SZÜKSÉGES-E MATEMATIKA A SZOFTVERFEJLESZTÉSHEZ?

Kolumbán Sándor, vállalkozó, szoftverfejlesztő, egyetemi oktató, Székelyudvarhely

Bevezető

A címben felvetett kérdést nem kell sokat magyarázni. Sok tehetséges informatikus diáktól (és gyakorló szakmabelitől) hallom, hogy aki jó informatikus az „megúszta" a matematikát. Ezt a véleményt már csak azért sem róhatom fel senkinek, mert hajdanán én is így gondoltam. Matematikából bukásra álló informatikus gimnazistaként másképp nehéz lett volna pozitívan állni a jövőhöz. Azóta már, több mint egy évtizednyi szoftverfejlesztői tapasztalattal a hátam mögött, ezt nagyon másképp látom. Ebben a cikkben néhány tapasztalatomat és gondolatomat osztom meg abban a reményben, hogy ezzel a tehetséges informatikusok következő nemzedékét és oktatóikat is meggyőzöm a matematika tanulás és tanítás fontosságáról informatikusok számára.

Talán sokan hallottak a kalapács törvényről[1], miszerint, ha csak egy kalapács áll a rendelkezésünkre, vonzó mindent szegként kezelni. Minden szentnek maga felé hajlik a keze, ezért úgy illik, hogy előrebocsájtsam, nekem két kalapácsom van, az informatika és a matematika. A középiskolai bukást matekból elkerültem, később alkalmazott matematikusi diplomát is szereztem, és az élet nagyon sok (szinte minden) területén matematikai modellekben gondolkodom, a róluk ismert törvényeken és tételeken keresztül értelmezem a világot. Egy matematikában jól képzett informatikus (nálam hitelesebb személytől származó jellemzés) írása ez.

Három egymást kiegészítő gondolatmenetet fogok végigvezetni, amelyek számomra egységes egésszé állnak össze, de azt remélem, hogy minden olvasó számára lesz olyan érv vagy megállapítás, ami számára is meggyőző lesz. Az első részben azokat az általános matematika által élesített készségeket említem, amelyek tapasztalatom szerint elengedhetetlenek a szoftverfejlesztésben (állásinterjún ezeket mindig tesztelem is). Ezek után néhány olyan példát mutatok be a saját szakmai pályafutásomból, ahol tisztes matematikai készségek nélkül az adott projekt nem lett volna sikeres. Harmadsorban azt a fájó tapasztalatot írom le, ami több éve frusztrál, sok pénzt veszít rajta mindenki, és aminek én matematikára nyitott, abban jól képzett informatikusokban látom a megoldását.

Képességek

Véleményem szerint a két legfontosabb készség, amit csak matematika művelésével lehet fiatal korban megszerezni és később elengedhetetlenek egy szoftverfejlesztőnek, az *absztrakciós képesség* és a *teljesség igénye.*

A matematika természeténél fogva egy absztrakt nyelvezet, amivel valóvilágbeli jelenségeket és problémákat írunk le, majd a matematika eszköztárával azokra megoldást keresünk, vagy következtetéseket vonunk le. Az a képesség, amivel egy adott nyelvezettel megfogalmazott, valamilyen tématerület-specifikus problémát át tudunk fogalmazni egy másikra, elengedhetetlen egy jó szoftverfejlesztő számára. Az ügyfeleink problémáit meg kell értenünk és azt egy másik nyelvre kell fogalmazni, majd felismerni, hogy mik a jó eszközök a probléma megoldására.

Egy ilyen példa, amikor földrakások térfogatát kell megbecsülni. Autópálya építéseknél komoly probléma, hogy a megrendelt sok kamion földből tényleg az összes beérkezett-e az építkezési telepre, vagy mindegyikről le-le esett egy pár köbméter és azzal valamelyik munkavezető telkén oldódott meg a tereprendezés.

