Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Automatizálási és Alkalmazott Informatikai Tanszék

Mezei Alexandra

FELHŐ SZOLGÁLTATÁS FELHASZNÁLÁSA RENDSZER HATÁSFOK NÖVELÉS ÉS TERVEZÉS OPTIMALIZÁLÁS TERÜLETÉN

KONZULENSEK
Dr. Ekler Péter, Weitzl Zoltán

BUDAPEST, 2016

Tartalomjegyzék

Összefoglaló

A nagy múltra visszatekintő, globálisan operáló cégek számára is kihívást jelent lépést tartani az informatika fejlődésével és felismerni a bennük rejlő lehetőségeket. A felhő alapú technológiák megjelenése hatalmas lehetőséget hordoz magában a meglévő termék-életciklusok szinte minden állomásának megreformálására. Kis, - közepes és nagyvállalatok egyaránt profitálhatnak az alkalmazása következtében. Ezek a vállalatok, illetve iparágak a termékeik létrehozására már rendelkeznek jól bevált és az egyes termékekre vonatkozó féltve őrzött tervezési eljárásokkal. A kiforrott termék-életciklusokhoz való ragaszkodás azonban magában hordozza egy valóban innovatív gondolkodású versenytárs megjelenésének veszélyét, mely ideális versenykörnyezetben nagy múltú vállalatok gyors hanyatlását is előidézhetik.

A dolgozatom első részében a felhő alapú technológiának egy olyan alkalmazási területét szeretném reflektorfénybe helyezni, mely mérsékelt informatikai portfólióval rendelkező kis-, közepes és nagyvállalatok számára adhat útmutatást, ösztönözést meglévő eljárásaik, az általuk előállított termékek életciklusának megreformálására. Az elméleti megfontolások tárgyalása után a levonható következtetések két gyakorlatba is átemelhető példán keresztül szemléltetem.

A megújuló energetika területén a folyamatok hatásfoka elsődleges fontosságú. Előzetes feltáró munkám alapján egy terület, melyben a rendszerhatásfok növelhető a napenergia-átalakító rendszerek indulási és leállási időszakai. Az indulás és leállás feltételeinek ellenőrzése időt vesz igénybe. A végső cél ennek az időnek a minimalizálásán keresztül a megtermelt energia maximalizálása. A feladat tehát felhő alapú technológiák által nyújtott szolgáltatások igénybe vétele napenergia-átalakító rendszerek rendszerhatásfokának javítása céljából. A dolgozatomban tehát valós körülmények között szerzett üzemadatok alapján, alapvető statisztikai módszerek felhasználásával megvizsgálom, hogy a rendelkezésre álló információ mennyisége és minősége elegendő lehet-e egy optimalizáló algoritmus kidolgozásához. Megvizsgáltam továbbá, hogy további, ingyenesen hozzáférhető adatok bevonásával (pl. helyi időjárás) optimalizálható-e a működés és ezen adatok alapján javaslatot teszek egy felhőben futtatható vezérlési és egy szabályozási algoritmus megvalósítására, melynek alkalmazásával a rendszer-hatásfok növelhető.

A megújuló energiatermelés elterjedése nagyszámú elosztott energiatermelő egységek erősáramú hálózatra csatlakozását vonta maga után. Az ezekben a berendezésekben alkalmazott erősáramú szűrőkörök költsége adja a berendezések árának döntő hányadát. Üzemi adatok visszacsatolása nélkül könnyen alakulnak ki olyan termékcsaládok, melyek ugyan általános körülmények között alkalmazhatóak, ám az esetek döntő hányadában túlméretezettek, így feleslegesen drágák. Véleményem szerint a tervezési időben alkalmazott eljárásokat magukba foglaló szimulátorok felhőben való futtatásával a vázolt problémakör hatékonyan támadható.

A gyakorlati rendszerek többségét nem egy differenciálegyenlet, hanem differenciálegyenletek rendszere írja le.

A dolgozatomban egy rugalmas tengellyel összekötött hajtógép és terhelés, valamint egy hálózatra csatlakozó AC/DC átalakító szűrőköre eseteit mutatom be két eltérő mérnöki tudományterületből vett példaként arra. hogy az említett differenciálegyenlet-rendszerek általában azonos, vagy a lényeget tekintve hasonló alakra hozhatók, így a viselkedésük leírásához felhasználható – felhőben futtatható - matematikai apparátus is azonos lehet. A dolgozatom harmadik részében két darab, általánosan alkalmazható, időtartománybeli szimulációs eljárás alapját képező matematikai módszer alkalmazását összegzem (Runge-Kutta variánsok, állapot átviteli mátrix alapú megoldás). A rendelkezésre álló algoritmusok alapján elkészítettem egy felhőben futtatható szimulációs környezet vázát, melynek segítségével a rendszer viselkedésének tetszőlegesen pontos vizsgálata lehetővé válik.

Abstract

As I see it, in near future, any product – independent of the related field of engineering – is going to be designed and produced more efficiently thanks to the application of CPS (Cyber-Physical Systems) technology. A significant increase in design and production efficiency and thus reduction in cost will be achieved. It is important to point out the concept of an engineering-discipline-independent product. The concepts stated in this essay remain valid regardless of a product being associated to electrical-, mechanical-, chemical-, biological- or any other field of engineering.

Regardless of being an innovation leader or a manufacturing-focused entity, CPS-based technologies hold unexplored possibilities and a huge potential for any company. Small, medium and large businesses can all benefit from its application. It will lead to a radical renewal of traditional, long established companies, of industries characterized by products having since years, or decades passed the saturation point on their technology evolution curve. Process of creating of such artefacts is a well understood, documented and securely protected secret of a company. A common attribute of such product lifecycles are unintended obstacles eliminating, or at least significantly reducing the possibility of cutting-edge technology affecting the core of the mechanisms applied. The idea behind "Industrie 4.0" can be interpreted as to promote the effective transposition of up-to-date know-how created by information technology to traditional engineering areas, where they can serve as the driving force behind economic growth, plus providing a real challenge for education and daily-level operations planning.

Process efficiency is a priority in the field of renewable energy systems. Based on my preliminary exploration work in this area, system efficiency could be increased by optimizing starting and stopping periods of solar energy conversion systems. Evaluating start-up and shut-down conditions takes fair amount of time. The ultimate goal lies in the maximization of produced energy through minimizing these time intervals. The services provided by cloud-based technologies are suggested to be exploited in order to improve the system efficiency of solar energy conversion systems. Statistical methods running in the cloud and manipulating operating data obtained under real conditions are suggested to evaluate quantity and quality of information sufficient to develop an efficiency optimization algorithm. Inclusion of freely available data involving e. g. local weather history is also taken into account.

The widespread application of renewable energy production systems lead to a large number of distributed cogeneration units being connected to the power system. The power filters used in these converters account for the majority of cost of the equipment. For the majority of applications, these units may be oversized, thus unnecessarily expensive.

In my opinion, the problem may be efficiently tackled by running the simulators and algorithms used in design time - or during operation of the device - in the cloud to allow for the gathered data to be used for cost reduction through design process optimization. The majority of practical systems (e. g. LCL filter of an electrical network or a shaft connecting a load and a driving machine) can be described by a differential equation, or more generally by a set of differential equations. The algorithms used for the solution of these differential equations (thus used for simulation during design) are generally similar. In the third part of my work I use two generally applicable time-domain simulation cores (Runge-Kutta variants, state transmission matrix based solution) running in a cloud-based environment allowing for arbitrarily accurate analysis of dynamic system behavior.

1 Bevezetés

Manapság a felhő alapú szolgáltatások számtalan felhasználási lehetőségével találkozhatunk az élet szinte bármely területén. Az ipar számára is kiváló lehetőséget jelent ezeknek a technológiáknak a beépítése akár a gyártási folyamat, akár a különböző termék-életciklusok fejlesztési szakaszaiba. Az "*Industrie 4.0*" koncepció [1] szerint az informatikusok által birtokolt tudás és szemléletmód a hagyományos mérnöki tudományok területeire való hatékony átültetése a közeljövő gazdasági fejlődésének mozgatórugója és egyben az oktatás és ipari munkaszervezés egyik nagy kihívása.

Bizonyos szakterületeken tevékenykedő (bio-, vegyész-, gépész és erősáramú villamos-) mérnökök által tervezett termékek közös jellemzője, hogy hasonló portfóliójú vállalatok hasonló termékpalettáival kell versenyezniük a fogyasztói piacokon. Ezekben a szektorokban a valóban innovatív, az informatika területéhez köthető megoldások csak ritkán, akkor is csak viszonylag lassú dinamikával terjednek el. Kis, - közepes és nagyvállalatok, illetve iparágak a termékeik létrehozására már rendelkeznek jól bevált, évek tapasztalata és tudása eredményeként kidolgozott tervezési eljárásokkal, melyek által létrehozott termékek már évek, akár évtizedek óta elérték technológiai fejlődésük telítődési szakaszát. A tervezett termékek életciklusában akaratlanul is létrejönnek olyan gátak, melyek a fejlődés élvonalába tartozó technológiák beépülését nagymértékben megnehezítik. Ezek a kiforrott termék-életciklusok és tervezési eljárások magukban hordozzák azonban annak a veszélyét, hogy megjelenik egy vagy több valóban innovatív gondolkodású versenytárs a piacon, ami ideális versenykörnyezetben akár nagy múltú vállalatok gyors hanyatlását is előidézheti.

Véleményem szerint a felhő alapú technológiák alkalmazásával tetszőleges mérnöki szakterület keretein belül tervezett termékek előállítása hatékonyabbá és ez által olcsóbbá tehető. Fontos kiemelni, hogy az általános értelemben vett termék egy példa, mely akár egy tetszőleges másik is lehetne. A dolgozatom kidolgozása során törekedtem arra, hogy megfelelő példákkal alátámasszam azt az elképzelést, hogy a bemutatott felhő alapú technológia koncepciója szakterületeken átívelően, tetszőlegesen választott termékre alkalmazható.

1.1 Motiváció

A dolgozatom első részében tehát a felhő alapú technológiának egy olyan alkalmazási területét szeretném reflektorfénybe helyezni, mely mérsékelt informatikai portfólióval rendelkező, korlátozott know-how birtokában levő kis-, közepes és nagyvállalatok számára adhat útmutatást meglévő eljárásaik, az általuk előállított termékek életciklusának megreformálására.

A legfontosabb, amivel minden ipari szereplőnek tisztában kell lenni az adatfelhalmozás fontossága. Eszközök sorozatgyártása költségérzékeny folyamat.

Új funkciók bevezetése ritka eseménynek számít, főleg akkor, ha implementálásuk nem jár együtt rövidtávú haszonnövekedéssel, illetve költségcsökkenéssel. Szemléletváltásra azonban szükség van a vállalatok részéről. Fontos megérteni, hogy a felhő alapú technológiák területén szerzett tapasztalat befektetés a jövőbe, ami annál hamarabb térülhet meg, minél korábban kezdünk el segítségükkel adatokat gyűjteni. Ha figyelembe vesszük a folyamatos adatgyűjtés szinte teljes hiányát, valószínűsíthető hogy az ezen a területen elsőnek lépő vállalkozások jelentős versenyelőnyre tehetnek szert. Ez a kialakuló versenyelőny nem csupán a technológiai előrelépésen, hanem a marketing adta lehetőségek megfelelő kihasználásán is alapulhat.

A tudásfelhalmozás jól bevált folyamata a „Big Data" adathalmazok rendelkezésre állása következtében változás előtt áll, egy újfajta – tapasztalaton alapuló - tudás felhalmozási folyamat jelenik meg. A hivatkozott cikkben[2] is adatfelhalmozás segítségével törekszenek optimalizálni egy adott iparág termelési folyamatát. Ennek az új folyamatnak egyik jellemzője, hogy a hatására történő beavatkozás megelőzheti a tudományos értelemben vett megértést, mely a veszélyeket és lehetőségeket egyaránt magába hordozhat.

Az új termékek vagy szolgáltatások tervezésének, vagy a már meglévő megoldások kibővítésének egyik módja tehát a rendelkezésre álló adathalmazokban rejlő információ kinyerése lehet, mely információt kinyerő algoritmusok kutatása egyre inkább előtérbe kerül. Az ipari szereplők (első sorban kis-és közepes vállalatok) számára nem is ezeknek az algoritmusoknak a felkutatására, hanem az adatok mihamarabbi rendszeres begyűjtésére kellene összpontosítaniuk, hogy a jövőben, a szükséges információ rendelkezésre álljon. Ebben a megközelítésben az algoritmusok felkutatása első sorban a nagyvállalatok által támogatott kutatási-fejlesztési szektor feladata lenne.

Fontosnak érzem megemlíteni, hogy az információ-technológiai felhő által nyújtott szolgáltatásokat vizsgálva nem találunk olyat, melyet egy kis vagy közepes vállalkozás saját infrastrukturális megoldások segítségével nem alakíthatott volna ki már évekkel ezelőtt is. Ezeknek a szolgáltatásoknak a gyakorlati elterjedését lényegében két tényező hátráltatta: a költség és a technológiai hozzá nem értés.

Mivel ezek az új termék funkcionalitások nem mutatnak ki azonnali pozitív hatást a vállaltra nézve, így elképzelhetetlennek tartják, hogy a sorozatgyártott termékeket ezekkel az új funkciókkal (amik első sorban kommunikációs csatornák implementálását jelenti) lássák el. Az „Internet of Things" szlogen mögött húzódó folyamatok azonban mégiscsak ebbe az irányba mutatnak. Egy meglévő termék felhőképessé tétele nyilvánvalóan a tervezési és a gyártási költségek növekedését vonja maga után. Az adódó lehetőségek beépítése a költség-haszon elemzés folyamatába nehéz, első sorban a hatékony, közérthető és elterjedt adatfeldolgozó algoritmusok hiánya következtében. Véleményem szerint a költség oldalon kalkulálható rizikót a marketing és többletszolgáltatások területén elérhető hasznok legalább kiegyensúlyozzák.

A felhőszolgáltatások igénybe vételével a technikai akadályok szinte teljes egészében megszűnnek, illetve a technológiai hozzá- nem-értés is olyan jelentéktelen mértékűvé válhat, aminek áthidalására csupán a meglévő előítéletek és rossz tapasztalatok magunk mögött hagyása szükségeltetik.

A továbbiakban az eddig elhangzott általános gondolatmenetet próbálom meg konkretizálni.

A termékek tervezése egy összetett, szakterületeken átívelő feladat. A tervezés folyamata az esetek döntő többségében felosztható egy specifikációs, egy tervezési, egy bemérési, egy gyártási, egy telepítési és egy üzemeltetési fázisra. Visszacsatolás a telepítési és üzemeltetési fázisokról vagy teljesen hiányzik, de ha létezik is, általában nagyon lassú. A folyamat többi fázisairól nem is beszélve. A berendezések pl. villamos és mechanikus részegységei - a megfelelő biztonsági tényezők figyelembe vételével - cég specifikus, tapasztalati úton szerzett eljárások, vagy szakkönyvekben összefoglalt általános eljárások alapján terveződnek. A kialakult folyamatok könnyen vállnak merev, berögződött, érzésen alapuló szabályokká. Az említett részegységek rendszerint költséges, a berendezés árának jelentős hányadát kitevő összetevők. Ennek ellenére az optimalizálásukra nincs vagy csak nagyon korlátozottak a lehetőségek. A megtervezett és bemért részegységek feltehetően az esetek túlnyomó többségében túlméretezetten, többé-kevésbé pazarló módon üzemelnek. A tervező elvégezte a feladatát, a berendezés üzemel, problémák nem lépnek fel, ám az alkalmazott megoldás pazarló, a költséget a felhasználó fizeti meg.

Hasonlóan, az alkalmazható szabályozási algoritmusok elvileg széles skáláját a gyakorlatban nagymértékben korlátozzák az imént említett tényezők. Itt ugyan adott a fejlődés lehetősége, ám a gyakorlatban kevés vállalat engedheti meg magának az alkalmazás specifikus, de akár csak az általánosan alkalmazható algoritmusoptimalizálást. Ezen a szakterületen talán a kockázat kezelése a legnagyobb hátráltató tényező, mely véleményem szerint (felhő alapú) adatgyűjtéssel, szimulációval és beavatkozással hatékonyan kezelhető.

Több előnnyel is bírhat tehát, ha az adott termék rendelkezik egy felhő alapú szolgáltatáshoz való csatlakozási felülettel. A termék üzemelése során az üzemi adatok folyamatosan tárolásra kerülnek, melyek később feldolgozhatók, de a meglévő adatok például cégen belüli ötletbörzék, versenyek alapja lehet, melyek eredménye új termékek vagy szolgáltatások kialakulásához vezethet.

A tervezési folyamat során időben felhalmozott tudáson alapuló eljárások, modern szimulációs eszközök felhasználásával, esetleg tapasztalatokon alapú megérzések alapján készülnek szoftverek, valamint villamos és konstrukciós tervek. A jövőre vonatkozó következtetésekből fakadóan a hibázás lehetősége minden esetben fennáll, hiszen a tankönyvekből kiolvasható képletek levezetései elhanyagolásokkal élnek, az üzemi tapasztalatokból nyert következtetéshez rendelkezésre álló minták száma korlátozott, a személyes tapasztalatok pedig gyakran szubjektívek.

Felhő alapú adatgyűjtés bevezetésével első sorban a rendelkezésre álló minták számának növelésével ez a bizonytalanság csökkenthető. Szubjektív személyes tapasztalatok általánosíthatók, számítási eljárások pontosíthatók.

Gyakorlati példákkal is szemléltetni szeretném a korábban részletezett adatgyűjtéssel és feldolgozással kapcsolatos alkalmazási előnyöket.

Egy több éve sorozatgyártott termékben alkalmazott szabályozástechnikai megoldás egy konkrét környezetben képtelen volt megfelelően működni. Mint kiderült, a tervezés és az első üzembe helyezések során behangolt szabályozó kör az adott környezeti feltételek mellett instabillá vált. A probléma orvoslása több hetet igénybe vett és a szabályozási kör átstrukturálását vonta maga után. Az alkalmazott eljárás azonban több kérdést is felvetett:

- Melyik szabályozástechnikai megoldás kerüljön a sorozatgyártott termékbe?
- A jelenleg működő berendezések között lehet, amelyik a stabilitás határhelyzetéhez közel üzemel?
- A jelenleg működő berendezések közül melyik kerülhet instabil helyzetbe a környezeti változások közepette?

