Universidad
Rey Juan Carlos

Escuela Técnica Superior De Ingeniería Informática

Máster Universitario en Informática Gráfica, Videojuegos y Realidad Virtual

Curso Académico 2019/2020

Trabajo fin de Máster

Solvers de Elasticidad y Restricciones en Simulación

Autora: Clara Peñalva Carbonell

Director: Miguel Ángel Otaduy Tristán

Resumen

En este proyecto se presenta un estudio centrado en los *solvers* de físicas modernos *Projective Dynamics* para acelerar el cálculo de deformaciones, realizando una comparativa con los modelos *Position-Based Dynamics* y tomando de referencia el método de integración *Backward Euler Implícito*. Para ello, se realiza un análisis teórico de los diferentes métodos y se implementan, prestando atención al desarrollo de las formulaciones dependientes del escenario que se simule. Finalmente, se presenta una batería de pruebas que pretenden dar respuesta sobre la idoneidad de cada *solver* en problemas estáticos y dinámicos.

Abstract

This project presents a study focused on modern physics solvers Projective Dynamics to accelerate the calculation of deformations, making a comparison with the Position-Based Dynamics models and taking as reference the Implicit Backward Euler integration method. Therefore, a theoretical analysis of the different methods is carried out and implemented, paying attention to the development of formulations that depends on the scenario that is simulated. Finally, a battery of tests is presented that aim to provide an answer on the suitability of each solver in static and dynamic problems.

Índice general

Índice de figuras

Índice de tablas

Índice de algoritmos

1. Preámbulo

1.1 Motivación

Hoy en día, hay una gran cantidad de métodos de integración disponibles para resolver simulaciones por computador. Algunos requieren de horas e incluso días para completarse, mientras que otros suceden en milésimas de segundo. Dada la gran cantidad de opciones, un problema común en el desarrollo de aplicaciones de simulación reside en la selección de alguno de estos algoritmos, lo que requiere un análisis de las ventajas y defectos de cada uno.

No existe un único método capaz de abarcar todos los usos posibles que requiere el mercado. Por ejemplo, en algunas situaciones se requieren simulaciones que trabajen en tiempo real, pudiendo o no sacrificar la precisión, otras necesitan poder representar una mayor cantidad de tipos de materiales, etc. En definitiva, existen multitud de escenarios posibles y una amplia gama de métodos de integración que ofrezcan unas características u otras.

Este trabajo pretende centrarse en el análisis de este problema para integradores en tiempo real, es decir, que ejecutan un tiempo de simulación comparable al tiempo real transcurrido en el cómputo, con una frecuencia mínima de 24Hz.

1.2 Objetivos

El proyecto busca explorar los *solvers* de físicas modernos *Projective Dynamics* para acelerar el cálculo de deformaciones, realizando una comparativa con los modelos *Position-Based Dynamics* y tomando de referencia el método de integración de Newton. Por ello, por una parte, como finalidad principal destaca:

- Implementar el sistema masa-muelle sobre el método de *Backward* Euler Implícito con varias iteraciones resolviendo el sistema por Newton-Raphson para recoger los valores de referencia.

- Implementar el sistema masa-muelle sobre métodos con enfoque basado en restricciones (PBD) y sobre métodos basados en *Projective Dynamics*.
- Diseñar, ejecutar y analizar las distintas pruebas de evaluación de los métodos:
    - Analizar su adecuación a la simulación de problemas estáticos y dinámicos.
    - Estudiar su precisión con respecto a una referencia.

Por otra parte, podemos enumerar otros objetivos secundarios, que servirán para llevar a cabo los expuestos anteriormente:

- Realizar un estudio previo de los métodos para adecuar sus parámetros con el fin de que los resultados de las pruebas sean analizables.
- Confeccionar un sistema para realizar la recogida de datos de las diversas pruebas de cada método.
- Desarrollar un flujo de trabajo unificado con el que visualizar las simulaciones de todos los métodos.

1.3 Estructura

A fin de alcanzar los objetivos fijados, la organización de este trabajo se divide en cinco partes: Preámbulo, Estado del Arte, Aplicación de los métodos, Estudio y discusión de resultados y Conclusiones. Para orientar en la lectura de este trabajo, a continuación se van a describir los contenidos de cada capítulo:

- Capítulo 1: Pone en contexto el proyecto, exponiendo las motivaciones para la elección del tema y especificamos los objetivos que se quieren conseguir.
- Capítulo 2: Realizaremos un estudio teórico de los *solvers* escogidos. Contendrá información relacionada con el proyecto y que es de utilidad para su comprensión.
- Capítulo 3: Estableceremos las características del escenario experimental y especificaremos cómo debemos adaptar cada *solver*.
- Capítulo 4: Detallaremos las métricas a seguir para obtener los datos de las pruebas, mostraremos los resultados y los razonaremos.
- Capítulo 5: Realizaremos una recapitulación del desarrollo del trabajo, haciendo hincapié en las dificultades encontradas y los resultados obtenidos.

2. Estado del arte

2.1 Introducción

La animación basada en la física está ampliamente extendida como una cuestión principal en el ámbito de los gráficos por computación. Hoy en día se utiliza en sectores tan dispares como la industria del cine, los videojuegos, simuladores de entrenamiento como, por ejemplo, operarios de grúas portuarias o médicos cirujanos. Por ello, existen una gran cantidad de modelos de simulación que satisfacen distintas necesidades del mercado. En este capítulo nos centraremos en proporcionar las bases conceptuales para entender las diferencias y similitudes entre los distintos los métodos de integración escogidos.

En primer lugar, es útil comprender las partes en las que se dividen tradicionalmente los algoritmos de simulación física por computador. Estos sistemas poseen una serie de parámetros constantes que definen las características de la evolución del problema. Entre ellos, el más relevante es el *paso de tiempo*. Este determina el intervalo de tiempo que avanza el mundo virtual. Usualmente, los *solvers* tienen dos pasos conceptuales:

1. La recopilación de datos necesarios para la resolución del sistema.
2. La resolución del sistema.

Es importante destacar que en el segundo paso, la resolución del sistema, cualquier *solver* va a ser capaz de efectuar una cantidad, arbitraria o predefinida, de iteraciones internas que aumentan la precisión del sistema. Independientemente de la cantidad de trabajo, al final de cada paso el *solver* habrá cubierto el *paso de tiempo* (dt) establecido.

Estos sistemas de resolución pueden tratar con diferentes modelos materiales que representan con mayor o menor fidelidad sustancias reales. Un ejemplo es la discretización de masa-muelle, donde se divide el material a simular en nodos de masa que están conectados entre ellos por *muelles* virtuales de diversas rigideces. Puesto que este modelo de discretización está ampliamente extendido en las aplicaciones en tiempo real, vamos a enfocar las explicaciones de los métodos de integración desde esta visión.

Una de las características centrales de múltiples modelos es una métrica que define la deformación de estos *muelles*. Como veremos a continuación, podemos encontrar dos interpretaciones de esta medida, una con sentido físico y otra con sentido matemático.

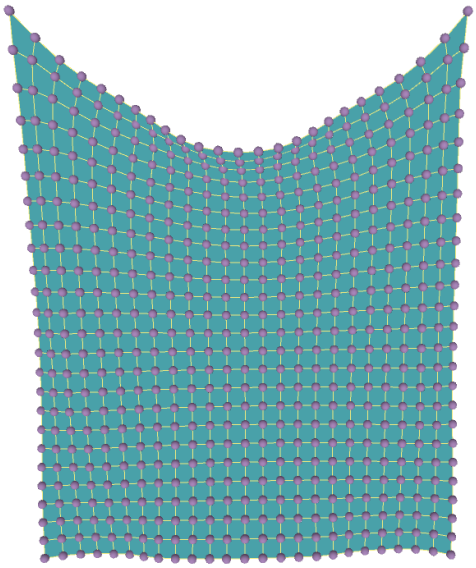

Figura 1. Tela simulada utilizando una discretización masa-muelle.

Al hablar de estos muelles abstractos, se debe explicar cómo responden ante las deformaciones. En su caso más sencillo, podemos hablar de la métrica de deformación del muelle como:

$$C = |x_1 - x_2| - l_0 \qquad 1$$

Donde x_1 y x_2 son los dos extremos del muelle y l_0 es su longitud de reposo. Si se presta atención en detalle a esta expresión, se puede ver cómo, independientemente de la dimensionalidad y orientación del muelle, C aumenta cuanto más lejos está la distancia de las posiciones de su reposo. Por lo tanto, un sistema de masa-muelle poco deformado, tendrá C_i bajas, y un sistema muy deformado, altas.

En la mecánica clásica el uso principal de C radica en definir Energías. Dentro de su lenguaje, C es referido como deformación y en el caso simple de un muelle, tomando la ley de Hooke, se define esta energía como:

$$E = 1/2 \cdot k \cdot C^T \cdot C \qquad 2$$

Donde k es la rigidez del muelle. Esta energía, de manera similar a C, también define la medida de desviación de un sistema con respecto a su estado de reposo. Además, permite

derivar otras dos magnitudes: la Fuerza y la Hessiana, vectorial y matricial respectivamente.

$$F = -\frac{\partial E}{\partial x} \quad (3)$$

$$H_x = -\frac{\partial^2 E}{\partial x^2} \quad (4)$$

Donde x son las posiciones y *v* las velocidades de los diferentes nodos del problema, dispuestas como un solo vector columna. De manera abstracta, son dos métricas que hablan de las direcciones en las que el sistema ha de evolucionar para minimizar sus energías y, por tanto, estar cerca del estado de reposo.

Llegados a este punto, vamos a introducir el *solver* que consideraremos como *ground truth* durante todo el trabajo, *Backward* Euler (Baraff, 1997), (Baraff & Witkin, 1998). Este método se caracteriza por su robustez. Primero debemos recordar que queremos simular un objeto deformable y, por tanto, partimos de la segunda ley de Newton para simular dinámica clásica:

$$\sum F_i \; = \; m \cdot a \quad (5)$$

Donde *F* son las fuerzas del sistema, *m* es la masa y a es la aceleración. Queremos averiguar las posiciones que tendrá el sistema a lo largo del tiempo, considerando una discretización de éste en intervalos (*dt*), cumpliendo la ecuación anterior. Además, suponemos que conocemos el estado inicial del sistema, x_0 y v_0. Por lo que debemos establecer una ecuación diferencial que defina las posiciones $f(x)$. Si seguimos el desarrollo de Tylor al revés, tenemos:

$$x(t) = x(t+dt) - v(t+dt) \cdot dt, \qquad x(t+dt) \; = \; x(t) \; + \; v(t+dt) \cdot dt \quad (6)$$

$$v(t) = v(t+dt) - a(t+dt) \cdot dt, \qquad v(t+dt) = v(t) \; + \; a(t+dt) \cdot dt \quad (7)$$

Queda por integrar en este sistema la ecuación de la 2ª Ley de Newton. Si despejamos en ella la aceleración, vemos que podemos remplazar este término en la ecuación de la velocidad, quedando el sistema así:

$$v(t+dt) = v(t) + M^{-1}F\big(x(t+dt)\big) \cdot dt \qquad 8$$

$$x(t+dt) \;=\; x(t) \;+\; v(t+dt) \cdot dt \qquad 9$$

A continuación, debemos linealizar la fuerza del sistema. Utilizamos el método Newton-Raphson, representando $x(t+dt) - x(t) \;=\; \Delta x$:

$$F(x(t+dt) = F(x(t) + \frac{dF}{dx}\,(\Delta x) \qquad 10$$

Y, sustituyendo en la ecuación de velocidad de *Backward* Euler (8):

$$v(t+dt) = v(t) + M^{-1}F(x(t) \cdot dt + \frac{dF}{dx}\,(x(t+dt) - x(t)) \cdot dt \qquad 11$$

Como aparece el cálculo de $x(t+dt)$, lo sustituimos por la Ecuación 9, multiplicamos los términos por *M* para eliminar el cálculo de matrices inversas y reordenamos:

$$\left(M - dt^2 \cdot \frac{dF}{dx}\right) \cdot v(t+dt) = M \cdot v(t) + dt \cdot F(x(t)) \qquad 12$$

Finalmente, tenemos un sistema a resolver de la forma $A \cdot v(t+dt) \;=\; b$, que podemos resolver mediante factorización de Cholesky, gradiente conjugado o similares. Resuelta la velocidad, para conocer la posición simplemente debemos calcular la Ecuación 9.

