Grid rendszerek használata vasbeton hídgerendák tervezésében

PASZTUHOV DÁNIEL, SZEBERÉNYI IMRE
BME Irányítástechnika és Informatika Tanszék
{dani, szebi}@iit.bme.hu
SIPOS ANDRÁS ÁRPÁD
BME Szilárdságtani és Tartószerkezeti Tanszék
siposa@silver.szt.bme.hu

Lektorált

Kulcsszavak: grid, portál, ipari alkalmazás, vasbeton gerendák, párhuzamos számítás

Cikkünkben bemutatunk egy valós, mérnöki feladatot és annak megoldását egy hatékony párhuzamos algoritmussal, valamint egy – felhasználói felületek előállítását hatékonyabbá tevő – webes eszközt is, melynek segítségével kényelmes, webes felületen indíthatók párhuzamos és grid-feladatok, valamint lehetőség van parancssoros programok indítására is.

1. Bevezetés

Bár a grid nagy számítási kapacitása miatt az ipari alkalmazások széles körében alkalmazható lenne, a használat nehézkessége eddig meggátolta az ilyen jellegű felhasználás elterjedését. Cikkünkben bemutatunk egy hatékony párhuzamos algoritmust, amellyel egy mérnöki probléma megoldása számítható. Az algoritmus alkalmas arra, hogy a számításokat grid-környezetben hajtsuk végre. A hozzá tartozó felhasználói felület lehetővé teszi az ipari felhasználók számára az eljárás egyszerű használatát. Segítségével lehetőség van a szükséges adatok bevitelére, azok fájlba történő elmentésére, korábbi adatfájlok beolvasására és természetesen a számítás elindítására, valamint az eredmények letöltésére is. A felületet létrehozó konfigurálható portlet egy újonnan kifejlesztett nyelvet használ a felület működésének leírására. A keretrendszer alkalmas arra, hogy a parancssoros adatbeviteltől a grafikai megoldásokig az alkalmazások széles körét biztosítsa a felhasználói felület számára. Megközelítésünk konfigurálhatóság szempontjából eltér a Grid környezetekhez fejlesztett portálrendszertől (Genius [16,17], P-GRADE [18], BIRN [19], LEAD [20]).

A párhuzamos algoritmus előfeszített vasbetongerendák, elsősorban hídgerendák térbeli alakváltozásainak számítására szolgál. A probléma erősen nemlineáris jellege – a geometriai és anyagi nemlinearitás egyszerre befolyásolja az egyensúlyi alakot – ellenére az eljárás megbízható, nem kell tartani hamis megoldásoktól vagy divergens viselkedéstől. A szakirodalomban [5-7] eddig javasolt eljárások bizonyítottan vezethetnek hamis megoldásra, vagy produkálhatnak kaotikus viselkedést. Módszerünk megbízhatóságának ára a nagy számításigény, hiszen pontos eredményhez mintegy 1 millió rúdalakot kell kiszámítani.

Cikkünk második fejezetében röviden ismertetjük az algoritmust, kiemelve a vasbeton rudak deformációinak számításában rejlő nehézségeket, az eljárás részletes ismertetése több publikációban is megtalálható [1-3]. A harmadik fejezet a felhasználói felülettel szemben támasztott követelményeket tartalmazza, a negyedik fejezetben pedig az új Conflet keretrendszert mutatjuk be, mely alkalmas más eljárások hasonló igényeinek gyors és hatékony kielégítésére.

2. Vasbeton hídgerendák térbeli deformációjának számítása

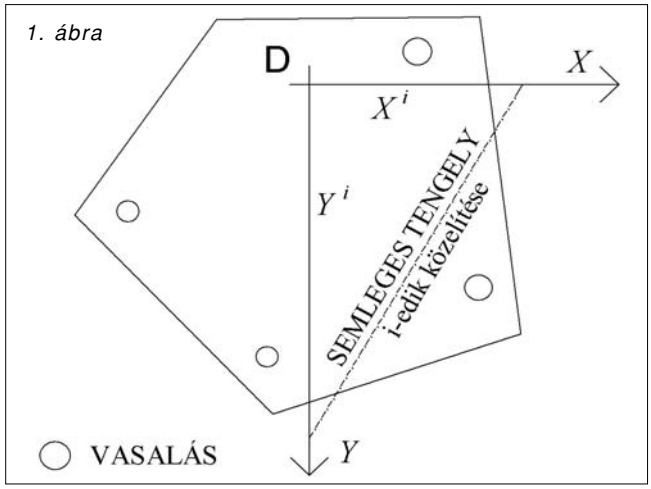
A bevezetőben említett algoritmus vasbetongerendák és -oszlopok térbeli deformációit határozza meg az anyag és a geometria nemlineáris viselkedésének figyelembe vételével. A számítás során a nyíró- és a normálerőkből származó alakváltozásokat elhanyagoljuk, továbbá feltesszük, hogy a sík keresztmetszetek a deformáció után is síkok maradnak.

