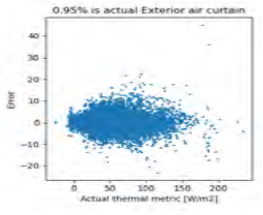
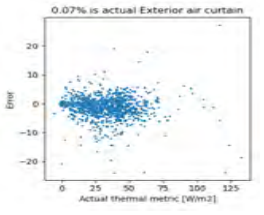
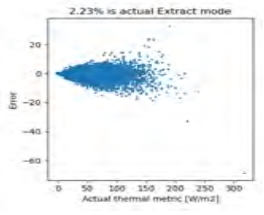
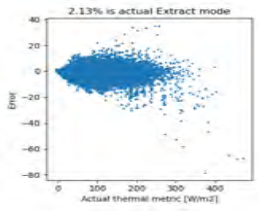
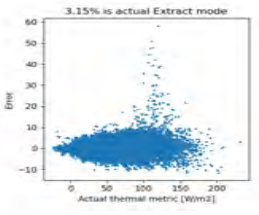
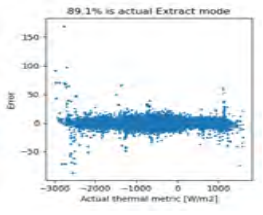
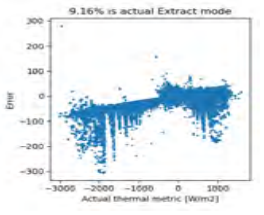
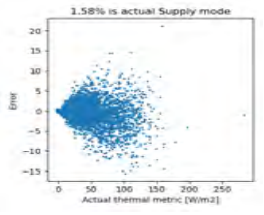
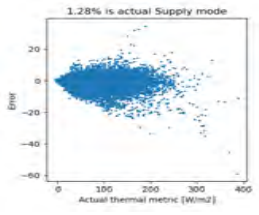
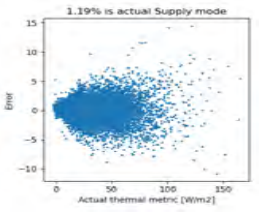
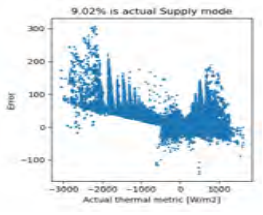
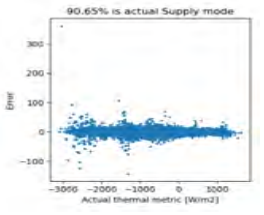
| | **Visual metric** | **Thermal metric** |
|---|---|---|
| **Estimates vs Actual values** |  |  |
| **Errors vs Actual values** |  |  |
| **Error histogram** |  |  |

Table 8: Analytical errors of Thermal metric prediction in different modes using the designed DSF control model

| | Predicted Air Buffer | Predicted Interior Air | Predicted Exterior Air | Predicted Exhaust Air | Predicted Supply Air |
|---|---|---|---|---|---|
| Actual Air Buffer |   |   |   |   |   |
| Actual Interior Air |   |   |   |   |   |
| Actual Exterior Air |   |   |   |   |   |
| Actual Exhaust Air |   |   |   |   |   |
| Actual Supply Air |   |   |   |   |   |

# 6 Future development

From the limitations of the black-box model toward gray-box modeling, it is possible to train hybrid or gray-box models for predicting Visual metric and Thermal metric instead of black-box models. As reviewed in Section 3.1, being predefined with simplified physical relations, the (physical) hybrid models can better performance over the training data and gain more performance stability on unseen data. It is also doable to train an ML-concerned gray-box model combining different learning models (e.g. ANN, RF/ET, GMB/XGB) using ensemble method, or to apply *online transfer learning* [36] that take advantage of *"online learning"* using online data stream for the data drift and growing very large over the time as well as *"transfer learning"* with model initialization in the new training or retraining process with the parameters of the pre-trained model. Another potential and more attractive approach are to apply RL to design a smart control model and compare for any improvement in model performance.

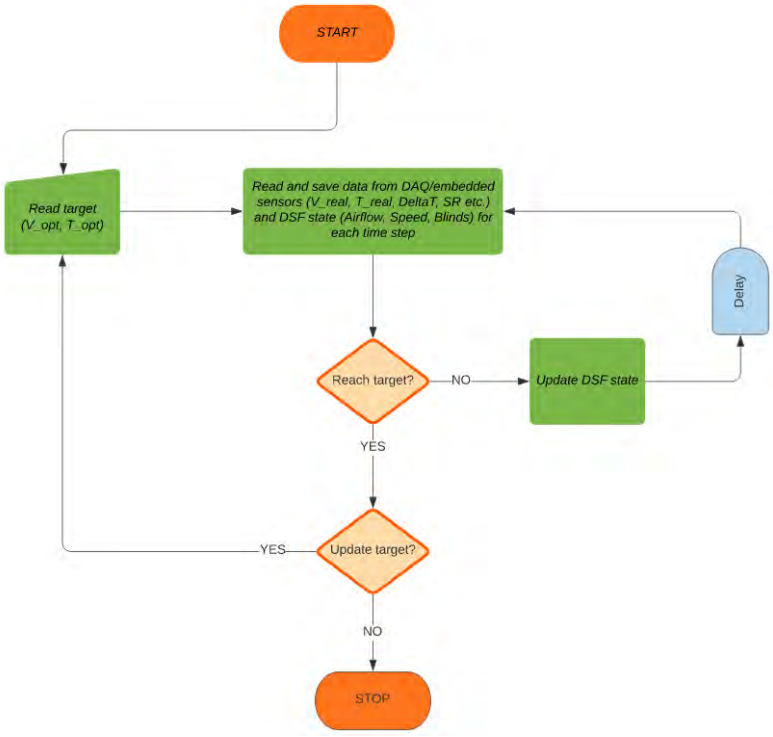



Figure 35: Data collection for training real-time models

As demonstrated in Figure 35, the control model can be also applied to collect data for training the real-time models when used as a part of the process that automatically updates the DSF state. The operational modes of Airflow path, Airflow speed, and Blinds angle are set by the DSF controller according to the inputs of `T_opt, V_opt` and dynamic boundary conditions required by the controller. Simulation values of the control inputs can be used to drive the controller whereas the airflow speed can be translated and stored as fan speed instead. Correspondingly, the real data

of Thermal metric, Visual metric, and boundary conditions read from the DAQ controller or the DSF embedded sensors are saved to the workstation together with the DSF state at each time step until the target is reached within a defined limit of errors, for example, `V_real = V_opt` +/- 100 W/m$^2$ and `T_real = T_opt` +/- 50 W/m$^2$. The target can be constantly updated to repeat the process until enough data are collected for training. The data collection process is however probably as slow as the period of up to a year though. Subsequently, the collected data that is real-time with thermal inertia and time-shift effect of the airflow modes on Thermal metric involved can be used to train time-series models using e.g. TDNN or XGBoost algorithm. A similar process can be also applied to RL. The learning agent is the DSF state updater with the operational modes as actions and boundary conditions as a learning environment. The agent can be concurrently collecting data and learning with pre-defined rewards/penalties on each action towards the target. Like DL with neural networks or other ML methods, RL can improve with experience. It takes time whereas the result is possibly expected with unforeseen success or failure unless undertaking the model till the end of the training process.

Related to model deployment or inference (making predictions on live data), there are available solutions of today's hardware and edge computing for ML when inference speed is a bottleneck on the target device. The continuous development of edge computing takes place with the explosive growth of IoT devices along with real-time applications of intensive computation. In Edge computing, computation and data storage are brought to edge devices or edge nodes, which allows a large amount of data to be processed and analyzed in real-time close to the data collection source rather than to be uploaded directly to the cloud or a centralized data processing system far away. Noteworthy, training or inference on the edge that can be offline has obvious advantages when using a highly available network with fast response time and minimal network latency, and cost savings in less data transmission bandwidth. [37]. As an example of hardware solutions for inference on the edge, benchmarking Tensorflow models on the new Raspberry Pi 4 (model B) using Google Coral's USB Accelerator packing Edge TPU (tensor processing unit) has been proved for remarkably faster speed than other available platforms [39], in which comparison in performance of the RPi 4 vs the older versions was also found interesting. However, continuously using RPi 4 and auxiliary devices like Coral USB Accelerator probably cause overheating problems over time, the solution to their thermal profiles such as a passive heat sink or an active cooling fan will need to be taken into account as well. When there is a need for online training/deploying large ML models, cloud computing may be considered to take in use instead, for example, Google Cloud TPU supports also PyTorch, scikit-learn, and XGBoost.

# 7 Conclusions

This thesis develops and demonstrates a methodology to apply ML to design a black-box model of the DSF that is flexible to control the thermal and visual aspects separately via dynamic operational modes. However, there are many feasible ML approaches to learn a control model for the DSF for better performance. Anyway, the work has been relatively fulfilled with the author's efforts and modest ML background.

A wrap-up to answer the first research question of how to apply ML to design the black-box control model of an active DSF is: (1) to review related work applying the most outperforming ML algorithms for black-box models such as ANN and ensembles of DTs using bagging method (RF/ET) or gradient boosting (XGBoost), (2) to do data analysis for proper ML problems stated to solve (regression of Visual metric and Thermal metric, classification of Airflow path), (3) to apply standards for model training and improving processes such as standardization, transformation, train-validation-test data split, hyperparameter tuning, regularization, (4) to define the criteria to evaluate and select the good performing models, (5) to combine the selected models to design controller components working logically, and (6) to define the accepted limit of application of the designed DSF controller. This methodology just provides an approach lacking realistic information on real data to assess for efficiency and reliability of the model in practical application though. Therefore, as an answer to the second research question, the reliability of the designed DSF control model relies on the performance of its component predictive models used. Each predictive model performance is evaluated according to the criteria including bias-variance trade-off, $R^2$ score for regression or accuracy of classification, and some user-defined criteria for model inference such as model size in the real world. These criteria are however to proceed to evaluate the model performance in the practice of model retraining and future works for training new models with available real-time data. To evaluate the overall performance of the designed DSF controller with XGBoost models, the classifier of Airflow modes is most critical as adding extra errors to other predictive models of thermal metrics. Meanwhile, the regressor of visual metric brings almost the same accuracy for the predictor of blind angle, regardless of the airflow mode predicted. The better performance of the predictive models, the higher quality of the component controllers individually, and the overall performance of the DSF controller generally. In brief, the DSF controller is as reliable as the training dataset of the predictive models used. The black-box models are limited to work fine in their knowledge and experience only.

In general, it is believed that ANN or any other ML black-box model will probably suffer from high variance in its inferences on new data due to significant and unavoidable disturbances added and data drift out of the range training data covered in real-time. Precise prediction of the DSF black-box control model for the thermal and visual metrics will be extremely challenging even if it can output the right operational modes but so wrong predicted the thermal and visual gains are unreliable. The proposed solution that may not fix completely model instability in real-time but can still improve the performance is to enlarge training data and retrain the black-box models applying online transfer learning. Finally, it is highly recommended

to proceed with the RL approach or hybrid models that are promising to overcome the limitations of black-box models.

In conclusion, the methodology represented to design the DSF black-box control model prepares an approach to plan DAQ for real data as well as enlarges the background for other approaches to develop possibly better control models of DSF.

# References


[1] Wikipedia. Curtain wall (architecture). [Accessed 30.11.2020]. [Online]. Available: https://en.wikipedia.org/wiki/Curtain_wall_(architecture).

[2] Mohammed, K. I., Alibaba, H. Z. Solar Control and Shading Strategies for Double Skin Facades in Hot Climate. *Imperial Journal of Interdisciplinary Research (IJIR),* 2018, vol. 4, p. 205–215.

[3] Streicher, W. (Editor). BESTFACADE, Best Practice for Double Skin Facades, EIE/04/135/S07.38652, WP 1 Report "State of the Art". Wien:., 2005.

[4] Chaudhary, G. Decoupling the thermal and visual performance in glazing systems: a novel methodology for the numerical investigation of the case of double skin facade systems. Master's Thesis, Norwegian University of Science and Technology, Norway, 2019.

[5] Lucchino, E. C., Goia, F., Lobaccaro, G. and Chaudhary, G. modeling of double skin facades in whole-building energy simulation tools: A review of current practices and possibilities for future developments. *Build Simul,* 2019, vol. 12, p. 3–27.

[6] Pomponi, F., Piroozfar, P. A. E., Southall, R., Ashton, P., Farr, E. R. P. Energy performance of Double-Skin Façades in temperate climates: A systematic review and meta-analysis. *Renewable and Sustainable Energy Reviews,* 2016, vol. 54, p. 1525–1536.

[7] Aznar, F., Echarri, V., Rizo, C. and Rizo, R. (2018) modeling the thermal behaviour of a building facade using deep learning. *PLoS ONE 13(12): e0207616,* https://doi.org/10.1371/journal.pone.0207616

[8] Bourdeau, M., Zhai, X. Q., Nefzaoui, E., Guo, X. and Chatellier, P. Modeling and forecasting building energy consumption: A review of data-driven techniques. *Sustainable Cities and Society,* 2019, vol. 48, p. 101533.

[9] Sarasti, J. Evaluation of load shifting potential in Espoo district heating network using a model predictive approach. Master's Thesis, Aalto University School of Engineering, Finland, 2017.

[10] Bhatia, A. Early Design Methodology for Energy Efficient Building Design. Ph.D. dissertation, International Institute of Information Technology, Hyderabad, India, 2019.

[11] Qolomany, B., Al-Fuqaha, A., Gupta, A., Benhaddou, D., Alwajidi, S., Qadir, J. and Fong, A. C. Leveraging Machine Learning and Big Data for Smart Buildings: A Comprehensive Survey. *IEEE Access,* 2019, vol. 7, p. 90316–90356. DOI: 10.1109/ACCESS.2019.2926642.

[12] Brownlee, J. What is Machine Learning? [Accessed 2.10.2020]. [Online]. Available: https://machinelearningmastery.com/what-is-machine-learning/#:~:text=Machine%20Learning%20is%20the%20training,decision%20against%20a%20performance%20measure.

[13] Silva, M. Statistical Modeling - The Pragmatic Guide, Part 2 - Interpreting Machine Learning Models. [Accessed 20.12.2020]. [Online]. Available: https://towardsdatascience.com/statistical-modeling-the-full-pragmatic-guide-7aeb56e38b36

[14] Tyagi, N. Defining Predictive Modeling in Machine Learning. [Accessed 7.12.2020]. [Online]. Available: https://medium.com/analytics-steps/defining-predictive-modeling-in-machine-learning-887c23b7a278.

[15] Pintelas, E., Livieris, I. E. and Pintelas, P. A. Grey-Box Ensemble Model Exploiting Black-Box Accuracy and White-Box Intrinsic Interpretability. *Algorithms,* 2020, vol. 13, no. 1:17.

[16] Rashidi, H. H., Tran, N. K., Betts, E. V., Howell, L. P. and Green, R. Artificial Intelligence and Machine Learning in Pathology: The Present Landscape of Supervised Methods. *Academic Pathology,* 2019, vol. 6. DOI: 10.1177/2374289519873088

[17] Zhu, X. Semi-Supervised Learning with Graphs. Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, Pennsylvania, 2005.

[18] Scikit-Learn. Documentation of Scikit-Learn 0.23.2. [Accessed 3.11.2020]. [Online]. Available: https://scikit-learn.org/0.23/.

[19] Binz, K. Intro to Regularization [Online]. Available: https://kevinbinz.com/category/latex/.

[20] Dabbura, I. Coding Neural Network - Regularization. [Accessed 1.11.2020]. [Online]. Available: https://towardsdatascience.com/coding-neural-network-regularization-43d26655982d

[21] Brownlee, J. Bagging and Random Forest Ensemble Algorithms for Machine Learning. [Accessed 9.12.2020]. [Online]. Available: https://machinelearningmastery.com/bagging-and-random-forest-ensemble-algorithms-for-machine-learning/.

[22] Yazici, M. T., Basurra, S. and Gaber, M. M. Edge Machine Learning: Enabling Smart Internet of Things Applications. *Big Data Cogn. Comput.,* 2018, vol. 2, no. 3:26.

[23] Brownlee, J. How to Develop an Extra Trees Ensemble with Python. [Accessed 10.12.2020]. [Online]. Available: https://machinelearningmastery.com/extra-trees-ensemble-with-python/.

[24] Brownlee, J. Boosting and AdaBoost for Machine Learning. [Accessed 15.12.2020]. [Online]. Available: https://machinelearningmastery.com/boosting-and-adaboost-for-machine-learning/.

[25] Brownlee, J. Gradient Boosting with Scikit-Learn, XGBoost, LightGBM, and CatBoost. [Accessed 10.12.2020]. [Online]. Available: https://machinelearningmastery.com/gradient-boosting-with-scikit-learn-xgboost-lightgbm-and-catboost/.

[26] Saha, S. Understanding the log loss function of XGBoost. [Accessed 21.12.2020]. [Online]. Available: https://medium.com/datadriveninvestor/understanding-the-log-loss-function-of-xgboost-8842e99d975d.

[27] XGBoost Documentation. [Accessed 15.11.2020]. [Online]. Available: https://xgboost.readthedocs.io/en/latest/index.html

[28] Mavuduru, A. Why XGBoost can't solve all your problems. [Accessed 25.12.2020]. [Online]. Available: https://towardsdatascience.com/why-xgboost-cant-solve-all-your-problems-b5003a62d12a

[29] Fang, W., Chen, C., Song, B., Wang, L., Zhou, J. and Zhu, K. Q. Adapted tree boosting for Transfer Learning. arXiv:2002.11982v2 [cs.LG]

[30] Brownlee, J. A Gentle Introduction to XGBoost for Applied Machine Learning. [Accessed 1.11.2020]. [Online]. Available: https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/.

[31] Brownlee, J. A Gentle Introduction to the Rectified Linear Unit (ReLU). [Accessed 3.11.2020]. [Online]. Available: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/.

[32] Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization. arXiv:1412.6980v9 [cs.LG]

[33] Brownlee, J. Crash Course On Multi-Layer Perceptron Neural Networks. [Accessed 5.10.2020]. [Online]. Available: https://machinelearningmastery.com/neural-networks-crash-course/.

[34] Brownlee, J. A Gentle Introduction to Dropout for Regularizing Deep Neural Networks. [Accessed 13.10.2020]. [Online]. Available: https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/.

[35] Brownlee, J. A Gentle Introduction to Information Entropy. [Accessed 11.12.2020]. [Online]. Available: https://machinelearningmastery.com/what-is-information-entropy/.

[36] Zhao, P., Hoi, S. C. H., Wang, J. and Li, B. Online Transfer Learning. *Artificial Intelligence,* 2014, vol. 216, p. 76–102.

[37] Ai, Y., Beng, M. and Zhang, K. Edge computing technologies for Internet of Things. *Digital Communications and Networks,* 2018, vol. 4, p. 77–86.

[38] Zhao, P., Hoi, S. C. H., Wang, J. and Li, B. Online Transfer Learning. *Artificial Intelligence,* 2014, vol. 216, p. 76–102.

[39] Allan, A. Benchmarking Machine Learning on the New Raspberry Pi 4, Model B. [Accessed 18.12.2020]. [Online]. Available: https://www.hackster.io/news/benchmarking-machine-learning-on-the-new-raspberry-pi-4-model-b-88db9304ce4.

# A Training Visual metric XGB model

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold, GridSearchCV
from sklearn.pipeline import Pipeline
from xgboost import XGBRegressor
from sklearn.compose import TransformedTargetRegressor
from sklearn.metrics import r2_score, mean_absolute_error
from pickle import dump
import time
start_time = time.time()


# fix random seed for reproducibility
seed = 7
np.random.seed(seed)

# Load the dataset
source = 'data/high_blinds_on_visual.csv'
data = pd.read_csv(source, float_precision='high')
print('Dataset:',data.shape)

# Split train & test datasets
features = ['Azimuth','Altitude','SR_diffuse','SR_direct','Blinds']
label ='Visual'
X = data[features].values
y = data[label].values
validation_size = 0.33
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=validation_size, random_state=seed)

model_fname = 'models/visual_XGB.pkl'

TUNE = 0
TRAIN = 0
SAVE = 0

if TUNE:
  # Standardize the dataset
  scaler_X= StandardScaler().fit(X_train)
  scaledX_train = scaler_X.transform(X_train)
  scaledX_test = scaler_X.transform(X_test)
  scaledX = scaler_X.transform(X)
  scaler_Y= StandardScaler().fit(y_train.reshape(-1,1))
  scaledY_train = scaler_Y.transform(y_train.reshape(-1,1))
  scaledY_test = scaler_Y.transform(y_test.reshape(-1,1))
  scaledY = scaler_Y.transform(y.reshape(-1,1))

  params = {
    'learning_rate': [0.3],
```

```
    'max_depth': [10], #range(5,30,5)
    'n_estimators': range(50, 450, 50)
    #'gamma': range(0,1,2),
    #'alpha': [0,0.001,0.01,0.1],
    #'lambda': [0,0.001,0.01,0.1],
    }
  model = xgb.XGBRegressor(tree_method='gpu_hist',
        predictor='gpu_predictor',
        random_state=seed)
  kfold = KFold(n_splits=3, shuffle=True, random_state=seed)
  grid_search = GridSearchCV(model, params,
        scoring='neg_mean_squared_error',
        n_jobs=-1, cv=kfold)
  grid_result = grid_search.fit(scaledX_train, scaledY_train)
  # summarize results
  print("Best: %f using %s" % (grid_result.best_score_,
          grid_result.best_params_))
  means = grid_result.cv_results_['mean_test_score']
  stds = grid_result.cv_results_['std_test_score']
  params = grid_result.cv_results_['params']
  for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))

  # plot
  DEPTH = 0 # 1, n_estimators = 100 (defaul)
  if DEPTH:
    param=range(5, 30, 5)
    plt.errorbar(param, means, yerr=stds)
    plt.title("XGB max_depth vs NMSE")
    plt.xlabel('max_depth')
    plt.ylabel('NMSE')
    plt.show()
  TREES = 0 # 1, max_depth = 10
  if TREES:
    param=range(50, 450, 50)
    plt.errorbar(param, means, yerr=stds)
    plt.title("XGB n_estimators vs NMSE")
    plt.xlabel('n_estimators')
    plt.ylabel('NMSE')
    plt.show()

if TRAIN:
  # prepare the model with input scaling
  pipeline = Pipeline(steps=[('trans', StandardScaler()),
    ('model', XGBRegressor(tree_method='gpu_hist',
        predictor='gpu_predictor',
        learning_rate=0.3,
        max_depth=10, n_estimators=300,
        random_state=seed))])

  # prepare the model with target scaling
  model = TransformedTargetRegressor(regressor=pipeline,
        transformer=StandardScaler())

```

```
107   # fit model on training data
108   model.fit(X_train,y_train)
109   print(model.regressor['model'].get_xgb_params())
110
111   mae_train = mean_absolute_error(y_train,
112           model.predict(X_train))
113   r2_train = model.score(X_train,y_train)
114   mae_test = mean_absolute_error(y_test,
115           model.predict(X_test))
116   r2_test = model.score(X_test,y_test)
117   mae = mean_absolute_error(y, model.predict(X))
118   r2 = model.score(X,y)
119   print('Train: ', mae_train, r2_train)
120   print('Test: ', mae_test, r2_test)
121   print('Overall: ', mae, r2)
122
123   if SAVE:
124     # save the model
125     dump(model, open(model_fname,'wb'))
126
127 print("--- Done in %s seconds ---" % (time.time() - start_time))
```

# B Training Visual metric ETR model

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold, GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.compose import TransformedTargetRegressor
from sklearn.metrics import r2_score, mean_absolute_error
from pickle import dump
import time
start_time = time.time()


# fix random seed for reproducibility
seed = 7
np.random.seed(seed)

# Load the dataset
source = 'data/high_blinds_on_visual.csv'
data = pd.read_csv(source, float_precision='high')
print('Dataset:',data.shape)

# Split train & test datasets
features = ['Azimuth','Altitude','SR_diffuse','SR_direct','Blinds']
label ='Visual'
X = data[features].values
y = data[label].values
validation_size = 0.33
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=validation_size, random_state=seed)

model_fname = 'models/visual_ETR.pkl'

TUNE = 0
TRAIN = 0
SAVE = 0

if TUNE:
  # Standardize the dataset
  scaler_X= StandardScaler().fit(X_train)
  scaledX_train = scaler_X.transform(X_train)
  scaledX_test = scaler_X.transform(X_test)
  scaledX = scaler_X.transform(X)
  scaler_Y= StandardScaler().fit(y_train.reshape(-1,1))
  scaledY_train = scaler_Y.transform(y_train.reshape(-1,1))
  scaledY_test = scaler_Y.transform(y_test.reshape(-1,1))
  scaledY = scaler_Y.transform(y.reshape(-1,1))

  params = {
    'max_depth': [25], #range(10,45,5),
```

```
    'n_estimators': range(50, 550, 50)
    #'ccp_alpha': [0,0.000001,0.00001,0.0001]
    #'min_samples_split': range(2,12,2)
    }
  model = ExtraTreesRegressor(n_jobs=24, random_state=seed)
  kfold = KFold(n_splits=3, shuffle=True, random_state=seed)
  grid_search = GridSearchCV(model, params,
      scoring='neg_mean_squared_error',
      n_jobs=-1, cv=kfold)
  grid_result = grid_search.fit(scaledX_train,
          scaledY_train.ravel())
  # summarize results
  print("Best: %f using %s" % (grid_result.best_score_,
          grid_result.best_params_))
  means = grid_result.cv_results_['mean_test_score']
  stds = grid_result.cv_results_['std_test_score']
  params = grid_result.cv_results_['params']
  for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))

  # plot
  DEPTH = 0 # 1, n_estimators = 100 default
  if DEPTH:
    param=range(10, 45, 5)
    plt.errorbar(param, means, yerr=stds)
    plt.title("ETR max_depth vs NMSE")
    plt.xlabel('max_depth')
    plt.ylabel('NMSE')
    plt.show()
  TREES = 0 # 1, max_depth = 25
  if TREES:
    param=range(50, 550, 50)
    plt.errorbar(param, means, yerr=stds)
    plt.title("ETR n_estimators vs NMSE")
    plt.xlabel('n_estimators')
    plt.ylabel('NMSE')
    plt.show()

if TRAIN:
  # prepare the model with input scaling
  pipeline = Pipeline(steps=[('trans', StandardScaler()),
      ('model', ExtraTreesRegressor(n_jobs=24,
      max_depth=25, ccp_alpha=0.0000001,
      random_state=seed))])

  # prepare the model with target scaling
  model = TransformedTargetRegressor(regressor=pipeline,
        transformer=StandardScaler())

  model.fit(X_train,y_train)
  print(model.regressor['model'].get_params())
  mae_train = mean_absolute_error(y_train, model.predict(X_train))
  r2_train = model.score(X_train,y_train)
  mae_test = mean_absolute_error(y_test, model.predict(X_test))
```

```
107   r2_test = model.score(X_test,y_test)
108   mae = mean_absolute_error(y, model.predict(X))
109   r2 = model.score(X,y)
110   print('Train: ', mae_train, r2_train)
111   print('Test: ', mae_test, r2_test)
112   print('Overall: ', mae, r2)
113
114   if SAVE:
115     # save the model
116     dump(model, open(model_fname,'wb'))
117
118 print("--- Done in %s seconds ---" % (time.time() - start_time))
```

# C Training Visual metric ANN model

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split,
from sklearn.model_selection import KFold, GridSearchCV
from sklearn.preprocessing import StandardScaler
from keras.wrappers.scikit_learn import KerasRegressor
from keras.models import Sequential
from keras.layers import Dense, Dropout
from sklearn.metrics import mean_absolute_error, r2_score
from keras.callbacks import ModelCheckpoint
from keras.utils.vis_utils import plot_model
from keras.models import model_from_json
from keras import regularizers
from pickle import dump
import time
start_time = time.time()

def build_model(n_inputs, n_outputs=1, l2=0.0001,
    layers=(1000,500), init='normal'):
  model = Sequential()
  model.add(Dense(layers[0], input_dim=n_inputs,
      kernel_regularizer=regularizers.l2(l2),
      kernel_initializer=init, activation='relu'))
  model.add(Dropout(0.5))
  model.add(Dense(layers[1],
      kernel_regularizer=regularizers.l2(l2),
      kernel_initializer=init, activation='relu'))
  model.add(Dropout(0.5))
  model.add(Dense(n_outputs, kernel_initializer=init))
  model.compile(loss='mse', optimizer='adam', metrics=['mae'])
  return model

def save_model(model, struct_fname):
  # saving model
  json_model = model.to_json()
  json_file = open(struct_fname, 'w')
  json_file.write(json_model)
  json_file.close()
  print('Saved model to disk')

def load_model(struct_fname, weights_fname):
  # loading model
  json_file = open(struct_fname, 'r')
  model = model_from_json(json_file.read())
  json_file.close()
  model.load_weights(weights_fname)
  model.compile(loss='mse', optimizer='adam', metrics=['mae'])
  print("Loaded model from disk")
  return model


```

```
# fix random seed for reproducibility
seed = 7
np.random.seed(seed)

# Load the dataset
source = 'data/high_blinds_on_visual.csv'
data = pd.read_csv(source, float_precision='high')
print('Dataset:',data.shape)

# Split train & test datasets
features = ['Azimuth','Altitude','SR_diffuse','SR_direct','Blinds']
label ='Visual'
X = data[features].values
y = data[label].values
validation_size = 0.33
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=validation_size, random_state=seed)

model_fname = 'models/visual_ETR.pkl'

TUNE = 0
TRAIN = 0
SAVE = 0

if TUNE:
  # Standardize the dataset
  scaler_X= StandardScaler().fit(X_train)
  scaledX_train = scaler_X.transform(X_train)
  scaledX_test = scaler_X.transform(X_test)
  scaledX = scaler_X.transform(X)
  scaler_Y= StandardScaler().fit(y_train.reshape(-1,1))
  scaledY_train = scaler_Y.transform(y_train.reshape(-1,1))
  scaledY_test = scaler_Y.transform(y_test.reshape(-1,1))
  scaledY = scaler_Y.transform(y.reshape(-1,1))

  model = KerasRegressor(build_fn=build_model, verbose=0)
  # grid search epochs, batch size and optimizer
  layers = [(100,50),(200,100),(600,300),(800,400),
      (1000,500),(1200,600),(1400,700),(1600,800)]
  #layers = [(1200,600)]
  #l2 = [0.0001, 0.001, 0.01, 0.1]
  l2 = [0.0001]
  #epochs = [200, 400, 600, 800, 1000]
  epochs = [100]
  #batches = [1000, 10000, 100000, 300000]
  batches = [1000]
  n_inputs = [scaledX.shape[1]]
  kfold = KFold(n_splits=3, shuffle=True, random_state=seed)
  param_grid = dict(n_inputs=n_inputs, l2=l2, layers=layers,
      epochs=epochs, batch_size=batches, verbose=[1])
  grid = GridSearchCV(estimator=model, param_grid=param_grid,
    scoring='neg_mean_squared_error', cv=kfold, n_jobs=-1)
  grid_result = grid.fit(scaledX_train, scaledY_train)
  # summarize results
```

```
107   print("Best: %f using %s" % (grid_result.best_score_,
108           grid_result.best_params_))
109   means = grid_result.cv_results_['mean_test_score']
110   stds = grid_result.cv_results_['std_test_score']
111   params = grid_result.cv_results_['params']
112   for mean, stdev, param in zip(means, stds, params):
113     print("%f (%f) with: %r" % (mean, stdev, param))
114
115   # plot
116   LAYERS = 1
117   if LAYERS:
118     param=range(8)
119     plt.errorbar(param, means, yerr=stds)
120     plt.xticks(param,layers,fontsize=8)
121     plt.title("ANN hidden layers vs NMSE")
122     plt.xlabel('Hidden layers')
123     plt.ylabel('NMSE')
124     plt.show()
125     # plot
126   L2 = 0
127   if L2:
128     param=range(4)
129     plt.errorbar(param, means, yerr=stds)
130     plt.xticks(param,l2,fontsize=8)
131     plt.title("ANN l2 regularization vs NMSE")
132     plt.xlabel('l2')
133     plt.ylabel('NMSE')
134     plt.show()
135   BATCHES = 0
136   if BATCHES:
137     param=batches
138     plt.errorbar(param, means, yerr=stds)
139     plt.title("ANN batch size vs NMSE")
140     plt.xlabel('Batch size')
141     plt.ylabel('NMSE')
142     plt.show()
143   EPOCHS = 0
144   if EPOCHS:
145     param=epochs
146     plt.errorbar(param, means, yerr=stds)
147     plt.title("ANN epochs vs NMSE")
148     plt.xlabel('epochs')
149     plt.ylabel('NMSE')
150     plt.show()
151
152 if TRAIN:
153   # Standardize the dataset
154   scaler_X= StandardScaler().fit(X_train)
155   scaledX_train = scaler_X.transform(X_train)
156   scaledX_test = scaler_X.transform(X_test)
157   scaledX = scaler_X.transform(X)
158   scaler_Y= StandardScaler().fit(y_train.reshape(-1,1))
159   scaledY_train = scaler_Y.transform(y_train.reshape(-1,1))
160   scaledY_test = scaler_Y.transform(y_test.reshape(-1,1))
```

