Technická univerzita v Liberci
Hálkova 6, Liberec

Fakulta mechatroniky a mezioborových inženýrských studií

Ročníkový projekt

Model sanace turonské zvodně

Vedoucí ročníkové práce:
RNDr. Jan Novák, Ph. D.

Vypracoval:
Martin Plešinger

Liberec 2002 / 2003

Fakulta mechatroniky a mezioborových inženýrských studií

Zadání projektu (4. ročník)

Název projektu: **Model sanace turonské zvodně**

Pro obor: **Přírodovědné inženýrství**

Řešitelský tým: **Martin Plešinger**

Vedoucí učitel: **RNDr. Jan Novák, Ph. D.**

Doporučované předměty 4. ročníku: **Stavba a řešení modelů, Moderní metody v MKP**

Zadání:

- Seznámení s problematikou sanace turonské zvodně v oblasti Stráže pod Ralskem.
- Seznámení s metodou lineárního programování.
- Přepracování stávajícího programového systému pro řízení postupu sanace:
  - zdokonalení interaktivního způsobu zadávání parametrů úlohy,
  - rozšíření počtu sanačních technologií na čtyři,
  - formulace a zařazení omezení počtu nových vrtů do optimalizačního výpočtu,
  - vyřešení zobrazování vybraných parametrů v průběhu výpočtu ve formě map,
  - zajištění uživatelsky kvalitního prostředí.
- Demonstrace práce systému – výpočet jedné varianty postupu sanace.

Požadované náležitosti:

k zápočtu v zimním semestru: aktivní práce, předložení upraveného systému obrazovkových formulářů pro zadávání vstupních dat, koncepční řešení rozšíření počtu sanačních technologií v modelu.

k obhajobě: závěrečná zpráva (20 stran), prezentace výsledků (max. 20 minut), poster (8-10 stran), resumé v angličtině (1 strana).

Termíny:

odevzdání závěrečné zprávy: 23. květen 2003

obhajoba: v 1. týdnu zkouškového období letního semestru 2003

Hodnocení:

max. 30 bodů za realizaci,
max. 30 bodů za dokumentaci,
max. 20 bodů za prezentaci,
max. 10 bodů za poster,
max. 10 bodů za resumé.
celkem max. 100 bodů

Datum zadání: 23.10.2002 Podpis vedoucího učitele:

Prohlášení

Prohlašuji, že jsem tuto ročníkovou práci vypracoval samostatně pouze za odborného vedení vedoucího ročníkového projektu Jana Nováka a konzultantů Otty Severýna, Ladislava Kastla a Hany Čermákové, kterým bych chtěl touto cestou poděkovat. Zároveň bych chtěl poděkovat Bohuslavě Jelínkové za pomoc při korektuře textu.

Dále prohlašuji, že veškeré podklady, ze kterých jsem čerpal, jsou uvedeny v seznamech použité literatury.

V Liberci 7. května 2003

Martin Plešinger

Abstrakt

Chemická těžba uranu v oblasti Stráže pod Ralskem v letech 1968 až 1996 silně poškodila tamní životní prostředí. Rozsáhlý sanační program, prováděný firmou DIAMO s. p., si klade za cíl minimalizovat tento dopad produkce uranu na životní prostředí. Proces čištění oblasti je velmi náročný a to jak časově, tak technicky a pochopitelně i ekonomicky. Proto je proces rozdělen na několik částí podle jednotlivých sanovaných podoblastí a řady časových kroků. Pro řízení sanace v jednotlivých oblastech jsou sestaveny specializované softwary – optimalizační systémy.

Při práci se stávajícím systémem pro řízení sanace turonské zvodně se objevily některé nedostatky, které bylo potřeba odstranit. Tyto změny vyplynuly zejména z praktických zkušeností a informací získaných právě v průběhu práce se starým systémem. V systému bylo nutno provést změny, které vyplynuly z nově vznikajících potřeb v souvislosti s novou koncepcí sanace, například začlenění nové technologie zpracování roztoků do systému. Dále byla v systému provedená řada změn na úrovni spíše programátorské (úpravy zdrojových textů) a změny, které se týkaly unifikace vzhledu grafického rozhraní systému a zlepšení uživatelského komfortu. Provedené změny lze shrnout především do těchto oblastí:

1) Změny související s novou koncepcí sanace:

   – začlenění nové technologie zpracování čerpaných roztoků – technologie hydraulické bariéry – BAR – do systému;

   – implementace ekonomického modelu – modulu provádějícího ekonomické hodnocení variant a jednotlivých kroků – přímo do systému;

   – ukládání dat s výsledky do souborů *xls*.

2) Odstranění nedostatků stávajícího systému:

   – sjednocení a vylepšení práce systému se soubory; vytvoření mechanismu pro mazání dočasných souborů a pro zálohování souborů; dále změny týkající se globálního nastavení systému (hlavní konfigurační *ini* soubor); zavedení lepší hierarchie do adresářové struktury systému;

   – vyřešení problému s nastavením desetinného oddělovače v celém systému a zejména v komerčním softwaru XA;

   – usnadnění zobrazení odhadů a vybraných vrtů v programu GwsView;

   – formulování a implementace mechanismu respektování existujících vrtů a omezení počtu nových vrtů při výběru optimální množiny čerpacích vrtů programem XA;

3) Další změny v systému:

   – přepracování grafického rozhraní programu; zvýšení přehlednosti a usnadnění manipulace použitím záložek; vyznačení nesplněných podmínek v případě nenalezení optimálního řešení; vytvoření nové obrazovky pro výběr variant;

   – přepracování většiny souborů se zdrojovými texty;

   – vytvoření instalačního disku. □

Předkládaná práce je stručným popisem inovovaného systému TURON 2003. Práce se týká zejména popisu grafického rozhraní systému a práce s ním. Dále jsou zde popsány všechny změny, které byly v systému prováděny v souvislosti s tímto projektem. V práci jsou dále velmi stručně popsány základy matematických metod, na kterých celý optimalizační systém stojí – základní myšlenky optimalizačních metod založených na lineárním programování a stručný principiální popis primární formulace metody konečných prvků (FEM).

Abstract

The chemical mining of uranium in region Stráž pod Ralskem in years 1968 – 1996 has seriously damaged local environment. Large remediation program executing by the enterprise DIAMO s. e. tries to minimize this negative effect to the environment. The remediation process is very burdensome task in many points (time aspect, technical aspects, economic aspects). That's why the remediation process is split into several parts, to individual subregions (geological layers) and to series of time steps. Specialized optimization software is made for control remediation in several subregions.

Application of the previous version of optimization software that controls remediation process in Turonian aquifer has showed a queue of problems. It was necessary to eliminate these problems. It was also needed to accomplish changes that emerge from connections with new developed problems and with new remediation concept. One of these problems was for example integrating a new surface technology of solutions reprocessing into the control system. Otherwise many small changes (especially: source texts modification, graphic interface layout alteration, user comfort improvement, etc.) were accomplished. All these changes can be recapitulated in these steps:

1) Changes connected with a new remediation concept:

   – New surface technology integration – it is so-called hydraulic barrier;

   – Economic model creation – this is a module, that implements economical evaluation of single time steps and the whole variant;

   – Result data saving into *xls* file;

2) Remotion of imperfections in previous system:

   – System's file management refinement (back up mechanism to settings files and results files creating, temporary files deletion mechanism, etc.); otherwise global system settings changing (main settings *ini* file); better hierarchy to directory tree installation;

   – The problem of decimal separator setting in the whole system (and particularly in XA module) resolution;

   – Display projections and selected drill holes in GwsView module facilitation;

   – System to respect existing drill holes and to restriction quantity of new drill holes definition and creation;

3) Remainder changes:

   – Graphic interface layout changing; lucidity enhancing and manipulation with program improvement (using sheets); infeasible requirements highlighting (if the optimal solution didn't found); new variant selection form creation;

   – Majority of files with source texts modification;

   – Installation disk creation. □

This work is a concise description of the whole system TURON 2003. The paper particularly concerns description of graphic interface of the system and succinct manual to manipulation with it. Next the paper contains detail description of all accomplished changes. Otherwise the document includes very succinct introduction to backgrounds of mathematical methods, that are used in this system – basic ideas of optimizing methods based on linear programming and basic fundamentals of primary definition of finite element method (FEM).

Obsah

Přílohy

1 Úvod do problematiky sanace ložiska Stráž

1.1 Ložisko Stráž pod Ralskem

Ložisko Stráž pod Ralskem se nachází v severních Čechách v oblasti, která tvoří významnou součást severočeské křídové tabule. Jedná se o uranové ložisko, na kterém v letech 1968 až 1996 probíhala intenzivní chemická těžba kyselým loužením. K loužení uranu se používá kyselina sírová – H_2SO_4. Chemická těžba probíhá víceméně tak, že se nejprve do podzemí jedněmi (vtláčecími) vrty vtláčí kyselina sírová a jinými (čerpacími) vrty se z podzemí čerpají roztoky, v kterých je již obsažen rozpuštěný uran. Je celkem pochopitelné, že část kyseliny sírové (respektive síranových aniontů) zůstává v podzemí, kde pak může chemicky reagovat i po ukončení těžby například s mateční horninou. Jak se v průběhu těžby ukázalo, bylo nutné používat při loužení větší dávky kyseliny, než se standardně pro kyselé loužení používá. Vlivem toho došlo v podzemí k velké akumulaci nečistot, které tam po ukončení těžby zůstaly. Majoritním kontaminantem jsou pochopitelně kyselina sírová (síranové ionty) a její soli rozpuštěné ve vodě. Jedná se zejména o sírany hliníku – síran hlinitý $Al_2(SO_4)_3$, sírany čpavku – síran amonný $(NH_4)_2SO_4$ a různé kamence – komplexní sírany hliníku, typickým příkladem je kamenec amonný respektive síran hlinitoamonný (mineralogický název soli je čermíkit nebo také tschermigit) – $NH_4Al(SO_4)_2$.

V místě těžby uranu existují dvě zavodněné geologické vrstvy nad sebou. Tyto vrstvy jsou částečně oddělené poloizolační vrstvou spodního turonu. Tato izolační nepropustná vrstva odděluje dva základní zavodněné kolektory – zvodně. Jedná se o zvodeň cenomanskou a zvodeň turonskou. Obě zvodně jsou geologicky tvořeny převážně pískovci, jsou však, jak již bylo výše uvedeno, odděleny navzájem téměř nepropustnou vrstvou. Přičemž situace je taková, že geologicky starší vrstvy jsou vrstvy cenomanské a leží tedy pod geologicky mladšími vrstvami turonu. Právě ve spodní cenomanské zvodni se ovšem nachází uranové ložisko. Veškeré těžební vrty proto zákonitě musí procházet i vrstvou turonu. Cenomanské vody nebyly nikdy nijak vodohospodářsky využívány právě díky přítomnosti ložiska a tedy díky přirozenému vysokému obsahu radioaktivních kontaminantů. Současně byly tyto vody bezpečně odděleny víceméně nepropustnou vrstvou spodního turonu od vod turonských. Vody svrchního turonského kolektoru naopak patří k jedněm z nejvýznamnějších zdrojů pitné vody v ČR. Tyto vody jsou tedy intenzivně vodohospodářsky využívané. A v neposlední řadě je třeba mít na paměti to, že nemalá část strážského bloku nese statut chráněné oblasti s přirozenou akumulací vod.

Původní předpoklady o minimálním dopadu těžby na ekosystém byly, jak se později ukázalo, mylné. Mimo jiné kvůli tomu, že bylo při loužení použito větších dávek kyseliny sírové, a také zřejmě kvůli podcenění geologické stavby celé oblasti. Spodnoturonský izolátor je totiž přirozeně narušen, zejména vlivem různé tektonické činnosti v oblasti, a není tedy úplně nepropustný. Velké množství vrtů, které procházejí nepropustnou vrstvou, a také ne příliš dobrý stav některých vrtů, hlavně z období počátků chemické těžby, izolační schopnosti nepropustné vrstvy ještě snižuje. Vlivem těchto poruch v izolátoru začalo docházet k průniku kontaminantů do turonské zvodně. Část kontaminantů navíc do turonského kolektoru prosákla z povrchu v průběhu těžby. Největším s tím spojeným rizikem je možnost migrace kontaminace do spodních vod, respektive do vodních zdrojů a zejména do zdrojů pitné vody využívaných v oblasti Mimoně. K takovému průsaku by mohlo dojít již v horizontu několika desítek let. Dále pak hrozí průsak kontaminace do vodoteče povodí Labe. Kontaminace Labe je sice záležitost velice dlouhodobá, nicméně zejména z ekologického hlediska se jedná o problém nepřehlédnutelný. V současné době do turonských vod migrovalo již cca 0,5% hmotnosti veškerých kontaminantů z cenomanského kolektoru, což je přibližně 30000 tun nečistot. Kontaminace však není rozložená souvisle v celé oblasti, ale je lokalizovaná v jakýchsi „čočkách“ – v místech průsaků. V těchto místech dosahují koncentrace cca 5 – 30 gramů celkového znečištění (TDS) na jeden litr vody. Největší podíl znečištěných vod však dosahuje koncentrací výrazně slabších – přibližně v rozmezí 0,5 – 1 gramu celkové solnosti na litr vody. Nesouvislé rozložení znečištění je dobře patrné na obrázku 2.3 v následující kapitole.

1.2 Sanace ložiska Stráž pod Ralskem

V souvislosti s možným hrozícím šířením kontaminace již několik let probíhá intenzivní sanace. Sanace probíhá současně v obou zamořených kolektorech (v turonské i cenomanské zvodni). Míra kontaminace turonské zvodně je sice téměř zanedbatelná ve srovnání s mírou kontaminace zvodně cenomanské, ovšem z turonských vod hrozí přímá migrace kontaminantů do vodních zdrojů. Také počet různých sledovaných látek s kritickou koncentrací je v turonských vodách menší (v turonu se sledují koncentrace iontů SO_4^{2-}, NH_4^+ a celkové znečištění, v cenomanu se navíc sleduje například koncentrace hliníku – Al). Tyto rozdíly mezi oběma oblastmi vedly k tomu, že v současné době probíhá sanace obou oblastí víceméně nezávisle. To například znamená, že jsou strategie sanace turonu a cenomanu řízeny individuálně.

Se sanací obou oblastí je pochopitelně spojena řada závažných problémů nejen technologických, ale také například ekonomických. Kontaminované roztoky je potřeba nějak dostat na povrch a s tím souvisí potřeba využívat a tedy udržovat stávající vrtná pole, která byla dříve určena pro těžbu. Současně je také potřeba vytvářet nové čerpací vrty v místech, kde není jiná možnost čerpání. Vyčerpané roztoky je ovšem potřeba na povrchu nějak zpracovávat. S tím souvisí povrchové technologie, které je potřeba pořídit, respektive udržovat v provozu, pokud již existují. Příkladem takových technologií mohou být například: odparka – pro odpařování vody z roztoků, neutralizace roztoků, čistění roztoků na membránových technologiích, výroba kamence z velmi silných roztoků, ale i například opětovné vtláčení slabých roztoků zpět do země do tzv. hydraulické bariéry nebo vypouštění těch nejslabších roztoků, které splňují hygienické normy, do vodoteče.

Pořizování a využívání technologií i udržování vrtné sítě a pořizování nových vrtů je činnost velice nákladná. Ukazuje se, že tyto problémy (minimalizace počtu vytvářených nových vrtů, maximalizace využití stávajících technologií, atd.) je vhodné řešit různými optimalizačními systémy. Obě kontaminované oblasti vykazují značné rozdíly. Z cenomanské zvodně je typicky čerpáno mnohem větší množství roztoků, které mají navíc mnohem větší koncentrace, než je tomu u zvodně turonské. Toto se projevuje zejména u používaných technologií. Zatímco u turonských roztoků se používá neutralizace roztoků (v počáteční fázi sanace) a hydraulická bariéra a vypouštění roztoků do vodoteče (v průběhu celé sanace, zejména v závěrečné fázi), u cenomanských roztoků je nutné používat i technologie, které vyžadují vyšší koncentrace roztoků, například odpařování vody z roztoků a výroba kamence. Vzhledem k rozdílům v kontaminaci a hlavně v použitých technologiích v turonské i cenomanské zvodni se pro obě oblasti používají nezávislé optimalizační a ekonomické modely.

1.3 Význam optimalizačního systému TURON 2003

Smyslem každého optimalizačního systému je vždy bezesporu zefektivnění nějaké činnosti. Systém TURON 2003 je, jak již bylo výše zmiňováno, systém k řízení sanace kontaminovaného turonského kolektoru ležícího nad ložiskem Stráž. Důraz byl kladen zejména na tamní cenný ekosystém, a to jak z pohledu čistě ekologického, tak z pohledu vodohospodářského. Sanaci takové oblasti je však potřeba řídit i s ohledem na ekonomické možnosti. Z tohoto naznačeného pohledu provádí tento optimalizační systém několik činností:

(a) Přímé řízení sanace, tedy určování odkud se má co čerpat, jakou rychlostí a na jakou technologii se mají vyčerpané látky poslat. Model určuje, kde by se měly otevřít nové vrty, a říká nám, zda je dosavadní postup sanace efektivní (zda dochází ke snižování koncentrací kontaminantů v podzemí).

(b) Modelování transportních procesů v podzemí nás přímo informuje o tom, zda například nehrozí nebezpečí přímého průsaku kontaminace do řeky, do zdrojů pitné vody atd. Z tohoto pohledu nás systém informuje o ekologické situaci v podzemí.

(c) V systému je zabudován ekonomický model, který nám umožňuje hodnotit jednotlivé zvolené strategie sanace.

Díky těmto vlastnostem systému je možné vybrat takový postup sanace, který bude v rámci možností nejrychlejší možný, ekonomicky realizovatelný a co nejméně poškozující životní prostředí.

Literatura použitá v kapitole 1

[1] Prezentace DIAMO s. p. na CD-ROM, Stráž pod Ralskem, Česká republika, 2000

[2] Application of numerical simulation system on Turonian aquifer remediation control, Čermáková, Novák , Mužák, TU Liberec, Česká republika, 2001;
Uranium in the Aquatic Environment, Proceedings of the International Conference Uranium Mining and Hydrogeology III and the International Mine Water Association Symposium, Freiberg, Germany, 15. – 21. September, 2002

[3] Vývoj a aplikace optimalizačních modelů pro sanaci horninového prostředí na ložisku Stráž, Model sanace cenomanské zvodně, Ing. Hana Čermáková, Csc., Příbram, Česká republika, 2000;
Sborník symposia Hornická Příbram 2000

[4] Encyklopedický přehled minerálů, Jan Hus Bernard, Rudolf Rost a kol., Academia Praha, Česká republika 1992 – mineralogické názvy kamenců v odstavci 1.1.

[5] Ročníková práce Realizace preprocesingu pro kalibraci 3D transportního modelu, kapitola 1.1, Technická univerzita Liberec, David Alimov, Česká republika, 2002

2 Matematické metody použité v optimalizačním systému – teoretické základy

Optimalizační model sanace turonské zvodně je realizován v systému (v programovém balíku) TURON 2003. Tento systém se skládá z několika základních součástí. První významnou součástí je grafické prostředí – systém obrazovek k zadávání jednotlivých parametrů optimalizace jako jsou: možnost zapnutí / vypnutí jednotlivých technologií, nastavení požadovaných parametrů těchto technologií (koncentrace roztoků, požadované objemy roztoků). Tento obrazovkový systém bude podrobně popsán v dalších kapitolách spolu s popisem celého programu. Druhou součástí programu je vlastní optimalizační model, založený na metodách lineárního programování. Třetí součástí je model proudění a transportu látek, založený na metodě konečných prvků. Podrobný popis funkce celého systému bude popsán níže. Tato kapitola nás má pouze stručně seznámit se základními principy použitých matematických metod.

2.1 Lineární optimalizační model – lineární programování

Ze začátku je třeba poznamenat, že termín programování v názvu této matematické metody neznamená programování v „klasickém“ smyslu tak, jak ho chápeme dnes. Nejedná se o počítačové programování, ale o programování matematické. Matematické programování je zde synonymem pro plánování, vytváření scénářů (příkladem problému LP je třeba přiřazení technologických operací na různých součástech jednotlivým strojům, které umožňuje maximalizaci celkové výroby při respektování určitých požadavků výroby a omezených kapacit strojů, minimalizaci odpadu a například optimalizaci využití energetických a ekonomických prostředků). Jedná se tedy o disciplínu spadající do matematických optimalizačních metod.

2.1.1 Úvod do lineárního programování – historie

Lineární programování patří mezi nejlépe propracované součásti matematického programování. Zabývá se teorií i numerickými metodami určování extrémů lineárních funkcí mnoha proměnných s lineárními omezujícími podmínkami. Lineární programování má v současné době mimořádně širokou oblast působnosti. Klasickou „první oblastí“ použití lineárního programování bylo plánování ekonomických aktivit. Toto označení se poprvé začalo objevovat v pracích matematiků G. B. Dantziga, T. C. Koopmanse, A. Charnese, W. W. Coopera, A. Hendersona, L. V. Kantoroviče a mnoha dalších na počátku padesátých let dvacátého století. Ačkoli problémy týkající se extremalizace lineárních funkcí s lineárními omezujícími podmínkami byly řešeny i dříve, teprve výsledky Dantzigových prací podnítily prudký rozvoj lineárního programování. Mezi hlavní klasické problémy lineárního programování patří například: tzv. dopravní problém (přeprava výrobků), směšovací úlohy, úloha plánování výroby, tzv. přiřazovací problém, atd.

2.1.2 Obecný problém lineárního programování

Jednoduchý matematický problém úlohy lineárního programování lze formulovat například následujícím způsobem.

Minimalizace / maximalizace lineární funkce f v proměnné z (viz také příklad na další straně):

$$f\left(x_1, x_2, x_3, \ldots x_i, \ldots x_n, z\right) = 0$$

Pokud platí tyto lineární podmínky:

$$g_{j=1\ldots m}\left(x_1, x_2, x_3, \ldots x_i, \ldots x_n\right) \geq 0$$

A zároveň:

$$x_{i=1\ldots n} \geq 0$$

V tomto případě máme lineární funkci n+1 proměnných a k tomu přidružených n+m lineárních podmínek.

2.1.3 Použití lineárního programování na konkrétním příkladu – metoda grafického řešení

Pro větší názornost zde uvedeme trochu konkrétnější příklad. Strojírenský podnik vyrábí dva typy výrobků A, B. Oba typy výrobků musí být nejprve odlity ve slévárně, tyto odlitky je potřeba ještě osoustružit v obrobně, pak již mohou jít hotové výrobky na odbyt. Kapacity jednotlivých technologických pracovišť jsou omezené a doba práce na jednotlivých typech výrobků je různá na každém pracovišti. Úkolem je určit výrobní sortiment z výrobků A, B, tedy určit optimální počet vyráběných výrobků A, B, dávající maximální objem výroby (měřený v cenách za prodané výrobky) a realizovatelný v daném podniku.

Celkovou situaci podniku vystihuje následující tabulka:

Pracoviště	Kapacity pracovišť (v normohodinách)	Spotřeba kapacity pracovišť na jeden výrobek (v normohodinách)	
		A	B
Slévárna	2000	20	40
Obrobna	3000	50	30
Odbyt výrobku A	1500	30	0
Odbyt výrobku B	2000	0	50
Zisk z výroby na jeden výrobek (v korunách)		60000	80000

Lineární model daného problému po triviálních úpravách:

Lineární funkce:

$$(1) \quad z = 60000x_A + 80000x_B$$

x_A – vyráběné množství výrobku A,
x_B – vyráběné množství výrobku B,
z – celkový zisk, který se snažíme maximalizovat.

Navíc musí být splněny tyto lineární podmínky:

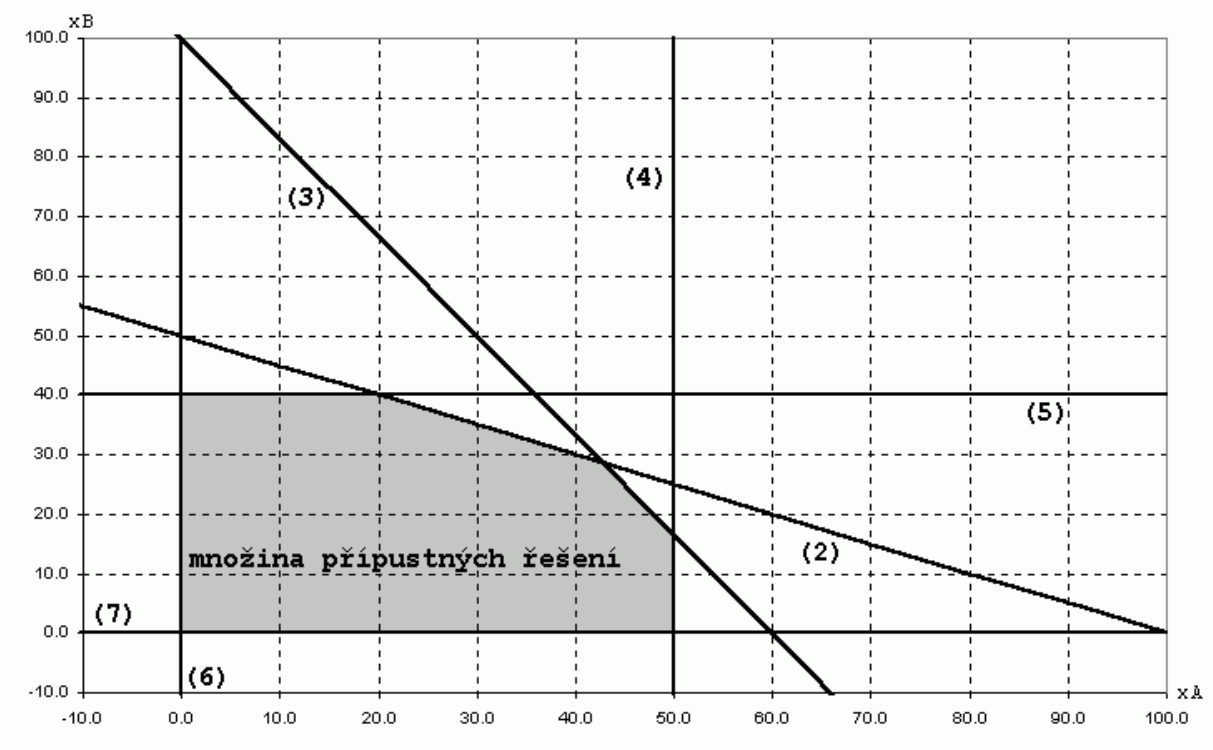
$$(2) \quad x_A + 2x_B \leq 100$$
$$(3) \quad 5x_A + 3x_B \leq 300$$
$$(4) \quad x_A \leq 50$$
$$(5) \quad x_B \leq 40$$

A pochopitelně tyto podmínky:

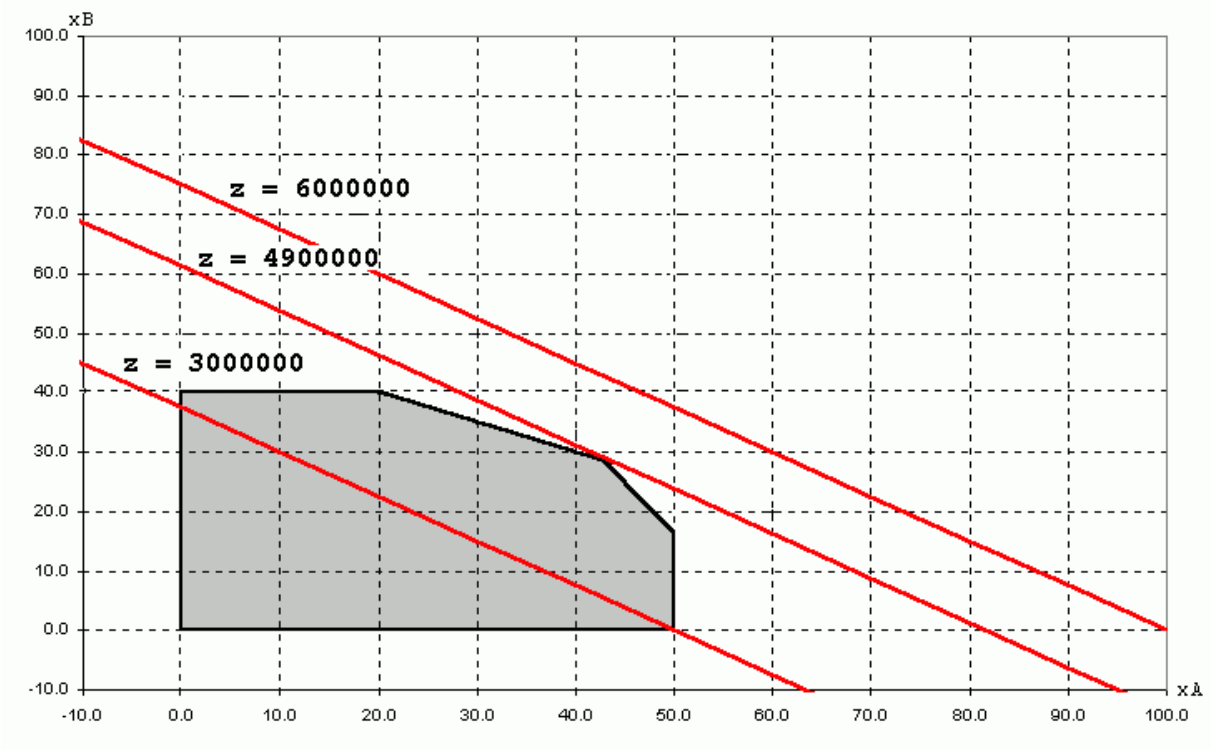
$$(6) \quad x_A \geq 0$$
$$(7) \quad x_B \geq 0$$

Takové hodnoty x_A a x_B, které splňují všech šest omezujících podmínek (2) – (7) nazýváme *přípustným řešením daného problému*. Přípustné řešení problému, které navíc splňuje vztah (1) s maximálním z, nazýváme *optimálním řešením problému*. Protože je tento problém pouze dvourozměrný (hledáme dvojici čísel x_A, x_B), můžeme snadno graficky zobrazit množinu všech přípustných řešení. Každá omezující podmínka rozdělí rovinu (x_A, x_B) na dvě poloroviny, z nichž jedna je podmínkou „vyloučena“ a druhá „povolena“. Množinu všech přípustných řešení pak dostaneme jako průnik všech „povolených“ polorovin od všech omezujících podmínek. Abychom v grafu nalezli optimální řešení, zobrazíme si izočáry funkce (1) pro různé konstantní hodnoty z. Celé grafické řešení tohoto problému je dobře patrné na následujících obrázcích (obr. 2.1 a 2.2).

Obr. 2.1 – množina přípustných řešení



Obr. 2.2 – nalezení optimálního řešení; na obrázku je vykreslena rovnice (1) pro tři různé konstantní hodnoty z



Optimální řešení v tomto případě tedy představuje bod $(x_A, x_B) = [43, 29]$. Optimální sortiment je tedy tvořen 43 kusy výrobku typu A a 29 kusy výrobku typu B (po zaokrouhlení). Objem produkce, respektive celkový zisk, v tomto případě činí 4900000 Kč.

2.1.4 Metody lineárního programování a komerční software na řešení lineárních problémů

Takovouto geometrickou interpretaci řešení lze samozřejmě přenést i do n–rozměrných úloh, a tedy do n–rozměrných prostorů. Pochopitelně, že pokud máme nějakou úlohu o mnoha neznámých řešit opravdu efektivně, nelze to provádět kreslením grafů. V současnosti je k dispozici řada různých metod vycházejících z maticové analýzy (z lineární algebry), například tzv. *simplexová metoda*. Obecně se ale vždy v principu jedná o hledání nějak definovaného bodu s extrémní hodnotou uvnitř n–rozměrného polyedru.

Zajímavou a také rozhodně nezanedbatelnou skupinu problémů lineárního programování tvoří tzv. *celočíselné problémy*. Výše uvedený příklad je jedním z nich. Tyto problémy jsou charakteristické tím, že neznámé hodnoty proměnných x_i musí být celá čísla. Tento požadavek vyplývá většinou z formulace úlohy, jako například: „Jaký počet výrobků typu A ...“, kde je pochopitelné, že počet výrobků musí být celé číslo. Taková formulace zadání nám přirozeně přidá další podmínky do lineárního modelu. Pro řešení takových problémů však většinou není potřeba používat speciální celočíselné metody. Pokud se jedná o ne příliš složité úlohy, celočíselné řešení můžeme, jako v našem případě, vytvořit pouze vhodným zaokrouhlením obecného optimálního řešení.

Pochopitelně je dnes možné (a také nezbytné) řešení úloh lineárního programování pomocí výpočetní techniky. Je k dispozici řada softwarů, které implementují metody řešení takových problémů. Jako příklad takových programů lze uvést velmi jednoduchý program STORM, který má poměrně intuitivní ovládání, jeho velkou nevýhodou je však to, že je určen pro operační systémy MS–DOS. Softwarový balík MS–Office, konkrétně program MS-Excel, také podporuje řešení jednoduchých úloh lineárního programování. Pro řešení praktických lineárních problémů s velkým množstvím neznámých většinou slouží specializované komerční softwary. Jedním z takových je i program XA, který je součástí systému TURON 2003. Které konkrétní metody lineárního programování tento software implementuje, mi není bohužel známo.

2.2 Model proudění a transportu látek – FEM

Prakticky všechny softwarové balíky sloužící pro modelování nejrůznějších fyzikálních dějů, počínaje vedením tepla v materiálech, přes programy zabývající se klasickou pružností a pevností, až po programy modelující například proudění podzemních vod a transportu kontaminace v těchto vodách, používají momentálně asi jednu z nejčastěji jmenovaných matematických metod – *FEM*. FEM je mezinárodní označení této metody a je to zkratka anglického názvu *finite element method*, česky se nazývá *metoda konečných prvků* – zkráceně *MKP*. Jedná se, podobně jako například u tradiční metody sítí, o numerickou metodu, respektive numerické metody. (FEM se v současnosti souhrnně označuje celá řada metod odvozená od základní metody konečných prvků.) FEM je momentálně asi nejefektivnější numerickou metodou, ale metody založené na konečných diferencích (metoda sítí), které nejsou tak přesné, se jí podařilo vytlačit hlavně díky velkému rozmachu a obrovskému navýšení výkonu výpočetní techniky, neboť veškeré implementace této metody jsou výpočetně velmi náročné.

2.2.1 Historie FEM

Možnost diskretizace úlohy respektive zkoumané oblasti na konečné prvky narozdíl od nekonečně malých diferenciálů přesné analýzy, se zdála zpočátku pro matematiky nezajímavá. První náhrada funkce dvou proměnných kombinací lineárních funkcí po trojúhelníkových prvcích (R. Courant 1943) zůstala zcela nepovšimnuta a ani velké úspěchy v inženýrské praxi nevzbudily pozornost analytiků. Až po první mezinárodní konferenci (1st Conference On Matrix Methods in Engeneering, 1965), kde se prezentovala FEM, postřehli i matematici, co inženýři už dávno pociťovali, totiž že vznikl kvalitativně nový aparát, který stojí za důkladný výzkum. Kolem roku 1968 byla FEM dosti přesně definovaná, jako zobecněná Ritz–Galerkinova variační metoda, užívající bázových funkcí s malým kompaktním nosičem, úzce spjatým se zvoleným dělením řešené oblasti na konečné prvky. V téže době bylo matematiky dokázáno, že FEM generuje soustavy lineárních rovnic podstatně lépe numericky podmíněné než tehdy ještě běžně užívaná metoda sítí.

2.2.2 Základní principy FEM, příklad vedení tepla

V tomto odstavci se pokusíme osvětlit základní princip FEM. Na začátku máme parciální diferenciální rovnici, jejíž řešení hledáme. Toto řešení obvykle hledáme v nějaké omezené oblasti (například pokud hledáme průběh napětí a deformací v mostním nosníku, je touto oblastí právě daný nosník). Diferenciální rovnici, která zpravidla obsahuje časové i prostorové derivace integrujeme přes celou oblast. Dostaneme tak integrální rovnici velmi úzce související s původní rovnicí diferenciální. Obě strany rovnice před integrováním ještě vynásobíme nějakou funkcí, může se jednat o téměř libovolnou funkci. Jediná podmínka na tuto funkci je její integrovatelnost na celé oblasti; tato funkce se označuje jako *testovací (váhová) funkce*. Integrální rovnici, kterou

dostaneme tímto postupem, nazýváme *slabá formulace*. Slabou formulací jí nazýváme proto, že řešením této rovnice není již jen jedna funkce, ale celá třída funkcí. Tato třída funkcí se pak nazývá *slabé řešení* původní diferenciální rovnice. Jednotlivé funkce tvořící slabé řešení se od tzv. silného řešení, tedy od skutečného řešení diferenciální rovnice, mohou lišit. Liší se však pouze na tzv. *množině míry nula*. Množina míry nula je například spočetná množina bodů a křivek, která je podmnožinou množiny bodů tvořících nějakou plochu – rovinu. Tento rozdíl mezi silným a slabým řešením je dán tím, že změna spočetného množství funkčních hodnot dané funkce nezmění hodnotu libovolného určitého integrálu této funkce. Násobení původní rovnice testovací funkcí pochopitelně na výsledek nemá vliv (z hlediska klasické matematické analýzy; při numerických výpočtech je naopak volba vhodné testovací funkce velmi důležitá). Ještě je třeba poznamenat, že jako testovací funkce, přestože jí může být téměř jakákoliv funkce integrovatelná na dané oblasti, se používá tzv. *funkce s malým kompaktním nosičem*. Funkce s malým kompaktním nosičem jsou takové funkce, které mají většinu funkčních hodnot na zkoumané oblasti nulových, pouze na malé jednoduše souvislé podoblasti dané oblasti mají funkční hodnoty nenulové. Typicky se volí váhové funkce například tak, aby jejich integrál přes celou zkoumanou oblast byl navíc roven jedné (normalizované váhové funkce).

Příklad diferenciální rovnice; rovnice vedení tepla; ustálený stav (derivace čehokoliv podle času je nulová):

$$-\nabla\cdot\left(\lambda\cdot\nabla\vartheta\right)=Q \quad \lambda\equiv const$$
$$-\lambda\cdot\left(\nabla\cdot\nabla\vartheta\right)=Q$$
$$-\lambda\cdot\Delta\vartheta=Q$$
$$-\Delta\vartheta=Q'$$

Přičemž:

$\lambda=\lambda(x,y,z)\equiv const$ – je koeficient přestupu tepla; konstantní v celém objemu,
$\vartheta=\vartheta(x,y,z)$ – je teplota; hledaná neznámá funkce,
$Q=Q(x,y,z)$ – je teplo.

Klasické (silné) řešení této rovnice bychom získali například metodou sítí, aplikovanou na poslední odvozenou tzv. Laplaceovu rovnici.

A její slabá formulace a slabé (variační) řešení:

$$-\nabla\cdot\left(\lambda\cdot\nabla\vartheta\right)=Q$$
$$-\nabla\cdot\left(\lambda\cdot\nabla\vartheta\right)\cdot\varphi=Q\cdot\varphi$$
$$-\int_\Omega\nabla\cdot\left(\lambda\cdot\nabla\vartheta\right)\cdot\varphi\cdot d\Omega=\int_\Omega Q\cdot\varphi\cdot d\Omega$$
$$\int_\Omega\lambda\cdot\nabla\vartheta\cdot\nabla\varphi\cdot d\Omega-\int_{\partial\Omega}\lambda\cdot\nabla\vartheta\cdot\varphi\cdot\mathbf{n}\cdot d\left(\partial\Omega\right)=\int_\Omega Q\cdot\varphi\cdot d\Omega$$

Kde:

$\varphi=\varphi(x,y,z)$ – je testovací, respektive váhová funkce,
$\Omega,\partial\Omega$ – je zkoumaná oblast a její hranice,
$\mathbf{n}=\mathbf{n}(x_{\partial\Omega},y_{\partial\Omega},z_{\partial\Omega})$ – je vnější normálový vektor k hranici zkoumané oblasti.

Poslední trochu nepřehledná úprava integrálů je rozvinutí integrálu pomocí divergenční věty. Tato úprava nás mimo jiné přivádí na myšlenku započtení okrajových podmínek do úlohy. Některé z okrajových podmínek se totiž také projeví jako integrály přes okraj zkoumané oblasti. V souvislosti s okrajovými podmínkami a hranicí oblasti je vhodné uvést, že se vyskytuje zajímavá skupina problémů, u nichž se v čase mění tvar oblasti, tedy i hranice oblasti (derivace hranice oblasti podle času není nulová). Pak se pochopitelně mění v čase i okrajové podmínky. My jsme však uvažovali stacionární úlohu (všechny veličiny včetně tvaru hranice oblasti a okrajových podmínek jsou tedy konstantní v čase). Nicméně tento text si nedává za cíl vysvětlit metodu

konečných prvků v plné šíři, ale spíš naznačit principy této metody, a proto se započtením okrajových podmínek nebudeme zabývat.

Dalším krokem pak je diskretizace zkoumané oblasti na síť konečných prvků. Je zřejmé, že touto diskretizací integrály degenerují na pouhé součty. Integrální rovnice tak přejde na soustavu lineárních rovnic. Tuto soustavu lze velmi dobře upravit například vhodnou volbou testovacích funkcí.

Předchozí rovnice, v které jsou již doplněny okrajové podmínky:

$$\int_{\Omega} \lambda \nabla \vartheta \nabla \varphi d\Omega + \int_{\partial\Omega_3} \sigma \vartheta \varphi d(\partial\Omega_3) = -\int_{\partial\Omega_2} q_N \varphi d(\partial\Omega_2) + \int_{\partial\Omega_3} \sigma \vartheta_D \varphi d(\partial\Omega_3) + \int_{\Omega} Q \varphi d\Omega$$

Kde

σ, q_N, ϑ_D – jsou parametry známé z okrajových podmínek,

$\partial\Omega_1 + \partial\Omega_2 + \partial\Omega_3 = \partial\Omega$ – je rozdělení hranice oblasti na části s Dirichletovou, Neumannovou a Newtonovou okrajovou podmínkou; pozor Dirichletova podmínka (hranice oblasti s indexem 1) se v rovnici nevyskytuje.

Diskretizací oblasti se „diskretizuje“ také hledaná funkce (rozložení teploty); provedeme následujícím způsobem:

$\vartheta \approx \sum_{i=1}^{m} \Theta_i \cdot w_i = \Theta_i \cdot w_i$ – v druhém typu zápisu je vynechán symbol sumy; tzv. Einsteinova notace.

Přičemž:

Θ_i – je m neznámých koeficientů,

w_i – jsou předem známé funkce, nejlépe s velmi malým kompaktním nosičem.

Přesnější řečeno neznámou funkci aproximujeme lineární kombinací předem známých funkcí w_i. Diskretizací funkce zde tedy rozumíme to, že místo celého spojitého průběhu neznámé funkce hledáme pouze jakési neznámé diskrétní koeficienty této lineární kombinace. Kompaktní nosič funkcí w_i má obvykle neprázdný průnik pouze s jedním, nebo několika málo vzájemně sousedními elementy oblasti. Tyto funkce navíc volíme pokud možno co nejjednodušší. Typicky jsou to funkce po částech lineární, nebo dokonce po částech konstantní. To proto, aby bylo nalezení koeficientů lineární kombinace aproximace hledané funkce co nejsnazší. Navíc by měla být hledaná funkce aproximovatelná pomocí funkcí w_i pokud možno jednoznačně (což lze u takto jednoduchých funkcí snadno zaručit) a aproximace by měla odpovídat našim požadavkům na fyzikální vlastnosti hledané funkce. Například pokud chceme v našem příkladu určit rozložení teploty v nějaké oblasti, přičemž víme, že bude posléze potřeba počítat ještě gradient rozložení teploty, nemá smysl takovou veličinu aproximovat po částech konstantní funkcí.

Rovnice má po aproximaci hledané funkce následující tvar. Zde je pro přehlednost použit druhý typ zápisu (tzv. Einsteinova notace, neboli konvence sčítacích indexů):

$$\Theta_i \cdot \int_{\Omega} \lambda \nabla w_i \nabla \varphi d\Omega + \Theta_i \cdot \int_{\partial\Omega_3} \sigma w_i \varphi d(\partial\Omega_3) = -\int_{\partial\Omega_2} q_N \varphi d(\partial\Omega_2) + \int_{\partial\Omega_3} \sigma \vartheta_D \varphi d(\partial\Omega_3) + \int_{\Omega} Q \varphi d\Omega$$

Protože máme m neznámých (koeficienty lineární kombinace z aproximace hledané funkce), potřebujeme právě m různých rovnic, abychom je mohli jednoznačně určit. Musíme tedy zvolit m různých testovacích funkcí. Pro každou různou testovací funkci pak dostaneme jednu rovnici. Vzhledem k tomu, že testovací funkce by měly být funkce s kompaktním nosičem a neznámá funkce (rozložení teploty) byla právě pomocí takových funkcí aproximována, je vhodné za j–tou testovací funkci zvolit právě funkci w_j.

J–tá rovnice pak tedy vypadá takto:

$$\Theta_i \cdot \int_{\Omega} \lambda \nabla w_i \nabla w_j d\Omega + \Theta_i \cdot \int_{\partial\Omega_3} \sigma w_i w_j d(\partial\Omega_3) = -\int_{\partial\Omega_2} q_N w_j d(\partial\Omega_2) + \int_{\partial\Omega_3} \sigma \vartheta_D w_j d(\partial\Omega_3) + \int_{\Omega} Q w_j d\Omega$$

Nyní však již je možné vypočítat všechny integrály, protože všechny funkce, které jsou pod integrály, známe. Tuto rovnici tedy můžeme přepsat následujícím způsobem:

$$\Theta_i \cdot d_{ij} + \Theta_i \cdot e_{ij} = b_j$$
$$\Theta_i \cdot A_{ij} = b_j$$

Celou soustavu rovnic pak můžeme jednoduše zapsat pomocí vektorů:

$$\mathbf{A} \cdot \mathbf{\Theta} = \mathbf{b}$$

Nalezení řešení v se tento moment již omezilo „pouze“ na vyřešení soustavy m lineárních rovnic o m neznámých. Toto je ovšem dalším velkým problémem, neboť ono m se obvykle pohybuje minimálně v řádech stovek tisíc a soustavy, které mají několik desítek milionů rovnic, také nejsou neobvyklé. Velkou výhodou takových soustav je ovšem to, že matice A jsou velmi málo zaplněné. Mají tedy jen velmi málo nenulových prvků a s ohledem právě na to je třeba volit vhodnou numerickou metodu řešení této soustavy. Tyto nulové prvky v matici vznikají právě díky tomu, že pod integrály se vyskytuje součin funkcí w_i . w_j, což jsou, jak víme, funkce s malým kompaktním nosičem, jsou tedy nenulové pouze na malé podoblasti. Pokud mezi sebou vynásobím dvě takové funkce, které ovšem nemají nenulový průnik nosičů, jejich součin je konstantní nula.

Na závěr zbývá jen dodat, že různých odvození tohoto řešení je několik. Výše naznačené odvození by se dalo označit například jako *odvození přes slabé řešení*. Ke stejnému výsledku lze dospět například pomocí variačních postupů (FEM se jako taková řadí mezi variační metody), pomocí tzv. *hledání stacionárního bodu kvadratického funkcionálu*. Odvození je také možné provést například pro neznámou funkci, která je vektorová. Takovou funkcí je právě například rychlost proudění v úlohách proudění podzemní vody. Takové odvození pak vede na tzv. *smíšenou formulaci*, zatímco problém zde řešený je tzv. *primární formulace*.

