# Programmeren voor Natuur- en Sterrenkunde

Docent: Ronald Bruijn
R.Bruijn@uva.nl
Assistenten 2020:
Brían OFearraigh
Iris de Ruiter
Jelle Conijn
Simon Îlic
Thijs Juan van Eeden
William Torre

Versie 17
Cursus April 2020
Gebaseerd op versie 10 'Numerieke Natuurkunde' van M. Vreeswijk en I. van Vulpen en 'Numerieke Natuurkunde 1: Monte Carlo Method' van K. Kroeninger
Deze syllabus is verkrijgbaar via `http://numnat.mprog.nl`

# Contents

# Cursus 2020: speciale informatie

Deze cursus was ontwikkeld om gegeven te worden in klassenverband. Door de huidige situatie rondom het *coronavirus* en de opgelegde maatregelen, zal de cursus in ieder geval bij aanvang "op afstand", dus volledig met behulp van electronische communicatiemiddelen aangeboden worden.

## Reeds bestaande hulpmiddelen

Een aantal electronische hulpmiddelen worden al jaren gebruikt voor deze cursus, en daar zal niets aan veranderen. Je hebt bijvoorbeeld deze syllabus via de website `http://numnat.mprog.nl` verkregen. Daarnaast zal het inleveren van de opdrachten via de site gebeuren, de opdrachten zelf staan in deze syllabus. Je kan ook extra materiaal, benodigd voor sommige opdrachten (in het bijzonder hoofdstukken 7 en 10) op de site vinden. *Dit staat allemaal op de site en in deze syllabus uitgelegd.*

## Hoe hulp te krijgen/vragen te stellen

Er zijn twee kanalen via welke je hulp kan krijgen of vragen kan stellen :

- **Ed Forum** Ed is een forum waarop je (geschreven) vragen kunt stellen. De vragen op dit forum kunnen op elk ogenblik geplaatst worden. In eerste instantie zullen de assistenten ernaar kijken tijdens de ingeroosterde uren proberen de vragen in volgorde van plaatsing te behandelen. Het systeem laat ook toe om python code te formatteren en uit te voeren. Als je een vraag stelt - via $New\ Thread$ - moet je aangeven om welke opdracht het gaat. Er is ook de mogelijkheid vragen te stellen in de categorie "General". Deze categorie is ook geschikt voor vragen over de gang van zaken bij de cursus.
  **N.B.** Hoewel het systeem toelaat dat je vragen ook zichtbaar maakt voor je medestudenten, is het niet toegestaan dat jullie elkaar helpen. Wat natuurlijk wel kan is het bediscussieren van de uitleg en theorie in de syllabus. In eerste instantie dien je je vragen te beperken tot de assistenten en docent. Je wordt er hierbij aan herinnerd dat het een individueel vak is en dat de fraude- en plagiaat regeling van de UvA geldt.

- **Zoom** *één-op-één* Je kan gesprekken van elk 10 minuten aanvragen met één van de assistenten. Dit gebeurt via het $Calendly$ systeem, dat je verzoek tot gesprek inplant en je van een tijd en $Zoom$ link voorziet. Voor het gebruik van Zoom (zoom.us) is het nodig een account aan te maken. De eerste keer dat je het gebruikt zal je dat kunnen doen (of eerder). Houdt rekening met de extra tijd dat dit vergt.

- **e-mail naar docent** Voor zaken die niet direct betrekking hebben op de opdrachten, kan je altijd naar naar de docent mailen.

*Links naar deze hulpmiddelen staan op de website van dit vak (in de linkerbalk).*

# 1 Inleiding

In dit vak, *Programmeren voor Natuur- en Sterrenkunde 2*, zal je de kennis en ervaring met de programmeertaal *Python* die je opgedaan hebt bij het vak *Programmeren voor Natuur- en Sterrenkunde* toepassen (en uitbreiden) voor het oplossen van een aantal problemen in de Natuurkunde. De nadruk zal dan ook liggen op het vertalen van deze problemen naar een computer-programma dat een oplossing of meer inzicht kan verschaffen. Het vakgebied, waarbij met behulp van computers natuurkundige (of sterrenkundige) vraagstukken worden aangepakt, wordt ook wel eens *numerieke natuurkunde* genoemd.

## 1.1 Begripsbepaling/Afbakening

Numerieke Natuurkunde $\Leftrightarrow$ Computational Physics.

- Twee hoofdthema's :
  - In Klassieke Mechanica, Electro-Magnetisme, Quantummechanica, Stromingsleer etc. heb je steeds te maken met **vergelijkingen**. De toestand van een systeem en zijn verdere ontwikkeling in de tijd wordt beschreven door een (differentiaal)-vergelijking en bijbehorende **randcondities**.
    Gevraagd : los de vergelijking op!
    Soms kan dat langs analytische weg; vaak/meestal kan dat **niet** . Technieken nodig uit : *numerical analysis* .
  - In Statistische Fysica, Vaste-stoffysica etc. hebben we vaak te maken met **veel-deeltjes** problemen. De nadruk daarbij ligt niet zozeer op het oplossen van de onderliggende vergelijkingen, maar op het reconstrueren van 'meetbare' eigenschappen op basis van onderliggend collectief gedrag. Dit gebeurt door **computer-simulatie**.
    Wanneer de gesimuleerde processen een **kans**-element bevatten, spreken we van *Monte Carlo methoden*.
- De nadruk zal liggen op de eerste categorie problemen, waarvoor technieken nodig zijn uit : *numerical analysis*. Daarnaast zullen er ook problemen behandeld worden waar *Monte Carlo methoden* voor gebruikt kunnen worden. De opdrachten staan in deze syllabus. De laaste opdracht (heel sectie **??**) mag je vervangen door een vrije opdracht.
- De vrije opdrachten vereisen wat meer zelfstandigheid. Je moet echter wel bedenken dat deze opdrachten gemiddeld moeilijker zijn, maar vooral dat je deze opdrachten met weinig hulp zal moeten afronden. Het ligt ook niet vast wat het eindresultaat is; kijk maar hoe ver je komt en interpreteer (Vergelijk met theorie) het resultaat zelf. Hier staat tegenover dat er flink wat bonuspunten (wel maximaal 1 voor de hele cursus) te verdienen zijn met deze keuzeopdrachten. Van alle opdrachten zijn er artikelen/verslagen aanwezig en beschikbaar via de docent met de relevante fysica en formules.

