ALGORITMY MATEMATICKÉ VIZUALIZACE

REŠERŠNÍ PRÁCE

Martin Petřek

Fakulta jaderná a fyzikálně inženýrská
obor: softwarové inženýrství

PRAHA 2001

Obsah

Úvod

Matematická vizualizace je poměrně mladá vědní disciplína, která vznikla z potřeby umět přijatelně zobrazit a znázornit složité matematické objekty, které existovaly zatím jen v mysli matematiků. Původně byla spíše řazena jako podoblast diferenciální geometerie, numerické matematiky a počítačové grafiky, ale po čase se začaly metody matematické vizualizace hojně používat i v jiných oblastech vědy. Dnes je předmět zájmu matematické vizualizace již natolik široký, že nelze přesně vymezit, co do vizualizace patří a co ještě ne, protože téměř v každém oboru je nutné umět prezentovat a zobrazovat data, ať už napočítaná nebo naměřená.

Tato rešeršní práce pojednává o základních metodách matematické vizualizace. Zejména je kladen důraz na metody spojené se zobrazováním vektorových polí a toků. Tyto se dnes uplatňují zvláště ve vědeckých ústavech, které se zabývají modelováním a simulací proudění v tekutinách.

Dále jsou ukázány metody pro zobrazení a animaci křivek s volnou hranicí, neboť tato oblast je momentálně předmětem výzkumu na Katedře Matematiky FJFI. Následuje přehled na trhu dostupných vizualizačních balíků, které umožňují pomocí implementovaných metod vědecká data zobrazovat a prezentovat. Jedná se o programy MATLAB, GRAPE, ALBERT a Elemvis.

Závěr pak shrnuje podané informace s odkazem na další výzkum v této oblasti. Práce též obsahuje několik dodatků, kde jsou diskutovány základní matematické (numerické) metody, které jsou při vizualizaci objektů a dat potřebné. Poslední částí je seznam zdrojů a literatury, ze kterých jsem při psaní čerpal.

V Praze 13. října 2001 *Martin Petřek*

Kapitola 1

Matematická vizualizace

1.1 Základní metody zobrazování

Při řešení problémů a úloh z technické praxe zpravidla vždy dochází k přesnější matematické formulaci problému a to zejména zavedením veličin a rovnic, které nám daný problém charakterizují. Veličiny (matematické nebo fyzikální) mívají většinou povahu funkcí. Tyto funkce lze rozdělovat podle parametrů a podle vrácených hodnot. Je tedy jasné, že právě podle typů funkcí by se daly rozdělit i metody a techniky vizualizace, které tyto funkce zobrazují. Několik z nich je v následujícím seznamu.

Vizualizační techniky

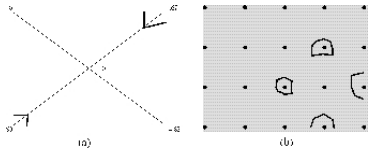
- skalární
- vektorové a objemové
- proudících toků a polí
  - medicínské
  - informativní
  - prostorové (stereoskopické) a techniky virtuální reality
    - ostatní

Vizualizace na čase nezávislých dat ve 2D

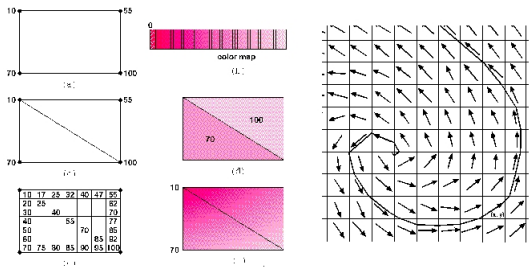
Vrstevnice
Oblíbenou technikou k zobrazení skalárních funkcí jsou metody, které generují množinu křivek reprezentující konstantní funkční hodnoty (vrstevnice,

izočáry). Generování těchto „vrstevnic“ však nemusí být triviální, protože máme-li funkční hodnoty určeny tradičně v uzlech sítě, je nutné při zobrazení jednotlivých hladin body interpolovat, a tak může dojít k situacím, kdy není jasné, kudy máme vrstevnici dále vést.

Předpokládejme nyní pravidelnou mřížku ve 2D (buňky jsou obdélníky). Na odstranění mnohoznačnosti pro výběr vrstevnice se provede rozdělení buňky na čtyři trojúhelníky (spojením úhlopříček). Pak se spočítá průměrná hodnota ve středu buňky ze všech vrcholů a naleznou se průsečíky podél každé hrany trojúhelníka (na hledané hladině). Průsečíky pak můžeme normálně spojit, protože k víceznačnosti už nedojde.



Obrázek 1.1: Tvorba pomocných bodů vrstevnic při nejednoznačném výběru



Obrázek 1.2: Typy stínování a tvorba proudnic

Stínování

Mějme funkci opět definovanou ve vrcholech pravidelné sítě. Chceme-li zobrazit funkční hodnotu barvou, je nutné nějak zobrazit barvu mezi jednotlivými vrcholy. Použijeme-li *konstantní* stínování, stačí rozdělit obdélník na dva trojúhelníky (např. podél úhlopříčky) a spočítat průměr z vrcholů nebo systematicky volit jeden z vrcholů. Vzniklé trojúhelníky jsou navíc automaticky konvexní a tudíž vhodné pro ořezávání. Při *Gouraudově* stínování se používá bilineární interpolace mezi vrcholy, takže přechod je více plynulý.

Konstantní stínování se používá, chceme-li vidět i jednotlivé elementy, a tak se lépe orientovat, Gouraudovo pak, má-li být výsledek „zahlazený“.

Šipky

Zobrazení vektorů pomocí šipek je často užívaná metoda pro svou jednoduchou implementaci. V uzlech známe složky vektoru, tak je jednoduše zob-

razíme, případně vektory „naškálujeme“ nějakou vhodnou konstantou, aby šipky nebyly příliš dlouhé. Tato technika však přináší problémy. Ve výsledném obrázku vzniká zmatek. Šipky zabírají hodně místa (zvláště při vykreslení i s hlavičkami), někde jsou nahuštěny, jinde naopak dost řídnou. Tyto problémy se ještě zvýrazní, máme-li promítat 3D oblast na dvojrozměrnou plochu.

Barevné kolečko

Tato technika spočívá v převodu složek vektoru (x, y, z) na hodnoty barevného modelu HSV (Hue, Saturation, Value) a interpretace vektoru jako obarveného pixelu. Ukazuje se, že převodní funkce do logaritmicky škálovaných stupnic (h,s,v) dává přijatelné výsledky a je dobře vnímána lidským zrakem.

Proudnice

Při vizualizace toků se obvykle konstruhují tzv. *proudnice*, což jsou křivky tečné k vektorům rychlostí. Používají se různé interpolační metody, většinou se vychází ze dvou bodů v každé buňce. Pro pravidelné sítě se používá bilineární interpolace, pro nepravidelné pak například Hardyho multikvadratická metoda. Některé další metody jsou popsány v 1.2 na straně 7.

Analýza kritických bodů

Zde se k zobrazení topologie vektorového pole (2D) spojují kritické body. Tím jsou myšleny body, kde například zaniká vektorová veličina. Křivky se ještě obarví nebo označí podle typů kritických bodů, které spojují. Příkladem jsou třeba sedlové body, uzly přitažlivé/odpudivé, víry aj.

Vizualizace na čase nezávislých dat ve 3D

Vizualizace 3D dat na 2D plochu (display) je oproti předchozímu případu daleko náročnější. Přestože projekce do 2D je relativně jednoduchá, vznikají problémy s nejednoznačností. Z výsledku je těžké určit relativní vzdálenosti a představit si inverzní zobrazení zpětně na 3D objekt. Proto se pro lepší představu používají pomocné techniky, které zobrazují osvětlení a stínování objektu, vržené stíny, pomocné projekční čáry a jiné záchytné body.

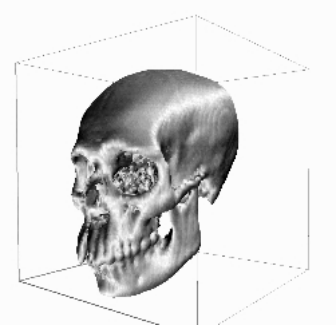
Další možnosti zobrazení je třeba animování dat se změnou barevné palety při průchodu objektem v řezech. Nyní následuje několik používaných metod.

Surface fitting

Jedná se o rozšíření vrstevnic do 3D. Známá je například metoda “Marching Cubes” (MC), kde se prochází objekt *voxel*[1] po voxelu a spojují se body se stejnou, uživatelem částečně definovanou hodnotou. Pak lze tato spojení převést na zobrazitelný povrch. Metoda má problémy s funkcemi, kde je velké

[1] objemová jednotka – podobně jako *pixel* ve 2D

množství sedlových bodů, neboť dochází ke tvorbě děr. Na jejich odstranění v současné době existuje několik metod.



Obrázek 1.3: Zobrazení metodou Surface fitting

Direct volume rendering
DVR algoritmy mapují data scény přímo do výsledného obrázku. Nejsou zde žádné pomocné geometrické primitivy. Přes scénu se posílají paprsky, které procházejí 3D oblastí s daty. Na průmětnu se pak zobrazí hodnota voxelu, na kterou paprsek narazil a která zapadla do předem určené kategorie. Tím se nám na průmětně zobrazí v podobě barev jen hodnoty, které potřebujeme a získáme tak rozložení, které nás zajímá. Existují dvě modifikace, kdy se buď posílají paprsky z oka pozorovatele přes průmětnu (obdoba *ray-tracingu*) nebo se ze všech voxelů vysílají k pozorovateli paprsky a zobrazují se pak jejich průsečíky s průmětnou.

Zobrazení v čase proměnlivých dat

Při zobrazování časově závislých dat by každého jistě napadla metoda, kdy napočítáme jednotlivé časové hladiny a výslednou posloupnost snímků zobrazíme v animační sekvenci. Tento mechanizmus se hojně používá. Je však nutné dávat pozor na různé nechtěné efekty s časovou animací spojené. Může se například stát vlivem diskrétních časových hladin, že vznikne tzv. *časový aliasing* tzn. nepravý nebo lépe falešný pohybový vjem. Příkladem nám může být i obyčejná kamera snímající pohyb cyklisty, kdy při určité frekvenci vzniká dojem rotace kola na opačnou stranu. Podobný dojem může vznikat i při translačním pohybu, kdy časová diskretizace není dostatečná pro rychlý pohyb v animaci – vzniká dojem stroboskopu. K nápravě se používají filtry: rozmazávající (blurring), roztřepující (aliasing) a kruhový (ringing). Objekt se v pohybu rozmaže, což působí věrohodněji. Na tuto operaci fungují poměrně dobře lineární filtry. Jedná-li se o „hladký“ pohyb, je účinná i lineární interpolace v čase. Pokud je však pohyb složitější (např. tok tekutiny), je

potřeba použít interpolaci založenou více na fyzikální povaze pohybu. Viz též 1.2 na straně 7.

1.2 Vizualizace vektorových polí

Zobrazování pole dat, zvláště pole vektorů rychlostí, jsou jednou ze základních úloh. Existuje několik rozdílných přístupů k této problematice. Nejjednodušší metody k zobrazení vektorů v uzlech nějaké pravidelné sítě, která pokrývá danou oblast, obvykle produkují matoucí a neurovnané výstupy kvůli rozdílné hustotě pole (tzv. *naškálování* pole) v dané prostorové oblasti, což vede k mnohonásobným překrýváním a zmatku. Navíc malé struktury, jako třeba víry, se můžou naopak v jiných místech zobrazovat nedostatečně. Cílem je proto získat přijatelnější metody.

