Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta-Katedra fyziky

Numerické výpočty MHD rovnic ve sluneční fyzice

Bakalářská práce

Vedoucí práce: RNDr. Petr Jelínek, Ph.D.

Autor: Marek Paraniak

Anotace

Tématem této bakalářské práce je seznámení s problematikou historie a používání „magnetohydrodynamických (MHD)“ rovnic pro modelování sluneční atmosféry a její aktivity. Začátek obsahuje úvod do historie a pozadí vývoje matematických metod, které tvoří základy většiny MHD simulačních programů. Dále je uveden popis instalace konfigurace a ovládání astrofyzikálního simulačního programu Athena 3.1 pod operačním systémem Linux. První část je věnována popisu programu a jeho instalaci. V další části je popsána struktura a konfigurace programu a nakonec je ukázáno praktické využití a několik testovacích výpočtů a vizualizace jejich výstupů.

Abstract

The subject of this bachelor thesis is the description of MHD „Magnetohydrodynamic“ equation and using for solar atmosphere activity modelling. It starts with brief description of history and background of development mathematical method used for numerical simulation. Furthermore mathematical algorithms used for most MHD modeling or simulation software. Next part of this thesis contain description of Athena 3.1 software structure and mainly its configuration under Linux system. Following part contains software structure description. Final part of this bachelor thesis shows some calculation examples and few visualization of software output.

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě Pedagogickou fakultou, elektronickou cestou ve veřejně přístupné části databáze STAG provozované Pedagogickou fakultou Jihočeské univerzity v Českých Budějovicích a na jejích internetových stránkách.

V Českých Budějovicích 20.1.2010 ……..…….

Obsah

Poděkování

Na tomto místě bych rád poděkoval svému vedoucímu bakalářské práce RNDr. Petru Jelínkovi, Ph.D. za četné cenné rady a připomínky při zpracování mé práce.

1 Magnetohydrodynamika

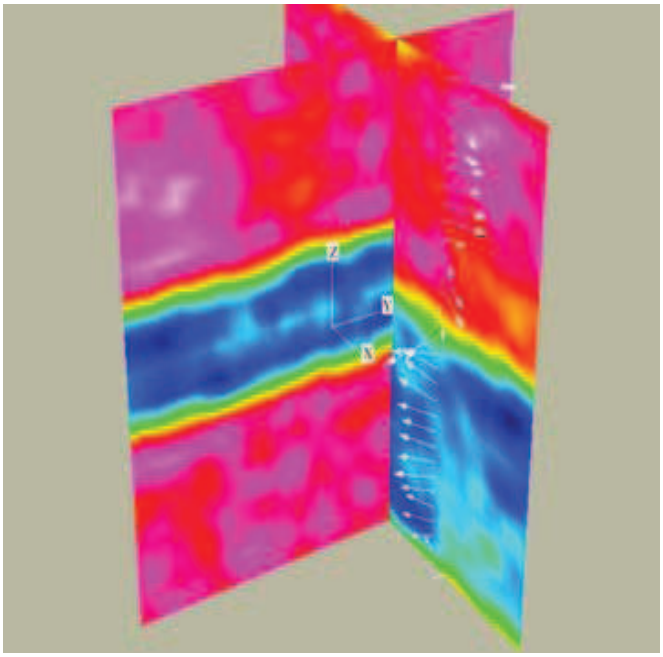
Kvalifikované odhady tvrdí, že více jak 95 % standardní hmoty v kosmu je ve formě plazmatu. Naše Slunce je také z převážné části tvořeno plazmatem.

Interakce plazmatu a magnetického pole Slunce dává vzniknout některým fascinujícím jevům. Jako jsou sluneční skvrny, sluneční erupce, sluneční vítr [1], koronální smyčky [2].

Porozumění co nejširšímu spektru sluneční aktivity nám dává lepší obraz, jak vzdáleného vesmíru, tak hlavně vlivu, jaký Slunce má na naší soustavu a převážně na naší vlastní planetu. Jednou z metod používaných k popisu a modelování chování sluneční atmosféry je magnetohydrodynamika (dále jen MHD).

Sluneční MHD je matematická disciplína, která se snaží popsat jednotným a co nejpřesnějším způsobem interakce mezi slunečním magnetickým polem plazmatem a sluneční atmosférou takovým způsobem, aby se dala ve výpočtech či matematických modelech použít jako kontinuální médium.

V současnosti jsme svědky revoluce v našem poznání sluneční aktivity. A jsme také svědky velikých pokroků na poli teoretického modelování a hlavně předpovídání sluneční aktivity. Ta má pro nás klíčový význam, například při modelování toku slunečního větru, viz. obrázek 1.



Obrázek 1: MHD model slunečního větru.

2 Historie a vývoj MHD

Historicky vychází vývoj MHD rovnic z Computational Fluid Dynamics, CFD [3] jak se často tato výpočetní metoda označuje, umožňuje pomocí matematických zákonitostí obecně modelovat proudění tekutin. Tedy můžeme vytvořit virtuální model zařízení či procesu a sledovat vývoj proudění v takto namodelovaném prostředí. Základem téměř všech MHD rovnic použitých ve sluneční fyzice jsou rovnice CFD od C. Naviera a G. G. Stokese, které byly publikovány v letech 1822 - 1845 a které popisují chování kapalin a plynů. Rovnice CFD obsahují:
rovnici kontinuity:

$$\frac{\partial \rho}{\partial \rho} + \frac{\partial}{\partial x_j}\{\rho u_j\} = 0 \tag{1}$$

a rovnici pro energii:

$$\frac{\partial}{\partial t}(\rho u_i) + \frac{\partial}{\partial x_j}\left[\rho u_i\, u_j + p\delta_{ij} - \tau_{ji}\right] = 0\,, i = 1,2,3 \tag{2}$$

Tyto rovnice mají jednu společnou charakteristiku. Musí mít nastaveny takzvané „hraniční podmínky“. Sice dokáží popisovat i hustotu kapaliny v různých teplotních režimech. Nicméně nedokáží popsat objemově veliké toky, které protékají v podmínkách sluneční atmosféry a nezahrnují do výpočtů vnitřní gravitaci hmoty. Mají jasně definovanou popisovanou plochu, která je z kosmologického hlediska poměrně malá.

Základy MHD byly položeny švédským fyzikem H. Alfvénem již v roce 1942. Za tyto a další práce v oboru sluneční fyziky obdržel v roce 1970 Nobelovu cenu.

Obrázek 2: Švédský fyzik Hannes Alfvén zakladatel magnetohydrodynamiky.

Mezi jeho hlavními objevy byly i tyto MHD rovnice, které popisují chování plazmatu v podmínkách sluneční tedy i hvězdné atmosféry.

Rovnice kontinuity:

$$\frac{\partial \varrho}{\partial t} = -\nabla.(\varrho \mathrm{v}) \tag{3}$$

Pohybová rovnice:

$$\frac{\partial \mathrm{v}}{\partial t} + \varrho(\mathrm{v}.\nabla) = -\nabla p + \frac{1}{\mu_0}(\nabla \mathrm{x} \mathrm{B})\mathrm{x}\mathrm{B} \tag{4}$$

Indukční rovnice:

$$\frac{\partial \mathbf{B}}{\partial t} = \nabla \mathrm{x}\ (\mathrm{v}\ \mathrm{x}\ \mathbf{B}) \tag{5}$$

Rovnice pro energii:

$$\frac{\partial \mathbf{U}}{\partial t} = -\nabla\ \mathbf{S} \tag{6}$$

Maxwellova rovnice pro divergenci magnetického pole:

$$\nabla.\mathbf{B} = 0 \tag{7}$$

Celková energie je vyjádřená jako součet tlakové, kinetické a magnetické energie:

$$U = \frac{p}{\gamma - 1} + \frac{\varrho}{2} v^2 + \frac{\boldsymbol{B}}{2\mu_0} \tag{8}$$

Rovnice pro tok:

$$\mathrm{S} = \left(U + p + \frac{B^2}{2\mu_0}\right).\mathrm{v} - (\mathrm{v}.\mathbf{B})\frac{\mathrm{B}}{\mu_0} \tag{9}$$

kde ρ = hustota, u = rychlost proudění, B = magnetické pole, p =tlak plynu.

3 Athena 3.1

Athena 3.1 je software založený na Godunovových algoritmech vyššího řádu a je primárně určen pro výpočty hlavně astrofyzikální dynamiky plynů. Vývoj softwaru byl podporován v rámci programu NSF ITR (Information Technology Research for National Priorities) [4]. Součástí projektu je také zveřejnění softwaru společně s dokumentací a výukovými materiály a jeho uvolnění pro širokou komunitu astrofyziků.

V aktuální verzi Athena 3.1 jsou implementovány tyto algoritmy:

a.) magnetohydrodynamické rovnice [5] pro výpočty v jedno, dvou a tří rozměrných prostorech.

b.) Stavová rovnice ideálního plynu s libovolně stanovenou hodnotou γ (včetně $\gamma = 1$, isotermická stavová rovnice).

c.) Interpolace charakteristik prvního, druhého a třetího řádu pomocí primitivních proměnných.

d.) Číselné toky vypočtené pomocí aproximačních, nebo lineárních Riemanovských řešitelů. Počáteční podmínky stanovené pomocí statického gravitačního potenciálu.

e.) Vlastní gravitace vypočtená pomocí FFT (Fast Fourier Transformation) [6].

f.) Libovolný počet pasivních skalárů přenášených tokem fluida.

g.) Paralelizace výpočtů pomocí rozkladu pomocí MPI domény.

Algoritmus a kód je vyvíjen kolektivem těchto spolupracovníků Jim Stone (Princeton University), Tom Gardiner (Cray Research), Peter Teuben (University of Maryland) a John Hawley (University of Virginia). Vývoj kódu začal v roce 2000 a stále probíhá.
Primární motivací pro vývoj Atheny 3.1, byla adaptace Godunovových [7] rovnic pro astrofyzikální výpočty magnetohydrodynamických stavů hmoty. Použitím jednokrokovým Eulerovým algoritmem je možné algoritmus použít společně s metodou AMR - Adaptive mesh refinement [6]. Kód je optimalizován a implementován pro použití na moderních výkonných procesorech a pamětech.

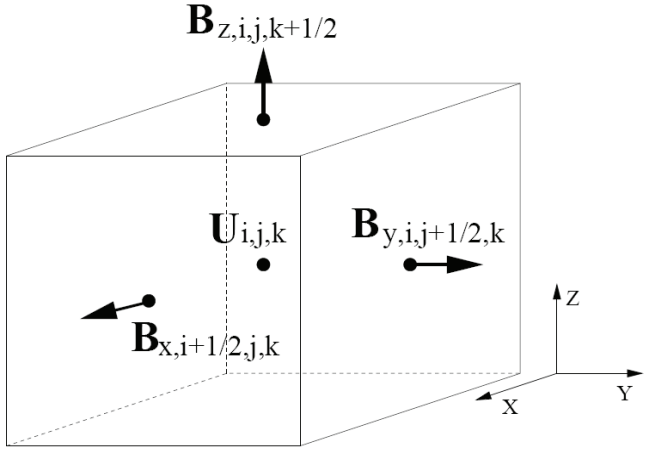
4 Doménová struktura

Informace o celé výpočetní doméně jsou uvedeny v doménové struktuře:

```
typedef struct Grid_Block_s{
int ixs, jxs, kxs; /* Minimální počet souřadnic v tomto bloku */
int ixe, jxe, kxe; /* Maximální počet souřadnic v tomto bloku */
int my_id;         /* process ID (rank procesu při paralelním
                   zpracování pomocí MPI) */
}Grid_Block;
typedef struct Domain_s{
Grid_Block ***grid_block; /* 3D pole výpočetních bloků vyskládané
                   v této doméně */
int ixs, jxs, kxs; /* Minimální počet souřadnic buněk v celé
                   doméně */
int ixe, jxe, kxe; /* Maximální počet souřadnic buněk v celé
                    doméně */
}Domain
```

Doména obsahuje pole výpočetních bloků, které po integraci informací o souřadnicích vytvoří souvislý prostor. Informace o každé výpočetní buňce a stejně tak o příslušných proměnných v doméně jsou organizovány do struktury zvané Grid. Příslušné proměnné v buňkách jsou organizovány obdobně.

Na obrázku 3 je patrné umístění magnetických polí na čelních stranách buněk a uložení proměnných uvnitř buněk ve struktuře zvané GAS (plyn). Hlavní grid obsahuje plynové struktury, magnetické pole umístněné na přední části buňky a informaci o souřadnicích.