A rúd azon tartományát, ahol csak nyomófeszültségek ébredhetnek, *betonnak* nevezzük, azon tartományokat pedig, ahol húzófeszültségek is ébredhetnek, *acélnak* hívjuk. A keresztmetszet alakja és a vasak helyzete tetszőleges *(1. ábra)*.

2.1. A keresztmetszet görbületének számítása

A sík keresztmetszetek törvénye miatt a hajlítás hatására a keresztmetszet egy egyenes, a semleges tengely körül fordul el. Célunk a semleges tengely, a gör-

1. ábra



bület és az elcsavarodás számítása. A rúd egy keresztmetszetének igénybevétele tipikusan külpontos nyomás és egyidejű csavarás. A semleges tengely meghatározása egy nem-lineáris egyenletrendszer megoldását követeli meg, hiszen a húzott oldalon a beton bereped, vagyis a dolgozó keresztmetszetet az ismeretlen semleges tengely határolja. Egy, az egyensúlyi egyenletekből származtatott kétdimenziós leképzéssel a semleges tengely helye kevés, 10-15 iterációs lépésen belül egyértelműen meghatározható. A lineáris anyagtörvényhez tartozó leképzés származtatásáról, illetve a konvergenciájának bizonyításáról [1,3,4] számol be részletesen. A beton valóságos viselkedését sokkal pontosabban modellező, fellágyuló anyagtörvényhez is definiálható egy konvergens iteráció, mely egyértelműen meghatározza a semleges tengely helyét és a görbületet.

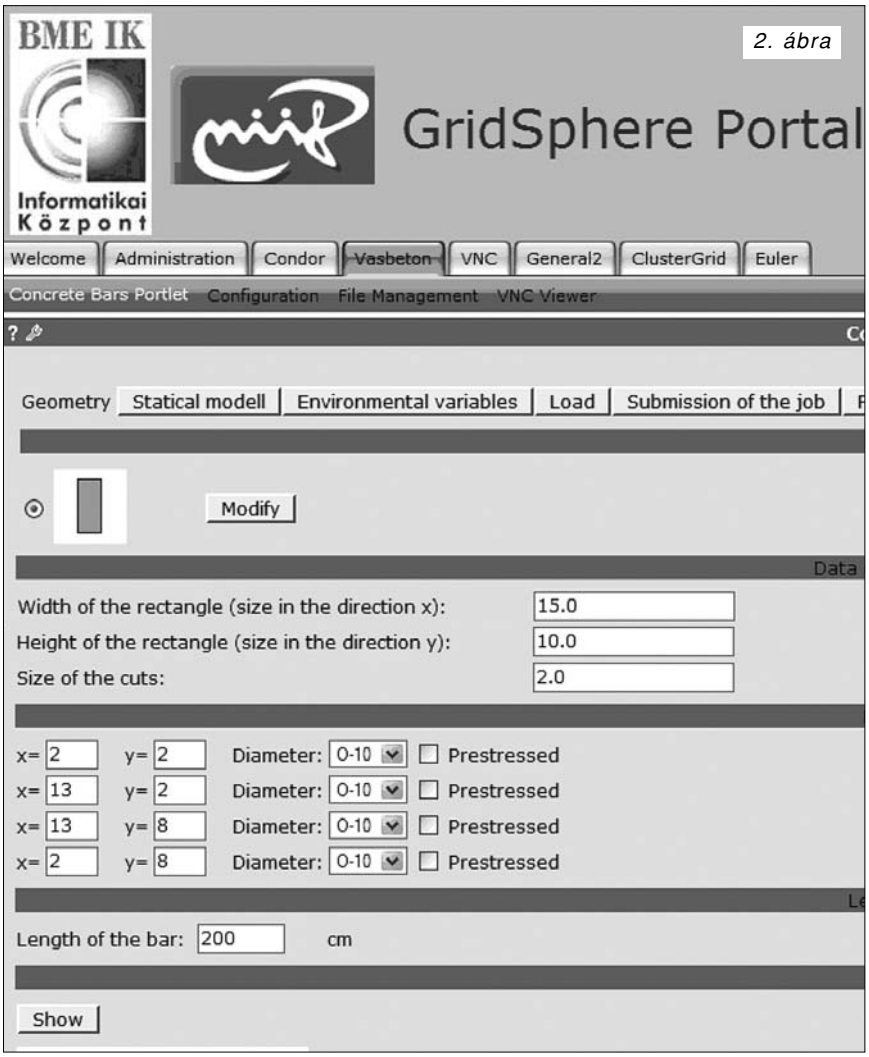
Vasbeton rudak számításánál nem csupán a beton anyagtörvényének nemlinearitását kell figyelembe venni, hanem a beton zsugorodását és kúszását is. A zsugorodási többletgörbületet az EUROCODE szabvány (EC2) [15] alapján közvetlenül lehet számítani. A kúszást a kúszási tényező segítségével számított effektív rugalmassági modulussal vesszük figyelembe. Pontosabb számításokhoz lehetőség van az úgynevezett Trost-féle eljárás használatára. A kúszás és a zsugorodás számítása az algoritmus megbízhatóságát nem befolyásolja. Az előfeszítés kapcsán a megengedhető feszítőerőt, illetve a feszültség-veszteségeket szintén az EC2 szabvány szerint számítjuk.

