BUDOVANIE VIRTUÁLNEHO LABORATÓRIA A NÁVRH REGULÁTOROV CEZ SIEŤ

Martin Foltin, Samuel Bartoš, Michal Kollárčik, Ivan Sekaj

Fakulta elektrotechniky a informatiky , Slovenská Technická Univerzita,
Ilkovičova 3, 812 19 Bratislava, Slovenská Republika

ABSTRAKT

Práca sa venuje tvorbe a rozvoju internetových aplikácií s použitím prostriedku Matlab WebServer. Užívateľom poskytuje formou interaktívneho formulára analýzu lineárnych systémov ako aj syntézu PID regulátorov pomocou genetických algoritmov. Nároky na softvérové i hardvérové vybavenie klienta sú pritom minimálne. Požadované je iba pripojenie na Internet.

1. Úvod

V súčasnosti viacero technických univerzít v Európe a aj vo svete má vytvorené alebo je v štádiu budovania tzv. virtuálnych laboratórií, ktoré predstavujú programové prostredie prístupné cez sieť (cez Internet) a umožňujúce elektronické vzdelávanie a simuláciu prevádzky reálnych procesov v technologickej praxi, prípadne diaľkové spúšťanie fyzikálnych modelov technologických procesov. Používatelia týchto virtuálnych laboratórií sa po pripojení na príslušnú webovskú stránku môžu jednak v textovej forme vzdelávať čítaním hypertextových materiálov a jednak môžu spúšťať a ovplyvňovať počítačovo simulovanú prevádzku virtuálnych procesov. Predkladaná verzia virtuálneho laboratória predstavuje prevratný spôsob riešenia výpočtu. Použitá softvérová architektúra presúva totiž ťažisko výpočtu od koncového užívateľa na server. Tým pádom sa odbúrava potreba špecifického a hlavne finančne náročného softéru u užívateľa, ktorý si vystačí s bežným internetovým prehliadačom.

2. Internetová aplikácia Ural

Technické riešenie internetovej aplikácie Ural (`http://ural.elf.stuba.sk`) vyplynulo jasne z povahy dominantného programového prostriedku Matlab – Webservera. Ide o klasickú internetovú klient-server aplikáciu doplnenú ďalšími nie nevyhnutne potrebnými internetovými nástrojmi akými sú skriptovací jazyk php (Personal home page Hypertext Preprocessor) a databázový server mysql (sql pre structured query language).

Nevyhnutné hardvérové vybavenie na strane servera tvorí konfigurácia umožňujúca štandardnú prácu s produktom Matlab R13. Matlab-Webserver je program, ktorý beží ako služba (service), a preto vyžaduje operačný systém Windows NT, 2000 resp. XP alebo Linux. Počas práce sme úspešne testovali pod OS Windows (NT4.0, 2000 a aj XP). Ostatný softvér zabezpečujúci služby komunikácie – webserver Apache – má minimálne nároky. Na strane klienta sa požaduje štandardný internetový prehliadač. Na obidvoch stranách je potrebné pripojenie do siete Internet.

Princíp aplikácií klient-server spočíva v komunikácii cez požiadavkový dialóg. Využíva sa výhradne protokol http (hyper-text transfer protokol), takže akákoľvek práca sa javí ako prehliadanie a vypĺňanie formlárov klasickej www-stránky.

Jadro internetovej aplikácie Ural tvorí Matlab R13 s nainštalovaným Matlab-Webserverom a službou Apache webserver, ktorá zabezpečuje kompletnú vstupno-výstupnú komunikáciu.

Po zaslaní požiadavky zo strany klienta o vygenerovanie novej stránky prebieha spracovanie na strane servera. Požiadavky prijíma vždy služba Apache určujúca ktorý program cez cgi (common gateway interface) má vykonať sekundárne spracovanie. V našom prípade to je Matlab-Webserver (matweb.exe). Ten tvorí akýsi most medzi www-rozhraním a prostredím Matlab-u. V požiadavke zo strany klienta je presne určené ktorý m-file sa má spracovať a zaslané sú taktiež niektoré premenné a ich hodnoty. Na serveri permanentne beží Matlab, ktorý spravuje všetky výpočty. Výsledky simulácie sú spätne cez Matlab-Webserver a Apache zaslané klientovi vo forme html (hyper-text markup language) dokumentu esteticky prezentujúceho výstupné údaje (čísla, tabuľky, grafy, atď,...)

