Szakdolgozat

Juhász Andrea

Debrecen

2009

Debreceni Egyetem
Természettudományi és Technológiai Kar
Szilárdtest Fizika Tanszék

Mechanikai kísérletek realisztikus modellezése

Témavezető:

Dr. Szabó István
Szilárdtest Fizika Tanszék Vezető

Készítette:

Juhász Andrea
Mérnök informatikus

Debrecen

2009

Tartalomjegyzék

1. Bevezetés

A szimulációk szerves részét képzik a mai világnak. Céljuk, hogy a modell szintjén az utánzott jelenséggel megegyező, vagy ahhoz hasonló élményt biztosítsanak a felhasználó számára, és így lehetővé tegyék az eredeti változat elhagyását egy adott feladat megvalósításához.

Ahhoz hogy a szimulációnk minél inkább reprezentálhassa nekünk a valóságot, a modellünknek kell a lehető legpontosabbnak lennie. A modellezés kialakulása egészen mélyre nyúlik vissza, elég csak a hajó-makettekre, épületmodellekre gondolnunk. Merész vállalkozás lenne felsorolni az összes modellezhető rendszert, mivel a közgazdasági rendszerek, erőforrás-, környezeti rendszerek, ipari rendszerek, számítógép és információs rendszerek, biológiai, kémiai, fizikai rendszerek is ide sorolhatóak és önmagukban is elég összetettek.

A számítógép megjelenése új teret nyitott a modellezés és szimuláció világának. Mára számos program áll a felhasználók rendelkezésére, mellyel szimulációkat végezhetnek. A gazdasági életben mára elengedhetetlenek lettek ezek a programok, hiszen a repülőgépgyártástól, autótervezéstől, elektronikai, informatikai eszközök tervezésétől kezdve számos területen a szimulációk futtatásával csökkenthetők a fejlesztési költségek.

A szakdolgozat célja bizonyos mechanikai kísérletek modelljének, majd a modell alapján a számítógépes szimulációjának elkészítése. A feladatokat a szabadon elérhető fizikai szimulációs rendszerek áttekintésével, és egy rendszer kiválasztásával kellett megoldani. A dolgozatban bemutatott szimulációs környezetek segítségével az ismetetők alapján az olvasó maga is képessé válhat a céljainak megfelelő szimulációk megvalósítására.

Napjainkban már például a Phun megjelenésével komolyabb fizikai ismeret nélkül, szórakoztató, játékos módon bárki ismerkedhet a szimuláció világával. Az eljárás-, és objektumorientált nyelvek is tartalmaznak grafikus elemeket, tehát akik komolyabb programozási ismeretekkel rendelkeznek, Java, C, C++ stb. forráskódok formájában is elkészíthetik a szimulációkat. De léteznek kimondottan erre a célra kifejlesztett szoftverek, mint például az Easy Java Simulations vagy a Simulink.

2. A mechanika áttekintése

A klasszikus, más néven newtoni mechanika a testek mozgásával, valamint a mozgásokat kiváltó okokkal, törvényekkel foglalkozik. A klasszikus mechanika kizárólag alacsony sebességek esetén alkalmazható, természetesen az alacsony sebességet a fénysebességhez viszonyítva értjük. A fénysebességgel összemérhető sebességek esetén a speciális relativitás törvényszerűségeit kell alkalmazni, atomi méretű, tehát mikrorendszerekre pedig a kvantummechanika nyújtja az ideális összefüggéseket.

A mechanika részterületei

Kinematika

A kinematika a fizika, azon belül is a mechanika, mint tudományág részterülete. Feladata a mozgások leírására vonatkozik, de alapvetően matematikai jellegű. A mozgások leírása alatt azt értjük, hogy egy tetszőleges időpontban meghatározzuk egy test helyét, valamint egy másik testhez viszonyított helyzetét. A test helyzetének meghatározásához úgynevezett vonatkoztatási rendszerre van szükségünk.
A vonatkoztatási rendszer a hely és az idő megadását lehetővé tevő viszonyítási objektumok rendszere a fizikában. Anyagi jellegű fogalom, ami azt jelenti, hogy, alappontjait és alapirányait valós anyagi testek jelölik ki. Bármilyen anyagi rendszer választható vonatkoztatási rendszernek. A koordinátarendszer a vonatkoztatási rendszer matematikai leírása. Egy tetszőleges pont helyzete a térben, vagy a síkban koordináták segítségével adható meg. A koordinátarendszer tekinthető síknak, vagy térnek, melyben egy kezdőpontot és tengelyeket jelölünk ki, melyektől a koordinátákat mérjük.

Az inerciarendszer vagy tehetetlenségi rendszer a fizikai alaptörvények szempontjából a legfontosabb rendszer, amiben érvényes Newton tehetetlenségi törvénye. Newton úgy gondolta, létezik egy ilyen abszolút rendszer, amihez képest minden egyenes vonalú egyenletes mozgást végző rendszer is inerciarendszer. Ma úgy gondoljuk, ilyen abszolút rendszer nincs. Inerciarendszernek tekinthető minden olyan koordinátarendszer, amelyre

igazak a newtoni axiómák, és ily módon a magukra hagyott testek nyugalomban vannak, vagy egyenes vonalú, egyenletes mozgást végeznek. Ha feltételezzük, hogy létezik egy ilyen inerciarendszer, azzal végtelen sok másik vonatkoztatási rendszer létét is feltételezzük, mivel minden az általunk eredetileg feltételezetthez képest egyenletes, egyenes vonalú mozgást végző koordinátarendszer is inerciarendszer.

Egy inerciarendszert megtalálni gyakorlati közelítéssel lehet, attól is függően, milyen pontosan akarunk vagy tudunk mérni. Sok szempontból a Föld felszínéhez rögzített koordinátarendszer is inerciarendszernek tekinthető, de ehhez el kell hanyagolnunk a Föld forgását. A Naprendszer jobb közelítése egy inerciarendszernek, a Tejútrendszer pedig még jobb. [1]

Léteznek nem tehetetlenségi rendszerek, melyeket gyorsuló vonatkoztatási rendszereknek hívjuk. Ilyen például egy mozgó jármű, vagy a forgó Föld. Ezen rendszerekben nem érvényesek a Newton-törvények, mert a testek erőhatás nélkül is látszólag gyorsulnak. Tehetetlenségi- vagy inerciaerők bevezetésével azonban helyre lehet formálisan állítani a mechanikai alaptörvényeket.

Ilyen erők pl:

- transzlációs tehetetlenségi erő – transzlációsan gyorsuló koordinátarendszerben
- centrifugális erő – forgó koordinátarendszerben
- Coriolis-erő – forgó koordinátarendszerben mozgó test esetén
- Euler-erő – változó szögsebességgel forgó koordinátarendszerben

Kinematika esetén szükséges a tömegpont fogalmának ismerete. A jelenségek egyszerűsítése céljából, a valós testek matematikai absztrakciójaként pontszerű részecskéket, azaz tömegpontokat használunk. A tömegpont helyvektorral, tömeggel, sebességgel, gyorsulással jellemezhető. A valós testek mérete természetesen nem mindig hanyagolható el.

Egy tömegpont helyzete, egy, a térben rögzített ponthoz viszonyított helyzetével adható meg, gyakran origónak (O) nevezzük. Eszerint tehát a részecske helyzete az O pontból a tömegpontba húzott $\vec{r}$ helyvektorral adható meg. Amennyiben mozgó részecskéről van szó, akkor az $\vec{r}$ vektor az idő függvényében változik.

Dinamika

Inerciarendszerben a mozgásállapot megváltozása mindig erő hatására történik. A dinamika feladata az erők (a mozgás okainak) leírása.

Tömeg

Az SI alapegysége, mértékegysége a kg, de ezen felül számos mértékegységet használnak még ma is, mint pl az uncia, naptömeg. Aszerint, hogy milyen kölcsönhatásban vesz részt a test, két típusát különböztetjük meg:

- A súlyos tömeg jellemzi a test gravitációs kölcsönhatásban való részvételének mértékét.
- Tehetetlen tömeg, a test erőhatással szembeni tehetetlenségének mértéke.

Lendület

Lendületnek, vagy impulzusnak nevezzük a tömeg és sebesség szorzatát.
A lendület általában véve a test azon törekvésének mértéke, hogy mozgásának nagyságát és irányát megtartsa.

Erő

Isaac Newton angol fizikus négy, tömeggel rendelkező mozgó test viselkedését leíró törvényeit Newton törvényeknek nevezzük. Ezen törvények alkotják a klasszikus mechanika alapját. Newton második törvénye kimondja, hogy egy pontszerű test 'a' gyorsulása egyenesen arányos a testre ható, a gyorsulással azonos irányú 'F' erővel, és fordítottan arányos a test 'm' tömegével.

A törvény képlettel kifejezett, elterjedt formája: $\mathbf{F} = m\mathbf{a}$, ahol

- **F** az erő vektora
- *m* a gyorsítandó tömeg
- **a** a gyorsulás vektora

Munka, energia

Ha egy test a rá ható erők hatására elmozdulást végez, akkor mechanikai munkát végez az erő a testen. Az energia pedig, a testek munkavégző képességét jelenti.

Statika

A statika a testek erőhatás alatti egyensúlyának feltételeivel foglalkozik.

3. A számítógép megjelenése a fizikában

A fizikai kutatásban a számítógépek számos területen felhasználásra kerültek. Ilyen területek például:

- Mérésvezérlés számítógéppel: bonyolultabb méréseket ma már számítógép segítségével végzik, egy központi számítógép segítségével kerülnek az adatok begyűjtésre, tárolásra.
- Adatfeldolgozás, adatok kiértékelése: a korábban begyűjtésre került adatokból számítógép segítségével állítjuk elő a számunkra lényeges információkat.
- Numerikus matematika: Algebrai és differenciál egyenletek numerikus megoldásában fontos szerepet játszik.
- Számítógépes algebra: algebrai műveletek egyszerűbb elvégzését szolgálják az erre a célra megírt számítógépes programok, mint például az integráltáblázatok használatának kiküszöbölésével.
- Grafikus elemek: talán a leglényegesebb előnye a grafikus megjelenítés, ugyanis a kinyert adatok megjelenítésével újabb függvénykapcsolatokat tárhatunk fel a már kinyert információkból.

4. A modell alapfogalma, csoportosítása jellemzői

Szó szerinti fordításban a latin modus, modulus szó mértéket, módot, módozatot jelent. A mindennapi életben azonban ennél jóval szélesebb körű az értelmezése.