## 1.2 Literatuur : een keuze

- Boeken en Internet :

| | |
|---|---|
| C.F. Gerald et al. | Applied Numerical Analysis |
| T.Pang | An Introduction to Computational Physics |
| | www.physics.unlv.edu/∼pang/cp.html |
| S.E. Koonin et al. | Computational Physics |
| R.Landau et al. | Computational Problems for Physics |
| W.H.Press et al. | Numerical Recipes, the Art of Scientific Computing |
| N.J.Giordano | Computational Physics |
| | www.physics.purdue.edu/∼ng/comp-phys.html |
| J.L.Zachary | Introduction to Scientific Programming |
| | www.cs.utah.edu/∼zachary/IntroSciProg.html |

- Vaktijdschriften :
  Journal of Computational Physics
  Computer Physics Communications
  Computer in Physics

## 1.3 Onze filosofie

- Niet alles 'afleiden'; het is geen wiskunde-college.
  Wel : 'geef gevoel voor'.
- Behandel slechts een klein deel van de onderwerpen;
  per onderwerp :slechts een klein deel van de technieken.
- Voorbeelden/opdrachten die aansluiten bij andere colleges
- Aandacht voor 'kwaliteit' programmatuur

## 1.4 Overzicht van onderwerpen

Er is een uitgebreide reeks opdrachten beschikbaar.

- *Numerical Analysis*
  - Numerieke basistechnieken
  - Gewone differentiaal vergelijkingen van de 1e orde
  - Gewone differentiaal vergelijkingen van de 2e orde, type 'initial value'
  - Gewone differentiaal vergelijkingen van de 2e orde, type 'boundary value'
  - Partiële differentiaal vergelijkingen, type 'elliptisch'
  - Partiële differentiaal vergelijkingen, type 'parabolisch'
- *Monte Carlo Methoden*

- Toevalsgetallen
- Monte-Carlo integratie
- Dronkemans wandeling (Random Walk)
- 'File-probleem'

## 1.5 Voorkennis

In deze cursus wordt voorkennis aangenomen van de programmeertaal **Python**, zoals behandeld in het vak *Programmeren voor Natuur- en Sterrenkunde*
In het bijzonder is het nuttig om kennis over de volgende elementen paraat te hebben:

- Het datatype **list** is onontbeerlijk om geïndexeerde lijsten van data op te slaan. Hierbij kan de index refereren naar een ruimte of tijd coordinaat, een element uit een verzameling etc. Kijk bij het gebruik van lists of andere datastructuren dat je ze met de juiste grootte initialiseert en vervolgens niet de maximale index overschrijdt bij het toekennen van een waarde. Het is uiteraard mogelijk om complexere datastructuren te gebruiken, eventueel uit een van de vele python modules
- De **for loop** is handig om bepaalde bewerkingen herhaaldelijk uit te voeren, waarbij eventueel de opeenvolgende elementen uit een datastructuur worden gebruikt of gemanipuleerd.
- Het gebruik van **functies** is zeer wenselijk om de programma code te organiseren.

Verder is het handig om nog even naar de python sectie van *Programmeren voor Natuur- en Sterrenkunde* te kijken, te vinden op `https://progns.mprog.nl/` (onder 'other').

## 1.6 Laptops/Python installaties

De opdrachten dien je op een eigen meegebrachte laptop te maken. Je hebt dus een python distributie nodig. We raden het pakket *Anaconda* aan. Instructies voor het installeren kan je vinden op de site van *Programmeren voor Natuur- en Sterrenkunde*, te vinden op `https://progns.mprog.nl/naslag/installatie-computer`. We gebruiken python versie **3.7**.

## 1.7 Weergeven resultaten

Om de resultaten van de verschillende opgaven weer te geven, is het uitprinten van getallen niet de beste manier. Om de resultaten grafisch weer te geven zijn er grafische paketten beschikbaar voor python. Een van deze paketten is *matplotlib*. Documentatie van dit pakket is is te vinden op `matplotlib.org`. Om de handige verzamling plotfuncties *matplotlib.pyplot* te gebruiken moet dient deze natuurlijk geïmporteerd worden met

```
import matplotlib.pyplot as plt
```

of iets equivalents.

## 1.8 Testen van je programma

Bij een aantal opdrachten wordt er gevraagd om delen van je programma, bijvoorbeeld een functie, te testen. Deze tests horen bij de opdracht en je programma dient deze dus uit te voeren. Het is natuurlijk niet zo dat je alleen de gevraagde tests kan doen. Het is altijd handig om tussentijds onderdelen van je programma te testen.

## 1.9 Toetsing

De toetsing geschiedt aan de hand van de in te leveren opdrachten. Deze opdrachten dienen individueel gemaakt te worden. De plagiaatregeling van de UvA is van toepassing. `http://www.uva.nl/plagiaat`

De maximale score kan behaald worden als alle opdrachten zijn voltooid. Opdrachten 3 tot en met 6 tellen bij elkaar voor 25% mee. Opdrachten 7 en 8 bij elkaar voor 25% en opdracht 9 en 10 elk 25%. Er is maximaal 1 bonuspunt te halen met extra of origineel werk. De programmacode van de opdrachten dient voltooid en werkend ingeleverd te worden samen met de grafische resultaten.

Zoals boven beschreven, zijn de opdrachten in 4 sets verdeeld:

- **Set 1**: Hoofdstuk/Opdracht 3 t/m 6
- **Set 2**: Hoofdstuk/Opdracht 7 en 8
- **Set 3**: Hoofdstuk/Opdracht 9
- **Set 4**: Hoofdstuk/Opdracht 10

Voor de eerste 2 sets zijn elk 16 contacturen beschikbaar. Voor de laatste twee sets (9 en 10) zijn elk 12 contacturen beschikbaar. Elk van de sets heeft een eigen deadline, om 23.59u op de dag in de lijst beneden:

- **Set 1**: 10 april 2020
- **Set 2**: 30 april 2020
- **Set 3**: 10 mei 2020
- **Set 4**: 20 mei 2020

De code dient voorzien te zijn van commentaar. Een aantal tips:

- Begin je programma met je naam, studentnummer en programma naam.
- Geef een korte omschrijving van wat je programma doet.

- Licht toe wat je variabelen voorstellen en geef ze duidelijke namen.
- Geef een korte omschrijving bij elke functie en klasse. Hierbij moet je ook toelichten wat de argumenten zijn.
- Geef een korte omschrijving bij elk blok of paragraaf van je code. Bijvoorbeeld bij een for-loop waarin veel gebeurt.
- Geef niet op elke regel commentaar! Het is dus niet de bedoeling dat je bijvoorbeeld schrijft "tel 1 op bij i".
- Geef wel commentaar bij kleine manipulaties die niet zo voor de hand liggen
- Schrijf het commentaar vooral voor jezelf! Je moet later je programma terug kunnen lezen en begrijpen.
- Schrijf het commentaar terwijl je bezig bent, zodat je niet achteraf moet uitzoeken wat je ook alweer gedaan hebt.

Bekijk de webpagina voor meer tips.

