Simulering av Elastiska Material

Grupp 9

Kalle Bladin
Erik Broberg
Emma Forsling Parborg
Martin Gråd

13 mars 2014

Sammanfattning

Här ska sammanfattningen skrivas in

Innehåll

Figurer

Tabeller

1 Inledning

Inledande text är alltid lite trevligt

1.1 Introduktion

1.2 Syfte

Denna rapport syftar till att undersöka hur verklighetstrogna simuleringar av elastiska material kan göras i tre dimensioner med hjälp av mass-fjäder-dämparsystem. Simuleringarna som demonstreras är menade att kunna göras i realtid, och erbjuda en användare viss interaktion.

1.3 Bakgrund

1.3.1 Fysikalisk modellering

En modell som kan användas för simulering av elastiska material är ett så kallat ”mass spring damper system” (Gelenbe, Lent & Stakellari 2012) eller MSD-system; en modell som beskrivs av partiklar sammankopplade med fjädrar och dämpare.

Det enkla systemet i Figur X utgör grunden för den struktur som används i detta projekt. Genom att utöka systemet med partiklar och samma typ av kopplingar, kan många olika material och former beskrivas, exempelvis tyg, gelé, skumgummi eller liknande elastiska material. Beroende på parametrarna fjäderkonstanter, dämparkonstanter, massor och fjäderlängder kan realistiska simuleringar av dessa material uppnås.

Som bakgrund till den fysikaliska modellen låg de grundläggande rörelse- och kraftlagarna (Halliday 2007) som beskrevs av Newton. Newtons andra lag anges i Ekvation 2,

där F är den resulterande kraften på en partikel, x?? är partikelns acceleration, m är partikelns massa och t är tidsvariabeln. Newtons tredje lag summeras i ett citat:

> ”If A puts a force on B, then B puts a force on A, and the two forces are equal in magnitude and have opposite direction.” (Halliday 2007)

Fjädrarna modellerades med Hookes lag som anges i ekvation 3,

där F är kraften som fjädern uträttar, k är fjäderkonstanten, l(t) är fjäderns utsträckning, och l0 är fjäderns vilolängd.

Dämparna modellerades utifrån ekvation 4,

där F är kraften som dämparen uträttar, b är dämparkonstanten och l är fjäderns utsträckning.

1.3.2 Numerisk lösning av differentialekvationer

1.3.3 Euler

1.3.4 Runge Kutta

2 Metod

2.1 Förstudier

I förstudierna ingick dels framtagandet av en kraftekvation, dels utvecklingen av MSD-system i en och två dimensioner. Som verktyg användes MATLAB för att få en visuell återkoppling på kraftekvationen, med hjälp av programmets inbyggda renderingsfunktion.

2.1.1 Kraftekvation

2.1.2 Utveckling av MSD-system

2.1.2.1 En dimension

Då det enkla systemet med två sammankopplade partiklar utökades med fler partiklar och kopplingar, krävdes en mer överskådlig notation för att beskriva systemet. I figur 1 nedan introduceras den nya notationen av ett system med två sammankopplade massor.

Figur 1: Två sammankopplade partiklar

I MATLAB utvecklades först ett MSD-system i en dimension, där partiklar sammankopplades i serie enligt Figur 2 nedan.

Figur 2: Fyra sammankopplade partiklar i serie

Genom att simulera detta system med Euler-metoden samt undersöka hur partiklarnas positioner förändrades över tid, kunde implementationen av den framtagna kraftekvationen enkelt verifieras.

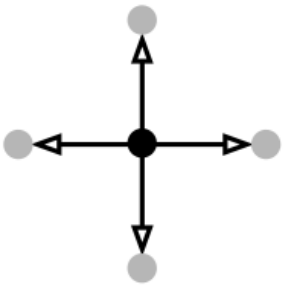
2.1.2.2 Två dimensioner

Nästa steg i utvecklingen av MSD-systemet var att implementera det i två dimensioner. Detta innebar en övergång från skalärer till vektorer i de framtagna ekvationerna.

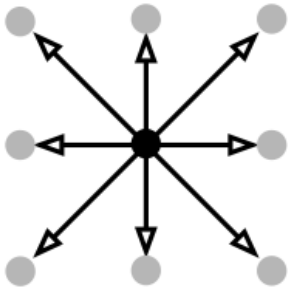
För att kunna generera godtyckligt stora MSD-system krävdes en regel för hur partiklarna skulle kopplas ihop med varandra. Två exempel på en partikels möjliga kopplingar visas i Figur 4 nedan.