A modell szóval jelölik például

- azt a rendszert, amely egy másik rendszerben (a modellezettben) végbemenő jelenséghez hasonló jelenséget valósít meg;
- egyes termékek mintáit (vonatkozhat gépekre, ruhákra, játékokra);
- közlekedési eszközök kicsinyített másolatát (pl. Matchbox);
- épületek geometriailag hű kisebbítését, amelyek rendszerint szemléltető célt szolgálnak és inkább kell makettnak nevezni;
- az olyan szemléltető eszközöket, amelyek valamely nagyon nagy (vagy nagyon kicsiny) objektum oktatási bemutatására szolgálnak (pl. a hidrogénatom modellje, vagy a planetárium). [2]

A modell fogalma

A modell megadására számos definíció szolgálhat válaszul a Magyar Nyelv Értelmező Szótárán keresztül biológiai, filozófiai szótárakon keresztül.
A mi feladatunkhoz legközelebbi választ a Természettudományi Lexikon adhat, mely szerint:

Bonyolult fizikai rendszerek egyszerűsített, minden részletében áttekinthető, gyakorlatilag megvalósított vagy szemléletesen elképzelt, arányosan lekicsinyített vagy felnagyított, matematikailag szabatosan leírható, idealizált mása, amely többé-kevésbé helyesen szemlélteti a vizsgált rendszer vagy folyamat geometriai, kinetikai, dinamikai vagy más fizikai, illetve sztochasztikus sajátosságait. A modellalkotásnál tudatában kell lenni annak, hogy a modell nem azonos a vizsgált rendszerrel vagy folyamattal, és nem tükrözi maradéktalanul összes tulajdonságait. A helyesen alkotott modell mégis magán viseli az objektív anyagi világban meglevő rendszer vagy lejátszódó folyamat fontos ismérveit, és így alkalmas a döntő törvényszerűségek feltárására és szemléltetésére. A kutatás megfelelő

stádiumában a modellalkotásnak nagy a heurisztikus jelentősége, és a modelleknek a fizika fejlődése során mindig fontos szerepe volt.
Leegyszerűsítve tekinthetjük a modellt valamely tárgynak (rendszerint kicsinyített) másaként. [2]

Hasonlósági reláció

Akkor és csakis akkor tekinthetünk valamit modellnek, ha ismerjük a modellezettel való összefüggését: azokat a jellemzőket, amelyek szerint a modell és modellezett hasonlóak egymáshoz.

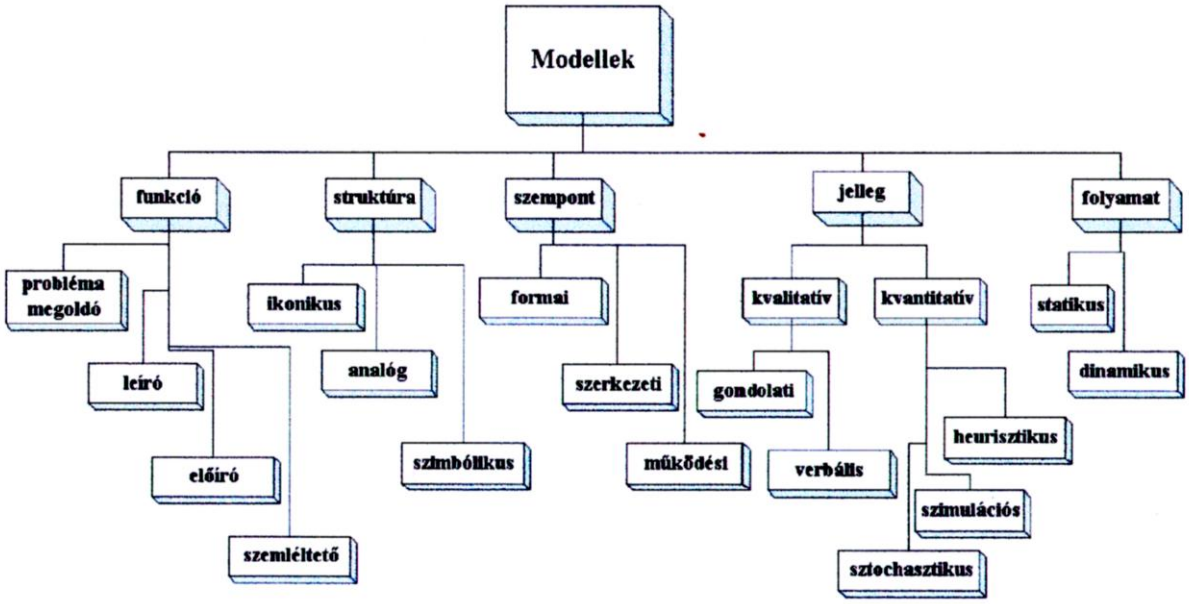
A hasonlóság a hétköznapi szóhasználatban élőlények, tárgyak, fogalmak valamilyen kapcsolatára utal; a tudományos megfogalmazás valamely tárgyrendszer és annak képe közötti összefüggésként értelmezi.
A modell és a modellezett közötti kapcsolat hasonlósági reláció. Az előbbiek szerint a modell és a modellezett mindig csak valamilyen meghatározott szempontból hasonló, más szempontok szerint viszont különböző. Így a modell mindig csak részleges lehet. Az ún. "teljes modell" csak egy van: maga a modellezett. A minden szempontból hasonlóság ugyanis azonosságot jelent. Ezért kell a hasonlósághoz mindig hozzátenni, hogy azt mely tulajdonságokra vonatkoztatjuk és ezzel - implicite - azt is megjelölni, hogy mely tulajdonságokban van különbözőség. Ellenkező esetben könnyen a káros analógia hibájába eshetünk. [3]

A hasonlósági reláció ekvivalencia reláció, tehát
-Reflexív,
-Szimmetrikus,
-Tranzitív.

E feltételek nem teljesülése nagyon súlyos hibákhoz vezethet.
A modell mindig csak bizonyos, meghatározott szempontok szerint hasonló a modellezetthez.
A minden szempontból hasonló modell maga a modellezett.

A modellek csoportosítása

A modellek csoportosításának (1 ábra) kiterjedt irodalma van. Még ha csak a technikai rendszerek modelljeit akarnánk felsorolni, akkor is megoldhatatlan feladatra vállalkoznánk. Gondoljunk csak arra: hányféle technikai rendszer van.



1 ábra *Modellek csoportosítása*

A modell

- célja szerint lehet taktikai (prediktív) vagy stratégiai (demonstratív).
- felépítése szerint: szimulációs vagy leíró (descriptív).
- tér-idő szemlélete szerint: diszkrét vagy folytonos.
- folyamatszemlélete szerint: determinisztikus vagy sztochasztikus

A modell funkciója (a modellezés célja) lehet például a modellezett:
- leírása,
- szemléltetése,
- elemzése,
- létesítésével (működésével) kapcsolatos előírás,

- működésével, várható tulajdonságaival kapcsolatos probléma
  megoldása.

A modellezett folyamat jellege szerint a modell:
- statikus,
- dinamikus.

A modell jellege lehet:
- kvalitatív,
- kvantitatív.

A feladat jellege lehet:
- direkt,
- indirekt (heurisztika, próbálgatás, genetikus, neurális módszerek),
- induktív. [4]

A modellek típusai

- Taktikai modellről akkor beszélünk, ha a modellezés célja valamilyen gyakorlati jellegű probléma megoldása során megalapozott jóslatok, készítése.
- Stratégiai modellről van szó akkor, ha a modell célja valamilyen jelenség összefüggéseinek tudományos, kutatási vagy egyéb álláspont-kifejtése.
- Szimulációs modell az a modelltípus, esetén a modell viselkedési elemei és a valóságos rendszer viselkedési elemei között egyértelmű a kapcsolat, tehát a modell a jelenséghez hasonló viselkedés mutatására képes.
- Leíró modell, valamilyen összefüggést matematikai formában fejez ki, a jelenséget bemutatja. A leíró modell a jelenséget nem szimulálja.
- Diszkrét modell, diszkrét skálán dolgozik, ami azt jelenti, hogy csak a természetes számokra van értelmezve a térbeli és időbeli felbontás. Például évenként, naponként, óránként ad kimenetet.

- Folytonos modell, amely változóit a valós számok halmazán értelmezi.
- Vegyes - egészértékű modell, egészértékű és folytonos változókat is használ.
- Determinisztikus modellről akkor van szó, ha a modell meghatározott bemeneti adatokra pontosan meghatározott számokat ad eredményül. A determinisztikus modellben a beállított paraméterek és bemeneti adatok egyértelműen meghatározzák a modell kimenetét.
- Sztochasztikus modell, a determinisztikus modellel ellentétben, nem konkrét kimeneti értékkel dolgozik, hanem, valamilyen gyakorisági eloszlással. Sztochasztikus modelleket fejlesztünk olyan esetekben, amikor a véletlen szerepét is szeretnénk a vizsgált folyamatban figyelembe venni. A sztochasztikus modellek közül azokat a modelleket, melyek szimulációs modellek, tágabb értelemben Monte Carlo modelleknek, vagy Monte Carlo szimulációknak is nevezzük.

Rendszerek modellezése

A rendszerek térben és időben léteznek. Így beszélhetünk a rendszerek szerkezetéről, és folyamatairól.
Ha a szerkezetet szeretnénk modellezni, elegendő lenne a geometriai hasonlóság figyelembevétele. Ez azonban csak akkor engedhető meg, ha a célunk mindössze a forma bemutatása, és nem a szerkezeti kapcsolatok elemzése.
Vannak olyan feladatok, melyek megoldását lényegesen megkönnyíti, ha korábbi ismereteink alapján lehetőségünk van előzetes feltételezések és következtetések levonására, azaz megvizsgáljuk a más rendszerekkel való hasonlóságokat.
A hasonlóság fogalma nélkül minden rendszert külön kellene vizsgálni. A hasonlóság ismeretében viszont felhasználhatóak az egyes vizsgálati eredmények, a kellően még nem ismert rendszerek jellemzőinek előzetes meghatározásához.

Két rendszer működése abban az esetben hasonló egymáshoz, ha az egyik állapotváltozóinak értékéből a másik állapotváltozóinak értéke kölcsönösen és egyértelműen meghatározható. Ennek megfelelően definiáljuk a folyamatok hasonlóságát:
Hasonlóak az olyan rendszerek, amelyek megfelelő jellemzői arányosak.

A technikai rendszerekben végbemenő folyamatok leggyakrabban parciális differenciálegyenletekkel írhatók le. Ezek főbb típusai a hiperbolikus, a parabolikus, ill. az elliptikus egyenletek.

Hiperbolikus (vagy másképpen hullám-) egyenletek írják le pl. a húr, a vékony rugalmas membrán, a rúd torziós rezgését, a hanghullám terjedését folyadékban (gázban), az elektromágneses hullámok terjedését a térben. Hiperbolikus egyenletek a kvantummechanika alapegyenletei is.

Parabolikus (másként Fourier) egyenletek az ismert általános (instacionárius) mérlegegyenletek, amelyeknek széles körű felhasználási területei léteznek. Itt csak példaként említjük, hogy a diffúzió, a hővezetés, a szárítás, a hidrodinamika instacionárius folyamatait, a gazdasági változásokat ilyen egyenletek írják le.

