GH_CPython と OpenSeesPy を用いた Grasshopper 上における オープンソースな構造形態創生環境の構築

Construction of open source computational morphogenesis environment on Grasshopper using GH_CPython and OpenSeesPy

藤田　慎之輔*1
Shinnosuke Fujita*1

*1 北九州市立大学大学院 国際環境工学研究科 講師 工博
Lecturer, Graduate School of Environmental Engineering, The University of Kitakyushu, Dr. Eng.

キーワード：GH_CPython　OpenSeesPy　Grasshopper
Keywords：GH_CPython　OpenSeesPy　Grasshopper

1. はじめに

構造形態創生と呼ばれる手法は，21 世紀に入って隆盛を迎え，最適化技術を基盤としてアカデミックの世界で様々な手法が提案され，実務にも応用されてきた．しかしながら，2000 年代初頭から構造デザインの世界で用いられてきた形態創生プログラムは，そのほとんどが Fortran や C++ といったいわゆる重量言語で開発されている．科学技術計算の知識に乏しい初学者にとってそれらの重量言語は習得に時間を要するため，利用するにあたって敷居が高く，カスタマイズが難しいために，実務で広く普及したとは言い難かった．しかし近年，Rhinoceros のプラグインとして，建築の設計時に考慮するパラメータをリアルタイムでコントロールし，効率的に形態創生を行うことのできるツールとして Grasshopper[1] が登場し，先行して欧米で盛んに用いられるようになった。Grasshopper は初学者でも扱いやすいヴィジュアルプログラミングツールであるため，ここ 10 年ほどで日本にも急速に普及した. 意匠・構造の垣根を越えてそれらのツールによってパラメトリックに形態決定を行う設計手法は総じてコンピュテーショナルデザインなどと呼ばれ，設計の一手法としての地位を確立しつつある．

Grasshopper 上で構造形態創生を行う場合，galapagos[2] や octopus[3] などの発見的最適化ツールを最適化計算に用いて，構造解析には Karamba3D[4] を用いるのが一般的である. 知識の浅いものでも見様見真似で利用することができる大変便利なツールである一方で，galapagos や octopus は中身がブラックボックスでありカスタマイズが難しいことや，Karamba3D では扱える有限要素の種類が少なかったり，一般の構造解析専用ソフトと比べると機能不足の感が否めない.

一方で，フリーのプログラミング言語である Python は，強力な最適化ライブラリ群を有するほか，機械学習などの AI（人工知能）分野で最も使われている言語のひとつであり，非常に分かりやすい言語ルールと高い汎用性を持つため，初学者でも容易に扱うことができる利点がある. 近年では，カリフォルニア大学バークレー校がオープンソースとして公開している有限要素解析フレームワークである OpenSees の Python インタプリタである OpenSeesPy[5] が登場し，まだ開発段階ではあるものの，有限要素解析も自作プログラムを組むことなく手軽に利用可能な環境が構築されつつある．

OpenSeesPy と Python の情報技術ライブラリ群を組み合わせれば極めて高度なコンピュテーショナルデザインが可能であるが，インターフェースが未整備であるため実務に殆ど普及していない．

しかし，まだ OpenSeesPy と同様に開発段階ではあるものの，2017 年に GH_CPython[6] が登場し，grasshopper 上で Python の情報技術ライブラリ群を直接動かすことが可能となった.

本報告では，GH_CPython を用いて grasshopper 上で直接 Python プログラムをコーディングし，有限要素解析を OpenSeesPy により行い，Python の最適化ライブラリ群を利用して構造形態創生を行う一連の手順の紹介と，実際の適用事例について概説する．

2. Python の情報技術ライブラリ群の紹介

Python には，科学技術計算に関する膨大なライブラリが存在し，現在も世界中の技術者により開発が続けられ，その数は増え続けている．その中でも，建築構造物

日本建築学会情報システム技術委員会
第42回情報・システム・利用・技術シンポジウム論文集，110-113，2019年12月，東京
Proceedings of the 42nd Symposium on Computer Technology of Information, Systems and Applications，AIJ，110-113，Dec.，2019，Tokyo

