Student number	point total	req total	extra total	R1 Euler integrator (1p)	R2 Spring system (2p)	R3 Trapezoid integrator (2p)	R4 Pendulum system (2p)	R5 Cloth system (3p)	mod	notes / wtf / ...	RK4 (2p)	Spray system (1-3p)	Wind (2p)	Mouse drag/ poke (2-3p)	Frictionl. coll. (2p)	Particle spline editor (2-3p)	Cloth tearing (1+p)	Particle rendering (1-4p)	Implicit integr (8-10p)	GPU stuff (4+p)	Other extras (?p)	What other extras
218096	0	0	0																			
225157	0	0	0																			
270034	0	0	0																			
292009	0	0	0																			
292326	10	10	0	1	2	2	2	3														
292986	0	0	0																			
293545	4	2.5	1.5	1		1.5				R3,RK4: The added step or 1/2*step should be removed from all loops	1.5											
295323	0	0	0																			
297606	0	0	0																			
311210	0	0	0																			
347022	0	0	0																			
350006	7	7	0	1	2	2	2															
350475	0	0	0																			
353692	5.5	5.5	0	1	1.5	2	1	0		Don't attempt to fix problems by adjusting the given constants; R2: Incorrect spring constant and rest length; R4: other spring force missing from evalF and constants incorrect. Why haven't you used the springs_ vector?											
353757	2	2	0	1	1					R2: Use spring_ in reset to store k and rlen, use those when calling fSpring. Use the forces to calculate f[3] ((fSpring+fGravity+fDrag)/mass) i.e. the acceleration of the particle. Also set f[2] to the speed of the particle; you can get it from the old state vector.											
357083	0	0	0																			
362256	0	0	0																			
401311	0	0	0																			
424615	13.5	9	4.5	1	2	2	1.5	2.5		R4&R5: You should use n-1 when calculating rest length and starting positions, for n particles there are n-1 springs. Clear the springs vector in reset, otherwise the system becomes too stiff because there are extra springs. EvalF missing division by mass; RK4: Xt calculated incorrectly in each loop; should be: xt=x0+0.5*k1, xt=x0+0.5*k2, xt=x0+k3.	1.5			3							
425494	0	0	0																			
425614	12.5	10	2.5	1	2	2	2	3		Wind: fWind may return really large values and the cloth streches out of the image. Wind pretty directional.	2		0.5									
426419	0	0	0																			
426736	0	0	0																			
427492	0	0	0																			
427793	8.5	8.5	0	1	2	2	2	1.5		R5: evalf is correct, the indexing in reset() is off (making a pair of functions that do (x,y)<=>(i) would maybe help dealing with these)											
427845	0	0	0																			
428381	0	0	0																			
428789	3.5	3.5	0	1	0.5	2																
430324	0	0	0																			
430463	7	7	0	1	2	2	2															
431857	17	10	7	1	2	2	2	3		Wind randomized differently for each particle and frame, doesn't really look like wind. +1 for rendering the plane in the spray system.	2	3	1					1				
432241	10	10	0	1	2	2	2	3														
437631	3	3	0	1		2																
438397	0	0	0																			
472379	0	0	0																			
473158	0	0	0																			
473420	0	0	0																			
473637	0	0	0																			
474380	0	0	0																			
475389	0	0	0																			
475758	0	0	0																			
475910	0	0	0																			
476498	0	0	0																			
477170	0	0	0																			
477400	12	10	2	1	2	2	2	3		Clear springs_ at reset	2											
477617	6	6	0	1	2	1.5	1.5			R3: You should be adding to x0 in the final step instead of x; R4: Clear the springs in reset and calculate rest length based on the starting positions.											
477659	10	10	0	1	2	2	2	3														
477701	6	6	0	1	2	2	1															
478328	0	0	0																			
478470	0	0	0																			
478687	0	0	0																			
479505	0	0	0																			
479576	0	0	0																			
479725	0	0	0																			
480248	0	0	0																			
480303	8.5	8	0.5	1	1.5	2	1.5	2		fDrag should take the velocity as input, not position; R4&5: Use n-1 when calculating the lengths since there are n-1 springs; You're using the state_ variable instead of the input parameter state in evalF. These are not the same; R5: The spring forces don't utilize i2 at all; RK4 doesn't work	0.5											
480730	23	10	13	1	2	2	2	3		GPU RK4: +2	2				2		2	1		6		
481014	10	10	0	1	2	2	2	3														
481441	10	10	0	1	2	2	2	3														
493578	0	0	0																			
506355	8	8	0	1	2	2	2	1														
508793	-2	0	0						-2													
514020	0	0	0																			
516109	10	10	0	1	2	2	2	3		R4,5: Note that your evalFs occasionally use state_ and not state, making the better integrators not work as well as they should.												