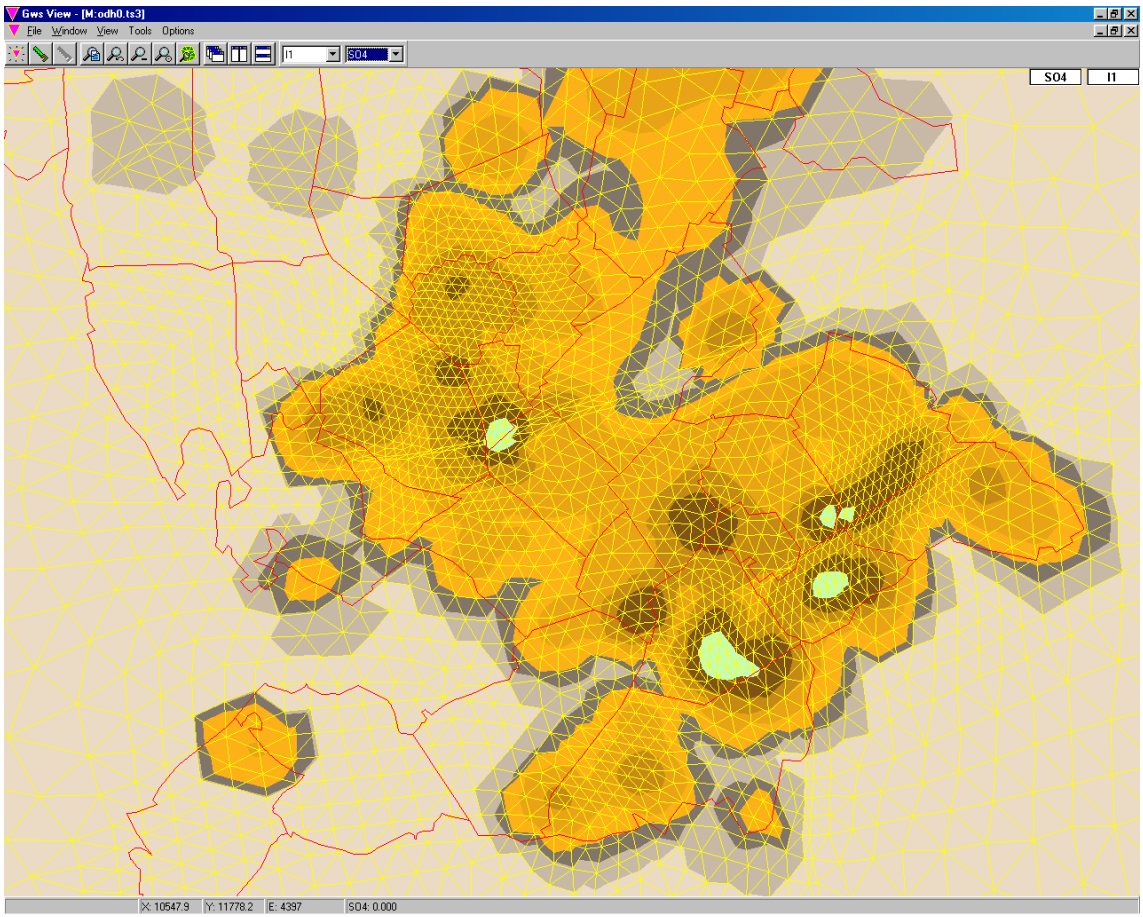
2.2.3 Softwary FEM, implementace metody v systému TURON 2003

V současné době je pochopitelně k dispozici velké množství komerčních FEM softwarů. Jako příklad můžeme uvést třeba systémy NE–XX, EDS UniGraphics – GFEM, Relax, Cosmos/M nebo například NEXIS. Špičkou mezi těmito software jsou systémy řady ANSYS. Tento systém je používán jako referenční FEM systém pro USA a země ES (celý program bez výjimky má certifikát ISO 9001). Ceny těchto komerčních balíků pochopitelně odpovídají náročnosti metody, kterou implementují. Pohybují se řádově od desítek tisíc až po miliony korun. V systému TURON 2003 je FEM implementována v modulech GEN_FLOW – pro výpočet proudění a GEN_TRAN – pro výpočet transportu látek. Výpočet proudění vychází z podobné rovnice jako je v předchozím příkladě vedení tepla, ale vychází z tzv. *Darcyho zákona*. Výpočet transportu látek, který navazuje bezprostředně na výpočet proudění, je ještě o řád složitější než problém proudění. Problém proudění tekutiny je totiž, termodynamicky vzato, vratný děj. Matice soustavy, která vznikne z takových rovnic, je tedy symetrická, což významně urychlí řešení požadované soustavy rovnic. Problém transportu však obsahuje i tzv. nevratné děje – typicky difuse kontaminantů, chemické reakce, atd.; takové děje nejsou symetrické vůči otočení směru toku času, což se projeví právě nesymetrií soustavy rovnic, kterou je potřeba řešit. Metoda konečných prvků je v systému TURON 2003 implementovaná ve své tzv. *smíšené hybridní formulaci*, označované někdy zkráceně jako *mixhybrid*. Komerční softwarové balíky jsou ovšem komplexní programy, které umožňují vše od nakreslení zkoumané oblasti v systému CAD, přes generování sítě konečných prvků až po samotný výpočet a zobrazení vypočtených výsledků. Stručně řečeno tedy zahrnují preprocesing dat, vlastní výpočet i postprocesing vypočtených dat. Moduly obsažené v systému TURON 2003 realizují pouze vlastní výpočet. Síť je konstantní, vytvořená pomocí externího programu GwsSit. Také zobrazování vypočtených výsledků je realizováno pomocí externího programu GwsView. Ukázka sítě, na které probíhá výpočet „turonu“, zobrazené v programu GwsView, je na obrázku na následující straně (obr. 2.3). Jedná se o pohled na horizontální strukturu sítě, výřez ve středu zkoumané oblasti.

*2.3 Topologie používaných FEM sítí

Význam existence sítě konečných prvků je pro metodu konečných prvků zcela zásadní. Na realizaci výpočtu na síti konečných prvků má zásadní význam i struktura a topologie sítě. Nevhodnou topologií sítě nebo nevhodnou volbou tvarů a rozměrů elementů lze významně ovlivnit například podmíněnost řešené soustavy rovnic a tím i přesnost numerického výpočtu. Současně by měla struktura sítě respektovat tvar oblasti, na které se provádí řešení. Tím se myslí například zjemnění sítě v místech, kde se předpokládá složitější tvar řešení, nebo například v případě geologického modelu, jakým je právě turonský model, to, že síť je rozdělená na jednotlivé vrstvy respektující geologické vrstvy a že horizontální hranice sítě jsou tvořeny významnými geologickými útvary. Taková síť i usnadňuje orientaci v modelu kupříkladu při zobrazování ve vizualizačních softwarech nebo při zadávání materiálových koeficientů do sítě (propustnosti v případě turonského modelu).

Obr 2.3 – zobrazení odhadů – horizontální rozložení kontaminace síranovými ionty. Je zde vidět „čočkovité“ rozložení kontaminace typické pro turonský kolektor. Zobrazení je provedeno v programu GwsView, na jednovrstvé síti komíny_1v



*2.3.1 Přesný popis tvaru sítě v systému TURON 2003

Pro výpočty bylo použita modelová síť s 5628 plošnými konečnými prvky (multielementy) a 56764 prostorovými konečnými prvky (elementy). Východní okraj modelované oblasti sleduje linii bazaltoidních žil Jeleních vrchů, severní okraj kopíruje Strážský zlom, západní je veden podél toku Ploučnice a jižní podél Ploužnice. Plocha modelované oblasti je 35 km^2. Model má celkem 9 vrstev. Spodní 3 vrstvy (vrstvy označované „I“) představují slinito-prachovité pískovce, horních 6 vrstev (vrstvy označované „J“) reprezentuje kvádrové pískovce svrchního turonu. Mocnost kvádrových pískovců má velkou hydraulickou vodivost (koeficient filtrace v intervalu 5 – 10 m^3/den) a akumulační schopnost. Z hydrogeologického hlediska je to nejdůležitější část kolektoru. Vzhledem ke sklonu báze svrchního turonu k jihu se mocnost tohoto souvrství pohybuje od jednotek metrů v blízkosti Strážského zlomu do 100 m při jižním okraji modelu. V souladu s tím se zvyšuje i počet modelových vrstev směrem k jihu. Na severu jsou vyšší vrstvy „odříznuty“ terénem. Výrazně nižší propustnost (o řád) je v okolí Ralska, čedičové žíly mají propustnost ještě o řád nižší.

Okrajové podmínky na části hranice území byly stanoveny podle měření hladin ve vrtech, na všech ostatních okrajích, a místy i na horní ploše modelu, jsou dány převážně nadmořskou výškou vodních toků. Tyto okrajové podmínky jsou stálé po celou dobu výpočtu.

***2.3.2 Elementy použité při realizaci sítě**

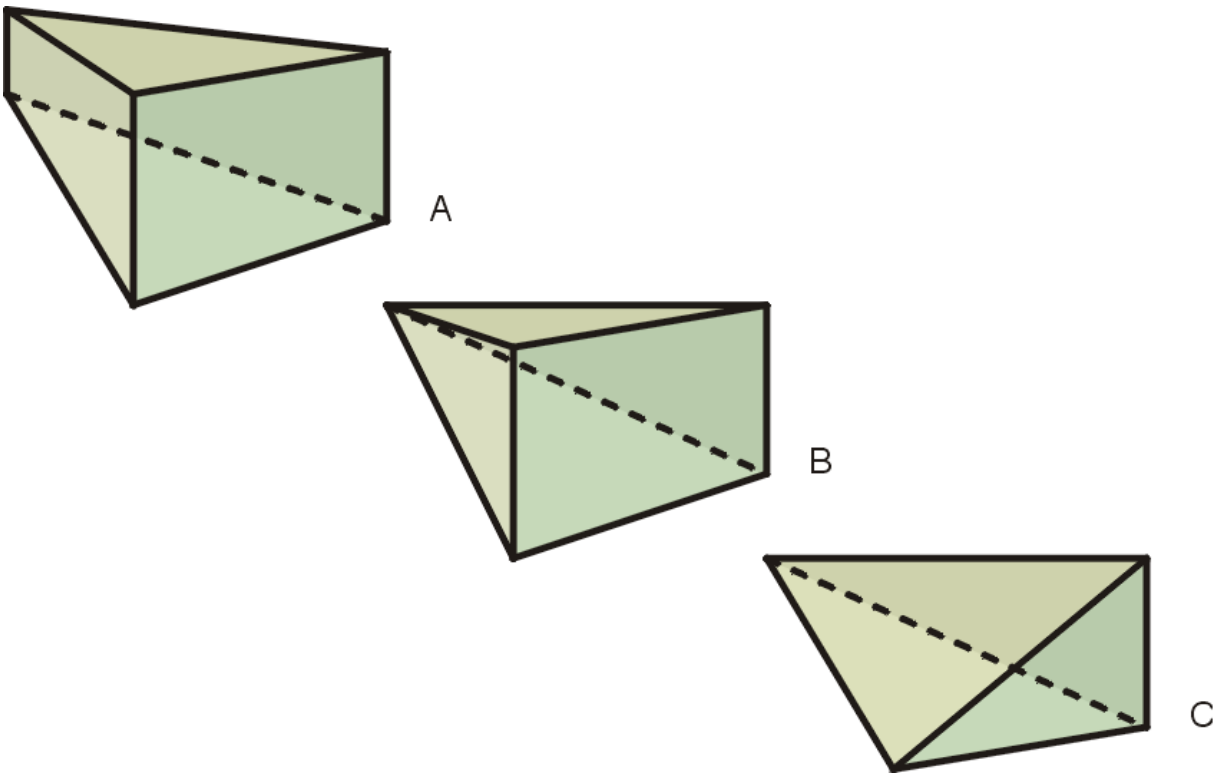
V síti použité v systému TURON 2003 se používají následující typy konečných elementů:

Trojboký hranol – s trojčetnou osou orientovanou vertikálně. To znamená, že při pohledu na horizontální rovinu sítě – při pohledu „seshora“ – se síť jeví jako trojúhelníková. Horní i spodní podstava navíc nemusí být navzájem rovnoběžné, a pochopitelně nejsou rovnoběžné ani s vodorovnou rovinou. Vertikální roviny (stěny hranolu) jsou však skutečně svislé, tzn. že horizontální souřadnice vrcholů horní a dolní podstavy se neliší.

Čtyřboký jehlan – s čtyřčetnou osou orientovanou víceméně téměř horizontálně. Takový element typicky vzniká degenerací z trojbokého hranolu tak, že se dva nad sebou ležící vrcholy obou podstav hranolu spojí v jeden vrchol (hlavní vrchol jehlanu). Při pohledu „seshora“ se i tyto elementy v síti jeví také jako trojúhelníky.

Simplex (čtyřstěn) – vzniká obdobně jako čtyřboký jehlan. Tentokrát však v trojbokém hranolu degenerují dvě dvojice nad sebou ležících vrcholů obou podstav. Takový element má na rozdíl od předchozích dvou jen čtyři stěny (předchozí měly pět stěn). Při pohledu „seshora“ se opět jeví jako trojúhelník.

Obr 2.4 – ukázky typů používaných elementů; A – trojboký hranol; B – čtyřboký jehlan; C – simplex (čtyřstěn)



Klasická tzv. *vícevrstvá síť* se, jak už název sám napovídá, skládá z několika horizontálních vrstev, jak již bylo popsáno v předchozím odstavci. Tyto vrstvy jsou v podstatě tvořeny vedle sebe ležícími trojbokými hranoly. Díky tomu, že stěny hranolů jsou svislé, vznikají v síti množiny – řady – uzlů ležících nad sebou ve svislici. Takovou řadu uzlů se stejnými horizontálními souřadnicemi (liší se pouze různými vertikálními souřadnicemi), nazýváme *multiuzel*. Současně s tím v síti vznikají i skupiny elementů ležících v zákrytu pod sebou (nad sebou) v jakýchsi komínech či sloupcích. Tuto strukturu – makroprvek – nazýváme *multielement*.

Jednotlivé vrstvy sítě přirozeně kopírují terén (horní vrstva), respektive zvrásnění jednotlivých geologických vrstev. Druhé dva typy elementů (jehlan a simplex) se uplatní v případě, že nějaká vrstva uprostřed sítě „končí“. Tomuto mechanismu ukončení vrstvy pomocí degenerovaných elementů říkáme *vykliňování vrstvy*.

Kromě klasické vícevrstvé sítě se v sytému TURON 2003 navíc používá i tzv. *jednovrstvá síť*. Tato síť je v pohledu „seshora“ nerozeznatelná od použité sítě vícevrstvé, má však jen jednu horizontální vrstvu. Taková síť slouží v tomto systému například k výpočtu odhadů. Odhady se počítají tak, že jakoby nakumulujeme kontaminaci, která je obecně rozložená ve všech vrstvách, do jedné vrstvy, ignorujeme tedy její vertikální rozložení. Výhodou tohoto postupu je to, že například při zobrazení odhadů okamžitě víme, kde je kontaminantů více a kde méně bez toho, aniž bychom museli složitě prohlížet jednotlivé vrstvy. Konkrétní implementaci sítě s tomto systému – výpis a popis souborů sítě – je možné si prohlédnout v přílohách.

Literatura použitá v kapitole 2

[1] Oborová encyklopedie – Aplikovaná matematika, RNDr. Jiří Nečas, SNTL, Praha, ČSSR 1977, (konkrétní příklad lineárního programování)

[2] Přehled užité matematiky, Prof. Karel Rektorys, Prometheus, Praha, Česká republika 1995, ISBN 80-85849-72-0 (ISBN 80-85849-62-3 ... druhý díl)

[3] FEM – Principy a praxe metody konečných prvků, V. Kolář, I. Němec, V. Kanický, Computer press, Praha 1997, ISBN 80-7226-021-9

[4] Přednášky k předmětu MKP, Doc. Jiří Maryška, TU Liberec, 2001-2002

[5] Model řízení sanace turonského kolektoru, DIAMO s. p., Česká republika, (Přesný popis tvaru sítě v systému TURON 2003)

3 Programová struktura optimalizačního systému

Systém TURON 2003 se skládá z několika základních součástí – programových modulů. Každý z těchto modulů je implementován v samostatném podprogramu systému. Seznam všech programů a podprogramů systému TURON 2003 naleznete také v přílohách. Základní schéma celého systému je vidět na obrázku 3.1. Hlavním programem, který řídí spouštění ostatních podprogramů a který řídí celý optimalizační proces, je program *TURON*. Tento program mimo jiné obsahuje systém obrazovek, který právě slouží jako rozhraní s uživatelem. Pomocí těchto obrazovek je možné zadávat různé parametry a omezení programu. Lze zde například povolovat / zakazovat použití jednotlivých povrchových technologií, můžeme stanovit koncentrace a objemy, které požadujeme načerpat. Je zde možné čerpání lokalizovat do oblastí (vyluhovacích polí) podle požadavků uživatele. Podrobný popis všech obrazovek včetně nastavení, která zde lze provádět, je v následující kapitole.

Celý složitý optimalizační problém sanace je v systému TURON 2003 linearizován. Zadávání jednotlivých parametrů na obrazovkách systému tak odpovídá kladení omezujících podmínek na nalezení optimálního řešení. Optimálním řešením nazýváme řešení lineární optimalizační úlohy ve smyslu lineárního programování. Hledáním optimálního řešení se tedy vlastně snažíme extremalizovat (maximalizovat nebo minimalizovat) nějakou účelovou funkci. Zvolit tuto účelovou funkci lze také na obrazovkách systému. Na výběr je zde hned několik různých typů úloh. Je možné minimalizovat respektive maximalizovat objem vyčerpaných kontaminantů (na výběr jsou i typy kontaminantů) a nebo je možné minimalizovat počet nových vrtů. Vlastní vyřešení této lineární úlohy má na starosti programový modul *XA*.

Na rozdíl od všech ostatních podprogramů systému se v případě programu *XA* jedná o komerční produkt, o komerční řešič úloh lineárního programování a celočíselného programování. Právě vzhledem k tomu, že se jedná o software dodávaný třetí osobou (SUN), zmíníme se pouze stručně o práci s tímto programem, zejména o jeho nastavování.

3.1 Nastavení programu XA – vstupy a výstupy – příklad

Pro ukázku nastavení programu XA použijeme příklad z kapitoly 2.1.3. Zadání příkladu tedy nebudeme opakovat. Největším problémem při zadávání vstupů tohoto programu je fakt, že program vyžaduje jejich velmi přesné sestavení. Program načítá vstupy ze souboru, který musí být velmi přesně sestaven. Umístění souboru se vstupy a souboru pro výstup předáváme programu XA pomocí parametru – souboru s nastavením.

Parametry programu XA – soubor s nastavením – soubor *MPT.CLP*:

```
C:\TEMP\PRIKLAD.MPS LISTINPUT  NO
                    OUTPUT  C:\TEMP\PRIKLAD.XA
                    PAGESIZE  2000
                    LINESIZE  79  TMARGINS   0
                    BMARGINS   0  FIELDSIZE  11
                    DECIMALS   3  EUROPEAN  NO
                    LMARGINS   0  COPIES     1
                    WAIT      NO  MUTE  NO
                    LISTINPUT NO  WARNING NO
                    SOLUTION  NO  CONSTRAINTS NO
                    COSTANALYSIS NO MARGINANALYSIS NO
                    MATLIST  NO   DEFAULTS NO
                    MAXIMIZE YES
                    OBJECTIVE ZISK
```

Sestavení tohoto souboru s parametry je víceméně stále stejné, v souboru je pouze potřeba aktualizovat umístění vstupů a výstupů. Ostatní parametry většinou není potřeba měnit. Ze pohledu uživatele programu je nejdůležitější sestavit právě soubor se vstupy, který obsahuje vlastní zadání lineárního problému. Následuje tedy soubor se vstupy, se zadáním, které odpovídá již dříve diskutovanému příkladu z kapitoly 2.

Soubor se vstupy programu XA – soubor *PRIKLAD.MPS* s komentáři:

```
NAME          PRIKLAD
ROWS
 N  ZISK                          // celkový zisk, účelová funkce, kterou maximalizuji
 L  SLEV                          // omezení dané kapacitou slévárny
 L  OBRB                          // omezení dané kapacitou obrobny
 L  ODBA                          // omezení dané odbytem výrobku typu A
 L  ODBB                          // omezení dané odbytem výrobku typu B
COLUMNS
    VYRA      SLEV      20       // výrobek typu A a jeho parametry, spotřeba kapacity
    VYRA      OBRB      50       //   jednotlivých výrobních a odbytových oddělení a
    VYRA      ODBA      30       //   příspěvek výrobku na celkovém zisku
    VYRA      ZISK      60000
    VYRB      SLEV      40       // dtto pro výrobek typu B
    VYRB      OBRB      30
    VYRB      ODBB      50
    VYRB      ZISK      80000
RHS
    MYRHS     SLEV      2000     // horní hranice omezující podmínky SLEV
    MYRHS     OBRB      3000     // dtto pro OBRB
    MYRHS     ODBA      1500
    MYRHS     ODBB      2000
RANGE
    MYRANGE   SLEV      2000     // rozsah povolených hodnot omezující podmínky SLEV
    MYRANGE   OBRB      3000     // dtto pro OBRB; rozsahy jsou pochopitelně směrem
    MYRANGE   ODBA      1500     //   k dolní hranici, horní je definována v sekci RHS
    MYRANGE   ODBB      2000
BOUND
 LO MYBOUND   VYRA      0        // omezení (spodní hranice) pro proměnnou VYRA
 LO MYBOUND   VYRB      0        // dtto pro VYRB; počet výrobků by neměl být záporný
ENDATA
```

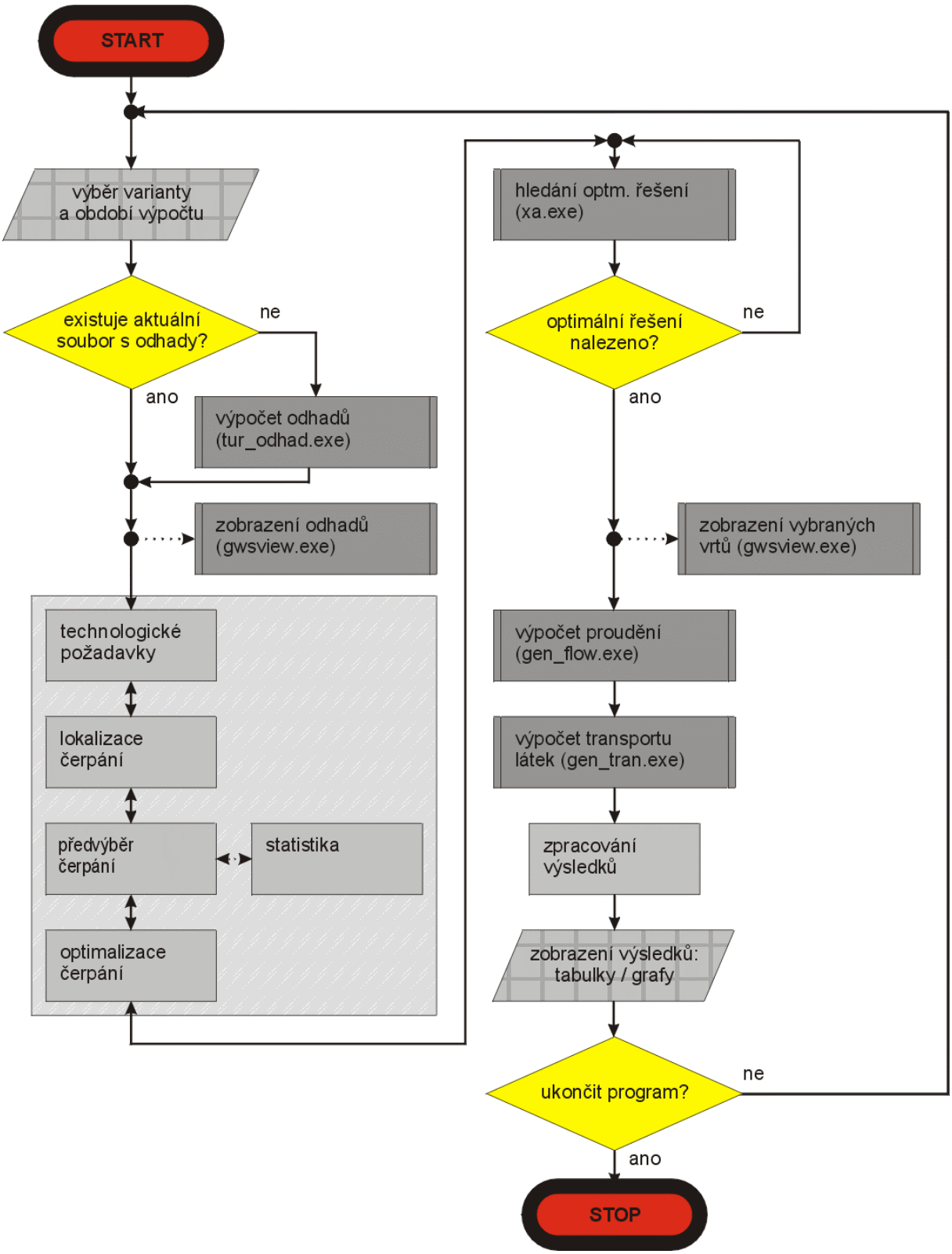
Takto sestavené vstupy ovšem nerespektují celočíselnost úlohy, program tedy jako optimální řešení nalezne necelý počet výrobků (viz také následující výpis souboru s výstupem, proměnné VYRA, VYRB). Program XA sice dokáže pracovat s celočíselnými proměnnými, ale pouze s takovými, které mají charakter logických proměnných a nabývají tedy pouze hodnot 0 a 1.

Zkrácený soubor s výstupem programu XA – soubor *PRIKLAD.XA*:

```
STATISTICS - FILE: PRIKLAD   TITLE: PRIKLAD          Sun Apr 06 12:23:55 2003
  XA VERSION 6.00   USABLE MEMORY 425K BYTES
  VARIABLES 2  MAXIMUM 6,500
     0 LOWER, 0 FIXED, 0 UPPER, 0 FREE, 0 INTEGER.
  CONSTRAINTS 4   MAXIMUM 1,500
     0 GE, 0 EQ, 4 LE, 0 NULL/FREE, 4 RANGED.
  CAPACITY USED BY CATEGORY-
    0.0% VARIABLE,  0.3% CONSTRAINT,   0.2%/ 8 NON-ZEROS, WORK 32,758
  OBJECTIVE FUNCTION IS MAXIMIZED.
  MPS FORMAT-
   OBJECTIVE: ZISK      RHS: MYRHS     RANGE: MYRANGE   BOUND: MYBOUND

O P T I M A L   S O L U T I O N ---> OBJECTIVE 4857142.857
SOLVE TIME 00:00:00  ITER 3  INVERT 2  MEMORY USED 0.1%

// následuje přehled splnění zadaných podmínek a hodnota účelové funkce ...

1  ZISK      BS   4,857,142.857 ...
2  SLEV      UL       2,000.000 ...
3  OBRB      UL       3,000.000 ...
4  ODBA      BS       1,285.714 ...
5  ODBB      BS       1,428.571 ...

// a nakonec následuje statistika hodnot proměnných ...

6  VYRA      BS          42.857 ...
7  VYRB      BS          28.571 ...
```

Z výsledků je patrné, že řešení opravdu nebylo celočíselné, nicméně se jedná pochopitelně o řešení správné (viz výsledky, které jsou k dispozici v příkladu v kapitole 2).

Obr. 3.1 – celkové schéma programu



3.2 Podprogram na výpočet odhadů

Pro správné nastavení požadavků na čerpání, právě například pro správnou lokalizaci čerpání, je potřeba dobře znát rozložení kontaminantů v podzemí. Mapa rozložení kontaminace v podzemí je však pochopitelně trojrozměrná. Kontaminace se mění jak v horizontálním směru, tak ve směru vertikálním. „Trojrozměrná“ data jsou samozřejmě potřeba pro správný výpočet proudění a transportu látek, nicméně například při hledání optimálních čerpacích vrtů je nám vertikální rozložení trochu na obtíž. Uživatel se při lokalizaci čerpání řídí údaji o tom, kde je kontaminace nejvyšší a orientace v trojrozměrné mapě by práci znesnadňovala. Mapa ve skutečnosti není trojrozměrná, ale v mapě lze postupně zobrazovat jednotlivé geologické vrstvy, v každé vrstvě pak vidíme horizontální rozložení kontaminace zvlášť.

Nejen uživatel se snáze orientuje v dvourozměrné mapě. Také data, která vstupují do lineárního modelu, se načítají z dvourozměrné mapy – jednovrstvé sítě. Otázkou zůstává, jakým způsobem by měl fungovat optimální algoritmus, který by převáděl data vícevrstvé sítě na data sítě jednovrstvé. Tento algoritmus nazýváme výpočtem odhadů a je implementován v podprogramu *TUR_ODHAD*. Tento program je vyvíjen firmou DIAMO s. p. Výpočet odhadů ve skutečnosti nedělá jen prosté zobrazení dat. Výpočet odhadů vytváří jednovrstvá data tak, že každému elementu sítě (potenciálnímu vrtu) přiřadí takové koncentrace, jaké bychom zde nejpravděpodobněji čerpali, kdybychom zde vrt vytvořili.

Odhady se vypočítávají z vícevrstvých dat rozložení kontaminantů – výsledku modelu transportu za předchozí období. Výpočet odhadů se ovšem provádí bezprostředně po zahájení výpočtu kroku varianty (viz posloupnost jednotlivých obrazovek systému – kapitola 4). Již vypočtené odhady tedy nemohou respektovat naše požadavky například na typy vrtů, které chceme používat. Na každém vyluhovacím poli totiž můžeme zvolit jiný typ vrtů (à 250, 125 a nebo 25 litrů za minutu). Jasné tak je, že pokud budou někde použity vrty s vyšším výkonem a současně jinde s nižším výkonem, vyčerpaný objem se bude alespoň v jednom případě lišit od odhadů. Současně s tímto problémem je u odhadů také problém s tím, že nejsou schopny akceptovat již existující vrty. U existujících vrtů je nutné respektovat nejen jejich čerpací výkon, ale i otevření, tedy vrstvy, z kterých jsou tyto vrty schopny čerpat. Odhady se tedy počítají tak, jako by byl vrt otevřen přibližně v místě s nejvyšší koncentrací.

Do systému zatím není zakomponován podprogram sestavený RNDr. Dresslerem, který umožňuje volit vydatnosti vrtů, nedokáže však respektovat otevření vrtů již existujících. Úplné korektní řešení by vyžadovalo vypracování nového podprogramu. Nicméně se zdá, že vývoj nového specializovaného modulu pro výpočet odhadů, který by navíc například uměl respektovat otevření a čerpací výkon existujících vrtů, je zbytečný. I se současným algoritmem pro výpočet odhadů totiž lze dosáhnout uspokojivých výsledků. Může se však stát, že po dokončení výpočtu transportu zjistíme, že skutečné čerpané objemy a koncentrace jsou nepatrně jiné, než jsme na začátku požadovali. Nalezení optimální varianty tak zabere více času, neboť je potřeba provést větší množství výpočtů.

3.3 Grafický postprocesor GwsView

Soubor s odhady je sice nezbytným zdrojem vstupních dat lineárního modelu, ale má sloužit i uživateli, který by se prostřednictvím něj měl seznámit se situací v podzemí. Soubor s odhady jako takový nám samozřejmě nic moc o rozložení kontaminace neřekne, je potřeba ho nějakým způsobem přehledně zobrazit a vykreslit, nejlépe na mapě oblasti. K tomuto účelu slouží další s podprogramů systému – grafický postprocesor *GwsView*. I tento program je vyvíjen státním podnikem DIAMO. Do rodiny programů Gws dále patří například program GwsView3D nebo GwsSit pro generování sítě. Tyto další programy však nejsou součástí systému.

V programu GwsView lze otevřít a zobrazit soubor s odhady přímo z prostředí systému TURON 2003. Soubor s odhady je pochopitelně zobrazován na jednovrstvé síti a je zobrazován na mapě vyluhovacích polí (viz také obrázek 4.2.5 v následující kapitole). V programu lze navíc volitelně zobrazit například přímo síť konečných prvků včetně číselných označení prvků a uzlů atd.

Z prostředí systému lze program GwsView ještě spouštět za účelem zobrazení množiny vybraných vrtů, pokud byla tato nalezena lineárním modelem. Vybrané vrty je navíc možné zobrazovat na podkladu odhadů i na podkladu souboru se skutečným rozložením kontaminace (výsledek transportního modelu za předchozí období). Při zobrazování vrtů lze v programu volitelně zapnout označení těchto vrtů a u zobrazení na vícevrstvých datech lze přecházet mezi jednotlivými vrstvami. Viz obrázky 4.7.8, 4.7.9 a 4.7.10, zde je také na první pohled patrný rozdíl v rozložení dat mezi jednovrstvým souborem odhadů a vícevrstvým souborem.

V programu GwsView je navíc možné zobrazovat binární soubory, které vznikají v dočasném adresáři při výpočtu transportu. Tyto soubory obsahují dynamická data o migraci kontaminantů. Soubory je však potřeba zobrazovat manuálně (nelze přímo ze systému) a navíc jsou systémem automaticky mazány.

3.4 Model proudění a transportu látek

Pokud již bylo nalezeno optimální řešení lineární úlohy, je tedy k dispozici seznam čerpacích vrtů, které byly pro dané období programem vybrány. Tyto vrty (mutlielementy) započítáváme do modelů proudění a transportu jako lokální zdroje respektive propady. Fungují zde tedy jako část okrajové podmínky. Po nalezení optimálního řešení je tedy automaticky sestavena okrajová podmínka. Počáteční podmínkou pro transportní model je výsledek předchozího období. Celý výpočet proudění i transportu látek je řízen dohromady jedním souborem.

Model proudění pracuje na principu smíšené hybridní metody konečných prvků a je realizován programem *GEN_FLOW*. Také tento program, stejně jako tak program *GEN_TRAN* pro výpočet transportu látek, je vyvíjen ve středisku matematického modelování ve státním podniku DIAMO. Zásadní částí modelu proudění je však řešič soustav lineárních rovnic, neboť metoda konečných prvků vede právě na soustavu lineárních rovnic. Řešič, respektive solver, však není součástí programu *GEN_FLOW* a je potřeba používat řešič externí. Systém je schopen používat řadu různých solverů, v současné době se však používá solver *NI1*. Tento řešič je oproti svým předchůdcům rychlejší, ovšem za tu cenu, že potřebuje více operační paměti. Všechny tyto řešiče jsou pro firmu DIAMO vyvíjeny na půdě Akademie věd.

Model transportu látek je založen na metodě konečných objemů a je implementován programem *GEN_TRAN*. Tento model bezprostředně navazuje na výsledky modelu proudění. Výsledkem tohoto modelu jsou již přímo množství, tedy objemy, koncentrace a hmotnosti vyčerpaných roztoků. Tyto údaje se pak již přímo zobrazují v tabulkách na poslední obrazovce systému (viz následující kapitola, odstavec 4.9). Dále je výsledkem transportního modelu soubor s rozložením kontaminace na konci výpočtu, tento slouží jako počáteční podmínka pro další krok výpočtu.

3.4.1 Modely standardní a modely s dvojí pórovitostí

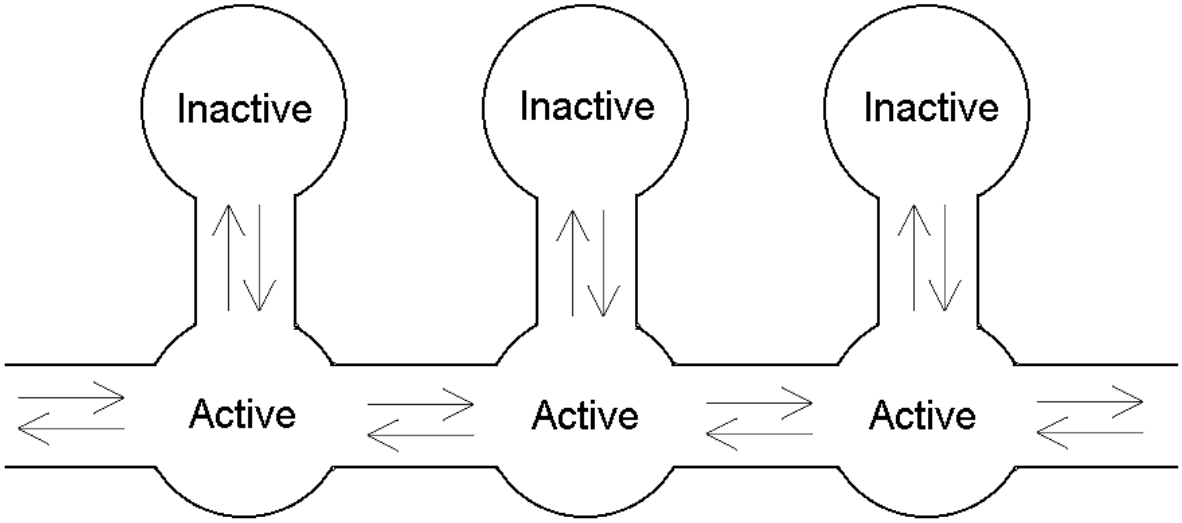
Vytváření modelů pracujících na principech konečných prvků není samozřejmě triviální záležitostí. Například právě při vytváření modelů podzemního proudění a transportu látek pro zdejší oblast bylo potřeba zavést do modelů tzv. dvojí pórovitost. Dvojí pórovitost materiálu si lze představit následujícím způsobem. V materiálu, kterým necháváme prosakovat tekutinu (zde pískovec), nejsou pochopitelně všechny póry stejné, mají různé rozměry, některé jsou lépe propustné, jiné naopak. Ve standardních modelech se však používají jen póry jednoho druhu (všechny různé póry se v modelu aproximují jen jedním typem pórů). V modelech s dvojí pórovitostí jsou obsaženy póry dvou různých typů. Jsou zde póry propustné neboli průchozí (aktivní), tedy takové, kterými mohou kontaminanty volně difundovat, a póry nepropustné, neprůchozí (neaktivní), tedy takové, které jsou slepé a „nikam nevedou“. Kontaminace se tedy z neaktivních pórů dostává stejnou cestou, jakou se dostala dovnitř. Tento jev může do značné míry ovlivnit migraci kontaminantů, neboť kontaminace se pohybuje v podzemí ve vodou saturovaném prostředí právě zejména mechanismem difuse (tedy proti směru gradientu koncentrace kontaminantů – Darcyho zákon). V neaktivních pórech téměř nedochází ke konvekci – proudění – kapaliny. Schematické znázornění matrice s dvojí pórovitostí je vidět na obrázku 3.2.

Úloha proudění řešená s ohledem na vícenásobnou pórovitost může být daleko přesnější, ovšem může také výpočet celkově zpomalit. Naopak model s nedostatečným rozlišením typů pórů může výpočet téměř znehodnotit. A právě proto, že model s jednoduchou pórovitostí dával nepřesné výsledky, byla zavedena pórovitost dvojí. Bylo by samozřejmě možné zavést i pórovitost vyšší, například trojí, ale nejeví se to zatím jako nutné. Model s dvojí pórovitostí také pochopitelně vyžaduje větší množství materiálových koeficientů, které jsou obsažené v síti konečných prvků. Navíc právě takové údaje, jako je například celková porosita materiálu, nebo poměr průchozích a neprůchozích pórů, musí být stanoveny experimentálně. Takové experimentální stanovení nejrůznějších parametrů modelu nazýváme kalibrací. Tato spočívá stručně řečeno v tom, že postupně měníme různé neznáme parametry modelu a porovnáváme je se skutečnou situací. Tuto činnost provádíme tak dlouho, dokud model neodpovídá realitě s dostatečnou přesností. Model proudění a transportu pro turonskou oblast byl kalibrován na datech z vyluhovacího pole VP-9C. Kalibrací byly například tyto materiálové koeficienty stanoveny takto:

Celková porosita:	0,25 – 0,27 (z celkového objemu matrice)
Aktivní póry:	0,07 – 0,08 (z celkového objemu matrice)

Zavedení dvojí pórovitosti je patrné kupříkladu u souborů systému. Například soubory mCAS.ts3 a mCAS_slow.ts3. Data souboru prvního z obou souborů odpovídají rozložení koncentrací kontaminantů v podzemí v aktivních pórech, naopak data souboru „slow“ odpovídají rozložení koncentrací v pórech neprůchozích.

Obr. 3.2 – struktura aktivních a neaktivních pórů



Literatura použitá v kapitole 3

[1] Application of numerical simulation system on Turonian aquifer remediation control (Dual porosity model), H. Čermáková, J. Novák, J. Mužák, TU Liberec, Česká Republika, 2000

[2] Calibration of dual porosity groundwater flow and contaminant transport model for simulation of cleaning VP-9C site. J. Mužák, M. Hork, In: Proceedings of contributed papers and posters. Algoritmy 2002, Vysoké Tatry – Podbanské, Slovensko, 2002

4 Interface – popis grafického rozhraní systému

V programu je možnost počítat různé varianty – strategie – sanace. Protože výpočet jedné varianty je poměrně složitá záležitost (ke každému kroku varianty se vztahuje velké množství souborů), je každá varianta umístěna ve zvláštním adresáři. Název varianty (například V01) je identický s názvem adresáře, v kterém je varianta (soubory varianty) umístěna. Výpočet varianty se provádí (zejména kvůli velké časové náročnosti) v mnoha krocích, které na sebe pochopitelně musí navazovat, nikoliv však bezprostředně. Z výsledků výpočtu předchozího kroku je tedy vždy možno začít počítat krok následující. Jeden krok výpočtu odpovídá časovému období délky půl roku (v modelu se půlrok bere v rámci zjednodušení jako 180 dní, jeden rok má 360 dní). Soubory odpovídající jednotlivým časovým krokům jsou pro přehlednost číslovány podle počátečního počtu dní v daném kroku. Pokud je tedy jméno varianty V01, v adresáři V01 můžeme po vypočtení prvních dvou kroků nalézt například tyto soubory: *vysledky0.xls* – souhrnné výsledky výpočtu za období 0 – 180 dní, nebo *vysledky180.xls* – souhrnné výsledky výpočtu za období 180 – 360 dní. Před samotným zahájením výpočtu je pochopitelně potřeba mít v adresáři varianty nějaké soubory, které charakterizují počáteční podmínky v čase 0. Tyto soubory obsahují počáteční rozložení kontaminace v oblasti – soubory *m0.ts3* a *m0_slow.ts3* – a seznam existujících vrtů – soubor *sexvrt0.txt*. Tyto soubory jsou do adresáře varianty automaticky nakopírovány při vytvoření varianty. Pro další kroky výpočtu jsou již tyto soubory generovány automaticky lineárním modelem (seznam vrtů) a transportním modelem (rozložení kontaminace).

Přesný souhrnný popis souborů, které se nacházejí v adresáři varianty, význam jednotlivých souborů a původ vzniku souboru, umístění adresářů s variantami atd. naleznete v přílohách.

4.1 Obrazovka 1 – výběr varianty

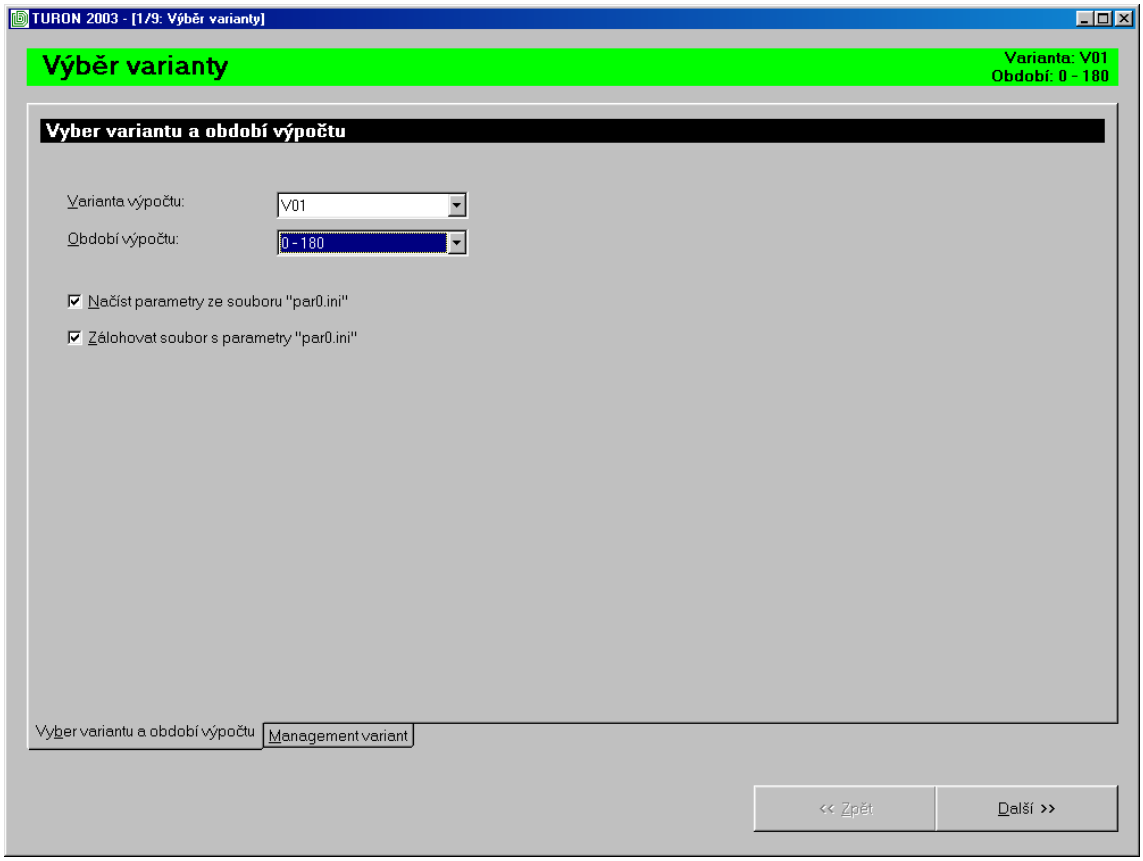
Na obrazovce výběru varianty si vybereme variantu a období které chceme počítat. Program nám sám nabídne seznam existujících variant a poté seznam období, pro která existují soubory s počátečními podmínkami. Pokud již někdy bylo námi vybrané období dané varianty počítáno, existuje *ini* soubor s parametry – s kompletním nastavením celého programu pro dané období. Tento soubor kromě parametrů může obsahovat i výsledky transportního modelu a výsledky ekonomického modelu (tato data soubor obsahuje pouze v případě, že byl daný krok dopočítán až do konce).

Program nám automaticky nabídne, zda chceme tento soubor s parametry načíst do programu a zda ho chceme zálohovat. Pokud existuje pro dané období také soubor s výsledky ve formátu *xls*, program nám ho také umožní zálohovat. Jména a strukturu *ini* souborů a mechanismus zálohování souborů nalezneme v přílohách a v kapitole týkající se úprav provedených v systému. Vzhled obrazovky pro výběr varianty viz obrázek 4.1.1.

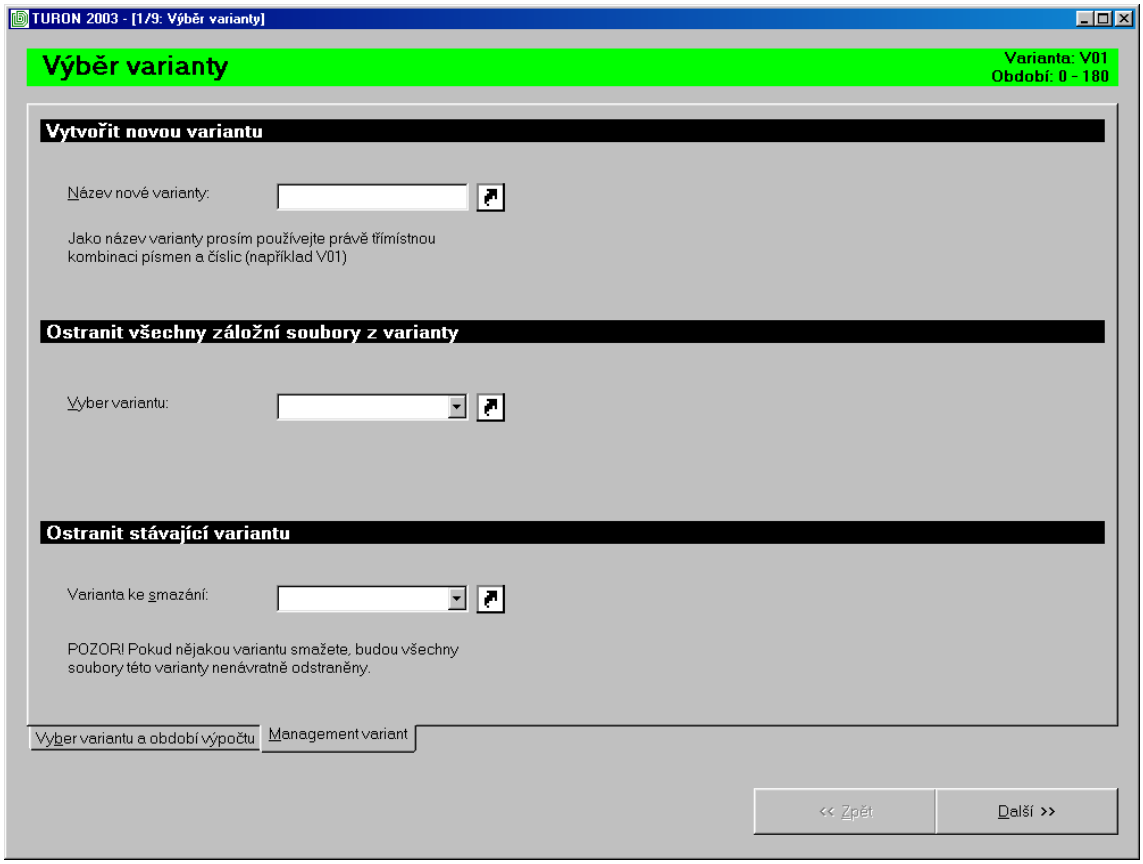
Na první obrazovce se také nachází jakýsi manažer variant, který umožňuje vytvoření nové varianty (automaticky vytvoří pří příslušný adresář a nakopíruje do něj potřebné soubory). Dále umožňuje celou variantu jednoduše smazat, případně umí z vybrané varianty smazat jen záložní soubory. Mazání záložních souborů se hodí zejména při vývoji programu, kdy dochází k častému přepočítávání variant a díky automatickému zálohování je brzy v adresáři varianty více záložních souborů než užitečných dat. Manažer variant viz obrázek 4.1.2.

Je zde také k dispozici popup menu (na pravém tlačítku myši), pomocí kterého lze celý program vypnout a lze zobrazit dialogové okno about. Součástí tohoto menu je také tzv. debug menu. Toto menu obsahuje funkce, které usnadňují vývoj programu. Například funkci pro zobrazení tzv. debug okna, ve kterém jsou vypsány všechny důležité proměnné vyskytující se v programu, nebo funkci, která nám umožňuje přeskočit z první obrazovky rovnou na obrazovku poslední. Debug obrazovka viz obrázek 4.1.4 (debug okno lze zapnout z kterékoliv obrazovky).

