

# Simulering av Elastiska Material

Grupp 9

*Kalle Bladin*

*Erik Broberg*

*Emma Forsling Parborg*

*Martin Gråd*

13 mars 2014

## Sammanfattning

Fysikbaserad animering av elastiska material kan sägas bestå av två skilda delar: fysikalisk simulering och grafikrendering. Materialen kan fysikaliskt beskrivas på flera olika sätt, men denna rapport är begränsad till att behandla endast en modell. Realismen i den simulering som presenteras är av stor vikt, varför fysikalisk korrekthet eftersträvas, och all beräkning grundar sig i vedertagna fysikaliska samband. För simuleringen undersöks två olika integreringsmetoder, Eulers stegmetod och Runge-Kuttas metod, och deras respektive för- och nackdelar med hänsyn till snabbhet och stabilitet.

De förstudier som presenteras härleder den ekvation som simuleringen bygger på. Denna grundar sig i sin tur på de fysikaliska samband som presenteras i Bakgrund. Förstudierna tar också upp modelleringen av det system som simuleras, och även de numeriska metoder som är implementerade. Resultaten från förstudierna valideras och ligger som grund för den slutgiltiga implementeringen.

De slutliga simuleringar som presenteras visar att den valda modellen kan efterlikna ett antal olika material och deras egenskaper. Utritningen sker med en bestämd frekvens, och noggrannheten i integreringarna bestäms utifrån denna. Undersökningen av integreringsmetoderna visar att Eulers stegmetod är att föredra i detta sammanhang.

# Innehåll

<b>1</b>	<b>Inledning</b>	<b>1</b>
1.1	Bakgrund . . . . .	1
1.2	Syfte . . . . .	1
<b>2</b>	<b>Metod</b>	<b>1</b>
2.1	Förstudier . . . . .	1
2.1.1	Fysikalisk modellering . . . . .	1
2.1.2	Numerisk lösning av differentialekvationer . . . . .	2
2.1.2.1	Euler . . . . .	2
2.1.2.2	Runge Kutta . . . . .	2
2.1.3	Kraftekvation . . . . .	2
2.1.4	Utveckling av MSD-system . . . . .	2
2.1.4.1	En dimension . . . . .	2
2.1.4.2	Två dimensioner . . . . .	2
2.1.5	Indexering av partiklar och kopplingar . . . . .	2
2.1.6	Interaktion med omgivning . . . . .	3
2.1.7	Implementering av kraftekvationen . . . . .	4
2.2	Implementering . . . . .	4
2.2.1	Grundläggande systemarkitektur . . . . .	4
2.2.2	Implementering av numeriska integreringsmetoder . . . . .	5
2.2.2.1	Eulers stegmetod . . . . .	5
2.2.2.2	Runge Kutta . . . . .	5
2.2.3	Rendering . . . . .	5
2.3	Stabilitets- och prestandatest . . . . .	5
<b>3</b>	<b>Resultat</b>	<b>5</b>
3.1	Resultat från förstudier . . . . .	5
3.1.1	En dimension . . . . .	5
3.1.2	Två dimensioner . . . . .	6
3.2	Resultat från implementering . . . . .	7
3.3	Resultat från stabilitets- och prestandatest . . . . .	7
<b>4</b>	<b>Diskussion</b>	<b>7</b>
<b>5</b>	<b>Slutsats</b>	<b>7</b>
<b>A</b>	<b>Beskrivning av systemet</b>	<b>7</b>
A.1	Systemkrav . . . . .	7
A.1.1	Hårdvara . . . . .	7
A.1.2	Mjukvara . . . . .	7
A.2	Köra programmet . . . . .	7

## Figurer

1	Två sammankopplade partiklar . . . . .	2
2	Fyra sammankopplade partiklar i serie . . . . .	2
3	Två olika sätt att koppla samman partiklar . . . . .	3
4	Två olika sätt att koppla samman partiklar . . . . .	3

5	Övergripande systemarkitektur . . . . .	5
6	Endimensionellt MSD-system i MATLAB . . . . .	6
7	Tvådimensionellt 4x4 MSD-system i MATLAB . . . . .	6

## Tabeller

# 1 Inledning

Inledande text är alltid lite trevligt

## 1.1 Bakgrund

Grundidén till systemet ligger i tidigare presenterade studier kring liknande ämnen. Partikelsystem i sin helhet kan anta många olika former. Elastiska material täcker flera sådana då det går att göra en väldigt generell modell för att simulera många olika material. Bristen med många fysikaliska simuleringar är att modellen endast täcker ett väldigt specifikt område som till exempel tygsimulering eller endast hårda material.