Ezek a kérdések folyamatos adatgyűjtés hiányában lényegében megválaszolhatatlanok, felhő alapú adatgyűjtés alkalmazásával azonban az ehhez hasonló helyzetekben például utólagos, céltudatos adatelemzéssel a megoldások egyszerűen kidolgozhatók. Ideális esetben az üzemelő rendszerek viselkedése proaktívan vizsgálható, a vázolt probléma kialakulása elkerülhető lenne.

Egy másik kérdés, ami felvetődhet, hogy az eszközt és a környezetét szabályozástechnikai szempontból szétcsatolni hivatott komponens (pl. szűrőkör) megfelelően lett-e méretezve. A szabályzó paraméterek, vagy a struktúra módosítása ugyanis csak az egyik lehetséges megoldása a problémának (nyilván a leggyorsabb). Kiderülhet azonban, hogy a módosított szabályzó alkalmazása mellett a szűrőkör mérete, ezáltal költsége csökkenthető. A szűrőkör méretének csökkentése az előbb vázolt problémától függetlenül is vizsgálható. Könnyen előfordulhat, hogy egy szűrőkör például egy sorozattermék részeként több évtizeden keresztül túlméretezetten üzemel, mert például egy olyan worst-case üzemre lett méretezve, ami elméletileg ugyan előállhat, a gyakorlati esetek többségében azonban távol esik a tényleges munkaponttól. Ebben az esetben az üzemi tapasztalatok szisztematikus begyűjtésének segítségével csökkenthetjük a sorozattermék árát is és egy optimális működés bevezetésének a lehetősége sem elérhetetlen többé. Számtalan irodalmat találhatunk az Industrie 4.0 jelentésével és elveivel kapcsolatban, a hivatkozott cikk[3] is ezzel a témakörrel foglalkozik. A dolgozat során ennek a koncepciónak egy olyan megvalósítását szeretném bemutatni, mely a felhő szolgáltatások segítségével könnyíti meg ezeknek az elveknek az iparban való elterjedését és hasznosítását.

1.2 Három nagy felhőszolgáltató összehasonlítása

Egy információtechnológiai felhőhöz történő csatlakozás árát több tényező határozza meg. A szolgáltatók által kínált csomagok konstrukciója mellett a felhasználói oldalon szükséges hardver, az átviendő adatmennyiség, és a kommunikáció gyakorisága is fontos szempont lehet.

A publikus felhőszolgáltatások piacán jelenleg három nagy szereplő dominál:

- Amazon Web Services (AWS),
- Google Cloud Platform (GCP), és
- Microsoft's Azure.

Ugyan az algoritmusaimat futtató környezet végső soron nem közülük került ki (Heroku), mégis jelentőségük következtében ezt a három szolgáltatást szeretném különböző szempontok szerint összehasonlítani. A szolgáltatás kiválasztásának legfontosabb szempontjai [4]:

- a szolgáltatott számítási kapacitás (compute),
- a szolgáltatott tárhely (storage),
- hálózat (networking), és természetesen
- a díjszabások (pricing structure).

A felhő alapú technológiák leírásában használt terminológia első sorban angol kifejezésekre épül, újszerű mivolta következtében magyar szakkifejezések nem terjedtek el, illetve nem alakultak ki. Ebből kifolyólag a következő – összefoglaló jellegű – fejezetekben angol kifejezéseket szerepeltetek, elősegítve így az érdeklődő olvasó számára az internetes kereső szolgáltatások hatékony használatát.

1.2.1 Számítási kapacitás (Compute)

Az úgynevezett EC2 (Elastic Compute Cloud) adja az AWS számítási szolgáltatásának magját. Az EC2 segítségével a felhasználóknak lehetőségük van saját virtuális gépük igény szerinti konfigurálására, de használhatnak már előre elkészített machine image-ket is. A felhasználó megadhatja a memória kapacitást, a virtuális gépek számát és választhat különböző régiók közül, ahol majd a virtuális gép futni fog. Az EC2 segítségével felügyelhetjük a terheléselosztást és a skálázást is.

A Google 2012-ben mutatta be a saját felhő szolgáltatását: Google Compute Engine (GCE). A GCE segítségével a felhasználók virtuális gépeket futtathatnak, hasonlóan az AWS-hez különböző régiókban. A szolgáltatás először nem volt mindenki számára elérhető, aztán később a Google folyamatosan bővítette az igénybe vehető szolgáltatásainak és jellemzőinek a listáját, mint például a terheléselosztás, skálázhatóság, operációs rendszer támogatás, gyorsabb perzisztens tárolók és példányok.

Szintén 2012-ben a Microsoft felhő szolgáltatása is bemutatásra került, de 2013 májusáig nem vált általánosan elérhetővé. Az Azure felhasználók VHD-kat (Virtual Hard Disk) választhatnak, ami jellegét tekintve hasonló az Amazon machine image-khez. Ezeknek a segítségével virtuális gépeket hozhatnak létre. A VHD lehet a Microsoft által, third party-k által, esetleg maga a felhasználó által meghatározott felépítésű és funkciójú. Minden VM esetén meg kell határozni a core-k számát és a memória nagyságát.

1.2.2 Tárhelyek és adatbázisok (Storage and databases)

Az AWS ideiglenes tárhelyet ad, ami egyszer lefoglalásra kerül, amikor példány elindul és felszabadul, mikor a példány leáll. Egy úgynevezett Block storage is a rendelkezésünkre áll, ami olyan, mint egy merevlemez, amit akár példányokhoz is hozzácsatolhatunk, de maradhatnak függetlenek is. Az AWS úgynevezett Object storage-ek is felkínál az S3 szolgáltatásával, valamint teljes mértékben támogatja a relációs és NoSQL adatbázisokat és a Big Data szolgáltatásokat is.

A Google's Cloud Platform az AWS-hez hasonlóan ideiglenes és perzisztens tárolókat is szolgáltat, Object storage-hez pedig a Google Cloud Storage-t használhatjuk. A GCP is támogatja a relációs adatbázisokat Google Cloud SQL néven. A legújabb Big Data technológiákat, mint pl. Big Query, Big Table és Hadoop is teljes egészében elérhetőek.

Az Azure ideiglenes tárolókat (D drive) és Page Blob-okat (Microsoft's Block Storage option) használ a VM alapú köteteihez, az Object Storage-hez pedig Block blob-kat és File serve-ket. Az Azure is támogatja a relációs és NoSQL adatbázisokat, a Big Data technológiákat a Windows Azure Table és HDInsight nevű szolgáltatások segítségével.

1.2.3 Hálózat (Networking)

Az Amazon's Virtual Private Clouds (VPCs) és az Azure's Virtual Network(VNET) lehetővé teszi a felhasználók számára, hogy a VM-ket elkülönített hálózatokba csoportosítsa a felhőben. VPC-ket ésVNET-ket használva a felhasználók hálózati topológiákat definiálhatnak, alhálózatokat, routing táblákat, privát IP cím tartományokat és hálózati átjárókat is készíthetnek.

Minden Google Compute Engine példány egy különálló hálózathoz tartozik, ami megadja a címtartományt és az átjáró címét minden példány számára, ami hozzá csatlakozik. Tűzfal beállításokat is megadhatunk a példányunknak, és ezen felül publikus IP címet is kaphat.

1.2.4 Díjszabások (Pricing structure)

Az AWS a használati órák számát számolja fel a felhasználóknak. A minimum használati idő egy óra. Az AWS példányok díjszabásának három csoportját különböztetjük meg:

- on demand: a vásárlók azért fizetnek, amit használnak,

- reserved: a vásárlók 1-3 évre lefoglalnak példányokat, amikért egy előre meghatározott összeget kell fizetniük felhasználás alapján,

- spot: a vásárlók ajánlatot tehetnek extra kapacitásért cserébe.

Az GCP a használati percek számát számolja fel a felhasználóknak. A minimum használati idő 10 perc. A Google további egy új hosszantartó-használati díjazást is bevezetett, aminek a segítségével egy az AWS-hez képest egyszerűbb és rugalmasabb megközelítést vezetett be.

Az Azure on-demand díjazás esetén a használati percek számát számolja fel.

1.3 Felhő technológiák alkalmazása az adott rendszerben

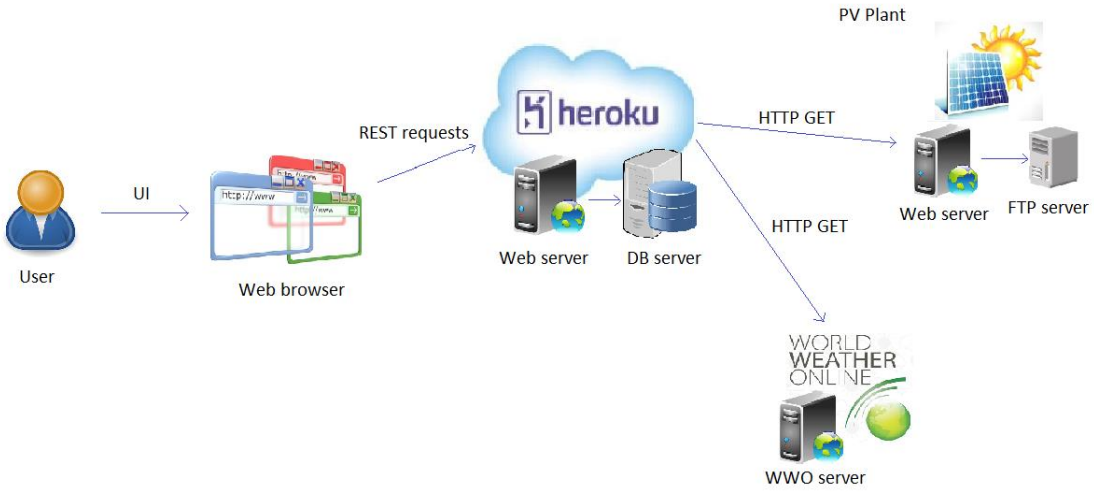
A dolgozatomban a felhő alapú technológiák két eltérő területen bevethető alkalmazási lehetőségeit szeretném bemutatni. Az ehhez szükséges –közös - rendszer felépítését szeretném a továbbiakban ismertetni.

Egy felhő alapú rendszer felépítése nagyon hasonlít egy kliens-szerver modell alapú rendszer felépítéséhez. Lényeges különbség, hogy a szerver pontos helye nem ismert, illetve a szerver üzemeltetése sem a felhasználó feladata. A felhasználó egy interfészen keresztül éri el a felhő által kínált szolgáltatásokat. Ahhoz, hogy egy felhő alapú rendszert alkossunk, sokféle szempontot kell figyelembe vennünk, mind a fejlesztői oldalról mind a felhasználói oldalról egyaránt[5].

A felhő által nyújtott szolgáltatások közül először egy Java alapú web alkalmazás hosztolását próbáltam ki. Elkészítettem egy egyszerű felépítésű *REST* alapú web alkalmazást. Az alkalmazás egy Tomcat[6] web szerveren fut, aminek a működését először lokálisan teszteltem, majd kerestem egy PaaS felhőszolgáltatást, aminek a segítségével az alkalmazás a felhőben futtathatóvá vált. A Heroku [7] egy ingyenes Cloud Application Platform, egy módszer web alkalmazások hosztolására. A szolgáltatás igénybe vételével a fejlesztők az alkalmazásaik implementálására koncentrálhatnak, nem kell energiát áldozni a szerverek menedzselésére, a skálázhatóság biztosítására. A Heroku esetében a rendelkezésre álló szerver típusok között megtalálható a mi rendszerünk teszteléséhez használt Tomcat web szerver is. A web alkalmazásainkat Git verziókezelő rendszer segítségével tölthetjük fel, de lehetőségünk van a saját WAR fájljaink futtatására is. A szerver oldali logika felépítése igen egyszerű. Mivel REST alapú kommunikációt használok, így http GET, POST, PUT és DELETE kérések segítségével kommunikálhatok a felhőben futó web szerverrel.

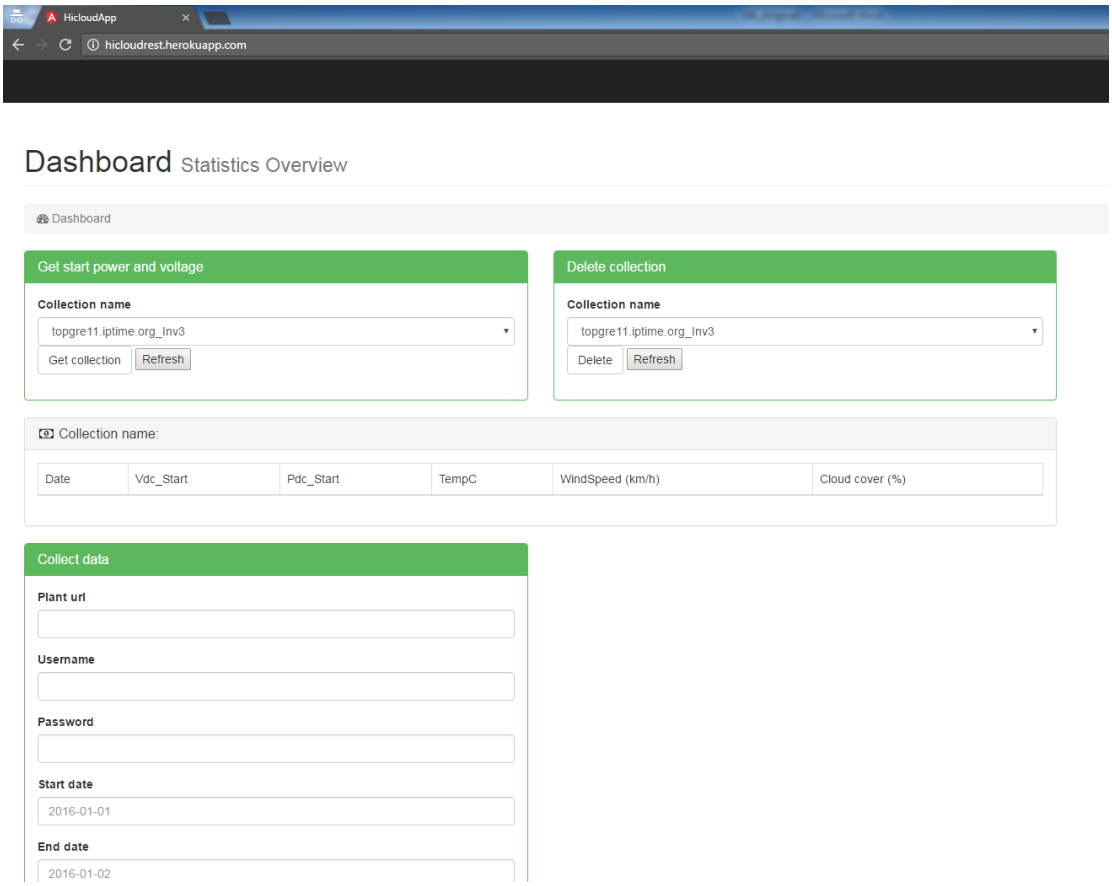
Mivel az adatgyűjtés és adatelemzés elengedhetetlen folyamatok az optimalizációs eljárások során, így szükségessé válik valamilyen adatbázis kezelő rendszer használata. Heroku használatakor különböző Addon-okat is csatolhatunk a web alkalmazásunkhoz.

A NoSql adatbázis kezelő rendszerek közül manapság a legelterjedtebb a MongoDb [8]. Egy MongoDb-t megvalósító Addon csatolása után ide kerül mentésre az adatgyűjtés során felhalmozott adatmennyiség. Az alábbi ábrán látható a rendszer felépítése és az egyes egységek közötti kommunikáció.



1. ábra Rendszer koncepció

Ahhoz, hogy az adatgyűjtés, az adatelemzés, és a későbbieknek a szimuláció futtatása és az eredmények kiértékelése egyszerűbb és szemléletesebb lehessen egy webes elérési felület is készítettem a web alkalmazáshoz. A fejlesztéshez a manapság egyre népszerűbbé váló Angular2[9] keretrendszert használtam, a weboldal kinézetét Bootstrap minták[10] segítségével alakítottam ki. Az alábbi képen látható az általam készített, böngészőből elérhető web alkalmazás. Az elkészült felhasználói felület segítségével a kutatásom során vizsgált megközelítési módok könnyebben konfigurálhatók.



2. ábra Elkészült web alkalmazás a böngészőben megnyitva

A hivatkozott cikkben leírtakhoz hasonlóan[11] én is kihasználom a felhő alapú szolgáltatások által nyújtott előnyöket, de én szeretném más szempontokból is megvilágítani a számítási felhő használatának előnyeit.

Az átviendő adatmennyiség és a kommunikáció gyakorisága szempontjából két esetet tételezek fel, melyek szorosan kapcsolódnak két részletesen bemutatott alkalmazási területhez. Az egyik ilyen az üzemi adatok napi rendszerességgel való begyűjtése, tárolása és feldolgozása. A másik pedig milliszekundumos, vagy az alatti időállandójú jelalakok beolvasása, feldolgozása és tárolása. Véleményem szerint a két eset jellegét tekintve jó példái a szóba jöhető alkalmazási területeknek. A rendelkezésre álló hardvert mind a két esetben azonosnak tételezem fel, feladata csupán a folyamatos internetelérés biztosítása a web alkalmazás felé.

A második fejezetben bemutatott megoldás egy viszonylag lassú adatgyűjtési és feldolgozási folyamatot igényel. Az adatokat napi rendszerességgel olvasatom és értékeltetem ki. A naperőmű napi működési adatai és a másnapra elérhető időjárási adatok ismeretében számolok ki egy beavatkozó jelet, aminek hatására a rendszer működése hatásfok szempontjából optimálisabbá válik. A napi rendszerességgel végzett adatgyűjtés következtében nagy adatmennyiség mozgatására nincs szükség. Az algoritmus végső állapotában berendezésenként napi két adat beolvasása és egy adat írása és IP címenként egy adat beolvasása elegendő. (Megjegyzés: az algoritmus fejlesztése során szükség volt nagyobb adatmennyiség feldolgozására is.)

A harmadik fejezetben bemutatott gondolat ezzel szemben nagy adatmennyiségek minél gyorsabb mozgatását és gyors adatfeldolgozást igényel. Szabályozástechnikai algoritmusok finomhangolása például teljesítményelektronikai eszközök kapcsolási frekvenciájával futó számítási eredmények begyűjtését igényli. A milliszekundum alatti valós idejű adatátvitel helyett az optimalizáció által igényelt mérési eredmények csomagban történő továbbítása terjedt el.