の形態創生を行うに際し有用となるであろう最適化手法ならびに機械学習に関するライブラリとしては，以下のものが挙げられる．

表 1　情報技術に関わる代表的な Python ライブラリ

手法の分類		主なライブラリ
数理計画法	滑降シンプレックス法	SciPy[7]
	Powell 法	SciPy
	共役勾配法	SciPy
	準ニュートン法	SciPy
	Newton-CG 法	SciPy
	記憶制限準ニュートン法	SciPy
	COBYLA 法	SciPy,pyOpt[8]
	逐次最小二乗法	SciPy,pyOpt
	逐次二次計画法	pyOpt
	拡張ラグランジュ関数法	pyOpt
	CONMIN 法	pyOpt
	KS 汎関数法	pyOpt
	フィルタ信頼領域法	pyOpt
	ペナルティ法	pyOpt
	改良型 Dhor's r-algorithm	pyOpt
	線形計画問題一般	PuLP[9] ,PICOS[10]
	二次計画問題一般	PICOS
	二次制約二次計画問題一般	PICOS
	二次錐計画問題一般	PICOS
	半正定値計画問題一般	PICOS
	幾何計画問題一般	PICOS
	混合整数計画問題一般	PICOS
	混合整数二次錐計画問題一般	PICOS
発見的最適化手法	粒子群最適化法	pyOpt,PyGMO
	ハーモニーサーチ	pyOpt,PyGMO
	遺伝的アルゴリズム	pyOpt,PyGMO[11] ,DEAP[12]
	蟻/蜂コロニー最適化	pyOpt,PyGMO
	力まかせ探索	SciPy
	ベイズンホッピング	SciPy,PyGMO
	差分進化	SciPy,PyGMO
	擬似焼きなまし法	SciPy,PyGMO
	モンテカルロ法	PyGMO
	共分散行列適応進化戦略	PyGMO
機械学習	scikit-learn[13] ,TensorFlow[14] ,Chainer[15] ,Theano[16]	

上記ライブラリはすべてオープンソースで公開されている．ここに紹介したライブラリ以外にも数多くのライブラリが存在し，情報技術に関わる様々なアルゴリズムが網羅されている．

また，Python には wxPython や PyQt，Tkinter といった強力な GUI ライブラリが存在するため，独自のグラフィックユーザーインターフェースを手軽に作成することができる．

3. 構造形態創生環境構築の流れ

本節では，GH_CPython と OpenSeesPy を用いた構造形態創生環境構築の流れを説明する．以下では，Windows10OS の PC に Rhinoceros6.0 および Anaconda Python 3.7 version があらかじめインストールされているものとする．

3.1. GH_CPython

GitHub もしくは Food4Rhino より GH_CPython.zip がダウンロードできるので，展開して GH_CPython.gha と FastColoredTextBox.dll をコンポーネントフォルダへ保存する．このとき，通常であれば C ドライブに GH_CPython というフォルダが生成される．同フォルダ内に Grasshopper.py というモジュールがあり，python のリストを Rhinoceros 上における点，線，面データに吐き出すことができるが，既知のエラーが存在することや，リストのみしか扱うことができず numpy の array 配列を扱えないなど使いにくい面がある．点や線の情報を Rhino 上に出力するだけであれば，同フォルダに以下の簡単なソースコードを適当な名前を付けて保存すれば，Python 側で作成した座標ベクトル群を Rhinoceros の点データや線データに変換する関数として呼び出すことができる．

```
def addPoint(r):
    x,y,z=str(r[0]),str(r[1]),str(r[2])
    return 'gCPy.Point('+x+','+y+','+z+')'

def addLine(r1,r2):
    x1,y1,z1=str(r1[0]),str(r1[1]),str(r1[2])
    x2,y2,z2=str(r2[0]),str(r2[1]),str(r2[2])
    return 'gCPy.Line('+x1+','+y1+','+z1+','+x2+',
    '+y2+','+z2+')'
```