### 1.9.1 Inleveren opdrachten

De opdrachten dienen electronisch ingeleverd te worden. Naast de programma code, moeten de grafische resultaten ook worden ingeleverd. Het inleveren geschiedt via :

`http://numnat.mprog.nl`

Ga naar het menu 'opdrachten' en volg vervolgens de link behorende bij de opdracht/hoofdstuk. Om een opdracht in te kunnen leveren, moet je ingelogd zijn met je UvA-id. Dit kan rechtsboven via *sign in*.
De programma-code die ingeleverd wordt,**dient direct uitvoerbaar te zijn** en de grafische weergaven in een gangbaar formaat (.pdf, .ps, .eps, .jpg, .png, .tif). Meerdere grafische resultaten kunnen eventueel bij elkaar in een bestand gestopt worden met (g)zip of tar.
Het laatst voor de deadline ingeleverde werk telt.

### 1.9.2 Feedback

Na het inleveren van een set opdrachten wordt deze nagekeken. De student-assistenten kunnen je in de volgende contact-momenten vertellen hoe je de opdrachten gedaan hebt en hoe je eventueel je werk kan verbeteren (als dat nodig is). Zorg dat je weet wie je vaste assistent is.

## 1.10 Overige informatie

Deze syllabus is te vinden op `http://numnat.mprog.nl`. Op deze site is ook hulpmateriaal en andere informatie aangaande deze cursus te vinden.

# 2 Basistechnieken

- Een aantal van deze technieken zijn al aan de orde gekomen bij *Programmeren voor Natuur- en Sterrenkunde* . In dit hoofdstuk worden ze nog eens overzichtelijk weergegeven. Ook zijn er wat nieuwe technieken bij die je later nodig zult hebben.

- Dit hoofdstuk bevat *geen opgaven* : Je hebt er al wat aan gedaan bij vak *Programmeren voor Natuur- en Sterrenkunde*

## 2.1 Numeriek differentiëren

- Probleemstelling ( functie van 1 variabele).
  Gegeven $f(x)$ op $(a \leq x \leq b)$
  Bereken $f^{'}(x)$ op $(a \leq x \leq b)$

- Verdeel interval $(a, b)$ in $N$ intervallen van gelijke lengte $h$ .
  Coordinaten van de intervalgrenzen (**roosterpunten**) zijn :

$$x_i = a + ih \quad (i = 0, 1, 2....., n)$$

- Representeer $f(x)$ in de vorm van een **tabel** :

$$f_i = f(x_i) \quad (i = 0, 1, 2, ....., n)$$

- Gebruik de Taylor-reeks-ontwikkeling :

$$\boxed{f(x_i + h) = f(x_i) + hf^{'}(x_i) + \frac{h^2}{2!}f^{''}(x_i) + \mathcal{O}(h^3)} \tag{2.1}$$

  In onze notatie :

$$\boxed{f_{i+1} = f_i + hf_i^{'} + \frac{h^2}{2!}f_i^{''} + \mathcal{O}(h^3)} \tag{2.2}$$

- Voorwaartse 2-punts benadering :

$$\boxed{f_i^{'} \simeq \frac{f_{i+1} - f_i}{h} + \mathcal{O}(h)} \tag{2.3}$$

- Achterwaartse 2-punts benadering :

$$\boxed{f_i^{'} \simeq \frac{f_i - f_{i-1}}{h} + \mathcal{O}(h)} \tag{2.4}$$

- 3-punts benadering :

$$\boxed{f_i^{'} \simeq \frac{f_{i+1} - f_{i-1}}{2h} + \mathcal{O}(h^2)} \tag{2.5}$$

- Benadering 2de afgeleide :

$$\boxed{f_i'' \simeq \frac{f_{i+1} - 2f_i + f_{i-1}}{h^2} + \mathcal{O}(h^2)} \tag{2.6}$$

- Merk op :
  - De **precisie** van de benadering neemt toe naarmate $h$ afneemt. De 'prijs' die je daarvoor betaalt is extra rekentijd ten gevolge van de toename van $N$
  - De **precisie** is (mede) afhankelijk van het gebruikte recept.
  - Overigens is er ook nog sprake van eem 'computer'-nauwkeurigheid.

## 2.2 Numeriek integreren

- Probleemstelling ( functie van 1 variabele).
  Gegeven $f(x)$ op $(a \leq x \leq b)$
  Bereken
$$\int_a^b f(x)dx$$

- Verdeel interval $(a, b)$ in $N$ intervallen van gelijke lengte $h$ en representeer $f(x)$ in de vorm van een **tabel** :
$$f_i = f(x_i) \quad x_i = a + ih \quad (i = 0, 1, 2, ....., N)$$

- Schrijf de integraal als :
$$\int_a^b f(x)dx = \sum_{i=0}^{N-1} \int_{x_i}^{x_{i+1}} f(x)dx$$

- Trapezium-regel (lineaire benadering) :
  Benader $f(x)$ op het interval $(x_i, x_{i+1})$ door :
$$f(x) = \frac{f_{i+1} + f_i}{2} + \mathcal{O}(h)$$
  dan volgt :
$$\boxed{\int_a^b f(x)dx \simeq \frac{h}{2}(f_0 + 2f_1 + 2f_2..... + 2f_{N-1} + f_N) + \mathcal{O}(h^2)} \tag{2.7}$$

- Simpson-regel (parabolische benadering, $N$ even):
  Benader $f(x)$ op het interval $(x_{i-1}, x_{i+1})$ door een parabool door $(f_{i-1}, f_i, f_{i+1})$
$$f(x) = f_{i-1} + (x - x_{i-1})\frac{f_i - f_{i-1}}{h} +$$
$$+\frac{(x - x_{i-1})(x - x_{i-1} - h)}{2}\frac{(f_{i+1} - 2f_i + f_{i-1})}{h^2} + \mathcal{O}(h^3)$$
  dan volgt :
$$\boxed{\begin{aligned}\int_a^b f(x)dx = \frac{h}{3}(f_0 + f_N) + \frac{4h}{3}(f_1 + f_3.... + f_{N-1}) + \\ + \frac{2h}{3}(f_2 + f_4.... + f_{N-2}) + \mathcal{O}(h^4)\end{aligned}} \tag{2.8}$$

- Merk (weer) op :
  - De **precisie** van de benadering neemt toe naarmate $h$ afneemt. De 'prijs' die je daarvoor betaalt is extra rekentijd ten gevolge van de toename van $N$
  - De **precisie** is (mede) afhankelijk van het gebruikte recept.

## 2.3 Oplossen van niet-lineaire vergelijkingen

- Probleemstelling ( functie van 1 variabele).
  Gegeven $f(x)$ op $(a \leq x \leq b)$
  Bereken de waarde van $x$ op het interval $(a, b)$ waarvoor $f(x) = 0$

- we behandelen $enkele$ (eenvoudige) methoden, die we bij de verdere opgaven nodig zullen hebben. deze methoden vereisen dat de functie $monotoon$ is op het interval $(a, b)$.