Elliptikus egyenletek a stacionárius mérlegegyenletek. Szemléltetésül: Elliptikusak a kondenzátorok, az elektrolitok és az elektroncsövek elektrosztatikus terét, a stacionárius hővezetést és diffúziót, a mágneses és gravitációs tereket, az örvénymentes folyadékáramlás és a porózus közegben áramló folyadék törvényszerűségeit leíró ún. Laplace-egyenletek.

A leíró egyenletek típusa szerint beszélhetünk hiperbolikus, parabolikus, ill. elliptikus feladatról. A vizsgált rendszer modelljének realizálása során “szabadon választhatunk” mindazon folyamatok között, amelyek ugyanolyan feladattípusba tartoznak. Hogy ez milyen előnyökkel jár, az nyilvánvaló. Elég csak arra gondolni, hogy míg egyes folyamatok jellemzőinek mérése még laboratóriumban is nagyon bonyolult, illetve hosszadalmas, addig másoké egyszerű és gyorsan megoldható. [5]

5. A szimuláció fogalma

A szimuláció egy vizsgálat, amikor is egy rendszer, folyamat fizikai vagy számítógépes modelljén tanulmányozzák a rendszer valódi és várható viselkedését.
A modelltől és az alkalmazott eljárástól függően számtalan szimulációs megoldás létezik. A modell lehet az eredeti jelenségtől vagy folyamattól csak méretben eltérő, de egyébként azonos jelhordozó közegre felépített kísérleti, vagy az eredeti jelhordozóktól különböző, de azokkal azonos módon viselkedő közegre épített fizikai modell.

Történeti áttekintése:

- **1990:** Egy meghatározott rendszer adott körülmények között lényeges tulajdonságainak vizsgálata elméleti modell segítségével. A modell működését általában számítógéppel vizsgálják, így az ismeretlen rendszerviselkedése kevés költséggel, kockázatmentesen ismerhető meg.

- **1997:** A kiindulási attribútum értékek ismeretében, a rendszer modellezése révén a végállapotra jellemző attribútum-értékek meghatározása. A szimuláció egy rendszer kiindulási értékeihez modellek (összefüggések) segítségével keresi a következményeket.

- **1997:** Szimuláció jellemzői:
    - A valóság komplex leírására törekszi
    - Az összetevők számát a számítási kapacitás függvényében növeli
    - Pontos képet ad a folyamat részleteiről
    - Képes az egyidejű hatások összegzésére
    - Az egész leírását a részletekre kapott eredmények eredője adja
    - A szimuláció sokkal kevesebb egyszerűsítést alkalmaz, mint a modellezés, így eredményei gyakran 20-30%-kal közelebb állnak a valósághoz. Azaz: 20-30% anyagmegtakarítás, 20-30%-kal nagyobb teljesítmény, 20-30%-kal kisebb kockázat... Ez az egyik magyarázata annak, hogy a szuperszámítógépek magas

ára dacára ma már a középvállalatok is egyre növekvő mértékben alkalmazzák a szimulációs technikát K+F tevékenységükben.

- **2004:** A modell objektumokból és törvényekből áll. A szimuláció maga a modell és a modell működtetése – eljárás, amely az objektumokon a változtatásokat a törvényeknek megfelelően elvégzi. Mikor kell szimulációt alkalmazni (kísérlettel szemben):
  - túl gyors
  - túl lassú
  - túl drága
  - túl veszélyes
  - túl bonyolult
  - nincs hozzá eszköz
  - etikai akadályai vannak
  - csak az eredmény látható
  - az eredmény sem látható
  - nem állíthatók be pontosan a feltételei
  - csak egyetlen példányban létezik
  - túl sokszor kell elvégezni

- **2005:** A szimulációs modell egy virtuális megjelenítése egy működő rendszernek vagy folyamatnak. Célunk az, hogy a valóság hasonmását megépítsük és vizsgálat alá vegyük, kísérleteket hajtsunk végre rajta, mielőtt bármilyen költséges változtatást hajtanánk végre a valós környezeten. [6]

A matematikai modell elhelyezhető megfelelő program formájában digitális számítógépben is, ekkor a digitális szimulációról beszélünk. A digitális szimuláció általában pontosabb eredményeket szolgáltat, mint az analóg, és matematikailag, bonyolult, nehezen kezelhető problémák esetén is alkalmazható. Az analóg számítógéppel végzett szimuláció azonban egyszerűbb, olcsóbb, gyorsabb. A két különböző számítógépes szimulációs módszer egymást jól kiegészíti, a kettő egyesítéséből jöttek létre az úgynevezett hibrid szimulációs rendszerek. A hibrid szimuláció során pl. a modellt a digitális számítógépbe program formájában viszik be. A digitális számítógép programozható (digitálisan vezérelhető)

eszközök segítségével köti össze az analóg számítógép megfelelői egységeit, digitál-analóg átalakítók segítségével beállítja a kívánt kezdeti értékeket. Igen nagy rendszerek csak digitális számítógép segítségével szimulálhatók, mert az analóg számítógép szükséges elemeinek száma az objektum bonyolultságával együtt nő, a gépben az elemek száma azonban korlátozott, míg digitális számítógépen a program mérete módosítható. A nagy rendszerek egyszerű, könnyen megtanulható, és a rendszer összetevői közötti kapcsolatokat jól tükröző digitális szimulációjára különleges programozási nyelveket fejlesztenek ki, ezeket szimulációs nyelveknek nevezzük. A szimulációs nyelvek száma igen nagy, mivel minden problémakör leírására más és más szimulációs nyelv használata nyújt optimálisabb megoldást.

6. Modellezés és szimuláció a világban

Az élet számos területén, amikor a jövőképről próbálunk valamit is megtudni, szimulációkra tudunk csak alapozni. A szimulációk minőségét, azt hogy mennyire is esik egybe a valósággal, a modell határozza meg. Erre nagyon jó példa az időjárás-előrejelzés.

Az oktatásban

A személyi számítógépek és a szimulációs programok megjelenésével lehetőség nyílt az előadó tanárok és diákok számára a fizikai kísérletek szimulációjára. Jelentősége elsősorban abban rejlik, hogy lehetőséget nyújtanak olyan kísérletek bemutatására, amelyeket az előadás keretein belül pl. adott körülmények között nem lehet elvégezni. Felmerült az a lehetőség is, hogy az elvégezhető kísérleteket is helyettesítsék számítógépes szimulációkra, de az erről alkotott vélemények erősen megoszlottak, így végül nem nyert teret ez a megoldás.
A Massachusetts Institute of Technology Fizikai Intézetben fennálló problémák más országok iskoláiban is ugyanúgy jellemzőek. A nem fizika szakos hallgatók kevésbé látogatják a fizika szemeszteretet, mivel száraznak és unalmasnak tartják az előadásokat. Az érdeklődés hiánya és a szemeszterek ritka látogatása miatt igen nagy arányban tesznek sikertelen vizsgát a hallgatók. Ezen problémára a szimuláció szolgált megoldásként. A hallgatók számítógépeken pár fős csoportokban végzik a szimulációkat. A feladat végzése során szert tesznek a szimulált kísérlet fizikai hátterére, valamit a megvalósítás során programozási ismereteket is szereznek.

A kémiában

Az anyag mikroszkopikus számítógépes szimulációja az utóbbi évtizedekben a modern anyagszerkezet-kutatás nélkülözhetetlen eszközévé vált. Az atomi mozgások, kölcsönhatások számítógéppel végzett modellezése lehetővé tette kísérleti eredmények értelmezését, segítette alapvető folyamatok megértését. Számítógépes szimulációkkal nehezen, esetleg kísérletileg közvetlenül nem megállapítható tulajdonságok határozhatóak

meg. Napjaink számítástechnikai kapacitásának kihasználásával, akár eddig ismeretlen anyagok esetén is számos kémiai és fizikai tulajdonság megjósolható, ami a gyógyszertervezésben és a nanotechnológiában jelentősek.

Kolloidkémiai kísérletek eredményeinek elemzésekor gyakran nehézséget okoz, hogy a vizsgált rendszerek rendkívül bonyolultak, összetettek, a kísérleti körülmények nehezen szabályozhatóak. Kolloidkémiai folyamatok számítógépes modellezése értékes információt szolgáltathat, hiszen akár atomi felbontásban vizsgálhatunk szerkezeti tulajdonságokat, emellett minden körülményt szabadon rögzíthetünk.

A hétköznapi életben

Egy amerikai tudóscsoport elkészítette a World Trade Center elleni, 2001. szeptember 11-én lezajlott terrortámadás szimulációját, mellyel az épületek szerkezeti egységeinek akkori viselkedését vizsgálták meg.
A 3D-s szimulációt a Purdue Egyetemen, egy kétéves, az amerikai Nemzeti Tudományos Alap részleges támogatásával létrejött projekt keretében készült el. A kutatás fő célja az volt, hogy az építészeti megoldásokat tovább fejlesszék oly módon, hogy akár hasonló katasztrófák esetében sem omoljanak össze a megsérült épületek.

A kutatás vezetője, Christoph Hoffmann informatikus professzor a sajtónak tett nyilatkozata szerint, a legnagyobb kárt a meggyulladó üzemanyag okozta, mert a robbanás és a keletkezett tűz eltávolította a legfontosabb tartóelemek szigetelését. Az ütközés és a tűz oly sok kulcsfontosságú tartóelemet semmisített meg, hogy az, az épület összeomlásához vezetett. A Világkereskedelmi Központ tervezésekor számoltak egy lehetséges repülőgép- és egyéb katasztrófákkal, ám az építkezés idején még a Boeing 707 számított a legnagyobb gépnek, s e jármű kevesebb üzemanyagot tartalmazott, mint a 767-es, így a várható tűz sem volt olyan méretű, mint amely 2001-ben a végzetes károkat okozta. A kísérlet legfontosabb eredménye annak felismerése volt, melyet az adatok alá is támasztottak, hogy ha egy ilyen gép, hasonló körülmények között, ám üzemanyag helyett vizet szállítva csapódott volna az épületbe, a tartóelemek és a szigetelés kibírta volna a becsapódást és evakuálni lehetett volna a bennlévő embereket.