例えば，上記のスクリプトを gh.py として保存し，以下のコードを GH_CPython コンポーネントとして作成すれば，スパンやライズ，メッシュの分割数をパラメトリックにコントロール可能な裁断球殻ラチスドームを描画することができる (図 1)．引数は c(下端の中心の座標), divx(X 方向分割数), divy(Y 方向分割数), spanx(X 方向スパン), spany(Y 方向スパン), h(ライズ) となっている．

```
import numpy as np
import gh

pitchx,pitchy=spanx/divx,spany/divy
R=spanx**2/8./h+spany**2/8./h+h/2.
r,ij=[],[]
for i in range(divx+1):
    for j in range(divy+1):
        x,y=pitchx*i-spanx/2.,pitchy*j-spany/2.
        z=np.sqrt(R**2-x**2-y**2)-(R-h)
        r.append([x+c[0],y+c[1],z+c[2]])
n,r=len(r),np.array(r)
for i in range(divx+1):
    for j in range(divy):
        ij.append([i*(divy+1)+j,i*(divy+1)+j+1])
for i in range(divy+1):
    for j in range(divx):
        ij.append([i+j*(divy+1),i+(j+1)*(divy+1)])
m,ij=len(ij),np.array(ij,dtype=np.int64)
plot_beam=[]
for e in range(m):
    i,j=ij[e,0],ij[e,1]
    plot_beam.append(gh.addLine(r[i],r[j]))
```

GH_CPython ではカラーコンタを描画するなどの高度なグラフィック操作を行うプラットフォームはまだ実装

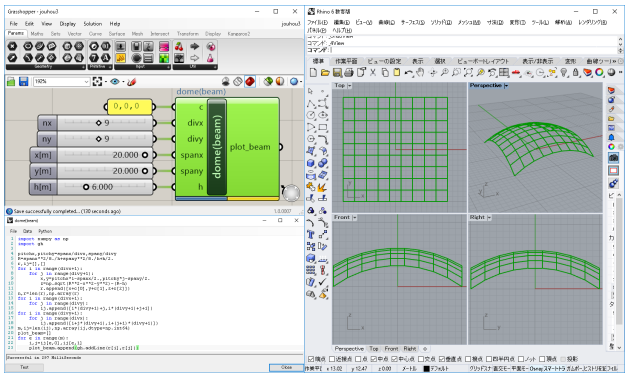


図 1　GH_CPython を利用したモデリング

されていない．Python ⇔ Grasshopper 間ではデータのやりとりさえできれば十分であるので，より高度な可視化表現が必要な場合にはデータの可視化は GH_CPython 側で行わずに Grasshopper の内部コンポーネントや，Rhino のグラフィックコントロールに強い C#や VisualBasic などで行うと良い．

