Poser Dynamics

Part V – Crash Course

Understanding cloth and having a mental model of the physics involved is necessary but not sufficient to make effective and efficient simulating systems. Creating those systems is a world in its own right. Especially those with an engineer's way of looking at things might be interested in this mini tour through the deep down dungeons. **Crash Course on Math, Physics and Sims** really is the math and physics loaded chapter. Any use of the underground escape tunnel brings you into Muppets Lab "where the future is made today". You have been warned.

June 2012

Contents

Crash Course on Math, Physics and Sims

Some people immediately zap away when they see even a two-character formula coming along, others insist in getting the rough engineering treatment to enhance their understanding. This chapter is for the latter part of the audience, although I can recommend the conclusions to everyone.

I'll start with some math first because 1) it simplifies handling the formulas and 2) in the forums people have stated to have some issues handling and grasping it. Second I'll do the physics, from an electrical point of view. This gives the proper resulting formulas in an easy way, but it's too far away from our actual simulations. So third, I'll present the physics from a mechanical point of view. Similar formulas but much closer to our actual cloth simulation. Then, fourth, I'll consider the simulation algorithms themselves. Finally, I'll present some conclusions that help you further in handling the Poser parameters and cloth simulation.

Complex Calculations

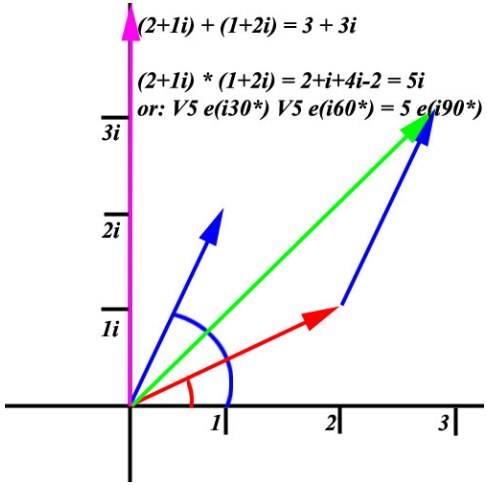
In order to handle straightforward object motion, steady currents and flows, I can simply deal with real numbers. But handling disruptions, oscillations, alternating current and more becomes a lot easier when I extent my repertoire to imaginary and complex numbers.

In that area, the magic bullet is called 'i' with the property $i^2 = -1$. I don't have to understand it, it's there, period.

Multiplying a real number b by i turns it into the imaginary i*b (or: ib for short), and repeating the operation turns it into $i*i*b = i^2b = -b$ which brings me back into the real world again. When I mix a real number a and an imaginary ib I get a complex number a+ib, and I can do all adding and subtraction while keeping the real and imaginary worlds separate: (a+ib) + (c+id) = (a+c) + i(b+d).

Like I can represent real numbers on a straight line, I can represent complex numbers in a simple plane. I can use dots or arrows to them, and adding and subtraction behave like simple adding and subtracting those arrows as in vector algebra.

Complex numbers however are not very friendly in multiplication and handling powers, exponentials and logarithms. As you can see in
$(a+ib)*(c+id) = ac + ibc + iad + i^2bd = (ac-bd) + i(bc+ad)$
real and imaginary parts of the numbers get mixed, which might turn longer calculations into a sheer mess.



The vector approach however gives us an alternative. Instead of considering the x and y components of the vector, we can consider its length R and angle q with the horizontal axis. This makes a+ib equivalent to R*[cos(q)+isin(q)]. Adding and subtracting from this approach becomes a nightmare, but multiplication, division and more becomes easy. Multiplication is: multiply the lengths and add the angles, which follows in a straightforward way from classic geometry (I'll save you the details).

To simplify our tool even more, I use natural exponents and logarithms to rewrite R to e^r (or r=ln(R)), and especially (cos(q) + isin(q)) to e^{iq}. In other words, a+ib becomes $e^{(r+iq)}$ where r=ln(sqrt(a^2+b^2)) and q= arctan(b/a). This is handy, since multiplication of values is equivalent to the addition of exponents, and taking powers is equivalent to multiplying exponents, so ($e^{(r+iq)}$ * $e^{(s+it)}$ => $e^{[(r+s)+i(q+t)]}$ and $[a+ib]^N$ => $[e^{(r+iq)}]^N$ => $e^{N(r+iq)}$.

