Építés – Építészettudomány
DOI: 10.1556/096.2025.00153

1828
AKADÉMIAI KIADÓ

Parametrikus tervezés az építőiparban: fogalmak, eszközök, esettanulmányok

Tóth Bálint[1*] – Bükkösi Raymond[2] – Lógó János[3]

[1] Budapesti Műszaki és Gazdaságtudományi Egyetem, Építőmérnöki Kar, Tartószerkezetek Mechanikája Tanszék, 1111 Budapest, Műegyetem rakpart 3. K mfsz. 63. (Politecnico di Milano.) E-mail: toth.balint@emk.bme.hu

[2] ARC-S Group Kft., Budapest. E-mail: raymond.bukkosi@arcs.hu

[3] Budapesti Műszaki és Gazdaságtudományi Egyetem, Építőmérnöki Kar, Tartószerkezetek Mechanikája Tanszék, 1111 Budapest, Műegyetem rakpart 3. K mfsz. 63. E-mail: logo.janos@emk.bme.hu

EREDETI KÖZLEMÉNY

Beérkezett: 2025. június 16. • Elfogadva: 2025. augusztus 18.

ÖSSZEFOGLALÓ

A számítógépek a 20. század végétől kezdve egyre meghatározóbb szerepet kaptak az építőipari és építészeti tervezésben. A különböző tervezési folyamatok jellemzésére számos új fogalom került bevezetésre (például digitális, komputációs, algoritmikus, generatív és parametrikus tervezés), amelyek sokszor átfedésekkel bírnak, és amelyek meghatározása a szakirodalomban is folyamatosan finomításra szorul. A tanulmány célja ezen fogalmak áttekintése és a gyakorlati alkalmazások vizsgálata esettanulmányokon keresztül. A dolgozatban kitérünk a leggyakrabban használt szoftveres platformokra (Tekla, Revit, Allplan, Grasshopper) és azok integrációjára, különös tekintettel a parametrikus tervezés és a numerikus számítási környezetek összekapcsolására. Az esettanulmányok során bemutatásra kerül egy stadion homlokzati burkolati rendszerének racionalizálása, egy szabadformájú épület szabadszereléséhez szükséges támogatórendszer, illetve egy Grasshopper–Matlab-alapú adatkapcsolat kiépítésének lehetősége. A tanulmány gondolatébresztő példákon keresztül világít rá, hogy a digitális és komputációs eszközök integrációja miként járulhat hozzá a tervezési és kivitelezési folyamat hatékonyságának növeléséhez.

KULCSSZAVAK

számítógépes tervezés, digitális tervezés, komputációs tervezés, algoritmikus tervezés, parametrikus tervezés, generatív tervezés, Grasshopper, BIM, geometriai racionalizálás, adatkapcsolat

* Levelező szerző

BEVEZETÉS

A számítógépek a 20. század végétől kezdve szerves részét képezik az építőipari és építészeti munkának. Az építészeti tervezés különböző aspektusaira – például a tömegformálásra[1] vagy a térszervezésre[2] – egyaránt hatással voltak, de ugyanígy jelentős szerepet kaptak a statikai tervezési folyamatokban is.[3] Ez az időszak rengeteg új fogalom megszületését eredményezte, amelyek a számítógép adta lehetőségek eltérő használatából és a különböző szakmai megközelítésekből táplálkoznak.

A tervezési módszerek közül a leggyakrabban használt fogalmak közé tartozik a digitális, komputációs, parametrikus, algoritmikus és generatív tervezés, az angol nyelvű szakirodalomban: digital design (DD), computational design (CD), parametric design (PD), algorithmic design (AD), generative design (GD). A definíciók és kategóriák tisztázásában Inês Caetano és szerzőtársainak cikke[4] nyújt segítséget, amely átfogó szakirodalmi áttekintést kínál a komputációs tervezés kulcsfogalmainak rendszerezéséhez és a köztük lévő kapcsolatok feltárásához.

A forrás alapján a CD területén két, részben átfedő, meghatározó halmaz azonosítható: a PD és a GD. A PD olyan megközelítés, amelyben a lehetséges tervek halmaza paraméterek segítségével határozható meg, míg a GD esetén algoritmusok alkalmazásával állíthatók elő tervek. A két megközelítés kombinálható, hiszen az algoritmusok paraméterekkel is vezérelhetők, bár ez nem feltétlenül szükséges.

Az AD szintén algoritmusokkal generált tervezési módszereket takar, ezért a szakirodalomban gyakran a GD szinonimájaként jelenik meg. Ugyanakkor AD esetében fennáll az algoritmus és a létrehozott modell közötti egyértelmű megfeleltetés, ami lehetővé teszi a modell egyes részeinek az algoritmus megfelelő részeihez való hozzárendelését. Ez a nyomonkövethetőség megkönnyíti a hibakeresést és elősegíti a hatékonyabb fejlesztést. AD eszerint GD részhalmazaként azonosítható.

Külön említésre méltó ugyanakkor, hogy CD nem feltétlenül kell, hogy digitális is legyen. Számos történeti példát ismerünk, amelyek analóg fizikai rendszerek révén végeztek számítási feladatokat. Frei Otto minimálfelületi kísérletei,[5] Gaudí fizikai modelljei[6] és Heinz Isler feszített membrán modelljei[7] egyaránt bizonyítják, hogy a számítás analóg eszközökkel is megvalósítható. Ezek az esetek jól mutatják, hogy a tervezési folyamat során a fizikai rendszerek maguk „végzik el" a számítást, segítve ezzel a formaalakítás és a szerkezeti optimálás folyamatát.