2.1.3 Indexering av partiklar och kopplingar

Ett stort problem, som ovanstående stycke implicerar, var att ta reda på vilka partiklar som skulle höra till varje koppling. För att detta skulle kunna lösas för godtyckligt stora objekt, behövdes ett systematiskt sätt att tilldela kopplingarna deras partiklar. Detta löstes med en funktion som utifrån index till en viss koppling gav indexar till de två partiklar som den kopplade samman. Genom att låta

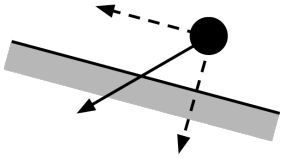


(a) Partiklar med fyra kopplingar

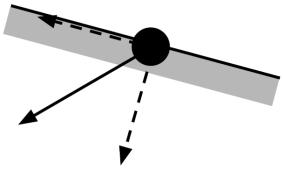


(b) Partiklar med åtta kopplingar

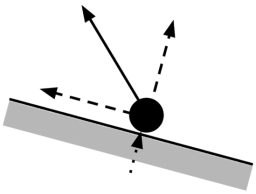
Figur 3: Två olika sätt att koppla samman partiklar



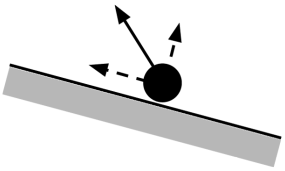
(a) Partikeln rör sig mot kollisionsplanet.



(b) Partikeln har kolliderat med planet.



(c) Partikelns position och hastighetsvektor ändras.



(d) Komponenterna multipliceras med dess respektive komponenter.

Figur 4: Två olika sätt att koppla samman partiklar

funktionen genomlöpa alla kopplingar innan simuleringen genomfördes kunde indexeringen användas för att finna de rätta konstanterna (massorna) och variablerna (positionerna och hastigheterna) för de kopplade massorna. En liknande funktion implementerades för att finna triangelindexar som användes för renderingen.

2.1.4 Interaktion med omgivning

Algoritm 1: Kollision

for *varje partikel* i **do**

if *partikeln befinner sig under kollisionsplanet:* **then**

- Spegla hastighetsvektorn i kollisionsplanet, se Figur 4c.
- Ortogonalprojicera positionsvektorn på kollisionsplanet, se Figur 4c.
- Multiplicera den komponent av hastighetsvektorn som är parallell med kollisionsplanet med en friktionskoefficient, se Figur 4d
- Multiplicara den komponent av hastighetsvektorn som är ortogonal med kollisionsplanet med en elasticitetskoefficient, se Figur 4d.

2.1.5 Implementering av kraftekvationen

Algoritm 2: Baserad på massorna

for *varje massa* i **do**
  for *varje massa* j*, som är kopplad till* i **do**
    - Beräkna den kraftvektor, f, som fjädern och dämparen mellan i och j påverkar massa i med.
    - Addera f till den totala kraften $F[i]$, som påverkar massa i.
  - Beräkna accelerationen för massa i genom att dividera den totala kraften $F[i]$ med dess massa.

Algoritm 3: Baserad på kopplingarna mellan massorna

for *varje koppling* c *(där* c *är kopplingen mellan massa* i *och* j*) :* **do**
  - Beräkna den kraft, f, som c påverkar massa i med.
  - Addera f till den totala kraften $F[i]$ som påverkar massa i.
  - Subtrahera f från den totala kraften $F[j]$ som påverkar massa j.

for *varje massa* i **do**
  - Beräkna accelerationen för massa i genom att dividera den totala kraften $F[i]$ med dess massa.