So, with this dirty trick, the issues of complex multiplication and power-lifting are reduced to addition and simple multiplication. That's a gain.

This is the basic tool, now the application of it in physics. Alternating currents, oscillating movements and the like can be described as x(t) = Rcos(wt) where R is the amplitude, and w=2 pi f, for oscillating frequency f. Each time wt makes a multiple of 2 pi, or each time ft makes a whole number, the oscillation starts anew. Now it has become a small step to state that the oscillation we see is just the real world part of a complex movement, which also has an imaginary component in a world we can or cannot imagine ☺.
But as a whole, its described with $x(t) = Re^{iwt}$ or even with $x(t) = e^{(r+iwt)}$.

The main reason for doing this, is that we need time-derivatives (and integrals) like dx/dt to get speed, acceleration and more. But the classical approach presents us a plethora of sine and cosine functions, and we really need a lot of geometry to get understandable results. In the new approach, we don't.

When $x(t) = e^{(r+iwt)}$ then $dx/dt = iw * e^{(r+iwt)}$, so $d2x/dt2 = -w2 * e^{(r+iwt)}$, from position to velocity to acceleration,
but also $\int x dt = 1/iw\, e^{(r+iwt)}$ and so $\int\int x dt2 = -1/w2\, e^{(r+iwt)}$, from acceleration to velocity to position.

Going Electrical

In abstraction, basic physics considers a 'field' and a 'thing', and moving the thing against the field requires energy (from us or from the thing itself), while moving the thing along the field returns that energy back. One unit of field times one unit of thing make 1 Joule of energy, and if I do so every second it requires or generates 1 Joule per second or: 1 Watt of power.

Less abstract, in electricity the 'field' is electric potential (char: V) determined in Volt and the 'thing' is electrical charge (char: Q) determined in Coulomb. When such an amount of charge passes a measuring point within 1 second, we've got an electrical current of 1 Coulomb per second, or: 1 Ampere.
When I move such a charge to a place with a 1 Volt higher potential it requires 1 Joule (char: E, in formula: E=V*Q), and

when I do so every second it requires a power of 1 Watt (char W, in formula: W = V*I). Especially this last formula on electrical power might hang somewhere in your memory, resulting from a physics class or so.

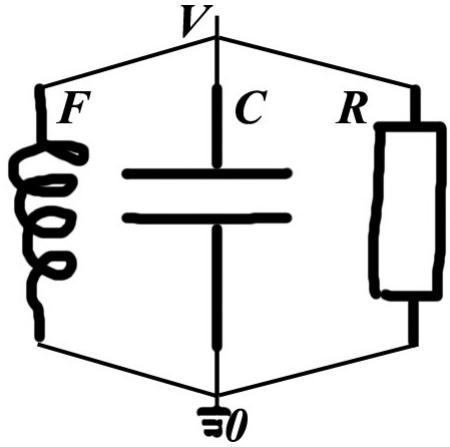
When I move a charge from one place to another, bridging a potential, it becomes harder for the next portion of charge to do the same. The source of the charges becomes short due to the displacement, or: anti-charged and as we know opposite charges pull each other. The destination of the charges gets a surplus or: gets charged, and similar charges push each other. In other words, by moving a charge across a field, it gets less interesting to continue, the field itself gets reduced. The ratio between these is understood as: electrical capacity (char C, measured in Farad).
In formula: dQ = C dV, when I move a charge dQ the field changes dV. When the device that includes the source and the destination of the charge is said to have a large capacity, this means that the displacement of large charges have little effect on the field.

Even when moving the charges hardly effects the field, the charge elements (e.g. electrons) have to bridge the gap between source and destination, collide to each other and to loads of molecules in between, and dissipate some heat. This is understood as electrical resistance (char R, measured in Ohm).
In formula: V = I*R or I=V/R, Ohm's law. A large resistance is like having air (or less) between two poles of a battery, they are electrically isolated, there is hardly any current and no energy loss. A small resistance is like short-circuiting the battery with a copper wire. Lots of current, and the heat might even melt the wire. As current I = dQ/dt we see that first we had a relationship between 'field' V and 'thing' Q, and now we've got a relationship between V and the first time-derivate of the 'thing': dQ/dt.

