Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS (2013)
J. Bender, J. Dequidt, C. Duriez, and G. Zachmann (Editors)

Exploring the Use of Adaptively Restrained Particles for Graphics Simulations

Pierre-Luc Manteaux François Faure Stéphane Redon Marie-Paule Cani

LJK-CNRS Grenoble University
Inria

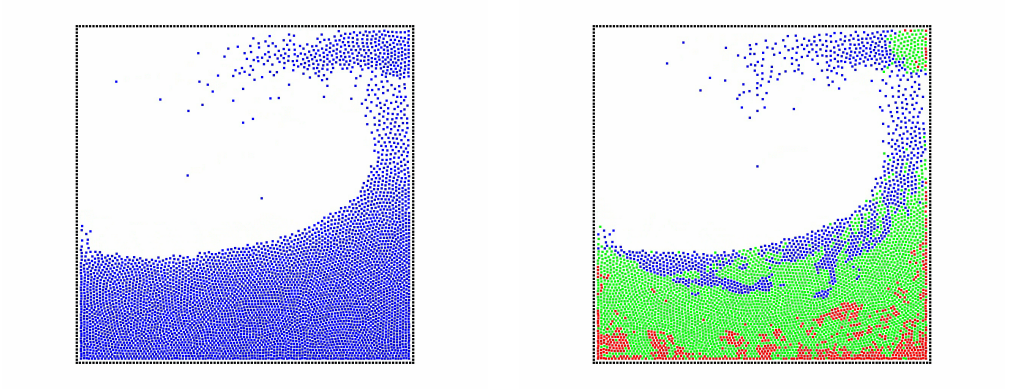


Figure 1: *A dam break simulation with 5000 particles simulated with WCSPH (on the left) and with our adaptive method (on the right). On the right image, blue corresponds to full-dynamics particles, green to transition particles and red to restrained particles.*

Abstract

In this paper, we explore the use of Adaptively Restrained (AR) particles for graphics simulations. Contrary to previous methods, Adaptively Restrained Particle Simulations (ARPS) do not adapt time or space sampling, but rather switch the positional degrees of freedom of particles on and off, while letting their momenta evolve. Therefore, inter-particles forces do not have to be updated at each time step, in contrast with traditional methods that spend a lot of time there.

We present the initial formulation of ARPS that was introduced for molecular dynamics simulations, and explore its potential for Computer Graphics applications: We first adapt ARPS to particle-based fluid simulations and propose an efficient incremental algorithm to update forces and scalar fields. We then introduce a new implicit integration scheme enabling to use ARPS for cloth simulation as well. Our experiments show that this new, simple strategy for adaptive simulations can provide significant speedups more easily than traditional adaptive models.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling —Physically based modeling

1. Introduction

Combining efficiency with visual realism had been one of the main goals of Computer Graphics research in the last decade. The general strategy for efficient graphical simulations is to concentrate the computational time on the most interesting parts of an animated scene (such as near the surface of a fluid), while simplifying the rest of the scene according

DOI: 10.2312/PE.vriphys.vriphys13.017-024

to some visual quality criteria. A number of adaptive simulation methods, aimed at controlling the trade-off between performance and precision, have been developed. Most of them consist in changing time or space sampling, using adaptive time steps or multi-scale models. Although several of them give impressive results, they are often difficult to implement, may-be restricted to specific applications, sometimes generate discontinuity artifacts due to sudden simplifications.

A different approach for adaptive simulation [AR12] was recently proposed in the context of molecular dynamics (MD). The key idea is that since most of the computation time is spent in computing interaction forces based on positions, particles with low velocity could be considered fixed in space - and the corresponding interaction forces constant - until they accumulate enough momentum to start moving again. While freezing objects to gain computation time has been extensively used in video games, the question of when and how to release them has not been extensively studied, and has mainly relied on *ad hoc* heuristics. Adaptively Restrained Particle Simulations (ARPS), in contrast, introduces a physically sound approach with proven correctness, and has been successfully used in the context of predictive, energy- and momentum-conserving particle simulation.

We present the first applications of ARPS to physically-based animation, and we complement the approach with two novel extensions, to cope with the specificity of our domain. Damping forces, not present in the classical MD framework, create specific difficulties that we tackle using a novel freeze criterion. Additionally, we derive an implicit integration method for applying ARPS to stiff objects. The remainder of this paper is organized as follows. We first briefly review the previous work on adaptive mechanical simulations in computer graphics. We then summarize the ARPS method in Section 3. The question of damping is studied in Section 4 through viscosity forces in SPH simulations. An extension to implicit integration is presented in Section 5 using a cloth-like use case. Practical implementation and parameter tuning are then addressed in Section 6. We finally discuss results and perspectives in Section 7.

