マトリックス：世界をシミュレート　第1部ー粒子モデル

http://plus.maths.org/issue42/features/dartnell/index.html
by Lewis Dartnell

モデルの構築は，科学や工学の研究の多くの分野の核心である．モデルの本質は複雑な系の表現で，複雑な系のふるまいの理解をするために，異なった種々の方法で系の単純化がなされてきた．例えば，航空技師は，風胴テストのために戦闘機のミニチュア模型を作るかもしれない．現代は，たいへん高速に数学モデルをコンピュータ上で走らせるモデリングが，ますます盛んになっている．超音速気流のコンピュータモデルは，信じられないくらい複雑だが，プログラムのデザインとシミュレーションは非常に基礎的な原理に基づいている．モデルのふるまいに関するこの論文の前半で，たいへん興味深い自然系研究のコンピュータモデルが，いとも簡単にプログラムできることを述べる．先端研究に同様なモデルを用いている科学者の何と少ないことか．

開始

どちらに行くのか？魚の行動をシミュレートする方法を見出せ．

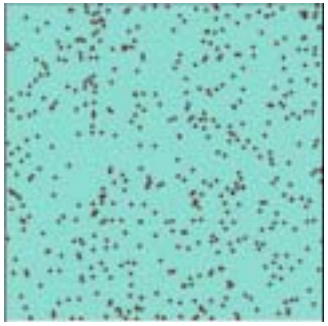
これは，数学的なモデリングとコンピュータシミュレーションへの体験入門である．だが，プログラミングそのものの学習には深くはかかわらない．これまでにプログラムを見たことがないとしても，心配は要らない．シミュレーションのすべては，このウエブサイトでJavaビデオとして見ることができる．もし諸君がコンピュータプログラミングをすこしやったことがあるなら，この論文で用いた全コードをダウンロードして，改良や調節など試してみたいだろう．これらのシミュレーションを書いたり，アニメーション作りに用いたソフトウェアはプロセッシングと呼ばれ，無料でダウンロードでき，ＰＣやMac バージョンで利用可能だ．プロセッシングはコンピュータ科学者とアーティストの協力でリリースされ，自分で容易に改良や開始ができる．プロセッシングが使われた世界中の種々プロジェクトすべてのリストをホームページで一見されたし．

よいモデルを組み立てる本質は，複雑な問題をいかにうまく単純化し，系の重要な特徴を抽出し，モデルのふるまい解析を混乱させるものは取り除くように考えることだ．例えば、ライフル銃射程の下方に発射された銃弾弾道の単純なモデルは，明らかに重力の影響を考慮する必要があるが，空気抵抗のわずかな影響は無視してよい．この場合の空気抵抗は，2次オーダーの効果と呼んでよい．別の系のモデリング，航空機の翼による揚力では，空気の影響を無視することはできないが他の因子は無視できる．コツは，そのふるまいが理解しやすいように，モデルをできうる限り単純に保つことと，意味のない結果が生じないように重要と思われる因子をあまりカットしないようにし，入力

因子の一つを変えて全系の応答の影響を見ることとのバランスにある．
この最初の論文のすべての例は，各点が空間内を色々な規則に従い動き回る粒子モデルとして知られるものである．