So the next step is to investigate what happens with that second time-derivative, or: dI/dt. This just means: alternating current. When alternating current is applied onto a coil, it generates an electric field. This is understood as electrical induction, the fundament under dynamos, generators, turbines and the like. Char: F (after Faraday), measured in Henry

(both did the discoveries at the same time).
In formula: V = F*dI/dt or dI/dt = V / F, a large induction generates a strong field from the alternating current.

Now, let me bring all the elements together. I've got a device with two poles, I apply an alternating voltage onto the poles, and the charges and currents in the device start working in parallel:



Working in parallel implies that I have to add the individual currents, so:
(I1+I2+I3) = I total = Voltage / "Impedance" (Impedance is some kind of resistance for alternating currents and voltages, taking capacity and induction into account).

- Alt. Voltage $V\exp^{\{iwt\}}$
- Capacity $I = dQ/dt = C * iw * V\exp^{\{iwt\}}$
- Resistance $I = 1/R * V\exp^{\{iwt\}}$
- Induction $I = 1/F * 1/iw * V\exp^{\{iwt\}}$
- In parallel $I = [\, C * iw + 1/R + 1/F*1/iw \,] * V\exp^{\{iwt\}} = 1/IMP * V\exp^{\{iwt\}}$
- So IMP $= 1/[\, iwC + 1/R + 1/iwF \,]$, after taking the inverse on both sides
  $= iwR / [-w2\,RC + iw + R/F]$, after multiplying with iwR on both sides
  $= iwR\,[\,(R/F-w2RC) - iw]/[\,(R/F-w2RC)2 + w2]$, after multiplying with $[\,(R/F-w2RC) - iw]$
  $= w * [w/R + i\,(1/F-w2C)] / [(1/F-w2C)2 + (w/R)2]$
- If R very small (short circuit), then 1/R becomes dominant and IMP => (w2/R) / (w2/R2) = R2/R = R

- If R very large (isolation), then IMP = iw / (1/F-w2C) and becomes quite large when 1/F-w2C => 0, that is: w2 => 1/FC. Then the impedance => R again.
  This w = sqrt(1/FC) represents the resonance frequency of the system, it behaves at a minimal energy loss.

Now we've got the essential formula, let's apply them on a mechanical system.

Going Mechanical

The mechanical route follows the same approach. The 'field' is force (char F, measured in Newton) and the 'thing' is displacement (say X, measured in meter). Applying a force of 1 N over 1 m requires 1 J of energy, and applying that force over a route with a speed of 1 m/s requires 1 W of power. Note the similarities with electrical.

- A spring pulls and pushes, proportional to its stretch: F = S * X. The stronger the spring, the more force is required to stretch it (with the same amount of displacement). Spring strength does not have a specific name, it's measured in N/m.
- Spring strength is not related to any mass, but when a mass is attached to a string, the force will generate an appropriate acceleration: F = M * a, Newtons Law, where a is $d^2 X/dt^2$, the second time derivative of the 'thing' we're considering. M is mass, in kilograms.
- So we've seen forces proportional to distance X (the spring) and to its second time-derivative acceleration A (gravity like). So all we need to make it like electrical, is a force that is proportional to the first time-derivative of distance, which is: speed, or velocity v. This introduces: damping D as in F = -D*v. The faster you move, the more the force will work against you. Air-resistance is one example of them.
- Now we can make the step from electrical to mechanical, just by substituting the equivalent concepts
  - So electrical capacity C relates to 1/S spring strength
  - So electrical resistance R relates to 1/D damping

    - So electrical induction F relates to M mass
- And so w = sqrt(S/M) represents the resonance frequency of the system. The resonance becomes noticeable when there is only little damping. Strong springs with little mass with make high frequencies, loosening the springs and/or raising the mass will reduce the resonance frequency. Damping will kill the oscillations and will produce a stable result faster.