Matematické pole budeme reprezentovat pomocí toku. Nechť je dáno vektorové pole $v : \Omega \times R_0^+ \to R^n$ pro nějakou oblast $\Omega \subset R^n$. Potom odpovídající tok $\phi : \Omega \times R_0^+ \to R^n$ je popsán systémem obyčejných diferenciálních rovnic

$$\partial_t \phi(x,t) = v(\phi(x,t),t)$$

za počáteční podmínky $\phi(x,0) = x$.

Dráha samostatné částice nám však jen velmi málo řekne o celkovém charakteru toku. Proto se můžeme ptát po nějakém lepším modelu, který by reprezentoval tok globálně na celé zkoumané oblasti. Ukážeme si dva odlišné přístupy k možnému řešení tohoto důležitého problému zpracovávání vektorových polí. Oba přístupy jsou založeny na řešení vhodného systému parciálně diferenciálních rovnic (PDE).

Nejprve můžeme řešit samotné rovnice proudění pro vhodné počáteční a okrajové podmínky. Tak explicitně vypočítáme Lagrangianovy obecné souřadnice (typicky poloha a čas) vzhledem ke vhodnému snímku. To je ovšem možné chápat jako texturovou mapu, která překrývá danou oblast svým vzorem a byla předepsána na odpovídající oblasti působnosti. Máme-li problém toku počítat v podobě přítoku u hranic oblasti, můžeme zvolit souřadnice jakožto přirozené souřadnice snímku. To tedy znamená, že metoda může být interpretována jako současné sledování všech bodů na vstupu do oblasti a studování jejich vývoje v čase. Numericky vlastně řešíme základní hyperbolický transportní problém s ohledem na danou rychlost. Tento přístup lze zvláště výhodně použít na v čase se měnící pole rychlostí, kde se vlastně ve skutečnosti popisuje geometrie dráhy částic. Texturové mapování je v čase spojité a tudíž umožňuje animaci.

Vedle toho však můžeme hledat metodu, která by „rozprostřela“ proudnice do hrubého obrazce, podle přidruženého vektorového pole $v(x,t)$ pro

daný pevný čas t. Dobrou vlastností by byla možnost postupně zvyšovat „drsnost“ tohoto obrazce, vzoru. Metody, které jsou založeny na takovém zvýraznění jistých struktur obrázku jsou dobře známé v oboru zpracování a analýza obrazu. Skutečně, nelineární rozptyl umožňuje vyhlazování šedých nebo barevných obrázků při zachování nebo zvýraznění hran. Jestliže takový rozptyl se silným vyhlazováním podél proudnic a zvýrazněním hran v ortogonálním směru aplikujeme na nějaký zpočátku náhodný šumový snímek, můžeme postupně vytvářet „drsnější“ vzory, které budou charakterizovat proudové pole v nějakém časovém okamžiku. Na numerické řešení problému rozptylu můžeme aplikovat metodu konečných prvků.

Nelineární rozptyl přidružený k vektorovému poli

V této metodě umožnuje nelineární anizotropní rozptyl aplikovaný na nějaký počáteční náhodný šumový obraz intuitivní a věrohodné zobrazení složitých proudících polí. Metoda je založena na lineárně integrální konvoluci, kdy na snímku s bílým šumem dochází ke korelaci intenzit barev (stupňů šedi) právě podél proudnic. Ukazuje se, že vhodnou volbou na konvoluční jádro je jádro Gaussovo. To je však při vhodné úpravě známé jakožto základní řešení rovnice tepla. To tedy znamená, že lineárně integrální konvoluce není nic jiného než řešení tepelných rovnic v jednodimenzionálním prostoru na proudnici. Body umístěné na ostatních integrálních křivkách tu nemají přímý vliv. Z toho důvodu je tloušťka výsledného vzoru na obrázku při lineárně integrální konvoluci stejná jako tloušťka vzoru na počátečním náhodném obrázku (obvykle jeden pixel). Zvyšování této tloušťky však vede k širokým pruhům a méně ostrým přechodům mezi zobrazením proudnic. Metoda je tedy založená na diskrétním přístupu k pixelům a můžeme ji pokládat za diskrétní difuzní proces podél proudnic.

Pokud se řeší spojitý případ rozptylu s podobnými vlastnostmi, vede problém na anizotropický rozptyl, který je řízen vhodnou rozptylovací maticí.

Ukažme si tedy základy nelineárního rozptylu při zpracování obrazu. Uvažujme funkci $\rho : R_0^+ \times \Omega \to R$ která řeší parabolický problém

$$\begin{aligned} \frac{\partial}{\partial t}\rho - \operatorname{div}(A(\nabla\rho_\varepsilon)\nabla\rho) &= f(\rho), \text{ na } R^+ \times \Omega, \\ \rho(0,\cdot) &= \rho_0, \text{ na } \Omega \\ \frac{\partial}{\partial \nu}\rho &= 0, \text{ na } R^+ \times \partial\Omega \end{aligned}$$

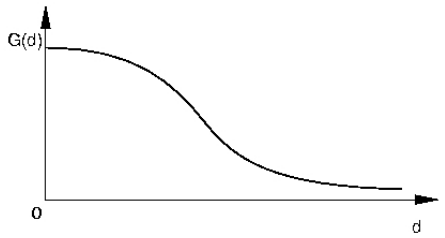
pro danou počáteční intenzitu $\rho_0 : \Omega \to [0,1]$
Veličina $\rho_\varepsilon = \Xi_\varepsilon * \rho$ je *ztlumení* dané hustoty, což je nutné provést kvůli uvedenému parabolickému problému. V našem případě interpretujeme hustotu

jako intenzitu obrázku – tj. skalární stupeň šedi. Řešení $\rho(\cdot)$ pak můžeme pokládat jako třídu snímků $\{\rho(t)\}_{t\in R^+}$, kde čas t slouží jako parametr. Poznamenejme, že při triviální volbě $A = 1$ a $f(\rho) = 0$ obdržíme standartní lineární rovnici tepla s izotropickým vyhlazovacím efektem. Při zpracovávání obrazu je ρ_0 dané jako počáteční obrázek, obsahující nechtěný šum. Rozptyl je pak řízen gradientem intenzity obrázku. Velké gradienty vyznačují hrany a okraje, které mají být zvýrazněny. Naopak malé hodnoty gradientu signalizují oblasti s přibližně stejnou intenzitou. Pro „vyhlazení“ šumu předepíšeme rozptylový koeficient

$$A = G(||\nabla\rho_\varepsilon||)$$

kde $G : R_0^+ \to R^+$ je monotoní klesající funkce s $\lim_{d\to\infty} G(g) = 0$ a $G(0) = \beta$ kde $\beta \in R^+$ je konstanta (obr. 1.4). (např. $G(d) = \frac{\beta}{1+||d||^2}$)

Jestliže bychom v argumentu funkce G nahradili ztlumený gradient $\nabla\rho_\varepsilon$ gradientem původním $\nabla\rho$, obdrželi bychom parabolický problém v oblastech vysokých gradientů, který už není dále dobře řešitelný. Ztlumením se vyhneme tomuto nedostatku a dostaneme se k žádanému vyhlazovacímu efektu. Nicméně pro zvýraznění strmých gradientů a tím i hran v obrázku, musíme tlumení Ξ_ε volit opatrně, aby nedošlo k velkým odlišnostem od původního obrázku. Obrázek 1.5 ukazuje vyhlazení obrázku a zvýraznění hran právě pomocí nelineárního rozptylu. Funkce $f(\cdot)$ navíc koriguje vyhlazování tak, aby se konečný snímek příliš nelišil od počátečního. Lze ji volit například jako $f(\rho) = \gamma(\rho_0 - \rho)$, kde γ je kladná konstanta (Obr. 1.7).



Obrázek 1.4: Tvar funkce $G(\cdot)$, která předepisuje difuzní koeficient

Nyní přejdeme k anizotropnímu rozptylu. Pro dané vektorové pole $v : \Omega \to R^n$ uvažujme lineární rozptyl ve směru vektorového pole i rozptyl ve směru k němu kolmém. Předpokládejme, že v je spojité a nenulové na Ω. Pak ale existuje třída spojitých ortogonálních transformací $B(v) : \Omega \to SO(n)$ takové, že $B(v)v = e_0$, kde $\{e_i\}_{i=0,\dots,n-1}$ je standartní báze v R^n (Obr. 1.6). Uvažujme rozptylovou matici

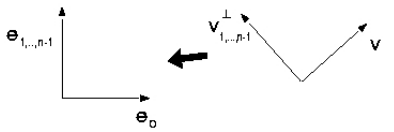
$A = A(v, \nabla\rho_\varepsilon)$ a definujme

$$A(v,d) = B(v)^T \begin{pmatrix} \alpha(||v||) & \\ & G(d) \end{pmatrix} B(v)$$

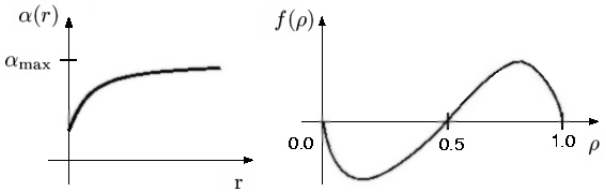
Obrázek 1.5: Šumový obrázek vlevo lze pomocí lineární difúze vyhladit a zvýraznit mu hrany

kde $\alpha : R^+ \to R^+$ řídí lineární rozptyl ve směru v tj. rozptyl podél proudnice a výše zmíněný koeficient $G(\cdot)$ řídí rozptyl v ortogonálním směru. Funkci $\alpha(\cdot)$ zvolíme monotónně (Obr. 1.7) tak, že

$$\alpha(0) > 0 \quad \text{a} \lim_{s\to\infty} \alpha(s) = \alpha_{\max}$$



Obrázek 1.6: Souřadnicová transformace $B(v)$



Obrázek 1.7: Graf funkce pro lineární difúzi $\alpha(\cdot)$ a funkce udržující původní vzhled obrázku

Pro takto navržený rozptyl nemá smysl udávat nějaký konkrétní počáteční obraz. Za ρ_0 zvolíme nějaký náhodný šum určitého frekvenčního rozsahu. Z toho důvodu se vzorky z původního obrázku „rozmáznou“ po proudu i proti proudu pole, zatímco tečné okraje k těmto vzorkům se zvýrazní. Pro časovou animaci je ještě nutné v průběhu vývoje udržovat kontrast snímku, neboť dochází k jeho snižování. Je proto vhodné volit funkci $f : [0, 1] \to R^+$ (Obr. 1.7). kdy

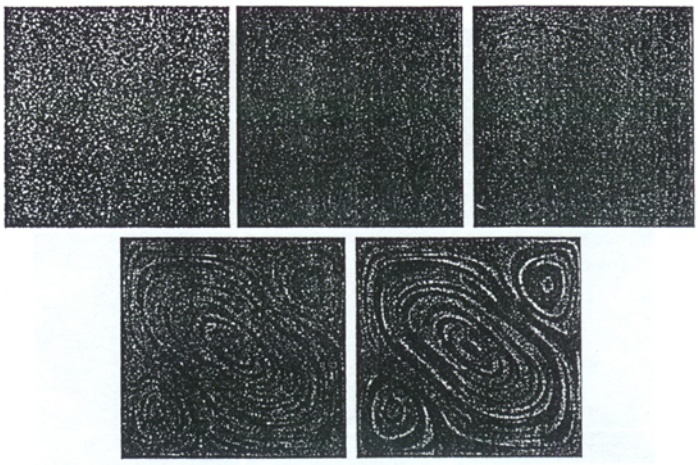
$$f(0) = f(1) = 0, f > 0 \text{ na } (0.5; 1) \text{ a } f < 0 \text{ na } (0; 0.5)$$

Shrnutím předchozích úvah tedy dostaneme rovnici

$$\frac{\partial}{\partial t} - \operatorname{div}(A(v, \nabla\rho_\varepsilon)\nabla\rho) = f(\rho)$$

kde jako počáteční snímek volíme náhodný šum ρ_0 a výpočtem obdržíme snímek zobrazující dané vektorové pole přijatelně a intuitivně. (Obr. 1.8 a Obr. 1.9)

Obrázek 1.8: Vizualizace proudění kolem válcové překážky metodou anizotropické difúze.