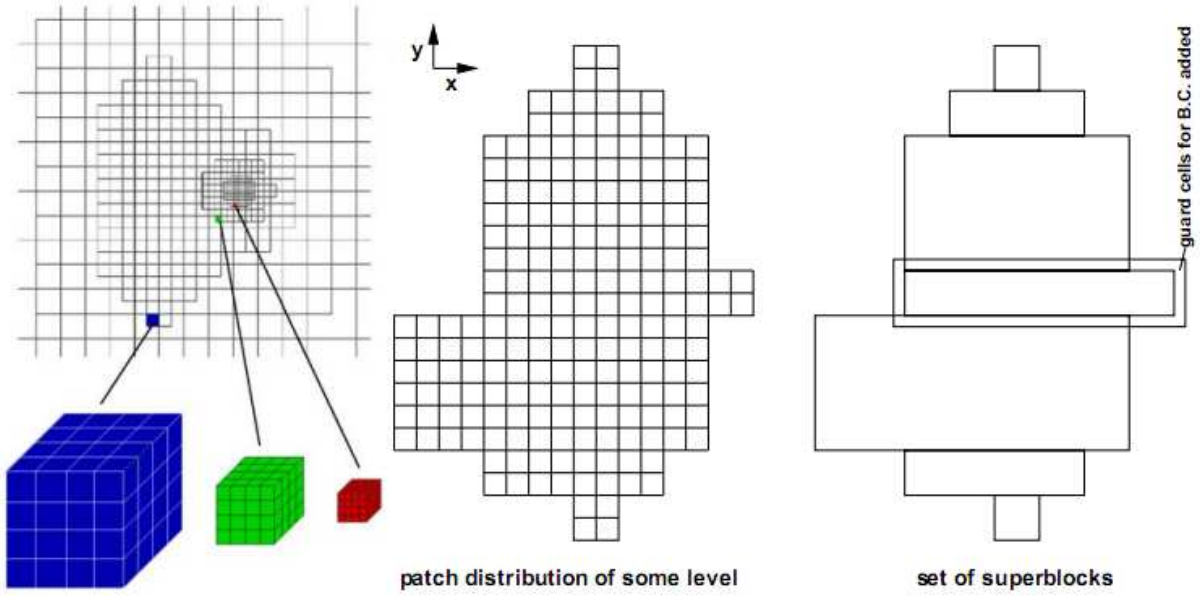
Obrázek 3: Výpočetní doména struktura.

Struktura gridu v kódu programu je popsána zde:

```
typedef struct Grid_s{
Gas ***U;          /* ukazatel na  3D „plynové“ pole */
#ifdef MHD
Real ***B1i,***B2i,***B3i; /* ukazatel na 3D pole rozhraní „B“ */
#endif             /* MHD */
Real x1_0;         /* x1-počáteční pozice souřadnice ix = 0 */
Real x2_0;         /* x2- počáteční pozice souřadnice jx = 0 */
Real x3_0;         /* x3- počáteční pozice souřadnice kx = 0 */
Real dx1,dx2,dx3;  /* velikost buňky */
Real dt,time;      /* časový krok, absolutní čas */
int nstep;         /* počet provedených integračních kroků */
int Nx1,Nx2,Nx3;   /* počet zón v osách  x1, x2, x3 */
int is,ie;         /* start/konec indexu buněk v ose x1 */
int js,je;         /* start/konec indexu buněk v ose x2*/
int ks,ke;         /* start/konec indexu buněk v ose x3*/
int idisp;         /* souřadnice ix = index i + idisp */
int jdisp;         /* souřadnice jx = index j + jdisp */
int kdisp;         /* souřadnice kx = index k + kdisp */
char *outfilename; /* základní jméno pro výstupní soubory */
int my_id;         /* ID procesu (nebo ranku v MPI),
                   který aktualizuje Síť */
int nproc;         /* celkový počet procesů ve výpočtu */

int rx1_id, lx1_id;/*ID gridů R/L ve směru x1 (implicitně = -1)*/

int rx2_id, lx2_id;/*ID gridů R/L ve směru x2 (implicitně = -1)*/

int rx3_id, lx3_id;/*ID gridů R/L ve směru x3 (implicitně = -1)*/
}Grid;
```

5 Metoda AMR - Adaptive Mesh Refinement

S rozvojem výpočetní techniky byly objeveny i nové metody modelování plazmatu. Jednou z nich je i metoda AMR - Adaptive Mesh Refinement. [3]
Výše zmíněná metoda a algoritmus na ní založený umožňuje měnit flexibilně přesnost výpočtů pro požadovanou oblast. Čili můžeme zanedbat oblasti, které nás nezajímají a flexibilně změnit přesnost v požadované fyzikální oblasti.



Obrázek 4: Výpočetní síť metody AMR použitá v programu Athena 3.1.

AMR metoda rozkládá složitý výpočet do tzv. „Dynamické domény" umístěné v kartézských souřadnicích. Tato doména je složena z velikého množství tzv. buněk. Jejich počet se může měnit v závislosti na složitosti výpočtu a výpočetním výkonu zařízení.

6 Linux výběr distribuce

Výše jmenovaný program je velmi flexibilně napsán v zásadě pro jakoukoliv Linuxovou distribuci. Je tedy čistě na uživateli jakou zvolí, ale ukázalo se během testování na distribucích Ubuntu, Mandrake, Opensuse, Debian, Dreamlinux, Slackware, že velmi záleží na jádře systému respektive na jeho úpravách výrobcem. Některé distribuce např. oblíbené Ubuntu, Mandrake, nebo OpenSuse velmi zasahují do samotného jádra, takže kompilování aplikace příkazem `make` selhávalo.

Proto jsme pro zkušební instalaci zvolili distribuci Dreamlinux, která používá jádro ze systému Debian. Na této distribuci při instalaci a kompilaci programu k chybám nedocházelo.

7 Získání programu

Program Athena 3.1 je volně ke stažení na internetu [9], má velmi mnoho vlastností a pro jeho plné využití je třeba dobře prostudovat manuál umístěný ve složce **doc** v instalačním adresáři. Zde jsou některé nejvýznamnější vlastnosti:

a.) Paralelní zpracování dat (pro fungování na víceprocesorových clusterech je třeba používat MPI knihovnu MPICH [10]).

b.) Vizualizace výstupů.

c.) Pro 1D vizualizaci výstupu je možno používat SuperMongo® [11].

d.) Pro 2D vizualizace ImageMagick® [12].

e.) Pro 3D vizualizace OpenDX® [13], nebo VisIt® [14] tento program je, ale vhodný pouze pro běh na výkoných clusterech.

8 Instalace programu

Příkazem `tar -xvf` rozbalíme program do libovolného adresáře na stanici. Samozřejmě v něm musíme mít nastavená příslušná přístupová práva. Po rozbalení se vytvoří adresářová struktura zobrazená v tabulce 1:

Tabulka 1: Adresářová struktura programu Athena 3.1 po rozbalení.

/athena 3.1	
/doc	Dokumentace,manuály(v AJ) tento adresář mimo jiné obsahuje „Programers Guide“ kde jsou vysvětleny použité algoritmy
/src	Zdrojový kód, a soubory include (nutné pro kompilaci.)
/prob	Soubory s předefinovanými problémy k výpočtu (viz**: /src/problem.c** pro symbolické linky.)
/tst	Různé testovací vstupní soubory pro řešení problémů
/1D-hydro	Úlohy 1 D Hydro.
/1D-mhd	Úlohy 1 D Magneto Hydro Dynamické
/2D-mhd	Úlohy 2 D Magneto Hydro Dynamické
/3D-mhd	Úlohy 3 D Magneto Hydro Dynamické
/vis	Vizualizační nástroje a skripty
/dx	Skripty pro vizualizaci výstupních dat externím programem OpenDX.
/idl	Skripty pro vizualizaci výstupních dat externím programem IDL .
/sm	Skripty pro vizualizaci výstupních dat externím programem Super Mongo
/vtk	Kód pro připojení VTK soubor

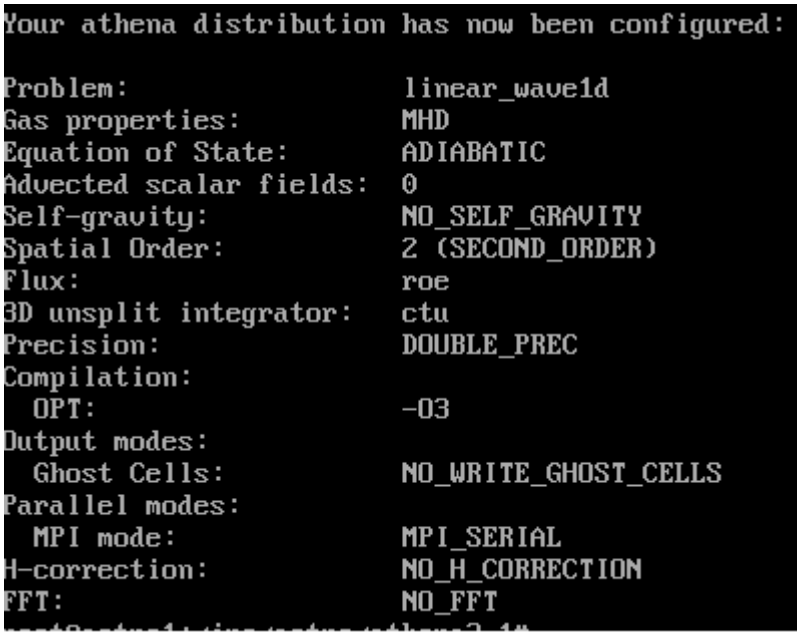
Kromě adresářů výše zmíněných se po kompilaci vytvoří ještě následující adresář /**bin** ten obsahuje spustitelné soubory vytvořené příkazem `make` ze souboru **makefile**.

Konfigurační skript vytvoříme spuštěním příkazu `autoconf` v domovském adresáři.

Instalace otestujeme touto sekvencí příkazů:

```
configure
make all
make test
```

Po úspěšné implicitní kompilaci vypadá implicitní výstup zhruba takto:



Obrázek 5: Implicitní výstup po úspěšné základní konfiguraci.

Implicitní nastavení je následující:

- Plyn: MHD
- Stavová rovnice: Adiabatická.
- Gravitace: Bez vlastní gravitace
- Prostrový řád : 2 řádu
- Tok : Roe
- Přesnost: Dvojitá přesnost
- MPI (Multi processing): MPI seriové atd.

Pokud se na obrazovce neobjeví žádné chyby, byla instalace úspěšná. Na linuxové distribuci Dreamlinux bylo v tomto případě ještě potřeba doinstalovat interpret příkazů **csh**.

9 Konfigurování Atheny 3.1

Po nainstalování programu je dalším důležitým krokem konfigurace programu Athena 3.1 pro vyřešení specifického fyzikálního problému. Důležitým souborem pro provedení tohoto úkolu je soubor `configure`, který má několik základních funkcí. Především slouží k vypínání, nebo zapínání specifických funkcí (features) programu a k výběru balíků funkcí implementovaných v Atheně 3.1 Tyto funkce se zapínají příkazem „On“ (Zapnuto), nebo „Off“ (Vypnuto). Například pokud je nutno použít funkce (Features), jednoduché (Single precision), nebo dvojité (Double precision) přesnosti, nebo pro nastavení výstupního formátu souboru, (např. pro různé vizualizační nástroje). Balíky funkcí (packages) se používají v případě, že řešený problém se více větví a má více než jednu možnost řešení. Základní balíky fyzikálních funkcí například obsahují hydrodynamické (hydrodynamics), nebo magnetohydrodynamické úlohy (magnetohydrodynamics), adiabatickou, nebo isotermickou stavovou rovnici, ale také různé algoritmické funkce, stupeň přesnosti ve výpočtu, Riemanovský řešitel [15] atp.

Možnosti těchto funkcí jsou řízeny na na úrovní kódu jazyka C pomocí předkompilovaných makrofunkcí. Takže není nutné, aby uživatel přímo editoval žádné specifické soubory.
Pokročilejší funkcí skriptu `configure` je nastavování kompilátoru a linkeru tak, aby reflektoval aktuální stav systému po vygenerování skriptu příkazem `autoconf.`
Používá se následující syntaxe:

```
configure [--enable -funkce] [--disable-funkce] [--with-balik funkcí=volba]
```

Funkce a balíky funkcí jsou platnými volbami v programu Athena 3.1 volba je hodnota, na kterou se balík funkcí má nastavit. Platné funkce pro Athenu 3.1 jsou uvedeny v tabulce 2, dále jsou v tabulce 3 uvedeny i platné balíky funkcí spolu se všemi nastavitelnými hodnotami.