A digitális eszközök fejlődésével párhuzamosan az additív gyártás, különösen a 3D nyomtatás is egyre inkább kutatott téma az építőiparban. Ezen technológiák jól ki tudják használni a számítógépes tervezési eszközök adta lehetőségeket, például topológiaoptimálással vagy generatív tervezési eljárásokkal elérhető geometriák előállítása kapcsán.[8] Az acélszerkezetek 3D nyomtatása, például a wire arc additive manufacturing (WAAM) technológia,[9] segítségével és a beton-

[1] Aish–Woodbury 2005. 1658.
[2] Shihai et al. 2023. 107156.
[3] Maher–Burry 2003.
[4] Caetano–Santos–Leitão 2020.
[5] Glaeser 1972.
[6] Crippa 2007.
[7] Chilton 2000.
[8] Buonamici et al. 2021.
[9] Meng–Gardner 2025.

szerkezetek 3D nyomtatásával[10] egyaránt új kihívásokat és lehetőségeket jelent a tervezők számára. A Budapesti Műszaki és Gazdaságtudományi Egyetem kutatási projektjei is aktívan foglalkoznak nyomtatott betonszerkezetek tulajdonságainak vizsgálatával.[11]

A generatív modellek előállítása során keletkező pluszadatok a későbbi tervezési, gyártási, kivitelezési fázisokban is hasznosak lehetnek, illetve további előnyük, hogy sok esetben váratlan ötletek (úgynevezett „happy accidents") is felmerülhetnek, melyek újfajta inspirációt adhatnak a tervezőknek.[12] Az algoritmikusan előállított modellek rugalmasan alakíthatóak, gyors, finom módosítások hajthatóak végre rajtuk a megfelelő kezdeti paraméterek definiálsa esetén és általában jól beilleszthetőek a Building Information Modelling (BIM) alapú tervezési folyamatokba.[13] Hátrányuk azonban, hogy a meglévő logikai kapcsolatok jelentősebb módosítása, új paraméterek bevezetése már nem mindig egyszerű.

Összességében elmondható, hogy a CD rendkívül sokféle lehetőséget kínál, de ugyanakkor magában hordozza az AD túlzott használatának veszélyét is. A sok lehetőség hatására hajlamosak lehetünk mindent kiszervezni a számítógépnek, és adott esetben olyan feladatokat is automatizálni, amelyeket manuálisan gyorsabban és kreatívabban tudnánk megoldani.[14] A tanulási folyamat szempontjából pozitív, a hatékony munkavégzés szempontjából viszont kifejezetten káros, hogy az algoritmikus tervezési lehetőségekkel való ismerkedés során az ember hajlamos minden problémát ilyen módon megoldani, ami sokszor több időt és energiát vehet igénybe, mintha a hagyományos módon, manuálisan végeznénk el a feladatot. Emiatt nagyon fontos a problémákat az elején átgondolni, megtervezni az algoritmus felépítését, és mérlegelni, hogy valóban szükség van-e ilyen módon nekilátni a tervezésnek vagy modellezésnek.

SZOFTVERES HÁTTÉR

Az előző fejezetben bemutatott fogalmak mind szoros kapcsolatban állnak a különböző szoftveres eszközökkel és programnyelvekkel. Ebben a fejezetben a hazai gyakorlatban leginkább használatos, illetve a fenti tervezési folyamatokhoz leginkább illeszkedő eszközök és ezek integrálási lehetőségei kerülnek áttekintésre.

Az építőmérnöki digitális tervezési (DD) folyamatokban a legismertebb platformok közé tartozik a Tekla, a Revit, az Archicad és az Allplan. Ezek a szoftverek különböző szintű automatizálást és rugalmasságot biztosítanak a tervező számára, attól függően, hogy milyen célokra kívánják őket használni. A Tekla például kifejezetten alkalmas részletes acélszerkezeti modellezésre és gyártási dokumentációk készítésére, míg a Revit erőssége a BIM-alapú építészeti modellezés és információkezelés. Az Allplan főként a vasbeton-modellezési és -dokumentálási feladatokban nyújt segítséget. Az építészeti tervezésben meghatározó szoftver még az ArchiCAD. Ezen platformok fejlesztése folyamatos, és potenciálisan a fentiekben ismertetetteknél szélesebb körű felhasználási lehetőségeket is kínálnak, ugyanakkor jellemzően a megadott területeken alkalmazzák őket. Felépítésük alapján a PD kategóriájába sorolhatók, bár ez elsődleges vizsgálatkor ez nem feltétlenül nyilvánvaló.

---

[10] Li et al. 2025a. 2001–2030; Santhosh–Raphael–Santhanam 2025. 04025025.
[11] Thajeel–Balázs 2025.
[12] Chien–Yeh 2012.
[13] Li et al. 2025b.
[14] Lawson 2005.

A fent ismertetett megközelítésekhez képest a Grasshopper (GH) funkcionálisan különböző. Bár a magyarországi mérnöki szakmai diskurzusban gyakran a parametrikus tervezéssel azonosítják, a bevezetésben bemutatott értelmezés szerint azonban kiemelendő, hogy a PD a GH használata nélkül is megvalósítható. Továbbá a gyakorlatban a GH környezetében létrehozott megoldások jellemzően az algoritmikus tervezés (AD) körébe sorolhatók. A GH tulajdonképpen egy vizuális programozási nyelv, amely a Rhino környezetében működik, mely egy általános, nem egyenközű, racionális B-spline (Non-uniform Rational B-Splines), vagyis NURBS alapú 3D modellező szoftver. Elsajátításához nem szükséges előzetes programozói tudás, ám támogatja a Python és a C# nyelveket is. Továbbá jól dokumentált, így könnyen fejleszthetőek hozzá saját pluginok is. Hozzá hasonló még a Revithez tartozó, szintén vizuális Dynamo felülete vagy az ArchiCAD-hez tartozó GDL nyelv.

A kiemelkedő megközelíthetőségnek köszönhetően a legtöbb CAD és végeselemes szoftver ma már biztosít valamilyen kapcsolódási pontot a GH platformhoz. Statikai tervező szoftverekkel is összekapcsolható, így nemcsak a geometria, hanem a statikai váz tervezésében és optimalizálásában is segítségünkre lehet. Könnyű tanulhatósága miatt az algoritmikus tervezésben a vizuális programozás kiemelt szerepet kapott az utóbbi években.

Néhány speciálisabb feladat esetén azonban szükség lehet hagyományos programnyelvek használatára is. Ezek elsőre sokszor távolinak tűnhetnek az építőipari szakemberek számára, ám elsajátításuk nagyban hozzájárulhat bonyolult projektek gyorsabb és hatékonyabb megvalósításához.

Előfordulhat, hogy a rendelkezésre álló eszközök nem elegendőek bizonyos ipari projektek esetében, vagy kutatási projektünk során szeretnénk integrálni meglévő ipari eszközöket. Ilyen helyzetekben az alapvető szkriptelési tudás jelentősen megkönnyítheti a tervezési folyamatot és segíthet a projekt hatékonyabb lebonyolításában.[15] Ennek megsegítése érdekében a cikkhez egy konkrét példa is kapcsolódik mely gondolatébresztőként szolgál.