Como se observa, aquí es donde entran en juego las fuerzas (Ecuación 3) y la Hessiana descrita en la Ecuación 4 . Una vez se resuelve este sistema de ecuaciones lineales tantas veces como iteraciones se hayan determinado, el paso acaba y las posiciones del sistema se desplazan las cantidades calculadas. Cabe señalar que la evaluación de las matrices Hessianas y la resolución del sistema lineal en este método conllevan un alto coste computacional, puesto que se calculan en cada iteración.

Hasta ahora hemos visto cómo utilizar C como una deformación que a su vez define una energía E. Sin embargo, existe otra manera de hacer cumplir la minimización de las

diferentes C_i: las restricciones. De forma simple, una restricción es toda propiedad que se desee conservar (Bender, et al., 2017).

Siguiendo con la discretización de masa-muelle, es posible abordar este problema partiendo de la misma propiedad constante C y reformular la fuerza a partir de la energía de restricción para simular un sistema dinámico con restricciones. En este sentido, se pueden integrar de igual manera que en el ejemplo anterior: se crea un sistema lineal, donde se necesitan las Hessianas, pero en este caso basándose en restricciones. Esta formulación de restricciones se conoce como restricciones débiles, ya que no existe una garantía de que se cumplan. Es decir, diferentes fuerzas y restricciones competirán en un mismo paso con intensidad proporcional a sus respectivas rigideces, y se concluirá con las posiciones en un punto de "compromiso" que también dependerá de las velocidades.

Por otra parte, existen las llamadas restricciones fuertes donde se parte de técnicas de optimización, que hacen uso de los multiplicadores de Lagrange, para minimizar el sistema lineal. (Witkin, 1997). Estas técnicas amplían la dimensionalidad del problema, añadiendo un grado de libertad por cada restricción.

En suma, C define una manera de medir cómo de deformado se encuentra un elemento, por ejemplo, un muelle virtual en el caso de una discretización de masa-muelle. Pero también podría ser cualquier otra propiedad a conservar. Esta medida puede integrarse como Fuerzas y Hessianas a través de Energías, o como restricciones. Y en ambas perspectivas se plantea un problema que puede estudiarse como un problema de optimización. Este es el funcionamiento general de los *solvers*, y en la siguiente sección pasaremos a discutir las características específicas de cada uno de los que se estudian en este trabajo, mostrando sus similitudes y diferencias.

2.2 Position Based Dynamics

Como hemos visto en el apartado anterior, una aproximación muy extendida para simular sistemas dinámicos está basada en las fuerzas, siguiendo la mecánica continua, como hace el método de Newton-Raphson sobre el integrador *backward* Euler. Sin embargo, los métodos basados en la posición (*Position-Based*, en inglés) siguen otra filosofía y omiten las velocidades y aceleraciones, modificando directamente las posiciones. Estos métodos se introdujeron como una alternativa para simular objetos deformables consiguiendo resultados visualmente plausibles con mayor eficiencia.

Siguiendo la dinámica de masa-muelle y el enfoque de restricciones, ambos introducidos en la sección 2.1, el algoritmo *Position-Based Dynamics* (PBD) define a los objetos como un conjunto de nodos y de restricciones. Además, cada restricción tiene un parámetro de rigidez asociado que define la fuerza de la restricción en un rango de cero a uno. Estos parámetros se encuentran prefijados y se enmarcan dentro de la primera fase conceptual que habíamos establecido en la sección 2.1. A continuación, se muestra el algoritmo completo, omitiendo las restricciones de colisión:

Algoritmo 1: bucle de simulación de *Position Based Dynamics*

1: $v_i = v_i + \mathrm{dt} \cdot w_i \cdot f_{\mathrm{ext}}(x_i)$
2: $p_i = x_i + dt \cdot v_i$
3: **mientras** i < iteracionesSolver **hacer**
4: **para todas** restricciones **hacer**
5: computar Δx usando la Ecuación 14
6: $x_{i+1} = x_i + \Delta x$
7: **fin para**
8: $i = i + 1$
9: **fin mientras**
10: $v_i = (p_i - x_i)/dt$
11: $x_i = p_i$

Si nos fijamos en el algoritmo anterior, entrando en la resolución del sistema (líneas 5-6), vemos cómo se refleja que utilizando la visión fundamentada en restricciones y la discretización basada en nodos conectados, el problema queda simplificado a la proyección de cada restricción que conforma el sistema a simular:

$$s_j = \frac{-C_j(x_i)}{\nabla C_j \, M^{-1} \nabla C_j^T} \qquad 13$$

$$\Delta x = s_j M^{-1} \nabla C_j(x_i) \qquad 14$$

Donde *j* es el índice de la restricción y *M* es la matriz de masas de los nodos que conformen la restricción. Esta resolución en lugar de resolver todas las restricciones como un todo, proyecta cada restricción y las resuelve por separado. Esto está basado en el algoritmo "Gauss-Seidel no lineal" porque se asemeja a las iteraciones de Gauss-Seidel para resolver sistemas lineales. Por otro lado, en las ecuaciones no se considera la rigidez (*k*) de la restricción. La forma propuesta por (Müller, et al., 2007) es multiplicar el resultado de Δx por *k*.

Como vemos, enfocando el mismo problema desde la perspectiva de minimizar el valor *C* a base de restricciones, la linealización del método se realiza individualmente por cada restricción. Esto permite que no nos encontremos ante un gran sistema para resolver, con el ahorro computacional que ello conlleva. Sin embargo, el principal problema del método PBD se encuentra en la dependencia entre la rigidez, el número de iteraciones y el *paso de tiempo*. Esto también está unido al hecho de que el valor de rigidez no tiene una base física dentro del método PBD. Por tanto, cuesta parametrizar correctamente los resultados para obtener materiales reales.

2.2.1 Extended Position-Based Dynamics (Macklin, et al., 2016)

El problema relacionado con el *paso de tiempo* y la rigidez y su dependencia con la cantidad de iteraciones en PBD se abordó con una extensión llamada *Extended Position-Based Dynamics* (abreviado a XPBD) (Macklin, et al., 2016). XPBD se deriva de una formulación de *compliant constraint* (Servin, et al., 2006) que asocia la inversa de la rigidez, o *compliance*, $\alpha = 1/k$ con cada restricción.

Se introduce el concepto de multiplicador de Lagrange a través de una nueva variable, λ. Estos multiplicadores se utilizan en técnicas para resolver problemas de optimización con restricciones. Conectándolo a las ecuaciones de movimiento se obtienen estas ecuaciones restringidas con el término *compliant*, $\alpha \cdot \lambda$.

$$\lambda = -\alpha^{-1}C(x) \tag{15}$$

$$\begin{gathered} Ma - \nabla C^T(x)\lambda = 0 \\ C(x) + \alpha\lambda = 0 \end{gathered} \tag{16}$$

Estas ecuaciones no lineales se discretizan en el tiempo, dando lugar a un sistema que se resuelve a través del método de Newton. Sin embargo, esta forma implica la realización de cálculos costosos como, por ejemplo, las Hessianas, mostrados en la sección 2.1. Por ello, se realizan diversos cambios de aproximación para simplificar la implementación, conectando así con el algoritmo de PBD. Finalmente, el sistema queda así:

$$[\nabla C(x_i)M^{-1}\nabla C(x_i)^T + \tilde{\alpha}]\Delta\lambda = -C(x_i) - \tilde{\alpha}\lambda_i \tag{17}$$

Y despejando, $\Delta\lambda_j$ para una restricción *j*:

$$\Delta\lambda = \frac{-C_j(x_i) - \tilde{\alpha}_j\lambda_{ij}}{\nabla C_j M^{-1} \nabla C_j^T + \tilde{\alpha}_j} \quad 18$$

Donde $\tilde{\alpha} = \alpha/\mathrm{dt}^2$. Como vemos, la variable *s* de PBD (Ecuación 13) es en realidad un multiplicador de Lagrange incremental en un caso infinitamente rígido. Y la actualización de la posición se define:

$$\Delta x = M^{-1}\nabla C(x_i)^T \Delta\lambda \quad 19$$

Estos son todos los cambios que se aplican sobre el método PBD, en el siguiente algoritmo se detalla el método XPBD y destacan en negrita las modificaciones en relación al Algoritmo 1.

Algoritmo 2: bucle de simulación de *Extended Position Based Dynamics*

1: predecir posición $\tilde{x} = x^n + dt \cdot v^n + dt^2 \cdot M^{-1} f_{ext}(x^n)$
2: se inicializa $x_0 = \tilde{x}$
3: se inicializan los multiplicadores $\lambda_0 = 0$
4: **mientras** i < iteracionesSolver **hacer**
5: **para todas** restricciones **hacer**
6: **computar $\Delta\lambda$ usando la Ecuación** 18
7: computar Δx usando la Ecuación 19
8: $\boldsymbol{\lambda_{i+1} = \lambda_i + \Delta\lambda}$
9: $x_{i+1} = x_i + \Delta x$
10: **fin para**
11: $i = i + 1$
12: **fin mientras**
13: $x^{n+1} = x_i$
14: $v^{n+1} = \frac{1}{dt}(x^{n+1} + x^n)$

Se observa que en caso de establecer el valor de *compliance* a cero, se obtiene la formulación de PBD. Cabe destacar que XPBD no hace que PBD converja más rápido, si el *solver* finaliza antes de llegar a la convergencia, el resultado mostrará un cumplimiento artificial de las restricciones. No obstante, XPBD devuelve una solución consistente que corresponde a una energía potencial bien definida y con iteraciones infinitas mantiene su comportamiento esperado.

2.3. Projective Dynamics

Una alternativa al método presentado anteriormente es *Projective Dynamics*, presentado para simulación por primera vez en (Liu, et al., 2013) que define un paso local intermedio en la integración. Este método posteriormente se generalizó para energías cuadráticas (Bouaziz, et al., 2014) y finalmente para todo tipo de modelos estructurales (Overby, et al., 2017). En esta sección se presentarán el primer y último métodos mencionados.

Estos métodos se nutren de la formulación de optimización de la integración de Euler Implícito (Martin, et al., 2011), que presenta la siguiente forma:

$$x^{n+1} = \min_{x}\left(\frac{1}{2dt^2}(x^n - \mathrm{y})^T M(x^n - \mathrm{y}) + E(x)\right) \qquad 20$$

Donde $y = x^n + dt\, v^n + dt^2 \mathrm{M}^{-1} f_{ext}$, que se corresponde a un estado predicho por la primera ley de Newton: movimiento sin la presencia de fuerzas internas, y *E(x)* es la energía del sistema. De forma intuitiva, se puede considerar el primer término como el "inercial", que tratará de atraer x hacia *y*. El segundo término opondrá su resistencia con deformaciones elásticas.

2.3.1 Fast Simulation of Mass-Spring Systems (Liu, et al., 2013)

Este método trata de encontrar una forma de acelerar las soluciones numéricas de simulaciones de objetos deformables considerando la formulación de optimización de la integración de Euler Implícito, descrita en la Ecuación 20. Pudiendo, así, resolverla aplicando un método de descenso de coordenadas de bloques (*block coordinate descent method*, en inglés). Esto es, alternando en el *solver* un paso local a cada muelle y un paso global.

Para ello, parte de un sistema ya conocido, masa-muelle, para discretizar los objetos, con energías que siguen la Ley de Hooke (Ecuación 2). En primer lugar, debemos resaltar que Liu introduce en la energía potencial elástica una variable auxiliar (d_j), transformándola en un problema de minimización restringido:

$$\frac{1}{2}k(|x_1 - x_2| - l_0)^2 = \min_{|d|-l_0}\frac{1}{2}k|(x_1 - x_2) - d|^2 \qquad 21$$

Donde k es la rigidez del muelle, x_1 y x_2 son los dos extremos del muelle y l_0 es su longitud de reposo. Además, *d* define $|d| = l_0$, lo que se podría considerar una restricción, transformando la anterior ecuación en:

$$\frac{1}{2}k(|x_1 - x_2| - l_0)^2 = \frac{1}{2}k(|x_1 - x_2| - |d|)^2 \quad (22)$$

Esta variable auxiliar *d* es la proyección de la diferencia entre los dos extremos $(x_1 - x_2)$ dentro de una esfera cuyo radio es la longitud en reposo del muelle l_0, podríamos llamarla "dirección del muelle" Esto nos será relevante más adelante.