V tomto softwaru je nutno pro řešení specifického problému vždy znovu provézt kompilaci s novými hodnotami, nastavenými buď v příslušném konfiguračním souboru, nebo jako parametr na příkazové řádce. Jednotlivá řešení je možno postupně vyplňovat do souboru s libovolným názvem například: prob_zadani_jmeno.

Příkazem `chmod -777 prob_zadani_jmeno` zajistíme, že se tento soubor stane spustitelným. Do tohoto souboru je možné jednotlivě řešené úlohy zadávat a při příštím použití jednoduše odkomentovat řádku s příslušným příkazem (čili odstranit znak #).
Vhodné je také připojit nad příslušnou řádku komentář s popisem řešeného problému, pro případné další použití.

Dalším možným krokem je využití pokročilých funkcí linuxového shellu. Při jeho spuštění stačí zadat tuto klávesovou zkratku Ctrl+R a je možné hledat podle počátečních slov v historii příkazů shellu. Nevýhodou je, že zde je zaznamenáno vše, co jsme do shellu zapsali. Tedy i příkazy které se spouštění programu Athena 3.1 netýkají.

Pro řešení problému uvedeného v tabulce 3 (Úlohy jsou vzájemně nespojitelné), je možno vybrat balík pro řešení konkrétního problému, je možné jej i upravit případně i přidat řešení nového problému, který software zatím neumí řešit. A to tak, že se vytvoří soubor s příslušným názvem problému a uloží se do souboru v adresáři **Athena 3.1/src/prob/**.

Tabulka 2: Funkce programu Athena 3.1

Funkce	Implicitní nastavení	Popis
single	vypnuto	Výpočty jsou prováděny s jednoduchou přesností (implicitně dvojitá přesnost)
debug	vypnuto	Kompiluje kód s flagy nutnými pro debugging (analýzu chyb při běhu programu)
ghost	vypnuto	Způsobí vypsání ghost zones při výstupu
mpi	vypnuto	Paralelizace výpočtu pomocí knihovny MPI
h-correction	vypnuto	H-korekce pro eliminování
fft	vypnuto	Zkompiluje a linkuje FFTW blokovou dekompozici (Fourierova transformace)

Tento adresář slouží pro inicializaci řešení jednotlivých problémů. V základní konfiguraci obsahuje Athena 3.1 již velké množství předkonfigurovaných balíků s připravenými řešiteli, kteří jsou již připraveni pro různé fyzikální úlohy. (Implicitní soubor v defaultním nastavení Atheny 3.1 je **Athena 3.1/src/prob/linear_wave1d.c** –with-problem=linear_wave1d).

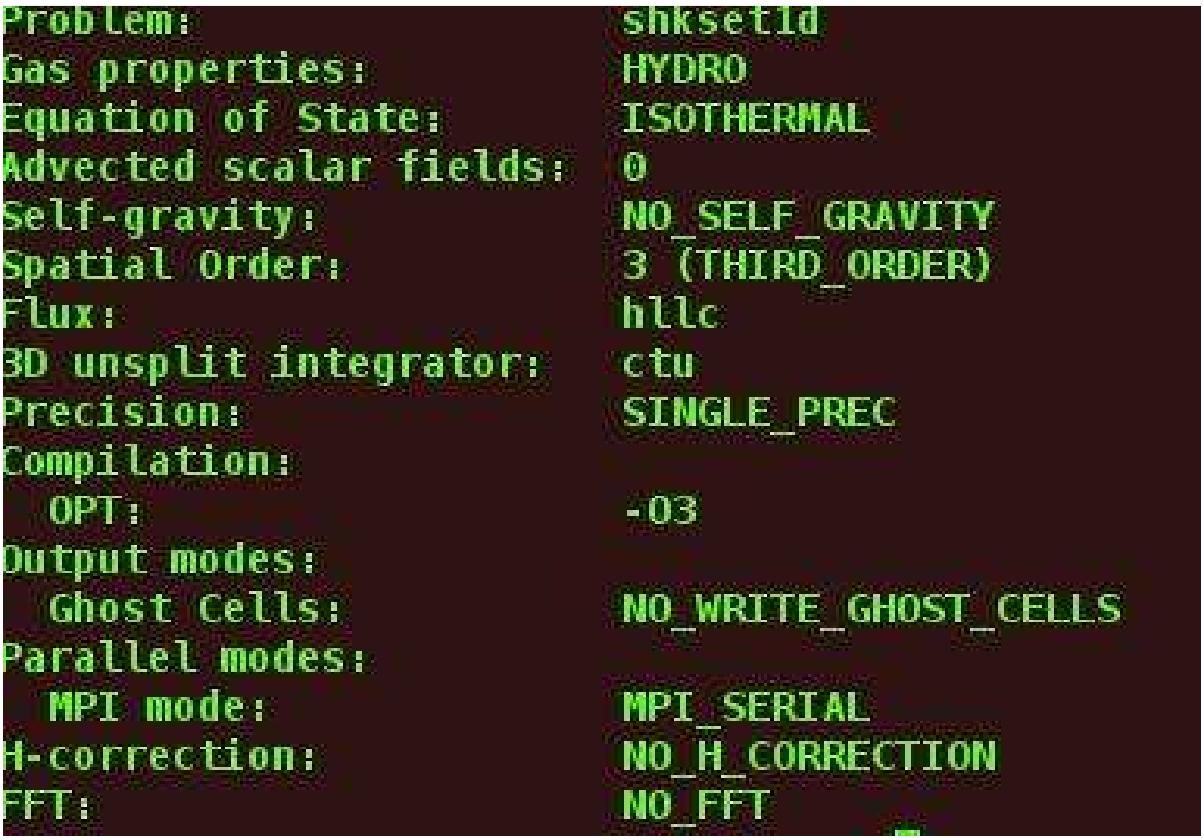
Popis jednotlivých souborů a algoritmů je uveden v programátorské příručce. Mějme na paměti, že skript `configure` vytvoří symbolické linky mezi programem `Problem.c` v adresáři **athena 3.1/src/** a příslušným souborem **athena 3.1/src/prob/**

Skript `configure` musí být zásadně spouštěn v kořenovém adresáři aplikace, tedy **athena 3.1/** Pokud spustíme skript `configure` s parametrem – `help` (`configure --help`), bude vypsáno větší množství informací včetně seznamu možných funkcí a výpočetních balíků.

Například pokud chceme konfigurovat Athenu 3.1 pro modelování isotermického hydrodynamické rázové vlny inicializované pomocí Roeova magnetického toku přes interpolaci třetího řádu s jednoduchou přesností, použijeme následující konfigurační zápis:

```
configure --with-problem=shkset1d --enable-single --with-
eos=isothermal --with-gas=hydro --with-order=3
```

Výstup na obrazovku po výše uvedené kompilaci vypadá pak takto:



Obrázek 6: Výstup na terminál po úspěšné konfiguraci „modelování hydrodynamické rázové vlny inicializované pomocí Roeova magnetického toku přes interpolaci třetího řádu s jednoduchou přesností stavová rovnice isotermická“.

Pro konfiguraci Athena 3.1 pro spuštění testu lineární vlny ve 3D adiabatické magnetohydrodynamiky s použitím HLLD toku pomocí van Leerova integrátoru a interpolaci druhého řádu ve dvojité přesnosti a paralelizovaným výpočtem, napíšeme na příkazovou řádku toto:

```
configure --with-flux=hlld --with-problem=linear wave3d --with-
integrator=vl --enable-mpi
```

Výstup na obrazovku po výše uvedené kompilaci vypadá takto:

```
Your athena distribution has now been configured:

Problem:                  linear_wave3d
Gas properties:           MHD
Equation of State:        ADIABATIC
Advected scalar fields:   0
Self-gravity:             NO_SELF_GRAVITY
Spatial Order:            2 (SECOND_ORDER)
Flux:                     hlld
3D unsplit integrator:    ctu
Precision:                DOUBLE_PREC
Compilation:
  OPT:                    -O3
Output modes:
  Ghost Cells:            NO_WRITE_GHOST_CELLS
Parallel modes:
  MPI mode:               MPI_SERIAL
H-correction:             NO_H_CORRECTION
FFT:                      NO_FFT
```

Obrázek 7: Výstup na terminál po úspěšné konfiguraci výpočtu lineární vlny ve 3D MHD s použitím HLLD toku s pomocí van Leerova integrátoru a interpolaci druhého řádu ve dvojité přesnosti a paralelizovaným výpočtem stavová rovnice adiabatická.

Skript `configure` vytvoří pro každou specifickou úlohu vlastní soubor `makefile` v adresáři **athena 3.1/src/** a jako vzor použije soubor **athena 3.1/src/makefile.in**. Po úspěšném spuštění `configure` zavolá parametry uvedené na příkazové řádce včetně těch implicitních.

10 Kompilace Atheny 3.1

Po spuštění skriptu `configure` je ještě nutné zkompilovat kód v kořenovém adresáři aplikace, tedy **athena 3.1/** spustíme tento příkaz pro kompilaci:

```
make all
```

Tímto se automaticky vytvoří adresář **athena 3.1/bin/** a spustitelný soubor. Dále se spustí příkaz make v adresáři **Athena 3.1/src/**, který zkompiluje a linkuje kód. Soubor `makefile` umístěný v kořenovém adresáři aplikace **athena 3.1/** také obsahuje další volby uvedené v následující tabulce:

Tabulka 3: Balíky funkcí

Balík	Výběr	Poznámka
Úloha	Jméno souboru	Použité jméno souboru v adresáři **athena 3.1/src/** po počáteční podmínky
Gas	Hydro Mhd*	Vytvoří kód pro hydrodynamiku Vytvoří kód pro MHD
Eos(stavová rovnice)	Adiabatic isothemal	Použije adiabatickou stavovou rovnici Použije isotermickou stavovou rovnici
Nscalars	#	Přidá #pasivní skaláry (implicitní počet je 0)
Gravity	fft	Zapne vnitřní gravitaci s použitím FFT (Fast Fourier transform)
Flux (tok)	roe force hlle hllc hlld	Roe-Riemanův [15] řešitel SILOVÝ magnetický tok HLLE Riemanův řešitel [16] HLLC Riemanův řešitel [17] (pouze Hydrodynamický) HLLD Riemanův řešitel (pouze MHD magnetohydrodynamický)
Řád	1 2 3	Prostorová rekonstrukce 1 řádu Prostorová rekonstrukce 2 řádu (lineární) Prostorová rekonstrukce 3 řádu
Integrátor	ctu vl	Rohový nedělený 3d integrátor van Leerův 3d nedělený integrátor

Tabulka 4: Parametry na příkazové řádce

Cíl	Poznámka
`all`	Vytvoří adresář **/athena 3.1./bin** a zkompiluje kód do spustitelného stavu
`compile`	Kompiluje kód
`clean`	Odstraní soubory s příponou .o z adresáře **/athena 3.1./src**
`help`	Vyvolá nápovědu
`test`	Spustí test instalace

Obvykle se soubory `makefile` nikdy neupravují ručně. Nicméně je možné `makefile` soubory uložené v adresáři **/athena 3.1./src** editovat (např. pro lepší optimalizaci výpočtu) a změnit flags kompilátoru raději než upravovat proměnné v prostředí. Ale je nutné mít na paměti, že při každé nové kompilaci programem `configure`, budou všechny soubory `makefile` přepsány a tím pádem veškeré ručně vytvořené úpravy kódu ztraceny.

Pokud tedy chceme do kódu zavést úpravy, které mají být stálé, musíme upravit šablonu `makefile.in` umístěnou taktéž v adresáři **Athena 3.1./src**. Nejlepším způsobem pro úpravu parametrů kompilátoru a cest k lokálním knihovnám je změna parametru MACHINE= na příkazové řádce souboru `make`.

Je možné zkompilovat program pro běh na několika stanicích současně, pokud je uvedeme na příslušném místě v souboru `makefile.in` a při příštím spuštění programu `configure` už není nutné zadávat znova do parametru názvy příslušných stanic. Pokud parametr MACHINE= neuvedeme, program použije jako implicitní hodnotu optimalizační úroveň 03 v gcc kompilátoru.