```
161   scaledY = scaler_Y.transform(y.reshape(-1,1))
162
163   # modelling and save training history
164   batch_size = 300000
165   epochs = 10000
166   n_inputs = scaledX.shape[1]
167   model = build_model(n_inputs)
168
169   # checkpoint
170   weights_fname='models/visual.weights.best.hdf5'
171   checkpoint = ModelCheckpoint(weights_fname, monitor='mae',
172         verbose=0, save_weights_only=True,
173         save_best_only=True, mode='min')
174   callbacks_list = [checkpoint]
175   hist = model.fit(scaledX_train, scaledY_train,
176       validation_split=validation_size,
177       epochs=epochs, batch_size=batch_size,
178       callbacks=callbacks_list, verbose=1)
179
180   # print the scores
181   score = model.evaluate(scaledX_test, scaledY_test, verbose=0)
182   print('Test score = ',score)
183   R2 = r2_score(scaledY, model.predict(scaledX))
184   print('Overall r2_score = ',R2)
185
186   if SAVE:
187     struct_fname = 'models/visual.struct.json'
188     save_model(model, struct_fname)
189     dump(scaler_X, open('models/visual.scalerX.pkl','wb'))
190     dump(scaler_Y, open('models/visual.scalerY.pkl','wb'))
191     history_fname = 'models/visual.history.pkl'
192     dump(hist.history, open(history_fname,'wb'))
193
194 print("--- Done in %s seconds ---" % (time.time() - start_time))
```

# D Training Airflow path XGB model

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import itertools
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from xgboost import XGBClassifier
import xgboost as xgb
from sklearn.metrics import accuracy_score, confusion_matrix
from pickle import dump, load
import pickle
import time
start_time = time.time()

print(xgb.__version__)
print(pickle.format_version)
print(xgb.Booster.load_model.__doc__)

def visualize_cm(cm, fname='no'):
    """
    Function visualizes a confusion matrix with and without
    normalization
    """
    plt.rc('legend', fontsize=10)
    plt.rc('axes', labelsize=10)
    plt.rc('xtick', labelsize=10)
    plt.rc('ytick', labelsize=10)


    fig, axes = plt.subplots(1, 2,figsize=(10,5))

    im1 = axes[0].imshow(cm, interpolation='nearest', cmap=plt.cm.
    Blues)
    fig.colorbar(im1, ax=axes[0])
    classes = ['buffer mode','exterior mode','extract mode',
            'interior mode','supply mode']
    tick_marks = np.arange(len(classes))
    axes[0].set_xticks(tick_marks)
    axes[0].set_xticklabels(classes,rotation=45)
    axes[0].set_yticks(tick_marks)
    axes[0].set_yticklabels(classes)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.
    shape[1])):
        axes[0].text(j, i, format(cm[i, j], 'd'),
                 horizontalalignment="center",
                 verticalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    axes[0].set_xlabel('predicted label $\hat{y}$')
    axes[0].set_ylabel('true label $y$')
```

```
50      axes[0].set_title('Without normalization')
51
52      cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
53      im2 = axes[1].imshow(cm, interpolation='nearest', cmap=plt.cm.
        Blues)
54      fig.colorbar(im2, ax=axes[1])
55
56      axes[1].set_xticks(tick_marks)
57      axes[1].set_xticklabels(classes,rotation=45)
58      axes[1].set_yticks(tick_marks)
59      axes[1].set_yticklabels(classes)
60
61      thresh = cm.max() / 2.
62      for i, j in itertools.product(range(cm.shape[0]), range(cm.
        shape[1])):
63          axes[1].text(j, i, format(cm[i, j], '.2f'),
64                   horizontalalignment="center",
65                  verticalalignment="center",
66                  color="white" if cm[i, j] > thresh else "black")
67
68      axes[1].set_xlabel('predicted label $\hat{y}$')
69      axes[1].set_ylabel('true label $y$')
70      axes[1].set_title('Normalized')
71
72      axes[0].set_ylim(-0.5,4.5)
73      axes[1].set_ylim(-0.5,4.5)
74
75      plt.tight_layout()
76      if not fname == 'no':
77        plt.savefig(fname)
78      plt.show()
79      percents = np.array([])
80      for i in range(5):
81          percents = np.append(percents,round(cm[i][i]*100,1))
82      return percents
83
84  # fix random seed for reproducibility
85 seed = 7
86 np.random.seed(seed)
87
88 # Load the dataset
89 source = 'data/high_blinds_on_thermal.csv'
90 data = pd.read_csv(source, float_precision='high')
91 print('Dataset:',data.shape)
92 features = ['DeltaT', 'Azimuth', 'Altitude', 'SR_diffuse',
93        'SR_direct', 'Visual', 'Thermal']
94 label = 'Airflow_path'
95 X = data[features].values
96 y = data[label].values
97
98 # Split train & test datasets
99 validation_size = 0.33
100 X_train, X_test, y_train, y_test = train_test_split(X, y,
101     test_size=validation_size, random_state=seed)
```

```
102
103
104 TRAIN = 0
105 SAVE = 0
106 if TRAIN:
107   # encode class values as integers
108   encoder = LabelEncoder().fit(y_train)
109   encodedY = encoder.transform(y)
110   encodedY_train = encoder.transform(y_train)
111   encodedY_test = encoder.transform(y_test)
112
113   # Standardize the dataset
114   scaler = StandardScaler().fit(X_train)
115   scaledX_train = scaler.transform(X_train)
116   scaledX_test = scaler.transform(X_test)
117   scaledX = scaler.transform(X)
118
119   # fit model on training data
120
121   model = XGBClassifier(tree_method='gpu_hist',
122         predictor='gpu_predictor',
123         learning_rate=0.3,
124         max_depth=10,
125         n_estimators=300,
126         objective='multi:softprob',
127         random_state=seed)
128   model.fit(scaledX_train, encodedY_train)
129   print(model)
130
131   # save the model
132   if SAVE:
133     #dump(model, open( 'airflow/airflow_XGB.pkl' , 'wb' ))
134     model.save_model('airflow/airflow_XGB.model')
135     # save the scaler
136     dump(scaler, open( 'airflow/airflow_scaler_XGB.pkl' , 'wb' ))
137     # save the encoder
138     dump(encoder, open( 'airflow/airflow_encoder_XGB.pkl' , 'wb' ))
139
140 print("--- Done in %s seconds ---" % (time.time() - start_time))
```

# E Training Airflow path RF model

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
#from cuml import RandomForestClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score, precision_score,
    confusion_matrix
import itertools
from pickle import dump, load
import time
start_time = time.time()

def visualize_cm(cm, fname='no'):
    """
    Function visualizes a confusion matrix with and without
    normalization
    """
    plt.rc('legend', fontsize=10)
    plt.rc('axes', labelsize=10)
    plt.rc('xtick', labelsize=10)
    plt.rc('ytick', labelsize=10)


    fig, axes = plt.subplots(1, 2,figsize=(10,5))

    im1 = axes[0].imshow(cm, interpolation='nearest', cmap=plt.cm.
    Blues)
    fig.colorbar(im1, ax=axes[0])
    classes = ['buffer mode','exterior mode','extract mode',
            'interior mode','supply mode']
    tick_marks = np.arange(len(classes))
    axes[0].set_xticks(tick_marks)
    axes[0].set_xticklabels(classes,rotation=45)
    axes[0].set_yticks(tick_marks)
    axes[0].set_yticklabels(classes)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.
    shape[1])):
        axes[0].text(j, i, format(cm[i, j], 'd'),
                 horizontalalignment="center",
                 verticalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    axes[0].set_xlabel('predicted label $\hat{y}$')
    axes[0].set_ylabel('true label $y$')
    axes[0].set_title('Without normalization')

    cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
```

```
49      im2 = axes[1].imshow(cm, interpolation='nearest', cmap=plt.cm.
    Blues)
50      fig.colorbar(im2, ax=axes[1])
51
52      axes[1].set_xticks(tick_marks)
53      axes[1].set_xticklabels(classes,rotation=45)
54      axes[1].set_yticks(tick_marks)
55      axes[1].set_yticklabels(classes)
56
57      thresh = cm.max() / 2.
58      for i, j in itertools.product(range(cm.shape[0]), range(cm.
    shape[1])):
59          axes[1].text(j, i, format(cm[i, j], '.2f'),
60                   horizontalalignment="center",
61                  verticalalignment="center",
62                  color="white" if cm[i, j] > thresh else "black")
63
64      axes[1].set_xlabel('predicted label $\hat{y}$')
65      axes[1].set_ylabel('true label $y$')
66      axes[1].set_title('Normalized')
67
68      axes[0].set_ylim(-0.5,4.5)
69      axes[1].set_ylim(-0.5,4.5)
70
71      plt.tight_layout()
72      if not fname == 'no':
73        plt.savefig(fname)
74      plt.show()
75      percents = np.array([])
76      for i in range(5):
77          percents = np.append(percents,round(cm[i][i]*100,1))
78      return percents
79
80  # fix random seed for reproducibility
81 seed = 7
82 np.random.seed(seed)
83
84 # Load the dataset
85 source = 'data/high_blinds_on_thermal.csv'
86 data = pd.read_csv(source, float_precision='high')
87 print('Dataset:',data.shape)
88 features = ['DeltaT', 'Azimuth', 'Altitude', 'SR_diffuse',
89       'SR_direct', 'Visual', 'Thermal']
90 label = 'Airflow_path'
91 X = data[features].values
92 y = data[label].values
93
94 # Split train & test datasets
95 validation_size = 0.33
96 X_train, X_test, y_train, y_test = train_test_split(X, y,
97      test_size=validation_size, random_state=seed)
98
99
100 TRAIN = 0
```

```
SAVE = 0
if TRAIN:
  # encode class values as integers
  encoder = LabelEncoder().fit(y_train)
  encodedY = encoder.transform(y)
  encodedY_train = encoder.transform(y_train)
  encodedY_test = encoder.transform(y_test)

  # Standardize the dataset
  scaler = StandardScaler().fit(X_train)
  scaledX_train = scaler.transform(X_train)
  scaledX_test = scaler.transform(X_test)
  scaledX = scaler.transform(X)

  #  modelling with CPUs
  model = RandomForestClassifier(criterion='entropy',
    max_depth=25, ccp_alpha=0.0001, n_jobs=16)
  model.fit(scaledX_train, encodedY_train.ravel())
  print(model.get_params())

  if SAVE:
    # save the model
    dump(model, open('models/airflow_RFC.pkl', 'wb'))
    # save the scaler
    dump(scaler, open('models/airflow_scaler_RFC.pkl', 'wb'))
    # save the encoder
    dump(encoder, open('models/airflow_encoder_RFC.pkl', 'wb'))

print("--- Done in %s seconds ---" % (time.time() - start_time))
```

# F Training Airflow path ANN model

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from keras.utils import np_utils
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras import regularizers
from keras.callbacks import ModelCheckpoint
from sklearn.metrics import accuracy_score, confusion_matrix
import itertools
from keras.models import model_from_json
from pickle import dump, load
import time
start_time = time.time()

def visualize_cm(cm, fname='no'):
    """
    Function visualizes a confusion matrix with and without
   normalization
    """
    plt.rc('legend', fontsize=10)
    plt.rc('axes', labelsize=10)
    plt.rc('xtick', labelsize=10)
    plt.rc('ytick', labelsize=10)


    fig, axes = plt.subplots(1, 2,figsize=(10,5))

    im1 = axes[0].imshow(cm, interpolation='nearest', cmap=plt.cm.
   Blues)
    fig.colorbar(im1, ax=axes[0])
    classes = ['buffer mode','exterior mode','extract mode',
            'interior mode','supply mode']
    tick_marks = np.arange(len(classes))
    axes[0].set_xticks(tick_marks)
    axes[0].set_xticklabels(classes,rotation=45)
    axes[0].set_yticks(tick_marks)
    axes[0].set_yticklabels(classes)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.
   shape[1])):
        axes[0].text(j, i, format(cm[i, j], 'd'),
                 horizontalalignment="center",
                 verticalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    axes[0].set_xlabel('predicted label $\hat{y}$')
    axes[0].set_ylabel('true label $y$')
    axes[0].set_title('Without normalization')
```

```
50
51      cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
52      im2 = axes[1].imshow(cm, interpolation='nearest', cmap=plt.cm.
        Blues)
53      fig.colorbar(im2, ax=axes[1])
54
55      axes[1].set_xticks(tick_marks)
56      axes[1].set_xticklabels(classes,rotation=45)
57      axes[1].set_yticks(tick_marks)
58      axes[1].set_yticklabels(classes)
59
60      thresh = cm.max() / 2.
61      for i, j in itertools.product(range(cm.shape[0]), range(cm.
        shape[1])):
62          axes[1].text(j, i, format(cm[i, j], '.2f'),
63                   horizontalalignment="center",
64                  verticalalignment="center",
65                  color="white" if cm[i, j] > thresh else "black")
66
67      axes[1].set_xlabel('predicted label $\hat{y}$')
68      axes[1].set_ylabel('true label $y$')
69      axes[1].set_title('Normalized')
70
71      axes[0].set_ylim(-0.5,4.5)
72      axes[1].set_ylim(-0.5,4.5)
73
74      plt.tight_layout()
75      if not fname == 'no':
76        plt.savefig(fname)
77      plt.show()
78      percents = np.array([])
79      for i in range(5):
80          percents = np.append(percents,round(cm[i][i]*100,1))
81      return percents
82
83  def build_model(n_inputs,n_outputs=1):
84    init = 'normal'
85    model = Sequential()
86    model.add(Dense(1000, kernel_regularizer=regularizers.l2(0.0001),
87              input_dim=n_inputs, kernel_initializer=init, activation=
        'relu'))
88    model.add(Dropout(0.5))
89    model.add(Dense(500, kernel_regularizer=regularizers.l2(0.0001),
90        kernel_initializer=init, activation='relu'))
91    model.add(Dropout(0.5))
92    model.add(Dense(n_outputs, activation='softmax'))
93    model.compile(loss='categorical_crossentropy',
94        optimizer='adam', metrics=['acc'])
95    return model
96
97  def save_model(model, struct_fname):
98    # saving model
99    json_model = model.to_json()
100   json_file = open(struct_fname, 'w')
```

```
101   json_file.write(json_model)
102   json_file.close()
103   print('Saved model to disk')
104
105 def load_model(struct_fname, weights_fname):
106   # loading model
107   json_file = open(struct_fname, 'r')
108   model = model_from_json(json_file.read())
109   json_file.close()
110   model.load_weights(weights_fname)
111   model.compile(loss='categorical_crossentropy',
112       optimizer='adam', metrics=['acc'])
113   print("Loaded model from disk")
114   return model
115
116  # fix random seed for reproducibility
117 seed = 7
118 np.random.seed(seed)
119
120 # Load the dataset
121 source = 'data/high_blinds_on_thermal.csv'
122 data = pd.read_csv(source, float_precision='high')
123 print('Dataset:',data.shape)
124
125 TRAIN = 0
126 SAVE = 0
127 if TRAIN:
128   # Split train & test datasets
129   features = ['DeltaT', 'Azimuth', 'Altitude', 'SR_diffuse',
130         'SR_direct', 'Visual', 'Thermal']
131   label = 'Airflow_path'
132   X = data[features].values
133   y = data[label].values
134   # encode class values as integers
135   encoder = LabelEncoder().fit(y)
136   encodedY = encoder.transform(y)
137   # convert integers to dummy variables (i.e. one hot encoded)
138   dummy_y = np_utils.to_categorical(encodedY)
139
140   # Split train & test datasets
141   validation_size = 0.33
142   X_train, X_test, y_train, y_test = train_test_split(X, dummy_y,
143       test_size=validation_size, random_state=seed)
144
145   # Standardize the dataset
146   scaler = StandardScaler().fit(X_train)
147   scaledX_train = scaler.transform(X_train)
148   scaledX_test = scaler.transform(X_test)
149   scaledX = scaler.transform(X)
150
151   # modelling and save training history
152   n_inputs = scaledX_train.shape[1]
153   n_outputs = y_train.shape[1]
154   model = build_model(n_inputs,n_outputs)
```

```
155   # checkpoint
156   weights_fname='models/airflow.weights.best.hdf5'
157   checkpoint = ModelCheckpoint(weights_fname, monitor='acc',
158       verbose=0, save_best_only=True, mode='max')
159   callbacks_list = [checkpoint]
160   history = model.fit(scaledX_train, y_train,
161       validation_split=validation_size,
162       epochs=1000, batch_size=500000,
163       callbacks=callbacks_list, verbose=1)
164
165   # print the scores
166   scores = model.evaluate(scaledX_test, y_test, verbose=0)
167   print('Test score: ', model.metrics_names[1], scores[1])
168   if SAVE:
169     # save model
170     dump(history.history, open('models/airflow.history.pkl','wb'))
171     dump(scaler, open('models/airflow.scaler.pkl','wb'))
172     dump(encoder, open('models/airflow.encoder.pkl','wb'))
173     save_model(model,'models/airflow.struct.json')
174
175 print("--- Done in %s seconds ---" % (time.time() - start_time))
```

# G Training Thermal metric XGB models

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from xgboost import XGBRegressor
from sklearn.compose import TransformedTargetRegressor
from sklearn.metrics import mean_absolute_error, r2_score
from pickle import dump, load
import time
start_time = time.time()

# fix random seed for reproducibility
seed = 7
np.random.seed(seed)

# Airflow mode:
#'Buffer mode'
#'Supply mode'
#'Extract mode'
#'Interior air curtain'
#'Exterior air curtain'
MODE = 'Buffer mode'
NAME = MODE.split()[0]
model_name = 'models/'+NAME+'_XGB.pkl'

# Load the dataset
source ='data/high_blinds_on_thermal.csv'
data = pd.read_csv(source)
data = data.loc[data['Airflow_path']==MODE]
print(data.shape)

# Split train & test datasets
if MODE == 'Buffer mode':
  features = ['DeltaT', 'Azimuth', 'Altitude', 'SR_diffuse',
          'SR_direct', 'Blinds', 'Visual']
else:
  features = ['DeltaT', 'Azimuth', 'Altitude', 'SR_diffuse',
        'SR_direct', 'Blinds', 'Speed', 'Visual']
label = 'Thermal'
X = data[features].values
y = data[label].values
validation_size = 0.33
X_train, X_test, y_train, y_test = train_test_split(X, y,
      test_size=validation_size, random_state=seed)


TUNE = 0
TRAIN = 0
SAVE = 0
```

```
53
54 if TUNE:
55   # Standardize the dataset
56   scaler_X= StandardScaler().fit(X_train)
57   scaledX_train = scaler_X.transform(X_train)
58   scaledX_test = scaler_X.transform(X_test)
59   scaledX = scaler_X.transform(X)
60   scaler_Y= StandardScaler().fit(y_train.reshape(-1,1))
61   scaledY_train = scaler_Y.transform(y_train.reshape(-1,1))
62   scaledY_test = scaler_Y.transform(y_test.reshape(-1,1))
63   scaledY = scaler_Y.transform(y.reshape(-1,1))
64
65   params = {
66     'learning_rate': [0.3],
67     'max_depth': [10], #range(5,30,5)
68     'n_estimators': range(50, 450, 50)
69     #'gamma': range(0,1,2),
70     #'alpha': [0,0.001,0.01,0.1],
71     #'lambda': [0,0.001,0.01,0.1],
72     }
73   model = xgb.XGBRegressor(tree_method='gpu_hist',
74       predictor='gpu_predictor', random_state=seed)
75   kfold = KFold(n_splits=3, shuffle=True, random_state=seed)
76   grid_search = GridSearchCV(model, params,
77     scoring='neg_mean_squared_error', n_jobs=-1, cv=kfold)
78   grid_result = grid_search.fit(scaledX_train, scaledY_train)
79   # summarize results
80   print("Best: %f using %s" % (grid_result.best_score_,
81           grid_result.best_params_))
82   means = grid_result.cv_results_['mean_test_score']
83   stds = grid_result.cv_results_['std_test_score']
84   params = grid_result.cv_results_['params']
85   for mean, stdev, param in zip(means, stds, params):
86     print("%f (%f) with: %r" % (mean, stdev, param))
87
88   # plot
89   DEPTH = 0 # 1, n_estimators = 100 default
90   if DEPTH:
91     param=range(5, 30, 5)
92     plt.errorbar(param, means, yerr=stds)
93     plt.title("XGB max_depth vs NMSE")
94     plt.xlabel('max_depth')
95     plt.ylabel('NMSE')
96     plt.show()
97   TREES = 0 # 1, max_depth = 10
98   if TREES:
99     param=range(50, 450, 50)
100    plt.errorbar(param, means, yerr=stds)
101    plt.title("XGB n_estimators vs NMSE")
102    plt.xlabel('n_estimators')
103    plt.ylabel('NMSE')
104    plt.show()
105
106 if TRAIN:
```

```
107   # prepare the model with input scaling
108   pipeline = Pipeline(steps=[('trans', StandardScaler()),
109       ('model', XGBRegressor(tree_method='gpu_hist',
110         predictor='gpu_predictor',
111         learning_rate=0.3, max_depth=10,
112         n_estimators=300, random_state=seed))])
113
114   # prepare the model with target scaling
115   model = TransformedTargetRegressor(regressor=pipeline,
116           transformer=StandardScaler())
117
118   # fit model on training data
119   model.fit(X_train,y_train)
120   print(model.regressor['model'].get_xgb_params())
121
122   mae_train = mean_absolute_error(y_train, model.predict(X_train))
123   r2_train = model.score(X_train,y_train)
124   mae_test = mean_absolute_error(y_test, model.predict(X_test))
125   r2_test = model.score(X_test,y_test)
126   mae = mean_absolute_error(y, model.predict(X))
127   r2 = model.score(X,y)
128   print('Train: ', mae_train, r2_train)
129   print('Test: ', mae_test, r2_test)
130   print('Overall: ', mae, r2)
131
132   if SAVE:
133     # save the model
134     dump(model, open(model_name,'wb'))
135
136 print("--- Done in %s seconds ---" % (time.time() - start_time))
```

# H Training Thermal metric ETR models

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split,
from sklearn.model_selection import KFold, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.compose import TransformedTargetRegressor
from sklearn.metrics import mean_absolute_error, r2_score
from pickle import dump, load
import time
start_time = time.time()

# fix random seed for reproducibility
seed = 7
np.random.seed(seed)

# Airflow mode:
#'Buffer mode'
#'Supply mode'
#'Extract mode'
#'Interior air curtain'
#'Exterior air curtain'
MODE = 'Buffer mode'
NAME = MODE.split()[0]
model_name = 'models/'+NAME+'_ETR.pkl'

# Load the dataset
source ='data/high_blinds_on_thermal.csv'
data = pd.read_csv(source)
data = data.loc[data['Airflow_path']==MODE]
print(data.shape)

# Split train & test datasets
if MODE == 'Buffer mode':
  features = ['DeltaT', 'Azimuth', 'Altitude', 'SR_diffuse',
          'SR_direct', 'Blinds', 'Visual']
else:
  features = ['DeltaT', 'Azimuth', 'Altitude', 'SR_diffuse',
        'SR_direct', 'Blinds', 'Speed', 'Visual']
label = 'Thermal'
X = data[features].values
y = data[label].values
validation_size = 0.33
X_train, X_test, y_train, y_test = train_test_split(X, y,
      test_size=validation_size, random_state=seed)

TUNE = 0
TRAIN = 0
SAVE = 0

```

```
53 if TUNE:
54   # Standardize the dataset
55   scaler_X= StandardScaler().fit(X_train)
56   scaledX_train = scaler_X.transform(X_train)
57   scaledX_test = scaler_X.transform(X_test)
58   scaledX = scaler_X.transform(X)
59   scaler_Y= StandardScaler().fit(y_train.reshape(-1,1))
60   scaledY_train = scaler_Y.transform(y_train.reshape(-1,1))
61   scaledY_test = scaler_Y.transform(y_test.reshape(-1,1))
62   scaledY = scaler_Y.transform(y.reshape(-1,1))
63
64   params = {
65   'max_depth': [25], #range(10,45,5),
66   'n_estimators': range(50, 550, 50)
67   #'ccp_alpha': [0,0.000001,0.00001,0.0001]
68   #'min_samples_split': range(2,12,2)
69   }
70   model = ExtraTreesRegressor(n_jobs=24, random_state=seed)
71   kfold = KFold(n_splits=3, shuffle=True, random_state=seed)
72   grid_search = GridSearchCV(model, params,
73         scoring='neg_mean_squared_error',
74         n_jobs=-1, cv=kfold)
75   grid_result = grid_search.fit(scaledX_train, scaledY_train.ravel
       ())
76   # summarize results
77   print("Best: %f using %s" % (grid_result.best_score_,
78           grid_result.best_params_))
79   means = grid_result.cv_results_['mean_test_score']
80   stds = grid_result.cv_results_['std_test_score']
81   params = grid_result.cv_results_['params']
82   for mean, stdev, param in zip(means, stds, params):
83     print("%f (%f) with: %r" % (mean, stdev, param))
84
85   # plot
86   DEPTH = 0 # 1, n_estimators = 100 default
87   if DEPTH:
88     param=range(10, 45, 5)
89     plt.errorbar(param, means, yerr=stds)
90     plt.title("ETR max_depth vs NMSE")
91     plt.xlabel('max_depth')
92     plt.ylabel('NMSE')
93     plt.show()
94     DEPTH = 0
95   TREES = 0 #1, max_depth = 25
96   if TREES:
97     param=range(50, 550, 50)
98     plt.errorbar(param, means, yerr=stds)
99     plt.title("ETR n_estimators vs NMSE")
100    plt.xlabel('n_estimators')
101    plt.ylabel('NMSE')
102    plt.show()
103
104 if TRAIN:
105   # prepare the model with input scaling
```

```
106   pipeline = Pipeline(steps=[('trans', StandardScaler()),
107       ('model', ExtraTreesRegressor(n_jobs=24,
108       max_depth=25, ccp_alpha =1e-6, random_state=seed))])
109
110   # prepare the model with target scaling
111   model = TransformedTargetRegressor(regressor=pipeline,
112           transformer=StandardScaler())
113
114   # fit model on training data
115   model.fit(X_train,y_train)
116   print(model.regressor['model'].get_params())
117   mae_train = mean_absolute_error(y_train, model.predict(X_train))
118   r2_train = model.score(X_train,y_train)
119   mae_test = mean_absolute_error(y_test, model.predict(X_test))
120   r2_test = model.score(X_test,y_test)
121   mae = mean_absolute_error(y, model.predict(X))
122   r2 = model.score(X,y)
123   print('Train: ', mae_train, r2_train)
124   print('Test: ', mae_test, r2_test)
125   print('Overall: ', mae, r2)
126
127   if SAVE:
128     # save the model
129     dump(model, open(model_name,'wb'))
130
131 print("--- Done in %s seconds ---" % (time.time() - start_time))
```

# I Training Thermal metric ANN models

```
# Load libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split,
from sklearn.model_selection import KFold, GridSearchCV
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential
from keras.layers import Dense, Dropout
from sklearn.metrics import mean_absolute_error, r2_score
from keras.callbacks import ModelCheckpoint
from keras.models import model_from_json
from keras import regularizers
from pickle import dump, load
import time
start_time = time.time()

def build_model(n_inputs, n_outputs=1, l2=0.0001,
      layers=(1000,500), init='normal'):
  model = Sequential()
  model.add(Dense(layers[0], input_dim=n_inputs,
      kernel_regularizer=regularizers.l2(l2),
      kernel_initializer=init, activation='relu'))
  model.add(Dropout(0.5))
  model.add(Dense(layers[1],
      kernel_regularizer=regularizers.l2(l2),
      kernel_initializer=init, activation='relu'))
  model.add(Dropout(0.5))
  model.add(Dense(n_outputs, kernel_initializer=init))
  model.compile(loss='mse', optimizer='adam', metrics=['mae'])
  return model

def save_model(model, struct_fname):
  # saving model
  json_model = model.to_json()
  json_file = open(struct_fname, 'w')
  json_file.write(json_model)
  json_file.close()
  print('Saved model to disk')

def load_model(struct_fname, weights_fname):
  # loading model
  json_file = open(struct_fname, 'r')
  model = model_from_json(json_file.read())
  json_file.close()
  model.load_weights(weights_fname)
  model.compile(loss='mse', optimizer='adam', metrics=['mae'])
  print("Loaded model from disk")
  return model

# fix random seed for reproducibility
seed = 7
```

```
53 np.random.seed(seed)
54
55 # Airflow mode:
56 #'Buffer mode'
57 #'Supply mode'
58 #'Extract mode'
59 #'Interior air curtain'
60 #'Exterior air curtain'
61 MODE = 'Buffer mode'
62 NAME = MODE.split()[0]
63
64 struct_fname = 'models/'+NAME+'.struct.json'
65 weights_fname = 'models/'+NAME+'.weights.best.hdf5'
66 history_fname = 'models/'+NAME+'.history.pkl'
67 scalerX = 'models/'+NAME+'.scalerX.pkl'
68 scalerY = 'models/'+NAME+'.scalerY.pkl'
69
70 # Load the dataset
71 source ='data/high_blinds_on_thermal.csv'
72 data = pd.read_csv(source)
73 data = data.loc[data['Airflow_path']==MODE]
74 print(data.shape)
75
76 # Split train & test datasets
77 if MODE == 'Buffer mode':
78   features = ['DeltaT', 'Azimuth', 'Altitude', 'SR_diffuse',
79           'SR_direct', 'Blinds', 'Visual']
80 else:
81   features = ['DeltaT', 'Azimuth', 'Altitude', 'SR_diffuse',
82         'SR_direct', 'Blinds', 'Speed', 'Visual']
83 label = 'Thermal'
84 X = data[features].values
85 y = data[label].values
86 validation_size = 0.33
87 X_train, X_test, y_train, y_test = train_test_split(X, y,
88       test_size=validation_size, random_state=seed)
89
90 TUNE = 0
91 TRAIN = 0
92 SAVE = 0
93
94 if TUNE:
95   # Standardize the dataset
96   scaler_X= StandardScaler().fit(X_train)
97   scaledX_train = scaler_X.transform(X_train)
98   scaledX_test = scaler_X.transform(X_test)
99   scaledX = scaler_X.transform(X)
100  scaler_Y= StandardScaler().fit(y_train.reshape(-1,1))
101  scaledY_train = scaler_Y.transform(y_train.reshape(-1,1))
102  scaledY_test = scaler_Y.transform(y_test.reshape(-1,1))
103  scaledY = scaler_Y.transform(y.reshape(-1,1))
104
105  model = KerasRegressor(build_fn=build_model, verbose=0)
106  # grid search epochs, batch size and optimizer
```

```
107   layers = [(100,50),(200,100),(600,300),(800,400),
108       (1000,500),(1200,600),(1400,700),(1600,800)]
109   #layers = [(1200,600)]
110   #l2 = [0.0001, 0.001, 0.01, 0.1]
111   l2 = [0.0001]
112   #epochs = [200, 400, 600, 800, 1000]
113   epochs = [100]
114   #batches = [1000, 10000, 100000, 300000]
115   batches = [1000]
116   n_inputs = [scaledX.shape[1]]
117   kfold = KFold(n_splits=3, shuffle=True, random_state=seed)
118   param_grid = dict(n_inputs=n_inputs, l2=l2,
119         layers=layers, epochs=epochs,
120         batch_size=batches, verbose=[1])
121   grid = GridSearchCV(estimator=model, param_grid=param_grid,
122       scoring='neg_mean_squared_error', cv=kfold, n_jobs=-1)
123   grid_result = grid.fit(scaledX_train, scaledY_train)
124   # summarize results
125   print("Best: %f using %s" % (grid_result.best_score_,
126           grid_result.best_params_))
127   means = grid_result.cv_results_['mean_test_score']
128   stds = grid_result.cv_results_['std_test_score']
129   params = grid_result.cv_results_['params']
130   for mean, stdev, param in zip(means, stds, params):
131     print("%f (%f) with: %r" % (mean, stdev, param))
132
133   # plot
134   LAYERS = 1
135   if LAYERS:
136     param=range(8)
137     plt.errorbar(param, means, yerr=stds)
138     plt.xticks(param,layers,fontsize=8)
139     plt.title("ANN hidden layers vs NMSE")
140     plt.xlabel('Hidden layers')
141     plt.ylabel('NMSE')
142     plt.show()
143     # plot
144   L2 = 0
145   if L2:
146     param=range(4)
147     plt.errorbar(param, means, yerr=stds)
148     plt.xticks(param,l2,fontsize=8)
149     plt.title("ANN l2 regularization vs NMSE")
150     plt.xlabel('l2')
151     plt.ylabel('NMSE')
152     plt.show()
153   BATCHES = 0
154   if BATCHES:
155     param=batches
156     plt.errorbar(param, means, yerr=stds)
157     plt.title("ANN batch size vs NMSE")
158     plt.xlabel('Batch size')
159     plt.ylabel('NMSE')
160     plt.show()
```

```
  EPOCHS = 0
  if EPOCHS:
    param=epochs
    plt.errorbar(param, means, yerr=stds)
    plt.title("ANN epochs vs NMSE")
    plt.xlabel('epochs')
    plt.ylabel('NMSE')
    plt.show()


if TRAIN:
  # Standardize the dataset
  scaler_X= StandardScaler().fit(X_train)
  scaledX_train = scaler_X.transform(X_train)
  scaledX_test = scaler_X.transform(X_test)
  scaledX = scaler_X.transform(X)

  scaler_Y= StandardScaler().fit(y_train.reshape(-1,1))
  scaledY_train = scaler_Y.transform(y_train.reshape(-1,1))
  scaledY_test = scaler_Y.transform(y_test.reshape(-1,1))
  scaledY = scaler_Y.transform(y.reshape(-1,1))

  # modelling and save training history
  batch_size = 300000
  epochs = 10000
  n_inputs = scaledX.shape[1]
  model = build_model(n_inputs)
  # checkpoint
  checkpoint = ModelCheckpoint(weights_fname, monitor='mae',
        verbose=0, save_best_only=True, mode='max')
  callbacks_list = [checkpoint]
  hist = model.fit(scaledX_train, scaledY_train,
      validation_split=validation_size,
      epochs=epochs, batch_size=batch_size,
      callbacks=callbacks_list, verbose=1)

  # print the scores
  y_pred = scaler_Y.inverse_transform(model.predict(scaledX_train))
  mae_train = mean_absolute_error(y_train, y_pred)
  r2_train = r2_score(y_train, y_pred)
  print('Train: ', mae_train, r2_train)
  y_pred = scaler_Y.inverse_transform(model.predict(scaledX_test))
  mae_test = mean_absolute_error(y_test, y_pred)
  r2_test = r2_score(y_test, y_pred)
  print('Test: ', mae_test, r2_test)
  y_pred = scaler_Y.inverse_transform(model.predict(scaledX))
  mae = mean_absolute_error(y, y_pred)
  r2 = r2_score(y, y_pred)
  print('Overall: ', mae, r2)

  # save model
  if SAVE:
    save_model(model, struct_fname)
    dump(scaler_X, open(scalerX,'wb'))
```

```
215     dump(scaler_Y, open(scalerY,'wb'))
216     dump(hist.history, open(history_fname,'wb'))
217 
218 print("--- Done in %s seconds ---" % (time.time() - start_time))
```