Al reformular el problema original de minimización sin restricciones (Ecuación 20) en un problema de optimización con restricciones, la ecuación a resolver queda así:

$$\min_{x \in R^{3m}, d \in U} \frac{1}{2}x^T(M + dt^2 L)x - dt^2 x^T J d + x^T b \quad (23)$$

Donde *M* es la matriz de masas de todos los nodos, *L* es la Laplaciana ponderada por rigidez y *J* es la matriz de incidencia de todos los muelles. Además, *b* contiene el término inercial y las fuerzas externas. Podemos observar estas expresiones a continuación:

$$M = \begin{bmatrix} m_1 \cdot I & 0 & \dots & 0 \\ 0 & m_2 \cdot I & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & m_n \cdot I \end{bmatrix} \quad (24)$$

$$L = \left(\sum_{i=1}^{s} k_i \, D_i D_i^T\right) \otimes I \quad (25)$$

$$J = \left(\sum_{i=1}^{s} k_i \, D_i S_i^T\right) \otimes I \quad (26)$$

$$b = M \cdot x + f_{ext} \cdot dt^2 \quad (27)$$

Donde *I* es la matriz identidad con la misma dimensionalidad que cada nodo y $\otimes$ representa el producto de Kronecker, D_i es la matriz de incidencia de cada muelle y *S* es el indicador de muelle ($S_{i,j} = k_{i,j}$).

Finalmente, d es la variable de proyección auxiliar para cada muelle que actualiza las longitudes de los muelles a valores óptimos de longitud de reposo mientras mantiene sus direcciones. Al proyectar el vector d, toma la siguiente forma para cada muelle *i*:

$$d_i = l_0 \frac{x_1 - x_2}{|x_1 - x_2|} \quad (28)$$

Asimismo, para resolver la minimización sobre x, se debe encontrar el punto crítico de la derivada de la función. En este caso debemos resolverla fijando d, igualándola a 0 y derivando sobre x.

$$(M + dt^2 L)x = J \cdot d \cdot dt^2 - b \quad (29)$$

Vemos que la ecuación de *x* adquiere la estructura común de resolución de sistemas lineales *Ax* = *b*.

El siguiente algoritmo muestra el bucle de simulación de este modelo.

Algoritmo 3: bucle de simulación de *Fast Simulation of Mass-Spring Systems*

1 En tiempo de precálculo:
1.1: computar *M* usando la Ecuación 24
1.2: computar *L* usando la Ecuación 25
1.3: computar *J* usando la Ecuación 26
1.4: computar $A = M + dt^2 \cdot L$

2 En tiempo de ejecución:
2.1: computar *b* usando la Ecuación 27
2.2: **mientras** *i* < iteracionesSolver **hacer**
2.3: **para todas** energías **hacer**
2.4: computar *d* usando la Ecuación 28
2.5: **fin para**
2.6: computar *x* usando la Ecuación 29
2.7: $i = i + 1$
2.8: **fin mientras**

Por una parte, en el paso local, línea 2.4 del Algoritmo 3, se actualizan las direcciones óptimas de cada muelle (variable *d*). Al ser un cálculo independiente por cada muelle, es altamente paralelizable. Además, si nos fijamos en la ecuación anterior (23) se puede apreciar que fijando *x* para resolver *d*, la dificultad de la minimización se ve reducida, (Ecuación 28).

Por otra parte, en el paso global, líneas 2.1 y 2.6 del Algoritmo 3, se resuelve el sistema lineal para obtener las posiciones actualizadas. Cabe destacar aquí otra ventaja de este método, en la resolución del *solver* vemos que las matrices *M*, *L* y *J* son independientes a las posiciones (*x*) y a la variable *d*. Por tanto, se pueden precalcular y utilizar en todas

las iteraciones del método, permitiendo, así, un ahorro computacional. Además, se aprecia que el cálculo de *b* es invariable dentro de la optimización del problema y se calcula fuera de las iteraciones del *solver* de cada paso. (línea 2.1).

En conclusión, este método resuelve la simulación reformulando el problema, introduciendo una optimización con restricciones. Esto posibilita que la resolución se pueda realizar en dos pasos, local, proyectando *d* en los muelles, y global, resolviendo el sistema lineal. Además, permite el precálculo de parte de las matrices que conforman el sistema lineal. Todo ello con la certeza de que convergerá a la misma solución que el método Euler Implícito con muelles de Hooke.

2.3.2 ADMM ⊇ Projective Dynamics (Overby, et al., 2017)

El método propuesto por Overby utiliza el algoritmo de optimización del método de multiplicadores de dirección alterna (*Alternating Direction Method of Multipliers*, en inglés, y abreviado a ADMM) para la integración de fuerzas no-elásticas. A diferencia del método descrito en la sección anterior, este *solver* generaliza *Projective Dynamics* a cualquier fuerza conservativa, incluyendo materiales hiperelásticos generales y restricciones duras, conservando su velocidad, paralelización y robustez. En el caso especial de las fuerzas lineales, es idéntico a *Projective Dynamics*. (Overby, et al., 2017) ADMM se enmarca dentro de la categoría de algoritmos *primal-dual* de optimización. A grandes rasgos, permite descomponer la función en dos términos y realizar de forma alterna los pasos de optimización. Esta característica es útil si los dos términos tienen estructuras simples pero incompatibles.

En la sección anterior vimos cómo (Liu, et al., 2013) deducía que al utilizar la formulación de optimización de integración implícita (Ecuación 20), se podía paralelizar y acelerar la resolución. En este caso Liu lo aplicaba a un sistema masa-muelle, pero la característica que permite dividir la resolución en dos bloques (local y global) es también aplicable sobre otros sistemas y, por tanto, el problema presenta una estructura que permite resolverlo de manera eficiente utilizando ADMM.

En primer lugar, debemos introducir la forma que presenta ADMM:

$$\min_{x,z} = f(x) + g(z) \qquad 30$$

$$s.t.\ Ax + Bz = c$$

Además, el algoritmo introduce una variable dual u e itera las siguientes reglas de actualización:

$$x^{n+1} = \underset{x}{arg\,min}\left(f(x) + \frac{\rho}{2}|Ax + Bz^n - c + u^n|^2\right) \quad 31$$

$$z^{n+1} = \underset{z}{arg\,min}\left(g(x) + \frac{\rho}{2}|Ax^{n+1} + Bz - c + u^n|^2\right) \quad 32$$

$$u^{n+1} = u^n + (Ax^{n+1} + Bz^{n+1} - c) \quad 33$$

Si se desea conocer en profundidad el desarrollo de ADMM, remitimos al artículo (Overby, et al., 2017). La forma que adquiere el problema de optimización (Ecuación 20) en la estructura de ADMM (Ecuación 30) es la siguiente:

$$\min_{x,z}\left(\frac{1}{2dt^2}(x^n - x^{n+1})^T M(x^n - x^{n+1}) + U_*(z)\right) \quad 34$$

$$s.t. W(Dx - z) = 0$$

Donde, $U_*(z)$ es la suma de las diferentes energías, *z* es una nueva variable, que satisface $z = Dx$ y a través de ella se definirá la restricción descrita en la ecuación anterior. Cabe señalar que en el método de Liu, presentado en la sección 2.3.1, también se hace mención a una variable (*d*) que añade una restricción sobre el problema a resolver. De igual forma que en este método, Overby también proyecta las coordenadas locales sobre las restricciones.

Además, *D* se corresponde con la matriz de reducción de las energías. Como hemos comentado, una característica relevante de este problema es la capacidad para paralelizar y dividir cálculos. De esta propiedad se extrae la posibilidad de realizar el cómputo de las energías de forma independiente. La matriz *D* define las coordenadas locales de cada energía, esto es, los nodos a los que afecta una energía concreta. Finalmente, *W* es la matriz de ponderación de restricciones, ($W = \omega I$), siendo ω el peso que se escoja para representar el crecimiento de las energías con respecto a la restricción, el usual es $\omega = \sqrt{k}$.

Una vez definido el problema con ADMM, se escogen las reglas:

$$f(x) = \frac{1}{2dt^2}|x - \tilde{x}|_M^2 \quad 35$$

$$g(x) = U_*(z) \quad 36$$

$$A = WD \quad 37$$

$$B = -W \quad 38$$

$$c = 0 \quad 39$$

Con todo ello, se aplican las reglas mostradas sobre la Ecuación 34 y se obtienen las ecuaciones que resuelven el problema:

$$x^{n+1} = \underset{x}{arg\,min}\left(\frac{1}{2dt^2}|x - \tilde{x}|_M^2 + \frac{1}{2}|W(Dx - z^n + \bar{u}^n)|^2\right),$$

$$x^{n+1} = (M + dt^2 D^T W^T WD)^{-1} \cdot \left(M\tilde{x} + dt^2 D^T W^T W(z^n - \bar{u}^n)\right), \quad 40$$

$$(M + dt^2 D^T W^T WD) \cdot x^{n+1} = \left(M\tilde{x} + dt^2 D^T WW(z^n - \bar{u}^n)\right),$$

$$z_i^{n+1} = \underset{z_i}{arg\,min}\left(U_i(z_i) + \frac{1}{2}|W_i(D_i x^{n+1} - z_i + \bar{u_i}^n)|^2\right) \quad 41$$

$$\bar{u_i}^{n+1} = \bar{u_i}^n + D_i x^{n+1} - z_i^{n+1} \quad 42$$

Donde el índice *i* hace referencia al muelle i-ésimo y $\bar{u} = W^{-1}u$. Si nos fijamos en la última expresión de x^{n+1}, observamos que adquiere la forma *Ax = b* de resolución de sistemas lineales. Teniendo construidas las ecuaciones principales del sistema, queda definir el algoritmo donde se calculan:

Algoritmo 4: bucle de simulación de Integración implícita con ADMM

1 En tiempo de precálculo:

1.1: computar *M* usando la Ecuación 24
1.2: computar *D* según las características de las energías que se simulen
1.3: computar $W = \omega I$
1.4: computar dt2DWW = $dt^2 \cdot D^T \cdot W^T \cdot W$ (servirá en el cálculo de *x*)
1.5: computar $A = M +$ dt2DWW $\cdot D$ (servirá en el cálculo de *x*)

2 En tiempo de ejecución:

2.1: inicializar $x_n = x_{n-1} + v_{n-1} \cdot dt + M^{-1} \cdot f_{ext} \cdot dt^2$
2.2: inicializar u = 0
2.3: **mientras** *i* < iteracionesSolver **hacer**
2.4: **para todas** energías **hacer**
2.5: computar *z* usando la Ecuación 41
2.6: computar *u* usando la Ecuación 42
2.7: **fin para**
2.8: computar *x* usando la Ecuación 40

2.9: $i = i + 1$
2.10: **fin mientras**
2.11: computar $v_n = (x_n - x_{n-1}) \cdot (1/dt)$

En relación al algoritmo, vemos que sigue una estructura similar al propuesto por (Liu, et al., 2013) en el apartado anterior (Algoritmo 3). En primer lugar, se distingue la distinción entre la sección de precálculo y cálculo en tiempo de ejecución. Además, dentro de la segunda sección, se diferencian los cálculos locales de las energías y las actualizaciones globales a todo el sistema.

En definitiva, simular objetos con la formulación de optimización de Euler Implícito con ADMM permite mejorar las prestaciones que mostraba el método de (Liu, et al., 2013), puesto que este se veía restringido a el modelo masa-muelle. Utilizando la formulación presentada por Overby el método es capaz de introducir diferentes tipos de fuerzas, tanto lineales como no lineales, así como restricciones duras y colisiones (Overby, et al., 2017).

2.4 Conclusiones

Hemos presentado métodos que realizan simulaciones en tiempo real, por una parte, modelos de *position-based* y, por otra parte, modelos de *projective dynamics*. Vemos que este problema, partiendo de propiedades a conservar, *C*, se pueden definir energías o restricciones. Además, en ambas se puede considerar la resolución de la simulación dinámica como una optimización.

En los siguientes capítulos abordaremos estos *solvers* aplicándolos a un escenario concreto y estudiaremos su interés, pudiendo compararlos analizando sus fortalezas y desventajas en relación al resto de métodos.

3. Aplicación de los métodos

Los métodos presentados en el capítulo anterior son útiles para formar una imagen del estado del arte. Sin embargo, también es necesario discutir sobre bajo qué circunstancias se van a someter a prueba los diferentes algoritmos. En este capítulo estableceremos las características de una escena de pruebas común que sirva como nexo comparativo, y examinaremos las especificidades del problema para cada uno de los métodos. Dado su interés en el paradigma de uso actual, compararemos XPBD, *Fast Simulation of Mass-Spring Systems* y ADMM ⊇ *Projective Dynamics*, teniendo en mente que el método implícito de Newton puede hacerse valer como referencia de base.