519656	0	0	0																			
525653	0	0	0																			
525666	8	8	0	1	1.5	2	1	2.5		Don't attempt to fix problems by adjusting the given constants; R2: Fixed particle should be at origin; R3: Particles too close to one another. Position should be multiplied by d.length(). Rest length should be the initial distance between particles. R4&5: You should use n-1 when calculating rest length and starting positions, for n particles there are n-1 springs. And you don't clear springs vector at reset causing e.g. the cloth to be more unstable (no points reduced).											
525792	0	0	0																			
525925	14	10	4	1	2	2	2	3			2	2										
526490	12	10	2	1	2	2	2	3		Clear springs_ in reset	2											
526717	10	9	1	1	2	2	1.5	2.5		R4&R5: Missing division by mass. fDrag should be per particle, not per spring; RK4: Remove the division by 2 when multiplying k2 and k3 by step and fix the typo in the calculation of k4.	1											
526746	0	0	0																			
527143	0	0	0																			
527347	3	3	0	1		2																
527389	0	0	0																			
527444	0	0	0																			
527923	25.5	10	15.5	1	2	2	2	3		Mouse drag position doesn't match mouse. +2 to the GPU cloth for RK4.	2	3	2	2.5						6		
528634	10	9.5	0.5	1	2	2	2	2.5		R2,4,5: Forces not divided by mass	0.5											
528883	0	0	0																			
529293	0	0	0																			
529303	0	0	0																			
529617	0	0	0																			
529992	26	10	16	1	2	2	2	3		Wind could be a bit smoother and more natural; GPU: +2 for RK4 and +1 for collision	2	3	2		2					7		
530185	0	0	0																			
530907	0	0	0																			
530981	5	5	0	1	2	2				R2, evalF has different spring coefficient (???)												
540094	0	0	0																			
540311	12	10	2	1	2	2	2	3			2											
541543	10	10	0	1	2	2	2	3														

Student number	point total	req total	extra total	R1 Euler integrator (1p)	R2 Spring system (2p)	R3 Trapezoid integrator (2p)	R4 Pendulum system (2p)	R5 Cloth system (3p)	mod	notes / wtf / ...	RK4 (2p)	Spray system (1-3p)	Wind (2p)	Mouse drag/ poke (2-3p)	Frictionl. coll. (2p)	Particle spline editor (2-3p)	Cloth tearing (1+p)	Particle rendering (1-4p)	Implicit integr (8-10p)	GPU stuff (4+p)	Other extras (?p)	What other extras
544375	12	9	3	1	2	2	2	2		R5: Rest lengths and starting positions incorrect. What's up with the seemingly arbitrary constants 1.15 and 2.47 len_diag calculation likely doesn't do what you intend, you are constructing Vec3f from 2 floats as input where the first float is implicity converted to Vec2f. The lengths should be len = width/(x_-1) for structural, len*sqrt(2) for shear, and len*2 for flexion; Wind pretty much constant, the per particle strength variation doesn't make much sense.	2		1									
544566	0	0	0																			
549749	0	0	0																			
552969	0	0	0																			
556347	0	0	0																			
561578	0	0	0																			
563068	0	0	0																			
570116	0	0	0																			
586210	17.5	10	7.5	1	2	2	2	3		Collision: Very unstable because velocity stays the same (matrix multiplications do not change anything). Changing to spherical coordinate system is uneccessary. Wind: model is pretty simple	2	3	1.5		1						
587170	20	9.5	10.5	1	2	2	2	2.5		R4,5: Don't change constants unneccessarily; R5: Reset position not exactly correct. Incorrect fixed points; Wind: could be smoother; GPU RK4 + wind: +3p	2		1.5							7		
587921	15.5	9.5	6	1	2	2	1.5	3		Clear springs_ in reset. Your evalF functions are missing division by mass; Wind: The force is too low to have a noticeable effect. Dynamic wind is good but could be smoother, e.g. you could use cubic or cosine interpolation.	2	3	1									
588137	0	0	0																			
588441	7.5	7.5	0	1	2	2	1.5	1		R4: Missing division by mass												
589291	0	0	0																			
589437	9.5	9.5	0	1	2	2	2	2.5		R5: force not divided by mass to get acceleration												
589848	0	0	0																			
590112	9	9	0	1	2	2	2	2		R5: no division by mass and quadratic complexity due to the nested loops; both of these contribute to the slow behavior. You should treat the springs in a separate loop instead of going through all springs for all particles; then you wouldn't have to test if the spring was correct but could simply compute the forces for each one.												
