# 複雑系プログラミング特論 まとめ

この講義では，いくつかの複雑系に関するプログラミングを行ってきました．複雑系と一言で言っても内容は様々なため，できるだけ広いトピックを取り上げ，その雰囲気を味わってもらいたいと思いつつ進めてきました．

## 複雑系的な面白さ

（この講義で扱ったような意味での）複雑系，特に構成論的アプローチに基づくモデル研究では，これをするとこういう結果になるというような，あらかじめ容易に予測される結果を出すためにプログラミングするという訳ではなく，むしろ，自明でない現象があらわれる条件を，モデルを作って動かして探るという過程を一般的にとりま す．

とは言うものの，やみくもに適当にモデルを作って面白い結果がでる訳ではないので，ロジスティックマップや一次元CA，シェリングの分居モデルといったやや基礎的なモデルを題材として，特に，系の挙動を特徴づけるいくつかのパラメータが与える影響を可視化し，調べることを主眼においてきました．
なので，単にこうなることを確認しましょうというより，どうなるかとか，なぜそうなるかといった考察を含めたやや取っ付きにくい課題をいくつかだしました．
例えばCAで言えば状態の分布と情報量など，系の振る舞いを様々なレベルや切り口で可視化することが，こういった系の理解に重要なことが伝われればと思います．

同時に，複雑系的な手法が具体的な問題解決に応用される例として，遺伝的アルゴリズムを用いた最適化についても紹介しました．

系をつくって動かして理解したり，問題解決する過程の面白さの一端が伝わったなら幸いです．

## pythonでのプログラミング

この講義では，pythonを使ってプログラムの実習を行いました．個人的な経験上，プログラミングは必要にならないと身につかないという考えから，モデルの記述 に必要なものを適宜少しずつ紹介する，活用法についても今取り組んでいる具体例をもちいて説明する方法をとりました．そのメリットがあったことを期待しています が，一方，もう少し系統立った説明をする機会もあった方がより良かったかもしれません．

今回いくつかの例を示した通り，pythonの大きな特徴は，いわゆる疑似コードに近い直感的なプログラミングが可能なことでした．リストの内包表現など，やや特 殊な記述法を知っている必要がありますが，プロトタイピングのための言語としてはとても便利なもので，実習を通じてその一端は伝わったのではないかと思います．実 際，欧米では，計算機科学を専攻するプログラミングの授業でCを使わずにpythonをはじめに使う場合が増えています．

もう一つのpythonを使うメリットは，可視化の為にmatplotlibというライブラリを利用なことでした．プログラミム中の配列やリストの中身を 表示する単純なグラフであれば，簡単にきれいなものがかけることを主に紹介しましたが，用途に応じて詳細な設定をかなり自由に設定できるのも利点の一つです．

今回は，可視化の為のmatplotlibと数値計算用のnumpyというライブラリを用いましたが，様々なライブラリが用意されており，用途に応じて活用できます．

一方，大きな弱点は，ご存知の通り実行速度があまり早くないことです．しかし，現在，pythonの実装にはいくつかあって，pypyな どのより高速な実装も開発されているので，かつてのJavaがそうだったように，（他の言語と比較して）次第に気にならない程度になってくるかもしれません．

とは言うものの，すべてのプログラミングをpythonで行う必要はなくて，例えば計算パワーが必要な本格的な計算は（pythonよりは高速な） JavaやCで，データの処理はpython+matplotlibなど，適宜使い分けるのは適切だと思います．皆さんの多くは，研究を進めていくにあたっていろんなデータ処理を行うことになるかと思いますが，今回紹介したpython+matplotlibが，みなさんの研究でプロトタイピングやデータ処理，可視化の役 に立てば幸いです．

## 評価

当初述べたとおり，成績評価はレポート提出と半ばで行ってもらった発表により評価します．