3.1 Escena de pruebas

Una vez especificados los métodos a comparar, se han de definir requisitos para las pruebas. Teniendo en cuenta que los experimentos tendrán una alta cantidad de repeticiones, y que se desean unos resultados consistentes, la lógica apunta a reducir la dimensionalidad del problema a la mínima expresión que permita manifestar fenómenos físicos trascendentes. Por estos motivos se ha concluido que una cuerda de 10 nodos fijada por sus dos extremos es un buen escenario que permite obtener los datos considerados relevantes en el Capítulo 4.

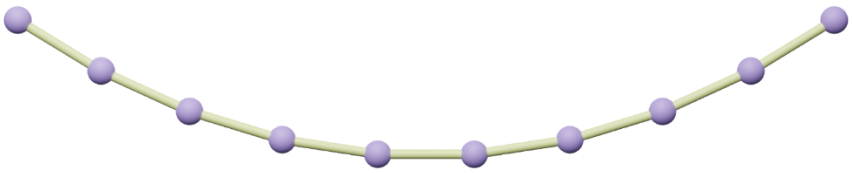

Figura 2. Catenaria discretizada con 10 nodos.

Además, la selección de una prueba de este tipo nos permite basarnos en una forma mecánica bien estudiada en ingeniería y arquitectura: la catenaria. Esta forma se define precisamente como la tomada por una cadena cuando cuelga por sus dos extremos y es empujada por la gravedad. Como detalle, esta curva es usada en ingeniería y arquitectura para encontrar los "caminos" de transmisión de fuerza óptimos de las estructuras, y ahorrar materiales estableciendo estructuras de soporte con la misma forma invertida.

En la escena que hemos diseñado, se sigue la misma estructura interna de la cuerda para todos los *solvers*. Se establece un muelle por cada par de nodos consecutivos:

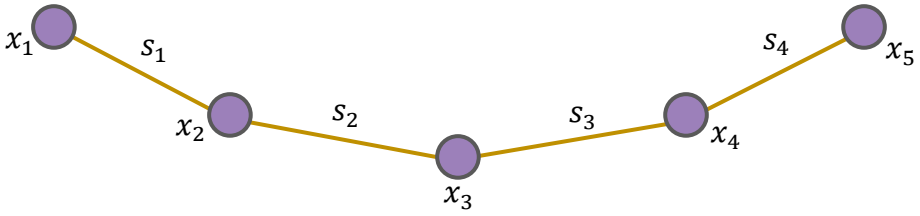

Figura 3. Esquema de la estructura interna de la cuerda.

Siguiendo esta discretización, tendremos en total 10 nodos y 9 muelles. Donde todos los nodos tendrán una misma cantidad de masa de valor 1 kg cada uno.

3.2 Condiciones de contorno

Es relevante prestar atención al fijado de los dos nodos de los extremos en los *solvers* que obtienen la actualización de las posiciones mediante la resolución de un sistema lineal. En este sentido, ambos métodos basados en *Projective Dynamics* siguen esta estructura. Las alternativas para inmovilizar estos nodos son:

- Sobrescribir los valores de posiciones de los nodos fijos a cada paso.
- Eliminar completamente los grados de libertad de la resolución de problemas.
- Establecer condiciones de contorno en el sistema lineal antes de su resolución.

Siendo conscientes de que para un ejemplo de estas dimensiones no es necesario realizar soluciones tremendamente elaboradas, es un buen experimento mental discernir el método más deseable para un simulador, dado que se desea extrapolar resultados a simulaciones a mayor escala.

En primer lugar, la sobreescritura de posiciones implica perder el efecto que ejercen los nodos fijos sobre el resto en la resolución del sistema. Este se resolvería considerando unas condiciones que no serían ciertas después.

En segundo lugar, la eliminación completa de los grados de libertad ofrece más optimalidad, dado que hace desaparecer completamente el problema. A parte de alguna sutil redefinición de nodos para que se soporten muelles con un solo extremo móvil, el gran problema de este procedimiento es que convertir nodos fijos en móviles durante la simulación supone una ampliación todas las matrices y vectores globales que intervienen

en el problema. En animación controlada por artistas estos cambios pueden suceder una gran cantidad de veces para cada escena y para cada objeto, lo que también convierte en indeseable este método.

Finalmente, también se puede determinar el fijado de nodos realizando una modificación antes de la resolución del sistema lineal del problema. Por ejemplo, en un sistema de dos nodos:

$$\begin{pmatrix} A_1 & A_2 \\ A_3 & A_4 \end{pmatrix} \begin{pmatrix} \mathrm{x}_1 \\ \mathrm{x}_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \quad 43$$

Si se quiere fijar el segundo $x_2 = \bar{x}_2$, queremos resolver en realidad este sistema:

$$\begin{pmatrix} A_1 & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 - A_2\bar{x}_2 \\ \bar{x}_2 \end{pmatrix} \quad 44$$

Para llegar a él, debemos seguir los siguientes pasos:

1. Crear un vector auxiliar *u*:

$$\begin{pmatrix} 0 \\ \bar{x}_2 \end{pmatrix} \quad 45$$

2. Calcular $A \cdot u$:

$$\begin{pmatrix} A_1 & A_2 \\ A_3 & A_4 \end{pmatrix} \begin{pmatrix} 0 \\ \bar{x}_2 \end{pmatrix} = \begin{pmatrix} A_2\bar{x}_2 \\ A_4\bar{x}_2 \end{pmatrix} \quad 46$$

3. Redefinir b:

$$b = b - A \cdot u \quad 47$$

4. Definir los valores 0 y 1 de la matriz *A* necesarios (como en la Ecuación 44) y las posiciones de los nodos fijados en *b*, en este ejemplo $\bar{x}_2$.

Así, el trabajo sucede antes de la resolución, y se permite la posibilidad de que el nodo correspondiente a x_2 se libere en un momento posterior de la simulación. Adicionalmente, en los métodos que fijan la matriz *A* para toda la simulación, este trabajo puede ser incluido como parte del preproceso de cálculo, lo que ahorra tiempo de cómputo de fijado a cada iteración.

3.3 Extended Position-Based Dynamics (Macklin, et al., 2016)

En el caso de la Dinámica Basada en Posiciones, el detalle más relevante de la implementación para nuestro escenario de prueba es la definición de los segmentos de la cuerda como restricciones de distancia:

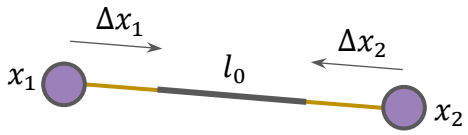

Figura 4. Gráfico de la restricción de distancia.

Para obtener $\Delta\lambda$ y Δx se debe calcular la restricción $C(x)$ teniendo en cuenta que el objetivo es que las posiciones de los dos nodos que la conforman vuelvan a su distancia de reposo, l_0:

$$\mathrm{C} = |x_1 - x_2| - l_0 = 0 \tag{48}$$

Además, también necesitamos calcular el gradiente de esta restricción:

$$\nabla C(x_1, x_2) = \begin{pmatrix} \frac{\partial C}{\partial x_1} & \frac{\partial C}{\partial x_2} \end{pmatrix} = (\nabla_{x_1} C \quad \nabla_{x_2} C) = \begin{pmatrix} \frac{x_1 - x_2}{|x_1 - x_2|} & \frac{-(x_1 - x_2)}{|x_1 - x_2|} \end{pmatrix} \tag{49}$$

Una vez tenemos estos dos cálculos, simplemente debemos sustituirlos en las ecuaciones que define el método XPBD (18 y 19) y obtendremos los incrementos de posición de los dos nodos. Como detalle, se observa que, para esta restricción, $\nabla C(x)$ es la diferencia de posiciones normalizada, un vector unitario, invirtiendo su signo para x_2. Por lo que el gradiente al cuadrado respecto a cada nodo es igual a 1. Por tanto, el cálculo de $\nabla C \cdot M^{-1} \nabla C^T$, utilizado en la Ecuación 18 de $\Delta\lambda$, queda así:

$$\nabla \mathrm{C} \cdot \mathrm{M}^{-1} \nabla \mathrm{C}^{\mathrm{T}} = (\nabla_{\mathrm{x}_1} \mathrm{C} \quad \nabla_{\mathrm{x}_2} \mathrm{C}) \cdot \begin{pmatrix} \frac{1}{\mathrm{m}_1} & 0 \\ 0 & \frac{1}{\mathrm{m}_2} \end{pmatrix} \cdot \begin{pmatrix} \nabla_{x_1} C \\ \nabla_{x_2} C \end{pmatrix}, \tag{50}$$

$$= \frac{\nabla_{x_1}^2 C}{m_1} + \frac{\nabla_{x_2}^2 C}{m_2} = \frac{1}{m_1} + \frac{1}{m_2}$$

De este mismo cálculo, podemos deducir la forma que tendrá también en nuestro problema la Ecuación 19:

$$\Delta x_1 = \frac{\nabla_{x_1} C}{m_1} \cdot \Delta\lambda$$
$$\Delta x_2 = \frac{\nabla_{x_2} C}{m_2} \cdot \Delta\lambda \qquad 51$$

El método del que parte la propuesta de este trabajo, (Müller, et al., 2007), proponía una técnica para esta familia de métodos donde se calcula una corrección de la posición a cada paso, considerando que cada nodo posee masa individual. Para asegurar que las restricciones no desplazaran los nodos fijos, definía la inversa de la masa como nula. Esto hace que en el cálculo de Δx (Ecuación 17) quede anulado y no se actualice la posición de los nodos. Esta técnica es la que utilizamos para fijar nodos para este *solver*.

3.4 Fast Simulation of Mass-Spring Systems (Liu, et al., 2013)

Como el nombre indica, este método está especialmente diseñado para el escenario en el que vamos a trabajar. Por tanto, la definición de los distintos componentes del algoritmo es directa.

Cabría señalar que para poder crear bien las matrices Laplaciana (L) y de incidencia (J), hay que tener en cuenta la conectividad de los nodos. En nuestro caso, está descrito en la sección 3.1. Teniendo esto y las ecuaciones que las definen, las matrices quedarían como sigue. Por una parte, *L*, la Laplaciana:

$$L = k \cdot \begin{pmatrix} \mathrm{I} & -\mathrm{I} & & & \\ -\mathrm{I} & 2\mathrm{I} & -\mathrm{I} & & \\ & -\mathrm{I} & \ddots & \ddots & \\ & & \ddots & 2\mathrm{I} & -\mathrm{I} \\ & & & -\mathrm{I} & \mathrm{I} \end{pmatrix} \qquad 52$$

Donde I se corresponde a la matriz identidad de tamaño “dimensión de nodos x dimensión de nodos”. La diagonal principal representa la cantidad de nodos con la que conecta cada nodo (los extremos conectan con 1 y los demás con 2). Las diagonales secundarias identifican la conectividad concreta. Por tanto, esta matriz adquiere la dimensionalidad de “grados de libertad x grados de libertad”. En nuestro caso, por ejemplo, es de: $(3 \cdot 10)\ x\ (3 \cdot 10)$.

Por otra parte, J, la matriz de incidencia:

$$\mathrm{J} = \mathrm{k} \cdot \begin{pmatrix} I & & \\ -I & I & \\ & -I & \ddots \\ & & \ddots \end{pmatrix} \qquad 53$$

Donde I vuelve a tener el mismo tamaño que antes. J transforma del espacio de grados de libertad al espacio de coordenadas reducidas, ponderado por los pesos. Por ello, su dimensión es "grados de libertad x coordenadas reducidas". Para el escenario de pruebas, por ejemplo, es de: $(3 \cdot 10)\ x\ (3 \cdot 9)$.

3.5 ADMM ⊇ Projective Dynamics (Overby, et al., 2017)

En primer lugar, este solver utiliza una matriz similar a Liu para que las energías dependan de las coordenadas locales (*D*), en este caso la llama "matriz de reducción". Es decir, cumple la misma función que la matriz de incidencia *J* (Ecuación 53). La diferencia es que adquiere la forma transpuesta, $D = J^T$ y no está ponderada por pesos:

$$D = \begin{pmatrix} I & -I & & \\ & I & -I & \\ & & \ddots & \ddots \end{pmatrix} \qquad 54$$

Donde I se corresponde a la matriz identidad de tamaño "dimensión de nodos x dimensión de nodos". Su dimensión es "coordenadas reducidas x grados de libertad". En nuestro caso, por ejemplo, es de: $(3 \cdot 9)\ x\ (3 \cdot 10)$.

