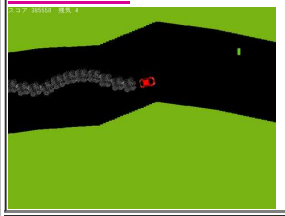
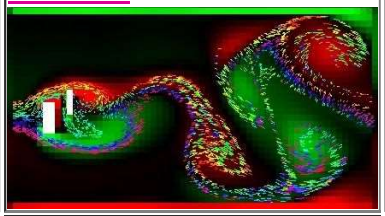
仮眠プログラマーのつぶやき

自分がプログラムやっていて、思いついたことをつぶやいていきます。

自作ゲームやツール、ソースなどを公開しております。

①ポンコツ自動車シュライシュラー DOWNLOAD 	②流体力学ソース付き DOWNLOAD 	③ミニスタヲズ DOWNLOAD 
④地下鉄でGO DOWNLOAD 	⑤ババドン DOWNLOAD 	⑥HLSLサンプル DOWNLOAD 
⑦圧縮拳（ツール） DOWNLOAD	⑧複写拳 DOWNLOAD	⑨布シミュレーション DOWNLOAD

livedoor プロフィール

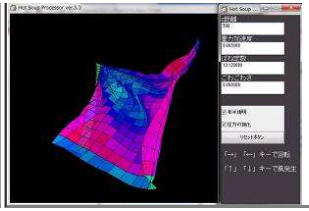
toropipp

記事検索

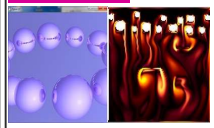
検索

月別アーカイブ

2017年04月
2016年01月
2015年03月
2014年11月
2014年10月
2014年03月
2014年02月
2013年08月
2013年07月
2013年06月
2013年05月



⑩hspでgpgpu
DOWNLOAD



⑪Handbrakeドラッグドロップツール
DOWNLOAD

⑫minecraft巨大電卓地形データ
DOWNLOAD

⑬流体力学CIP法
DOWNLOAD

⑭パズドラルート解析GPGPU版
DOWNLOAD

⑮ゲーム「流体de月面着陸」
DOWNLOAD

＜HSPで高速並列演算 | HLSLでリアルタイムレイトレーシング＞

2013年03月
2013年02月
2013年01月
2012年12月
2012年11月
2012年02月
2011年11月
2011年09月
2011年07月
2010年09月
2010年06月
2010年05月
2010年04月
2010年03月
2010年02月
2010年01月
2009年12月
2009年09月
2009年08月
2009年07月
2009年06月
2009年05月

カテゴリ別アーカイブ

HLSL (2)
流体力学 (4)
GPGPU (17)
ブログ (1)
アルゴリズム (11)
パーリンノイズ (4)
レイトレーシング (7)
HSP (14)
プラグイン作成 (5)
パズドラ (5)
雑談 (12)
技術話 (7)
技術解説「ババドン」(5)
HSPコンテスト (4)
セピア (3)

2011年09月24日 00:01　　流体力学 アルゴリズム

流体力学のプログラムを作りたい！その1

（決断は早いほうが‥ということでブログ移転しました！決してアメブロとライブドアブログで仮眠プログラマーが2人いるわけではないですよ。よろしくお願いします）

そして、本日はメインディッシュの　**流　体　力　学**　！！
過去の当たり判定やセピア変換などの記事は前菜かスープみたいなもんですよ

HSPコンテストでは流体シミュレーションがたくさんコメントをいただいててありがたやー
ニコニコの紹介動画の方はすでに黒歴史‥

今回は一番単純で簡単な方法で動く流体力学プログラムをつくろうと言うのが目標だ。（ところで当方は独学でやったため、もしかしたら専門用語の使い方が間違ってたり、計算式の認識の仕方が変だったりするかもしれないですが、ご了承下さい。）

【導入】

まず2Dでも3Dでも流体力学で避けて通れないのは**ナビエストークスの式**（NS式）([1]http://ja.wikipedia.org/wiki/%E3%83%8A%E3%83%93%E3%82%A8–%E3%82%B9%E3%83%88%E3%83%BC%E3%82%AF%E3%82%B9%E6%96%B9%E7%A8%8B%E

1億桁×1億桁 (5)
付属プラグイン速度測定 (1)
技術解説「ぽんこつ自動車」(4)
任意の軸回転 (1)
技術解説「地下鉄でGO！」(8)
ドローン (1)
電子工作 (1)

【お気に入りブログ】
- アド吉ブログ
- シープのオフィシャルブログ
- 「ファンの一人だよ。」
- 鋼鉄親子でゲーム漬け！
- ゲームって何？
- 空のつくり方