2.2 Implementering

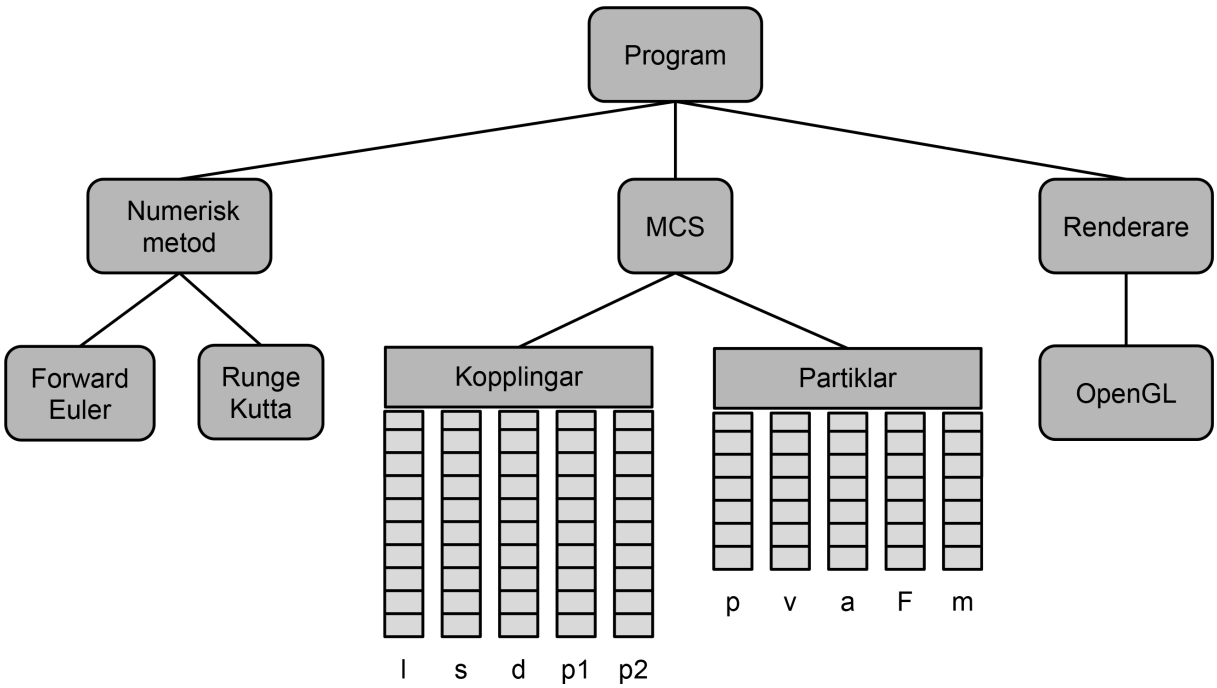
2.2.1 Grundläggande systemarkitektur

Att övergå till ett objektorienterat system innebar att en systemarkitektur behövde tas fram. För att ge en grundläggande förståelse av systemarkitekturen beskrivs denna med ett blockdiagram där de primära modulerna är separerade från varandra. Se Figur 5.
Klassen MCS (Mass Connection System) är den klass som lagrar all den huvudsakliga datan. Den innehåller samtliga partiklar och kopplingar för ett system.

Alla partiklar identifieras med ett index, och definieras av följande information:

- p, position
- v, hastighet
- a, acceleration
- F, total momentan kraftpåverkan
- m, massa

Samtliga kopplingar identifieras med ett index, och definieras av följande information:



Figur 5: Övergripande systemarkitektur

- l, vilolängd
- k, fjäderkonstant
- b, dämparkonstant
- p1, index till den partikel som utgör kopplingens ena ändpunkt
- p2, index till den partikel som utgör kopplingens andra ändpunkt

Vad som också framgår i Figur Figur 5 är att all numerisk integrering beräknas i en separat modul vilket gjorde det enkelt att ändra integreringsmetod vid behov, då den var oberoende av resten av systemet.