ESETTANULMÁNYOK

1. példa: Geodéziai visszamérés és adatkapcsolat szerepe a Biodom szabadszerelésében

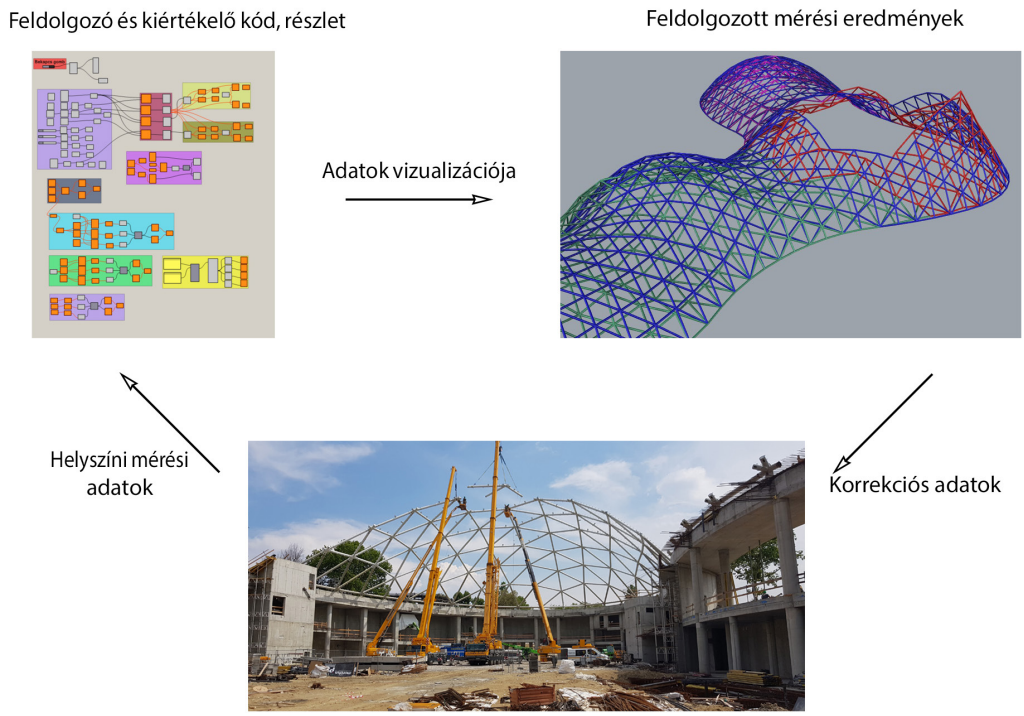
A Biodom projekt szabadformájú acélszerkezetének kivitelezése során a szerkezet geometriájából adódóan különleges kihívások merültek fel. Az egyik legfőbb kérdés az volt, hogy hogyan lehet a bonyolult geometriájú szerkezetet olyan módon megépíteni, hogy a szereléshez ne legyen szükség teljes körű, kiterjedt állványrendszer felállítására. A cél a „szabadszerelés" megvalósítása lett, amely során csak minimális, lokális állványzatra van szükség a szerkezet szerelésekor. A projekt a digitális tervezésben széles körűen használta a digitális tervezés eszközeit.

Szabadszerelés során a kivitelezésnek előre meghatározott peremfeltételek (például maximális lokális elmozdulás: 0–25 mm, globális maximum: 100 mm) mellett kellett biztosítani, hogy a szerkezet geometriai pontossága a szerelés minden ütemében teljesüljön. A fő probléma abból adódott, hogy a szerelés előrehaladtával a szerelendő perem folyamatosan mozgásban volt:

[15] Tóth–Bruggi–Lógó 2023.

a részlegesen merev kapcsolatok és a szerelési sorrend miatt a szerkezet ideiglenesen deformálódott, így a következő szerelési ütem pozíciója állandóan változott.

A szerelési folyamat zavartalan előrehaladása érdekében ezért egy folyamatos monitoring rendszert kellett felállítani, amely a helyszíni geodéziai mérések segítségével folyamatosan figyelte a szerkezet aktuális geometriai állapotát. A mérések kiértékelése a Rhino–GH környezetben történt, ahol a TXT formátumban beérkező Egységes Országos Vetületi rendszer (EOV) szerinti koordinátákat gyorsan és hatékonyan lehetett feldolgozni és összevetni a tervezett geometriai modellel egy automatizált rendszer segítségével. Az *1. ábra* jobb felső képe illusztrálja ezt. A sötétkék színnel (nyomtatott verzió esetén fekete) az adott építési fázisban számított elmozdulás látható. A többi szín (nyomtatásban: a világosabb árnyalatok) a szerelés közbeni, valós idejű mérés eredményeit ábrázolja, amelyeket egy meghatározott tényezővel felnagyítottunk a könnyebb értelmezhetőség érdekében. A tervezett geometria vonalát – amely a tervezési állapotot szemlélteti – egy további színnel jelöltük, azonban ez a vizsgált ábrán nem szerepel. A valós állapotok gyors ellenőrzését követően a szükséges korrekciók – például a csomópontok visszaemelése – gyorsan meghatározhatók voltak.



1. ábra. Monitoring rendszer adatfolyamának áttekintése

A GH–Tekla kapcsolat biztosította, hogy a tervezési modell és a kivitelezési folyamat összhangban maradjon: a GH-ben feldolgozott adatok visszacsatolásra kerültek a Tekla modellbe, így a kivitelezés pontosan nyomon követhető, és frissíthető volt. Ez lehetővé tette, hogy a szerelési egységek elhelyezése mindenkor az aktuális állapothoz igazítva történjen meg, és a szerkezet szerelése folyamatosan és ütemezetten haladhasson előre.