# J Outputting from Blinds Angle Controller

```
import os
from multiprocessing import Process, Queue
import numpy as np
import pandas as pd
from pickle import load
import time
start_time = time.time()

n_proc = int(os.getenv('SLURM_CPUS_PER_TASK',1))
print('Using %d processors' % n_proc)

# function to estimate blinds angle and visual metric for single
    data point
def predict_blinds_angle(model, Azimuth, Altitude, SR_diffuse,
    SR_direct, V_opt):
  if V_opt < 0.1:
    Blinds_pred = 0
    V_pred = 0
  else:
    blinds = np.array(range(0,100,10))
    inputs = np.zeros((len(blinds),5))
    inputs[:,0] = Azimuth
    inputs[:,1] = Altitude
    inputs[:,2] = SR_diffuse
    inputs[:,3] = SR_direct
    inputs[:,4] = blinds
    Visual_pred = model.predict(inputs)
    Visual_opt = np.zeros(Visual_pred.shape)
    Visual_opt[:,]=V_opt
    Visual_err = np.subtract(Visual_opt,Visual_pred)
    index_min = np.argmin(np.absolute(Visual_err))
    Blinds_pred = blinds[index_min]
    V_pred = Visual_pred[index_min]
  return Blinds_pred, V_pred

# function for each process
def func_proc(q, model, X):
  n = X.shape[0]
  Blinds_pred = np.zeros((n,1))
  Visual_pred = np.empty((n,1))
  for i in range(n):
    if i%1000==0:
      print(i)
    blinds_pred, visual_pred = predict_blinds_angle(model, X[i,0],
            X[i,1], X[i,2], X[i,3], X[i,4])
    Blinds_pred[i] = blinds_pred
    Visual_pred[i] = visual_pred
  q.put([Blinds_pred,Visual_pred])

if __name__ == '__main__':
  # Load model
  model = load(open('models/visual_XGB.pkl','rb'))
```

```
  print('Loaded model successfully')

  # Load the dataset
  source = 'data/high_blinds_on_visual.csv'
  data = pd.read_csv(source)
  print(data.shape)

  # predict Visual_pred_base
  X = data[['Azimuth','Altitude','SR_diffuse','SR_direct','Blinds'
   ]].values
  Visual_pred_base = model.predict(X)
  data['Visual_pred_base']=Visual_pred_base

  # predict blinds + visual
  X = data[['Azimuth','Altitude','SR_diffuse','SR_direct','Visual'
   ]].values
  n = X.shape[0]
  k = int(n/n_proc)
  # define processes
  q = []
  p = []
  for i in range(n_proc):
    q.append(Queue())
  for i in range(n_proc-1):
    i_start = i*k
    i_end = (i+1)*k
    p.append(Process(target=func_proc,
        args=(q[i],model,X[i_start:i_end,:])))
  i_start = (n_proc-1)*k
  p.append(Process(target=func_proc,
        args=(q[n_proc-1],model,X[i_start:,:])))
  for i in range(n_proc):
    p[i].start()
  Blinds_pred = np.zeros((n,1))
  Visual_pred = np.zeros((n,1))
  for i in range(n_proc-1):
    blinds, visual = q[i].get()
    i_start = i*k
    i_end = (i+1)*k
    Blinds_pred[i_start:i_end] = blinds
    Visual_pred[i_start:i_end] = visual
    p[i].join()
  blinds, visual = q[n_proc-1].get()
  i_start = (n_proc-1)*k
  Blinds_pred[i_start:] = blinds
  Visual_pred[i_start:] = visual
  p[n_proc-1].join()
  data['Blinds_pred_XGB']=Blinds_pred
  data['Visual_pred_XGB']=Visual_pred

  # Save results
  data.to_csv('data/V_pred.Blinds_ctrl.csv', index=False)

print("--- Done in %s seconds ---" % (time.time() - start_time))
```

# K Outputting from Airflow Speed Controller

```
import os
import numba
from numba import prange, cuda
import numpy as np
import pandas as pd
from pickle import load
import time
start_time = time.time()
print('Using %d processors' % int(os.getenv('SLURM_CPUS_PER_TASK',1)))

@cuda.jit
def get_blinds_visual(X,A,B,V):
  i=cuda.grid(1)
  if i < X.shape[0]:
    if i%100000==0:
      print(i)
    for a in A:
      if round(X[i,0])==round(a[0]) and round(X[i,1])==round(a[1])
      and round(X[i,2])==round(a[2]) and round(X[i,3])==round(a[3])
      and round(X[i,4])==round(a[4]) and round(X[i,5])==round(a[5])
    :
        B[i] = a[6]
        V[i] = a[7]

# Load dataset
source='data/V_pred.Blinds_ctrl.csv'
data = pd.read_csv(source)
print(data.shape)
A = data[['Azimuth','Altitude','SR_diffuse','SR_direct','Blinds','Visual',
        'Blinds_pred_XGB','Visual_pred_XGB']].values
source='data/Airflow_ctrl.csv' # collected from training airflow path XGB model
df = pd.read_csv(source)
print(df.shape)
X=df[['Azimuth','Altitude','SR_diffuse','SR_direct','Blinds','Visual']].values
blinds = np.zeros((df.shape[0],1))
visual = np.zeros((df.shape[0],1))
B_device = cuda.to_device(blinds)
V_device = cuda.to_device(visual)
X_device = cuda.to_device(X)
A_device = cuda.to_device(A)
threadsperblock = 32
blockspergrid = (X.shape[0] + (threadsperblock - 1)) // threadsperblock
get_blinds_visual[blockspergrid, threadsperblock](X_device,A_device,
              B_device,V_device)
blinds = B_device.copy_to_host()
visual = V_device.copy_to_host()
```

```
46 df['Blinds_pred_XGB']=blinds
47 df['Visual_pred_XGB']=visual
48
49 # Save results
50 df.to_csv('data/V_pred.Blinds_ctrl.Airflow_ctrl.csv', index=False)
51
52 print("--- Done in %s seconds ---" % (time.time() - start_time))
```

Listing 1: Collecting V_pred Blinds_ctrl Airflow_ctrl

```
import numpy as np
import pandas as pd
from pickle import load
import matplotlib.pyplot as plt
from sklearn.metrics import mean_absolute_error, r2_score
from multiprocessing import Process, Queue
import os
import time

start_time = time.time()

n_proc = int(os.getenv('SLURM_CPUS_PER_TASK',1))
print('Using %d processors' % n_proc)

def predict_airflow_speed(model, inputs):
  T_pred = model.predict(inputs.reshape(1,-1))
  T_pred = T_pred[0]
  Speed_pred = 0
  return Speed_pred,T_pred

def func_proc(q, model, X):
  n = X.shape[0]
  Speed_pred = np.zeros((n,1))
  Thermal_pred = np.zeros((n,1))
  for i in range(n):
    if i%100000==0:
      print(i)
    speed_pred,thermal_pred = predict_airflow_speed(model, X[i,:])
    Speed_pred[i]=speed_pred
    Thermal_pred[i]=thermal_pred
  q.put([Speed_pred,Thermal_pred])

if __name__ == '__main__':
  # Load model and dataset
  model = load(open('models/Buffer_XGB.pkl','rb'))
  print('Loaded model')
  source ='data/V_pred.Blinds_ctrl.Airflow_ctrl.csv'
  data = pd.read_csv(source)
  data = data.loc[data['Airflow_pred_XGB']=='Buffer mode']
  print('Loaded dataset: ',data.shape)

  # Set up processes
  X = data[['DeltaT','Azimuth','Altitude','SR_diffuse','SR_direct',
        'Blinds_pred_XGB','Visual_pred_XGB']].values
  n = X.shape[0]
  k = int(n/n_proc)
  # define processes
  q = []
  p = []
  for i in range(n_proc):
    q.append(Queue())
  for i in range(n_proc-1):
    i_start = i*k
    i_end = (i+1)*k
```

```
    p.append(Process(target=func_proc,
      args=(q[i],model,X[i_start:i_end,:])))
  i_start = (n_proc-1)*k
  p.append(Process(target=func_proc,
      args=(q[n_proc-1],model,X[i_start:,:])))
  for i in range(n_proc):
    p[i].start()

  Speed_pred = np.zeros((n,1))
  Thermal_pred = np.zeros((n,1))
  for i in range(n_proc-1):
    speed, thermal = q[i].get()
    i_start = i*k
    i_end = (i+1)*k
    Speed_pred[i_start:i_end] = speed
    Thermal_pred[i_start:i_end] = thermal
    p[i].join()
  speed, thermal = q[n_proc-1].get()
  i_start = (n_proc-1)*k
  Speed_pred[i_start:] = speed
  Thermal_pred[i_start:] = thermal
  p[n_proc-1].join()
  data['Speed_pred_XGB']=Speed_pred
  data['Thermal_pred_XGB']=Thermal_pred

  # save results
  data.to_csv('data/result_buffer.csv',index=False)

print("--- Done in %s seconds ---" % (time.time() - start_time))
```

Listing 2: Outputting Speed_ctrl and T_pred in Buffer mode

```
import numpy as np
import pandas as pd
from pickle import load
import matplotlib.pyplot as plt
from sklearn.metrics import mean_absolute_error, r2_score
from multiprocessing import Process, Queue
import os
import time

start_time = time.time()

n_proc = int(os.getenv('SLURM_CPUS_PER_TASK',1))
print('Using %d processors' % n_proc)

def predict_airflow_speed(model, inputs):
  speeds = [0.088888889, 0.044444444000000013,
   0.022222222000000003, 0.011111111000000002, 0.005555556,
   0.002777778]
  X = np.zeros((len(speeds),8))
  for i in range(6):
    X[:,i]=inputs[i]
  X[:,-2]=speeds
  X[:,-1]= inputs[-2]
  T_opt = inputs[-1]
  Thermal_pred = model.predict(X)
  Thermal_opt = np.zeros(Thermal_pred.shape)
  Thermal_opt[:,]=T_opt
  Thermal_err = np.subtract(Thermal_opt,Thermal_pred)
  index_min = np.argmin(np.absolute(Thermal_err))
  Speed_pred = speeds[index_min]
  T_pred = Thermal_pred[index_min]
  return Speed_pred,T_pred

def func_proc(q, model, X):
  n = X.shape[0]
  Speed_pred = np.zeros((n,1))
  Thermal_pred = np.zeros((n,1))
  for i in range(n):
    if i%100000==0:
      print(i)
    speed_pred,thermal_pred = predict_airflow_speed(model, X[i,:])
    Speed_pred[i]=speed_pred
    Thermal_pred[i]=thermal_pred
  q.put([Speed_pred,Thermal_pred])

if __name__ == '__main__':
  # Load model
  model = load(open('models/Interior_XGB.pkl','rb'))
  print('Loaded model')
  source = 'data/V_pred.Blinds_ctrl.Airflow_ctrl.csv'
  data = pd.read_csv(source)
  data = data.loc[data['Airflow_pred_XGB']=='Interior air curtain']
  print('Loaded dataset: ',data.shape)

```

```
  # Set up processes
  X = data[['DeltaT','Azimuth','Altitude','SR_diffuse','SR_direct',
      'Blinds_pred_XGB','Visual_pred_XGB','Thermal']].values
  n = X.shape[0]
  k = int(n/n_proc)
  # define processes
  q = []
  p = []
  for i in range(n_proc):
    q.append(Queue())
  for i in range(n_proc-1):
    i_start = i*k
    i_end = (i+1)*k
    p.append(Process(target=func_proc,
        args=(q[i],model,X[i_start:i_end,:])))
  i_start = (n_proc-1)*k
  p.append(Process(target=func_proc,
        args=(q[n_proc-1],model,X[i_start:,:])))
  for i in range(n_proc):
    p[i].start()

  Speed_pred = np.zeros((n,1))
  Thermal_pred = np.zeros((n,1))
  for i in range(n_proc-1):
    speed, thermal = q[i].get()
    i_start = i*k
    i_end = (i+1)*k
    Speed_pred[i_start:i_end] = speed
    Thermal_pred[i_start:i_end] = thermal
    p[i].join()
  speed, thermal = q[n_proc-1].get()
  i_start = (n_proc-1)*k
  Speed_pred[i_start:] = speed
  Thermal_pred[i_start:] = thermal
  p[n_proc-1].join()
  data['Speed_pred_XGB']=Speed_pred
  data['Thermal_pred_XGB']=Thermal_pred

  # save results
  data.to_csv('data/result_interior.csv',index=False)

print("--- Done in %s seconds ---" % (time.time() - start_time))
```

Listing 3: Outputting Speed_ctrl and T_pred in Interior air curtain mode

```
import numpy as np
import pandas as pd
from pickle import load
import matplotlib.pyplot as plt
from sklearn.metrics import mean_absolute_error, r2_score
from multiprocessing import Process, Queue
import os
import time

start_time = time.time()

n_proc = int(os.getenv('SLURM_CPUS_PER_TASK',1))
print('Using %d processors' % n_proc)

def predict_airflow_speed(model, inputs):
  speeds = [0.088888889, 0.044444444000000013,
   0.022222222000000003, 0.011111111000000002, 0.005555556,
   0.002777778]
  X = np.zeros((len(speeds),8))
  for i in range(6):
    X[:,i]=inputs[i]
  X[:,-2]=speeds
  X[:,-1]= inputs[-2]
  T_opt = inputs[-1]
  Thermal_pred = model.predict(X)
  Thermal_opt = np.zeros(Thermal_pred.shape)
  Thermal_opt[:,]=T_opt
  Thermal_err = np.subtract(Thermal_opt,Thermal_pred)
  index_min = np.argmin(np.absolute(Thermal_err))
  Speed_pred = speeds[index_min]
  T_pred = Thermal_pred[index_min]
  return Speed_pred,T_pred

def func_proc(q, model, X):
  n = X.shape[0]
  Speed_pred = np.zeros((n,1))
  Thermal_pred = np.zeros((n,1))
  for i in range(n):
    if i%100000==0:
      print(i)
    speed_pred,thermal_pred = predict_airflow_speed(model, X[i,:])
    Speed_pred[i]=speed_pred
    Thermal_pred[i]=thermal_pred
  q.put([Speed_pred,Thermal_pred])

if __name__ == '__main__':
  # Load model
  model = load(open('models/Exterior_XGB.pkl','rb'))
  print('Loaded model')
  source ='data/V_pred.Blinds_ctrl.Airflow_ctrl.csv'
  data = pd.read_csv(source)
  data = data.loc[data['Airflow_pred_XGB']=='Exterior air curtain']
  print('Loaded dataset: ',data.shape)

```

```
  # Set up processes
  X = data[['DeltaT','Azimuth','Altitude','SR_diffuse','SR_direct',
      'Blinds_pred_XGB','Visual_pred_XGB','Thermal']].values
  n = X.shape[0]
  k = int(n/n_proc)
  # define processes
  q = []
  p = []
  for i in range(n_proc):
    q.append(Queue())
  for i in range(n_proc-1):
    i_start = i*k
    i_end = (i+1)*k
    p.append(Process(target=func_proc,
        args=(q[i],model,X[i_start:i_end,:])))
  i_start = (n_proc-1)*k
  p.append(Process(target=func_proc,
        args=(q[n_proc-1],model,X[i_start:,:])))
  for i in range(n_proc):
    p[i].start()

  Speed_pred = np.zeros((n,1))
  Thermal_pred = np.zeros((n,1))
  for i in range(n_proc-1):
    speed, thermal = q[i].get()
    i_start = i*k
    i_end = (i+1)*k
    Speed_pred[i_start:i_end] = speed
    Thermal_pred[i_start:i_end] = thermal
    p[i].join()
  speed, thermal = q[n_proc-1].get()
  i_start = (n_proc-1)*k
  Speed_pred[i_start:] = speed
  Thermal_pred[i_start:] = thermal
  p[n_proc-1].join()
  data['Speed_pred_XGB']=Speed_pred
  data['Thermal_pred_XGB']=Thermal_pred

  # save results
  data.to_csv('data/result_exterior.csv',index=False)

print("--- Done in %s seconds ---" % (time.time() - start_time))
```

Listing 4: Outputting Speed_ctrl and T_pred in Exterior air curtain mode

```
import numpy as np
import pandas as pd
from pickle import load
import matplotlib.pyplot as plt
from sklearn.metrics import mean_absolute_error, r2_score
from multiprocessing import Process, Queue
import os
import time

start_time = time.time()

n_proc = int(os.getenv('SLURM_CPUS_PER_TASK',1))
print('Using %d processors' % n_proc)

def predict_airflow_speed(model, inputs):
  speeds = [0.088888889, 0.044444444000000013,
   0.022222222000000003, 0.011111111000000002, 0.005555556,
   0.002777778]
  X = np.zeros((len(speeds),8))
  for i in range(6):
    X[:,i]=inputs[i]
  X[:,-2]=speeds
  X[:,-1]= inputs[-2]
  T_opt = inputs[-1]
  Thermal_pred = model.predict(X)
  Thermal_opt = np.zeros(Thermal_pred.shape)
  Thermal_opt[:,]=T_opt
  Thermal_err = np.subtract(Thermal_opt,Thermal_pred)
  index_min = np.argmin(np.absolute(Thermal_err))
  Speed_pred = speeds[index_min]
  T_pred = Thermal_pred[index_min]
  return Speed_pred,T_pred

def func_proc(q, model, X):
  n = X.shape[0]
  Speed_pred = np.zeros((n,1))
  Thermal_pred = np.zeros((n,1))
  for i in range(n):
    if i%100000==0:
      print(i)
    speed_pred,thermal_pred = predict_airflow_speed(model, X[i,:])
    Speed_pred[i]=speed_pred
    Thermal_pred[i]=thermal_pred
  q.put([Speed_pred,Thermal_pred])

if __name__ == '__main__':
  # Load model
  model = load(open('models/Extract_XGB.pkl','rb'))
  print('Loaded model')
  source ='data/V_pred.Blinds_ctrl.Airflow_ctrl.csv'
  data = pd.read_csv(source)
  data = data.loc[data['Airflow_pred_XGB']=='Extract mode']
  print('Loaded dataset: ',data.shape)

```

```
  # Set up processes
  X = data[['DeltaT','Azimuth','Altitude','SR_diffuse','SR_direct',
      'Blinds_pred_XGB','Visual_pred_XGB','Thermal']].values
  n = X.shape[0]
  k = int(n/n_proc)
  # define processes
  q = []
  p = []
  for i in range(n_proc):
    q.append(Queue())
  for i in range(n_proc-1):
    i_start = i*k
    i_end = (i+1)*k
    p.append(Process(target=func_proc,
        args=(q[i],model,X[i_start:i_end,:])))
  i_start = (n_proc-1)*k
  p.append(Process(target=func_proc,
        args=(q[n_proc-1],model,X[i_start:,:])))
  for i in range(n_proc):
    p[i].start()

  Speed_pred = np.zeros((n,1))
  Thermal_pred = np.zeros((n,1))
  for i in range(n_proc-1):
    speed, thermal = q[i].get()
    i_start = i*k
    i_end = (i+1)*k
    Speed_pred[i_start:i_end] = speed
    Thermal_pred[i_start:i_end] = thermal
    p[i].join()
  speed, thermal = q[n_proc-1].get()
  i_start = (n_proc-1)*k
  Speed_pred[i_start:] = speed
  Thermal_pred[i_start:] = thermal
  p[n_proc-1].join()
  data['Speed_pred_XGB']=Speed_pred
  data['Thermal_pred_XGB']=Thermal_pred

  # save results
  data.to_csv('data/result_exhaust.csv',index=False)

print("--- Done in %s seconds ---" % (time.time() - start_time))
```

Listing 5: Outputting Speed_ctrl and T_pred in Exhaust mode

```
import numpy as np
import pandas as pd
from pickle import load
import matplotlib.pyplot as plt
from sklearn.metrics import mean_absolute_error, r2_score
from multiprocessing import Process, Queue
import os
import time

start_time = time.time()

n_proc = int(os.getenv('SLURM_CPUS_PER_TASK',1))
print('Using %d processors' % n_proc)

def predict_airflow_speed(model, inputs):
  speeds = [0.088888889, 0.044444444000000013,
   0.022222222000000003, 0.011111111000000002, 0.005555556,
   0.002777778]
  X = np.zeros((len(speeds),8))
  for i in range(6):
    X[:,i]=inputs[i]
  X[:,-2]=speeds
  X[:,-1]= inputs[-2]
  T_opt = inputs[-1]
  Thermal_pred = model.predict(X)
  Thermal_opt = np.zeros(Thermal_pred.shape)
  Thermal_opt[:,]=T_opt
  Thermal_err = np.subtract(Thermal_opt,Thermal_pred)
  index_min = np.argmin(np.absolute(Thermal_err))
  Speed_pred = speeds[index_min]
  T_pred = Thermal_pred[index_min]
  return Speed_pred,T_pred

def func_proc(q, model, X):
  n = X.shape[0]
  Speed_pred = np.zeros((n,1))
  Thermal_pred = np.zeros((n,1))
  for i in range(n):
    if i%100000==0:
      print(i)
    speed_pred,thermal_pred = predict_airflow_speed(model, X[i,:])
    Speed_pred[i]=speed_pred
    Thermal_pred[i]=thermal_pred
  q.put([Speed_pred,Thermal_pred])

if __name__ == '__main__':
  # Load model
  model = load(open('models/Supply_XGB.pkl','rb'))
  print('Loaded model')
  source ='data/V_pred.Blinds_ctrl.Airflow_ctrl.csv'
  data = pd.read_csv(source)
  data = data.loc[data['Airflow_pred_XGB']=='Supply mode']
  print('Loaded dataset: ',data.shape)

```

```
  # Set up processes
  X = data[['DeltaT','Azimuth','Altitude','SR_diffuse','SR_direct',
      'Blinds_pred_XGB','Visual_pred_XGB','Thermal']].values
  n = X.shape[0]
  k = int(n/n_proc)
  # define processes
  q = []
  p = []
  for i in range(n_proc):
    q.append(Queue())
  for i in range(n_proc-1):
    i_start = i*k
    i_end = (i+1)*k
    p.append(Process(target=func_proc,
        args=(q[i],model,X[i_start:i_end,:])))
  i_start = (n_proc-1)*k
  p.append(Process(target=func_proc,
        args=(q[n_proc-1],model,X[i_start:,:])))
  for i in range(n_proc):
    p[i].start()

  Speed_pred = np.zeros((n,1))
  Thermal_pred = np.zeros((n,1))
  for i in range(n_proc-1):
    speed, thermal = q[i].get()
    i_start = i*k
    i_end = (i+1)*k
    Speed_pred[i_start:i_end] = speed
    Thermal_pred[i_start:i_end] = thermal
    p[i].join()
  speed, thermal = q[n_proc-1].get()
  i_start = (n_proc-1)*k
  Speed_pred[i_start:] = speed
  Thermal_pred[i_start:] = thermal
  p[n_proc-1].join()
  data['Speed_pred_XGB']=Speed_pred
  data['Thermal_pred_XGB']=Thermal_pred

  # save results
  data.to_csv('data/result_supply.csv',index=False)

print("--- Done in %s seconds ---" % (time.time() - start_time))
```

Listing 6: Outputting Speed_ctrl and T_pred in Supply mode

# Liite 3 Simulointitulokset - asuinkerrostalo

Aalto University
School of Engineering

# Intelligent building envelope solutions in Finnish climatic conditions - Simulation of new and old apartment building with and without active cooling system

17.12.2020

Azin Velashjerdi Farahani, Juha Jokisalo, Natalia Korhonen, Risto Kosonen

# Overview of the study

**Old and new buildings**

- **Apartment**

**Envelope solutions**

- Window properties

**Main Goals**

✓To see the effects of each solution on cooling and heating energy consumption.

✓To see the effects of each solution on the indoor air temperature

# Methods

## Simulation tool

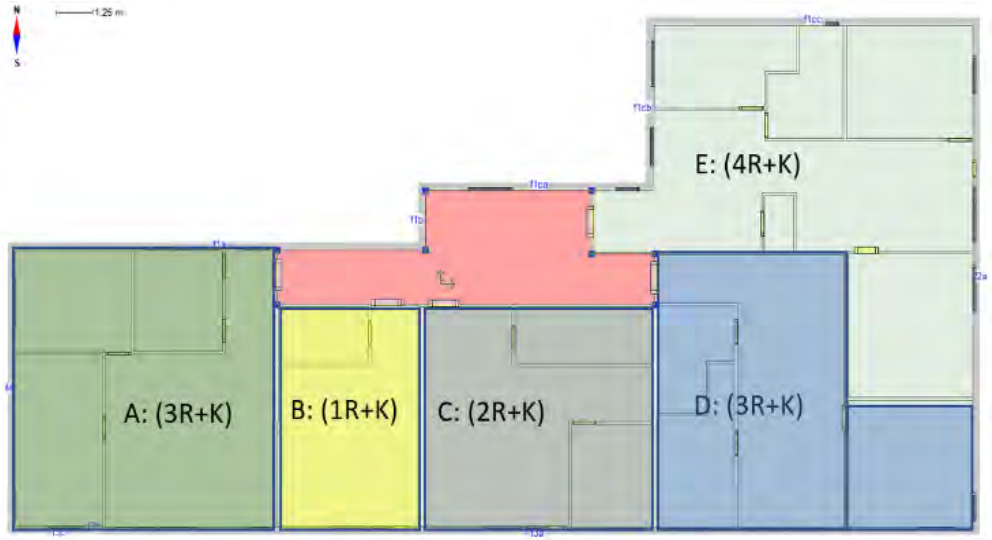
- **IDA ICE 4.8**
- **The simulation period**: one year
- **Time resolution of the results:** 1 hour.
- **Whole building simulation**

## Climatic data

**Weather data:** TRY (2020)

# Building properties

| | New building | Old building |
|---|---|---|
| Year of construction | after 2012 | 1960s, 1950s |
| Heated net floor area ($m^2$) | 1943.5 | |
| U-value of external wall ($W/m^2K$) | 0.17 | 0.6 |
| U-value of external window ($W/m^2K$) | 1 | 2.5 |
| Air tightness of building envelope (q50) ($m^3/h,m^2$) | 4 | 6 |
| Heating system | District heating, 70/40°C water radiator, 65 W/m² | District heating, 70/40°C water radiator, 100 W/m² |
| Ventilation system | Constant air volume (CAV) mechanical supply (17°C) and exhaust ventilation system, Heat recovery efficiency: 0.65 | mechanical exhaust ventilation |
| | total air exchange rate of the building: 0.52 ACH. | |
| Split cooling unit | SCOP: 3, Living rooms, Cooling capacity of the units: 45 W/m² | |



| Simulation Cases of the old apartment building, IBES Project | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Cases | | Wall construction | | | | Window properties | | | | | | | Setpoint | |
| | | Average U value ($W/m^2K$) | Window to wall ratio (%) | Air Leakage q50 ($m^3/h,m^2$) | Window Type | U value ($W/m^2K$) | Tsol | g-value | External shading | Integrated shading | Window opening | | Heating setpoint °C | Cooling setpoint °C |
| **Passive Solutions** | **Case 2** | 0.6 | 19 % | 6 | Clear double glazing window | 2.5 | 0.7 | 0.76 | window indentation 5cm | The manually controlled blinds between the outer panes are used when some one is in the room according to the occupancy profile. | None | | 21 | 24 |
| | **Case 3** | 0.6 | 19 % | 6 | Double glazing argon filled solar protection windows | 1 | 0.16 | 0.19 | window indentation 5cm | None | None | | 21 | 24 |
| | **Case 4** | 0.6 | 19 % | 6 | Double glazing argon filled solar protection windows | 1 | 0.16 | 0.19 | Window indentation 5cm | The manually controlled blinds between the outer panes are used when some one is in the room according to the occupancy profile. | None | | 21 | 24 |

| Simulation Cases of the old apartment building, IBES Project | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Cases | | Wall construction | | | | Window properties | | | | | | Setpoint | |
| | | Average U value ($W/m^2K$) | Window to wall ratio (%) | Air Leakage q50 ($m^3/h,m^2$) | Window Type | U value ($W/m^2K$) | Tsol | g-value | External shading | Integrated shading | Window opening | Heating setpoint °C | Cooling setpoint °C |
| **Automated Solutions** | **Case 5** | 0.6 | 19 % | 6 | Clear double glazing window | 2.5 | 0.7 | 0.76 | Window indentation 5cm | The electrically controlled blinds between the outer panes are used when the intensity of inside solar radiation on the windows from May to Sep exceeds 100 $W/m^2$ . | None | 21 | 24 |
| | **Case 6** | 0.6 | 19 % | 6 | Clear double glazing window | 2.5 | 0.7 | 0.76 | Window indentation 5cm | None | 10% of the window area of the largest window of each room opens when the outside air temperature is between 12 °C and 22°C and the zone air temperature exceeds 23 °C. | 21 | 24 |
| | **Case 7** | 0.6 | 19 % | 6 | Clear double glazing window | 2.5 | 0.7 | 0.76 | The electrically controlled awnings on the windows without a balcony open when the wind speed is less than 8 m/s and outdoor air temperature exceeds 15 °C meanwhile the outside solar radiation on vertical surface exceeds 100 $W/m^2$ | None | None | 21 | 24 |
| | **Case 8** | 0.6 | 19 % | 6 | Triple IGU with LowE #6 | Electrochromic, When the outdoor solar radiation on the vertical surface exceeds 450 $W/m^2$ the darkest state of the glazing is on. | | | Window indentation 5cm | None | None | 21 | 24 |
| | | | | | | 0.97 | 0.29-0.01 | 0.31-0.05 | | | | | |

The following cases are simulated in the new apartment building with and without the active cooling system:

- Base case
- Case 4
- Automated solutions (Case 5, Case 6, Case 7, Case 8)

| Simulation Cases of the new apartment building, IBES Project | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Cases | | Wall construction | | | | Window properties | | | | | | | Setpoint | |
| | | Average U value ($W/m^2K$) | Window to wall ratio (%) | Air Leakage q50 ($m^3/h,m^2$) | Window Type | U value ($W/m^2K$) | Tsol | g-value | External shading | Integrated shading | Window opening | | Heating setpoint °C | Cooling setpoint °C |
| **Base Case** | **Case 1** | 0.17 | 15 % | 4 | Triple glazing argon filled lowE window | 1 | 0.3 | 0.35 | window indentation 30cm | None | None | | 21 | 24 |
| **Passive Solutions** | **Case 2** | 0.17 | 15 % | 4 | Triple glazing argon filled lowE window | 1 | 0.3 | 0.35 | window indentation 30cm | The manually controlled blinds between the outer panes are used when some one is in the room according to the occupancy profile. | None | | 21 | 24 |
| | **Case 3** | 0.17 | 15 % | 4 | Double glazing argon filled solar protection windows | 1 | 0.16 | 0.19 | window indentation 30cm | None | None | | 21 | 24 |
| | **Case 4** | 0.17 | 15 % | 4 | Double glazing argon filled solar protection windows | 1 | 0.16 | 0.19 | Window indentation 30cm | The manually controlled blinds between the outer panes are used when some one is in the room according to the occupancy profile. | None | | 21 | 24 |

## Simulation Cases of the new apartment building, IBES Project

| Cases | | Wall construction | | | Window properties | | | | | | Setpoint | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Average U value ($W/m^2K$) | Window to wall ratio (%) | Air Leakage q50 ($m^3/h,m^2$) | Window Type | U value ($W/m^2K$) | Tsol | g-value | External shading | Integrated shading | Window opening | Heating setpoint °C | Cooling setpoint °C |
| **Automated Solutions** | **Case 5** | 0.17 | 15 % | 4 | Triple glazing argon filled lowE window | 1 | 0.3 | 0.35 | Window indentation 30cm | The electrically controlled blinds between the outer panes are used when the intensity of inside solar radiation on the windows from May to Sep exceeds 100 $W/m^2$ . | None | 21 | 24 |
| | **Case 6** | 0.17 | 15 % | 4 | Triple glazing argon filled lowE window | 1 | 0.3 | 0.35 | Window indentation 30cm | None | 10% of the window area of the largest window of each room opens when the outside air temperature is between 12 °C and 22°C and the zone air temperature exceeds 23 °C. | 21 | 24 |
| | **Case 7** | 0.17 | 15 % | 4 | Triple glazing argon filled lowE window | 1 | 0.3 | 0.35 | The electrically controlled awnings on the windows without a balcony open when the wind speed is less than 8 m/s and outdoor air temperature exceeds 15 °C meanwhile the outside solar radiation on vertical surface exceeds 100 $W/m^2$ | None | None | 21 | 24 |
| | **Case 8** | 0.17 | 15 % | 4 | Triple IGU with LowE #6 | Electrochromic, When the outdoor solar radiation on the vertical surface exceeds 450 $W/m^2$ the darkest state of the glazing is on. | | | Window indentation 30cm | None | None | 21 | 24 |
| | | | | | | 0.97 | 0.29-0.01 | 0.31-0.05 | | | | | |

# Results of the cases with the active cooling system



Contact: juha.jokisalo@aalto.fi

# Break down of Energy Consumption, old apartment building

| Old Apartment Building, Annual energy consumption break down, kWh/m$^2$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Systems | Base case | Passive and Manual solutions | | | Automated solutions | | | |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| | No solar protection | Manual blinds | Solar protection windows | Manual blinds + Solar protection windows | Electrical blinds | Electrical openable windows | Electrical awnings | Electrochromic windows |
| **District heating** | | | | | | | | |
| Space heating + AHU | 136.1 | 136.7 | 131.7 | 131.9 | 138.1 | 136.5 | 136.3 | 130.6 |
| DWH | 42.4 | 42.4 | 42.4 | 42.4 | 42.4 | 42.4 | 42.4 | 42.4 |
| **Total** | 178.5 | 179.1 | 174.1 | 174.3 | 180.5 | 178.9 | 178.7 | 173.0 |
| **Electricity** | | | | | | | | |
| Space cooling electricity | 2.1 | 1.8 | 0.8 | 0.7 | 1.1 | 0.8 | 1.5 | 0.8 |
| HVAC aux | 2.3 | 2.3 | 2.3 | 2.3 | 2.3 | 2.3 | 2.3 | 2.3 |
| Lighting | 7.6 | 7.6 | 7.6 | 7.6 | 7.6 | 7.6 | 7.6 | 7.6 |
| Equipment | 21.0 | 21.0 | 21.0 | 21.0 | 21.0 | 21.0 | 21.0 | 21.0 |
| **Total** | 33.0 | 32.7 | 31.7 | 31.6 | 32.0 | 31.7 | 32.4 | 31.7 |

| The percentage of effect of each solution on district heating and space cooling electricity in comparison to Case 1, The old apartment building | | | | | | | |
|---|---|---|---|---|---|---|---|
| Systems | Case 2 | Case 3 | Case 4 | Case 5 | Case 6 | Case 7 | Case 8 |
| District Space Heating + AHU | 0.4 % | -3.2 % | -3.1 % | 1.5 % | 0.3 % | 0.1 % | -4.0 % |
| Space Cooling Electricity | -14.3 % | -61.9 % | -66.7 % | -47.6 % | -61.9 % | -28.6 % | -61.9 % |