<< 2011年9月 >>

日	月	火	水	木	金	土
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

livedoor NEWS

- 「死の組」になったイラン代表 ケイロス監督が強烈コメント
- 宇宙戦隊キュウレンジャーで共演する岐洲匠と大久保桜子 熱愛か
- ジャニーズJr.阿部顕嵐と私立恵比寿中学・星名美怜が熱愛か「文春」報道
- 世間から猛反発を受けている白鵬 CM大幅減の危機？
- 中居正広「めちゃイケ」で最後の日本一周へ 2018年1月2日放送

5%BC%8F より）

$$\frac{\partial \boldsymbol{v}}{\partial t} + (\boldsymbol{v} \cdot \nabla)\boldsymbol{v} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \boldsymbol{v} + \boldsymbol{F}$$

（画像はwikipediaより）

左辺 − 第1項：時間微分項、第2項：移流項（対流項）

右辺 − 第1項：圧力項、第2項：粘性項（拡散項）、第3項：外力項

ちなみにこれでも簡単なほうらしい。これだけだと未知数が1つおおいままなので、∇・v = 0 という方程式を連立して解けということだ（vは速度ベクトル場）。

$$\nabla \cdot \boldsymbol{v} = \operatorname{div} \boldsymbol{v} = 0$$

（画像はwikipediaより）

三角形の逆みたいな記号があるけどなんだこれは・・内積の掛け算で使う「・」があるけど、ベクトルの計算式とかあるのか？と、私は最初見た時いろいろわからないとこだらけで、結局全部を本当に理解するのに2ヶ月くらいかかった・・

結局、∇は「なぶら」といって、∇の次に来る記号のベクトル場を空間微分するといった意味らしくて・・・・まぁ細かいところは今はいい動くプログラムができてから式の意味をじっくり考えても遅くはない。

∇とかわからなくてもプログラムは作れる。

今はとりあえず**動くプログラムを作ることが最優先**。

いろいろとすっ飛ばして、数値流体力学の解法を決めよう。

【粒子法か格子法か】

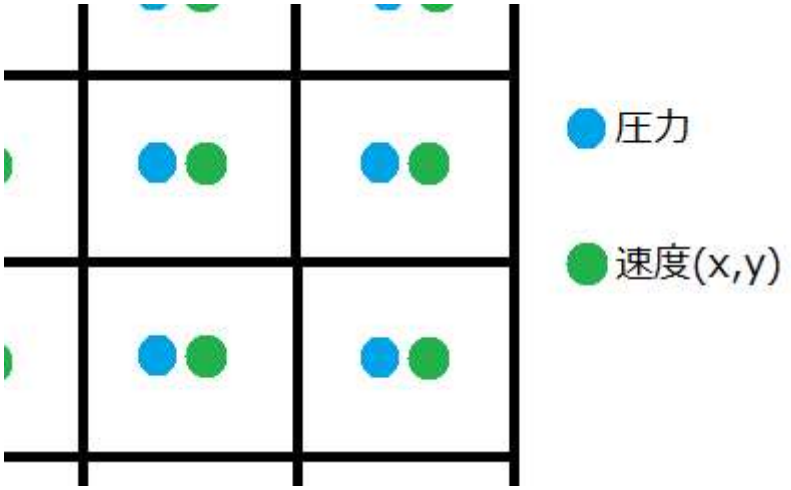
粒子法と格子方という方法があって、前者は水（流体）を粒子として計算、後者は水の入った空間を何百という格子で分割して各々の格子の圧力や速度を計算するというもの。

どうも格子法の方が簡単だということが調べでわかった。

格子法で流体力学の完成形、
例
[13]http://www.youtube.com/watch?v=4Ov6tguRr8E

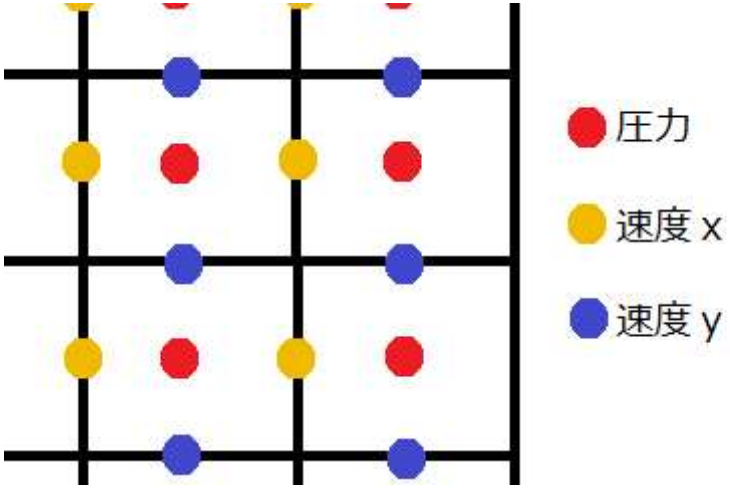
格子法では、2Dの場合、各々の格子の中の圧力、x速度、y速度、を時間ステップごとに更新していくものだ。