7. A szimulációs programok, mechanikai kísérletek szimulációja

A Phun bemutatása

A Phun egy leginkább nevelési és oktatási célra kifejlesztett szoftver, amely rajzfilmszerű dizájnnal és a fizika törvényszerűségeivel ad lehetőséget modellek szimulálására. Ebben a játéknak is tekinthető szerkesztőprogramban különféle kétdimenziós idomokat, idomcsoportokat rajzolhatunk, kombinálhatjuk őket, illetve – ez igen fontos – elláthatjuk őket fizikai tulajdonságokkal. Az elemekből tehát bármit építhetünk – rajzolhatunk –, s ezeket az elemeket aztán interakcióra késztethetjük – a lehetőségek gyakorlatilag végtelenek: az elemeket lehet egymáshoz is rögzíteni, meghajtani, csoportosítani, másolni stb. Magát a programot a Department of Computing Science at Umeå egyetem, Emil Ernerfeldt nevű hallgatója készítette el. Az Algoryx Simulationnél került továbbfejlesztésre a program és az óta népszerű internetes rajongótáborral rendelkezik, talán ezért is jelennek meg újabb és újabb bétaverziói.

2 ábra *Phun felhasználói felülete*

A program a munkaterületből áll, valamint egy menüsorból (2. ábra).

A menüsor főbb elemei:

- File menü, melyben betölthetünk korábbi szimulációt, törölhetjük az adott munkánkat.
- Sim control menüpont segítségével elrejthetjük és előhívhatjuk a szimulációt vezérlő gombokat. Itt van lehetőség a zoom állítására, a szimuláció elindítására valamit megállítására is.
- Toolbar menüponttal lehet elrejteni, előhívni az eszközöket, melyeket a programban használhatunk. Pl. alakzatok rajzolása, sík rajzolása, rögzítő elemek stb. (3. ábra). A menü fontosabb elemei a rajzoló elemek. Az első két menüponttal bármilyen alakot rajzolhatunk, leginkább egy ceruzához lehet hasonlítani. Összetettebb alakzatok létrehozásához használhatunk rögzítő elemet, mellyel az alakzatokat egymáshoz rögzíthetjük (Fixate tool), motorral láthatjuk el (Hinge tool).

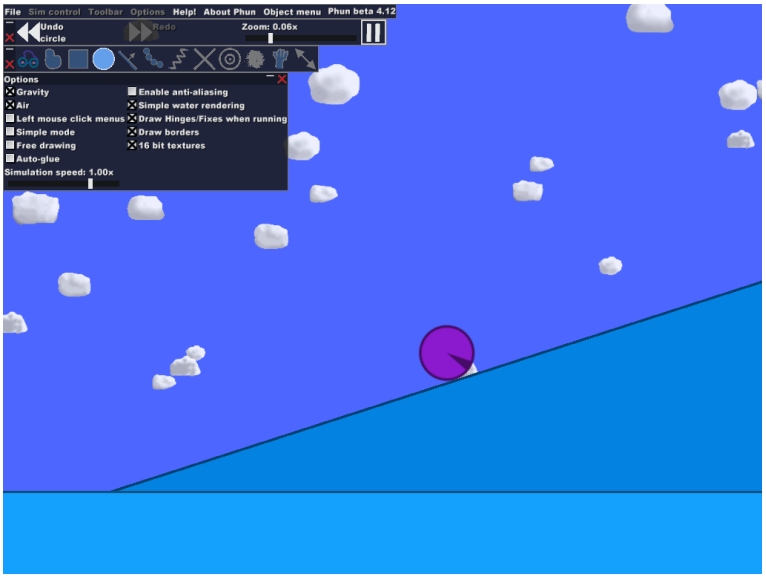
3. ábra *Phun Toolbar menüpontja*

Az elemek fizikai tulajdonságait helyi menüből tehetjük meg, itt állíthatjuk be a halmazállapotot, sebességet, forgást stb.

Egy lejtőn való súrlódás nélkül lecsúszó test szimulálásához (4. ábra) igazából nincs is matematikai háttérre szükségünk, egyszerűen fantázia kérdése, hogy mit és hogyan szeretnénk szimulálni.

4. ábra *Lejtőn lecsúszó test Phunban*

Egyszerű lépésekkel változtathatunk az elemek alakján, színén, a lejtő szögén.



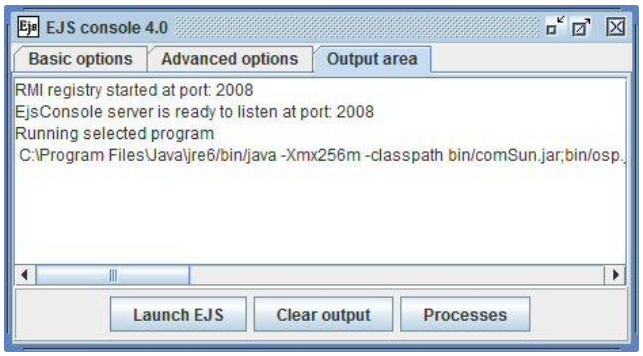
5. ábra *Lejtőn leguruló test Phunban*

Az Easy Java Simulations (EJS)

Az EJS szintén a szimulációs programok közé tartozik, melyet a JAVA nyelvre alapoztak. Használatához viszont a felhasználónak meg kell fogalmaznia a probléma, feladat matematikai hátterét is. Ez a program célja elsősorban nem az, hogy a professzionális programozók feladatát megkönnyítse, hanem inkább azoknak a felhasználóknak készült, akik a szimulációs problémát tudományos oldalról közelítik meg, azaz jobban érdekli őket a szimuláció folyamata, tartalma, maga a jelenség, mint a programozási eszközök, programozás folyamata.

Az Easy Java Simulations felépítése

- EJS konzol: a program indításakor a konzol jelenik meg (6. ábra), alapvető és speciális beállításokra van itt lehetőség. Első indításkor lényeges, hogy itt be kell állítani a munkaterületet, azaz, hogy hová szeretnénk elmenteni a munkánkat. Ez a beállítás persze bármikor megváltoztatható.



6. ábra *Easy Java Simulations konzolja*

- EJS interface: maga a munkaterület. Oldalmenüvel rendelkezik, amely mindig elérhető és használható, függetlenül attól, hogy éppen milyen munkapanelen dolgozunk. Az oldalmenü elemei:
  - Új szimuláció készítése
  - EJS Digital Library
  - Mentés és Mentés másként
  - Keresés, mely segítségével a workspace-en belül kereshetünk
  - Szimuláció indítása
  - A csomag szimulációja, itt menthetjük el a .JAR kiterjesztésű állományunkat, amelyet később bármely gépen futtathatunk, amelyre JAVA van telepítve.
  - Beállítások menü, a futással, a készülő HTML oldallal, szerzővel kapcsolatos beállításokat változtathatjuk meg.
  - Az utolsó menüpont pedig az Információ menü, amely egy link az Easy Java Simulations oldalára:

    http://www.um.es/fem/EjsWiki/pmwiki.php?n=Main.HomePage

  A Output, azaz a kimeneti area is állandó eleme a programnak, itt kaphatunk információt a fordításról.

Az EJS interfacen belül 3 panel található:

- Description azaz a Leírás arra szolgál, hogy leírjuk, szövegesen bemutassuk a szimulációnkat. Lehetőség van itt multimédiás elemek használatára is. Alapjában véve egy egyszerű HTML szerkesztőről van szó.
- Modell panel segítségével hozhatjuk létre a változókat, itt rendelhetünk a változókhoz kezdőértéket, megírhatjuk az algoritmusokat, melyek meghatározzák a modellünk viselkedését az időben. További panelekre oszlik, melyekkel egy konkrét szimuláció bemutatásakor még részletesebben foglalkozom.
- View panelen tudjuk magának a szimulációnak a grafikus, azaz látványi elemeit megadni, megtervezni.

A lejtőn lecsúszó test problémája Easy Java Simulations-ban

A lejtő egy teher felemeléséhez szükséges erő csökkentésére szolgáló egyszerű gép, akár az ék, vagy a csavar. Feltalálása egészen a történelem előtti időkig nyúlik vissza. Fizikai jelentősége a 17. században Galilei kísérletéhez kapcsolódik, ugyanis Galilei lejtőn legurított golyók hatására fogalmazta meg híres tételét, mely kimondja, hogy a szabadon eső testek mozgása nem függ a test tömegétől. Szintén lejtővel végzett kísérletei vezettek ahhoz a felismeréshez is, hogy egy egyenletesen gyorsuló test mozgása során megtett út négyzetesen arányos az közben eltelt idővel. A lejtő felépítését tekintve, lényegében egy a vízszintessel kis szöget bezáró sík, melyre egy nehezebb testet könnyebb felvontatni, mint függőlegesen megemelni.
A lejtő használatával a munka mennyisége nem csökken, hiszen a felvontatás ideje lényegesen megnő, mintha függőlegesen emelnénk, viszont az erőszükségletek jelentősen csökkennek.

A lejtőn lecsúszó test G súlyára súrlódásmentes esetben két erő hat, egy F_1 erő (1), mely a lejtővel párhozamos, és egy F_2 erő (2), amely merőleges arra. Erre a következő összefüggések írhatók fel:

$$F_1 = G \cdot \sin \alpha \quad (1)$$

$$F_2 = G \cdot \cos \alpha \quad (2)$$

Abban az esetben, ha a testre nem hat vontatóerő, a következő összefüggés - amely a gyorsulásra vonatkozik - írható fel (3):

$$m \cdot a = F_1 = G \cdot \sin \alpha = m \cdot g \cdot \sin \alpha \quad (3)$$

Így a lejtő menti gyorsulás az F_1 erővel (4) lesz egyenlő, ami a lejtővel párhuzamos irányú:

$$a = F_1 = g \cdot \sin \alpha \quad (4)$$

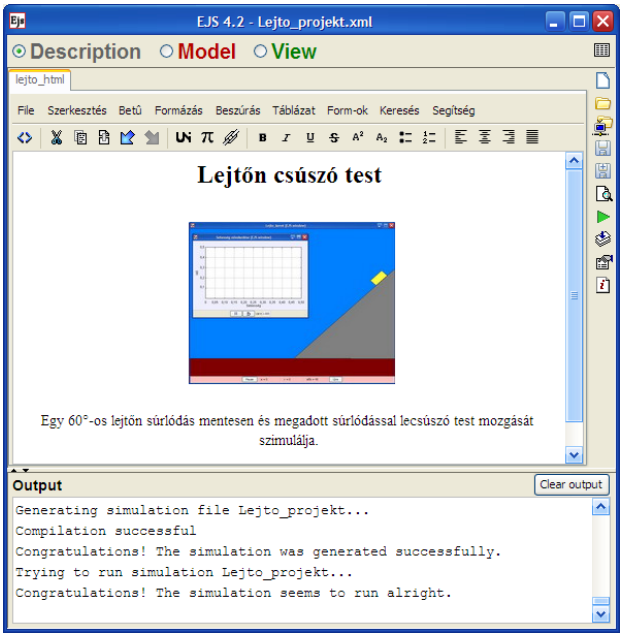
Ezen ismeretek segítségével egyszerűen megfogalmazhatóak a matematikai egyenletek:

```
vx=g*Math.sin(angle);
vy=g*Math.cos(angle);
dt=dt+0.0001;
v=a*dt;
x=x-v*dt;
y=y-v*dt;
```

A pontok kiszámítása a pillanatnyi sebességeken alapul. A v értéke folyamatosan újraszámolásra kerül, mivel az a gyorsulást ismerjük. A megtett út, pálya számítása ezen sebesség felhasználásával történik x, y komponensekre felbontva.