590332	13.5	9.5	4	1	2	2	2	2.5		R5: only spring force divided by mass. Fluid system has weird forces and no regenerating particles, Wind has constant direction and per-particle random force, Mouse dragging has one global force in the unit direction, doesn't lead to much interactivity.	2	0.5	1	0.5								
590426	18.5	10	8.5	1	2	2	2	3			2	3			2		1.5					
593177	0	0	0																			
593452	9.5	9.5	0	1	2	2	1.5	3		R4&5: Missing division by mass												
593876	8.5	8.5	0	1	2	2	2	1.5														
594367	6	6	0	1	2	2	1			R4: drag should be per particle instead of per spring, lerp in reset() should use n_-1 instead of n_ to compute interpolation factor (so last particle is at end_point)												
594590	3	3	0	1		2																
594930	0	0	0																			
595201	15	9.5	5.5	1	2	2	2	2.5		R5: not dividing force by mass for acceleration, Wind: rerandomized every frame	2	2	1.5									
595612	12	10	2	1	2	2	2	3		R5: Clear the springs vector in reset	2											
596048	9.5	9.5	0	1	2	2	2	2.5		R5: The rest length of the shear springs was just a bit off. It should have been sqrt(xDiff^2 + yDiff^2)												
596242	20	10	10	1	2	2	2	3		GPU RK4: +2	2				2					6		
596747	0	0	0																			
596789	0	0	0																			
596792	11.5	9.5	2	1	2	1.5	2	3		R3: The euler step should be x0 + step*f0	2											
596857	7	6	1	1	2	2	1			R4: Each particle only takes into account one spring while there are two springs connected to each particle. Missing division by mass from fGravity and fDrag. In reset use index (i+i)*2 when calculating the variable "length" and n_-1 for the start positions; RK4: Correct kind of structure but doesn't quite implement the formula correctly. Note that k1 should just be step*f0 (or just f0 if the multiplication by step is included in the final calculation of xm) and work forward from there.	1											
597445	15.5	10	5.5	1	2	2	2	3		The wind changing completely every frame is a bit unrealistic	2		1.5	2								
598088	16	10	6	1	2	2	2	3			2	2			2							
598318	0	0	0																			
602851	0	0	0																			
603067	8.5	8.5	0	1	1.5	2	2	2		R2: Typo in distance calculation (pos1 v. pos2); R4, 5: You don't clear springs vector at reset causing e.g. the cloth to be more unstable (no points reduced). R5: Rest lengths of structural and flex springs are incorrect												
603096	10	10	0	1	2	2	2	3														
603326	0	0	0																			
604105	11	9	2	1	2	2	1.5	2.5		Note that your evalFs use state_ and not state, making the better integrators not work as well as they should. R4, 5: Drag applied multiple times per particle. Should be added at the same time as gravity. R5: You should use n-1 when calculating rest length and starting positions, for n particles there are n-1 springs.	2											
606064	0	0	0																			
606268	0	0	0																			
608952	9.5	9.5	0	1	2	2	2	2.5		R4,5: Forces not divided by mass. And you don't clear springs vector at reset causing e.g. the cloth to be more unstable (no points reduced).												
609142	21.5	10	11.5	1	2	2	2	3		Nice idea with the vector field wind but you should also randomize it; Collisions: A bit unstable. Cool smoke rendering!	2	3	1		1.5			4				
609155	0	0	0																			
609168	12	10	2	1	2	2	2	3		Clear the springs at reset	2											
610827	4.5	4.5	0	1	2	1	0.5			R3: In the first loop, you should have calculated a state xh like you did x1 in eulerStep and after calculate fh using evalF and xh.												