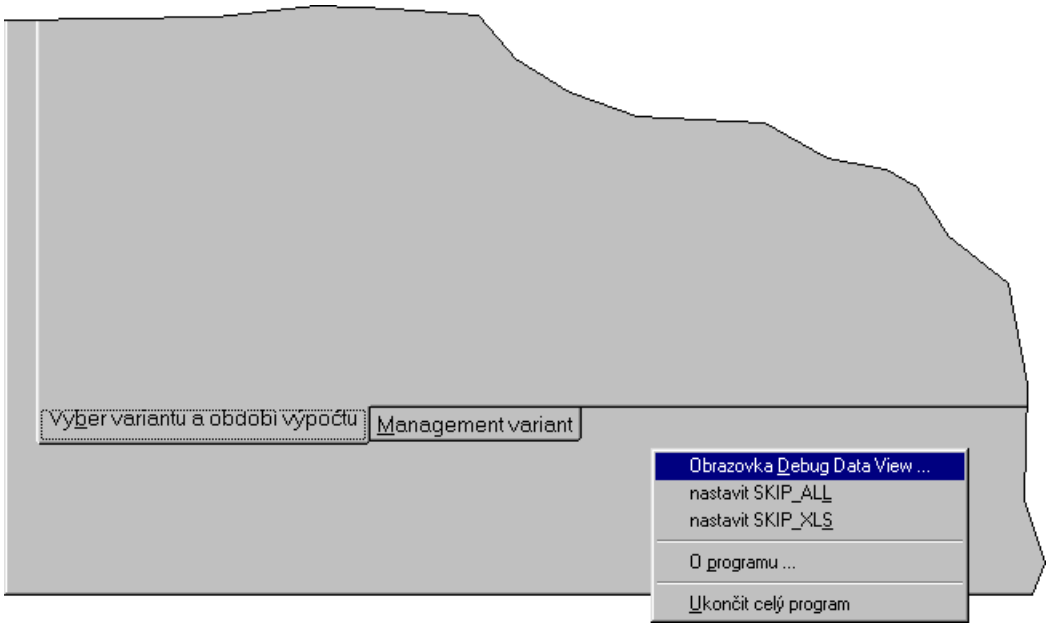
Obr. 4.1.1 – obrazovka 1 se zadanými parametry – název varianty V01, období 0 – 180 dní



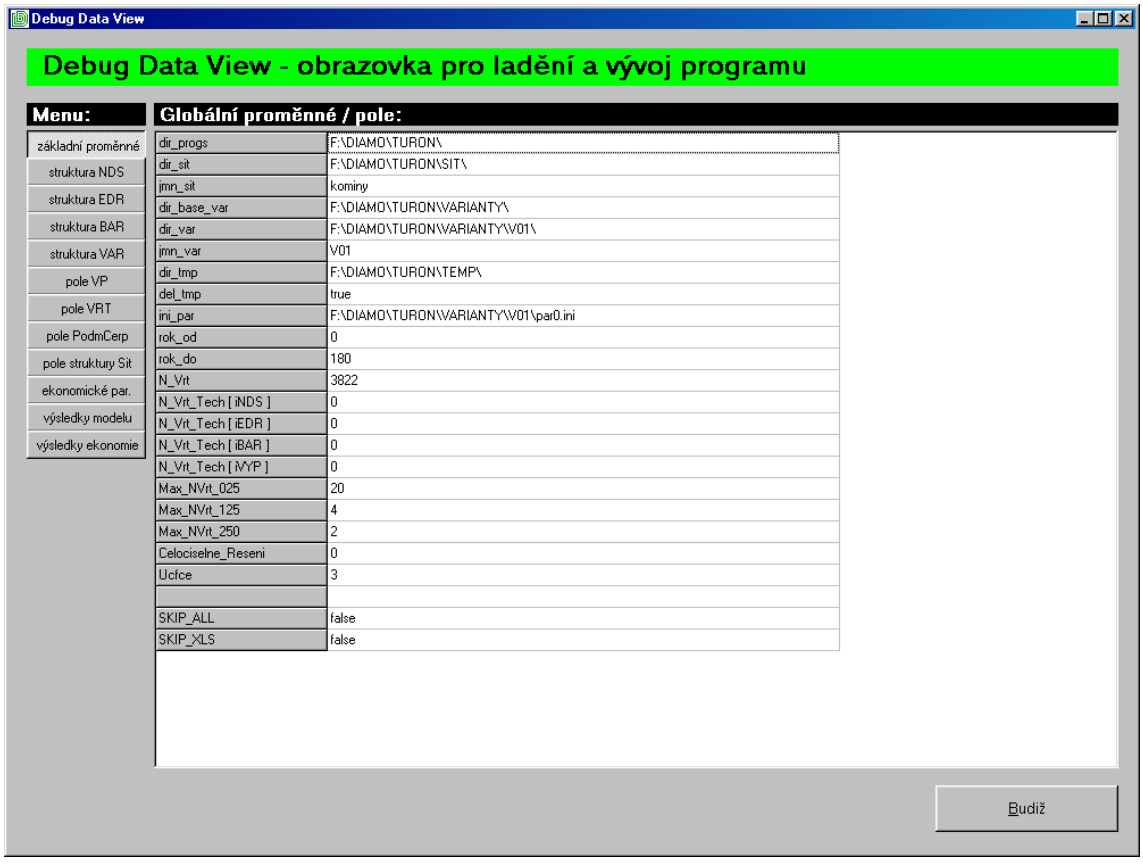
Obr. 4.1.2 – obrazovka 1; manager variant



Obr. 4.1.3 – obrazovka 1; popup menu se zapnutým debug menu



Obr. 4.1.4 – obrazovka Debug Data View



4.2 Obrazovka 2 – výpočet odhadů

Soubory obsahující počáteční podmínky (počáteční rozložení kontaminace) každého jednotlivého kroku, tedy soubory *mCAS.ts3* a *mCAS_slow.ts3*, jsou soubory definované na takzvané vícevrstvé síti. Jejich „nevýhodou“ je to, že data v nich uložená respektují vertikální rozložení kontaminace. Nevýhodou to můžeme nazvat proto, že například při zadávání lokalizace čerpání (viz kapitola 4.4) musíme mít celkem přesnou představu o rozložení kontaminace, ovšem pouze o jejím horizontálním rozložení (abychom se nesnažili čerpat někde, kde nic není). Také vybrání vhodných čerpacích vrtů (viz kapitoly 4.6 a 4.7), které je realizováno komerčním softwarem na řešení lineárních problémů, by bylo výrazně složitější, pokud bychom chtěli akceptovat vertikální rozložení kontaminace.

Algoritmus výpočtu odhadů tedy vlastně realizuje funkci, která nám říká, jaké roztoky s jakým zastoupením jednotlivých kontaminantů bychom nejpravděpodobněji čerpali z nějakého místa na povrchu, pokud bychom zde udělali čerpací vrt. Bohužel současný program na výpočet odhadů například neumí akceptovat existující vrty, což je celkem zásadní nedostatek. Pokud je totiž někde již hotový vrt, je otevřen jen pro určité vrstvy (je otevřen jen v určité hloubce). Tento fakt pochopitelně značně ovlivní kvalitu čerpaného roztoku, kontaminace totiž pochopitelně může být i v místech, kde vrt otevřen není. Výpočet odhadů tedy pouze transformuje data uložená v *ts3* souborech (*mCAS.ts3* a *mCAS_slow.ts3*), respektive pouze data uložená v prvním z obou souborů (*mCAS.ts3*).

Vypočtené odhady jsou uloženy do souboru *odhCAS.ts3*, který je již realizován na jednovrstvé síti. Vzhledem k tomu, že výpočet odhadů manipuluje s daty výše uvedených souborů, je potřeba opakovat výpočet při každé aktualizaci těchto souborů.

Při přechodu z obrazovky 1 na obrazovku 2 mohou nastat v podstatě tři různé situace:

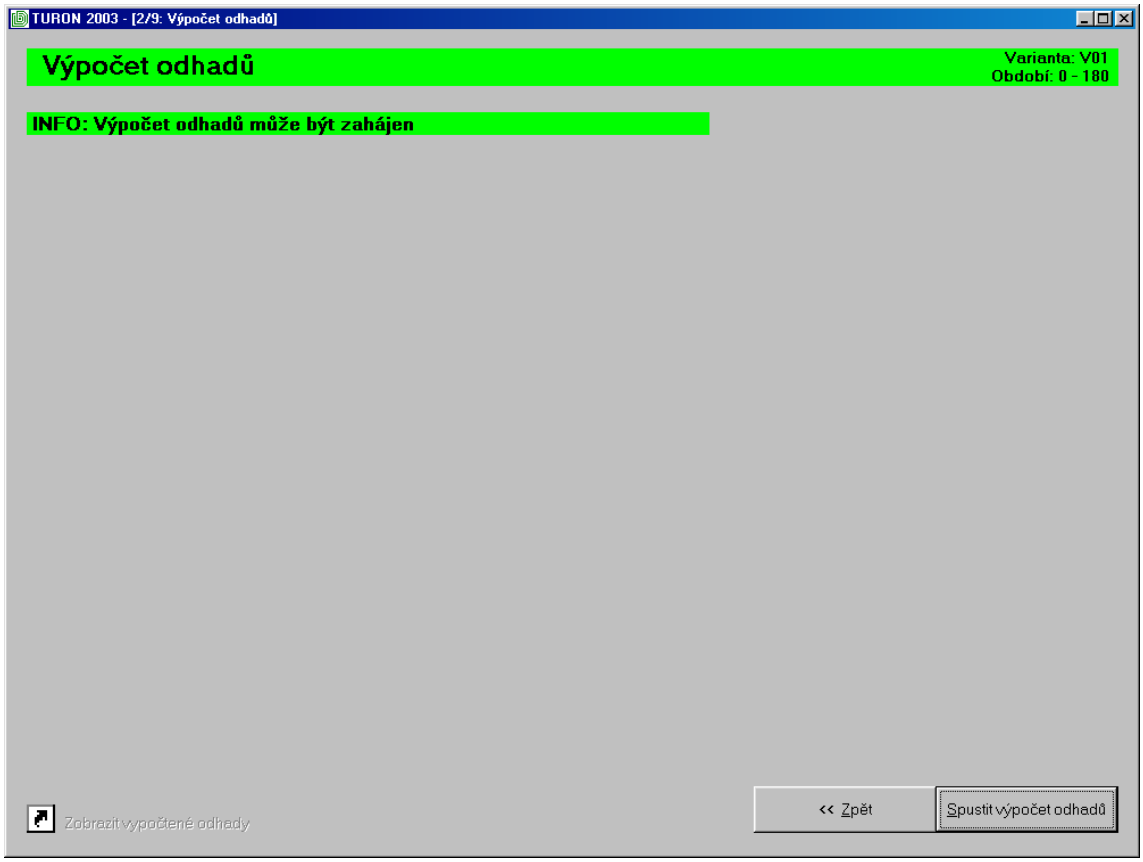
(a) Soubor s odhady zatím neexistuje a je tedy nutné zahájit výpočet odhadů. Toto je v podstatě standardní situace při prvním počítání libovolného kroku nějaké varianty. Po vypočtení odhadů je možné zobrazit odhady pomocí grafického procesoru GwsView a je možné pokračovat na další obrazovku.

(b) Soubor s odhady je již k dispozici, ale odhady nejsou aktuální. To znamená že soubor s odhady byl vytvořen dříve než soubory, z kterých se odhady vypočítávají. Tato situace tedy typicky nastává když přepočítáváme nějakou již vypočtenou variantu. V takovém případě je možné již vypočtené odhady zobrazit pomocí programu GwsView. Výpočet nových odhadů je však nutný a programem je také striktně vyžadován. Po novém výpočtu a aktualizaci souborů je opět možné odhady zobrazit a pokračovat na další obrazovku.

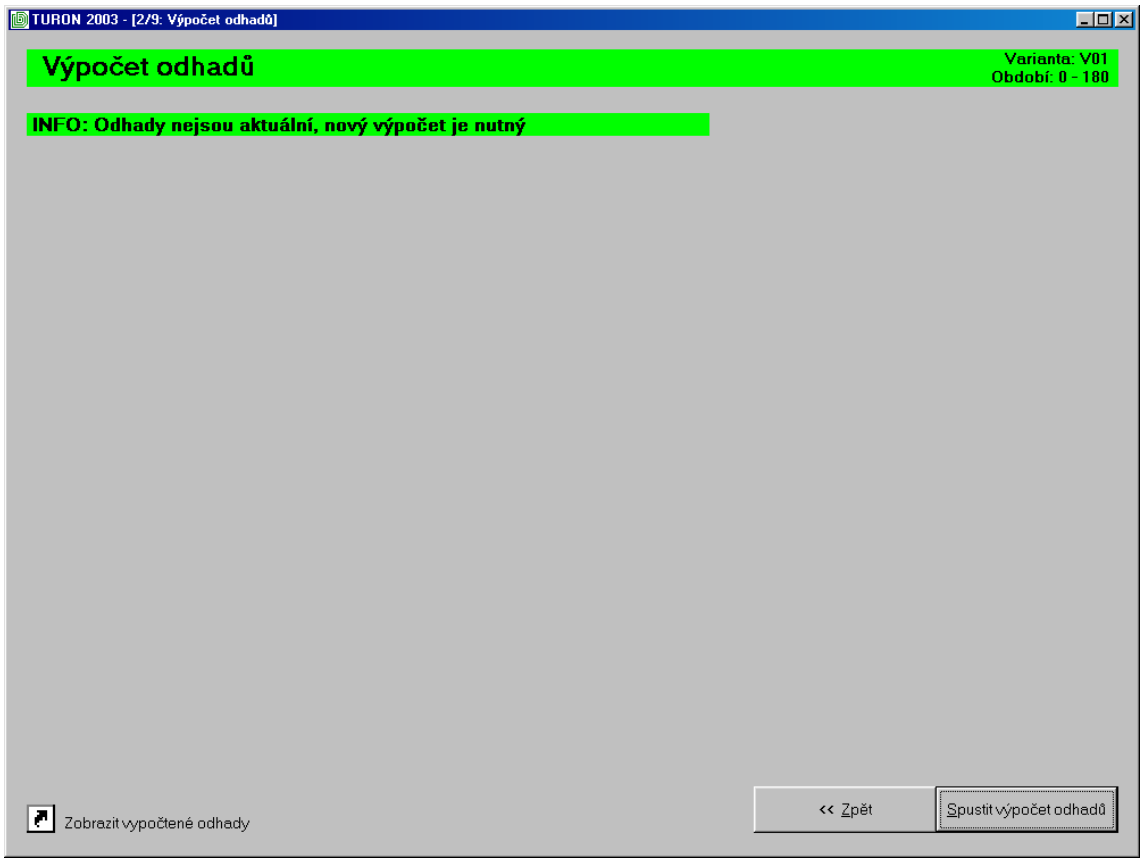
(c) Soubor s odhady existuje a je aktuální. To je situace, která nastává například v případě přepočítávání kroku za období 0 – 180 dní, neboť soubory *m0.ts3* a *m0_slow.ts3* se nemohou programově měnit. V takovém případě je opět možno odhady zobrazit. Nový výpočet odhadů pochopitelně není nutný a program ho ani neumožňuje. Může tedy přímo pokračovat na další obrazovku.

Výpočet odhadů je realizován externím programem tur_odhad.exe. Zobrazování dat taktéž probíhá prostřednictvím externího programu – grafického postprocesoru – GwsView. Přesný popis všech souborů a programů, které jsou součástí systému TURON 2003, a jejich úlohu v tomto systému, naleznete v přílohách.

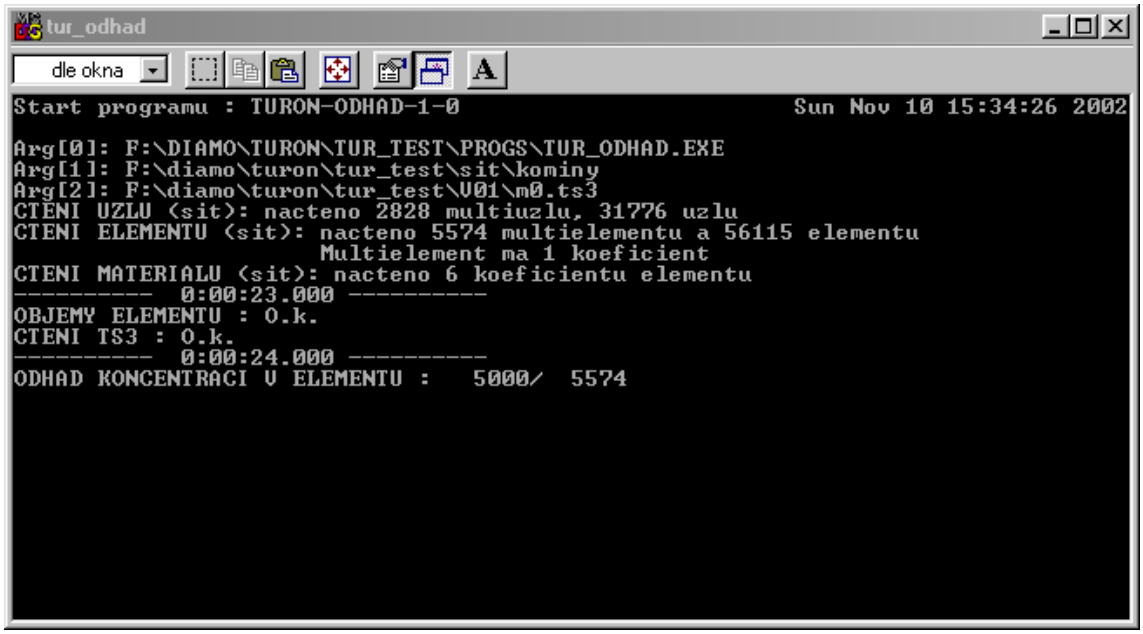
Obr. 4.2.1 – obrazovka 2; výpočet odhadů může být zahájen



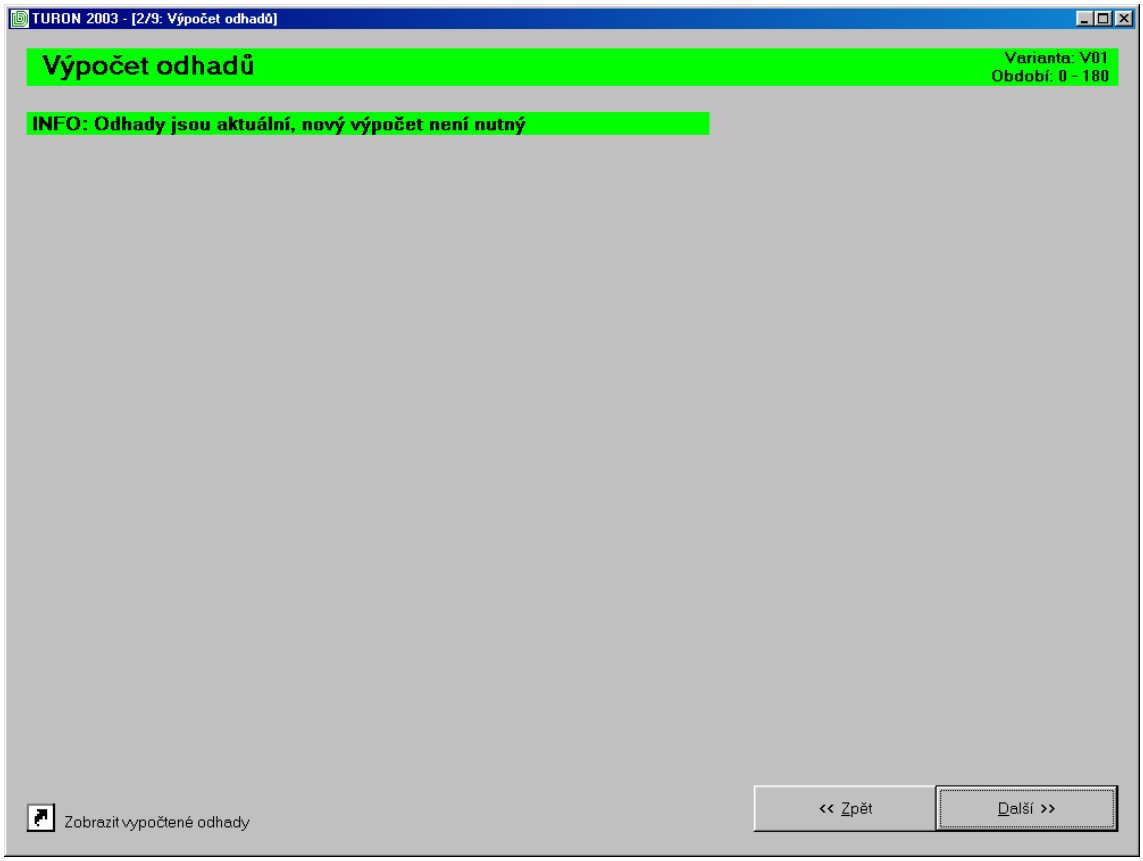
Obr. 4.2.2 – obrazovka 2; soubor s odhady existuje, ale není aktuální



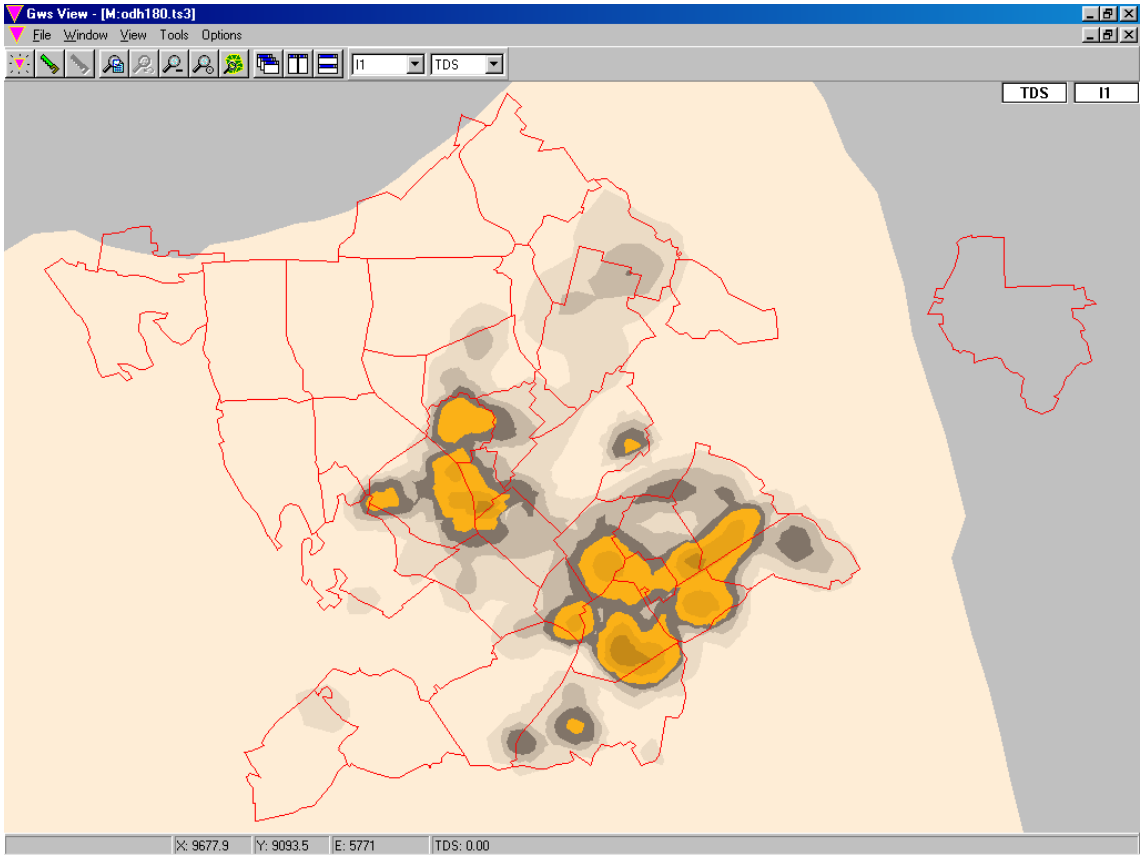
Obr. 4.2.3 – obrazovka2; výpočet odhadů pomocí programu TUR_ODHAD



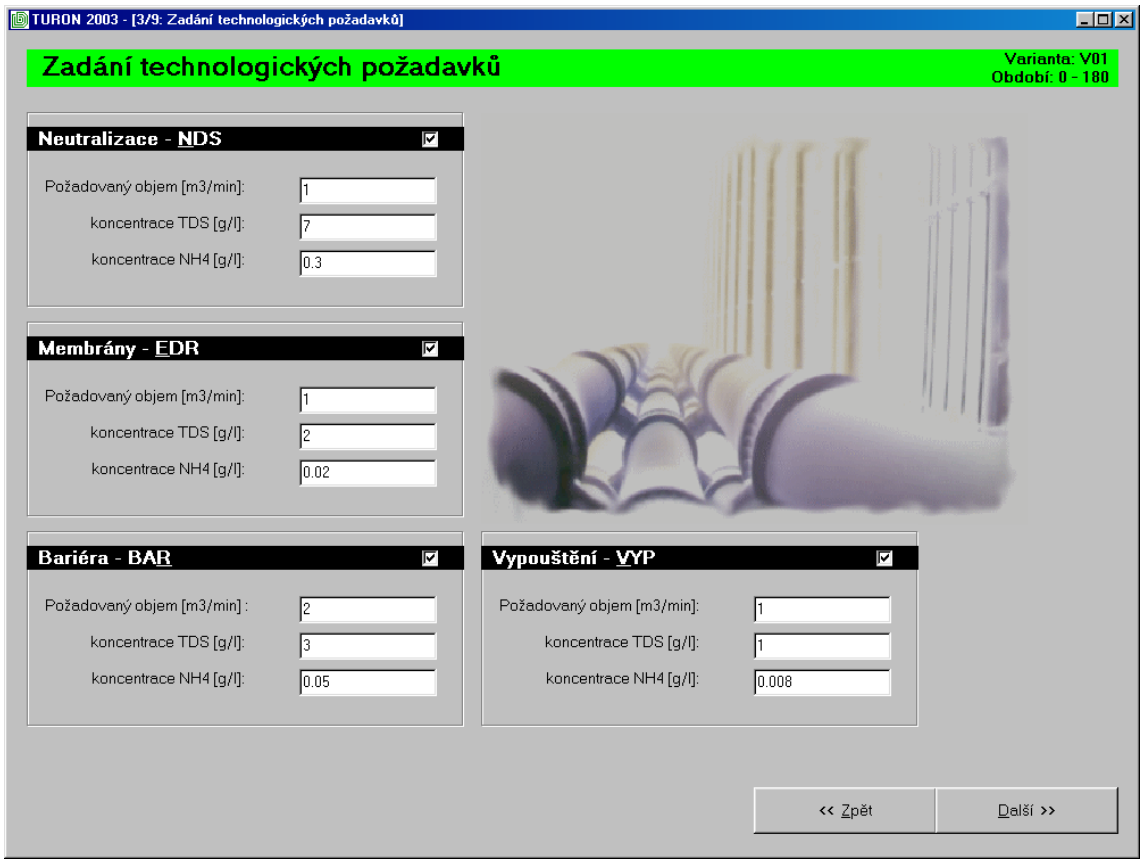
Obr. 4.2.4 – obrazovka 2; soubor s odhady existuje a (již) je aktuální



Obr. 4.2.5 – zobrazení odhadů v programu GwsView



Obr. 4.3.1 – obrazovka 3; zadání technologických požadavků



4.3 Obrazovka 3 – zadání technologických požadavků

Na této obrazovce je možné zapnout jednotlivé technologie zpracování kontaminovaných roztoků. Zde se také nastavují základní parametry (požadavky) těchto roztoků. V systému se „počítá“ se čtyřmi technologiemi, které připadají v úvahu při sanaci turonské zvodně. Jedná se o následující čtyři technologie:

Neutralizace – v systému označovaná zkratkou *NDS* a na obrazovkách zobrazovaná modrou barvou. Jedná se o klasickou neutralizaci. Technologie odstraňuje z roztoků zejména ionty SO_4^{2-}, *Al*, *Fe*. Dále se zde, v samostatné části neutralizační stanice, z roztoků odstraňují ionty NH_4^+ chlorací. Technologie je vhodná pro nejsilnější roztoky, které se v turonském kolektoru vyskytují. Protože vlivem čerpání vysoce koncentrovaných roztoků koncentrace kontaminantů v podzemí poměrně rychle klesá, je potřeba tuto technologii časem – v závěru procesu sanace – vyřadit ze scénáře sanace. Neutralizační stanice totiž nepracuje se slabými roztoky dostatečně efektivně (z ekonomického hlediska).

Membrány – respektive membránová technologie, v systému označovaná zkratkou *EDR* a zobrazována červeně. Roztoky, které jsou zpracovávány pomocí membránové technologie, jsou vlastně rozděleny na dva typy roztoků – koncentrát a diluát. Diluát je tvořen v podstatě vyčištěnou vodou, respektive roztoky s tak nízkou solností, že mohou být vypouštěny. Koncentrát, kterého je samozřejmě výrazně méně, pak jde na další zpracování odparkou. Membránová technologie typicky pracuje se středně silnými roztoky. Její provoz je však velmi finančně náročný a ukazuje se, že tyto roztoky lze efektivněji zpracovávat na neutralizační stanici, proto se uvažuje o jejím vypuštění z plánu sanace turonské zvodně.

Bariéra – hydraulická bariéra, v systému označovaná zkratkou *BAR*, zobrazovaná zeleně. Nejedná se o technologii v pravém slova smyslu, nicméně se jedná o postup, jak naložit se středně slabými a slabými roztoky. V podstatě se jedná o vtláčení těchto roztoků zpět do země. Bariéra se používá k zabránění migrace kontaminantů do bývalého hlubinného dolu a k udržení umělého rozdílu hladin podzemní vody právě v okolí hlubinného dolu. Zde je potřeba hydrobariéru udržovat minimálně do doby, než bude důl zaplaven vodou. Je tedy potřeba zde vtláčet vodu do podzemí a turonské roztoky s nepříliš vysokými koncentracemi se jeví jako velmi vhodné. Jedná se tedy vlastně o obrazné „zabití dvou much jednou ranou“, což je pochopitelně oboustranně výhodné.

Vypouštění – vypouštění, označováno *VYP*, zobrazováno žlutě. Také se nejedná o technologii v pravém slova smyslu, ale pouze o způsob zpracovávání roztoků, jako v předchozím případě. V podstatě se jedná o prosté vypouštění roztoků do řeky. Roztoky se však vypouštějí až po předčištění ve směšovací nádrži. Zde se smíchávají s vyčištěnou vodou z ostatních technologií a dochází zde ještě k částečnému čistění sedimentací. Do řeky je ovšem možno vypouštět pouze ty nejslabší roztoky, které splňují přísné hygienické normy.

Pro každou technologii je možné na této obrazovce zapnout následující parametry. Požadovaný objem, tedy celkový požadovaný nátok na danou technologii v krychlových metrech za minutu. Požadované koncentrace TDS a NH4 v gramech na litr roztoku. Vzhled celé obrazovky 3 viz obrázek 4.3.1.

4.3.1 Poznámka – složky kontaminace sledované v systému TURON 2003

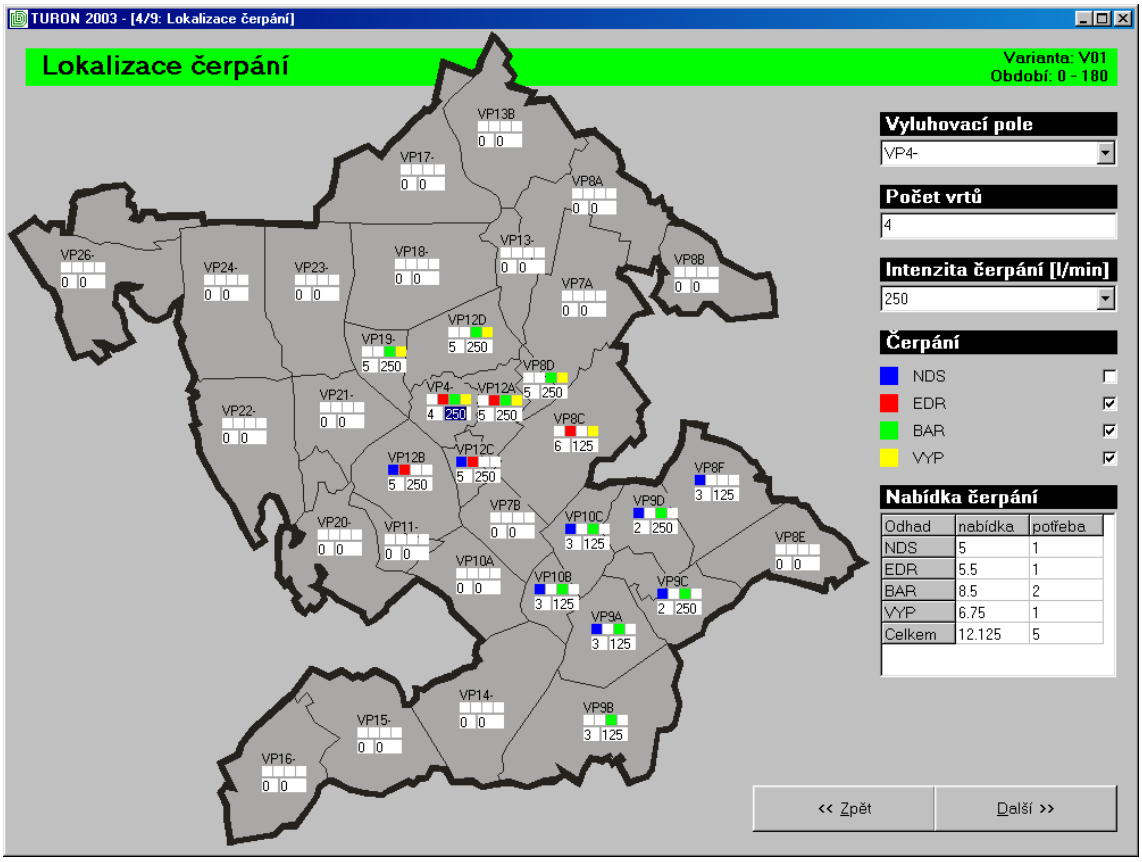
V turonském systému se celkově sledují tři složky kontaminace, jak již bylo zmíněno v úvodní kapitole (kapitola 1.2). Jedná se o následující tři složky:

Celkové znečistění – koncentrace celkového znečistění v oblasti; v programu označováno zkratkou *TDS*. *(TDS = total dissolved solids = celkové rozpuštěné látky)*

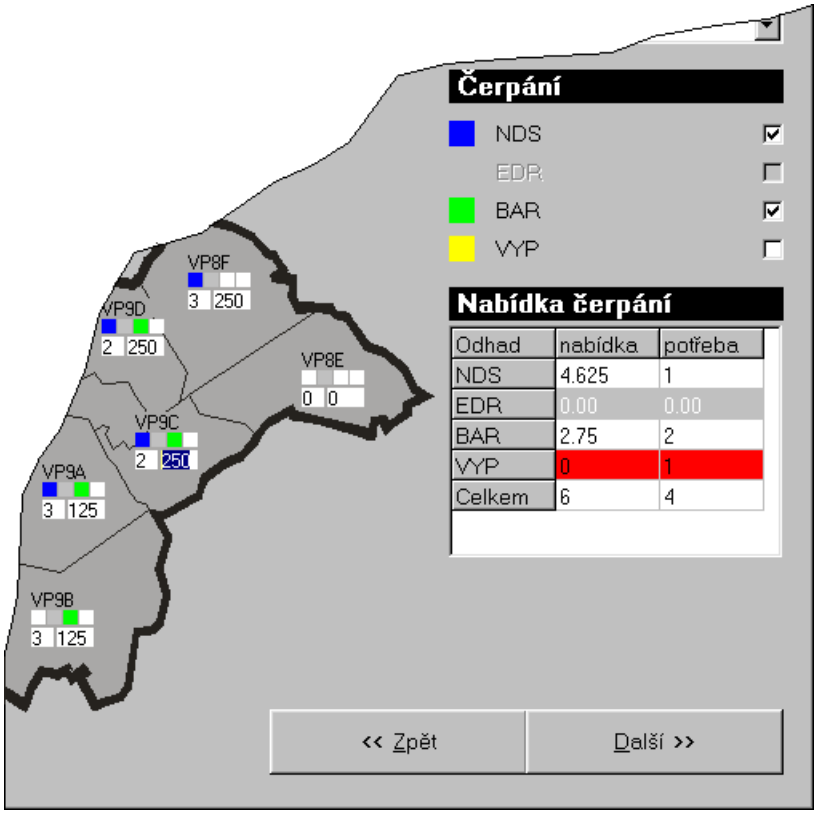
SO_4^{2-} – koncentrace síranových aniontů; v programu označováno zkratkou *SO4*.

NH_4^+ – koncentrace amonných kationtů; v programu označováno zkratkou *NH4*.

Obr. 4.4.1 – obrazovka 4; lokalizace čerpání. Provázání obou postupů zadávání parametrů je dobře patrný, aktuálně je vybráno vyluhovací pole VP4-



Obr. 4.4.2 – zvýrazněné řádky v tabulce; VYP – převyšuje potřeba nabídku; EDR – vypnutá technologie



4.4 Obrazovka 4 – lokalizace čerpání

Jak již nadpis kapitoly napovídá, obrazovka 4 slouží k lokalizaci čerpání. Protože čerpané koncentrace načítáme ze souboru odhadů, pracujeme stále s jednovrstvou sítí. Čerpání tedy stačí lokalizovat horizontálně – tedy „na mapě“. Nejmenší oblasti (horizontálně rozlišitelné pro uživatele) jsou jednotlivá vyluhovací pole. V oblasti je celkem 35 vyluhovacích polí. Ke každému vyluhovacímu poli je přiřazeno několik elementů (multielementů) sítě, toto přiřazení je jednoznačně definované v souboru *LOKAL.ELM* (viz přílohy). Pro každé vyluhovací pole je možno nastavit maximální počet vrtů, které lze v daném poli vybrat (typicky do deseti vrtů u výkonnějších vrtů; u vrtů liftových můžeme vybrat až 30 vrtů) a vydatnost těchto vrtů (liftové vrty – 25, čerpadlové vrty – 125, nebo 250 litrů za minutu; při inicializaci je hodnota nastavena na 0). Je také možné (a nutné) přiřadit tyto vrty v daném vyluhovacím poli některé ze zapnutých technologií. Z jednoho vyluhovacího pole je pochopitelně možné čerpat na více technologií. Obecně je vlastně možné čerpat na všechny technologie, které byly na předchozí obrazovce zapnuty. Výběr vyluhovacího pole na nějakou technologii se provádí zaškrtnutím příslušného políčka. Jednotlivé technologie jsou rozlišeny barevně, jak bylo popsáno v předchozí kapitole.

Všechna nastavení týkající se vyluhovacích polí lze provést dvojím způsobem, buď přímo v mapě vyluhovacích polí, nebo v pravé části obrazovky. V mapě vybíráme vyluhovací pole tím, že na ně klikneme. Výběr technologií, na které se bude čerpat, provádíme zaškrtáváním barevných políček na mapě, nastavení počtu vrtů a intenzity čerpání můžeme provádět také přímo na mapě. V levé části obrazovky si opět můžeme vybrat vyluhovací pole, tentokrát však ze seznamu. K vyluhovacímu poli je možno přiřadit počet vrtů, intenzitu čerpání a čerpací technologie. Oba postupy zadávání dat jsou pochopitelně ekvivalentní a jsou vzájemně provázány. Pro upřesnění viz obrázek 4.4.1. V pravé dolní části obrazovky je tabulka nabídky čerpání. V této tabulce můžeme porovnat nabídku, která je definovaná počtem možných vrtů ve všech vybraných vyluhovacích polích a jejich vydatností, a potřebu, která je definovaná z technologických požadavků (ty byly stanoveny na předchozí obrazovce). Nabídka a potřeba je v tabulce rozepsaná pro jednotlivé technologie. Pokud je na nějaké technologii (nebo celkově) větší potřeba než nabídka, příslušný řádek tabulky zčervená a program nám neumožní přejít na další obrazovku. Tabulka nabídka čerpání viz obrázek 4.4.2.

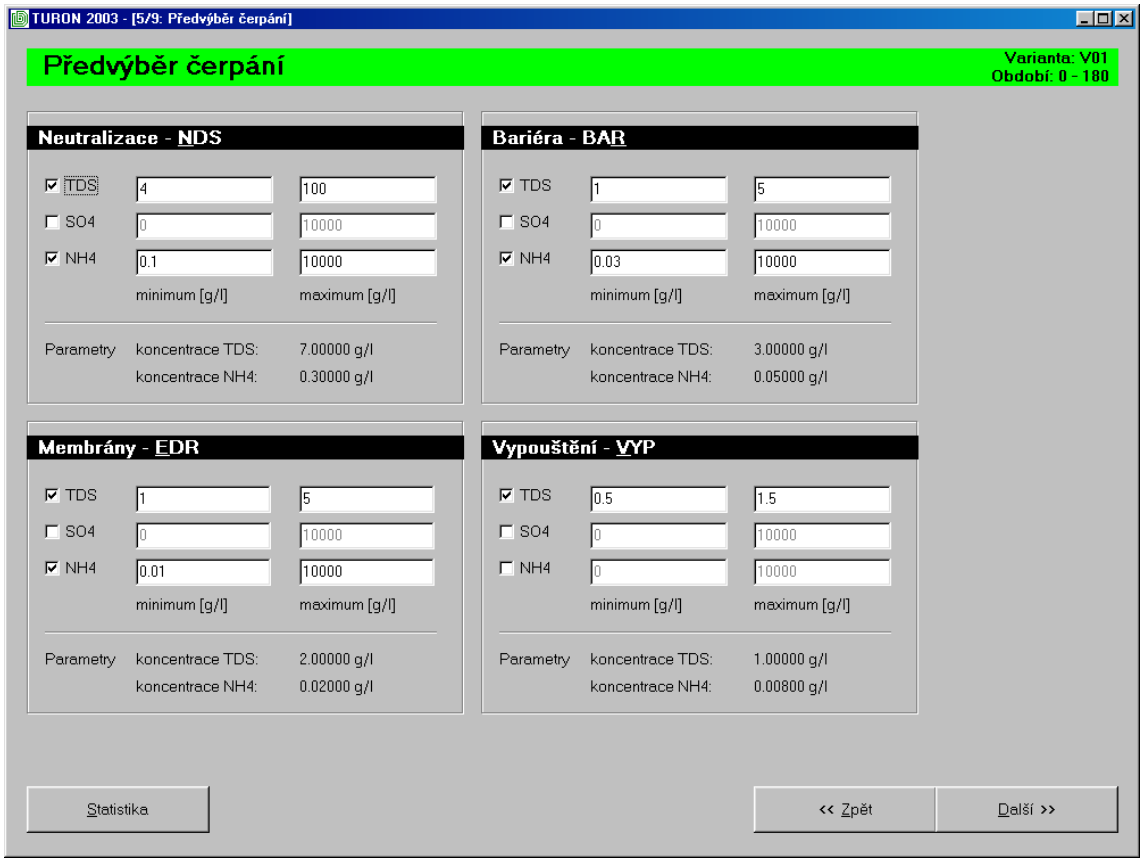
4.5 Obrazovka 5 – předvýběr čerpání a statistika

Obrazovka předvýběr čerpání nám umožňuje pro jednotlivé zapnuté technologie zapnout / vypnout a stanovit rozmezí – minimální a maximální – hodnoty koncentrací jednotlivých sledovaných složek kontaminace. Všechny hodnoty se zadávají v gramech na litr. Hodnoty, které jsou technologicky požadované (viz obrazovka 3 – zadání technologických požadavků; kapitola 4.3), jsou zde pro přehlednost pro každou technologii vypsané. Pokud rozmezí hodnot koncentrací nechceme stanovovat, necháme příslušnou položku vypnutou. Pokud položku zapneme, je potřeba tyto hodnoty stanovit. V případě složek TDS a NH4, kde jsou stanoveny technologické požadavky, by se navíc měla do uživatelem nastaveného rozmezí vejít právě i hodnota těchto požadavků. Celá obrazovka 5 je vidět na obrázku 4.5.1.

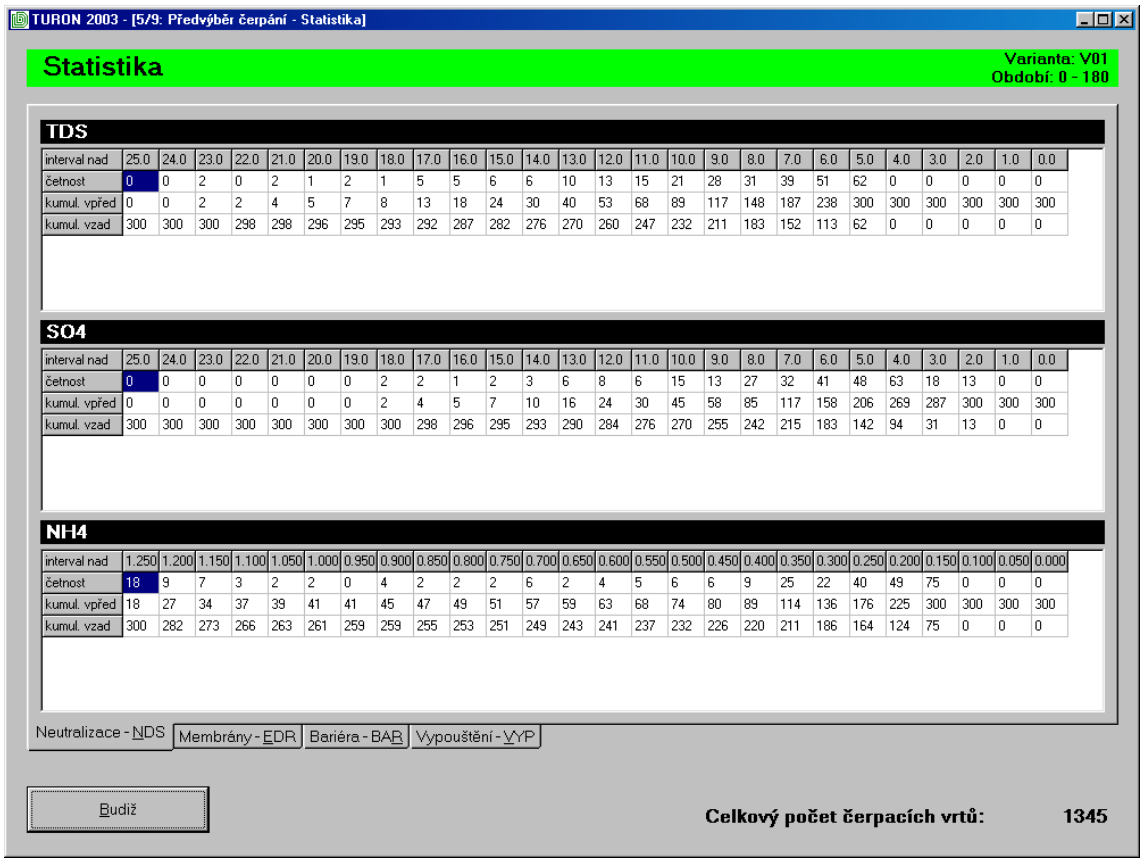
Předvýběr čerpacích vrtů je velmi důležitá součást programu. Dosud jsme stanovili v podstatě všechno, co je potřeba k vybrání optimální množiny čerpacích vrtů. Nalezení této množiny je ekvivalentní vyřešení lineárního problému, ke kterému je zde použit komerční software XA. Lineární model vlastně z množiny potenciálních vrtů vybere takové, které jsou schopny splnit technologické požadavky (požadovaný čerpaný objem a požadované koncentrace atd.) a které navíc splňují jistá omezení (například maximální počet vrtů na daném vyluhovacím poli). Množina potenciálních vrtů vlastně normálně obsahuje všechny vrty, což je cca 6000 vrtů. Možnosti poněkud zastaralého (přesto však kvalitního) softwaru XA jsou bohužel značně omezené a takto rozsáhlou úlohu by řešil velmi dlouho, pokud by vůbec byl schopen ji řešit. Předvýběr nám tedy umožní tuto množinu potenciálních vrtů omezit a vrty, z kterých bychom například čerpali roztoky s menšími než minimálními koncentracemi, z této množiny vyřadit. Množina potenciálních vrtů se tak výrazně zmenší a program XA je schopen bez problémů najít řešení takové úlohy. Omezení množiny potenciálních vrtů, a to jak lokalizací čerpání (obrazovka 4), tak intervalem koncentrací (obrazovka 5), navíc přispívá k výrazné linearizaci této úlohy. Zejména právě díky rozdílným koncentracím roztoků v různých vrtech, a to v rozmezí až několika řádů (0,1 – 10000 g/l), by byla úloha normálně totiž silně nelineární.

Ke sledování počtu předvybraných vrtů slouží obrazovka statistika (tlačítko „Statistika“ na této obrazovce). Na obrazovce statistiky se nám zobrazí celkový počet čerpacích vrtů. Dále pak počty vrtů vybrané na všechny vybrané technologie, rozepsané pro různé koncentrace jednotlivých složek. Počet všech předvybraných vrtů by měl být pro nalezení optimálního řešení maximálně cca 1300. Pokud počet vrtů překročí hodnotu 1600, program o tom uživatele automaticky informuje. Obrazovka statistiky je na obrázku 4.5.2.

Obr. 4.5.1 – obrazovka 5; předvýběr čerpání



Obr. 4.5.2 – obrazovka 5, předvýběr čerpání – statistika



4.5.1 Poznámka – vztah vrtů k elementům respektive multielementům

Celou dobu zde pracujeme střídavě s vrty a s elementy respektive s multielementy. V programu (v síti konečných prvků) však žádné vrty definované nejsou. Program za vrt považuje pro něj nejmenší horizontálně rozlišitelný prvek tedy multielement sítě. Proto zde můžeme mluvit o tom, že v daném vrtu jsou takové a takové koncentrace sledovaných složek kontaminace. Vybráním vrtu se pak tedy rozumí vybrání daného multielementu. Za vrt však nemůže být považován libovolný multielement, ale pouze takový, který leží na vyluhovacím poli. Přiřazení jednotlivých multielementů (vrtů) ke konkrétním vyluhovacím polím je jednoznačně definováno v souboru *LOKAL.ELM* v adresáři sítě (viz přílohy).

Na obrazovce 7 je možné již přímo zobrazit vrty, vybrané modelem k čerpání v daném období. Díky tomuto vztahu vrt – multielement, je při zobrazení vidět jak jednotlivé vybrané vrty „sedí“ v horizontálních těžištích multielementů (viz obrázek 4.7.8).