2.2. A rúd alakjának számítása

A rúd deformációit a görbület és az elcsavarodás hosszmenti integrálásával kaphatjuk meg, amennyiben a rúd egyik végének térbeli helyzetét és az ott működő erőket és nyomatékokat ismerjük. Ebben az esetben *kezdetiérték-feladatról* beszélünk. A mérnöki gyakorlatban tipikusan a rúd mindkét végén vannak ismeretlen geometriai vagy statikai mennyiségek. Ekkor egy *peremérték-feladatot* kell megoldanunk, mely jóval nagyobb számítási kapacitást igényel, mint egy kezdeti-érték feladat kiszámítása.

A peremérték-feladat megoldására kézenfekvő numerikus eljárás a szimplex-módszer [4,8,9], amely nagyszámú kezdetiérték-feladat megoldására vezeti vissza a problémát. Egyetlen rúdból álló szerkezetek esetén változók a rúd egyik végén a peremfeltételek által nem rögzített alakváltozási jellemzőket *változóknak* nevezzük. A rúd másik végén a peremfeltételek által előírt mennyiségeket *függvényeknek* nevezzük. A megoldásokat a változók és a teherparaméter d=n+1 dimenziós terében keressük, amelyet a továbbiakban a probléma *Globális Reprezentációs Terének* (GRS) nevezünk. A GRS egy-egy pontja tehát egy-egy kezdetiérték-feladatnak felel meg és ezek között keressük a vizsgált mechanikai probléma peremérték-feladatának megoldásait. A szimplex módszernél első lépésben a GRS-t diszkretizáljuk egy szimplex-hálóval. (A fenti példa 2D-s GRS-ében ez háromszögekre való felosztást jelent.) Ezután kiintegráljuk a kezdetiérték-feladatot a szimplexek összes csúcspontjában, meghatározva a függvények e pontbeli értékét. A szimplexek belsejében lineáris interpolációt alkalmazva keressük a feladat megoldásait, azon pontokat, ahol minden függvény értéke zérus. Itt tehát egy egyszerű lineáris egyenletrendszert oldunk meg.

A példa teljes megoldásához a GRS összes szimplexében el kell végezni a számítást, ami rendkívül munkaigényes, a számítási igény GRS dimenziójával exponenciálisan nő. Ezért érdemes a módszert nagy teljesítményű, párhuzamos, illetve Grid technológiájú rendszerekre implementálni. A szimplex módszer különösen alkalmas erre, mivel a számításokat szimplexenként külön, egymástól függetlenül lehet végezni, tehát a számítási munka kis, önálló egységekre bontható.



2. ábra

3. A felhasználói felülettel szemben támasztott követelmények

Ahhoz, hogy egy „valós életből” vett gerendát számolni tudjunk, megközelítőleg 200-1000 paraméter megadására van szükség. A paraméterek száma alapvetően a keresztmetszet bonyolultságától függ *(2. ábra)*. A bevitt koordináta-adatoknak megfelelő alakzatot felhasználói ellenőrzés céljából a képernyőn meg kell jeleníteni.