612155	0	0	0																			
612540	0	0	0																			
612812	0	0	0																			
621308	0	0	0																			
647175	0	0	0																			
647502	23.5	10	13.5	1	2	2	2	3		Frictionl. coll: Collision with complete friction instead; Spray: Particles are not usually created all at once but there is a predefined rate at which the sprinkler emits particles. The current implementation seems a bit unnatural as the particles are emitted at clear discrete steps. GPU RK4: +1.5p. In evalF.glsl restLen should have w-1, not w.	2	2.5	2		1.5					5.5		
648080	15	9	6	1	2	2	1.5	2.5		R4: Use the starting positions to get rest length; R4&R5: fDrag should be per particle, not per spring; R5: Rlen should be width / (x_-1). Wind: Simple per particle randomization along a single axis, quite unrealistic.	2	2	1				1					
648569	7	7	0	1	2	2	2			Clear the springs in reset												
648860	0	0	0																			
649458	0	0	0																			
650191	0	0	0																			
650560	0	0	0																			
650829	8	8	0	1	2	2	2	1		R5: only structural springs attempted, multiplication by two done both when storing spring indices and after reading them, drag added multiple times to some of the particles												
651640	9.5	9.5	0	1	2	2	2	2.5		R4&5: You don't clear springs vector at reset causing e.g. the cloth to be more unstable (no points reduced). You should use n-1 when calculating rest length and starting positions, for n particles there are n-1 springs.												
651802	0	0	0																			
652209	23.5	10	13.5	1	2	2	2	3		R4,5: You don't clear springs vector at reset causing e.g. the cloth to be more unstable (no points reduced). Spray: The bouncing doesn't work properly because you're trying to do it in evalF. You'll need another function where you detect collisions; GPU RK4: +2	2	1.5	2	2						6		

Student number	point total	req total	extra total	R1 Euler integrator (1p)	R2 Spring system (2p)	R3 Trapezoid integrator (2p)	R4 Pendulum system (2p)	R5 Cloth system (3p)	mod	notes / wtf / ...	RK4 (2p)	Spray system (1-3p)	Wind (2p)	Mouse drag/ poke (2-3p)	Frictionl. coll. (2p)	Particle spline editor (2-3p)	Cloth tearing (1+p)	Particle rendering (1-4p)	Implicit integr (8-10p)	GPU stuff (4+p)	Other extras (?p)	What other extras
652584	34.5	10	24.5	1	2	2	2	3		Frictionl. coll: Collision with complete friction instead; RKF: You don't actually change the step size of the future steps; GPU RK4: +2. The implicit integration scheme is not strictly implicit euler since it doesn't use the normal newton search that updates the jacobian every frame. There is also some subtle issue with (likely) some of the jacobian evaluations.	2	3	2		2.5			1	6	6	2	RKF (2p)
653156	0	0	0																			
653347	10	10	0	1	2	2	2	3														
654142	12	10	2	1	2	2	2	3			2											
654294	12	10	2	1	2	2	2	3		Well spotted; the different stiffness is due to not clearing the spring list (reset() is called once for every system at launch so you always have at least two copies of springs). No point reduction as this is more of a mistake on our part.	2											
654618	9	9	0	1	2	1	2	3		R3: This is the mid point integrator, not a trapezoidal one.												
655109	0	0	0																			
655361	0	0	0																			
655390	0	0	0																			
656014	8.5	8.5	0	1	2	2	1.5	2		Don't change the given constants. R2&R4: Rlen should be based on the particle distances in the starting position; R4&R5: You're using the member variable "state_" instead of the input parameter "state" in evalF, these are not the same states. The number of springs is n_ - 1, use this when calculating starting positions and rlens.												