Egy rendszer felhőben történő szimulációjához szükséges bemeneti adatok (például az erősáramú hálózat egy periódusa) adatcsomag formájában érkeznek. Az adatcsomagok méretének a vezérlő algoritmust futtató beágyazott rendszerben rendelkezésre álló memória szab határt.

2 Rendszer hatásfok növelése felhő szolgáltatások segítségével

A megújuló energetika területén a folyamatok hatásfoka elsődleges fontosságú, mely a viszonylag magas befektetett tőke - bekerülési költség - megtérülésének záloga. A hagyományos értelemben vett, a vevő irányába kiválóan kommunikálható eszköz-hatásfokok maximalizálása a technológia elért fokán egyre nehezebb. A rendszer-hatásfokra természeténél, interdiszciplináris jellegénél fogva kevesebb hangsúly fektetődik, számonkérése az egymástól függetlenül működő komponensbeszállítók egymással versenyző érdekei miatt nehéz. Előzetes feltáró munkám alapján egy terület, melyben a rendszerhatásfok növelhető a napenergia-átalakító rendszerek indulási és leállási időszakai. Ilyen indulási és leállási szekvenciák az aktuális időjárási viszonyok függvényében naponta többször is lejátszódhatnak. Az indulás és leállás feltételeinek ellenőrzése időt vesz igénybe. A végső cél ennek az időnek a minimalizálásán keresztül a megtermelt energia maximalizálása. Az e módon előállított többlet energia arányaiban véve ugyan kicsi, ám egy olyan piaci környezetben ahol az eszközhatásfok tized százalékokkal való megnövelése a realitásoktól elrugaszkodott vágyálomnak tekinthető sokat számít és jelentős marketing előnyökkel bír. A feladat tehát felhő alapú technológiák által nyújtott szolgáltatások igénybe vétele napenergia-átalakító rendszerek rendszerhatásfokának javítása céljából. A kérdéskör tárgyalását az alábbi fejezetekben vázolt szemszögekből közelíteném meg.

2.1 Megvalósíthatósági vizsgálatok

Ebben a fejezetben a napenergia átalakító rendszerek indulási és leállási időszakainak megfigyelésén alapuló kutatási és fejlesztési folyamatot szeretném bemutatni. Megvizsgálom, hogy valós körülmények között szerzett üzemadatok alapján, alapvető statisztikai módszerek felhasználásával a rendelkező információ megfelelő lehet-e egy optimalizáló algoritmus kidolgozásához.

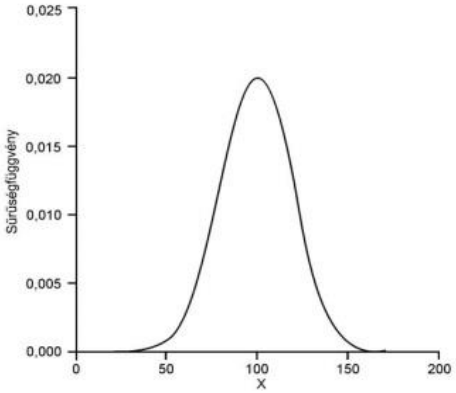
Az erőművekhez tartozik egy-egy web szerver. Az erőművekhez tartozó inverterekről folyamatosan gyűlnek bizonyos üzemadatok, melyek egy FTP szerveren tárolódnak. Minden nap keletkezik egy fájl, amiben az adatok 5 perces mintavétel időközönként kerülnek lementésre.

Ezen tárolt üzemadatok közül az indulási feszültség és indulási teljesítmény értékeket kezdtük el vizsgálni. A naperőművek éjszaka nem üzemelnek. Reggel elindulnak, amint az üzemükhöz szükséges körülmények rendelkezésre állnak. Az erőmű által termelt teljesítmény DC feszültség és árama szorzata. A DC feszültség értékét a környezeti hőmérséklet, a DC áram értékét a napsugárzás határozza meg. A nap emelkedésével a napelemek üresjárási feszültsége már viszonylag korán beáll a környezeti hőmérséklet által meghatározott értékére. Az inverter az ún. üresjárási feszültség adott szintet meghaladó értéke után az erősáramú hálózatra csatlakozik.

A gond az, hogy az első sorban a környezeti hőmérséklettől függő üresjárási feszültség nem határozza meg egyértelműen a modulokból felvehető teljesítményt. Előfordulhat, hogy az indulási feszültséghez nulla közeli áram (így teljesítmény) tartozik. Az energiatermelés viszont a folyamatos hálózati üzem feltétele. Teljesítmény hiányában az eszköz lecsatlakozik a hálózatról, és a folyamat - 10 perc nagyságrendű várakozási idő eltelte után - kezdődik előröl.

A gyakorlatban az inverterek tervezésekor, illetve üzembe helyezésekor előre meghatározott indulási feszültség értékeket állítanak be és évtizedeken keresztül eszerint működnek. Mivel a reggeli indulás (erősáramú hálózatra csatlakozás) pillanatában (amikor a bemeneti feszültség eléri a beállított indulási feszültség értékét) a rendszer pillanatnyi termelése szinte bizonyosan nem esik egybe a pillanatnyi fogyasztással, így vagy termelési többlet, vagy hiány keletkezik. Hiány esetén a berendezés leáll, majd adott idő elteltével újra próbálkozik. A termelési többlet az erőmű tulajdonosa számára veszteség, hiszen annak a jele, hogy a naperőmű már percekkel, esetleg tíz percekkel korábban is elkezdhette volna az energiatermelést. Az említett, esetlegesen több tíz perces várakozási idő további veszteségként jelentkezik. A célom, hogy egy felhőben futó algoritmus felhasználásával, napi rendszerességgel automatikusan megtaláljam és beállítsam azt a megfelelő indulási feszültségszintet, amivel egy adott napon az indulás pillanatában az energiatermelő rendszer üzeme optimális lesz.

A megfigyeléseim kezdetekor azzal az egyszerű alapfeltevéssel éltem, hogy a napi rendszerességű indulási folyamatok egymástól független események sorozatának tekinthetők, így az indulási feszültség és teljesítmény értékek normális eloszlást követnek. Gauß eloszlás esetén az egyes események (azaz a mindenkori, aznapi indulási feszültség és teljesítmény) bekövetkezésének valószínűségei egy haranggörbe mentén helyezkednek el. Amennyiben a feltételezésem igaznak bizonyulna, elegendő lenne hosszú távú minták átlagának számításával frissíteni a beállított indulási feszültség értékét a gyakorlatban beállított konstans érték helyett. Minél több független eseményt vizsgálunk, annál inkább a középérték körül központosulnak az egyes valószínűség értékek és a rendszer hatásfok szempontjából egyre optimálisabban indulna el. A görbe szélessége, vagyis a szórás az adatmennyiséggel arányosan csökkenne, a haranggörbe alakja egyre keskenyebb lenne.



3. ábra Gauß eloszlás

A rendelkezésre álló adatok száma folyamatosan (napról napra) bővült. Eleinte még csak a rendelkezésre álló kisebb adatmennyiséget figyelembe véve vizsgálódtam. Öt erőmű és az ahhoz tartozó teljesítményelektronikai átalakítók adatain különböző statisztikai vizsgálatokat végeztem. A manuálisan gyűjtött üzemadatok .csv fájl formátumban állnak rendelkezésre. Ebből az adathalmazból szűrtem ki az indulási feszültség és teljesítmény értékét. Indulási teljesítményként a napi adatok közül az első nem nulla értékű teljesítmény értéket kerestem meg, míg az indulási (üresjárási) feszültségként az ezt megelőző mintához tartozó feszültség értéket tároltam el. Egy Excel táblában mind az öt erőműre kigyűjtöttem egy kb. 2 hetes (17 napos) intervallumra nézve ezeket az adatokat.

Uoc	Pstart	Uoc	Pstart	Uoc	Pstart	Uoc	Pstart
#1				#2			
#1		#2		#1		#2	
703,5	5,3	687,1	4,4	711	5,1	708,5	5
711,7	7,5	716,2	7,6	713,4	5,3	711,8	5,3
669,8	3,1	683,2	3,7	665,1	3,8	662,4	3,7
692,3	6,2	697,5	6,2	684,9	4,5	683,7	4,3
688,9	4,5	692,6	4,6	687,4	6,1	685,8	4,4
705,6	6,2	710	6,2	719	5,1	718,3	5
704,8	5,8	643,1	1,4	726,5	5,7	725,4	5,4
645,1	1,5	649,6	1,5	719,4	5,2	719,3	5,2
674,9	3,4	680,5	3,8	678,8	3,8	677,6	3,8
700,5	7,4	637,4	2,1	705	8	714,3	4,7
693,2	5,3	697,5	5,3	719,9	5,4	718,9	5,3
702,1	7,5	637,8	1,8	724,3	5,5	723	5,3
700,7	9,1	692,5	6,8	665,7	3,1	665,6	3,1
703,6	9,5	645,8	2,7	674,7	4	674,3	5,2
673,8	4,8	680,1	5	690,4	6,7	687,7	3,8
696,3	7,2	648,8	2,1	720,2	5,2	718,8	5,3
704,9	6,4	645	1,5	729,9	5,8	728,9	10,8
MIN	MIN	MIN	MIN	MIN	MIN	MIN	MIN
645,1	1,5	637,4	1,4	665,1	3,1	662,4	3,1
MAX	MAX	MAX	MAX	MAX	MAX	MAX	MAX
711,7	9,5	716,2	7,6	729,9	8	728,9	10,8
TARTOMÁNY	TARTOMÁNY	TARTOMÁNY	TARTOMÁNY	TARTOMÁNY	TARTOMÁNY	TARTOMÁNY	TARTOMÁNY
66,6	8	78,8	6,2	64,8	4,9	66,5	7,7
VÁRHATÓ ÉRTÉK	VÁRHATÓ ÉRTÉK	VÁRHATÓ ÉRTÉK	VÁRHATÓ ÉRTÉK	VÁRHATÓ ÉRTÉK	VÁRHATÓ ÉRTÉK	VÁRHATÓ ÉRTÉK	VÁRHATÓ ÉRTÉK
692,4529412	5,923529412	673,2176471	3,923529412	702,0941176	5,194117647	701,4294118	5,035294118
SZÓRÁS	SZÓRÁS	SZÓRÁS	SZÓRÁS	SZÓRÁS	SZÓRÁS	SZÓRÁS	SZÓRÁS
17,25235917	2,092107016	27,00266735	2,042219813	22,27965638	1,170721246	22,62000676	1,645805721
NORM SÜRÜSÉG	NORM SÜRÜSÉG	NORM SÜRÜSÉG	NORM SÜRÜSÉG	NORM SÜRÜSÉG	NORM SÜRÜSÉG	NORM SÜRÜSÉG	NORM SÜRÜSÉG
2,31%	19,07%	1,48%	19,53%	1,79%	34,08%	1,76%	24,24%

4. ábra Excel tábla példa kétheti adattal

Hogy átfogó képet kaphassak a vizsgálandó adatok jellegéről, első lépésként meghatároztam az egyes inverterekre vonatkozóan az adathalmazok minimum és maximum értékét és ez alapján az értéktartományt.

Első ránézésre feltűnt, hogy a teljesítményadatok felbontása nagy valószínűséggel nem megfelelő pontos következtetések levonására.

A kW-os nagyságrendben érkező adatok felbontása túl nagy. Sajnos 500kW-os, 1 MW-os energia-átalakító eszközök esetén ez a pontosság 2%, illetve 1%, ami a gyakorlati felhasználási esetek többségében (pl. energiaszámláló) elfogadható érték. Így a felbontás javulására sajnos kevés az esély. Pontatlan indulási teljesítmény mérés mellett az optimalizáció már csak az indulási feszültség értékeire alapozhat. Az indulási feszültségek felbontása szerencsére megfelelőnek bizonyult. Mivel normális eloszlás esetén a valószínűség érték az átlag körül a legmagasabb, így meghatároztam az átlagot is, illetve a normális eloszlás sűrűségfüggvényének segítségével kiszámoltam, hogy – a rendelkezésre álló adatok alapján- az egyes indulási feszültség és teljesítmény értékek milyen valószínűséggel vennék fel az adott sokaság átlagát. Említettem, hogy magas valószínűség esetén az indulási feszültségek átlagát, mint új indulási feszültséget beállítva optimálisabb rendszerviselkedés állna elő. Kisméretű sokaság esetén látható, hogy az előbb említett valószínűség érték (tehát annak a valószínűsége, hogy egy adott napon az indulási feszültségértéke a rendelkezésre álló minták átlaga lesz) alacsony. További mintákra volt szükségem annak megállapítására, hogy ez az alacsony érték a minták alacsony számából fakad-e.

A következő lépésben megpróbáltam minél több adatot begyűjteni a vizsgálataimhoz. Ehhez nyújt hatalmas segítséget a felhő adta szolgáltatások igénybe vétele, hiszen a rendelkezésre álló tárolókapacitás skálázható, az adatgyűjtési és az adat manipulációs folyamatok automatizálhatók.

Az inverterek FTP szerverein kb. 1 évre visszamenően kerülnek tárolásra az üzemadatok. Ezt az egy éves adatmennyiséget emelem át egy saját MongoDB adatbázisba, majd ezen a felhőben tárolt adatmennyiségen végeztem el a felhőben futó statisztikai számításokat.

Az adatbázisban eltároljuk az inverter azonosítóját (erőmű web szerverének domain neve + az inverter id-ja), a dátumot (éééé-hh-nn formátumban), a mintavétel idejét, a mért DC feszültséget és DC teljesítményt. Ebből az adathalmazból a fent említett algoritmus adaptációját lefuttatva határozom meg az indulási feszültség és teljesítmény értéket az adott napra vonatkozóan. Lehetőségem van az egyes inverterekre lebontva is megjeleníteni a kiszámolt statisztikát, illetve egy összesítő táblázatban megnézhetem az összes, az adatbázisban tárolt inverterre vonatkozó kiszámolt értékeket.

Statistics result

mjsolar1-2.iptime.org Inv0	Min	Max	Range	Mean	Standard dev	Normal density	Normal dist
Vdc	620.3	799.9	179.6	687.1432584	30.24591016	1.32%	0.5
Pdc	0.9	15.9	15.0	5.5747191	2.88848733	13.81%	0.5

5. ábra Egy inverterre vonatkozó statisztikai adatok

Summary

Inverter	Vdc_Min	Vdc_Max	Vdc_Range	Vdc_Mean	Vdc_Standard dev	Vdc_Normal density	Vdc_ dist
sunshin01.iptime.org_Inv1	629.7	868.5	238.8	725.0981030	40.93021595	0.97%	0.5
rlaeka.iptime.org_Inv1	632.9	832.9	200.0	694.7193548	30.45377216	1.31%	0.5
sunshin01.iptime.org_Inv0	658.7	862.4	203.7	726.3040323	39.66504446	1.01%	0.5
rlaeka.iptime.org_Inv0	640.3	796.3	156.0	695.4253369	29.38627929	1.36%	0.5
topgre11.iptime.org_Inv1	627.5	761.6	134.1	685.8320955	22.29297080	1.79%	0.5
topgre11.iptime.org_Inv2	626.6	781.2	154.6	688.3912698	24.22116931	1.65%	0.5
topgre11.iptime.org_Inv3	628.5	766.3	137.8	685.7986772	22.49095128	1.77%	0.5
topgre11.iptime.org_Inv0	607.9	782.4	174.5	685.9857143	24.78261257	1.61%	0.5
powerecho01.iptime.org_Inv0	622.6	760.7	138.1	679.4289398	27.20362962	1.47%	0.5
powerecho01.iptime.org_Inv1	622.1	762.4	140.3	678.2406877	26.42344735	1.51%	0.5
mjsolar1-2.iptime.org_Inv0	620.3	799.9	179.6	687.1432584	30.24591016	1.32%	0.5
mjsolar1-2.iptime.org_Inv1	628.5	802.2	173.7	689.3835196	36.95651062	1.08%	0.5

Count

6. ábra Összegző statisztikai táblázat

Az Excel tábla létrehozásakor ez a letöltés kézzel történt, de ekkora adatmennyiség esetén nyilván ezt a folyamatot automatizálnunk kellett, így az erőművekhez tartozó web szerverekről http GET kérések segítségével töltöm le az adatokat. Egy GET kérést kellett indítani az erőművek web szervere felé. Ahhoz, hogy beléphessük akár az erőműhöz tartozó weboldalra, meg kell adnunk egy felhasználó nevet és egy jelszót. A http kérésünk fejlécében így Basic auth segítségével azonosítanunk kellett magunkat, utána jöhet csak a GET kérés tartalma. A GET kérésben megadhatjuk az inverter azonosítóját és a dátumot, és válaszként egy .csv formátumú text-et kapunk, ami feldolgozva elmentjük az adatbázisba az adatokat.

Az adatbázisban tárolt adatok formátuma nagyon hasonlít egy JSON fájl formátumára. Ebben az esetben dokumentumokat, kulcs-érték párokat tárolunk, és ezekhez férhetünk hozzá. A bemutatott statisztikai számítások elvégzéséhez egy Java könyvtárat használok, amiben a szükséges függvények implementációja már rendelkezésre áll. A leggyakrabban használt és legnépszerűbb ilyen library ismereteim szerint az Apache Commons Math Library. Egy double tömbön az átlag számítását például egy ilyen egyszerű függvény segítségével végezhetjük el.

```
public double getMeanValue(List<Double> numberArray) {
     DescriptiveStatistics stats = new DescriptiveStatistics();
     for (int i = 0; i < numberArray.size(); i++) {
          stats.addValue(numberArray.get(i));
     }
```

```
    double mean = stats.getMean();
    return mean;}
```

Az összefoglaló táblázatban látott adatok alapján, megállapíthatjuk, hogy az indulási feszültségre vonatkozó feltételezéseink nem helyénvalóak. A vizsgálatba bevont adatok számának növelésével a szórás értékek nem csökkentek, a minták átlagának valószínűsége sem nőtt. Ebből a megállapításból kiindulva megállapítható hogy egy naperőmű indulási feszültségeinek értékei nem tekinthetőek egymástól függetlennek.

2.2 Optimalizálási lehetőségek vizsgálata

Ebben a fejezetben megvizsgálom, hogy további ingyenesen hozzáférhető adatok bevonásával, ebben az esetben a helyi időjárás adatok segítségével optimalizálható-e a működés.