Díky tomuto vztahu je také možné považovat čerpací vrty – multielementy – za speciální typy okrajových podmínek ve výpočtu proudění a transportu látek.

4.6 Obrazovka 6 – volba účelové funkce a optimalizace čerpání

Na této obrazovce se nastavují poslední optimalizační parametry pro lineární model.

(a) *Volba účelové funkce*. Její význam je naprosto zřejmý – minimalizace / maximalizace obsahu jednotlivých složek kontaminace, respektive minimalizace počtu nových vrtů. Jedná se o funkci, kterou se snažíme v lineárním modelu extremalizovat (viz kapitola 2 – matematické metody).

(b) *Omezení pro počet nových vrtů*. Význam také naprosto jasný. Maximální počet nových vrtů s danou vydatností, které může model požadovat (některé vrty již existují, některé vrty je potřeba vytvářet nově, což je velmi nákladné). Typické nastavení je například deset vrtů à 25 l/min, dva vrty à 125 l/min a jeden vrt à 250 l/min.

(c) *Celočíselné řešení.* Program XA, který implementuje lineární model, umožňuje některé proměnné v tomto modelu považovat za celočíselné (také viz kapitola 2 – matematické metody). V programu se totiž může stát, že lineární model, aby dospěl k optimálnímu řešení, vybere některé čerpací vrty například jen z padesáti procent. Tento neduh lze do jisté míry ignorovat u existujících vrtů, ovšem v případě nových vrtů je to zásadní nedostatek. Právě proto, že vytváření nových vrtů je velmi nákladné, tak je rozdíl, zda se vytvoří dva nové vrty a ty se pak plně využívají nebo zda se má vytvořit osm nových vrtů a využívat je na pětadvacet procent. Naopak nevýhodou celočíselného řešení je to, že celočíselný výpočet trvá velmi dlouho (někdy řádově i desítky minut) a program pak třeba ani nenajde optimální řešení. Z tohoto důvodu je možné použít buď klasické – neceločíselné – řešení, nebo je možné za celočíselné proměnné považovat pouze nové vrty, nebo lze provést výpočet plně celočíselně. Tak zvané úplné celočíselné řešení však někdy nemusí vůbec existovat. Pokud totiž budeme uvažovat všechny vrty celočíselně, nemusí se nám podařit splnit požadovaný nátok na technologie. Výkony čerpacích vrtů jsou totiž jen 25, 125 a 250 l/min a pokud budeme mít stanoven nevhodný nátok na některou z technologií (viz obrazovka 3), například 1.01 m^3/min, nelze pochopitelně tento problém řešit celočíselně.

(d) *Koeficienty kompenzace poklesu koncentrací.* V průběhu jednoho výpočetního kroku (180 dní) se koncentrace – právě zejména vlivem čerpání – mění. Výsledky transportního modelu potom neodpovídají předpokladům zadaným v optimalizačním modelu. Tomu však lze předejít úpravou požadavků před vstupem do lineárního modelu. Protože koncentrace v podzemí nejčastěji klesají, požadavky musíme nejčastěji zvyšovat. Tyto koeficienty pak jsou typicky čísla nepatrně vyšší než jedna.

Tento jev se však projevuje zejména v závěrečné fázi sanace, kdy jsou z podzemí již odčerpány silné roztoky. V prvních výpočetních krocích tedy jsou tyto koeficienty přímo rovny jedné.

(e) *Požadavek na dodržení průměrných koncentrací.* Pokud nejsou zadány (zaškrtnuty) tyto požadavky, jsou v lineárním modelu všechny proměnné, které představují množství různých kontaminantů pro různé technologie, omezeny pouze seshora. Seshora jsou omezeny tak, jak bylo nastaveno na druhé obrazovce při zadání technologických požadavků (viz též kapitola 4.2), dolní mez pro jednotlivé

kontaminanty neexistuje. Pokud požadujeme dodržení průměrných koncentrací, jsou limity nastaveny jak seshora tak zezdola, ovšem s velmi malým intervalem. Pro kontaminant SO4 se tyto meze nenastavují vůbec. Nastavení mezí je dobře patrné na obrázku 4.7.1 v následující kapitole.

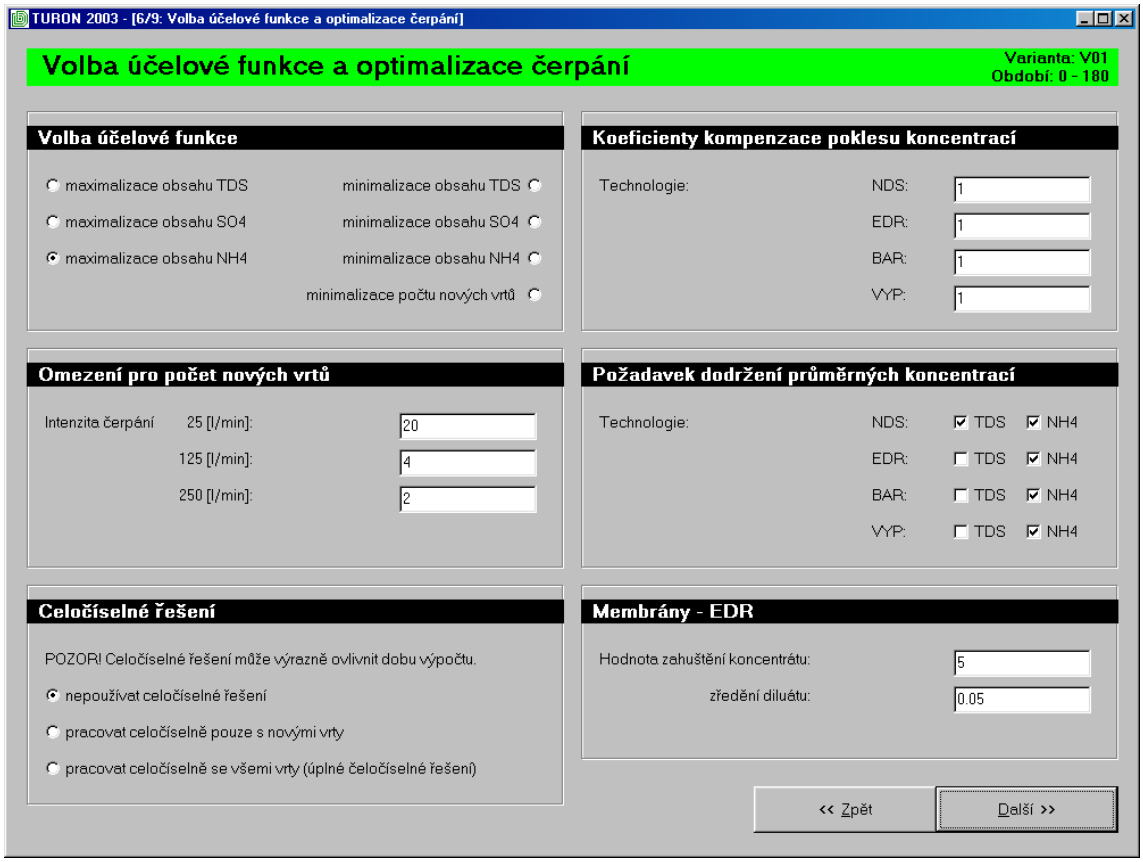
Pokud dodržení průměrných koncentrací nepožadujeme, výsledky lineárního modelu se mohou značně lišit od technologických požadavků. To se dělá, když je v podzemí již málo látek, takže nelze zajistit plně efektivní provoz technologií.

(f) Omezení vztahující se k membránové technologii – *zahuštění koncentrátu* a *zředění diluátu* – souvisí s principem této technologie. Do technologie vstupuje středně silný roztok, který se tam zahušťuje. Technologie má pak logicky dva výstupy, zahuštěný roztok – *koncentrát*, a zředěný roztok – *diluát*. Zahuštění koncentrátu je pak parametr, který nám říká, kolikrát se má zahustit koncentrát oproti koncentraci roztoku, který do technologie vstupuje. Typická hodnota zahuštění koncentrátu je 5. Zředění diluátu nám říká, kolikrát se zvýší koncentrace diluátu oproti roztoku vstupujícímu do technologie. Takový popis je sice přesný, ale může se zdát trochu nejasný. Diluát má nižší koncentraci než roztok původní. Pokud tedy tento parametr definujeme jako multiplikativní konstantu (kolikrát se zvýší koncentrace oproti původnímu roztoku), je jasné, že se vždy jedná o číslo menší než jedna. Typická hodnota tohoto parametru 0.1 – 0.05.

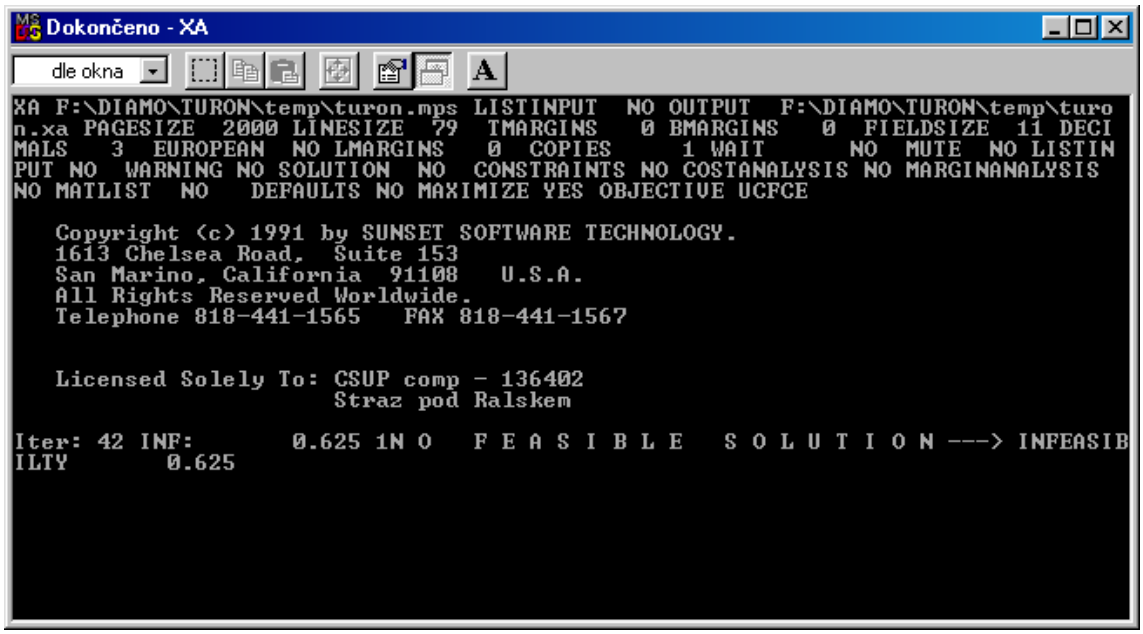
Po nastavení všech těchto parametrů se při přechodu na další obrazovku spustí program XA, realizující lineární model. Tento program se pokusí nalézt optimální řešení zadaného problému. Nalezením optimálního řešení se rozumí určení konkrétních vrtů, z kterých se má po dané období čerpat a které splňují všechny zadané požadavky. Program XA se spustí jako DOSová konzole. Výpočet vlastního problému bývá většinou velmi rychlý, pouze pokud počítáme celočíselně, může výpočet trvat řádově i několik minut. Program nám nahlásí, zda nalezl optimální řešení nebo ne – zobrazí hlášení „*OPTIMAL SOLUTION*“ v případě úspěchu. V případě, že byl problém zadán celočíselně, zobrazí se hlášení „*INTEGER SOLUTION*“. Pokud optimální řešení nenalezne, nebo pokud nenalezne korektní celočíselné řešení k již nalezenému optimálnímu řešení, zobrazí program hlášení „*NO FEASIBLE SOLUTION*“. Navíc program v případě nalezení řešení zobrazí extremalizovanou hodnotu účelové funkce nebo v případě neúspěchu zobrazí chybu, respektive odchylku od zadaných podmínek, které se musel dopustit (*INFEASIBILITY*). Celá obrazovka 6 je vidět na následujícím obrázku 4.6.1, program XA je na obrázku 4.6.2.

I na této obrazovce jsou jednotlivé položky, jež jsou vázány na konkrétní technologie (například právě zahuštění koncentrátu a zředění diluátu), dostupné pouze tehdy, když je daná technologie zapnutá (viz obrazovka 2 – kapitola 4.2 – zadání technologických požadavků).

Obr. 4.6.1 – obrazovka 6; volba účelové funkce a optimalizace čerpání



Obr. 4.6.2 – obrazovka 6; program XA nenalezl optimální řešení



4.7 Obrazovka 7 – výsledky lineárního modelu

V případě, že program XA nalezl optimální řešení tohoto problému, na obrazovce se optimální řešení ohlásí. Na této obrazovce je několik tabulek s celkem zřejmým významem:

Omezení modelu – v této tabulce jsou zapsány všechny omezující podmínky dané úlohy a jejich splnění. V tabulce je vždy zapsán název podmínky, její splnění, horní a dolní omezení (pokud existují).

Viz obrázek 4.7.1. První položkou v této tabulce je účelová funkce, ta pochopitelně nemá žádná omezení, je zde také zapsána hodnota, na kterou byla maximalizována. Dále je zde vidět několik typů podmínek. První z nich, tzv. „tvrdé“ podmínky (například Q_NDS – řádek 3, nebo Q_EDR – řádek 11), jsou omezeny z obou stran – ovšem s nulovým intervalem hodnot. Tyto podmínky musí být splněny přesně. Další podmínky (například TDS_NDS – řádek 4) jsou podmínky, kde se sice také klade omezení z obou stran, ovšem interval hodnot není nulový. Dále jsou zde vidět podmínky s omezením pouze seshora (TDS_EDR) a podmínky zcela „volné“, tedy bez omezení (SO4_EDR).

Také je zde vidět (jak již bylo zmiňováno v minulé kapitole), že byl právě například pro technologii NDS zapnut požadavek na dodržení průměrných koncentrací TDS a NH4 (TDS_NDS, NH4_NDS), zatímco pro technologii EDR byl zapnut pouze požadavek na dodržení průměrných koncentrací NH4 (TDS_EDR, NH4_EDR), což je vidět i na obrázku 4.6.1.

Dále viz obrázek 4.7.2. V tomto případě nebylo nalezeno optimální řešení. Je zde vidět, že podmínky 24 a 28 nebyly splněny. V obou těchto případech byla hodnota těchto proměnných pod dolní mezí. Jedná se o položky TDS_BAR a TDS_VYP. U obou položek jsou nastaveny obě dvě meze, což naznačuje, že byly zapnuty požadavky na dodržení koncentrací TDS pro technologie BAR a VYP, které způsobily nenalezení optimálního řešení.

Nenalezení optimálního řešení nemusí být způsobeno jen nesplněním omezení modelu. Chyba může vzniknout i při výběru vrtů (následující tabulka), kde může být nějaký vrt vybrán s hodnotou vyšší než jedna. To by znamenalo, že vrt budeme využívat z více než sta procent, což je samozřejmě nesmyslné, a také model to považuje za chybu.

Vybrané vrty – tato tabulka obsahuje seznam všech vrtů povolených předvýběrem. U vybraných vrtů je nastavena hodnota, s kterou je model vybral. Hodnota vybraných vrtů nám v podstatě říká, jakou „poměrnou část“ vrtu budeme využívat. Je-li tedy vrt vybrán například s hodnotou 0.451, znamená to, že budeme využívat 45.1% vydatnosti tohoto vrtu.

Viz obrázek 4.7.3. Zde je vidět část tabulky s jedním vybraným vrtem (položka 321). Tento vrt byl vybrán s hodnotou 0.725. Hodnota vrtu by logicky neměla vybočit z intervalu nula jedna. Může se však stát, že byl některý vrt vybrán s hodnotou vyšší než jedna, což je ovšem nahlášeno jako chyba a řádek s touto chybou je zvýrazněn červenou barvou stejně jako v tabulce omezení modelu. Ideální situace je, pokud jsou vrty vybírány s hodnotou buď nula nebo jedna a nic mezi tím. Toho lze dosáhnout použitím volby celočíselné řešení na obrazovce 6.

U všech vrtů jsou ještě v této tabulce zobrazeny doplňující informace – koncentrace jednotlivých kontaminantů v tomto vrtu v gramech na litr, lokalizace vrtu ve vyluhovacím poli, vydatnost vrtu v krychlových metrech za minutu. Poslední (respektive druhou) položkou v tabulce je název vrtu, který se vlastně skládá z technologie, na kterou byl vrt vybrán, a z čísla multielementu, kterým je vrt reprezentován. Z názvů vrtů je také patrné, že jeden vrt může být předvybrán na více technologií, vybrán by však měl být jen na jednu. Občas se ale může stát, že vrt je vybrán například na dvě technologie, na obě však s hodnotou 0.5. Takovou situaci model však jako chybu nenahlásí, protože celková hodnota vrtu je jedna, což je v pořádku. Tato situace ovšem nastává velmi zřídka a z pohledu ryze praktického příliš nevadí. Program proto tuto situaci zatím nijak neošetřuje. Takovému vrtu je programem natvrdo přidělena jen jedna z obou technologií.

Tyto dvě hlavní tabulky vlastně obsahují přímo návratová data programu XA. Všechny ostatní tabulky již obsahují jen výtahy z těchto tabulek.

Seznamy vrtů – tato tabulka obsahuje pouze stručný seznam vybraných existujících a nových vrtů. Je zde vlastně seznam vrtů, u kterých jsou zaškrtnuty příslušné atributy, podle toho zda je vrt vybraný lineárním modelem. Zda tento vrt existuje, nebo zda je nový. Ostatní vrty zde zobrazeny nejsou. Mohou se tak tedy v této tabulce vyskytnout vlastně jen tři různé kombinace atributů. Viz obrázek 4.7.4.

Vrt byl vybrán modelem a zatím neexistuje, je tudíž novým vrtem (např. vrt 6).
Vrt byl vybrán modelem a již existuje (např. vrt 45).
Vrt již existuje, ale modelem vybrán nebyl (např. vrt 6126).

Vyluhovací pole – informace o jednotlivých vybraných vyluhovacích polích setříděné podle technologií. Ke každému vyluhovacímu poli je zde celkový objem vyvedených látek na danou technologii a koncentrace jednotlivých kontaminantů. Viz obrázek 4.7.5.

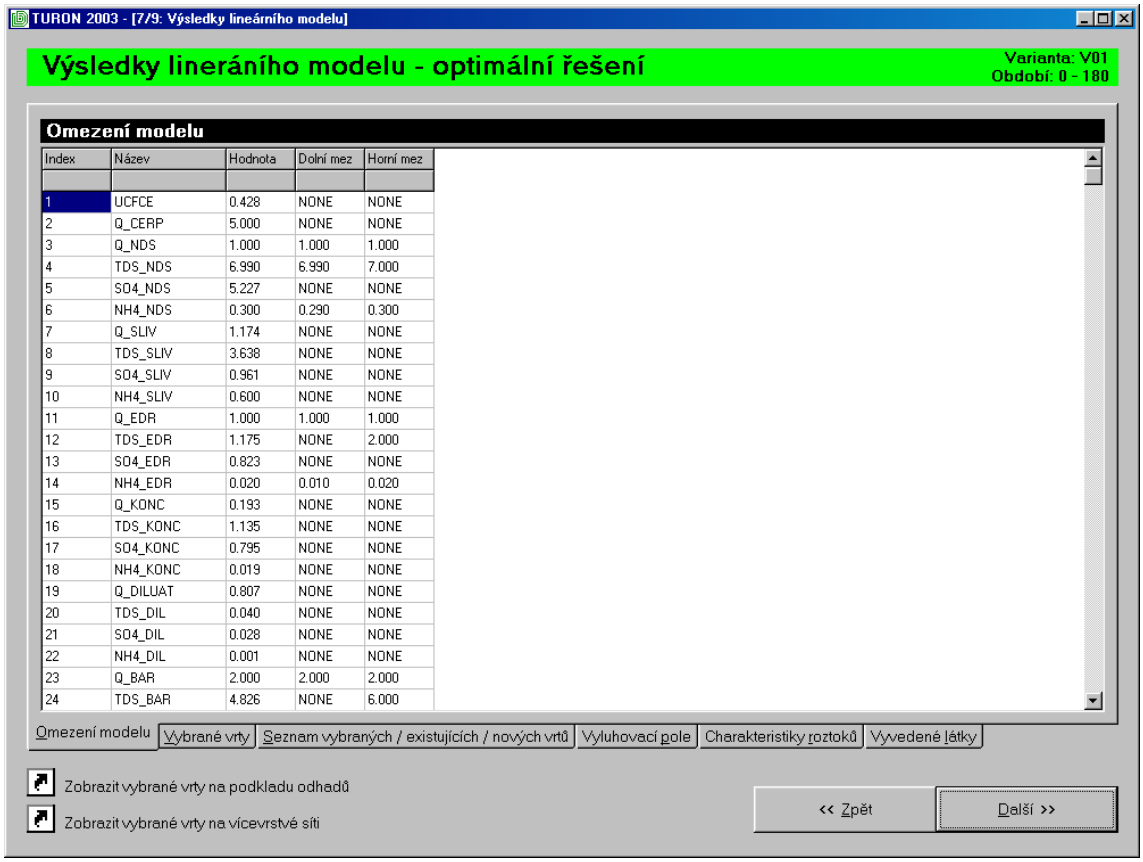
Charakteristiky roztoků – celkové objemy (v kubických metrech za minutu) a koncentrace sledovaných složek (v gramech na litr) připadající na jednotlivé technologie. Obsahuje také objemy a koncentrace nátoku na technologie, slivu z neutralizační technologie a objemy a koncentrace produktů (koncentrátu a diluátu) vystupujících z membránových technologií. Viz obrázek 4.7.6.

Vyvedené látky – celkové objemy (v kubických metrech za minutu) a hmotnosti vyvedených sledovaných složek (kilogramy za minutu) připadající na jednotlivé technologie. Viz obrázek 4.7.7.

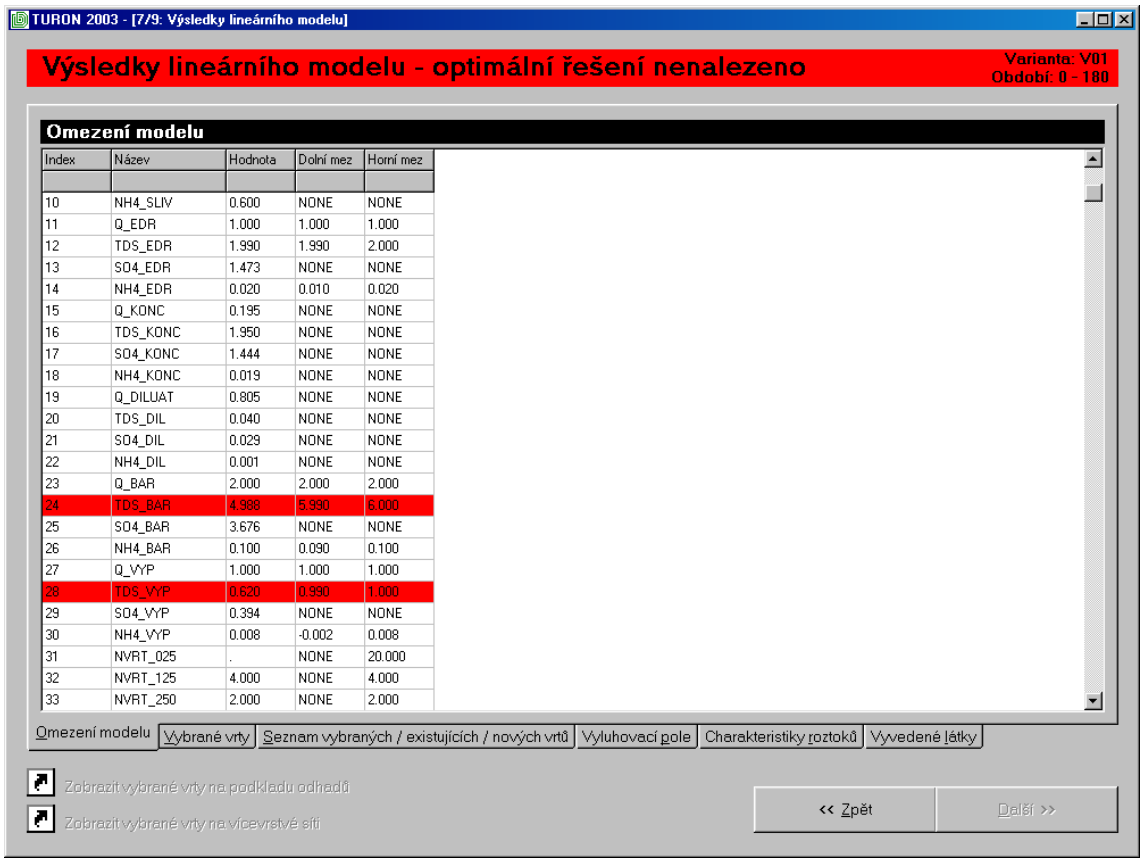
Pokud bylo nalezeno optimální řešení, je na sedmé obrazovce také možno zobrazit vybrané vrty, a to jak na jednovrstvé síti (na podkladu odhadů – soubor *odhCAS.ts3*), tak na vícevrstvé síti (na podkladu počátečních dat – soubory *mCAS.ts3* a *mCAS_slow.ts3*). Při zobrazení vybraných vrtů na odhadech vidíme vlastně vrty zobrazené na datech, podle kterých byly tyto vrty vybrány. Ze souboru odhadů totiž načítáme data do lineárního modelu. Zatímco při zobrazení na vícevrstvé síti vidíme vrty na skutečných „reálných“ datech. Protože se jedná o vícevrstvé soubory, je možné v programu GwsView tyto jednotlivé vrstvy procházet a sledovat rozložení kontaminace v každé vrstvě zvlášť. Na obrázcích 4.7.9. a 4.7.10 jsou vidět zobrazené vybrané vrty na podkladu odhadů a na vícevrstvé síti. Na obrázku 4.7.8 je zvětšený detail, na kterém je vidět, jak se vrt zobrazuje do horizontálního těžiště multielementu, zároveň je zde vidět, že nové vrty jsou ve zobrazení označeny jinak, a tudíž není těžké je identifikovat.

Pokud program XA nenalezl optimální řešení, program nám neumožní pokračovat ve výpočtu a nelze ani zobrazit vybrané vrty. Musíme se tedy vrátit a pozměnit omezující podmínky lineárního modelu, jak již zda bylo diskutováno.

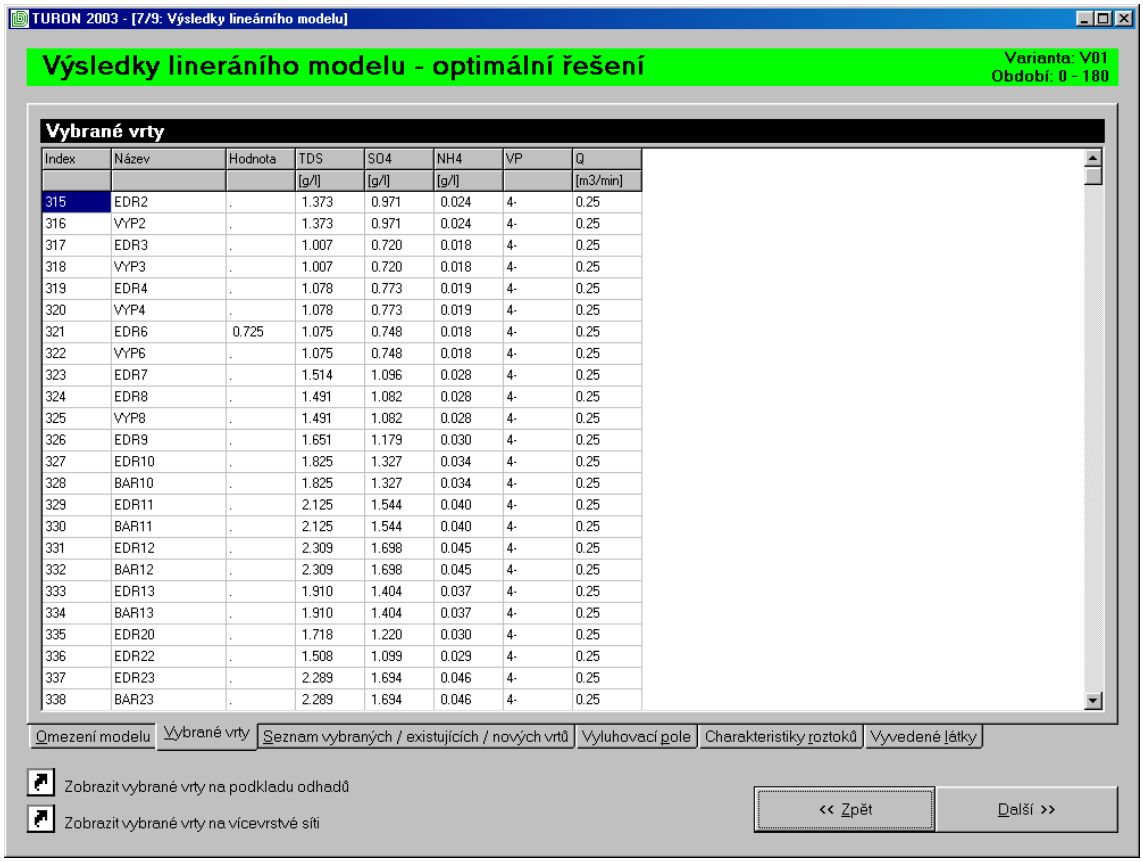
Obr. 4.7.1 – obrazovka 7; tabulka omezení modelu; optimální řešení nalezeno



Obr. 4.7.2 – obrazovka 7; tabulka omezení modelu; optimální řešení nenalezeno

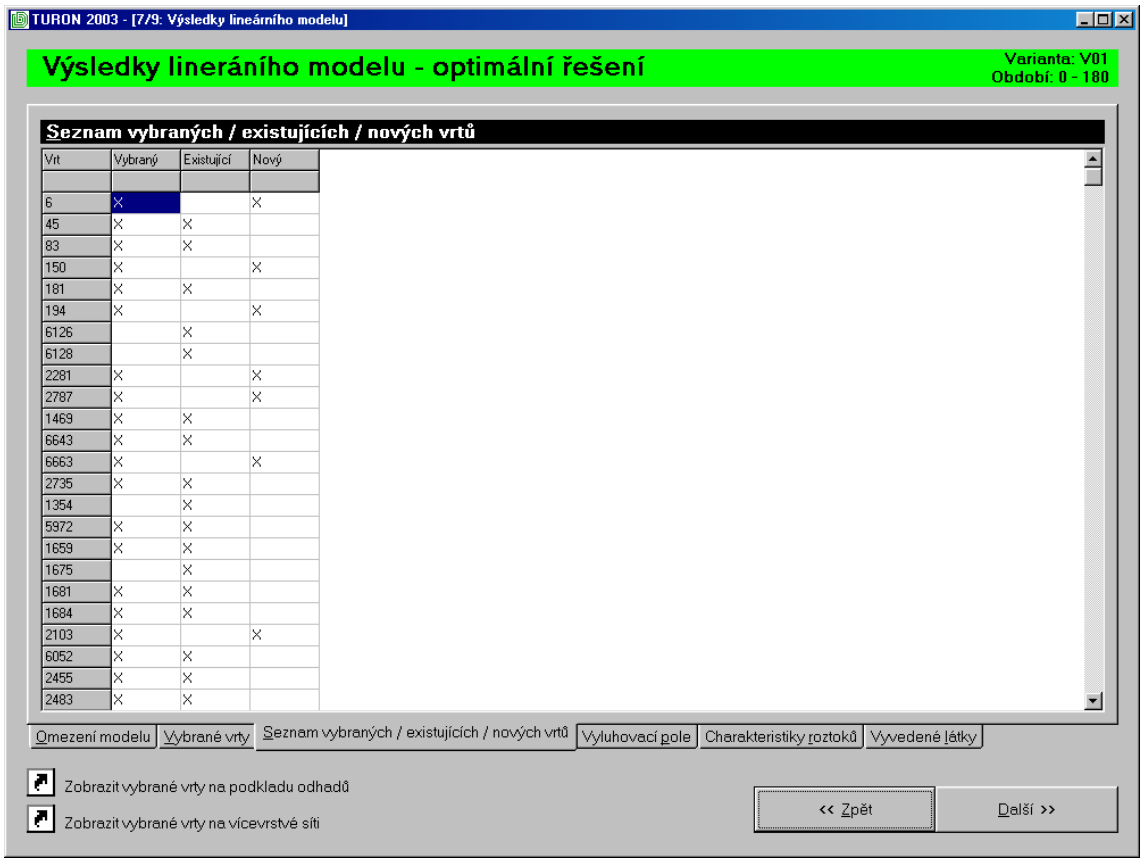


Obr. 4.7.3 – obrazovka 7; tabulka vybrané vrty



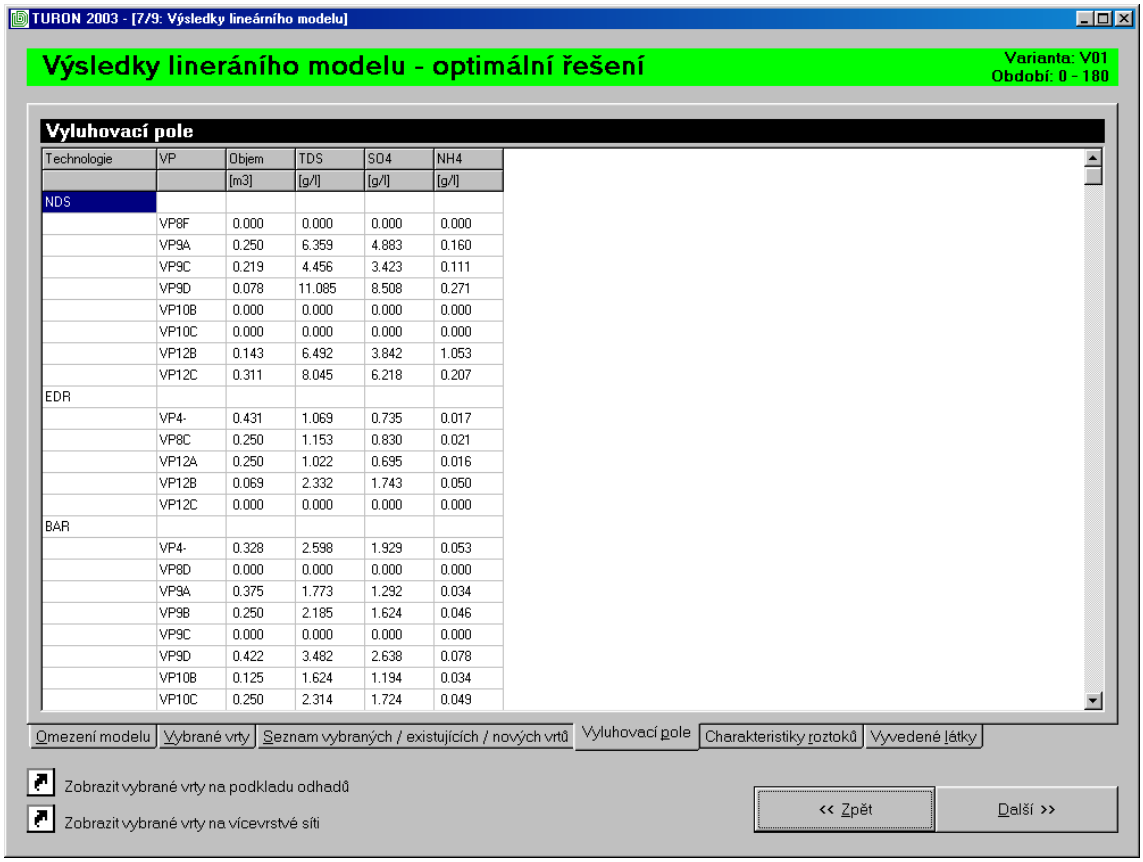
Index	Název	Hodnota	TDS	SO4	NH4	VP	Q
			[g/l]	[g/l]	[g/l]		[m3/min]
315	EDR2	.	1.373	0.971	0.024	4-	0.25
316	VYP2	.	1.373	0.971	0.024	4-	0.25
317	EDR3	.	1.007	0.720	0.018	4-	0.25
318	VYP3	.	1.007	0.720	0.018	4-	0.25
319	EDR4	.	1.078	0.773	0.019	4-	0.25
320	VYP4	.	1.078	0.773	0.019	4-	0.25
321	EDR6	0.725	1.075	0.748	0.018	4-	0.25
322	VYP6	.	1.075	0.748	0.018	4-	0.25
323	EDR7	.	1.514	1.096	0.028	4-	0.25
324	EDR8	.	1.491	1.082	0.028	4-	0.25
325	VYP8	.	1.491	1.082	0.028	4-	0.25
326	EDR9	.	1.651	1.179	0.030	4-	0.25
327	EDR10	.	1.825	1.327	0.034	4-	0.25
328	BAR10	.	1.825	1.327	0.034	4-	0.25
329	EDR11	.	2.125	1.544	0.040	4-	0.25
330	BAR11	.	2.125	1.544	0.040	4-	0.25
331	EDR12	.	2.309	1.698	0.045	4-	0.25
332	BAR12	.	2.309	1.698	0.045	4-	0.25
333	EDR13	.	1.910	1.404	0.037	4-	0.25
334	BAR13	.	1.910	1.404	0.037	4-	0.25
335	EDR20	.	1.718	1.220	0.030	4-	0.25
336	EDR22	.	1.508	1.099	0.029	4-	0.25
337	EDR23	.	2.289	1.694	0.046	4-	0.25
338	BAR23	.	2.289	1.694	0.046	4-	0.25

Obr. 4.7.4 – obrazovka 7; tabulka seznam vybraných / existujících / nových vrtů

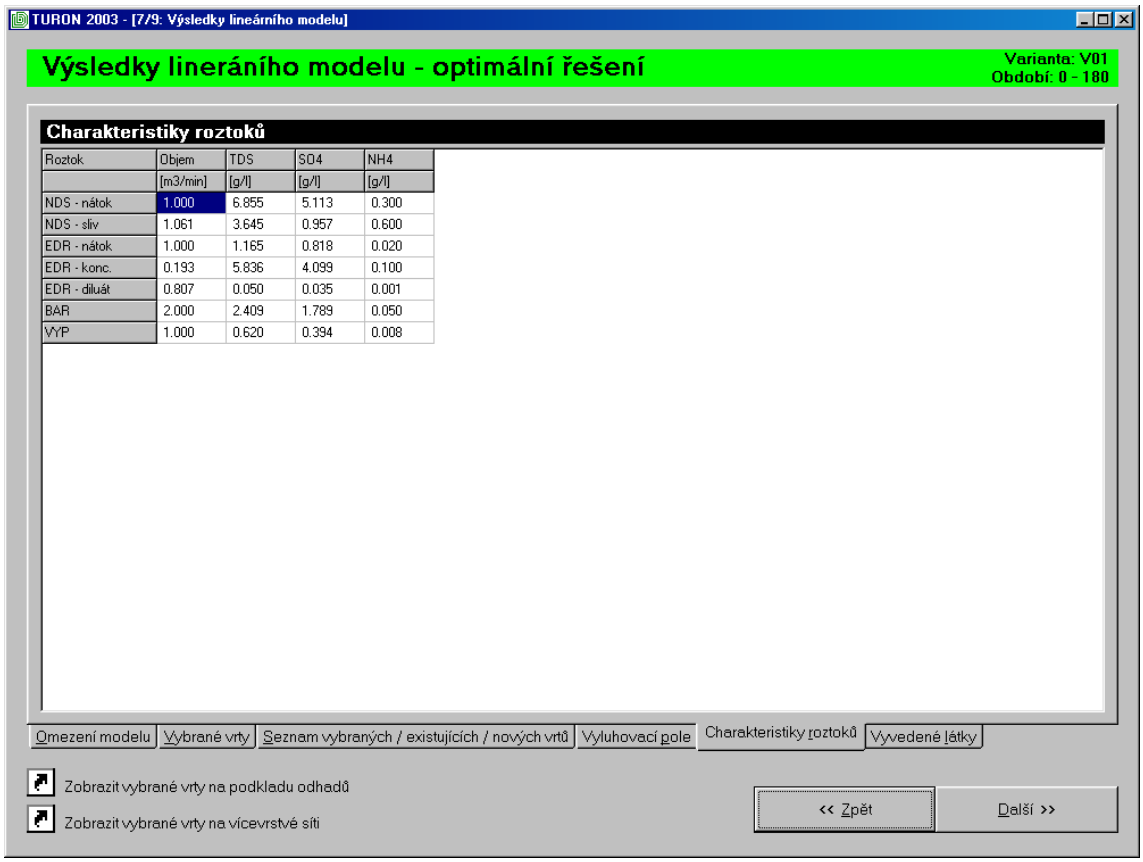


Vrt	Vybraný	Existující	Nový
6	X		X
45	X	X	
83	X	X	
150	X		X
181	X	X	
194	X		X
6126		X	
6128		X	
2281	X		X
2787	X		X
1469	X	X	
6643	X	X	
6663	X		X
2735	X	X	
1354		X	
5972	X	X	
1659	X	X	
1675		X	
1681	X	X	
1684	X	X	
2103	X		X
6052	X	X	
2455	X	X	
2483	X	X	

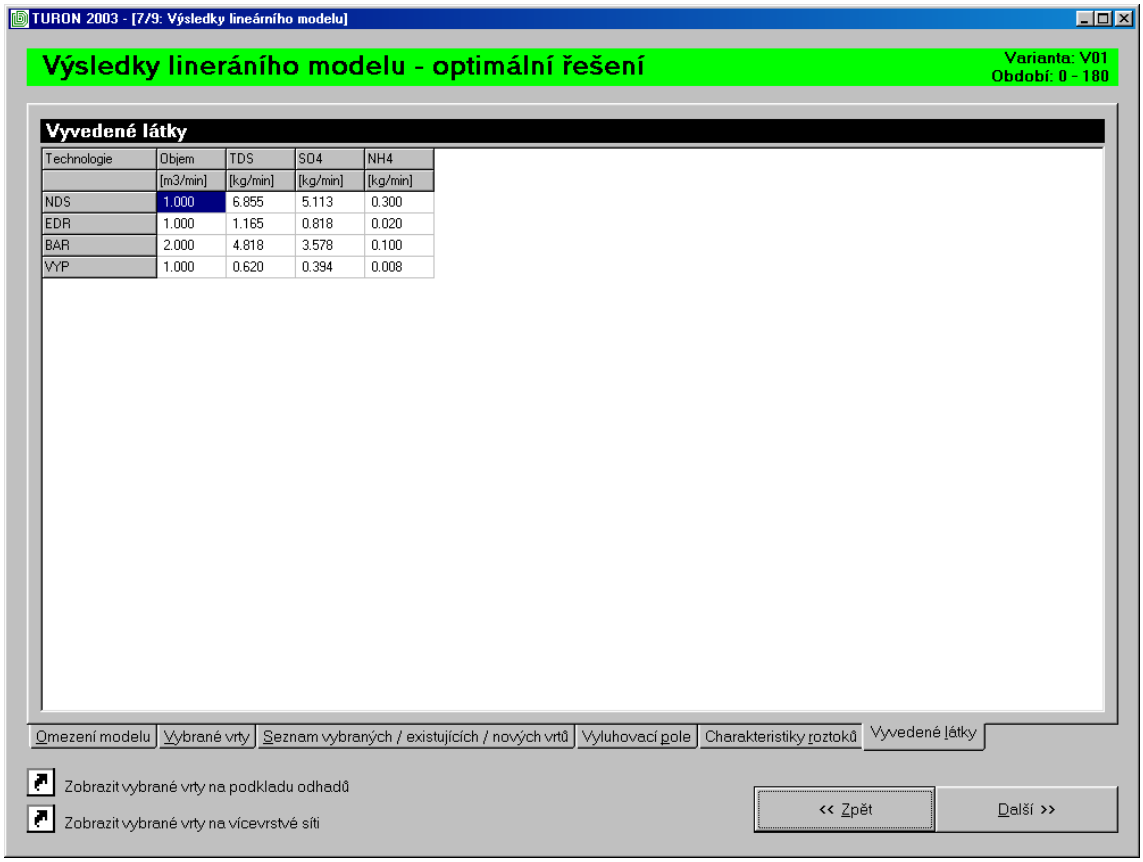
Obr. 4.7.5 – obrazovka 7; tabulka vyluhovací pole



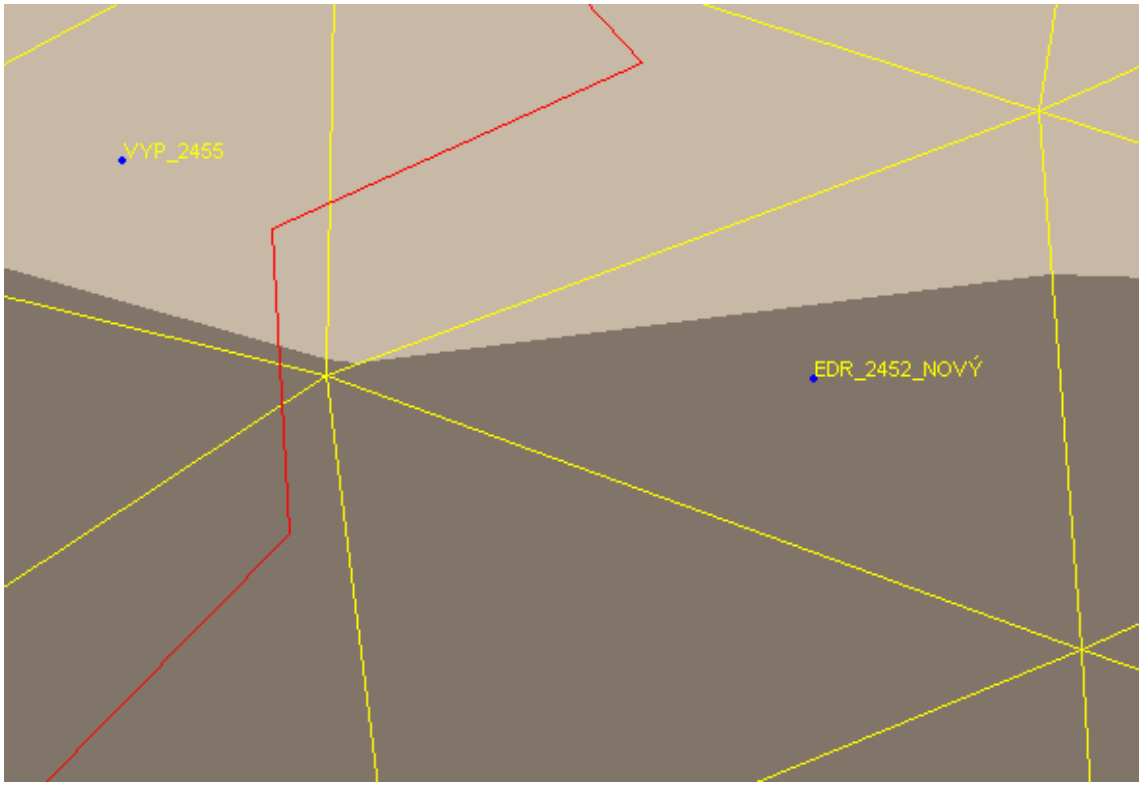
Obr. 4.7.6 – obrazovka 7; tabulka charakteristiky roztoků



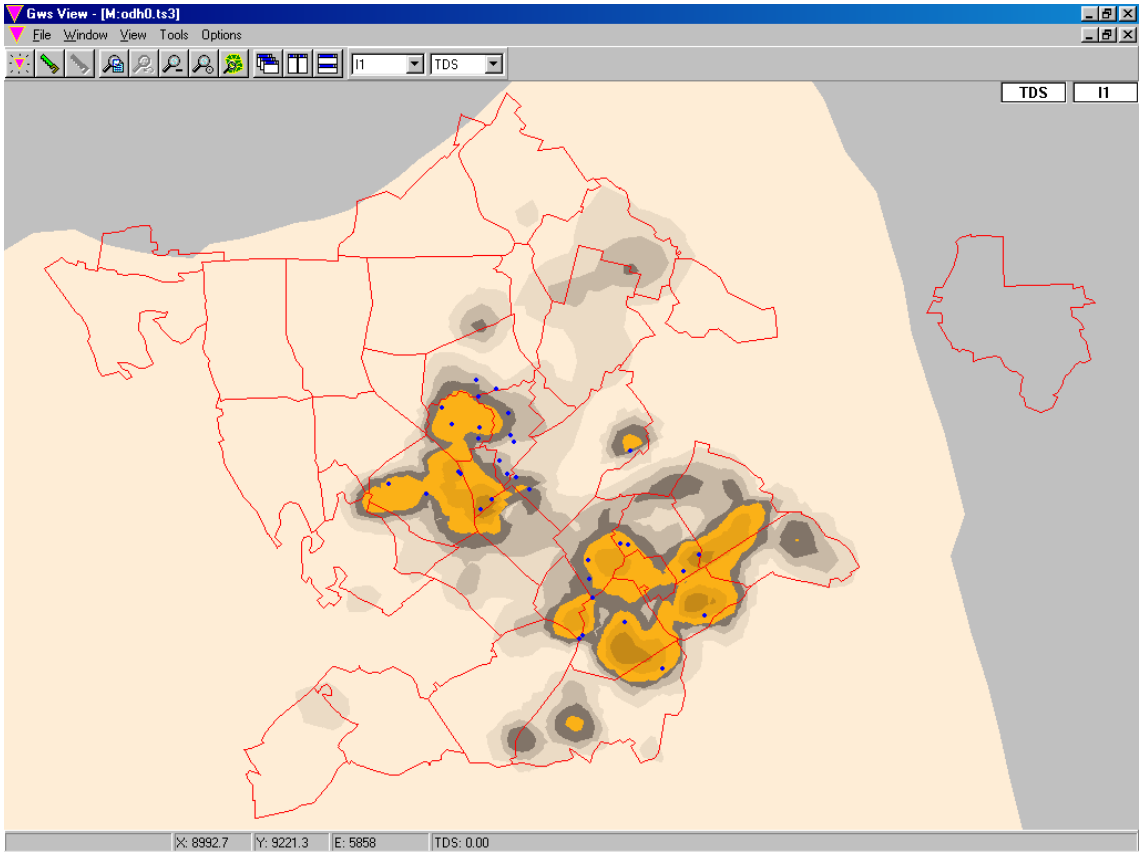
Obr. 4.7.7 – obrazovka 7; tabulka vyvedené látky