10.1 Kompilace bez příkazu configure

Většina fyzikálních parametrů v programu Athena 3.1 je řízena, nastavována a ovládána za pomoci předkompilovaných makrofunkcí. Kompletní sada maker je uložena v souboru **/athena 3.1./src/defs.h.in.** Skript `configure` vytvoří hlavičkový soubor **/src/defs.h**, který obsahuje sadu maker potřebnou pro řešení konkrétního fyzikálního problému. Můžeme si, ale vytvořit vlastní soubor **defs.h** a řešit i úplně nové problémy, které Athena 3.1, ještě sama neumí. Tímto získáváme velikou flexibilitu výpočetních schopností programu. (Nicméně je třeba mít na paměti, že při nové kompilaci programem `configure` bude soubor **/src/defs.h** přepsán a tím pádem veškeré ručně vytvořené úpravy kódu ztraceny) Kroky kompilace jsou řízeny souborem `makefile`, který je vygenerován skripty z adresáře **src/makefile.in**.

Pokud je nutno obejít konfigurační skript musí být soubory `makefile` vytvořeny ze šablon ručně. Samozřejmě stejně jako v předchozích případech budou po opětovném spuštění skriptu `configure` veškeré změny ztraceny.

10.2 Spouštění programu Athena 3.1

Po úspěšné kompilaci programu musí existovat ve složce **/athena 3.1/bin** spustitelný soubor `athena`. Existují dva způsoby jak spouštět program. Editace vstupního souboru bude popsána níže v dokumentu:

- Ruční editace parametrů ve zpracovávaném souboru
- Přímé spuštění programu s parametrem -i ....cesta/k/souboru.

Například pro výpočet rázové vlny Brio & Wu v 1D zadáme do shellu tento příkaz:

```
athena –i   /tst/1D-mhd/athinput.brio-wu
```

Program nejdříve zavolá hodnoty všech vstupních parametrů na standardní výstup stdout. Během hlavní integrační smyčky vypíše na obrazovku cyklické hodnoty a časovou značku a po skončení činnosti vypíše výsledek na obrazovku. Do programu Athena 3.1 jsou implementovány některé funkce, které se zadávají jako parametr na příkazovém řádku.
Příkaz `athena -h` vypíše tyto následující parametry na obrazovku:

```
athena -h

Athena version 3.1 - 01-JAN-2008
Last configure: Wed Jan 9 09:25:53 EST 2009
Usage: athena [options] [block/par=value ...]
Options:
-i <file> Alternativní vstupní soubor [athinput]
-d <directory> Alternativní běhový adresář [current dir]
-h Vytiskne tuto nápovědu
-n Předá vstupy, ale nespustí program
-c Vypíše konfigurační údaje a skončí
-r <soubor> Restartuje simulaci se příslušným souborem
Configuration details:
Problem: linear_wave3d
```

```
Gas properties: MHD
Equation of State: ADIABATIC
Passive scalars: 0
Self-gravity: none
Order of Accuracy: 2 (SECOND_ORDER)
Flux: hlld

Unsplit 3D integrator: ctu
Precision: DOUBLE_PREC
Output Modes:
Ghost Cells: disabled
Parallel Modes:
MPI: MPI_SERIAL
H-correction: disabled
FFT: disabled
```

Parametr `-d` vytvoří nový adresář a do něj uloží soubory vytvořené programem. Parametr `-n` je důležitý pro vypsání ladících informací na obrazovku.

Jakýkoliv platný vstupní parametr, který je součástí vstupního souboru, je možné zadat i z příkazového řádku a tato volba má přednost před parametry uvedenými ve vstupním souboru.

Toho je možné úspěšně využít pro ladění programu společně s parametrem `-d`. Pokud je použit parametr `-c` program vypíše na obrazovku parametry nastavení programu, s kterými byl zkompilován.

10.3 Úprava vstupního souboru

Běhové parametry programu jsou definovány ve vstupním souboru. Ten obvykle nese jméno `athinput.jméno_problému` kde `jméno_problému` je řetězec string. Často je tento řetězec stejný jako název souboru vytvořený generátorem úloh ve složce **athena/src/prob**, který je použit pro inicializování dat.

V některých případech složitějších problémů je nutné definovat jméno souboru ručně (některé generátory funkcí se mohou využít pro řešení více problémů najednou). Jako příklad parametrů vstupního souboru programu Athena 3.1 je zde vybrán soubor **athena 3.1/tst/1d-mhd/athinput.brio-wu.** Další řešené úlohy jsou uloženy v adresáři **athena 3.1/tst/**. Konfigurační soubor vypadá takto, pro přehlednost jsou jednotlivé parametry uvedeny v následující tabulce:

Tabulka 5: Parametry konfiguračního souboru

Parametr	Hodnota	Popis parametru:
maxout	= 3	# Maximální počet výstupních bloků souborů
<Output1>		
out_fmt	= tab	# Tabulátorový datový výstup
dt	= 0.0025	#Časový přírůstek mezi výstupy
<Output2>		
out_fmt	= hst	# Historie datového výstupu
dt	= 0.0025	# Časový přírůstek mezi výstupy
<Output3>		
out_fmt	= bin	# Binární datový výstup
dt	= 0.0025	# Časový přírůstek mezi výstupy
<Time>		
cour_no	= 0.8	# Číslo courant, friedrichs, & lewy (cfl)
nlim	= 10000	# Max. Počet cyklů
tlim	= 0.1	# Časový limit
<Grid>		
Nx1	= 800	# Počet zón ve směru x1
x1min	= 0.0	# Minimální hodnota zóny x1
x1max	= 1.0	# Maximální hodnota zóny x1
ibc_x1	= 2	# Zarážka vnitřní hranice zóny (x1)
obc_x1	= 2	# Zarážka vnější hranice zóny (x1)
Nx2	= 1	# Počet zón ve směru x2
x2min	= 0.0	# Minimální hodnota zóny x2
x2max	= 1.0	# Maximální hodnota zóny x2
ibc_x2	= 2	# Zarážka vnitřní hranice zóny (x2)
obc_x2	= 2	# Zarážka vnější hranice zóny (x2)
Nx3	= 1	# Počet zón ve směru x3
x3min	= 0.0	# Minimální hodnota zóny x3
x3max	= 1.0	# Maximální hodnota zóny x3
ibc_x3	= 2	# Zarážka vnitřní hranice zóny (x3)
obc_x3	= 2	# Zarážka vnější hranice zóny (x3)
NGrid_x1	= 1	
NGrid_x2	= 1	
NGrid_x3	= 1	
<problem>		
gamma	= 2.0	# Gamma = c_p/c_v
dl	= 1.0	# Hustota v levé polovině souřadnic
pl	= 1.0	# Tlak
v1l	= 0.0	# Rychlost ve směru x
v2l	= 0.0	# Rychlost ve směru y
v3l	= 0.0	# Rychlost ve směru z
b1l	= 0.75	# Magnetické pole v x
b2l	= 1.0	# Magnetické pole v y
b3l	= 0.0	# Magnetické pole v z
dr	= 0.125	# Hustota v pravé polovině souřadnic
pr	= 0.1	# Tlak

v1r	= 0.0	# Rychlost ve směru x
v2r	= 0.0	# Rychlost ve směru y
v3r	= 0.0	# Rychlost ve směru z
b1r	= 0.75	# Magnetické pole v x
b2r	= -1.0	# Magnetické pole v y
b3r	= 0.0	# Magnetické pole v z
hk_dir	= 1	# Směr pohybu vlny (výbuchu) -- (1,2,3) = (x1,x2,x3)

Parametry jsou seskupeny do pojmenovaných bloků. Název každého bloku je definován v hranatých závorkách na jednotlivé řádce nad příslušnými parametry.
Prázdná řádka nad, nebo pod názvem bloku je ignorována. Pod každým názvem bloku je seznam parametrů ve formátu syntaxe:

```
název_parametru = hodnota #komentář
```

Mezery po názvu parametru, stejně tak mezera po znaku =, jsou ignorovány. Všechno za znakem a včetně znaku #, je také ignorováno. Hodnota pro každý parametr musí být uvedena vždy na zvláštní řádce. Pro dokumentaci jednotlivých parametrů jsou umožněny komentáře za znakem #. Maximální počet znaků v jedné řádce je omezen na 256. Název bloku i parametr jsou též citlivé na velikost písmen. Čili B3r b3R a b3r znamenají pokaždé jiný parametr. Vstupní soubor je čten velmi flexibilním programem zvaným „Parser“ speciálně napsaným pro program Athena 3.1. Ten je umístěn v **/athena 3.1./src/par.c**.

Celý vstupní soubor je načten na samém počátku běhu hlavního programu a názvy parametru stejně tak i jejich hodnoty jsou umístěny v paměti. Při běhu jednotlivých procedur programu je k nim možno přistupovat. Parser umožňuje, aby názvy parametrů mohy být v bloku uvedeny v libovolném pořadí a dále také řeší to, aby nekorektně napsané či neplatné parametry nebyly zahrnuty do výpočtu. Ve vstupním souboru nikdy nejsou nastaveny nějaké implicitní hodnoty pro běhové parametry. Pro každý požadovaný parametr musí být ve vstupním souboru uvedena hodnota. Pokud Parser (překladač) potřebuje pro výpočet určitou hodnotu nějakého parametru uvést ve vstupním souboru a tato hodnota ve vstupním souboru uvedena není, Parser pošle na standardní výstup chybovou hlášku a ukončí běh hlavního programu. V tomto případě je název chybějícího parametru detekován při běhu programu.

Parametrem může být číslo integer, číslo s plovoucí čárkou, nebo řetězec. Pokud je to nutné program Parser provede automatickou konverzi datových typů např. číslo s plovoucí čárkou na typ double (předpokládá se ovšem, že uživatel bude znát rozdíly mezi základními datovými typy real, integer, string atd..) Parametry je možné samozřejmě zadávat také pomocí příkazové řádky, což

umožňuje velmi pružně testovat kód případně i hledat potřebné hodnoty parametrů syntaxe příkazu je „`block/parametr=hodnota`". V kapitolách níže jsou popsány jednotlivé bloky parametrů i jednotlivé parametry.

10.3.1 Blok <comment>

Slouží pro komentáře k programu a k jeho zpřehlednění. Hodnoty zde uvedené nejsou použity pro výpočty programu.

10.3.2 Blok <Job>

Parametry v tomto bloku slouží k řízení úloh, které Athena 3.1 provádí a jsou použity programem `main.c`.

- problem_id: řetězec použitý jako jméno výstupního souboru. Obvykle stejné jako jméno vstupního souboru. Maximální délka názvu je omezena na 256 znaků.
- maxout: udává maximální počet bloků které jsou načteny ze vstupního souboru. Ve výstupních blocích od <output1> až do <outputN>, (kde N= hodnota maxout) jsou hledány platné hodnoty. Chybějící výstupní bloky jsou ignorovány.

10.3.3 Blok <Output#>

Parametry zde uvedené přímo, ovlivňují charakter dat na výstupu. Například binární soubor, výpis programu, obrázek atd.
out: Proměnná pro formáty obrázků, pgm, ppm, fits.
Aktuálně povolené hodnoty jsou: M1, M2, M3, E, B1c,B2c, B3c, ME, V1, V2, V3, P, S, cs2. Pokud je hodnota proměnné nastavena na all bude výstup obsahovat d, M1, M2, M3 a v závislosti na konfiguraci také může obsahovat E, B1c, B2c, B3c.
Výstupní typ souboru musí být jeden z těchto formátů: bin, dx, hst, tab, rst, vtk.
out_fmt: výstupní formát např. bin, dx, hst, tab, rst, vtk, fits, pdf, pgm, ppm.
Detailní popis formátů je uveden v manuálu.

a.) dat_fmt: Nepovinné pole pro řetězec, který nastavuje tabulátorový výstup do souboru (např,12%.5e). Tato hodnota nesmí být v uvozovkách ani nesmí obsahovat mezery.
b.) dt: časový přírůstek mezi výstupy (při výpočtu problému).
c.) time: čas příštího výstupu (při výpočtu problému). Pokud není nastaven je použito implicitní hodnoty (pro nový výpočet) nebo aktuální hodnota (pro restart úlohy).
d.) id: jakýkoliv řetězec přidaný do názvů výstupních souborů.
e.) dmin/dmax: hodnoty max/min aplikované na výstupní soubory (užitečné pro obrázky).

f.) palette: barevná paleta použitá u obrázků. v současnosti jsou platné tato možnosti rainbow, jh_colors, idl1, idl2, step8, step32, heat.

g.) ix1, ix2, ix3: rozsahy os pro směry ix1, ix2 nebo ix3 přes, které jsou data průměrována. Například ix1=: zprůměruje celou osu x1 a seřadí ve 2D. ix1=5: spočítá průměr od hodnoty 5 do konce osy ix1. ix1=:10 spočítá průměr od začátku osy ix1 do 10. ix1=5:10 spočítá průměr od čísla 5 do 10 a ix1=5 vynese z řady na osu ix1 pouze číslo 5 a nastaví hodnotu 5 pro index i.

h.) usr_expr_flag: je třeba nastavit na hodnotu 1, pokud potřebujeme pro výpočet použít uživatelsky definovaný výraz.