箱の中のガス分子

最初のモデルは，箱につめられたガス分子の大変基礎的な物理シミュレーションだ．見やすく単純にするため正方形内に閉じ込められた2次元粒子を見る．この場合の物理は簡単である．各粒子は，箱の壁に当たるまで，出発したときと同方向・同スピードで直線上に運動し続ける．壁に衝突すると，粒子は跳ね返り方向を変えるがスピードは変えない．
映画と同じで，この論文のアニメーションは流体の運動を印象づけるフレームのシリーズからなる．プログラムの仕事は各粒子の位置，方向，スピードを各時間ステップごとに前のステップの情報に基づき計算することである．
この論文のすべての例では，各時間ステップごとのデータを配列または行列に蓄える．行列の各行は異なる粒子に関する必要な情報を蓄える．このガスの例では，各粒子の状態は4つの数で記述される．これらのうちの2つは平面内の位置 (x,y) である．残りの2つは，粒子の運動の2成分を記述する．これらは$deltaX, deltaY$と呼ばれる（"X位置の変化"，"Y位置の変化"の意）．粒子の速度も定義する．この場合の行列は4つの列を持ち，全粒子に対する十分な行数を持つ．シミュレーションの各時間ステップごとに，各粒子の位置 (x,y) は，その速度に依存した更新を受ける．
このプロセスのモデルを記述するプログラムの構造は，かなり直接的である．どのように機能するか一般的な要点がつかめるかコードを見てみよう．
最初にするのは，モデルの最も重要なパラメータ（含まれる分子の数，各粒子の位置と速度の情報をストアする行列など）の定義である．
次に，プロセッシングは，世界の大きさーこの例では正方形，アニメーションのフレームレートーを決め，シミュレーションのセットアップをする必要がある．それからシミュレーションの初期化をする．これはデータを蓄える行列で，各粒子の初期位置，速度の必要な情報を順次蓄える．今回は，粒子のランダム散布を選んだ．次に，トルコ石色の背景と各粒子の (x,y) 位置に置かれた赤い円で系の現状を記述するコードがある．
最後に，シミュレーションの中核，アップデート関数が来る．各時間ステップごとに，この関数は，行列中の各粒子に順次あたり，時間ステップ内に，その速度が箱の境界を越えその粒子を飛び出させないかどうか計算する．もし飛び出すなら，箱の壁で跳ね返り，粒子速度は変化するが，そうでなければ，その速度ベクトルで決められる距離だけ前進し続ける．かくして，各粒子は新しい位置と新しい速度ベクトルをもち，行列は新しい値にアップデートされる．



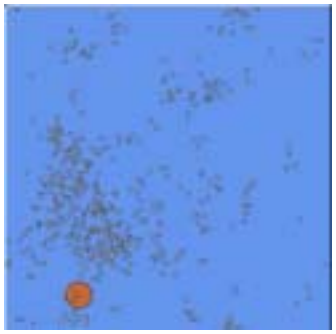
箱に閉じ込められたガス分子のモデル．Java始動には映像をクリックオン．

全プログラムは，粒子が系をアップデートした後，再び描画し，粒子はもう一度アップデートされ，これが繰り返されるように，ループにセットされる．高速な現在のコンピュータでは，これがたいへん高速に行われるので，ガス分子のスムーズなアニメーションになる．
この絵はプログラムのJAVAバージョンを開くようになっているので，プロセッシングソフトウエアをインストールしなく

ても，動作を見ることができる．
この過剰に単純化したモデルは，2つの問題点があることに気づくだろう．第一に，我々のプログラムではdeltaX，deltaYに整数値だけ使っていること．このため粒子が動ける異なる方向の多くがなくなってしまう．粒子のいくつかは，両サイドでバウンドし往ったり来たりするので，完全に水平や垂直に動くのが見られる．第二に，このコードは，分子間の相互作用を考慮していない．分子は決して互いに衝突せず，箱の中を跳ね回るだけで，完全予測可能である．周期的に，はじめがそうであったように，全分子が外向きに散らばる前に，真ん中で密なクラスターが形成される．こえは明らかに部屋の中の空気分子では起こらないことだ．空気分子は絶えず互いにぶっつかりあい予測不可能な運動を生み出している．
もし諸君が，少しコンピュータプログラム知っているなら，この単純な例をスタートとして，もっと洗練され現実的なものを作りたいと思うだろう．これをどのように行うかの若干の助言は，プログラムファイルの末尾にある．
運動のモデリングのもう一つの例に移ろう．今度は，粒子の相互作用の規則がもう少し複雑である．