[1] Abraham H. Maslow: The Psychology of Science (1966, p. 15)

Informatikusként tudnunk kell, hogy ezt a problémát az elérhető informatikai eszköztárral hogyan lehet költséghatékonyan megoldani. Először meg kell tudnunk határozni, hogy mik a probléma lényegi összetevői. Ez a lépés tökéletesen analóg az iskolapadból ismert pármondatos szöveges feladatokkal. A különbség annyi, hogy míg Pistike kirándulás végén megmaradt almái számának (a) meghatározásához csak annyira kell rájönnünk, hogy a busz színe lényegtelen, a kezdeti almák száma (x) és a megevett almák száma (y) a releváns ($a = x - y$), addig a való életben százas nagyságrendű oldalszámú leírások és hosszú megbeszélések során válik ismertté a feladat. A projektek sikeréhez elengedhetetlen, hogy az informatikus csapat értse, hogy mit is kell megoldani. Ezekre az absztrakciós feladatokra a növekvő bonyolultságú „szöveges feladatok" megoldása az egyetlen, ami felkészíti a diákokat.

A probléma megértése és modellezése után a következő lépés annak megoldása. Nagyon sok esetben van egy kezdeti ötlet, kézenfekvő rövidítő, amivel egy *elég* jó megoldás adható a problémára. Szoftverprojektek esetében ez abban testesül meg, hogy a fejlesztés elindul egy irányba, amivel ígéretes eredmények érkeznek, majd később kiderül, ez a megközelítés csak részlegesen oldja meg a problémát. Ami még rosszabb, hogy ezt már a legelején is tudni lehetett volna. A földrakásos példánál maradva, ilyen lehet egy olyan megközelítés, ahol a helyszínen geodéták által megadott méréseket viszünk be egy mobilalkalmazással egy programba. Erről később kiderül, hogy a geodéták drágák, nem elérhetőek, és hajlamosak lehetnek a haver tereprendezését segíteni. Ezeknek a kezdeti rossz irányoknak a következménye, hogy a projekt egy késői fázisában derül ki, hogy majdnem elölről kell kezdeni mindent. Itt szokták elhajtani az informatikusokat.

Az ilyen kezdeti melléfogások elkerüléséhez elengedhetetlen az a matematikában lépten nyomon előkerülő irányelv, hogy csak a teljes, minden körülményt figyelembe vevő megoldások jók. Az $a = bx + c$ típusú feladatokra nem elég jó megoldás az $x = \frac{a-c}{b}$, hiszen kezdeni kell valamit azzal az esettel is, ha $b = 0$. Ezt a teljességre törekvést, az erre való igényt számomra a matematika egyre bonyolultabb tételeinek kimondása és bizonyítása testesíti meg. A tételek kimondása esetén mindig az a cél, hogy beazonosítsuk, milyen helyzetekben működik egy adott algoritmus vagy állítás. A bizonyítások során pedig minden lehetséges helyzetre ki kell térni, csak így lesz teljes a bizonyítás és használható a tétel. Ugyanez igaz a szoftverekre is. Az a szoftver, ami a felhasználási eseteinek csak 99%-ában működik helyesen, az valójában teljesen használhatatlan.

Ez a teljességre való törekvés, és annak valamilyen szinten formális igazolása magunk és a kollégák felé elengedhetetlen az új technológiák koncepciójának megértésében vagy bonyolult szoftverarchitektúrák tervezése során.

Egyébként a földrakás-mérős feladat egyik, a jelenlegi technológiával kiváló megoldása, hogy távvezérlésű robotrepülőkkel több irányból fényképeket készítenek a földhalmokról[2]. Ezekből később bonyolult képfeldolgozási módszerekkel ki lehet számolni a földhalmok térfogatát. Így napi szinten válik lehetővé egy akár 50 km hosszú autópályaszakasz összes építkezési helyszínének felmérése. Ezáltal követhető, hogy az építkezés az igért ütemben halad-e, illetve az éjszakai szarkák elhordása is detektálható.

Példák

Az előző, elvontabb érvelés után néhány konkrét példát hoznék fel, amiken én is dolgoztam. Ezeket úgy válogattam össze, hogy a középiskolás anyag ipari felhasználását is illusztrálják, valamint kedvenc témám, a valószínűségszámítás is érintett legyen. Az olyan alapvető matematikai fogalmak, mint vektoranalízis, mátrixok, projekciók, határértékek, integrálok, deriváltak már a középiskolás anyagnak is részei. Ezen kívül tipikusan az egyetemi képzésben kerül terítékre a valószínűségi eloszlások, határeloszlások és sztochasztikus folyamatok témaköre.