- **De bisectie methode**.
  Deze methode veronderstelt dat de functiewaarden op de uiteinden van het $zoek\text{-}interval$ $(a, b)$ een $tegengesteld\ teken$ hebben. We weten dan dat het nulpunt moet liggen $tussen$ $(x_l = a)$ en $(x_r = b)$. De methode gaat dan als volgt :

  - Bereken de functiewaarde op

    $$x_m = \frac{x_l + x_r}{2}$$

  - Wanneer de functiewaarde in $x_m$ hetzelfde teken heeft als in $x_l$

    $$vervang \quad x_l \quad door \quad x_m$$

    Wanneer de functiewaarde in $x_m$ hetzelfde teken heeft als in $x_r$

    $$vervang \quad x_r \quad door \quad x_m$$

  - Ga hiermee door $totdat$ de $lengte$ van het interval $(x_l, x_r)$ zodanig klein wordt, dat de positie van het nulpunt $voldoende\ nauwkeurig$ vastligt.
  - In het hiernavolgende $voorbeeld$ wordt , langs $iteratieve\ weg$ de wortel uitgerekend van een ingevoerd getal met de bisectie methode.

```
"""
Dit programma lost de volgende vergelijking op:
x^2-wortel^2 = 0
De wortel^2 is een gevraagde input door het programma.
"""
eps = 0.0001 # minimum intervallengte

"""
Inlezen van wortelsq
"""
wortelsqstr = raw_input("Geef wortelsq in: ")
print "We lossen op: x^2 -",wortelsqstr," = 0"
wortelsq = float(wortelsqstr)
"""
begingrenzen  zoekinterval
"""
xl = 0.
xr = wortelsq
if xr < 1. :
    xr = 1.

"""
Hier begin de while loop
"""
while (xr - xl) > eps :
    xm = (xl + xr) /2.  # midden van interval
    fl = xl*xl-wortelsq # de vgl. in xl
    fm = xm*xm-wortelsq # de vgl. in xm

    if fl*fm <= 0.:     # indien teken wisselt zijn we voorbij de oplossing
        xr = xm         # rechtergrens omlaag
    else :
        xl = xm         # linkergrens omhoog

    print "xl xr", xl, xr

"""
eindresultaat
"""
wortel = (xr+xl)/2.
onzekerheid = (xl-xr)/2.

print " wortel = ", wortel," onzekerheid = ", onzekerheid
```

- **De secant methode**
  Ook deze methode veronderstelt dat de functiewaarden op de uiteinden van het $zoek\text{-}interval\ (a, b)$ een $tegengesteld\ teken$ hebben.

  $$f_l = f(a) \qquad f_r = f(b) \qquad f_l \star f_r < 0$$

  De methode gaat dan als volgt :

  - Construeer een rechte lijn door ( $x_l$ , $f_l$) en ( $x_r$ , $f_r$) , en bereken het snijpunt met de x-as
    $$x_s = x_l - f_l \frac{x_r - x_l}{f_r - f_l}$$
  - Bereken de functiewaarde $f_s$ in $x_s$
  - Wanneer de functiewaarde in $x_s$ hetzelfde teken heeft als in $x_l$

    $$vervang \quad x_l \quad door \quad x_s \quad en \quad vervang \quad f_l \quad door \quad f_s$$

    Wanneer de functiewaarde in $x_s$ hetzelfde teken heeft als in $x_r$

    $$vervang \quad x_r \quad door \quad x_s \quad en \quad vervang \quad f_r \quad door \quad f_s$$

  - Om dit iteratieve proces te $beeindigen$ zijn er twee mogelijkheden :
    * $Stop$ zodra de absolute waarde van $f(x_s)$ kleiner wordt dan 'een of andere' $afsnij\text{-}waarde$ (dicht bij $0$).
    * $Stop$ zodra $x_s$ (bijna) niet meer $verandert$.

- **Methode vanuit 1 startpunt**
  Soms is de situatie zodanig, dat er voor een functie $geen\ start\text{-}interval$ ter beschikking is, maar slechts $1\ startpunt$. het is dan niet mogelijk om direct de bisectie- of de secant- methode toe te passen. we gaan dan als volgt te werk :

  - Noem het $startpunt\ x_0$
  - Kies een $stapgrootte\ h$, bijv. $h = 0.01\ x_0$
  - Bereken de functiewaarde $f_0$ in het startpunt $x_0$
  - Bereken de functiewaarde $f_1$ in $(x_1 = x_0 + h)$
  - Wanneer $f_1 \star f_0 < 0$ is, hebben we het startinterval te pakken.
  - Wanneer $f_1 \star f_0 > 0$ is, vergelijken we $|f_1|$ met $|f_0|$ .
    Als $|f_1| < |f_0|$ dan 'lopen' we kennelijk in de goede richting : ga door met 'stapjes' $h$, totdat het startinterval is gevonden.
    Als $|f_1| > |f_0|$ dan 'lopen' we kennelijk in de verkeerde richting : keer het teken van $h$ om, en ga door met 'stapjes' $h$, totdat het startinterval is gevonden.
  - Zodra het startinterval is gevonden, kunnen we overschakelen op de bisectie- of de secant-methode.

- **opmerkingen** : Het is duidelijk dat $alle$ beschreven methoden slechts 'werken' wanneer ze worden 'voorzien' van bruikbare $start$- informatie over de functie waarvan het nulpunt moet worden bepaald. Een $analyse\ vooraf$ van de grootheden die in een bepaald probleem een rol spelen, is hiervoor noodzakelijk.

## 2.4 Numerieke Precisie

In computers worden getallen gerepresenteerd door een eindig aantal bits. Vastekomma/Gehele getallen (Engels: *Integers*) en zwevendekommagetallen (wetenschappelijke notatie, of Engels: *floating-point*) hebben elk een andere representatie. Typisch gebruiken we in python 32-bit vastekomma of 64-bits (dubbele precisie/ *double precision*) zwevendekommagetallen. Deze eindige aantallen bits leiden tot beperkingen in de kleinste en grootste getallen die gerepresenteerd kunnen worden en in een limiet op de nauwkeurigheid. Dubbele preciesie getallen hebben ongeveer 15-16 significante getallen en een bereik tussen de $4.9{\cdot}10^{-324}$ en $1.8{\cdot}10^{308}$. De machine-nauwkeurigheid (Engels: *machine precision*) is gedefinieerd als het grootste getal dat bij 1.0 opgeteld kan worden zonder dat de opgeslagen waarde te veranderen. Dus als $1.0_c$ de computer representatie van 1.0 is en $\epsilon_m$ de machine precisie :

$$1.0_c + \epsilon_m := 1.0_c \tag{2.9}$$

# 3 Herhaling: priemgetallen

Deze eerste opdracht dient ter verfrissing van de kennis van de programmeertaal python en van de werkomgeving. De opdracht is rechtstreeks overgenomen uit de python cursus, maar dient wel weer gedaan te worden.

## 3.1 Opdracht 1

De opdracht luidt: Schrijf een programma $primes.py$ dat het duizendste priemgetal berekent en print. Print ook de lijst van alle 1000 priemgetallen.
Computing hints:

- een manier om te testen of een getal $a$ een veelvoud is van getal $b$ ($b$ deelt $a$ met rest 0) is het gebruik van de %-operator. In python code geeft $a\%b$ de rest ($8\%3$ is 2). Controleer de werking op de command-line.
- Gebruik list om reeksen getallen te bewaren. Bekijk de documentatie.