イメージ1 （通常格子）



さて格子法でやると決めた後は、どの種類の格子にするかを決める必要がある。大きく分けてさっきの通常格子（上）とスタガード格子（下）がある。

イメージ2 （スタガード格子）



（スタガード格子の方が少し複雑で難しい。が、プログラムを作っている途中でわかったが最終的にこっちの方が楽になる）
というわけでスタガード格子でやることに決定！

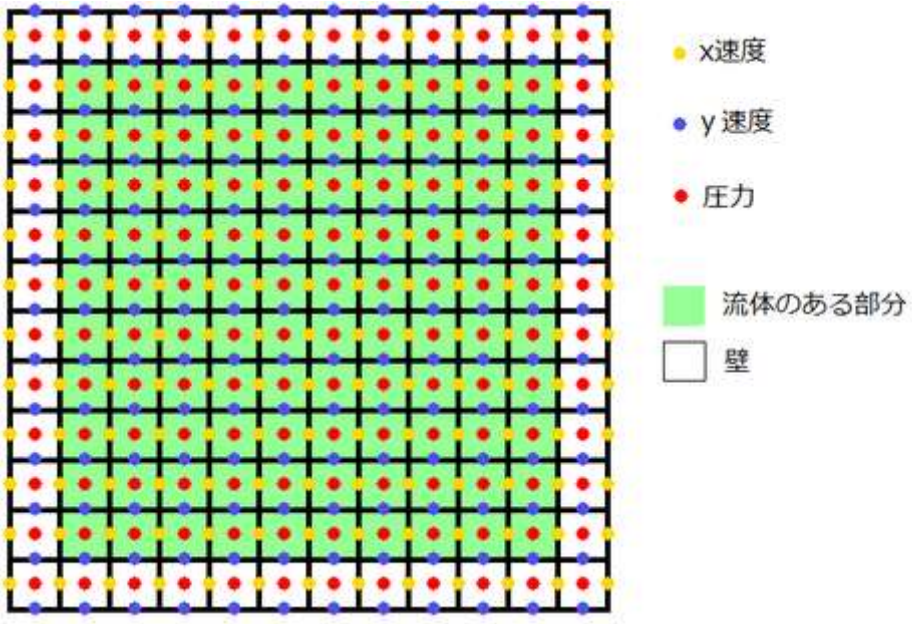
これは東工大のサイトが参考になった。
[2.1]http://www.eto.titech.ac.jp/contents/sub03/chapter08-1.html

で、あとはスタガード格子の速度と圧力をNS式でどのように処理して時間ステップを進めるかということだが・・・

まずはNS方程式をじっくり眺める前に、アルゴリズムで扱うデータについ考えてみよう

【全体像と変数】

イメージ3 （全体像）



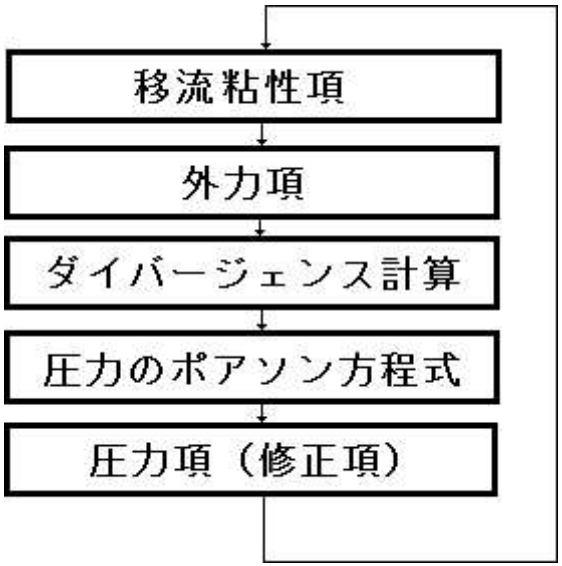
分り易くするために流体の分割は10×10格子とする。
壁を含めると全計算領域は12×12格子である。
データ構造はどうなっているかというとまずは圧力格納変数「p」、これは格子数と同じでいいので12×12の配列変数となる。
次にx速度格納変数「vx」、これは13×12要素あるので13×12の配列変数となる。逆にy速度格納変数「vy」は12×13の配列変数となる。
ついでに後々、ダイバージェンス(▽・v=0)の計算で使う各格子での「発散」値を格納するデータ「s」も作る必要がある。これは圧力と同じく12×12の配列変数となる。
また「p」「vx」「vy」の一時的な記憶変数として「p_after」「vx_after」「vy_after」もそれぞれ同じ配列で用意。

そして、以降**話を分り易くするためにΔxやΔyは1として考える**（少し衝撃的な決断だが）なので以降の式にΔxΔyが省略されているがそこはご了承下さい。
まともにプログラムが動いてからΔxやΔyをいじってみればいいのである。
またΔtは1にすると計算が発散するので、今は0.1くらいだと思ってもらえればいい。