[2] `http://ventustech.hu`

Szélerőmű szimuláció

Az első példám egy tengeri szélerőmű farm szimulációs szoftvere. Ennek a projektnek a célja, hogy egy szélerőműfarm teljes működését szimulálja (akár az építés első kapavágásától az építésen keresztül a napi ügymenetig), és ebből különböző adatokat meghatározzon. A legfontosabb az adott idő alatt megtermelt villamosenergia értékesítéséből származó bevétel, illetve a hajók és szerelők aktuális helyének követése. A bevétel (B) a piaci energiaár ($c(t)$) és az aktuálisan megtermelt energia ($p(t)$) szorzatának idő szerinti integrálja: $B = \int p(t)c(t)\mathrm{d}t$.

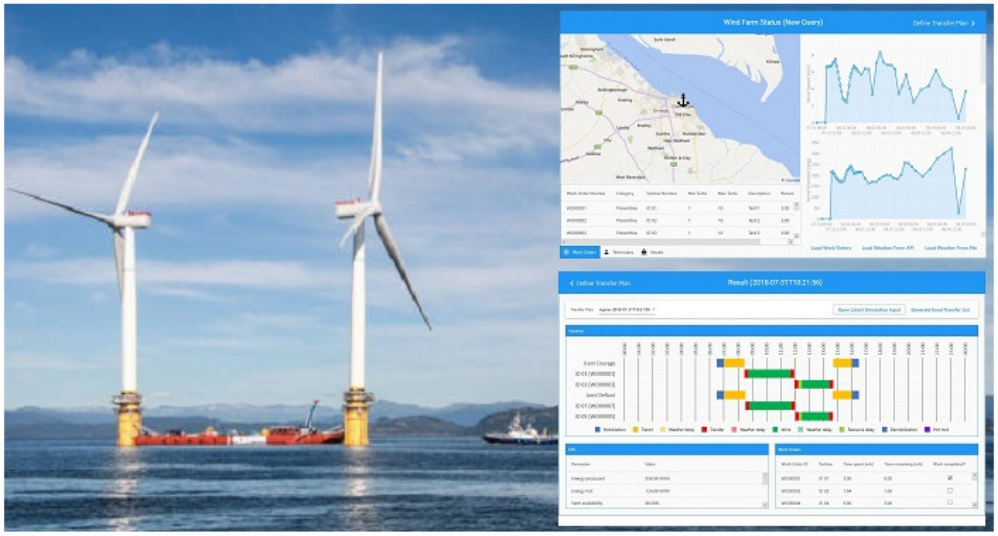
Ez az integrál a szimulációban darabonként analitikusan számolható, mert a piaci ár egy lépcsős függvény, a megtermelt energia pedig analitikusan számolható a turbinák típusa és a szimuláció számára megadott időjárásadatok alapján. A használt képletek levezetése nem túl bonyolult, de annyira specifikusak, hogy semmilyen könyvben nem fellelhetőek, így nekem kellett levezetni őket. Az így kapott formulák egyik sajátossága, hogy matematikailag helyesek ugyan, de a számítógépen nem működnek, amiért nagyrészben a numerikus hibák felelősek. Az

$$f(x) = \begin{cases} a, & x < \pi \\ b, & x = \pi, \\ c, & x > \pi \end{cases}$$

függvénydefiníció matematikailag helyes, azonban a számítógépek véges számábrázolási képessége, π irracionális volta és a kerekítési hibák miatt az $x = \pi$ feltétel a gyakorlatban sosem teljesül. Gondolhatnánk, hogy ez igazából segítség, hiszen el is lehet felejteni azt a kis jelentéktelen ágat, amibe numerikusan sosem botlunk bele. Ettől azonban a szimulációs eredmények valójában használhatatlanná válnának. A képleteket úgy kellett átdolgoznom programozás előtt, hogy bizonyos irányú határértékek különféle, a szélerőművek működéséből adódó feltételeket teljesítsenek, így sikerült egy többéves munka alapjait lefektetni.

Ugyanebben a projektben a hajók és technikusok helyét térképen kell megjeleníteni. Mivel a szélerőművek tipikusan a földgömbnek csak egy kis részét fedik le, ezért elég kétdimenziósnak és síknak tekinteni a tengert a szimulációs térben. Azonban a szimulációs tér és a valós földrajzi koordináták közötti transzformációk meghatározása és implementálása 3D geometriai ismereteket igényelt.