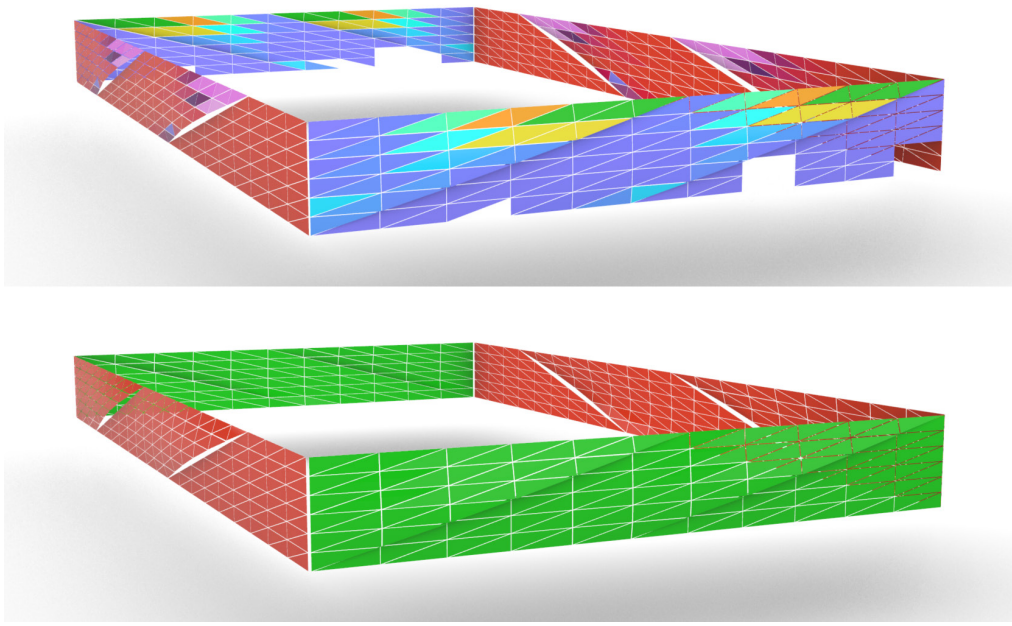
A bemutatott módszer tehát kulcsfontosságú szerepet játszott abban, hogy a szabadformájú acélszerkezet a szabadszerelés elvének megfelelően, minimális állványzat felhasználásával és a geometriai pontosság biztosításával épülhessen meg. A helyszíni visszamérési rendszer és a gyors adatintegráció révén a szerelés során fellépő geometriai elmozdulások azonnal detektálhatók és korrigálhatók voltak, így a kivitelezés pontosan és hatékonyan valósulhatott meg.

2. példa: A paraméteres racionalizálás esettanulmánya az MTK Stadion homlokzati lemezburkolatának tervezésében

Az algoritmikus tervezés gyakorlati alkalmazását jól illusztrálja az MTK Stadion homlokzati lemezburkolatának optimálása, amely során a burkolati elemek kiosztásának racionalizálására és a gyártási folyamatok hatékonyabbá tételére került sor. A tervezési folyamat kezdetén az építésziroda egy kétszeresen görbült felületű homlokzati modellt hozott létre, amelyet látszólag egységes, háromszög alapú burkolati elemekre osztottak fel. A részletes elemzés azonban rávilágított, hogy a 700 elemből álló homlokzat valójában 87 különböző méret- és alakváltozatú háromszögből állt össze az áttörések miatti eltérések számítása nélkül, ami jelentősen megnehezítette volna a gyártást és a kivitelezést.

A gyártástechnológiai és kivitelezési szempontból is célszerűbb megoldás érdekében a célkitűzés egy egyenletesebb, ismétlődő elemekből álló burkolati rendszer létrehozása lett.

A kétszeresen görbült felület matematikai meghatározásához egy félszinuszhullám és egy Gauss-görbe kombinációjából előállított digitális modell lett alkalmazva, amely lehetővé tette a



2. ábra. A kapott elrendezés (felül) és a racionalizált homlokzati elemek (alul) terület alapján színezve

geometria pontos, egyértelmű definiálását és a későbbi módosítások gyors végrehajtását. Ez lehetővé tette, hogy a tervezési folyamat során az építész tervezőcsapattal folyamatos, jelen idejű egyeztetés valósuljon meg a geometriai változtatásokról. A gyakorlatban: közösen finomhangoltuk a paramétereket, amíg az építészek rá nem bólintottak a megengedhető geometriai eltérésre.

A kétszer görbült felület és ennek egy sík vetülete is fel lett osztva háromszögekre az eredeti elképzelésnek megfelelően. Az így kapott sík háromszögeket a súlypontjuknál fogva a kétszeresen görbült felület háromszögeinek súlypontjába mozgatva és normálvektoraikat összehangolva alakítottuk ki az új elrendezést. Ennek eredményeként egy olyan burkolati rendszer jött létre, amely 2 különböző variációra csökkentette az elemtípusok számát, miközben követte az építészeti koncepciót. A kiinduló és a racionalizált homlokzat összehasonlítása alább látható *(2. ábra)*.

Mivel a felület a kétszeres görbülete miatt nem volt síkba teríthető, szükség volt a csatlakozó lemezek közötti hézagok lokális módosítására annak érdekében, hogy a vizuálisan egységes burkolati felület érzetét megőrizzük. A tervezési folyamatban a kiosztás optimálásának támogatására a GH környezetében elérhető Galapagos evolúciós optimalizáló algoritmus került alkalmazásra, amely lehetővé tette az egyes lemezelemek pozíciójának finomhangolását és a rendszer paraméteres optimalizálását. Az elemek a befogók mentén tudtak mozogni a saját síkjukban, míg a célfüggvény értéke a legkisebb és legnagyobb hézag méretéből lett számítva globálisan. Az eredeti 50 mm hézag helyett a végső elrendezésben 42–88 mm között mozogtak ezek a méretek. Ezek a különbségek vizuálisan elhanyagolhatóak a homlokzet jellemző méreteihez képest *(3. ábra)*.