En segundo lugar, la actualización del paso local del *solver* implica saber qué tipo de energía se está simulando. Nosotros estamos representando la energía de elasticidad de Hooke. Tomando de referencia el Apéndice B de (Overby, et al., 2017), y recordando que nuestra variable de peso era $\omega_i = \sqrt{k}$, podemos calcular la actualización de *z*, creando una variable $p \in C$, que minimice $|\omega_i \cdot (p_i - (D_i \cdot x^{n+1} + \overline{u_i}^n))|$:

$$p_i^{n+1} = l_0 \cdot \frac{D_i \cdot x^{n+1} + \overline{u_i}^n}{|D_i \cdot x^{n+1} + \overline{u_i}^n|} \qquad 55$$

Una vez obtenida, *z* adquiere la siguiente forma, que, si simplificamos, en nuestro caso donde $\omega_i^2 = \mathrm{k}$:

$$z_i^{n+1} = \frac{1}{\omega_i^2+k} \cdot (k \cdot p_i + \omega_i^2 \cdot (D_i \cdot x^{n+1} + \overline{u_i}^n)),$$

$$z_i^{n+1} = \frac{1}{2} \cdot (p_i^{n+1} + D_i \cdot x^{n+1} + \overline{u_i}^n) \quad (56)$$

Una vez hemos obtenido la proyección acerándonos lo máximo posible a la restricción, se actualiza la variable dual ($\bar{u}$) siguiendo la Ecuación 42. Se aprecia una diferencia notable entre este método y el anterior en cuanto a la complejidad del paso local. Como contrapartida, ADMM permite introducir una mayor cantidad de energías en el *solver*.

4. Estudio y discusión de resultados

En el capítulo anterior hemos estudiado las especificidades de los diferentes métodos a comparar para el caso concreto de nuestra escena de prueba: la cuerda catenaria. Siguiendo en esa línea, pasaremos a definir el proceso experimental: el repaso de los objetivos establecidos y la metodología escogida para alcanzarlos.

4.1 Introducción al proceso experimental

Si recordamos la sección 1.2, los objetivos definidos relevantes para este capítulo eran:

- Implementar los métodos mencionados sobre un sistema masa-muelle.
- Analizar su adecuación a la simulación de problemas estáticos y dinámicos.
- Estudiar su precisión con respecto a una referencia.

Por lo tanto, es necesario seleccionar una o más métricas que permitan estudiar los requisitos presentados. Consideramos que unos buenos candidatos son:

- Para el análisis dinámico, la medición de las oscilaciones que experimenta la cuerda antes de terminar en el estado de reposo, así como el periodo entre oscilaciones.
- Para el análisis estático, el error numérico del estado de reposo final alcanzado con respecto a una solución base.

En la siguiente sección daremos una manera de estudiar estas métricas.

4.2 Pruebas

En la sección anterior definimos tres métricas necesarias para obtener la información que deseamos: las oscilaciones y el periodo de la energía cinética hasta la convergencia del sistema, y el error numérico de este estado de convergencia.

En primer lugar, es importante definir la condición de convergencia. Dada la naturaleza de la prueba, la expectativa es una oscilación eterna, en un espacio sin rozamiento con el

aire o entre los elementos de simulación. Sin embargo, todos los métodos de resolución aproximada de sistemas lineales introducen amortiguamiento numérico en sus resultados, que se expresa como un decremento paulatino de la energía total del sistema, a diferente velocidad dependiendo del *solver*. Dado este marco, una buena manera de medir cuándo el sistema ha convergido, es analizar los sucesivos picos de energía cinética y establecer un umbral absoluto de parada en estos máximos, común para todos los métodos. La búsqueda de estos picos también nos es útil para definir el periodo medio del sistema. Hemos fijado el umbral de parada en 0,001J para un pico de energía cinética. En cuanto la simulación detecte un pico cuyo valor absoluto sea menor al umbral, considerará que el sistema ha llegado al reposo.

En segundo lugar, el error numérico en el estado de convergencia de la catenaria se puede medir comparando con un modelo de referencia. En nuestro caso hemos escogido *Backward* Euler Implícito, dado que pese a su elevado coste es usado comúnmente como verdad fundamental.

4.3 Resultados

Para realizar los experimentos, se han variado tres parámetros: el *paso de tiempo*, el número de iteraciones y la rigidez de la cuerda. Respectivamente, cada uno de los valores ha tenido 3, 3 y 2 muestras distintas, elevando la cantidad de experimentos por modelo de simulación a 18. Los tres modelos probados, sumados a una referencia de *Backward* Euler Implícito para cada una de las rigideces, hacen un total de 56 pruebas.

En cuanto al *paso de tiempo*, se han escogido tres valores distintos, desde uno de muy poca discretización temporal con un gran error, 0.1s, a otro de discretización fina pero poco avance relativo al tiempo de cómputo, 0.001s, es decir, 1000 iteraciones por segundo simulado.

En cuanto al número de iteraciones de cada método, dado que no se pueden comparar directamente, se han escogido mediante proceso experimental una serie de valores que daban resultados similares.

En cuanto a las dos rigideces seleccionadas, se han determinado valores de 10^3 y 10^6 para examinar un abanico amplio de dificultad de resolución, ya que 10^3 no es

complicado de cubrir para ningún método, pero 10^6 eleva el error y el amortiguamiento numérico de manera relevante.

Pasamos a presentar los resultados, ordenados por *solver*:

- Valores de referencia, *Backward* Euler Implícito:

dt (s)	iter.	rigidez	error	nº máx. E_k	T/2 medio	error relat. T medio
0,001	5	1000	-	723	0,6773	-
0,001	5	1E+06	-	175	0,2060	-

Tabla 1. Valores de referencia *Backward* Euler Implícito con rigidez de 10^3 y 10^6.

- XPBD:

dt (s)	iter.	rigidez	error	nº máx. E_k	T/2 medio	error relat. T medio
0,1	1	1000	1,22E+01	14	1,0857	0,60
0,01	1	1000	2,85E-01	76	0,6817	0,01
0,001	1	1000	2,36E-03	720	0,6777	0,00
0,1	20	1000	2,21E-01	9	0,6889	0,02
0,01	20	1000	6,07E-03	74	0,6712	0,01
0,001	20	1000	5,46E-03	719	0,6776	0,00
0,1	50	1000	1,71E-01	9	0,6778	0,00
0,01	50	1000	6,07E-03	74	0,6712	0,01
0,001	50	1000	5,46E-03	719	0,6776	0,00

Tabla 2. Resultados de XPBD con rigidez de 10^3.

dt (s)	iter.	rigidez	error	nº máx. E_k	T/2 medio	error relat. T medio
0,1	1	1E+06	1,97E+01	14	1,0500	4,10
0,01	1	1E+06	3,14E+00	45	0,4342	1,11
0,001	1	1E+06	2,07E-01	199	0,2290	0,11
0,1	20	1E+06	5,72E+00	7	0,5571	1,70
0,01	20	1E+06	6,22E-01	25	0,2660	0,29
0,001	20	1E+06	3,56E-04	175	0,2060	0,00
0,1	50	1E+06	3,81E+00	6	0,4833	1,35
0,01	50	1E+06	2,83E-01	22	0,2336	0,13
0,001	50	1E+06	3,56E-04	175	0,2060	0,00

Tabla 3. Resultados de XPBD con rigidez de 10^6.

- *Fast Simulation of Mass-Spring Systems*:

dt (s)	iter.	rigidez	error	nº máx. E_k	T/2 medio	error relat. T medio
0,1	1	1000	3,58E-03	2	20,5000	29,27
0,01	1	1000	7,45E-03	13	0,6577	0,03
0,001	1	1000	4,16E-03	87	0,6708	0,01

0,1	30	1000	2,06E-03	9	0,6889	0,02
0,01	30	1000	2,98E-03	61	0,6705	0,01
0,001	30	1000	3,70E-03	227	0,6737	0,01
0,1	70	1000	2,12E-03	9	0,6889	0,02
0,01	70	1000	2,98E-03	61	0,6705	0,01
0,001	70	1000	3,70E-03	227	0,6737	0,01

Tabla 4. Resultados de *Fast Simulation of Mass-Spring Systems* con rigidez de 10^3.

dt (s)	iter.	rigidez	error	nº máx. E_k	T/2 medio	error relat. T medio
0,1	1	1E+06	3,16E-04	2	1108,6999	5381,81
0,01	1	1E+06	3,44E-04	2	90,6399	439,06
0,001	1	1E+06	3,50E-04	2	7,4464	35,15
0,1	30	1E+06	3,16E-04	2	49,8000	240,78
0,01	30	1E+06	2,32E-03	5	0,2140	0,04
0,001	30	1E+06	3,43E-04	184	0,2054	0,00
0,1	70	1E+06	3,16E-04	2	22,1500	106,54
0,01	70	1E+06	5,79E-04	20	0,2014	0,02
0,001	70	1E+06	3,46E-04	184	0,2054	0,00

Tabla 5. Resultados *de Fast Simulation of Mass-Spring Systems* con rigidez de 10^6.

- ADMM ⊇ *Projective Dynamics*:

dt (s)	iter.	rigidez	error	nº máx. E_k	T/2 medio	error relat. T medio
0,1	1	1000	1,37E-02	11	0,9455	0,40
0,01	1	1000	3,51E-03	75	0,6756	0,00
0,001	1	1000	3,59E-03	726	0,6740	0,00
0,1	20	1000	1,92E-03	9	0,6889	0,02
0,01	20	1000	4,04E-03	74	0,6714	0,01
0,001	20	1000	3,52E-03	725	0,6740	0,00
0,1	50	1000	1,78E-03	9	0,6889	0,02
0,01	50	1000	4,04E-03	74	0,6714	0,01
0,001	50	1000	3,52E-03	725	0,6740	0,00

Tabla 6. Resultados de ADMM ⊇ *Projective Dynamics* con rigidez de 10^3.

dt (s)	iter.	rigidez	error	nº máx. E_k	T/2 medio	error relat. T medio
0,1	1	1E+06	6,18E-03	13	7,2692	34,29
0,01	1	1E+06	1,82E-02	49	0,7378	2,58
0,001	1	1E+06	3,30E-04	187	0,2206	0,07
0,1	20	1E+06	2,27E-03	8	1,5250	6,40
0,01	20	1E+06	2,12E-03	20	0,2225	0,08
0,001	20	1E+06	3,16E-04	175	0,2061	0,00
0,1	50	1E+06	4,55E-03	7	0,9714	3,72
0,01	50	1E+06	3,29E-04	19	0,2111	0,02
0,001	50	1E+06	2,31E-04	175	0,2061	0,00

Tabla 7. Resultados de ADMM ⊇ *Projective Dynamics* con rigidez de 10^6.

4.4 Conclusiones

A fin de facilitar la comprensión de los datos de la sección anterior, durante las conclusiones extraeremos pequeñas muestras de ellos. Además, cada método se representará mediante una abreviatura y un color (con matices de tonalidades en caso de visualizar diversos datos de un mismo *solver*):

Abreviatura y color	Nombre del *solver*
XPBD	XPBD (*Extended Position-Based Dynamics*)
Fast m-s	*Fast Simulation of Mass-Spring Systems*
ADMM	ADMM ⊇ *Projective Dynamics*

Tabla 8. Codificación de cada *solver* en los resultados.

4.4.1 Estudiando la estática del problema

En esta sección nos centramos en la convergencia de los métodos. Consideramos los valores de Euler Implícito como *ground truth* (Tabla 1), el resto de métodos calculan su error en base a la diferencia de posiciones en estado de reposo.

En primer lugar, podemos observar cómo, en general, al aumentar el trabajo del sistema, ya sea reduciendo el *paso de tiempo* (*dt*) como aumentando las iteraciones por cada paso del *solver*, el error de las métricas desciende. Esto es un comportamiento totalmente esperable.