Említettem, hogy az üresjárási feszültség a mindenkori időjárási viszonyok függvénye. A helyi időjárás adatok ingyenesen hozzáférhetőek és további vizsgálatra adnak lehetőséget. Mivel az inverterekről egy évre visszamenőleg áll rendelkezésre információ, így egy olyan API-t kerestem, ahol az időjárási adatok is visszamenőlegesen hozzáférhetők. Az erőművekhez tartozó web szerver IP címe ismert, így legkönnyebben IP cím alapján tudtam meghatározni és letölteni az adott területhez és az adott naphoz tartozó időjárási adatokat. A World Weather Online API Premium szolgáltatása minden kritériumot teljesített. Ez a szolgáltatást csak egy bizonyos időtartamra vonatkozóan ingyenes, az indítható kérések száma is korlátozott (napi 500 kérés), így a tervezettnél tovább tartott begyűjteni az időjárásra vonatkozó információkat.

Az adatok letöltéséhez szintén egy http GET kérést kell indítanunk az API felé, ebben a kérésben szerepeltetni kell a regisztráció során kapott API kulcsot. A GET kérésünk paramétereit változtatva adhatjuk meg milyen jellegű információra van szükségünk. A mi esetünkben egy API kéréshez tartozó Uri alakja:

```
http://api.worldweatheronline.com/premium/v1/past-
weather.ashx?key=******************&q=121.158.105.136&format=json&d
ate=2015-10-13&tp=24
```

Láthatjuk, hogy első paraméterként az API key-t adjuk meg, ezután következik az IP cím, a válasz formátuma (JSON), hogy melyik napra vagyunk kíváncsiak és az idő intervallum, ami a mi esetünkben egy 24 órás átlag. 24 órás átlag, mert első közelítésben az előző napi 24 órás átlag és a következő napi indulási feszültség között kívánunk összefüggést keresni.

A visszakapott JSON formátumú válasz formátuma az API dokumentációjában részletesen le van írva. A formátum alapján a GSON library segítségével osztályokat vettem fel, amiket annotációkkal ellátva a sztring-ként kapott válasz automatikusan Java objektummá alakítható. Az így keletkezett objektumokból a szükséges információ egyszerűen kinyerhető.

A legfontosabb információk - melyeket le is tároltunk az indulási adatok mellé az adatbázisba - az adott napi átlaghőmérséklet, a szélsebesség és a felhőborítottság mértéke. Az adatbázisba lementett adatokat a web felületen egy táblázatban is megtekinthetjük az adott inverter kiválasztása után.

Az előző fejezetben vázoltakhoz hasonlóan most is először egy kisebb adatmennyiség és Excel segítségével végeztem el a korrelációk vizsgálatát.

A korreláció két tetszőleges érték közötti lineáris kapcsolat nagyságát és irányát jelzi. A korrelációs együttható (r) előjele a kapcsolat irányát mutatja meg, a nagysága (0-1 közötti szám) pedig az együtt járás szorosságát, az összefüggés erejét mutatja. A statisztikában a tapasztalt korrelációt a következőképpen számítják[12],

$$r_{xy} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})*(y_i-\bar{y})}{s_x*s_y},$$

ahol *n* az adatok száma, $\bar{x}$ és $\bar{y}$ az egyes változók várható értékei, $\bar{s_x}$ és $\bar{s_y}$ pedig a szórást jelölik. Az általunk gyűjtött adatok esetén a várható érték az átlag, az ettől való eltérést szorozzuk és összegezzük, osztjuk az adatok darabszámával, majd a szórások szorzatával. Az Excel KORREL függvénye ezt a korrelációt számítja ki.

Első megközelítésben az indulási feszültség és teljesítmény közötti korrelációt vizsgáltam. Kevés adatmennyiség esetén (pl. a kéthetes adatgyűjtés eredménye) aránylag erős korreláció figyelhető meg a két érték között.

Ezután kiválasztottam egy tetszőleges invertert és az egy év alatt gyűjtött adatok (indulási feszültség, teljesítmény, hőmérséklet, szélsebesség, felhőborítottság) közti korrelációt vizsgáltam. Érdekes módon minél több a rendelkezésre álló adat, annál kisebb a korreláció értéke az indulási feszültség és teljesítmény között. Ugyan ezt a vizsgálatot egy másik inverter esetén elvégezve már nem tapasztaltam ezt az összefüggést, de a gyenge korreláció ott is kimutatható. Érdemes lenne megvizsgálni, hogy ez a gyenge korreláció az indulási teljesítmény mérésnek pontosításával emelhető-e.

	Korrelációs együttható
Vdc ~ Pdc	0,149251959
Vdc ~ Temp	-0,659491721
Pdc ~ Temp	0,440175731
Vdc ~ Cloud	-0,217457428
Pdc ~ Cloud	-0,055604329
Vdc ~ Wind	0,178246311
Pdc ~ Wind	-0,045629049

7. ábra Pearson-féle korreláció számítás eredménye

A legnyilvánvalóbb összefüggés, ami ebből a vizsgálatból kiolvasható az indulási feszültség és a hőmérséklet közötti fordított irányú korreláció.

Ez az összefüggés a PV modulok fizikai tulajdonságával magyarázható. A napelemek villamos tulajdonságai függnek a környezeti hőmérsékletétől, az üresjárási feszültség fordítottan arányos a hőmérséklettel.

A környezeti hőmérséklet változása befolyásolja tehát leginkább a modulok feszültségét, vagyis napról napra vizsgálva, a hőmérséklet csökkenésével az indulás pillanatában fennálló üresjárási feszültség növekedni fog.

A korrelációs vizsgálatot minden elérhető inverterre elvégeztem. A korrelációszámítást Java nyelven implementáltam és ez az algoritmust a felhőben futtattam. Az eredmények az alábbi táblázatban láthatóak.

Correlation

Inv name	Vdc ~ Pdc	Pdc ~ Temp	Vdc ~ Temp	Vdc ~ CloudCover
topgre11.iptime.org_Inv3	0.09761829311393425	0.5400720489994267	-0.48461576118226773	-0.004029146013721344
sunshin01.iptime.org_Inv1	0.1642051244404863	-0.04630921604134186	-0.7155703399918674	-0.18737164232076242
topgre11.iptime.org_Inv2	0.22413577747682456	0.08723308063859232	-0.5535636129661985	0.011104838233803238
rlaeka.iptime.org_Inv1	0.2110601225945782	0.07769051950829585	-0.5472363242512026	-0.14513556438823805
mjsolar1-2.iptime.org_Inv0	0.17589719142380275	0.4308306098768293	-0.6417333515303381	-0.22340050005769543
topgre11.iptime.org_Inv1	0.12205616342474986	0.6040131894869617	-0.4669899329089134	-0.0324661171561512
sunshin01.iptime.org_Inv0	0.2006270969108083	0.053682121347778174	-0.6523380901368284	-0.12994681363298163
powerecho01.iptime.org_Inv0	0.37594373568512585	0.2018525379787083	-0.40300999591324915	-0.02986684747197387
mjsolar1-2.iptime.org_Inv1	0.5531864752200536	-0.06158068342615252	-0.6773286609355473	-0.2566130837855516
topgre11.iptime.org_Inv0	0.10485877192222522	0.5888008331730962	-0.486826504874109	-0.05056852126053001
powerecho01.iptime.org_Inv1	0.21566349020920458	0.053827864118553344	-0.3826924074453173	-0.07263230150812067
rlaeka.iptime.org_Inv0	0.15273695819585895	0.2653497692021779	-0.5587699865549272	-0.14863615959819979

8. ábra Felhőben futtatott Pearson-féle korrelációs számítási eredmények

A táblázatban kiemeltem az üresjárási feszültség és a hőmérsékletre vonatkozó korrelációs együtthatók értékét. Nagyobb adatmennyiség esetén is jól látható a negatív korreláció, illetve az is megfigyelhető, hogy az egy erőműhöz tartozó inverterek nagyjából ugyanazt a korrelációs együttható értéket adja erre a két paraméterre vonatkozóan.

A korrelációszámításnak a jelfeldolgozásban[13] is nagy szerepe van. Segítségével megállapítható, hogy két jel mennyire hasonlít egymáshoz. Ezt az alkalmazott technikát más területen is hasznosítani lehet (kommunikáció, jelfeldolgozás, képfeldolgozás). A korábban említett Pearson-féle korreláció mellett ez egy más fajta korreláció, ami nem a lineáris kapcsolat erősségét, hanem az adatok hasonlóságára világít rá[14]. Ezt a korrelációs együtthatót az alábbi összefüggés szerint számolhatjuk ki: először összeszorozzuk a két paraméter összetartozó értékeit, majd ezeket summázzuk. Végül a két paraméter közül kiválasztunk egy úgy nevezett „referencia jelet”.

Ennek a paraméternek az összetartozó értékeit önmagával összeszorozzuk és ezeket az értékeket is summázzuk. Végül a két kapott eredményt elosztjuk egymással, ezzel normalizáljuk, illetve egy százalékos hasonlóságot is számolhatunk belőle.

Fontos kiemelni, hogy egy azon erőműhöz tartozó inverterek összehasonlítása a felhő egy újabb alkalmazási területére világít rá, ami angol terminológiával a „Predictive Maintenance” kategóriába esik.

Uoc_1.1 * Uoc_1.2	Uoc_1.1 * Uoc_1.1	Pstart_1.1 * Pstart_1.2	Pstart_1.1 * Pstart_1.1	Uoc_2.1 * Uoc_2.2	Uoc_2.1 * Uoc_2.1	Pstart_2.1 * Pstart_2.2	Pstart_2.1 * Pstart_2.1
483374,85	494912,25	23,32	28,09	503743,5	505521	25,5	26,01
509719,54	506516,89	57	56,25	507798,12	508939,56	28,09	28,09
457607,36	448632,04	11,47	9,61	440562,24	442358,01	14,06	14,44
482879,25	479279,29	38,44	38,44	468266,13	469088,01	19,35	20,25
477132,14	474583,21	20,7	20,25	471418,92	472518,76	26,84	37,21
500976	497871,36	38,44	38,44	516457,7	516961	25,5	26,01
453256,88	496743,04	8,12	33,64	527003,1	527802,25	30,78	32,49
419056,96	416154,01	2,25	2,25	517464,42	517536,36	27,04	27,04
459269,45	455490,01	12,92	11,56	459954,88	460769,44	14,44	14,44
446498,7	490700,25	15,54	54,76	503581,5	497025	37,6	64
483507	480526,24	28,09	28,09	517536,11	518256,01	28,62	29,16
447799,38	492944,41	13,5	56,25	523668,9	524610,49	29,15	30,25
485234,75	490980,49	61,88	82,81	443089,92	443156,49	9,61	9,61
454384,88	495052,96	25,65	90,25	454950,21	455220,09	20,8	16
458251,38	454006,44	24	23,04	474788,08	476652,16	25,46	44,89
451759,44	484833,69	15,12	51,84	517679,76	518688,04	27,56	27,04
454660,5	496884,01	9,6	40,96	532024,11	532754,01	62,64	33,64
7925368,46	8156110,59	406,04	666,53	8379987,6	8387856,68	453,04	480,57
	0,9717092		0,6091848		0,9990618		0,9427138

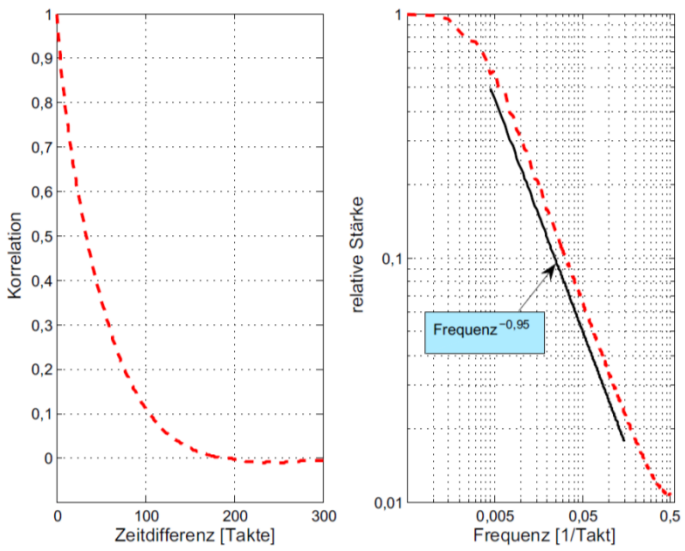
9. ábra Korreláció számítás eredménye

A fenti táblázatban az egyes erőművekhez tartozó invertereket hasonlítottuk össze az indulási feszültség és teljesítmény alapján. A kijött korrelációs együtthatók értékéből következtetni lehet az adott inverter esetleges hibás működésére, hiszen két ugyanott és ugyanazon körülmények között működő inverternek nagyon hasonlóan kellene működnie, mégsem ezt látjuk az indulási teljesítmény esetén pl. az első vizsgált erőmű invertereinek esetében. A különbségek okát valószínűleg a modulok közötti fizikai kapcsolat hibájában kell keresni. A feszültségek közti eltérés soros kapcsolás kötéseinek, a teljesítmények közti eltérés párhuzamos kapcsolás kötéseinek hibájára utal.

A kutatási folyamat eredményeként arra következtethetünk, hogy a vizsgált rendszer esetében egy matematikai értelemben vett komplex rendszerről[15] beszélhetünk. Erre első sorban a környezeti hőmérséklettől való függés alapján következtethetünk. Adott földrajzi hely időjárási viszonyai ugyanis (pl. a földrengésekhez hasonlóan) a komplexitás területéhez sorolható matematikai apparátussal írható le. Sokféle definíció létezik a komplexitás fogalmára, de mindegyikben definíció alapján kétféle aspektusa különböztethető meg: egy rendszer lehet komplex struktúrájú és komplex viselkedésű. Az általunk vizsgált rendszer esetében valószínűleg mindkét aspektus teljesül, hiszen struktúra szempontjából kijelenthetjük, hogy sok összetevőből (időjárás hatása, modulok telepítési hibái) áll és ezen összetevők között nem határozható meg egy egyszerű, lineáris kapcsolat.

Viselkedés szempontjából sem tehetünk konkrét megállapításokat, hiszen a rendszernek viszonylag változatos és nehezen kiszámítható reakciói lehetnek egy adott eseményre vonatkozóan. A komplex rendszerek egy viszonylag új tudományterületnek nevezhető, ami azt kutatja, hogy egy rendszer részei hogyan hozzák létre az egész rendszer együttes viselkedését és a rendszer milyen kölcsönhatásban áll a környezetével. Komplex rendszerek szinte bármilyen életterületen felfedezhetőek, ebben a kontextusban egy adott földrajzi területhez köthető időjárási viszonyok változása is egy komplex rendszernek tekinthető.

Bizonyos események gyakoriságának vizsgálatakor - komplex rendszerek esetén - fontos szempont az időbeli lefolyás, vagyis az események sorrendjének a vizsgálata. Egy adott esemény hogyan befolyásolja az időben rákövetkező esemény valószínűségét. Komplex rendszerek viselkedésének vizsgálata, illetve egy rendszer komplex voltának a vizsgálata a frekvenciatartományban történhet. Földrengések megfigyelésekor pl. időbeli korreláció mutatható ki adott erősségű földrengések bekövetkezésének valószínűsége között, ami megmutatja, hogy a rendszer egy korábbi állapota milyen hatással van a jelenlegi, vagy a jövőbeli állapotára.



10. ábra Földmozgások időbeli korrelációja

A földrengések esetében megállapítható, hogy egy rengés hosszú idővel a lecsengése után is hatással van a rendszer állapotára. Ezzel analóg módon, a helyi időjárástól való függésen keresztül egy adott napi indulási feszültség is függ az előző napok, de akár hetek időjárásától (vagyis mért indulási feszültségeitől). Az egyes események nem tekinthetők függetlennek. Ha egy komplex rendszer viselkedését a frekvencia tartományban vizsgáljuk, akkor egy érdekes jelenséget figyelhetünk meg, az úgy nevezett 1/f zaj jelenséget. Ez az 1/f zaj jelenség számtalan (komplex) rendszerben megfigyelhető, például a tőzsdeindexek napi (évi) alakulásában. Ha további kutatásokat szeretnénk végezni a rendszer viselkedésével kapcsolatban, akkor ez a tudományterület és jelenség vizsgálata egy nagyon jó kiindulópontnak tekinthető az esetleges további összefüggések feltárásában.

Összességében elmondható, hogy amennyiben egymástól független események hatásai az idő múlásával kiegyenlítődnek, a rendszer minden egyes új esemény bekövetkezésével egy középérték irányába tolódik el. Vizsgálataim alapján, a számítási felhőben futó algoritmusok felhasználásával arra a következtetésre jutottam, hogy naperőművek indulási viselkedése nem írható le egymástól független események sorozataként. Egy 1/f karakterisztikájú spektrummal jellemzett (komplex) rendszer ilyen kiegyenlítődő tendenciát nem mutatna. Amennyiben a mi rendszerünk is egy ilyen komplex rendszernek tekinthető, akkor kijelenthetjük, hogy nem tudunk egy olyan megfelelő középértéket találni, ami akár hosszabb időre (pl. több 10 évre) vonatkozóan mindig optimális működést biztosítana.

Ahhoz, hogy egy valóban optimális működést elérjünk gyakori, folyamatos beavatkozás szükséges a rendszer működésébe. Erre a célra felhő alapú szolgáltatások igénybe vétele megfelelő lehet.

2.3 Javaslat egy felhőben futtatható vezérlési és egy szabályozási algoritmus megvalósítására

A fejezetben a korábban szerzett kutatás eredményeinek felhasználásával javaslatot próbálok tenni egy felhőben futtatható vezérlési és egy szabályozási algoritmus megvalósítására [16], melynek alkalmazásával a rendszer-hatásfok növelhető lenne. Az ismertetett eredmények tükrében körvonalazódni látszik, hogy a hatásfok rendszerszintű optimalizációjára sem egy egyszerű vezérlési, sem egy egyszerű szabályozási algoritmus nem található. A javasolt struktúra egy vezérlő és egy szabályzó tag ötvözete, melyben a vezérlő a szabályzó kimentére csatol elő, így rövidítve a beállási időn.