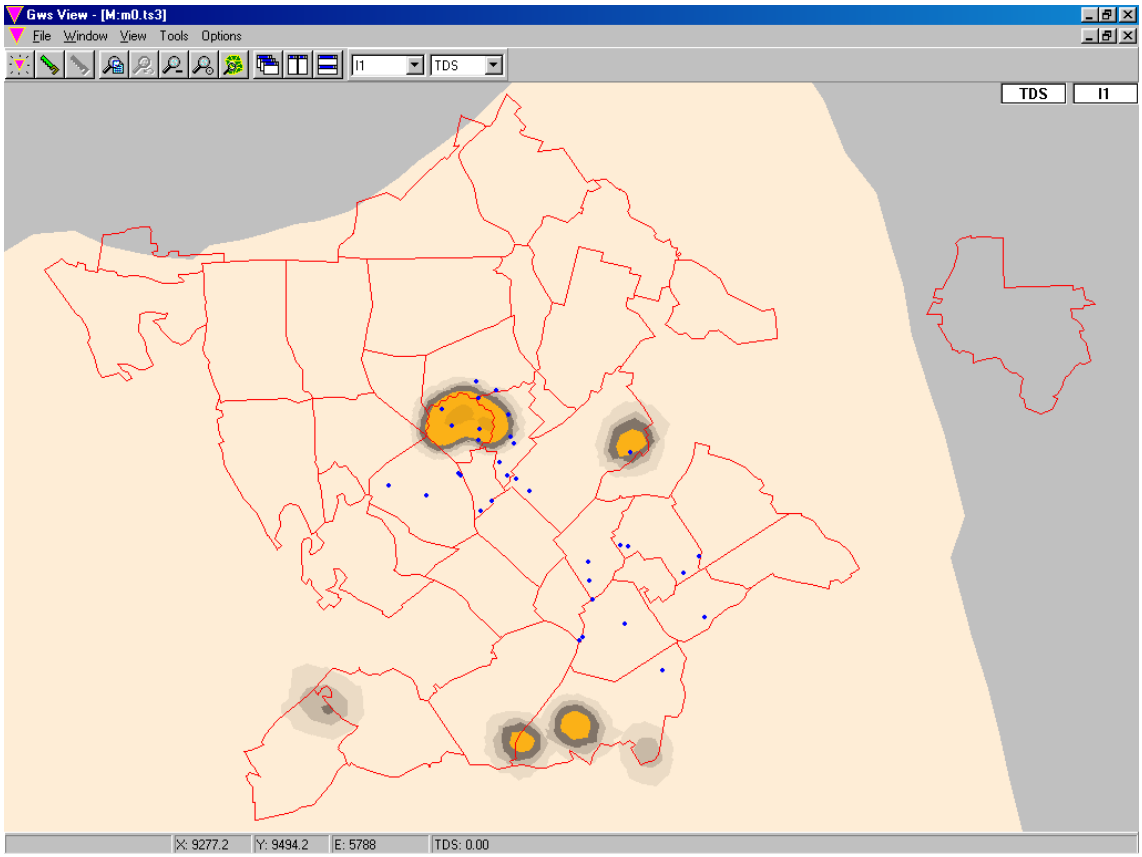
Obr. 4.7.8 – zobrazení vybraných vrtů na podkladu odhadů v programu GwsView – detail; rozdíl mezi novým a existujícím vrtem; je dobře patrné, jak je vrt umístěn v horizontálním těžišti mutlielementu



Obr. 4.7.9 – zobrazení vybraných vrtů na podkladu odhadů v programu GwsView



Obr. 4.7.10 – zobrazení vybraných vrtů na vícevrstvé síti v programu GwsView – zobrazena vrstva I1



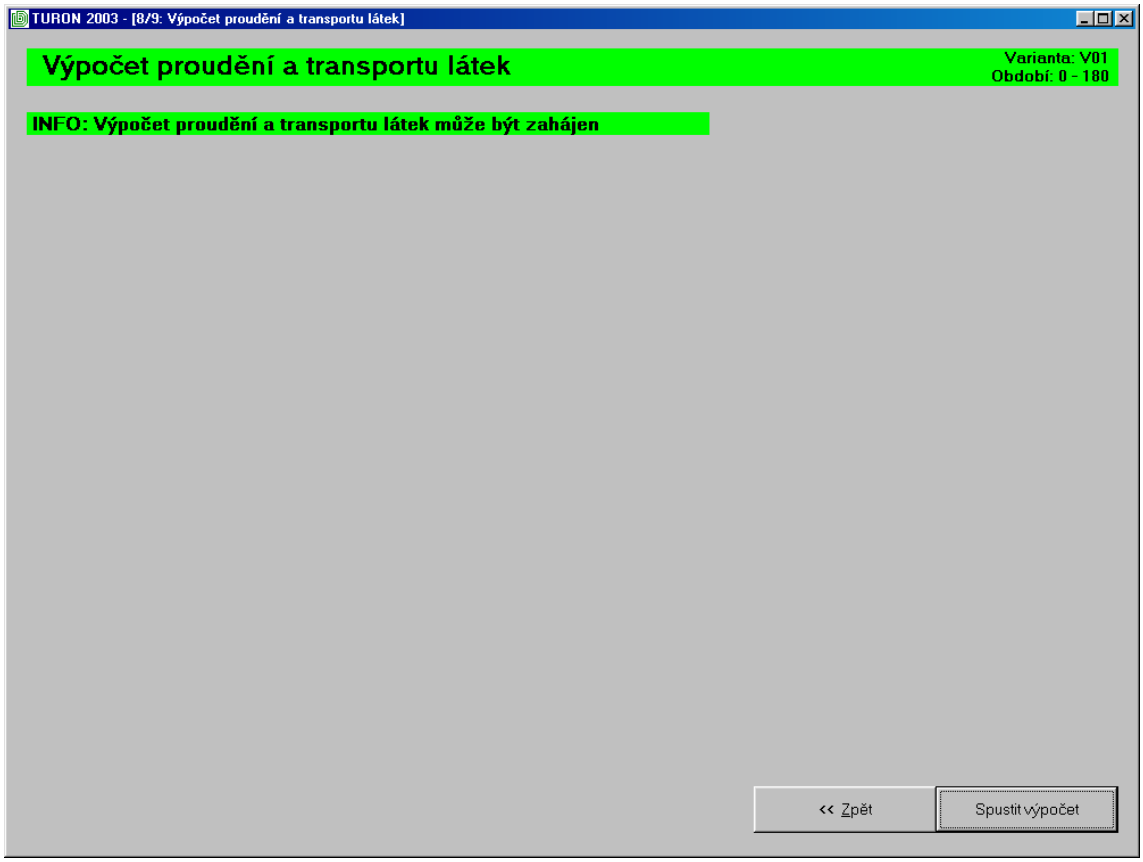
4.8 Obrazovka 8 – výpočet proudění a transportu látek

Pokud již máme vybrané vrty, můžeme sestavit kompletní okrajové podmínky pro výpočet proudění a transportu látek. Okrajová podmínka je sestavena ze dvou částí – konstantní podmínka (trvale uložená v souborech v adresáři sítě, viz přílohy) a nekonstantní podmínka, která je právě tvořená vybranými čerpacími vrty (multielementy, respektive konkrétními vrstvami v nich). Počáteční podmínku pro výpočet transportu látek tvoří soubory s aktuálním stavem kontaminace (např. *m180.ts3* a *m180_slow.ts3*).

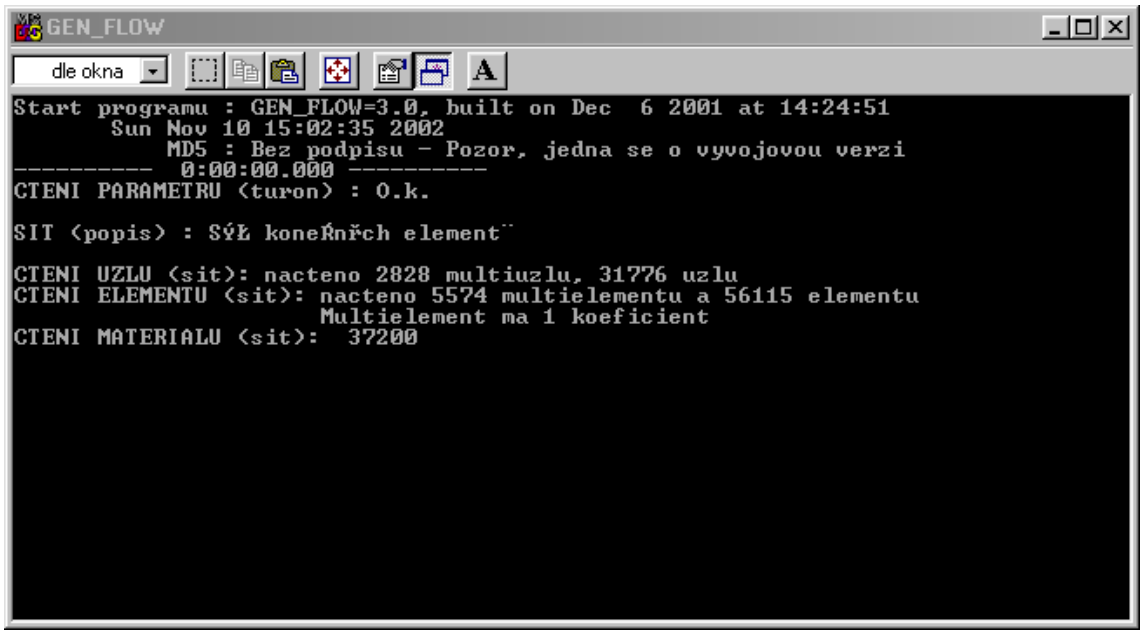
Jak již bylo několikrát řečeno, výpočet probíhá ve dvou hlavních fázích. Výpočet proudění podzemní vody realizovaný programem GEN_FLOW a výpočet transportu látek realizovaný programem GEN_TRAN. Třetí závěrečnou fází je zpracování výsledků. Všechny tři fáze se odrážejí v chování programu.

Při vstupu na osmou obrazovku systém automaticky hlásí, že je připraven k spuštění výpočtu proudění a transportu látek. Nyní se buď můžeme vrátit na předchozí obrazovku, nebo zahájit výpočet (viz obrázek 4.8.1). Po zahájení výpočtu nás program informuje o tom, co se aktuálně děje – zda běží výpočet proudění, transportu látek, nebo zda probíhá zpracování výsledků (viz obrázky 4.8.2, 4.8.3). Po ukončení výpočtu můžeme pokračovat na další obrazovku (viz obrázek 4.8.4). Pozor – pokud se nyní vrátíme zpět, budeme muset při dalším vstupu na tuto obrazovku absolvovat výpočet proudění a transportu znovu.

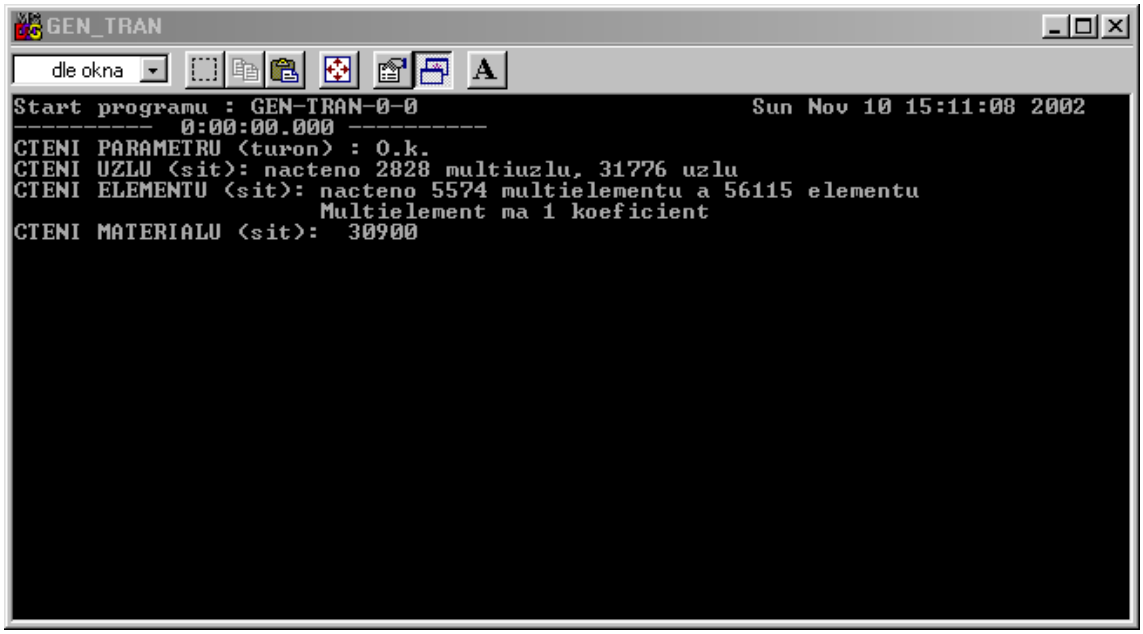
Obr. 4.8.1 – obrazovka 8; výpočet může být zahájen



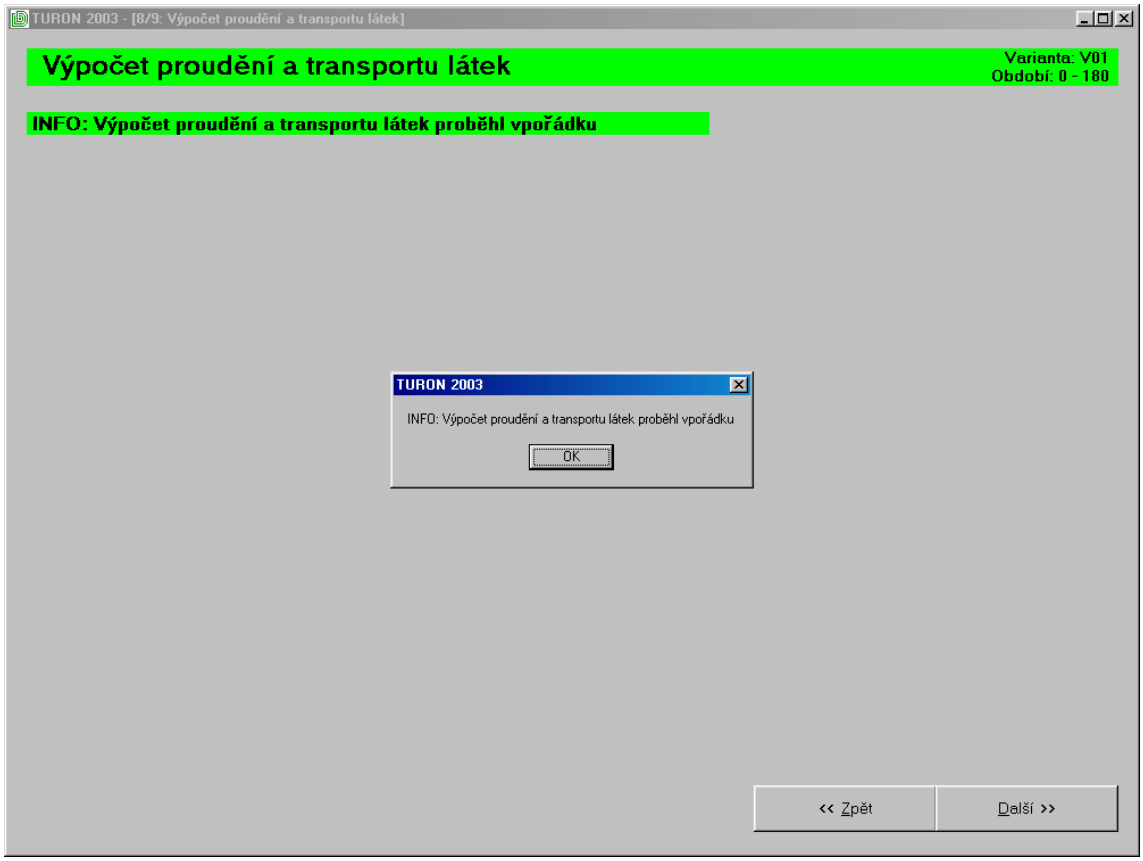
Obr. 4.8.2 – obrazovka 8; probíhá výpočet proudění – program GEN_FLOW



Obr. 4.8.3 – obrazovka 8; probíhá výpočet transportu látek – program GEN_TRAN



Obr. 4.8.4 – obrazovka 8; výpočet proběhl v pořádku



4.9 Obrazovka 9 – výsledky modelu transportu

Na této obrazovce jsou již k dispozici jen výsledky výpočtu proudění a transportu látek a ekonomického modelu v několika tabulkách:

Průběh varianty – celý průběh dané varianty; zde se zobrazují všechny doposud vypočtené kroky. Je zde celková hmotnost vyvedených látek, parametry a požadavky jednotlivých technologií. Vedle požadavků technologií jsou zde i skutečné vypočtené hodnoty těchto parametrů. Viz obrázek 4.9.1.

dtto bez požadavků – tato tabulka je naprosto identická s tabulkou předchozí, pouze s tím rozdílem, že zde jsou vynechány hodnoty požadavků. Tato tabulka tedy obsahuje pouze vypočtené hodnoty. Viz obrázek 4.9.2.

Ekonomické hodnocení – výstup ekonomického modelu. Tabulka obsahuje náklady na jednotlivé technologie (okamžité i kumulované), náklady na pořizování nových vrtů, celkové náklady a jednotkové náklady, tedy celkové náklady přepočítané na jeden kilogram TDS. Viz obrázek 4.9.4.

dtto Grafy – grafy ekonomického hodnocení. Vykreslují se zde do grafů okamžité náklady jednotlivých technologií v čase, celkové náklady jednotlivých technologií v čase a vývoj průměrných jednotkových nákladů v čase (viz obrázky 4.9.5 – 4.9.7). Jednotlivé grafy je možné ukládat do souborů a to ve formátu *BMP*, *WMF* a *EMF*. Obrázky se ukládají do adresáře varianty (pokud již soubor se stejným jménem existuje bude automaticky zálohován).

Výsledky – NDS – vypočtené výsledky aktuálního kroku vztažené na jednu technologii (NDS). Zejména kvantifikace rozdílů mezi požadavky a vypočtenou skutečností.

Výsledky – EDR – vypočtené výsledky aktuálního kroku vztažené na jednu technologii (EDR). Zejména kvantifikace rozdílů mezi požadavky a vypočtenou skutečností.

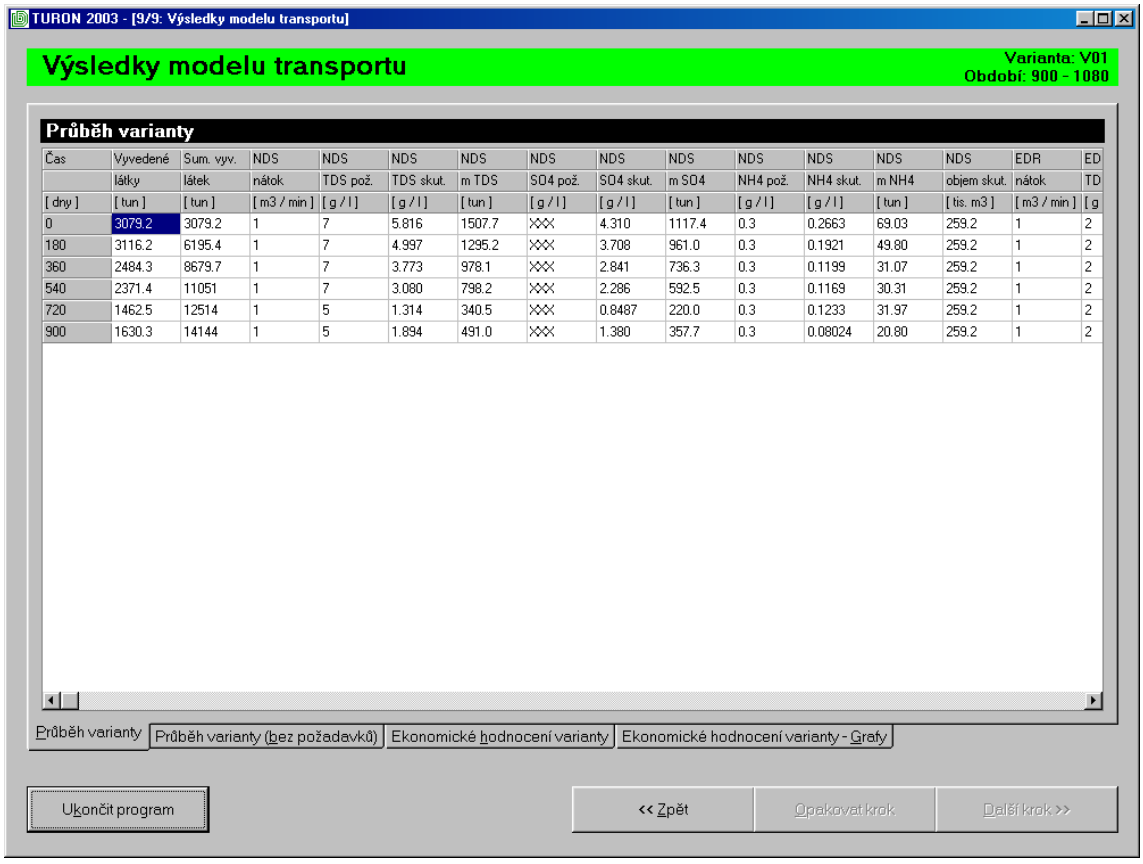
Výsledky – BAR – vypočtené výsledky aktuálního kroku vztažené na jednu technologii (BAR). Zejména kvantifikace rozdílů mezi požadavky a vypočtenou skutečností.

Výsledky – VYP – vypočtené výsledky aktuálního kroku vztažené na jednu technologii (VYP). Zejména kvantifikace rozdílů mezi požadavky a vypočtenou skutečností.
Tabulky s výsledky - viz obrázek 4.9.3.

Všechny tyto tabulky se automaticky ukládají do souboru ve formátu *XLS* – MS Excel (soubor *vysledkyCAS.xls*). Po dokončení výpočtu můžeme z této obrazovky celý systém buď ukončit, nebo můžeme pokračovat ve výpočtu. Pokračovat se dá dvojím způsobem. Můžeme opakovat tento krok výpočtu, můžeme tedy provést výpočet znovu pro to stejné období. Také můžeme pokračovat ve výpočtu dalšího kroku varianty. V obou případech systém automaticky přeskočí zpět na první obrazovku. Systém pouze automaticky vyplní název varianty a období výpočtu. Tyto položky však mohou být pochopitelně uživatelem změněny.

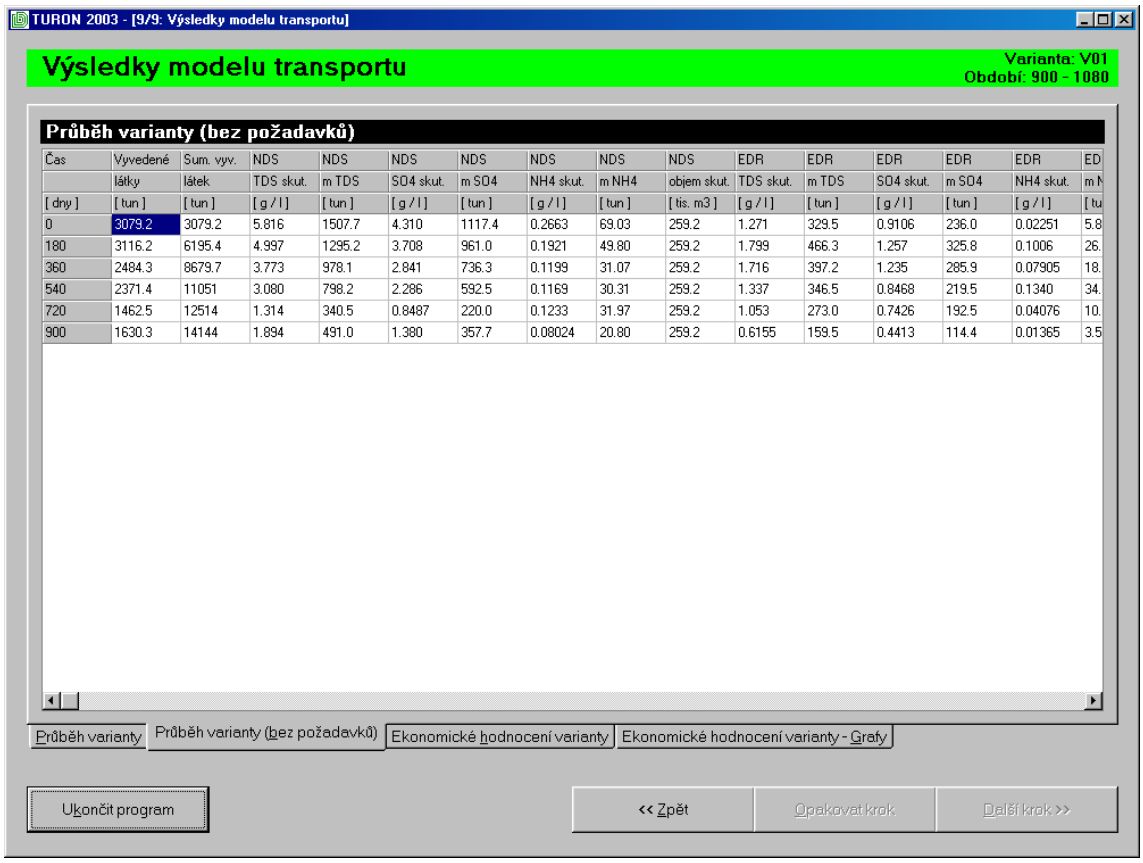
Pokud máme zapnuté tzv. debug menu (viz kapitola 4.1, obrázek 4.1.3), můžeme se z první obrazovky dostat pomocí volby SKIP_ALL v tomto menu rovnou na poslední obrazovku. V takovém případě se zobrazí pochopitelně jen první čtyři tabulky, které se vesměs načítají z výsledků uložených v *ini* souborech s parametry. Poslední tabulky, které zobrazují výsledky aktuálně vypočteného kroku, k dispozici nejsou. Rovněž pokud jsme na obrazovce 2 vybrali například jen tři technologie, z posledních čtyř tabulek se zobrazí jen příslušné tři. V případě že máme zapnuté SKIP_ALL, jsou na posledním okně zablokována i tlačítka „opakovat krok“ a „další krok výpočtu“. To je dobře patrné například na obrázcích 4.9.1 – 4.9.4.

Obr. 4.9.1 – obrazovka 9; tabulka průběh varianty (byla použita volba SKIP_ALL)



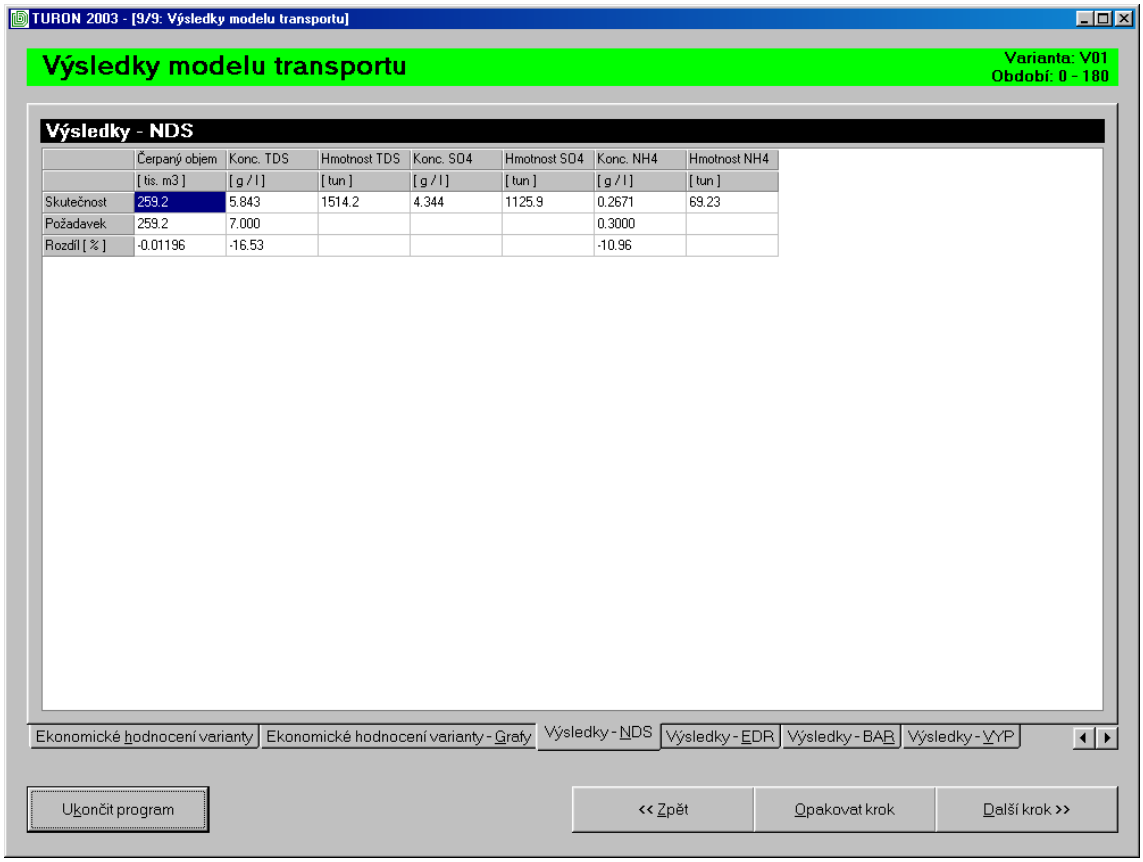
Čas	Vyvedené	Sum. vyv.	NDS	NDS	NDS	NDS	NDS	NDS	NDS	NDS	NDS	NDS	NDS	EDR	ED
	látky	látek	nátok	TDS pož.	TDS skut.	m TDS	SO4 pož.	SO4 skut.	m SO4	NH4 pož.	NH4 skut.	m NH4	objem skut.	nátok	TD
[dny]	[tun]	[tun]	[m3 / min]	[g / l]	[g / l]	[tun]	[g / l]	[g / l]	[tun]	[g / l]	[g / l]	[tun]	[tis. m3]	[m3 / min]	[g
0	3079.2	3079.2	1	7	5.816	1507.7	XXX	4.310	1117.4	0.3	0.2663	69.03	259.2	1	2
180	3116.2	6195.4	1	7	4.997	1295.2	XXX	3.708	961.0	0.3	0.1921	49.80	259.2	1	2
360	2484.3	8679.7	1	7	3.773	978.1	XXX	2.841	736.3	0.3	0.1199	31.07	259.2	1	2
540	2371.4	11051	1	7	3.080	798.2	XXX	2.286	592.5	0.3	0.1169	30.31	259.2	1	2
720	1462.5	12514	1	5	1.314	340.5	XXX	0.8487	220.0	0.3	0.1233	31.97	259.2	1	2
900	1630.3	14144	1	5	1.894	491.0	XXX	1.380	357.7	0.3	0.08024	20.80	259.2	1	2

Obr. 4.9.2 – obrazovka 9; tabulka průběh varianty bez požadavků (byla použita volba SKIP_ALL)



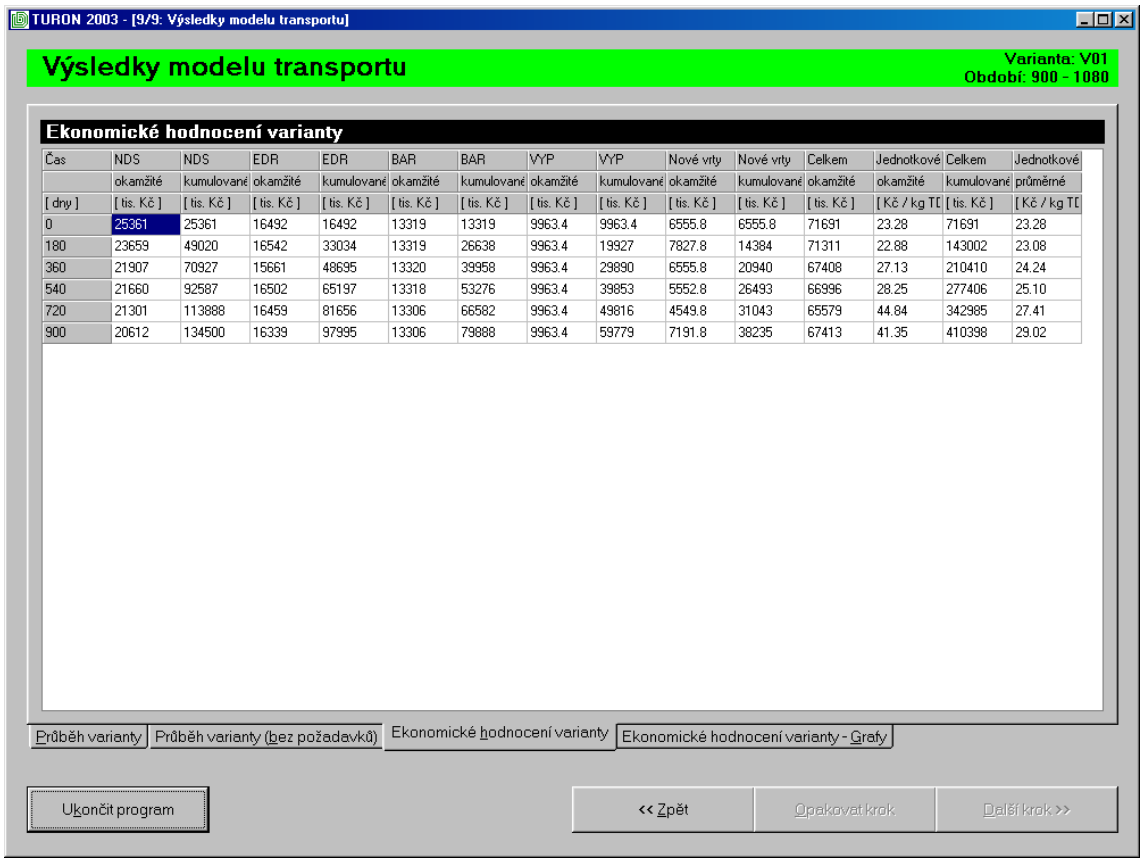
Čas	Vyvedené	Sum. vyv.	NDS	NDS	NDS	NDS	NDS	NDS	NDS	EDR	EDR	EDR	EDR	EDR	ED
	látky	látek	TDS skut.	m TDS	SO4 skut.	m SO4	NH4 skut.	m NH4	objem skut.	TDS skut.	m TDS	SO4 skut.	m SO4	NH4 skut.	m N
[dny]	[tun]	[tun]	[g / l]	[tun]	[g / l]	[tun]	[g / l]	[tun]	[tis. m3]	[g / l]	[tun]	[g / l]	[tun]	[g / l]	[tu
0	3079.2	3079.2	5.816	1507.7	4.310	1117.4	0.2663	69.03	259.2	1.271	329.5	0.9106	236.0	0.02251	5.8
180	3116.2	6195.4	4.997	1295.2	3.708	961.0	0.1921	49.80	259.2	1.799	466.3	1.257	325.8	0.1006	26.
360	2484.3	8679.7	3.773	978.1	2.841	736.3	0.1199	31.07	259.2	1.716	397.2	1.235	285.9	0.07905	18.
540	2371.4	11051	3.080	798.2	2.286	592.5	0.1169	30.31	259.2	1.337	346.5	0.8468	219.5	0.1340	34.
720	1462.5	12514	1.314	340.5	0.8487	220.0	0.1233	31.97	259.2	1.053	273.0	0.7426	192.5	0.04076	10.
900	1630.3	14144	1.894	491.0	1.380	357.7	0.08024	20.80	259.2	0.6155	159.5	0.4413	114.4	0.01365	3.5

Obr. 4.9.3 – obrazovka 9; tabulka výsledků jedné technologie – NDS (volba SKIP_ALL nebyla použita)



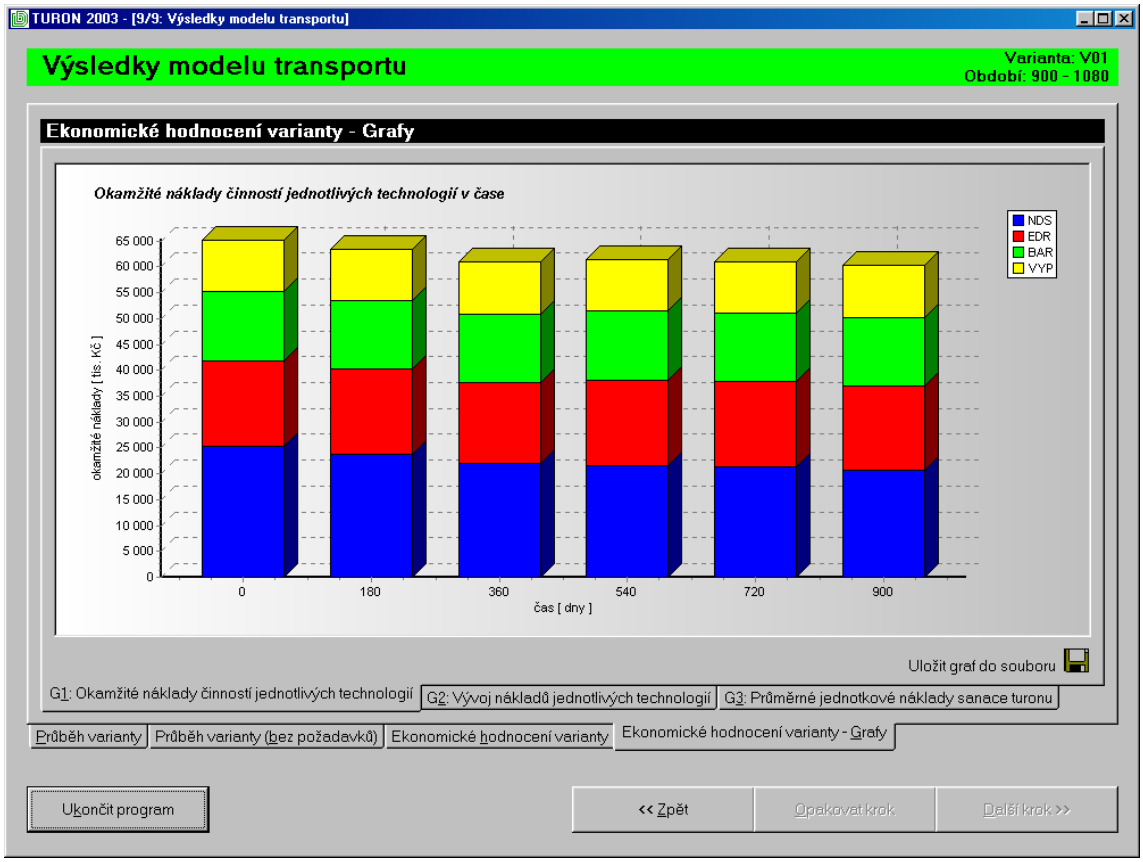
	Čerpaný objem	Konc. TDS	Hmotnost TDS	Konc. SO4	Hmotnost SO4	Konc. NH4	Hmotnost NH4
	[tis. m3]	[g / l]	[tun]	[g / l]	[tun]	[g / l]	[tun]
Skutečnost	259.2	5.843	1514.2	4.344	1125.9	0.2671	69.23
Požadavek	259.2	7.000				0.3000	
Rozdíl [%]	-0.01196	-16.53				-10.96	

Obr. 4.9.4 – obrazovka 9; tabulka ekonomické hodnocení varianty (byla použita volba SKIP_ALL)

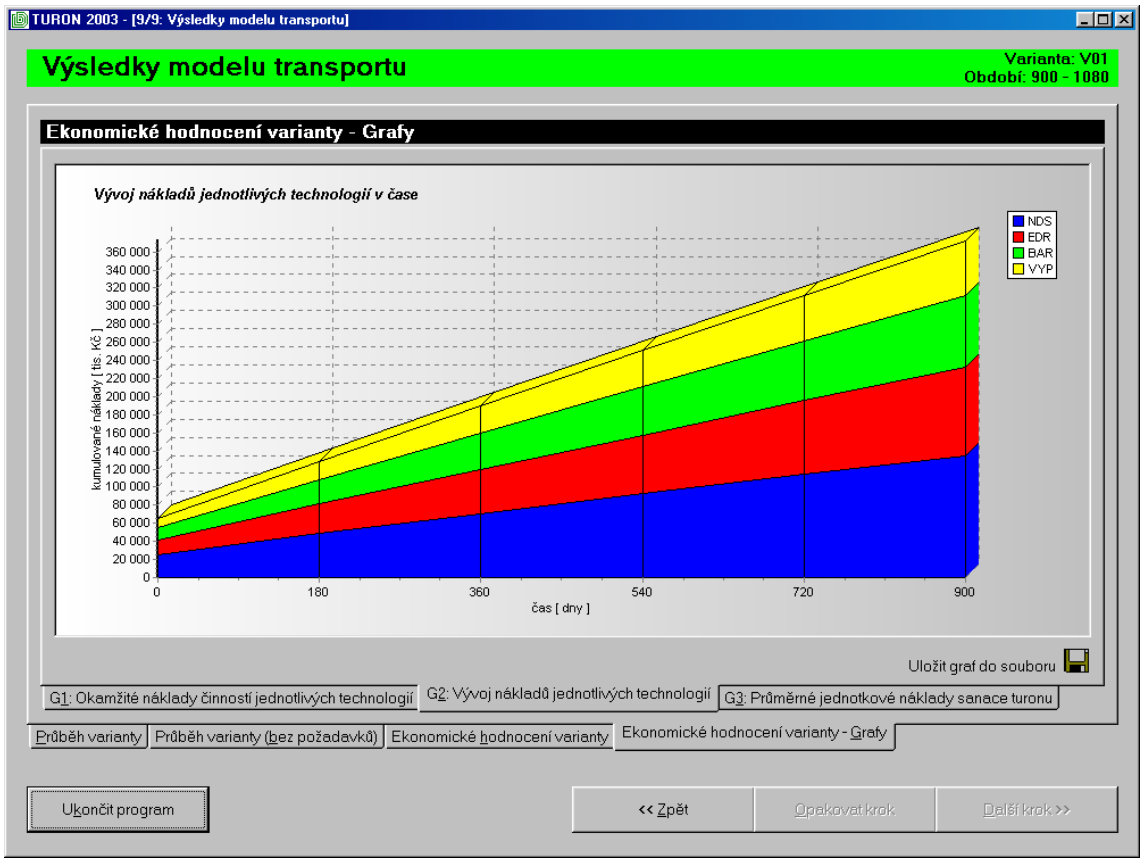


Čas	NDS	NDS	EDR	EDR	BAR	BAR	VYP	VYP	Nové vrty	Nové vrty	Celkem	Jednotkové	Celkem	Jednotkové
	okamžité	kumulované	okamžité	kumulované	okamžité	kumulované	okamžité	kumulované	okamžité	kumulované	okamžité	okamžité	kumulované	průměrné
[dny]	[tis. Kč]	[tis. Kč]	[tis. Kč]	[tis. Kč]	[tis. Kč]	[tis. Kč]	[tis. Kč]	[tis. Kč]	[tis. Kč]	[tis. Kč]	[tis. Kč]	[Kč / kg TI	[tis. Kč]	[Kč / kg TI
0	25361	25361	16492	16492	13319	13319	9963.4	9963.4	6555.8	6555.8	71691	23.28	71691	23.28
180	23659	49020	16542	33034	13319	26638	9963.4	19927	7827.8	14384	71311	22.88	143002	23.08
360	21907	70927	15661	48695	13320	39958	9963.4	29890	6555.8	20940	67408	27.13	210410	24.24
540	21660	92587	16502	65197	13318	53276	9963.4	39853	5552.8	26493	66996	28.25	277406	25.10
720	21301	113888	16459	81656	13306	66582	9963.4	49816	4549.8	31043	65579	44.84	342985	27.41
900	20612	134500	16339	97995	13306	79888	9963.4	59779	7191.8	38235	67413	41.35	410398	29.02

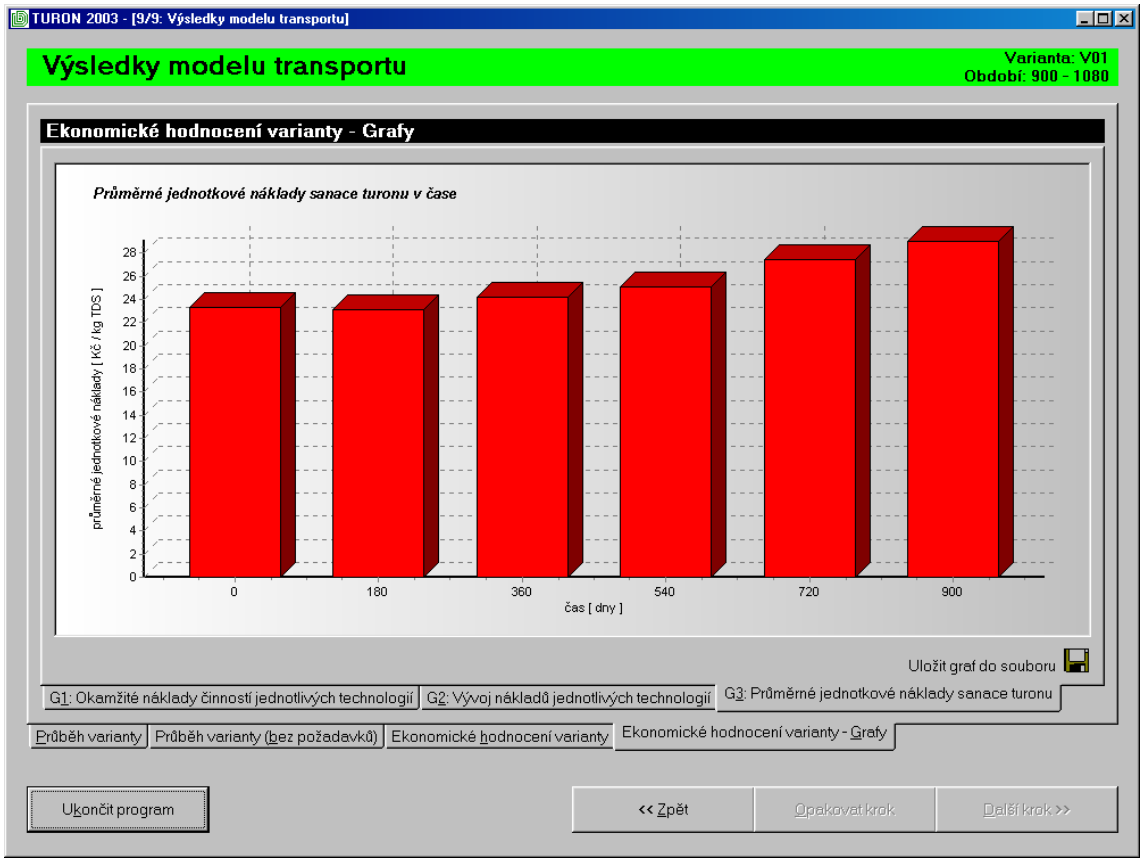
Obr. 4.9.5 – obrazovka 9; graf okamžité náklady činností jednotlivých technologií v čase



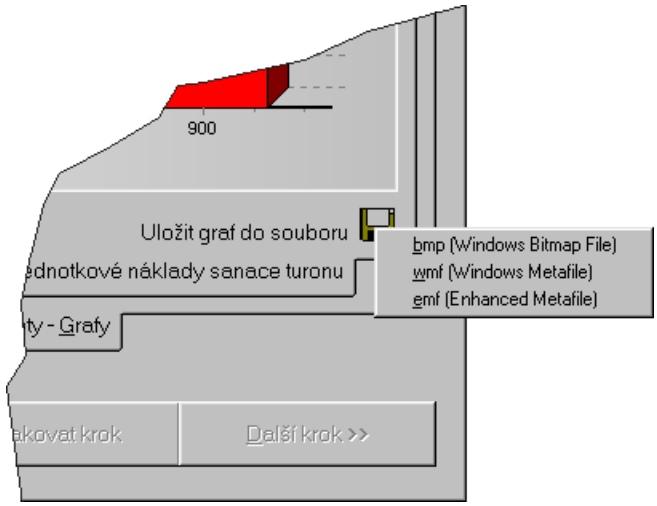
Obr. 4.9.6 – obrazovka 9; graf vývoj nákladů jednotlivých technologií v čase



Obr. 4.9.7 – obrazovka 9; graf průměrné jednotkové náklady sanace turonu v čase



Obr. 4.9.8 – obrazovka 9; popup menu na uložení grafu do souboru



Seznam vyobrazení v kapitole 4

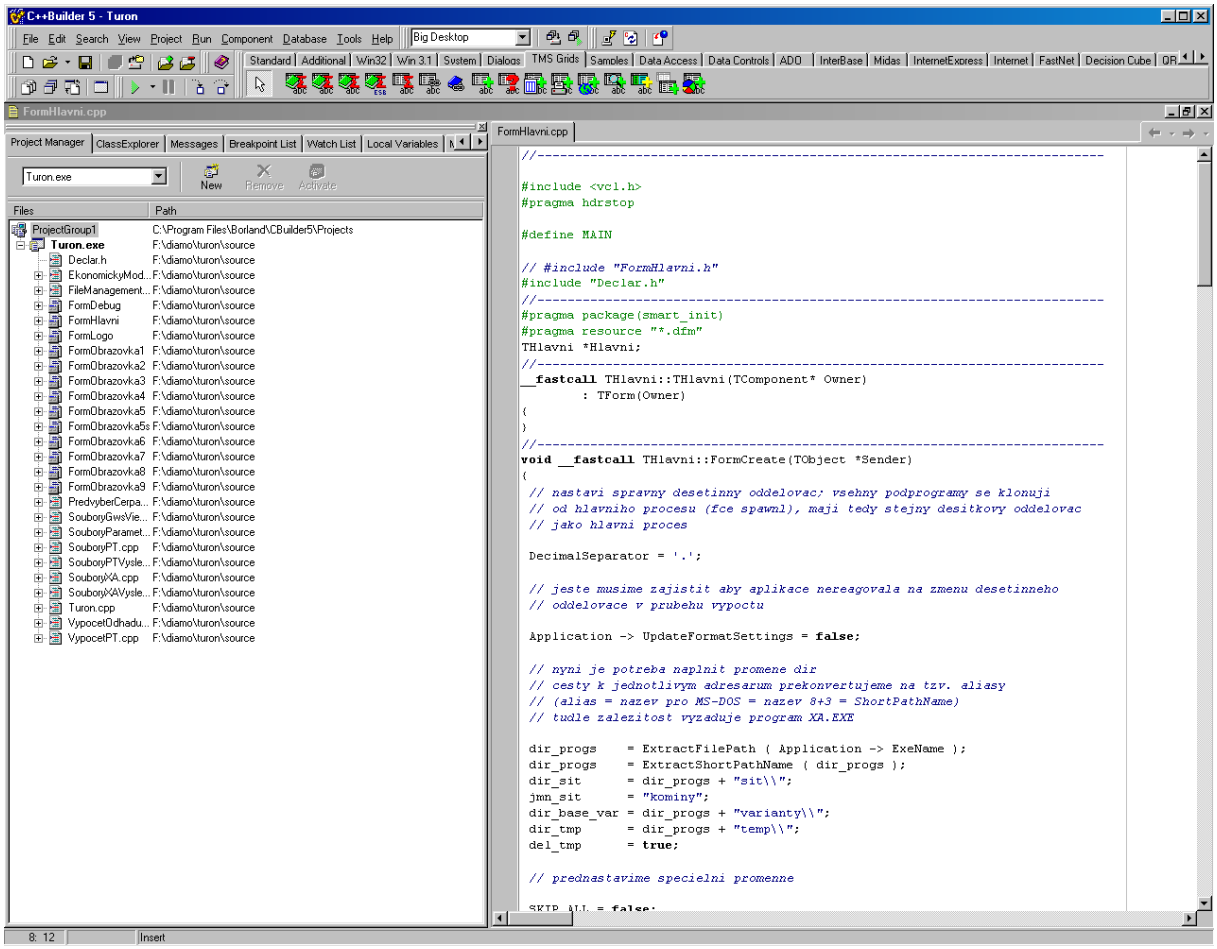
5 Změny provedené v systému v rámci ročníkového projektu

Systém pro optimalizaci sanace turonské oblasti byl vyvíjen již od roku 1999 ve státním podniku DIAMO. Kromě vývoje důležitých podprogramů (výpočet odhadů, modely proudění a transportu látek) bylo ve státním podniku vyvíjeno i grafické rozhraní optimalizačního systému. Toto rozhraní – systém obrazovek – je nyní implementováno v souboru *turon.exe* a je neustále pozměňováno podle aktuálních potřeb. V této kapitole se budeme věnovat výhradně vývoji tohoto grafického interface programu. Pokud se tedy bude kdekoliv v této kapitole hovořit o programu, myslí se tím výhradně program *turon.exe*

Program je v současné době vyvíjen ve vývojovém prostředí Borland C++Builder 5 firmy Inprise – http://www.borland.com – doplněném o některé komponenty (TADVStringGrid) firmy TMS Software – http://www.tmssoftware.com – tyto komponenty jsou k dispozici na druhém instalačním CD prostředí C++Builder 5 a na internetu. Na vývoji programu se v průběhu tří let vystřídala řada programátorů ze státního podniku DIAMO a Technické Univerzity v Liberci. Každý z nich pochopitelně vložil do zdrojových kódů programu část svých představ o programování (formátování textu, rozdělení programu do jednotlivých modulů atd.). Tyto drobné záležitosti však vedly k velké roztříštěnosti zdrojových textů a k jejich značné nepřehlednosti. Řada procedur a datových proměnných se v programu vyskytovala duplicitně, některé procedury byly naopak definovány nebo jen deklarovány a nebyly v programu vůbec použity. Dokonce se ve složce se zdrojovými texty programu vyskytovaly i soubory se starými zdrojovými texty, které již vůbec nebyly používány, a soubory, které s programem vůbec nesouvisely.