2. Previous work

There have been two main ways to address adaptivity in Computer Graphics: time adaptivity and space adaptivity. Time adaptivity has been used to perform as large time steps as possible without compromising stability. [DC96] locally adapt the time step based on the Courant-Friedrichs-Lewy criterion [PTVF92] for early SPH simulation, and this was later extended to more recent SPH formulations [IAGT10]. The same criterion was derived and used for deformable solids, using adaptive space sampling as well [DDBC99, DDCB01]. Time adaptivity has also been used to conservatively handle collisions [HVS*09].

Space adaptivity has been first used in mass-spring systems [HPH96, GCS99] and then extended to continuous models such as FEM [WDGT01]. [DDCB01] use non-nested meshes, while [GKS02] propose to consider adaptivity from the shape functions viewpoint on a single mesh. [SSIF07] constrained T-nodes within other independent nodes. [MKB*08] solved multi-resolution junctions with polyhedral elements. [OGRG07] combine adaptivity and multigrid solution. Real-time remeshing has been applied to 1D elements such as rods and wires [LGCM05, ST08, SLNB11] and to 2D surfaces like cloth [BD12], [NSO12] and paper [NPO13]. In 3D, adaptive meshes have been used to simulate cutting [CDA00], plasticity [BWHT07, WRK*10] and thin fluid features [WT08], [ATW13]. Adaptive shape matching has been proposed using a meshless, octree-based approach [SOG08]. In addition, adaptive SPH fluid simulations were recently proposed [APKG07], [SG11], [GP11], [OK12].

An interesting alternative to adaptive space sampling is adaptive deformation fields. [RGL05] dynamically create rigid clusters of articulated bodies, while [KJ09] decompose of the displacement field on a dynamically reduced set of deformation modes.

Despite decades of improvements, it seems that adaptive models are not yet mature or general enough to be used in mainstream software. Adaptivity is typically difficult to apply because it requires significant changes in the models or the equation solvers. In contrast, ARPS require comparatively small changes to the simulators and may become an interesting alternative.

3. Adaptively Restrained Particles

Basic ideas: Adaptively Restrained Particle Simulations (ARPS) [AR12] was recently developed to speed up particle simulations in the field of Molecular Dynamics. They rely on Hamiltonian mechanics, where the state of a system is described by a position vector $\mathbf{q}$ and a momentum vector $\mathbf{p}$, and its time evolution is governed by the following differential equations:

$$\frac{d\mathbf{p}}{dt} = -\frac{\partial \mathcal{H}}{\partial \mathbf{q}}$$
$$\frac{d\mathbf{q}}{dt} = +\frac{\partial \mathcal{H}}{\partial \mathbf{p}}$$

Here, the Hamiltonian $\mathcal{H}$ is the total mechanical energy given by:

$$\mathcal{H}(\mathbf{q},\mathbf{p}) = \frac{1}{2}\mathbf{p}^T M^{-1}\mathbf{p} + V(\mathbf{q}) \quad (1)$$

where the first term corresponds to the kinetic energy, while the second represents the potential energy. In [AR12], an *adaptively restrained* (AR) Hamiltonian is introduced:

$$\mathcal{H}_{AR}(\mathbf{q},\mathbf{p}) = \frac{1}{2}\mathbf{p}^T \Phi(\mathbf{q},\mathbf{p})\mathbf{p} + V(\mathbf{q}) \quad (2)$$

The matrix Φ is a block-diagonal matrix used to switch on or off the positional degrees of freedom of the particles dur-

ing the simulation. Each 3x3 block corresponds to a particle i equal to $\Phi_i(q_i, p_i) = m_i^{-1}[1 - \rho_i(q_i, p_i)]\mathbf{I_{3x3}}$. The function $\rho_i \in [0,1]$ is called the *restraining function*. When $\rho_i = 0$, $\Phi_i = m_i^{-1}$ and the particle is *active*: it obeys standard (full) dynamics. When $\rho_i = 1$, $\Phi_i = 0$ and the particle is *inactive* (not moving). When $\rho_i \in [0,1]$, the particle is in transition between the two states. The restraining function ρ_i of each particle is used to decide *when* to switch positional degrees of freedom on or off. In [AR12], ρ_i depends on the particle kinetic energy. The function uses two thresholds, a restrained-dynamics threshold ε^r and a full-dynamics threshold ε^f. It is defined as :

$$\rho_i(p_i) = \begin{cases} 1, & \text{if } 0 \leq K_i(p_i) \leq \varepsilon_i^r \\ 0, & \text{if } K_i(p_i) \geq \varepsilon_i^f \\ s(K_i(p_i)) \in [0,1], & \text{elsewhere} \end{cases} \tag{3}$$

where $K_i = p_i^2/2m_i$ is the kinetic energy, and s is a twice-differentiable function. In practice a 5^{th}-order spline is used.