3. ábra. Megvalósult szerkezet

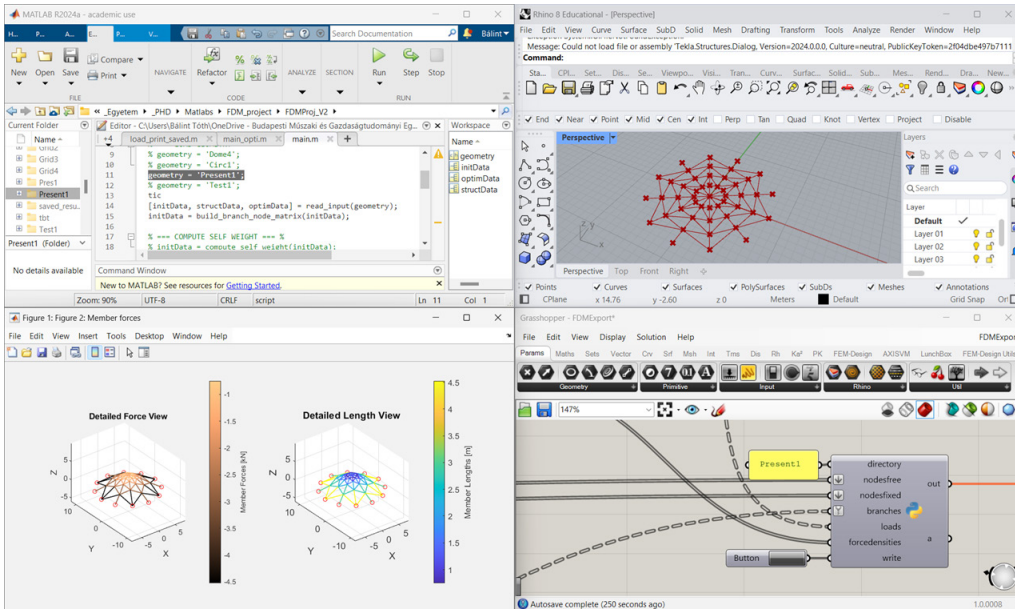
A bemutatott esettanulmány rávilágít arra, hogy a parametrikus tervezési eszközök és optimalizáló algoritmusok hogyan járulhatnak hozzá a komplex geometriai felületek gyárthatóságának és szerelhetőségének javításához, miközben megőrzik az eredeti építészeti elképzelést és az esztétikai követelményeket. A módszer eredményeként egy gazdaságosabb és könnyebben kivitelezhető homlokzati rendszer jött létre, amely egyúttal kielégíti a vizuális követelményeket is.

3. példa: Grasshopper- és Matlab-alapú adatkapcsolat alkalmazása kutatási környezetben

A számítógépes tervezés és a kutatási célú numerikus számítások összekapcsolására jó példát nyújt egy Matlab-környezetben fejlesztett számítási kód és a Rhino környezetében működő GH platform integrálása, amely a kutatási feladatok során az adatok előkészítését és kezelését jelentősen megkönnyítette. Ebben az esetben a Matlab szoftver végezte a numerikus számításokat, illetve az eredmények generálását, míg a GH a geometriai adatok gyors és rugalmas előállítását támogatta.

A Matlab kód önmagában képes a numerikus számítások elvégzésére és az eredmények kiértékelésére, azonban a számítások bemenő adatait jelentő geometriai modell manuális bevitele nehézkes és időigényes folyamat lett volna, hiszen a matematikai modell elkészítéséhez a csomópontok koordinátáit, valamint a rudak kezdő- és végpontjait egyaránt meg kellett adni. Ezek kézi előállítása és ellenőrzése jelentős hibalehetőséget jelentett volna, ráadásul az iteratív próbafuttatások során időveszteséget is okozott volna.

A fenti problémát áthidalva a Rhino környezetében működő GH vizuális programozási környezet került használatra a kívánt kiinduló geometria létrehozására. A GH rugalmasságának és



4. ábra. A Grasshopper által létrehozott mappa beolvasása Matlab segítségével

intuitív kezelhetőségének köszönhetően a bonyolultabb geometriai konfigurációk is gyorsan és pontosan megrajzolhatóak Rhino segítségével vagy közvetlenül generálhatóak GH-ban, a tervezési folyamat során pedig a szükséges támaszok és terhek is könnyedén definiálhatók.

A geometriai adatok – például a pontok koordinátái és a rudak kezdő- és végpontjainak adatai – néhány soros Python script segítségével exportálhatók szabványos CSV vagy TXT fájlformátumba közvetlenül a GH-ból. Ezek a fájlformátumok egyszerűen beolvashatók Matlab-környezetben, így az adatok integrálása a numerikus számításokat végző programhoz gyors és hibamentes folyamatként valósult meg *(4. ábra)*.

A Matlab kód át lett alakítva annak érdekében, hogy a GH-ból exportált adatfájlokat közvetlenül be tudja olvasni, és automatikusan létre tudja hozni a szükséges geometriai és statikai modellt. Ennek köszönhetően a kutatás során könnyen tesztelhetők voltak a kód különböző, akár bonyolultabb geometriákra vonatkozó futtatásai is.

Ez a megközelítés jól szemlélteti, hogy a vizuális programozási környezetek (GH) és a numerikus számításokat futtató programnyelvek (Matlab) kombinálása hogyan könnyítheti meg a kutatási munka során az adatok előállítását és kezelését, miközben lehetővé teszi a gyors kísérletezést és az eredmények hatékonyabb generálását.

Kiegészítő példa: Grasshopper és Matlab közötti élő kapcsolat kialakítása Python segítségével

A következő példa a GH és a Matlab közötti adatkapcsolat létrehozásának lehetőségét illusztrálja. Célja, hogy inspirációt nyújtson más felhasználóknak a két környezet összekapcsolásához. Ennek demonstrálására egy élő kapcsolatot létesít a két szoftver között, ami a Rhinóban vagy GH-ban létrehozott geometria jelen idejű megjelenését támogatja egy Matlab plot ablakban. A kiinduló kód segítségével a logika átültethető ennél bonyolultabb feladatok megoldásába is.

A GH környezetében vagy a Rhino felületén először létre kell hozni a kívánt geometriai modellt, amely a szerkezeti elemek tengelyvonalait tartalmazza. Ezután a GH-ban egy Python komponens segítségével a „lines" bemenethez kell csatlakoztatni a definiált geometriai adatokat. A Python kód folyamatosan figyeli a „lines input" állapotát és ellenőrzi a geometria változását a GH-ban.

A kód futtatásakor létrehoz egy mappát az asztalon, amely két TXT fájlt tartalmaz. Az egyik fájl a csomópontok koordinátáit rögzíti, míg a másik a tengelyek kezdő- és végcsomópontjainak indexeit tartalmazza. Az első oszlop mindkét esetben a sorok indexelése. A Python script folyamatosan frissíti ezeket a fájlokat, így a geometriai változtatások valós időben követhetők.

A Matlab kód állítható időintervallumonként olvassa ezeket a fájlokat és a beolvasott adatok alapján automatikusan generálja a kívánt grafikai megjelenítést, a konkrét példában a tengelyek ábrázolását. Így a GH-ban végrehajtott geometriai módosítások azonnal megjelennek a Matlab környezetében is.