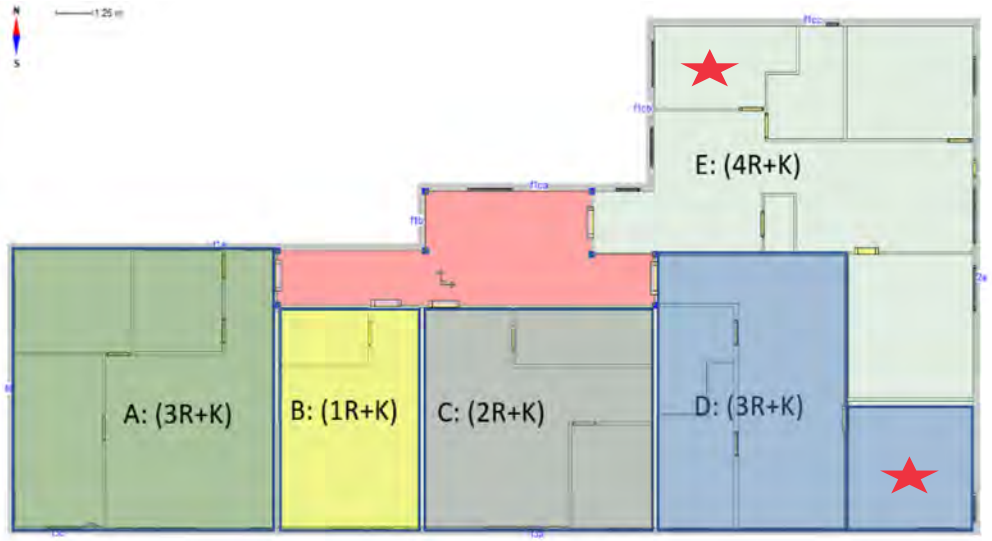
# Break down of Energy Consumption, new apartment building

| New Apartment Building, Annual energy consumption break down, kWh/m2 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Systems | Base case | Passive and Manual solutions | | | Automated solutions | | | |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| | No solar protection | Manual blinds | Solar protection windows | Manual blinds + Solar protection windows | Electrical blinds | Electrical openable windows | Electrical awnings | Electrochromic windows |
| **District heating** | | | | | | | | |
| Space heating + AHU | 34.9 | 35.0 | 36.1 | 36.1 | 35.0 | 35.1 | 34.9 | 35.7 |
| DWH | 42.4 | 42.4 | 42.4 | 42.4 | 42.4 | 42.4 | 42.4 | 42.4 |
| **Total** | 77.3 | 77.4 | 78.5 | 78.5 | 77.4 | 77.5 | 77.3 | 78.1 |
| **Electricity** | | | | | | | | |
| Space cooling electricity | 2.3 | 2.1 | 1.6 | 1.5 | 1.9 | 0.4 | 2.0 | 1.8 |
| HVAC aux | 5.8 | 5.8 | 5.8 | 5.8 | 5.8 | 5.8 | 5.8 | 5.8 |
| Lighting | 7.6 | 7.6 | 7.6 | 7.6 | 7.6 | 7.6 | 7.6 | 7.6 |
| Equipment | 21.0 | 21.0 | 21.0 | 21.0 | 21.0 | 21.0 | 21.0 | 21.0 |
| **Total** | 36.7 | 36.5 | 36.0 | 35.9 | 36.3 | 34.8 | 36.4 | 36.2 |

| **The percentage of effect of each solution on district heating and space cooling electricity in comparison to Case 1, The new apartment building** | | | | | | | |
|---|---|---|---|---|---|---|---|
| Systems | Case 2 | Case 3 | Case 4 | Case 5 | Case 6 | Case 7 | Case 8 |
| District Space Heating + AHU | 0.3 % | 3.4 % | 3.4 % | 0.3 % | 0.6 % | 0.0 % | 2.3 % |
| Space Cooling Electricity | -8.7 % | -30.4 % | -34.8 % | -17.4 % | -82.6 % | -13.0 % | -21.7 % |

# Indoor Air Temperature Results

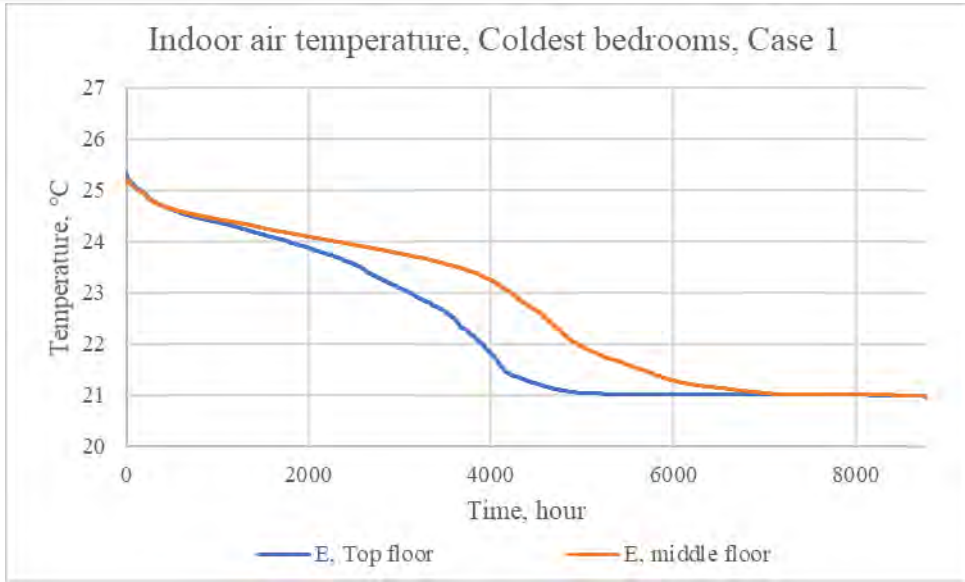
- The coldest and warmest bedrooms of the building are defined based on the degree hour above 24 °C. Bedroom no.2 of apartment D is the warmest and bedroom no.2 of the apartment E is the coldest.
- The annual duration curves in the coldest and warmest bedrooms of the building are analyzed.
- The indoor air temperature in the coldest and warmest bedrooms of the building in the typical days of each season, are studied.

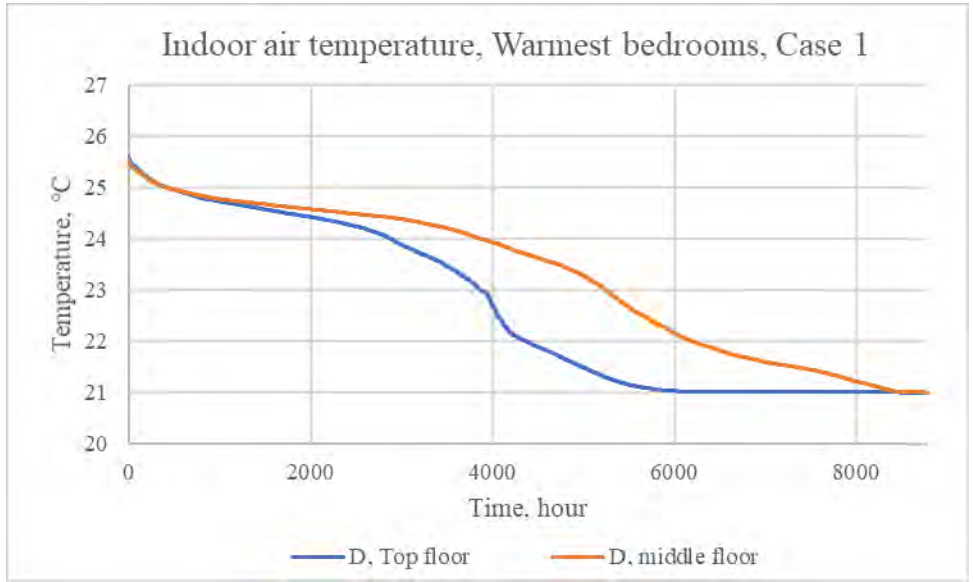



- ✓ Apartment D's bedroom is located on the south side of the building and has two windows on its south and east walls.
- ✓ Apartment E's bedroom is located on the north side of the building and has a window on its west wall.

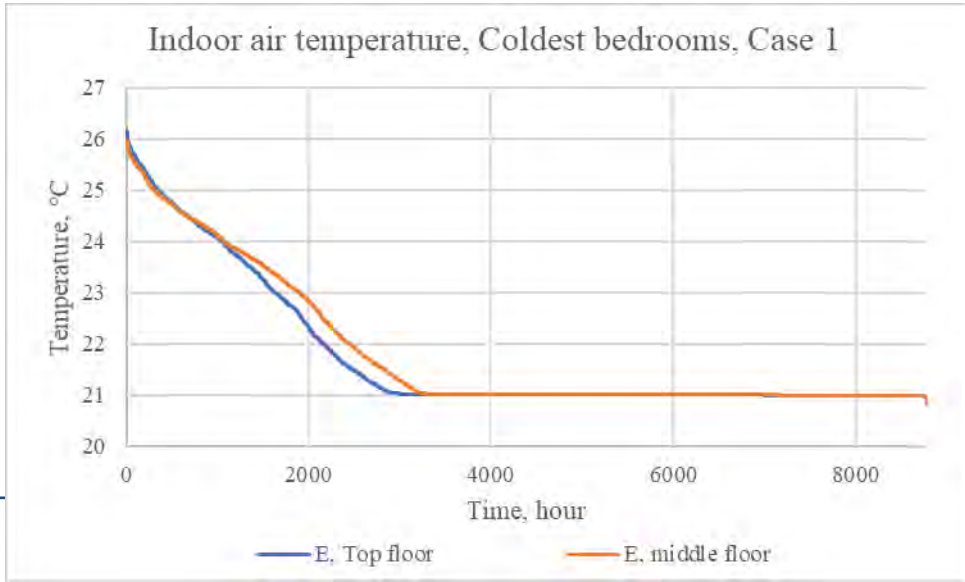
# Indoor Air Temperature Results

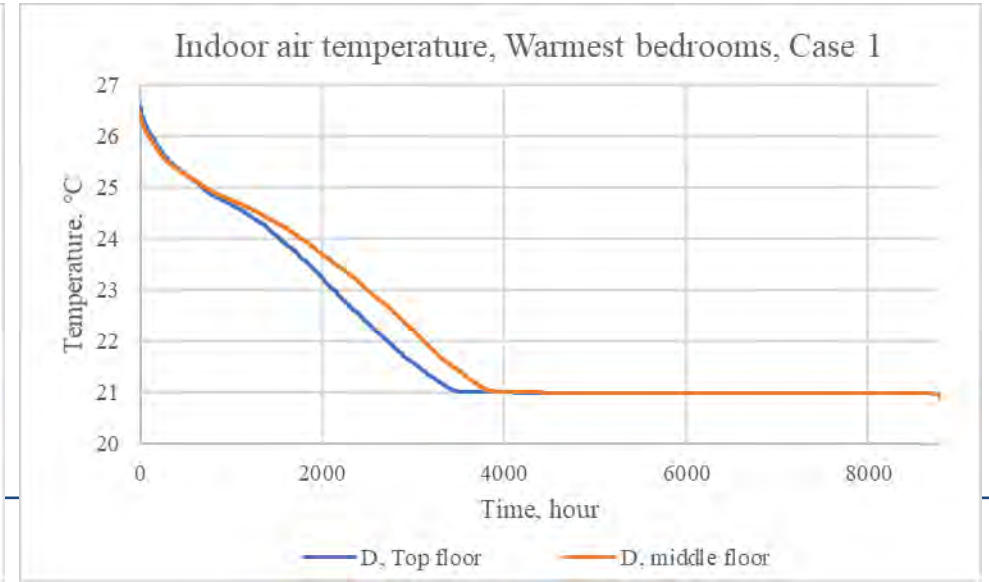
New apartment building



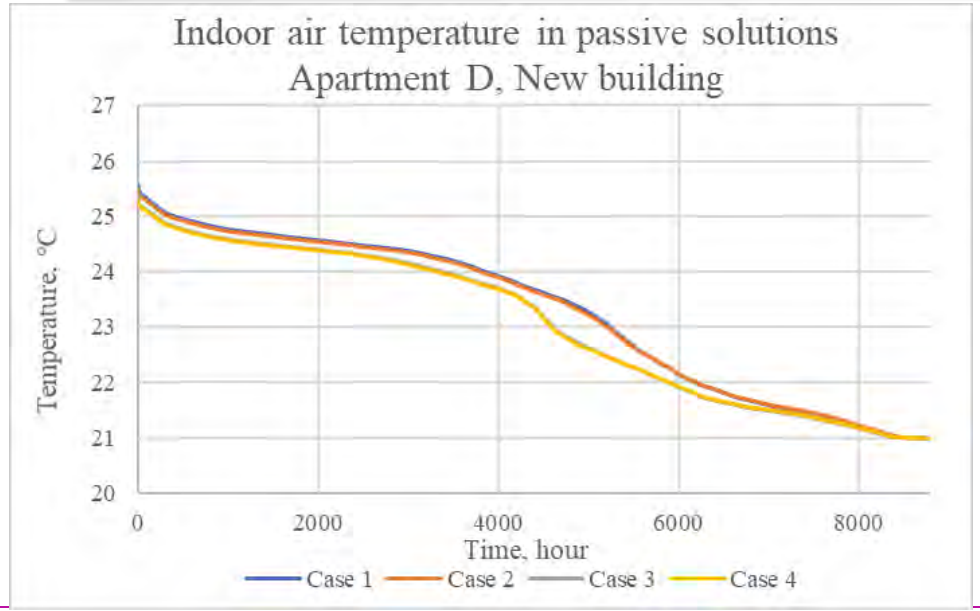

Old apartment building



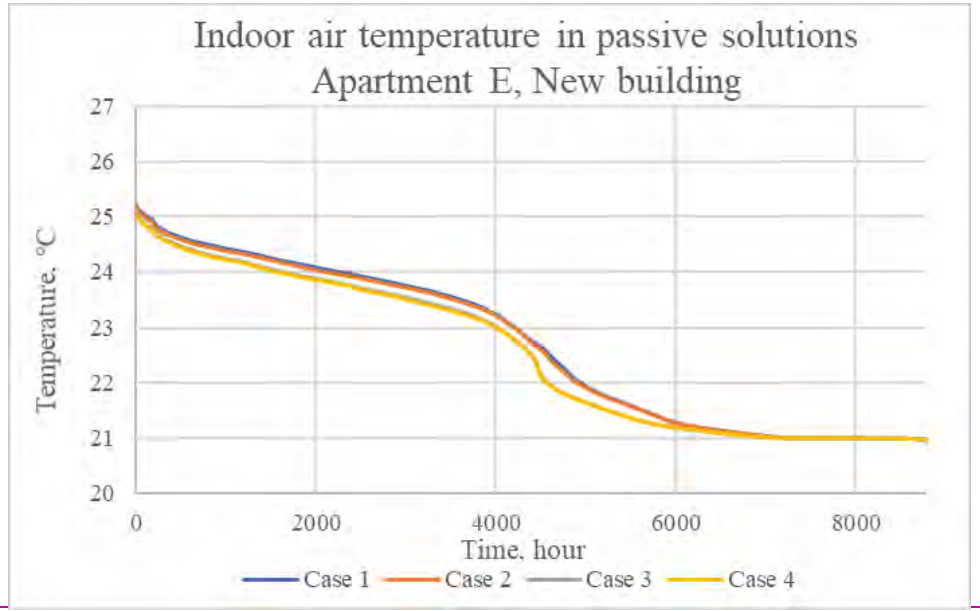




The results of middle floor in the coldest and warmest bedrooms are analyzed, since the number of middle floors is 3 and the temperature is higher.

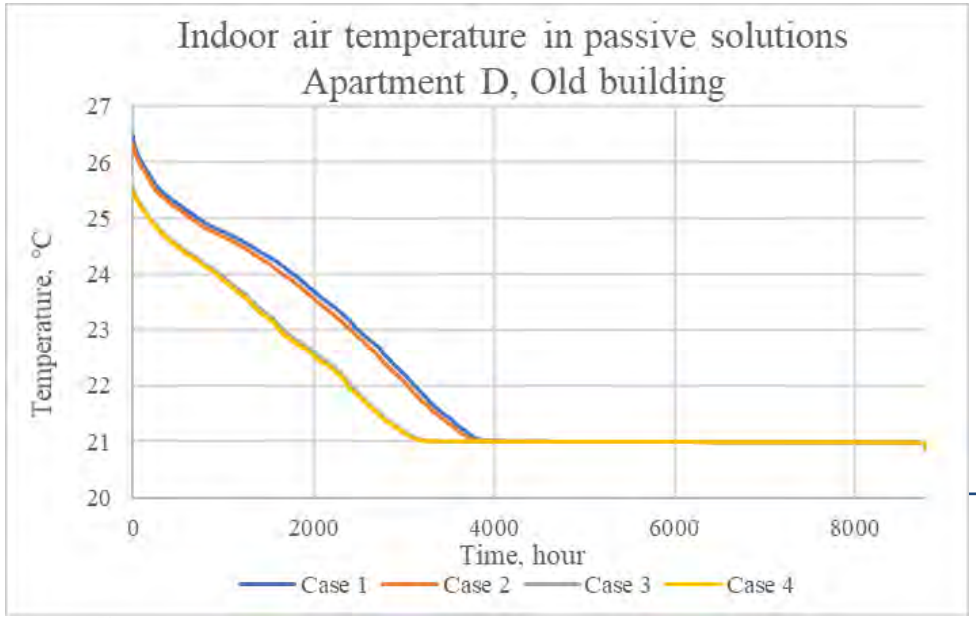
# Indoor air temperatures, Passive solutions: (1. Base case, 2.Manual blinds, 3.SP windows, 4. manual blinds and SP windows)



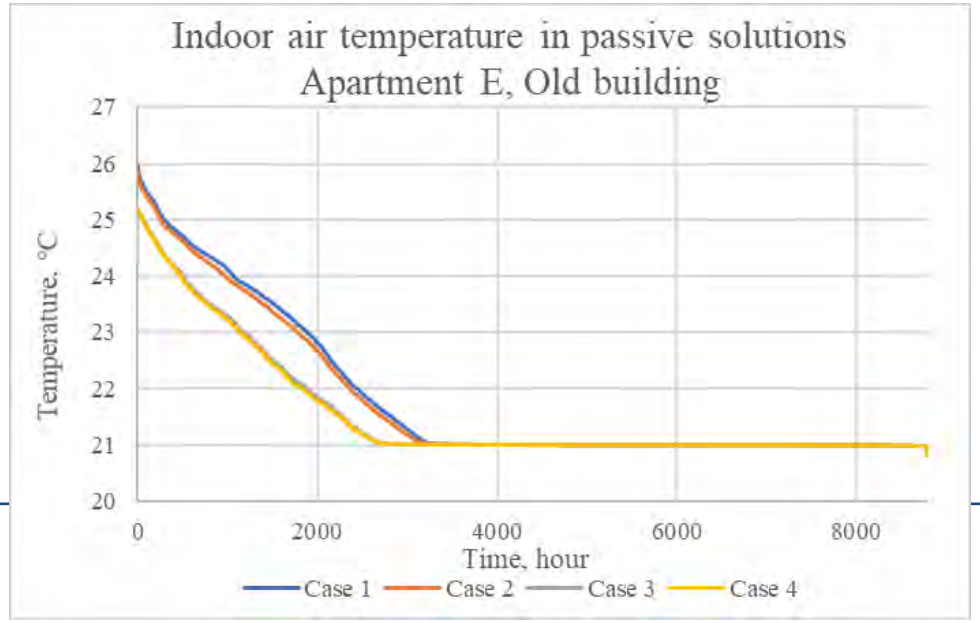

| Degree hours above 24 °C | |
|---|---|
| Case 1 | 2320 |
| Case 2 | 2196 |
| Case 3 | 1617 |
| Case 4 | 1550 |




| Degree hours above 24 °C | |
|---|---|
| Case 1 | 952 |
| Case 2 | 862 |
| Case 3 | 602 |
| Case 4 | 562 |

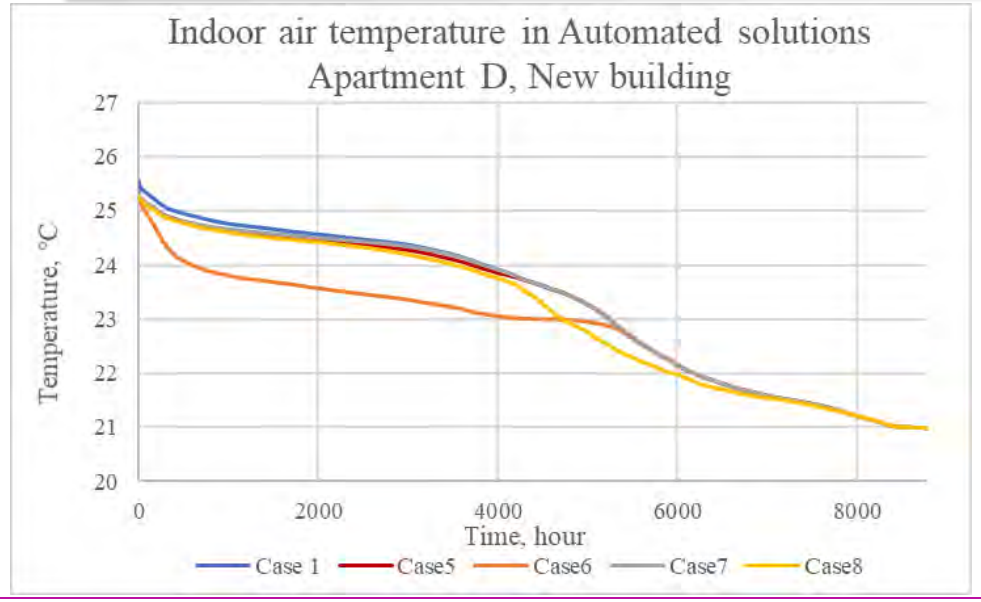



| Degree hours above 24 °C | |
|---|---|
| Case 1 | 1642 |
| Case 2 | 1472 |
| Case 3 | 577 |
| Case 4 | 542 |

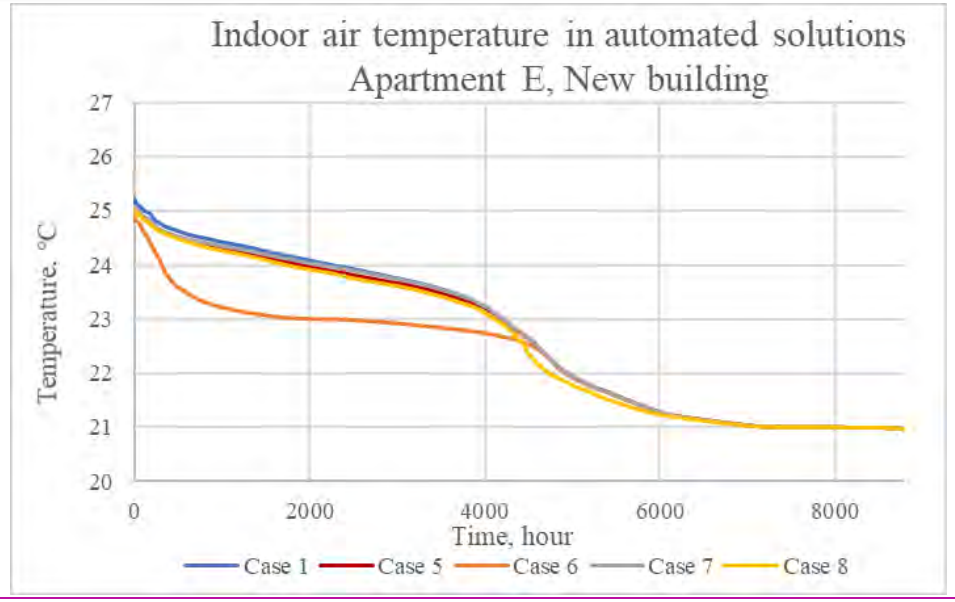



| Degree hours above 24 °C | |
|---|---|
| Case 1 | 793 |
| Case 2 | 681 |
| Case 3 | 269 |
| Case 4 | 251 |

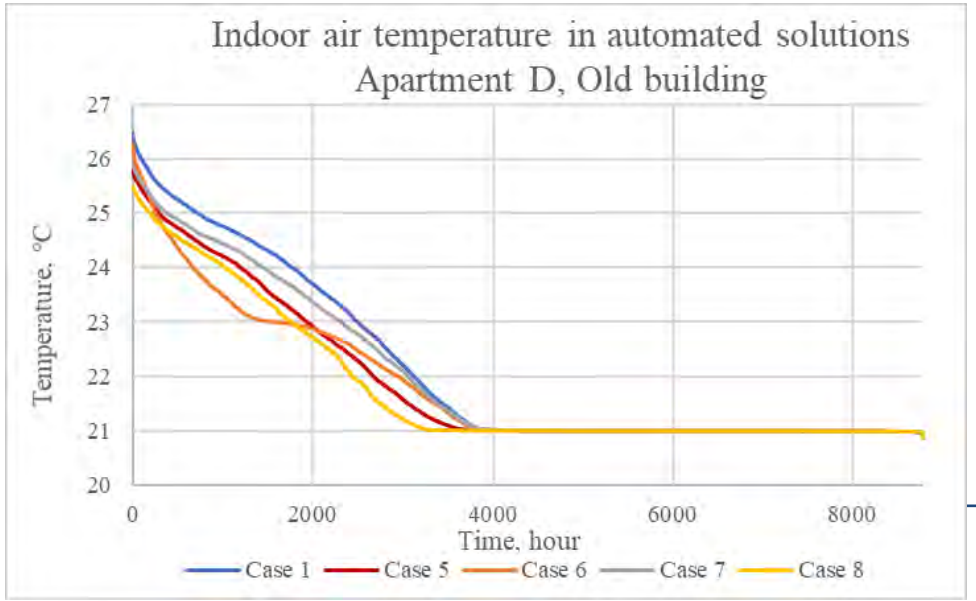
# Indoor air temperatures, Automated solutions: (5. El. blinds, 6.El. openings, 7.El. awnings, 8.Electrochromic windows)



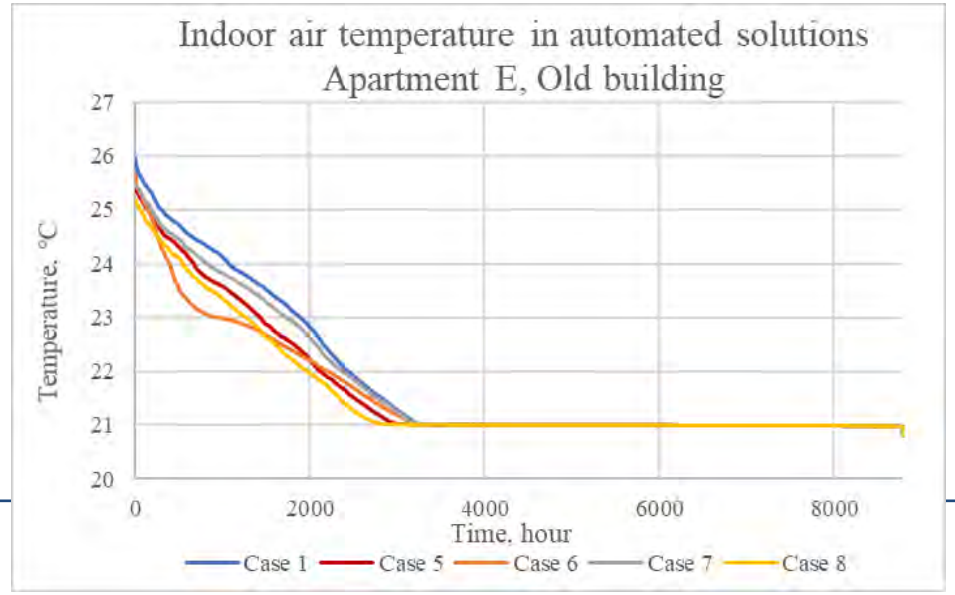

| Degree hours above 24 °C | |
|---|---|
| Case 1 | 2320 |
| Case 5 | 1889 |
| Case 6 | 278 |
| Case 7 | 2034 |
| Case 8 | 1694 |




| Degree hours above 24 °C | |
|---|---|
| Case 1 | 952 |
| Case 5 | 703 |
| Case 6 | 145 |
| Case 7 | 785 |
| Case 8 | 630 |



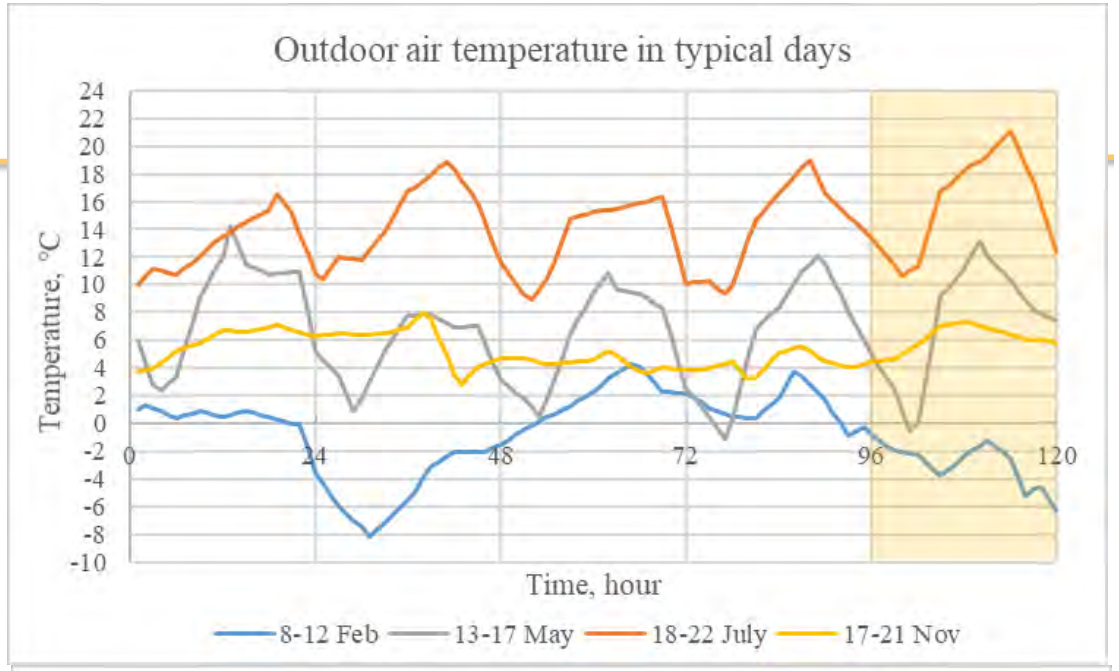

| Degree hours above 24 °C | |
|---|---|
| Case 1 | 1642 |
| Case 5 | 815 |
| Case 6 | 581 |
| Case 7 | 1056 |
| Case 8 | 610 |

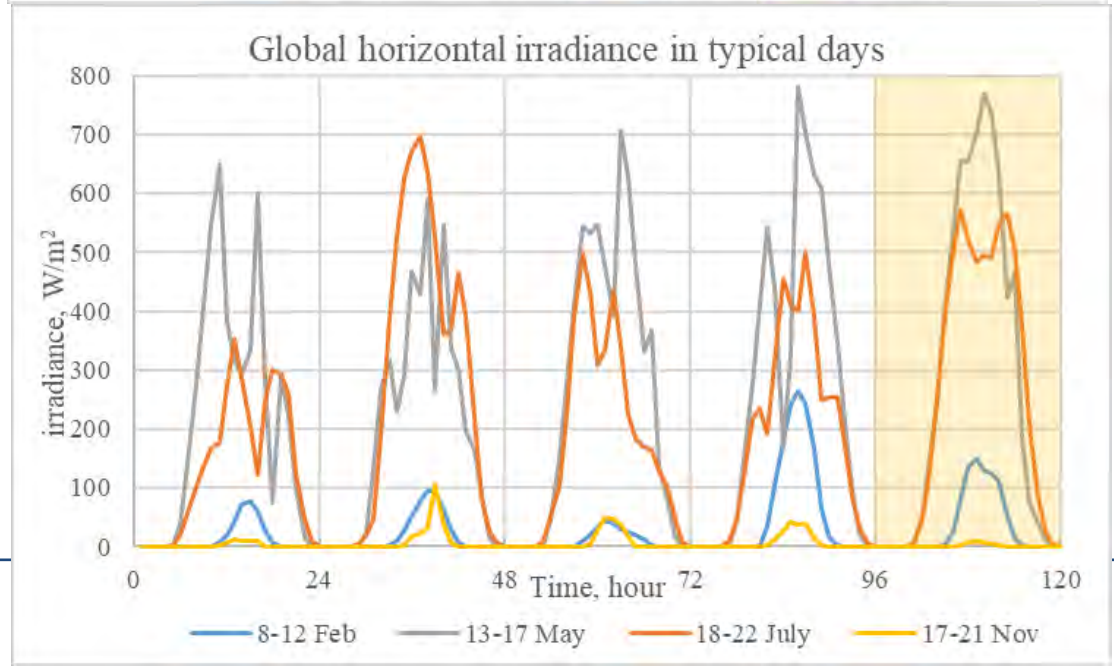



| Degree hours above 24 °C | |
|---|---|
| Case 1 | 793 |
| Case 5 | 415 |
| Case 6 | 297 |
| Case 7 | 510 |
| Case 8 | 280 |

# Typical days in each season

- The average temperature of December, January, February, and March (winter) is calculated and compared to the daily average temperature of each day in these months, the closest number to it, is selected as the typical day in the winter Which is Feb 12.
- The same calculation is done in April and may and the typical day in the spring is May 17.
- The same calculation is done in June, July, and August and the typical day in the Summer is July 22.
- The same calculation is done in September, October, and November and the typical day in the autumn is Nov 21.