Adaptive equations of motion: The adaptive equations of motions are derived from the AR Hamiltonian (2):

$$\frac{d\mathbf{p}}{dt} = -\frac{\partial \mathcal{H}_{AR}}{\partial \mathbf{q}} = -\frac{\partial V(\mathbf{q})}{\partial \mathbf{q}}$$
$$\frac{d\mathbf{q}}{dt} = \frac{\partial \mathcal{H}_{AR}}{\partial \mathbf{p}} = M^{-1}[I - \rho(\mathbf{p})]\mathbf{p} - \frac{1}{2}\mathbf{p}^T M^{-1}\frac{\partial \rho(\mathbf{p})}{\partial \mathbf{p}}\mathbf{p}$$

Applied to a particle, one can derive the rate of position change, which we call *effective velocity*, as:

$$\dot{q} = \frac{1}{m}\left((1 - \rho(p))p - \frac{1}{2} \parallel p \parallel^2 \frac{\partial \rho(p)}{\partial p}\right) \tag{4}$$

While the momenta evolve as in classical Hamiltonian mechanics, position evolves differently. When a particle's momentum is small enough, the particle becomes inactive and stops moving. However, even if the particle is inactive, its momentum may change. Therefore its kinetic energy may become large enough again for the particle to resume moving. In general, particles switch between active and inactive states during the simulation.

A simple example: Consider a 1D harmonic oscillator : a particle attached to the origin with a perfect spring. Fig. 2 shows a phase portrait of the corresponding AR system. In classical mechanics, the trajectory of the state in this (position, momentum) space is an ellipse, the size of which depends on the (constant) energy of the system. Using ARPS, the position is constant (vertical straight parts) as long as the kinetic energy is small enough, while it is an ellipse as long as the kinetic energy is big enough. These trajectories are connected by a transition corresponding to an energy between the two thersholds of eq. (3). The closed trajectory corresponds to a constant *adaptively restrained energy* H_{AR}.

Generalization: Due to the similarity of the adaptive kinetic energy with the standard kinetic energy, one can show that particle systems simulated using ARPS exhibit the expected properties of standard physical simulation, namely the conservation of momentum and (adaptive) energy. It is therefore possible to perform macroscopically realistic simulations with reduced computation time.

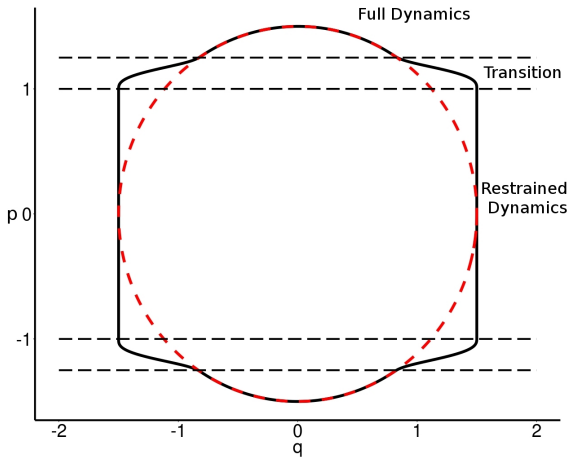


Figure 2: *Phase portrait of a harmonic oscillator. The red dotted ellipse corresponds to standard Hamiltonian mechanics, while the solid black line corresponds to ARPS. During restrained dynamics momentum is accumulated. Then a transition deals with the accumulated energy before getting back to the full dynamics.*

Computational performance: [AR12] obtained significant speedup exploiting immobility of particles. An incremental method was used to update the particles forces at each time step, while saving time on inactive particles :

1. All forces that were acting on each active particle at the previous time step are substracted based on previous position.
2. New forces based on current positions are added to each active particle.

The computational performance comes from the absence of force computation between two inactive particles and the absence of neighbor search for inactive particles. As these two steps are common bottlenecks in particle simulation, significant speedup were achieved.

Potential benefits of extension to Computer Graphics: Molecular dynamics often inspired particle-based simulations in Computer Graphics. The same bottleneck, namely inter-particles forces computation based on neighbor search, is present in the two fields, so we can expect interesting performance for ARPS in graphics. The remainder of this paper explores two applications of ARPS to graphical simulations:

1. Particle-based fluid simulation. In this case, damping

forces are involved in contrast with the classic use of ARPS. We propose a method to handle them as well as an incremental algorithm to update the forces and the scalar fields.
2. Stiff object simulation. We take the example of a cloth simulation. We will propose an implicit formulation of ARPS and a hybrid solver to exploit inactivity of particles.

It is clear that ARPS is not well-suited for simulations where all degree of freedom move: classical spatial adaptation is better suited in this case. In contrast, ARPS is best suited for simulations where most parts are immobile but may resume moving at any time. Even if these situations are not the most visually exciting, they are very common in Computer Graphics: they include simulation of characters clothing when many of the characters are at rest, surgical simulations with local user interaction, and the animation of large volumes of liquid, when most of it already came to rest.