Az anyagi jellemzők megfelelnek az EC2 szabványnak, azaz a felhasználónak választhat a különféle minőségű betonok, acélok és előfeszítő pászmák közül. A felhasználói felület a megfelelő anyagjellemzőket a kiválasztott minőséghez rendeli hozzá. További paraméterek adják meg környezeti jellemzőket (például relatív páratartalom, a szerkezet életkora megterheléskor stb.) Szükséges továbbá a szerkezet terheinek (koncentrált és megosztó terhelést) bevitele.

Ha az összes paramétert megadtuk, a feladat elindítható, vagy a paraméterkészlet fájlba menthető, hogy onnan visszaolvashassuk egy későbbi feladatindítás érdekében. Indítás előtt a Globális Reprezentációs Tér adatai módosíthatóak.

4. A Conflet keretrendszer

A felhasználói felület a Conflet (CONFigurable portLET – konfigurálható portlet) rendszerrel készült. A Conflet rendszer egy egyszerűen használható keretrendszer az alkalmazásfejlesztők számára, hogy segítségével felhasználóbarát felületeket készítsenek grid-környezetben futó feladatok indításához, lecsökkentve ezzel a fejlesztésre fordított időt.

A rendszer arra való, hogy oldalakat, oldalcsoportokat jelenítsen meg és beolvassa róla a feladat különböző paramétereit, elvégezzen egyszerű számításokat (ideértve egy kép generálását), fájlokat hozzon létre és végül grides feladatot (vagy parancssoros távoli programot) indítson. Sok ilyen típusú feladat létezik, így a Conflet fő célkitűzése, hogy minimalizálja a fejlesztő által elvégzett munkát.

A Conflet a nyílt forráskódú, ingyenes GridSphere Portál Keretrendszer [10,11] szolgáltatásaira épít, de olyan rugalmasra lett tervezve, hogy bármely más (nem feltétlenül portál, de mindenképpen Java alapú) keretrendszerrel együtt tudjon működni.

Egy Conflettel létrehozott feladatindító alkalmazás futás közben is megváltoztatható a konfigurációs fájlok (azaz azon fájlok, amik az alkalmazás kinézetét és viselkedését befolyásolják) egy halmazának feltöltésével. A két fő fájltípust *view*-nak és *controller*-nek nevezzük. Egy view határozza meg egy oldal kinézetét, megvalósítását tekintve JSP fájl. A controller határozza meg egy oldal viselkedését, azaz azon akciókat, melyeket a portlet végrehajt egy gomb megnyomására vagy egy link meghívására. Mindkét fájltípusból több példány létezhet és ezen példányok nincsenek összekötve sem: az aktuális view megváltozhat az aktuális controller megváltoztatása nélkül és fordítva is, valamint egy view több controllerhez is tartozhat és egy controller több view-nak is lehet a párja.

4.1. Controller és View

A controllerek megalkotásához létrehoztunk egy XML-alapú vezérlő nyelvet. A controller nyelv string változókat és egyszerű vezérlési szerkezeteket (elágazás, ciklus) használ.

Egy Conflettel előállított felhasználói felületben több view (Java Server Pages (JSP) [14] és GridSphere UI Tab Library) segítségével létrehozott oldal is létezhet. A különféle oldalakat csoportosíthatjuk, úgynevezett fülekbe szervezhetjük. A fülek oldalon belüli pozíciója egy saját jelölővel adható meg a Conflet Tag Library-ből. Egy példa controller és view látható a *3. és 4. ábra*. Az utóbbi a 2. ábra középső részének leírása.

A view és controller fájlok főbb jellemzői a következők:

- Karakterlánc típusú változók definiálhatók és használhatók a view és controller fájlokban. A változók könnyen összefűzhetők egymással vagy egy konstanssal.
- Az űrlapelemek értéke a rendszerben változóként jelennek meg, míg tulajdonságaik controller parancsokkal állíthatók be. Az űrlapelemek értékei fájlba menthetők, és visszatölthetők onnan.