RESULT: IMP = $w * [wD + i (1/M-w^2/S)] / [(1/M-w^2/S)^2 +(wD)2]$, times S^2M^2 at both sides:
= $wSM * [wDSM + i(S-w^2M)] / [(S-w^2M)^2 +(wDSM)^2]$
=> 1/D at the resonance frequency w = sqrt(S/M)

Mechanical systems and electrical systems do compare, but to a certain extent. Electrical components can be made quite ideal: capacitors with a good isolation and no leakage (extreme resistance), resistors with hardly capacity or induction by their own, inductors with metal kernels and so on. And those systems are pretty linear over a large range of values, from electron beams in a CRT monitor to outdoor flashes in a thunder storm, the same laws keep applying.

Mechanical components are far less ideal: springs do have mass and damping by themselves, and the stretching is far from linear. When the spring is made of an iron wire coil, there are upper and lower limits to its expansion or compression depending on the length and thickness of the wire used. So in fact the formula should read $F = S_0 X + S_1 X^3 + S_2 X^5 + \ldots$

Odd powers only, as all forces work in one direction: against the displacement.
The same holds for damping, as can be found in any aero/fluid-dynamics class or textbook. For low speeds it's proportional, but at higher speeds the additional terms kick in quickly: $F = D_0 X + D_1 X^3 + \ldots$

All this means that the math still can be done but that the final resulting formulas become longer and less friendly.

Conclusions

By looking at complex numbers in addition to normal, real numbers, I got tools for expressing fluctuations in a more elegant way. By applying them to electrical circuits with alternating current, I got the formulas for the "impedance" of a system responding to alternating inputs. By applying those formulas to a mechanical context, I got a descriptor for a mechanical system, responding to external forces.

I found that such a system is characterized by

- Mass (M), making it harder for a force to accelerate an object
- Damping (D), making it easier for a force to reduce an objects speed, and making the object lose energy faster
- Spring Strength (S), making it harder for a force to displace an object
- Resonance, the tendency to vibrate with a frequency sqrt(S/M). This is reduced by Damping.

Going Sim

Creating a computer simulation of a mechanical system requires the build of a ball-and-spring network. Like a mesh, the vertexes are small balls with carry all the mass, while the edges connecting the vertices are small (mass-less) springs which do the pushing and pulling. Balls and springs both take their share in damping, and the balls pick up the external forces and limitations like gravity, attached weights, air-damping, collision and friction to objects.

In order to make the simulation work like the real thing, various technical (constructive) aspects require attention. Building goods simulations is a serious profession. To name a few:

- **Resonance**
  mentioned already as a 'natural aspect' of electrical and mechanical circuits. What does it do?
  Say, the resonance frequency is: 100. This means that a vertex is oscillating around its proper place to be, at 100

times a second, real world time (as the values for Damping, Spring strength, Mass, Object size etc. are real world values). If the simulation recalculates the positions of the vertices in steps of 1/1000 of a second, the oscillation would be clearly visible: a predictable instability. The oscillation is samples ten times a period.
If the simulation recalculates the vertex position each 1/60 of a second, the neat oscillation would appear as a random jumping around, as a noise, an unpredictable instability. The oscillation is samples less than once a period.
In all cases, an instability which makes that the sim needs longer calculation times to arrive at a steady result. In other words; we have to wait till the oscillation-energy is faded away. The best way to speed that up, is to increase the damping, as this stands for energy loss per second.

- **Mesh density**
  in how many subparts do I chop the system? In cloth, which can be considered a surface, halving sizes in two dimensions imply quadrupling the amount of balls and springs, and calculation time. In Poser I've seen pieces of cloth chopped into pieces of 5x5cm (divides 1 m^2 into 400 balls) to 1x1cm (10.000 balls), but a cotton fibre is as thin as 0.1mm (100 million balls per m^2) and no one is building a sim like that. But by not doing so, one runs the risk of "finite element artifacts".