4. Extension to SPH fluid simulation

SPH fluid simulation is widely used in computer graphics and many methods have been proposed [DC96], [MCG03], [SP09], [ICS*13]. SPH approximates fluid dynamics with a set of particles. The particles are used to interpolate properties of the fluid anywhere in the space. Each particle samples fluid properties such as density, pressure or temperature. All these properties are updated based on the particle neighbors and are used in short-ranged inter-particle forces. For a detailed and comprehensive introduction to SPH, you can refer to [Mon05]. To integrate ARPS, we chose WCSPH (Weakly Compressible Smoothed Particle Simulation) [BT07], a standard SPH formulation [DC96], [MCG03]. We limited our simulation to the main inter-particles forces: pressure and viscosity. Classically a SPH algorithm follows three steps:

1. Update mass density and pressure
2. Compute inter-particles forces : pressure, viscosity
3. Integrate velocities and positions

With ARPS, time can be saved on each computation step involving pairwise terms. In SPH, inter-particles forces and density field computation are the perfect candidates. As proposed in [AR12], we use an incremental algorithm to update only quantities involving active particles.

4.1. Viscosity

Viscosity forces involve particles velocities. The viscosity force of particle i with respect to particle j is :

$$f_{ij} = \begin{cases} -m_i m_j \Pi_{ij} \nabla W_{ij} & v_{ij}^T q_{ij} < 0 \\ 0 & v_{ij}^T q_{ij} \geq 0 \end{cases} \quad (5)$$

Π_{ij} is given as :

$$\Pi_{ij} = -\nu \left(\frac{v_{ij}^T q_{ij}}{| q_{ij} |^2 + \varepsilon h^2} \right) \quad (6)$$

W_{ij} denotes a convolution kernel, v_{ij} the difference of velocities between the two particles, q_{ij} the difference of positions between the two particles, m_i is the mass, h is the particle smoothing radius and $\nu = \frac{2\alpha h c_s}{d_i + d_j}$ is a viscous term where α is a viscosity constant, d_i the particle i density. $\varepsilon = 0.01$ is a constant to avoid singularities.

However, velocity is not explictly represented in ARPS, and can be seen in two different ways. We may define it based on the momentum and set $v_i = p_i / m_i$, or based on the change of position $\dot{q}_i$. In the first case, we can get time-varying forces even for inactive particles, which we want to avoid. We therefore use the effective velocity of the particle, as defined in eq.(4). Applied to a harmonic oscillator, this results in the behavior illustrated in Fig. 3. The more the particle is damped the longer it remains inactive, which is an intuitive behavior.

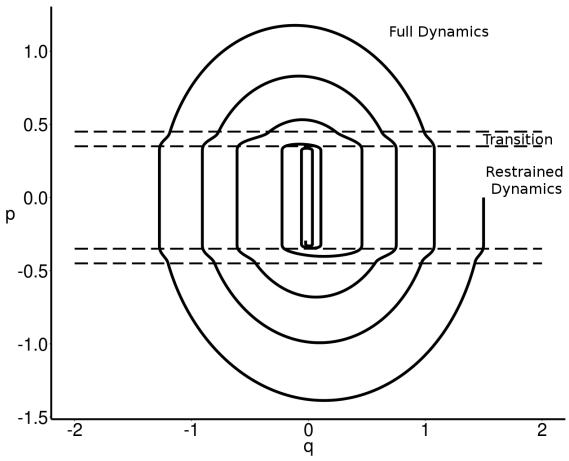
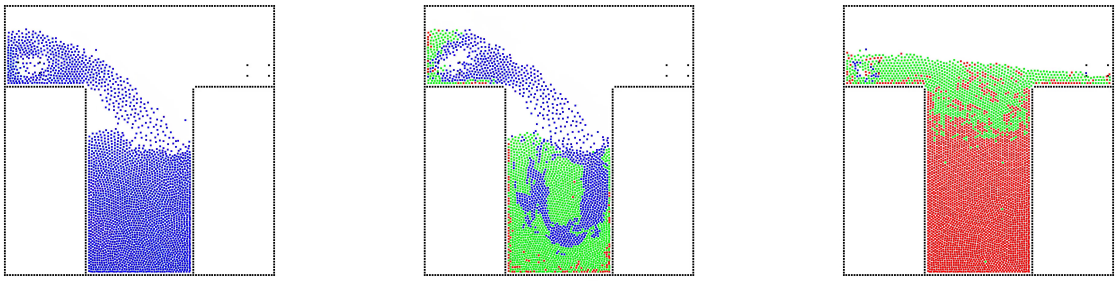


Figure 3: *Phase portrait of our damping approach in ARPS. As with a classic damped oscillator we obtain a spiral phase portrait.*

4.2. Modified inactivity criterion

Since our damping force vanishes along with the effective velocity of the particle, it drags down the kinetic energy asymptotically close to the inactivity threshold, without ever reaching it. Consequently, particles only subject to damping forces never become inactive, and we do not spare computation time, even when the particles get nearly static. To remedy this problem, we consider inactive the particles which effective velocity fall below a user-defined threshold.