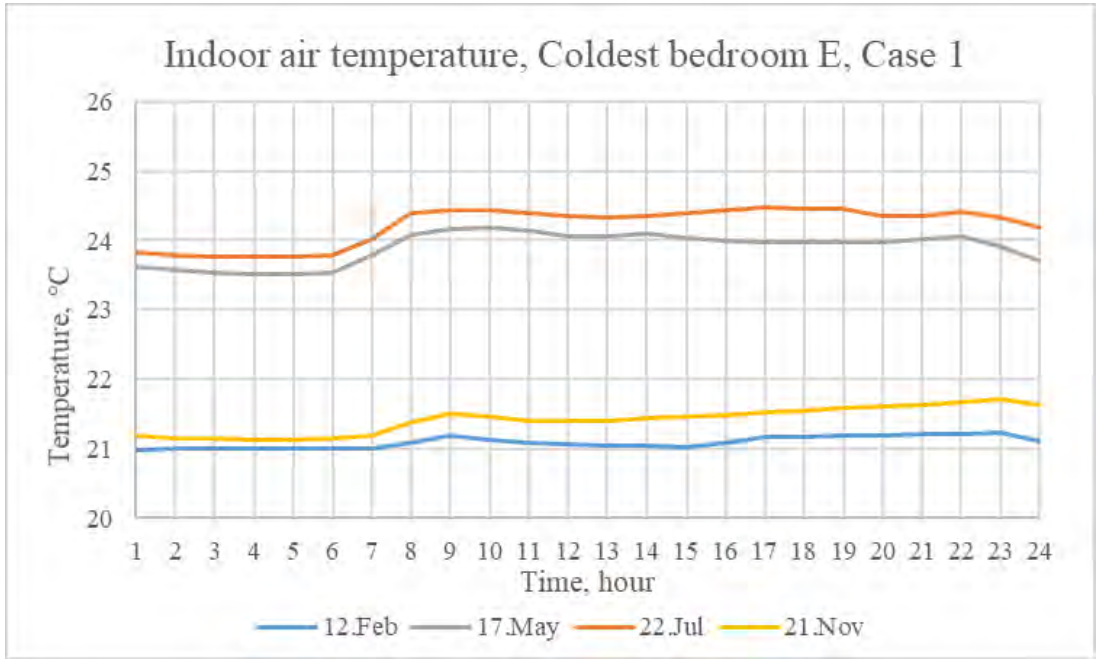



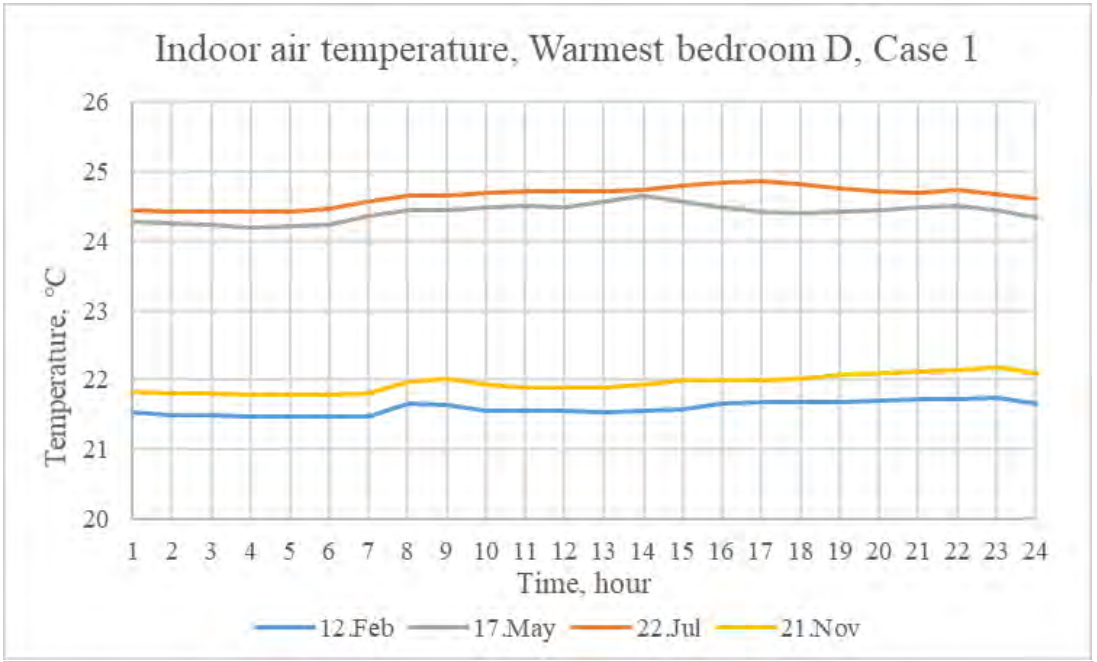

The yellow highlight in the graphs shows the typical days.

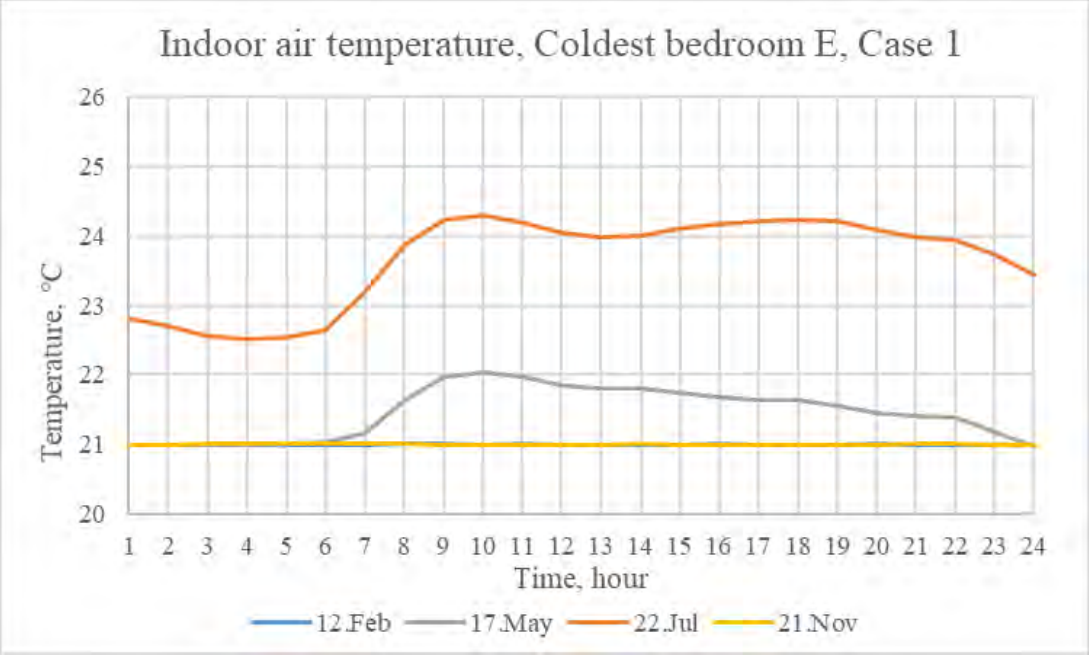
New apartment building

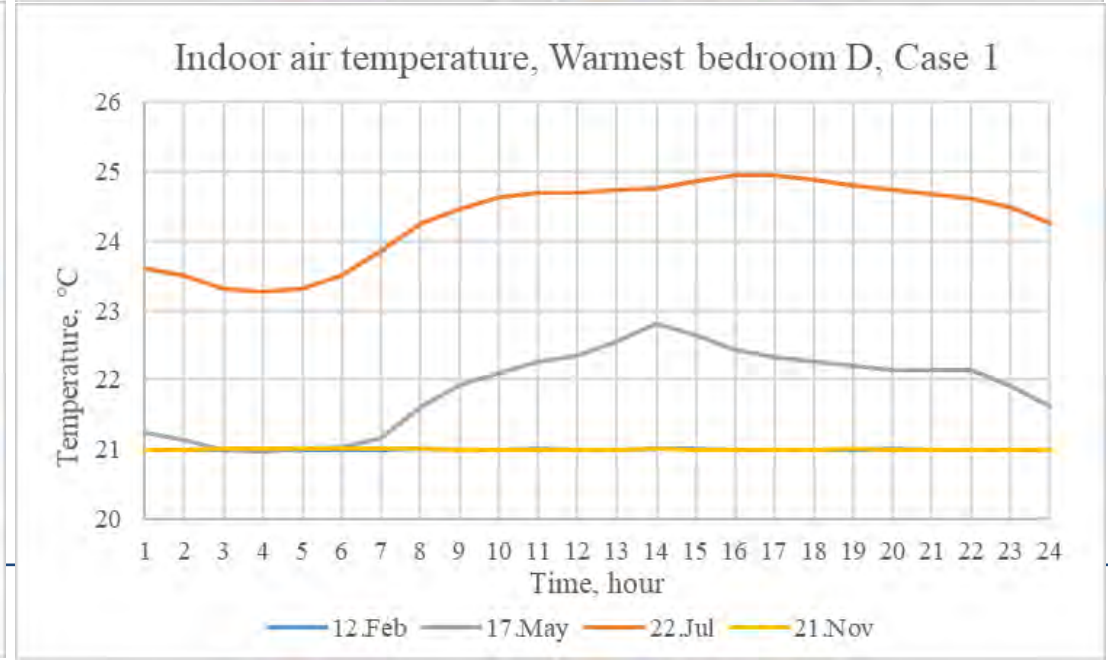
Old apartment building

Base case

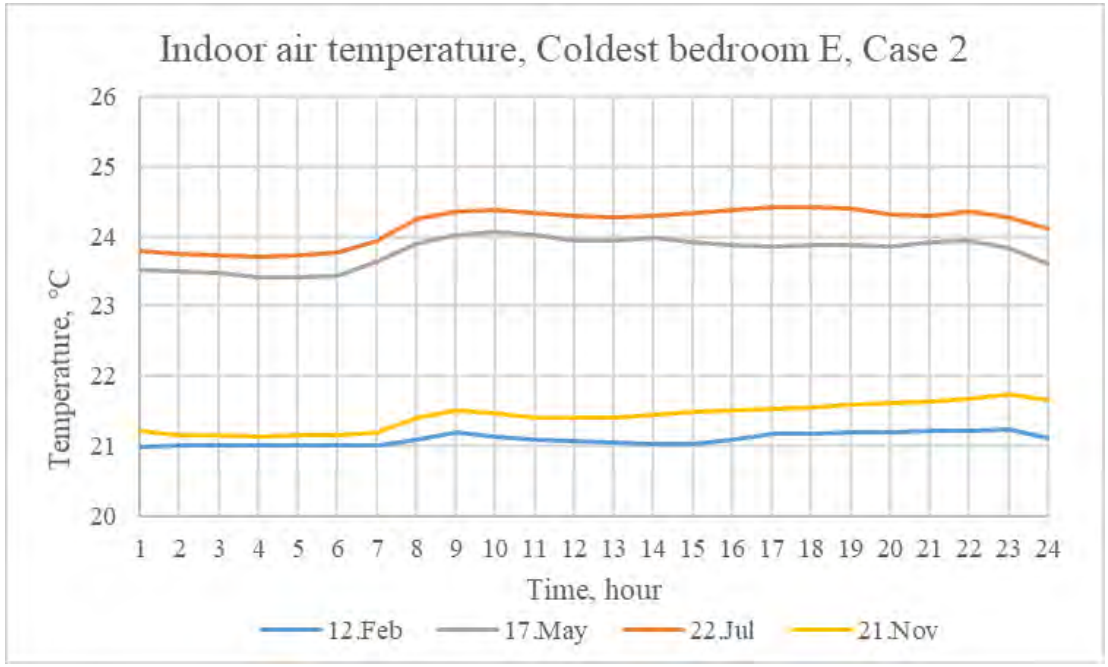


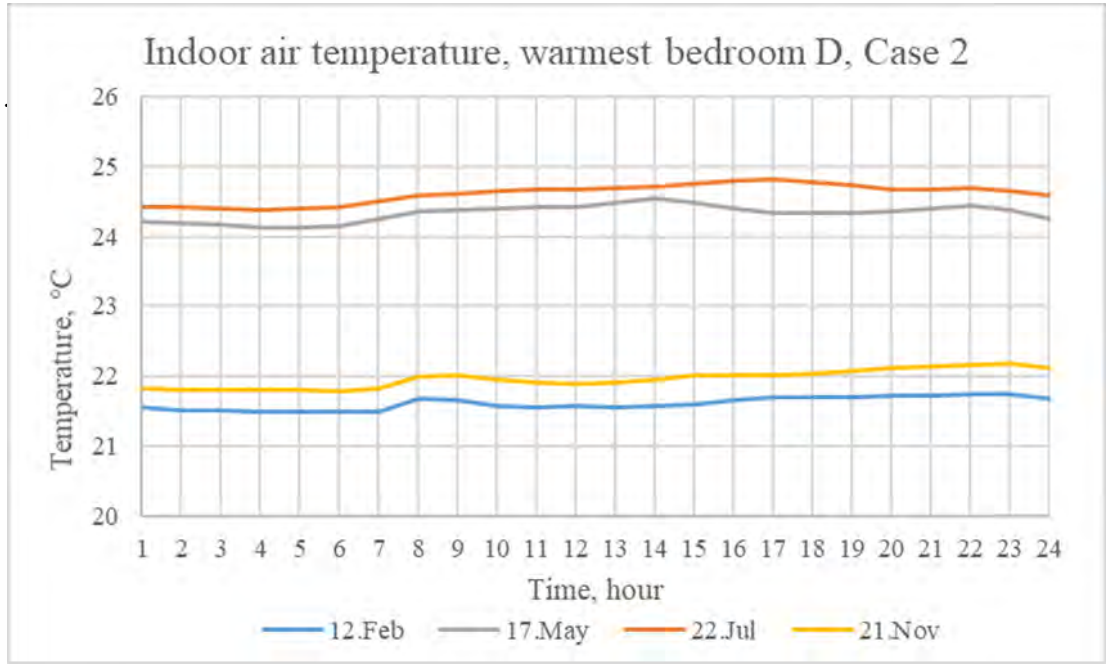



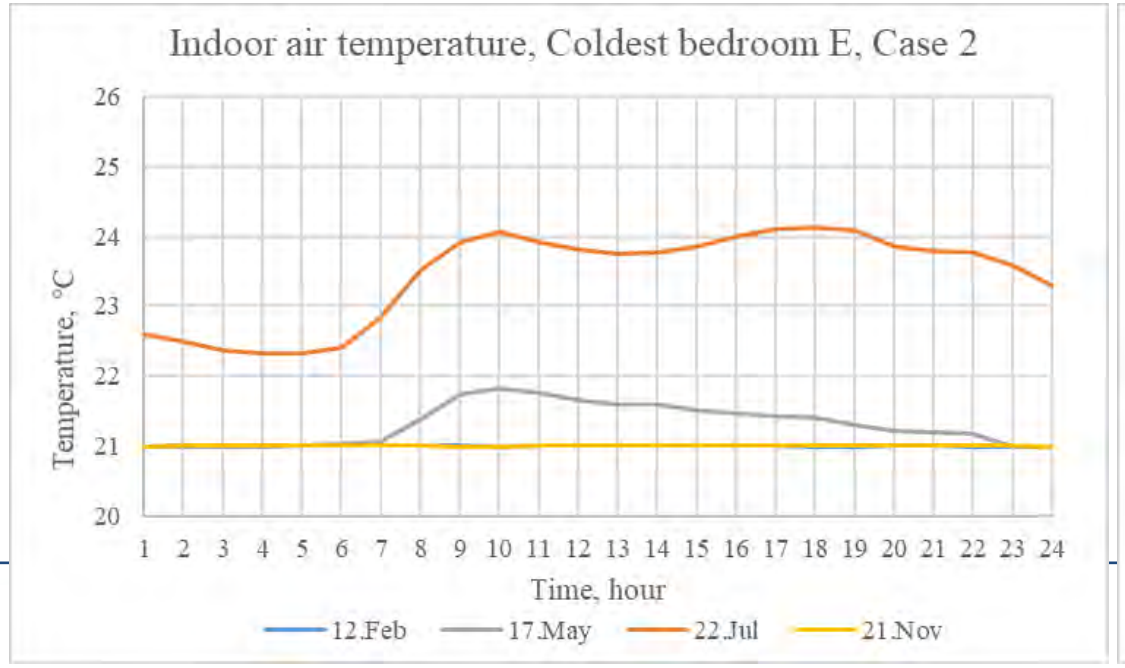



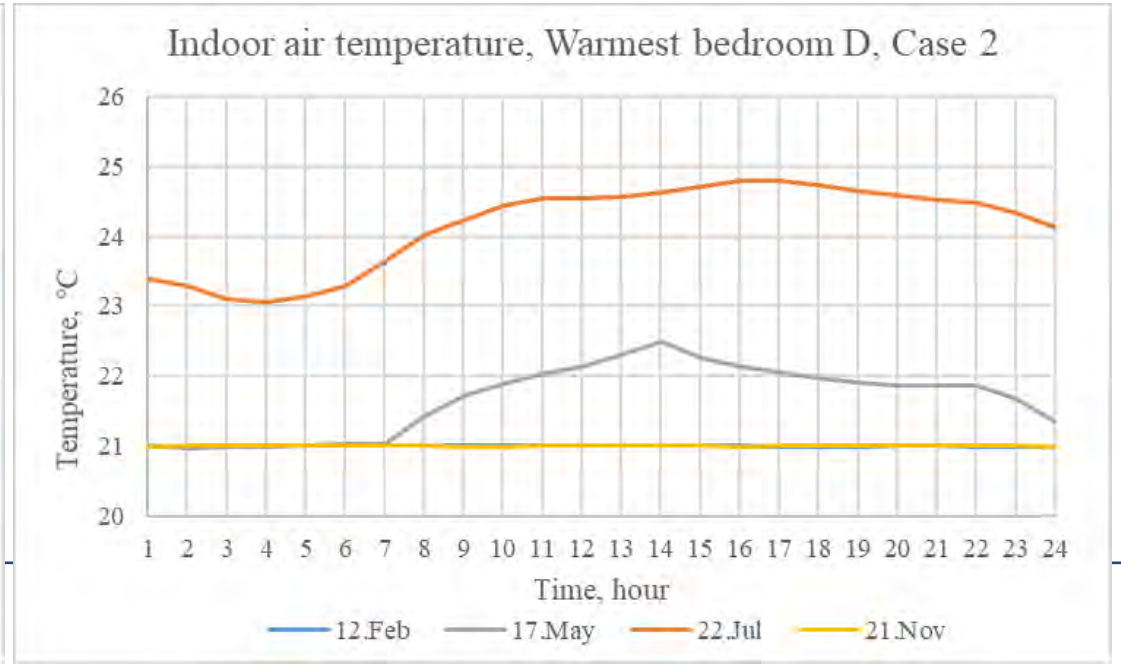
New apartment building



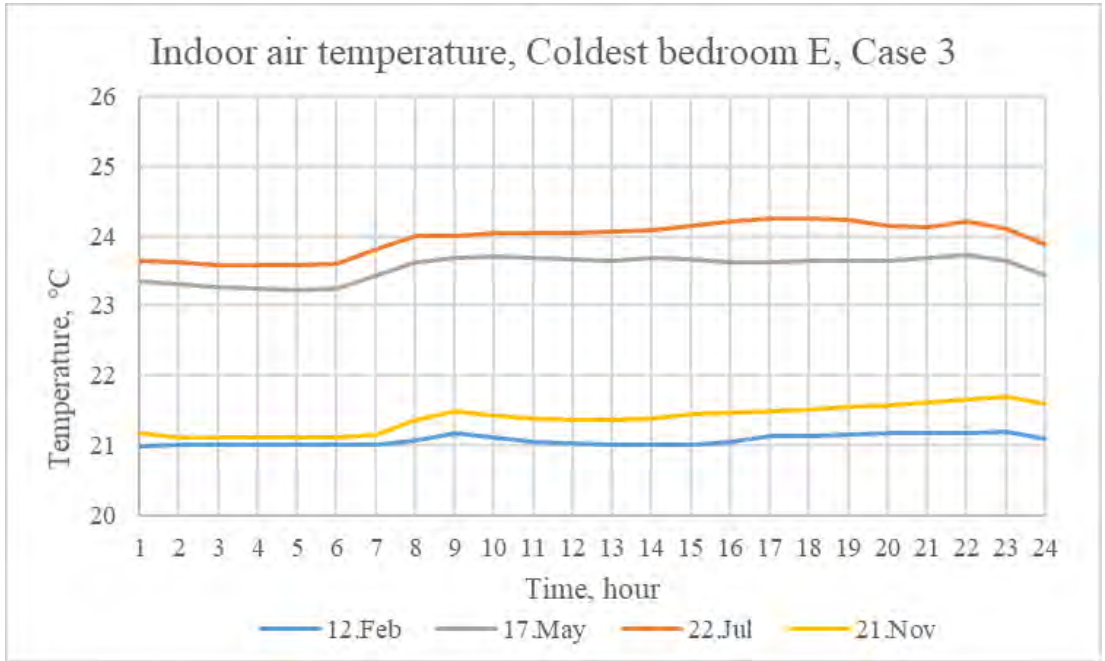

Old apartment building

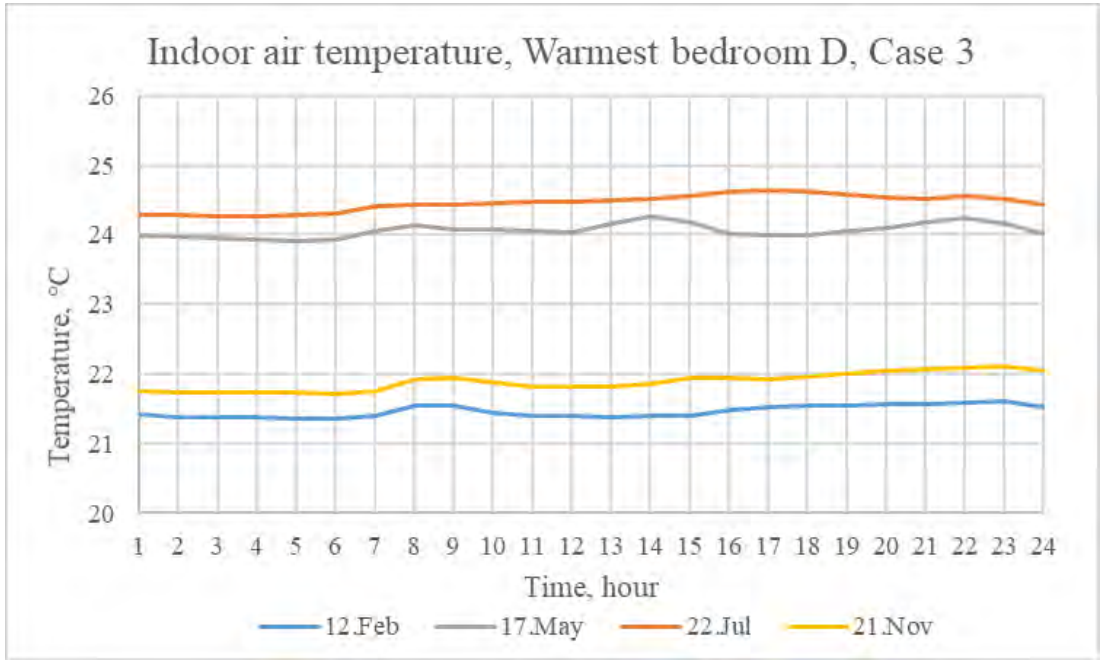


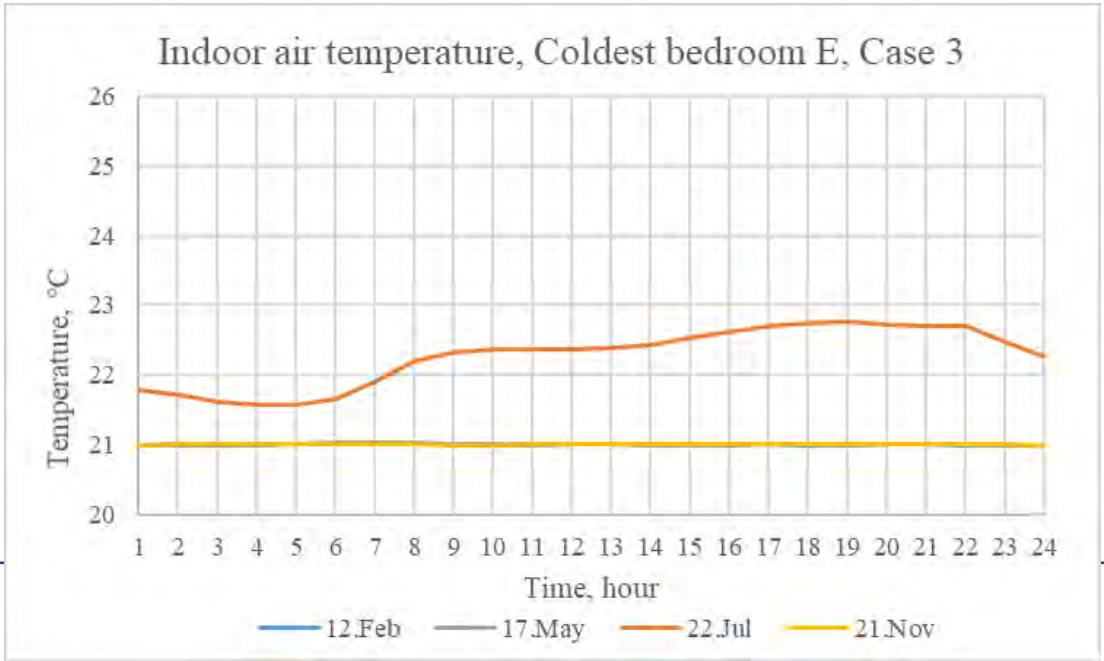

Case 2, Manual blinds

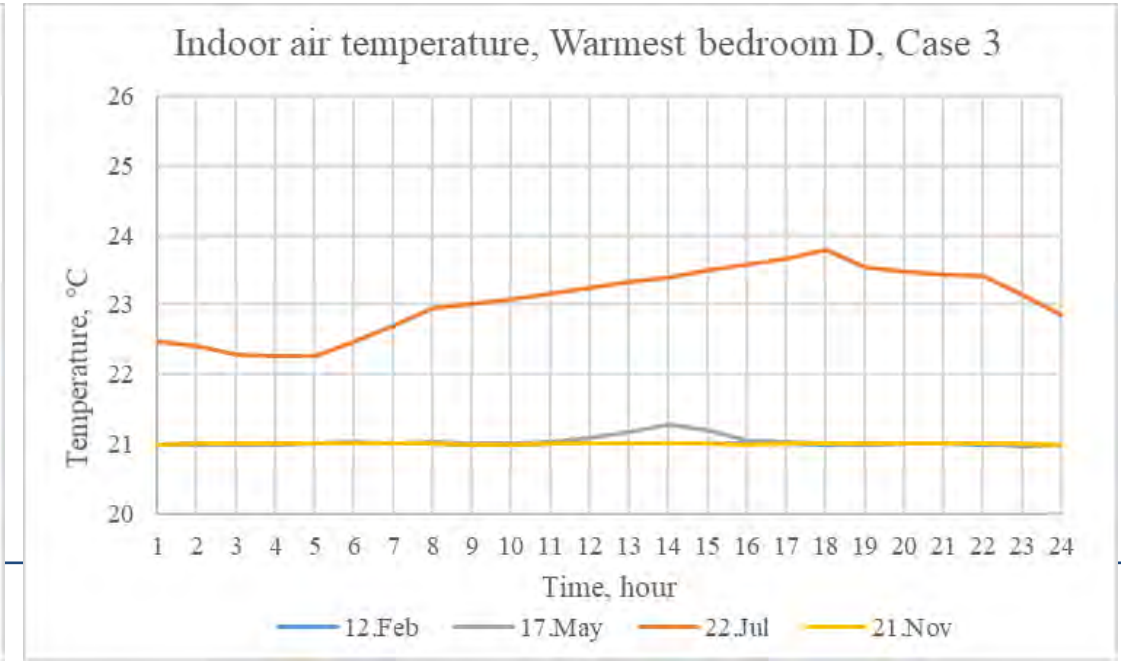
New apartment building




Old apartment building

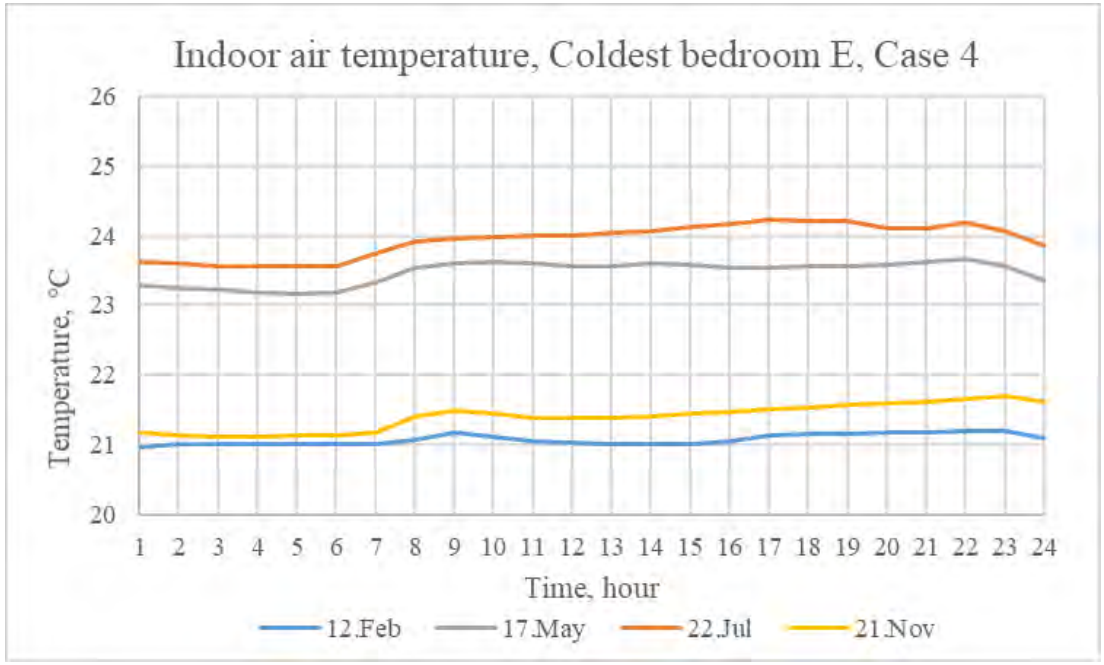


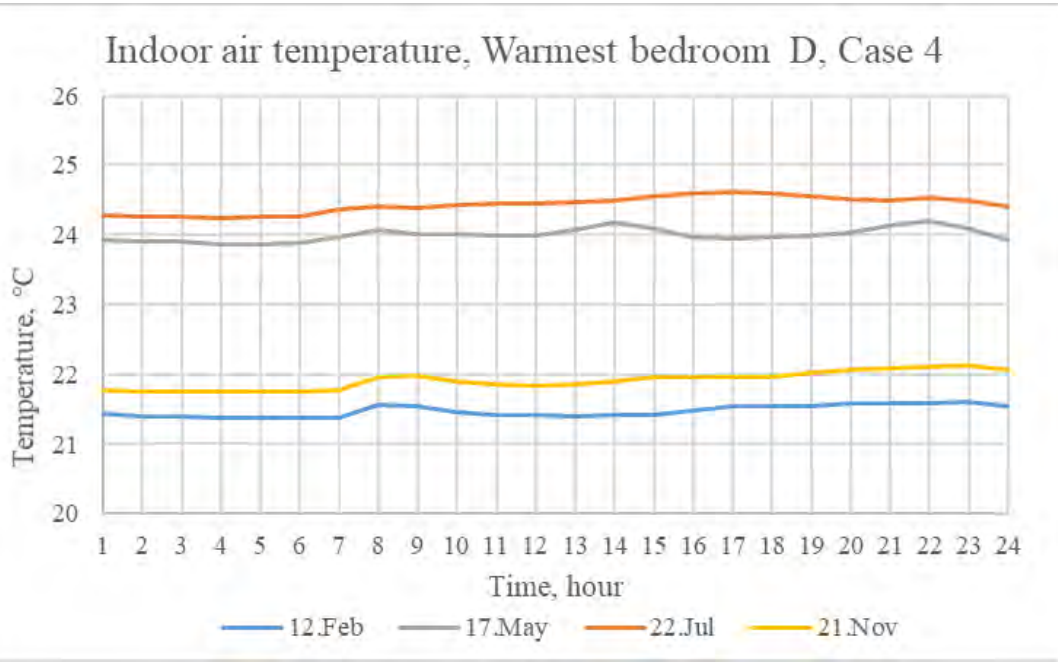




Case 3, Solar protection windows

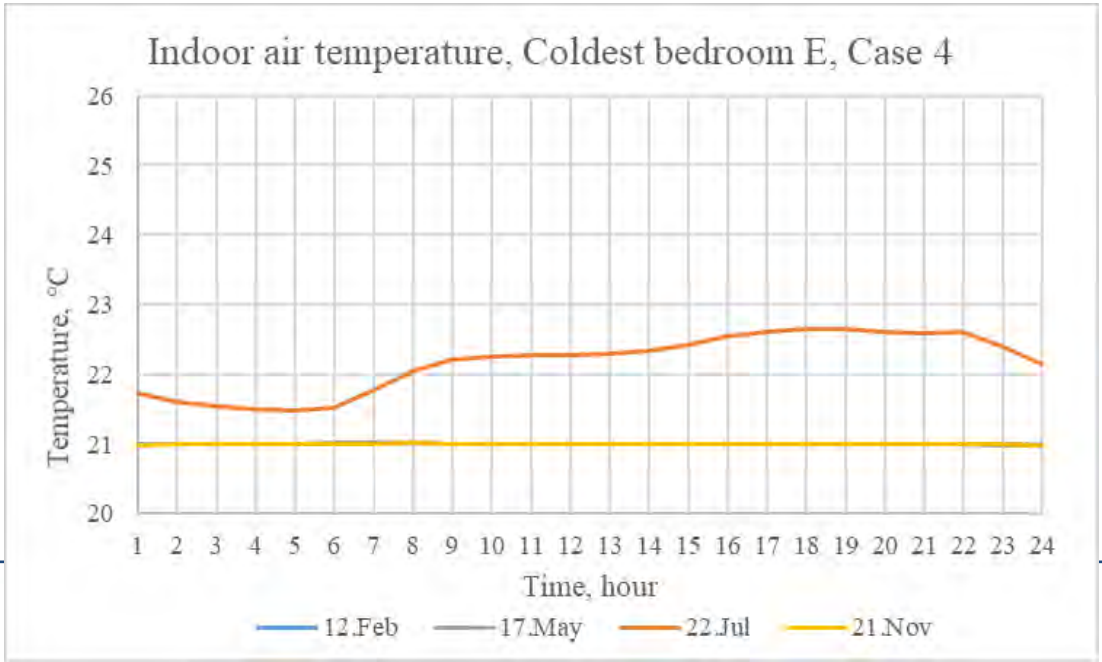
Case 4, Manual blinds + Solar protection windows