Ez a példa jól mutatja, hogy a Python programozási nyelv segítségével a GH és a Matlab közötti kommunikáció viszonylag egyszerűen megvalósítható. Ez a megközelítés lehetővé teszi a geometriai modellek és a numerikus számítások közötti valós idejű adatcserét, amely akár oktatási vagy kutatási célokra is hasznos lehet. A fent leírt GH és matlab scriptek az *A és B melléklet*ben csatolva találhatóak meg. Fontos megjegyezni, hogy a megfelelő működés érdekében a GH-n belüli Python komponensnek az egyetlen bemenete "lines" elnevezést kell kapjon, és ezt a bemenetet "List Access" opcióra kell állítani.

ÖSSZEFOGLALÁS

A dolgozatban bemutatásra került, hogy a komputációs és digitális tervezés milyen jelentős mértékben vált a kortárs építőipar és építészeti gyakorlat szerves részévé a 20. század végétől kezdődően. Ismertetésre kerültek azok a kulcsfogalmak – például a digitális, komputációs, algoritmikus, generatív és parametrikus tervezés –, amelyek a számítógépes tervezési folyamatok alapját képezik. A szakirodalomban ezek tisztázására tett kísérletek hasznos támpontot nyújtanak a fogalmak rendszerezéséhez és a közöttük lévő kapcsolatok feltárásához.

A dolgozatban bemutatásra kerültek a hazai szerkezettervezési gyakorlatban leginkább használt modellező szoftverek (Tekla, Revit, Allplan, GH) és azok sajátosságai. Külön hangsúlyt kapott a Rhino vizuális programozási környezete (GH), amely kiemelkedő rugalmasságot biztosít a tervezők számára. A szoftveres integráció kérdése szintén tárgyalásra került, rámutatva arra, hogy a digitális környezetek közötti adatkapcsolat milyen módon támogathatja a tervezési, kivitelezési és kutatási folyamatokat.

A három esettanulmány illusztrálta a komputációs és digitális tervezés gyakorlati előnyeit. Az első példa az MTK Stadion homlokzati lemezburkolatának racionalizálási folyamatát elemezte, amely során parametrikus modellezési technikák és evolúciós algoritmusok alkalmazásával sikerült a gyártási és kivitelezési folyamatokat kezelhetőbbé és gazdaságosabbá tenni. A második esettanulmány a Biodom projekt szabadformájú acélszerkezetének szabadszereléséhez használt monitoring rendszert mutatta be, amelynél a GH és Tekla közötti adatkapcsolat, valamint a geodéziai visszamérési rendszer integrálása tette lehetővé a szerelési folyamat pontos követését és a szerkezet geometriai pontosságának biztosítását minimális állványhasználat mellett. A harmadik esettanulmány egy GH–Matlab adatkapcsolati példa révén világított rá arra, hogy a parametrikus modellek és a numerikus számítások közötti valós idejű adatcsere miként teheti hatékonyabbá a kutatási folyamatokat.

A dolgozatban hangsúlyosan kiemelésre került, hogy a komputációs tervezés eszközei rendkívül sokféle lehetőséget kínálnak a tervezők számára, ugyanakkor az algoritmikus tervezés túlhasználatának veszélye is fennáll. Ezért elengedhetetlen, hogy a tervezési feladatok elején átgondoltan történjen a probléma definiálása, az algoritmus felépítése és annak mérlegelése, hogy az algoritmikus eszközök alkalmazása valóban a leghatékonyabb megoldást biztosítja-e.

A dolgozat célkitűzése tehát az volt, hogy a számítógépes tervezés alapfogalmait, szoftveres hátterét és gyakorlati alkalmazását bemutassa, miközben felhívja a figyelmet arra, hogy az algoritmikus tervezési folyamatok csak a megfelelő alkalmazás esetén képesek a hatékonyság növelésére. A komputációs és digitális tervezés helyes és tudatos használata ugyanakkor jelentős potenciált kínál az építészeti és mérnöki tervezés jövőbeli fejlődéséhez.

KÖSZÖNETNYILVÁNÍTÁS

A 2024-2.1.2-EKÖP-KDP-2024-00005 számú projekt a Kulturális és Innovációs Minisztérium Nemzeti Kutatási Fejlesztési és Innovációs Alapból nyújtott támogatásával, az EKÖP_KDP-24-1-BME-24 pályázati program finanszírozásában valósult meg.

IRODALOMJEGYZÉK

Aish–Woodbury 2005 — Aish, Robert – Woodbury, Robert: Multi-level Interaction in Parametric Design. In *Smart Graphics. SG 2005. Lecture Notes in Computer Science*. Vol. 3638. Springer, Berlin, Heidelberg 2005. https://doi.org/10.1007/11536482_13

Buonamici et al. 2021 — Buonamici, Francesco – Carfagni, Monica – Furferi, Rocco – Volpe, Yari – Governi, Lapo: Generative Design: An Explorative Study. *Computer-Aided Design and Applications* 18 (2021) 1. 144–155. doi: https://doi.org/10.14733/cadaps.2021.144-155

Caetano–Santos–Leitão 2020 — Caetano, Inês – Santos, Luís – Leitão, António: Computational design in architecture: Defining parametric, generative, and algorithmic design. *Frontiers of Architectural Research* 9 (2020) 2. 287–300. ISSN 2095-2635. https://doi.org/10.1016/j.foar.2019.12.008. (https://www.sciencedirect.com/science/article/pii/S2095263520300029)

Chien–Yeh 2012 — Chien, Sheng-Fen – Yeh, Yee-Tai: On creativity and parametric design: A preliminary study of designer's behaviour when employing parametric design tools. In *Digital Physicality – Proceedings of the 30th eCAADe Conference* 1 (2012) 245–253. Czech Technical University in Prague, Faculty of Architecture. https://doi.org/10.52842/conf.ecaade.2012.1.245

Chilton 2000 — Chilton, John: *Heinz Isler: The engineer's contribution to contemporary architecture*. Thomas Telford, 2000.

Crippa 2007 — Crippa, Maria Antonietta: *Gaudí: The complete buildings* (R. Donnell, Trans.). Taschen, 2007.

Glaeser 1972 — Glaeser, Ludwig: *The work of Frei Otto*. The Museum of Modern Art, 1972.