鳥の群れ

モデル化する系は鳥の群れである．はじめの例のように完全に独立に動くのではなく，各粒子は他の粒子の動きに反応して動く．同様な標準構造をガスのプログラムにも用いるだろう．最初はパラメータと世界の大きさを決め，全粒子の位置と速度を初期化する．次に，系の現在の状態を表示する関数のループをまわし，次のタイムステップでの系の変化の計算を行う規則を実施する．
最初の例は，閉空間に閉じ込められた粒子の運動のシミュレーションをねらった．今度は，開かれた戸外で飛ぶ鳥がどのように群れを作るか調べるのが関心事だ．エッジがあるような世界は困る．なぜなら，そのような人工的な特徴の周りでは，シミュレーションが適切にふるまわないであろうから．モデリングの共通のコツは，世界空間を，粒子が左から去れば右から現れる，上下も同様［周期的境界条件］に世界空間を定義することだ．上下は互いにつながりチューブのよう．左右端を丸く曲げてつなぐ．ドーナツの表面の世界のようだ（数学ではトーラスという）．



鳥の群れモデル．単純な規則の集合が，他の鳥との相互作用を定義し各粒子の運動に作用する．燈色の円は視野範囲を示す．イメージをクリックするとシミュレーションのJavaバージョンが走る．再スタートは［Ctrl］＋［F5］（Windows）または，［z］＋［R］（Mac）．

このモデルでの第二の発展は，直線運動を続ける粒子のかわりに，鳥たちは互いに相互作用をし近隣者に依存し飛ぶ方向を変化させることだ．各時間ステップごとに，プログラムは，次々に各鳥を選び，選んだレンジ内で他のすべての鳥の飛行方向の平均を計算し，選んだ鳥はこの方向に舵をとる．ランダムに選んだある鳥の視野レンジの場を燈色の円で表示させる．プログラムで値を蓄える変数はeyesight．プログラムがどのように機能するか一般的な要点を拾い上げるため，コンピュータ・コードを一寸見てみよう．次に左のイメージのシミュレーションを走らせて見よう．
始めはランダムであった鳥たちの運動が，直ちに秩序のある振る舞いに変わる．鳥の巨大な群れが，大体同じ方向に飛び行くように自己組織化される．孤立した鳥が一団の中に引き込まれる．そして時折2つの大きいグループがお互いに近づいて迷うとき，個々の鳥は他の群れに参加するために引き剥がされる．2つの雲がスムーズに混ざ

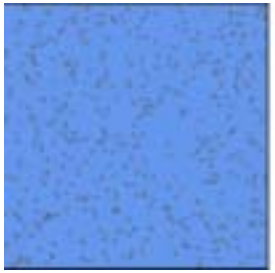
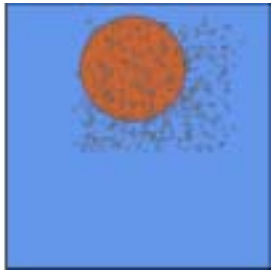
り合うようだ．ちょっと迷って，そして次に新しいグループの先頭が決まる．この設計された行動を本当の鳥一団の動的関係と比較してみよう．
これを書く時点で，あるホップベースの飲物の英国テレビで放映されている素晴らしい広告がある．夕映え空のツバメの群れに魅せられる．タグ ライン「属します」で終わる．この広告はYouTube から見ることができる．たとえすでに消されていたとしても，他の鳥の群れビデオを見いだすのは全く容易であろう．

