Some ideas on snake swimming with the help of FreeFEM

F. Hecht

LJLL, Sorbonne Universite, Paris projet Alpines, Inria de Paris

with E. Trelat, G. Lance

12 oct. 2021.

http://www.freefem.org mailto:frederic.hecht@upmc.fr

Outline

Outline

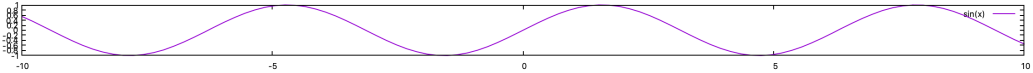
My goal is to understand and optimize the swimming of an object, of a living being. It is still how to transform a vertical movement into a horizontal one.

Movie:snake swimming

Moreover it is necessary that the model is simple: if possible 2D and stationary.

Outline

We will suppose that the swimming speed is constant in time, and to simplify simplify the snake will be 2D (???) infinite and defined by a line that moves periodically with a constant speed V.



That is, the function giving the position of the snake over time will be given by

$(x, y = f(x + Vt))$ where f is the snake shape function (periodical of period L which we will suppose null at the ends $f(0) = f(L) = 0$) and where V is the translation speed of the snake shape.
Moreover we will only look at the upper part for symmetry reasons.
So, in the snake frame of reference: Run:movement-snake.edp and in the reference frame in translation with a speed of $(V, 0)$, there is no more motion.
The model this all contrains.

In summary, we have 3 reference frames:

- $\mathcal{R}^o$ the frame of the snake which links it to the object ,
- $\mathcal{R}^c$ the frame of computational which moves with a blue speed$(V, 0) = V^c$ relative to the blue frame of reference $\mathcal{R}^o$ (ie. $(x^c, y^c) = (x^o + Vt, y^o)$),
- $\mathcal{R}^e$ the frame of the experiment (we see the snake moving) which moves at an unknown speed relative to the reference frame $\mathcal{R}^o$.

And in the reference frame $\mathcal{R}^o$ the shape of the snake is given by $(x^o, y^o = f(x^o + Vt))$ and so the speed of deformation is $(0, y = Vf'(x^o + Vt)) = (0, y = Vf'(x^c))$, and moreover we can notice that $u^o = u^c - V^c$.

Remark: We could do a better because the velocity is clearly not in $(0, v_1)$ in the snake's frame of reference, the snake is inextensible so we can calculate a more reasonable speed.

The Navier-Stokes equations are stable by change of Galilean frame, we will place ourselves in the computational reference frame $\mathcal{R}^c$.

Our problem is to find $(\boldsymbol{u}, p)$ in $\Omega =]0, L[\times]f(x), H[$ where f is the periodic function of period L defining the shape of the snake, H the top of the computational box and such that

$$(\boldsymbol{u} \cdot \nabla)\boldsymbol{u} - \nu\Delta\boldsymbol{u} + \nabla p = 0, \quad \nabla \cdot u = 0$$

$\boldsymbol{u}$ is periodic on the vertical edges, we will put a zero force condition on the upper edge $\boldsymbol{u}$ (Γ_∞), and the velocity imposed by the motion on the lower edge (Γ_s).
The velocity of this edge in the snake frame of reference is therefore :
$\boldsymbol{V}_s^o = (0, Vf'(x + Vt)$ In the computational reference frame the boundary condition is $\boldsymbol{V}_s^c = (-V, Vf'(x))$.
Finally, the velocity of the snake will be given by minus the velocity at infinity (on the upper edge) in the snake reference frame, to calculate the velocity in the reference frame , we solve:

$$(\boldsymbol{u}^o \cdot \nabla)(\boldsymbol{u}) - \nu\Delta\boldsymbol{u}^o + \nabla p = 0, \quad \nabla \cdot \boldsymbol{u}^o = 0$$

The space for the velocities $\boldsymbol{V} = H^1_{per}(\Omega)^2$ where $H^1_{per}(\Omega)$ is the space H^1 with periodicity in x (i.e. $u(x) == u(x+L)$)

$$\boldsymbol{P} = p \in L^2(\Omega)$$

and the space with homogeneous boundary conditions:

$$\boldsymbol{V}_0 = \{\boldsymbol{v} \in \boldsymbol{V} / \boldsymbol{v} = 0 \text{ on } \Gamma_s\}$$

.

The variational formulation is then to find $(\boldsymbol{u}, p) \in \boldsymbol{V} \times \boldsymbol{P}$ such that $\boldsymbol{u}_{|\Gamma_s} = \boldsymbol{V}_s^c$

$$\forall \boldsymbol{v} \in V_0,\ \forall q \in \boldsymbol{P} \int_\Omega (\boldsymbol{u}.\nabla \boldsymbol{u}).\boldsymbol{v} + \nu \nabla \boldsymbol{u} : \nabla \boldsymbol{v} - p \nabla . v - q \nabla . u = 0$$

Remark : in this case, the pressure is not defined to a constant due to the boundary condition on Γ_∞ .