Hoewel een computer je in staat stel om snel te rekenen is het toch belangrijk om voor elk probleem de optimale strategie te bepalen. Hier bijvoorbeeld:

- Behalve 2 zijn even getallen nooit een priemgetal
- Verzin hoe je per priem-kandidaat bijhoudt of het wel/niet priem is als je over kandidaat delers heenloopt. Bedenkt van tevoren hoe je de lijst met gevonden priemgetallen gaat opslaan.
- Wanneer kan je stoppen ? Als je wilt bepalen of 37 een priemgetal is, welke kandidaat delers bekijk je voordat je zeker weet dat het een priemgetal is ?
- Print voor elke kandidaat informatie zodat je weet waar je bent in de berekening en je zit of de computer ook echt jouw strategie volgt.
- Zorg dat het programma stopt bij het 1000ste priemgetal. Bedenk dat je programma waarschijnlijk niet het eerste priemgetal genereerd (2).

Als je wilt controleren of je programma goed werkt, kan je je gevonden lijst priemgetallen hier matchen met een lijst bekende priemgetallen:

`http://primes.utm.edu/lists/small/1000.txt`

## 3.2 Opdracht 2:

Welke getallen (onder $n = 1000$, dus onder het 1000-ste priemgetal) vormen de langste reeks aaneengesloten niet-priemgetallen ? Start met de code uit vraag 1.

## 3.3 in te leveren

**Lever in:**

- *primes.py*, dat de volgende uitkomsten produceert bij uitvoering:

  - Het duizendste priemgetal.
  - Volledige lijst van de eerste duizend priemgetallen.
  - De langste aaneengesloten reeks niet-priemgetallen onder het duizendste priemgetal.

**N.B.: :** Het ingeleverde programma moet uitvoerbaar zijn en de uitkomsten duidelijk weergeven, bijvoorbeeld met een toelichting als *Het duizendste priemgetal is : ....* Alleen maar getallen naar het scherm sturen is niet de bedoeling!

# 4 Gewone Differentiaal Vergelijkingen (ODE's) 1e orde

## 4.1 Probleemstelling

- Probleemstelling:

| | |
|---|---|
| Onafhankelijke variabele | $x$ op $(a \leq x \leq b)$ |
| Afhankelijke variabele | $y(x)$ |
| Vergelijking | $y^{'} = f(x, y)$ |
| Randconditie | $y(a)$ gegeven |
| Gevraagd | $y(x)$ op $(a \leq x \leq b)$ |

- Voorbeeld : radioactief verval van atoomkern ; het aantal deeltjes dat per tijdseenheid vervalt, wordt gegeven door :

$$\boxed{\frac{d}{dt}N(t) = -\lambda N(t)} \tag{4.1}$$

waarin $\lambda$ de levensduur van de betreffende kern is.

- Aanpak :
Verdeel interval $(a, b)$ in $N$ intervallen van gelijke lengte $h$ en representeer $y(x)$ en $f(x, y)$ in de vorm van **tabellen** :

$$y_i = y(x_i) \qquad x_i = a + ih \quad (i = 0, 1, 2, ....., n)$$

$$f_i = f(x_i, y_i) \qquad (i = 0, 1, 2, ....., n)$$

$y_0$ is gegeven door de randconditie.
Te bepalen : $y_i \quad (i = 1, 2, ...., n)$

## 4.2 numerieke methoden

- Gebruik weer de Taylor-reeks-ontwikkeling :

$$\boxed{y(x_i + h) = y(x_i) + hy^{'}(x_i) + \frac{h^2}{2!}y^{''}(x_i) + \mathcal{O}(h^3)} \tag{4.2}$$

In onze notatie :

$$\boxed{y_{i+1} = y_i + hy_i^{'} + \frac{h^2}{2!}y_i^{''} + \mathcal{O}(h^3)} \tag{4.3}$$

- Methode van Euler :

$$y_i^{'} = \frac{y_{i+1} - y_i}{h} + \mathcal{O}(h) = f_i$$

$$\boxed{\Longrightarrow y_{i+1} = y_i + hf_i + \mathcal{O}(h^2)} \tag{4.4}$$

- Methode van Adams-Bashforth :

$$y_{i+1} = y_i + \int_{x_i}^{x_{i+1}} y_i' dx = y_i + \int_{x_i}^{x_{i+1}} f(x,y) dx$$

Benader $f(x,y)$ via lineaire extrapolatie door $(x_{i-1}, y_{i-1})$ en $(x_i, y_i)$ :

$$f(x,y) \simeq \frac{x - x_{i-1}}{h} f_i - \frac{x - x_i}{h} f_{i-1} + \mathcal{O}(h^2)$$

$$\boxed{\Longrightarrow y_{i+1} = y_i + h\left(\frac{3}{2} f_i - \frac{1}{2} f_{i-1}\right) + \mathcal{O}(h^3)} \tag{4.5}$$

Merk op : doordat we bij de berekening van $y_{i+1}$ niet alleen gebruik maken van $y_i$ , maar ook van $y_{i-1}$, neemt de nauwkeurigheid toe.

- Methode van Runge-Kutta, 2e orde

$$y_{i+1} = y_i + \int_{x_i}^{x_{i+1}} f(x,y) dx$$

Benader de integraal

$$y_{i+1} = y_i + h f(x_{i+\frac{1}{2}}, y_{i+\frac{1}{2}}) + \mathcal{O}(h^3)$$

Gebruik Euler :

$$y_{i+\frac{1}{2}} = y_i + \frac{h}{2} f(x_i, y_i) + \mathcal{O}(h^2)$$

Dan wordt het recept :

$$\boxed{\begin{aligned} k &= h f(x_i, y_i) \\ y_{i+1} &= y_i + h f(x_i + \frac{h}{2}, y_i + \frac{k}{2}) + \mathcal{O}(h^3) \end{aligned}} \tag{4.6}$$

- Methode van Runge-Kutta, 4e orde :
Betere benadering van de integraal resulteert in het volgende recept

$$\boxed{\begin{aligned} k_1 &= h f(x_i, y_i) \\ k_2 &= h f(x_i + \frac{h}{2}, y_i + \frac{k_1}{2}) \\ k_3 &= h f(x_i + \frac{h}{2}, y_i + \frac{k_2}{2}) \\ k_4 &= h f(x_i + h, y_i + k_3) \\ y_{i+1} &= y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) + \mathcal{O}(h^5) \end{aligned}} \tag{4.7}$$

- Pas op !! de aangegeven orde van de fout betreft de **locale** fout.
De **globale** fout is een factor $h^{-1}$ groter.

- Merk (weer) op :
  - De **precisie** van de benadering neemt toe naarmate $h$ afneemt.
  - De **precisie** is (mede) afhankelijk van het gebruikte recept.