Si entramos al análisis concreto de cada método, en XPBD se aprecia una peor aproximación de entrada en relación al resto de métodos en todas las rigideces. Mostramos, por ejemplo, las primeras aproximaciones con rigidez 10^3:

solver	dt (s)	iter.	rigidez	error
XPBD	0,1	1	1000	**1,22E+01**
Fast m-s	0,1	1	1000	3,58E-03
ADMM	0,1	1	1000	1,37E-02
XPBD	0,1	20	1000	**2,21E-01**
Fast m-s	0,1	30	1000	2,06E-03
ADMM	0,1	20	1000	1,92E-03

Tabla 9. Relación entre el error y las aproximaciones más burdas con rigidez de 10^3.

Partiendo ya en "desventaja", notamos que el decrecimiento de error al disminuir el *dt* dentro las mismas iteraciones y al variar las iteraciones dentro de un mismo rango de *dt* es más acentuado que en los métodos basados en *Projective Dynamics* (Figura 5). Esto es, de entrada realiza una corrección de las posiciones más burda pero en cuanto se le aumenta ligeramente el trabajo (*dt* o *iteraciones*), consigue realizar buenas aproximaciones a nivel visual. Sin embargo, en general no llega al nivel de precisión del resto de métodos, aunque para un escenario donde no se necesitase la mayor precisión, se comporta correctamente.

Esto mismo lo podemos apreciar visualmente en las siguientes figuras:

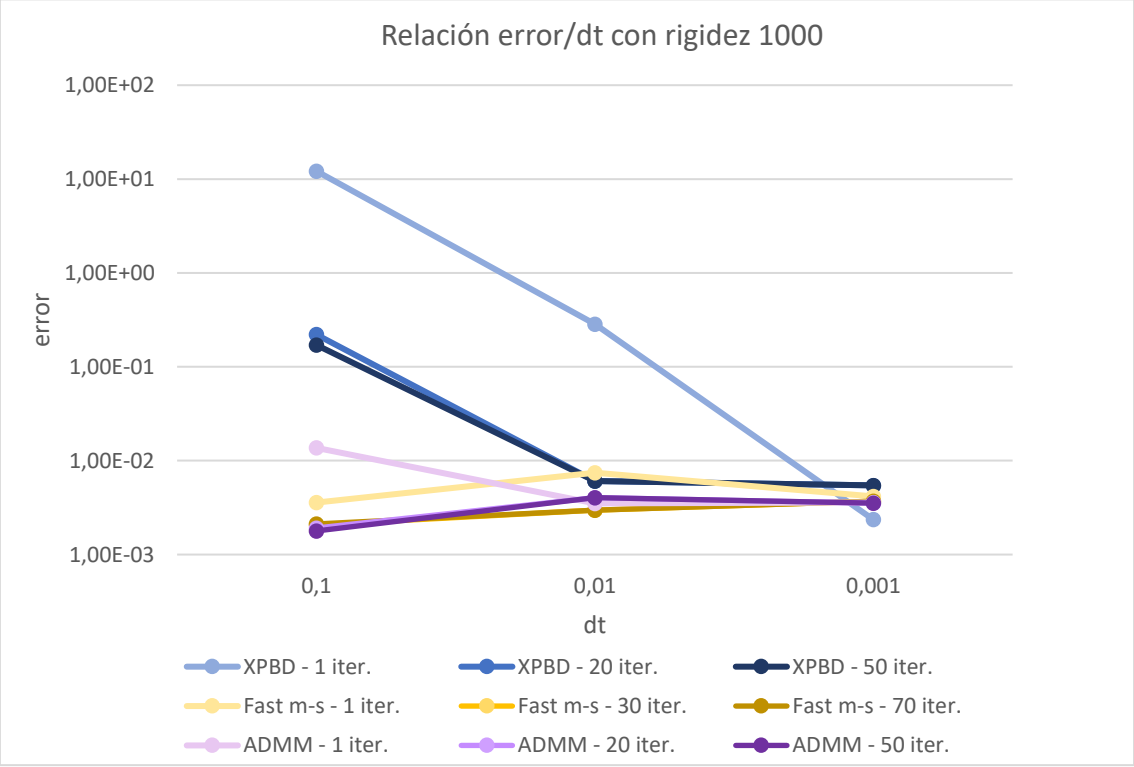

Figura 5. Relación entre el error y el *dt* con rigidez 10^3 en todos los métodos.

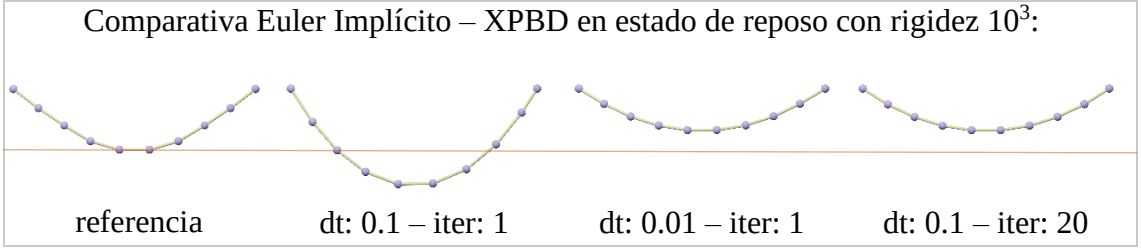

Figura 6. Comparación estado de reposo entre Euler Implícito y XPBD (rigidez: 10^3).

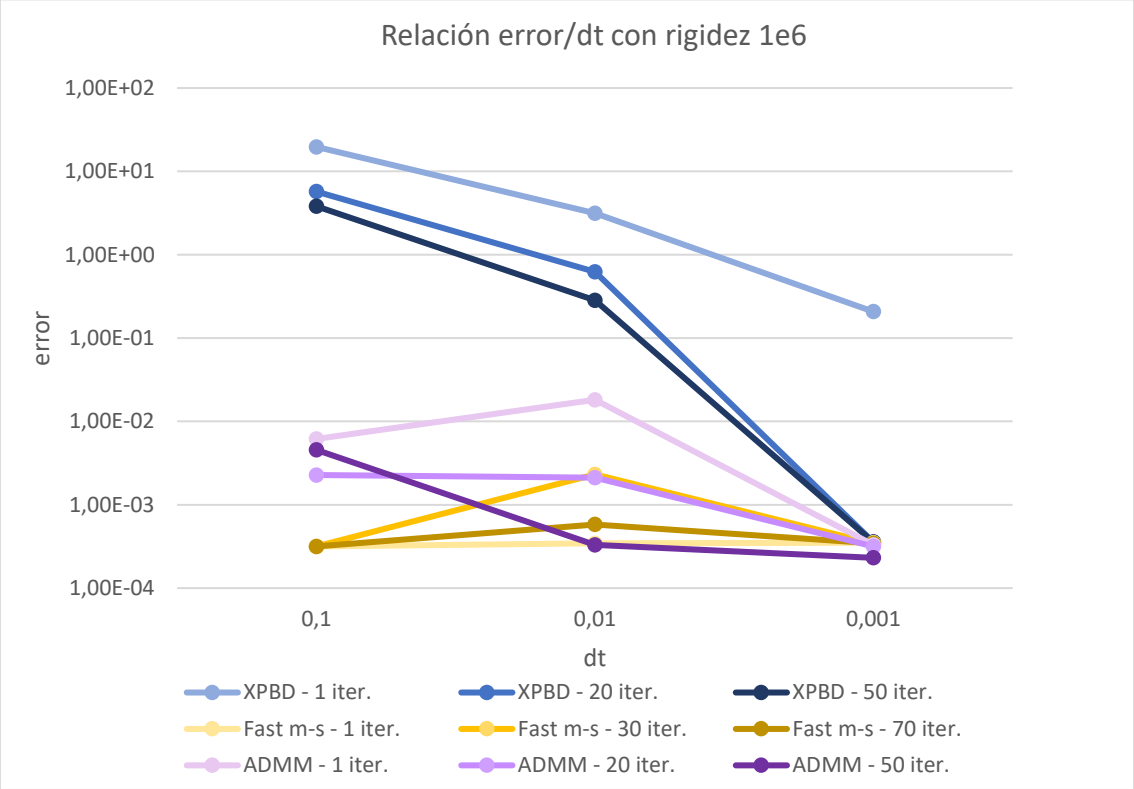

Figura 7. Relación entre el error y el *dt* con rigidez 10^6 en todos los métodos.

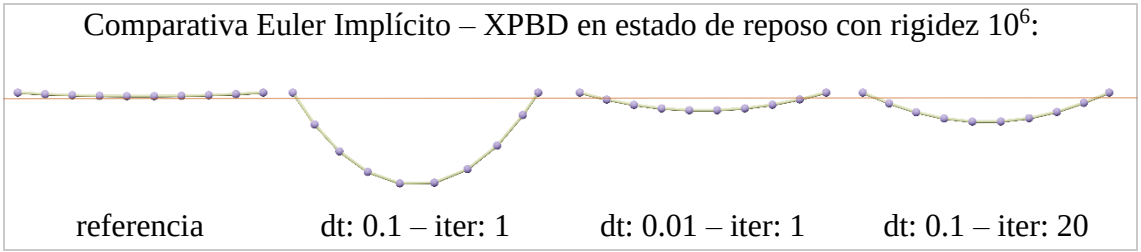

Figura 8. Comparación estado de reposo entre Euler Implícito y XPBD (rigidez: 10^6).

Se aprecia que a XPBD le cuesta más aproximar a la solución en el escenario con mayor rigidez en general (Figura 7 y Figura 8). Necesita llegar a las 50 iteraciones o, en su defecto, disminuir el *dt* al máximo para aumentar la precisión. Esto tiene sentido ya que en el escenario planteado las restricciones de la catenaria trabajan de forma totalmente opuesta a la gravedad, es decir es la situación donde más esfuerzo realizan. Al aumentar su rigidez, mayor será la corrección que deban aplicar. No obstante, constatamos que ya no existe la relación que PBD introducía entre el paso de tiempo y la rigidez y su dependencia con la cantidad de iteraciones.

Si dejamos a un lado XPBD, entrando al análisis de los métodos basados en *Projective Dynamics* (PD), vemos que los errores de convergencia no alcanzan niveles críticos en ninguna situación planteada. El error más grande registrado está en el orden de 10^{-2}.

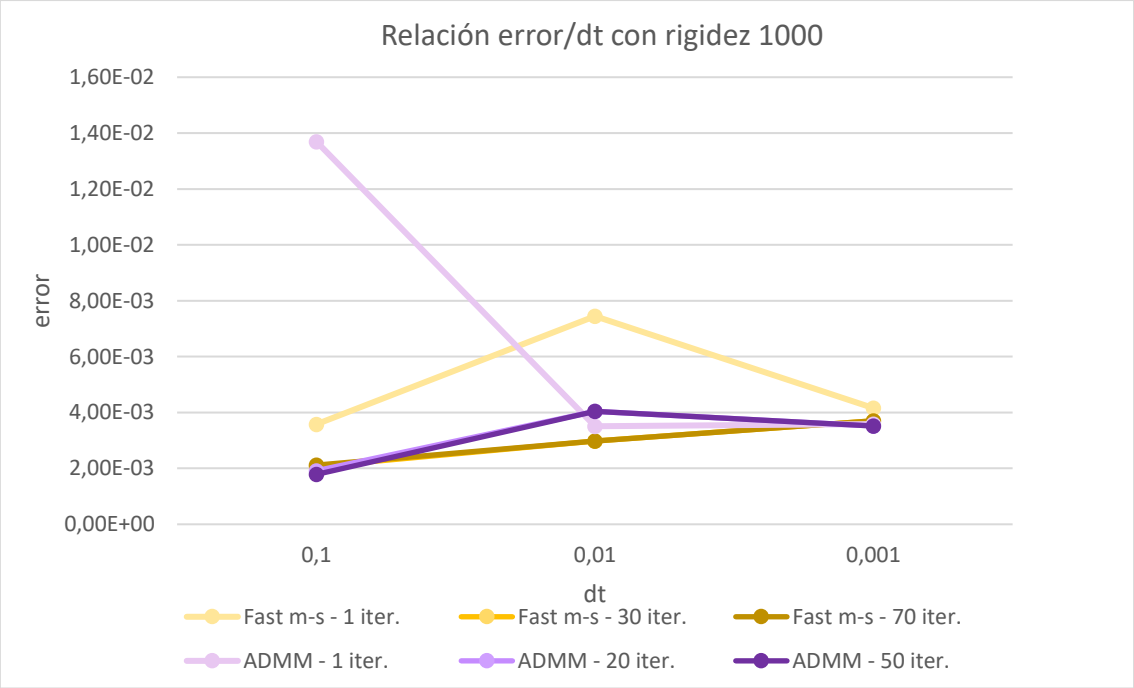

Figura 9. Relación entre el error y el *dt* con rigidez 10^3 en *Projective Dynamics*.