Keďže ide o klasickú internetovú komunikáciu cez protokol http, zbieranie údajov na následnú simuláciu je možné riešiť jednoducho, ale pre užívateľa nie veľmi príjemne: jeden veľký formulár. V tomto prípade statický formulár, ktorý si nevie zapamätať už raz vložené údaje, a pri opakovanej simulácií sa vyžaduje opätovné vypĺňanie formulára. Často sa to môže stať značnou prekážkou v práci. Preto sme vytvorili za pomoci jazyka php a databázy mysql program – dynamicky generovaný formulár, ktorý dokáže postupne od uživateľa vyzbierať potrebné údaje a vie ich dočasne skladovať v sql databáze. Identifikácia jednotlivých užívateľov prebieha pomocou identifikačného reťazca phpsessionid. Na základe toho je každému návštevníkovi priradený jeden riadok tabuľky v databáze, kde sa ukladajú a modifikujú nastavenia jednotlivých premenných (prenos, regulátor, obmedzenia, atď,...). Zároveň si uživateľ tento reťazec pomocou internetového prehliadača cez cookies (koláčiky) ukladá a v prípade opätovnej návštevy bude môcť využiť hodnoty, s ktorými v minulosti pracoval.

Sprievodný program, tiež šaman alebo wizard, asistuje uživateľovi na každom jeho kroku pri práci na Urale. Navigácia je jednoducho intuitívna pomocou zaužívaných tlačítok ďalej (next), späť (back) a štart. Šaman umožňuje uživateľovi zadať prenosovú funkciu vyšetrovaného systému a ponúka mu jeho analýzu za pomoci základných funkcií Matlab-u. Ďalej je možné z hlavného menu simulovať odozvu systému na skokovú zmenu želanej hodnoty, prípadne si otestovať nastavenie vlastného PID regulátora. Hlavnou funkciou aplikácie Ural je ale návrh parametrov PID regulátora pomocou genetických algoritmov ďalej (GA) [1, 2].

3. Číslicová simulácia jednoduchého uzavretého regulačného obvodu

Našim cieľom pri programovaní číslicovej simulácie uzavretého regulačného obvodu pre internetovú aplikáciu Ural bolo skrátiť výpočtový čas simulácie na minimum, pri zachovaní čo najlepšej vernosti napodobnenia správania sa reálneho systému. K použitiu vlastného programu na simuláciu sme pristúpili po zistení, že zaužívaný simulačný program Matlab-Simulink nespĺňa naše náročné požiadavky na rýchlosť simulácie. Pre názornosť uveďme niekoľko údajov. Veľkosť populácie v GA, použitom v Urale je 30, čo pri 1000 generáciach (iteráciach) predstavuje nutnosť spustenia 30000 simulácií. Je nutné poznamenať, že časová náročnosť simulácie je závislá od nastavenej doby simulácie, periódy vzorkovania a od zložitosti zadaného systému. Pri nastavenej dobe simulácie 10 sekúnd, perióde vzorkovania 0,1 sekundy a použití systému 3. rádu bez dopravného oneskorenia bežal GA pri simulovaní Matlab-Simulinkom rýchlosťou približne 1 generácia (30 simulácii) za sekundu. Po následnom nahradení Matlab-Simulinku našou vlastnou číslicovou simuláciou sme dosiahli rýchlosť až 100 generácii (3000 simulácii) za sekundu[1]. Veľkú zásluhu na tom má Just-in-time accelerator implementovaný vo verzii Matlab 6.5 (Release 13).

Samotné jadro simulácie pracuje na báze riešenia diferenčných rovníc. Z toho vyplýva nutnosť získania prenosovej funkcie systému v z-oblasti. V internetovej aplikácii Ural má uživateľ na výber, či zadá spojitý alebo diskrétny prenos. Ak zadá prenos diskrétny,

[1] Test rýchlosti výpočtu bol vykonaný na počítači s procesorom Intel P4 1,8GHz, WindowsXP

nepotrebujeme robiť žiadne úpravy. V prípade, že zvolí prenos spojitý, vystačíme si v Matlabe s príkazom *tf* na vytvorenie prenosovej funkcie zo zadaného čitateľa a menovateľa, následne túto prenosovú funkciu prepočítame so zadanou periódou vzorkovania na diskrétnu príkazom *c2d* a nakoniec pomocou príkazu *tfdata* získame čitateľa a menovateľa v z-oblasti v potrebnom tvare. Dopravné oneskorenie v spojitom prípade prepočítame na oneskorenie vo vzorkách a príslušným počtom núl rozšírime polynóm menovateľa prenosu v z-oblasti. Prenos kauzálneho systému v z-oblasti bude v tvare:

$$G_P(z) = \frac{b_1 z^{-1} + b_2 z^{-2} + \ldots + b_m z^{-m}}{a_0 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_n z^{-n}} = \frac{Y(z)}{U(z)}$$