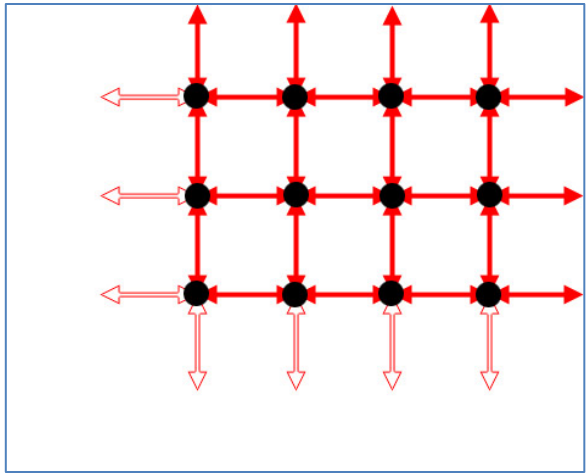
- **Non linearity**
  as long as the mechanics, the systems and the forces considered are linear, increasing mesh density has no effect. For instance, the cloth has a mass density of 1kg/m2, and whether I chop it into 10.000 or 100 million balls will not have effect on the effects of gravity or air damping on the cloth. And when the cloth is 1 m x 1 m and I chop 1 horizontal fibre into 100 little springs, or 10.000, will not have effect on its sheer stretching behavior. But when things are non-linear, mesh density makes a difference.

- Resonance.
  as discussed in the electrical and mechanical sections, each system has some kind of resonance frequency for which the damping is minimal. If I give the system a kickstart, it will send all sorts of signals around in all sorts of frequencies (noise like) but the resonance one stays alive the longest. For mechanics, it's proportional to sqrt(S/M) for spring strength (say: stretch resistance) S and mass M.
  Now I chop the cloth into pieces, at mesh density d. For instance, d=100 chops the 1x1m cloth into fibers of 1 cm (with stretch resistance S/100) and pieces of 1x1cm (with mass M/10.000).
  As a result, the resonance frequency goes up a tenfold (sqrt(d) to be precise). As stated above, this affects the noisiness, the randomness of the vertex positions.
- Folding
  one can imagine that making an angle between two adjacent polys in a mesh is easy for very small distortions but becomes quite hard for the larger angles. Each extra degree of bending requires more and more force to accomplish. So if the 1x1m cloth is divided into 10x10cm pieces and I need a 30° between two adjacent pieces, or the cloth is divided into 1x1 cm pieces and I need 3° between two adjacent pieces but for 10 pieces in a row, the latter requires less force to accomplish.
  In Poser terms: with the same fold-resistance, finer meshes will fold easier. Mesh density makes a difference.

- **Edge effects**

  You might have experienced it when making waves with a heavy rope or garden hose (if not, give it a try). When the other end is loose it jumps up end down wildly. Actually, when your waves in the rope have some amplitude, the loose end will show you the double. You can create some constant wave in the rope if you manage to pick a frequency that creates a specific length of the wave: such that there are ¼ , ¾ , 1 ¼ etc waves in the length of the rope.

  When you fix the other end, to a wall or a pole, the other end will be still at any time. You can still make those standing waves but at different frequencies, such that there are ½, 1, 1 ½ etc waves in the length of the rope.



  Not only ropes and garden hoses, but simulations as well suffer from edge effects. At the end of the ball/spring network, balls have less springs attached. Does that require heavier balls, more strength for the remaining springs attached to those balls, or just the other way around? Does the sim require fixation (like the rope at the wall) or does it allow for jumping around (like the open end case)?

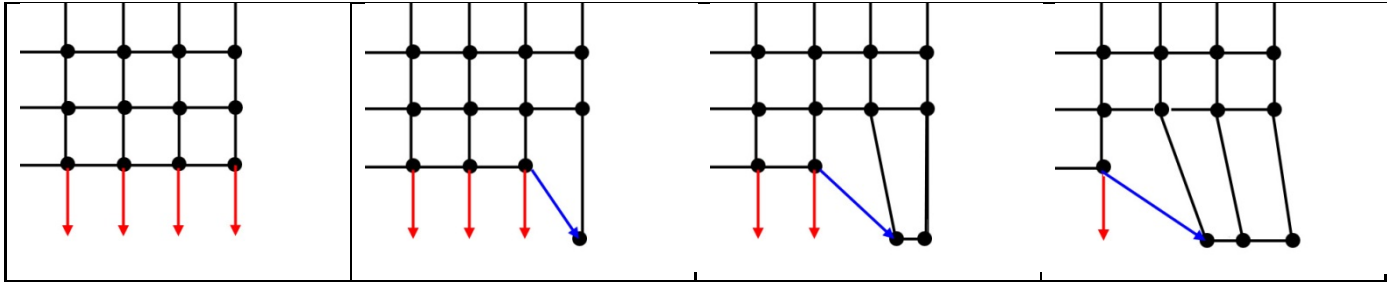
  Anyway, as the image illustrates, the vertices at the edge of a mesh behave different because they are missing forces, as the springs (in between edges) are not there.