Lawson 2005 — Lawson, Bryan: Oracles, draughtsmen, and agents: The nature of knowledge and creativity in design and the role of IT. *Automation in Construction* 14 (2005) 3. 383–391. ISSN 0926-5805. https://doi.org/10.1016/j.autcon.2004.08.005

Li et al. 2025a — Li, Shuai – Lan, Tian – Nguyen, Hung-Xuan – Tran, Phuong: Frontiers in construction 3D printing: Self-monitoring, multi-robot, drone-assisted processes. *Progress in Additive Manufacturing* 10 (2025) 2001–2030. https://doi.org/10.1007/s40964-024-00794-8

Li et al. 2025b — Li, Shoufu – Yu, Tao – Liu, Yyang – Li, Yan – Lei, Bo – Qiao, Wenjing – Zhang, Yuyan: Research on Parametric Modeling and Model Transformation of Proton Hospital Based on BIM. *Buildings* 15 (2025) 10. 1658. https://doi.org/10.3390/buildings15101658

Maher–Burry 2003 — Maher, Andrew – Burry, Mak C.: The Parametric Bridge: Connecting Digital Design Techniques in Architecture and Engineering. In *Crossroad of Digital Discourse. Proceedings of the 2003 Annual Conference of the Association for Computer-Aided Design in Architecture*. Indianapolis (Indiana) 24–27 October 2003. (ACADIA) 39–47. https://doi.org/10.52842/CONF.ACADIA.2003.039

Meng–Gardner 2025 — Meng, Xin – Gardner, Leroy: Standardisation framework for metal additive manufacturing in construction. *Automation in Construction* 176 (2025) 106–267. ISSN 0926-5805. https://doi.org/10.1016/j.autcon.2025.106267

Santhosh–Raphael–Santhanam 2025 Santhosh, S. Gokul – Raphael, Benny – Santhanam, Manu: Concrete 3D Printing of Shape-Optimized Lattice Beams Incorporating Nature-Inspired Patterns. *Journal of Architectural Engineering* 31 (2025) 3. 04025025. DOI: https://doi.org/10.1061/JAEIED.AEENG-1942

Shihai et al. 2023. 107156. Shihai Wu – Wei-Zhen Lu – Chao Qin – Baohua Wen – Dizi Wu – Yujing Xiang: A goal-oriented planning approach for two-dimensional cutting components in architectural design: Coupling BIM and Parametric design (PD). *Journal of Building Engineering* 76 (2023) 3. 107156. DOI: https://doi.org/10.1016/j.jobe.2023.107156

Thajeel–Balázs 2025 Thajeel, Marwa M. – Balázs, György L.: 3D Printed Concrete: Fresh and Hardened Properties, *Periodica Polytechnica Civil Engineering* 69 (2025) 1. 12–27. https://doi.org/10.3311/PPci.37602

Tóth–Bruggi–Lógó 2023 Tóth, Bálint – Bruggi, Matteo – Lógó, János: Comparative Study on the Optimal Topologies [Összehasonlító tanulmány optimális topológiákra]. *Építés – Építészettudomány* 51 (2023) 3–4. 195–213. https://doi.org/10.1556/096.2023.00094

MELLÉKLETEK

A) melléklet: Python kód Grasshopper Python komponenséhez

```
"""
Projekt: Geometriai adatok exportálása Grasshopperből
Szerző: Tóth Bálint, Bükkösi Raymond, Lógó János
Dátum: 2025.06.11.

Ez a kód oktatási célokra készült, és bemutatja, hogyan lehet a Rhino-Grasshopper
környezetből származó geometriai adatokat (vonalak és poligonok) csomópontokra és
rúdelemekre bontani, majd egyszerű szöveges fájlokba exportálni.
A kód Python nyelven íródott, a Rhino.Geometry és a scriptcontext könyvtárak fel-
használásával.
A kód részletes magyar nyelvű kommentárokat tartalmaz a könnyebb megértés érdeké-
ben.

A kód szabadon felhasználható oktatási célokra.
"""

# Rhino.Geometry és scriptcontext könyvtárak importálása
import Rhino.Geometry as rg
import scriptcontext as sc
import os

# Feltételezzük, hogy a Grasshopper környezetben a 'lines' bemenetben szereplő
GUID-kat kapjuk meg
# Ezeket átalakítjuk Rhino geometriai objektumokká
gh_geos = [sc.doc.Objects.Find(x).Geometry for x in lines]

# Inicializáljuk a csomópontok listáját és a rúdelemek listáját
nodes = []          # Csomópontok listája
node_indices = {}   # A csomópontok egyedi indexelése a duplikációk elkerülésére
elements = []       # Rúdelemek listája (start_idx, end_idx)
```

```
# Segédfüggvény: egy adott pontot felvesz a nodes listába, ha még nincs benne
# A pont koordinátái 6 tizedesjegyre kerekítve kerülnek összehasonlításra a pontosság miatt
def get_or_add_node(pt):
    key = (round(pt.X, 6), round(pt.Y, 6), round(pt.Z, 6))
    if key not in node_indices:
        node_indices[key] = len(nodes)
        nodes.append(pt)
    return node_indices[key]

# A beérkező geometriai objektumok feldolgozása
for geo in gh_geos:
    if isinstance(geo, rg.LineCurve):
        # Ha a geometria egy egyenes vonal (LineCurve), akkor kinyerjük a kezdő- és végpontját
        pt1 = geo.PointAtStart
        pt2 = geo.PointAtEnd
        start_idx = get_or_add_node(pt1)
        end_idx = get_or_add_node(pt2)
        elements.append((start_idx, end_idx))
    elif isinstance(geo, rg.PolylineCurve):
        # Ha a geometria egy poligon (PolylineCurve), akkor végigmegyünk az összes szegmensén
        pl = geo.ToPolyline()
        for i in range(pl.Count - 1):
            pt1 = pl[i]
            pt2 = pl[i + 1]
            start_idx = get_or_add_node(pt1)
            end_idx = get_or_add_node(pt2)
            elements.append((start_idx, end_idx))
    else:
        # Egyéb típusokat figyelmen kívül hagyunk (például felületek)
        pass

# A kimeneti fájlokat az Asztalon hozzuk létre, egy "Examples" nevű mappában
desktop = os.path.join(os.path.expanduser("~"), "Desktop")
examples_dir = os.path.join(desktop, "Examples")
if not os.path.exists(examples_dir):
    os.makedirs(examples_dir)

# A csomópontokat tartalmazó és az elemeket tartalmazó szöveges fájlok elérési útvonalai
node_file = os.path.join(examples_dir, "nodes.txt")
elem_file = os.path.join(examples_dir, "elements.txt")

# A csomópontok fájlba írása
# A sorok formátuma: index X Y Z
with open(node_file, "w") as f:
    for i, pt in enumerate(nodes):
        f.write("{0} {1:.3f} {2:.3f} {3:.3f}\n".format(i+1, pt.X, pt.Y, pt.Z))

# A rúdelemek fájlba írása
# A sorok formátuma: index start_node_index end_node_index
with open(elem_file, "w") as f:
    for i, (n1, n2) in enumerate(elements):
        f.write("{0} {1} {2}\n".format(i+1, n1+1, n2+1))

# Visszajelzés a felhasználónak: hány csomópont és hány rúdelem lett exportálva
a = "{} csomópont és {} rúdelem exportálva.".format(len(nodes), len(elements))
```