## 4.3 Opgave : radio-actief verval

### 4.3.1 de fysische achtergrond

- het radio-actief verval van een atoom wordt beschreven door

$$\boxed{\frac{d}{dt}N(t) = -\lambda N(t) \qquad met \qquad \lambda = \frac{\ln 2}{T_{\frac{1}{2}}}} \tag{4.8}$$

  waarin $T_{\frac{1}{2}}$ de halveringstijd is van het vervallende atoom.

- volgens de 'handboeken' bestaat er het volgende verval :

$$\boxed{{}^{76}Br \quad \rightarrow \quad {}^{76}Se} \tag{4.9}$$

  de halveringstijd van Broom is $16.1$ uur.
  het eindproduct Selenium is stabiel.

- we gaan uit van een beginsituatie waarin alleen Broom aanwezig is.
  de **opdracht** is om uit te rekenen hoeveel Broom en Selenium er aanwezig is, als functie van de tijd.

### 4.3.2 De formules

- noem de beginhoeveelheid Broom : $N_0$

- noem de hoeveelheden Broom en Selenium als functie van de tijd : $N_1(t), N_2(t)$.
  noem de vervalsconstante van Broom : $\lambda_1$

- de hoeveelheid Broom, als functie van de tijd, wordt gegeven door :

$$\boxed{\frac{d}{dt}N_1(t) = -\lambda_1 N_1(t) \quad met \;\; N_1(t=0) = N_0} \tag{4.10}$$

- de hoeveelheid Selenium wordt verkregen door 'sommatie' van het Broom verval.

- de vergelijkingen voor $N_1$ heeft een analytische oplossing ;

$$\boxed{N_1(t) = N_0 \exp(-\lambda_1 t)} \tag{4.11}$$

- **Merk op** dat de differentiaalvergelijking bij deze opgave de vorm heeft van $y' = f(y)$ . in het 'recept' voor de Runge-Kutta methodes vervalt derhalve de afhankelijkheid van $x$.

### 4.3.3 De procedure/De opdracht

- bij de numerieke berekening nemen we steeds kleine stapjes in de tijd, waarbij we de hoeveelheid vervallend Broom berekenen.
  - gebruik als tijdstap $(0.01)*$ de halveringstijd van Broom.
  - neem 750 tijdstappen.
- dit resultaat wordt gebruikt om de 'resterende' hoeveelheid Broom te berekenen, evenals de hoeveelheid geproduceerd Selenium.
- gebruik de analytische oplossing om de precisie van de numerieke procedure te controleren. doe dit door op elk tijdstip de absolute waarde uit te rekenen van het verschil tussen de numerieke en de analytische waarde, waarbij dit verschil wordt gedeeld door de analytische waarde.
- gebruik eerst de methode van Euler, en dan de methode van Runge-Kutta (4e orde)
- om het resultaat van de berekeningen zichtbaar te maken, is het 'uitprinten' van getallen niet erg geschikt : gebruik een **grafisch pakket** .

### 4.3.4 In te leveren

**Lever in** :

- *uw programma-code (decay.py)*
  Dit programma moet de volgende uitkomsten produceren bij uitvoering:
  - Grafische weergave van de hoeveelheid Broom en Selenium als functie van tijd voor één van de twee methoden (Euler/Runge-Kutta).
  - Grafische weergave van de precisie van de hoeveelheid Broom als functie van tijd voor beide methoden (Euler en Runge-Kutta).
- *De grafische resultaten als boven beschreven.* :

# 5 Gewone Differentiaal Vergelijkingen (ODE's) 2e orde ; 'initial value type'

## 5.1 Probleemstelling

- Probleemstelling:

| | |
|---|---|
| Onafhankelijke variabele | $x$ op $(a \leq x \leq b)$ |
| Afhankelijke variabele | $y(x)$ |
| Vergelijking | $y^{''} = f(x, y, y^{'})$ |
| Randcondities | $y(a) = y_0$ en $y^{'}(a) = y_0^{'}$ gegeven. |
| Gevraagd | $y(x)$ op $(a \leq x \leq b)$ |

- Voorbeeld : de vergelijking van Newton uit de Klassieke Mechanica

$$\boxed{\frac{d^2y(t)}{dt^2} = \frac{F(y)}{m}} \tag{5.1}$$

als de plaats $y(t)$ en de snelheid $y^{'}(t)$ op het tijdstip $t = 0$ gegeven zijn, en natuurlijk ook de massa $m$ en de kracht $F$ , dan ligt het verdere verloop van de plaats en de snelheid vast, en kan worden berekend.

- wanneer de randcondities op deze wijze zijn gegeven (beide in $hetzelfde$ punt), spreken we van '$initial\ value\ type$'.

## 5.2 Numerieke Methoden

- vergelijkingen van deze vorm, en met deze randcondities, kunnen worden **herleid** tot twee gewone differentiaal- vergelijkingen van de 1e orde, met ieder hun eigen randconditie.

- introduceer de snelheid als 'hulp'variabele. de vergelijkingen worden dan :

$$\boxed{\frac{dy(t)}{dt} = v(t) \qquad \frac{dv(t)}{dt} = \frac{F(y)}{m}} \tag{5.2}$$

met $y(t = 0)$ en $v(t = 0)$ gegeven door de randcondities.

- de numerieke aanpak wordt nu als volgt :
  - introduceer een tijdstap $\delta t$, en discretiseer tijd, plaats, snelheid en kracht :

$$t_i = t_0 + i\delta t \quad y_i = y(t_i) \quad v_i = v(t_i) \quad F_i = F(y_i) \quad (i = 0, 1, 2, ..)$$

  - de vergelijkingen worden dan :

$$\boxed{\frac{dy_i}{dt} = v_i \qquad \frac{dv_i}{dt} = \frac{F_i}{m}} \tag{5.3}$$

met $y_0$ en $v_0$ gegeven door de randcondities.

- de rekenprocedure wordt dan :
  - bereken $y_1$ uit $y_0$ , gebruikmakend van $v_0$
  - bereken $v_1$ uit $v_0$ , gebruikmakend van $F_0$ en $m$.
  - bereken $y_2$ uit $y_1$ , gebruikmakend van $v_1$.
  - etc.
- merk op dat deze aanpak slechts 'werkt' omdat de randcondities van plaats en snelheid **op hetzelfde tijdstip** gegeven zijn.

## 5.3 Opgave : Beweging deeltje bij potentiaal (klassiek)

### 5.3.1 De fysische achtergrond

- we nemen een probleem uit de Klassieke Mechanica : de 1-dimensionale beweging van een deeltje in aanwezigheid van een **potentiaalberg** $V(x)$
- de vergelijking van Newton heeft dan de vorm :

$$\boxed{\frac{d^2x(t)}{dt^2} = -\frac{1}{m}\frac{dV(x)}{dx}} \tag{5.4}$$

  waarbij $m$ de massa van het deeltje is.