これでデータ構造はOKではじっくり、NS方程式をプログラムに落としこんでいこう。

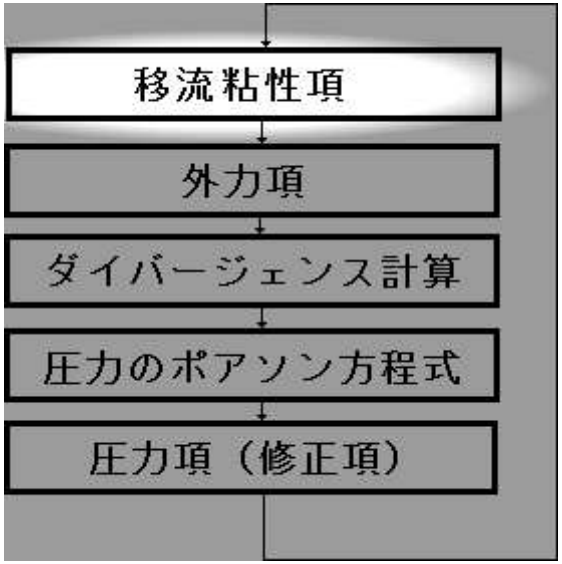
【フローチャート】

フローチャートはこうなる



まず上から見てみる

①移流粘性項



最初に移流と粘性を分けて考える。

・移流

移流つまり、流れる前の速度格納変数 vx vy をつかって一定時間流れた後の速度を求める、ということだ。
流れ後の速度格納変数をそれぞれvx_after　vy_afterとすると

（i,jは配列の要素、整数）

（ u>=0かつv>=0の場合 ）
vx_after[i][j] = vx[i][j] − u*(vx[i][j] − vx[i−1][j])*Δt − v*(vx[i][j]−vx[i][j−1])*Δt

（ u<0かつv>=0の場合 ）
vx_after[i][j] = vx[i][j] − u*(vx[i+1][j] − vx[i][j])*Δt − v*(vx[i][j]−vx[i][j−1])*Δt

（ u>=0かつv<0の場合 ）
vx_after[i][j] = vx[i][j] − u*(vx[i][j] − vx[i−1][j])*Δt − v*(vx[i][j+1]−vx[i][j])*Δt

（ u<0かつv<0の場合 ）
vx_after[i][j] = vx[i][j] − u*(vx[i+1][j] − vx[i][j])*Δt − v*(vx[i][j+1]−vx[i][j])*Δt

ここでuとvは、代入が行なわれる変数vx_after[i][j]の位置にあるx速度、y速度を意味する。よってu=vx[i][j]でいいがvは、上のスタッガード格子の性質上、x速度の定義点とy速度の定義点がずれているため

v=(vx[i−1][j] + vx[i][j] + vx[i−1][j+1] + vx[i][j+1])/4

と、4つの近くの格子点の情報を足して4でわる必要がある。

ここまででvxの移流の話が終わり、次はvyの移流である。
上の計算でvxと置き換えるだけでいいがuとvは

u=(vy[i][j−1] + vy[i+1][j−1] + vy[i][j] + vy[i+1][j])/4

v=vy[i][j]
とする。

これを計算領域内すべてのi jで計算する。
全計算が終わったならvx vy にvx_after vy_afterを代入する。計算領域はイメージ図1で言う緑の部分と接するか重なるところだけ。
つまり最外層の1層だけは参照されるのみで代入は行なわれない。

またまた東工大のサイトが参考になった。
[2.2]http://www.eto.titech.ac.jp/contents/sub03/chapter03.html

移流のイメージとしては例えば以下の画像の黄緑の部分の速度が(1, 1)で白の部分が(0, 0)、話を分り易くするためにΔt=1だとしたら

イメージ4 （移流）

こうなる

しかし実際はΔtや速度の値が 0.148 とか0.009とか不規則な数であり、これにより移流計算するごとに速度の情報が崩れまくる。
特にこれから述べる一次風上差分法はとてもその形が崩れやすい。

数値流体力学ではこのプロファイルの崩壊をいかに防いで移流計算するかということに、そのシミュレーション精度がかかっている。
精度を上げるためには、Cubic セミ・ラグランジアン法やCIP法を用いるとよい。
今回は単純で簡単な一次風上差分を採用した。