3. ábra

```
<?xml version="1.0"?>
<actions version="1.0.1"
  xmlns:condor="http://n0.iit.bme.hu/gridsphere/condor.xsd"
  xmlns:ssh="http://n0.iit.bme.hu/gridsphere/ssh.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="leiro.xsd">
  <init>
    <for var="i" begin="1" end="${rf_n}">
      <listbox beanId="rf_${i}_dia" file="rfdia.dat"
selectedid="6"/>
      <checkbox beanId="psrf_${i}" selected="false"/>
    </for>
  </init>
  <default>
    <if>
      <or-each var="i" begin="1" end="${rf_n}">
        <string string="${psrf_${i}}"/>
      </or-each>
      <then>
        <enable-tab tabid="prestress"/>
      </then>
      <else>
        <disable-tab tabid="prestress"/>
      </else>
    </if>
  </default>
  <action name="show">
    <generate-postfix var="pngpostfix"/>
    <function
class="hu.bme.iit.gridsphere.vasbeton.geometria.Geometria"
jar="vasbeton.jar" output="cache/geomkep${pngpostfix}.png">
      <input>
        <line>pic</line>
        <line>rectangle</line>
        <line-each var="i" begin="1" end="${paramnum}">
          ${param_${i}}
        </line-each>
      </input>
    </function>
    <image beanId="ready"
src=".${user_name}/cache/geomkep${pngpostfix}.png"/>
  </action>
  <action name="change">
    <next view="geomselect.jsp" ctl="geomselect.xml"/>
  </action>
  <action name="next">
    <next/>
  </action>
</actions>
```

```
<ui:frame>
  <ui:tablerow header="true">
    <ui:tablecell>
      <conflet:message key="DATA_OF_CROSS_SECTION"/>
    </ui:tablecell>
  </ui:tablerow>
</ui:frame>
<ui:frame>
  <ui:tablerow>
    <ui:tablecell width="350">
      <ui:hiddenfield beanId="paramnum" value="3"/>
      <conflet:message key="WIDTH_OF_RECTANGLE"/>:
    </ui:tablecell>
    <ui:tablecell>
      <ui:textfield beanId="param_1" value="15.0"/>
    </ui:tablecell>
  </ui:tablerow>
  <ui:tablerow>
    <ui:tablecell width="350">
      <conflet:message key="HEIGHT_OF_RECTANGLE"/>:
    </ui:tablecell>
    <ui:tablecell>
      <ui:textfield beanId="param_2" value="10.0"/>
    </ui:tablecell>
  </ui:tablerow>
  <ui:tablerow>
    <ui:tablecell width="350">
      <conflet:message key="SIZE_OF_CUTS"/>:
    </ui:tablecell>
    <ui:tablecell>
      <ui:textfield beanId="param_3" value="0.0"/>
    </ui:tablecell>
      </ui:tablerow>
</ui:frame>
```