10.3.4 Blok <Time>

Parametry v tomto bloku slouží pro definování časových hodnot pro počítanou úlohu (např. čas ukončení výpočtu). Jsou použity programem `main.c`

- tlim: doba za jakou se zastaví integrace, v jednotkách definovaných v řešené úloze.
- nlim: maximální počet cyklů hlavní smyčky před zastavením. Implicitně je tato hodnota nastavena na -1 a zastaví se za čas uvedený v tlim.
- cour_no: číslo udávající dimenzi výpočtu musí být menší než 1.0 pro 1D a 2D pro 3 D musí hodnota být nižší než 0,5

10.3.5 Blok <grid>

Zde uvedené hodnoty nastavují vlastnosti pro výpočetní síť. Využívá je pro svou činnost tato procedura `init_grid_block.c`.

Nx1, Nx2, Nx3: počet buněk v sítí v osách x1, x2 a x3. x1min, x2min, x3min: souřadnice počátečních buněk na osách x1,x2 a x3. (levý okraj první buňka). x1max, x2max, x3max: souřadnice koncových buněk na osách x1, x2 a x3 (pravý okraj poslední buňka).

`ibc_x1,obc_x1:`

`ibc_x2,obc_x2:`

`ibc_x3,obc_x3:`

10.3.6 Blok <parallel>

Parametry v tomto bloku řídí rozklad ve výpočetní domény do bloků pro paralelní zpracování MPI. Doména může být rozložena v libovolném směru na pevně daný počet zpracovávaných datových bloků. To umožňuje provézt různé druhy dekompozice.

· NGrid x1: Počet MPI bloků v ose X1.

· NGrid x2: Počet MPI bloků v ose X2.

· NGrid x3: Počet MPI bloků v ose X3.

Tento blok může být při seriovém zpracování vypuštěn, nebo počet bloků může být nastaven na hodnotu 1 pro každou osu.

10.3.7 Blok <problem>

Parametry v tomto bloku jsou používány generátorem úloh a závisí na typu řešené úlohy. Například úloha **athena 3.1/src/prob/shkset1d.c.** (je použita při výpočtu Brio & Wu rázová vlna), vyžaduje následující hodnoty v bloku.

- gamma: poměr specifické teploty použité ve stavové rovnici.
- *l: hodnoty proměnné * v levé straně
- *r: hodnoty proměnné * v pravé straně

Parametry *l a *r jsou specifické pro řešení úlohy Brio & Wu rázová vlna. Celkově lze říci, že vstupní soubory pro jiné úlohy v tomto bloku mají různé parametry.

11 Formát datového výstupu

Datový výstup je v kódu Athena 3.1 nastaven pomocí parametru bloku <output> ve vstupním souboru.

Pokud chceme mít více výstupních formátů, je nutno nastavit blok <output> pro každý formát zvlášť. To jest, pro každý formát uvést zvláštní blok <output>. Počet výstupních souborů není omezen. Výstupní soubory jsou pojmenovávány podle následující souborové masky basename.Id.#dupmpid.outid.type, kde basename je zděděno z názvu souboru z parametru <job>/problem_id. Parametr id# určuje číslo procesoru použitého při paralelizaci výpočtu pomocí MPI. Proměnná # určuje úroveň neboli rank procesu v MPI (root proces nemá žádné Id#, ani se neukazuje při seriových výpočetních úlohách) Hodnota dumpid je integer naplněný nulami <job>/numdigits, řetězec outid je specifikován v parametru <output>/id a parameter type, který určuje formát výstupu a může být bin, tab, hst, vtk, rst, pdf, pgm, ppm, fits.

Výstupní soubory history dump neobsahují dumpid, nebo outid. Parametry formátů bloků byly již dostatečně vysvětleny v odstavci 10.3. V níže uvedených odstavcích jsou popisy souborových formátů.

History Dumps: (typ=hst) je formátovaná tabulka různých integrovaných hodnot v závislosti na parametru <output>/dt, kde každá hodnota je napsána v jedné řádce tabulky. Soubor obsahuje tlim/dt řádků a reprezentuje časový vývoj hodnot. Soubor je vytvářen funkcí `dump_history.c` editací tohoto souboru je možné upravit množství výstupních hodnot a další parametry. Data jsou přidávána do souboru, pokaždé když je tato funkce volána.

Binární výstup: (typ=bin). Neformátovaný zápis závislých hodnot přes všechny aktivní zóny. Pokud je příkazem `configure` aktivována volba DX bude vytvořena hlavička OpenDX souboru a vytvořen soubor `.dx`, jehož jméno bude odpovídat jménu binárního souboru. Tato hlavička umožní, aby byl binární soubor přečten programem OpenDX (viz. Kapitola 9.3). Vytvoří se nový soubor s časovým intervalem <output>/dt. Tento soubor vytváří funkce `dump_binary.c`

Tabulátorový výstup: (typ=tab) Formátovaná tabulka všech závislých proměných ze všech zón. Vytvoří nový soubor v časovém intervalu <output>/dt Vytváří se funkcí dump_table.c Užitečná funkce hlavně pro 1D tisky.

Ppm výstup: (typ=ppm) Dvourozměné obrázky.

Pgm výstup: (typ=pgm) Černobílé obrázky ve formátu pgm. Škála, orientace a průměrování použité pro vytváření jednotlivých obrázku jsou stejné jako u formátu ppm. Generováno funkcí `output_pgm.c`.

Distribuční funkce pravděpodobnosti: (typ=pdf) Výstup z vybraných proměnných je vytvářen pomocí funkce `output_pdf.c`.

Obrázek FITS (Flexible Image Transport System): (typ=fits) To samé jako obrázky ppm, ale ve formátut FITS je vytvářen pomocí funkce `output_fits.c`.

Obrázky VTK (Virtualization Toolkit): (typ=vtk) podobné binárnímu výstupu, ale ve formátu VTK. Užitečné pro 3D simulace. Je vytvářen pomocí funkce `output_vtk.c`.

Výpis pro restart výpočtu (Restart dumps): (typ=rst). Vytvoří binární výstup všech proměnných (pokud je to nutné i s dvojitou přesností), které se poté dají použít při restartování simulace. Pro restart simulace je nutný vstupní soubor s hodnotami v ASCII. Úlohy řešené pomocí paralelizace MPI vytvoří restart soubor pro každý paralelně běžící proces. Výstupní soubory v programu Athena 3.1 jsou vždy na pozadí přepsány.

11.1 Soubory restartu

Soubory restartu jsou užitečné, pokud výpočet musí pokračovat od posledního bodu přerušení. Soubory, obsahují dostatek informací k pokračování výpočtu, jsou uloženy s potřebnou přesností a restart výpočtu vygeneruje identická data nutná ke kontinuálnímu pokračování výpočtu. Athena 3.1 definuje vlastní formát pro soubory restartu. Soubor `restart.c` obsahuje všechny funkce potřebné ke čtení a zápisu do těchto souborů. Pokud chceme provádět výpočet s možností restart souboru, vložme do vstupního souboru tento zápis:

```
<output2> out_fmt = rst# soubor restartu
dt = 1.0 # časový krok mezi výstupy
```

V tomto příkladě musí být hodnota <job>/maxout vetší nebo rovna číslu 2. Časový přírůstek <output>/dt je udáván v měřítku času použitém ve výpočtu a měl by dávat požadovanou frekvenci zápisu do výstupních souborů (doporučovaná hodnota je vytvářet soubor restartu minimálně každých 6 hodin).

Při paralelizaci výpočtů pomocí knihovny MPI dojde k vytvoření restart souboru pro každý výpočetní proces (vlákno) při opětovném startu úlohy musí být opět použitého stejného počtu procesorů. Pokud úloha vyžaduje uživatelská data musí být tato přidána do restart souborů.
Athena 3.1 poskytuje mechanizmus, který automaticky přidá tato data. V generátoru úloh jsou tyto dvě funkce:

```
void problem_write_restart(Grid *pG, Domain *pD, FILE *fp)
{
return;
}
```

```
void problem_read_restart(Grid *pG, Domain *pD, FILE *fp)
{
return;
}
```

Obvykle jsou tyto funkce prázdné, ale pokud je to nutné dají použít ke čtení a zápisu parametrů, nebo nastavení specifických parametrů úlohy a hraničních podmínek, atp. Generátor úloh **src/prob/rt.c** obsahuje příklady použití. Další informace o použitých strukturách a příkladech jsou popsány v programátorské příručce. Pokud chceme zahrnout do výpočtu restart soubor zadáme na v shellu parametr –r.

```
athena -r myfile.rst
```

Při použití restart souboru není nutné specifikovat parametr -i vstupní.soubor. Protože restart soubor již obsahuje původní data ze vstupního souboru ve formátu ASCII která byla načtena od začátku výpočtu funkcí `par.c`. Tato funkce také vytváří pro restart soubory automaticky zaznamenává vstupní parametry použité při startu úlohy. Restart soubor je možno editovat běžným editorem. Pokud je vstupní soubor specifikována i při použití restart souboru jako v tomto případě,

```
athena -r mujsoubor.rst -i mujvstupni.soubor
```

pak jsou hodnoty v restart souboru přepsány hodnotami ze vstupního souboru.

Podobně, ale mohou být přepsány také parametry ve vstupním souboru samotném například:

```
athena -r myfile.rst time/tlim=20.0
```

Obvykle parametr time/tlim musí být modifikován při restartu. Při paralelním zpracování MPI stačí zadat pouze restart soubor pro root (rank0) proces všechny ostatní dědičné procesy vytvoří svoje vlastní restart soubory na základě jména hlavního zpracovávaného souboru.

11.2 Přidání uživatelsky definovaného výrazu

Často je užitečné, aby výstup byl jiný než standartní datový typ definovaný v parametru <output>/out type. Například pokud je třeba vytvořit obrazové ppm soubory hustoty kinetické energie. Toto je možné jednoduše nastavit pomocí funkce `<output>/usr.`

Jsou nutné následující kroky. Nejdříve je nutné vytvořit příslušnou funkci. Musí být datového typu Real a seznam argumentů musí obsahovat plynovou strukturu a rozměry datového pole. Následující příklad je vybrán ze zdrojového souboru **src/prob/field loop.c** a vypočítá z-komponentu aktuální hustoty z buněk i, j, k.

```
static Real current(const Grid *pG, const int i, const int j,
const int k)
{
return ((pG->B2i[k][j][i]-pG->B2i[k][j][i-1])/pG->dx1 -
(pG->B1i[k][j][i]-pG->B1i[k][j-1][i])/pG->dx2);
}
```

Pak se použije funkce `get usr expr ()`, která je obsažena v každém generátoru úloh, aby program vrátil vypočtenou hodnotu v případě, že řetězec <output>/out má příslušnou hodnotu. Jako příklad, aktuální hustota je vypočítána s pomocí výše obsažené funkce pokud řetězec <output>/out je J3 použitý v následujícím kódu.

```
Gasfun_t get_usr_expr(const char *expr)
{
if(strcmp(expr,"J3")==0) return current;
return NULL;
}
```

Pokud chceme z vypočítané hodnoty hustoty vytvořit film použijeme do řetězce výstupního bloku <output>/out hodnotu ‘J3’ a hodnotu <output>/usr=1, spolu s příslušnými parametry (např. nastavení snímkování, časové intervaly, škálu, atd.).

11.3 Přidání uživatelsky definovaného formátu

Je také velmi jednoduché přidávat uživatelsky definované zcela nové výstupní formáty. Je nutno učinit ve dvou krocích. Prvním krokem je napsání nové výstupní funkce do souboru, který obsahuje generátor úloh. Funkce v generátoru úloh musí mít následující syntaxi:

```
void special_output (Grid *pGrid, Domain *pDomain, Output *pOut);
```

Více informací o programování doménové struktury případně, struktury výstupních souborů lze

nalézt v programátorské příručce.