Obrázek 1.9: Několik časových kroků vývoje od šumového obrázku ke konečnému snímku (proudění ve 2D).

1.3 Vizualizace křivek s volnou hranicí

Rovnici popisující dynamiku anizotropních křivek lze zapsat jako

$$\alpha(\theta)v_\Gamma = g(\theta)\kappa - F,$$

kde $v_\Gamma \equiv v_\Gamma(\mathbf{x}, t)$ je normálová rychlost a $\kappa(\mathbf{x}, t)$ znamená křivost v bodě $\mathbf{x}$ křivky $\Gamma(t)$ v čase t. Předpokládejme oblast $\Omega_s(t) \subset R^2$ reprezentující pevnou látku a $\Omega_l(t) \subset R^2$ reprezentující kapalnou látku v čase t. Fázové rozhraní $\Gamma(t) = \partial\Omega_s(t) \cap \partial\Omega_l(t)$ má potom tvar křivky. Pokud $\mathbf{n}_\Gamma$ bude označovat vnější normálu Ω_s, je $\kappa = \text{div } \mathbf{n}_\Gamma$ a $v_\Gamma = -\vec{v_\Gamma} \cdot \mathbf{n}_\Gamma$. Proměnná θ znamená úhel mezi $\mathbf{n}_\Gamma$ a osou x_1. Úhel θ je z nějakého intervalu v R. Kladné funkce α, g, F jsou dány povahou materiálu. Je-li $F \equiv const$, pak reprezentuje řídící sílu úměrnou rozdílu termodynamickýcjh potenciálů obou fází. V případě, že $g = 1, \alpha = const$ je pohyb *izotropický*.

Získáme-li numerickým výpočtem vývoj křivky v různých časových hladinách, lze ji zobrazit několika způsoby. Prosté zobrazení jednotlivých bodů nám sice podá intuitivní představu, avšak nikterak valnou (zvláště v případě kdy máme málo bodů). Daleko lepší výsledek dostaneme, pokud použijeme nějaký druh interpolace nebo aproximace.

Parametrickou křivku můžeme obecně vyjádřit ve tvaru:

$$Q(t) = \sum_{k=0}^{n} t^k X_k, \quad \text{kde } X_k \in R^3$$

Neznámé koeficienty určíme pomocí *řídících bodů* a *vektorů*. Definujeme tak vlastně body, kde má křivka procházet, nebo jaký směr má mít tečna v daném bodě.

Patrně nejvhodnější a hojně používané křivky na interpolaci (aproximaci) jsou křivky třetího stupně tj. *Kubiky*

$$\begin{aligned} x(t) &= a_x t^3 + b_x t^2 + c_x t + d_x \\ y(t) &= a_y t^3 + b_y t^2 + c_y t + d_y \\ z(t) &= a_z t^3 + b_z t^2 + c_x t + d_z \end{aligned}$$

Označíme-li vektor $Q(t) = (x(t)\ y(t)\ x(t))$ a koeficienty vložíme do matice

$$C = \begin{pmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{pmatrix}$$

můžeme napsat

$$Q(t) = (t^3\ t^2\ t\ 1) \cdot C = T(t) \cdot C$$

Konstantní matici C můžeme rozepsat do součinu

$$C = MG,$$

kde matice M je typu 4×4 a nazývá se *bázová matice*. Čtyřprvkový vektor G reprezentuje vliv vnějších parametrů. Obsahuje řídící body, tečné vektory atp. Výsledný výpočet křivky pak probíhá podle vztahu

$$Q(t) = (t^3\ t^2\ t\ 1) \cdot \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{pmatrix} \cdot \begin{pmatrix} G_1 \\ G_2 \\ G_3 \\ G_4 \end{pmatrix}$$

1.3.1 Fergusonovy kubiky

Vzhledem k tomu, že potřebujeme křivku interpolovat, bude výhodné použít například *Fergusonovy kubiky* (1964 J. C. Ferguson). Tyto křivky jsou určeny dvěma body P_0, P_1 a dvěma tečnými vektory P_0' a P_1' v nich. Body P_0, P_1 určují polohu křivky (křivka jimi prochází). Směr a velikost vektorů pak určuje míru jejího vyklenutí. Čím je velikost vektoru větší, tím více se k němu křivka přimyká. Jsou-li oba vektory nulové, křivka degeneruje na úsečku P_0P_1.

Předpis pro výpočet Fergusonovy kubiky má tvar:

$$Q(t) = (t^3\ t^2\ t\ 1) \cdot \begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} P_1 \\ P_2 \\ P_1' \\ P_2' \end{pmatrix}$$

Rozepíšeme-li jednotlivé složky, dostaneme:

$$Q(t) = P_0F_1(t) + P_1F_2(t) + P_0'F_3(t) + P_1'F_4(t),$$

kde F_1, F_2, F_3, F_4 jsou *kubické Hermitovské polynomy* tvaru:

$$\begin{aligned} F_1(t) &= 2t^3 - 3t^2 + 1, \\ F_2(t) &= -2t^3 + 3t^2, \\ F_3(t) &= t^3 - 2t^2 + t, \\ F_4(t) &= t^3 - t^2 \end{aligned}$$

Pro hodnoty $t = 0$ resp. $t = 1$ prochází křivka body P_0 resp. P_1.

Největší přednost Fergusonových křivek se projeví při jejich navazování. Spojitost dvou kubik docílíme totožností posledního bodu křivky Q_1 a prvního bodu křivky Q_2. Spojitost C^1 zaručíme stejnými avšak opačně orientovanými vektory ve spojovacím uzlu. Slabší spojitost G^1 (geometrickou) zísáme zajištěním lineární závislostí těchto dvou vektorů.

1.3.2 Kochanek-Bartels

Pro interpolaci naší křivky je nyní potřeba určit způsob, jakým budeme počítat směrové vektory v navazujících uzlech. Můžeme použít algoritmus autorů *Kochanek-Bartels*, který je založen na Fergusonových kubikách. Vstupem je posloupnost interpolovaných bodů P_i a pro každý bod také tři koedicienty a_i, b_i, c_i. Ty určují chování křivky v okolí příslušného bodu. Tečné vektory v i-tém bodě a $i+1$-ním bodě interpolované posloupnosti se získají podle vztahu:

$$P'_i = \frac{(1-a_i)(1+b_i)(1+c_i)}{2}(P_i - P_{i-1}) + \frac{(1-a_i)(1-b_i)(1-c_i)}{2}(P_{i+1} - P_i)$$

$$P'_{i+1} = \frac{(1-a_{i+1})(1+b_{i+1})(1-c_{i+1})}{2}(P_i - P_{i-1}) + \frac{(1-a_{i+1})(1-b_{i+1})(1+c_{i+1})}{2}(P_{i+1} - P_i)$$

Význam koeficientů je následující:

- a_i – určuje, jak ostře křivka v bodě P_i „zatáčí“
- b_i – určuje šikmost křivky
- c_i – řídí její spojitost

1.3.3 Program Mvis

Program Mvis byl vytvořen za účelem zobrazení vývoje křivek s volnou hranicí. Hlavním požadavkem byla automatická tvorba animace, ze souboru vstupních dat, které byly získány přímým numerickým výpočtem. Dále možnost uživatelského rozhraní, které by umožňovalo přehrávat a zastavovat animaci, posouvat jednotlivé snímky a umožnit přehravání ve smyčce. Vzhledem k tomu, že numerické výpočty probíhaly na výpočetních stanicích se systémem UNIXového typu, byl navíc přidán požadavek pro standartní rozhraní *X-windows*.

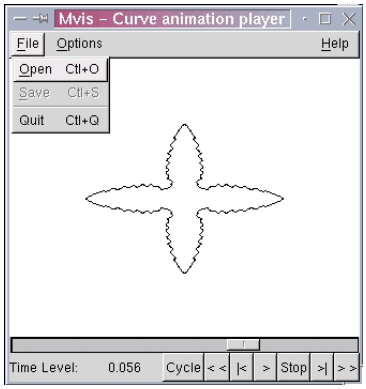
Program je implementován v knihovně *GTK widget library* v systému *Linux*. Ta je volně dostupná v různých distribucích linuxu, nebo volně stažitelná z Internetu. Existuje dokonce i verze pro operační systém *MS-Window*.

Program načte datový soubor, který má následující formát.

```
# Time level T =   .00000
    1.51730   1.51730
    1.50750   1.52230
    1.51730   1.51730
      .
      :
```

```
# Time level T =   .00800
    1.69837   1.50288
    1.69837   1.49712
    1.68973   1.48962
      .
      :
```

Obsahuje tedy jednotlivé časové hladiny a v nich příslušné body křivky. Počet bodů může být v každé hladině různý. Řádky obsahující pouze znak `<enter>` jsou ignorovány. Případné chyby v datech zatím program neumí rozpoznat. Program je vidět na obrázku 1.10.



Obrázek 1.10: Ukázka grafického rozhraní programu Mvis

Kapitola 2

Vizualizační programy

2.1 Matlab

Programový systém MATLAB (produkt firmy MathWorks) se stal velmi oblíbeným a mnohde dostačujícím nástrojem pro vědecké a inženýrské výpočty. Implementuje základní algoritmy pro operaci s čísly, funkcemi, maticemi a vektory, které jsou stěžejní pro příklady z praxe. Asi největší výhodou systému je to, že je navržen dosti obecně a promyšleně, což se odráží v jeho snadné rozšiřitelnost o doplňkové balíky (tzv. *toolboxy*), které se hodí pro konkrétnější úlohy. Příkladem takového toolboxu může být třeba známá nadstavba SIMULINK, umožňující zobrazovat schémata elektronických obvodů s možností simulace výpočtu. Tvorbou takových balíků se zabývá spousta vědeckých týmů; značnou část lze nalézt a stáhnout přímo z Internetu (`www.mathworks.com`). Obecnost systému však přináší i jisté negativní vlastnosti, jako je menší přesnost a rychlost výpočtu daného problému. Proto stále platí, že pro konkrétní úlohu je algoritmus výpočtu implementovaný třeba v C++ rychlejší a přesnější, než při použití obecnějšího výpočetního jádra MATLABu. Na druhé straně je pravdou, že nové verze MATLABu mají své jádro již velmi dobře optimalizované, a protože MATLAB nabízí uživateli komfort v podobě jednoduché syntaxe a přehledné ONLINE nápovědy, mnoho uživatelů dává přednost právě jemu. MATLAB však neslouží jen pro výpočty, ale i pro vizualizaci výsledků a jejich grafickou prezentaci s možností konverze do grafických formátů a tisku.