3.2. OpenSeesPy

OpenSeesPy は pip を使って簡単にインストールできる．コマンドラインで `pip install openseespy` とすればよい．GitHub 上に Documentation がまとめられているが，OpenSeesPy もまだ開発途上の新しいライブラリであるため，解説が不十分でサンプルコード数も少なく初学者にはやや扱いづらい．日本の建築の構造設計を行う上では，梁要素とせいぜい板要素さえ使えれば十分であり，他の有限要素を利用する機会は殆どない．そこでここでは梁要素と板要素で構成される図のような構造物の弾性解析を行うソースコードを紹介する．通常 Python では index は 0 から数えるが，OpenSeesPy では，節点や要素，断面，材料番号の index が 1 から始まる点に注意する．また，単位は全て kN,m で揃えている．OpenSees では梁要素におけるコードアングルを角度で入力するのではなく，部材座標軸の方向ベクトルで入力する点が一般的な FEM ソフトウェアと異なる．なお，同ベクトルは単位ベクトルである必要はない．

```
import openseespy.opensees as ops
ops.wipe()
ops.model('BasicBuilder', '-ndm', 3)
n,m1,m2=8,8,1#節点数, 梁要素数, 板要素数
r=[[0.,0.,0.],[5.,0.,0.],[5.,5.,0.],[0.,5.,0.],
[0.,0.,5.],[5.,0.,5.],[5.,5.,5.],[0.,5.,5.]]#節点座標
for i in range(n):
    ops.node(i+1,r[i][0],r[i][1],r[i][2])#節点入力
ij=[[1,5],[2,6],[3,7],[4,8],[5,6],[6,7],[7,8],[8,5]]#梁要素節点関係
matNo=[1,1,1,1,2,2,2,2]#梁要素の断面材料番号
A=[1.207e-2,8.337e-3]#断面積
E=[2.05e+8,2.05e+8]#ヤング係数
G=[7.885e+7,7.885e+7]#せん断弾性係数
J=[3.569e-4,3.59e-7]#ねじり定数
Iy=[2.32e-4,2.35e-4]#強軸断面二次モーメント
Iz=[2.32e-4,1.74e-5]#弱軸断面二次モーメント
v=[[0.,1.,0.],[0.,1.,0.],[0.,1.,0.],[0.,1.,0.],
[0.,0.,1.],[0.,0.,1.],[0.,0.,1.],[0.,0.,1.]]#部材座標軸
ijkl=[[5,6,7,8]]#板要素節点関係
matNo2=[1]#板要素の断面材料番号
E2=[2.1e+7]#ヤング係数
poi=[0.2]#ポアソン比
t=[0.15]#板厚
fix=[[1,1,1,1,1,1,1],[2,1,1,1,1,1,1],[3,1,1,1,1,1,1],
[4,1,1,1,1,1,1]]#境界条件
p=[[5,10.,0.,0.,0.,0.,0.],[8,10.,0.,0.,0.,0.,0.]]#外力
for e in range(m1):#部材座標軸と梁要素の材料断面入力
    i,j,m=ij[e][0],ij[e][1],matNo[e]-1
    ops.geomTransf('Linear',e+1,v[e][0],v[e][1],v[e][2])
    ops.element('elasticBeamColumn',e+1,i,j,A[m],E[m],
    G[m],J[m],Iy[m],Iz[m],e+1)
for i in range(m2):#板要素の材料断面入力
    ops.section('ElasticMembranePlateSection',i+1,E[i],
    poi[i],t[i],0.)
for e2 in range(m2):#板要素
    [n1,n2,n3,n4],m=ijkl[e2][:],matNo2[e2]
    ops.element('shellMITC4',m1+e2+1,n1,n2,n3,n4,m)
for f in fix:#境界条件入力
    ops.fix(f[0],f[1],f[2],f[3],f[4],f[5],f[6])
ops.timeSeries('Constant',1)
ops.pattern('Plain', 1, 1, '-fact', 1.0)
for f in p:#外力入力
    ops.load(f[0],f[1],f[2],f[3],f[4],f[5],f[6])
ops.system('BandSPD'),ops.numberer('RCM')
ops.constraints('Plain'),ops.algorithm('Linear')
ops.integrator('LoadControl', 1.0)
ops.analysis('Static')
ops.analyze(1)
disp,reaction=[],[]
for i in range(n):
    disp.append(ops.nodeDisp(i+1))
```

図 2 および表 2 の解析モデルを入力した上記のソースコードを実行すれば，弾性変位を得ることができる．ソースコードには含めていないが，断面力や応力，反力も必要に応じて得ることができる．