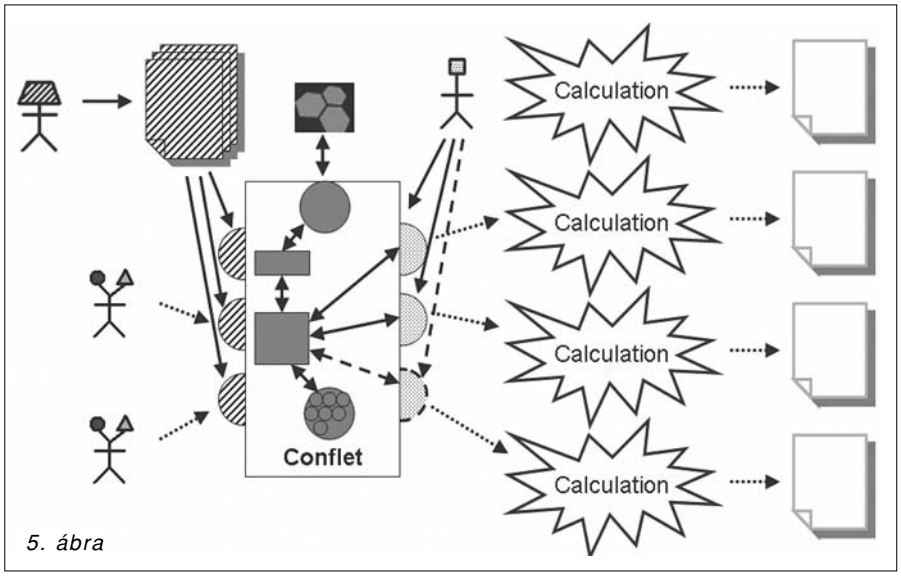
4. ábra

- Fájlok hozhatók létre, tölthetők le és fel a távoli gépre vagy akár az alkalmazásszerver gépére. A fájlok tartalma reguláris kifejezésekkel változókba tehető. A fájl feldolgozásához és írásához ciklusok definiálhatók.
- Névvel ellátott változócsoportok hozhatók létre és tárolhatók XML-fájlokban. A csoportok nevei listadobozba tölthetők, míg a kiválasztott nevű csoportba tartozó változók egy utasítással aktiválhatók.
- A controllerhez Javaban kiterjesztéseket írhatunk, melyek segítségével egyszerű számításokat végezhetünk el, vagy akár képeket hozhatunk létre. A futtatáshoz a Java Security Managert használjuk. A kiterjesztések az archívumnév (JAR), az osztály neve, valamint a ki- és bemenet megadásával hívhatók meg.
- Akciók (parancssorozatok) rendelhetők nemcsak egyes konkrét felhasználói eseményekhez, hanem akár egy oldal első betöltődéséhez (init), vagy egy tetszőleges akció lefutásához (default). Az akciók egymásból is meghívhatók.
- A rendszerhez beépülő modulok (plugin) segítségével új middleware-t, új storage-et és új, távoli helyen parancsot futtatni képes modult (accessor) tudunk adni. A beépülő modulok új parancsokat is tartalmazhatnak.
- A parancsnyelv alapparancsai is könnyedén kiterjeszthetők. Az alapparancsokat tartalmazó modul (Conflet Language Bundle, CLB) könnyedén lecserélhető, tartalma különféle adminisztrációs fájlok módosítása nélkül megváltoztatható. A CLB nem csak új alapparancsokat, hanem új konfigurációs fájltípusokat is tartalmazhat, így a rendszer – bizonyos keretek közt – rugalmasan bővíthető.
- A controllerben lévő hibák a Java stack trace-hez hasonlóan jelennek meg.

4.2. Architektúra

A Conflet többrétegű architektúrát használ, melyet a lenti, *5. ábra* ábrázol.

- A *user interface* modul (a Conflet téglalapján belül a legfelső, szürke kör) köti össze a portál motort (vagy más, megjelenítéshez használt Java környezetet) a rendszer központi moduljával a „Conflet interface” modulon keresztül. Feladata, hogy a fő modulból származó oldalakat megjelenítse, és a felhasználói interakciókat a fő modulhoz visszairányítsa.
- A *Conflet interface* modul (a Conflet téglalapján belül a felső lapos téglalap) definiálja azokat az interfészeket és absztrakt osztályokat, melyek a „user interface” modul és a fő modul közötti interakcióhoz használhatók. (Nem tekintjük külön rétegnek).
- A *fő modul* (a Conflet téglalapján belüli négyzet) keretszolgáltatásokat nyújt a CLB-nek, úgy mint parancs- és konfigurációs fájl-osztályok betöltése, fájlok betöltése, az adatok kezelése, kommunikáció és a CLB elemeinek meghívása.
- A *Conflet Language Bundle* (a Conflet téglalapján belüli kis körökkel teli kör) tulajdonképpen parancsosztályok és konfigurációs fájl-osztályok halmaza kiegészítve néhány segédosztállyal, melyek a CLB koherenciáját hivatottak biztosítani. A controller parancsait a CLB osztályai valósítják

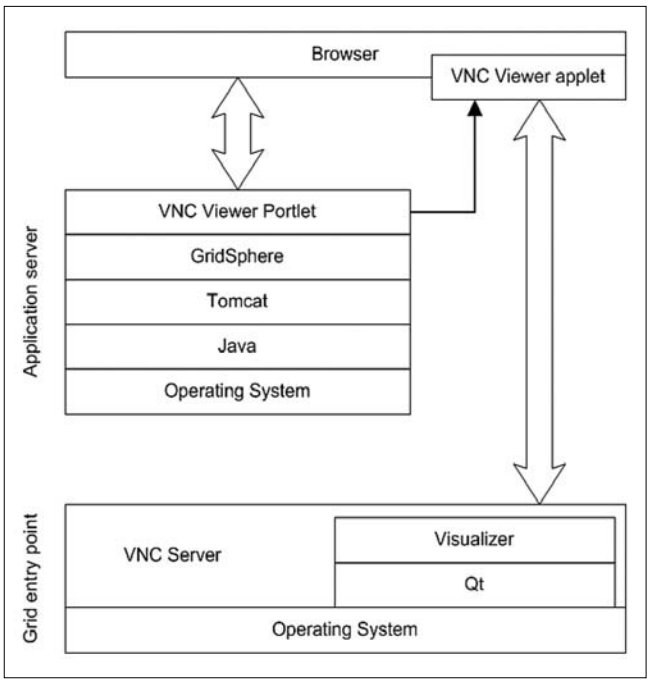


5. ábra

meg, a konfigurációs fájl-osztályok pedig az egyes típusok (például view, controller stb.) működését biztosítják.
- *Plugin modulok* (a Conflet téglalapjának jobb oldalán lévő félkörök) definiálhatók, hogy hozzáférést biztosítsunk a különféle külső szolgáltatásokhoz, mint a grid, egy cluster, távoli storage-ek, vagy program-végrehajtók. A Plugin modulokban további parancsok lehetnek.