(a) (b) (c)

Figure 4: *A permanent flow simulation with 4240 particles. (a) is a classic WCSPH simulation. (b) is our adaptive method at the same time of (a) with restrained particles in red. (c) is our adaptive method once the permanent flow is installed.*

4.3. Performance

We performed two experiments to measure computation time. The first one (Table 1) is a fall of 5000 particles in a box. As soon as most particles come to rest, and become inactive the speedup can be significant. For 15s, the mean speedup is 3.8. The speedup can locally reach 25.7. We can see in Figure 1 that during speed movements most of the particles are active so that the adaptive simulation stay close to reference simulation. Therefore small scale details like splashes can be preserved.

Simulation Time	SPH	ARPS	Speed-up
15s	893s	232s	{0.91, 25.73, 3.85}

Table 1: *Fall of a block of water - Computation time and speedup {min, max, mean}*

The second experiment (see Table 2) is the creation of a permanent flow with 4240 particles. As we can see in Figure 4, once the permanent flow is installed a large amount of particles are restrained. We reach an interesting speedup while keeping a motion close to the reference.

Simulation Time	SPH	ARPS	Speed-up
30s	2166s	814s	{0.83, 3.99, 2.66}

Table 2: *Fluid permanent flow - Computation time and speedup {min, max, mean}.*

5. Extension to stiff objects: Implicit Integration

In this section we explore the application of ARPS to stiff object simulation and propose an implicit integration scheme which saves computation time for particles at rest. Implicit integration for cloth simulation was introduced in [BW98]. An introduction to implicit integration is proposed in [WBK01]. While originally formulated on velocity, it can be straightforwardly expressed on momentum. Instead of integrating the momentum using the forces at the current time step, implicit integration uses the forces at the end of the current step. As we do not know these forces we end up with a non linear function and after linearization with a linear system to solve to obtain the next momentum:

$$(I - h^2 K M^{-1})\Delta p = h(f + hKM^{-1}p)\,, \tag{7}$$

where $K = \dfrac{\partial f}{\partial q}$ is the stiffness matrix and M is the mass matrix. Solving the linear system is more costly than explicit integration, but it allows the use of larger time steps without any loss of stability, enabling to advance much faster.

5.1. ARPS Implicit Integration

We derive an implicit integration scheme from Adaptively Restrained equations of motion. The linear system has to take into account the state of the particles. The discrete equations of motions for implicit Euler are:

$$\begin{aligned} \Delta p &= h f(q_{n+1}, p_{n+1}) \\ \Delta q &= h\left(M^{-1}(1-\rho(p_{n+1}))p_{n+1} \right. \\ &\left. -\frac{1}{2} p_{n+1}^T M^{-1} \frac{\partial \rho(p_{n+1})}{\partial p} p_{n+1}\right) \end{aligned} \tag{8}$$

We perform a Taylor-Young expansion of $f(q_{n+1}, p_{n+1})$ and introduce Δq in the expended momenta equation. We then perform a Taylor-Young expansion of $\rho(p_{n+1})$ in the momentum equation, which gives us the following equation system:

$$(I - h^2 K R M^{-1})\Delta p = h(f + hKM^{-1}s) \tag{9}$$

R is a block-diagonal matrix where each 3×3 block R_{ii} is:

$$R_{ii} = I - \rho(p_n^i) - p_n^i \frac{\partial \rho(p_n^i)}{\partial p_i}^T - \frac{1}{2} p_n^i p_n^{iT} \frac{\partial^2 \rho(p_n^i)}{\partial p_i^2}^T - \frac{\partial \rho(p_n^i)}{\partial p_i} p_n^{iT}, \quad (10)$$

while s is a $3N$ vector where N is the number of particles, and each s_i is :

$$s_i = p_n^i - \rho(p_n^i) p_n^i - \frac{1}{2} p_n^{iT} p_n^i \frac{\partial \rho(p_n^i)}{\partial p_i} \quad (11)$$

Note that if all particles are inactive then we have $R = 0$ and $s = 0$ and we get an explicit formulation:

$$I\Delta p = hf \quad (12)$$

Conversely, if all particles are active then $R = I$ and $s = p$ and we get the classical implicit formulation of eq.(7). We loop over time using algorithm 1.