A vezérlési algoritmus megvalósításakor a megfigyelhető feszültség – hőmérséklet összefüggésből indultam ki. Az algoritmus alapja, hogy az időjárás előrejelzés adatokon keresztül és a múltban megszerzett ismeretek felhasználásával a másnapra előre jelzett hőmérséklet alapján egy becslést adjunk az indulási feszültség értékére. A becsléshez először kikeressük az adatbázisból a korábban már mért, ehhez a hőmérséklet értékhez tartozó indulási feszültségeket. Ezekhez a feszültség értékekhez tartoznak teljesítmény értékek is. A becslés során kiválasztjuk a legkisebb pozitív teljesítmény értéket a már korábban leszűkített adathalmazból és az új indulási feszültség érték az ehhez a teljesítmény értékhez tartozó feszültség érték lesz. Az algoritmus a meglévő adatokon nem változtat, a rossz előrejelzést nem bünteti, de a jó előrejelzés hatása a jövőben a legújabb értékek eltárolásán keresztül jelentkezik. Az algoritmus évszakok figyelembe vételével (például a múltba tekintés hosszának korlátozásán keresztül) tovább finomítható.

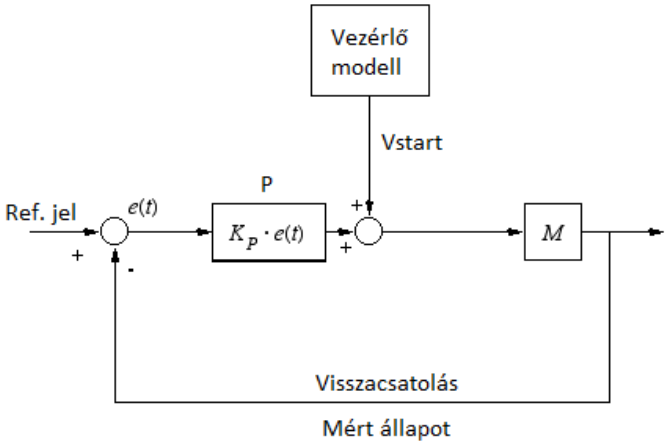
Az implementálás első lépéseként lekérdezzük a World Weather Online Api segítségével a holnap várható átlagos napi hőmérsékletet. A GET kérésre kapott JSON válasz formátuma ugyanaz, mint mikor a múltbeli adatokat kérdeztem le, így a hőmérséklet értéket ugyanazzal a metódussal kapom meg. Ezután az adatbázisból kikerestetem az előre jelzett hőmérséklethez tartozó feszültség értékeket.

Mivel nem csak erre lesz szükség, hanem a teljesítmény értékre is, így a már meglévő *StartData* objektum listával térek vissza, ami mindkettőt tartalmazza. Az objektum listán minimumkeresést elvégezve a teljesítmény értékek alapján határozható meg a vezérlő által becsült feszültség érték.

A javasolt struktúrában a szabályzó tag feladata ennek a mindenkori becsült értéknek a finomhangolása. A szabályozási algoritmus megvalósítása során is törekedtem a legalapvetőbb megoldást adó szabályzó kidolgozására. A megoldás tovább fejlesztése további kutatási témák alapjául szolgálhat. A szabályzó alapjele az indulási teljesítmény 0 értéke, hiszen energiamentes indulás a cél.

A szabályzó egy arányos tag. Adott technológia korlátok következtében a vezérlő és a szabályzó közös beavatkozó jelét egy indulási feszültség minimum és maximum értéke közé korlátozni szükséges.

Míg a vezérlés például az indulás előtt, például éjjel számolja újra az indulási feszültség értékét, addig a szabályzó hatása az indulás után, vagy nap közben frissül. A szabályzó így az előző napi hiba hatására a következő napra kiszámolt értékben avatkozik be. A vázolt struktúra lengésekre hajlamos, hiszen az időjárás kiszámíthatatlansága következtében az előző napi hibák alapján levont következtetések érvényessége megkérdőjelezhető. A szabályzó egyértelműen pozitív hatásának a kimutatása további kutatási téma alapját képezheti.



11. ábra Szabályzó blokkdiagram

Az előre csatolás nélküli, illetve a priori ismereteket teljes mértékben figyelmen kívül hagyó tisztán szabályzón alapuló megoldást feltételezve a szabályzó beállása nagyon lassú lenne. A beállított erősítés függvényében a beállási idő a 100 napos nagyságrendet is elérheti. Ha vesszük az előbb említett minimum és maximum értékek számtani közepét és a vezérlő hatása helyett ezzel inicializálunk, akkor már lényegesen jobb beállási időt kapnánk. De mivel rendelkezünk már korábban mért adatokkal és beláttuk a feszültség és a hőmérséklet közötti alapvető összefüggést, így minden adott ahhoz, hogy felhasználjuk ezeket az adatokat egy az adott körülményekre vonatkozóan viszonylag jó becslés kiszámításához. A vezérlési és a szabályozási algoritmus egyaránt Java nyelven implementálásra került és a felhőben is eredményesen futottak.

Mint láthatjuk, az optimalizáló algoritmusok kidolgozása során előnyösen felhasználtunk több, a felhő által kínált szolgáltatást. Ahogy már a bevezető részben is megállapítottam, a folyamatos adatgyűjtés segítségével a vállalatok lényeges versenyelőnyre tehetnek szert a hasonló termékpalettával rendelkező piaci versenytársaival szemben. Az optimalizáló algoritmus nagyon fontos eleme az adatgyűjtés, hiszen a segítségével a szabályzó beállási ideje jelentős mértékben csökkenthető.

Az elérhető számítási kapacitás sem korlátoz, hiszen az algoritmusok a felhőben futnak, ahol nagy teljesítményű virtuális gépek segítségével pillanatok alatt rendelkezésünkre állhat a szükséges információ. A kommunikáció kapcsolat létesítéséhez a végpontokat el kell látni egy http protokollon keresztül a felhővel kommunikáló interfésszel. Így az optimalizáló algoritmus bárhol, az üzembe helyezést követően, akár automatikusan is futtathatóvá válna, így növelve hosszútávon a rendszer hatásfokát.

3 Tervezési eljárások optimalizálása felhő segítségével

A dolgozat bevezető részében tárgyalt elvet egy konkrét példával megvilágítva: a megújuló energiatermelés elterjedése nagyszámú elosztott energiatermelő egységek erősáramú hálózatra csatlakozását vonták maguk után. Az ezekben a berendezésekben alkalmazott erősáramú szűrőkörök mérete adja a berendezések árának döntő hányadát. Üzemi adatok visszacsatolása nélkül, az alkalmazott worst-case tervezési eljárások folyománya ként könnyen alakulnak ki olyan termékcsaládok, melyek ugyan általánosan alkalmazhatóak, ám az esetek jelentős hányadában túlméretezettek, így feleslegesen drágák. Ugyanakkor offline szimulációs eszközök a naprakész tervezési eljárások szerves részeivé váltak. Véleményem szerint az alkalmazott offline szimulátorok online szimulátorként felhőben való futtatásával a vázolt problémakör – a túlméretezett eszközökön kiküszöbölésén keresztüli költségcsökkentés - hatékonyan támadható. Állításom szerint amennyiben a tervezési szakaszban használt szimulációs algoritmusok megfelelő változatait egy eszköz életciklusnak üzemi szakaszában a felhőben futtatjuk és az eredményeket a tervezési szakaszban felhasználjuk, költséghatékonyabb eszközöket tervezhetünk. Fontos, hogy ajánlásom szerint ne közelítő szimulációt, hanem a tervezési fázisban használt eljárások lehető legpontosabb replikációját implementáljuk. A felhő által az ehhez szükséges számítási kapacitás rendelkezésre áll. A javasolt eljárás szerint a számítási felhőben futó algoritmus periodikusan beolvas üzemi adatokat (gerjesztéseket és válaszokat), ezeket az adatokat a rendszer szimulációja számára bemeneti és kimeneti jelalakká alakítja. A szimulált viselkedés és a valós viselkedés közötti eltérés szabályzók automatikus hangolásának, a tervezési eljárások finomításának, rejtett összefüggések feltárásának, „Predicitve maintenance” eljárásoknak és túlméretezések feltárásának alapja lehet.

A motivációban említett, a felhőben futó szimuláció az időtartományban írja le a rendszer viselkedését, így átmeneti állapotok tetszőlegesen pontos (időbeni felbontású) vizsgálatát is lehetővé teszi. Tranziens viselkedés leírása szükséges, hiszen a jól bevált tervezési eljárások rendszerint állandósult állapotban érvényes alapfeltevésekből indulnak ki, ezekhez képest a tranziens folyamatok elemzése további hasznos információt szolgáltathat. Az általam javasolt költségcsökkentő hatású optimalizálás az egyik megközelítésben a felhőn keresztüli adatgyűjtésen alapuló, a folyamatok tranziens jellegét figyelembe vevő szimuláción alapul. Gyakran alkalmaznak ugyan szimulációs szoftvereket tranziens folyamatok vizsgálatára, ám a tervezési időben történő vizsgálatok igazán hatékony alkalmazása - a lehetséges esetek nagy száma miatt – a gyakorlatban nem lehetséges. Átmeneti állapotok leírása lényegében - általános esetben nemlineáris, változó együtthatós - differenciálegyenletek megoldását jelenti. Nemlineáris, változó együtthatós rendszerek leírására a felhő a rendelkezésre álló, skálázható számítási kapacitás következtében kimondottan alkalmas. Differenciálegyenletek numerikus megoldását, a megfelelő algoritmus kiválasztását kiterjedt irodalom segíti. A gyakorlati rendszerek többségét nem egy differenciálegyenlet, hanem differenciálegyenletek rendszere írja le.

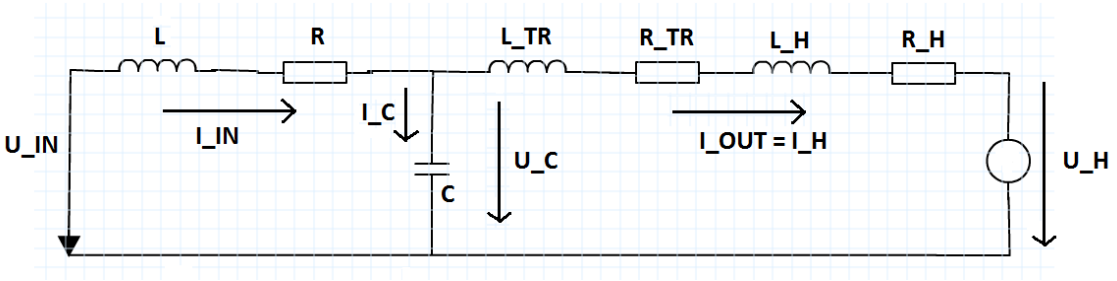
Ezek a differenciálegyenlet-rendszerek általában azonos, vagy a lényeget tekintve hasonló alakra hozhatók. Tekintsük például egy rugalmas tengellyel összekötött hajtógép és terhelés, valamint egy hálózatra csatlakozó AC/DC átalakító szűrőköre eseteket. A két egymástól távol álló szakterületről vett rendszerre a levezetés eredményeként kapott állapotteres leírás jól látható hasonlóságot mutat, így a folytonos/diszkrét időtartománybeli viselkedésük leírásához felhasználható matematikai apparátus is azonos.

3.1 Gyakorlati rendszerek leírása differenciál egyenletek segítségével

Ebben a fejezetben a korábban említetett példarendszerek (a rugalmas tengellyel összekötött hajtógép és terhelés, valamint egy hálózatra csatlakozó AC/DC átalakító szűrőköre) differenciálegyenletekkel történő leírását és az ehhez kapcsolódó levezetést szeretném bemutatni, így szemléltetve, hogy a rendszereket leíró egyenletek hasonló alakra hozhatók.

3.1.1 AC/DC átalakító szűrőköre

Az első részben egy AC/DC átalakító szűrőkörére vonatkozó rendszer leírást és levezetést szeretném bemutatni.



12. ábra AC/DC átalakító szűrőköre

A rendszer energiatárolói:

- C: szűrőkondenzátor
- L: inverter oldali induktivitás
- L_TR: az inverter és a hálózat közti csatoló transzformátor induktivitása
- L_H: az erősáramú hálózat induktivitása

Kirchhoff huroktörvénye alapján (feszültségek előjeles összege = 0) az alábbi összefüggések írhatók fel (1):

$$U_{IN} - U_C = I_{IN} * R + L * \frac{dI_{IN}}{dt}$$

$$\frac{dI_{IN}}{dt} = \frac{U_{IN} - U_C - R * I_{IN}}{L}$$

Kirchhoff csomóponti törvénye alapján az alábbi egyenlet vezethető le (2):

$$I_{in} - I_C = I_H$$

$$I_C = I_{IN} - I_H$$

$$C * \frac{dU_C}{dt} = I_{IN} - I_H$$

$$\frac{dU_C}{dt} = \frac{I_{IN} - I_H}{C}$$

Kirchhoff huroktörvénye felhasználásával a harmadik differenciálegyenlet (3):

$$U_C - U_H = L_{TR} * \frac{dI_{OUT}}{dt} + R_{TR} * I_{OUT} + L_H * \frac{dI_{OUT}}{dt} + R_H * I_{OUT}$$

$$U_C - U_H = (L_{TR} + L_H) * \frac{dI_{OUT}}{dt} + (R_{TR} + R_H) * I_{OUT}$$

$$\frac{dI_{OUT}}{dt} = \frac{U_C - U_H + (R_{TR} + R_H) * I_{OUT}}{L_{TR} + L_H}$$

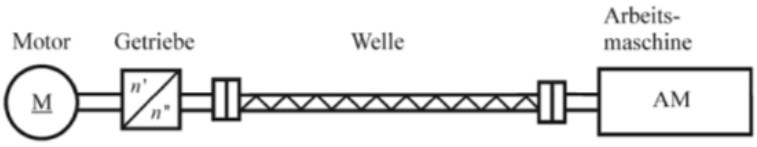
A három differenciálegyenletet az alábbi mátrixos alakban is felírható az alatta olvasható általános alak alapján:

$$\begin{bmatrix} \dot{I_{IN}} \\ \dot{U_C} \\ \dot{I_{OUT}} \end{bmatrix} = \begin{bmatrix} -R/L & -1/L & 0 \\ 1/C & 0 & -1/C \\ 0 & 1/(L_{TR} + L_{H)} & -(R_{TR} + R_H)/(L_{TR} + L_H) \end{bmatrix} * \begin{bmatrix} I_{IN} \\ U_C \\ I_{OUT} \end{bmatrix} + \begin{bmatrix} 1/L & 0 \\ 0 & 0 \\ 0 & 1/(L_{TR} + L_H) \end{bmatrix} * \begin{bmatrix} U_{IN} \\ U_H \end{bmatrix}$$

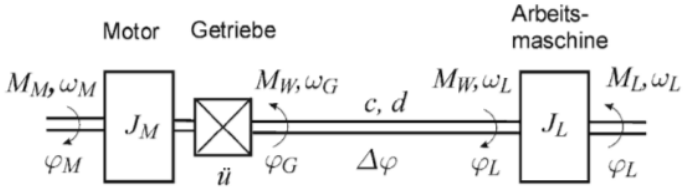
$$\dot{\bar{x}} = \bar{\bar{A}} * \bar{x} + \bar{\bar{B}} * \bar{u}$$

3.1.2 Rugalmas tengellyel összekötött hajtógép és terhelés

Ebben a fejezetben a második példarendszer, egy rugalmas tengellyel összekötött hajtógép és terhelőgép esetére felírható differenciálegyenlet-rendszer levezetését szeretném ismertetni [17].



13. ábra Rugalmas tengellyel összekötött hajtógép és terhelés



14. ábra Motor- tengely - gép rendszer

A rendszer energiatárolói:

- Rugalmas tengely, mint rugó
- Motor tehetetlenségi nyomaték (J_M)
- Terhelőgép tehetetlenségi nyomaték (J_L)

A tengelyre ható M_W nyomaték hatására a rugalmas tengely megcsavarodik.

M_W hatását az áttétel motor oldalán M'_W -vel jelöljük és az alábbi képlettel számolhatjuk ($\ddot{u}$ az áttétel):

$$M'_W = \frac{M_W}{\ddot{u}}$$

$$\ddot{u} = \frac{\omega_M}{\omega_G}$$

Az irodalomban fellelhető Voigt – Kelvin modell alapján:

$$M_W = c * \varphi + d * \dot{\varphi}$$

A motorra nyomatékának figyelembe vételével:

$$M_M - M'_W = J_M * \frac{d\omega_M}{dt} = M_M - \frac{M_W}{\ddot{u}} = M_M - \frac{c * \varphi + d * \dot{\varphi}}{\ddot{u}}$$

A terhelést gyorsító nyomaték $M_W - M_L$ pedig:

$$M_W - M_L = J_L * \frac{d\omega_L}{dt} = c * \varphi + d * \dot{\varphi} - M_L$$

A rugalmas tengelyre felírható szögkülönbségekből:

$$\varphi = \varphi_G - \varphi_L$$

$$\varphi = \varphi_G - \varphi_L = \omega_G - \omega_L = \frac{\omega_M}{\ddot{u}} - \omega_L$$

A fenti összefüggések felhasználásával a rendszerre felírható állapotegyenlet:

$$\dot{\omega}_M = \frac{1}{J_M} * (M_M - \frac{c}{\ddot{u}} * \varphi - \frac{d}{\ddot{u}} * (\frac{\omega_M}{\ddot{u}} - \omega_L))$$

$$\dot{\omega}_L = \frac{1}{J_L} * (c * \varphi + d * \left(\frac{\omega_M}{\ddot{u}} - \ \omega_L\right) - M_L)$$

$$\dot{\varphi} = \frac{\omega_M}{\ddot{u}} - \omega_L$$

Ezt a három differenciálegyenletet ebben az esetben is felírható mátrixos alakban:

$$\begin{bmatrix} \dot{\omega_M} \\ \dot{\varphi} \\ \dot{\omega_L} \end{bmatrix} = \begin{bmatrix} -d/J_M & -c/J_M & d/J_M \\ 1 & 0 & -1 \\ d/J_L & c/J_L & -d/J_L \end{bmatrix} * \begin{bmatrix} \omega_M \\ \varphi \\ \omega_L \end{bmatrix} + \begin{bmatrix} 1/J_M & 0 \\ 0 & 0 \\ 0 & -1/J_L \end{bmatrix} * \begin{bmatrix} M_M \\ M_L \end{bmatrix}$$

$$\dot{\bar{x}} = \bar{\bar{A}} * \bar{x} + \bar{\bar{B}} * \bar{u}$$

A bemutatott két példa alapján szemléletes, hogy egy általános kialakítású, a felhőben futó, lineáris, állandó együtthatós differenciálegyenlettel leírható rendszer viselkedését modellező szimulátor széles körben, szakterületeken átívelően alkalmazható. Egy ilyen szimulátor segítségével, valamint egy előre kialakított interfészen keresztül begyűjtött adatokon keresztül tetszőleges rendszer viselkedését megfigyelhetjük, reprodukálhatjuk és elemezhetjük.