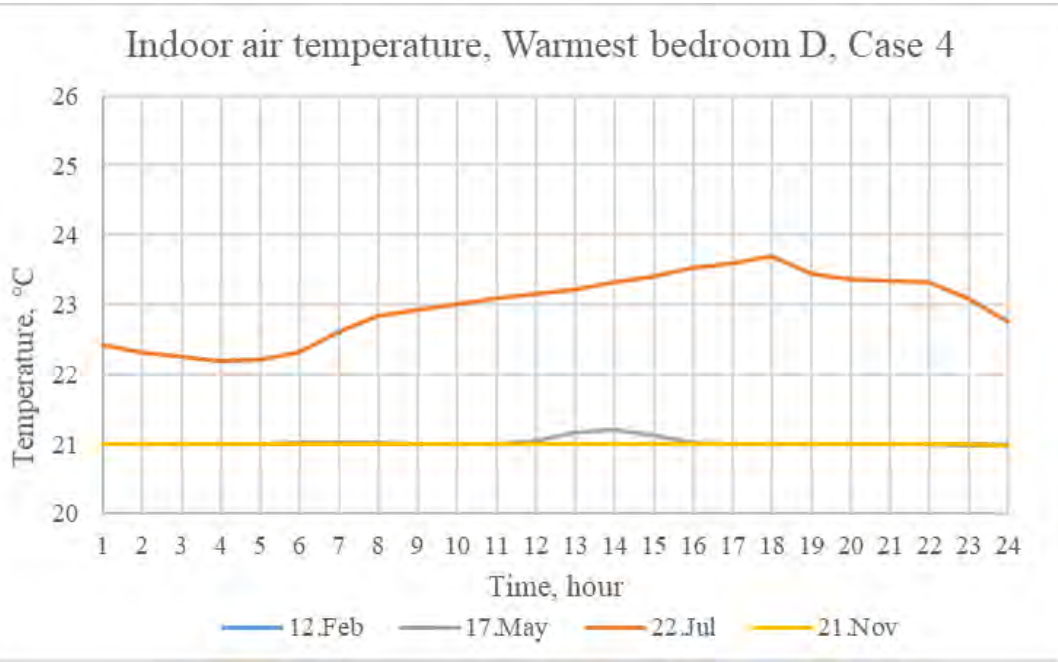
New apartment building



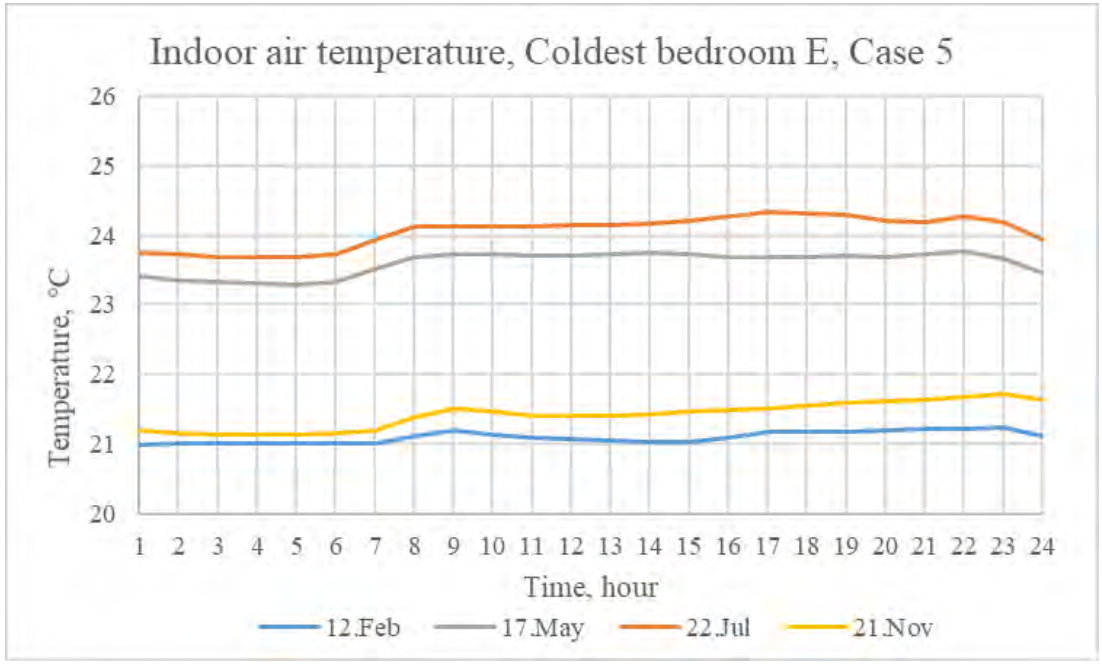



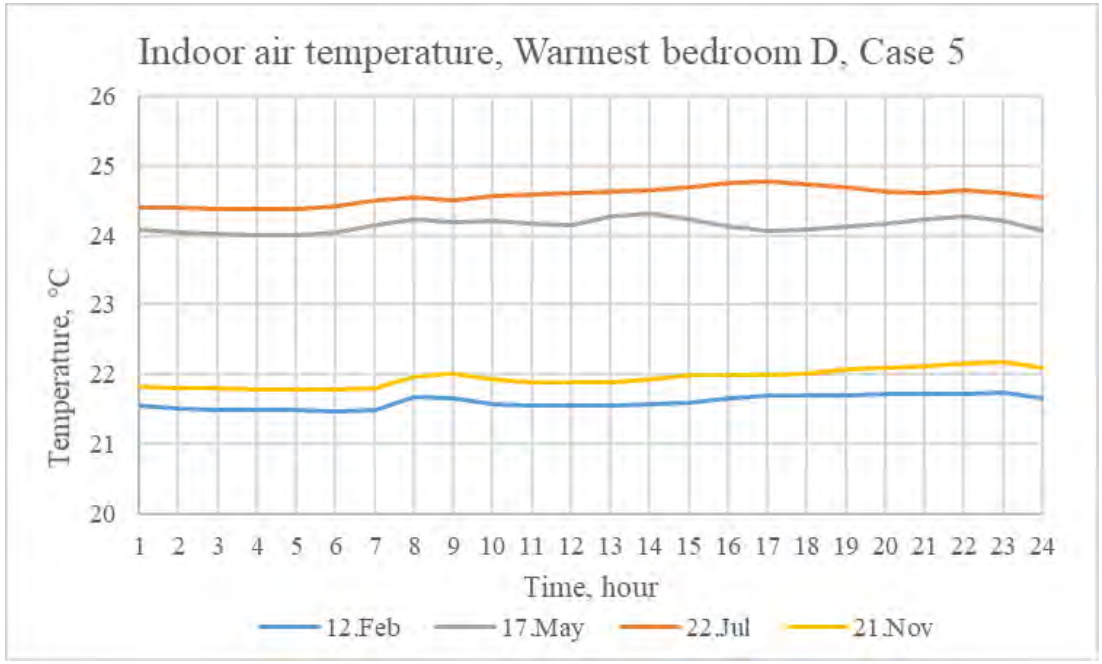

Old apartment building



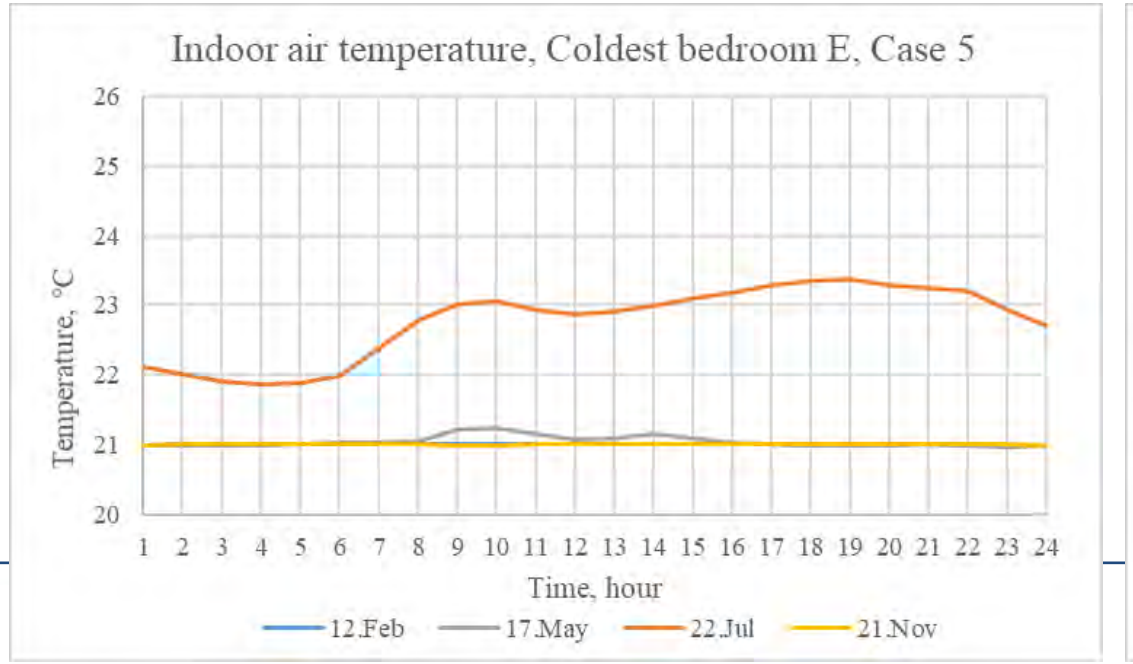



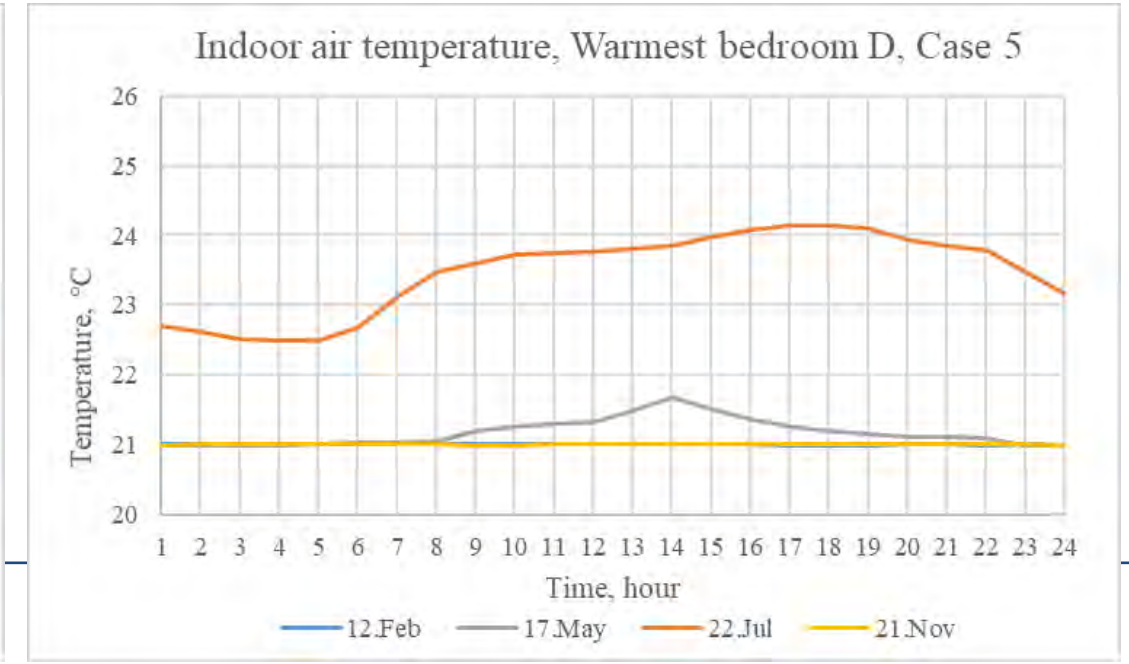
New apartment building



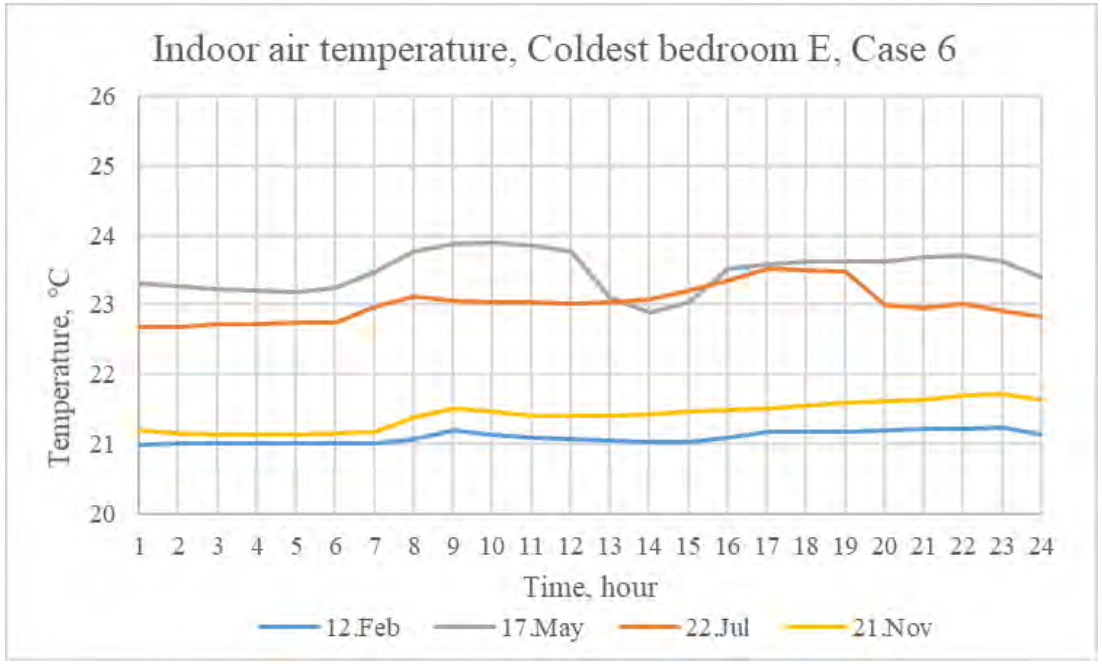

Old apartment building

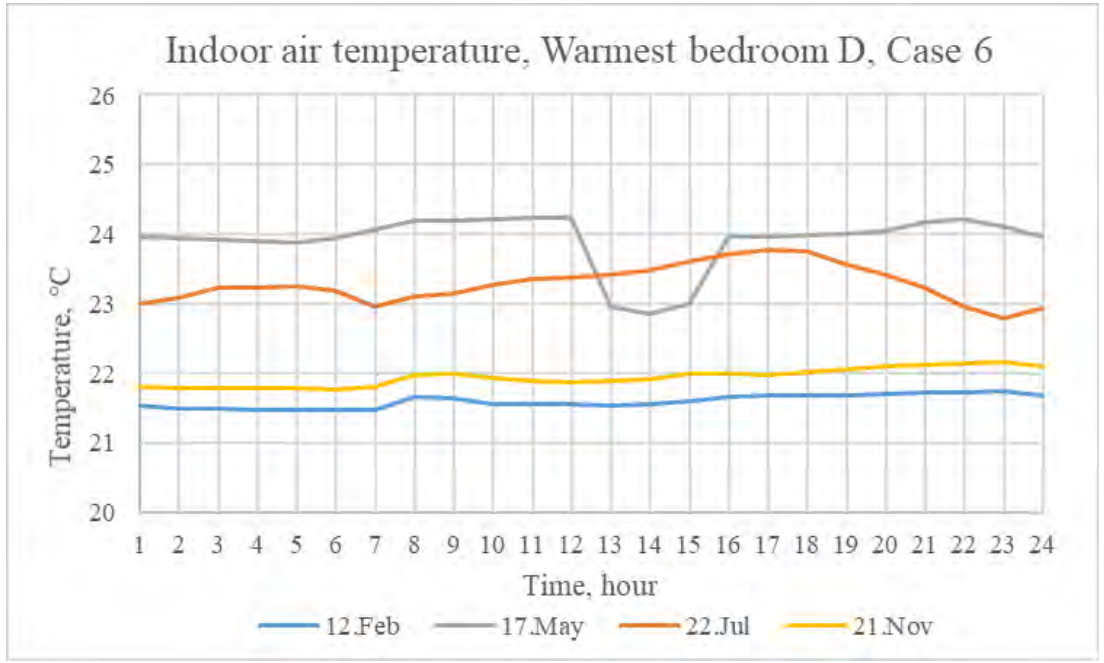


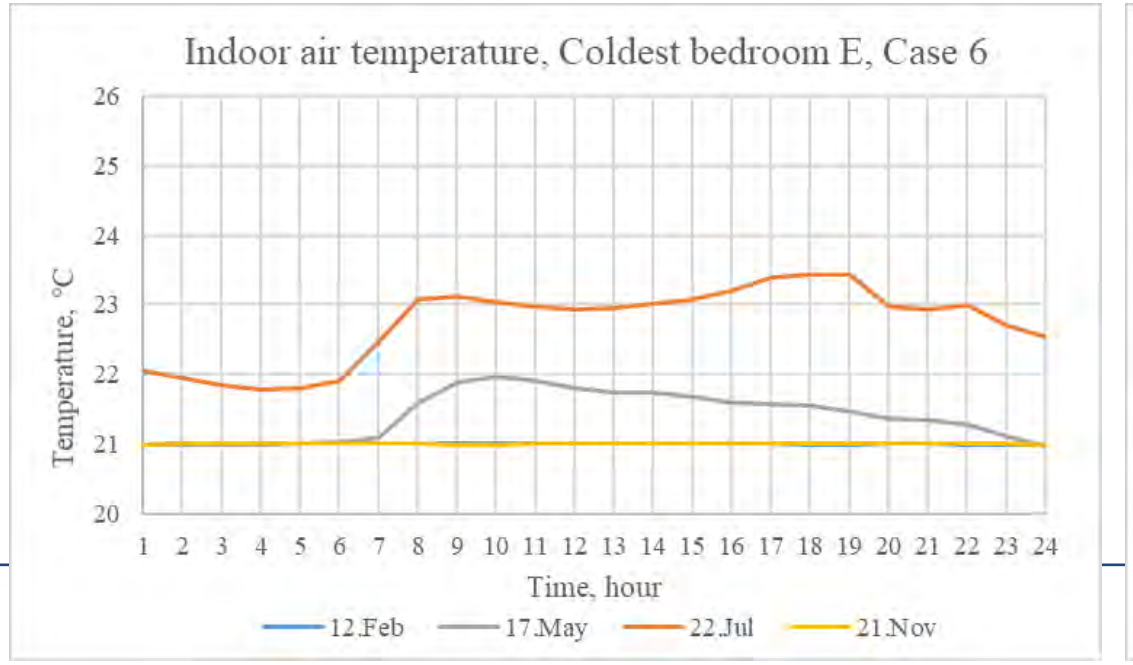

Case 5, Electrically controlled blinds

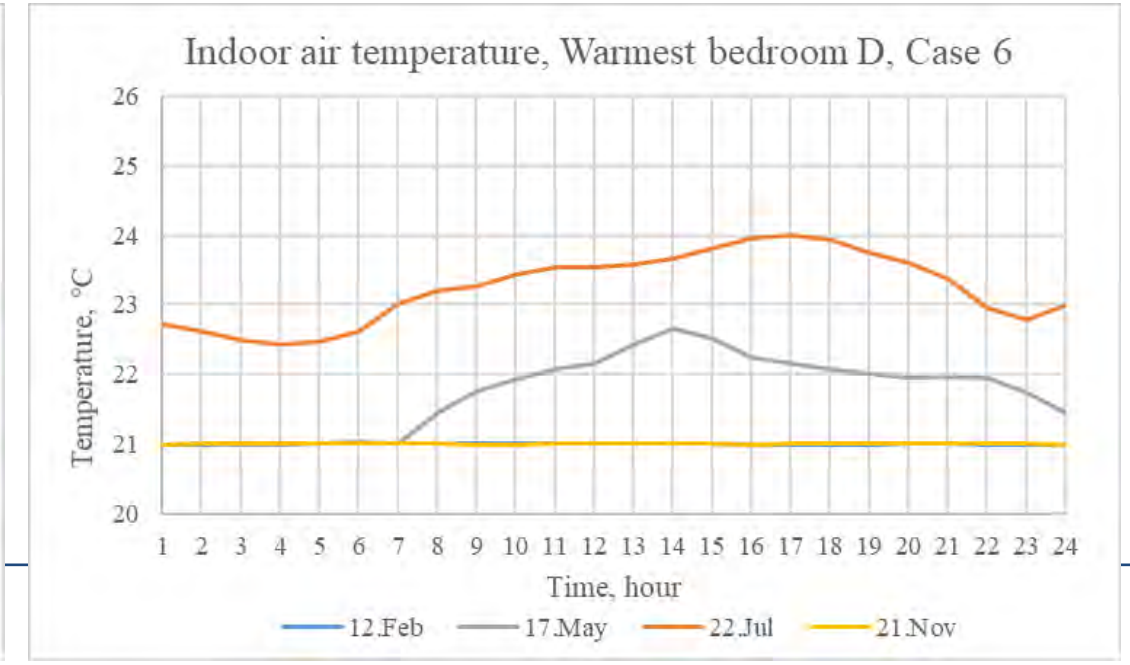
New apartment building




Old apartment building



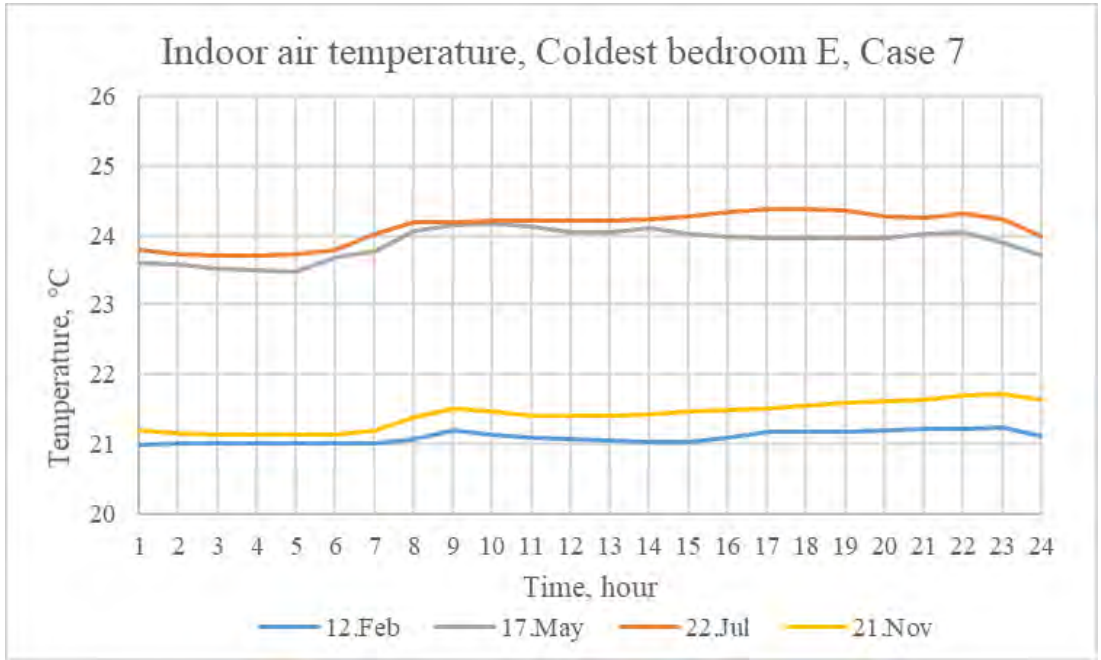



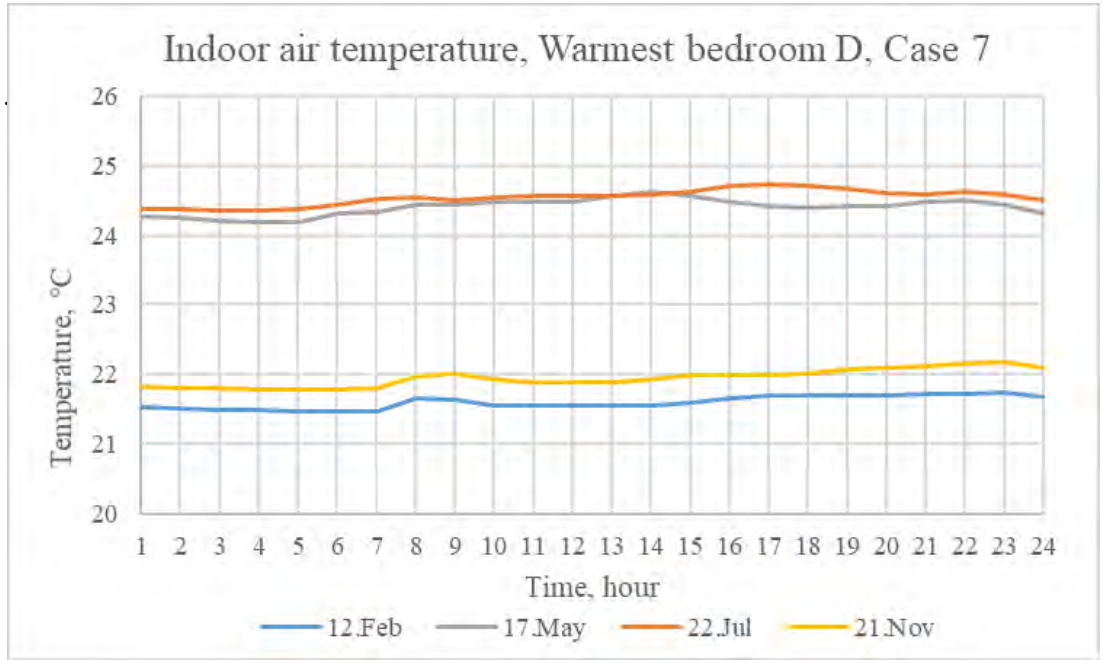

Case 6, Electrically controlled openings

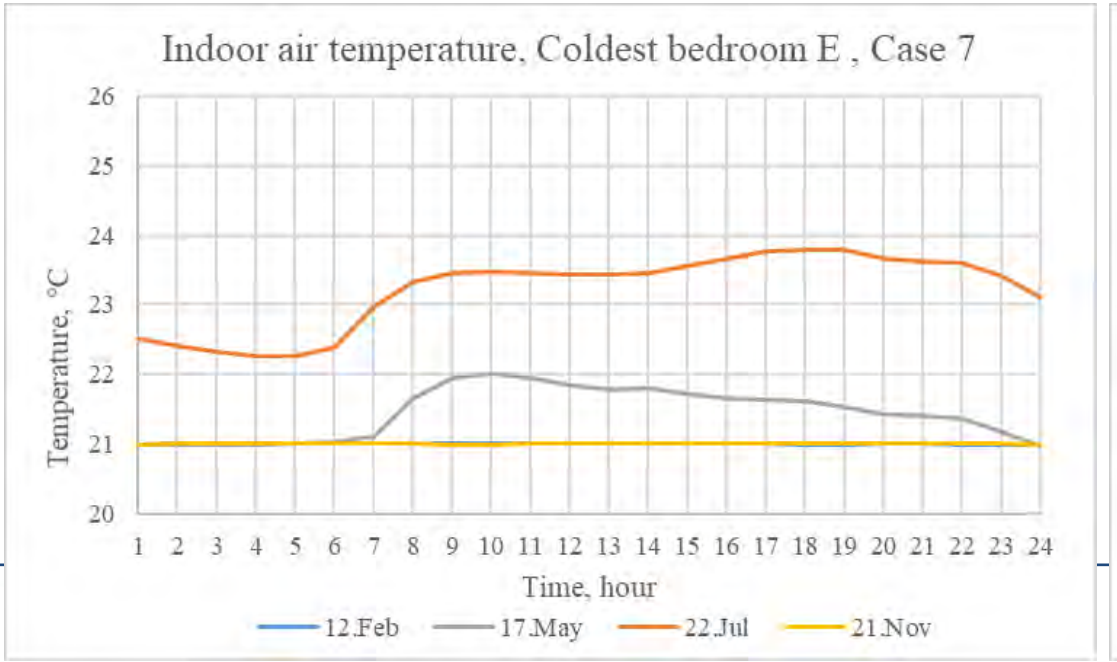
New apartment building

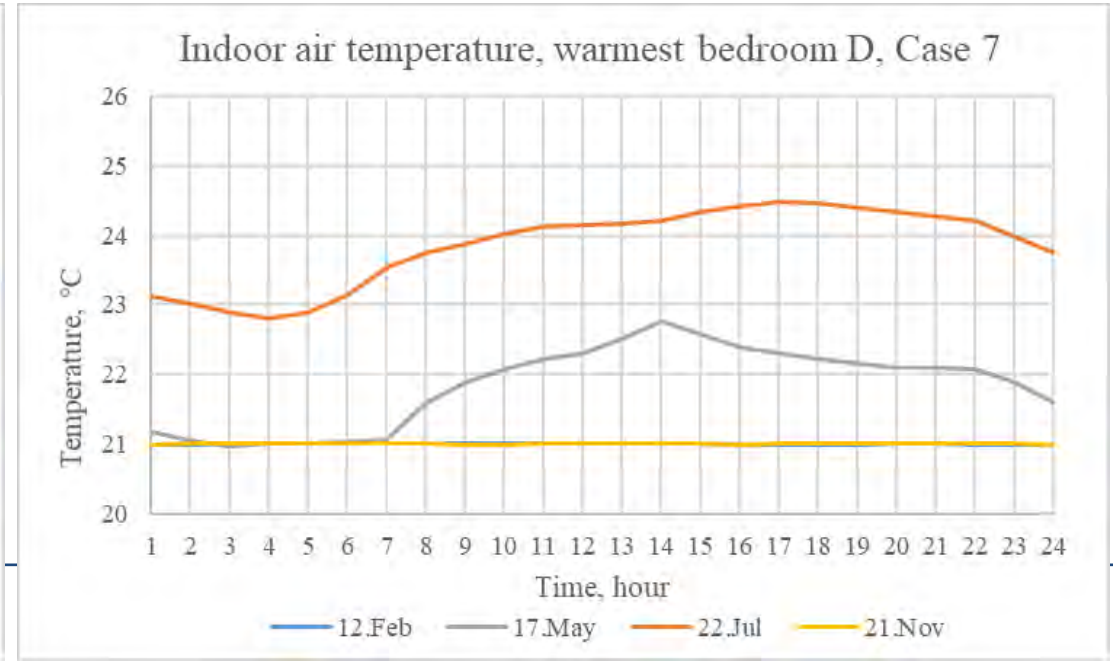
Old apartment building

Case 7, Electrically controlled Awning



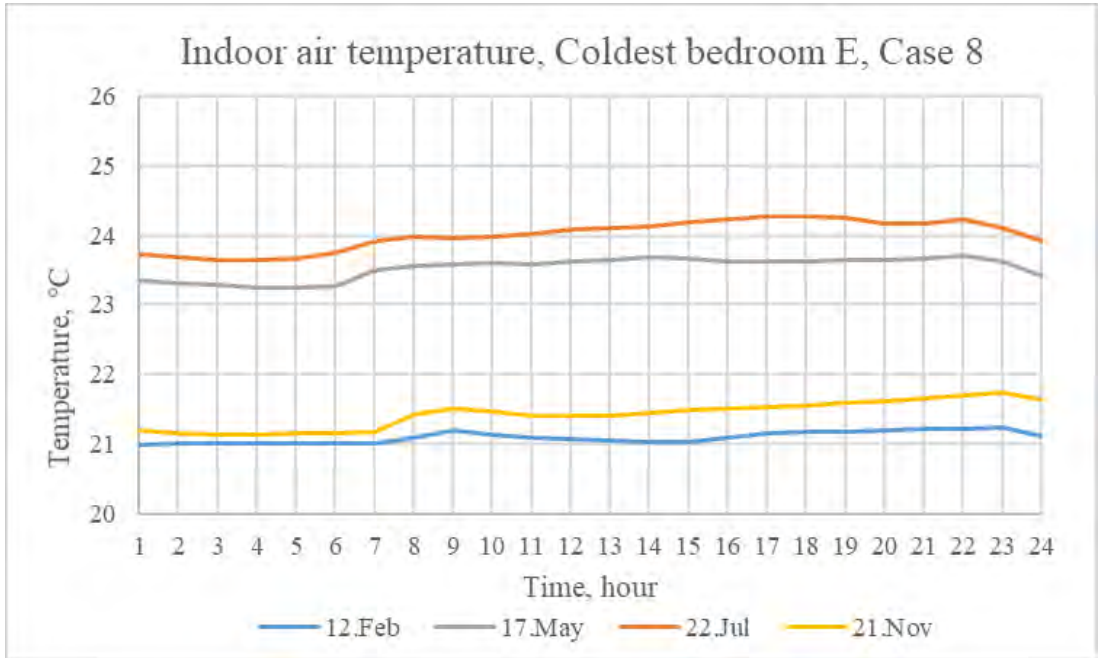



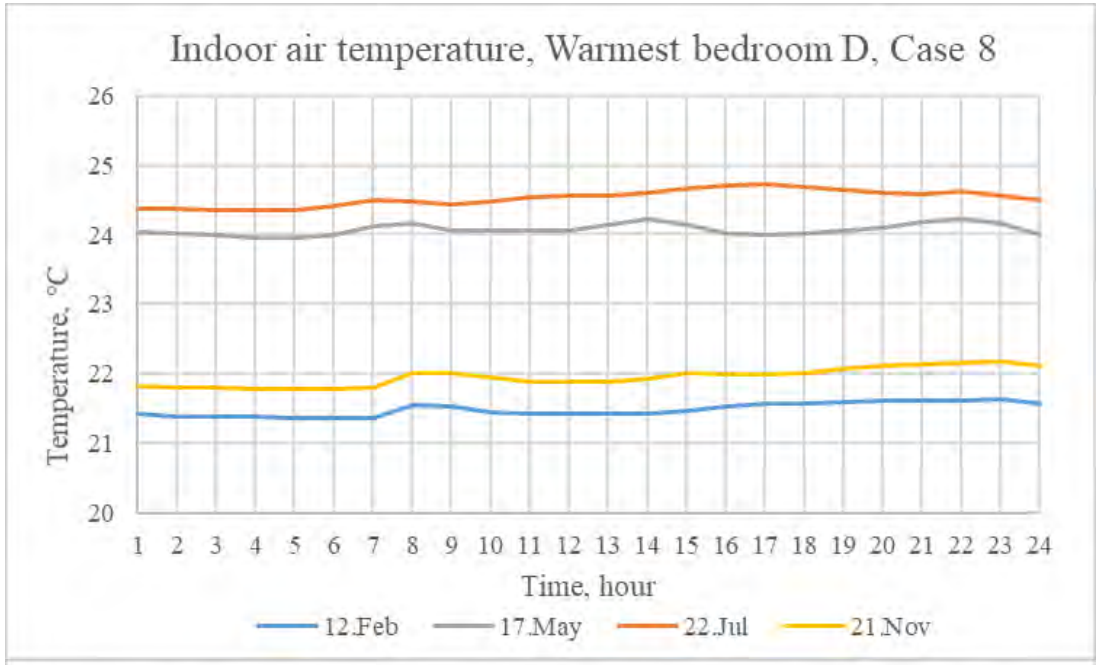



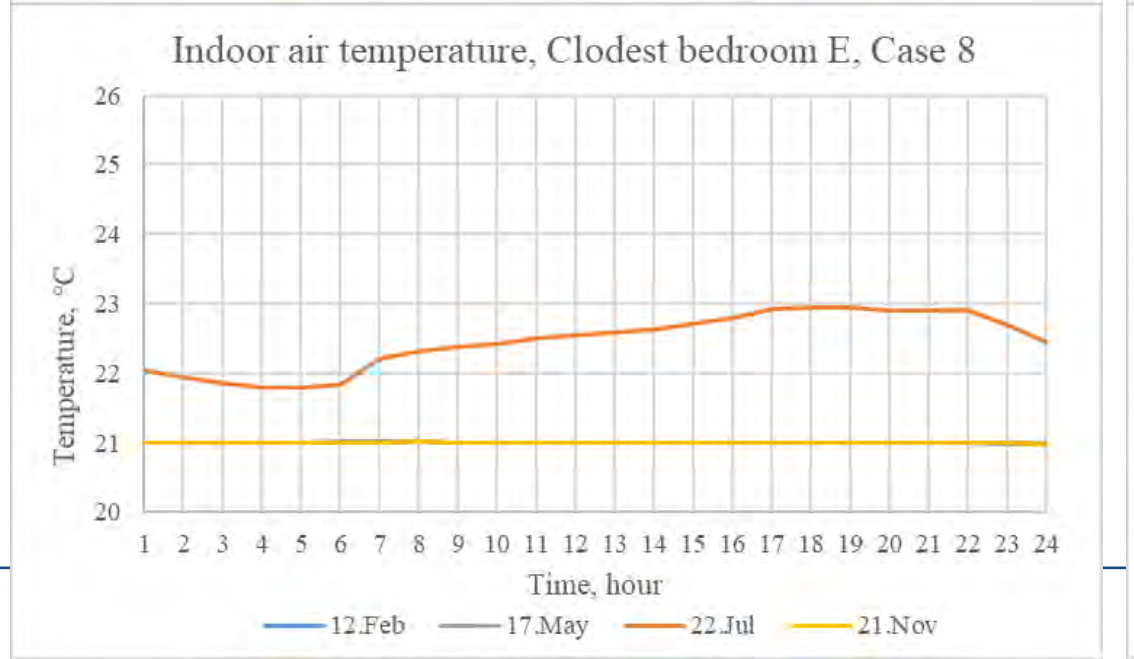



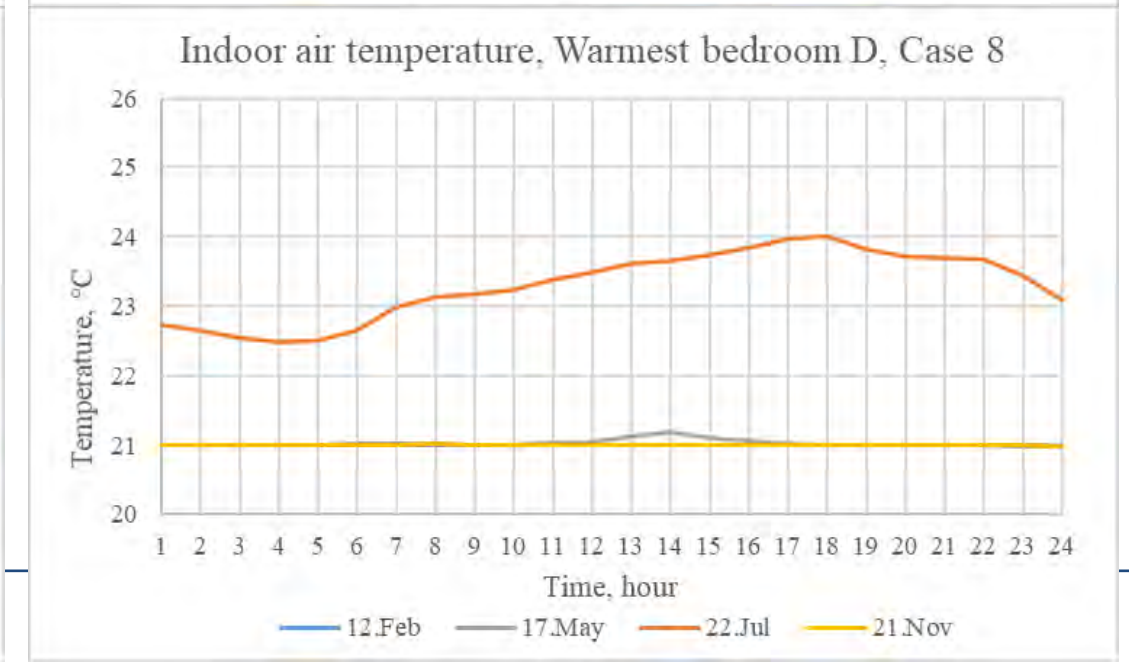
New apartment building

Old apartment building



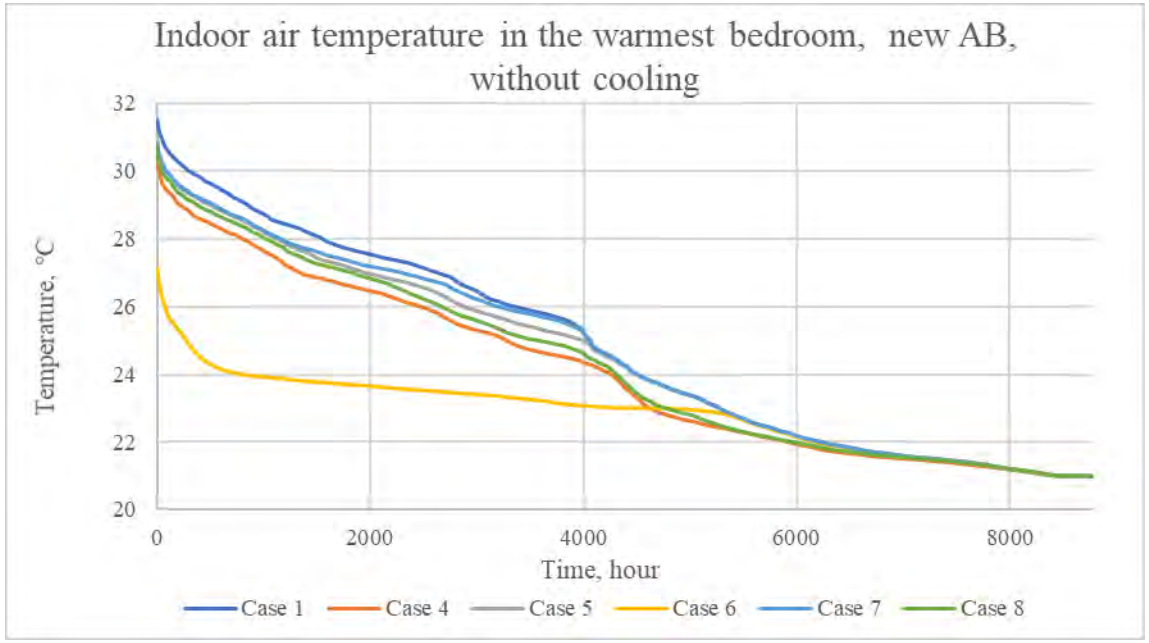

Case 8, Electrochromic windows

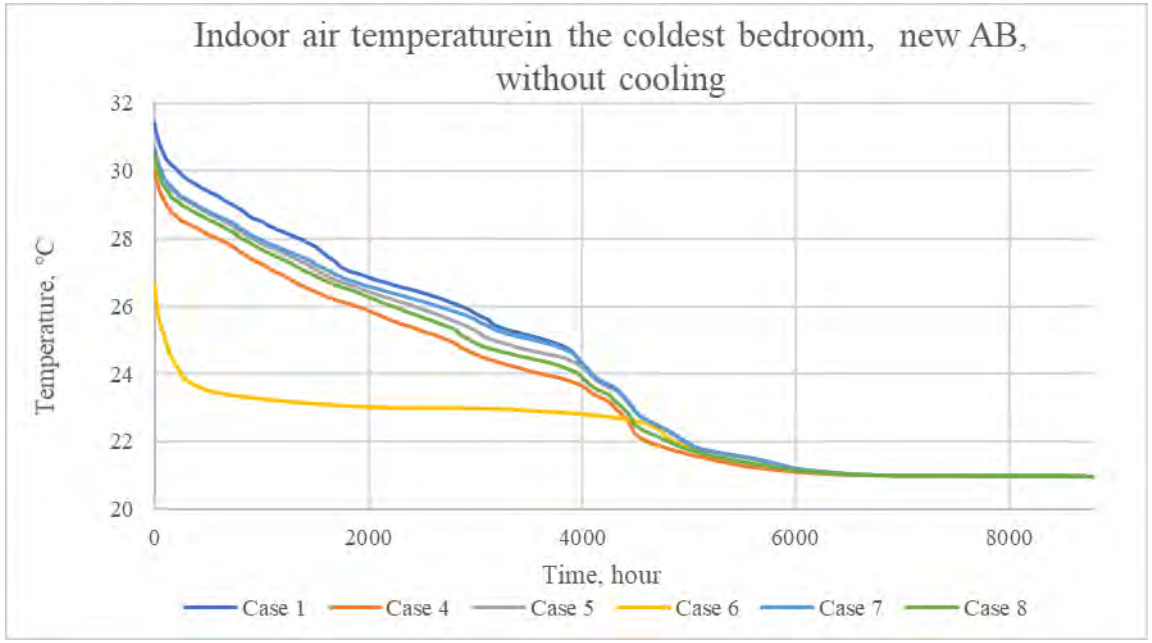
# Results of the cases without the active cooling system



Contact: juha.jokisalo@aalto.fi

# Indoor air temperatures, New apartment without the active cooling system:




| Degree hours, Warmest bedroom | | | |
|---|---|---|---|
| Cases | | above 24 °C | above 27 °C |
| Case 1 | base case | 14994 | 3938 |
| Case 4 | SP + Man.blinds | 10198 | 1636 |
| Case 5 | El. Blinds | 12705 | 2568 |
| Case 6 | El. Opening | 625 | 0 |
| Case 7 | El. Awnings | 13553 | 2770 |
| Case 8 | Electrochromic windows | 11604 | 2232 |

| Degree hours, Coldest bedroom | | | |
|---|---|---|---|
| Cases | | above 24 °C | above 27 °C |
| Case 1 | base case | 12674 | 3159 |
| Case 4 | SP + Man.blinds | 8100 | 1205 |
| Case 5 | El. Blinds | 10557 | 2061 |
| Case 6 | El. Opening | 224 | 0 |
| Case 7 | El. Awnings | 11300 | 2208 |
| Case 8 | Electrochromic windows | 9680 | 1775 |

# Conclusions

- Considering the effects of each solution on **district heat consumption of spaces and ventilation**:
  - ✓ **In the old building**: The lowest district heat consumption is for cases with **electrochromic windows and solar protection** windows with about 3% to 4% decrease (about 5 kWh/m$^2$) because of the window low U value in comparison to the base case's original poorly insulated windows. Other solutions have no noticeable effect on district heat consumption with less than 0.5%.
  - ✓ **In the new building**: Cases with **electrochromic windows and solar protection** windows have an increase of about 3% (about 1.5 kWh/m$^2$) in district heat consumption because of the reduction in solar gains in winter. Other solutions have no noticeable effect on district heat consumption with less than 0.5%.
- Considering the effects of each solution on **electricity consumption of space cooling**:
  - ✓ **In the old building**: The lowest electricity consumption of space cooling is for cases with **electrochromic windows and solar protection** windows, and electrically controlled openable windows with 62% decrease. However, the absolute value of this decrease is small (1.4 kWh/m$^2$). The electrically controlled blinds, electrically controlled awnings and manual blinds are the next effective ones with about 48%, 28.6%, and 14.3% decrease, respectively.
  - ✓ **In the new building**: The electrically controlled openable window is the case with the lowest electricity consumption of space cooling with 83% decrease. However, the absolute value of this decrease is small (1.9 kWh/m$^2$). Solar protection and electrochromic windows are the next effective ones with 30% and 20% decrease, respectively. Electrically controlled blinds, electrically controlled awnings, and manual blinds are the cases with the highest electricity consumption of space cooling.

# Conclusions

- Considering the indoor air temperature conditions in the studied bedrooms:

**In the old building**:

Range of degree hours above 24°C in the warmest bedroom with:
- The passive solutions: 542-1472Kh
- The automated solutions: 581-1056Kh

The order of three best solutions with the lowest degree hours:
1. Man. blinds +SP windows (542Kh)
2. SP windows (577Kh)
3. EL. openable windows (581Kh)

The highest degree hours are got with man. blinds (1472Kh).

**In the new building with the active cooling system:**

Range of degree hours above 24°C in the warmest bedroom with:
- The passive solutions: 1550-2196Kh
- The automated solutions: 278-2034Kh

The order of three best solutions with the lowest degree hours:
1. EL. openable windows (278Kh)
2. Man. blinds +SP windows (1550Kh)
3. SP windows (1617Kh)

The highest degree hours are got with man. blinds (2196Kh)

# Conclusions

**In the new building without the active cooling system:**

Range of degree hours above 24°C in the warmest bedroom with:

- Man. blinds +SP windows : 10198Kh
- The automated solutions: 625-13553Kh

The order of three best solutions with the lowest degree hours:

1. EL. openable windows (625Kh)
2. Man. blinds +SP windows (10198Kh)
3. Electrochromic windows(11604Kh)

Degree hours above 27°C in the warmest bedroom with EL. openable windows is 0.

