Snailgate: Summary of Simulation

June 28, 2017

1 Physics Problem

We are trying to solve the following problem:

$$m\ddot{u} = R(u), \text{ for each node}$$

, where m is the mass, $u(t)$ represents position of the node through time and $R(u)$ is the resulting force on the node. Notice that the force on the node $R(u)$ is affected by the position of all other nodes in the system and also from gravity, water level and others.

Since the mass is constant, we could consider it equal to 1 and work with the following formula:

$$\ddot{u} = R(u)$$

But this is a second order ODE. To make it easier to solve, we first transform it in a first order ODE by declaring velocity v as $v(t) = \dot{u}(t)$.

Then, we have the following system for each node:

$$\begin{cases} \dot{u} = v \\ \dot{v} = R(u) \end{cases} \text{, for each node}$$

Now, we have a first order system of ODEs. Notice that this two equations repeat for all nodes in our simulation, so the size of the system depends on the discretization that we use. A fine discretization creates a larger (and probably more difficult) system to solve.

Since we are dealing with a 2 dimensional problem, we need to represent values on x and y directions. A way of doing this is considering $u = (u_x, u_y)$ and $v = (v_x, v_y)$ and using this variables in the system. So, for each node, we end up with four variables.

Consider now that we have a large vector containing all u and v variables for all nodes. We

have then:

$$U = \begin{bmatrix} \vdots \\ u_x^{(i)} \\ u_y^{(i)} \\ v_x^{(i)} \\ v_y^{(i)} \\ u_x^{(i+1)} \\ u_y^{(i+1)} \\ v_x^{(i+1)} \\ v_y^{(i+1)} \\ \vdots \end{bmatrix} \text{ and } F = \begin{bmatrix} \vdots \\ v_x^{(i)} \\ v_y^{(i)} \\ R_x^{(i)} \\ R_y^{(i)} \\ v_x^{(i+1)} \\ v_y^{(i+1)} \\ R_x^{(i+1)} \\ R_y^{(i+1)} \\ \vdots \end{bmatrix}$$

Finally, we can represent our first order ODE system as:

$$U'(t) = F(U)$$

2 Resolution methods

With this formulation, it is easy to write Forward and Backward methods for this problems.

2.1 Forward Euler (Explicit method)

In the case of Forward Euler, we simply discretize and approximate the derivative on the left side of equation $U'(t) = F(U)$:

$$\begin{aligned} \frac{U(t+k) - U(t)}{k} &= F(U(t)) \\ \frac{U^{n+1} - Un}{k} &= F(U^n) \\ U^{n+1} &= U^n + kF(U^n) \end{aligned}$$

Notice that, when solving for time $t + k$ (which corresponds to iteration $n + 1$) we can solve U explicitly by simply using the value of U^n and applying F on it.

2.2 Backward Euler (Implicit method)

In the case of Backward Euler, we aim to use F applied on the future iteration. Then, it is not possible to solve $U^{n+1} = U(t + k)$ explicitly and we have to solve the system below:

$$\begin{aligned} \frac{U(t+k) - U(t)}{k} &= F(U(t+k)) \\ \frac{U^{n+1} - Un}{k} &= F(U^{n+1}) \\ U^{n+1} &= U^n + kF(U^{n+1}) \end{aligned}$$

Notice that if F is linear, meaning that $F(U^{n+1}) = AU^{n+1}$ for some matrix A, we would have a linear system $(I - kA)U^{n+1} = U^n$.

However, since we are dealing with water pressure and elasticity, which do not give rise to linear forces, we have to solve a nonlinear system:

$$\begin{aligned} U^{n+1} - U^n - kF(U^{n+1}) &= 0 \\ G(U^{n+1}) &= 0 \end{aligned}$$

Thus, we are basically searching for values of U^{n+1} for which $G(U^{n+1})$ is 0. For solving this, we can make use of known literature techniques as the famous Newton method for nonlinear systems.