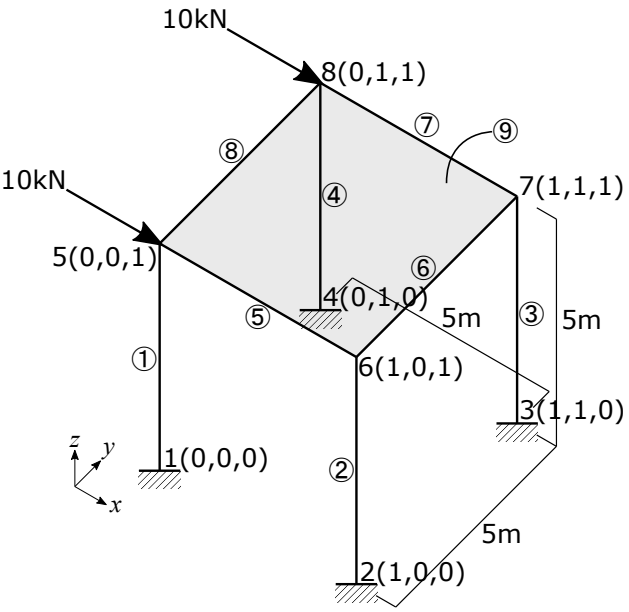


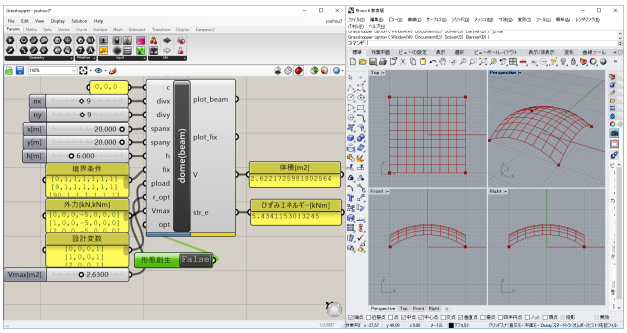
図 2　OpenSees で解析する例題骨組

表 2　例題骨組の部材断面情報

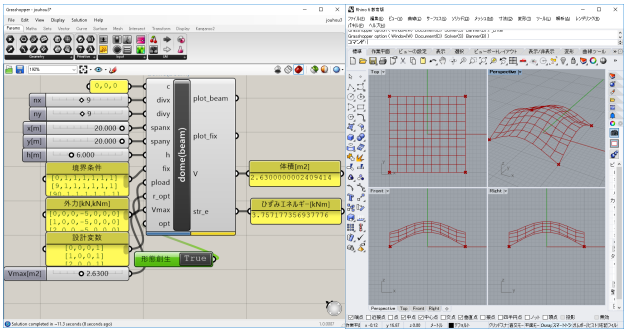
要素番号	要素種別	断面・板厚	ヤング率	ポアソン比
①	梁要素	□-350x9	205GPa	0.3
②	梁要素	□-350x9	205GPa	0.3
③	梁要素	□-350x9	205GPa	0.3
④	梁要素	□-350x9	205GPa	0.3
⑤	梁要素	H-400x200x8x13	205GPa	0.3
⑥	梁要素	H-400x200x8x13	205GPa	0.3
⑦	梁要素	H-400x200x8x13	205GPa	0.3
⑧	梁要素	H-400x200x8x13	205GPa	0.3
⑨	板要素	t=150mm	21GPa	0.2

3.3. Python ライブラリを用いた GH 上での形態創生

GH_CPython 通して OpenSeesPy を用いれば grasshopper 上で様々な構造解析を行うことができる．あとは，Python の豊富な情報技術ライブラリを用いれば，grasshopper 上で構造形態創生を手軽に行うことができる．1 例として，図 2 で作成したラチスドームについて，部材断面をすべて H-400x200x8x13(図 1 の例において⑤〜⑧に指定した断面)，4 隅を固定支持として，各節点に鉛直下向きに 5kN の節点外力を受けた状態で，支持点を除く節点の鉛直方向座標を設計変数，体積の上限値を 2.63m^2 としてひずみエネルギーを最小化するスクリプトを組み，実行した結果を図 3 に示す．最適化ライブラリには表 1 における pyOpt を用い，逐次二次計画法により最適化計算を行った．支持点間のライズがカテナリー状に上昇する構造最適化の世界ではおなじみの形状を grasshopper 上で得ることができた．