Algorithm 1 Implicit integration scheme

for each time step **do**
  compute ρ, R, s, f.
  compute $A = I - h^2 KRM^{-1}$
  compute $b = hf + h^2 KM^{-1} s$
  solve $A\Delta p = b$
  compute $p_{n+1} = p_n + \Delta p$
  compute $q_{n+1} = q_n + hM^{-1}(R\Delta p + s)$
end for

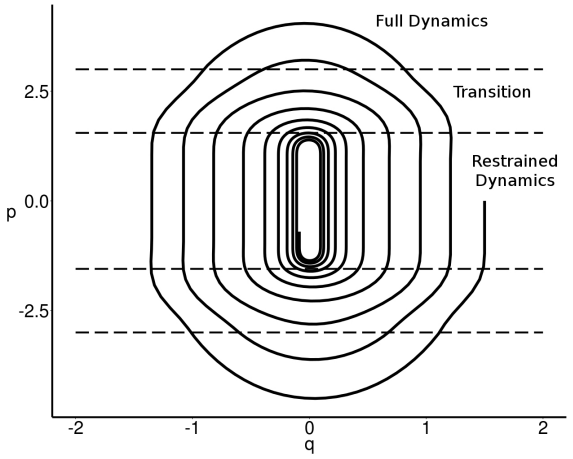


Figure 5: *Phase portrait of a harmonic oscillator simulated using implicit ARPS.*

Figure 5 shows the phase portrait of a harmonic oscillator simulated using our implicit formulation. As expected, the well-known numerical damping effect of implicit Euler provides us with the same behavior we could observe with a damped harmonic oscillator. To include a damping term in the physical model, we derived an implicit formulation which includes a damping term $f_d = -\gamma \dot{q}$:

$$(I + h\gamma M^{-1} R - h^2 KRM^{-1})\Delta p = h(f + hKM^{-1} s + f_d) \quad (13)$$

Solving the equation: We exploit inactive particles to save computation time. As discussed earlier, inactive particles can be handled using explicit integration, which is much simpler. When a particle is inactive and has no active neighbors we do not need to include it in the linear system. We thus build the minimal linear system, which only contains active particles and their neighbors. These particles are implicitly integrated, while the others are explicitly integrated. Figure 6 shows a hanging cloth with active and inactive particles. At the beginning all the particles become active. Then a moving front of inactivation/reactivation traverses the cloth at decreasing frequency. The cloth finally finds a rest position, where all the particles are inactive and simulated explicitly, saving computation time. The particles can become active again if external forces or imposed motion are applied. Table 3 shows performances we achieved with our hybrid solver. As soon as a large number of particles become inactive the simulation is explicitly integrated and interesting speedup can raise.

Simulation Time	Implicit	Hybrid	Speed-up
20s	16.9s	6.2s	{0.77, 15.16, 2.73}

Table 3: *Implicit solver vs Hybrid solver. Computation time and speedup {min, max, mean}.*

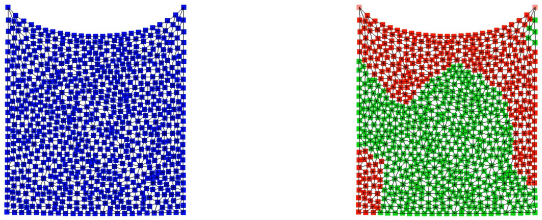


Figure 6: *Hanging cloth. Left: traditional implicit simulation. Right: implicit ARPS simulation with a varying set of active and inactive particles.*

However, while smoothly varying external forces are well handled by our simulator, we noticed instabilities when interacting strongly with the model. They seem to occur during the transition between the transitive and the full-dynamics states. A more thorough study of the influence of the transition function ρ on the stability of the system would be necessary to come up with robust implicit ARPS simulations.

This transition should be really well taken to avoid any instabilities.

6. Implementation

6.1. Parameters

ARPS use two parameters, ε^r and ε^f of Equation 3. The main goal of ARPS in computer graphics is to save time when nothing happens. So we generally want a low ε^r not to miss interesting movements. When sudden movements occur, we want a normal reaction, so we want the inactive particles to quickly become active. This requires a short transition, *i.e.* ε^f close enough to ε^r. However, due to discrete time integration, a short transition may be stepped over, or not enough sampled, which may result in instabilities. Currently we manually set the parameters, and defer the automatic tuning to future work. In table 4 we refer the thresholds used in our simulations.

	ε^r	ε^f	Tolerance
SPH	1-e6	2-e5	8e-5
Cloth	0.05	1	1e-4

Table 4: *ARPS thresholds for SPH and Cloth simulation*

6.2. Linear solver

A linear equation solver is necessary in implicit integration, as presented in Section 5. In contrast with most formulations, implicit ARPS generally results in an unsymmetrical equation matrix, due to the matrix products in eq.(9). We currently use a sparse LU solver from umfpack library, but it would be interesting to try a Conjugate Gradient method for unsymmetrical matrices to control the computation time, as it is usually done in implicit integration.