Druhým krokem je zakomponování volání datového formátu kdekoliv ve výpočetní proceduře. Formát volání formátu datového výstupu má následující syntaxi:

```
void data_output_enroll (Real time, Real dt, int num, const
VGFunout_t fun,
const char *fmt, const Gasfun_t expr, int n,
const Real dmin, const Real dmax, int sdmin, int sdmax)
```

Argumenty použité v této funkci mají následující funkce:

a.) **time:** čas výstupu, obvykle aktuální doba simulace

b.) **dt:** interval mezi výstupy

c.) **num:** počáteční číslo výstupu

d.) **fun:** jméno výstupní funkce (ukazatel výstupní funkce) Ve výše uvedeném příkladu je to speciální výstup, ale také to mohou být obrázky ppm o určitém množství

e.) **fmt:** řetězec formátu použitelný například pro chybový výpis

f.) **expr:** název funkce (ukazatel funkce) určující množství vytvořených obrázků při používání rutin např.output, ppm, output, pgm, output, fits

g.) **n:** v tomto případě obrázky obsahují v názvu řetězec "out#", v tomto řetězci je nahrazen znak # hodnotou uvedenou v argumentu n

h.) **dmin, dmax:** při vytváření datových výstupů, typ obrázek se pro data mohou použít implicitně uvedené hodnoty min/max pro osy nebo zde uvedené údaje dmin a dmax

i.) **sdmin sdmax:** Zarážky, které indikují použití automaticky nastavené škály (sdmin / sdmax =0), nebo pevné hodnoty (sdmin / sdmax = 0).

V nejjednodušším případě při volání datového výstupu jsou nastaveny všechny argumenty na 0, nebo NULL.tak jako v následujícím příkladě:

```
data_output_enroll(pGrid-
>time,0.1,0,special_output,NULL,NULL,0,0.0,0.0,0,0);
```

11.4 Specifikace hraničních podmínek

Aktuální implementace hraničních podmínek využívá funkce ukazatelů, nebo také zarážek. Zarážky jsou používány k nastavení implicitních funkcí z úplného seznamu uvedeného v **/src/set bvals.c**. Každá z těchto funkcí nastavuje množství „ghost“ zón podle algoritmu, který nastaví hodnota ukazatele.

Použití funkce ukazatele velmi zjednodušuje nastavení nových hraničních podmínek. Například k přidání nových hraničních podmínek na vnitřní osu X1 uživatel napíše novou funkci, která nastaví hodnoty „ghost“ zón a vloží je do generátoru úloh a zavede tuto novou funkci tak, že na konec souboru uvede tuto řádku:

```
set_bvals_fun(right_x1,special_bc_function_name);
```

V tomto případě (special_bc_function_name) je název funkce vytvořené v kroku 1. První argument set_bvals_fun nastavuje hraniční podmínky, na které je funkce použita: specifikuje levou či pravou x1 pro vnitřní nebo vnější hranici osy x1 a podobně specifikuje levou či pravou x2 pro vnitřní nebo vnější hranici osy x2 a samozřejmě specifikuje levou či pravou x3 pro vnitřní nebo vnější hranici osy x3. Příklady jsou uvedeny v těchto souborech generátoru úloh `dmr.c`, `noh.c`, a `shkset3d.c`. Každý z těchto souborů obsahuje speciální hraniční podmínky uvedené v tomto příkladě. Musíme dávat pozor na následující:

Zarážky pro hraniční podmínky je nutno definovat ve vstupním souboru pouze pro směry ve kterých je sít integrována. To jest, pokud je Nx1>1 a Nx2=Nx3=1 právě tehdy jsou vnitřní hraniční podmínky ibc x1 a vnější hraniční podmínky obc x1 požadovány ve vstupním souboru. Pokud nastavíme vnitřní x1 hranice, zarážka ibc x1 v souboru parametrů není požadována. Opět může být parametr v souboru uveden, ale nebude kontrolován.

11.5 Řešitelé (generátory) úloh zahrnutí v Athena 3.1

Program obsahuje velké množství řešených úloh, které jsou v manuálu nazvány „problem“. Tyto soubory jsou uloženy v adresáři **/src/prob**. V tabulce 6 je uveden jejich úplný seznam.

Tabulka 6: Seznam řešitelů (Problem Generator) úloh v programu Athena 3.1.

blast.c	dmr.c
linear_wave3d.c	shkset2d.c
field_loop.c	lw_implode.c
rotor.c	shkset3d.c
carbuncle.c	cpaw1d.c
kh.c	noh.c
rt.c	shu-osher.c
cpaw3d.c	cpaw2d.c
linear_wave1d.c	orszag-tang.c
shk_cloud.c	twoibw.c
linear_wave2d.c	pgflow.c,
shkset1d.c	

12. Gravitace v Atheně 3.1

12.1 Zdrojové podmínky a závislost na statickém potenciálu

Zpracování zdrojových podmínek v programu Athena 3.1 je podstatně rozdílné od přechozích verzí. Teď jsou dovoleny pouze funkce statického gravitačního potenciálu a jejich zpracování přísně konzervuje celkovou energii v toku.
Pokud chceme zahrnout statický gravitační potenciál do výpočtu, musíme zahrnout speciální uživatelsky definovanou funkci, která definuje gravitační potenciál a jeho souřadnice (x1 ,x2 ,x3) do výpočtu. To učiníme tak, že ji přidáme jako argument, který specifikujeme v generátoru úloh například:

```
static Real grav_pot (const Real x1, const Real x2, const Real x3)
```

Argumenty v tomto případě jsou souřadnice x-, y- a z-, jejich potenciál musí být ověřen. Tato funkce je zavedena do integrátoru pomocí funkce ukazatele zvané StaticGravPot. Tento ukazatel je definován v souboru **src/globals.h**. Pokud chceme uživatelsky definovanou funkci zavést, přidáme kdekoliv do souboru generátoru úloh tuto řádku.

```
StaticGravPot = grav_pot;
```

Pokud funkce StaticGravPot nebude v souboru uvedena, tak nebude ani v integraci zahrnuto gravitační zrychlení. Soubory úloh `pgflow.c` `rt.c` obsahují příklady, jak zahrnout zdrojové podmínky v našich vlastních výpočtech. Speciálně funkce `pgflow.c` nastaví podmínky pro test gravitačního potenciálu.

12.2 Vlastní gravitace při použití FFT (Fast Fourier transformation)

Athena 3.1 je schopna zahrnout do výpočtu vnitřní gravitaci kapaliny. Používá unikátní blokový rozklad algoritmu FFT založený na knihovnách FFTW. Pokud je třeba použít tuto funkci, tak musí být mít nainstalovány knihovny FFTW3.x a musí být modifikován konfigurační soubor celého hlavního programu **athena 3.1/makeoptions.in**. Dále je nutné spustit konfiguraci s níže uvedeným parametrem:

```
--with-gravity=fft --enable-fft options
```

Numerický algoritmus implementovaný v Athena 3.1 přesně konzervuje celkový gravitační moment kapaliny. Gravitační konstanta je načtena jako blokový parametr ve vstupním souboru. Test lineární vlna (linear wave), který je součástí kódu, je možno použít pro test vlastní gravitace. Algoritmus funguje, ale pouze pro periodické hraniční podmínky.

13 Běh Atheny 3.1 na více procesorech za použití MPI

Athena 3.1 umožňuje paralelizaci výpočtu pomocí technologie doménového rozkladu založeného na knihovně MPI. Kód je možno spustit na jakémkoliv clusteru s distribuovanou pamětí, kde je tato knihovna nainstalována. Pokud je požadavek používat tuto funkci, je třeba při konfiguraci základního programu MPI zapnout. To se provede následujícím příkazem:

```
configure --enable-mpi
```

Tento příkaz překompiluje makra, která do funkcí připojí příslušný MPI kód.

Při kompilování musí být příslušné MPI knihovny připojeny do programu. Nejjednodušším způsobem, jak toho dosáhnout, je zapsat do souboru **src/makefile.in**, kde je specifikován kompilátor,jeho parametry, linker a také specifické knihovny clusteru.
Například parametr:

```
make all MACHINE=hydra
```

Zahrnuje příslušné kompilátory MPI knihovny pro cluster Beowulf na Princentonské univerzitě, který se jmenuje „hydra“. Pokud se nakopírují soubory z „hydry“ na nový cílový počítač s názvem „mymachine“ po příslušné úpravě a kompilaci tímto příkazem:

```
make all MACHINE=mymachine
```

a program vytvoří příslušné spustitelné soubory.

Vstupní soubor pro příslušnou řešenou úlohu musí ale být upraven. Musí být do něj přidán blok <parallel>, který specifikuje požadovaný druh doménového rozkladu výpočtu.
Například následující uvedený soubor:

```
<parallel>

NGrid_x1 = 1
NGrid_x2 = 10
NGrid_x3 = 1
```

vyústí v maticový rozklad s 10 poli v ose-y (je potřeba 10 procesorů).

Zatímco tento zápis:

```
<parallel>

NGrid_x1 = 1
NGrid_x2 = 2
NGrid_x3 = 3
```

vyústí v rozklad se dvěmi poli v ose y a třemi poli v ose z.
(celkově je potřeba 6 procesorů).

A tento zápis:

```
<parallel>

NGrid_x1 = 4
NGrid_x2 = 4
NGrid_x3 = 4
```

vyústí v blokový rozklad se čtyřmi bloky v každém směru (To znamená, že celkový počet procesorů nutných k výpočtu je 64).

V zásadě je možné provést rozklad do jakéhokoliv počtu domén, ale počet aktivních zón v jakémkoliv MPI bloku, a v libovolném směru by neměl být nižší než 4.

Každý paralelní MPI výpočet musí být spuštěn příkazem `mpiexec,` nebo `mpirun.` Počet použitých procesorů je specifikován například na příkazové řádce pomocí parametru `np#` (kde # udává počet použitých procesorů). Počet použitých procesorů uvedených na příkazové řádce musí souhlasit s počtem MPI bloků uvedených v bloku <parallel> ve vstupním souboru. Jinak Athena 3.1 vytiskne chybovou hlášku a skončí.

V adresáři **athena 3.1/doc** jsou uvedeny užitečné skripty pro PBS (Parallel Batch System), který se často používá k plánování úloh na paralelních clusterech. Data zpracovávaná pomocí MPI budou zapsána do oddělených souborů pro každý proces. Užitečné programy pro spojení několika vtk souborů vygenerovaných paralelním zpracováním jsou umístěny v adresáři **athena 3.1/vis/vtk**.

14 Vizualizace výstupu

Athena 3.1 nemá implicitní grafický program pro zpracovávání a vizualizaci dat. Místo toho se můžeme rozhodnout, který vizualizační nástroj nejlépe vyhovuje uživatelským potřebám, výstupní data jsou ve formátu, který může příslušný vizualizační program zpracovat. Athena 3.1 obsahuje kód pro několik různých grafických formátů včetně zdrojového kódu. Budoucí verze budou obsahovat lepší vizualizační nástroje.

Následující část obsahuje popis užitečných vizualizačních balíků pro datové soubory Atheny 3.1 (předpokládejme, že kód programu již byl spuštěn a datové soubory jsou již k dispozici).

14.1 Supermongo

Je populární softwarový balík pro vytváření 1D obrazových souborů typu SM 11. Jednoduchá makrofunkce SM, která dokáže číst tabulkový vstup a zpracovat ho do obrazového formátu je umístěna v adresáři **/athena3.1/vis/sim**.

14.2 Procedury IDL (Interactive Data Language)

IDL (Interactive Data Language) je velmi výkonný a profesionální externí program, který může číst jak binární tak VTK [18] soubory a vytvářet z datových souborů obrázky. Procedury pro tento program jsou uloženy v adresáři **athena 3.1/vis/idl**. Pokud je nutno tyto procedury spouštět musíme být balík IDL v systému nainstalován. Z adresáře **athena 3.1/bin** je nutno následující příkaz a z binárního souboru se mohou vytvořit 1D obrazové soubory. Vizualizaci datových souborů spustíme tímto příkazem:

```
idl
IDL> .run ../vis/idl/pltath.pro
IDL> nine_plot,'Brio-Wu.0040.bin',1
```

Soubor `pltath.pro` obsahuje užitečné procedury pro zpracování souborů.