2.1.1 Grafika a prezentace

Grafika v MATLABu umožňuje snadné zobrazení a prezentaci získaných výsledků. Je možné vykreslit různé druhy grafů: dvourozměrné, třírozměrné, histogramy, apod. MATLAB také umožňuje otevřít více oken pro zobrazení

grafů najednou nebo zobrazit více grafů v jednom okně. Pro pokročilé uživatele je určeno stínování třírozměrných grafů s určením zdroje dopadajícího světla, animace třírozměrných grafů, zobrazení kontur a mnoho dalších grafických funkcí. Většinu těchto efektů je možné docílit jedním nebo několikamálo příkazy a jejich vykreslení je rychlé díky použitému algoritmu Z-buffer. Obrázky v grafických oknech MATLABu navíc nejsou statické. Každý již nakreslený objekt má přiřazen identifikátor, jehož prostřednictvím je možné měnit vlastnosti objektu a tím i jeho vzhled. Tento grafický systém, nazvaný *Handle Graphics*, umožňuje vytvořit v okně také ovládací prvky (tlačítka apod.) a vytvořit tak graficky ovládané uživatelské rozhraní. Je zde k dispozici i nástroj pro vytvoření uživatelského rozhraní interaktivně, beznutnosti programování. MATLAB dále podstatně rozšiřuje možnosti práce s trojrozměrnými objekty včetně nových technik násvitu, stínování a perspektivních zobrazení objektů. Vysoká kvalita zobrazení je zajištěna použitím 24-bitových barev.

Nyní následuje stručný přehled nejpoužívanějších funkcí pro zobrazení dat, neboť kompletní seznam funkcí by zabral příliš mnoho místa a lze nalézt, jak již bylo zmíněno, v ONLINE nápovědě (příkazem `helpwin` nebo `helpdesk`).

2.1.2 Přehled funkcí pro 2D i 3D grafiku

△ Dvojdimenzionální grafy

- `plot(x1,y1,...)`
  - zobrazí jednotlivé body nebo seznamy bodů (v podobě vektorů x a y)
  - v argumentu funkce lze také uvézt řetětec na upřesnění stylu vykreslování například ':r' pro tečkovaný červený(red) výstup
  - podobné funkce jsou též `loglog, semilogx, semilogy` které vykreslují stejně ovšem na logaritmických stupnicích.
- `polar(theta,rho,...)`
  - vykresluje body v polárních souřadnicích
- `axis(xmin, ymin,xmax,ymax)`
  - nastavuje rozmezí souřadných os
  - lze volat jako dvojslovo například `axis equal` pro stejná měřítka os (kružnice vypadá skutečně jako kružnice a ne elipsa)

- `zoom(factor)` – přibližuje nebo oddaluje graf nebo povoluje/zakazuje zoomování obrázku (`zoom on, zoom off`)
- `grid on/off` – nakreslí do grafu mřížku
- `box on/off` – uzavře graf do obdelníku (tj. nakreslí i horní a pravou osu)
- `hold on/off` – nakreslený graf nebude následujícím voláním `plot` překreslen
- `title, legend, xlabel, ylabel` – umožňují vložit titulek, legendu (pro více funkcí v jednom grafu), názvy os grafu
- `plotedit on/off` – připojí uživatelské ovládací prvky ke grafu
- `text, gtext` – umožňují vložit textový popisek na dané souřadnice v grafu (v případě `gtext` tam musí uživatel ukázat myší)

△ **Trojdimenzionální grafy**

- `plot3` – zobrazuje body trojrozměrného prostoru podobně jako `plot` o ve 2D
- `mesh(X,Y,Z,C)` – zobrazuje barevnou 3D-síť definovanou čtyřmi maticemi v argumentu; implicitně je C=Z
- `surf` – zobrazí trojrozměrně povrch(plochu) podobně jako `mesh`
  - barvy jsou použity v závislosti na nastavené `colormap`
  - zároveň je použit stínovací model příkazu `shading`
- `surfl` – jako `surf` ale zároveň nasvěcuje
- `fill(X,Y,Z,C)`
  - vyplní mnohoúhelník(ve 3D) podle specifikované barvy(v každém bodě)
  - je-li potřeba, tak se poslední bod automaticky doplní na první vrchol
- `colormap` *barevmodel* – nastavuje barevný model
  - možnosti jsou: `hsv, hot, gray, bone, copper, pink, white, flag, lines, colorcube, vga, jet, prism, cool, autumn, spring, winter, summer`

- `caxis(V)` – nastavuje měřítko „barevné“ osy
  - `V` je dvojprvkový vektor `[cmin, cmax]`
  - `cmin` a `cmax` odpovídají číslům barev v `colormap`
  - parametrem `'auto'` se vrátíme k původnímu nastavení
- `shading` *typ* – nastavuje typ stínování (`falt, interp, faceted`)
- `hidden on/off` – při použití `mesh` kreslí síť průhlednou, nebo neprůhlednou
- `brighten(`***value***`)` – zesvětluje, nebo ztmavuje paletu
  - ztmavení $-1 < value < 0$
  - zesvětlení $0 < value < 1$
- `lighting` *typ* – nastavuje typ osvětlovacího modelu (`flat, gourad, phong, none`)
- `material` *typ* – nastavuje vlastnost materialu (`shiny, dull, metal`)
- `specular, diffuse, surfnorm` – vracejí odrazovou složku světla, difuzní složku, nebo normálu povrchu
- `vrml(`***h, filename***`)` – exportuje grafický výstup(*handle h*) do souboru (*filename*) ve formátu VRML 2.0

– dále lze scénu otáčet, volit kamery, umisťovat světla, tisknou, atd.; tyto příkazy je ale lepší vyhledat přímo v nápovědě (`helpwin`)

△ **Speciální grafy**

- `area(`***x,y***`)` – vykreslí vyplněnou plochu tvořenou seznamy bodů(x,y)
- `bar(`***x,y***`)` – vykreslí vertikální obdelníky specifikované v monotoním seznamu bodů x a v seznamu bodůy
- `barh` – jako `bar` ale horizontálně
- `bar3, bar3h` – podobně ale graf je 3D; parametrem *width* lze nastavit šířku „kvádru“
- `comet,comet3 (`***x,y***`)` – vykreslují animaci dráhy
- `errorbar(`***x,y,l,u***`)` – vykresluje specielní typ grafu pro zobrazení odchylek $< y + l; y - u >$

- `ezplot(f)` – rychlé a jednoduché vykreslení funkce jedné reálné proměnné $f = f(x)$; parametr je *string* (např. 'sin')
  - lze uvést i parametr `[a,b]` jako interval vykreslení
- `ezpolar` – jako `ezplot` ale polární souřadnice
- `feather(u,v)` – vykresluje vektory rychlostí
  - parametr u representuje úhly šipek od horizontální osy, v délku šipek na horizontální ose
- `fplot(fce, lim)` – podobně jako `ezplot`
- `hist,pareto` – vykresluje histogram resp. sloupečky seřazené do klesajícího pořadí
- `pie, pie3 (x)` – vykresluje koláčové grafy podle parametrů x; rozdělí podle celkového součtu
- `plotmatrix (x, y)` – vykresluje rozptylovou matici; není-li x zadáno, kreslí `plotmatrix(y,y)`
- `ribbon (x,y)` – podobně jako `plot`, ale kreslí „stuhu“ ve 3D; lze nastavit šířku stuhy
- `stem (x,y), stem3` – vykresluje diskrétní hodnoty y na x v podobě čar z bodu [x,0] do bodu [x,y], kde je navíc jestě kolečko; parametr x lze vynechat; `stem3` pro 3D verzi
- `stair (x,y)` – schodovitý graf podobně jako stupňovité funkce
- `contour (z)` – kreslí vrstevnice funkce z na určitých hladinách
- `contourf` – jako `contour` ale ještě vybarvené
- `contour3` – opět jako `contour` ale ve 3D; vrstevnice jsou kresleny na své odpovídající hladině
- `clabel (cs,h)` – k vrstevnicím vypíše i její výšku; *cs* je vlastní funkce, h je handle na grafický objekt
- `quiver (x,y,u,v), quiver3` – vykreslí vektory rychlostí v x, y o složkách u,v ; `quiver3` je 3D verze
- `meshc` – kombinace `contour` a `mesh`

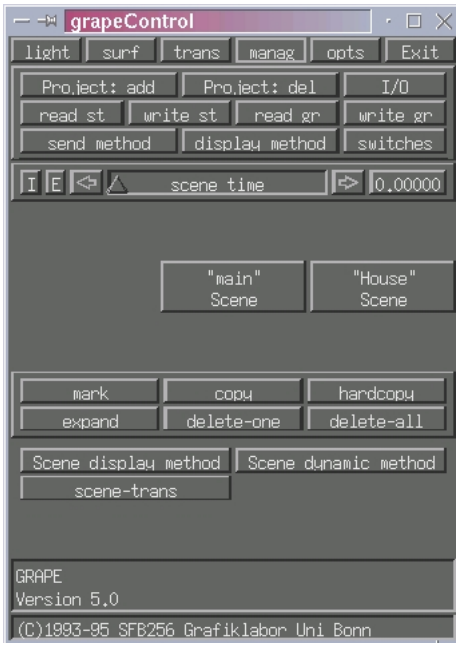
- `meshz` – `mesh` s postranními stěnami
- `slice (x,y,z,v,sx,sy,sz)` – zobrazení funkce z $[x, y, z] \rightarrow R$ „po řezech“ (*sx,sy,sz*)
- `surfc` – kombinace `contour` a `surf`
- `trisurf (tri,x,y,z,c)` – vykresluje funkci dvou proměnných jako `surf`, ale na vlastní triangulační síti; *tri* je matice $M \times 3$, řádky obsahují indexy do *x,y* a *z* obsahuje hodnoty, *c* případně ještě barvy
- `trimesh (tri,x,y,z,c)` – jako předchozí, ale kreslí pouze síť, ne povrch
- `waterfall` – stejné jako `mesh`, ale sloupcové čáry se nevykreslují
- `image (C)`, `imagec` – zobrazí matici *C* jako obdelníkový obrázek; je-li *C* typu $M \times N$ hledá se hodnota v *Colormap* podle hodnoty matice; je-li *C* typu $M \times N \times 3$ interpretují se tři složky jako RGB

2.2 Systém Grape

Systém GRAPE bývá překládán svými autory (Grape team – Institut für Angewande Mathematik Universität Freiburg) jako GRAphics Programming Environment. Tedy programovací prostředí pro grafiku. Jedná se o systém umožňující zobrazování geometrických, matematicky popsaných objektů a dalších technických tvarů. Právě proto se uplatňuje i ve vědecké vizualizaci.

Systém je už od samého jádra objektově navržen a nabízí tak možnost vytvářet třídy se svými metodami, rozšiřovat je a měnit podle vlastních představ. Vše zajišťují zabudované mechanizmy dědičnosti (*inheritance*), navzdory tomu, že je systém naprogramován v jazyku C. Knihovna tříd se skládá ze tříd pro geometrické objekty jako křivky (curves), povrchy (surfaces), objemy (volumes) a z jejich podtříd, které obsahují dodatečné informace jako funkce pro metodu konečných prvků, informace o zjemnění sítě a popis funkcí. Objekty lze kombinovat spolu s dalšími; statické třídy mohou být použity ve třídách časově závislých pro dynamický popis systémů. Příkladem toho může být třeba transformace křivky v čase, na čase závislé vektorové pole aj. Objekty mohou být však i čistě geometrického charakteru nebo to mohou být instance tříd pro uživatelské rozhraní, ovládací prvky atp.

V systému jsou zahrnuty algoritmy na výpočet parametricky definovaných křivek a objektů, na výpočet minimálních povrchů a H-povrchů. Dále algoritmy pro vývoj křivek a povrchů v prostoru, algoritmy řešící obyčejné



Obrázek 2.1: Grafické okno GRAPEu s ovládacím panelem

i parciální diferenciální rovnice a další funkce umožňující pracovat s objekty v Euklidovských, sférických či hyperbolických prostorech. Nedílnou součástí je i možnost vyhodnocovat funkce konečných prvků.