## 1.2 Syfte

Denna rapport syftar till att undersöka hur verklighetstroga 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. Det är även viktigt att modellen ska kunna täcka många olika delar av newtonsk fysik; allt från elastiskt tyg till tärningskast med hårda tärningar.

# 2 Metod

## 2.1 Förstudier

I förstudierna ingick dels framtagandet av en kraftekvation, dels utvecklingen av MSD-system i en och två dimensioner. Detta grundar sig i den fysikaliska modelleringen samt numerisk lösning av differentialekvationer. Som verktyg användes MATLAB för att få en visuell återkoppling på kraftekvationen, med hjälp av programmets inbyggda renderingsfunktion.

### 2.1.1 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,  $a$  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 utträttar,  $k$  är fjäderkonstanten,  $l(t)$  är fjäderns utsträckning, och  $l_0$  är fjäderns vilolängd.

Dämparna modellerades utifrån ekvation 4,

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

## 2.1.2 Numerisk lösning av differentialekvationer

### 2.1.2.1 Euler

### 2.1.2.2 Runge Kutta

## 2.1.3 Kraftekvation

## 2.1.4 Utveckling av MSD-system

### 2.1.4.1 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.

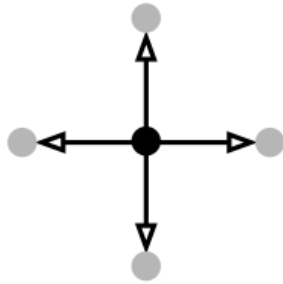
### 2.1.4.2 Två dimensioner

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.

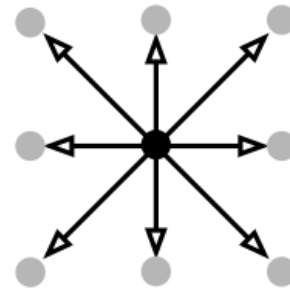
För att kunna generera godtyckligt stora MSD-system krävdes en regel för hur partiklarna skulle kopplas ihop med varandra. Två exempel på en partikels möjliga kopplingar visas i Figur 4 nedan.

## 2.1.5 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



(a) Partiklar med fyra kopplingar



(b) Partiklar med åtta kopplingar

Figur 3: Två olika sätt att koppla samman partiklar

index till en viss koppling gav indexar till de två partiklar som den kopplade samman. Genom att låta 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.