A ferdén csíkos részek a konfiguráció menetét jelölik. A trapézfejű pálcikaember (konfigurátor) elkészíti a konfigurációs fájlokat, melyek egy-egy felületként jelennek meg a felhasználó szemszögéből.

A pöttyözött vonal a feladat indításának menetét szemlélteti: a kétfejű felhasználó kapcsolatba lép a Conflet egyik felületével, majd az – a beépülő modulokon keresztül – elindítja a feladatot, ami vizualizálható eredményt ad.



6. ábra

5. Megjelenítés

Az eredmények megjelenítéséhez két feladatot kellett megoldani. Egyrészt egy szoftvereszközt kellett kifejleszteni, amely képes megjeleníteni grafikusan a számítás eredményét, másrészt pedig a kifejlesztett szoftvereszközt integrálni kellett a portál architektúrájába. A vizualizáció architektúrája a *6. ábrán* látható.

5.1. Az eredmények megjelenítéséhez használt alkalmazás

A számolást végző algoritmus eredménye a Globális Reprezentációs Térben a bifurkációs diagram, amit a megjelenítő eszköz két- vagy háromdimenziós formában megjelenít. Ha kiválasztjuk a diagram egy pontját, az alkalmazás kirajzolja a hozzá tartozó rúdalakot, valamint megadja a maximális vízszintes és függőleges lehajlást és elfordulást. Ez azt jelenti, hogy a ponthoz tartozó egyetlen kezdetiérték-feladatot a munkaállomás megoldja, ehhez nem szükséges a párhuzamos környezet.

Az alkalmazást a platformfüggetlen Qt [12] keretrendszer segítségével implementáltuk. A megjelenítő bármely szabványos X-Window vagy Windows környezetben képes futni. Mivel a futtatásra szolgáló távoli gép általában szabványos UNIX környezet, kézenfekvő a programot ott futtatni. Alternatíva, hogy a fájlokat letöltve a felhasználó a saját munkaállomásán jeleníti meg az eredményeket. Előbbi esetben megoldandó probléma volt, hogy miként jelenítsük meg a távoli gép képét a felhasználó számítógépén úgy, hogy lehetőleg ne kelljen semmiféle programot telepítenie. A probléma megoldására fejlesztettük ki a VNC Viewer portletet.

5.2. VNC Viewer portlet

A VNC (Virtual Network Computing) [13,14] lehetővé teszi, hogy a felhasználó grafikus kapcsolatba lépjen bármely géppel az interneten. Egy általános célú grafikus felhasználói felületet biztosít állapotmentes protokoll fölött. A szoftvercsomag két komponensből áll: a szerver a távoli gépen fut, fenntartja a kapcsolatot a kliens és a grafikus környezet között, vagy grafikus szolgáltatásokat nyújt; a kliens (viewer) pedig megjeleníti a munkakörnyezet képét a felhasználó képernyőjén. A kliensnek különféle verziói léteznek, van többek között Java applet verziója is, ami szempontunkból a legfontosabb változat; ezt integráltunk a portálunk környezetébe.

Az elkészült portáloldal a következő szolgáltatásokat nyújtja:
- A távoli gépen VNC munkaasztalok (szerverek) hozhatók létre és törölhetők, valamint csatlakozhatunk hozzájuk.
- Úgynevezett program profilok hozhatók létre, melyek a későbbiekben lehetővé teszik, hogy a távoli munkaasztalunkon mindössze három egérkattintással új programot indíthassunk.

Biztonsági szempontok miatt a VNC jelszó alapú azonosítást használ. Ahhoz, hogy az egyszeres bejelentkezés (single sign-on) követelményeit teljesíteni tudjuk, ezt a tulajdonságot a biztonság szem előtt tartása mellett át kellett alakítanunk, lehetővé kellett tennünk, hogy a felhasználók újabb jelszó beírása nélkül tudjanak VNC szerverükhöz kapcsolódni úgy, hogy azt más az interneten ne tehesse meg.

6. Összefoglalás

Cikkünkben bemutattunk egy globálisan konvergens algoritmust vasbeton hídgerendák térbeli deformációjának számítására. Az algoritmushoz illeszkedő felhasználói felület lehetővé teszi a Grid rendszerek ipari alkalmazását. Bemutattunk továbbá egy, az eredmények megjelenítésére alkalmas eszközt is.