Az EJS program nem csak a JAVA kódot hozza létre, megkönnyítve a programozó dolgát, hanem emellett készít egy Java-appletet és egy alap weblapot is. A weblap szempontjából fontos a Description munkapanel (7. ábra).

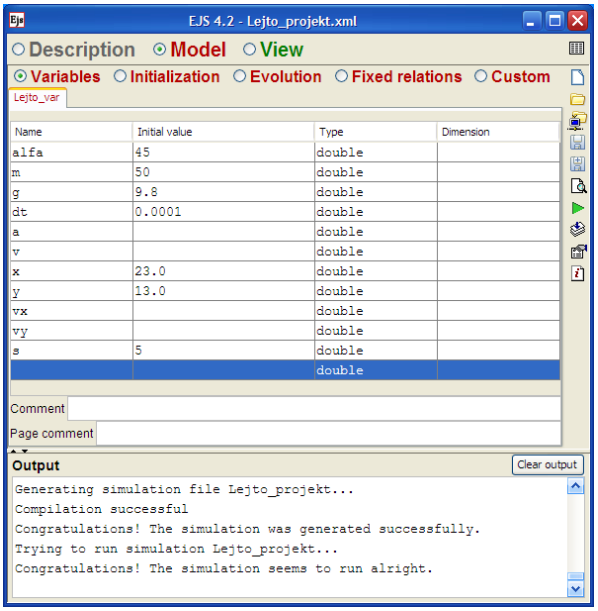
Használatát tekintve, igen egyszerű. Lehetőség van a betűk formázására, szövegek igazítására, képek beszúrására. Az internetes használatot megkönnyítve, form-ok beszúrása is lehetséges (szövegmezők, nyomógombok, jelölőnégyzetek, jelszó stb.).



7. ábra *Description panel*

A modell panelen (8. ábra) történik a probléma matematikai fizikai hátterének a lefektetése. Először, mint számos programozási nyelvben, meg kell határoznunk a változókat, melyeket egy táblázat kitöltésével tehetünk meg. A Variables fülön megadjuk a változó nevét, kezdőértékét, egy legördülő menüből kiválasztjuk a változó típusát, valamint dimenziót adhatunk meg.

Itt adom meg a lejtőn lecsúszó test tömegét, a lejtő szögét, a gravitációs állandót és az időt. A többi számomra hasznos adatot pedig ezekből az értékekből számolom ki. Az itt megadott változókat nem tudjuk módosítani, miközben fut a szimuláció.



8. ábra *Modell panel*

Vannak olyan változók, melyeknek nem tudjuk a kezdőértékét vagy bonyolult számítások végzésére van szükség, ezeket az Initialization fülön tehetjük meg.

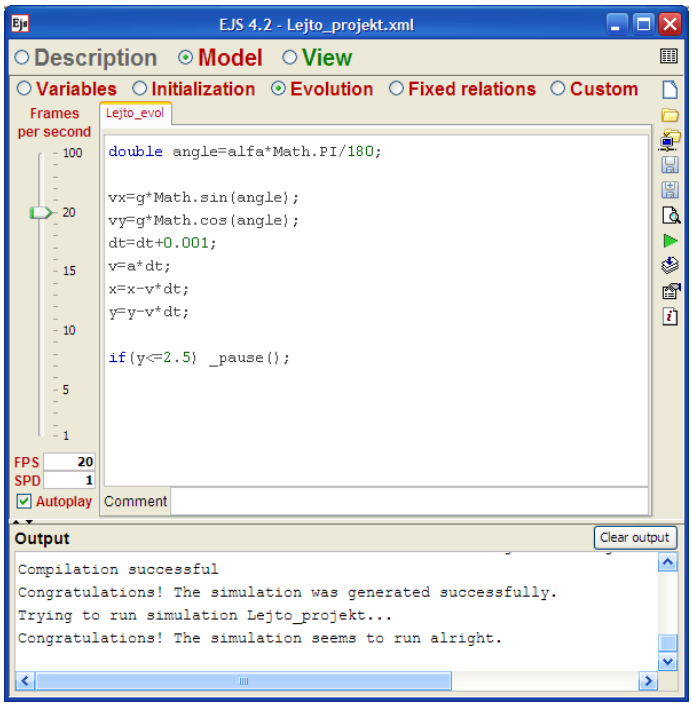
Ahhoz hogy a fokban megadott szöggel számolni tudjunk, át kell váltanunk radiánba:

```
double angle=alfa*Math.PI/180;
```

Valamint kiszámíthatjuk a test gyorsulását is, mivel a folyamat lejátszódása alatt nem fog változni az értéke:

```
a=g*Math.sin(angle);
```

Az Evolution fülön (9. ábra) a modell időbeli változását írhatjuk le matematikai, valamint differenciál egyenletek segítségével.



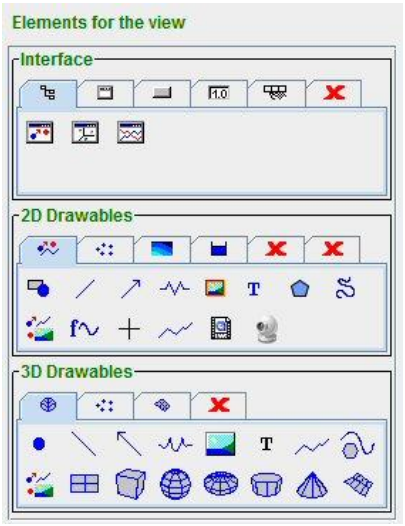
9. ábra *A Modell panel Evolution ablaka*

Fontos megemlíteni az ODE szerkesztőt, mely differenciál egyenletek szerkesztésére szolgál.

Az utolsó fő panel a View, amellyel a vizuális elemeket lehet elkészíteni. Készíthetünk a segítségével 2 dimenziós vagy akár 3 dimenziós szimulációkat is. A változókat hozzárendelhetjük az elemekhez, valójában itt történik a modell és a szimuláció összekapcsolása.

A View elemei 3 csoportba sorolhatóak (10. ábra):

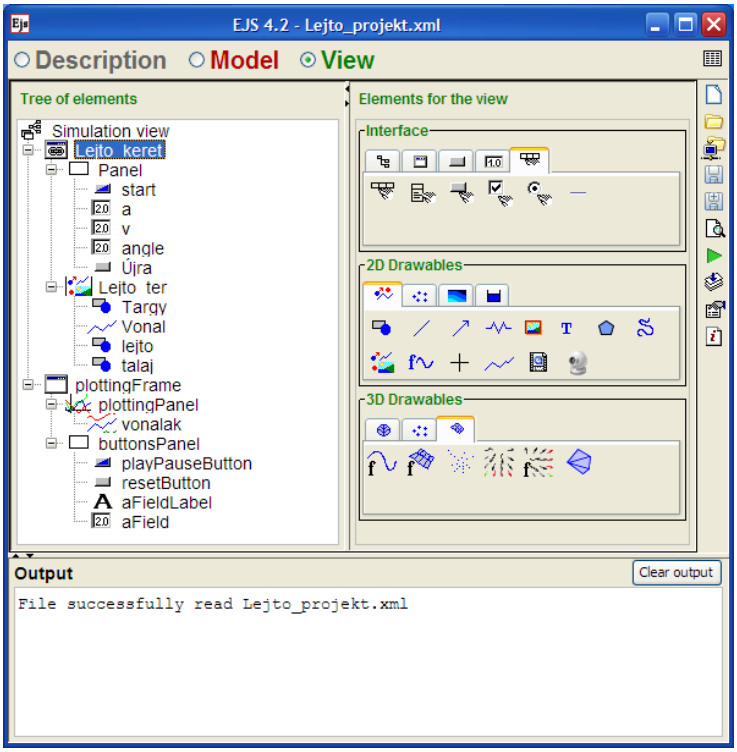
- Felületelemek, ide tartozik maga az ablak, amelyben megjelenik a szimulációnk, valamint a kereten belül elhelyezhető panelek nyomógombok, jelölőnégyzetek.
- 2 dimenziós elemek között találhatunk egyszerű alakzatokat, kört, sokszögeket, vonalakat, görbéket, beszúrhatunk akár kamerával készült képeket vagy saját image fájlt.



10. ábra *View eszköztár*

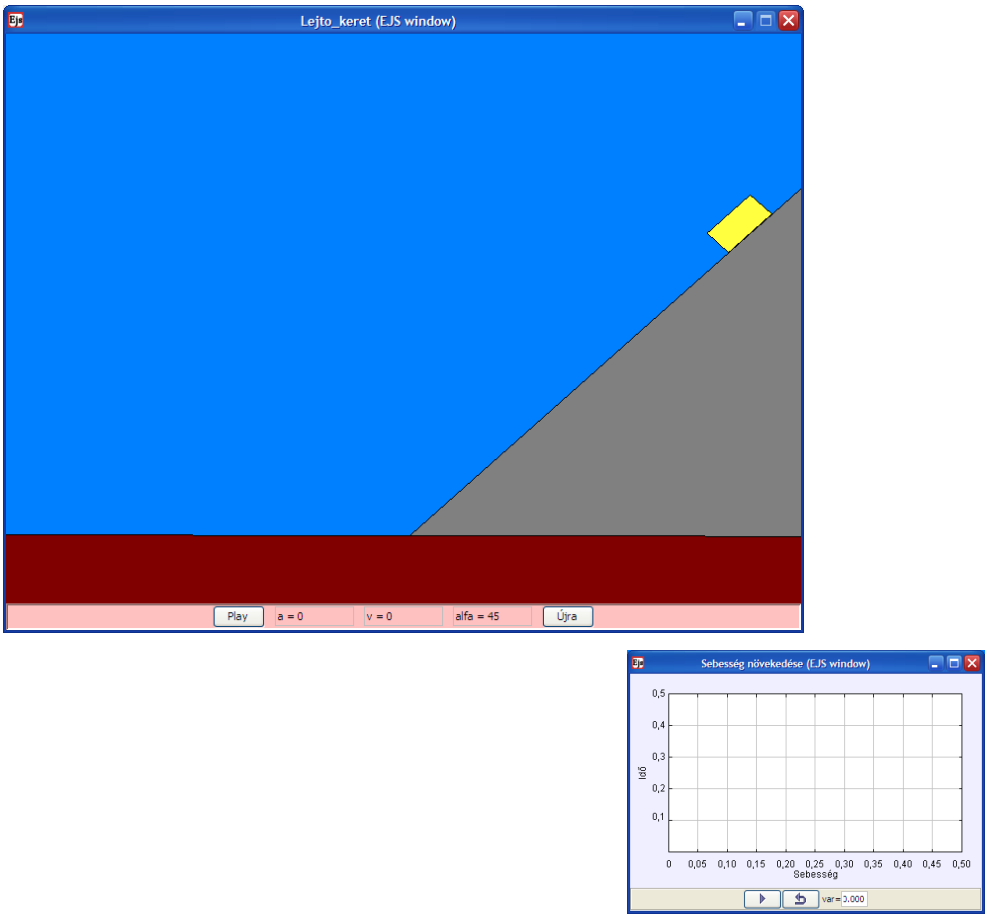
- 3 dimenziós, azaz térbeli elemek között pedig megtalálhatjuk a térbeli testeket, pontcsoportokat, de ezeket kizárólag a 3 dimenziós panelen belül helyezhetjük el.