  All this is up to the sim builder, and there is no real escape. No sims without edge effects, unless one can make calculations on an edgeless world (a globe, e.g.). For cloth and clothes, one might give the edges specific treatment, like is done with real clothes. A rim, with higher mass / density, less stretching and less folding.

- **Calculation order**

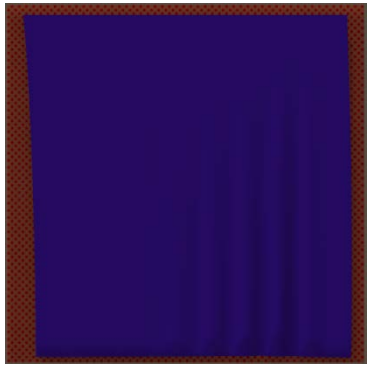
  There are various ways to get the results of a simulation. Iterating step by step is always required, but one can take a ball/spring network in iteration N, and use that information only to calculate the next iteration N+1. A layered approach, so to speak. Another, more continuous approach, is to take a vertex with the information from iteration N, plus the already derived information for surrounding vertices from iteration N+1, and calculate the iteration N+1 for that vertex.

  Again, this is up to the sim builder and there is no escape. The first approach requires more memory (as two nets have to be kept in store) and requires more iterations (as less info is used per step), but is easier to implement as a multi-threaded routine, it will be less prone to edge effects, and if any these will be distributed quite evenly along all edges. The second approach requires less resources, less iterations, but is typically a single-thread routine which will add directional effects to the edges. That is; it will calculate the vertices in a specific order, like top to bottom left to right so edge effects will typically appear in the top left or bottom right corner.



First, there is gravity only. But having the first ball moved will skew the forces on the second, and so on.

Then, you will get this, a piece of cloth skewing to the right-bottom corner when let loose.



I suspect modern simulators like Marvellous Designer to be of the first kind, and Poser Cloth Sim of the second kind as it stems from the Poser 5 period (even before) and has hardly seen a methodical update after then.

- **Unsimulated aspects**
  Aspects of cloth which are absent in the sim, are: thread thickness, and weave density. Thicker threads make thicker cloth, with more weight per square inch, are harder to fold especially over large angles, are harder to stretch, and might increase the friction between threads. This friction between threads by itself affects shear and folding, affects air damping, and so on. A high weave density (threads x thickness per stretching meter) has similar effects.
  Modern advanced cloth simulations take threads and weave aspects into account. Poser cloth room limits itself to global cloth descriptors which include named details. From that, Poser simulates non-woven materials as well; leather, fleece, rubber and the like. This implies a simulation mechanism which is further away from physical details in the real world.

Perhaps you're familiar with software creation. Then you're aware that only 20% of the effort really goes to the program's functionality, and 80% is about the user interface, preventing people from inputting erroneous values or clicking buttons at the wrong moment or selecting impossible combinations of options; that is: making the software user aware. For sims, things are about similar. 20% is about the simulation itself and 80% is about handling the resource requirements, the artifacts from resonance and edges and non-linearities of nature.
This implies that you might have to upgrade yourself from a Poser User to a Poser Sim User.

Going Poser Cloth Room

Offering a cloth simulator to end users introduces a guaranteed set of shortcomings which are not well understood, by those users at least. Not simulating various aspects of cloth (thread thickness, weave density, weave structure) makes it harder to translate the real world to sim parameters, and impatience combined with insufficient resources introduces algorithms with additional artifacts (like skewing cloth).

Mesh structures, mesh densities and various resistance and damping settings will affect the presence or absence of edge effects and resonance, which – at least from an end users perspective – introduce seemingly random vertex movements and floppy edges. Increasing mesh densities, resistance and damping as well as longer simulation times (more frames) kill those effects eventually. Sometimes raising the damping might just make things worse instead of better. But for most people it's unclear that they have to cope with simulation artifacts that have nothing to do with cloth behavior or with their use of Cloth Room.