2.2.2 Implementering av numeriska integreringsmetoder

2.2.2.1 Eulers stegmetod

2.2.2.2 Runge Kutta

2.2.3 Rendering

2.3 Stabilitets- och prestandatest

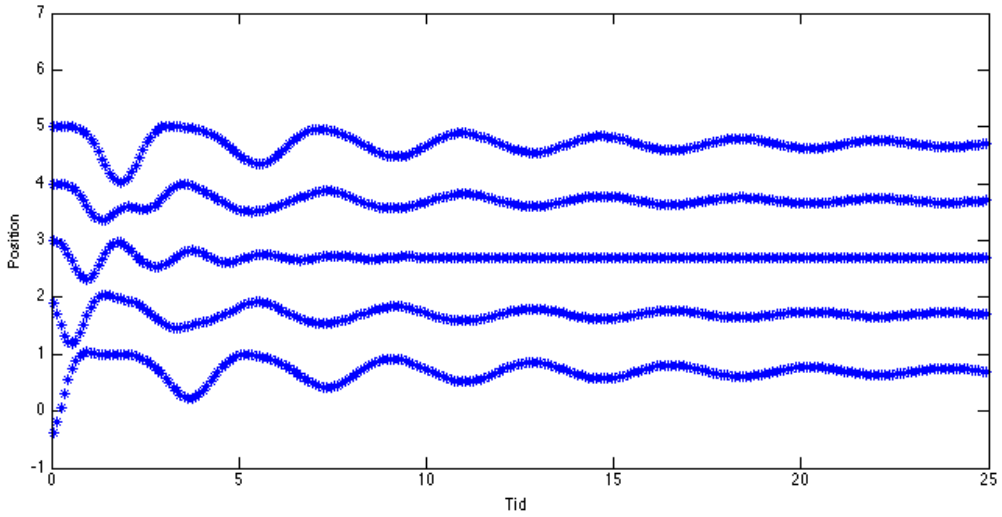
3 Resultat

3.1 Resultat från förstudier

3.1.1 En dimension

Resultatet från förstudierna i en dimension presenteras i Figur 6. Här illustreras fem sammankopplade partiklars samverkan efter att en av dem har givits en förskjuten startposition. Den kraft som bildas

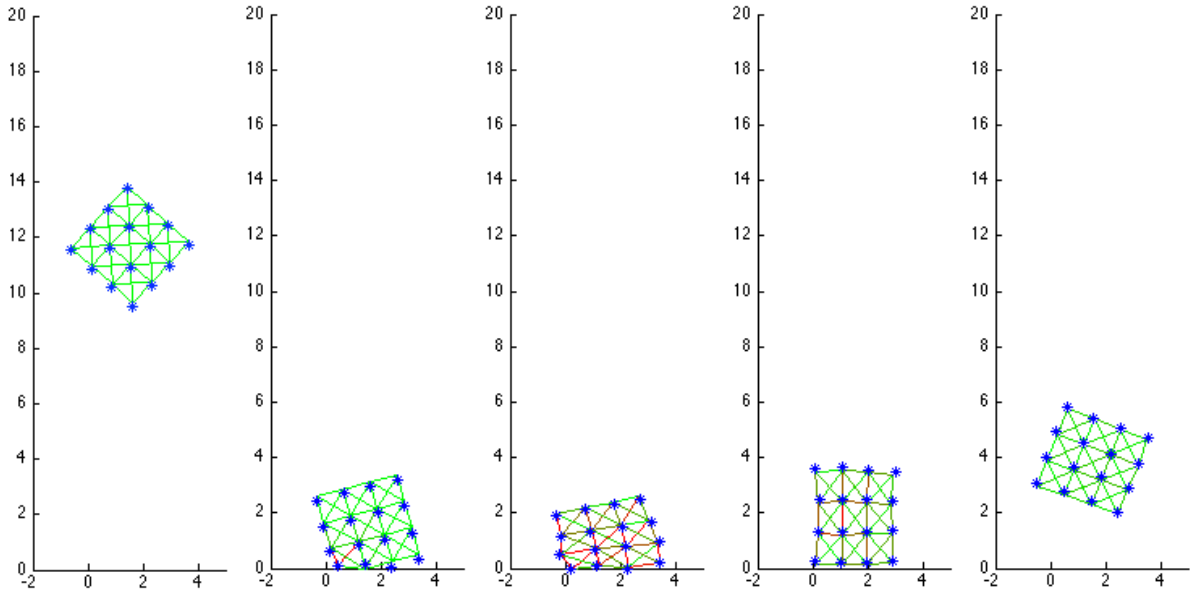
av denna förskjutning kan ses påverka de andra partiklarna och fortplantas genom systemet över tid. Krafterna ses också dämpas och tillslut avta, vilket stabiliserar systemet.



Figur 6: Endimensionellt MSD-system i MATLAB

3.1.2 Två dimensioner

I Figur 7 åskådliggörs resultatet av förstudierna i två dimensioner. Här illustreras ett 4x4 MSD-system som faller från sin startposition och därefter kolliderar med ett markplan. Fjädrarnas inverkande krafter visualiseras genom att låta färgen bero på kopplingens längd. Kopplingen är grön i sitt viloläge och färgen går mot rött ju mer längden avviker från denna.



Figur 7: Tvådimensionellt 4x4 MSD-system i MATLAB

3.2 Resultat från implementering

3.3 Resultat från stabilitets- och prestandatest

4 Diskussion

5 Slutsats

Referenser

[1] Shari Lawrence Pfleeger och Joanne M. Atlee, *Software Engineering, Fourth Edition, International Edition*, Pearson 2010

A Beskrivning av systemet

A.1 Systemkrav

A.1.1 Hårdvara

A.1.2 Mjukvara

A.2 Köra programmet