3.2 Időtartománybeli szimulációs eljárások implementálása

Ahhoz, hogy a szimulációs környezet váza és a szimulátor működése könnyebben érthető legyen, ismertetem a gyakorlatban alkalmazott időtartománybeli szimulációs eljárások elméleti hátterét képező numerikus módszereket.

Két lehetséges numerikus módszert vizsgáltam:

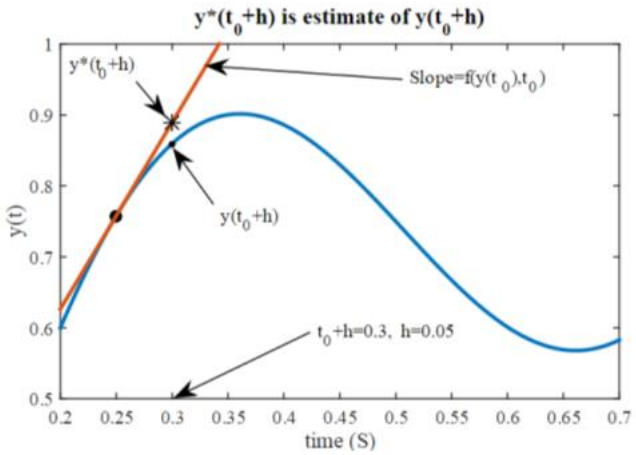
- Runge-Kutta alapú numerikus differenciálegyenlet megoldó algoritmust, és
- a rendszer diszkrét idejű állapotteres leírásából kiinduló állapot átviteli mátrixot.

Az egyes megoldási algoritmusok közti különbség az elérhető pontosságban és a futási időben rejlik.

3.2.1 Runge-Kutta módszerek

Ebben a fejezetben a Runge-Kutta alapú módszer elméleti háttérét[18] szeretném röviden összefoglalni és hogy a gyakorlatban ennek a módszernek a segítségével hogyan is oldható meg egy adott differenciálegyenlet[19].

Differenciálegyenletek megoldására többféle módszer létezik, az egyik ilyen csoport az úgy nevezetett Runge-Kutta típusú módszerek. Előnyük, hogy nem igénylik a függvény magasabb deriváltjainak számítását, pontossága a lépésköz (az eljárás alkalmazása során „*h*” lépésközzel haladunk előre az időben) változtatásával szabályozható. Az alapelképzelés Euler közelítésen nyugszik, ami pontatlanságából kifolyólag a gyakorlati esetek többségében közvetlenül nem alkalmazható. Az R-K módszerek családjába tartozó eljárások lényege, hogy a keresett függvényre az adott pontban vett meredekség, vagyis a derivált értékének segítségével kapunk közelítő megoldást. Az éppen aktuális pontra vonatkozó érték számításához az előző pontokban becsült értékeket használjuk fel.



15. ábra Euler approximáció egy példán keresztül

Első lépésként elsőrendű Runge-Kutta algoritmus segítségével próbáltam becslést találni egy általános függvény adott pontban vett deriváltjának értékére. Az eljárás során az adott y(t) függvény Taylor sorának első tagjából indulunk ki. A soron következő taktusokban megismételjük a folyamatot az előző taktusban kapott becsült érték (*y*(h)*) felhasználásával Összefoglalva, az alábbi algoritmust addig ismételjük, míg el nem érünk egy adott végpontba:

- a meredekség becslője

$$k_1 = f(y^*(t_0), t_0)$$

- a megoldás becslője a következő időpontban

$$y^*(t_0 + h) = y^*(t_0) + k_1 * h$$

Egy elsőrendű Runge-Kutta módszer a gyakorlati esetek többségében továbbra is elfogadhatatlan hibával dolgozik. Az egyik kézenfekvő módszer ennek a hibának a csökkentésére a *t=0* és a *t=h* között félúton lévő pontban (k_2) vett derivált értékének felhasználása. Az elsőrendű Runge-Kutta módszert használjuk fel, a függvény *y(½h)* – ban vett értékének előállítására. Ezt a becslést használjuk k_2 értékének előállítására (ami az intervallum középpontjában a meredekség közelítő értéke lesz). Ezután már felhasználhatjuk k_2-t, hogy *y*(h)* értékének meghatározására. A leírtak alapján az alábbi algoritmus írható fel a másodrendű Runge-Kutta módszerre vonatkozóan:

- a meredekség becslője

$$k_1 = f(y^*(t_0), t_0)$$

- a függvény értékének becslője t_0+h/2 pontban

$$y\left(t_0 + \frac{h}{2}\right) = y^*(t_0) + k_1 \frac{h}{2}$$

- a meredekség becslője t_0+h/2 pontban

$$k_2 = f\left(y_1\left(t_0 + \frac{h}{2}\right), t_0 + \frac{h}{2}\right)$$

- a megoldás becslője a következő időpontban

$$y^*(t_0 + h) = y^*(t_0) + k_2 * h$$

Az előző bekezdésben bemutatott eljárással, amennyiben két becslést használunk a meredekségre, akkor a közelítés egy pontosabb eredmény ad, mint ha csak egy meredekséggel dolgoznánk. Ezek után magától értetődő, hogy ha minél több becslést használunk a meredekségre, annál pontosabb lesz az eredmény. Ez a feltevés további magasabb rendű Runge-Kutta módszerekhez vezet.

A negyedrendű Runge-Kutta módszer algoritmusa a másodrendű módszer kiterjesztéseként:

- felhasználja a k_1 meredekséget, hogy kiszámolja a félúton vett meredekséget, akkor megkapjuk k_2-t is
- felhasználja a k_2 meredekséget, hogy kiszámolja a félúton vett meredekséget, akkor k_3 egy másik becslése lesz a félúton vett meredekségnek
- végső soron pedig felhasználja k_3-t, hogy végiglépkedjen az időintervallumon (t_0+h-ig), és k_4 egy becslés lesz a végpontban vett meredekségre

A meredekségek súlyozott összegét véve kapjuk meg a végső becslésünket $y^*(t_0+h)$ értékére. Az analitikus megoldás numerikus megfelelőjét a bemutatott számítás teljes szimulálni kívánt időtartamra történő kiterjesztésével kapjuk.

A hivatkozott oldalon[19] megtalálható ezeknek az algoritmusoknak a részletesen magyarázata, illetve a függelékben megtalálható egy példa differenciálegyenlet Runge-Kutta megoldásainak Java nyelven történő implementációjára is.

3.2.1.1 Algoritmusok felhőben futtatása

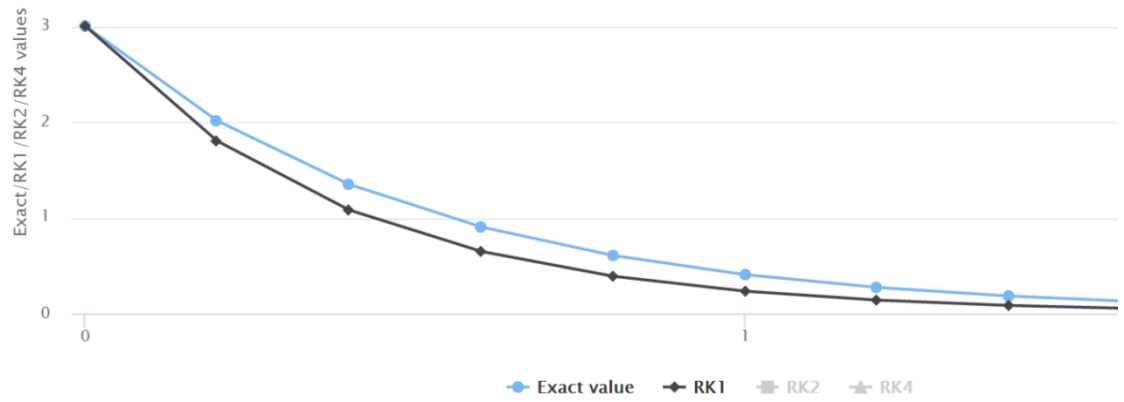
Az előbb említett példa differenciálegyenlet Runge-Kutta megoldását felhőben is lefuttattam. Az alábbi táblázatban összegzem az eredményeket, amelyeket a 0 és 2 közötti intervallumban, $h = 0.2$ lépésközzel kaptam meg. Az első oszlopban láthatók a példa differenciálegyenlet analitikus megoldásának eredményei, melyet az ismert függvénybe t értékeinek behelyettesítésével kaptunk. A tőle jobbra fekvő oszlopokban különböző rendszámú Runge-Kutta módszerek futtatásával kapott eredmények láthatók.

Runge-Kutta results

t	exact	rk1	rk2	rk4
0	3	3	3	3
0.2	2.0109601381069178	1.7999999999999998	2.04	2.0112
0.4	1.3479868923516647	1.0799999999999998	1.3872	1.34830848
0.6000000000000001	0.9035826357366061	0.6479999999999999	0.9432959999999999	0.903906004992
0.8	0.6056895539839662	0.3887999999999999	0.64144128	0.6059785857466369
1	0.4060058497098381	0.23327999999999993	0.4361800704	0.4062480438845454
1.2000000000000002	0.27215385986823754	0.13996799999999995	0.29660244787199996	0.27234868862019923
1.4000000000000001	0.1824301878756539	0.08398079999999997	0.20168966455295997	0.18258256085098157
1.6	0.12228661193509868	0.05038847999999998	0.13714897189601277	0.12240334879449805
1.8	0.0819711673418777	0.030233087999999984	0.09326130088928869	0.08205920503183148
2	0.054946916666202564	0.018139852799999988	0.0634176846047163	0.055012491053339815

16. ábra Runge-Kutta módszerek futásának eredménye

Az alábbi ábrákon az egyes oszlopok értékei láthatók grafikonként ábrázolva. A kékkel ábrázolt görbe az analitikus megoldás eredménye az adott intervallumban. Az első ábrán az elsőrendű Runge-Kutta módszer alkalmazásával kapott eredmények láthatók. Mint ismeretes, az ennél a módszernél jelentkező viszonylag nagy hiba a grafikonon is jól látható.

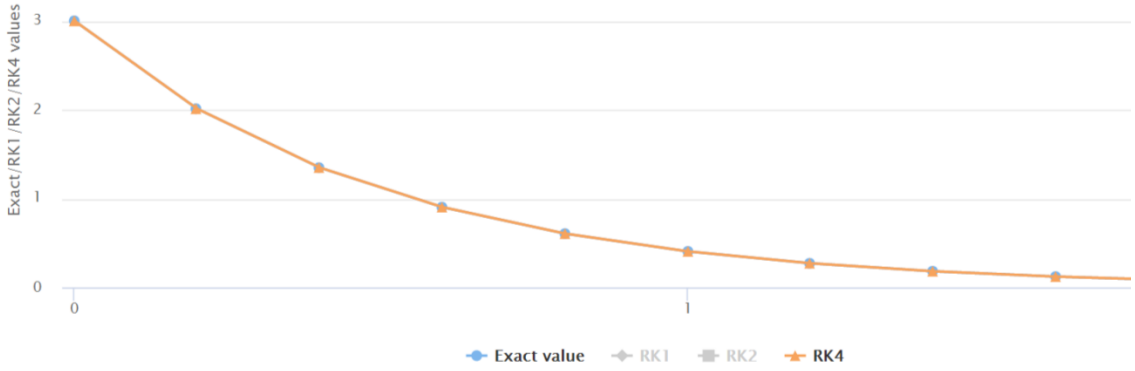


17. ábra Elsőrendű RK módszer eredményének ábrázolása

A második ábrán a másodrendű Runge-Kutta módszer eredményeit láthatjuk. Itt már jelentősen csökken a hiba mértéke, ami az előző ábrával összehasonlítva szembetűnő.

18. ábra Másodrendű RK módszer eredményének ábrázolása

Alább a negyedrendű Runge-Kutta algoritmus futtatásával kapott eredményeket láthatjuk. A grafikonon jól látszik, hogy a két görbe szinte egybeesik. Mint ismeretes, a negyedrendű algoritmusnak felhasználásával már egy jó becslést kapunk a megoldásra nézve.



19. ábra Negyedrendű RK módszer eredményének ábrázolása

3.2.2 Állapot átviteli mátrix alapú megoldások

Egy másik, időtartományban történő szimuláció alapját képező matematikai módszer a rendszer diszkrét idejű állapotteres leírásában szereplő állapot átviteli mátrixot felhasználó eljárás. Mint már korábban is láthattuk a gyakorlati rendszerek többségét differenciálegyenlet rendszerek segítségével írhatjuk le. Szabályozástechnikai, valamint jel-és rendszerelméleti ismereteink alapján folytonos időtartományban az alábbi rendszeregyenlet írható fel:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t), \qquad x(0) = x_0$$

$$y(t) = \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t)$$

Ismert matematikai összefüggéseket alkalmazva[20] a kimenetet kifejezve az alábbi egyenletet írhatjuk fel:

$$\boldsymbol{y}(t) = \boldsymbol{C}e^{\boldsymbol{A}t}\boldsymbol{x}_0 + \int_0^t \boldsymbol{C}e^{\boldsymbol{A}(t-\tau)}\boldsymbol{B}\boldsymbol{u}(\tau)d\tau + \boldsymbol{D}\boldsymbol{u}(t)$$

Mivel ezt az eljárást is a felhőben futtatva kerül alkalmazásra, a folytonos időtartománybeli modellből áttérnünk a modell diszkrét időtartománybeli leírására:

$$\boldsymbol{x}(k+1) = \boldsymbol{A}_d\boldsymbol{x}(k) + \boldsymbol{b}_d u(k), \qquad x(0) = x_0$$

$$y(k) = \boldsymbol{c}^T\boldsymbol{x}(k) + du(k)$$

A diszkrét időtartománybeli modell paramétereit a következő összefüggések segítségével kaphatjuk meg:

$$\boldsymbol{A}_d = e^{\boldsymbol{A}T}, \qquad \boldsymbol{b}_d = \int_0^T e^{\boldsymbol{A}\alpha}d\alpha\boldsymbol{b}$$

$$\boldsymbol{c}_d^T = \boldsymbol{c}^T, \qquad d_d = d$$

A későbbiek ismertetett szimulációs algoritmus futtatása során az $\boldsymbol{A}$ és $\boldsymbol{B}$ mátrix értékeit használjuk fel az egyenletrendszer megoldásához. Ezeknek a mátrixoknak az értékeit a rendszer paraméterei (energiatárolók, veszteségek) és a mintavételi idő határozzák meg. Hagyományos tervezési eljárások során ezeknek a paramétereknek fix értékeket adnak, ezekkel szimulálnak, így ellenőrzik a rendszer várható működését. A választott értékeket például szabványok, vagy a priori ismeretek alapján határozzák meg a szakemberek. Ezeket a paramétereket felhasználva történik a mátrixok értékeinek offline számítása. A szakemberek a tervezési fázis alatt a rendelkezésre álló szimulációs algoritmus és a paraméterek így becsült értékének felhasználásával törekednek egy általánosan alkalmazható megoldás kidolgozására. A mai globalizált világban az így tervezett berendezések egymástól gyökeresen eltérő üzemi körülmények között működnek. Egy energia átalakító eszköz ugyan azon a paraméterek alapján tervezve üzemel a szomszédaitól teljesen elszigetelt Koreai Köztársaság villamos energia rendszerére csatlakozva, és például az Európai Unió hatalmas kiterjedésű, kooperatív energiarendszerében. Ugyan azon komponensek integrálódnak minden elkészült és legyártott termékbe. Egy ilyen globálisan alkalmazható megoldás rendszerint tág, worst-case tervezési elvet igényel, mely nem feltétlenül jelenti minden esetben költséghatékony megoldást.

Az állapot átviteli- és a rendszer bemeneti mátrixok elemeinek offline számítása esetén a diszkrét idejű állapotegyenlet megoldása kevesebb számításai kapacitás lekötését igényli az R-K algoritmusokkal összehasonlítva, de a szimuláció ebben a formában – az offline számítás megismétlése nélkül – a rendszerelemek csak egy bizonyos előre meghatározott értékeivel és állandó szimulációs lépésközzel végezhető el. Egy lehetséges szimulációs ciklus így a következő lépésekből áll:

- a mátrixok elemeinek offline számítása,
- szimuláció,
- az eredmények kiértékelése,
- a mátrixok elemeinek offline újraszámítása,
- a fenti lépések ismétlése egy kritérium eléréséig.

Ezzel szemben a Runge-Kutta módszerek az időben változó rendszerelem-értékekkel és változó szimulációs lépésközzel dolgozhatnak, így értékük módosítása a szimuláció során is megtörténhet.

Az állapotteres leíráson alapuló megoldás ismertetett hátránya a mátrixelemek online újraszámításával kiküszöbölhető. Ehhez az állapot átviteli mátrix exponenciális függvényének számítása szükséges[21], mely számára a szükséges számítási kapacitás a felhőben adott költségszinten elérhető. Online újraszámítás esetén az üzembe helyezéskor a helyszínen érvényes paraméterek figyelembe vételével a mátrix értékei frissíthetővé válnak, így az adott körülményekhez mérten egy az előre meghatározott fix paraméterekhez képest optimálisabb működés is elérhető lehet.

3.3 Szimulációs környezet vázának elkészítése

A dolgozatomnak ebben a szakaszában egy olyan szimulációs környezet vázát szeretném bemutatni, mely a jelenleg széles körben alkalmazott tervezési eljárások reformjának, az ezen keresztüli költségcsökkentésnek felhő alapú motorja lehet. Élve az online visszacsatolás, beavatkozás és a felhő alapú technológiában rejlő számítási kapacitás adta lehetőségekkel az erőforrás pazarló tervezési eljárások és a felhasználásukkal tervezett egységek kiszűrhetők, ezáltal anyagi és marketing szempontból egyaránt kiaknázható versenyelőnyt biztosítva az alkalmazó vállalatok számára.

A szimulációs környezet, amit a továbbiakban részleteznék egy hálózatra csatlakozó AC/DC átalakító szűrőkörének a viselkedését szimulálja. A célunk tehát a szimulációs eljárások felhőben futtatása, aminek a segítségével akár üzembe helyezéskor, akár a rendszer későbbi működése során egy optimális és költséghatékony működés érhető el.