**Comparing the results of the new apartment building with and without the active cooling system:**

- The degree hours above 24°C in the warmest bedroom are significantly higher in the cases without the active cooling system.
- The amounts of decrease of the degree hours above 24°C in comparison to Base case, in the cases without the active cooling, are more than in the cases with the active cooling by 3-8 %.

# Conclusions

- Comparing old and new apartment building:
  - ✓ Considering the indoor air temperature of warmest bedrooms' duration curves, In the old building, the air temperature is 21℃ most of the year and then it goes up to 26-27℃.While in the new building, it is 21℃ in just about 500 hours of the year and then it goes up to 25℃.
  - ✓ The degree hours above 24℃ in the bedrooms of the new apartment is higher than the old building's in all the cases except the one with the electrically controlled openings.
  - ✓ The middle floor is significantly warmer than the top floor in both old and new buildings.

# Liite 4 Simulointitulokset – toimistorakennus

Aalto University
School of Engineering

# Intelligent building envelope solutions in Finnish climatic conditions - Simulation of new and old Office building

07.05.2021

Azin Velashjerdi Farahani, Juha Jokisalo, Risto Kosonen

# Overview of the study

**Old and new buildings**

- **Office**

**Envelope solutions**

- Window properties

**Main Goals**

✓To see the effects of each solution on cooling, heating, and lighting energy consumption.

✓To see the effects of each solution on the indoor air temperature

# Methods

## Simulation tool

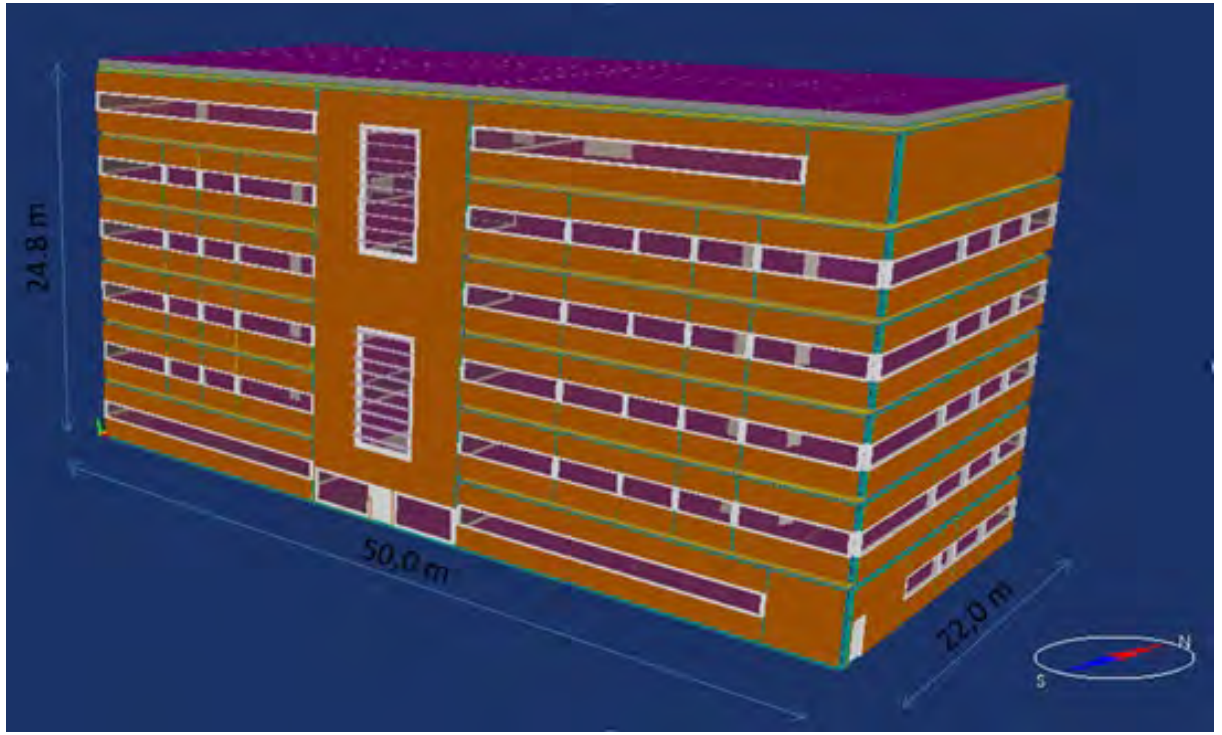
- **IDA ICE 4.8**
- **The simulation period**: one year
- **Time resolution of the results:** 1 hour.
- **Whole building simulation**

## Climatic data

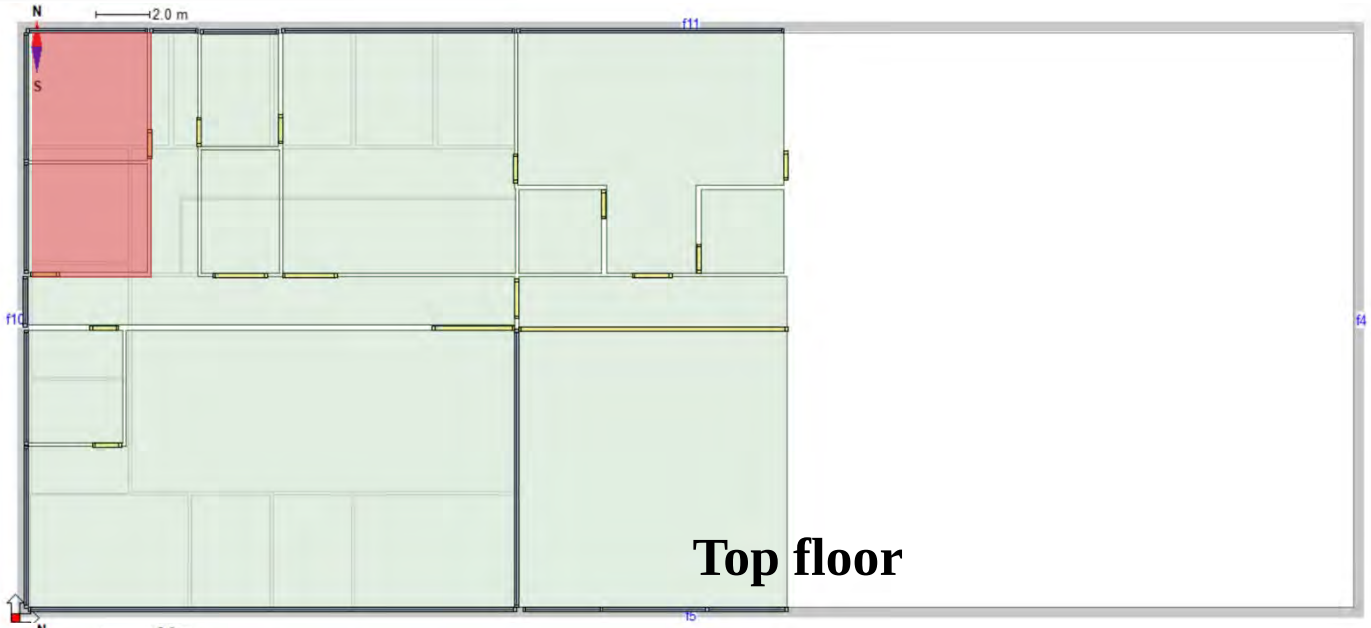
**Weather data:** TRY (2020), Helsinki-Vantaa

## Example building

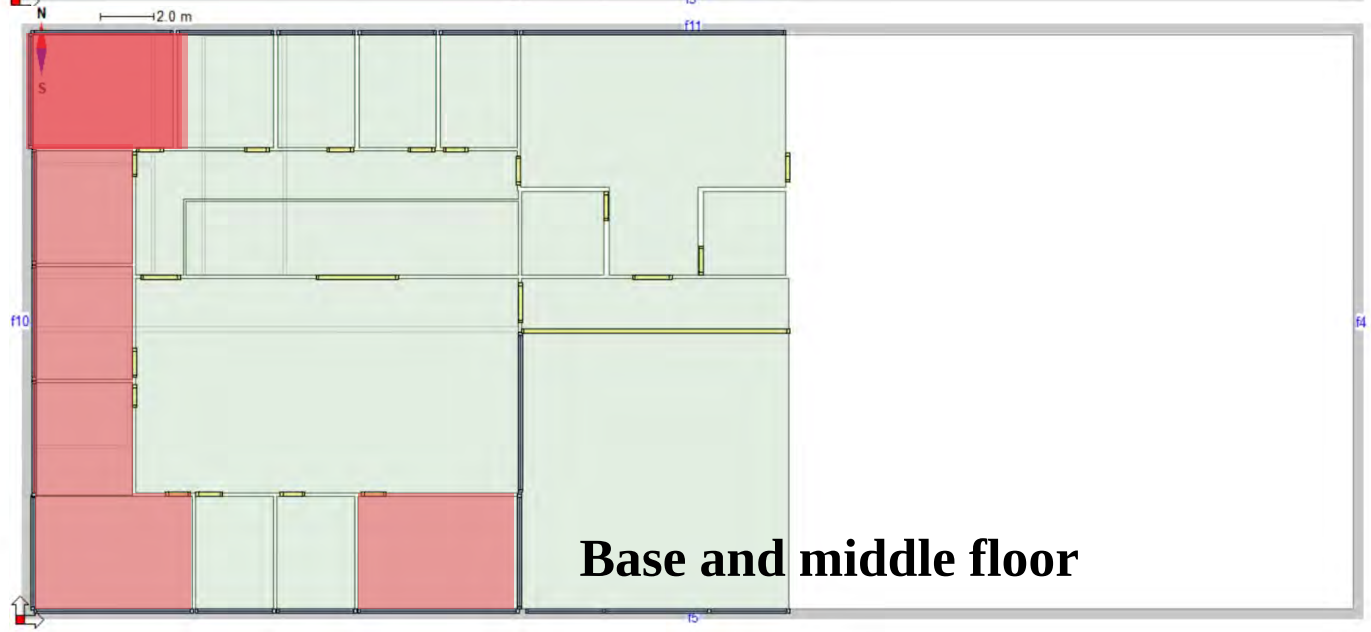
Finnish example building which has been used to show cost optimality of building regulations for European commission.



# Plan of the buildings




**Top floor**

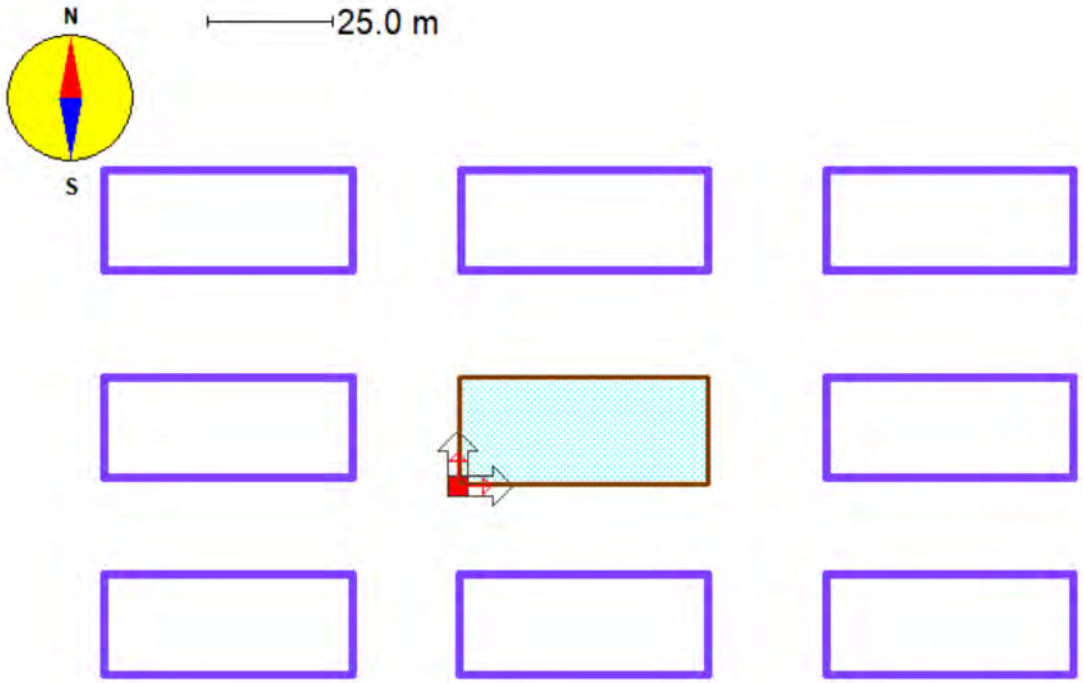



**Base and middle floor**

- ✓6 floors
- ✓Heated net floor area: 5744 $m^2$
- ✓Occupied hours: 7-18 weekdays
- ✓Window to wall ratio : 50%
- ✓Total Number of occupants: 307
- ✓Average occupancy density: 1 per 20 $m^2$

**Meeting room**

# Orientation and surrounding of the buildings




- ✓ 8 similar buildings on each side of the building
- ✓Distance between the buildings: 20 m
- ✓The façade with largest windows faces South.

# Example buildings, Building properties

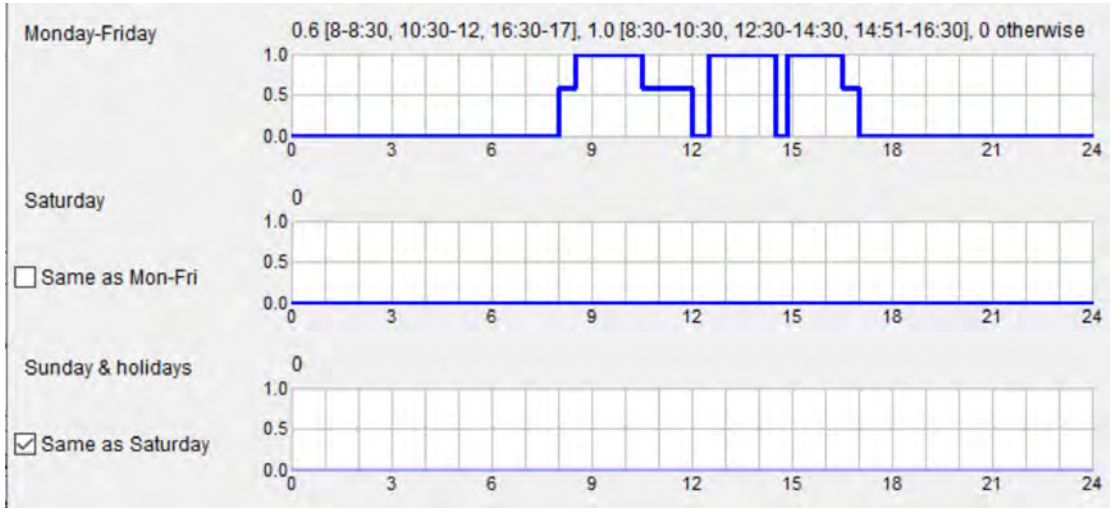
| | Building Properties | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | envelope properties | | Window properties | | | | | | |
| | Orientation | Shading building | U value ($W/m^2K$) | Air Leakage $q_{50}$ ($m^3/h,m^2$) | U value ($W/m^2K$) | ST | Tvis | g-value | External shading | Integrated shading | Window opening |
| **Old building** | South | Total of 8 same buildings on north, east, south and west side of the example building. | External wall 0.29<br>Roof 0.23<br>Base floor 0.37 | 6 | 2.1 | 0.58 | 0.74 | 0.68 | window indentation 5 cm | None | None |
| **New building** | | | External wall 0.17<br>Roof 0.09<br>Base floor 0.16 | 2 | 1 | 0.3 | 0.61 | 0.35 | | | |

# Example buildings, System properties

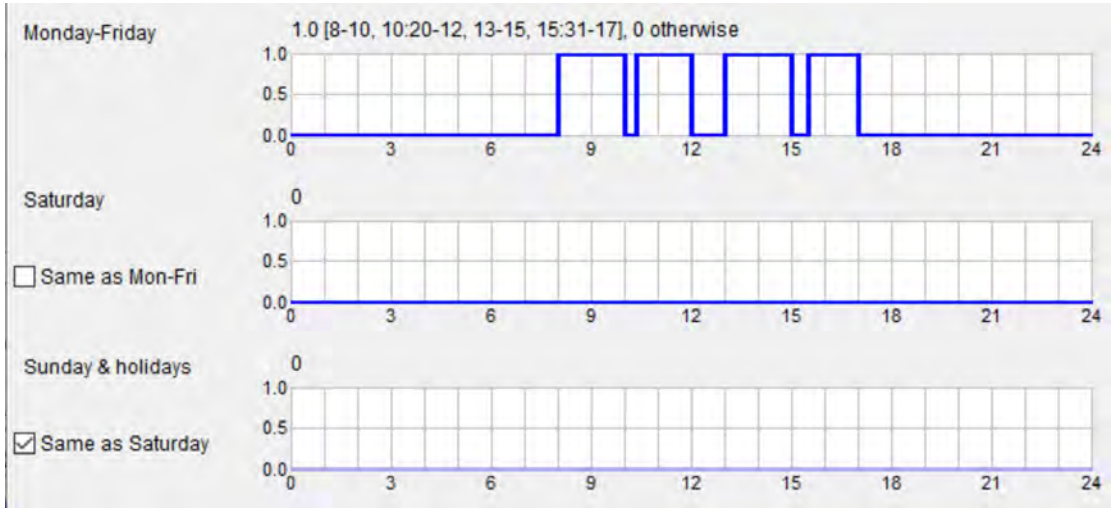
| | System properties | | | | | | |
|---|---|---|---|---|---|---|---|
| | Setpoint | | Mechanical cooling system | District heating system | Main ventilation 06-19 in weekdays (one hour before occupied hours) | Basic ventilation (unoccupied hours) | Daylight control |
| | Heating setpoint ℃ | Cooling setpoint ℃ | | | | | |
| **Old building** | 21 | 25 | Ideal space cooling,<br>Cooling in main AHU, supply air temperature 16 ℃<br>COP:2.5<br>(available between 06-19 in weekdays) | Space heating, 70/40 ℃ water radiators (efficiency 0.8),<br>Substation efficiency 0.97 | CAV<br>Office rooms 1.5 l/sm$^2$<br>Meeting rooms 4 l/sm$^2$<br>Other rooms 1.5 l/sm$^2$<br>Night flush ventilaiton between 22-06 in weekdays (Sunday night to Friday Morning) Apr_Sep, available when outdoor temperature exceeds 12 ℃, exhaust air temperature exceeds 23 ℃, and the oudoor temperature is at least 2 ℃ bellow exhaust air temperature.<br>Heat recovery of 0.6 with defrost protection 4 ℃ (Minimum exhaust air temperature after heat recovery)<br>SFP:2.5 | CAV, 0.15 l/sm$^2$ All rooms | None |
| **New building** | | | Ideal space cooling,<br>Cooling in main AHU, supply air temperature 16 ℃<br>COP:3<br>(available between 06-19 in weekdays) | Space heating, 45/35 ℃ water radiators (efficiency 0.9),<br>Substation efficiency 0.97 | VAV<br>Office rooms 0.22 l/sm$^2$ (if CO2 < 600ppm)-2 l/sm$^2$ (if CO2 > 900ppm)<br>Meeting rooms 0.22 l/sm$^2$ (if CO2 < 600ppm)- 4 l/sm$^2$ (if CO2 > 900ppm)<br>CAV<br>Other rooms 2 l/sm$^2$<br>Night flush ventilaiton between 22-06 in weekdays (Sunday night to Friday Morning) Apr_Sep, available when outdoor temperature exceeds 12 ℃, exhaust air temperature exceeds 23 ℃, and the oudoor temperature is at least 2 ℃ bellow exhaust air temperature.<br>Heat recovery of 0.8 with defrost protection 1 ℃ (Minimum exhaust air temperature after heat recovery)<br>SFP:1.8 | | Lighting OFF (if daylight level at workspace > 700lx)<br>Lighting fully ON (if daylight level at workspace < 500lx) |