- we geven de potentiaalberg de vorm van een 'Gaussische verdeling' rond het punt $x = x_{pot}$ , met een $breedte = 1$ , en een maximale hoogte $V_{max}$ :

$$\boxed{V(x) = V_{max} \exp\left[-\frac{(x - x_{pot})^2}{2}\right]} \tag{5.5}$$

- we starten op het tijdstip $t = 0$ met een deeltje met een gegeven beginsnelheid $v = v_0$, en op een plaats $x = x_0$ waar de potentiaal $V(x_0) \simeq 0$.
  we laten het deeltje lopen in de richting van de potentiaalberg, en berekenen het verloop van plaats en snelheid als functie van de tijd.
- voor het gemak kiezen we de massa van het deeltje ($m = 1$). daardoor wordt de snelheid gelijk aan de impuls.

### 5.3.2 De formules

- We kunnen de (2e orde) vergelijking van Newton schrijven als een combinatie van twee (1e orde) vergelijkingen :

$$\boxed{\begin{aligned} \frac{dx(t)}{dt} &= p(t) \\ \frac{dp(t)}{dt} &= -\frac{dV(x)}{dx} = V_{max}\,(x - x_{pot}) \exp\left[-\frac{(x - x_{pot})^2}{2}\right] \end{aligned}} \tag{5.6}$$

- Na discretisatie van tijd, plaats, en snelheid/impuls worden de vergelijkingen

$$\boxed{\begin{aligned}\frac{dx_i}{dt} =& p_i \\ \frac{dp_i}{dt} =& -\frac{dV(x=x_i)}{dx} = V_{max}\left(x_i - x_{pot}\right)\exp\left[-\frac{(x_i - x_{pot})^2}{2}\right]\end{aligned}} \tag{5.7}$$

met $x_0$ en $p_0$ gegeven.

- een praktische opmerking : opdat het programma de potentiaalberg $goed\ ziet$ , moet de $tijdstap$ in combinatie met de beginimpuls zodanig worden gekozen, dat de $verplaatsing$ van het deeltje per tijdstap aanzienlijk kleiner is dan de breedte van de potentiaalberg.

$$\boxed{vdt << 1} \tag{5.8}$$

### 5.3.3 De opdracht

- interessante grootheden bij dit probleem zijn niet alleen de ontwikkeling van plaats en impuls van het deeltje, als functie van de tijd, maar ook de ontwikkeling van de kinetische energie $E_{kin}$, de potentiele energie $E_{pot}$ en de totale energie $E_{tot}$ :

$$\boxed{E_{kin}(t) = \frac{p^2(t)}{2} \qquad E_{pot}(t) = V\left(x(t)\right) \qquad E_{tot}(t) = E_{kin}(t) + E_{pot}(t)} \tag{5.9}$$

- de totale energie moet **behouden** zijn, en gelijk aan de beginenergie :

$$\boxed{E_{tot}(t) = E_{tot}(t=0) = \frac{p^2(t=0)}{2} = constant} \tag{5.10}$$

- verder zijn er twee interessante situaties te ondescheiden :
  - de beginenergie is $kleiner$ dan de hoogte van de potentiaalberg
  - de beginenergie is $groter$ dan de hoogte van de potentiaalberg
- de **opdracht** luidt :
  - bereken voor **beide** 'soorten' beginenergieen, het verloop in de tijd van plaats, impuls en de energieen. Gebruik hierbij de methode van Adams-Bashforth.
  - gebruik een functie om de afgeleide van te potentiaal te berekenen.
  - maak het resultaat langs **grafische** weg zichtbaar.

### 5.3.4 In te leveren

**Lever in** :

- *programma-code (particle1dim.py)* welke bij uitvoering de volgende resultaten produceert :
  - Grafische weergaven van de plaats,impuls en drie energieën als functie van tijd voor beide situaties.
- *grafische resultaten*

# 6 Het vibratie-spectrum van het waterstof molecuul

## 6.1 Fysische probleembeschrijving

- We beschouwen een molecuul, bestaand uit twee gelijksoortige atomen (bijv. waterstof $H_2$ of zuurstof $O_2$).
- De potentiaal tussen beide atomen (ten gevolge van de electrische wisselwerkingen tussen de atoomkernen en de electronen) heeft de volgende componenten :
  - een **afstotende** component, op **zeer kleine** afstanden.
  - een **aantrekkende** component, op **grote** afstanden.
- een voorbeeld van een dergelijke potentiaal is afkomstig van Lennard-Jones

$$\boxed{V(r) = 4V_0 \left[ \left(\frac{a}{r}\right)^{12} - \left(\frac{a}{r}\right)^{6} \right]} \tag{6.1}$$

  - $V(r) = 0$ voor $r = a$
  - op kleine afstanden ($r < a$) is $V(r)$ positief : de atomen stoten elkaar af.
  - op afstanden ($r > a$) is $V(r)$ negatief : de atomen trekken elkaar aan.
  - de minimumwaarde van de potentiaal $= -V_0$. deze wordt bereikt bij

$$\boxed{r_{min} = 2^{\frac{1}{6}} a} \tag{6.2}$$

  - op grote afstanden ($r \gg r_{min}$) gaat de (negatieve) potentiaal asymptotisch naar 0.
  - De waarden van $V_0$ en $a$ hangen af van het betreffende atoom.
- De atomen kunnen ten opzichte van elkaar $vibreren$ : een beweging waarbij ze zich beurtelings 'van elkaar af' en 'naar elkaar toe' bewegen.
  De beweging 'van elkaar af' wordt (op de duur) gestopt doordat de aantrekkende kracht te groot wordt.
  De beweging 'naar elkaar toe' wordt (op een gegeven moment) gestopt doordat de afstotende kracht te groot wordt.
- Tijdens de beweging geldt de behoudswet van energie-impuls :

$$\boxed{E_{totaal} = \frac{p^2(r)}{2m} + V(r) = constant} \tag{6.3}$$

  In de 'omkeerpunten' $r_{in}$ en $r_{out}$ geldt :

$$\boxed{E_{totaal} = V(r_{in}) = V(r_{out})} \tag{6.4}$$

- Bekend is dat voor gebonden systemen op atomaire schaal **niet alle** waarden van $E_{totaal}$ zijn toegestaan: er is een **spectrum** van energie-**eigenwaarden**. De quantisatie-regels van Sommerfeld-Bohr komen er op neer dat alle gebonden toestanden moeten voldoen aan de volgende eis :

$$\boxed{\int_{r_{in}}^{r_{out}} p(r)dr = \sqrt{2m}\int_{r_{in}}^{r_{out}} [E_n - V(r)]^{\frac{1}{2}}\, dr = (n+\frac{1}{2})\hbar\pi} \tag{6.5}$$

  Hierin is $\hbar$ de constante van Planck (gedeeld door $2\pi$),
  en n = 0, 1, 2,.....