A szimuláció elemeinek létrehozása egy könyvtárfa szerkezet létrehozásával történik. Első lépésként az ablakkeretet kell létrehoznunk, majd pedig így tudjuk a további elemeket elhelyezni. A konkrét pozíciókat, elemek méretét, valamint a változókkal való összerendelést, az egyes elemek Properties menüjében tudom beállítani.



11 ábra *View panelben a lejtőn lecsúszó test problémájának könyvtárszerkezete*

A következő könyvtárszerkezet kialakításával és a megfelelő beállításokkal és paraméterekkel a következő szimulációs ablakot hoztuk létre (12. ábra):



12. ábra *Lejtőn lecsúszó test egy lehetséges megoldása*

Ferde hajítás problémája

A kinematika területe foglalkozik a mozgásokkal, helyváltoztatásokkal, tehát ide tartozik egy test hajítása is. Mivel a test méretei a mozgáshoz képest elhanyagolhatóak, ezért anyagi, vagy tömegpontként tekintünk rá. A mozgás meghatározásához, azaz a tömegpont mindenkori helyzetének leírásához elsősorban a mindenkori viszonyítási alapot kell lefektetnünk. Egy 2 dimenziós szimulációs mozgás esetén a vonatkoztatási rendszer matematikai megfelelője, egy jól definiált koordinátarendszer alkalmas. Amennyiben térbeli mozgásról beszélünk, a test helyzete az általánosan alkalmazott Descartes-féle koordinátarendszerben ábrázolható. Mint ismeretes, ezt a rendszert három, páronként egymásra merőleges, egy közös pontból, a koordinátarendszer origójából induló egységvektorral illetve az ezen egységvektorokra illeszkedő egyenesekkel, a koordinátarendszer tengelyeivel adhatjuk meg.

X,Y komponensek

Ahhoz, hogy a test pályáját szimulálni tudjuk egy koordinátarendszerben, ahhoz meg kell határoznunk, az a vízszintes (x tengely) és függőleges (y tengely menti) koordinátákat.

Ezt azért tehetjük meg, mert a mozgás egy $v_0 \times cos\alpha$ sebességű vízszintes egyenletes mozgás, és egy függőleges, $v_0 \times sin\alpha$ kezdősebességű függőleges hajítás (szabadesés) eredője.

Ezért a vízszintes elmozdulása a testnek (1):

$$x = v_0 \cdot \cos(\alpha) \cdot t \qquad (1)$$

A függőleges elmozdulása (2) pedig:

$$y = v_0 \cdot \sin(\alpha) \cdot t - \frac{g}{2} \cdot t^2 \qquad (2)$$

A kísérlet szimulálásához, ezen képletek segítségével, kiszámíthatjuk a mozgás koordinátáit.

```
x = x + vx*dt;

y = y + vy*dt - 0.5*g*dt*dt;

vy = vy - g*dt;
```

A v_0 kezdősebesség értéke ebben az esetben megfelel a test kiindulópontjának (x,y) koordinátáival, így tehát rekurzív módon számítjuk ki a mozgás koordinátáit. A vx és vy értékeit az Initialization panelen már meghatároztunk, amik a következők:

```
double radians = alfa*Math.PI/180.0;

vx = v * Math.cos(radians);

vy = v * Math.sin(radians);
```

A pályaalakja egy lefelé nyíló parabola lesz, melynek egyenlete (3) a következő:

$$y = \tan(\alpha) \cdot x - \frac{g}{2 \cdot (v_0)^2 \cdot \cos^2 \cdot \alpha} \cdot x^2 \quad (3)$$

A szimuláció során szükségünk van egyéb információk ismeretére is, de ahhoz hogy ezt kiszámíthassuk szintén szükséges a fizikai képetek ismerte.

A test emelkedésének ideje (4), azaz mikor érjük el a parabola csúcspontját:

$$t_{em} = \frac{v_0 \cdot \sin \alpha}{g} \quad (4)$$

Az emelkedés maximális értéke (5), tehát a parabola csúcspontjának y koordinátája:

$$h_{em} = \frac{v_0^2 \cdot \sin^2 \alpha}{2g} \quad (5)$$

Valamint a dobás hossza (6), azaz ahol a test földet ér:

$$x_{max} = \frac{v_0^2 \cdot \sin 2\alpha}{g} \quad (6)$$

A fenti értékek a program futása során csak abban az esetben változnak, hogyha valamely deklarált változó értékét a menüben megváltoztatjuk, tehát ezen értékek kiszámítását az Inicialization panelen érdemes megejteni:

```
tem=(v*Math.sin(radians))/g;

hem=(v*v*Math.sin(radians)*Math.sin(radians))/(2*g);

xmax=(v*v*Math.sin(2*radians))/g;
```

A View panelen a Menü középső panelén jelennek meg ezek az értékek. A kezdeti sebesség, a ferdehajítás szögének változtatásával, az értékek újraszámításra kerülnek. Ehhez

egy beépített függvényt használunk, amely az _initialize(). Feladata, hogy a megadott változók értékeit az újabb szimulációk során frissítse.

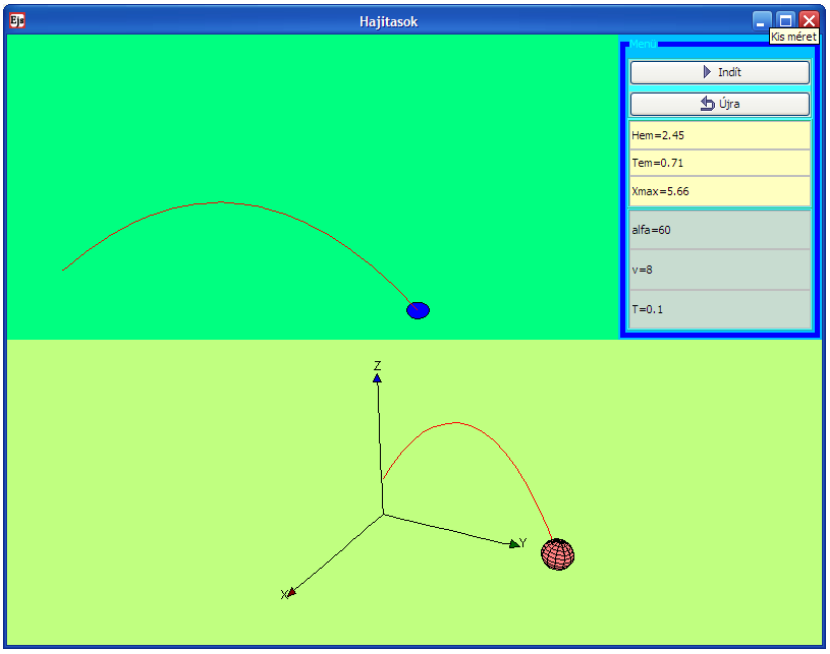
A szimuláció alapbeállításit a változók és értékeik deklarálásakor határozzuk meg.
A gravitációs állandó: g=9.8. Értéke nem változhat.

A kezdősebesség, a hajítás szöge és az idő, a menü harmadik panelén bármikor megváltoztatható, de a program indításakor a kezdő deklarációs értékek fognak szerepelni, így már a program indításakor kiszámolásra kerülnek azok a változók értékei, melyeket nem láttunk el kezdőértékkel.

v=8m/s
alfa=60°
dt=0.1s

A program felépítése

A program egyetlen ablakból áll (13. ábra), mely két lényegi részre van bontva, az egyik a menü.
A menün található az indítógomb, amely elindítja és le is állíthatja a szimuláció menetét.
Az Újra gomb segítségével visszaállíthatjuk a labdákat az alaphelyzetükbe, amely a (0,1.5) koordinátákon található. Erre szintén az _initialize() függvényt használjuk.



13 ábra *Ferde hajítás egy lehetséges megvalósítása*

A középső panelelem az információs rész, ahol további információkat számítások értékeit láthatjuk.

Ha a programot az alapértelmezett értékekkel lefuttatjuk, a test emelkedésének ideje:

$$T_{em}=0,71s$$

A dobás maximális magassága:

$$H_{em}=2.45m$$

A 1.5 m-es magasságból hajítva a dobás hossza:

$$X_{max}=5.66m$$

A menü harmadik panelrészén a beállítások találhatóak. A szög, kezdősebesség és idő értékek változtatásával a szimuláció ismét lefuttatható.

A menü csak abban az esetben látható, ha éppen nem futtatjuk a szimulációt. Ahhoz, hogy ez lehetséges legyen, a panel beállításainál kell a Visible tulajdonságnál, egy _isPaused() eljárást meghívni. Ekkor már tehát csak abban az esetben fog látszani a menü, ha a szimulációnk éppen szünetel.

Az ablak másik lényegi része a szimulációs tér, mely szintén két külön részből áll. A felső részen, azaz a Drawing panelen látható a ferde hajítás 2 dimenziós szimulációja. Az alsó részén pedig, ugyanannak a hajításnak a 3 dimenziós változata a Drawing panel 3D-n. A hajítás 3 dimenziós szimulációja esetén a kép az x,y,z tengelyeken forgatható, tehát különböző szögekből és nézetekből is megtekinthető a mozgás folyamata. A mozgás ívét egy úgynevezett Trail elemmel kirajzolhatjuk, így láthatjuk, hogy a mozgás pályája valóban egy lefelé ívelő parabolát ír le.

A Matlab bemutatása

A Matlab egy interaktív környezet, mérnöki és tudományos feladatok, szimulációk elvégzésére. A Matlab a matrix laboratory elnevezés rövidítése. A szoftver egy mátrix-orientált nyelv, mely leginkább kétdimenziós, azaz téglalapszerű mátrixokkal való műveletekre alkalmas, természetesen a speciális esetek is kezelhetők vele, mint például a skalár (egyelemű tömb), valamint sor és oszlopvektorok.