### 2.1.6 Interaktion med omgivning

---

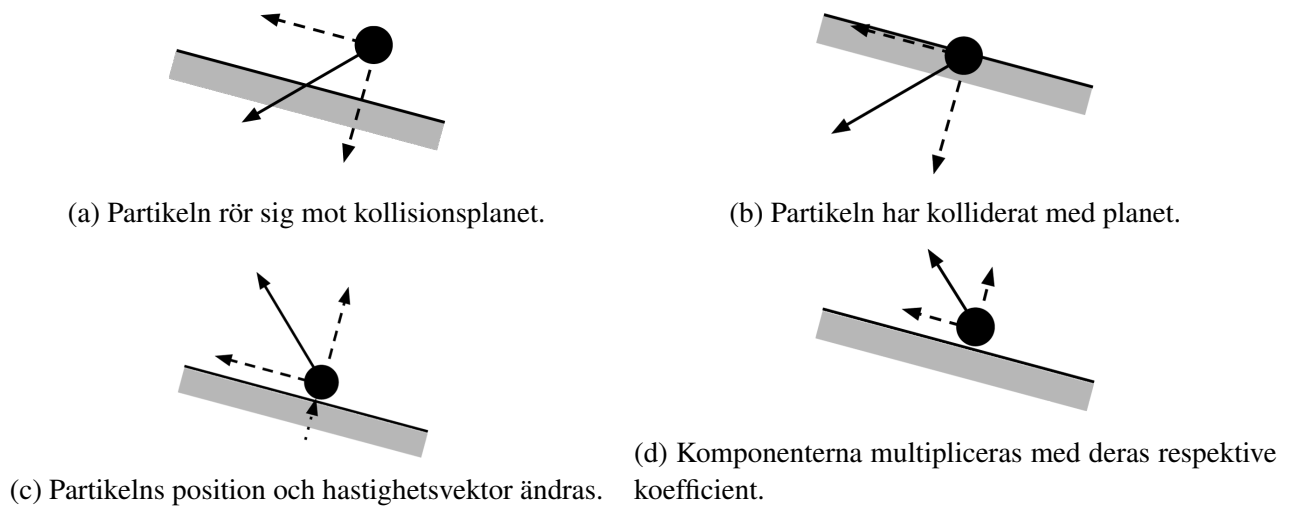
#### Algoritm 1: Kollisionshantering

---

**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
- Multiplicera den komponent av hastighetsvektorn, som är ortogonal med kollisionsplanet, med en elasticitetskoefficient, se Figur 4d.