A szimulációs algoritmus Java nyelven történő implementálásához a rendelkezésemre állt a szimuláció C nyelven megírt kódrészlete.

Elsőként az állapot átviteli mátrix alapú eljárást mutatnám be.

A C nyelven megírt szimulációs program esetében az egyes mátrix értékek MATLAB segítségével (offline) kerültek számításra, így a mátrixok elemei a kódrészletbe beégetve jelennek meg. Az algoritmusban található függvény az állapotváltozók korábbi értékeinek, a gerjesztések értékeinek és a mátrix elemeinek felhasználásával a már korábbi fejezetben taglalt differenciálegyenlet rendszert oldja meg.

A mátrixok elemeinek újraszámításához a mátrixok exponenciális függvényének számítása szükséges, ami csak négyzetes mátrixok esetén végezhető el. A mátrixműveletek elvégzéshez a *Jblas*[22] illetve a *Jeigen*[23] nevű Java külső könyvtárat használtam, amiket kifejezetten lineáris algebrai műveletek elvégzéséhez készítettek. Az alábbi kódrészletben egy mátrix skalárral való szorzását végeztem el:

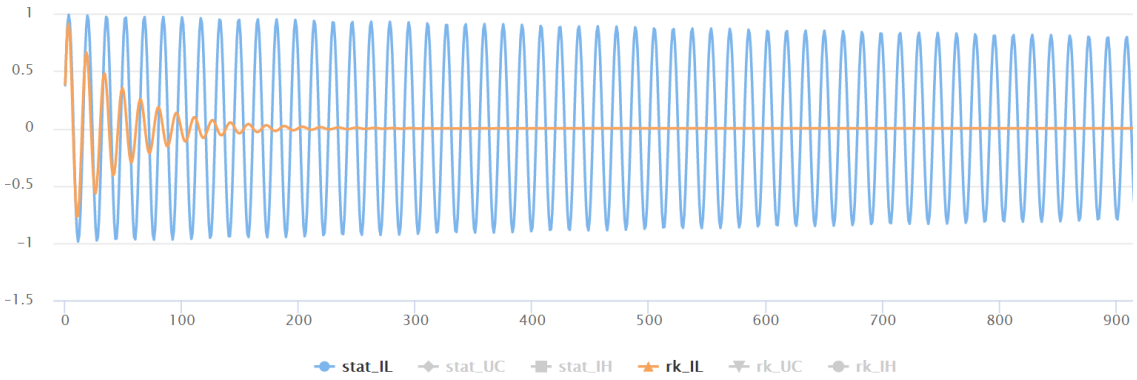
```
double t = 1.0/f_samp;
DoubleMatrix A = new DoubleMatrix(a_cont);
DoubleMatrix B = new DoubleMatrix(b_cont);
DoubleMatrix A_t = A.mul(t);
DoubleMatrix B_t = B.mul(t);
```

Mivel a B mátrixunk egy 2x3-as oszlopmátrix, így nem számolhatjuk az exponenciális függvényét. MATLAB környezetben c2d nevű függvény [24] segítségével történik a folytonos időből diszkrét időbe történő áttérés. A függvény implementációját tanulmányozva a rendszermátrixot és a bemeneti mátrixot négyzetes mátrixszá alakítom, majd ennek a mátrixnak veszem az exponenciális függvényét. Az eredményből le kell vennünk az A és B mátrixnak megfelelő almátrixokat és meg is kaptunk az új értékeket.

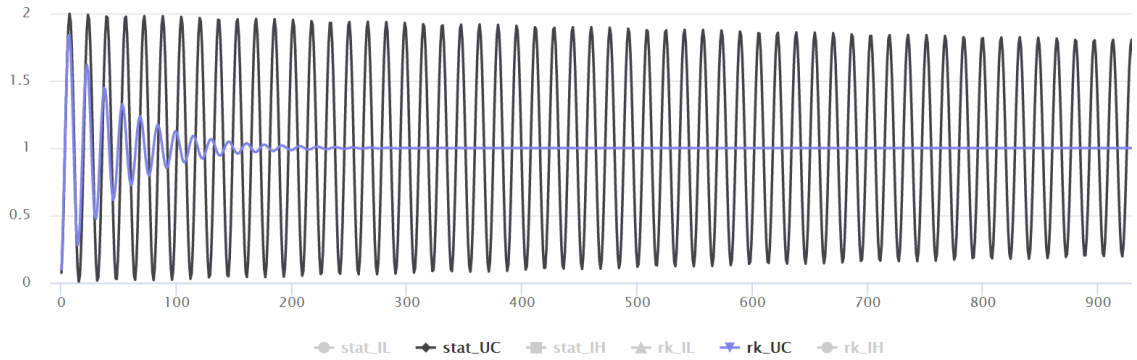
A Runge-Kutta alapú megoldások közül a másodrendű és a harmadrendű Runge-Kutta módszerek kerültek implementálásra. A paraméterek, amiket a szimulációhoz használunk, megfelelnek az állapot átviteli mátrix esetén használt paramétereknek. Ezen módszer esetén a lépésköz és a paraméter értékei is online változtathatóak.

A szimulációt a paraméterek megadásával a felhőben is futtathatjuk. Lehetőségünk van az állapot átviteli mátrixos módszer esetén a mátrixok újraszámítására és a gerjesztő jelek megadására. A Runge-Kutta módszer esetén lehetőségünk van a lépésköz és a szimulációs paraméterek értékeinek változtatására is. A szimuláció eredményeit egy táblázatban tekinthetjük meg és hasonlíthatjuk össze a már meglévő eredményeikkel. A kapott eredményeket akár adatbázisba is lementhetjük, így összehasonlíthatjuk a már korábban kapott eredményekkel.

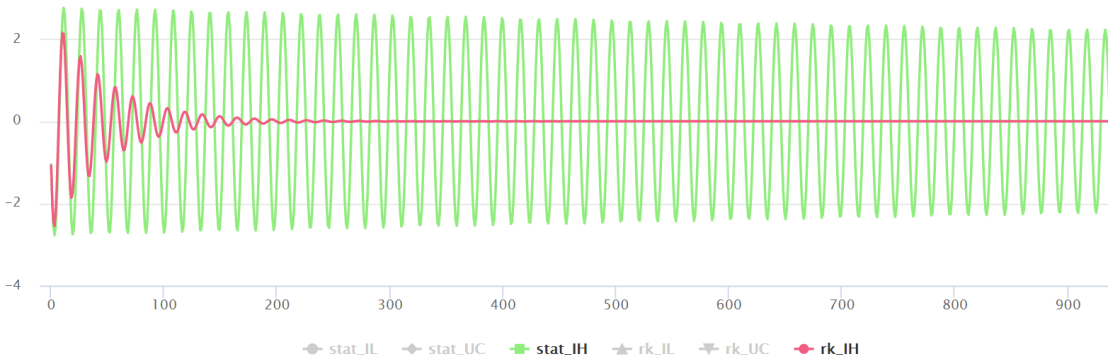
Az következő grafikonokon egy szimulációs futás eredményeit láthatjuk:



20. ábra I_L állapotváltozó értékei



21. ábra U_C állapotváltozó értékei



22. ábra I_H állapot változó értékei

A szimuláció során megadtuk a rendszer paraméter értékeit, gerjesztésnek pedig mindkét jel értékét konstans egyre állítottuk. Azt várnánk a rendszertől, hogy az idő múlásával az állapotváltozók egy állandósult állapotot vesznek fel. A grafikonokon látszik, hogy ez meg is történik.

4 Összefoglalás

Az utolsó fejezetben szeretném összefoglalni a dolgozat megírása és a rendszer elkészítése közben szerzett eredményeket és tapasztalatokat. A második alfejezetben kitérnék még a rendszer adta továbbfejlesztési lehetőségekre is.

4.1 Eredmények, tapasztalatok

A projekt során láthattuk, hogy a különböző felhő alapú megoldások hogyan hasznosíthatóak az egyes termékek fejlesztési életciklusa során. A jól elkülönülő fejezetekben és alfejezetekben kitértem két nagy alkalmazási területre, és egy-egy példán keresztül bemutattam a felhő alapú szolgáltatások használatának előnyeit az adott szakterületen. A dolgozatom írása során törekedtem arra, hogy az elméletben szerzett tudást a gyakorlatba is átültessem, és így a gyakorlatban szerzett eredményeket az adott területnek megfelelően hasznosítsam.

A bevezető részben részletesen kifejtettem, hogy a felhő alapú szolgáltatások milyen lehetőségeket nyitnának meg a kis –és közepes vállalatok, valamint az egyes iparágak számára a termék fejlesztési ételciklus során. Összehasonlítottam a manapság népszerű és leggyakrabban használt felhő szolgáltatókat és bemutattam az általam elképzelt rendszer felépítését és működését.

A második részben a rendszer hatásfok növelése volt a kitűzött cél, vagyis hogy az adott felhő alapú szolgáltatásokat hogyan hasznosíthatnánk ezen a területen. Összegzésképp elmondható, hogy az adatgyűjtés jelenti a legfontosabb részét a folyamatnak. Felhőben „tárolt” adatbázis segítségével több éves üzemadatok alapján hatékony adatanalízist végezhetünk és a rendelkezésre álló információ és a belőlük levont következtetések alapján különböző optimalizáló algoritmusokat is futtathatunk a felhő segítségével, amivel egy hosszú távú, optimalizált működés biztosítható.

A harmadik részben először egy elméleti betekintést szerettem volna nyújtani a különböző időtartománybeli szimulációs eljárások módszereibe. Láthattuk, hogy a gyakorlatban előforduló rendszerek többsége leírható differenciálegyenlet rendszerek segítségével, és ezek az egyenletek az ismertetett elméleti módszerek felhőben való futtatásának segítségével szintén megoldhatóvá válnak. Elkészült a bemutatott rendszer szimulációs környezetének váza is, melynek alkalmazásával a hagyományos tervezésből adódó rejtett többletköltségek radikálisan csökkenthetővé válhatnának.

A dolgozat elkészítése során betekintést nyerhettem más tudományterületek világába is, és láthattam, hogy az informatika nagymértékben segítheti az elméleti tudás gyakorlatba való átültetését és ez által a különböző iparágak fejlődését is. A felhő alapú technológiák megjelenése pedig további lehetőségeket nyitott a vállalatok és az ipar számára a különböző tervezési eljárásaik megreformálására is.

4.2 Továbbfejlesztési lehetőségek

A továbbfejlesztés lehetősége az összes bemutatott alkalmazási területen adott, sőt még ki is terjeszthető ebben a dolgozatban nem részletezett területeken való kutatással is.

A rendszer hatásfok optimalizálás területén további részletekbe menő adatgyűjtés is lehetséges, mint például az időjárás adatok óránkénti begyűjtése, valamint további optimalizálási algoritmusok kidolgozása sem elképzelhetetlen. A fejezet végén ismertetett komplex rendszerek tudományterülete is további kutatási irányokat kínál a témával kapcsolatban.

További szimulációs eljárások felhőben való futtatása is lehetővé válik, valamint a már meglévőket közvetetten is összekapcsolhatjuk a rendszerrel, hogy valós időben próbálhassuk ki őket - akár az üzembe helyezési vagy telepítési fázis során-, és dolgozhassuk fel az eredményeiket.

Felhő alapú technológiát nem csupán ezeken a területen lehetne hasznosítani a termék fejlesztési folyamat során. Sok vállalat életében előfordult már, hogy a fejlesztési ciklus során más vállalatokkal is együtt kellett működnie bizonyos feladatok elvégzése során, mint például tesztek elvégzése, szimulációs eredmények összevetése, vagy bizonyos rendszer komponensek összehangolása esetén. Előfordulhat az is, hogy egy hazai vállalat, mint például a Hyundai Technologies Center Hungary Kft. is szoros együttműködésben áll a külföldi anyavállalattal. Ezekben az esetekben nagyon nehéz összehangolni az együttműködés és a közös munka menetét. Sok esetben, ahhoz hogy az eredményeket megfelelő minőségben össze lehessen vetni, meg kell tanulnia és be kell szereznie az adott vállalatnak egy teljesen más fejlesztőkörnyezetet, ami időt, pénzt is sok energiát is felemészthet. Ebben a helyzetben is nagyszerű lehetőség lehet egy felhő alapú platform és környezet kidolgozása a szereplőknek, melynek segítségével nincs szükség más fejlesztői környezethez való alkalmazkodáshoz, hanem egy közös megegyezésen alapú platform segítségével egységes kiértékelési lehetőséget kapnak az eredményeikhez.

A napelemes rendszer esetén korábban már elkészült egy Android alapú alkalmazás is. Ezt az alkalmazást is tovább lehetne bővíteni egy interfésszel, aminek a segítségével kommunikálnánk a felhővel, és szintén bekapcsolódhatna egy kibővített rendszer koncepcióba. A hajtásrendszer projekt során pedig egy PC-n futó szoftver is készül, ezt a szoftver is beépíthetnénk a rendszerbe további adatgyűjtés vagy az üzembe helyezés segítése céljából.

Irodalomjegyzék

[1] Selmar Allenhof : *Implementation of an example application facilitating Industrie 4.0*

[2] *Business understanding, challenges and issues of Big Data Analytics for the servitization of a capital equipment manufacturer (2016. augusztus)*, http://ieeexplore.ieee.org/document/7363897/

[3] Design Principles for Industrie 4.0 Scenarios (2016. augusztus), http://ieeexplore.ieee.org/document/7427673

[4] Felhő szolgáltatások összehasonlítása (2016. július), http://cloudacademy.com/blog/public-cloud-war-aws-vs-azure-vs-google

[5] Mark C. Chu-Carroll : *Code in the Cloud* 10.-19.

[6] Apache Tomcat (2016. május), http://tomcat.apache.org/

[7] Heroku (2016. május), https://www.heroku.com/about

[8] MongoDb (2016. május), https://docs.mongodb.org/getting-started/java/query/

[9] Angular2 (2016. május), https://angular.io/docs/ts/latest/guide/server-communication.html

[10] Bootstrap template (2016. június), http://startbootstrap.com/template-categories/all/

[11] Cloud Computing for Power System Simulations at ISO New England—Experiences and Challenges (2016. augusztus), http://ieeexplore.ieee.org/document/7471484/

[12] Korreláció számítás (2016. augusztus), http://www.die-wege.de/mod/book/view.php?id=250&chapterid=401

[13] Korrelation (2016. augusztus), https://www.uni-koblenz.de/~physik/informatik/DSV/Faltung.pdf

[14] U. Karrenberg: *Signale - Prozesse – Systeme*

[15] Frank-Michael Dittes: *Komplexität, Warum die Bahn nie pünktlich ist*

[16] Philipp K. Janert : *Feedback Control for Computer Systems*

[17] Rolf Müller : *Modellierung, Analyse und Simulation elektrischer und mechanischer Systeme mit Maple und MapleSim*

[18] Benkó Sándor: *Számítógépes módszerek az erősáramú elektrotechnikában*

[19] Approximation of Differential Equations by Numerical Integration (2016. szeptember), http://lpsa.swarthmore.edu/NumInt/NumIntIntro.html

[20] Jan Lunze: *Regelungstechnik 2: Mehrgrößensysteme*

[21] Leslie Hogben: *Handbook of Linear Algebra*

[22] Jblas (2016. szeptember), http://jblas.org/

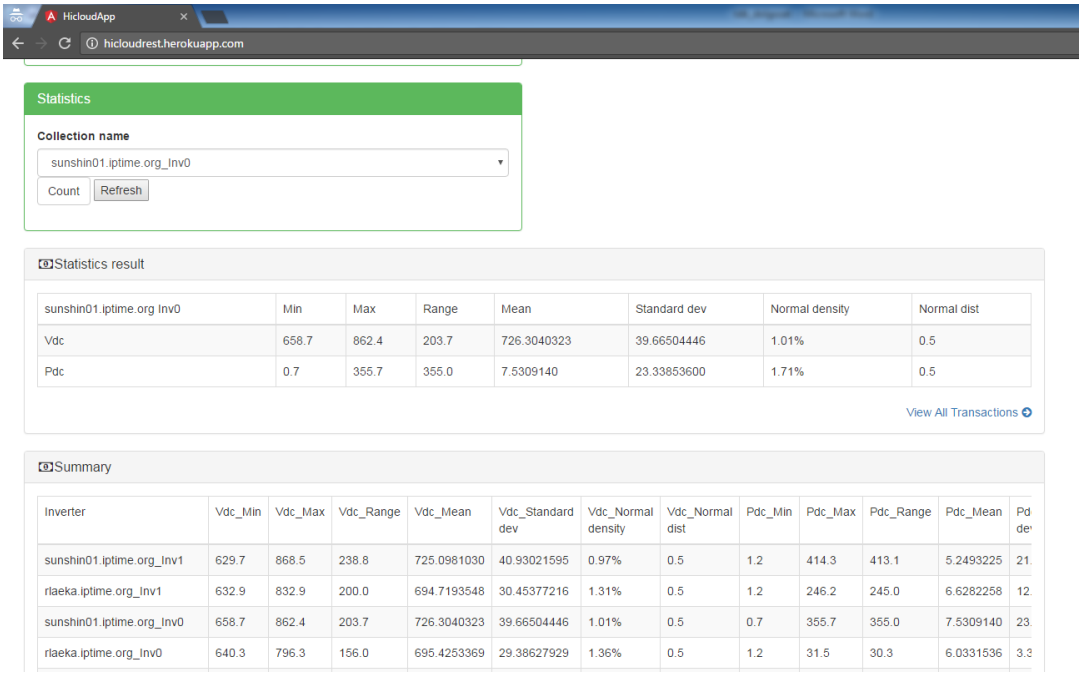
[23] Jeigen (2016. szeptember), https://github.com/hughperkins/jeigen

[24] Matlab c2d function (2016. szeptember), https://www.mathworks.com/help/control/ref/c2d.html