657068	0	0	0																			
657181	28	10	18	1	2	2	2	3		GPU RK4: +2, reasonable non-uniform tearing: +2	2	3	2		2		3			6		
657437	10	9	1	1	2	2	2	2		R4, R5: drag should be added once per particle, not once per spring. The correct rest length for the cloth would be width/(x_-1). The wind model is overly simple, each particle having its own random force.			1									
657767	24	10	14	1	2	2	2	3		Wind direction static. Implicit Euler: the integrator itself is fine (the example does 20 iterations and breaks when dy_vec.squaredNorm()<=.001, otherwise they're the same), but on evalJ you directly write to coeffRef(acc_i1+i, pos_i1+j) and coeffRef (acc_i2+i, pos_i2+j) -- these will be touched by several springs! You should initialize them to 0 before the loop and then add the contribution of each spring; after these changes your solution is quite stable (can use max step size in cloth).			1		2				7		4	Implicit optimization
657893	0	0	0																			
663434	0	0	0																			
665173	9.5	9.5	0	1	2	1.5	2	3		R3: There should not be a +step in the first loop; R4, 5: You don't clear springs vector at reset causing e.g. the cloth to be more unstable (no points reduced).												
665678	28	10	18	1	2	2	2	3		Strain heatmap graded for particle rendering. The bullet is closer to frictionless collision, so +1 for that and +1 for the trapezoid.	2		1		2		1	3		6	3	RKF (3p)
666208	0	0	0																			
666211	13	10	3	1	2	2	2	3		R4,5: You could have had a completely separate loop for the spring forces and you could have indexed f with i1 and i2. This way you wouldn't have had to loop through springs at each particle; Wind: Simple model	2		1									
666253	0	0	0																			
710015	8	8	0	1	2	2	1.5	1.5		R4, R5: drag should be added once per particle, not once per spring. Not dividing forces by mass to get acceleration.												
715298	17	10	7	1	2	2	2	3			2	3	2									
716734	0	0	0																			
717377	0	0	0																			
717539	0	0	0																			
718020	0	0	0																			
718208	9.5	9.5	0	1	2	2	2	2.5		R4&5: You should use n-1 when calculating rest length and starting positions, for n particles there are n-1 springs.												
718512	13.5	9.5	4	1	2	2	1.5	3		R4: Spring force not divided by mass	2		2									
718826	22.5	10	12.5	1	2	2	2	3		Drag: To prevent the selected particle from twitching about, it shouldn't be moved in evalF	2	3	2	2.5	2		1					
719032	9	9	0	1	1.5	2	1.5	3		R2: As stated in the comments, the fixed particle is intended to be located in the origin while the other particle starts at start_pos. Rest length of 0 also doesn't really seem to make sense, using the length of the spring at the starting position is a good value to use; R4&5: Clear the springs_ vector at reset. EvalF missing division by mass.												
721619	0	0	0																			
721923	0	0	0																			
723154	0	0	0																			
723329	11.5	9.5	2	1	2	2	2	2.5		R5: you're skipping some springs that should be there: each comparison of i>x_ and i>2*x_ should be >= instead of >. the change fixes the asymmetry at the top left corner. RK4: function4 should be evaluated with xm3 instead of k3, k4 should be function4*step and not function3*step. Wind is strongly directional, per-particle randomization doesn't really give you a "random wind direction".	1		1									
723468	11	9	2	1	2	2	2	2		R5: Missing shear and flexion springs from the result; this is the danger of treating each spring manually. Flexions should have double the rest length., and rest length should be width/(x_-1), 0.075 would correspond to width/x_.	2											
723484	12	10	2	1	2	2	2	3			2											
723565	9	9	0	1	2	2	2	2		R5: Missing flexion and shear springs												
723976	0	0	0																			
724483	0	0	0																			
726915	10	10	0	1	2	2	2	3														
728696	0	0	0																			
729297	0	0	0																			
732323	0	0	0																			
737551	12	10	2	1	2	2	2	3			2											
765714	0	0	0																			
765756	11.5	9.5	2	1	2	2	1.5	3		R4: Force not divided by mass; R5: Looping through the springs in a separate loop to get the spring forces and using i1 and i2 to index f would have been a lot easier. No need to make all those conditions. The springs_ vector is useless in your code.	2											
765785	12	9	3	1	2	2	1.5	2.5		R4: Rest length shoud be 1.f / (n_-1) * end_point. length(). R4&R5: Forces not divided by mass. R5: Particle count has been changed to 10x10 and they are not exactly in the correct position at reset. The whole first row of particles are fixed. Wind: Simple wind	2		1									
765882	10	10	0	1	2	2	2	3														
766108	0	0	0																			
767136	0	0	0																			
769396	0	0	0																			
772419	0	0	0																			
784465	0	0	0																			
784847	0	0	0																			
784902	0	0	0																			
785053	0	0	0																			
785134	13.5	10	3.5	1	2	2	2	3		Wind: rerandomized every frame which does not seem realistic	2		1.5									
785163	0	0	0																			
785228	26.5	9.5	17	1	1.5	2	2	3		R2: In fSpring, the resulting force should be still divided by (pos2-pos1).length(). The implicit integrators themselves look more or less correct, but there has to be some subtle mistake (likely in the evalJs) since they don't work with large timesteps.	2							7		8	Optimized implicit (4p), implicit midpoint (2p), implicit trapezoid (2p)
785257	10	10	0	1	2	2	2	3														
785325	0	0	0																			
785354	13.5	10	3.5	1	2	2	2	3		The wind changing completely every frame is a bit unrealistic	2		1.5									
785367	13.5	10	3.5	1	2	2	2	3		The wind changing completely every frame is a bit unrealistic	2		1.5									
785435	0	0	0																			
785448	0	0	0							Code doesn't compile												
785451	3	3	0	1	2					R2: Forces not divided by mass (The mass is here 1 so it does not make a difference though).												