Součástí ročníkového projektu bylo provedení určitých zásadních systémových změn v tomto programu. Jak se však postupem času ukázalo, provedení těchto změn by bylo daleko snazší, kdyby měly zdrojové texty jednotnější formu. Součástí projektu se tak stalo i částečné přepracování těchto souborů.

Obr. 5.1 – ukázka vývojového prostředí C++Builder 5; nahoře je na popředí lišta s komponentami fy TMS Software – komponenty TMS Grids, zde je i komponenta TADVStringGrid. V levé části obrázku je vidět seznam souborů se zdrojovými texty programu



5.1 Drobné změny v systému TURON 2003

5.1.1 Správa souborů – mazání dočasných souborů

Vzhledem k tomu, že program pracuje s velkým množstvím souborů, a to jak celkově, tak v rámci jedné varianty, dokonce i na úrovni jednoho kroku varianty, bylo nutné zavést pevná pravidla, jak zacházet se soubory a to jak na úrovni uživatelského rozhraní, tak na úrovni zdrojových textů.

Zejména v průběhu výpočtu transportu látek program vytváří velké množství souborů, které po dokončení výpočtu již nejsou potřeba. Jedná se hlavně o tři soubory, které celkem zabírají cca 300 MB. Tyto soubory zůstávaly permanentně v dočasných adresářích systému. Nároky na volný diskový prostor se tím ve skutečnosti nijak nezvětšovaly, protože tyto soubory se neustále přepisovaly při dalších krocích výpočtu, ale to, že program okolo sebe nechává velké množství nejrůznějších nepotřebných souborů nepůsobí příliš elegantně.

Většina těchto dočasných souborů se implicitně umísťuje do adresáře *temp* (adresář pro dočasné soubory), který je standardně umístěn v kořenovém adresáři systému TURON 2003. Mazání obsahu tohoto adresáře bylo jaksi programově ošetřeno, nikoliv však dostatečně efektivně. Mazání souborů bylo prováděno standardním použitím interního příkazu *del* programu *command.com*, prostřednictvím volání funkce *system()*, která je v programovacím jazyce C++ standardně k dispozici a která volá systémový příkaz.

Prvořadou nevýhodou tohoto řešení bylo to, že při mazání obsahu tohoto adresáře byl příkaz *del* použit s některými parametry – *del /F /Q *.**, které nejsou běžně přístupné v různých operačních systémech Windows. Například parametr Q znamená, že se systém nebude dotazovat, zda se mají opravdu smazat soubory s příslušnou maskou. Tento parametr je však dostupný pouze v příkazových interpretech systémů Windows NT, 2000, XP, nikoliv však například v operačním systému Windows 98, na kterém shodou okolností probíhal vývoj programu. Soubory adresáře *temp* tedy vůbec nebyly mazány a program čas od času hlásil zdánlivě nelogické chyby. Při prvním pohledu se zdálo jako nejvhodnější nahradit tento příkaz příkazem *del ?**, který také smaže všechny soubory v daném adresáři, na nic se neptá a navíc funguje ve všech verzích příkazového interpretu *command.com*, odpovídajících verzím DOSu starším než *MS DOS 6.22*. V průběhu testování programu se však ukázalo i toto řešení jako nedostatečné, protože některé soubory v adresáři *temp* někdy přesto zůstávaly. Navíc pokud je v dočasném adresáři vytvořen podadresář (což by se ovšem nemělo stát), obě verze příkazu *del* s tím nic nezmohou a adresář tam zůstane, dokud ho nesmaže uživatel manuálně.

Druhou nevýhodou, která je spíš „vadou na kráse“ programu, bylo to, že při spuštění příkazu pomocí funkce *system()*, musí ve Windows na popředí vyskočit okno programu *command.com*, v kterém se příkaz provede, a toto okno se vzápětí zavře. Právě díky tomu, že se okno téměř okamžitě automaticky zavírá, nebylo možné kontrolovat, zda při provedení příkazu nedošlo k nějaké chybě. Také samotné zobrazení DOSovského okna nepůsobilo na uživatele příliš estetickým dojmem.

Navíc byl tento příkaz použit vždy přímo (nebyl zahrnut v nějaké složitější proceduře), tudíž v každé metodě příslušející události, při které se měl být obsah *tempu* smazán, bylo složité formátování příkazu implementováno znovu.

Nejlepším řešením tohoto problému se ukázalo vytvoření procedury, která je při různých událostech spouštěna. Procedura v dočasném adresáři nalezne všechny soubory a smaže je, pak nalezne všechny podadresáře, pokud zde nějaké jsou, a rekurzivně zavolá sama sebe na tyto podadresáře. Tato procedura ve skutečnosti nepotřebuje žádný parametr, neboť umístění adresáře *temp* je zapsáno v globálních datech programu. Mazání obsahu dočasného adresáře se také nyní provádí častěji. Dříve se mazal obsah tohoto adresáře jen v případě, že jsme po dokončení jednoho kroku výpočtu zahájili další krok výpočtu, popřípadě jsme si vyžádali opakování výpočtu tohoto kroku. Nyní probíhá mazání navíc při ukončení programu a preventivně i při jeho spuštění. Z bezpečnostních důvodů, a také pro potřeby ladění programu, lze ovšem mazání obsahu dočasného adresáře zabránit pomocí nastavení určitých proměnných v konfiguračním souboru *turon.ini*. Viz také přílohy a zdrojové texty.

Na závěr je třeba dodat, že kromě dočasných souborů, které se vytvářejí v adresáři temp, se v systému vytváří řada dočasných souborů i na různých jiných místech. Přesměrování takových souborů do dočasného adresáře je víceméně zbytečné. Všechny tyto soubory se mažou automaticky v momentě, kdy již nejsou potřeba. Příkladem takového souboru je *tur_odhad.log*. Jedná se o logovací soubor programu na výpočet odhadů, který se vytváří v adresáři sítě. Dalšími jsou například soubory hydrodynamiky, které se vytvářejí v adresáři varianty (soubory *00-00000.hdm*, *00-00000.hdn*).

5.1.2 Správa souborů – zálohování souborů s nastavením parametrů kroku varianty a s výsledky výpočtu

Zejména v průběhu ladění programu, kdy bylo potřeba načítat různé konfigurace dat a zároveň pracovat (pro jednoduchost) stále se stejnou variantou, se ukázalo jako vhodné mít možnost zálohovat soubory s nastavením jednotlivých kroků, eventuelně i soubory s výsledky výpočtu. Zálohování těchto nejdůležitějších souborů je také vhodné v případě, že se provádí přepočítání již existující varianty. Soubory se zálohují automaticky, ovšem na žádost uživatele. Automatickým zálohováním na žádost uživatele se myslí to, že program sám přiřadí zálohovanému souboru nové jméno podle určité konvence a umístí ho do nějakého adresáře (přímo do adresáře varianty). Uživatel má ale možnost volby, zda zálohování povolí nebo ne.

K zálohování obou souborů (s parametry i s výsledky) dochází na první obrazovce, kde si vybíráme, kterou variantu a které období chceme počítat. Pokud program zjistí, že pro dané období vybrané varianty tyto soubory již existují, automaticky zobrazí volby pro zálohování těchto souborů (viz obrázek 4.1.1 v předchozí kapitole).

Příklady původních a záložních souborů (nacházejí se v adresáři varianty):

par0.ini	*par0.ini_2002-12-15_16-10-40_backup.ini*
par180.ini	*par180.ini_2002-12-17_00-59-13_backup.ini*
vysledky180.xls	*vysledky180.xls_2002-12-17_01-34-09_backup.xls*

Syntaxe názvů přidělovaných záložním souborům:

<jméno a přípona pův. soub.>_<datum (rrrr-mm-dd)>_<čas (hh-mm-ss)>_backup.<přípona pův. soub.>

Díky této syntaxi pojmenovávání záložních kopií je možné přesně určit, kdy vznikly. Hlavní výhodou je však to, že se záložní soubory navzájem nepřepisují a je tedy možné jich vytvořit, kolik je potřeba, a zároveň je lze celkem snadno rozlišit. Na první obrazovce systému je navíc jakýsi manager variant, který umožňuje smazání všech existujících záložních souborů z adresáře varianty (viz obrázek 4.1.2).

Zálohování ostatních souborů varianty se jeví jako zbytečné, a to nejenom kvůli tomu, že by tyto záložní soubory zabíraly zbytečné místo, ale zejména proto, že zálohováním souborů s parametry máme k dispozici všechny potřebné informace o daném kroku a můžeme ho bez problémů zopakovat. Pozor! Pro absolutně přesné opakování výpočtu je však nutné mít také stejné soubory *mCAS.ts3* a *mCAS_slow.ts3* (tyto soubory se však nezálohují). Vždy ale vycházejí z prvopočátečního stavu na začátku celé varianty (*m0.ts3* a *m0_slow.ts3*) a jednotlivé *ini* soubory s nastavením všech kroků (mezi *0* a *CAS*) již mohou být zálohovány.

Na závěr je potřeba poznamenat, že zálohování souborů by mělo sloužit pouze jako pomocný mechanismus. Pro vypočítávání diametrálně odlišných variant slouží jejich identifikace a možnost jejich rozlišení pomocí jmen (respektive adresářů, v kterých se soubory variant nacházejí).

5.1.3 Hierarchie adresářů systému a hlavní konfigurační soubor

Původní verze programu mohla mít téměř libovolné umístění jednotlivých adresářů. Byl zde hlavní adresář (v programu nazývaný *dir_base*), v kterém se nacházely adresáře jednotlivých variant, adresář obsahující hlavní program, podprogramy a dynamické knihovny systému (*dir_progs*), adresář se soubory sítě (*dir_sit*) a adresář pro dočasné soubory (*dir_tmp*). Nastavení cest k těmto jednotlivým adresářům bylo při spuštění programu načítáno ze souboru *cesty.ini*, který ovšem musel ležet ve stejném adresáři jako hlavní program (v adresáři *dir_progs*). Soubor *cesty.ini* tak vlastně obsahoval i cestu k hlavnímu programu systému, který ji z něj načítal.

Výhodou tohoto řešení je to, že jednotlivé adresáře je možno umístit kamkoliv, takže pokud jsme například měli systém nakopírovaný na disku C, kde již nebyl dostatek volného místa, mohli jsme adresáře variant a dočasný adresář, které zabírají nejvíce místa, přesunout na jiný disk. Dále je toto řešení vhodné proto, že je možné snadno měnit síť, na které probíhá výpočet. Naopak nevýhodou bylo právě poněkud nepřehledné umístění „cesty sama k sobě“ v ini souboru a dále poněkud krkolomné načítání těchto cest ze souboru. Cesty byly v souboru uloženy celé (absolutní cesty, tedy včetně označení písmen disku), a tak při přenosu celého systému na jiný počítač, nebo i při přesouvání systému v rámci jednoho počítače z původního adresáře do jiného adresáře bylo nutné manuálně aktualizovat cesty k jednotlivým složkám v souboru *cesty.ini*.

V současné době je hierarchie adresářů víceméně pevná (viz příloha P1), ovšem do hlavního adresáře (adresář s programem *turon.exe*) lze umístit konfigurační soubor *turon.ini*, ve kterém lze některá tato pevná nastavení změnit. Lze nastavit jiný adresář sítě (a jméno sítě), adresář s variantami a dočasný adresář. Navíc lze v tomto souboru nastavit, zda má systém obsah dočasného adresáře mazat nebo nikoliv. Všechny cesty již mohou být v konfiguračním souboru umístěny relativně vzhledem k programu *turon.exe*, mohou být uvedeny s dlouhým nebo i se zkráceným jménem a navíc mohou, ale nemusí být zakončeny zpětným lomítkem (backshlash „\“). Dále je v souboru několik nastavení, s jejichž pomocí lze program částečně optimalizovat (například zobrazení debug obrazovky; viz obrázek 4.1.4).

Součástí adresářové struktury se, jak je vidět v příloze, stala i složka se zdrojovými texty programu. Ty byly dříve umístěny mimo hlavní adresář systému.

5.1.4 Problém sdílených složek

Další problém, který se vyskytoval v systému, nesouvisí ani tak s programováním, jako spíše s nastavením adresářů programu, a proto ho uvádím v souvislosti s hierarchií složek. Program pro výpočet proudění (*gen_flow.exe*) generuje soustavu rovnic, kterou je potřeba vyřešit, aby bylo možné navázat výpočtem transportu látek. Výpočet soustavy realizuje program *ni1.exe*, který je volán dávkovým souborem *solver.bat*. Program pro výpočet proudění je bohužel udělán tak nešťastně, že oba soubory (řešič i dávkový soubor) volá zcela obecně a ne z konkrétního adresáře.

Aby bylo možné volat tyto soubory obecně, je potřeba, aby cesta k těmto souborům (standardně jsou umístěny mezi programy systému) byla zapsána v systémové proměnné *PATH*, jejíž obsah se specifikuje v souboru *autoexec.bat*, nebo aby tyto soubory ležely v takovém adresáři, který už proměnná *PATH* obsahuje (nejlépe tedy některý sdílený adresář *Windows*). Původně byl tento problém řešen prvním způsobem, což při manipulaci s programem znamenalo provádět neustále změny v souboru *autoexec.bat*. Při přenesení systému na jiný počítač se tento problém opakoval. Pro snadnou manipulaci s programem v rámci jednoho počítače se tedy ukázalo jako výhodnější umístit tyto dva soubory do některé složky, která již je sdílená v proměnné *PATH*. Jako nejvhodnější byla vybrána složka pro příkazy systému Windows (*C:\Windows\Command*).

Pokud jsou soubory umístěny zde, tak se, jak již bylo řečeno, značně ulehčí manipulace s programem (lze ho libovolně přesouvat jako celek z adresáře do adresáře bez jakýchkoliv změn v jakýchkoliv souborech). Současně s tím se vyřeší i problém s omezenou kapacitou proměnné *PATH*. Tato systémová proměnná totiž může obsahovat jen omezený počet znaků.

Složka programů systému TURON 2003 navíc ještě obsahuje řadu dynamických knihoven (soubory **.dll* a **.bpl*). Tyto soubory jsou vyžadovány většinou programů (podprogramů) systému. Vzhledem k tomu, že tyto knihovny leží ve stejné složce jako programy, neměl by vzniknout žádný problém, nicméně všechny dynamické knihovny jsou pro jistotu také umístěny do systémové složky systému Windows (*C:\Windows\System*).

Umístění jednotlivých souborů do složek Windows provádí automaticky instalátor programu. Popis umístění jednotlivých souborů je také v souboru *readme.pdf*, který je součástí systému. Při přenosu systému na jiný počítač tak můžeme použít buď instalátor, nebo příslušné soubory do systémových složek nakopírovat manuálně. Pokud však například nemůžeme zapisovat do těchto systémových složek, alternativní řešení pomocí modifikace souboru *autoexec.bat* je stále nasnadě.

5.1.5 Problém desetinné čárky / tečky – program XA

Komerční software XA, který reprezentuje lineární model v systému, striktně vyžaduje použití desetinné tečky při psaní reálných čísel. Dalším nastavením, které bylo potřeba dělat při přenesení systému na jiný počítač, tak byla změna desetinného oddělovače z tečky na čárku (typicky pouze v české mutaci systému Windows). Nastavení desetinné tečky (respektive jednotné nastavení desetinného oddělovače) vyžadoval i samotný program *turon.exe*, neboť *ini* soubory obsahující parametry nastavení kroku varianty, které program generuje, obsahují řadu desetinných čísel. Bylo tedy potřeba, aby při čtení a zápisu souboru s parametry program pracoval minimálně vždy se stejným oddělovačem.

Tento problém poměrně dlouho odolával úspěšnému vyřešení. Ze začátku, jak bylo řečeno výše, bylo nutné manuálně měnit desetinný oddělovač před prvním spuštěním programu v ovládacích panelech. Prvním pokusem o řešení programu se tak stal pomocný podprogram, který se spustil, pokud bylo při spuštění programu

turon.exe detekováno nastavení desetinného oddělovače na čárku. Tento pomocný modul automaticky změnil oddělovač na tečku zápisem do registru a zasláním systémové zprávy *WM_SETTINGCHANGE* o tom informoval běžící programy, poté navrátil řízení zpět programu *turon.exe*.

Toto řešení však stále nebylo dost elegantní, neboť na počítači se mohou vyskytnout různé programy, které budou vyžadovat různě nastavený desetinný oddělovač. Na druhou alternativní cestu k vyřešení tohoto problému ukazoval fakt, že každý spuštěný program (proces) si zkopíruje obsahy všech systémových proměnných do svých vlastních globálních dat, která se pochopitelně dají modifikovat. Téměř každý program vytvářený v prostředí C++Builder / Delphi má k dispozici globální proměnnou *DecimalSeparator*, do které se právě po spuštění programu zkopíruje nastavení desetinného oddělovače z Windows. Obsah této proměnné však lze i natvrdo změnit přímo v programu. Tím se vyřešil problém s desetinným oddělovačem v programu *turon.exe*. Program XA však stále nepracoval korektně.

Program XA byl z hlavního programu spouštěn pomocí funkce *system()*, která je k dispozici v standardních knihovnách C++. Tato funkce provede volání příkazového interpretu (*command.com*), který pak provede požadovaný příkaz, jenž byl předán jako parametr funkce *system()*. Za systémový příkaz lze pochopitelně považovat i spuštění programu (může být i s parametrem). Hlavní program při spuštění této funkce čeká, dokud se funkce neukončí, tudíž se synchronizací chodu obou programů nebyl problém. Nahrazením funkce *system()* funkcí *spawnl()*, která slouží k vytváření (spouštění) podprocesů, se problém vyřešil definitivně. Tato funkce umí vytvářet podprocesy v několika režimech, jeden z nich je i *P_WAIT*. Spuštění podprocesu v tomto režimu donutí rodičovský proces čekat na jeho návratovou hodnotu, čímž je opět zaručena dokonalá synchronizace obou programů. Navíc oddělení podprocesu se provádí jakýmsi klonováním, při kterém dojde ke zkopírování všech globálních proměnných se systémovými údaji z rodičovského procesu na proces dceřinný. Podproces tak navíc zdědí i „správné“ nastavení desetinného oddělovače.

Posledním nastavením, které bylo potřeba provést v souvislosti s tímto problémem, bylo zabránění ve změně desetinného oddělovače „zvenku“. Kdyby totiž jiný proces v době běhu systému TURON 2003 modifikoval desetinný oddělovač Windows, systém by na to reagoval změnou svého vlastního desetinného oddělovače; (pokud nějaký proces změní desetinný oddělovač, informuje o tom ostatní prostřednictvím systémové zprávy *WM_WININICHANGE* respektive *WM_SETTINGCHANGE*, ostatní procesy ji standardně akceptují a přizpůsobí se). Změnou nastavení jisté globální proměnné (*UpdateFormatSettings = false*) dosáhneme toho, že aplikace tyto zprávy ignoruje.

Veškerá tato nastavení byla testována pouze na operačním systému Windows 98. Nelze tudíž zaručit, že budou korektně fungovat i v systémech řady NT (změna desetinného oddělovače na tečku přímo v ovládacích panelech pochopitelně zůstává stále možná).

5.1.6 Další problémy, které se mohou v systému vyskytnout a jejich alternativní řešení – program XA

Program XA je poněkud starší nástroj, který není určen pro práci pod Windows, a s tím souvisí i řada dalších problémů, které se u něj mohou vyskytovat. Ve starších verzích systémů TURON se nacházel (kromě inicializačního soubor *cesty.ini*) soubor *mpt.clp*, který sloužil jako inicializační soubor programu XA. Tento program používá jako vstup a výstup soubory s přesně definovanou, relativně složitou a značně nepřehlednou strukturou. Umístění a názvy souboru se vstupy a souboru určeného pro výstup navíc program XA načítá právě z inicializačního souboru *mpt.clp*, o jehož umístění ho informuje parametr, s kterým je spouštěn. Tento soubor byl tedy dříve pevnou součástí systému, a tak při manipulaci se systémem se opět musela měnit veškerá nastavení týkající se cest a umístnění vstupů a výstupů uložená v tomto souboru, jako v případě souboru *cesty.ini*. Nyní je tento inicializační soubor programu XA generován automaticky systémem TURON 2003.

Pokud systém instalujeme do počítače pomocí standardního instalačního programu, systém se snaží implicitně nainstalovat do adresáře „*C:\Program Files\Diamo a. s\TURON 2003*“. Tato cesta však obsahuje i tzv. dlouhé názvy souborů (neodpovídají standardu 8+3 tak, jak byl používán ve starých systémem DOS – maximálně osm písmen pro jméno souboru a tři písmena na příponu souboru, navíc nesmí jméno ani přípona obsahovat tzv. nepovolené znaky jako jsou například mezera, tečka, čárka atd.). Operační systémy Windows 9x a NT používají kromě tzv. dlouhých názvů i tzv. aliasy. Tyto aliasy jsou tvořeny takovým způsobem, že právě odpovídají standardu 8+3. Ke každému dlouhému názvu je na disku jednoznačně přiřazen jeden alias. Pro specifikaci vstupů a výstupů programu XA je tedy nutné použít tyto aliasy. Přiřazení aliasů ke dlouhému jménu adresáře, souboru, nebo celé cesty lze provést například pomocí API funkce *GetShortPathName()* nebo pomocí funkce *ExtractShortPathName()*, která je standardně k dispozici ve vývojovém prostředí C++Builder.

Některé souborové systémy nemusí podporovat ukládání dvojích jmen (dlouhých názvů i aliasů). U některých se dá vytváření aliasů zapnout respektive vypnout, a implicitně mohou být vypnuty. Některé souborové systémy pro některé dlouhé názvy aliasy neukládají (většinou u těch které se liší od standardu 8+3 jen tím, že rozlišují velikost písmen), nebo se dá tato vlastnost zapnout respektive vypnout. Navíc kromě omezení názvů souborů a adresářů na formát 8+3, je ve starých operačních systémech DOS ještě požadavek další, týkající se celé cesty. Cesta (rozuměj posloupnost všech vnořených podadresářů od kořenového adresáře, včetně všech oddělovacích znaků „\“ a včetně názvu souboru) totiž nesmí být delší než osmdesát znaků – to odpovídá šířce obrazovky v textovém režimu v DOSu (80x25 znaků) a u některých starých souborových systémů například nemohlo být více než osm vnořených podadresářů. Moderní diskový souborový systém tak například nemusí vytvářet aliasy souborům a adresářům, které jsou „hlouběji“ než osmdesát znaků. Jak se v těchto různých případech budou chovat funkce pro hledání aliasů a jak se zachová program XA (a celý systém TURON 2003), není odzkoušeno.

Nicméně pokud bude program XA hlásit nějaké problémy, je nejvhodnější natvrdo nastavit v ovládacích panelech desetinný oddělovač na tečku (viz předchozí kapitola). Dále je potřeba zkontrolovat soubor *mpt.clp*, který se v době, kdy je spuštěn program XA, nachází v dočasném adresáři *temp*. Tento soubor by měl odkazovat na další dva soubory se vstupy (*turon.mps*) a výstupy (*turon.xa*) programu XA. Tyto soubory je také dobré v případě výskytu chyby zkontrolovat; oba se nacházejí také v adresáři *temp*. (Chybná činnost některých funkcí je možná i například díky rozdílné interpretaci některých systémových volání v různých typech systémů Windows - vše bylo testováno pouze na Windows 98.)

5.1.7 Zobrazování odhadů a vybraných vrtů v GwsView

Zobrazovací systém GwsView je vyvíjen samostatně spolu s dalšími programy ve státním podniku DIAMO. Jedná se o sadu programů týkajících se zejména generování, práce a vizualizace sítí konečných prvků, které mají takovou strukturu, jak je popsaná v přílohách (síť konečných prvků používaná v tomto systému je také generovaná promocí systémů Gws – konkrétně GwsSit).

Program GwsView byl součástí systému TURON již od počátku, ale například zobrazování odhadů bylo nutno provádět ručně. To znamenalo spustit vizualizační program z nějakého souborového manažeru, načíst příslušnou síť (například *komíny_1v*) a otevřít soubor s odhady. Vybrané vrty nešlo zobrazovat vůbec, protože seznam vybraných vrtů se nikam neukládal (vyjma okrajové podmínky k danému časovému období). Prvním krokem tedy bylo vygenerování souboru s vybranými vrty (např. soubor *svyvrt0.txt* pro období 0 – 180 dní; struktura souboru je popsaná v přílohách P2; tento formát souboru je podporován programem GwsView). Soubor se seznamem vybraných vrtů se generuje v okamžiku, když je programem XA nalezeno optimálního řešení. Samo matematické nalezení optimálního řešení totiž fakticky odpovídá úspěšnému vybrání množiny čerpacích vrtů (viz také kapitoly 4.6 a 4.7). Seznam těchto vrtů pak lze získat na základě výstupu program XA.

Mechanismus zobrazování jak odhadů, tak i vybraných vrtů, se pak redukuje pouze na spuštění programu GwsView s příslušnými parametry. Tyto parametry jsou textové soubory se strukturou standardních *ini* souborů Windows, mají však přípony **.stv* a **.zpv*. Sestavení takových souborů programově tedy není nijak složité. Spouštění programu GwsView se provádí opět pomocí funkce *spawnl()*, jako například při spouštění programu XA. Vzhledem k tomu, že vlastní systém (hlavní program) nemusí čekat na ukončení činnosti grafického postprocesoru a neočekává od něj žádná návratová data, můžeme podproces spustit v režimu *P_NOWAIT* (na rozdíl například od programu XA, který se spouští režimu *P_WAIT*). Přesná programová realizace spouštění postprocesoru a generování **.stv* a **.zpv* souborů viz zdrojové texty.

5.2 Zásadní změny v systému TURON 2003

5.2.1 Přidání technologie BAR

Původně byly v programu zakomponovány pouze tři technologie – neutralizační stanice (NDS), membránové technologie (EDR) a vypouštění roztoků do řeky (VYP). V průběhu času se však ukázalo, že například provoz membránové technologie je příliš nákladný a vzhledem ke koncentracím roztoků nacházejících se v turonské zvodni neekonomický. Membránové technologie tak jsou v současné době z dlouhodobého scénáře sanace turonu vypuštěny – nepoužívají se (používají se pouze pro „cenomanské“ roztoky, které mají vyšší koncentrace). Nicméně technologie je v optimalizačním systému stále k dispozici pro případ, že by byla potřeba, respektive pro případ, že by například klesly náklady na provoz membránových technologií. Přesnější popis všech používaných technologií je v kapitole týkající se popisu systému obrazovek – konkrétně jsou technologie popsány v odstavci 4.3.

Naopak se ukázalo vhodné začlenit do systému technologii novou, tak zvanou hydraulickou bariéru (BAR). Bariéra uměle udržuje zvýšenou hladinu podzemní vody v oblasti mezi vyluhovacími poli a mezi blízkým hlubinným dolem (ten leží mimo oblast vyluhovacích polí). Původně bránila úniku roztoků do šachty, dnes již jen vyrovnává hladinu podzemní vody v okolí dolu, protože ten se zatápí. Původně se do bariéry vtláčely kyselé důlní vody, které byly daleko silnější než vody turonské, v současnosti tak vlastně dochází k čištění této oblasti. Zároveň se hydraulická bariéra stává technologií pro zpracování slabých a středně slabých turonských roztoků.

Vzhledem k tomu, že co se týká způsobu zpracování roztoků, je tato technologie velmi podobná technologii vypouštění (nic se nečistí), byla původně hydraulická bariéra simulovaná právě pomocí technologie vypouštění. Zásadní nevýhodou tohoto řešení je pochopitelně to, že nelze v programu současně provozovat obě technologie BAR i VYP. Bylo tedy nutné začlenit tuto technologii do systému. Celé předělání programu bylo rozděleno na několik částí podle toho, jak si to vyžádaly okolnosti.

První fází přidání technologie bylo její zakomponování do grafického rozhraní programu. Tento krok byl zvolen jako první proto, že se jednalo o krok nejjednodušší. Současně s tím bylo nutné do programu přidat veškeré globální proměnné (přidání a předělání potřebných struktur, přidání potřebných funkcí a metod atd.).

Druhou fází pak bylo propojení všech nových vizuálních komponent s globálními proměnnými programu a implementace nově vzniklých funkcí a metod. Veškeré tyto činnosti byly v podstatě prováděny duplikováním již existujících programových celků souvisejících s technologií VYP. Největším problémem se jevila modifikace podmínek lineárního modelu, nicméně i tento problém byl vyřešen „okopírováním“ podmínek technologie VYP.

Posledním nedostatkem, který se po přidání technologie v programu vyskytl, je problém ryze programátorský. Na všech obrazovkách jsou jednotlivé technologie řazeny jaksi „podle důležitosti“ a jejich pořadí je následující: NDS – EDR – BAR – VYP. V programu jsou jednotlivé technologie seřazeny do polí a manipuluje se s nimi pomocí indexů v tomto poli. Vzhledem k tomu, že původně program obsahoval pouze tři technologie, je zde zavedeno následující indexování: i(NDS) = 0, i(EDR) = 1, i(VYP) = 2. Přidání technologie BAR jako poslední způsobilo to, že má pochopitelně index nejvyšší i(BAR) = 3. Posloupnost technologií v programu je tedy jiná: NDS – EDR – VYP – BAR. Tento nedostatek se však snad brzy podaří vyřešit.

5.2.2 Respektování existujících vrtů a omezení počtu nových vrtů

Problém respektování existujících vrtů je velmi podstatný a je potřeba ho řešit nejen v turonském systému, ale i v systému cenomanském. Veškeré výpočty se v programu realizují na síti konečných prvků, tato síť je tvořena multielementy, jak již bylo řečeno, a každý multielement může reprezentovat maximálně právě jeden vrt. Optimalizační program si při hledání umístění čerpacích vrtů vybírá podle toho, kde je nejvhodnější koncentrace kontaminantů. Čerpáním z takového místa po dobu jednoho půlroku se však poměry v podzemí změní – koncentrace se sníží. Podobná koncentrace jako byla v místě tohoto čerpacího vrtu byla nejspíš i v jeho nejbližším okolí, tedy i v okolních multielementech. Optimalizační program si pak tedy při dalším kroku výpočtu nejspíše vybere nějaký multielement v nejbližším okolí původního místa čerpání. To proto, že v tomto místě koncentrace prudce poklesla, zatímco v jeho okolí se změnila méně, a ve vzdálenějším okolí téměř vůbec. Doba sanace je obecně stanovena na několik desítek let (cca 20 – 30 let). Výpočet jedné varianty tedy čítá například 50 kroků. Ukazuje se, že během tolika kroků výpočtu optimalizační model stihne vybrat téměř všechny možné multielementy za čerpací vrty, což pochopitelně není realizovatelné. Síť konečných prvků obsahuje cca 6000

multielementů, za potenciální vrty lze ovšem považovat pouze ty, které se kryjí s některým vyluhovacím polem. Přidružení multielementů k vyluhovacím polím je definováno v souboru *lokal.elm*, který je v adresáři sítě. Tento soubor obsahuje záznamy typu: *číslo multielementu – jméno vyluhovacího pole* (viz také přílohy). Tímto omezením je počet potenciálních čerpacích vrtů stažen na 3821. Vytvoření takového množství vrtů je pochopitelně nesmyslné. Je to nejen příliš drahé, ale i celkově nepraktické – například vrty s velkou vydatností (250 l/min) nemohou být moc blízko vedle sebe, protože by si příliš „konkurovaly".

Na vyluhovacích polích je již jakási stávající síť vrtů, které mohou být použity k čerpání. Zde tedy není většinou potřeba nové vrty vytvářet. Pochopitelně jsou tu i oblasti, kde to nutné je. Pokud ovšem někde děláme nové vrty, chtěli bychom je pochopitelně využívat po delší období a ne po půl roce dělat další nové vrty jen o několik metrů vedle. Optimalizační model tedy musí při výběru vrtu zohlednit existující vrtnou síť. Pokud si model vynutí vytvoření nových vrtů, měl by jich být jen omezený počet a program by si je měl zapamatovat a přidat je ke stávající vrtné síti, aby je mohl respektovat v dalším kroku výpočtu.

Prvním pokusem o řešení tohoto problému bylo omezení počtu multielementů, které jsou přiřazeny k jednotlivým vyluhovacím polím. To lze jednoduše provést zredukováním počtu záznamů v souboru *lokal.elm*. Toto řešení bylo zatím pokusně realizováno pouze v cenomanském modelu. Řešení je to sice částečně funkční a použitelné, nikoliv však dostatečně elegantní. Nejlepší řešení tohoto problému je přímé omezení lineárního modelu vytvořením dalších podmínek na výběr vrtů.

Vlastní implementace respektování existujících vrtů a omezení počtu nových vrtů byla rozdělena do několika oddělených logických kroků. Prvním z nich bylo vytvoření mechanizmu zapamatování existujících vrtů. Soubor se seznamem existujících vrtů (*sexvrtCAS.txt*) je potřeba mít jako jakousi počáteční podmínku pro každý krok každé varianty, navíc pro první krok libovolné varianty musí být tento seznam stejný (pouze seznam vrtů existujících v čase 0). Soubory *sexvrtCAS.txt* se tedy chovají obdobně jako soubory *mCAS.ts3* a *mCAS_slow.ts3*. Soubor se seznamem existujících vrtů je tedy vždy po nalezení optimálního řešení aktualizován o nově vybrané vrty a uložen pod novým jménem (s novou hodnotou *CAS* v názvu souboru). Seznam existujících vrtů musí obsahovat všechny důležité informace o vrtu (takové parametry vrtu, které již nelze ovlivnit, pokud vrt existuje). Tyto parametry jsou: číslo vrtu respektive multielementu (tedy vlastně jeho horizontální umístění), otevření vrtu (v kterých vrstvách je otevřen, tedy jeho vertikální lokalizace) a čerpací výkon vrtu (25, 125 nebo 250 l/min).

Druhým krokem bylo respektování existujících vrtů. Když už měl model k dispozici seznam existujících vrtů, mohl tyto vrty vybírat přednostně. K tomu slouží na šesté obrazovce v oddílu „Volba účelové funkce" položka „minimalizace počtu nových vrtů" (viz též obrázek 4.6.1). Účelová funkce je zde funkce, kterou (v našem případě) maximalizuje lineární model (viz také kapitola 2, odstavec týkající se lineárního programování). Při sestavení souboru se vstupy programu XA vlastně jen sestavíme seznam vrtů, které prošly předvýběrem, s potřebnými parametry. Důležitým parametrem každého vrtu je právě hodnota, kterou přispívá do účelové funkce, je-li vybrán. Existujícím vrtům (tedy takovým, které jsou v souboru *sexvrtCAS.txt*) přiřadíme hodnotu 1, ostatním vrtům přiřadíme hodnotu –1. Lineární model pak při své snaze účelovou funkci maximalizovat, preferuje existující vrty před vrty neexistujícími.

Posledním krokem bylo již omezení počtu nových vrtů. K tomuto účelu jsou na obrazovce šest (viz obrázek 4.6.1 – oddíl „Omezení pro počet nových vrtů") zřízeny položky, kde lze stanovit maximální počet nových vrtů pro jednotlivé čerpací výkony vrtů. Tyto hodnoty jsou pak jednoduše vloženy do lineárního modelu jako další omezující podmínky. Například vybráním nového vrtu s výkonem 250 l/min se inkrementuje příslušná proměnná, která nesmí překročit tyto podmínky. Lineární model tak může vybrat jen omezený počet nových vrtů, jinak není schopen nalézt optimální řešení.

V souvislosti s omezením počtu nových vrtů se objevil ještě jeden problém. Lineární model normálně nepracuje celočíselně tzn., že může například vybrat vrt jen z padesáti procent. Problém však vzniknul v momentě, když program ve snaze dostát podmínce na počet nových vrtů, vybral ve skutečnosti například desetkrát více nových vrtů ovšem s hodnotou 10%. Z tohoto důvodu bylo potřeba, aby měl uživatel možnost vyžádat si celočíselné řešení, pokud to bude nutné (viz obrázek 4.6.1 – oddíl „Celočíselné řešení", viz také kapitola 4.6). Toto další omezení se implementuje zadáním jiného typu podmínky do vstupních dat lineárního modelu. Druhotným problémem je zde ovšem to, že nalezení celočíselného řešení může trvat velmi dlouho a pro tzv. „úplné celočíselné řešení", optimální řešení vůbec nemusí existovat (z čerpacích výkonů vrtů – 25, 125 a 250 l/min nemusí jít celočíselně splnit náš požadavek nátoku na technologii – například 1.01 m^3/min; takto speciální požadavky však většinou nemáme). Tento druhotný problém by šlo ovšem také řešit například rozšířením intervalu pro splnění nátoku na technologii o ±0.125 m^3/min.

5.2.3 Zabudování ekonomického modelu do systému

Ekonomickým modelem se rozumí algoritmus, který dokáže odhadnout náklady spojené s daným krokem varianty. Algoritmus je implementován v samostatném modulu (*EkonomickyModel.cpp*). Ekonomické hodnocení se spouští automaticky při přístupu na poslední obrazovku systému. Zde je také k dispozici tabulka s jeho výsledky. V tabulce jsou vypsány různé druhy nákladů seřazené podle období (viz obrázek 4.9.4). Tyto výsledky se podobně jako ostatní výsledky vázané na dané období (krok varianty) ukládají do souboru s parametry (*parCAS.ini*), odkud jsou pak programem do souhrnných tabulek načítány. Tabulka ekonomického hodnocení obsahuje okamžité i kumulované náklady na jednotlivé technologie a na vytváření nových vrtů (je tedy patrné, že ekonomický model je úzce vázán na schopnost programu rozpoznat již existující a nové vrty). Dále obsahuje celkové náklady opět okamžité i kumulované a náklady jednotkové (tedy náklady přepočtené na kilogram TDS), okamžité a průměrné. Všechny výsledky, respektive celá tabulka ekonomického hodnocení, se navíc ukládají i do XLS souboru s výsledky (*vysledkyCAS.xls*). Některé z nákladů se vykreslují do grafů, které jsou taktéž na poslední obrazovce systému (viz obrázky 4.9.5 – 4.9.8). Tyto grafy lze ukládat do souborů v několika grafických formátech, obrázky se ukládají přímo do adresáře varianty.

Problémem ekonomického hodnocení je závislost na velkém množství různých ekonomických parametrů, které se velmi často mění (například cena kilowatthodiny elektrické energie, atd.). Aby bylo možno tyto parametry snadno modifikovat, načítají se všechny z *ini* souboru (*ekonopar.ini*), který je umístěn v základním adresáři systému.

5.3 Ryze programátorské změny a změny v designu programu

Jak bylo už uvedeno výše, na vývoji tohoto programu se podílela celá řada programátorů, a tak byly zdrojové texty programu poměrně značně nepřehledné. Program je vyvíjen ve vývojovém prostředí C++Builder, ve kterém lze používat pro programování alternativně jazyk C nebo jazyk C++. Vývojové prostředí samo o sobě dodržuje určitý styl (formu) psaní programů. Toto prostředí navíc rozšiřuje klasické možnosti C / C++ o celou řadu nových knihoven, funkcí, typů atp. čehož lze s výhodou využít.

Toto vývojové prostředí bylo již vybráno dříve. Bylo tak vybráno zřejmě proto, že vhodně kombinuje možnosti jazyka C / C++ s možnostmi vizuálního programování, jako je například v prostředích Delphi nebo Visual Basic.

5.3.1 Změny týkající se formulářů – rozhraní MDI

Každý formulář (obrazovka, dialogové okno) je v prostředí C++Builder realizován třemi zdrojovými soubory. Například okno s názvem *Form1* bude sestaveno s těchto souborů:

Form1.dfm – obsahuje informace o rozložení komponent na formuláři a o jejich statických nastaveních a o přiřazení metod (procedur) k jednotlivým událostem, které mohou komponenty generovat.

Form1.h – hlavičkový soubor (standardní hlavičkový soubor C / C++), který obsahuje definici objektu se jménem daného okna. Tento objekt je potomkem jakéhosi obecnějšího objektu okna (dědí jeho vlastnosti), jako jednotlivé položky obsahuje právě instance objektů (komponent), které okno obsahuje, a jako metody obsahuje procedury ošetřující události, jež mohou tyto komponenty generovat. Dále tento soubor obsahuje právě jednu instanci objektu daného okna.

Form1.cpp – obsahuje definice metod okna a popřípadě další pomocné procedury.

Celá aplikace je implementována pomocí rozhraní *MDI*, což znamená, že je zde jedno hlavní okno (*MDI Main*) a jedno klientské okno (*MDI Client*), které ovšem není ve skutečnosti realizováno žádným formulářem. Aplikace pak může obsahovat několik dalších oken – potomků (*MDI Child*). Tato okna nejsou klasická okna Windows, ale mají velmi podobné vlastnosti. Tato okna potomků se zobrazují v klientském okně (může jich být zobrazeno i několik najednou, pokud je však okno potomka maximalizované, je v klientském okně zobrazeno právě jedno).

V systému TURON 2003 je jedno hlavní okno a tedy i jedno klientské okno, která jsou realizována hlavním formulářem (okno má atribut *fsMDIForm*). Jednotlivé obrazovky systému jsou pak realizovány pomocí oken potomků (okna mají atribut *fsMDIChild*). Tento systém byl zaveden již dříve, ale právě díky postupnému rozšiřování programu takto nebyly implementovány všechny obrazovky, ale jen některé. Současně s tím byly i soubory, které obrazovky realizují, pojmenované různě a poněkud nepřehledně. V současné verzi systému jsou již všechny obrazovky realizované pomoci rozhraní *MDI* (vyjma okna, které tvoří logo při spuštění programu, a tzv. „debug“ okna). Všechny zdrojové soubory příslušející k formulářům už také mají sjednocené pojmenování *Form*.** (např. *FormObrazovka1.dfm*).

Primárním smyslem rozhraní *MDI* je ale vytvářet aplikace, které obsahují větší množství *Child* oken, jež jsou ovšem (vzhledem k programu) rovnocenná (*MDI = Multiple Document Interface*). Příkladem takových aplikací může být například Word nebo Excel. Zde si uživatel může otevřít více oken – dokumentů – a program se všemi pracuje stejně. V systému TURON 2003 jsou ale okna (jednotlivé obrazovky), která nejsou rovnocenná a nemělo by být možno se mezi nimi libovolně přepínat, přechod z jednoho okna na druhé má být proveden pouze po stisknutí tlačítka *Další* respektive *Zpět*. *MDI* rozhraní ovšem ve své podstatě toto přepínání mezi jednotlivými okny umožňuje. Přepínání mezi okny se provádí pomocí klávesové zkratky *Ctrl + Tab*. Při stisknutí těchto kláves tedy dojde v programu k přechodu na jinou obrazovku, ale bez příslušných změn v kontextu programu, proto je potřeba tuto klávesovou zkratku v programu nepoužívat. Optimální by bylo softwarově zamezit tomuto přepnutí například sledováním a ošetřením události *WM_MDIACTIVATE*.

5.3.2 Ukládání výsledků do souboru XLS – komponenta TADVStringGrid

Jedním z hlavních požadavků na systém byla možnost jednoduchého, rychlého a snadného zpracování výsledků. Do adresářů variant se sice výsledky ukládaly, ale pouze ve formě souborů **.df0*, **.df1* a **.df2*, které tyto požadavky rozhodně nesplňují. Na poslední obrazovce sytému TURON 2003 však již byla data z těchto souborů přehledně dekódována do různých tabulek. Nejvhodnější tedy bylo zajistit ukládání těchto tabulek přímo do souboru – například Excelu, v kterém pak již lze s daty jednoduše manipulovat.

Ukládání tabulek do souborů Excelu lze zajistit poměrně snadno, stačí například místo tabulek typu *TstringGrid*, které jsou standardními komponentami vývojového prostředí, použít tabulky typu *TADVStringGrid*. Tato komponenta pochází z vývojových dílen fy TMS Software a je k dispozici (k volnému stažení, za předpokladu nekomerčního využití) na webových stránkách této firmy a je také standardně na originálním instalačním disku vývojového prostředí C++Builder / Delphi. (Vzhledem k tomu, že na instalačním CD systému TURON 2003 jsou k dispozici i kompletní zdrojové texty hlavního programu, je zde také instalační balíček s touto komponentou.)

Tabulka typu *TADVStringGrid* již dokáže automaticky ukládat svůj obsah do libovolného souboru formátu *XLS*, *CSV*, *HTM*, *TXT*, atd. Tabulku lze také vložit jako jeden list (*sheet*) do již existujícího souboru Excelu. Díky těmto vlastnostem komponenty je možno uložit všechny tabulky z obrazovky s výsledky modelu do jednoho souboru (*vysledkyCAS.xls*), který obsahuje několik listů; na každém z nich je právě jedna tabulka.

Tento postup při ukládání tabulek má ovšem i své nevýhody. Jak bylo experimentálně zjištěno, na některých konfiguracích počítače (zejména záleží na verzi Windows, Office a Internet Exploreru) dojde při ukládání tabulek do Excelu k chybě, jejíž původ se zatím nepodařilo blíže zjistit. Z tohoto důvodu je v tzv. debug menu (viz obrázek 4.1.3) a v konfiguračním souboru *turon.ini* k dispozici volba *SKIP_XLS*. Po zapnutí této volby systém automaticky přeskočí ukládání XLS souboru a k chybě tak nedojde.