14.3 Formát OpenDX

Balík OpenDX umí číst binární soubory z programu Athena 3.1 tak, že přidá do hlavičky souborů příponu.dx. Před použitím formátu musíme samozřejmě program zkonfigurovat s parametrem `-dx` na příkazové řádce. A samozřejmě na systému musí být nainstalován příslušný OpenDX balíček. Program Athena ve verzi 2.0 obsahuje několik příkladů binárních souborů, které ještě nebyly zakomponovány do verze Athena 3.1.

14.4 VTK (Visual Tool Kit)

Dle zkušeností je pro 3D vizualizace velice užitečný balík VisIt. Formát VTK, který program Athena 3.1 vytvoří, je již možno přímo číst programem VisIT. Pro zpracování datových souborů vytvořených paralelizací v síti MPI je v adresáři **athena 3.1/vis/vtk** připravený „C“ kód, který spojuje vytvořené soubory do jednoho. Tento kód je velmi užitečný pro paralelizované zpracovávání většího množství dat na výpočetně výkonných mainframových serverech či clusterech.

14.5 2D Animace

Athena 3.1 umí vytvářet 2D obrázky, které mohou být zobrazeny přímo nebo snadno animovány. Například pokud chceme, aby byla vytvořena série obrázků s proměnou ve formátu ppm, spojíme je a zobrazíme je postupně přímo v programu ImageMagick tímto příkazem:

```
animate *.ppm
```

Je možno také použít program xanim, který vyžaduje převod ppm obrázků do formátu FLI stačí pak zadat tyto příkazy:

```
ls Wind*ppm > list1
ppm2fli -g80x80 list1 Wind.fli
xanim Wind.fli
```

15 Příklady spouštění Athena 3.1

15.1 Post instalační testy programu Athena 3.1

Výkonostní test instalace Athena 3.1 se spouští automaticky pomocí souboru `make`. Tento test se skládá z lineárního konvergerenčního testu v gridu (síti) o 512 zónách. Program vypočítá úlohu a vrátí chybu L1 při porovnávání rychlé magnetosonické vlny s analytickým řešením. Pokud se nepodaří výkonostní test, nebo úloha vrátí příliš velikou chybu, pak je něco špatně v instalaci programu případně v kompilaci kódu.
Pokud je nutno spustit výkonostní testy, musíme spustit následující sekvenci příkazů. Testy se musí spustit v kořenovém adresáři programu Athena 3.1. Příslušné skripty, jsou vygenerovány příkazem `autoconf` při instalaci programu. Pro spuštění výkonnostního testu spustíme tento příkaz:

```
Configure

make all

make test

(cd tst/1D-mhd; ./run.test)

zone-cycles/cpu-second = 3.067215e+05
zone-cycles/wall-second = 3.055470e+05
L1 norm for density: 6.333390e-11
```

Hodnota zone-cycles/cpu-second je užitečný ukazatel činnosti kódu, ukazuje počet buněk sítě aktualizovaných za sekundu. Tento údaj záleží na fyzikálních podmínkách, geometrii problému a samozřejmě na procesoru použitém pro výpočet. Hodnoty výše uvedené jsou pro procesor Intel Xeon 3.08GHz. Chyba je absolutní a neměla by nikdy být větší než 10^{-10}.

15.2 Spuštění 1D testovací úlohy „Athena 3.1: „The Brio & Wu shocktube “

Jako příklad jak konfigurovat, kompilovat a spouštět Athenu 3.1 a vizualizovat výstup pro 1D úlohu si ukážeme kroky pro spouštění „Brio & Wu shocktube“ (rázová vlna). Tento postup předpokládá, že program Athena 3.1 je již korektně nainstalován.

Prvním krokem ke konfiguraci je tento příkaz:

```
cd Athena 3.1
configure --with-problem=shkset1d
```

Po jeho provedení program vytiskne na obrazovku různé diagnostické kroky a údaje z prováděné činnosti. Dalším nutným krokem je kompilace. Tu provedeme následujícím příkazem:

```
make all
```

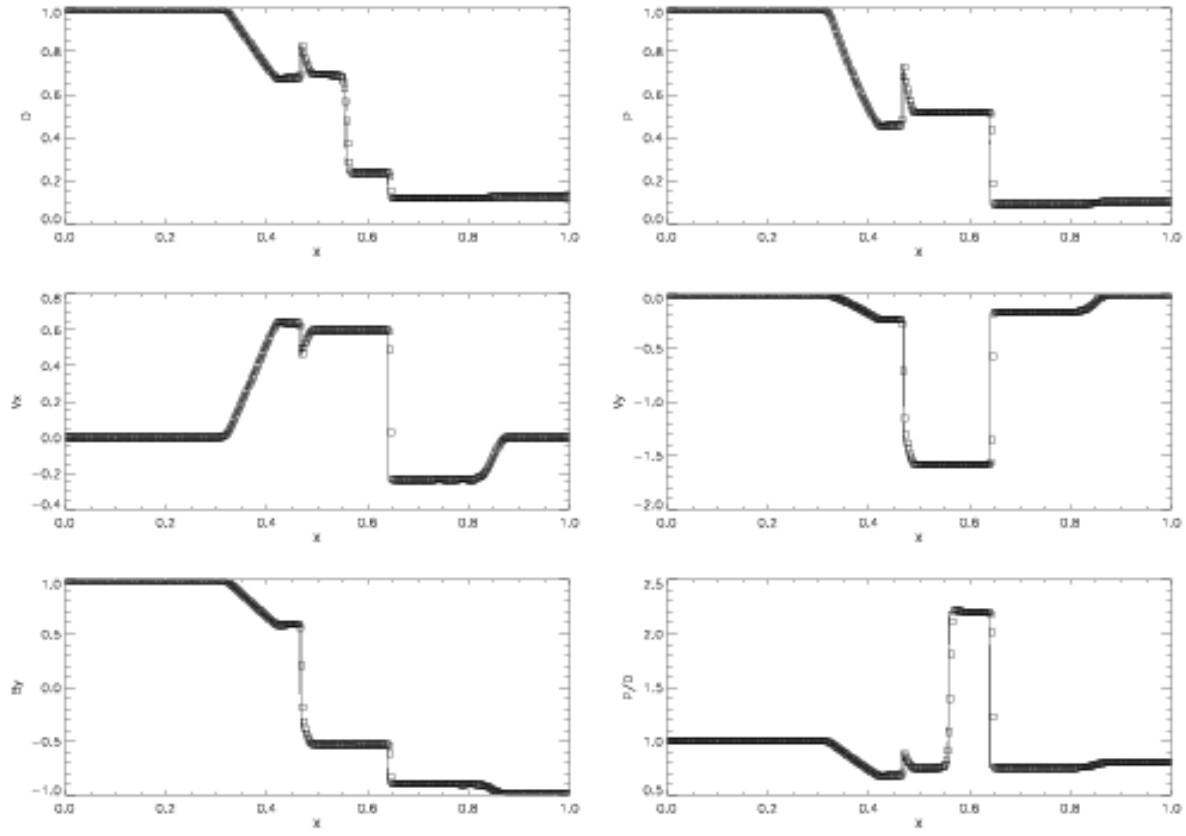
Implicitní kompilátor vytiskne na obrazovku diagnostické údaje. Poté můžeme kód spustit z domovského adresáře programu **athena 3.1/bin** a to tímto příkazem:

```
cd bin
athena -i ../tst/1D-mhd/athinput.brio-wu
```

Program vytiskne na obrazovku informace o každé operaci, kterou provádí. Měl by vytvořit asi 40 binárních souborů pojmenovaných `Brio-Wu*.bin` dále vytvoří 40 tabulkových souborů pojmenovaných `Brio-Wu*.tab` a také soubor `Brio-Wu.hst`. Je zde několik způsobů jak mohou být tato data zobrazena. Jedním ze způsobů je použít IDL skripty uložené ve složce **athena 3.1/vis/idl** a zadat tyto příkazy:

```
idl
IDL> .r ../vis/idl/pltath.pro
IDL> four_plot,'Brio-Wu.0040.bin'
```

Athena 3.1 používá v této úloze řešitel Roe v gridu (síti) se 400 výpočetními buňkami a 10 000 buňkami s jednorozměrným polem. Obrázek 8 ukazuje zleva doprava rychlou refrakci pomalé sloučené vlny a kontaktní diskontinuity. Výsledky mohou být porovnány s referenčním obrázkem.



Obrázek 8: Výsledky testu „Brio-Wu shocktube“ zpracované pomocí procedury IDL four_plot.

15.3 Spuštění 2D testovací úlohy „Orzsag-Tang vortex“ v programu Athena 3.1

Jako příklad jak konfigurovat, kompilovat a spouštět Athenu 3.1 a vizualizovat výstup pro 2D úlohu si ukážeme jak spustit 2D test „Orzsag-Tang vortex“ s algoritmem 3 řádu a HLLD toky. Dále vytvoříme z výstupních dat obrázky. Základním předpokladem samozřejmě je, že musíme mít program Athena 3.1 korektně nainstalován. Prvním krokem ke konfiguraci je tento příkaz:

```
cd Athena 3.1
configure –with –order=3 –with-problem=orzsag-tang –with flux=hlld
```

Konfigurační skript během provádění operace vytiskne různé diagnostické kroky a údaje.
Dalším nutným krokem je kompilace:

```
make all
```

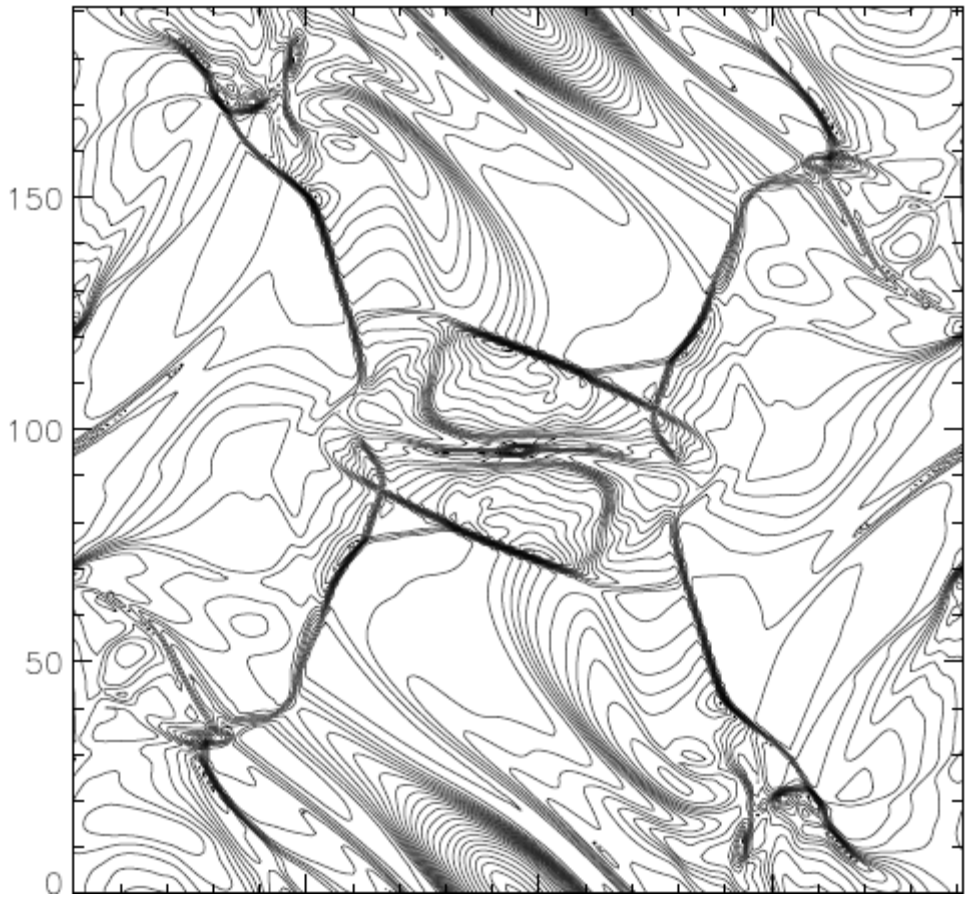
Implicitní kompilátor vytiskne na obrazovku diagnostické údaje. Poté můžeme kód spustit z domovského adresáře programu **/athena 3.1/bin** následujícím příkazem:

```
cd bin
athena -i ../tst/1D-mhd/athinput.orzag-tang
```

Kód vytiskne na obrazovku informace o každé operaci, kterou provádí. Na procesoru Intel Xeon 3.08GHz trvá výpočet asi 4 minuty. Program by měl vytvořit asi 100 binárních souborů pojmenovaných `Orzag-tang.*.bin`, 250 obrázků tlaku plynu pojmenovaných `OrzsagTang.*.P.ppm` a nakonec 250 obrázků hustoty plynu pojmenovných `OrzsagTang.*.D.ppm` a také soubor OrzsagTang.hst. Je zde několik způsobů, jak mohou být tato data zobrazena. Jedním ze způsobů je použít IDL skripty uložené ve složce **athena 3.1/vis/idl**.