Az elemek IEEE szabvány szerinti double ábrázolási módot feltételeznek, ami lehetővé teszi, hogy a valós számok mellett 3 kitüntetett kód is ábrázolható legyen: *NaN, inf, -inf*. Ezek rendre a definiálatlan értéket (*NaN*, not a number), valamint a pozitív (*inf*) és negatív (*-inf*) végtelent jelölik. A programban a double típuson kívül más típus nem értelmezett, tehát a ciklus változók típusa is rendre double típusú. A mátrixokra a standard függvények alkalmazhatóak, mint pl. sin, cos, sqrt, exp, log, log10, mint ahogy a matematikai függvények is mint pl. fix, floor, ceil, round, rem, real, imag, conj, abs, angle, sign stb. Ezen függvények végrehajtása elemenként történik. A megszokott módon használható feltételes utasítás, azaz if else, valamint alkalmazhatunk ciklusokat is (for, while).

A rendszermodellezést és szimulációt a Simulink blokkorientált nyelv támogatja. A különböző szakterületek jelfeldolgozási feladatait a Matlabra épülő toolboxok segítik:

- Control System
- Signal Processing
- Optimization
- System Identification
- Fuzzy Logic
- Neural Network
- Image Processing
- Robust Control
- Extended Symbolic Math
- Real-Time Workshop

A Matlab felépítését tekintve, a program elindításakor a parancs ablak (Command Window) jelenik meg az un. Matlab prompt-tal: (>>) Ezen kívül létrehozhatunk grafikus ablakokat is, melyek 2 dimenziós, vagy 3 dimenziós grafikai objektumok megjelenítésére szolgálnak. A plot utasítással lehetőség van több görbe ábrázolására is, melyeket akár különböző színekkel és vonaltípusokkal láthatunk el.

Numerikus módszerek használatakor gyakran van szükség nagy mennyiségű adathalmazok kezelésére. Ezen adathalmazok kezelése nehézkes lenne a parancs ablakot használva, egymás után begépelve a szükséges utasításokat. E helyett használhatjuk az úgynevezett `M` állományokat. Ezek szöveges állományok, melyek Matlab utasításokat tartalmaznak. A szöveges állományokat nevük azonosítja. Utasításait a program egymás után értelmezi és végrehajtja.

A .m állományokban lehetőség van ciklusok szervezésére, elágazó utasítások használatára és egyéb programozási eszközök bevezetésére is.
Speciális M állományok a függvények. A programozónak tehát módjában áll saját függvények létrehozására is, specialitása abban rejlik, hogy a function kulcsszóval kezdődik.

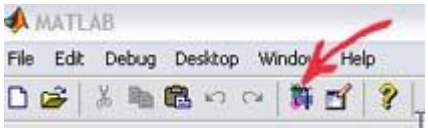
Simulink

A Simulink egy blokk-orientált nyelv. A Simulink a blokkokból felépített dinamikus (differenciál- és/vagy differencia- és/vagy algebrai egyenletekkel leírható) rendszerek szimulációjára alkalmas program. A Simulink-et két lépesben használjuk fel, a modell-alkotásban és a modell-analízisben. A modellt blokkokból építjük fel, melyek típusosztályokba vannak szervezve. A blokkdiagram bevitele a standard grafikus programokéhoz hasonló.

A modell szimulálásakor több paraméter bevitele szükséges:

- szimuláció időtartománya
- alkalmazott numerikus integrálási módszer
- lépésköz nagysága
- és az elvárt pontosság

A Simulink két módon indítható, vagy a Matlab parancs ablakából a >>Simiulink paranccsal, vagy pedig a menüsorból a Simulink ikonnal (14. ábra)

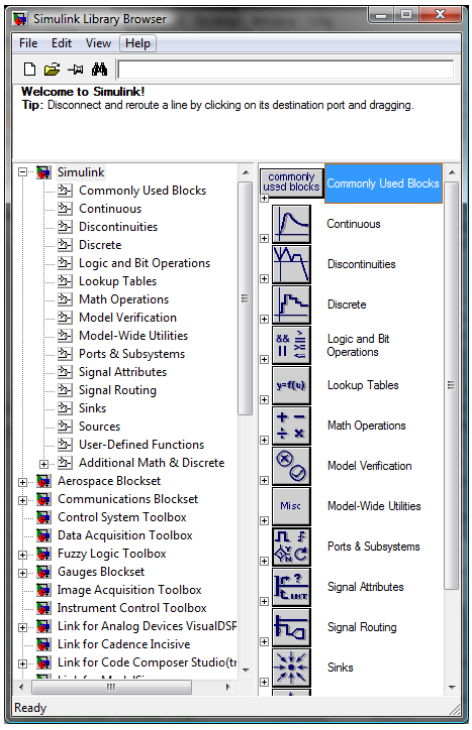


14. ábra *A Simulink indítógombja*

Ekkor megjelenik a Simulink típusosztálya (könyvtárszerkezete) (15. ábra), amely tartalmazza a blokkokat (elemeket).

Az új fájl ikonra kattintva megjelenik a munkafelület, azaz a modell ablak.

A szimulációt alkotó elemeket a „fogd és húzd” módszerrel hozhatjuk létre a munkaterületen. A blokkok típus szerinti kategóriákba, úgynevezett blokk könyvtárakba vannak sorolva a könnyebb áttekinthetőségért. Az alkotóelemek közötti matematikai összefüggéseket az összekötések jelzik.

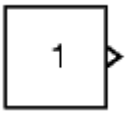


15. ábra *A Simulink könyvtárszerkezete, mely a blokkokat tartalmazza*

A szimuláció építőelemei

Állandó érték (16. ábra): egy valós vagy komplex állandó értéket generál. Alapértelmezettként az értéke 1.

A beállítások menüben megváltoztathatjuk az értékét. Mivel egy állandó értékről van szó, ezért csak kimenettel rendelkezik. A dimenzió-beállítástól függően generálhatunk vele skalár, vektor vagy mátrixot. A Constans blokk kimeneti jele megegyezik az állandó értékként szereplő paraméterrel, azonban a kimenet típusa megváltoztatható.



16. ábra *Konstans*

A fő ablaktábláján adható meg az étéke, megadható minimális és maximális érték is.

Integrátor (17. ábra): feladata a kimenetek előállítása a bementetek integrálásával. A blokk kimenetét a következő egyenletek (1) képviselik:

$$y(t) = \int_{t_0}^{t} u(t)dt + y_0$$

$$x = y(t)$$

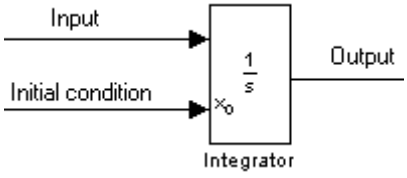
$$x_0 = y_0$$

$$\dot{x} = u(t) \qquad (1)$$

A párbeszédpanelben lehetőség van:

- Alsó, felső korlátok létrehozására
- Kezdeti érték megadására
- Újraindításra

A kezdeti feltételek megadásakor kétféle lehetőségünk van. Használhatunk belső paramétereket, ekkor csak az Input bemenet áll a rendelkezésünkre, vagy dolgozhatunk külső paraméterrel is, ekkor megjelenik egy következő bemeneti port is a blokkon. Amennyiben korlátozó feltételeket akarunk megadni a kimenetre, a Limit output jelölőnégyzettel tehetjük meg.



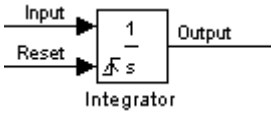
17. ábra *Integrátor*

Ebben az esetben meg kell adni az alsó és felső határértékeket (korlátokat). Lehetőség van a telítettség jelzésére is a Show saturation jelölőnégyzet segítségével. Ekkor egy újabb kiment jelenik meg a blokkon.

A kimenet értéke 3 érték egyikét veheti fel:

- 1 érték jelzi a felső telítettségi határt
- 0 érték esetén egyik határértéket sem értük el
- -1 érték esetén pedig elértük az alsó telítettségi szintet.

A kezdőértéket visszaállíthatjuk egy úgynevezett külső reset segítségével, amelyre egy újabb bemeneti port jön létre (18. ábra).

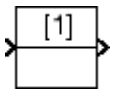


18. ábra *Reset inputtal rendelkező Integrátor*

A resethez 5 mód áll a rendelkezésünkre:

- Rising (emelkedő) abban az esetben állítódnak vissza a kezdőértékek, ha a jel 0 vagy negatív értékről pozitívra változik.
- Falling (Csökkenő) a kezdőértékek visszaállítása akkor történik, ha a jel értéke pozitív értékről 0, vagy negatív értékre csökken.
- Either beállítást használva az újraindítás akkor történik meg, ha a jel értéke 0-ról változik.
- Level esetében arról van szó, hogy visszaállítás történik, ha a jel nem 0 az aktuális időpontban, vagy pedig nem 0 értékről 0-ra vált.
- Level hold esetén akkor történik reset, ha a jel értéke nem 0 értékű az adott időpontban.

IC blokk (19. ábra): segítségével kezdeti értékeket hozhatunk létre. Be és kimenettel egyaránt



rendelkezik. A szimuláció indításakor a megadott érték lesz a kimenete, viszont ezután már a bemeneti értéket adja tovább a kimenetre.

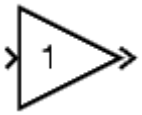
19. ábra
IC blokk

Ha az IC-blokk örökli, vagy meghatároz egy nulla minta időeltolódás (t_{offset}), az IC kimenete t időpontban,

$$t = n * t_{period} + t_{offset}\ t = n * t_{idő} + t_{offset}$$

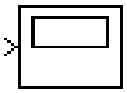
ahol n az a legkisebb egész szám úgy, hogy $t \geq t_{start.}$.

Szorzás (20. ábra): a matematikai értelemben vett szorzásra kell gondolnunk. Paraméterként megadhatjuk a szorzó értékét, és megjelölhetjük, hogy elem- vagy mátrixszorzásról van e szó. Dolgozhatunk valós vagy komplex skalárral, vektorral, vagy mátrixszal, bármely numerikus adattípus használható. Alapértelmezett értéke az 1.



20. ábra
Szorzás

Scope (21. ábra): a szimuláció során keletkezett jelek megjelenítésére szolgál.



Az így létrejövő ablakot mozgathatjuk, átméretezhetjük, a Scope paramétereit változtathatjuk a szimulációnk során.

21. ábra
Scope

A blokk a bemenetek számának megfelelő koordinátarendszert használ. A bemenő adatokat a szimulációs idő függvényében jeleníti meg. A koordinátarendszerben lehetőség van a tengelyek méretének megadására. Nagyíthatjuk a görbéket.

Ground (talaj): ezt az elemet akkor használjuk, ha valamely blokk bemenetére nem szeretnék más blokkot kötni. A blokk kezdőértéke 0.