6.3. Choice of the restraining function and criterion

In ARPS the restraining function is a 5^{th}-order spline. The spline directly depends on particle kinetic energy which is the *restraining* criterion. The implicit solver involves second derivatives of the restraining function, which may have large values, leading to instabilities. We found that controlling the state of the particles based on momenta norm rather than kinetic energies seems to mitigate this and lead to more stable simulations. We plan to investigate this issue in future work.

7. Discussion and concluding remarks

We have shown that ARPS, a new, simple approach to adaptive simulation, can effectively be applied to Computer Graphics, and we have demonstrated two specific applications. The most successful one is the SPH simulation, for which we have obtained significant speedups with only minor changes to the original simulation method. In the case of stiff material, we have obtained promising results for implicit integration, and we will address stability issues in future work, starting with a careful study of the restraining function.

Another interesting avenue is to employ non-physically-based transition criteria. The current one, based on kinetic energy, is well adapted to molecular dynamics simulation. In Computer Graphics, however, we are more interested in visual results. In future work, we thus plan to investigate the tuning of the transition thresholds based on visibility or distance to the camera, to even more focus the computational power where it most contributes to the quality of the result.

8. Acknowledgement

This work was partly supported by ERC advanced grant EXPRESSIVE.

References

[APKG07] ADAMS B., PAULY M., KEISER R., GUIBAS L. J.: Adaptively sampled particle fluids. *ACM Trans. Graph. 26*, 3 (July 2007). 2

[AR12] ARTEMOVA S., REDON S.: Adaptively restrained particle simulations. *Phys. Rev. Lett. 109* (Nov 2012), 190201. 2, 3, 4

[ATW13] ANDO R., THÜREY N., WOJTAN C.: Highly adaptive liquid simulations on tetrahedral meshes. *ACM Trans. Graph. (Proc. SIGGRAPH 2013)* (July 2013). 2

[BD12] BENDER J., DEUL C.: Efficient cloth simulation using an adaptive finite element method. In *Virtual Reality Interactions and Physical Simulations (VRIPhys)* (2012), Bender J., Kuijper A., Fellner D., Éric Guérin, (Eds.). 2

[BT07] BECKER M., TESCHNER M.: Weakly compressible sph for free surface flows. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2007), SCA '07, Eurographics Association, pp. 209–217. 4

[BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 43–54. 5

[BWHT07] BARGTEIL A. W., WOJTAN C., HODGINS J. K., TURK G.: A finite element method for animating large viscoplastic flow. In *ACM Transactions on Graphics (Proc. SIGGRAPH)* (2007), vol. 26. 2

[CDA00] COTIN S., DELINGETTE H., AYACHE N.: A hybrid elastic model allowing real-time cutting, deformations and force-feedback for surgery training and simulation. In *The Visual Computer* (2000), vol. 16. 2

[DC96] DESBRUN M., CANI M.-P.: Smoothed particles: a new paradigm for animating highly deformable bodies. In *Proceedings of the Eurographics workshop on Computer animation and simulation '96* (New York, NY, USA, 1996), Springer-Verlag New York, Inc., pp. 61–76. 2, 4

[DDBC99] DEBUNNE G., DESBRUN M., BARR A. H., CANI M.-P.: Interactive multiresolution animation of deformable models. In *Eurographics Workshop on Computer Animation and Simulation'99, September, 1999* (Milan, Italie, Sept. 1999), Magnenat-Thalmann N., Thalmann D., (Eds.), Computer Science, Springer, pp. 133–144. 2

[DDCB01] DEBUNNE G., DESBRUN M., CANI M.-P., BARR A. H.: Dynamic real-time deformations using space and time adaptive sampling. In *Proc. ACM SIGGRAPH* (2001). 2

[GCS99] GANOVELLI F., CIGNONI P., SCOPIGNO R.: Introducing multiresolution representation in deformable object modeling. In *ACM Spring Conference on Computer Graphics* (1999). 2

[GKS02] GRINSPUN E., KRYSL P., SCHRÖDER P.: Charms: a simple framework for adaptive simulation. In *ACM Transactions on Graphics (Proc. SIGGRAPH)* (2002), vol. 21. 2

[GP11] GOSWAMI P., PAJAROLA R.: Time adaptive approximate SPH. In *Proceedings Eurographics Workshop on Virtual Reality Interaction and Physical Simulation* (2011). 2

[HPH96] HUTCHINSON D., PRESTON M., HEWITT T.: Adaptive refinement for mass/spring simulations. In *Eurographics Workshop on Computer Animation and Simulation* (1996), pp. 31–45. 2

[HVS*09] HARMON D., VOUGA E., SMITH B., TAMSTORF R., GRINSPUN E.: Asynchronous Contact Mechanics. *SIGGRAPH (ACM Transactions on Graphics) 28*, 3 (Aug 2009). 2