Samotný vizualizér GRAPEu se pak skládá z okna pro grafický výstup a z ovládacího panelu s různými vrstvami. Standartní vrstvy jsou určeny pro osvětlení a změny vlastností scény, pro transformace, pohyb kamer a animaci. Uživatel si má možnost definovat další vlastní vrstvy (max. 6), umožňující interaktivně měnit potřebná data. V systému je ještě implementován mechanizmus správy projektů, která umožňuje jednoduše organizovat práci a zvyšuje modularitu, podobně jako možnost dodávat dynamické knihovny u konkurenčních systémů.

Přehled existujících projektů ve verzi 5.3

Amandus: Počítá na parametru závislý minimální povrch zadaného Weierstrassovskou reprezentací

Build: Tvorba povrchů pro časově proměnnou triangulaci z makro plátů (*macro patches*); data se čtou ze souboru, který lze editovat interaktivně

Clippping: dokumentace nebyla k dispozici

Curve: Počítá na parametru závislé křivky z zadaného souboru

Diri: Kombinuje *Build, Refine a Dual* na prostředí pro počítání minimálního povrchu a jeho sdružené struktury

Dual: Dirichletův minimalizátor opět pro úlohy s minimálním povrchem; Počáteční triangulaci lze vytvořit v *Build* nebo *Explicit*

Explicit: Počítá povrch (závislý na parametru); popis povrchu je opět v datovém souboru

Grid_Edit: dokumentace nebyla k dispozici

House: Demonstrační program

Image: Vnitřní projekt používaný projektem *RGB Video*

Model: Transformace geometrie – fyzická, oproti transformacím scény v *surface menu*

Move_Trace, Probe: dokumentace nebyla k dispozici

Refine: Interaktivní zjemňování sítě

RGBVideo: Projekt na tvorbu bitmapových souborů a video sekvencí. Zatím je k dispozici na stanicích typu IRIS GL.

Scene_Tree: Prohlížeč hierarchické struktury dat, užitečný pro rozsáhlé hierarchie dat.

Surface: Základní projekt pro práci s parametricky závislými povrchy; parametry se čtou z popisoujícího souboru

Time_Object: Základní projekt pro práci s geometrickými objekty; parametry se čtou ze souboru

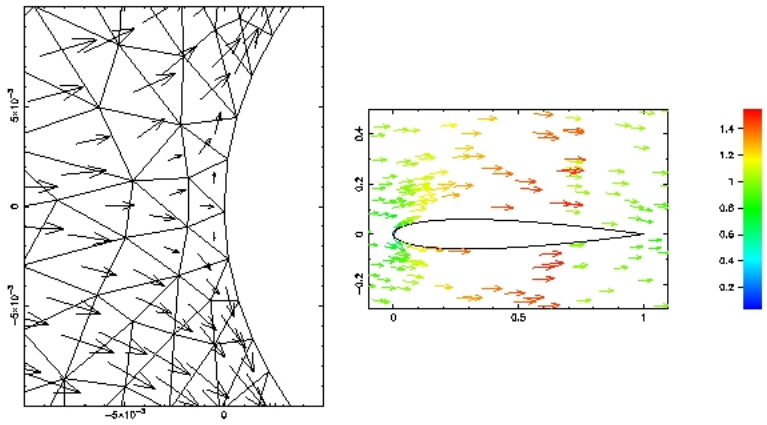
Tr2d_Edit: dokumentace nebyla k dispozici

2.3 Balík Elemvis

Při používání systémů pro vizualizaci a prezentaci matematických dat často nastane situace, kdy je potřeba výstup provést nikoliv na obrazovku počítače nebo obrázkový soubor, ale na fyzické médium, což bývá většinou papír nebo průhledné igelitové folie. Možnost tisknout obrazová data dnes musí nabízet

snad každý program pro vizualizaci a prezentaci, který obsahuje uživatelské rozhraní. Programy tedy fungují tak, že uživatel si zvolí buď výstup na tiskárnu nebo export do obrázkového souboru. Zařazení takových výstupů např. do publikací pak může přinášet problémy.

Matematicko-grafická prezentace dat je většinou spojena se sazbou rovnic, proto se ve většině odborných publikací používá profesionální sázecí systém TEX. Výstupem programu je soubor `*.dvi` (DeVice Independent – nezávislý na zařízení) který se ještě většinou převádí do formátu *PostScript*. (`*.ps`) Bylo by tudíž nadmíru výhodné, kdyby existoval program, který umí grafická data zobrazovat přímo do PostScriptu, neboť je tento formát podporován většinou tiskáren a je možné PostScriptové soubory jednoduše začleňovat do publikací. Balík Elemvis nabízí právě tuto možnost.



Obrázek 2.2: Ukázka výstupu balíku Elemvis

Elemvis je vizualizační nástroj vyvinutý na Matematicko-fyzikální fakultě Univerzity Karlovy. (viz [6]) Je navržen na vizualizaci dat obdržených pomocí metody konečných prvků. Data jsou aplikována v triangulační síť a to centrálně na každý element. Vstupem jednotlivých programů je konfigurační soubor `*.cfg`. Elemvis umožňuje zobrazení triangulační sítě, izočar, vektorového pole (šipky); je zde i program na vytvoření vlastní barevné palety. Je samozřejmě možné jednotlivé zobrazování kombinovat.

Balík je psán v jazyce Fortran77, autor poskytuje zdrojový kód. Ke kompilaci jsou potřeba knihovny PGPLOT a standartní X-Windows.

Datové soubory

Triangulační síť musí být uložena v souboru `trig`. Tam jsou uloženy tyto informace: celkový počet bodů `nPoints`, počet elementů `nElem`, po-

zice uzlů `x(i)` a `y(i)` spolu s indexy vrcholů každého elementu (`in(i,1) in(i,2) in(i,3)`), kde in(i, j) značí index j-tého vrcholu i-tého elementu ($j = 1, \ldots, 3$). Veličiny `x(i) y(i)` jsou reálné (typ *real*), ostatní jsou celočíselné (typ *int*). Uložení v souboru je následující:

```
nPoints
nElem
x(1)  y(1)
.
:
x(nElem)  y(nElem)
in(1,1)  in(1,2) in(1,3)
.
:
in(nElem,1) in(nElem,2) in(nElem,3)
```

Skalární hodnoty (na jednotlivých elementech) jsou v souboru `value`, který obsahuje reálné (*real*) hodnoty v každém elementu.

```
nElem
value(1)
.
:
value(nElem)
```

Vektorové veličiny (vektor o složkách u a v) jsou uloženy v souboru `uANDv`. Ten je tvaru:

```
nElem
u(1)  v(1)
.
:
u(nElem) v(nElem)
```

Soubor s barevnou paletou jménem `basepal` obsahuje počet barev `nBase` (typ *int*) a jednotlivé barevné složky RGB `r(i) g(i) b(i)` každá v intervalu $< 0; 1 >$.

```
nBase
r(1)  g(1) b(1)
.
:
r(nBase) g(nBase) b(nBase)
```

Konfigurační soubory pro jednotlivé programy

Konfigurační soubory každé utility obsahují řádky s čísly, což jsou parametry. Struktura (tj. pořadí parametrů) je pevně daná, ovšem bohužel pro každý program jiná, a tak je lepší mít pro každý program jeden konfigurační soubor a hodnoty jen měnit (nemazat!). Konfigurační soubory jsou v instalaci balíku zahrnuty a zdokumentovány. V každém lze nastavit výřez a jméno souboru pro výstup do *PostScript*u.

Seznam programů:

mesh – kreslí triangulační síť

valel – kreslí skalární hodnoty; v souboru `valel.cfg` lze nastavit i rozsah funkčních hodnot v barevné paletě

isol – kreslí izočáry; v souboru `isol.cfg` se nastavuje interval mezi sousedními hladinami a jedna počáteční hladina

cbsipky – kreslí vektory v podobě šipek (černobíle); v souboru `cbsipky.cfg` se nastavuje relativní vdálenost mezi šipečkami a relativní velikost šipeček

sipky – kreslí vektory barevně (v závislosti na normě); nastavuje se navíc rozsah palety jinak jako u `cbsipky`

colisol – barevná varianta `isol`

elem+mesh – kombinuje `mesh` a `valel`

vytvorpaletu – vytvoří barevnou paletu (256 barev) interpolací z barev uvedené v souboru `basepal`; například pro černobílý přechod se ve `basepal` uvedou 2 barvy (0 0 0) a (1 1 1) výsledek bude v souboru `paleta`

2.4 Albert

Programový balík ALBERT vyvíjený na Německé univerzitě ve Freiburgu (`htt://www.mathematik.uni-freiburg.de/IAM/ALBERT`) je určen pro numerické výpočty parciálních diferenciálních rovnic. Jedná se o knihovny napsané v jazyce C, které implementují základní výpočetní operace pro uvedený typ rovnic. Výpočty jsou založené na metodě konečných prvků (FEM) a tudíž lze odtušit, že balík klade důraz na implementaci algoritmů pro operace

s krycí sítí (tzv. *mesh*). Jsou zde metody pro tvorbu, zjemňování(*refinement*) i snížení počtu prvků v síti(*coarsening*). Proces zjemňování lze provádět lokálně a „řízeně“ tj. při výpočtu je možné provádět „odhad“ pro další zjemnění a to pouze v oblastech, kde je to nezbytně nutné. Výpočty lze samozřejmě aplikovat ve dvou i třech dimenzích; lze tvořit sítě triangulační, obdelníkové, simplexové. Pro zjemnění lze užít rekurzivních dělení. Síť je možné načíst z vedlejšího souboru, který je v obyčejném textovém formátu.

Jak bylo zmíněno výše, jedná se o knihovny funkcí. Autoři si nekladli za cíl vytvořit program podobný třeba MATLABu. V prvé řadě šlo o implementaci metod pro rychlý a přesný výpočet. Je tudíž zřejmé, že uživatele balíku ALBERT nečeká objektová struktura, ani žádné uživatelské rozhraní. Volají se tedy jednotlivé funkce, vše je zaměřeno na maximální rychlost a přesnost.

Přesto, že je systém vytvořen primárně pro výpočty, jsou zde k dispozici i základní funkce pro grafický výstup. Funkce se odkazují na standartní grafické rozhraní unixových systémů(*X-windows*) a dále na funkce implementované rozhraním (*GL/OpenGL*), které je nyní již také de facto standartem. Tato rozhraní lze v současné verzi použít pouze pro 2D grafiku. K zobrazení ve 3D lze využít možnosti knihovny GLTOOLS
(`http://www.wias-berlin.de/~gltools`) nebo použít „postprocessingový přístup“ a data nejprve uložit a k zobrazení použít systém GRAPE, kde trojdimenzionální zobrazování nechybí.