このサイトの座標のところがすごく理解の助けになった。
[3]http://www.eco.zag.jp/env_univ/euler.html

・粘性

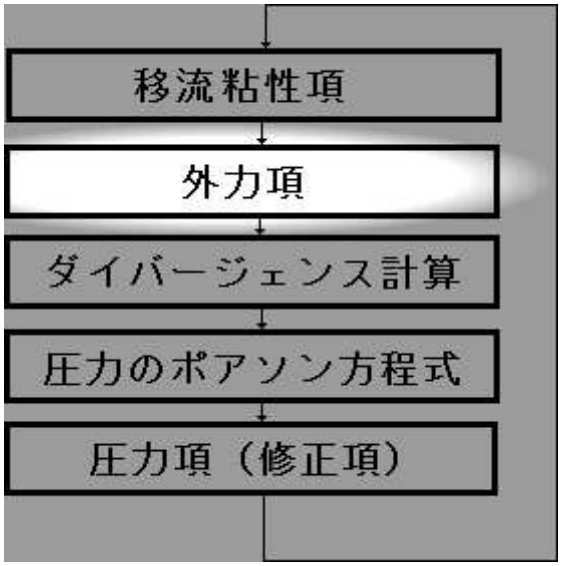
次に粘性であるがこれはただ単に、1つの点の速度が周囲4つ点に拡散するという意味である。
どのくらいの割合で拡散するかはRe(**レイノルズ数**)で決まる。

vx_after[i][j]=vx[i][j]-1/Re*(vx[i+1][j]+vx[i][j+1]+vx[i-1][j]+vx[i][j-1])*Δt

vyでも同じく・・・

全計算が終わったならvx vy にvx_after vy_afterを代入する。

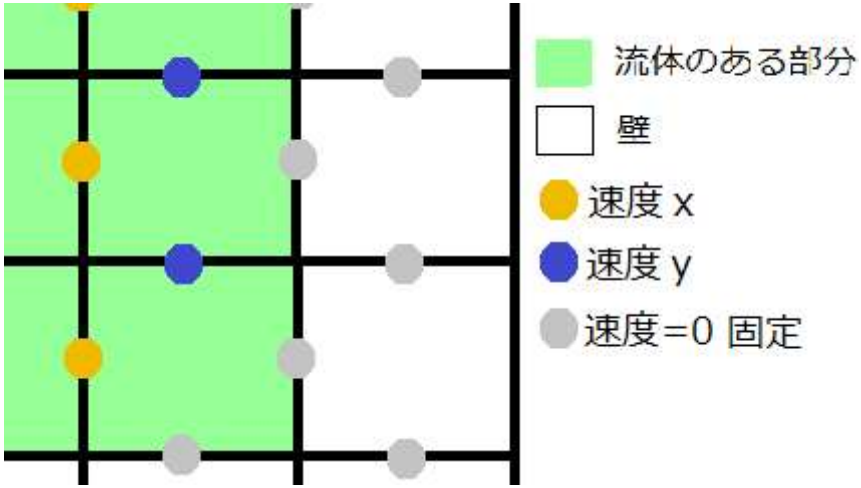
②外力項



マウスで流体に速度をつけたいのなら、ここで速度に加算する。

また、壁と接する速度定義点は本来0でないといけないはずだが、前行程の移流粘性項で近隣の速度定義点から影響を受け0でなくなってしまった定義点があった場合に、ここで**0に戻す**。
そして流出境界などもここで設定する。