Figur 4: Två olika sätt att koppla samman partiklar

### 2.1.7 Implementering av kraftekvationen

---

**Algoritm 2:** Algoritm 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:** Algoritm 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.
  
```

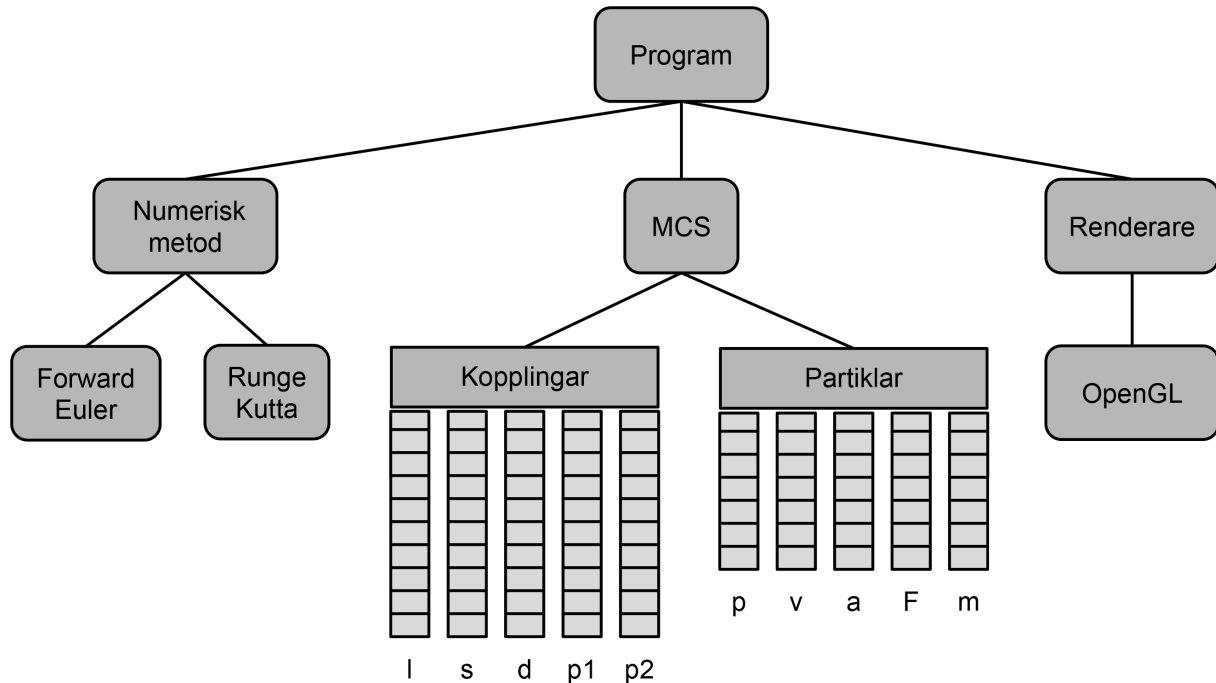
---



## 2.2 Implementering

### 2.2.1 Grundläggande systemarkitektur

Att övergå till ett objektorienterat system innebär att en systemarkitektur behövs 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.



Figur 5: Övergripande systemarkitektur

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:

- 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 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

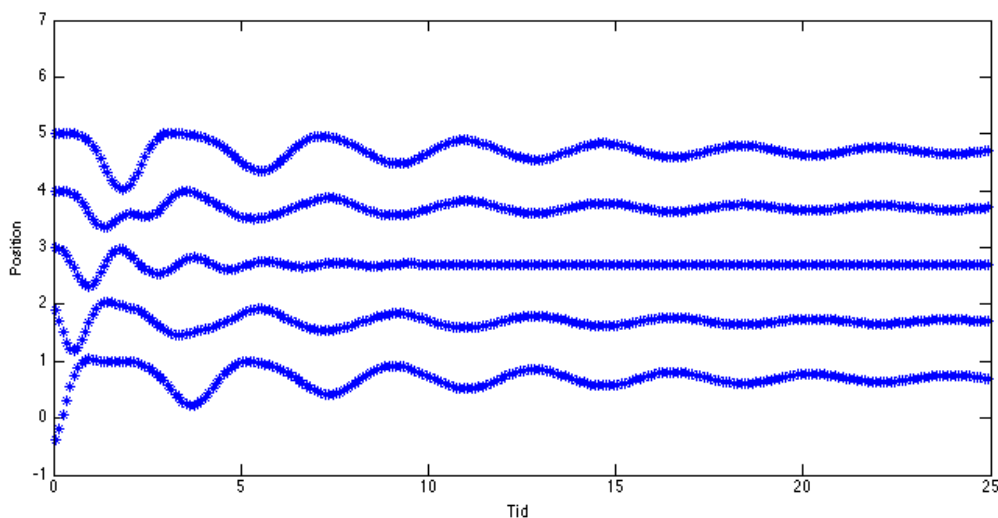
## 2.3 Stabilitets- och prestandatest

# 3 Resultat

## 3.1 Resultat från förstudier

### 3.1.1 En dimension

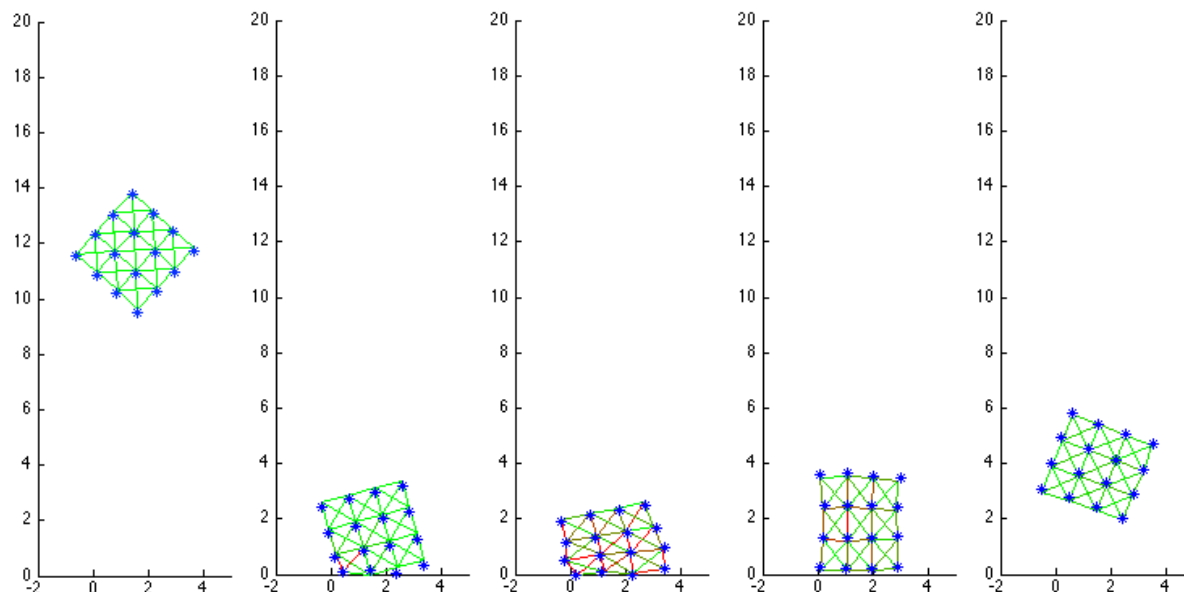
Resultatet från förstudierna i en dimension presenteras i Figur 6. Här illustreras fem sammankopplade partiklars samverkan efter att en av dem har givits en förskjuten startposition. Den kraft som bildas av denna förskjutning kan ses påverka de andra partiklarna och fortplantas genom systemet över tid. 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 inverkan 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