2.4.1 Funkce pro 2D grafiku

```
typedef void * GRAPH_WINDOW;

GRAPH_WINDOW graph_open_window(char *title, char (geometry,
                        REAL *world, MESH *mesh);
void graph_close_window(GRAPH_WINDOW win);
```

Parametry jsou:

title: název okna, v případě hodnoty `nil` je nastavena implicitní hodnota

geometry: určení velikosti okna ve formátu X-windows “HxW” nebo “HxW+X+Y” je-li `nil`, užije se výchozí velikost

world: souřadnice, určující část triangulační oblasti, která se má vykreslit (`xmin,xmax,ymin,ymax`) je-li `nil`, užije se buď `mesh` nebo se zobrazí čtverec $[0,1] \times [0,1]$

mesh: nastavitelná triangulace(pokud je `world` rovno `nil`) viz. výše; pokud `world` i `mesh` jsou `nil`, použije se čtverec $[0,1] \times [0,1]$

`graph_open_window()` vrací identifikátor okna typu `GRAPH_WINDOW`, který je pak potřeba v `graph_close_window()`; pokud nastane chyba, lze porovnávat vrácenou hodnotu s `NO_WINDOW`, což je chybová konstanta:

```
const GRAPH_WINDOW NO_WINDOW;
```

Standartní barevné konstanty

```
typedef float GRAPH_RGBCOLOR[3];   // 0 <= red,green,blue <= 1

const GRAPH_RGBCOLOR rgb_black, rgb_white, rgb_red,
const GRAPH_RGBCOLOR rgb_green, rgb_blue, rgb_yellow;
const GRAPH_RGBCOLOR rgb_magenta, rgb_cyan, rgb_grey50;
```

Pro „vyčištění“ okna barvou `c` lze použít:

```
void graph_clear_window(GRAPH_WINDOW win,GRAPH_RGBCOLOR c);
```

Jako výchozí hodnota barvy je použita `rgb_black`.

Kreslení triangulace

Triangulační síť lze vykreslit funkcí:

```
void graph_mesh(GRAPH_WINDOW win,MESH *mesh,
                GRAPH_RGBCOLOR c,FLAGS flag);
```

Parametry jsou:

win: grafické okno

mesh: triangulační síť

c: barva – implicitně je nastavena na `rgb_white`

flag: atributy pro specielní vlastnosti vykreslení:

- `GRAPH_MESH_BOUNDARY` je-li nastaven, tak se vykreslí jen okrajové hrany, jinak všechny; pokud je barva `nil`, kreslí se Dirichletovské okrajové hrany modře, Neumannovské červeně
- `GRAPH_MESH_ELEMENT_MARK` trojúhelníky označené pro zjemnění se vyplní červeně, trojúhelníky označené pro zjednodušení sítě se vyplní modře, neoznačené se vyplní bílou barvou

- `GRAPH_MESH_VERTEX_DOF` u každého vrcholu se připíše první stupeň volnosti (*DOF*)
- `GRAPH_MESH_ELEMENT_INDEX` napíšou se indexy prvků; lze použít, pokud se knihovna zkompilovala s parametrem `USE_OPENGL==0` a `EL_INDEX==1`

Funkce na zobrazení izočar

Křivky dané datovou strukturou `DOF_REAL_VEC` (spojový seznam bodů) lze vykreslit pomocí:

```
void graph_levels(GRAPH_WINDOW win, DOF_REAL_VEC *v, int n,
                  REAL *levels, GRAPH_RGBCOLOR *c, int refine);
void graph_level(GRAPH_WINDOW win, DOF_REAL_VEC *v, REAL level,
                 GRAPH_RGBCOLOR c, int refine);
```

Rutinka `graph_levels()` se použije ke kreslení násobných izočar. Parametry jsou:

win: grafické okno

v: `DOF_REAL_VEC` sktruktura obsahující data funkce

n: počet izočar pro kreslení

levels: `REAL` vektor rozměru `n` – hladiny, ve kterých se mají kreslit izočáry; je-li rovno `nil`, tak se vykreslí `n` hladin rovnoměrně rozdělených mezi maximem a minimem funkce

c: vektor barev pro každou izočáru(`n`); je-li `nil`, použijí se implicitní hodnoty (`rgb_white`)

refine: volitelný parametr pro zjemnění na izočáře. Izočáry se vypočítávají pomocí lineární interpolace na vrcholech trojúhelníka. Je-li `refine`≥ 0, každý trojúhelník se rekurzivně rozdělí na $4^{\texttt{refine}}$ menších trojúhelníků a funkce se vyhodnotí ve všech vrcholech těchto malých trojúhelníků. Je-li `refine<0`, použije se výchozí hodnota definovaná ve `v->admin->bas_fcts->degree-1`.

Funkce `graph_level()` kreslí samostatnou izočáru na hladině `level` barvou `c` při zjemnění `refine`; případy kdy `c=nil` nebo `refine`< 0 se řeší stejně jako v předchozí funkci.

Zobrazení skalární funkce

Funkci lze zobrazit pomocí barevných trojúhelníků.

```
void graph_values(GRAPH_WINDOW win, DOF_REAL_VEC *v,
                  int refine);
```

Hodnoty barvy jsou nastaveny impicitně; barva je rozdělena mezi maxima a minima `v`

Zobrazení vektorových funkcí

Podobně jako v předchozím případě existuje návrh rutin pro zobrazení vektorových funkcí. V současné době však ještě nejsou implementovány.

```
void graph_levels_d(GRAPH_WINDOW win, DOF_REAL_D_VEC *v, int n,
                REAL *levels, GRAPH_RGBCOLOR *c, int refine);
void graph_level_d(GRAPH_WINDOW win, DOF_REAL_D_VEC *v,
               REAL level, GRAPH_RGBCOLOR c, int refine);
void graph_values_d(GRAPH_WINDOW win, DOF_REAL_D_VEC *v,
                int refine);
```

Plánuje se také zobrazení pomocí šipek pomocí rutiny:

```
void graph_vec(GRAPH_WINDOW win, DOF_REAL_D_VEC *v, REAL scale,
               GRAPH_RGBCOLOR *c, int refine);
```

Zobrazení po částech konstantních funkcí

Hodnoty funkcí po částech konstantních, jako jsou lokální odhady chyb, lze zobrazit pomocí:

```
void graph_el_est(GRAPH_WINDOW win, MESH *mesh,
                  REAL (*get_el_est)(EL *el));
```

Parametry jsou:
`win`: grafické okno
`mesh`: triangulace
`get_el_est`: ukazatel na funkci, která vrací hodnotu jednoho prvku; tato funkce je na elementu konstantní.

2.4.2 Rozhraní GLTOOLS

Gltools lze používat pro 2d i 3d triangulace pokud se konstanta `DIM` rovná konstantě `DIM_OF_WORLD`. Definuje se následující typ a funkce:

```
typedef void* GLTOOLS_WINDOW;

GLTOOLS_WINDOW open_gltools_window(const char *, const char *,
                                   const REAL *, MESH *,int);
void close_gltools_window(GLTOOLS_WINDOW);
void gltools_mesh(GLTOOLS_WINDOW, MESH *, int);
void gltools_drv(GLTOOLS_WINDOW, const DOD_REAL_VEC *,
                 REAL, REAL);
void gltools_drv_d(GLTOOLS_WINDOW, const DOD_REAL_D_VEC *,
                   REAL, REAL);
void gltools_est(GLTOOLS_WINDOW, MESH *, REAL (*)(EL *),
                 REAL, REAL);
```

Popis:

- `open_gltools_window(title, geometry, world, mesh, dialog)` – funkce vrací `GLTOOLS_WINDOW` které je otevřeno pro výstup; v případě chyby vrací `nil`; `title` je řetezec pro název okna, je-li `nil`, je použit výchozí; `geometry` určuje velikosti okna ve formátu X-windows "HxW" nebo "HxW+X+Y" je-li `nil`, užije se výchozí velikost; volitelný parametr `world` je ukazatel na pole *světových souřadnic* (xmin, xmax, ymin, ymax) ve 2d a (xmin,xmax,ymin,ymax,zmin,zmax) ve 3d, určuje výřez oblasti sítě zobrazované v okně. Případ `nil` se řeší jako u podobné funkce
  `graph_open_window` na straně 27. Použije se buď `mesh` nebo výchozí oblast $[0,1]^{\texttt{DIM}}$; `mesh` je ukazatel na triangulační síť; jsou-li oba `world` i `mesh` `nil`, použije se oblast $[0,1]^{\texttt{DIM}}$; hodnota `dialog` nastavuje, zda (ne)bude zobrazování v interaktivní módu – `dialog=0` nebo jiná hodnota
  `close_gltools_window(win)` – uzavírá okno s parametrem, který byl získán z předchozí funkce

- `gltools_mesh(win,mesh,mark)` – zobrazí v okně `win` síť `mesh`; je-li `mark` nenulová, zobrazí se i počástech konstantní funkce `el->mark`

- `gltools_drv(win,u,min, max)` – zobrazuje `DOF_REAL_VEC u` do grafického okna `win`; `min` a `max` definují rozsah funkce pro display, je-li `min` $\geq$ `max` je určen automaticky

- `gltools_drv_d(win, ud, min,max)` – zobrazuje `DOF_REAL_D_VEC ud` do grafického okna `win`
- `gltools_est(win,mesh, get_el_est, min,max)` – zobrazuje v okně `win` odhadovou chybu na `mesh` jako počástech konstantní funkcí přístupnou přes `get_el_est()`; `min` a `max` definují rozsah pro display, je-li `min` $\geq$ `max`, je určen automaticky; lze použít pro libovolnou po částech konstantní funkci.

2.4.3 Rozhraní pro GRAPE

Data týkající se geometrie sítě a hodnot funkce lze zapsat do pomocného souboru pomocí `write_mesh[_xdr]()` a funkce

`write_dof_real[_d]_vec[_xdr]()`

a poté použít nějaký program využívající GRAPE pro zobrazení sítě `mesh`. Použití zobrazování `H-mesh` z GRAPE zatím není ještě dostatečně odladěno, plánuje se v blízké budoucnosti.

Závěr

Cílem této práce bylo seznámit se s problémy a úlohami, které se týkají matematické vizualizace. V první části první kapitoly byly popsány základní metody a algoritmy, které jsou potřebné k zobrazování vědeckých dat. Byly uvedeny ty nejpoužívanější, avšak také ty, o které se další metody silně opírají a které jsou též základem softwarových programů, umožňujících provádět vizualizaci dat. Vlivem dynamického rozvoje v této oblasti lze očekávat, že tato část práce bude zastarávat. Může však posloužit jako stručný úvod do dané problematiky; pro zájemce jsou v literatuře uvedeny odkazy, kde lze čerpat další informace o metodách a algoritmech, jimž jsou věnovány celé monografie.

Druhá a třetí část se zabývá detailnějším popisem metod pro konkrétní úlohy, přičemž je jasné, že i zde není vývoj zdaleka ukončen.

Celá druhá kapitola popisuje matematickou vizualizaci „z pohledu uživatele“. Je zde uveden stručný přehled užitečných funkcí nejrozšířenějších vizualizačních programů. Tato část může sloužit jako uživatelská příručka. Je jasné, že detailní popis všech funkcí by zabral několik stovek dalších stran, proto jsou popsány jen ty nejzákladnější. Pro výčet a popis ostatních jsou opět uvedeny odkazy na příslušné programové manuály.

Dodatky

3.1 Interpolace síťových funkcí

3.1.1 Interpolace funkcí jedné proměnné

Nechť je na intervalu $< a, b >$ reálné osy x dána síť $a = x_0 < x_1 < \ldots < x_n = b$ a v jejích uzlech jsou dány hodnoty $\{f_k\}_{k=0}^n$ funkce $f(x)$ definované na $< a, b >$. Zformulujeme úlohu po částech kubické interpolace. Na intervalu $< a, b >$ se hledá funkce $g(x)$, která vyhovuje těmto požadavkům:

1. $g(x)$ náleží do třídy $C^2(a, b)$, tj. je spojitá spolu se svými derivacemi druhého řádu včetně;

2. na každém z intervalů $< x_{k-1}, x_k >$ je $g(x)$ kubický polynom tvaru

$$g(x) \equiv g_k(x) = \sum_{l=0}^{3} a_l^{(k)} (x_k - x)^l, \quad k = 1, \ldots, n;$$

3. v uzlech sítě $\{x_k\}_{k=0}^n$ jsou splněny rovnosti

$$g(x_k) = f_k, \quad k = 0, 1, \ldots, n;$$

4. $g''(x)$ vyhovuje okrajovým podmínkám

$$g''(a) = g''(b) = 0$$

Přednosti interpolace, kterou jsme zvolili, se objasní v dalším, až dokážeme přirozenou extrémní vlastnost takto definované funkce $g(x)$.