Discret version

To begin with we will solve a linear version (Stokes) neglecting the term $(\boldsymbol{u}.\nabla\boldsymbol{u})$, but shifted by one velocity Vr to compute the velocity $\boldsymbol{u}^c$ where $\boldsymbol{u}^o$ in both reference frames.

```
real Vr = 0;
mesh Th = square(nx,ny,[x/nper,y],flags=0);
Th = movemesh(Th,[x,-fy+y*(Htop+fy)]);

fespace Wh(Th,[P2,P2,P1],periodic=[[2,y],[4,y]]);

solve Stokes2( [u1,u2,p],[v1,v2,q] ) =
  int2d(Th)( UgradV(-vR,0,u1,u2)'*[v1,v2]+mu*(Grad(u1,u2):
     Grad(v1,v2))  - div(u1,u2)*q -div(v1,v2)*p)
  + on(3,u2=0)
  + on(1,u1=vdent-vR,u2=Uy)
;
```

Run:Stokes-Serpent.edp

The optimisation problem

The goal is to find the best snake shape that maximizes $J(f)$ the velocity generated at infinity so $J(f) = \int_{\Gamma_\infty} u_1$.
A few remarks before doing anything:

- I feel like a sawtooth shape would be better? and $J(f)$ would be zero if f is antisymmetric? (same mistake)
- To have a constant volume we will impose $\int_0^L f = 0$,
- $J(f)$ grows with $||f|_\infty$, so we have to put constraints, the snake clearly has a constraint on the curvature, but to simplify we will just put a constraint on the derivative at first.
- Finally, we should evaluate the energy spent by the snake to move and find the form at given energy given.

We will use "IPOPT" to solve this problem, so we have to calculate the gradient with respect to f.

Outline

```
mesh Th0 = square(NX,NX*H,[x,y*H]);
meshL ThL = change(extract(Th0,refedge=ll), flabel = ...);
fespace WhL(ThL,P1),  PhL(ThL,P0);
WhL X0=0.15*sin(x*pi*2); // Donnée initial ...
WhL fm,cgm; // discrétisation de la donnée, de ça derive !!!
func real J(real[int] &X)
{
    fm[] = X;
    cgm[]= M11^-1*(bgm=M1dx*fm[]); // proj L2 de dx ..
    Th = movemesh(Th0,[x,fm+y*(H-fm)/H]);
    solve Stokes( [u1,u2,p],[v1,v2,q] ) =
     int2d(Th)(2*mu*(SGrad(u1,u2):SGrad(v1,v2))
              - div(u1,u2)*q - div(v1,v2)*p - eps*p*q)
     + on(1,u1=0,u2=vdent*cgm) ;
    real cost = -int1d(Th,3)(u1) ;
    plot(Th,[u1,u2],cmm="␣J=␣"+cost +"␣" +iter0);
    return cost;
}
```

A Remark

The control $f \in H^2(0,1) \cap H^1(0,1)$ is discretized with `WhL` one-dimensional $\mathbb{P}_1$ finite elements with the command `fm`. Hence, its numerical derivative **`dx`**`(fm)` is discretized with $\mathbb{P}_0$ finite-elements, i.e., is piecewise constant. However, the boundary condition **`on`**`(1,u1=0,u2=vd*gm)` implies that `u2`, which is continuous, we do a L^2 projection on `WhL`.

```
cgm[]= M11^-1*(bgm=M1dx*fm[]); // proj L2 de dx ..
```

The Derivative

```
func real[int] dJ(real[int] &X) {
real[int] dJ(WhL.ndof);
fm[] = X; func gm = dx(fm);  cgm[]= M11^-1*(bgm=M1dx*fm[]); // proj L2 de dx ..
Th = movemesh(Th0,[x,fm+y*(H-fm)/H]);  ThC = movemesh(ThL,[x,fm+y*(H-fm)/H]);

solve Stokes( [u1,u2,p],[v1,v2,q] ) =
int2d(Th)(2*mu*(SGrad(u1,u2):SGrad(v1,v2)) - div(u1,u2)*q - div(v1,v2)*p )
+ on(1,u1=0,u2=vdent*cgm) ;

solve StokesAdjoint( [v1,v2,q],[w1,w2,g] ) = //adjoint problem
int2d(Th)(2*mu*(SGrad(v1,v2):SGrad(w1,w2))
- div(w1,w2)*q - div(v1,v2)*g )
- int1d(Th,3)(w1)+ on(1,v1=0,v2=0);

func uk = vdent*(q - 2*mu*dy(v2) + mu*gm*( dx(v2)+dy(v1) ));
func vk = - gm*( -q*dy(u1) + 2*mu*dx(v1)*dy(u1) + mu*dy(u2)*( dx(v2) + dy(v1) ))
   + ( mu*dy(u1)*( dx(v2)+dy(v1) ) + 2*mu*dy(v2)*dy(u2) - q*dy(u2) ) ;

varf vMassC(u,v) = int1d(ThC,qforder=3)(u*v);
varf vb1(u,v) = int1d(ThC)(uk*v);   varf vb2(u,v) = int1d(ThC)(vk*v);
matrix Mc = vMassC(WhC,WhC);
real[int] bc(WhC.ndof);
uL[] = Mc^-1*(bc=vb1(0,WhC));  vL[] = Mc^-1*(bc=vb2(0,WhC));
varf vargrad(uu,vv) = int1d(ThL)(vL*vv + uL*dx(vv));
dJ = vargrad(0,WhL);   return dJ;}
```

The contraint

```
func real[int] C(real[int] &X)
{ real[int] cont(1+PhL.ndof);
    fm[] = X;    // contraintes volumes ∫_0^a f(x)^2 dx < M
    cont[0] = int1d(ThL)(fm);    // |f'(x)| ≤ M
    cont(1:PhL.ndof) = mDx*X;
    return cont;
}

matrix dc;
func matrix jacC(real[int] &X)
{
    real[int,int] dcc(1,WhL.ndof); dcc = 0.0;
    dcc(0,:) = aCvol;
    dc = dcc;
    dc = [[dcc],[mDx]]; // [contrainte volume,contrainte perim]
    return dc;
}
```

Run:opt-Stokes-HLT.edp

Conclusion

Merci,

Gontran Lance soutient le 26 novembre 2021 au LJLL à 14h00, couriel: gontranlance@gmail.com pour plus détail et Info.

Les Journées FreeFEM et MMG auront lieu les 8, 9 et 10 décembre

http://freefem.org