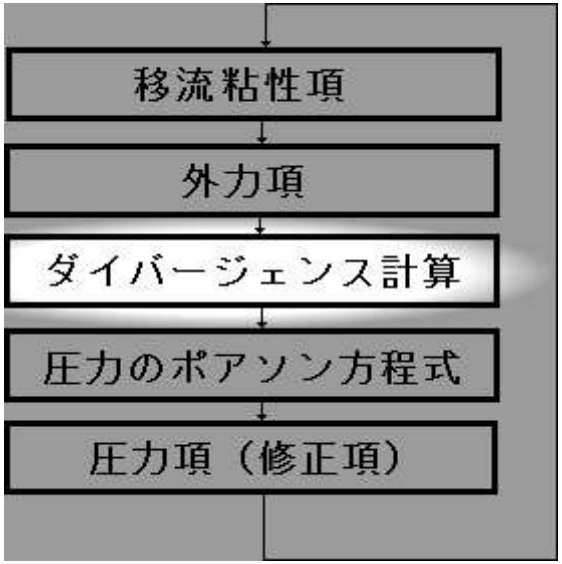
イメージ5 （壁の速度固定）



またまた東工大のサイトが参考になった！すばらしい！

[2.3]http://www.eto.titech.ac.jp/contents/sub03/chapter08-4.html

③ダイバージェンス計算



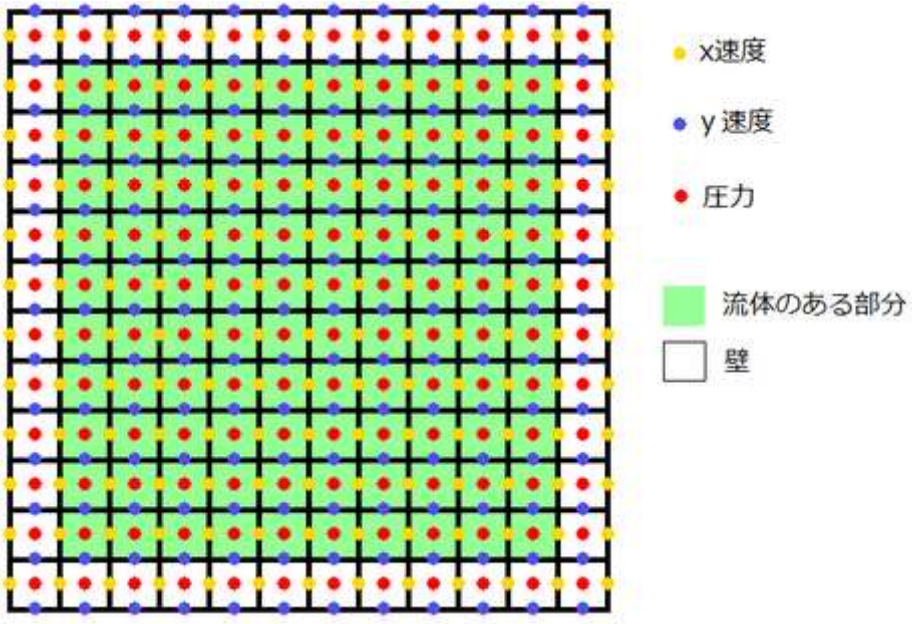
⑤で圧力値を使うため、ここでは速度からダイバージェンスを求める計算をし、④でダイバージェンスから圧力を求める計算をする。
「s」はダイバージェンス、つまり湧出を格納する変数だ。

s[i][j]=(−vx[i][j] −vy[i][j] +vx[i+1][j] +vy[i][j+1])/Δt

これは一定区間の空間から出ていく水の量をあらわしている。
なぜΔtで割るのかは、
[2.4]http://www.eto.titech.ac.jp/contents/sub03/chapter01.htmlを参照のこと。正直私はよくわからない。

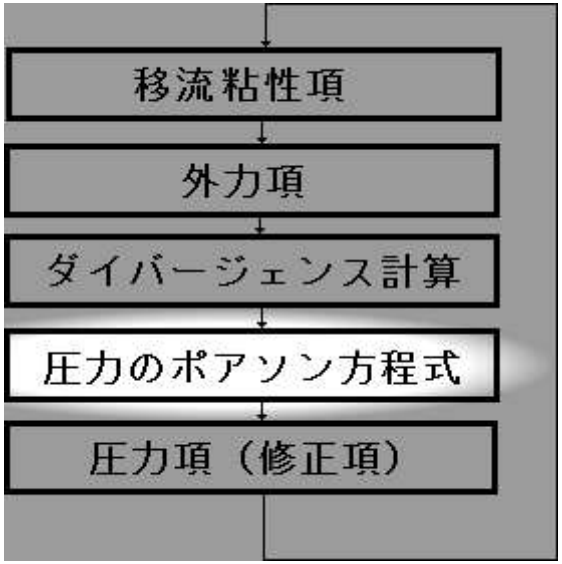
位置　i,j　の格子内で見た湧出は、それを囲む4つの速度から求められる。圧力定義点を囲む速度定義点は、先ほどの↓をじっくり見ればわかるであろう。ちなみに湧出と圧力の定義点は全く同じ位置である。

イメージ3（全体像）



ところで湧出を求める意味なのだがこれは、記事の上の方で▽・v = 0 を連立するという話につながる。▽・v = 0 つまり、どのi,jでも s[i][j] が0になる必要があると言っているのだ。
移流や粘性、外力項などで速度情報が何度も書き換えられたため、今のままでは▽・v = 0 を満たしていないと考えられる。これでは非圧縮の条件が満たされない。
これをいっきに解決するのが主に次の④である。

④圧力のポアソン方程式



ここから、流体特有の非圧縮という特性をプログラムで再現する、ある意味数値流体力学の真骨頂とも言える部分の計算に突入する！
上で求めたダイバージェンス値から、圧力を算出する。ダイバージェンスから圧力を求めるには、以下の反復法のSOR法を使うといい。

SOR法

$$p[i][j]=(1.0-\omega)*p[i][j] + \omega*(p[i-1][j] + p[i+1][j] + p[i][j-1] + p[i][j+1] - s[i][j]$$

ωは加速係数といって1.8～1.9で一番収束が早くなる。
$\Delta x \Delta y$を変えたい場合は
[2.5]http://www.eto.titech.ac.jp/contents/sub03/chapter07.html
の(7-4)式を参照のこと