Ukážeme, že uvedená úloha nalézt interpolační po částech kubickou funkci $g(x)$ má jediné řešení. Využijeme k tomu podmínky 1 až 4.

Protože druhá derivace funkce $g(x)$ je spojitá a lineární na každém intervalu sítě $< x_{i-1}, x_i >, i = 1, \ldots, n$, můžeme pro $x_{i-1} \leq x \leq x_i$ psát

$$g''(x) = m_{i-1} \frac{x_i - x}{h_i} + m_i \frac{x - x_{i-1}}{h_i},$$

kde $h_i = x_i - x_{i-1}, m_k = g''(x_k)$. Budeme obě strany dvakrát integrovat podle x a dostaneme

$$g(x) = m_{i-1}\frac{(x_i - x)^3}{6h_i} + m_i\frac{(x - x_{i-1})^3}{6h_i} + A_i\frac{x_i - x}{h_i} + B_i\frac{x - x_{i-1}}{h_i,}$$

kde A_i, B_i jsou integrační konstanty. Vypočteme je z podmínky $g(x_{i-1}) = f_{i-1},\ g(x_i) = f_i$. Dosadíme $x = x_i$ a $x = x_{i-1}$ do předchozí rovnice a obdržíme

$$\begin{aligned} m_i\frac{h_i^2}{6} + B_i &= f_i, \\ m_{i-1}\frac{h_i^2}{6} + A_i &= f_{i-1} \end{aligned}$$

Nakonec budeme mít

$$\begin{aligned} g(x) = m_{i-1}\frac{(x_i - x)^3}{6h_i} + m_i\frac{(x - x_{i-1})^3}{6h_i} + \\ + \left(f_{i-1} - \frac{m_{i-1}h_i^2}{6}\right)\frac{x_i - x}{h_i} + \left(f_i - \frac{m_i h_i^2}{6}\right)\frac{x - x_{i-1}}{h_i} \end{aligned} \tag{3.1}$$

$$\begin{aligned} g'(x) = -m_{i-1}\frac{(x_i - x)^2}{2h_i} + m_i\frac{(x - x_{i-1})^2}{2h_i} + \\ + \frac{f_i - f_{i-1}}{h_i} - \frac{m_i - m_{i-1}}{6}h_i \end{aligned} \tag{3.2}$$

Z 3.2 určíme limity derivace zprava a zleva v bodech $x_1, x_2, \ldots, x_{n-1}$:

$$\begin{aligned} g'(x_i - 0) &= \frac{h_i}{6}m_{i-1} + \frac{h_i}{3}m_i + \frac{f_i - f_{i-1}}{h_i} \\ g'(x_i + 0) &= -\frac{h_{i+1}}{3}m_i - \frac{h_{i+1}}{6}m_{i+1} + \frac{f_{i+1} - f_i}{h_{i+1}} \end{aligned}$$

Podle předpokladu 1 jsou funkce $g''(x)$ a $g'(x)$ spojité na $< a, b >$. Z podmínky spojitosti $g'(x)$ v bodech $x_1, x_2, \ldots, x_{n-1}$ dostaneme $n - 1$ rovnic

$$\frac{h_i}{6}m_{i-1} + \frac{h_i + h_{i+1}}{3}m_i + \frac{h_{i+1}}{6}m_{i+1} = \frac{f_{i+1} - f_i}{h_{i+1}} - \frac{f_i - f_{i-1}}{h_i}$$

Tyto rovnice doplníme s přihlédnutím k okrajovým podmínkám rovnicemi $m_0 = m_n = 0$ a obdržíme soustavu lineárních algebraických rovnic pro neznámé $m_1, m_2, \ldots, m_{n-1}$

$$Am = Hf$$

Čtvercová matice A je tvaru

$$A = \begin{pmatrix} \frac{h_1+h_2}{3} & \frac{h_2}{6} & 0 & \dots & 0 & 0 \\ \frac{h_2}{6} & \frac{h_2+h_3}{3} & \frac{h_3}{6} & \dots & 0 & 0 \\ 0 & \frac{h_3}{6} & \frac{h_3+h_4}{3} & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ 0 & 0 & 0 & \dots & \frac{h_{n-1}}{6} & \frac{h_n+h_{n-1}}{3} \end{pmatrix}$$

vektory m a f a obdélníková matice H jsou dány vzorci

$$\begin{aligned} m &= (m_1, m_2, \dots, m_{n-1})^{\mathbf{T}} \\ f &= (f_1, f_2, \dots, f_n)^{\mathbf{T}} \end{aligned}$$

$$H = \begin{pmatrix} \frac{1}{h_1} & \left(-\frac{1}{h_1} - \frac{1}{h_2}\right) & \frac{1}{h_2} & \dots & 0 & 0 \\ 0 & \frac{1}{h_2} & \left(-\frac{1}{h_2} - \frac{1}{h_3}\right) & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \left(-\frac{1}{h_{n-1}} - \frac{1}{h_n}\right) & \frac{1}{h_n} \end{pmatrix} \tag{3.3}$$

Lze ukázat, že matice A je pozitivně definitní, a tedy regulární. Z toho plyne, že interpolační spline-funkce $g(x)$ je jednoznačně daná.

Kubické spline-funkce mají jednu velmi důležitou vlastnost. Uvažujme na $< a, b >$ třídu funkcí, které mají druhou derivaci integrovatelnou s kvadrátem.

$$u \in W_2^2(a,b), \quad u(x_k) = f_k, \quad k = 0, \dots, n$$

Minimalizujme funkcionál

$$\Phi(u) = \int_a^b (u''(x))^2 \mathrm{d}x$$

Lze ukázat, že tuto rovnici minimalizuje právě spline-funkce $g(x)$, proto se mnohdy spline-funkce definují právě jako funkce, které tento funkcionál minimalizují.

Máme-li v praxi úhly, které mají svírat tečny v krajních bodech, použijeme okrajové podmínky

$$g'(a) = f'_0, \quad g'(b) = f'_n \tag{a}$$

Známe-li křivost křivky v a a b, jsou podmínky

$$g''(a) = f''_0, \quad g''(b) = f''_n \tag{b}$$

Víme-li o interpolované funkci, že je periodická s periodou $b-a$, je na místě zadat okrajové podmínky

$$g'(a) = g'(b), \quad g''(a) = g''(b) \tag{c}$$

Podmínky (a) povedou s ohledem na 3.2 k rovnicím

$$\frac{2}{3}m_0 + \frac{1}{3}m_1 = \frac{2}{h_1}\left(\frac{f_1 - f_0}{h_1} - f_0'\right)$$
$$\frac{1}{3}m_{n-1} + \frac{2}{3}m_n = \frac{2}{h_n}\left(f_n' - \frac{f_n - f_{n-1}}{h_n}\right)$$

podmínky (b) k rovnicím

$$m_0 = f_0'', \quad m_n = f_n''$$

a konečně podmínky periodičnosti spline-funkcí (c) vedou na

$$\begin{aligned} m_0 &= m_n \\ \frac{h_n}{6}m_{n-1} + \frac{h_n + h_1}{3}m_n &= \frac{f_1 - f_n}{h_1} - \frac{f_n - f_{n-1}}{h_n} \end{aligned}$$

Těmito rovnicemi doplníme soustavu a vyřešíme. Podmínky lze samozřejmě kombinovat.

3.1.2 Interpolace funkcí dvou a více proměnných

Nechť $D = \{x, y; a \le x \le b, c \le y \le d\}$ je obdélník v rovině (x, y). Sestrojíme v D síť

$$D_h = \{x_k, y_l; a = x_0 < x_1 < \ldots < x_n = b_n, c = y_0 < y_1 < \ldots < y_m = d\}$$

Za těchto podmínek spočívá úloha po částech bikubické interpolace funkce $f(x, y)$, jejíž hodnoty jsou dány v uzlech sítě D_h, v konstrukci funkce $g(x, y)$, která vyhovuje následujícím podmínkám:

1.
$$g(x, y) \in C^2(D) \tag{3.4}$$

2. v každé buňce sítě je $g(x, y)$ bikubický polynom tvaru

$$g(x, y) = g_{kl}(x, y) = \sum_{i,j=0}^{3} a_{ij}^{kl}(x_k - x)^i(y_l - y)^j \tag{3.5}$$

3. na síti D_h nabývá $g(x,y)$předepsaných hodnot

$$g(x_k, y_l) = f_{kl}, \quad k = 0, \ldots, n, \ l = 0, \ldots, m \tag{3.6}$$

4. funkce $g(x,y)$ vyhovuje podmínce

$$\left.\frac{\partial^2 g}{\partial \nu^2}\right|_\Gamma = 0 \tag{3.7}$$

kde ν je vnější normála k hranici Γ oblasti D.

Nyní je třeba znát i druhé derivace funkce v uzlových bodech. Budeme-li nejprve řešit $m+1$ lineárních algebraických rovnic na přímkách sítě $y = y_j, \ j = 0, 1, \ldots, m$, dostaneme hodnoty funkce $g_{xx}(x,y)$ v uzlech sítě D_h. Pak budeme analogicky řešit $n+1$ úloh interpolace spline-funkcemi na přímkách $x = x_i, \ i = 0, 1, \ldots, m$ a najdeme hodnoty funkce $g_{yy}(x,y)$ na D_h. Předpokládejme nyní, že máme vypočítat hodnotu $g(x,y)$ v nějakém bodě (x,y) a nechť $x_{i-1} \le x \le x_i, \ y_{j-1} \le y \le y_j$. Ze vzorců analogických 3.1 můžeme najít hodnoty $g(x,y)$ v bodech (x_i, y) a (x_{i-1}, y):

$$g(x_{i-1}, y) = N_{i-1,j-1}\tfrac{(y_j-y)^3}{6\tau_j} + N_{i-1,j}\tfrac{(y-y_{j-1})^3}{6\tau_j} + \tag{3.8}$$
$$+ \left(f_{i-1,j-1} - \tfrac{N_{i-1,j-1}\tau_j^2}{6}\right)\tfrac{y_j-y}{\tau_j} + \left(f_{i-1,j} - \tfrac{N_{i-1,j}\tau_j^2}{6}\right)\tfrac{y-y_{j-1}}{\tau_j}$$

$$g(x_i, y) = N_{i,j-1}\tfrac{(y_i-y)^3}{6\tau_j} + N_{ij}\tfrac{(y-y_{j-1})^3}{6\tau_j} + \tag{3.9}$$
$$+ \left(f_{i,j-1} - \tfrac{N_{i,j-1}\tau_j^2}{6}\right)\tfrac{y_j-y}{\tau_j} + \left(f_{ij} - \tfrac{N_{ij}\tau_j^2}{6}\right)\tfrac{y-y_{j-1}}{\tau_j}$$