In the end, it's clear not what is meant by the various kinds of parameters:

- Resistance, against pulling or torque, is like electrical capacity: it stores energy in the cloth and releases it later on.
- Damping is like electrical resistance: it turns cloth movements into energy loss and brings the system to a standstill.
- Gravity, Friction and Wind make the forces (via Mass, or: Cloth Density), and so generate the energy for Resistance to store and for Damping to lose. Which effectively make these the drivers of the simulation.

Unsupported aspects

When I take the backdoor to enter the stage Cloth Room behind the scenes, that is: I take a look at the Python manual, I find some parameters which go unsupported by the properties dials in the 4th panel of the cloth room. These are all of them:

- Airdamping Dial: Air Damping
- Clothclothforce always 10, no way to set it manuall (python script required), unclear what is does
- Clothfriction Dial: Cloth Self Friction
- Dampingstretch Dial: Stretch Damping
- Density Dial: Cloth Density (grams per cm2, 1g/cm2 = 10kg/m2)
- DynamicFriction Dial: Dynamic Friction
- FrictionFromSolid Checkbox: Collision Friction. If checked, the Object parameter are used instead of the Cloth parameters. So either cloth shows a different behavior over rough and smooth object surfaces but the same for all cloth elements, OR cloth shows a different behavior for various cloth elements but the same for all collision objects. In one sim one cannot have both, various cloth elements over various surfaces.
- FrictionvelocityCutoff always 30 (cm/s), no Dynamic Friction below this speed, no way to set it manually
- ShearResistance Dial: Shear Resistance
- SpringResistance no way to set it manually, might be related to EdgeSprings, which are OFF (0)
- StaticFriction Dial: Static Friction
- Thickness is derived from Collision Depth and Collision Offset, which define distances between the outside of the object and the outside of the cloth, which are the parts we see. These collision parameters BTW are aspects of the objects, not of the cloth so in one sim all cloth elements have the same thickness for a specific object, and another thickness or another object. So Cloth Thickness is not very well represented. I guess it doesn't do anything
- U-BendRate not used, no way to set it manually
- U-BendResistance Dial: Fold Resistance
- U-Scale purpose is unclear, no way to set it manually

- UseEdgeSprings 0/1 whether to use Edge Springs calculations. No way to set it, though. It's OFF (0). Here you have the ability of Cloth Room to manage edge effects, as discussed in earlier chapters.
- U-StretchResistance Dial: Stretch Resistance
- V-Bendrate V-values are not used
- V-BendResistance ditto
- V-Scale ditto
- V-StretchResistance ditto

File layout

Cloth info about a cloth prop or cloth figure is stored in the pp2/ppz (prop) and cr2/crz (figure) respectively, and looks like:

```
vertsGroup _default_
	{
	v 0
	v 1
	…
	v 6003
	v 6004
	stitchVertsGroupProperties
		{
		U_Bend_Resistance 30.000000
		V_Bend_Resistance 5.000000		\ no dial
		U_Stretch_Resistance 40.000000
		V_Stretch_Resistance 50.000000		\ no dial
		Shear_Resistance 50.000000
		U_Scale 1.000000			\ no dial
		V_Scale 1.000000			\ no dial
		Density 0.010000
		Thickness 0.000000			\ no dial
		Spring_Resistance 5000.000000
```

```
            Air_Damping 0.020000
            Dynamic_Friction 0.100000
            Static_Friction 0.500000
            Friction_Velocity_Cutoff 30.000000      \ no dial
            Cloth_Cloth_Force 10.000000             \ no dial
            U_Bend_Rate 0.000000                    \ no dial
            V_Bend_Rate 0.000000                    \ no dial
            Cloth_Cloth_Friction 0.000000
            Damping_Stretch 0.010000
            get_friction_from_solid 0
            Use_Edge_Springs 0                      \ no setting
            anisotropic 0                           \ no setting
            }
      }
vertsGroup _choreographed_
      {
      }
vertsGroup _constrained_
      {
      v 3
      v 6
      …
      v 5752
      v 5753
      }
vertsGroup _softDecorated_
      {
      }
vertsGroup _rigidDecorated
      {
      }
```