(a) 最適化前



(b) 最適化後

図 3　Python ライブラリを用いた GH 上での形態創生例

4. まとめ

GH_CPython と OpenSeesPy を用いて，Galapagos や Karamba に頼らない構造形態創生事例を紹介した．OpenSeesPy も含めて，Python のライブラリはそのほとんどがオープンソースで公開されているので，コンパイル済で中身がブラックボックス化してしまう既存のコンポーネントを使用するよりもプログラムの中身を熟知した構造形態創生環境の構築が可能である．ただし，GH_CPython はまだ発展途上のコンポーネントであり，形態創生の様子をリアルタイム描画したり，高度なヴィジュアライゼーションには対応していないなど，機能面で不十分な点も存在する．その他には，gh-python-remote[17] を用いれば表 1 のライブラリ群が GH_CPython と同様にダイレクトに利用可能となるが，Python2 系でしか使用できない欠点があるため，現状最も有効な選択肢となりうるのはやはり GH_CPython であろう．gh-python-remote が近い将来 Python3 に対応するか，あるいは GH_CPython の機能がより充実するかすれば，より手軽に Python の情報技術ライブラリ群を grasshopper 上で利用できるようになると思われる．

[参考文献]

1）D. Rutten. Grasshopper -computing architectural concepts-. *the conference Advances in Architectural Geometry, Vienna, Austria*, pp. 18–21, 2010.9.

2）D. Rutten. Galapagos: On the logic and limitations of generic solvers. *Architectural Design*, Vol. 83, No. 2, pp. 132–135, 2013.3.

3）R. Vierlinger. Multi objective design interface. *Master Thesis of Technical University Vienna*, 2013.3.

4）C. Preisinger and M. Heimrath. Karamba -A toolkit for parametric structural design. *Structural Engineering International*, Vol. 24, No. 2, pp. 217–221, 2013.11.

5）M. Zhu, F. McKenna, and M. Scott. Openseespy: Python library for the opensees finite element framework. *SoftwareX*, Vol. 7, pp. 6–11, 2018.1.

6）Mahmoud AbdelRahman. Mahmoudabdelrahman/gh_cpython: Alpha release of gh_cpython plugin, `https://doi.org/10.5281/zenodo.888148`. 2017.9.

7）T. E. Oliphant. *Guide to NumPy*. Createspace Independent Publishing Platform, 2015.

8）R. E. Perez, P. W. Jansen, and J. R. R. A. Martins. pyopt: a python-based object-oriented framework for nonlinear constrained optimization. *Structural and Multidisciplinary Optimization*, Vol. 45, No. 1, pp. 101–118, 2012.1.

9）S. Mitchell, S. M. Consulting, and I. Dunning. Pulp: A linear programming toolkit for python. *The University of Auckland, Auckland, New Zealand*, 2011.9.

10）G. Sagnol. Picos documentation. a python interface to conic optimization solvers. release 0.1.1. *Zuse Institute Berlin (ZIB)*, 2012.1.

11）D. Izzo. Pygmo and pykep: Open source tools for massively parallel optimization in astrodynamics (the case of interplanetary trajectory optimization). *Proceedings of the International Conference on Astrodynamics Tools and Techniques - ICATT*, 2012.1.

12）F. A. Fortin et al. Deap: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, Vol. 13, pp. 2171–2175, 2012.7.

13）F. Pedregosa et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, Vol. 12, pp. 2825–2830, 2011.10.

14）M. D. Golub and D. Sussillo. Fixedpointfinder: A tensorflow toolbox for identifying and characterizing fixed points in recurrent neural networks. *The Journal of Open Source Software*, Vol. 3, No. 31, 2018.11. https://doi.org/10.21105/joss.01003.

15）S. Tokui, K. Oono, S. Hido, and J. Clayton. Chainer: a next-generation open source framework for deep learning. *Proceedings of workshop on machine learning systems (LearningSys) in the twenty-ninth annual conference on neural information processing systems (NIPS)*, Vol. 5, , 2015.12.

16）J. Bergstra et al. Theano: Deep learning on gpus with python. *Journal of Machine Learning Research*, Vol. 1, pp. 1–48, 2011.10.

17）GH Python Remote: `https://www.food4rhino.com/app/gh-python-remote`. Accessed 2019.8.