A fentieken kívül a projektben különösen fontos szerepe van a meghibásodások és egyéb véletlen jelenségek szimulálásának, amik számomra a legkedvesebb feladatok általában.

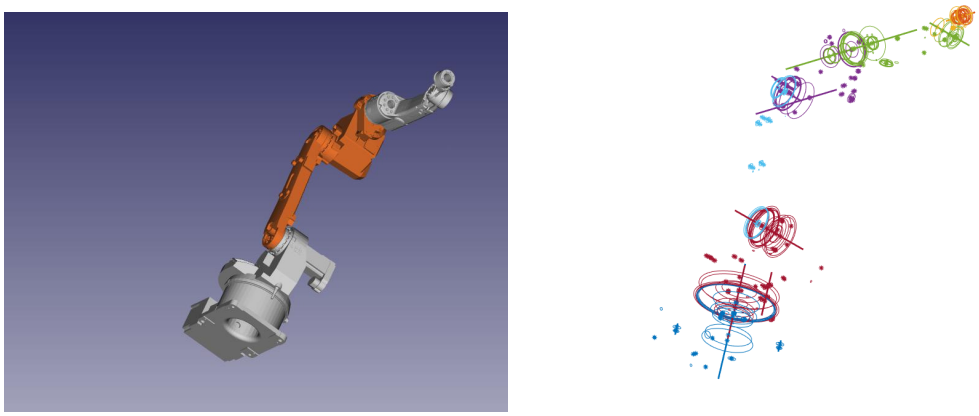


1. ábra. Képernyőkép a szélerőmű szimulációs rendszerből

Robot modellek feldolgozása

A második példaprojektemben háromszabadságfokú robotkarok 3D modelljeinek feldolgozása és azok kiterjesztett valósággal történő megjelenítése a feladat[3]. Ehhez a feladathoz egy kulcslépés, hogy a robotkar csuklóinak forgástengelyeit olyan háromdimenziós rácsvázak alapján kell meghatározni, amelyek csak a robotok külső felületét írják le.

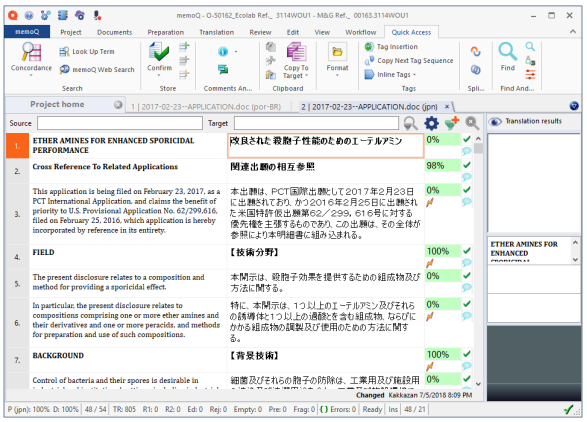
Ennek a feladatnak a megoldásához elengedhetetlen volt az alapvető mintafelismerési algoritmusok ismerete[4]. Ezeket úgy kellett átalakítani, hogy az aktuális feladathoz illeszkedjenek és értelmes időben le is fussanak. A munka során a 3 és 4 dimenziós vektoranalízis témakörébe tartozó ismeretek, és némi mátrixanalízis volt szükséges.



2. ábra. Robotmodell és a benne talált forgástengelyek

Automatikus szövegkiegészítés

Az utolsó példa egy fordítástámogató rendszerhez kapcsolódik. Ezekben a szoftverekben az alapvető működési modell az, hogy a fordítandó szöveget nyelvi szabályok szerint a rendszer mondatokra tördeli, majd azt egy fordító lefordítja. Ehhez természetesen szerteágazó módokon kap gépi támogatást a fordító, de az emberi fordítás és gépelés nem elkerülhető.



3. ábra. Képernyőkép a memoQ fordítórácsáról

Az egyik segítség az automatikus szövegkiegészítés gépelés közben[5]. Például a „This is a red roentgen machine.” mondat angol-magyar fordítása közben az első „r” betű leütésére azonnal

[3] `http://exliteron.com`
[4] `http://en.wikipedia.org/wiki/Hough_transform`
[5] `http://help.memoq.com/current/en/Places/predictive-typing.html`

meg kellett jelennie a „röntgen” és „röntgengép” szavaknak. A gépelés közbeni releváns találatok felajánlása olyan statisztikai alapú szöveg-feldolgozó algoritmusokat igényelt, ami hatékonyan implementálható, olyannyira, hogy két billentyűleütés között tudnia kell produkálni a fenti példához hasonló releváns találatokat.