B) melléklet: Matlab kód exportált fájlok beolvasásához és tengelyek megjelenítéséhez

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Projekt: Live Structure Viewer
% Szerző: Tóth Bálint, Bükkösi Raymond, Lógó János
% Dátum: 2025.06.11.
%
% Ez a kód oktatási célokra készült, és bemutatja, hogyan lehet Matlab
% környezetben folyamatosan figyelni a Rhino-Grasshopper környezetből
% exportált geometriai adatokat (csomópontokat és rúdelemeket) és
% azokat élőben megjeleníteni.
%
% A kód részletes magyar nyelvű kommentárokat tartalmaz a könnyebb
% megértés érdekében.
%
% A kód szabadon felhasználható oktatási célokra.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function live_structure_viewer()
    % === 1. Állományok előkészítése ===
    % Meghatározzuk az Asztal (Desktop) mappát a felhasználó profiljából
    desktop = fullfile(getenv('USERPROFILE'), 'Desktop');
    dataFolder = fullfile(desktop, 'Examples');
    nodeFile = fullfile(dataFolder, 'nodes.txt');   % Csomópontok fájl
    elemFile = fullfile(dataFolder, 'elements.txt');% Rúdelemek fájl

    % === 2. Ábraablak létrehozása ===
    fig = figure('Name', 'Live Structure Viewer');
    refreshPlot();  % Első frissítés (kezdeti megjelenítés)

    % === 3. Utolsó módosítási időpontok tárolása ===
    % Ezeket figyeljük a fájlok frissítéséhez
    lastNodeTime = getModifiedTime(nodeFile);
    lastElemTime = getModifiedTime(elemFile);

    % === 4. Időzítő objektum létrehozása ===
    % Minden 2 másodpercben ellenőrizzük a fájlokat
    t = timer( ...
        'ExecutionMode', 'fixedRate', ...
        'Period', 2.0, ...      % Ellenőrzési periódus (másodperc)
        'TimerFcn', @checkAndRefresh);  % Fő ellenőrző függvény

    start(t);  % Indítjuk az időzítőt

    % === 5. Ha bezárjuk az ábrát, leállítjuk az időzítőt és töröljük az erőforrá-
sokat ===
    fig.CloseRequestFcn = @(src, event) stopAndDeleteTimer(t, fig);

    % === 6. Beágyazott (nested) függvények ===

    % 6.1. Fájlmódosítás-ellenőrző függvény
    function checkAndRefresh(~, ~)
        newNodeTime = getModifiedTime(nodeFile);
        newElemTime = getModifiedTime(elemFile);
```

```
        if newNodeTime ~= lastNodeTime || newElemTime ~= lastElemTime
            lastNodeTime = newNodeTime;
            lastElemTime = newElemTime;
            refreshPlot();  % Ha változás van, frissítjük az ábrát
        end
    end

    % 6.2. Ábrafrissítő függvény
    function refreshPlot()
        try
            clf(fig);           % Ábraablak törlése (újrarajzolás előtt)
            hold on;
            axis equal;
            xlabel('X'); ylabel('Y'); zlabel('Z');
            title('Live Structure Viewer');

            % 6.2.1. Fájlok betöltése
            nodes = load(nodeFile);         % Csomópontok betöltése
            elements = load(elemFile);      % Rúdelemek betöltése
            coords = nodes(:, 2:4);         % Csomópontok XYZ koordinátái

            % 6.2.2. Rúdelemek kirajzolása (kék vonal)
            for i = 1:size(elements, 1)
                n1 = elements(i, 2);
                n2 = elements(i, 3);
                pt1 = coords(n1, :);
                pt2 = coords(n2, :);
                plot3([pt1(1), pt2(1)], [pt1(2), pt2(2)], [pt1(3), pt2(3)], 'b-',
'LineWidth', 0.5);
            end

            % 6.2.3. Csomópontok kirajzolása (piros pont)
            scatter3(coords(:,1), coords(:,2), coords(:,3), 3, 'ro', 'filled');
            view(30, 30);    % 3D nézet beállítása
            grid on;
        catch err
            disp(['Hiba a frissítés során: ', err.message]);
        end
    end

    % 6.3. Fájl módosítási idejének lekérdezése
    function t = getModifiedTime(file)
        if exist(file, 'file')
            info = dir(file);
            t = info.datenum;
        else
            t = 0;  % Ha nem létezik a fájl
        end
    end

    % 6.4. Időzítő és ábraablak leállítása és törlése
    function stopAndDeleteTimer(timerObj, figHandle)
        stop(timerObj);     % Időzítő leállítása
        delete(timerObj);   % Időzítő törlése
        delete(figHandle);  % Ábraablak bezárása
    end
end
```

Parametric Design in the Construction Industry: Concepts, Tools, Case Studies

SUMMARY

Since the end of the 20th century, computers have played an increasingly significant role in construction and architectural design. Throughout various design processes, numerous new concepts have been introduced—such as digital, computational, algorithmic, generative, and parametric design—which often overlap and whose definitions in the literature still require continuous refinement. The aim of this study is to present and systematize these concepts and to examine their practical applications through case studies. The paper discusses the most commonly used software platforms (Tekla, Revit, Allplan, Grasshopper) and their integration, with special emphasis on connecting parametric design with numerical computational environments. The case studies include the rationalization of a stadium facade cladding system, the supporting system required for the free-standing assembly of a complex building, and the implementation possibilities of a Grasshopper–Matlab data link. The study highlights how the integration of digital and computational tools can contribute to increasing the efficiency of the design and the execution processes.

KEYWORDS

computer-aided design, digital design, computational design, algorithmic design, parametric design, generative design, Grasshopper, BIM, geometric rationalization, data linkage