巨大スケール鳥群れを示すYouTube広告

この運動は優雅な振り付けに見える．個々の鳥が次に何処に飛ぶか正確に知っているように見える．鳥の巨大な流れが，滑らかに調和し一斉に方向を変える．群れの中にリーダーはいない．基本計画もない．すべての決定はグループ力学だけで決まる．何千という群れをなす鳥の魅了される複雑さのすべては，最近接の隣を越えた残りのグループが何をするかを知ることなしに，個人が行動するというたいへん単純な規則から発する．これはボトムアップ制御として知られる．単純な規則で相互作用している個体から，たいへん複雑なグループ行動が出現する例だ．たくさんの動物，ミツバチ，スズメバチ，アリ，シロアリ，．．．が，この種の「群れ知性」を使う．
しかし，群れの振る舞いの基礎となっているこれらの単純な規則は異常な環境では，まったく馬鹿な行動を惹き起こすことがある．自分自身ではまだ試す機会がなかったが，友達の友達から，諸君が羊の一団と一種にやれる面白いトリックの信頼できる情報を得ている．羊達の前で突然走ったとする．羊達は脅威を察知し動き去ろうとする．まだ可能な限り，諸君の隣人に近い状態を保って，同じ方向に動作することが最も安全であるという規則は働いている．もしあなたが羊より少し速く走り続けるなら，あなたは追いつき，そして一団の中央を通過して先頭で今走っている．グループの動的力学が，侵略者から逃げ出すという初期の個体の決断を引き継いだ，そして諸君は逃げようとするのではなく，群れ全体を諸君の後ろに従えていることになる．

コウモリとタカ

プロセッシングソフトとこのモデルのプログラッミングコードをダウンロードしたら，グループ全体のふるまいにどのように影響するかモデルのある特徴をいじることができる．例えば，鳥同士は何処まで見えているのか，eyeSight変数はたいへん重要なパラメータだ．我々のプログラムでは，この変数は20にセットされている（とりの視野を表している燈色の円の半径が20単位）．このパラメータは諸君の望む如何なる値にもセットできる．2つの異なるシナリオを見てみよう：”コウモリのように目が見えない”eyeSightは1とする．もう一つは”タカの目”eyeSightは100とする．地図をよぎって見る事ができる．下の2つのリンクをクリックすれあば，どちらのシミュレーションも見ることができる．

コウモリのように目が見えないeyeSight＝1　タカの目eyeSight＝100

第一のシナリオでは鳥たちは相互作用をまったくしない．だから実際は世界空間内でランダムに走るガス分子と同じだ．第二のシナリオでは，鳥たちはグループの反対側にいる個々の鳥の飛行方向に反応するくらい，たいへん遠くまでコミュニケーションする．グループのふるまいはたいへん速く一体化した固まった運動になる．どちらの場合にも，少なすぎるか多すぎる相互作用のために，すべての複雑な群れのふるまいは失われる．系のアウトプットはこのパラメータにたいへん敏感である．気体的ふるまいと固体的なふるまいの間にのある種の相転移がある．興味あるダイナミックスの見地ではどちらも死んだも同然だが，自然ではeyeSightの値がこれら両極端の中間値のときにのみ緊急行動が起こることが見られる．

揺動させる

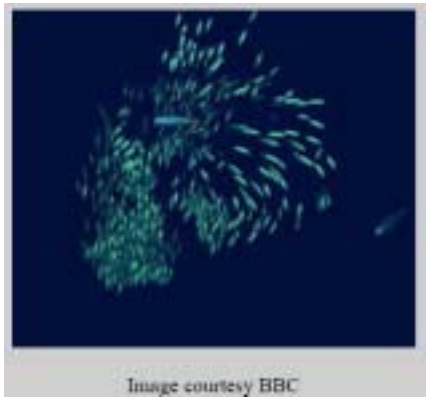
もう一つの重要なモデリングパラメータはランダムジグリングである（プログラム中にrandomJiggleと記す）．各鳥の周囲の平均飛行方位の査定がいつも正確になされる訳ではないという事実を考慮に入れる．プログラムは時時刻に各鳥の方位を更新するとき，ランダムジグリング変数により与えられる範囲内のランダム角で調整される．10°より小さいランダムジグル値なら，群れは最終的に一方向への一体運動になる．randomJiggleを180°にセットすると系は塵微粒子のブラウン運動のシミュレーションと同じになる．ガス分子の例のように，このモデルをさらに自然に近づけるいくつかの方法がある．このヒントのいくつかはプログラムファイルの末尾にある．