Z uvedeného tvaru prenosovej funkcie vyplýva nasledujúca diferenčná rovnica:

$$y(k) = \frac{1}{a_0} * \left[b_1 u(k-1) + b_2 u(k-2) + \ldots + b_m u(k-m) - a_1 y(k-1) - a_2 y(k-2) - \ldots - a_n y(k-n) \right]$$

kde $n \geq m$

Výpočet uvedenej diferenčnej rovnice sa realizuje v jednoduchom cykle *for* a výsledok výpočtu *y(k)* (hodnota výstupu v k-tom kroku) sa uloží do vektora, ktorý bude slúžiť na výpočet akčného zásahu, na vyhodnotenie účelovej funkcie pre potreby GA, prípadne na grafické znázornenie priebehu výstupnej veličiny. Pred výpočtom prvej vzorky je však nutné vynulovať prvých *n* prvkov vektorov *u*, *y* a *w* (*w* bude použité pri výpočte akčného zásahu, je to vektor konštánt, ktorých hodnota je rovná žiadanej hodnote), kde *n* značí rád menovateľa diskrétneho prenosu (vrátane dopravného oneskorenia). Z toho vyplýva, že v prípade grafického znázornenia priebehov prvých *n* prvkov vektorov nevykresľujeme.

Po výpočte vzorky výstupnej veličiny nasleduje výpočet vzorky akčného zásahu, z ktorej sa následne vypočíta nasledujúca vzorka výstupnej veličiny atď. V prípade, že vypočítaná vzorka akčného zásahu prekračuje hodnotu obmedzenia, stanovenú užívateľom, nastaví sa aktuálna vzorka akčného zásahu na hodnotu obmedzenia. Simulácia v internetovej aplikácii Ural, umožňuje taktiež pridanie šumu merania na výstup. Ak užívateľ zvolí amplitúdu šumu *a*, tak sa ku každému prvku výstupného vektora pripočíta náhodné číslo z intervalu <-*a*,*a*>. Výpočet akčného zásahu je potom realizovaný z už zašumeného vektora *y*. V simulácii v Urale sú implementované tri diferenčné rovnice na výpočet akčného zásahu, pričom prvé dve sú ekvivalentné, používajú len odlišné parametre:

1.) Obdĺžniková náhrada PID regulátora – rekurzívna forma výpočtu akčného zásahu:

$$u(k) = u(k-1) + P(1 + \frac{D}{TP})(w(k) - y(k)) - P(1 - \frac{TI}{P} + \frac{2D}{TP})(w(k-1) - y(k-1)) + \frac{D}{T}(w(k-1) - y(k-1))$$

P – proporcionálne zosilnenie regulátora
I – integračné zosilnenie regulátora
D – derivačné zosilnenie regulátora
T – perióda vzorkovania regulátora

2.) Obdĺžniková náhrada PID regulátora – rekurzívna forma výpočtu akčného zásahu:

$$u(k) = u(k-1) + K(1 + \frac{T_D}{T})(w(k) - y(k)) - K(1 - \frac{T}{T_I} + \frac{2T_D}{T})(w(k-1) - y(k-1)) + \frac{KT_D}{T}(w(k-1) - y(k-1))$$

K – zosilnenie regulátora

T_I – integračná časová konštanta regulátora
T_D – derivačná časová konštanta regulátora
T – perióda vzorkovania regulátora

3.) Takahasiho rekurzívna forma výpočtu akčného zásahu

$$u(k) = u(k-1) + K_P(y(k-1) - y(k)) + K_I T(w(k) - y(k)) + \frac{K_D}{T}(2y(k-1) - y(k-2) - y(k))$$

K_P – proporcionálne zosilnenie regulátora
K_I – integračné zosilnenie regulátora
K_D – derivačné zosilnenie regulátora
T – perióda vzorkovania regulátora

Hlavný cyklus, ktorý obsahuje dva podcykly na výpočet výstupu a akčného zásahu končí po výpočte poslednej vzorky, ktorej poradové číslo je zrejmé z nastavenej doby simulácie a periódy vzorkovania. Výsledkom simulácie sú vektory *u* a *y*, ktoré budú použité na vyhodnotenie účelovej funkcie.