# Occupancy, lighting, and equipment profiles

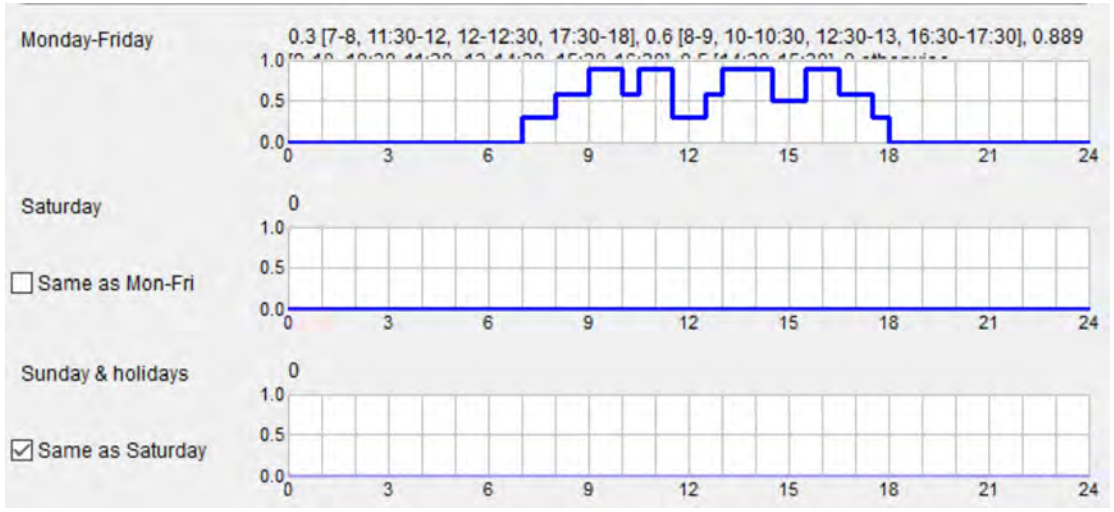
**Meeting room**




**Office room**




**Open office**




**Internal heat gains level:**

- Lighting: 9.9 W/m$^2$
- Equipment: 12 W/m$^2$

# Simulation cases

| Cases | | Description | Weather file |
|---|---|---|---|
| Base Case | Old-Base | The reference case. | TRY 2020 |
| | New-Base | | |
| Orientation | Old-West | Orientation is changed 90◦. | |
| | New-West | | |
| Site shading | Old-No site shading | There is no building shading around the example building. | |
| | New-No site shading | | |
| Passive solutions | Old-Manual blind | Manual blinds between the outer windowpanes according to the occupancy schedule during whole year, all the windows except the atrium windows. | |
| | New-Manual blind | | |
| | Old-S.P windows | Solar protection windows, U value: 1 $W/m^2K$, g-value: 0.19, ST: 0.16 | |
| | New-S.P windows | | |
| | Old-Manual blind & S.P. windows | Combination of manual blind and solar protection windows. | |
| | New-Manual blind & S.P. windows | | |
| Automated solutions | Old-Auto blind | Blinds between the outer window panes are in used when the intensity of inside solar radiation exceeds 100 $W/m^2$ for all the windows except the atrium windows. | |
| | New-Auto blind | | |
| | Old-Auto External blind | External blinds are in used when the intensity of inside solar radiation exceeds 100 $W/m^2$ for all the windows except the atrium windows. | |
| | New-Auto External blind | | |
| | Old-Openable windows | 10% of the largest window of each external wall of each room (cross ventilation) opens when the outside air temperature is between 12 °C and 22°C and the zone air temperature between 23 °C and 25°C. Not in the atrium. | |
| | New-Openable windows | | |
| | Old-Auto awning | Electrically openable awning, Awnings on the windows without a balcony open when the wind speed is less than 8 m/s and outdoor air temperature exceeds 15 °C meanwhile the outside solar radiation on the vertical surface exceeds 100 $W/m^2$. All the windows except the large atrium windows, small windows of the atrium at the street level have the awnings. Depth and height of the awning is equal to the 30% of the window height. | |
| | New-Auto awning | | |
| | Old-Electrochromic windows | Electrochromic windows, U value: 0.97 $W/m^2K$, when the outdoor solar radiation on the vertical surface exceeds 450 $W/m^2$ the darkest state of the glazing is on ST: 0.29-0.01, g-value: 0.31-0.05, Tvis: 0.61. All the windows. | |
| | New-Electrochromic windows | | |

# Break down of Energy Consumption, Old office building

**Total annual Energy consumption, Old office building (kWh/m$^2$, a)**

| System | | Old-Base | Old-West | Old-Nositeshading | Old-Manual blind | Old-S.P. windows | Old -Manual Blind&S.P. windows | Old-AutoBlind | Old-Auto external blind | Old-Openable windows | Old -Auto awning | Old-Electrochromic windows |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| District heating | Space heating | 50.8 | 54.5 | 46.7 | 53.6 | 42.2 | 43.5 | 54.1 | 55.7 | 51.0 | 51.0 | 40.7 |
| | AHU heating | 47.0 | 47.3 | 46.6 | 47.3 | 47.5 | 47.5 | 47.4 | 47.5 | 47.0 | 47.0 | 47.5 |
| | DHW | 7.0 | 7.0 | 7.0 | 7.0 | 7.0 | 7.0 | 7.0 | 7.0 | 7.0 | 7.0 | 7.0 |
| | Total | 104.8 | 108.8 | 100.4 | 108.0 | 96.7 | 98.1 | 108.5 | 110.2 | 105.0 | 105.0 | 95.1 |
| Difference total (%) | | | 3.8 % | -4.3 % | 3.0 % | -7.8 % | -6.5 % | 3.5 % | 5.1 % | 0.2 % | 0.2 % | -9.2 % |
| Difference (space+AHU) (%) | | | 4.0 % | -4.6 % | 3.2 % | -8.3 % | -6.9 % | 3.7 % | 5.5 % | 0.2 % | 0.2 % | -9.9 % |
| Electricity | Space cooling | 2.5 | 2.4 | 3.6 | 1.5 | 0.1 | 0.1 | 0.7 | 0.5 | 1.1 | 0.8 | 0.1 |
| | AHU cooling | 2.1 | 2.1 | 2.1 | 2.1 | 2.0 | 2.0 | 2.1 | 2.1 | 2.1 | 2.1 | 2.0 |
| | HVAC aux | 23.9 | 23.8 | 24.1 | 23.6 | 22.8 | 22.6 | 23.4 | 23.2 | 23.2 | 23.3 | 22.9 |
| | Lighting | 18.6 | 18.6 | 18.6 | 18.6 | 18.6 | 18.6 | 18.6 | 18.6 | 18.6 | 18.6 | 18.6 |
| | Equipment | 22.6 | 22.6 | 22.6 | 22.6 | 22.6 | 22.6 | 22.6 | 22.6 | 22.6 | 22.6 | 22.6 |
| | Total | 69.7 | 69.5 | 71.0 | 68.4 | 66.1 | 65.9 | 67.4 | 67.0 | 67.5 | 67.3 | 66.2 |
| Difference total (%) | | | -0.3 % | 1.9 % | -1.9 % | -5.1 % | -5.4 % | -3.3 % | -3.9 % | -3.1 % | -3.4 % | -5.0 % |
| Difference (space+AHU) (%) | | | -1.9 % | 24.7 % | -22.0 % | -53.6 % | -54.4 % | -38.8 % | -43.6 % | -31.4 % | -38.1 % | -53.9 % |

- **Solar protection windows with manual blinds** (tot. elec. -5.4%, tot. heating -6.5%)
- **Solar protection windows** (tot. elec. -5.1%, tot. heating -7.8%)
- **Electrochromic windows**(tot. elec. -5%, tot. heating -9.2%)

Lowest district heating (Higher window U-value in comparison with the existing poorly insulated windows in old building)
First 3 lowest electricity consumption (low solar gains cause cooling of ventilation to be almost enough, and space cooling electricity to be nearly zero)

# Break down of Energy Consumption, New office building

**Total annual Energy consumption, New office building (kWh/m$^2$, a)**

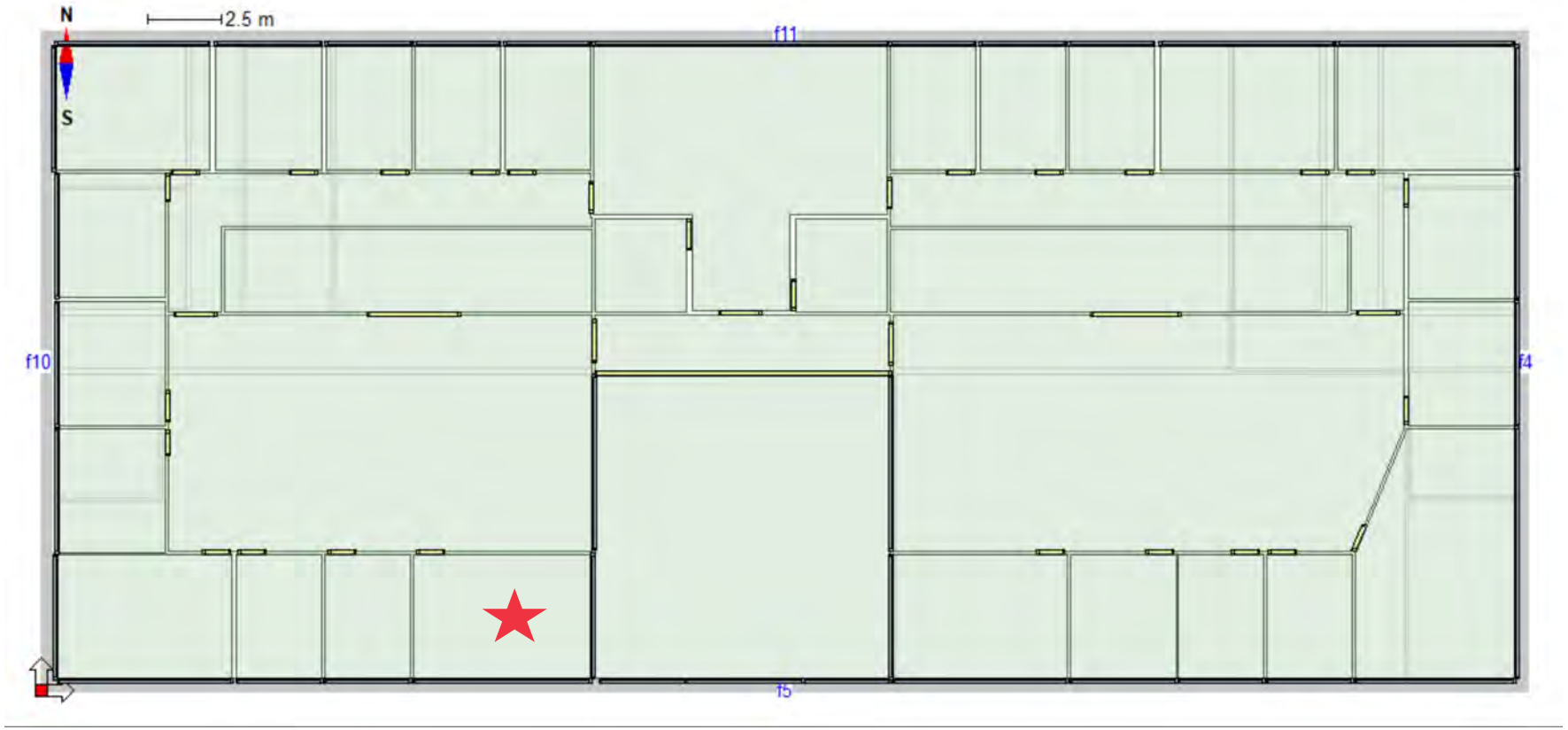
| System | | New-Base | New-West | New-Nositeshading | New-Manual blind | New-S.P. windows | New -Manual Blind&S.P. windows | New-AutoBlind | New-Auto external blind | New-Openable windows | New -Auto awning | New-Electrochromic windows |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| District heating | Space heating | 14.6 | 15.6 | 13.3 | 15.3 | 16.4 | 16.8 | 15.1 | 15.4 | 14.6 | 14.6 | 15.3 |
| | AHU heating | 7.5 | 7.6 | 7.4 | 7.6 | 7.7 | 7.7 | 7.6 | 7.6 | 7.6 | 7.5 | 7.6 |
| | DHW | 7.0 | 7.0 | 7.0 | 7.0 | 7.0 | 7.0 | 7.0 | 7.0 | 7.0 | 7.0 | 7.0 |
| | Total | 29.2 | 30.2 | 27.8 | 30.0 | 31.0 | 31.5 | 29.7 | 30.0 | 29.2 | 29.2 | 30.0 |
| Difference total (%) | | | 3.7 % | -4.8 % | 2.7 % | 6.4 % | 8.0 % | 2.0 % | 3.0 % | 0.1 % | 0.0 % | 2.9 % |
| Difference (space+AHU) (%) | | | 4.8 % | -6.3 % | 3.6 % | 8.5 % | 10.6 % | 2.7 % | 3.9 % | 0.2 % | 0.0 % | 3.8 % |
| Electricity | Space cooling | 3.3 | 2.9 | 4.0 | 2.6 | 1.6 | 1.4 | 2.3 | 2.0 | 1.0 | 2.1 | 1.8 |
| | AHU cooling | 0.8 | 0.8 | 0.8 | 0.8 | 0.8 | 0.8 | 0.8 | 0.8 | 0.6 | 0.8 | 0.8 |
| | HVAC aux | 5.8 | 5.8 | 5.8 | 5.8 | 5.8 | 5.7 | 5.8 | 5.8 | 5.3 | 5.8 | 5.8 |
| | Lighting | 16.2 | 16.9 | 15.9 | 17.4 | 16.2 | 17.4 | 17.0 | 17.1 | 16.2 | 16.6 | 16.2 |
| | Equipment | 22.6 | 22.6 | 22.6 | 22.6 | 22.6 | 22.6 | 22.6 | 22.6 | 22.6 | 22.6 | 22.6 |
| | Total | 48.7 | 49.0 | 49.0 | 49.2 | 47.0 | 47.9 | 48.5 | 48.3 | 45.8 | 47.9 | 47.1 |
| Difference total (%) | | | 0.7 % | 0.7 % | 1.1 % | -3.5 % | -1.6 % | -0.4 % | -0.8 % | -6.0 % | -1.6 % | -3.2 % |
| Difference (space+AHU) (%) | | | -8.8 % | 16.0 % | -16.8 % | -41.6 % | -46.6 % | -24.3 % | -32.0 % | -59.0 % | -28.9 % | -37.7 % |

- **Openable windows** (tot. elec. -6%, tot. heating +0.1%)
- **Solar protection windows** (tot. elec. -3.5%, tot. heating +6.4%)
- **Electrochromic windows** (tot. elec. -3.2%, tot. heating +2.9%)

Openable window is the best solution with the lowest electricity consumption and no change in district heating consumption.

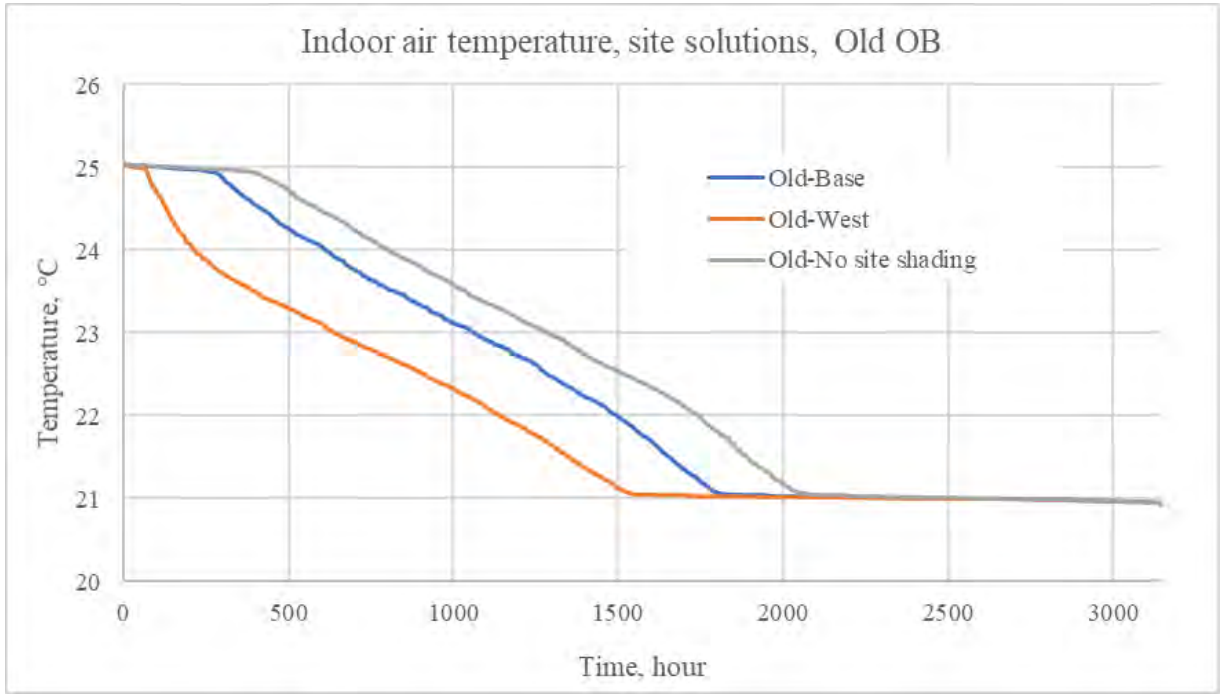
Electrochromic and solar protection windows are with the next two lowest electricity consumption but an increase in district heating consumption due to the lower solar heat gains.

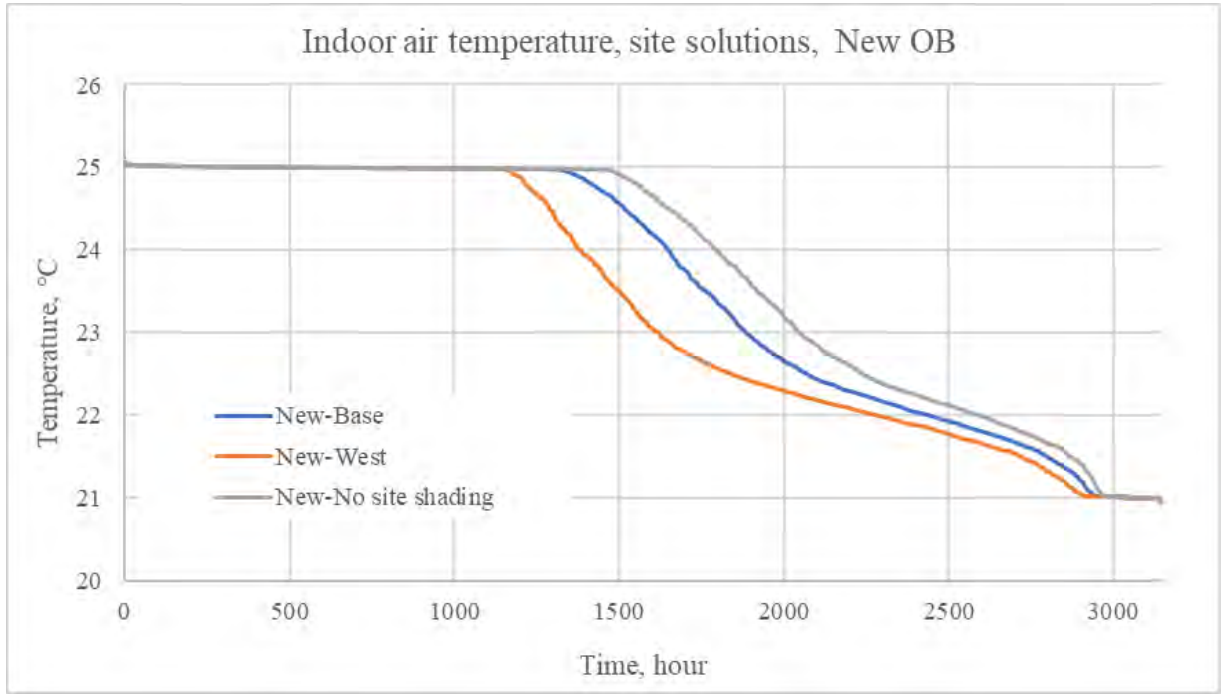
# Results, Indoor temperature conditions



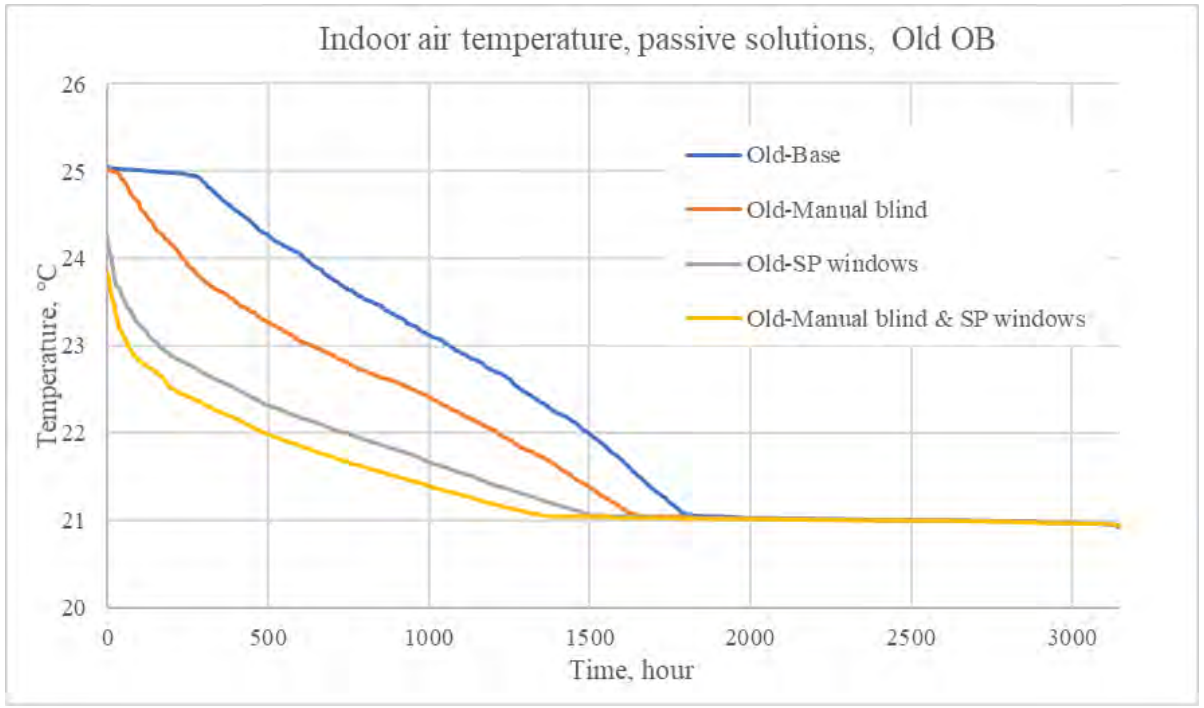

★**Warmest room with external windows, middle floor (meeting room)**

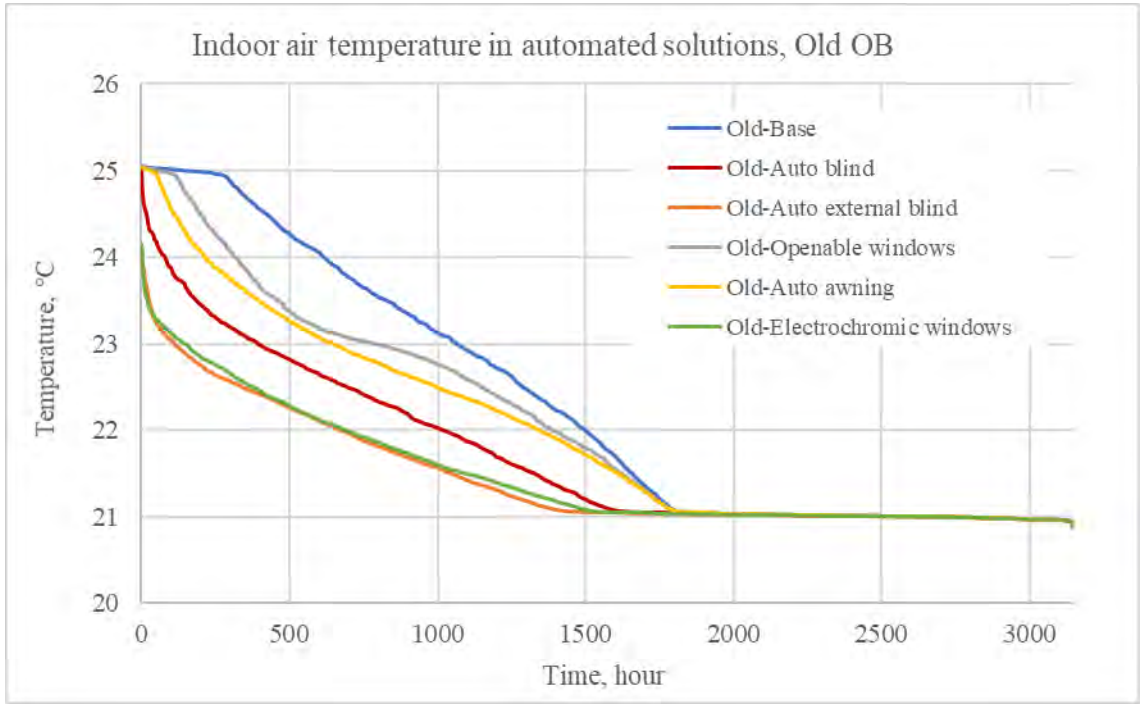
# Indoor air temperatures during occupied time, Site solutions



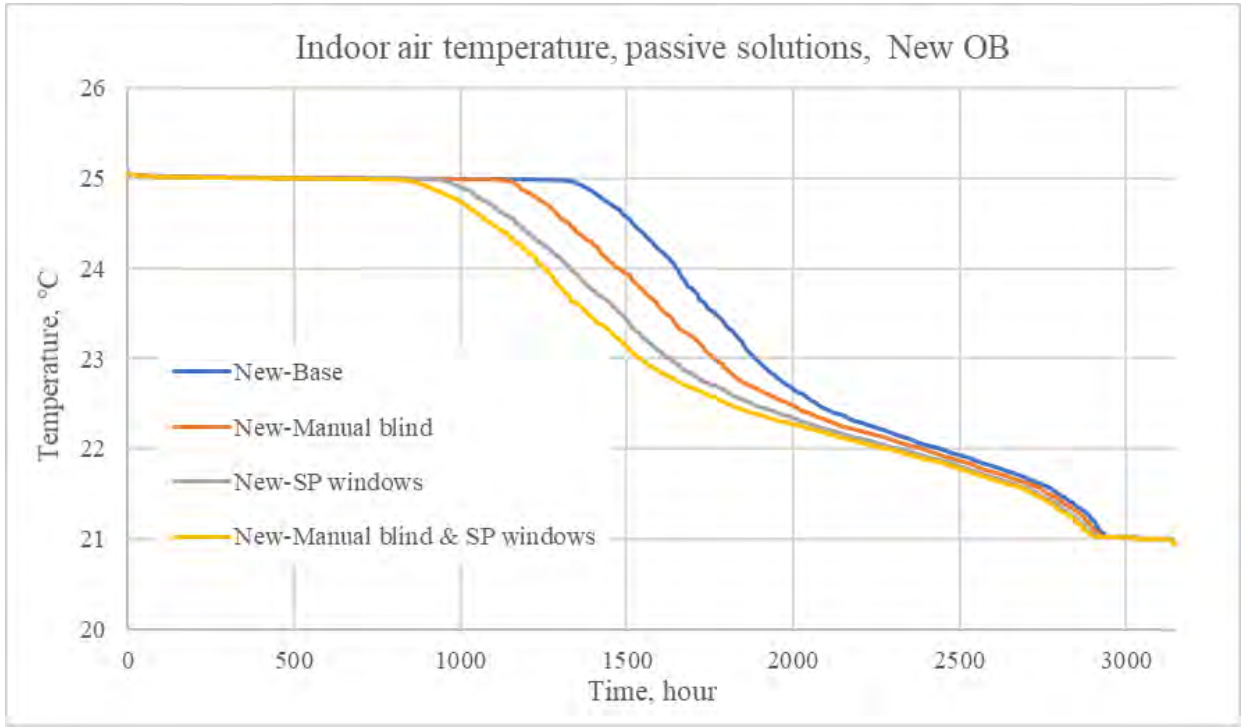
# Indoor air temperatures during occupied time, Passive and automated solutions, Old building

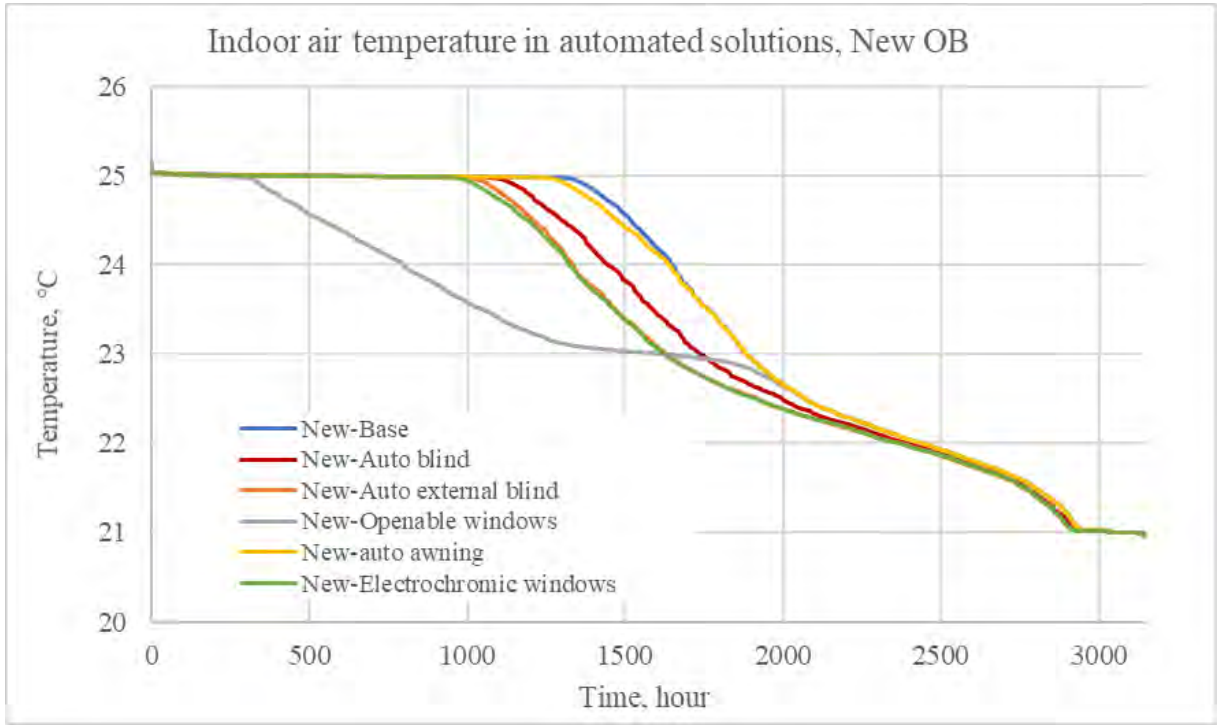



- Cases with lowest energy demand **(Solar protection windows with manual blinds, Solar protection windows, Electrochromic windows)** have the most comfortable indoor temperature conditions. Solar protection windows are slightly more effective.

# Indoor air temperatures during occupied time, Passive and automated solutions, New building




- Cases with lowest energy demand **(Openable windows, Electrochromic windows, Solar protection windows)** and **auto external blinds** have the most comfortable indoor temperature conditions. Solar protection windows are slightly more effective than the electrochromic windows.

# Conclusions

- Considering the effects of each solution on **district heat consumption**:
  - ✓ **In the old building**: The lowest space heating is for cases with **electrochromic windows and solar protection** windows because of high U value of the window in comparison to the base case's original poorly insulated windows. Other solutions increase space heating. The highest space heating is for cases with **auto external blind, auto blind, and manual blind** because of the reduction of solar gains in winter.
  - ✓ **In the new building**: All the solutions increase space heating consumption. Cases with **solar protection windows & manual blinds, auto external blinds, and electrochromic windows** have the highest district heat consumption because of the reduction of solar gains in winter.
- Considering the effects of each solution on electricity consumption :
  - ✓ **In the old building**: All the solutions decrease space cooling. The lowest space cooling is for cases with **manual blinds & solar protection window, solar protection, and electrochromic windows**, and **auto external blinds.**
  - ✓ **In the new building**: All the solutions decrease space cooling. The lowest electricity consumption is for cases with **Openable windows**, **manual blinds & solar protection, Solar protection windows, electrochromic windows**, and **auto external blinds.**
  - ✓ Lighting electricity consumption is increased by using manual or automated blinds, automated awning, automated external blinds and openable windows. Overall, the difference in total electricity consumption shows that **Openable windows, solar protection windows and electrochromic windows** are the first three solutions in terms of total electricity consumption in the new building.

# Conclusions

- Considering the indoor air temperature conditions in the warmest meeting room:

➢ **Overall, new building is warmer than old building whole year because of well-insulated construction.**

➢ **Site solutions:**

Removing the site shading significantly increase indoor air temperature while west orientation significantly reduce it.

➢ **Passive solutions:**

The most comfortable indoor temperature is for the case with combination of manual blinds and solar protection windows in both the old and new building.

➢ **Automated solutions:**

Electrochromic windows and external blinds are the most effective solutions for decreasing indoor air temperature. in both the old and new building.

In the new building, Openable windows significantly reduce the indoor air temperature.

➢ **Comparison:**

The case with lowest energy demand among passive strategies (**solar protection windows**) cause slightly more comfortable indoor conditions in comparison to the case with lowest energy demand among automated strategies (**electrochromic windows**) except for the openable windows in the new building.