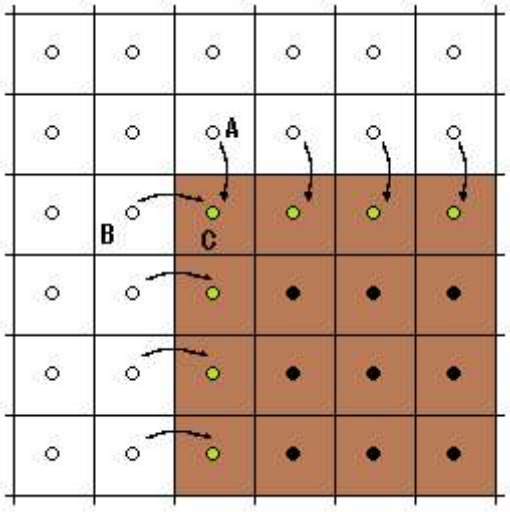
これを計算領域内のi,jで、何回も反復することで収束していく。
当然初期p値が解から遠ければ、収束までの反復回数は増える。ここが一番計算コストのかかる部分なので、ここをいかに効率良く解くかで、全体の処理速度が決まってくるといってもいい。

さて、ここで**2つ問題**がある。

1つ目は計算領域
これは移流項の速度計算と同じで、最外層1層は計算しない。
つまり配列が**12×12**なら、iとj （0～11） はそれぞれ　**1～10**の内である。最外層は参照だけされる。

格子が壁の場合は、壁内の圧力を計算しない。
隣の格子が壁の場合、SORの処理の前にその壁に圧力をコピーしておく↓

イメージ6



薄茶色が壁、白が流体の存在するところである。
このとき圧力定義点の白丸はSOR法で代入も参照も行なわれる。
壁の中の黄緑色の圧力定義点は、となりの流体の圧力が代入された後参照される。
黒丸は参照すら行なわれない。

つまり・・・例えば、Bの圧力をSOR法で計算しようとする時、Bの上下左右の圧力が参照されるため、Cの圧力が参照されることになるが、Cは壁内にあるため、ここにはBの圧力が使われる。よってBの圧力計算をする前に、CにBの圧力を代入しておく。次にAの圧力をSOR法で計算しようとする時、やはりCにはAの圧力を代入しておく必要がある。

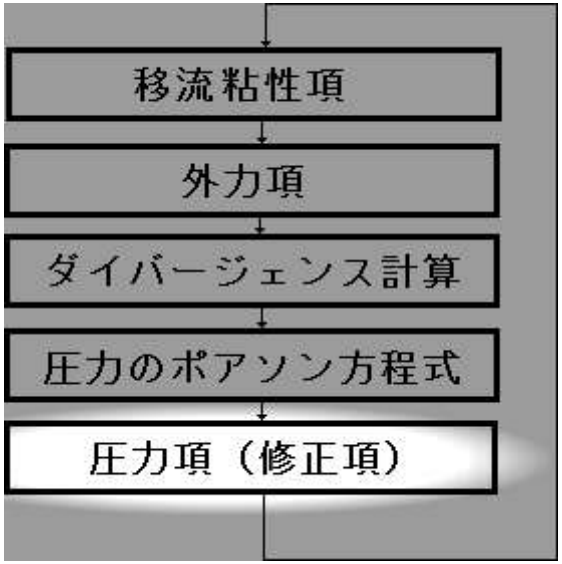
結果的に黄緑色の部分の本来の圧力というものは無視される・・
まぁ壁の中に圧力なんてないわけなんだが・・

ここでも東工大のサイトが一番参考になった。
というかコチラのほうがわかりやすい！もう熟読をオススメする！
[2.6]http://www.eto.titech.ac.jp/contents/sub03/chapter08-2.html

さらにこのサイトによると、格子の壁情報を格納する変数を作ると便利だということだ。
ところでSOR法は参照する4つの圧力情報が、計算後の情報と計算前の情報の2種類が混じり合っているように見えるが、それで良い。

2つ目は反復のループから抜け出す条件「p」に現在計算中の圧力情報を、「p_after」に1ループ前の圧力情報を格納しているとしてp[i][j] - p_after[i][j] の**絶対値**が、すべてのi,jで**一定値以下**なら「収束した」と判断する。
ただこんな作業を毎ループやっていると、計算量2倍になり非効率なので、私は10ループに1回にするなど、i,jを1つ飛ばしでやるなど工夫していたりする。

⑤修正項



圧力により速度が修正される。ここで、修正されるというのは、div v =0 になるように速度が調節されるということである。

vx[i][j]-=(p[i][j]-p[i-1][j])*Δt
vy[i][j]-=(p[i][j]-p[i][j-1])*Δt

⑤はこれだけ

これでやっと全ての処理が終わり。
vx vy p の値全てが次のタイムステップに進んだことになる。修正項のあとは、可視化ルーチンに回すなり、ランダムに配置した粒子の移動ルーチンに回すなり好きにすればいい。
そして処理は①へ