Pro zvýraznění zobrazení tlaku v obrázku v čase t=0,5 je nutno zadat následující příkaz:

```
IDL> .r ../vis/idl/pltath.pro
IDL>readbin,'OrzagTang.0050.bin'
IDL>contour,p,nlevels=30,isotropic,xstyle=1,ystyle=1''
```



Obrázek 9: Zvýrazněné kontury tlaku plynu v čase t=0,5 test „Orzsag-tang vortex“ zpracované pomocí procedury IDL.

V této úloze je také zajímavé sledovat animaci vývoje hustoty, nebo tlaku. Je to možné několika způsoby. Nejjednodušší je použít přímo programem vytvořené soubory. Například pokud je nainstalován program ImageMagick, je musíme zadat tento příkaz:

```
animate *.P.ppm
```

V případě zobrazení hustoty se postupuje podobně, jen se bude příkaz malinko lišit napíšeme toto:

```
animate *.D.ppm
```

15.4 Spuštění 3D testovací úlohy „Advection field loop test“v programu Athena 3.1

Jako příklad, jak konfigurovat, kompilovat a spouštět Athenu 3.1 a vizualizovat výstup pro 3D úlohu si ukážeme kroky pro spouštění úlohy „Advection field loop test“. Opět se předpokládá, že je program Athena 3.1 již korektně nainstalován. Prvním krokem ke konfiguraci je zadání příkazu:

```
cd Athena 3.1
configure –with-order=3 –with-problem=field_loop –with-flux=hlld
```

Konfigurační skript vytiskne různé diagnostické kroky a údaje. Dalším nutným krokem kompilace je zadání příkazu:

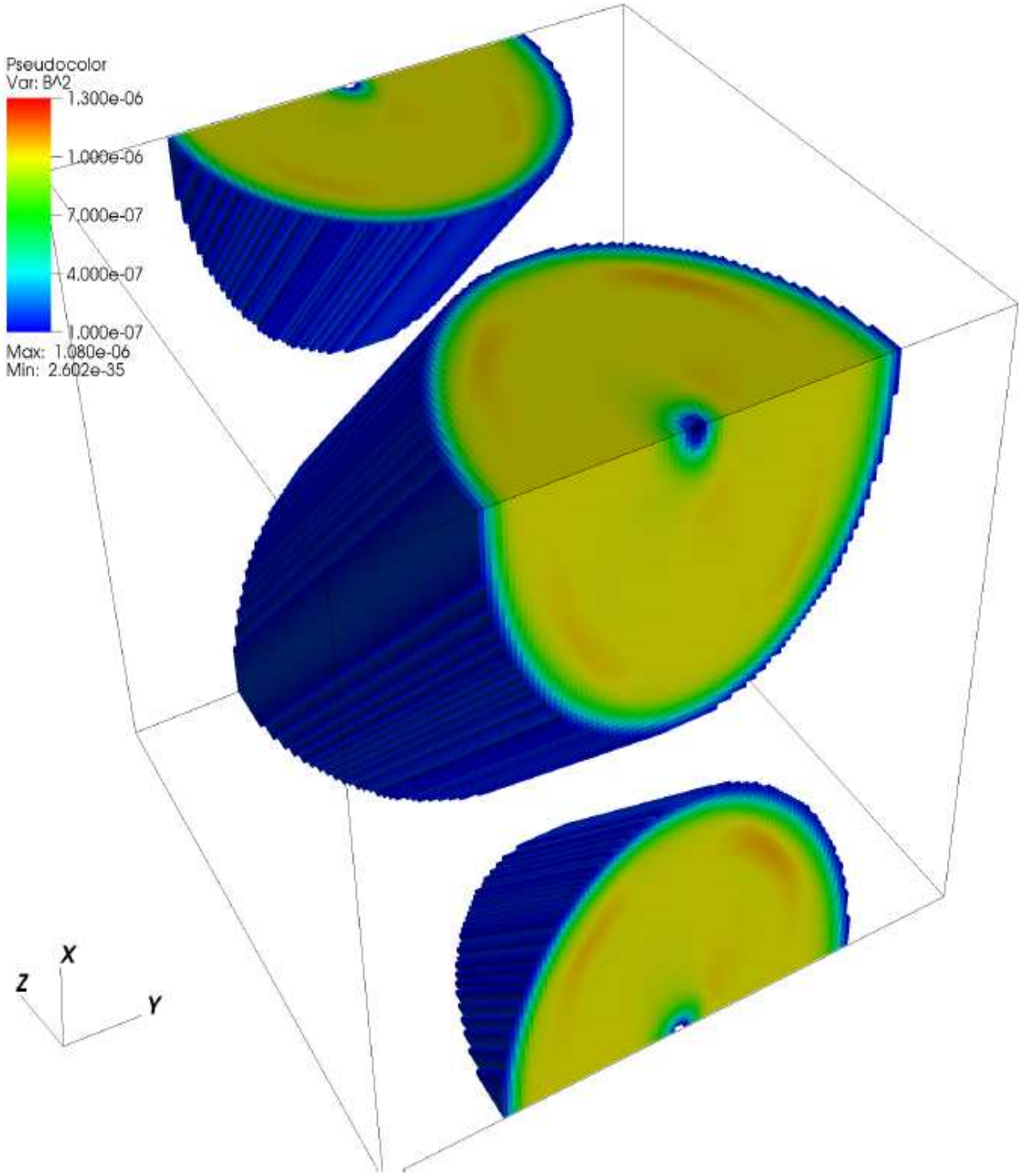
```
make all
```

Implicitní kompilátor vytiskne na obrazovku diagnostické údaje. Poté můžeme kód spustit z domovského adresáře **athena 3.1/bin** následujícím příkazem:

```
cd bin
athena –i tst/3d-mhd/athinput.field_loop4grid/Nx1=64 grid/Nx2=64
grid/Nx3=64 parallel/ grid/Nx1=1 parallel/ grid/Nx2=1 parallel/
grid/Nx3=1
```

Před spuštěním je nutno se přesvědčit, že vstupní soubor pro tento test je vhodně upraven (tzn je uveden iprob=4 v sekci <problem>) Zatímco implicitní vstupní soubor pro tuto úlohu používá paralelní zpracovávání s výpočetní sítí o velikosti 128^3 s blokovým rozkladem na 8 procesorů, je nutno vstupní parametry na příkazové řádce upravit na hodnoty výpočetní síť (grid) 64 a jen jeden procesor.

Procesoru Intel Xeon 3.08GHz trvá zpracování asi 30 minut. Program by měl vytvořit 3 VTK soubory, 250 obrázků formátu ppm zobrazujících aktuální hustotu na řezech v osách x a y a dále také soubor historie hst. Na obrázku 3 je zobrazen normovaný povrch magnetické energetické hustoty. Obrázek je vytvořen balíčkem VisIt. Alternativně je možno použít proceduru `readvtk` umístěnou v adresáři **vis/idl/pltath.pro** k načtení dat a jednu ze zobrazovacích rutin z balíku IDL k zobrazení obrázku.



Obrázek 10: Povrchový obraz magnetické energie, vizualizace vytvořena Programem VisIt, „Advection field loop test“ ve 3D prostoru.

15.5 Testy v Athena 3.1

V programu Athena 3.1 každý řešitel úloh v podadresáři **/src/prob** vytvoří další testovací soubor. Další popis těchto testů, příklady spuštění a výsledky můžete najít na webových stránkách a v dokumentaci.

16 Testovací souprava v Athena 3.1

Opravdová síla programu Athena 3.1, ale tkví v řešení nových úloh (tj. takových pro, které nejsou soubory řešitelů ještě vytvořeny ve zdrojové distribuci). Pro vytváření nových úloh je nutné dodržet následující postup.

Je nutno napsat novou funkci, která inicializuje úlohu. Tato funkce musí být typ void, musí mít název `problem` a musí mít jako parametry uvedené ukazatele na struktury GAS a Domain.
Tato funkce musí být umístěna v souboru v adresáři **/src/prob**.

Je nutno napsat novou funkci nazvanou `Userwork_in_loop` a `Userwork_after_loop` (nemusí mít žádné argumenty, pokud nejsou třeba) a uložit ji do souboru obsahujícím úlohu. Jak už název sám napovídá , tyto funkce mohou být použity k provedení speciálních na specifickém problému závislých operací, které proběhnou po hlavním výpočtu (například `linear_vawe.c`).
Pokud je třeba nastavit speciální hraniční podmínky je nutno napsat novou funkci, která je implementuje a zavést ji do výpočtu použitím funkce set_bvals_fun.
Pokud je potřeba speciální datový výstup je nutno napsat novou funkci, která jej implementuje a zavést ji pomocí funkce data_output_enroll.

Poté co jsou provedeny všechny výše uvedené úkoly, nakonfigurujeme a zkompilujeme kód za použití příslušných možností a spustíme v shellu s tímto parametrem:

```
–with-problem=název_nové_úlohy
```

Je pravděpodobné, že budeme muset použít programátorskou příručku, abychom byli schopni porozumět datovým strukturám, a názvům použitým v programu Athena 3.1. Pro začátek je vhodné použít řešitele uvedené v adresáři **/src/prob** jako šablony.

17 Závěr

Program Athena 3.1 je navzdory své, z dnešního pohledu nepatrné velikosti, opravdu mocným nástrojem pro řešení astrofyzikálních úloh. Vzhledem k způsobu ovládání je spíše uzpůsoben pro zkušenější uživatele výpočetní techniky, kteří jsou schopni pracovat v shellu [19] Linuxu. Nicméně velmi široké možnosti tohoto způsobu ovládání umožňují řešit i velmi komplexní úlohy. Po důkladném studiu manuálů a laborování s různými konfiguracemi je, ale jeho ovládání již triviální a skutečně záleží pouze na ochotě uživatele se jej naučit. Doba výpočtu velmi závisí na množství dat a použitém výpočetním hardwaru. Nicméně i při poměrně značném množství dat, slabém procesoru a nepříliš výkonných pamětech, výpočet vždy doběhne do konce. Odměnou za nepříliš komfortní ovládání je tedy vysoká stabilita a spolehlivost. Kód využívá beze zbytku plný výkon procesoru a pamětí. Software je napsán s ohledem na přesnost výstupu, jeho další zpracování a hlavně stabilitu celého procesu. Prakticky se nestane, že by proces tzv. zamrzl či nedokončil úlohu.

18 Seznam obrázků

19 Seznam Tabulek

20 Seznam literatury

[1] Priest, E. R.: Solar Magnetohydrodynamics, D. Reidel Publishing Company, London England 1982.

[2] Chung.T. J.: Computational Fluid Dynamics.Cambridge University Press, New York, 2002.

[3] Nakariakov V. M., Vervichte, E.: Coronal Waves and Oscilations Living Rev. Solar Phys., 2, 2005.

[4] http://www.nsf.gov/funding/pgm_summ.jsp?pims_id=5524&from=fund

[5] http://www.cs.utexas.edu/~dagh/Tutorial/jcb_mitra/tut_jcb_mitra.html

[6] http://en.wikipedia.org/wiki/Fast_Fourier_transform

[7] http://davis.lbl.gov/NASA/amrgodunov.pdf

[8] http://en.wikipedia.org/wiki/Adaptive_mesh_refinement

[9] http://www.astro.princeton.edu/~jstone/athena.html

[10] http://www.mcs.anl.gov/research/projects/mpi/mpich1/

[11] http://www.supermongo.net/

[12] http://www.imagemagick.org/script/index.php

[13] http://www.opendx.org/

[14] www.llnl.gov/VisIt/

[15] http://en.wikipedia.org/wiki/Riemann_solver

[16] http://www.crs4.it/HTML/int_book/NumericalMethods/subsection3_7_3.html#SECTION000730000000000000

[17] http://math.lanl.gov/~shenli/publications/hllc_mhd.pdf

[18] www.vtk.org

[19] http://en.wikipedia.org/wiki/Unix_shell.