4. Názorný príklad návrhu parametrov PID regulátora

Uvažujme model jednosmerného motora, daný prenosovou funkciou

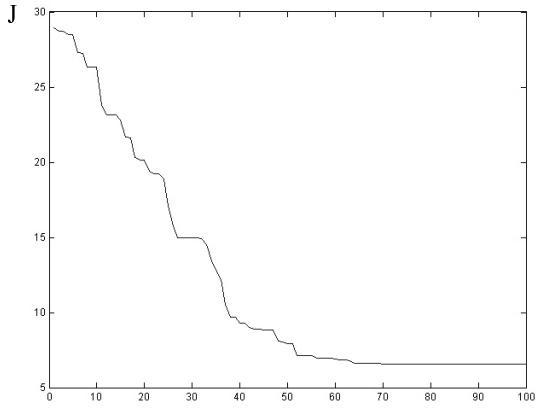
$$F(s) = \frac{1}{s^3 + 4.7s^2 + 7.177s + 2.35}$$

Zvolíme nasledovné parametre:

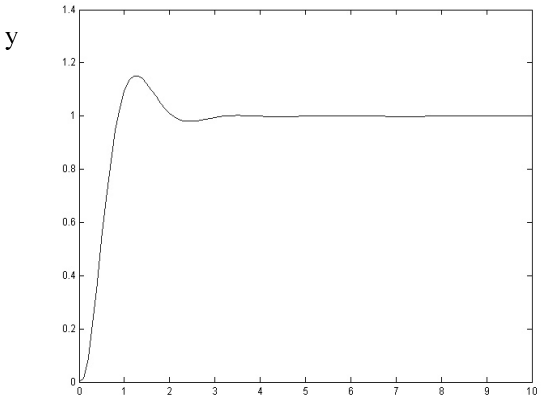
- rozsah parametrov 1. typu regulátora z intervalu (0,100), akčný zásah (-100,100)
- v ďalších nastaveniach: želaná hodnota 1, amplitúda šumu merania 0, čas simulácie 10s, perióda vzorkovania regulátora 0,1 s
- ďalšie hodnoty necháme nezmenené

Po ukončení chodu algoritmu dostávame nasledovné parametre PID regulátora:

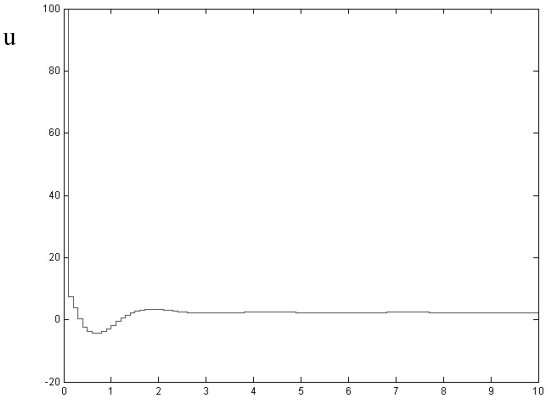
P=20.3251
I=40.566
D=9.49301



Obr. 1. Konvergencia účelovej funkcie



Obr. 2. Priebeh výstupnej veličiny



Obr. 3. Priebeh akčného zásahu

7. Záver

Naša aplikácia je príkladom ako spojením súčasných popredných internetových technológií akými sú skriptovacie jazyky a databázové produkty s výkonným výpočtovým prostredím Matlab, vznikajú užívateľsky príjemné rozhrania. Internetová aplikácia Ural (http://ural.elf.stuba.sk) umožňuje analýzu zvoleného lineárneho systému ako aj syntézu PID regulátora pomocou genetických algoritmov. V budúcnosti možno očakávať rozšírenie o ďalšie aplikácie určené na skvalitnenie výučby v oblasti automatizovaných systémov riadenia.

Použitá literatúra

1. Sekaj, I.: Genetic Algorithm-Based Control System Design and System Identification, Conferrence Mendel'99, Brno, Czech Republic, june 9^{th} –12^{th} 1999, pp. 139-144
2. I. Sekaj : Návrh genetických algoritmov, *interné učebné texty KASR FEI STU Bratislava, 2001*
3. Š. Kozák : Lineárne číslicové systémy I, STU Bratislava, 1991
4. Mathworks: Užívateľská príručka k Matlabu, verzia 6.5
5. H. Rawat: Programujeme PHP profesionálne, Computer Press, 2000

Tel : ++421 2 602 91 506 E-mail : foltin@kasr.elf.stuba.sk , samo@kasr.elf.stuba.sk sekaj@kasr.elf.stuba.sk