5.3.3 Změny týkající se „nevizuálních“ souborů

Kromě souborů, které generuje vývojové prostředí automaticky (zdrojové soubory formulářů), je v systému TURON 2003 řada dalších zdrojových souborů. Původně zde bylo těchto souborů jen několik málo, ovšem byly značně velké a orientace v nich byla nesnadná. Navíc každý soubor vypadal trochu jinak, neboť každý soubor vytvářel někdo jiný, a většina těchto souborů byla v průběhu doby různě modifikována, takže například i navzájem velmi úzce související procedury byly implementovány na odlišných místech. V některých modulech byly dokonce implementovány funkce, které se nikde nepoužívaly, nebo funkce, které byly pouze deklarované, ovšem implementované nebyly.

V současné době je již většina těchto souborů přepsána a jejich formátování je v rámci možností unifikováno. Pro celý systém je pouze jeden hlavičkový soubor *Declar.h* (vyjma hlavičkových souborů, které přináležejí formulářům), který obsahuje definice všech globálních konstant, struktur, definice všech globálních proměnných a deklarace všech globálních funkcí (původně měly některé moduly samostatné hlavičkové soubory, některé měly společný). Do souboru jsou ještě navíc vloženy hlavičkové soubory všech formulářů a některé běžně používané hlavičkové soubory (*stdio.h*, *math.h*, atd.).

V samostatných modulech (soubory **.cpp*) jsou výhradně implementovány globální funkce deklarované v souboru *Declar.h*. Tyto moduly jsou nyní také pojmenovány víceméně jednotně tak, aby bylo možno snadno určit, čeho se týkají. Navíc v hlavičkovém souboru jsou jednotlivé globální funkce popsány a je u nich uvedeno, v kterém modulu jsou implementovány. V programu se vyskytují například následující moduly:

FileManagement.cpp – v tomto modulu jsou implementovány funkce pro práci se soubory. Modul obsahuje funkce pro manipulaci se soubory, dále funkce mazání obsahu adresáře / mazání obsahu adresáře temp, nebo například funkce pro vytvoření záložní kopie souboru.

SouboryParametry.cpp – procedury pro načítání / vytváření souborů s parametry (*ini* soubory).

VypocetOdhadu.cpp – procedury pro spuštění výpočtu odhadů.

a další …

5.3.4 Ostatní drobné změny

V průběhu předělávání souborů byla také snaha o maximální využití programovacích prostředků, které skýtá moderní vývojové prostředí. Zejména využívání snazší manipulace s řetězci pomocí proměnných (objektů) typu *String* nebo *AnsiString*, které mají řadu výhod oproti klasickým řetězcům jazyka C / C++. Řetězce těchto typů lze například snadno sčítat, porovnávat, vkládat do nich čísla z číselných proměnných atd. Navíc většina vizuálních komponent v prostředí C++Builder má všechny textové vstupy a výstupy realizovány právě pomocí těchto objektů.

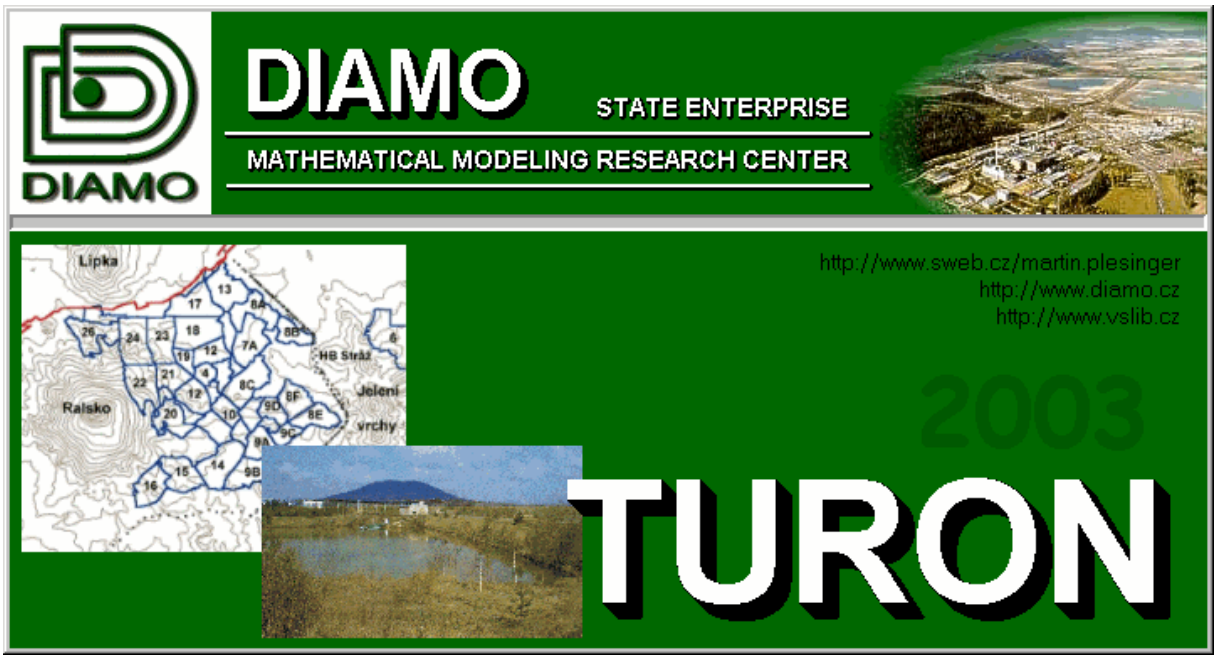
Využití takovýchto prostředků výrazně zjednoduší práci programátora, a to nejen při vytváření programu, ale i při jeho pozdějších modifikacích, neboť celý zdrojový text je výrazně přehlednější.

5.3.5 Designové změny

Mezi hlavní změny v designu aplikace patří sjednocení vzhledu jednotlivých oken. Například velké nadpisy jsou na všech obrazovkách na stejných místech, jsou stejně velké, stejně odsazené od okraje okna, stejně barevné a použit je stejný font. Značně nepřehledně ve starém turonském systému také působilo chaotické umístění jednotlivých tlačítek na obrazovkách. Na různých obrazovkách byla tlačítka různě velká a dokonce tlačítka se stejnou funkcí (konkrétně tlačítka *Zpět* a *Další*) byla umístěna na různých obrazovkách na různých místech, což při manipulaci s programem nepůsobilo příliš esteticky.

Programu také přibyla nezbytná ikona a při spuštění programu se navíc zobrazí logo s názvem programu.

Obr. 5.2 – logo, které se zobrazuje při spouštění programu TURON 2003



5.4 Ostatní úpravy v systému

5.4.1 Smazání přebytečných souborů

Turonský systém v původním stavu, tak jak byl, zabíral přibližně 100 MB sice včetně zdrojových textů, ale naopak bez dočasného adresáře. Zdrojové texty měly přibližně 100 souborů a vlastní systém měl dalších cca 120 souborů. Většina těchto souborů nebyla vůbec potřeba, a to jak ve zdrojových textech, tak v systému. Jednalo se o různé staré verze programů, zálohy zdrojových textů, ale i zdrojové texty, které se systémem vůbec nesouvisely, nepotřebné dynamické knihovny a inicializační soubory, fragmenty různých datových souborů a pozůstatky po generování sítě.

Vzhledem k tomu, že většina z těchto souborů byla úplně zbytečná, byly ze systému postupně odstraněny. Tímto způsobem se podařilo zredukovat počet souborů na cca 90, z čehož přibližně 60 souborů připadá na zdrojové texty. Naopak byly do systému přidány některé dynamické knihovny, které zde chyběly a jsou potřeba pro správný chod programu a nejsou standardní součástí Windows. Nyní systém po nainstalování na počítač zabírá cca 20 MB místa na disku. Přesnější seznam souborů, které se v systému nacházejí, nalezneme v přílohách.

5.4.2 Požadavky na hardware

V souvislosti s redukcí počtu souborů byly i experimentálně stanoveny přibližné parametry počítače, které jsou pro správný chod systému vyžadovány. Program sice zabírá pouze cca 20 MB na disku, při spuštění modelu transportu látek však zabrané místo naroste přibližně na 300 MB (velké soubory v dočasném adresáři). Data vypočtené varianty také zabírají řádově desítky megabytů. Optimální volný diskový prostor tak byl stanoven s ohledem na snadnou a rychlou manipulaci s velkými soubory cca na 1 GB na disku, na kterém je systém nainstalován. Samozřejmě dočasný adresář i adresář s variantami mohou být umístěny na jiném disku, než je systém TURON 2003.

Co se týče procesoru, bylo odzkoušeno, že procesor 300 MHz stačí, ale výpočet jednoho kroku trvá přibližně 30 minut. Samozřejmě v tomto případě platí: čím rychlejší procesor, tím lépe. Velikost operační paměti byla vymezena dvěma krajními body: 80 MB rozhodně nestačí pro úspěšné vypočtení kroku; 192 MB stačí. Přesnější hodnoty se bohužel nepodařilo zjistit. Nicméně největší nároky na operační paměť má program, který řeší soustavu lineárních rovnic úlohy proudění – solver NI1 – tento program způsobí pád výpočtu proudění na počítači s menší operační pamětí.

5.4.3 Vytvoření instalačního disku

K systému TURON 2003 byl vytvořen instalační soubor, respektive instalační CD, z kterého je možné nainstalovat jednotlivé komponenty systému. Při instalaci je na výběr několik typů instalace (kompaktní, úplná a uživatelská). Při uživatelské instalaci si můžeme vybrat, které z instalačních balíčků se mají na disk zkopírovat. Opět je na výběr několik balíčků (vlastní systém, soubory kopírované do sdílené systémové složky *Windows\Command*, soubory kopírované do systémové složky *Windows\System*, zdrojové texty programu a jedna vypočtená varianta). Součástí instalačního CD je i jakýsi manuál a návod k instalaci (soubor *readme.pdf*), který je později k dispozici i v nainstalované verzi programu. Dále je na CD instalační balíček s komponentou *TADVStringGrid* pro vývojové prostředí *Borland C++Builder 5*. Poslední součástí instalačního disku je tato dokumentace celého projektu.

6 Závěry a doporučení

Optimalizační software – systém TURON 2003 – respektuje všechny dosavadní požadavky, které byly formulované na základě zkušeností s optimalizací sanačního procesu. Systém plně vyhovuje současným technologickým potřebám a současně platným představám technologické koncepce sanace. Systém umožňuje regulérní zpracování řady kompletních variant sanace s možným zapojením čtyř plně nezávislých technologií a s možností respektování existujících vrtů.

Další práce na systému by měla směřovat především k permanentnímu prověřování systému, odhalování jeho možností a případných slabin a nedostatků. Dále bude model neustále rozšiřován a doplňován ve smyslu zjištěných potřeb. Takovým rozšířením by například mohlo být právě zabudování mechanismu respektování existujících vrtů do algoritmu výpočtu odhadů.

Výhledovým rozšířením optimalizačního softwaru by mohlo být také jeho propojení se systémem pro řízení sanace cenomanské zvodně a s dalšími systémy. Vývoj systému by také mohl být například ve znamení případného zkracování kroku simulace nebo zpřesňování numerických modelů.

P1 Seznam souborů a hierarchie adresářů systému TURON 2003

P1.1 Hierarchie adresářů v kořenovém adresáři systému

Adresář:	Popis:
sit	– adresář se soubory FEM sítě
varianty	– složka, ve které se vytvářejí složky jednotlivých variant
varianty\V01	– složka varianty V01
temp	– dočasná složka, zde se vytvářejí pomocné soubory během výpočtů
source	– složka se zdrojovými texty programu

Pochopitelně adresářů jednotlivých variant (jako je adresář V01) se může v adresáři *varianty* vyskytovat více. V každém takovém adresáři pak může být umístěna jedna varianta.

P1.2 Seznam souborů v kořenovém adresáři systému

P1.2.1 Programy a podprogramy v systému; soubory sdílené ve složce C:\Windows\Command

Soubor:	Popis:
turon.exe	– hlavní spouštěcí soubor celého systému, obsahuje systém obrazovek
ini_editor.exe	– program, pomocí kterého lze nastavovat parametry v ini souborech *ekonopar.ini* a *turon.ini*
gwsview.exe	– grafický postprocesor systému; lze spouštět i samostatně
tur_odhad	– program pro výpočet odhadů; jednoúčelový program
xa.exe	– program pro řešení lineárních problémů; jednoúčelový komerční program
gen_flow.exe	– program pro výpočet proudění; jednoúčelový program
gen_tran.exe	– program pro výpočet transportu látek; jednoúčelový program
solver.bat	– dávkový soubor volaný programem *gen_flow.exe*, tento skript spouští řešič lineárních rovnic
ni1.exe	– solver – řešič soustav lineárních rovnic, tento program je volán programem *gen_flow.exe* prostřednictvím skriptu *solver.bat*

Soubory *solver.bat* a *ni1.exe* jsou při standardní instalaci také nakopírovány do složky C:\Windows\Command. Pokud jsou ovšem tyto dva soubory umístěny ve výše uvedené systémové složce, která je implicitně sdílená v systémové proměnné PATH, nemusí být již umístěny v adresáři systému. Po instalaci zde však jsou pro přehlednost a úplnost a také pro snadné přenesení již nainstalovaného systému na jiný počítač. Pokud nechceme tyto soubory kopírovat do systémové složky, lze tento problém vyřešit alternativně. Více k tomuto tématu naleznete v kapitole týkající se problémů, které se mohou v systému vyskytnout, nebo v dokumentaci k systému (soubor readme.pdf, který je součástí instalace).

P1.2.2 Nestandardní dynamické knihovny v systému; soubory sdílené ve složce C:\Windows\System

Knihovna:	Popis:	Využití knihovny v programech:
borlndmm.dll	– Borland Memory Manager	*turon.exe*, *gwsview.exe*
cc3250mt.dll	– Borland C++ Multi-thread RTL (WIN/VCL MT)	*turon.exe*, *gwsview.exe*
cw3230.dll	– Dynamic Link Run Time Library	*gen_flow.exe*, *gen_tran.exe*, *tur_odhad.exe*
vcl50.bpl	– Borland Component Package	*turon.exe*, *gwsview.exe*
vclx50.bpl	– Borland Extended Component Package	*turon.exe*, *gwsview.exe*
vcldb50.bpl	– Borland Database Component Package	*gwsview.exe*
vclbde50.bpl	– Borland BDE DB Component Package	*gwsview.exe*
tee50.bpl	– Borland TeeChart Component Package	*gwsview.exe*
bcbsmp50.bpl	– ? blíže neurčená knihovna	*gwsview.exe*

Všechny tyto dynamické knihovny (soubory *.dll a *.bpl) jsou při standardní instalaci také nakopírovány do složky C:\Windows\System. Pokud je ovšem těchto devět souborů umístěno ve výše uvedené systémové složce, nemusí být již umístěny v adresáři systému. Po instalaci zde však jsou pro přehlednost a úplnost a také pro snadné přenesení již nainstalovaného systému na jiný počítač. Pokud nechceme tyto soubory kopírovat do systémové složky, lze tento problém vyřešit alternativně. Většina těchto knihoven je navíc standardně dodávána s vývojovým prostředím Borland C++Builder 5, respektive Borland Delphi 5 od firmy Inprise, a pokud je některé z těchto prostředí na počítači již nainstalováno, tyto soubory se v systémové složce již nejspíš nacházejí. Více k tomuto tématu naleznete v kapitole týkající se problémů, které se mohou v systému vyskytnout, nebo v dokumentaci k systému (soubor readme.pdf, který je součástí instalace). Program ni1.exe ještě navíc obsahuje volání na nestandardní knihovnu dformsg.dll, zřejmě ji však nepotřebuje ke svému chodu (odzkoušeno). Tato knihovna navíc ani nebyla s tímto programem dodána.

P1.2.3 Další soubory, které se mohou v kořenovém adresáři systému vyskytnout

Soubor:	Popis:
turon.ini	– hlavní konfigurační soubor systému TURON 2003
ekonopar.ini	– konfigurační soubor obsahující parametry ekonomického modelu
readme.pdf	– soubor se stručným popisem programu, zejména s postupem instalace a s popisem možných problémů, které se během chodu programu mohou vyskytnout; tento soubor pochopitelně není pro správný chod systému potřeba
tur_odhad.pif	– zástupce program *tur_odhad.exe*
xa.pif	– dtto
gen_flow.pif	– dtto
gen_tran.pif	– dtto
Turon.tds	– soubor s daty pro ladění programu turon.exe. V adresáři se vyskytuje v případě, že provádíme další vývoj aplikace. Tento soubor je automaticky vytvářen vývojovým prostředím Borland C++Builder 5 spolu se spustitelným souborem. Pokud jsme systém pouze instalovali, tento soubor se zde nevyskytuje; pro chod systému pochopitelně není potřeba
DeIsL1.isu	– pozůstatek po instalátoru; je zde v případě, že jsme program instalovali pomocí standardního instalátoru. Tento soubor není nutný pro správný chod systému
_DEISREG.ISR	– dtto
_ISREG32.DLL	– dtto

P1.3 Seznam souborů v adresáři sítě

P1.3.1 Základní soubory vícevrstvé sítě komíny

Soubor:	Popis:
kominy.sit	– definice sítě komíny
kominy.ste	– seznam multielementů sítě a jejich přiřazení k multiuzlům
kominy.stu	– seznam multiuzlů sítě a jejich horizontální souřadnice
kominy.stm	– materiálové konstanty pro jednotlivé elementy

P1.3.2 Soubory s okrajovou podmínkou na vícevrstvé síti komíny

Soubor:	Popis:
kominy.okp	– definice okrajové podmínky na síti komíny
konstant.oke	– „konstantní“ okrajová podmínka na elementech sítě komíny

V souboru okp je okrajová podmínka pouze definovaná, oproti tomu v souboru oke jsou již zapsány konkrétní podmínky (jsou v něm konkrétní číselná data). „Dynamická“ – nekonstantní – okrajová podmínka, obsahující vybrané vrty, respektive údaje o multielementech, ze kterých se má čerpat, se generuje až za běhu programu lineárním modelem. Pro výpočet proudění a transportu látek jsou potřeba oba tyto typy podmínek – „konstantní“ i „dynamicky“ generovaná. Soubor s touto „dynamickou“ podmínkou nalezneme pro každý krok výpočtu v adresáři s vypočtenou variantou. Viz také odstavec *P1.4.2 Seznam souborů po ukončení jednoto kroku výpočtu*.

P1.3.3 Ostatní soubory související s vícevrstvou sítí komíny

Soubor:	Popis:
kominy.stn	– seznam vnitřních i vnějších stěn sítě

Pokud se tento soubor nevyskytuje v adresáři sítě, systém si ho umí sám vytvořit (konkrétně vytváří tento soubor program GEN_FLOW). Pokud se však soubor v adresáři sítě již vyskytuje, znovu se nevytváří, pouze je programem načten. Tento soubor tedy v podstatě k výpočtu není potřeba, v rámci urychlení výpočtu je však součástí instalace systému.

P1.3.4 Základní soubory jednovrstvé sítě komíny_1v

Soubor:	Popis:
kominy_1v.sit	– definice sítě komíny_1v
kominy_1v.ste	– seznam elementů sítě a jejich přiřazení k uzlům
kominy_1v.stu	– seznam uzlů sítě a jejich horizontální souřadnice
kominy_1v.stm	– materiálové konstanty pro jednotlivé elementy

P1.3.5 Ostatní soubory, které se nacházejí v adresáři sítě

Soubor:	Popis:
poleobu.dxf	– soubor grafického systému AutoCAD; obsahuje mapu vyluhovacích polí; tento soubor slouží jako podklad při vykreslování dat v grafickém postprocesoru GwsView
lokal.elm	– obsahuje informace o přiřazení multielementů sítě k jednotlivým vyluhovacím polím

P1.4 Seznam souborů v adresáři varianty V01

P1.4.1 Seznam souborů před zahájením výpočtu – soubory nutné k zahájení výpočtu

Předpokládejme, že jsme úplně na začátku před zahájením výpočtu pro první období (0 – 180 dní).

Soubor:	Popis:
m0.ts3	– soubor s rozložením kontaminace na počátku prvního časového kroku (období 0 – 180 dní) dané varianty V01. Tento soubor je součástí standardní instalace programu, pokud tomu tak není, musíme ho do systému dodat. Tento soubor slouží k výpočtu odhadů pro dané období a je vlastně počáteční podmínkou pro výpočet transportu látek. Tytéž soubory, potřebné pro další časové kroky (soubory *m180.ts3*, *m360.ts3*, atd.), jsou automaticky generované systémem při zpracování dat modelu transportu.
m0_slow.ts3	– dtto jako m0.ts3. Tento soubor však na rozdíl od předchozího obsahuje data s rozložením kontaminace v takzvaných „*neprůchozích*“ pórech. Oproti tomu soubor předchozí obsahuje data o kontaminaci v takzvaných „*průchozích*“ pórech. V turonském modelu se totiž počítá s tzv. *dvojí pórovitostí materiálu matečné horniny*.
sexvrt0.txt	– seznam existujících vrtů na počátku sanace; obsahuje informace o horizontální a vertikální lokalizaci vrtu (číslo multielementu a otevření vrtu) a čerpací výkon vrtu.

Následující soubor není k zahájení výpočtu bezpodmínečně nutný, je zde však uveden, protože je zatím součástí standardní instalace systému. Pokud se tento soubor v adresáři varianty nevyskytuje, je systémem automaticky generován v průběhu procházení mezi jednotlivými obrazovkami.

par0.ini – soubor s parametry prvního kroku varianty (období 0 – 180 dní). V tomto souboru jsou uloženy informace o nastavení jednotlivých parametrů, které provádíme v průběhu procházení obrazovkami systému. Tento soubor však není potřeba pro zahájení výpočtu, je systémem generován automaticky.

P1.4.2 Seznam souborů po ukončení jednoho kroku výpočtu

Předpokládejme, že jsme právě dokončili výpočet pro první období (0 – 180 dní). Soubory v tomto odstavci jsou seřazeny víceméně chronologicky podle času vzniku.

Soubor: Popis:

m0.ts3 – soubor s počátečním rozložením kontaminantů (průchozí póry); nutný pro zahájení výpočtu; podrobněji popsáno v předchozím odstavci P1.4.1

m0_slow.ts3 – soubor s počátečním rozložením kontaminantů (neprůchozí póry); nutný pro zahájení výpočtu; podrobněji popsáno v předchozím odstavci P1.4.1

sexvrt0.txt – soubor se seznamem existujících vrtů; obsahuje důležité informace o vrtu

par0.ini – soubor s nastavením parametrů daného období; tento soubor je systémem generován automaticky a je automaticky aktualizován v průběhu procházení jednotlivými obrazovkami; podrobněji viz odstavec P1.4.1

odh0.ts3 – soubor s daty odhadů pro dané období. Tento soubor se vytváří při výpočtu odhadů ze souborů *m0.ts3* a *m0_slow.ts3*. Tento soubor obsahuje na rozdíl od dvou výše jmenovaných data, která se zobrazují pouze na jednovrstvé síti komíny_1v. Je to v podstatě „jednovrstvý obraz vícevrstvých souborů“ *m0.ts3* a *m0_slow.ts3*.

oke0.oke – „dynamicky“ generovaná okrajová podmínka definovaná na elementech sítě. Tato okrajová podmínka je pro každý krok jiná a zohledňuje vybrané vrty. Tyto čerpací vrty (popřípadě i vtláčecí vrty, tyto se ovšem v „turonském“ systému nerealizují) lze totiž považovat za speciální typ okrajové podmínky. Tento soubor slouží spolu se souborem *konstant.oke*, který se nachází v adresáři sítě a který obsahuje „konstatní“ okrajovou podmínku, pro program *GEN_FLOW* pro výpočet proudění. Viz také odstavec *P1.3.2 Soubory s okrajovou podmínkou na vícevrstvé síti komíny*.

sexvrt180.txt – soubor se seznamem existujících vrtů pro další krok výpočtu

m0.df0 – soubor s výsledky modelu transportu – výstup programu *GEN_TRAN*. Z tohoto souboru se čtou data vypisovaná na poslední obrazovce grafického systému

m0.df1 – dtto

m0.df2 – dtto

m180.ts3 – tento soubor je také výstupem modelu transportu – programu *GEN_TRAN*, slouží však jako vstupní data pro další krok výpočtu (období 180 – 360 dní). Soubor obsahuje rozložení kontaminace na konci tohoto (tedy i na začátku dalšího) kroku. Opět obsahuje data pouze pro průchozí póry.

m180_slow.ts3 – dtto pro neprůchozí póry

vysledky0.xls – tento soubor obsahuje všechny tabulky, které byly zobrazeny na poslední obrazovce grafického systému (výsledky modelu transportu). Jedná se o standardní formát souboru používaný programem MS Excel

P1.4.3 Záložní soubory, které se mohou vyskytovat v adresáři varianty

Příklady záložních souborů:

par0.ini_2002-12-15_16-10-40_backup.ini
par180.ini_2002-12-17_00-59-13_backup.ini
vysledky180.xls_2002-12-17_01-34-09_backup.xls

Syntaxe názvů přidělovaných záložním souborům:

<jméno a přípona pův. soub.>_<datum (rrrr-mm-dd)>_<čas (hh-mm-ss)>_backup.<přípona pův. soub.>

P1.5 Poznámka k seznamu souborů

V průběhu celého výpočtu se mohou ve všech adresářích systému TURON 2003 (nejen v dočasném adresáři temp) vyskytovat nejrůznější soubory, které zde nejsou popsány. Většinou se jedná například o různé logovací soubory, soubory se vstupy a výstupy programu XA, nebo pomocné soubory pro výpočet proudění a transportu látek. Konkrétně soubory týkající se výpočtu transportu látek mohou mít i několik desítek megabytů (tři soubory à 87.2 MB). Všechny tyto soubory by však měly při korektním ukončení programu zmizet.

P2 Struktura některých důležitých souborů

P2.1 Struktura některých souborů sítě

P2.1.1 Struktura souborů *.SIT

Textový soubor se strukturou *ini* souboru; v souboru je řada parametrů týkajících se sítě. Tento soubor v podstatě definuje celou sít. Soubor obsahuje například následující nejdůležitější sekce:

[GWS_SIT_DEF] – obsahuje data definující síť;
[GWS_SIT_NAZVY] – definuje používané názvy vrstev, ploch a materiálů sítě;
[GWS_SIT_KOEF] – určuje názvy materiálových koeficientů a data pro jejich grafické zobrazení.

Soubor může obsahovat ještě řadu různých sekcí, tyto tři jsou však nejdůležitější. Ukázka souboru kominy_1v.sit:

```
[GWS_SIT_DEF]
Verze=GWS-5.0
Popis=Síť konečných elementů
MaxUzlu=3000
MaxElm=7000
NVrstev=1
IHlavni=0
NKoef=4
NElmKoef=1
NMaterial=2
Krok=1
MeritkoZ=10
VrtyFile=*
DxfFile=
Pohled=0,60
BoxHledani=5
Alfa=0
CaryFile=
ClrVrty=10
ClrVrtyTxt=15
ClrPozadi=1

[GWS_SIT_NAZVY]
IPlocha=1
IVrstva=0
IKoef=2
IElmKoef=0
Vrstvy=I1 I2 I3 J1 J2 J3 J4 J5 J6 J7 J8 J9
Plochy=pI1 pI2 pI3 pJ1 pJ2 pJ3 pJ4 pJ5 pJ6 pJ7 pJ8 pJ9 hJ9
Material=i1  i2  i3 j1 j2 j3 j4 j5 j6 j7 j8 j9

[GWS_SIT_KOEF]
Stp0=Kx,0,0.01,0.05,0.1,0.2,0.5,1,1.5,2,3,4,5,10,15,20
Frm0=6,2
Stp1=Ky,0,0.01,0.05,0.1,0.2,0.5,1,1.5,2,3,4,5,10,15,20
Frm1=6,2
Stp2=Kz,0,0.01,0.05,0.1,0.2,0.5,1,1.5,2,3,4,5,10,15,20
Frm2=6,2
Stp3=X,0
Frm3=6,2

[GWS_SIT_ELMKOEF]
Stp0=K1/M1,0.000010,0.000100
Frm0=12,6

[GWS_SIT_POZADI]
nPozadi=0
Pozadi0=9,10000,poleobu.dxf

[GWS_SIT_GVRSTVY]
GVrstvy=I I I J J J J J J J J
```

P2.1.2 Struktura souborů *.STU

Textový soubor popisující multiuzly sítě; obsahuje věty s následující strukturou a významem:

OZNAC X Y POVRCH IDOLNI IHORNI VYSKY

OZNAC	– je číselné označení multiuzlu;
X , Y	– jsou horizontální souřadnice multiuzlu;
POVRCH	– je výška terénu v místě multiuzlu;
IHORNI, IDOLNI	– jsou čísla krajních vrstev v tomto multiuzlu;
VYSKY	– je seznam výšek uzlů v daném multiuzlu.

Ukázka části souboru kominy_1v.stu:

```
GWS_SIT_UZL
1 10732.4 11045 295.91  0 1  244.69 295.91
2 10765.4 11046.5 295.94  0 1  243.32 295.94
3 10748.7 11070.4 295.9  0 1  243.52 295.9
4 10744.9 11020.2 295.94  0 1  244.73 295.94
5 10735.5 11107.4 295.85  0 1  243.12 295.85
...
```

P2.1.3 Struktura souborů *.STE

Textový soubor popisující multielementy sítě; obsahuje věty s následující strukturou a významem:

OZNAC UZEL0 UZEL1 UZEL2 IDOLNI IHORNI KOEF0 KOEF1 ...

OZNAC	– je číselné označení multielementu;
UZEL0, UZEL1, UZEL2	– jsou číselná označení multiuzlů v rozích multielementu;
IDOLNI, IHORNI	– jsou čísla krajních vrstev v daném multielementu;
KOEF0, KOEF1, ...	– jsou koeficienty daného multielementu.

Ukázka části souboru kominy_1v.ste:

```
GWS_SIT_ELM
2 5 3 6 0 0 0
3 1 2 3 0 0 0
4 2 1 4 0 0 0
6 5 6 7 0 0 0
7 6 3 8 0 0 0
...
```

P2.1.4 Struktura souborů *.STM

Textový soubor popisuje materiálové konstanty elementů; obsahuje věty s následující strukturou a významem:

OZNAC CISLO_VRSTVY CISLO_MAT K0 K1 ...

OZNAC	– je číselné označení multielementu, pro který jsou hodnoty zadávány;
CISLO_VRSTVY	– je číslo vrstvy v daném mutlielementu, kterou tato věta popisuje;
CISLO_MAT	– je číslo materiálu v této vrstvě;
K0, K1, ...	– jsou koeficienty příslušné k danému elementu v dané vrstvě.

Ukázka části souboru kominy_1v.stm:

```
GWS_SIT_MAT
2 0 0 0.8 0.8 0.2 0.28
3 0 0 0.8 0.8 0.2 0.28
4 0 0 0.8 0.8 0.2 0.28
6 0 0 0.8 0.8 0.2 0.28
7 0 0 0.8 0.8 0.2 0.28
...
```

P2.1.5 Struktura souboru KOMINY.OKP

Textový soubor se strukturou *ini* souboru. Tento soubor obsahuje definici okrajové podmínky na celé síti. Ukázka souboru *kominy.okp*:

```
GWSOKP_DEF
.\kominy.sit
Současný stav
5574  2828  12
END

BODY_Q  OKP1
  KOEF  0
  VODA  0
  H  0
  VYZNAM  3
  BARVA  11
  MIXHYBRID   1
  OKNO_R  0  0  100  100
  OKNO_I  80  20  732  600  0 0 0
  DLG_I  0  0
  UZLY
  ENDUZLY
  ELM
  ENDELM
  HRANY
  ENDHRANY
  DOLNI  0
  HORNI  11
  QJED  0
  BODY
  ENDBODY
END
```

P2.1.6 Struktura souborů *.OKE

Jedná se o soubory obsahující přímo konkrétní okrajové podmínky definované na multielementech sítě. V systému se vyskytují dva základní zástupci těchto souborů – *konstant.oke* – soubor s tzv. „konstantní“ okrajovou podmínkou a soubory *okeCAS.oke* – soubor s tzv. „dynamickou“ okrajovou podmínkou, která je generovaná v závislosti na vybraných vrtech v daném kroku.

Ukázka části souboru s „konstantní“ okrajovou podmínkou *konstant.oke*:

```
GWS_OKP_MIXHYBRID
TURON, konstantni okrajove podminky
9991    13      4731    1       0       6       299.850         0.000
9991    13      4735    1       0       6       299.963         0.000
9991    13      4739    2       0       6       299.667         0.000
9991    13      4741    1       0       5       299.252         0.000
9991    13      4749    1       0       6       300.098         0.000
9991    13      4786    1       0       6       300.275         0.000
9991    13      4824    1       0       5       300.434         0.000
9991    13      4825    1       1       5       300.572         0.000
9991    13      4827    1       1       4       300.664         0.000
9991    13      4833    1       0       3       300.748         0.000
...
9991    25      5420    4       0       0       293.610         0.000
9991    25      5424    4       0       0       293.296         0.000
9991    25      5428    4       0       0       293.146         0.000
201     99      5929    6       2       3     -1000.000                 0.000
```

Ukázka části souboru s „dynamickou“ okrajovou podmínkou *oke0.oke* Povšimněme si, že soubor s „dynamickou“ podmínkou obsahuje i data z podmínky „konstantní“. Soubor *konstant.oke* se ve skutečnosti připojuje na konec každého souboru *okeCAS.oke*.

```
GWS_OKP_MIXHYBRID
TURON, VAR: V01 CAS: 0  SIT: F:\DIAMO\TURON\sit\kominy
201      4   3   6   0   0  -360.000     0.000
201      4   6   6   0   0  -360.000     0.000
201      6  10   6   0   0  -360.000     0.000
201      6 133   6   0   0  -118.080     0.000
201      6 6127   6   2   2  -241.920     0.000
201      4 4148   6   0   0   -41.400     0.000
201      6 1577   6   3   3  -180.000     0.000
201      6 6636   6   3   3  -180.000     0.000
201      6 6643   6   3   3  -180.000     0.000
201      6 1355   6   3   3  -360.000     0.000
201      6 1393   6   3   3  -360.000     0.000
...
201      4 568   6   3   3  -210.600     0.000
201      4 612   6   2   2  -360.000     0.000
201      3 2341   6   2   2  -168.120     0.000
201      3 292   6   3   3  -360.000     0.000
201      6 2847   6   0   0  -360.000     0.000
201      5 6205   6   3   3  -108.360     0.000
9991    13      4731    1       0       6       299.850        0.000
9991    13      4735    1       0       6       299.963        0.000
9991    13      4739    2       0       6       299.667        0.000
9991    13      4741    1       0       5       299.252        0.000
9991    13      4749    1       0       6       300.098        0.000
9991    13      4786    1       0       6       300.275        0.000
9991    13      4824    1       0       5       300.434        0.000
9991    13      4825    1       1       5       300.572        0.000
9991    13      4827    1       1       4       300.664        0.000
...
9991    25      5420    4       0       0       293.610        0.000
9991    25      5424    4       0       0       293.296        0.000
9991    25      5428    4       0       0       293.146        0.000
201     99      5929    6       2       3     -1000.000              0.000
```

P2.1.7 Struktura souboru LOKAL.ELM

Přiřazení jednotlivých multielementů k vyluhovacím polím. Textový soubor; obsahuje věty s následující strukturou a významem:

I_ELM I_POLE

I_ELM – je číselné označení multielementu;
I_POLE – je označení příslušného vyluhovacího pole;

Ukázka části souboru lokal.elm:

```
lok
2       4-
3       4-
4       4-
6       4-
7       4-
8       4-
9       4-
10      4-
11      4-
...
6697    8C
6698    8C
6699    8C
2506    8D
2521    8D
2813    8D
2814    8D
4032    8D
4040    8D
...
```

P2.2 Struktura některých souborů nacházejících se v adresáři varianty

V adresáři varianty se vyskytují i soubory s tzv. „dynamickou“ okrajovou podmínkou. Tyto však byly popsány v předchozím odstavci, který se týkal souborů sítě.

P2.2.1 Struktura soborů SEXVRT*.TXT

Tento soubor obsahuje seznam existujících vrtů pro dané období. Jedná se o textový soubor s následující strukturou:

CISLO_ELEMENTU OTEVRENO_OD OTEVRENO_DO KAPACITA_VRTU

CISLO_ELEMENTU	– je číslo elementu (multielementu), na kterém je vrt realizován;
OTEVRENO_OD, DO	– nám udává, od které a do které vrstvy je vrt otevřen;
KAPACITA_VRTU	– je objem roztoků čerpaný za minutu z tohoto vrtu.

Ukázka souboru *sexvrt0.txt* (hlavička souboru je zatím povinná):

```
; seznam existujicich vrtu v case 0, pred zahajenim vypoctu
;
; lokalizace (cislo elementu), otevreno od vrstvy, do vrtstvy, kapacita vrtu
;
45 0 3 25
83 0 3 25
181 0 3 250
6126 0 3 250
6128 0 3 250
1469 0 3 250
6643 0 3 250
2735 0 3 250
1354 0 3 250
5972 0 3 250
1659 0 3 250
1675 0 3 250
1681 0 3 250
1684 0 3 250
6052 0 3 250
2455 0 3 250
2483 0 3 250
2484 0 3 250
2496 0 3 250
2798 0 3 250
6161 0 3 250
417 0 3 125
418 0 3 125
501 0 3 125
311 0 3 125
2359 0 3 250
6227 0 3 250
```

P2.2.2 Struktura soborů PAR*.INI

Textový soubor se strukturou klasického *ini* souboru. Tento soubor obsahuje informace o nastavení všech parametrů, s kterými se dá v programu manipulovat, také obsahuje některé informace o výsledcích výpočtu. Tento soubor je nejdůležitějším řídícím soubor systému. Následuje komentovaná ukázka části souboru *par0.ini*:

Na začátku je hlavní sekce souboru obsahující informace o názvu varianty a o období, ke kterému se vztahuje.

```
[General]
Varianta=V01
RokOd=0
RokDo=180
```

Následují čtyři sekce týkající se základního nastavení technologií. Zapnutí technologie, objem nátoku na technologii, technologicky požadované koncentrace TDS a NH4, koeficient poklesu koncentrací atd.; přesný popis všech nastavení je v kapitole týkající se popisu systému obrazovek.

```
[NDS]
Zapnuto=1
Q_Natoku=1
Kc_TDS_Natoku=7
Kc_NH4_Natoku=0.3
Kc_Pokles=1
Podm_Kc_TDS=0
Podm_Kc_NH4=0
Podm_Kc_SO4=0

[EDR]
Zapnuto=1
Q_Natoku=1
Kc_TDS_Natoku=2
Kc_NH4_Natoku=0.02
Kc_Diluatu=0.05
Zahusteni=5
Kc_Pokles=1
Podm_Kc_TDS=0
Podm_Kc_NH4=0
Podm_Kc_SO4=0

[BAR]
Zapnuto=1
Q_Natoku=2
Kc_TDS_Natoku=3
Kc_NH4_Natoku=0.05
Kc_Pokles=1
Podm_Kc_TDS=0
Podm_Kc_NH4=0
Podm_Kc_SO4=0

[VYP]
Zapnuto=1
Q_Natoku=1
Kc_TDS_Natoku=1
Kc_NH4_Natoku=0.008
Kc_Pokles=1
...
```

Tato sekce obsahuje nastavení pro jednotlivá vyluhovací pole (je jich 35), tedy údaje, zda se má z daného vyluhovacího pole čerpat, respektive kolik vrtů je na daném poli zapnuto, na které technologie se má čerpat, a nakonec je zde vydatnost čerpacích vrtů na daném vyluhovacím poli.

```
[VP_Parametry]
VP4-_N_Cerp=4
VP4-_CerpNaEDR=1
VP4-_CerpNaNDS=0
VP4-_CerpNaBAR=1
VP4-_CerpNaVYP=1
VP4-_Q_CerpVrt=0.25
VP7A_N_Cerp=0
...
```

Pak jsou zde opět čtyři sekce k jednotlivým technologiím, tentokrát se však jedná o nastavení parametrů předvýběru čerpání.

```
[PredvybCerp_NDS]
TDS_Zap=1
SO4_Zap=0
NH4_Zap=1
TDS_Min=4
SO4_Min=0
NH4_Min=0.1
TDS_Max=100
SO4_Max=10000
NH4_Max=10000

[PredvybCerp_EDR]
TDS_Zap=1
SO4_Zap=0
NH4_Zap=1
TDS_Min=1
SO4_Min=0
NH4_Min=0.01
TDS_Max=5
SO4_Max=10000
NH4_Max=10000

[PredvybCerp_BAR]
TDS_Zap=1
SO4_Zap=0
NH4_Zap=1
TDS_Min=1
SO4_Min=0
NH4_Min=0.03
TDS_Max=5
SO4_Max=10000
NH4_Max=10000

[PredvybCerp_VYP]
TDS_Zap=1
SO4_Zap=0
NH4_Zap=0
TDS_Min=0.5
SO4_Min=0
NH4_Min=0
TDS_Max=1.5
SO4_Max=10000
NH4_Max=10000
```

A sekce s optimalizačními parametry. Obsahuje informace o volbě účelové funkce, o omezení na počty nových vrtů a o volbě celočíselného řešení.

```
[Optimalizace]
UcFce=1
Max_NVrt_025=10
Max_NVrt_125=2
Max_NVrt_250=1
Celociselne_Reseni=1
```

Na konci souboru jsou ještě dvě sekce, ve kterých jsou uloženy výsledky modelu transportu a výsledky ekonomického modelu.

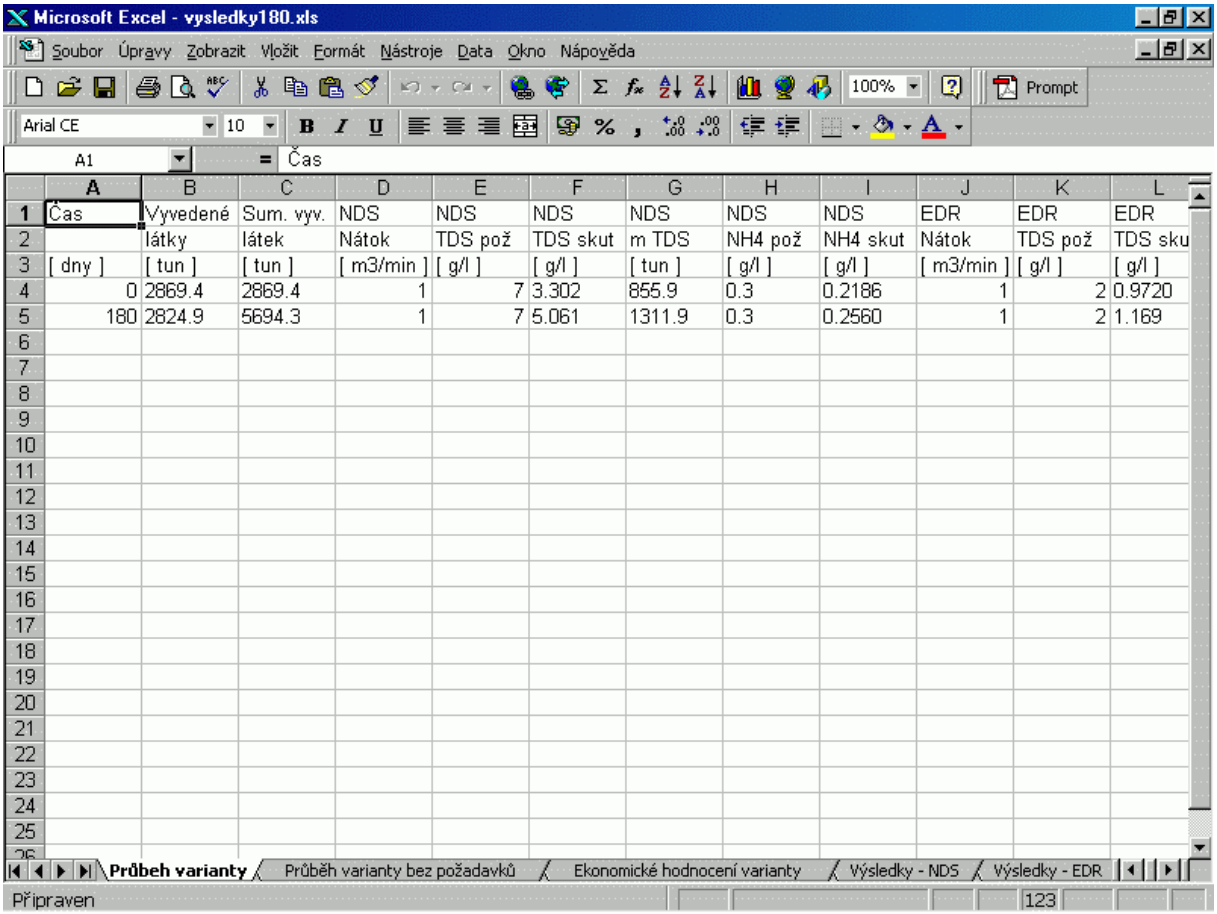
```
[Vysledky]
m_TDS_Krok=3133.613
kc_TDS_NDS=5.84255833066455
kc_SO4_NDS=4.34411522983073
kc_NH4_NDS=0.267131099784311
m_TDS_NDS=1514.21
m_SO4_NDS=1125.86
m_NH4_NDS=69.2321
objem_NDS=259.169
kc_TDS_EDR=1.49205828681217
kc_SO4_EDR=1.06743029540781
kc_NH4_EDR=0.0404396587976127
m_TDS_EDR=386.743
m_SO4_EDR=276.679
```

```
m_NH4_EDR=10.482
objem_EDR=259.201
kc_TDS_BAR=2.06236473688068
kc_SO4_BAR=1.53516358880783
kc_NH4_BAR=0.0425357904767048
m_TDS_BAR=1069.2
m_SO4_BAR=795.881
m_NH4_BAR=22.052
objem_BAR=518.434
kc_TDS_VYP=0.630630283062179
kc_SO4_VYP=0.421271522872211
kc_NH4_VYP=0.00957291831435836
m_TDS_VYP=163.46
m_SO4_VYP=109.194
m_NH4_VYP=2.48131
objem_VYP=259.201

[Ekonomie]
sum_o=71018.3471206089
NDS_o=25385.4240659112
EDR_o=16517.9588376976
BAR_o=13329.757622
VYP_o=9963.446595
vrt_o=5821.76
```

P2.2.3 Obsah souborů VYSLEDKY*.XLS

Jedná se o standardní soubor formátu XLS programů MS Excel. Na následujícím obrázku je vidět obsah souboru po otevření v Excelu (jedná se o starší soubor v současnosti se již ukládá do souboru více informací).

Obr. P2.1 – soubor *vysledky180.xls* otevřený v Excelu



	A	B	C	D	E	F	G	H	I	J	K	L
1	Čas	Vyvedené	Sum. vyv.	NDS	NDS	NDS	NDS	NDS	NDS	EDR	EDR	EDR
2		látky	látek	Nátok	TDS pož	TDS skut	m TDS	NH4 pož	NH4 skut	Nátok	TDS pož	TDS sku
3	[dny]	[tun]	[tun]	[m3/min]	[g/l]	[g/l]	[tun]	[g/l]	[g/l]	[m3/min]	[g/l]	[g/l]
4	0	2869.4	2869.4	1	7	3.302	855.9	0.3	0.2186	1	2	0.9720
5	180	2824.9	5694.3	1	7	5.061	1311.9	0.3	0.2560	1	2	1.169

P3 Zdrojové texty (stav ke dni 1. 4. 2003)

P3.1 Seznam souborů se zdrojovými texty

Jak spolu schematicky souvisejí jednotlivé soubory zdrojových textu (zejména soubory *h* a *cpp*) si můžete prohlédnout na obrázku P3.1 na následující straně

P3.1.1 Hlavní soubory projektu vytvářené vývojovým prostředím

Turon.bpr – soubor projektu – tento soubor lze otevřít v prostředí Borland C++Builder – přípona souboru *bpr* znamená *Builder PRoject*.

Turon.cpp – hlavní *cpp* soubor, obsahuje funkci *WinMain()*, příkazy, kterými se vkládají jednotlivé formuláře a moduly do projektu – *USERES()*, *USEFORM()*, *USEUNIT()*; v tomto souboru je také obsažen příkaz *USE()*, kterým je do projektu „vložen" soubor *Declar.h*. Pozor! Smysl této funkce je pouze v tom, že soubor *Declar.h* lze snadno otevřít ve vývojovém prostředí. Kompilátorem je však příkaz *USE()* ignorován (pomocí tohoto příkazu lze tedy do projektu vložit jakýkoliv textový soubor, který s projektem souvisí a je potřeba ho spolu s projektem editovat).

Turon.res – soubor s dalšími prostředky, které se využívají v projektu; typicky je v tomto souboru uložena ikona, informace o verzi programu, atd.

P3.1.2 Hlavní hlavičkový soubor projektu

Declar.h – nejdůležitější hlavičkový soubor tohoto projektu, obsahuje definice hlavních globálních proměnných, struktur, deklarace globálních funkcí atd.