Student number	point total	req total	extra total	R1 Euler integrator (1p)	R2 Spring system (2p)	R3 Trapezoid integrator (2p)	R4 Pendulum system (2p)	R5 Cloth system (3p)	mod	notes / wtf / ...	RK4 (2p)	Spray system (1-3p)	Wind (2p)	Mouse drag/ poke (2-3p)	Frictionl. coll. (2p)	Particle spline editor (2-3p)	Cloth tearing (1+p)	Particle rendering (1-4p)	Implicit integr (8-10p)	GPU stuff (4+p)	Other extras (?p)	What other extras
785493	4	4	0	1	1	2				R2: You should initialize spring_.k and spring_.rlen in reset() and then input them to fSpring in evalF.												
785503	0	0	0																			
785516	0	0	0																			
795551	10	10	0	1	2	2	2	3														
795577	10	10	0	1	2	2	2	3		R5: cloth stiffer and more unstable because you don't clear springs vector at reset (no points reduced)												
795593	6	6	0	1	2	2	1			R4: You should use n-1 when calculating starting positions. The way you're dividing by mass when calculating the spring forces causes you to do the division more than once per particle.												
795629	0	0	0																			
795658	0	0	0																			
795674	13.5	9	4.5	1	2	2	2	2		R5: The idea of flexions is a bit reversed; they should start from each particle and extend two particles away (over the neighbor that's connected by the structural spring), you have them starting from every other particle but connecting to the immediate neighbor. Wind is strongly directional, per-particle randomization doesn't really give you a "random wind direction".	2		1					1.5				
795713	10	10	0	1	2	2	2	3														
795865	13.5	10	3.5	1	2	2	2	3		R5: The initial particle positions at reset are not quite right; Wind: model is pretty simple	2		1.5									
796178	10	10	0	1	2	2	2	3		R4,5: You don't clear springs vector at reset causing e.g. the cloth to be more stiff and unstable (no points reduced).												
798257	0	0	0																			
801131	0	0	0																			
804646	10	10	0	1	2	2	2	3														
807711	0	0	0																			
809609	14.5	10	4.5	1	2	2	2	3		Tearing only affects the edge constraint, not springs, and the constraint doesn't stay broken. Wind randomization a bit odd. +0.5 for the big bang, interesting concept!	2	0.5	1.5				0.5					
811383	0	0	0																			
814872	0	0	0																			
818315	0	0	0																			
821289	0	0	0																			
822709	0	0	0																			
46596K	0	0	0																			
55055P	0	0	0																			
62727K	12	10	2	1	2	2	2	3			2											
64879R	11.5	10	1.5	1	2	2	2	3		RK4: One 1/2 too many in the first two loops	1.5											
65451T	0	0	0																			
67932J	0	0	0																			
69246M	12	10	2	1	2	2	2	3			2											
77241H	0	0	0																			
77388B	0	0	0																			
83107B	0	0	0																			
83854J	0	0	0																			
84171B	0	0	0																			
84805K	0	0	0																			
k28342	0	0	0																			
k90624	0	0	0																			
k93517	0	0	0																			