Budeme-li znát hodnoty $g_{xx}(x_{i-1}, y)$ a $g_{xx}(x_i, y)$, můžeme ze vzorců typu 3.1 najít hodnotu $g(x,y)$. Připomeňme, že funkce $g_{xx}(x,y)$ je po částech kubická funkce proměnné y. Řešme tedy $m+1$ jednorozměrných úloh na přímkách $y = y_j$ pro funkci $g_{xx}(x,y)$, jejíž síťové hodnoty už známe. Tak dostaneme na síti D_h funkci $g_{xxyy}(x,y)$. Označíme-li $K_{ij} = g_{xxyy}(x_i, y_j)$, máme

$$g_{xx}(x_{i-1}, y) = K_{i-1,j-1}\tfrac{(y_j-y)^3}{6\tau_j} + K_{i-1,j}\tfrac{(y-y_{j-1})^3}{6\tau_j} + \tag{3.10}$$
$$+ \left(M_{i-1,j-1} - \tfrac{K_{i-1,j-1}\tau_j^2}{6}\right)\tfrac{y_j-y}{\tau_j} + \left(M_{i-1,j} - \tfrac{K_{i-1,j}\tau_j^2}{6}\right)\tfrac{y-y_{j-1}}{\tau_j}$$

$$g_{xx}(x_i, y) = K_{i,j-1}\tfrac{(y_j-y)^3}{6\tau_j} + K_{ij}\tfrac{(y-y_{j-1})^3}{6\tau_j} + \tag{3.11}$$
$$+ \left(M_{i,j-1} - \tfrac{K_{i,j-1}\tau_j^2}{6}\right)\tfrac{y_j-y}{\tau_j} + \left(M_{ij} - \tfrac{K_{ij}\tau_j^2}{6}\right)\tfrac{y-y_{j-1}}{\tau_j}$$

Použijeme-li znovu vzorec 3.1, určíme hodnotu $g(x,y)$:

$$g(x,y) = g_{xx}(x_{i-1}, y)\tfrac{(x_i-x)^3}{6h_i} + g_{xx}(x_i, y)\tfrac{(x-x_{i-1})^3}{6h_i} + \tag{3.12}$$
$$+ \left(g(x_{i-1}, y) - \tfrac{g_{xx}(x_{i-1},y)h_i^2}{6}\right)\tfrac{x_i-x}{h_i} + \left(g(x_i, y) - \tfrac{g_{xx}(x_i,y)h_i^2}{6}\right)\tfrac{x-x_{i-1}}{h_i}$$

K výpočtu je nutné tedy jednou vyřešit $(n+1)+(m+1)+(m+1) = 2m+n+3$ lineárních algebraických soustav a určit $N_{ij}, M_{ij}, K_{ij}, i = 0, 1 \ldots, n, j = 0, 1, \ldots, m$. Dále dosadíme do vzorců uvedených výše a získáme tak hodnotu $g(x, y)$.

Problém ve dvou dimenzích lze snadno zobecnit i do více rozměrné oblasti typu kvádru v R^n.

3.2 Metoda sítí

Nechť $\Omega \subset R^n$ je omezená oblast s po částech hladkou hranicí a L je eliptický operátor na Ω.
Úloha:

$$\begin{aligned} \frac{\partial y}{\partial t} - Ly &= f \qquad \text{na } (0,T)\times\Omega \\ \alpha(x)\frac{\partial y}{\partial n} + \beta(x)y &= \gamma(x) \qquad \text{na } (0,T)\times\partial\Omega \text{ (okrajová podmínka)} \\ y(0,x) &= y_0(x) \qquad \text{na } \Omega \text{ (počáteční podmínka)} \end{aligned}$$

se nazývá *smíšená úloha* pro Parabolickou parciální diferenciální rovnici.

Typová úloha pro *metodu sítí*: $\Omega = (a, b), Ly = D \cdot \Delta y = D \cdot \frac{\partial^2 y}{\partial x^2}$

$$\frac{\partial y}{\partial t} - D\frac{\partial^2 y}{\partial x^2} = f(x,y) \qquad \text{na } (0,T)\times(a,b) \tag{3.13}$$

$$y(t,a) = \gamma_1,\ y(t,b) = \gamma_2 \tag{3.14}$$

$$y(0,x) = y_0(x) \quad \text{na } (a,b) \tag{3.15}$$

Nahraďme nyní derivace centrálními diferencemi $(u_{\bar{x}x})$.
Označme $\tau > 0$ časový krok (tj. $T = N_1\tau$), $\bar{\omega}_h$ prostorovou síť na (a, b)

$$y(k\tau, jh) \overset{\text{ozn}}{=} y_j^k \qquad k = 0, \ldots, N_T; j = 0, \ldots, m$$

Náhradu rovnice provedeme v časové hladině k $(t = k\tau)$. Tu označíme u, následující hladinu $k + 1$ označíme $\hat{u}$.
Současná náhrada časových a postupných derivací:

a) explicitní schéma

$$\frac{\hat{u} - u}{\tau} - Du_{\bar{x}x} = f \qquad \text{na } \omega_h$$

pak $\hat{u} = u + \tau D u_{\bar{x}x} + \tau f$. Rozepišme $\hat{u}$ po uzlech.

$$u_j^{k+1} = u_j^k + \frac{\tau}{h^2} D(u_{j+1}^k - 2u_j^k + u_{j-1}^k) + \tau f_j^k \tag{3.16}$$

Použití: ze známe hladiny k řešením u vypočítáme hladinu $k+1$ (začínáme pro $k=0$ počáteční podmínkou)
Vazba časového a prostorového kroku:

$$\begin{aligned} \hat{u} &= Au + \tau f \\ \Rightarrow u^k &= Au^{k-1} + \tau f^{k-1} = A^2 u^{k-2} + \tau A\ f^{k-2} + \tau f^{k-1} = \cdots = \\ &= A^k u^0 + \cdots \end{aligned}$$

Nechceme, aby z malé změny u^0 vznikla velká změna u^k; dále potřebujeme, aby $\sigma(A) \subset (-1,1)$

$$A = \begin{pmatrix} 1 - D\frac{2\tau}{h^2} & D\frac{\tau}{h^2} & 0 & \cdots \\ \frac{D\tau}{h^2} & 1 - D\frac{2\tau}{h^2} & D\frac{\tau}{h^2} & 0 & \cdots \\ \vdots & & \ddots & & \vdots \end{pmatrix}$$

vlastní čísla $\lambda_i = 1 - 4\frac{\tau}{h^2} D \sin^2 \frac{i\pi}{2n}$ pro $i = 1, \ldots, m-1$
vektory: $\left(\sin \frac{i\pi j}{m}\right)_{j=1}^{m-1}$
Stabilita za podmínky: $-1 < 1 - 4\frac{\tau}{h^2} D \sin^2 \frac{i\pi}{2m} < 1$, neboli $D\frac{\tau}{h^2} < \frac{1}{2}$

b) implicitní schéma

$$\begin{aligned} \frac{\hat{u} - u}{\tau} - D\hat{u}_{\bar{x}x} &= \hat{f} \quad \text{na } \omega_h \\ \hat{u} - \tau D\hat{u}_{\bar{x}x} &= u + \tau \hat{f} \end{aligned}$$

po uzlech

$$u_j^{k+1} - D\frac{\tau}{h^2}\left(u_{j+1}^{k+1} - 2u_j^{k+1} + u_{j-1}^{k+1}\right) = u_j^k + \tau f_j^{k+1}$$

opět určíme hladinu $k+1$ z hladiny k; řešíme soustavu lineárních rovnic pro $\hat{u}$

$$\begin{aligned} A\hat{u} &= u + \tau \hat{f} \\ \text{kde} & \\ A &= \begin{pmatrix} 1 + D\frac{2\tau}{h^2} & -D\frac{\tau}{h^2} & 0 & \cdots \\ \frac{-D\tau}{h^2} & 1 + D\frac{2\tau}{h^2} & -D\frac{\tau}{h^2} & 0 & \cdots \\ \vdots & & \ddots & & \vdots \end{pmatrix} \\ u^k &= A^{-1}(u^{k-1} + \tau f^k) = (A^{-1})^2 u^{k-1} + A^{-1}\tau f^k + (A^{-1})^2 \tau f^{k-1} = \\ &= \ldots = (A^{-1})^k u^0 + \ldots \end{aligned}$$

Tvrzení: $\sigma(A^{-1}) \subset (-1,1)$ (neboli Implicitní schéma je nepodmíněně stabilní). Chyba aproximace diferenciálního operátoru $\frac{\partial}{\partial t} - L$ pro schéma a) a b):

$$\begin{aligned} L &\rightarrow L_h \qquad (-y'' \rightarrow u_{\bar{x}x} \text{ s přesností } O(h^2)) \\ \frac{\partial y}{\partial t} &\rightarrow u_t, u_{\bar{t}} \quad \text{(tj. dopředná resp. zpětná diference) s chybou } O(\tau) \end{aligned}$$

c) Crank-Nicolsonovo schéma

$$\frac{\hat{u} - u}{\tau} - \frac{D}{2}(\hat{u}_{\bar{x}x} + u_{\bar{x}x}) = \frac{1}{2}(f + \hat{f})$$

Následující hladina $\hat{u}$ se najde z předchozí

$$\hat{u} - \tau\frac{D}{2}\hat{u}_{\bar{x}x} = u + \frac{\tau D}{2}u_{\bar{x}x} + \frac{\tau}{2}(\hat{f} + f) \quad \text{(opět implicitní povahy)}$$

schéma nahrazuje rovnici 3.13 na hladině $(k + \frac{1}{2})\tau$:
$\frac{\hat{u}-u}{\tau}$ tj. $\frac{u^{k+1}-u^k}{\tau}$ nahrazují časové derivace $\frac{\partial f}{\partial t}((k + \frac{1}{2}\tau), \cdot)$
s přesností $O(\tau^2)$ – centrální diference

$\frac{1}{2}(u^{k+1}_{\bar{x}x} + u^k_{\bar{x}x}) + \frac{1}{2}(f^k + f^{k+1})$ nahrazují $\frac{\partial^2 y}{\partial x^2}((k+\frac{1}{2})\tau, \cdot)$ a $f((k+\frac{1}{2})\tau, \cdot)$

s přesností $O(\tau^2)$
Shrnutí:
Crank-Nicolsonovo schéma má přesnost $O(\tau^2 + h^2)$

Literatura

[1] Becker, J.; Preusser, T.; Rumpf, M., PDE Methods in Flow Simulation Post Processing, August 17, 1999

[2] Beneš, M. and Mikula, K., Simulation of Anisotropic Motion by Mean Curvature – Comparison of Phase Field and Sharp Interface Approaches, Acta Mathematica Universitatis Comenianae, Volume 67 (1998), No. 1, 17–42

[3] Gale, T., GTK v1.2 Tutorial, www.gtk.org, February 23^{rd}, 2000

[4] Grape-Team; GRAPE Manual Version 5.3, Bonn, Freiburg,1997

[5] Hege, H.– C.; Polthier, K., (Eds.): Mathematical Visualization Algorithms, Applications and Numerics, Springer, Berlin (1998)

[6] Krška, F., Elemvis2 – package for scientific data visualization, http://adela.karlin.mff.cuni.cz/~krska/elemvis, 2001

[7] Limpouch, A., X Window System programování aplikací, Grada 1993

[8] Marčuk, G. I., Metody numerické matematiky, Academia 1987

[9] Moorhead, R. J.; Zhifan, Z.; Signal Processing Aspects of Scientific Visualization, 2001
http://www.erc.msstate.edu/labs/vail/pubs/sig_proc/spmag_06.html

[10] Petřek, J., Analýza systému SIMULINK, ročníkový projekt, VUT Brno, 2000

[11] Schmidt, A., Siebert, K. G., ALBERT: An adaptive hierarchical finite element toolbox, Mathematische fakultät Universität Freiburg

[12] Sochor, J.; Žára, J.; Beneš, B. : Algoritmy počítačové grafiky, ČVUT Praha, 1998

[13] The MathWorks, Inc, Matlab online manual

[14] Zich, J., Základy počítačové grafiky a animace, Rešeršní práce, 1999