A fejlesztő szemszögéből tekintve a Java/J2EE, XML és GridSphere Portál Keretrendszer technológiákon alapuló alkalmazásfejlesztési rendszer segítségével új fel-

használói felületeket lehet készíteni parancssoros programokhoz a bináris módosítása nélkül. Grid környezetben hasznos eszköz lehet feladatindító portletek létrehozására. Mivel magasszintű parancsokból álló parancsnyelvet használ, a fejlesztés gyorsabbá és hatékonyabbá válik, ráadásul nem kell Java-ban programozni, fordítani és adott esetben szervert újra indítani.

Cikkünkben ismertettük, hogyan definiálhatók felhasználói felületek a kifejlesztett keretrendszer segítségével.

Köszönetnyilvánítás

A cikkben ismertetett munka részben az OTKA TO46646, TS49885 számú, a Nemzeti Kutatás-fejlesztési Iroda NKFP 2/009/04, valamint a Pázmány Péter program RET-06/2005 projektjeinek támogatásával jött létre. A szerzők köszönik az EU INFSO-50883 projekt és a BVM Épelem Kft támogatását.

Irodalom

[1] Sipos, A. A., Domokos G.:
„Asymmetrical, spatial deformations of reinforced concrete columns and prestressed beams”,
fib Symposium „Keep Concrete Attractive”,
Budapest, 2005., Vol. II, pp.693–698.

[2] Sipos, A. A., Domokos G., Gáspár Zs.:
„A 2D Pelikan iteráció konvergencia-tulajdonságai”,
J. of Building Science, 2005, 33 (1-2), pp.205–217.

[3] Sipos, A. A.:
„Calculation of the spatial deformations of rods without tensile strentgh”,
PhD thesis, BME, 2007.

[4] Domokos, G.:
„Global description of elastic bars”,
Zeitschrift für Angew. Math. und Mech., 1994, No.74, T289-T291.

[5] Brondum-Nielsen, T.:
„Stress Analysis of Concrete Sections Under Service Load”,
ACI Journal, Proceedings, 1979, Vol. 76., No.2, pp.195–211.

[6] Brondum-Nielsen, T.:
„Serviceability Limit State Analysis of Cracked, Polygonal Concrete Sections Under Biaxial or Symmetric Bending”,
ACI Journal, Proceedings, 1986, Vol. 83., No.2, pp.209–218.

[7] Cosenza, E., Debenardi, P. G.:
„Calculation of Stresses, Deformations and Deflections of Reinforced and Prestressed Concrete Elements in Service”,
CEB Bulletin 235, 1997, pp.105–142.

[8] Gáspár, Zs., Domokos, G., Szeberényi, I.:
„A parallel algorithm for the global computation of elastic bar structures”,
Comput. Assist. Mech. Eng. Science, 1997, No.4, pp.55–68.

[9] Domokos G., Szeberényi I.:
„A Hybrid Parallel Approach to One-parameter Nonlinear Boundary Value Problems”,
Comput. Assist. Mech. Eng. Science, 2004, No.11, pp.15–34.

[10] M. Russell, J. Novotny, O. Wehrens,
„GridSphere: An Advanced Portal Framework”,
GridSphere Project Website (www.gridsphere.org)

[11] M. Russell, J. Novotny, O. Wehrens,
„GridSphere: A Portal Framework for Building Collaborations”,
GridSphere Project Website (www.gridsphere.org)

[12] Qt 4.0 Whitepaper,
http://www.trolltech.com/pdf/whitepapers/qt40-whitepaper-a4.pdf

[13] RealVNC Home Page,
http://www.realvnc.com

[14] JavaServer Pages 2.0 Specification,
http://jcp.org/aboutJava/communityprocess/final/jsr152

[15] EUROCODE EN 1992-1-1. April 2002, Rev. final draft.

[16] Genius Portal,
https://genius.ct.infn.it/

[17] Genius Portal,
http://egee.cesnet.cz/en/user/genius.html

[18] P. Kacsuk, G. Sipos
„Multi-Grid, Multi-User Workflows in the P-GRADE Portal”.
Journal of Grid Computing, Vol. 3., Issue 3-4, Kluwer Academic Publisher, 2006.
pp. 221–238.

[19] BIRN Portal,
https://portal.nbirn.net/

[20] LEAD Portal,
https://portal.leadproject.org/