A hatékony együttműködés hiánya

A fenti példák esetében is, de nem csak, az a tapasztalatom, hogy a nagy hozzáadott értéket képviselő szoftverek mindegyike valamilyen matematikai eredmény és az azt körülvevő technikai megvalósítás kombinációja. A Google hatalmas felhőparkja és a sok-sok kérést nagyon hatékonyan kiszolgáló infrastruktúrája mit sem érne, ha nem lenne egy olyan kereső algoritmusa, ami releváns találatokat produkálna villámgyorsan. Ennek az algoritmusnak az őse, a PageRank[6] is klasszikus sztochasztikus folyamatokhoz kapcsolódó eredményekre épült. Larry Page és Sergey Brin is olyan informatikusok, akik vették a fáradtságot, hogy matematikával foglalkozzanak, és azt informatikával kombinálva létrehozták a Google-t.

Rendszeresen olyan projektek kerülnek a látókörömbe, ahol teljesen egyértelmű, hogy bonyolult és magasszintű szoftverfejlesztői teljesítményre van szükség, de ez önmagában nem elég. Valahol a rendszer mélyén mindig olyan matematikára van szükség, ami túlmutat az átlag informatikus rutinmunkáján. Ilyenkor tipikusan az történik, hogy a megrendelő ugyanazt a feladatot több csapatnak is kiadja. Dolgoznak rajta informatikusok, akik szép mérnöki teljesítménnyel fejlesztenek egy nagyon szép szoftverrendszert, amiben kis túlzással átlagszámítás és gyökvonás a legbonyolultabb matematika. A végeredményért pedig kapnak 1 petákot. Dolgoznak rajta matematikusok, akik ügyes algoritmusokat és modelleket fejlesztenek, amik tesztadatokon és kis méretű példákon jól teljesítenek, de a valós környezetbeli adatmennyiségeken használhatatlannak bizonyulnak. Ők is keresnek 1 petákot. Ezek után jobb esetben a megrendelő össze tudja a kettőt integrálni és 2 petákért van egy tuti megoldása. Rosszabb esetben mindkét megoldás megy a fiókba, mert a megrendelő sem rendelkezik kellő tudással az összekombináláshoz.

Ha olyan csapatok dolgoznak az ilyen feladatok megoldásán, amelyekben vannak informatikusok, matematikusok és *matematikában jól képzett informatikusok*, akkor előfordulhat, hogy $1+1>2$, azaz a megrendelő sokkal többet hajlandó fizetni a magas szintű matematikára építő, mérnökileg is kiváló megoldásért. Az ilyen megoldások megtervezésére és kidolgozására azonban csak olyan informatikusok képesek, akik az alapvető matematikai eszközök széles tárházával rendelkeznek és értik azt a nyelvet, amin egy matematikussal lehet kommunikálni.

Zárszó

A fenti példák és a felsorakoztatott érvek segítségével remélem sikerül meggyőznöm minél több informatikusi pályára készülőt, hogy a középiskolai és az alapvető egyetemi matematika anyag készségszintű ismerete és az ennek elsajátítása során elvégzett gyakorló munka olyan képességeket fejleszt ki bennük, amelyek elengedhetetlenek a jó szoftvermérnökké váláshoz. Sőt továbbmegyek, szerintem nincs is más olyan mód a matematika gyakorlásán kívül, ami ezeket a képességeket fejlesztené. Erre minden informatikusnak szüksége van valamilyen szinten.

Aki kellő mennyiségű és mélységű matematikai ismeretet szed magára a fenti folyamat során, abból válhat olyan informatikus, aki összekötő szerepet tud betölteni a matematikusok és más informatikusok között azokban az $1+1>2$ típusú projektekben, amik a szakma javát adják.

Bár lehetne még folytatni a felsorolást, a mesterséges intelligencia térhódításának közepette talán nincs szükség több érvre amellett, hogy elengedhetetlen a matematika a szoftverfejlesztéshez.

[6] `http://hu.wikipedia.org/wiki/PageRank`