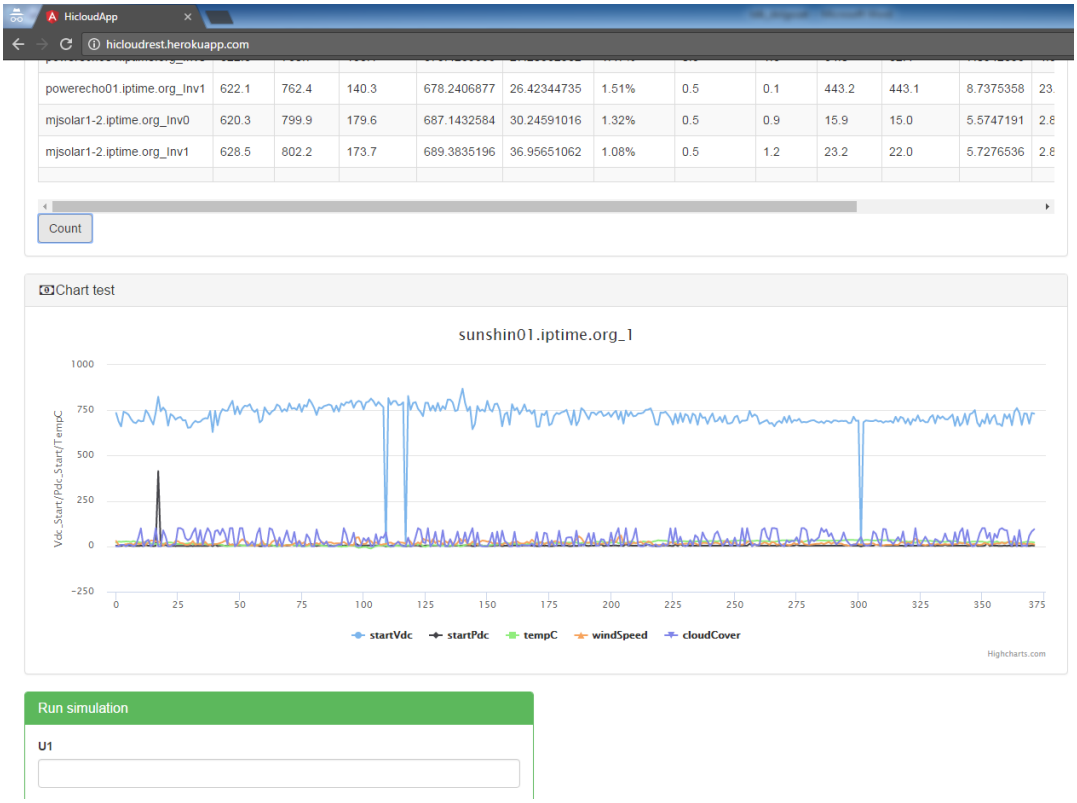
Ábrák jegyzéke

Függelék

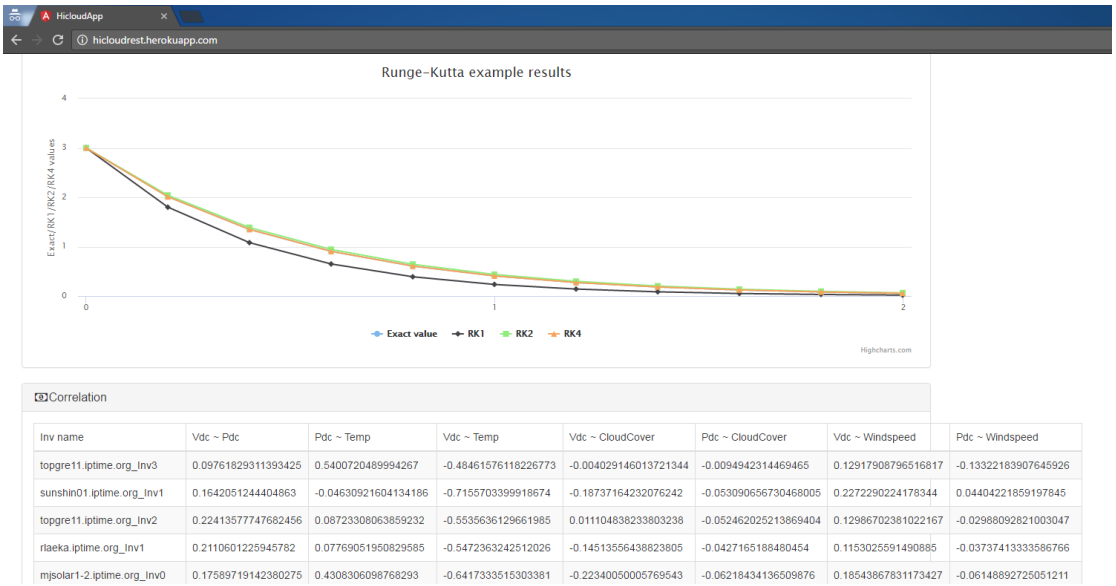
Néhány további kép az elkészült web alkalmazásról



23. ábra Statisztikai számítások a weblapon

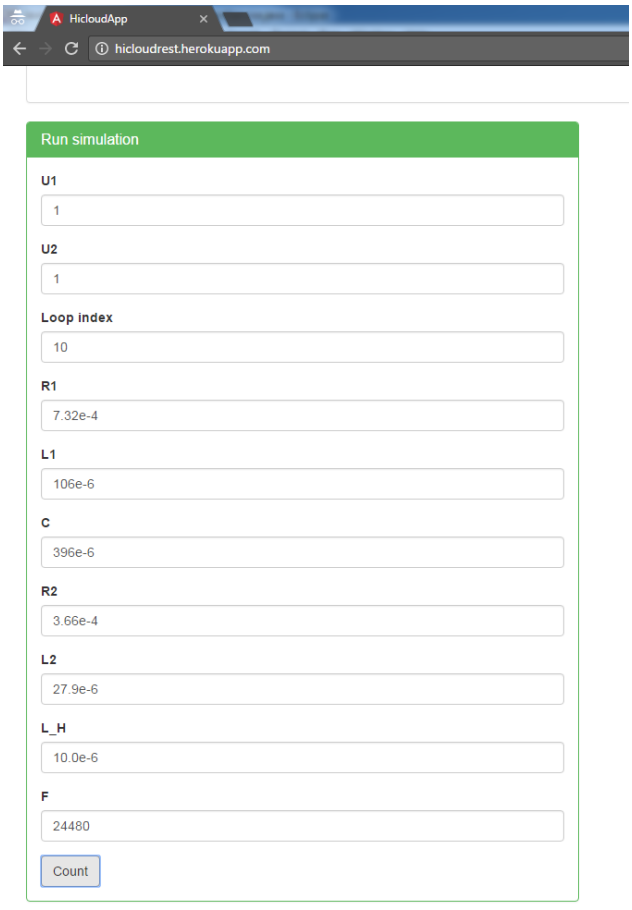


24. ábra Indulási feszültség, teljesítmény és időjárás adatok

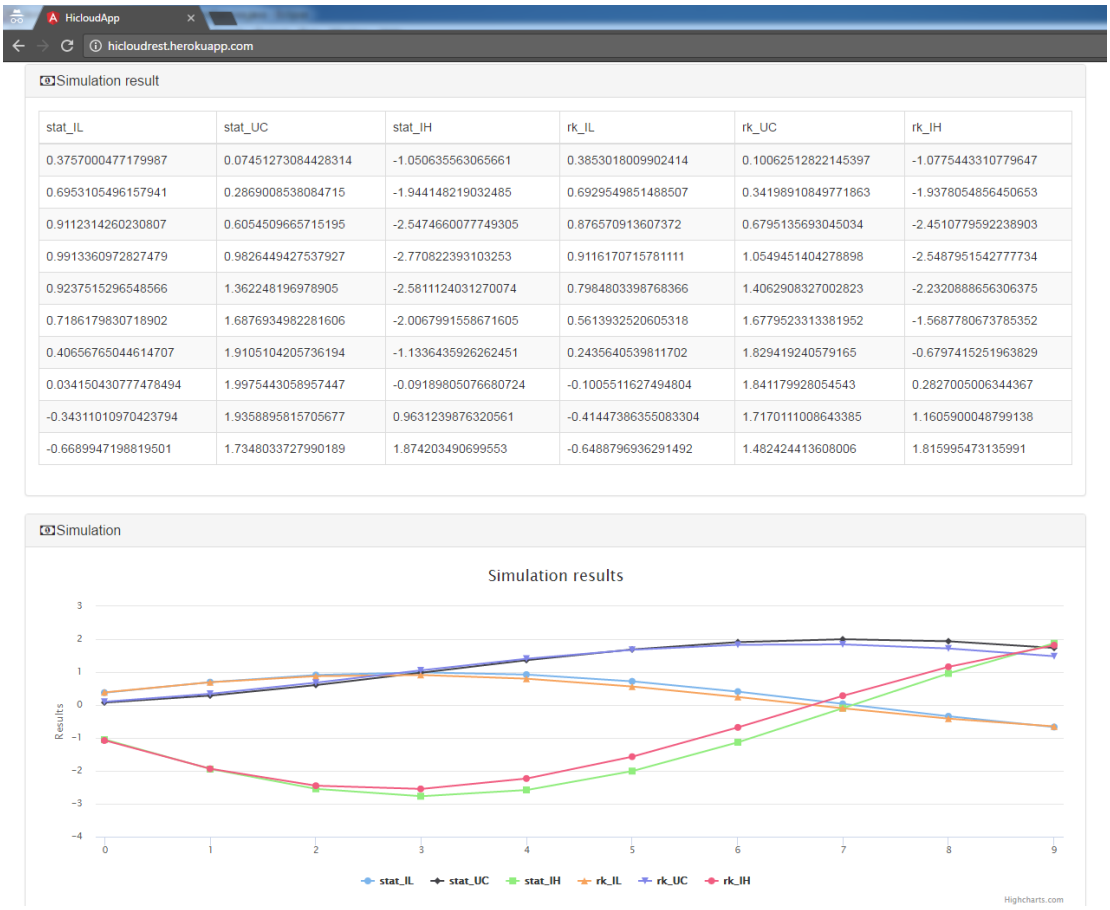


Inv name	Vdc ~ Pdc	Pdc ~ Temp	Vdc ~ Temp	Vdc ~ CloudCover	Pdc ~ CloudCover	Vdc ~ Windspeed	Pdc ~ Windspeed
topgre11.iptime.org_Inv3	0.09761829311393425	0.5400720489994267	-0.48461576118226773	-0.004029146013721344	-0.0094942314469465	0.12917908796516817	-0.13322183907645926
sunshin01.iptime.org_Inv1	0.1642051244404863	-0.04630921604134186	-0.7155703399918674	-0.18737164232076242	-0.053090656730468005	0.2272290224178344	0.04404221859197845
topgre11.iptime.org_Inv2	0.22413577747682456	0.08723308063859232	-0.5535636129661985	0.011104838233803238	-0.052462025213869404	0.12986702381022167	-0.02988092821003047
rlaeka.iptime.org_Inv1	0.2110601225945782	0.07769051950829585	-0.5472363242512026	-0.14513556438823805	-0.0427165188480454	0.1153025591490885	-0.03737413333586766
mjsolar1-2.iptime.org_Inv0	0.17589719142380275	0.4308306098768293	-0.6417333515303381	-0.22340050005769543	-0.06218434136509876	0.18543867831173427	-0.06148892725051211

25. ábra Runge-Kutta algoritmus futtatása a felhőben



26. ábra Szimulációs paraméterek megadása



27. ábra Szimuláció futásának eredménye

Runge-Kutta megoldások

Elsőrendű Runge-Kutta módszer

```
public Map<Double, Double> FirstOrderRK(){
     Map<Double, Double> values = new TreeMap<Double, Double>();
     double y0 = 3.0;
     double h = 0.2;
     int t = (int) (2.0 / h);

     double[] ystar = new double[t+1];
     ystar[0] = y0;

     values.put(new Double(0.0), new Double(y0));

     for(int i=0; i< t; i++){
          double t_exact = h * (i+1);
          double k1 = -2* ystar[i];
          ystar[i+1] = ystar[i] + ( k1*h );

          values.put(new Double(t_exact),
                         new Double(ystar[i+1]));

     }

     return values;
}
```

Másodrendű Runge-Kutta módszer

```
public Map<Double, Double> SecondOrderRk(){
     Map<Double, Double> values = new TreeMap<Double, Double>();

     double y0 = 3.0;
     double h = 0.2;
     int t = (int) (2.0 / h);

     double[] ystar = new double[t+1];
     ystar[0] = y0;

     values.put(new Double(0.0), new Double(y0));

     for(int i=0; i< t; i++){
          double t_exact = h * (i+1);
          double k1 = (-2) * ystar[i];
          double y1 = ystar[i] + k1*h/2;
          double k2 = (-2) * y1;
          ystar[i+1] = ystar[i] + ( k2*h );

          values.put(new Double(t_exact),
                         new Double(ystar[i+1]));

     }
     return values;
}
```

Negyedrendű Runge-Kutta módszer

```
    public Map<Double, Double> FourthOrderRk() {
         Map<Double, Double> values = new TreeMap<Double, Double>();

         double y0 = 3.0;
         double h = 0.2;
         int t = (int) (2.0 / h);

         double[] ystar = new double[t+1];
         ystar[0] = y0;

         values.put(new Double(0.0), new Double(y0));

         for(int i=0; i< t; i++){
              double t_exact = h * (i+1);
              double k1 = (-2) * ystar[i];
              double y1 = ystar[i] + k1*h/2;

              double k2 = (-2) * y1;
              double y2 = ystar[i] + k2*h/2;

              double k3 = (-2) * y2;
              double y3 = ystar[i] + k3*h;

              double k4 = (-2) * y3;

              ystar[i+1] = ystar[i] + (k1+2*k2+2*k3+k4)*h/6;

              values.put(new Double(t_exact),
                           new Double(ystar[i+1]));

         return values;
    }
```

Állapot átviteli mátrix alapú megoldás (szűrőkör)

```
public void STM_DiffSolver() {

     List<DenseMatrix> matrixResults = countMatrixValues();

     double Am[][] = get2dMatrix(matrixResults.get(0).rows,
                     matrixResults.get(0).cols,
                     matrixResults.get(0).getValues());
     double Bm[][] = get2dMatrix(matrixResults.get(1).rows,
                     matrixResults.get(1).cols,
                     matrixResults.get(1).getValues());

          double x_der[] = new double[X_VEC_DIM];

     x_der[X_VEC0] = Am[0][0] * state_variables[X_VEC0] + Am[0][1] *
                  state_variables[X_VEC1]+ Am[0][2] *
                  state_variables[X_VEC2] + Bm[0][0] *
                  u_v1 + Bm[0][1] * u_v2;

     x_der[X_VEC1] = Am[1][0] * state_variables[X_VEC0] + Am[1][1] *
                  state_variables[X_VEC1]+ Am[1][2] *
                  state_variables[X_VEC2]
                  + Bm[1][0] * u_v1 + Bm[1][1] * u_v2;

     x_der[X_VEC2] = Am[2][0] * state_variables[X_VEC0] + Am[2][1] *
                  state_variables[X_VEC1]+ Am[2][2] *
                  state_variables[X_VEC2]+
                  Bm[2][0] * u_v1 + Bm[2][1] * u_v2;

     // updates the state_variable values
     for (int i = 0; i < X_VEC_DIM; i++)
          state_variables[i] = x_der[i];
     }

}
```

Állapotátviteli mátrixok értékeinek újraszámítása Matlab kód alapján

```
    public List<DenseMatrix> countMatrixValues(){
        double[][] a_cont = new double[3][3];
        double[][] b_cont = new double[3][2];

        a_cont[0][1] = -1.0 / L1;
        a_cont[0][0] = -R1 / L1;
        b_cont[0][0] = 1.0 / L1;

        a_cont[1][0] = 1.0 / C;
        a_cont[1][2] = -1.0 / C;

        a_cont[2][1] = 1.0 / (L2 + L_H);
        a_cont[2][2] = -R2 / L2;
        b_cont[2][1] = -1.0 / (L2 + L_H);

        double t = 1.0 / f_samp;

        DenseMatrix  A = new DenseMatrix (a_cont);
        DenseMatrix  B = new DenseMatrix (b_cont);

        DenseMatrix  A_t = A.mul(t);
        DenseMatrix  B_t = B.mul(t);

        int nb = B.cols;
        int n = A.cols;

        DenseMatrix  Zeros = DenseMatrix.zeros(nb, n + nb);

        DenseMatrix  A_t_B_t = A_t.concatRight(B_t);

        DenseMatrix  S = A_t_B_t.concatDown(Zeros);

        DenseMatrix  S_expm = S.mexp();

        DenseMatrix A_t_exp = S_expm.slice(0, n, 0, n);
        DenseMatrix B_t_exp = S_expm.slice(0, n, n, n + nb);

        List<DenseMatrix> result = new ArrayList<DenseMatrix>();
        result.add(A_t_exp);
        result.add(B_t_exp);

        return result;

    }
```

Másodrendű Runge-Kutta módszer (szűrőkör)

```
public void RungeKutta2_dsp() {
     double tempvar[] = new double[X_VEC_DIM];
     double tmp[] = new double[X_VEC_DIM];
     double k1[] = new double[X_VEC_DIM];
     double k2[] = new double[X_VEC_DIM];

     tempvar = copy_variables(state_variables);

     k1 = var_grad_dsp(tempvar);

     tmp = add variables 1(tempvar, step * 2.0 / 3.0, k1);

     k2 = var_grad_dsp(tmp);

     state_variables = add_variables_2(tempvar, step / 4.0, k1,
                                       step * 3.0 / 4.0, k2);
}

public double[] var_grad_dsp(double[] X) {
     double[] dX = new double[X.length];

     dX[X_VEC0] = (U_IN - X[X_VEC1] - R_CH * X[X_VEC0]) / L_CH;
     dX[X_VEC1] = (X[X_VEC0] - X[X_VEC2]) / C_F;
     dX[X_VEC2] = (X[X_VEC1] - U_GRID - R_TR * X[X_VEC2]) / L_TR;

     return dX;
}

public double[] add_variables_1(double[] src, double stp1, double[]
dta1){
     double[] trg = new double[src.length];

     trg[X_VEC0] = src[X_VEC0] + stp1 * dta1[X_VEC0];
     trg[X_VEC1] = src[X_VEC1] + stp1 * dta1[X_VEC1];
     trg[X_VEC2] = src[X_VEC2] + stp1 * dta1[X_VEC2];

     return trg;
}

public double[] add_variables_2(double[] src, double stp1, double[]
dta1, double stp2, double[] dta2) {

     double[] trg = new double[src.length];

     trg[X_VEC0] = src[X_VEC0] + stp1 * dta1[X_VEC0]
                   + stp2 *dta2[X_VEC0];
     trg[X_VEC1] = src[X_VEC1] + stp1 * dta1[X_VEC1]
                   + stp2 *dta2[X_VEC1];
     trg[X_VEC2] = src[X_VEC2] + stp1 * dta1[X_VEC2]
                   + stp2 *dta2[X_VEC2];

     return trg;
}
```