[IAGT10] IHMSEN M., AKINCI N., GISSLER M., TESCHNER M.: Boundary handling and adaptive time-stepping for pcisph. In *Proceedings of Vriphys* (2010). 2

[ICS*13] IHMSEN M., CORNELIS J., SOLENTHALER B., HORVATH C., TESCHNER M.: Implicit incompressible sph. *IEEE Transactions on Visualization and Computer Graphics 99*, PrePrints (2013), 1. 4

[KJ09] KIM T., JAMES D. L.: Skipping steps in deformable simulation with online model reduction. In *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* (2009), vol. 28. 2

[LGCM05] LENOIR J., GRISONI L., CHAILLOU C., MESEURE P.: Adaptive resolution of 1d mechanical b-spline. In *Proc. ACM GRAPHITE* (2005). 2

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2003), SCA '03, Eurographics Association, pp. 154–159. 4

[MKB*08] MARTIN S., KAUFMANN P., BOTSCH M., WICKE M., GROSS M.: Polyhedral finite elements using harmonic basis functions. In *Proc. Eurographics Symposium on Geometry Processing* (2008). 2

[Mon05] MONAGHAN J. J.: Smoothed particle hydrodynamics. *Reports on Progress in Physics 68*, 8 (Aug. 2005), 1703–1759. 4

[NPO13] NARAIN R., PFAFF T., O'BRIEN J. F.: Folding and crumpling adaptive sheets. *ACM Transactions on Graphics 32*, 4 (July 2013), xxx:1–8. Proceedings of ACM SIGGRAPH 2013, Anaheim. 2

[NSO12] NARAIN R., SAMII A., O'BRIEN J. F.: Adaptive anisotropic remeshing for cloth simulation. *ACM Transactions on Graphics 31*, 6 (Nov. 2012), 147:1–10. Proceedings of ACM SIGGRAPH Asia 2012, Singapore. 2

[OGRG07] OTADUY M. A., GERMANN D., REDON S., GROSS M.: Adaptive deformations with fast tight bounds. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer animation* (2007). 2

[OK12] ORTHMANN J., KOLB A.: Temporal blending for adaptive sph. *Comp. Graph. Forum 31*, 8 (Dec. 2012), 2436–2449. 2

[PTVF92] PRESS W. H., TEUKOLSKY S. A., VETTERLING W. T., FLANNERY B. P.: *Numerical recipes in C (2nd ed.): the art of scientific computing*. Cambridge University Press, New York, NY, USA, 1992. 2

[RGL05] REDON S., GALOPPO N., LIN M. C.: Adaptive dynamics of articulated bodies. In *ACM Transactions on Graphics (Proc. SIGGRAPH)* (2005), vol. 24. 2

[SG11] SOLENTHALER B., GROSS M.: Two-scale particle simulation. *ACM Trans. Graph. 30*, 4 (July 2011), 81:1–81:8. 2

[SLNB11] SERVIN M., LACOURSIÈRE C., NORDFELTH F., BODIN K.: Hybrid, multiresolution wires with massless frictional contacts. In *IEEE Transactions on Visualization and Computer Graphics* (2011), vol. 17. 2

[SOG08] STEINEMANN D., OTADUY M. A., GROSS M.: Fast adaptive shape matching deformations. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2008). 2

[SP09] SOLENTHALER B., PAJAROLA R.: Predictive-corrective incompressible sph. In *ACM SIGGRAPH 2009 papers* (New York, NY, USA, 2009), SIGGRAPH '09, ACM, pp. 40:1–40:6. 4

[SSIF07] SIFAKIS E., SHINAR T., IRVING G., FEDKIW R.: Hybrid simulation of deformable solids. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2007). 2

[ST08] SPILLMANN J., TESCHNER M.: An adaptive contact model for the robust simulation of knots. In *Computer Graphics Forum (Proc. Eurographics)* (2008), vol. 27. 2

[WBK01] WITKIN A., BARAFF D., KASS M.: Physically based modeling. In *Online SIGGRAPH Course Notes* (2001). 5

[WDGT01] WU X., DOWNES M. S., GOKTEKIN T., TENDICK F.: Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes. In *Computer Graphics Forum (Proc. Eurographics)* (2001), vol. 20. 2

[WRK*10] WICKE M., RITCHIE D., KLINGNER B. M., BURKE S., SHEWCHUK J. R., O'BRIEN J. F.: Dynamic local remeshing for elastoplastic simulation. In *ACM Transactions on Graphics (Proc. SIGGRAPH)* (2010), vol. 29. 2

[WT08] WOJTAN C., TURK G.: Fast viscoelastic behavior with thin features. In *ACM Transactions on Graphics (Proc. SIGGRAPH)* (2008), vol. 27. 2