P3.1.3 Hlavní moduly projektu, které implementují globální funkce

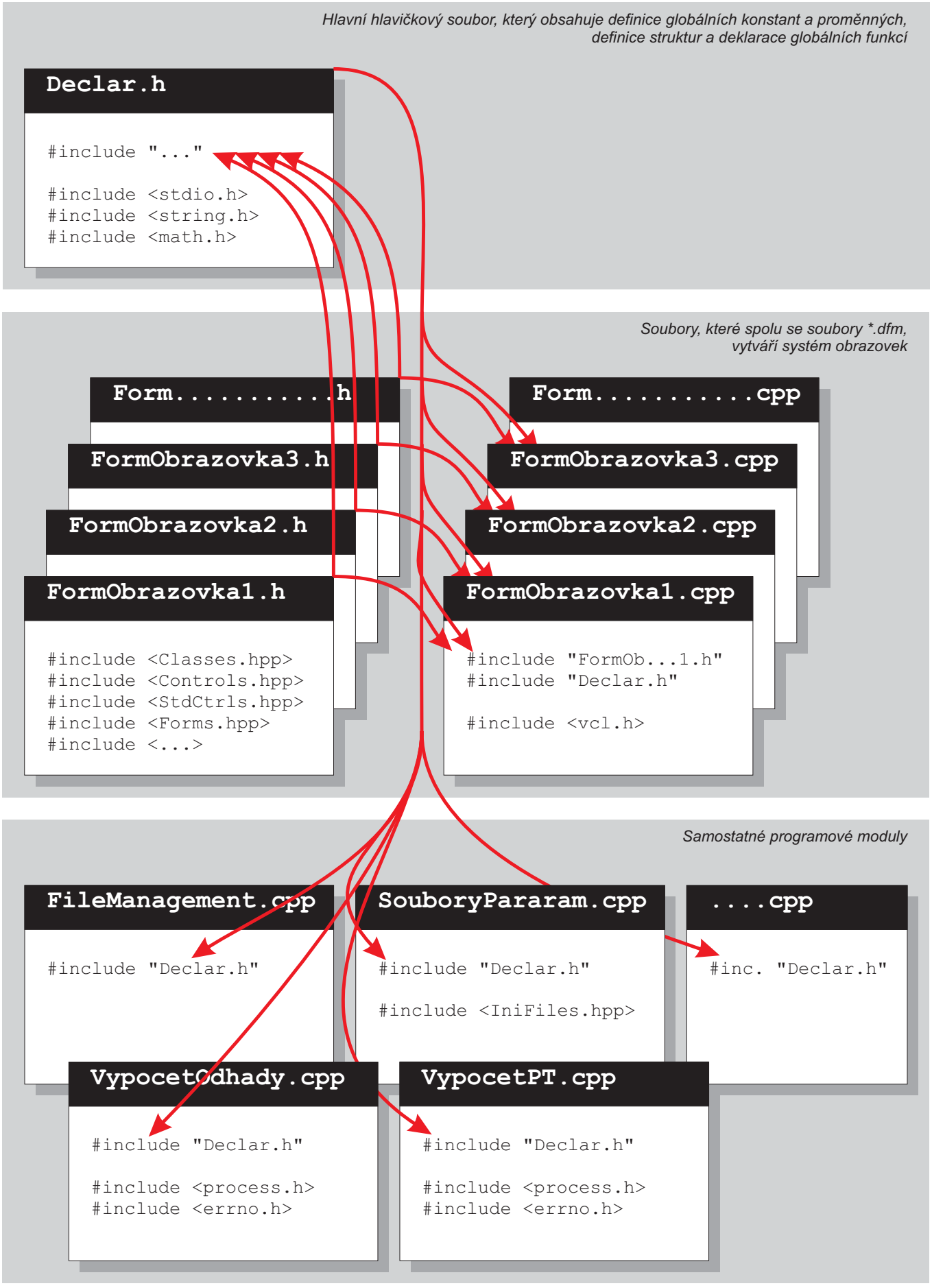
FileManagement.cpp – funkce pro práci se soubory
SouboryParametry.cpp – funkce pro manipulaci s parametry programu / práce s *ini* soubory
SouboryGwsView.cpp – funkce pro zobrazování dat v grafickém postprocesoru GwsView
VypocetOdhadu.cpp – funkce pro výpočet odhadů
PredvyberCerpani.cpp – funkce pro předvýběr čerpacích vrtů
SouboryXA.cpp – funkce, které připravují půdu pro program XA / spouštění programu XA
SouboryXAVysledky.cpp – funkce pro zpracování výsledků programu XA
SouboryPT.cpp – funkce, které vytváří soubory nutné pro výpočet proudění a transportu látek
VypocetPT.cpp – funkce, které spouští výpočet proudění a transportu látek
SouboryPTVysledky.cpp – funkce pro zpracování výsledků výpočtu proudění a transportu látek
EkonomickyModel.cpp – funkce, které implementují ekonomický model

P3.1.4 Obrazovky projektu

Všechny následující soubory jsou generovány automaticky vývojovým prostředím (mimo implementací v souborech *cpp*). Hlavičkový *h* soubor obsahuje definici struktury okna a jednu její instanci; *dfm* soubor obsahuje statické nastavení atributů jednotlivých komponent, použitých v tomto okně; *cpp* soubor obsahuje implementace metod, které ošetřují události, jež mohou být komponentami generovány.

FormLogo.h, .cpp, .dfm – logo programu, které se zobrazuje při jeho spuštění (normální okno)
FormHlavni.h, .cpp, .dfm – hlavní okno systému (okno typu *MDI Main / Client*)
FormObrazovka1.h, .cpp, .dfm
FormObrazovka2.h, .cpp, .dfm
FormObrazovka3.h, .cpp, .dfm
FormObrazovka4.h, .cpp, .dfm
FormObrazovka5.h, .cpp, .dfm
FormObrazovka5s.h, .cpp, .dfm
FormObrazovka6.h, .cpp, .dfm
FormObrazovka7.h, .cpp, .dfm
FormObrazovka8.h, .cpp, .dfm
FormObrazovka9.h, .cpp, .dfm
– jednotlivé obrazovky systému (okna typu *MDI Child*); zobrazují se v hlavním okně
FormDebug.h, .cpp, .dfm – pomocná obrazovka; slouží výhradně pro vývoj systému (klasické okno)

Obr. P3.1 - na obrázku je patrné, jak spolu souvisí jednotlivé soubory *.h a *.cpp



P3.2 Některé soubory se zdrojovými texty – výpis

Následují výpisy souborů se zdrojovými texty; zejména se jedná o hlavní hlavičkový soubor a vybrané soubory, v kterých jsou implementovány nejdůležitější globální funkce. Nelze zde pochopitelně vypsat všechny zdrojové texty, ty jsou k dispozici na instalačním CD systému TURON 2003.

Posupně jsou zde vypsány následující soubory:

Declar.h – hlavní hlavičkový soubor
VypocetPT.cpp – výpočet proudění a transportu

Ukázky souborů, které jsou generovány automaticky vývojovým prostředím:

Turon.cpp – tento soubor obsahuje funkci *WinMain()*
FormHlavni.h
FormHlavni.cpp – zde je vidět mechanismus nastavení desetinného oddělovače

P3.2.1 Soubor Declar.h

P3.2.2 Soubor VypocetPT.cpp

P3.2.3 Soubory generované automaticky Turon.cpp

P3.2.4 Soubory generované automaticky FormHlavni.h

P3.2.5 Soubory generované automaticky FormHlavni.cpp

(Všechny zdrojové texty následují.)

```
/*****************************************************************************/
/*  Delcar.h - hlavni definice + deklarace                                   */
/*****************************************************************************/
/*                                                                           */
/*  definice konstant                                                        */
/*  definice globalnich promennych                                           */
/*  definice globalnich struktur                                             */
/*                                                                           */
/*  deklarace globalnich funkci                                              */
/*                                                                           */
/*****************************************************************************/
/*  prepracovano     - 29. 12. 2002                                          */
/*****************************************************************************/

#ifdef MAIN
   #define EX
#else
   #define EX extern
#endif

// MAIN je definovane v souboru FormHlavni.cpp

/*****************************************************************************/
/*  OBRAZOVKY                                                                */
/*****************************************************************************/

#include "FormHlavni.h"

#include "FormObrazovka1.h"
#include "FormObrazovka2.h"
#include "FormObrazovka3.h"
#include "FormObrazovka4.h"
#include "FormObrazovka5.h"
#include "FormObrazovka5s.h"
#include "FormObrazovka6.h"
#include "FormObrazovka7.h"
#include "FormObrazovka8.h"
#include "FormObrazovka9.h"

#include "FormLogo.h"
#include "FormDebug.h"

/*****************************************************************************/
/*  STANDARDNI KNIHOVNY                                                      */
/*****************************************************************************/

#include <stdio.h>

/*****************************************************************************/
/*  DEFINICE GLOBALNICH KONSTANT                                             */
/*****************************************************************************/

#define N_TECH        4           // pocet pouzitych technologii
#define N_SSLO        3           // pocet sledovanych slozek
#define N_SSLOTS3     4           // pocet sledovanych slozek v souborech ts3
#define N_VP         35           // pocet vyluhovacich poli

/*
 v programu je snaha dodrzovat tuto konvenci pro indexovaci promenne:
 it - promenna pro indexovani technologii
 is - promenna pro indexovani sledovanych slozek
 ip - promenna pro indexovani vyluhovacich poli
*/

#define iNDS          0           // pro snadnejsi indexovani technologii
#define iEDR          1
#define iVYP          2
#define iBAR          3
#define iNOTECH      -1

#define iTDS          0           // pro snadnejsi indexovani slozek
#define iSO4          1
#define iNH4          2

#define iTDSTS3       0           // idx sloupcu v ts3 pro sledovane slozky
#define iSO4TS3       1           // POZOR! v ts3 souborech je navic berilium
#define iNH4TS3       2
#define iBERTS3       3
```

```
#define MAXVRT     6000           // max. pocet vrtu - struktura vrt

#define MAXELMOZN  6725           // max. oznaceni elementu + 1 (viz *.ste)
#define MAXUZLOZN  3778           // max. oznaceni uzlu + 1 (viz *.stu)
#define iX            0           // souradnice v poli uzlu
#define iY            1

#define ZERO       1e-7           // technicka nula

#define FAST       true           // pro vytvareni souboru *ts3 / *_slow.ts3
#define SLOW      false

/* konstanty pouzivane pri hledani souboru / adresaru */

#define ATTR_RHSA  0x27           // atributy rhsa (u souboru / adresare)
#define ATTR_D     0x10           // atribut d (pouze u adresare)

#define ATTR_FILE  0x27           // atr. souboru   ( == ATTR_RHSA )
#define ATTR_DIR   0x37           // atr. adresare  ( == ATTR_RHSA + ATTR_D )

/*
 soubor i adresar muze mit libovolny z techto atributu (volitelne atributy)
 r (read only) == 0x01
 h (hidden)    == 0x02
 s (system)    == 0x04
 a (archive)   == 0x20

 adresar ma navic jeste jeden atribut (poviny atribut)
 d (directory) == 0x10

 z toho jasne vyplyvaji hodnoty predchozich konstant
*/

/*****************************************************************************/
/*  DEFINICE SPECIALNICH GLOBALNICH PROMENNYCH                               */
/*****************************************************************************/

EX bool SKIP_ALL;

// pokud je SKIP_ALL == true, preskoci se z prvni obrazovky rovnou
// az na devatou a naopak; ini soubory se pri tom nijak nemodifikuji
// vhodne pro ladeni posledni obrazovky ... plesinger

EX bool SKIP_XLS;

// pokud je SKIP_XLS == true, nevytvari se na konci xls soubory
// to je z toho duvodu, ze metody SaveToXLS a SaveToXLSSheet nekdy
// zpusobi nesmyslnou chybu (chyba vznika pravdepodobne v zavislosti
// na verzi ms windows, ms office a ms internet explorer) ... plesinger

/*****************************************************************************/
/*  DEFINICE GLOBALNICH PROMENNYCH                                           */
/*****************************************************************************/

EX String dir_progs;              // cesta k programum
EX String dir_sit;                // cesta k siti
EX String jmn_sit;                // jmeno site
EX String dir_base_var;           // cesta k adresari se vsema variantama
EX String dir_var;                // cesta k adresari aktualni varianty
EX String jmn_var;                // jmeno pocitane varianty
EX String dir_tmp;                // cesta k temp adresari na docasne soubory
EX bool   del_tmp;                // true == mazat / false == nemazat adr. tmp
EX String ini_par;                // jmeno ini souboru s parametry

EX int rok_od;                    // cas zahajeni kroku vypoctu
EX int rok_do;                    // cas ukonceni kroku vypoctu

EX int N_Vrt;                     // pocet vrtu
EX int N_Vrt_Tech [ N_TECH ];     // pocet vrtu na danou technologii

EX int Max_NVrt_025;              // maximalni pocet novych vrtu  25 l/min
EX int Max_NVrt_125;              // maximalni pocet novych vrtu 125 l/min
EX int Max_NVrt_250;              // maximalni pocet novych vrtu 250 l/min

EX int Ucfce;                     // kod pro volbu ucelove funkce

EX int Celociselne_Reseni;        // hledat reseni celociselne
```

```
/*************************************************************************/
/*  DEFINICE STRUKTUR                                                    */
/*************************************************************************/

/*************************************************************************/
/*  struktura: sit                                                       */
/*************************************************************************/

struct S_Sit                                  // obsahuje ruzna prevodni pole
{
 int   elmozn_2_elmidx [ MAXELMOZN ];         // ozn.elm. > idx.elm. v poli VRT
 int   elmozn_2_uzlozn [ MAXELMOZN ] [ 3 ]; // ozn.elm. > 3x ozn.uzl. (*.ste)
 float uzlozn_2_sourdn [ MAXUZLOZN ] [ 2 ]; // ozn.uzl. > 2x sourdnc. (*.stu)
                                              // poradi souradnic: x, y
};

/*************************************************************************/
/*  struktura: technologie                                               */
/*************************************************************************/

struct S_Technologie
{
 String Jmeno;          // jmeno NDS, EDR, ...
 bool   Zap;            // zapnuto - (ne)pouzivam tuto technologii
 double QNatok;         // pozadovany objem nátoku na techn. [ m3 / min ]
 double QNabidka;       // mozny objem natoku na technologii [ m3 / min ]
 double kcTDS_In;       // zadaná koncentrace TDS nátoku
 double kcNH4_In;       // zadaná koncentrace NH4 nátoku
 double kcPokles;       // koeficient poklesu koncentraci
 bool   podmKcTDS;      // bude pozadovno dodrzeni koncentrace TDS
 bool   podmKcSO4;      // bude pozadovno dodrzeni koncentrace SO4
 bool   podmKcNH4;      // bude pozadovno dodrzeni koncentrace NH4
 double kcDiluatu;      // koncentrace TDS v diluatu (pouze EDR)
 double zahusteni;      // hodnota zahusteni koncentratu (pouze EDR)

 double cas                 [ 20 ]; // hodnoty casu pro casove rady    (*.DF0)
 double radaKc   [ N_SSLO ] [ 20 ]; // casova rada koncentraci slozek  (*.DF0)
 double radaQ    [ N_SSLO ] [ 20 ]; // casova rada mnoz. slozek        (*.DF2)
 double radaQSum [ N_SSLO ] [ 20 ]; // casova rada mnoz. slozek - suma (*.DF1)
 double radaObj             [ 20 ]; // casova rada objemu cerp.        (*.DF2)
 double radaObjSum          [ 20 ]; // casova rada objemu cerp. - suma (*.DF1)
};

/*************************************************************************/
/*  struktura: sledovana slozka (pridana zatim jen pro uplnost)          */
/*             slouzi zatim jen pro zapisovani do ini souboru            */
/*************************************************************************/

struct S_Sled_Slozka
{
 String Jmeno;       // jmeno TDS, SO4, ...
};

/*************************************************************************/
/*  struktura: vyluhovaci pole                                           */
/*************************************************************************/

struct S_VP
{
 String Jmeno;       // jmeno VP4-, VP7A, ...
 int    N_Cerp;      // pocet vrtu, ze kterych se v poli cerpa
 bool   CerpNDS;     //  == true, kdyz se z pole cerpa na NDS
 bool   CerpEDR;     //  == true, kdyz se z pole cerpa na EDR
 bool   CerpVYP;     //  == true, kdyz se z pole cerpa na VYP
 bool   CerpBAR;     //  == true, kdyz se z pole cerpa na BAR
 double Q_CerpVrt;   // intenzita cerpani z jednoho vrtu
};
```

```
/*****************************************************************************/
/*  struktura: vrt                                                           */
/*****************************************************************************/

struct S_Vrt
{
 int    Elm;                     // cislo elementu ve kterem se vrt nachazi
 bool   Exist;                   // existuje jiz vrt
 double Q;                       // intenzita cerpani z 1 vrtu
 double kcCerp        [ N_SSLO ]; // koncentrace sledovanych slozek
 String Lokal;                   // lokalizace elm.; oznaceni VP kt. patri
 int    IdxVP;                   // Index vyluhovaciho pole kteremu patri
 bool   TechLokal     [ N_TECH ]; // mozno cerpat na technologii?
                                 // urcuje lokalizace (idxtech.)
 bool   PodmPredvyber [ N_TECH ]; // splnuje vrt podm. predvyberu? (idxtech.)
 double QDiluat;
 double kcDiluat      [ N_SSLO ]; // koncentrace diluatu u sledovanych slozek
 double QSliv;
 double kcSliv        [ N_SSLO ];
 double QKonc;
 double kcKonc        [ N_SSLO ];
 double vaha;                    // vaha (hodnota v rozmezi <0,1>) s jakou
                                 //  byl element vybran lin. modelem
                                 //  0-nebyl vybran, 1- byl vybran
 int    Odvrst;                  // od vrstvy, ve kterych je vrt otevren
 int    Dovrst;                  // do vrstvy, ve kterych je vrt otevren
 int    Tech;                    // na kterou technologii byl vrt vybran
 int    Zarad;                   // na kterou technologii ma byt vrt
                                 //  pokud je to apriory vyzadovano
 bool   povolen;                 // povolen?
};

/*****************************************************************************/
/*  struktura: podminky predvyberu cerpani                                   */
/*****************************************************************************/

struct S_Podminky
{
 bool   Povoleno;          // je technologie povolena?
 bool   Zap [ N_SSLO ];    // ma vyznam slozky - TDS, SO4, NH4
 double Min [ N_SSLO ];    // dtto
 double Max [ N_SSLO ];    // dtto
};

/*****************************************************************************/
/*  struktura: vysledky modelu                                               */
/*****************************************************************************/

struct S_Vysledky
{
 double Konc [ N_SSLO ] [ N_TECH ]; // konc. slozek   [ kg / m3 ] = [ g / l ]
 double Hmot [ N_SSLO ] [ N_TECH ]; // hmotn. slozek  [ kg ]
 double Hmot_TDS_Krok;              // hmotn. latek v danem kroku [ kg ]
 double Objm            [ N_TECH ]; // objemy         [ m3 ]
};

/*****************************************************************************/
/*  DEFINICE GLOBALNICH PROMENNYCH - STRUKTUR                                */
/*****************************************************************************/

EX struct S_Sit           Sit;
EX struct S_Technologie   NDS , EDR , BAR , VYP;
EX struct S_Technologie * Technologie [ N_TECH ];
EX struct S_Sled_Slozka   Sled_Slozky [ N_SSLO ];
EX struct S_VP            VP          [ N_VP ];
EX struct S_Vrt           VRT         [ MAXVRT ];
EX struct S_Podminky      PodmCerp    [ N_TECH ];
EX struct S_Vysledky      Vysledky;
```

```
/*****************************************************************************/
/*  DEFINICE STRUKTUR                                                        */
/*  EKONOMICKE PARAMETRY / EKONOMICKY MODEL                                  */
/*****************************************************************************/

/*****************************************************************************/
/*  struktura: ekonomicke parametry                                          */
/*****************************************************************************/

struct S_EParametry
{
 /* promenne naklady (cerpani cerpadla / cerpani airlift / transport) */

 double prom_merna_sp_en_cerp;        // [ kWh / m3 ]
 double prom_merna_sp_en_airlift;     // [ kWh / m3 ]
 double prom_merna_sp_en_transport;   // [ kWh / m3 ]
 double prom_material;                // [ Kc / m3 ]
 double prom_opravy_udrzba;           // [ Kc / m3 ]
 double prom_laboratore;              // [ Kc / m3 ]
 double prom_sluzby;                  // [ Kc / m3 ]
 double prom_mzdy_vcetne_SZ;          // [ Kc / m3 ]
 double prom_celkem;                  // [ Kc / m3 ]
 double prom_ZR;                      // [ % ]

 /* fixni naklady (cerpani) */

 double fixn_vyrobni_rezie;           // [ tis. Kc / rok ]
 double fixn_spravni_rezie;           // [ tis. Kc / rok ]
 double fixn_ostatni_bez_odpisu;      // [ tis. Kc / rok ]
 double fixn_celkem;                  // [ tis. Kc / rok ]

 /* porizovaci naklady (cerpani) */

 double cena_cerpadla_4in;       // [ Kc ]
 double cena_cerpadla_6in;       // [ Kc ]
 double vystroj_cerpadl_vrtu;    // [ Kc ]
 double vrt_liftovy_stihly;      // [ Kc / m ]
 double vrt_cerpaci_plnostihly;  // [ Kc / m ]
 double vrt_cerpaci_siroky;      // [ Kc / m ]

 /* doplnkove udaje (cerpani) */

 double zivotnost_cerpadla;      // [ pocet let ]
 double prumerna_hloubka_vrtu;   // [ m ]
 double cena_el_energie;         // [ Kc / kWh ]
};

/*****************************************************************************/
/*  struktura: ekonomicke parametry NDS                                      */
/*****************************************************************************/

struct S_EParametry_NDS
{
 /* NDS promenne naklady */

 double prom_merna_sp_en;             // [ kWh / m3 ]
 double prom_merna_sp_en_kaly;        // [ kWh / m3 kalu ]
 double prom_material;                // [ Kc / m3 ]
 double prom_opravy_udrzba;           // [ Kc / m3 ]
 double prom_laboratore;              // [ Kc / m3 ]
 double prom_sluzby;                  // [ Kc / m3 ]
 double prom_mzdy_vcetne_SZ;          // [ Kc / m3 ]
 double prom_celkem;                  // [ Kc / m3 ]
 double prom_material_kaly;           // [ Kc / m3 kalu ]
 double prom_opravy_udrzba_kaly;      // [ Kc / m3 kalu ]
 double prom_doprava_kaly;            // [ Kc / m3 kalu ]
 double prom_laboratore_kaly;         // [ Kc / m3 kalu ]
 double prom_sluzby_kaly;             // [ Kc / m3 kalu ]
 double prom_mzdy_vcetne_SZ_kaly;     // [ Kc / m3 kalu ]
 double prom_celkem_kaly;             // [ Kc / m3 kalu ]
 double prom_ZR;                      // [ % ]

 /* NDS fixni naklady */

 double fixn_celkem;                  // [ tis. Kc / rok ]
 double fixn_podminene_naklady;       // [ Kc / m3 ]
```

```
 /* NDS ceny chamikalii */

 double cena_vapno_CaO;               // [ Kc / kg ]
 double cena_chlor_Cl2;               // [ Kc / kg ]
 double cena_siran_sodny_Na2SO4;      // [ Kc / kg ]
 double cena_chlorid_barnaty_BaCl2;   // [ Kc / kg ]
};

/*****************************************************************************/
/*  struktura: ekonomicke parametry EDR                                      */
/*****************************************************************************/

struct S_EParametry_EDR
{
 /* EDR promenne naklady */

 double prom_merna_sp_en;             // [ kWh / m3 ]
 double prom_material;                // [ Kc / m3 ]
 double prom_opravy_udrzba;           // [ Kc / m3 ]
 double prom_ostatni;                 // [ Kc / m3 ]
 double prom_celkem;                  // [ Kc / m3 ]
 double prom_celkem_odparka;          // [ Kc / m3 koncentratu ]
 double prom_ZR;                      // [ % ]

 /* EDR fixni naklady */

 double fixn_vyrobni_rezie;           // [ tis. Kc / rok ]
 double fixn_spravni_rezie;           // [ tis. Kc / rok ]
 double fixn_ostatni_bez_odpisu;      // [ tis. Kc / rok ]
 double fixn_celkem;                  // [ tis. Kc / rok ]
};

/*****************************************************************************/
/*  struktura: ekonomicke parametry BAR                                      */
/*****************************************************************************/

struct S_EParametry_BAR
{
 /* BAR promenne naklady */

 double merna_sp_en;             // [ kWh / m3 ]
};

/*****************************************************************************/
/*  struktura: ekonomicke parametry VYP                                      */
/*****************************************************************************/

struct S_EParametry_VYP
{
 /* VYP promenne naklady */

 double merna_sp_en;             // [ kWh / m3 ]
};

/*****************************************************************************/
/*  struktura: ekonomicke hodnoceni kroku varianty                           */
/*****************************************************************************/

struct S_EHodnoceni
{
 double tch_o [ N_TECH ];        // technologie - okamzite naklady
 double vrt_o;                   // nove vrty   - okamzite naklady
 double sum_o;                   // celkem      - okamzite naklady
};

/*****************************************************************************/
/*  DEFINICE GLOBALNICH PROMENNYCH - STRUKTUR                                */
/*  EKONOMICKE PARAMETRY / EKONOMICKY MODEL                                  */
/*****************************************************************************/

EX struct S_EParametry     EPar;
EX struct S_EParametry_NDS EParNDS;
EX struct S_EParametry_EDR EParEDR;
EX struct S_EParametry_BAR EParBAR;
EX struct S_EParametry_VYP EParVYP;
EX struct S_EHodnoceni     EHodnoceni;
```

```
/*************************************************************************/
/*  DEKLARACE GLOBALNICH FUNKCI                                          */
/*************************************************************************/
/*                                                                       */
/*  funkce jsou srovnane podle modulu, v kterych jsou implementovane     */
/*                                                                       */
/*  system oznaceni jednotlivych modulu je nasledujici:                  */
/*                                                                       */
/*  Soubory*.cpp    - zpracovava / zapisuje / cte nejake soubory         */
/*  Vypocet*.cpp    - spousti nejaky vypocet pomoci externiho programu   */
/*                                                                       */
/*  *XA*.cpp        - moduly tykajici se optimalizacniho softwaru XA     */
/*  *PT*.cpp        - moduly tykajici se vypoctu proudeni a transportu   */
/*                                                                       */
/*  (drive byly vsechny tyto moduly v jednom souboru, zejmena pro        */
/*   prehlednost jsem je rozhodil do nekolika mensich, a logicky         */
/*   vice ucelenych souboru)                                             */
/*                                                                       */
/*************************************************************************/

/*************************************************************************/
/*  FileManagement.cpp - sprava souboru                                  */
/*************************************************************************/

bool f_del ( AnsiString );                // smaze soubor
bool f_cpy ( AnsiString , AnsiString );   // zkopiruje soubor
bool f_mov ( AnsiString , AnsiString );   // presune / prejmenuje soubor
void d_clean ( AnsiString );              // vymaze obsah zadaneho adresare
void d_clean_temp ();                     // vymaze obsah adresare dir_tmp
void f_backup ( AnsiString );             // vytvori zalozni kopii souboru
void f_backup_del_all ( AnsiString );     // smaze vsechny zal. kopie v adr.

bool existuji_vsechy_soubory_slozky ();   // zkontroluje, jestli je vsechno
                                          // tam, kde to ma byt

/*************************************************************************/
/*  SouboryParametry.cpp - prirazeni jmen, prace s ini soubory           */
/*************************************************************************/

// prvni tri funkce pouze pojmenuji technologie / slozky / vyluhovaci pole
// tato jmena jsou vicemene potreba pouze jen manipulaci s ini soubory,
// proto jsou tyto funkce pohromade

void priradit_jmena_technologie ();       // priradi jmena technologiim
void priradit_jmena_slozky ();            // priradi jmena slozkam
void priradit_jmena_VP ();                // priradi jmena vyluhovacim polim
void nacti_globalni_parametry ();         // nacte parametry z turon.ini
void ini_parametry_nacist ( AnsiString ); // nacte parametry z ini souboru
void ini_parametry_ulozit ( AnsiString ); // ulozi parametry do ini souboru
void zmenit_parametry_dalsi_krok ();      // zmeni parametry pro dalsi krok
void zmenit_parametry_opakovat_krok ();   // zmeni parametry pro opak. kroku

void ini_vysledky_ulozit ( AnsiString );  // ulozi vysledky do ini souboru
void ini_ekonomie_ulozit ( AnsiString );  // ulozi ekonom. hodnoceni do ini s.

/*************************************************************************/
/*  SouboryGwsView.cpp - vytvareni souboru pro gwsview; zobrazeni dat    */
/*************************************************************************/

// zobrazeni dat v gwsview

void gwsview_odhady_1v_zobrazit ();           // odhady       - 1v sit
void gwsview_vybrane_vrty_zobrazit ();        // vybrane vrty - vv sit
void gwsview_vybrane_vrty_1v_zobrazit ();     // vybrane vrty - 1v sit

// vytvareni souboru *.stv a *.zpv pro zobrazeni dat v gwsview

void gwsview_odhady_1v_stv ();                // odhady       - 1v sit
void gwsview_odhady_1v_zpv ();
void gwsview_vybrane_vrty_stv ();             // vybrane vrty - vv sit
void gwsview_vybrane_vrty_zpv ();
void gwsview_vybrane_vrty_1v_stv ();          // vybrane vrty - 1v sit
void gwsview_vybrane_vrty_1v_zpv ();

void nacti_data_site ();                  // nacte idx m-elementu, m-uzlu
void vytvor_seznam_vybranych_vrtu ();     // vytvoreni seznamu vybranych vrtu
```

```
/*************************************************************************/
/*  VypocetOdhadu.cpp - vypocet odhadu koncentraci; tur_odhad            */
/*************************************************************************/

int  aktualni_stav_odhadu ();            // zjisti aktualni stav odhadu
bool vypocet_odhadu_koncentraci ();      // spusti ext. prg. tur_odhad

/*************************************************************************/
/*  PredvyberCerpani.cpp                                                 */
/*************************************************************************/

// nacte soubor lokal.elm + pomocna funkce

void cti_soubor_lokal_elm ();
int  index_VP ( char * );

// vyplni pole Sit.elmozn_2_elmidx

void vyplnit_elmozn_2_elmidx ();

// cte odhady koncetraci v danem multielmentu

void cti_koncentrace ();

// na ktere technologie muzu z elemetu cerpat

void moznost_cerpani_z_elementu ();

// funkce pro predvyber jendnotlivych vrtu na jednotlive technologie

void predvyber          ();
void predvyber_tech     ( int );
void predvyber_tech_VRT ( int , int );
int  pocet_VRT_na_tech  ( int );
bool VRT_povolen        ( int );

/*************************************************************************/
/*  SouboryXA.cpp - vytvareni souboru nutnych pro program XA, spust. XA  */
/*************************************************************************/

int  najdi_optimalni_reseni ();              // spusti program XA

// tyto funkce jsou jiz vyuzivany predchozi funkci

void pis_soubor_mpt_clp ();                  // ridici soubor programu XA
void pis_soubor_turon_mps ();                // soubor se vstupy prg. XA

// pomocne fce pro pis_soubor_turon_mps ()

void turon_mps_hlavicka    ( FILE * );
void turon_mps_omezeni_VP  ( FILE * );
void turon_mps_omezeni_elm ( FILE * );
void turon_mps_NDS ( struct S_Vrt * , FILE * );
void turon_mps_EDR ( struct S_Vrt * , FILE * );
void turon_mps_BAR ( struct S_Vrt * , FILE * );
void turon_mps_VYP ( struct S_Vrt * , FILE * );
void turon_mps_RHS_technologie ( FILE * );
void turon_mps_RHS_VP          ( FILE * );
void turon_mps_RHS_elm         ( FILE * );
void turon_mps_range           ( FILE * );
void turon_mps_bound_vrt ( struct S_Vrt * , FILE * );

// pomocna fce pro turon_mps_NDS / EDR / BAR / VYP ()

void turon_mps_ucelova_fce ( struct S_Vrt * , FILE * , String );

// pomocna fce pro turon_mps_EDR ()

void koncentrat_diluat ( S_Vrt * );
```

```
/*************************************************************************/
/*  SouboryXAVysledky.cpp - zpracovani vysledku linearniho modelu        */
/*************************************************************************/

// zpracuje soubor turon.xa a vyplni vsechny tabulky na Obrazovka7

void zpracuj_soubor_turon_xa_a_vypln_tabulky ();

// vyplni zahlavi vsech tabulek

void zahlavi_tabulka_omez ();
void zahlavi_tabulka_vrty ();
void zahlavi_tabulka_vrty_prehled ();
void zahlavi_tabulka_pole ();
void zahlavi_tabulka_roztoky ();
void zahlavi_tabulka_latky ();

// zpracovani souboru, vyplni tabulky GridOmez a GridVrty + pomocne fce

void zpracuj_soubor_turon_xa ();
bool prazdny_radek      ( char * );
void zpracuj_radek_omez ( char * , int );
void zpracuj_radek_vrty ( char * , int );

// vyplni tabulku GridVrtyPrehled

void vyplnit_tabulka_vrty_prehled ();

// vyplni tabulku GridPole + pomocna fce

void vyplnit_tabulka_pole ();
void vyplnit_tabulka_pole_tech ( int , int * );

// vyplni tabulku GridRoztoky + pomocne fce

void vyplnit_tabulka_roztoky ();
void vyplnit_tabulka_roztoky_NDS ();
void vyplnit_tabulka_roztoky_EDR ();
void vyplnit_tabulka_roztoky_BAR ();
void vyplnit_tabulka_roztoky_VYP ();

// vyplni tabulku GridLatky

void vyplnit_tabulka_latky ();

/*************************************************************************/

void pis_soubor_oke ();                       // okrajova podminka

/*************************************************************************/

// seznam existujicich vrtu

void sexvrt_nacist_ze_souboru ();             // nacteni seznamu
void sexvrt_pro_pristi_obdobi ();             // aktualizace seznamu
                                              // pro dalsi krok vypoctu

/*************************************************************************/
/*  SouboryPT.cpp - vytvareni souboru pro vypocet proudeni a tranportu   */
/*************************************************************************/

// funkce vytvareji soubory mmf, poc, pop potrebne pro vypocet PT

bool pis_soubor_mmf ( AnsiString );
bool pis_soubor_poc ( AnsiString , bool );
bool pis_soubor_pop ( AnsiString , bool );

/*************************************************************************/
/*  VypocetPT.cpp - vypocet proudeni a transportu; gen_flow, gen_tran    */
/*************************************************************************/

bool vypocet_proudeni ();                // spusti ext. prg. gen_flow
bool vypocet_transportu_latek ();        // spusti ext. prg. gen_tran
```

```
/*************************************************************************/
/*  SouboryPTVysledky.cpp - zpracovani vysledku vypoctu PT               */
/*************************************************************************/

// zpracuje a zkopiruje vypoctene soubory ts3 a dfN do dir_var

bool zpracuj_vysledky_vypoctu ();

// zpracuje a zkopiruje vypoctene soubory ts3 do dir_var

bool zpracuj_soubor_ts3 ( bool );

// zpracuje vypocnete soubory dfN

bool zpracuj_soubor_dfN ( int );

// pomocne funkce ke zpracovani souboru dfN

bool zpracuj_hlavicku_dfN ( FILE * );
bool zpracuj_radek_dfN ( int , int , char * );
bool zpracuj_radek_df0 ( int , char * );
bool zpracuj_radek_df1 ( int , char * );
bool zpracuj_radek_df2 ( int , char * );
void chyba_souboru_dfN ( int , char * );

/*************************************************************************/
/*  EkonomickyModel.cpp - implementace ekonomickeho modelu               */
/*************************************************************************/

bool nacti_ekonomicke_parametry ();  // nacte parametry z ini souboru
                                     // (vyplni struktury EPar)
bool zkontroluj_parametry ();        // zkontroluje parametry
                                     // (spousti se automaticky, interni fce)
void urci_naklady ();                // urci naklady
                                     // (vyplni strukturu EHondoceni)

// nasledujici fce jsou interni fce ekonomickeho modelu

double naklady_na_nove_vrty   ();                    // [ tis. Kc ]

double naklady_cerpani        ( double , int );      // [ Kc ]
double naklady_transport      ( double );            // [ Kc ]
double naklady_chemikalie_NDS ( double );            // [ Kc ]

double naklady_zpracovani_NDS ( double , double );   // [ tis. Kc ]
double naklady_zpracovani_EDR ( double , double );   // [ tis. Kc ]
double naklady_zpracovani_BAR ( double );            // [ tis. Kc ]
double naklady_zpracovani_VYP ( double );            // [ tis. Kc ]

/*************************************************************************/
```

```
/*****************************************************************************/
/*  VypocetPT.cpp - vypocet proudeni a transportu; gen_flow, gen_tran        */
/*****************************************************************************/
/*                                                                           */
/*  vypocet_proudeni ()                                                      */
/*  vypocet_transportu_latek ()                                              */
/*                                                                           */
/*****************************************************************************/
/*  prepracovano     -  8. 12. 2002                                          */
/*****************************************************************************/

#include <process.h>
#include <errno.h>
#include "Declar.h"

/*****************************************************************************/
/*  spusti externi program gen_flow, pro vypocet proudeni                    */
/*****************************************************************************/

bool vypocet_proudeni ()
{
 char prg [ 1024 ];
 char arg [ 1024 ];

 int ret;
 extern int errno;

 if ( ! SetCurrentDir ( dir_tmp ) )
 {
  ShowMessage ( "CHYBA: Nelze nastavit pracovní asdresáø: " + dir_tmp );
  return false;
 }

 if ( ! pis_soubor_mmf ( dir_tmp + "turon.mmf" ) )
 {
  ShowMessage ( "CHYBA: Chyba pøi vytváøení øídícího souboru turon.mmf" );
  return false;
 }

 f_cpy ( dir_var + "oke" + rok_od + ".oke" , dir_tmp + jmn_sit + ".oke" );

 sprintf ( prg , "%sgen_flow.exe" , dir_progs.c_str () );
 sprintf ( arg , "%sturon.mmf"    , dir_tmp.c_str ()   );

 ret = spawnl ( P_WAIT , prg , prg , arg , NULL );

 /* systemova chyba pri spousteni */
 if ( ret == -1 )
 {
  switch ( errno )
  {
   case E2BIG   : ShowMessage ( "ERROR SPAWNL: Arg list too long"    ); break;
   case EINVAL  : ShowMessage ( "ERROR SPAWNL: Invalid argument"     ); break;
   case ENOENT  : ShowMessage ( "ERROR SPAWNL: File name not found" ); break;
   case ENOEXEC : ShowMessage ( "ERROR SPAWNL: Exec format error"    ); break;
   case ENOMEM  : ShowMessage ( "ERROR SPAWNL: Not enough memory"    ); break;
   default      : ShowMessage ( "ERROR SPAWNL: Errno = " + errno     );
  }
  return false;
 }

 /* behova chyba programu */
 if ( ret !=  0 )
 {
  ShowMessage ( "ERROR SPAWNL: Runtime error of GEN_FLOW; code = " + ret );
  return false;
 }

 if ( ! FileExists ( dir_var + "00-00000.hdm" ) )
 {
  ShowMessage ( "CHYBA: Nepodaøilo se vytvoøit soubory hydrodynamiky hdm." );
  return false;
 }

 return true;
}
```

```
/*****************************************************************************/
/*  spusti externi program gen_tran, pro vypocet transportu latek            */
/*****************************************************************************/

bool vypocet_transportu_latek ()
{
 char prg [ 1024 ];
 char arg [ 1024 ];

 int ret;
 extern int errno;

 if ( ! SetCurrentDir ( dir_tmp ) )
 {
  ShowMessage ( "CHYBA: Nelze nastavit pracovní asdresáø: " + dir_tmp );
  return false;
 }

 if ( ! pis_soubor_poc ( dir_tmp + "turon.poc"      , FAST ) )
 {
  ShowMessage ( "CHYBA: Chyba pøi vytváøení souboru turon.poc" );
  return false;
 }
 if ( ! pis_soubor_poc ( dir_tmp + "turon_slow.poc" , SLOW ) )
 {
  ShowMessage ( "CHYBA: Chyba pøi vytváøení souboru turon_slow.poc" );
  return false;
 }
 if ( ! pis_soubor_pop ( dir_tmp + "popo.pop"       , FAST ) )
 {
  ShowMessage ( "CHYBA: Chyba pøi vytváøení souboru turon.pop" );
  return false;
 }
 if ( ! pis_soubor_pop ( dir_tmp + "popo_slow.pop"  , SLOW ) )
 {
  ShowMessage ( "CHYBA: Chyba pøi vytváøení souboru turon_slow.pop" );
  return false;
 }

 f_cpy ( dir_var + "m" + rok_od + ".ts3" , dir_tmp + "m" + rok_od + ".ts3" );

 sprintf ( prg , "%sgen_tran.exe" , dir_progs.c_str () );
 sprintf ( arg , "%sturon.mmf"    , dir_tmp.c_str ()   );

 ret = spawnl ( P_WAIT , prg , prg , arg , NULL );

 /* systemova chyba pri spousteni */
 if ( ret == -1 )
 {
  switch ( errno )
  {
   case E2BIG   : ShowMessage ( "ERROR SPAWNL: Arg list too long"   ); break;
   case EINVAL  : ShowMessage ( "ERROR SPAWNL: Invalid argument"    ); break;
   case ENOENT  : ShowMessage ( "ERROR SPAWNL: File name not found" ); break;
   case ENOEXEC : ShowMessage ( "ERROR SPAWNL: Exec format error"   ); break;
   case ENOMEM  : ShowMessage ( "ERROR SPAWNL: Not enough memory"   ); break;
   default      : ShowMessage ( "ERROR SPAWNL: Errno = " + errno    );
  }
  return false;
 }

 /* behova chyba programu */
 if ( ret !=  0 )
 {
  ShowMessage ( "ERROR SPAWNL: Runtime error of GEN_TRAN; code = " + ret );
  return false;
 }

 if ( ! FileExists ( dir_tmp + "roztok.ts3" ) )
 {
  ShowMessage ( "CHYBA: Nepodaøilo se vytvoøit soubory roztok.ts3." );
  return false;
 }

 return true;
}

/*****************************************************************************/
```

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop
USERES("Turon.res");
USEFORM("FormHlavni.cpp", Hlavni);
USEFORM("FormObrazovka1.cpp", Obrazovka1);
USEFORM("FormObrazovka2.cpp", Obrazovka2);
USEFORM("FormObrazovka3.cpp", Obrazovka3);
USEFORM("FormObrazovka4.cpp", Obrazovka4);
USEFORM("FormObrazovka5.cpp", Obrazovka5);
USEFORM("FormObrazovka5s.cpp", Obrazovka5s);
USEFORM("FormObrazovka6.cpp", Obrazovka6);
USEFORM("FormObrazovka7.cpp", Obrazovka7);
USEFORM("FormObrazovka8.cpp", Obrazovka8);
USEFORM("FormObrazovka9.cpp", Obrazovka9);
USEFORM("FormLogo.cpp", Logo);
USEFORM("FormDebug.cpp", Debug);
USEUNIT("FileManagement.cpp");
USEUNIT("SouboryParametry.cpp");
USEUNIT("SouboryGwsView.cpp");
USEUNIT("VypocetOdhadu.cpp");
USEUNIT("PredvyberCerpani.cpp");
USEUNIT("SouboryXA.cpp");
USEUNIT("SouboryXAVysledky.cpp");
USEUNIT("SouboryPT.cpp");
USEUNIT("VypocetPT.cpp");
USEUNIT("SouboryPTVysledky.cpp");
USEUNIT("EkonomickyModel.cpp");
USE("Declar.h", File);
//---------------------------------------------------------------------------
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
        try
        {
                Application->Initialize();
                Application->Title = "TURON 2003";
                Application->CreateForm(__classid(THlavni), &Hlavni);
                Application->CreateForm(__classid(TLogo), &Logo);
                Application->CreateForm(__classid(TObrazovka1), &Obrazovka1);
                Application->CreateForm(__classid(TDebug), &Debug);
                Application->Run();
        }
        catch (Exception &exception)
        {
                Application->ShowException(&exception);
        }
        return 0;
}
//---------------------------------------------------------------------------
```

```
//---------------------------------------------------------------------------

#ifndef FormHlavniH
#define FormHlavniH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <Menus.hpp>
//---------------------------------------------------------------------------
class THlavni : public TForm
{
__published:  // IDE-managed Components
        TPopupMenu *PopupMenu1;
        TMenuItem *MIDebug;
        TMenuItem *MIAbout;
        TMenuItem *MISeparator2;
        TMenuItem *MIClose;
        TMenuItem *MISeparator1;
        TMenuItem *MI_SKIP_ALL;
        TMenuItem *MI_SKIP_XLS;
        void __fastcall FormCreate(TObject *Sender);
        void __fastcall FormDestroy(TObject *Sender);
        void __fastcall MIDebugClick(TObject *Sender);
        void __fastcall MIAboutClick(TObject *Sender);
        void __fastcall MICloseClick(TObject *Sender);
        void __fastcall MI_SKIP_ALLClick(TObject *Sender);
        void __fastcall MI_SKIP_XLSClick(TObject *Sender);
        void __fastcall FormCloseQuery(TObject *Sender, bool &CanClose);
private:  // User declarations
public:   // User declarations
        __fastcall THlavni(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE THlavni *Hlavni;
//---------------------------------------------------------------------------
#endif
```

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#define MAIN

// #include "FormHlavni.h"
#include "Declar.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
THlavni *Hlavni;
//---------------------------------------------------------------------------
__fastcall THlavni::THlavni(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall THlavni::FormCreate(TObject *Sender)
{
 // nastavi spravny desetinny oddelovac; vsehny podprogramy se klonuji
 // od hlavniho procesu (fce spawnl), maji tedy stejny desitkovy oddelovac
 // jako hlavni proces

 DecimalSeparator = '.';

 // jeste musime zajistit aby aplikace nereagovala na zmenu desetinneho
 // oddelovace v prubehu vypoctu

 Application -> UpdateFormatSettings = false;

 // nyni je potreba naplnit promene dir
 // cesty k jednotlivym adresarum prekonvertujeme na tzv. aliasy
 // (alias = nazev pro MS-DOS = nazev 8+3 = ShortPathName)
 // tudle zalezitost vyzaduje program XA.EXE

 dir_progs    = ExtractFilePath ( Application -> ExeName );
 dir_progs    = ExtractShortPathName ( dir_progs );
 dir_sit      = dir_progs + "sit\\";
 jmn_sit      = "kominy";
 dir_base_var = dir_progs + "varianty\\";
 dir_tmp      = dir_progs + "temp\\";
 del_tmp      = true;

 // prednastavime specielni promenne

 SKIP_ALL = false;
 SKIP_XLS = false;

 // podivame se, jestli existuje soubor s globalnimi parametry
 // a pokusime se ho nacist

 nacti_globalni_parametry ();

 // projistotu se podivame zda tyto adresare existuji
 // a zda v nich jsou potrebne soubory

 if ( ! existuji_vsechy_soubory_slozky () )
 {
  ShowMessage ( ( AnsiString ) "CHYBA: Chybí soubory (adresáøe) nezbytnì" +
                               " nutné pro chod celého systému," +
                               " program nebude fungovat." );
 }

 // protoze mohlo dojit k novemu nacteni cest ze souboru turon.ini
 // zformatujeme tyto cesty opet do formatu 8+3

 dir_sit      = ExtractShortPathName ( dir_sit );
 dir_base_var = ExtractShortPathName ( dir_base_var );
 dir_tmp      = ExtractShortPathName ( dir_tmp );
```

```
 // jeste jedna kontrola, jestli je vsechno v poradku

 if ( dir_progs == "" ) {
  ShowMessage ( ( AnsiString ) "CHYBA: Neplatné jméno adresáøe dir_progs," +
                               " program nebude fungovat." );
 }
 if ( dir_sit == "" ) {
  ShowMessage ( ( AnsiString ) "CHYBA: Neplatné jméno adresáøe dir_sit," +
                               " program nebude fungovat." );
 }
 if ( dir_base_var == "" ) {
  ShowMessage ( ( AnsiString ) "CHYBA: Neplatné jméno adresáøe dir_base_var," +
                               " program nebude fungovat." );
 }
 if ( dir_tmp == "" ) {
  ShowMessage ( ( AnsiString ) "CHYBA: Neplatné jméno adresáøe dir_tmp," +
                               " program nebude fungovat." );
  del_tmp = false;
 }

 // a hned zezacatku promazeme adresar temp

 d_clean_temp ();

 // priradim technologie do pole technologie

 Technologie [ iNDS ] = & NDS;
 Technologie [ iEDR ] = & EDR;
 Technologie [ iBAR ] = & BAR;
 Technologie [ iVYP ] = & VYP;

 // na zaver priradime jmena technologiim, slozkam a vyluhovacim polim
 // tato jmena jsou vyzadovana pri cteni ini souboru

 priradit_jmena_technologie ();
 priradit_jmena_slozky ();
 priradit_jmena_VP ();

 // a nacteme ekonomicke parametry

 if ( ! nacti_ekonomicke_parametry () )
  ShowMessage ( "CHYBA: Došlo k chybì pøi naèítání ekonomických parametrù." );
}
//---------------------------------------------------------------------------
void __fastcall THlavni::FormCloseQuery(TObject *Sender, bool &CanClose)
{
 int ret = Application -> MessageBox ( "Opravdu chcete program ukonèit?" ,
                                       "TURON" , MB_YESNO|MB_ICONQUESTION );
 if ( ret == ID_YES ) CanClose = true;
 else                 CanClose = false;
}
//---------------------------------------------------------------------------
void __fastcall THlavni::FormDestroy(TObject *Sender)
{
 d_clean_temp ();
}
//---------------------------------------------------------------------------
void __fastcall THlavni::MIDebugClick(TObject *Sender)
{
 Debug -> Show ();
}
//---------------------------------------------------------------------------
void __fastcall THlavni::MI_SKIP_ALLClick(TObject *Sender)
{
 MI_SKIP_ALL -> Checked = ! MI_SKIP_ALL -> Checked;
}
//---------------------------------------------------------------------------
void __fastcall THlavni::MI_SKIP_XLSClick(TObject *Sender)
{
 MI_SKIP_XLS -> Checked = ! MI_SKIP_XLS -> Checked;
}
//---------------------------------------------------------------------------
void __fastcall THlavni::MIAboutClick(TObject *Sender)
{
 Logo -> ShowModal ();
}
//---------------------------------------------------------------------------
void __fastcall THlavni::MICloseClick(TObject *Sender)
{
 Hlavni -> Close ();
}
//---------------------------------------------------------------------------
```