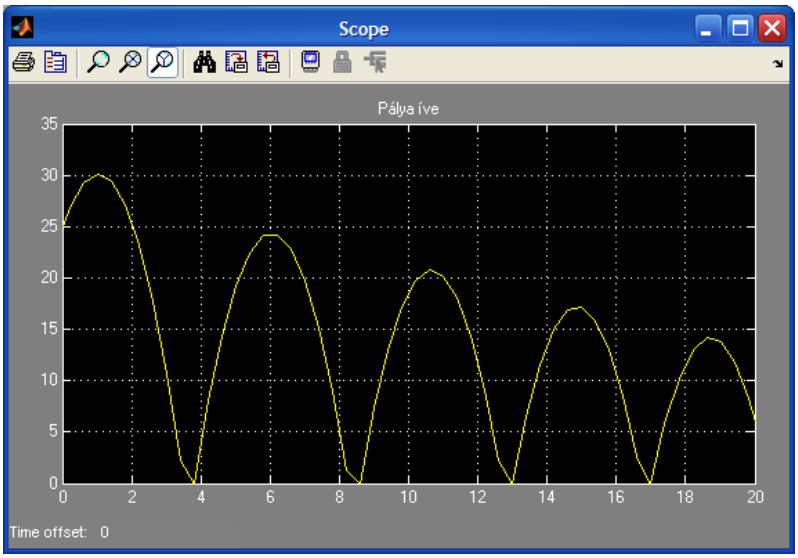
Terminátor (vég elem): abban az esetben alkalmazzuk, ha bármely blokk outputjára nem szeretnénk már más blokkot kötni, kimeneti értékével nem szeretnénk már további műveleteket végrehajtani.

Ezen elemek használatával kiküszöbölhető, hogy a modell szerkesztése után a futtatáskor hibaüzenetek jelenjenek meg hiányzó kapcsolatok miatt.

A pattogó labda pályája

A labda, mint a tárgyak nagy része elliptikus pályán esik le. Esés közben a gravitáció miatt természetesen gyorsul. A felpattanás legmagasabb pontján a golyó súlytalan, tehát mozgási energiája és a gravitáció pontosan egyensúlyban van. Mielőtt a golyó földet ér, megnyúlik, majd miután leér, felpattan, és ismét megnyúlik. Lényeges, hogy milyen gyorsan nyeri vissza a köralakját a golyó. A golyónak pattogás és nyúlás közben saját tömegét meg kell tartania.

A labda pályája a Simulink segítségével egyszerűen kirajzolható (22. ábra). Blokkdiagramja igen népszerű, elkészítése egyszerű, ezért is érdemes első lépésként ennek a feladatnak a reprezentálása.

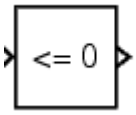


22. ábra P*pattogó labda pályája Simulinkben*

A blokkdiagram felépítése

A program működése a blokkdiagramról leolvasható (24. ábra). Két kezdeti feltétel van megadva, mely a sebességre, és arra vonatkozik, hogy a labda milyen magasról induljon. Az IC1 tartalmazza az indítás magasságát, ebben az esetben ez most 25 m, az IC kettő pedig a sebesség kezdeti értékét, ami most 20 m/s. Az Integrátor State értéke megegyezik az Output, azaz a normál kimenet értékével, visszaállításkor játszik fontos szerepet ez a kimeneti port.

A State értékét megvizsgáljuk minden egyes alkalommal, hogy 0 értékű e, erre a Compare To Zero blokk (23. ábra) áll a rendelkezésünkre.

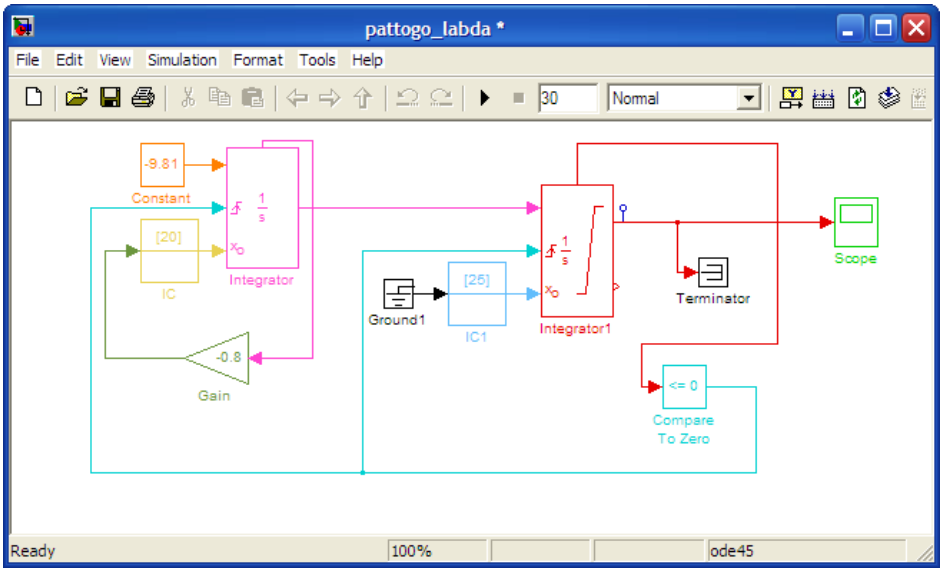


Egy boolean típusú értéket fog visszaadni (1, 0).

Ha az értéke igaz, abban az esetben az Integrator újraindul, az aktuális sebesség értékével újraszámol.

23. ábra

Compare to Zero



24. ábra *Pattogó labda blokkdiagramja*

A Gain értéke -0.8, ami az a labda rugalmasságát jelzi, a sebesség változtatásban van lényeges szerepe, azaz minden földet érés után a labda 20%-ot veszít előző sebességéből.

A program elkészítéséhez még nyitó, és záró elemekre van szükségünk (Ground, Terminator) valamint egy Scope-ra. A Scope szolgáltatja nekünk a labda pályájának képét.

Beállítását tekintve az y tengely maximális és minimális értékeinek beállítása lényeges lehet. A labda 25 méteres magasságból indul, ezért egy ennél nagyobb maximális értéket kell választanunk. Mivel az y tengely negatív részére nincs szükségünk, azaz a labda nem pattan a földszintnél mélyebbre, ezért a minimális értéknek célszerű 0-t választani. Ezen beállításokat futtatás közben is megtehetjük, az Axes propertiesben. De használhatjuk egyszerűen az Autoscale opciót is, ekkor a skálázás automatikusan történik.

8. Összefoglalás

Témaválasztásom célja elsősorban az volt, hogy betekintést nyerjek a számítógépes szimuláció világába, megismerjek olyan programokat, amik ezen szimulációk elkészítését segítik elő. Már egészen egyszerű problémák sem mindig oldhatók meg analitikus eszközökkel, megoldásukhoz numerikus eszközökre van szükség. A nagy mennyiségű számítások gyors elvégzésére képes számítógépek forradalmasították a tudományos kutatások módszereit. Lehetőséget adtak olyan nem lineáris és komplex jelenségek kiszámítására is, amelyekre azelőtt nem volt mód.

Szakdolgozatom első felében a fizikai hátteret ismertetem, mit is takar a mechanika, milyen részterületekből épül fel, valamit a modellezés és szimuláció fogalmát vezetem be további olyan információkkal, melyek fontosak lehetnek a teljes megismerésükhöz.
A következő nagyobb részben olyan szimulációs programokat ismertetek egészen az egyszerűbbektől a bonyolultabbakig, melyekkel mechanikai kísérletek szimulációja lehetséges.
Betekintést kaphatunk arról is, hogy a hétköznapi életben milyen szerepe is van a modellezésnek és szimulációknak, kezdve az oktatástól a kémián, gyógyszerészeten át a hétköznapi életig.

A szakdolgozatom írása során azért döntöttem amellett, hogy több szimulációs programot használok fel, és ezeken keresztül mutatom be- egy-egy szimuláció elkészítésének folyamatát, mert elég széleskörű választékkal rendelkezik a mai szimulációs programok listája.
A felhasználó a feladat típusához, elvárásaihoz mérten választhat, hogy mely program áll a legközelebb magához a felhasználóhoz, és a feladat jellegéhez.

A későbbiekben szeretném még az Easy Java Simulations használatához szükséges ismereteimet kibővíteni, és további, hasonló témakörrel foglalkozó szimulációs rendszereket megismerni.

9. Irodalomjegyzék

[1] Vonatkoztatási rendszer
http://hu.wikipedia.org/wiki/Vonatkoztat%C3%A1si_rendszer
2009.11. 12.

[2] A modell fogalma
http://web.t-online.hu/eszucs7/modell/Modell.htm#mfogalma
2009.11.12.

[3] Hasonlóság
http://web.t-online.hu/eszucs7/modell/Modell-2.htm
2009.11.13.

[4] A modell csoportosítása
http://web.t-online.hu/eszucs7/modell/Modellcsoportos.htm
2009.11.12.

[5] Rendszerek modellezése
http://web.t-online.hu/eszucs7/MODELL_MAT/Rendszerek_modellez.htm
2009.11.10.

[6] Szimuláció
https://miau.gau.hu/mediawiki/index.php/Szimul%C3%A1ci%C3%B3
2009.11.08.

Klasszikus mechanika
http://hu.wikipedia.org/wiki/Klasszikus_mechanika
2009.11.12.

Modellek típusai
http://hu.wikipedia.org/wiki/Modell_%28tudom%C3%A1ny%29
2009.11.10.

3D-s szimuláció készült a World Trade Center katasztrófájáról
http://itcafe.hu/hir/3d-s_szimulacio_keszult_a_world_trade_center_katasztrofajarol.html
2009.11.13.

A Phun hivatalos honlapja
http://www.phunland.com/wiki/Home
2009.11.15.

Maltlab, Simulink
http://www.jegyzet.hu/uploaded/597/iii-matl.pdf
2009.11.09.

Simulink blokkjegyzék
http://www.mathworks.com/access/helpdesk/help/toolbox/simulink/slref/f4-4889.html
2009.11.15.

Az Easy Java Simulations hivatalos honlapja
http://www.mathworks.com/access/helpdesk/help/toolbox/simulink/slref/f4-4889.html
2009.11.09.

Ferde hajítás
http://physical.blog.hu/2008/01/17/1_2_6_ferde_hajitas
2009.11.08.

Kun Ferenc: Számítógépes fizika kézirat
Debreceni Egyetem Elméleti Fizikai Tanszék
2001, Debrecen

Sudár Sándor: Számítógépes szimulációk és vizuális módszerek a fizikaoktatásban
http://www.kfki.hu/fszemle/archivum/fsz0510/sudar0510.html
2009.11.12.

10. Köszönetnyilvánítás

Szeretném megköszönni Dr. Szabó István Tanár Úrnak, hogy vállalta szakdolgozatom készítésének témavezetői szerepkörét, szakmai tudásával, építőjellegű tanácsaival, ötleteivel, és hasznos észrevételeivel támogatta, felügyelte a diplomamunkám elkészítését.