単純さと速度

モデル記述で重要な因子は，用いるアルゴリズムの効率である．”アルゴリズム”の起源は，アラブの数学者al-Khowarazmiの名前であるが，”仕事を完成させるための一連の指示”を意味するようになった．ここの鳥の群れの例で用いたアルゴリズムは，おそらくそれほど自然ではない．本当の鳥は，他のすべての鳥の位置を記録し，自身と他のすべての鳥間の斜辺距離を計算し，正確に5m内のものを選択し，それらの進路の算術平均の方位に向けて舵を切るなどということはしない．これらのステップは，追従が容易で，望ましい結果が得られるので，モデルとして選ばれたのだ．それは，なかなか冗長な方法だ．一つの鳥から他のすべての鳥までの距離を計算しなければならない．これを順番にそれぞれの鳥について，各時間ステップごとに行う．高速なデスクトップコンピュータでも，鳥の数が500より大きいとシミュレーションはゆっくりゆっくり進む．

Iain Couzin

イアンは，プリンストン大学とオックスフォード大学の両方に拠点を置く動物行動の専門家だ．バッタから魚，鳥に至るまでの群れのダイナミックスのコンピュータモデルを作った．彼の研究は，如何に動物の群れが集団としての決定をなすか驚異的な特徴を見いだすことと，アフリカの政府機関の行うバッタの致命的な群移動コントロールを手伝うことだ．最近まで，イアンは群れがどのように肉食動物のアプローチに反応するかに集中していた．諸君はイアンの研究の詳細をhttp://www.princeton.edu/~icouzin/で読むことができる．下の映像は彼のモデルの一つから作ったビデオの静止画だ．BBCのドキュメンタリシリーズ”肉食動物”で使われた．



そこでモデルのデザインでは，仕事をどのように達成するか考える必要がある．可能な限り速いアルゴリズムを使うか，現実の系の現実の方法に近づけるか，単純に効率は悪いかもしれないが仕事はできる方法にするか．プログラミングで，しばしば，特定の仕事部分のアルゴリズムを関数にするのは良いアイディアだ．こうすればプログラム内で特定の計算が必要になったときいつでも呼び出せる．（鳥の群れのコードを見れば，MeanHeading()関数でこれがなされるのを見るだろう．）

見守る間にシミュレーションがずっと速く走るようにしたければ，プログラムを2つの段階に分割するとよい．最初は，コンピュータを貪り尽くすようなすべての計算を通して行い，各時間ごとに系の状態を別々な行列として記録する．この計算の後は，例えば1000回の時間間隔の行列リストの束のような3次元行列に行き着く．シミュレーションを一回通しで行えば，すべての情報が保存される．第二のステージは時間ステップごとに順番に表示する（今度は計算処理のために待つ必要はない）．あたかもアニメーションを見るのに，行列の本の頁をパラパラするようなものだ．

我々は，単純な鳥の群れシミュレーションに絞ってきた．コンピュータプログラムの構造，特定の仕事をするアルゴリズムや関数の記述などのコンピュータモデリングの重要な様相が浮彫りになるからだ．このように単純なモデルでも，複雑で大変自然に近い群れのふるまいが作れる．これはまさに，動物研究者が作ろうとし，理解しようとした鳥の群れのふるまいや，魚の群れのコンピュータモデルだ．例えば，Iain Couzin博士が動物のふるまいを理解するために巨大グループのコンピュータモデルを使つている．彼のモデルは，この論文で見てきたものとまったく同じ原理に基づいており，同様の複雑性がみごとに出現する．モデルはさらに複雑になっており，（我々の2次元平面を超え）3次元で動き，冗長だが現実に近い各個体のふるまいを制御する規則の集合を持つ．