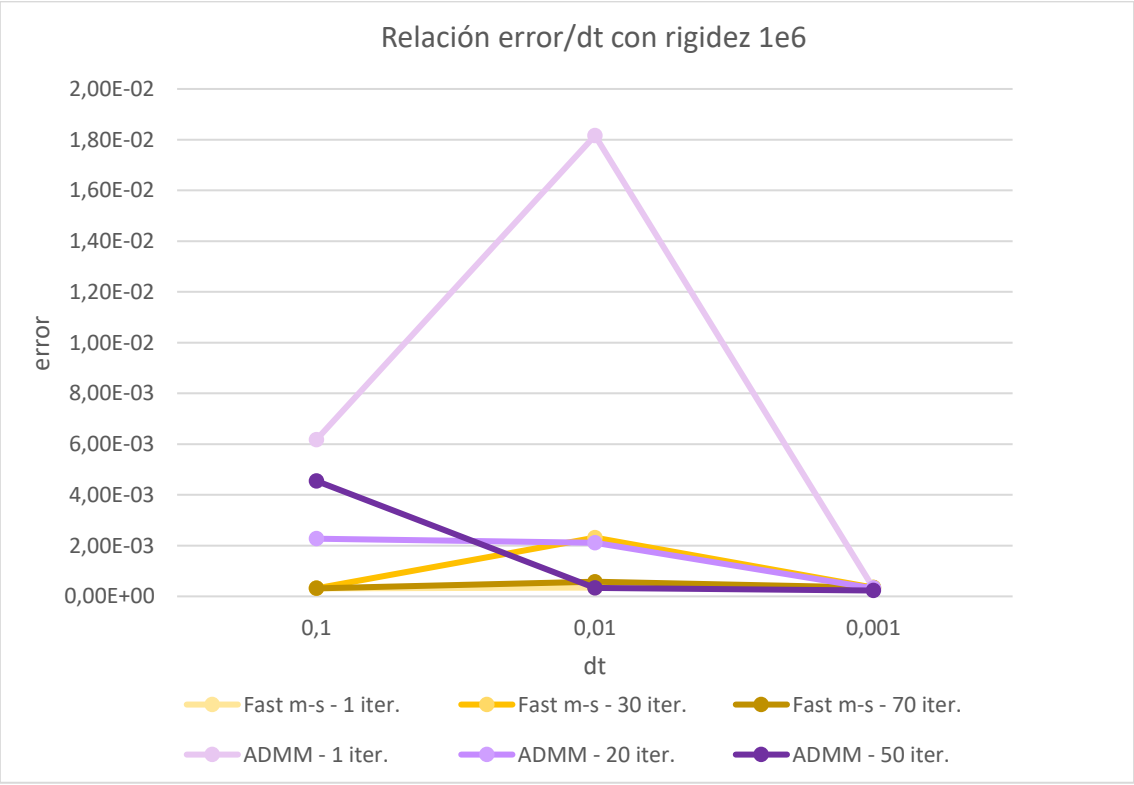

Figura 10. Relación entre el error y el *dt* con rigidez 10^6 en *Projective Dynamics*.

Para los casos de PD, podemos observar cómo reducir el paso de tiempo de 0.1 a 0.01 s, aumenta en general el error de convergencia del problema (ver la Figura 10). Esto puede parecer contraintuitivo pero tiene sentido, ya que en ambos casos de PD el comportamiento dinámico con poco paso de tiempo es muy degenerado, realiza pocas iteraciones, y converge rápidamente. Aumentar la discretización temporal permite que la cuerda se mueva más libremente, aumentando potencialmente el error al que puede llegar en su posición final. De nuevo, al aumentar el paso de tiempo a 0.001s el error vuelve a descender porque la discretización temporal fina implica un error causado menor a cada paso, que vuelve a compensar el comportamiento dinámico inicial.

En conclusión, desde una perspectiva estática, donde no importe el desarrollo dinámico la mejor solución siempre vendrá de *Projective Dynamics* en relación con PBD. Dentro de esta familia de métodos, no existe una gran diferencia. No obstante, debemos recordar que *Fast Simulation of Mass-Spring Systems* está diseñado especialmente para este tipo de escenarios, con discretización masa-muelle, y limita su utilidad fuera de este campo.

4.4.2 Estudiando la dinámica del problema

Observando el problema desde otra perspectiva, queremos estudiar el comportamiento que adquiere la cuerda a lo largo de la simulación en los distintos métodos. Para ello, tomamos como referencia la energía cinética. La evolución de esta medida a lo largo del tiempo refleja cómo se comporta visualmente el sistema. Además, sus máximos nos permiten ver cómo de eficiente es frente al esfuerzo. A continuación mostramos un esquema que relaciona los distintos marcadores que vamos a utilizar con el movimiento que efectúa la catenaria. Identificamos mediante 5 pasos el proceso de un periodo completo de la curva de la energía cinética y los relacionamos con el movimiento de la cuerda:

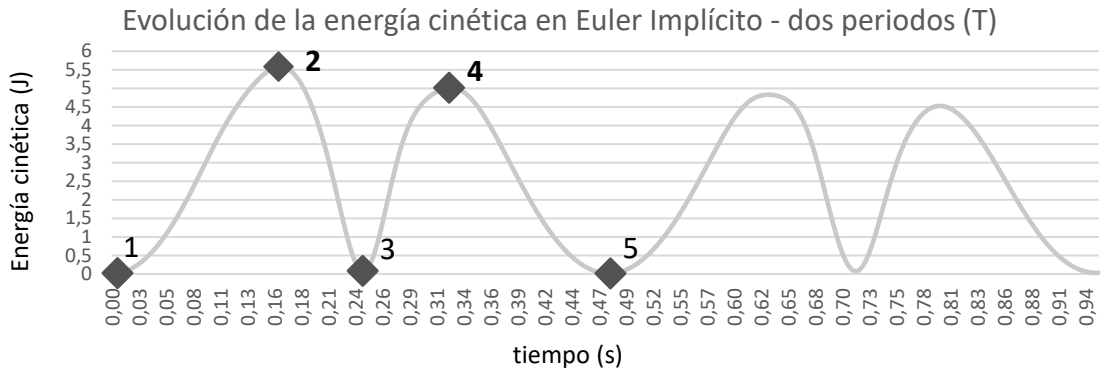

Figura 11. Muestra de dos T de la energía cinética en Euler Implícito con rigidez 10^6.

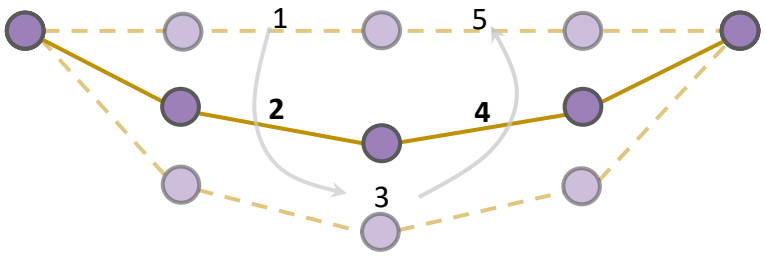

Figura 12. Esquema de la relación entre la energía cinética y el movimiento de la cuerda.

Por otro lado, también hemos calculado el periodo medio de la curva que registra la energía cinética que tiene cada caso del estudio. Concretamente, en los datos de la sección 4.3 se muestra la mitad del periodo, ya que nos resultaba más cómodo registrarlo por cada máximo que detectaba el sistema. Igualmente, puesto que el periodo permanece constante en el tiempo, aporta el mismo significado. Además, hemos calculado el error relativo respecto al valor de referencia de Euler Implícito para ver más fácilmente las diferencias. A través de esta métrica complementamos la información recogida por los máximos de energía.

Primeramente, al igual que en el estudio de la estática, la precisión se ve afectada de forma drástica con un *paso de tiempo* grande:

solver	dt (s)	iter.	rigidez	nº máx. E_k	T/2 medio	error relat. T medio
XPBD	0,1	1	1000	14	1,0857	0,60
Fast m-s	0,1	1	1000	2	20,5000	29,27
ADMM	0,1	1	1000	11	0,9455	0,40
XPBD	0,1	20	1000	9	0,6889	0,02
Fast m-s	0,1	30	1000	9	0,6889	0,02
ADMM	0,1	20	1000	9	0,6889	0,02

Tabla 10. Relación entre la energía cinética y *dt* 0.1s con rigidez de 10^3.

solver	dt (s)	iter.	rigidez	nº máx. E_k	T/2 medio	error relat. T medio
XPBD	0,1	1	1e6	14	1,0500	4,10
Fast m-s	0,1	1	1e6	2	1108,6999	5381,81
ADMM	0,1	1	1e6	13	7,2692	34,29
XPBD	0,1	20	1e6	7	0,5571	1,70
Fast m-s	0,1	30	1e6	2	49,8000	240,78
ADMM	0,1	20	1e6	8	1,5250	6,40
XPBD	0,1	50	1e6	6	0,4833	1,35
Fast m-s	0,1	70	1e6	2	22,1500	106,54
ADMM	0,1	50	1e6	7	0,9714	3,72

Tabla 11. Relación entre la energía cinética y *dt* 0.1s con rigidez de 10^6.

En esta ocasión el efecto del *dt* es mayor y por mucho que se aumenten las iteraciones de cada paso, las simulaciones presentan notables diferencias con el comportamiento de Euler Implícito. En general, salvo casos que detallaremos, se aprecia fácilmente que los *solvers* alcanzan con relativo éxito el comportamiento de referencia bajo poca rigidez. Por ello, encontramos más interesante centrar ahora el estudio en la alta rigidez.

Si nos fijamos en el método de *Fast Simulation of Mass-Spring Systems*, vemos que introduce un fuerte amortiguamiento numérico con pocas iteraciones. Tiene muy pocos máximos de energía y, por tanto, el periodo incrementa de forma drástica. Esto es, visualmente no efectúa prácticamente "rebotes" (Figura 13).

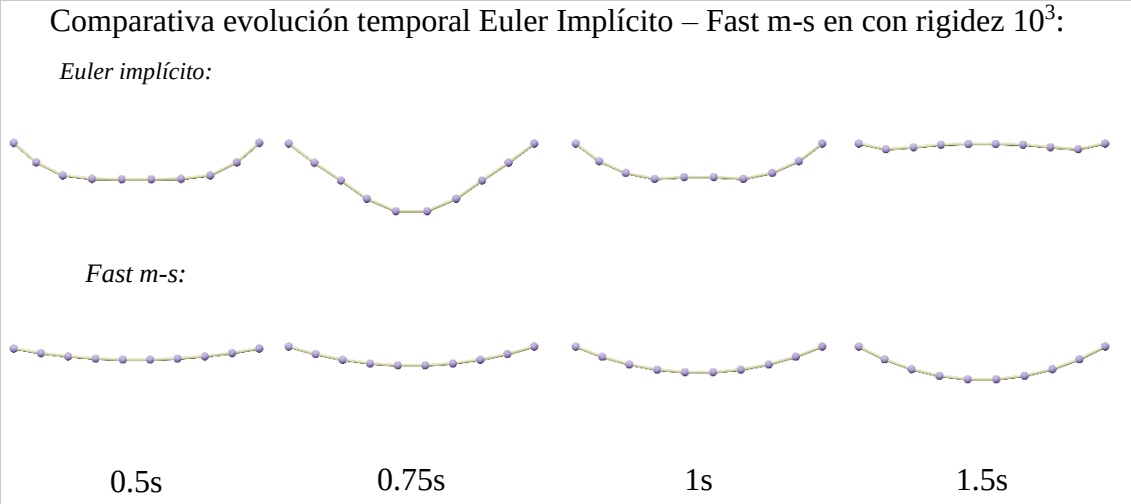

Figura 13. Comparación de la evolución temporal entre Euler Implícito y *Fast m-s* (rigidez: 10^3, dt: 0.1s, iter: 1).

Este mismo problema ya lo indican los autores en (Liu, et al., 2013). Además, si analizamos los datos, se ve acentuado cuanto mayor es la rigidez. Fijándonos en el valor elevado de rigidez, el *solver* consigue centrar el periodo a partir de iteraciones y *dt* con valores medios. (Figura 14) Pero, si analizamos las oscilaciones, vemos que no alcanza los valores del resto de métodos. Continúa teniendo mayor amortiguamiento. En las dos siguientes figuras se aprecia lo que estamos comentando. Están extraídas del caso medio: *dt* de 0.01s, iteraciones medias de cada *solver* y rigidez elevada.