最後に一つ注意すべき点は移流項。数値が発散するとすれば移流項だからだ。
Δtを適当に細かくしておけば発散しない。

ちなみに、この流体力学のサンプルソースを配布している。
↑のメッセージボードからDONWLOADできる
見るにはHSPのインストールが必要

【おしまい】

と、とりあえずNS式の意味を深く考えず、ここをこうすればいいと計算の手順の部分だけ書いてきたが、とりあえずこれで動くはずである！正直、最初に流体力学をやろうと思ったときは「▽」とか「ベクトル場」とか「ラプラシアン」とか全く分からなかったが、見よう見まねでプログラムを動かそうとしているうちにだんだんと式や単語の意味が分かってきた感じなのだ‥

私は、まずは動くプログラムを作ってから、式や単語の意味を考えるのが一番効率がいいと思う。なのでこの記事ではあえて、そんな専門的な言葉を使わないで進めてきた。
そもそも専門書を買って勉強していたりするわけではなく、威張って用語を使えるわけでもない。

数値流体力学は奥が深く、まだまだイロハのイくらいしか喋っていないだろうが、何かわかったことがあり次第ブログでつぶやいていこうと思う。

参考にしたサイトやらなにらや

[1]http://ja.wikipedia.org/wiki/%E3%83%8A%E3%83%93%E3%82%A8-%E3%82%B9%E3%83%88%E3%83%BC%E3%82%AF%E3%82%B9%E6%96%B9%E7%A8%8B%E5%BC%8F
[2]http://www.eto.titech.ac.jp/contents/
[3]http://www.eco.zaq.jp/env_univ/euler.html
[4]http://d.hatena.ne.jp/etopirika5/20110425/1303740299
[5]http://www.bee-www.com/log.htm
[6]http://journal.mycom.co.jp/articles/2007/10/09/cedec01/001.html
[7]http://www.4gamer.net/games/000/G000000/20070926041/screenshot.html?num=012
[8]http://fluid.me.seikei.ac.jp/lecture/cfd/cfd-08-NS-algorithm.pdf
[9]http://staff.aist.go.jp/kogaki.t/Dissertation/PDF_files/chap3.pdf

[10]http://homepage.mac.com/catincat/java/index.html
[11]http://oshiro.bpe.es.osaka-u.ac.jp/people/staff/imura/lecture/2007cg/fire071220.pdf
[12]http://www.youtube.com/watch?v=6rPL-QkUFf8&feature=related
[13]http://www.youtube.com/watch?v=4Ov6tguRr8E

次回：HLSLでリアルタイムレイトレーシング

「流体力学」カテゴリの最新記事

流体力学のゲーム完成！「流体 de 月面着陸」
流体力学のプログラムを作りたい！その3
流体力学のプログラムを作りたい！その2

「アルゴリズム」カテゴリの最新記事

「FFTモジュール活用例2巨大整数の乗算」のソース
FFTモジュール活用例1スペクトラムアナライザーのソース
円周率小数点以下第1000億桁まで計算するproject

コメント一覧

1. Nao　2011年09月24日 10:25

   えー来てみました
   アメブロ読んでたんですがね、全くわかりませんでしたが。
   こっちにも顔出すようにします。

2. ぴっぴ　2011年09月24日 12:31

   Naoさんはアメブロの時からお世話になってた唯一の読者かもしれません（泣）いつも読んでくださっていたんですねm(__ __;)ありがとうございます。
   心機一転これからもよろしくお願いします。

3. CANGOXINA　2011年10月08日 00:47

   メカマさんの記事からジャンプ！

   私も流体力学(専攻)や、HSP(独学ド素人)を勉強していましたので、
   記事の内容にとても興味を持ちました。

   これからたまにコメント残しますので、

よろしくお願いします～～。

※勝手にリンクさせていただきました。
ご迷惑ならリンク外しますので、ご連絡ください。

4. pippi　2011年10月09日 18:15

コメントありがとうございます！リンク嬉しいです！
流体力学は興味があって勉強してみたのですが、まだまだド素人なので暖かく見守っていて下さい（笑
こちらもCANGOXINAさんのブログに顔を出させてもらうつもりなので、よろしくお願いします！

5. seasalt　2011年11月16日 23:40

コメント、リンクありがとうございます！

こちらからもリンクさせていただきました、
というご報告と
コンテストではありがとうございました。
を言うのを忘れていましたので改めまして
ありがとうございました。

よろしくお願いしまう。

6. ぴっぴ　2011年11月19日 00:39

> seasaltさん
リンクして下さってありがとうございます。
こちらはとても更新の遅いブログですが、どうかよろしくお願いします！

コメントする

名前
メール
URL
情報を記憶　評価 -- ▼　顔　星

投稿する

〈 HSPで高速並列演算 | HLSLでリアルタイムレイトレーシング 〉