重力モデル

この種類の粒子運動モデリングの拡張は，地図をよぎって伸びる効果を含めることである．このわかりやすい例は，重力である．重力は消えそうなほどわずかかもしれないが，太陽の重力の影響は非常に長距離まで達する．processingを使い単純な重力モデルを作るには，太陽としてスクリーンの中心に小円を描き，他のすべての点から太陽までの方位と距離を計算する関数を記述する．これで，諸君のモデル化した世界にいる粒子が影響をこうむり速度を変じる重力を計算できる．ランダムな位置に置かれランダムな速度を持った惑星でシミュレーションを初期化し，これらの太陽の周りの円弧運動をアニメートしよう．次の時間ステップのこれらの位置は，現在の重力と速度で決定される．諸君はプログラムを，各惑星が後ろに軌道の軌跡を残すように変更することもできる．（draw()関数BYの背景コマンドを取り除く．ラインの始めに//でコメントにする）．

何十億という星がアンテナ銀河の衝突の間に形成された．下に示した銀河衝突モデリングで色々見いだそう．このイメージはハッブル宇宙望遠鏡で撮影された（NASA提供）．

太陽系の表示のために，各惑星の円軌道を生む速度を注意して解く必要がある．太陽系の圧倒的な支配力，太陽の重力だけを含めば，第一近似で正確なモデルができる．だが，巨大ガスの木星は他の惑星に注目すべき影響を与える．このような二次的な効果を含めるなら，さらに正確なモデルができる．水星の軌道を完全に観察に合わせるには，もっと複雑なレベルが必要で，アインシュタインの相対性理論ーこれはNewtonの重力の法則よりも，ある特定の状況では正確ーを含める必要がある．再度，モデリングのコツは，月面に人を着陸させるアポロプログラムが含める必要のある詳細の最小量を巧みに考慮することだ．単純な Newtonの重力モデルで地球と月の影響以外のすべてを無視している．
すべての粒子が互いに相互作用するもっと大きな重力系ダイナミックスのモデリングは，非常に高速なコンピュータで膨大な計算が必要で，極端な”計算浪費”である．しかし，このような数値シミュレーションは多くの研究者にとって極めて重要だ．例えば，下の枠中に，2人の主導的な研究者の仕事に焦点を合わせ，過去と未来の何十億年のイベントを見る．世界を打ち砕く衝突を通して月が作られ，我々自身の銀河と我々の最も近くの隣人の巨大な重力の引きが，何十億年もの年を超えお互いを分裂させるであろう．

Robin Canup
ロビンはデキサスのサウスウエスト研究所の宇宙科学者だ．彼女は月がいかにして形成されたかに興味を持ち，そしてそれが太陽系の初めに，若い地球はより小さい原始の惑星に衝突した時の「大きいピシャという音」イベントから生まれたという理論をテストした．彼女のコンピュータモデルは衝突の熱で地球全体が融け大量の岩が宇宙に放出され，その多くは衛星の軌道でに合体し月になった．http://www.boulder.swri.edu/~robin/でもっと多くを見ることができる．
John Dubinski
ジョンはトロント大学で全銀河のダイナミックスを研究している宇宙物理学者．我々の銀河，天の川，アンドロメダと呼ばれるらせん銀河の隣人は重力的に互いに引き合っていて，500,000 km / 時間でお互いに向かって落ちて行く．2人がお互いを破壊する（とき・から・につれて・ように）、これらの2つの巨大銀河が星の大きい細長い布をもぎ取って、合流し始めるであろうとき、ジョンは将来の30億年の時間のモデル作りにスーパーコンピュータ使い，これらの2つの巨大銀河が混合し始め，星の巨大な細長い布をもぎ取り2つの裂け目のようになる．驚異的なのは，このすべての混乱にもかかわらず個々星間のギャップは非常に大きいので，実際には一つも衝突しないであろう．我々の太陽の運命は不確実である．しかしそれはあるいは銀河間の宇宙の暗虚に排出されるか，混合銀河の密集しているコアに飛び込むかである．諸君は，http://www.cita.utoronto.ca/~dubinski/tflops/で，地球の夜空の景色も含めて，さらに色々な映画を見ることができる．

（訳：谷　克彦）