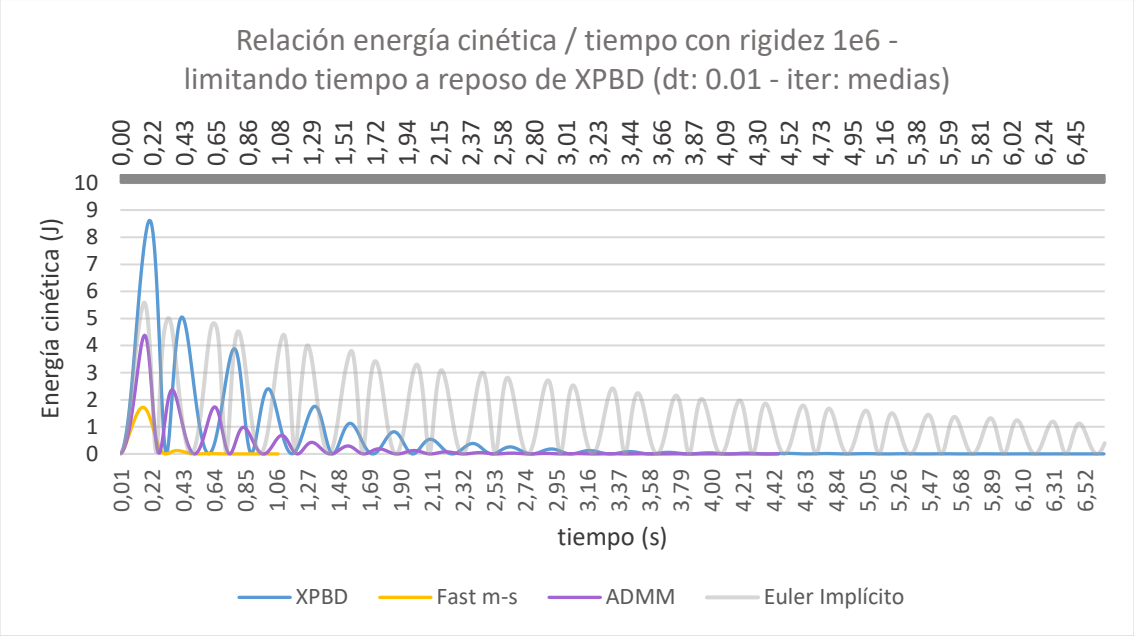

Figura 14. Relación entre la energía cinética y el tiempo (rigidez: 10^6, dt: 0.01s, iter: medias).

Y el mismo gráfico pero limitando los datos hasta el tiempo en que el método de *Fast m-s* llegue al reposo:

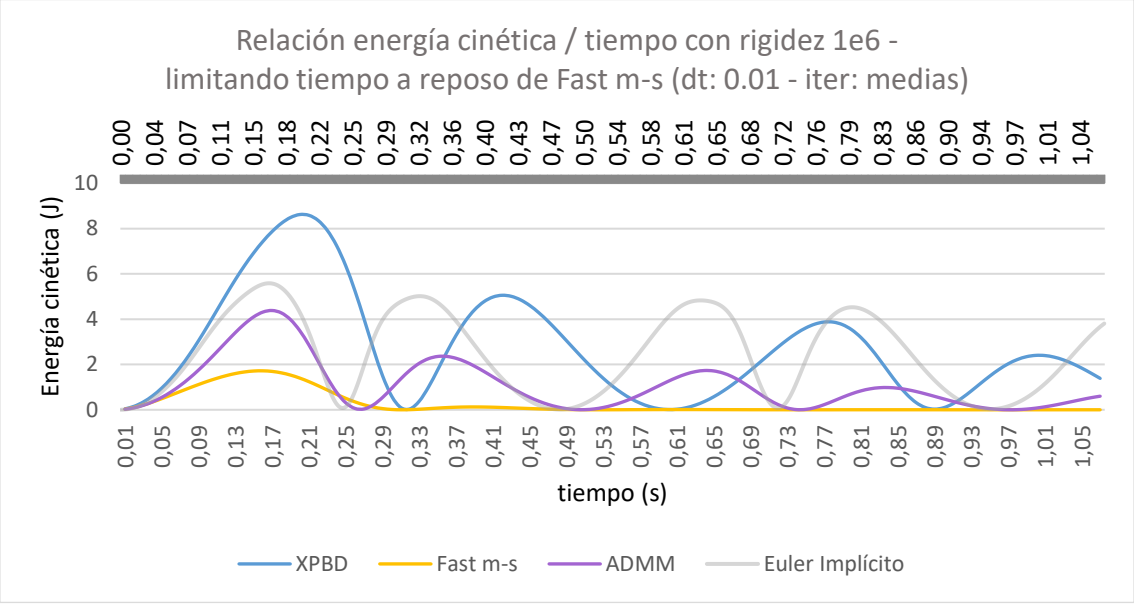

Figura 15. Relación entre la energía cinética y el tiempo acortada (rigidez: 10^6, dt: 0.01s, iter: medias).

Prestando atención al periodo, con un *paso de tiempo* medio se obtiene una buena aproximación en general. Entre los dos métodos que mejor se comportan, ADMM presenta mejores resultados en un rango medio en lo que respecta al periodo. Sin embargo, XPBD aproxima con menor error en los casos del *dt* grande (0.1s) (ver Tabla 11). En cuanto a los valores de *dt* fino (0.001s), ambos funcionan bien:

solver	dt (s)	iter.	rigidez	nº máx. E_k	T/2 medio	error relat. T medio
XPBD	0,001	1	1e6	199	0,2290	0,11
Fast m-s	0,001	1	1e6	2	7,4464	35,15
ADMM	0,001	1	1e6	187	0,2206	0,07
XPBD	0,001	20	1e6	175	0,2060	0,00
Fast m-s	0,001	30	1e6	184	0,2054	0,00
ADMM	0,001	20	1e6	175	0,2061	0,00

Tabla 12. Relación entre la energía cinética y *dt* 0.001s con rigidez de 10^6.

En cuanto a las oscilaciones, si observamos la Figura 15 anterior, todos los métodos siguen el mismo patrón, se refleja fácilmente las oscilaciones de la catenaria. En general XPBD ofrece mejores resultados, aunque realiza mayores oscilaciones a cambio de iniciar el sistema con una mayor cantidad de energía (la referencia está en torno a 6J i XPBD supera 8J). En el caso poco rígido, no hay grandes diferencias entre ADMM y XPBD.

En suma, no existe un método que destaque por encima del resto en el comportamiento dinámico. Sí quedan patentes las deficiencias de *Fast Simulation of Mass-Spring Systems*. A primera vista parecería que ADMM es superior a XPBD también en el caso dinámico si nos fijamos en la mayor precisión que alcanza, con *dt* 0.001. Pero una simulación real oscilaría en un *dt* de entre 0.042s i 0.017s, lo que hace que sea mucho más difícil discernir si una técnica es superior a la otra. Además, se ha de destacar que XPBD hace más oscilaciones antes de detenerse, por lo que su disipación, y en última instancia el comportamiento dinámico, es ligeramente mejor, lo que refuerza la idea de que no haya una elección definitiva para el caso dinámico.

5. Conclusiones

En el desarrollo de este trabajo, hemos realizado un análisis de los métodos de simulación en tiempo real más comunes en el estado del arte. De entre ellos, por su relevancia, hemos seleccionado XPBD (Macklin, et al., 2016), Fast Simulation of Mass-Spring Systems (Liu, et al., 2013) y ADMM ⊇ Projective Dynamics (Overby, et al., 2017). Primero hemos realizado un estudio teórico de cada uno de ellos. Una vez entendido su funcionamiento y características, nuestro objetivo más importante ha sido comparar los comportamientos de las técnicas y, para ello, hemos establecido una catenaria como escenario de pruebas, implementado las energías y restricciones necesarias. Una vez completados los métodos, hemos definido como métricas relevantes el error de convergencia, la cantidad de oscilaciones y el periodo de éstas para obtener los resultados experimentales.

Las pruebas han ido dirigidas a medir la calidad de simulación con respecto a un método base que se toma como *ground truth*: *Backward* Euler Implícito. El estudio se ha dividido en dos campos: la simulación estática y la simulación dinámica, que poseen requisitos distintos y por lo tanto también características deseables diferentes.

En cuanto a la simulación estática, los algoritmos más favorables han sido los basados en *Projective Dynamics*, teniendo ADMM preferencia, dada su mayor versatilidad.

Al contrario, en las simulaciones dinámicas, no ha habido un claro método que se destaque con respecto a los demás, ofreciendo resultados similares en las características medidas, pero sí que se ha de destacar que Liu ofrece resultados peores que ADMM y XPBD, por lo que se descarta como método viable para el ámbito dinámico.

Por lo tanto, podemos concluir que, dependiendo del tipo de simulación a realizar, los métodos más apropiados serían XPBD para el caso dinámico y ADMM para simulaciones tanto estáticas como dinámicas. Es interesante comentar cómo no existe una respuesta clara, habiéndose desarrollado los dos métodos mencionados de manera paralela en el tiempo. Esto hace que se determinen como estado del arte por motivos distintos y que el uso de ambos sea legítimo.

Como comentarios de trabajo futuro, sería interesante estudiar en profundidad el rendimiento temporal de cada uno de los métodos, dado que en nuestro caso hemos seguido las guías de los respectivos autores, que encontraban los números de iteraciones para las comparativas a base de prueba y error. Adicionalmente, se podría aumentar la dimensionalidad del problema para aumentar el error generado (relajando o no la rigidez alta que hemos establecido), pero eso dejaría fuera de la comparativa al método de Liu si se tratan casos fuera del rango de masa-muelle (FEM, por ejemplo). También se contemplaría seguir estudiando los métodos basados en *projective dynamics* existentes. Por ejemplo, una evolución de ADMM ⊇ Projective Dynamics (Overby, et al., 2017) es el *solver* propuesto por (Brown, et al., 2018) que introduce la posibilidad de modelar fuerzas disipativas.

6. Referencias bibliográficas

Baraff, D., 1997. Implicit Methods for Differential Equations. En: *Physically Based Modeling: Principles and Practice.* s.l.:SIGGRAPH ’97Course Notes.

Baraff, D. & Witkin, A., 1998. Large Steps in Cloth Simulation. *COMPUTER GRAPHICS Proceedings, Annual Conference Series.*

Bender, J., Müller, M. & Macklin, M., 2017. A Survey on Position Based Dynamics. *EUROGRAPHICS 2017 Tutorials.*

Bouaziz, S. y otros, 2014. Projective Dynamics: Fusing Constraint Projections for Fast Simulation. *ACM Transactions on Graphics.*

Brown, G. E., Overby, M., Forootaninia, Z. & Narain, R., 2018. Accurate Dissipative Forces in Optimization Integrators. *ACM Transactions on Graphics.*

Liu, T., Bargteil, A. W., O’Brien, J. F. & Kavan, L., 2013. Fast Simulation of Mass-Spring Systems. *ACM Transactions on Graphics*, 32(6), p. article nº 214 .

Macklin, M., Müller, M. & Chentanez, N., 2016. XPBD: Position-Based Simulation of Compliant Constrained Dynamics. *MIG '16 Proceedings of the 9th International Conference on Motion in Games,* pp. 49-54.

Martin, S., Thomaszewski, B., Grinspun, E. & Gross, M., 2011. Example-based elastic materials. *ACM Transactions on Graphics.*

Müller, M., Heidelberger, B., Hennix, M. & Ratcliff, J., 2007. Position based dynamics. *J. Vis. Comun. Image Represent.,* Volumen 18.2, pp. 109-118.

Overby, M., Brown, G. E., Li, J. & Narain, R., 2017. ADMM ⊇ Projective Dynamics: Fast Simulation. *IEEE Trans Vis Comput Graph,* 24 julio, 23(10), pp. 2222-2234.

Servin, M., Lacoursiere, C. & Melin, N., 2006. Interactive Simulation of Elastic Deformable Materials. *Proc. SIGRAD.*

Witkin, A., 1997. Differential Equation Basics. En: *Physically Based Modeling: Principles and Practice.* s.l.:SIGGRAPH ’97 Course Notes.