- Het berekenen van de energie-eigenwaarden komt dus neer op het oplossen van de vergelijking :

$$\boxed{\sqrt{2m}\int_{r_{in}}^{r_{out}} [E_n - V(r)]^{\frac{1}{2}}\, dr = (n+\frac{1}{2})\hbar\pi} \tag{6.6}$$

  waarbij $r_{in}$ en $r_{out}$ de oplossingen zijn van

$$\boxed{E_n = V(r) = 4V_0\left[\left(\frac{a}{r}\right)^{12} - \left(\frac{a}{r}\right)^{6}\right]} \tag{6.7}$$

- Om het rekenwerk te vereenvoudigen, gaan we over op andere eenheden:
  - gebruik $a$ als eenheid van lengte : $r \to ax$
  - gebruik $V_0$ als eenheid van energie : $E_n \to e_n V_0$
  - introduceer $\gamma = a\hbar^{-1}\sqrt{2mV_0}$

  Het berekenen van de energie-eigenwaarden komt nu neer op het oplossen van de vergelijking :

$$\boxed{\gamma\int_{x_{in}}^{x_{out}} [e_n - v(x)]^{\frac{1}{2}}\, dx = (n+\frac{1}{2})\pi} \tag{6.8}$$

  waarbij $x_{in}$ en $x_{out}$ de oplossingen zijn van

$$\boxed{e_n = v(x) = 4\left[x^{-12} - x^{-6}\right]} \tag{6.9}$$

  en waarbij (volgens de handboeken) voor $H_2$ geldt

$$\boxed{\gamma = 21.7} \tag{6.10}$$

- merk op dat in deze eenheden
  - $V(x) = 0$ voor $x = 1$
  - de minimumwaarde van de potentiaal $= -1$. deze wordt bereikt bij

$$\boxed{x_{min} = 2^{\frac{1}{6}}} \tag{6.11}$$

### 6.1.1 Opbouw van het programma: een advies

Deze opdracht resulteert in een tamelijk 'lang' programma. Het is belangrijk de zaak stap voor stap op te zetten, en na elke stap te testen of het bijbehorende programma-deel goed werkt. *Dit zal deel zijn van de opracht..* Zorg er natuurlijk ook voor dat je Je kan bijvoorbeeld je programma op de volgende manier structureren:

- maak een functie **vpot ( x )** die voor elke waarde van $x$ de waarde van de Lennard-Jones potentiaal $v(x)$ levert.
  Je kan aan de formule van de potentiaal een aantal dingen *zien*, die *nuttig* zijn om je functie te *testen* :
  - De potentiaal heeft een minimum waarde $v = -1$ in $x = x_{min} = 2^{\frac{1}{6}}$
  - Voor het snijpunt met de x-as geldt : $v(x = 1) = 0$
  - Voor 'kleine' waarden van $x$, is $x^{-12}$ de dominerende term
  - Voor 'grote' waarden van $x$, is $-x^{-6}$ de dominerende term
- Om je functie **vpot** te *testen*, kan je bijvoorbeeld als volgt te werk gaan :
  - Bereken de waarden van de potentiaal op het interval $0.96 < x < 5.96$ met tussenstapjes van $0.01$
  - Maak een grafische plot van het resultaat.
- Bij een *gekozen* waarde van de energie $e$ hoort een waarde voor de zgn. actie-integraal.
  $$s(e) = \gamma \int_{x_{in}}^{x_{out}} [e - v(x)]^{\frac{1}{2}} \, dx$$
  Om deze waarde te berekenen zijn nodig de beide oplossingen $x_{in}$ en $x_{out}$ van de vergelijking $e = v(x)$. Dus :
- Maak een functie **action ( e)** , die :
  - voor een *gegeven* waarde van $e$ de bijbehorende waarden van $x_{in}$ en $x_{out}$ berekent (dit kan langs analytische weg)
  - de bijbehorende waarde van de actie-integraal berekent via numerieke integratie (bijv. met de trapezium-regel)
- Om je functie **action** te *testen*, kan je bijvoorbeeld als volgt te werk gaan :
  - Roep de functie aan met achtereenvolgens
    $e = -0.9, -0.8, -0.7, ..... - 0.1$
    Print de waarden van $x_{in}$ , $x_{out}$ en van de actie-integraal als functie van $e$ uit. Kijk of dat er uit ziet als verwacht.
  - Als er zich energie-eigenwaarden bevinden binnen het bereik van de bovenstaande $e$'s, dan moeten we zien dat de waarden van de actie-integraal de waarden van $(n + \frac{1}{2})\pi$ omvatten. Verifieer dit.

- We hebben nu alle ingredienten die nodig zijn om de laagste energie-eigenwaarde $e_0$ te vinden. Daarvoor moet gelden:

$$s(e_0) - \frac{\pi}{2} = 0.$$

Dit *komt neer* op het numeriek oplossen van een (niet-lineaire) vergelijking, zoals besproken in het hoofdstuk 'Basistechnieken'. Kandidaat-methodes zijn : 'Bisectie' en 'Methode vanuit 1 startpunt'. Dus :

- Maak een functie
**eigen (estart, value)** die de waarde van de energie uitrekent waarvoor $s(e) - value = 0$.
Gebruik $estart$ als 'start-waarde' of als 'ondergrens' van de zoekmethode.
Geef de gevonden eigenwaarde van de energie terug als $returnwaarde$ van de functie.

- De energie-eigenwaarde $e_1$ moet voldoen aan :

$$s(e_1) - \frac{3\pi}{2} = 0.$$

Deze waarde kunnen we weer vinden met de functie **eigen**, aangeroepen met een andere waarde voor $value$. Daarbij is $estart = e_0$ een goede startwaarde .

- Voor 'hogere' energie eigenwaarden, kan dezelfde procedure worden voortgezet.

- **PS** een nauwkeurige waarde voor $\pi$ krijg je via de 'opdracht' $pi = acos\ (-1.)\ ;$

### 6.1.2 De opdracht

De opdracht is :

- *Bereken de eigenwaarden van de energie, voorzover ze* **kleiner dan** $-0.01$ *zijn.* Het testen van je programma, zoals boven voorgesteld, hoort bij deze opdracht. Zorg dat je programma ook de test output geeft.

- *lever in*

  - *uw programma-code (hspec.py)* dat bij uitvoering de volgende resultaten en controles produceert :

    * *Test van vpot* Een grafische weergave van de waarden van de potentiaal op het boven beschreven interval.
    * *Test van action* De waarden (print) van $x_{in}$, $x_{out}$ en van de actie-integraal voor de waarden van $e$ als boven aangegeven. Maak ook een uitspraak of de waarden van de actie-integraal de waarden $(n + \frac{1}{2})\pi$ omvatten.
    * *Resultaat* De eigenwaarden van de energie als in de opdracht gevraagd.

  - *grafische resultaten*

## 6.2 Nog tijd over? Vervolg onderdeel van de opdracht

- in de literatuur vinden we ,als experimenteel resultaat, een tabel met $twaalf$ energie-eigenwaarden $kleiner\ dan\ -0.01$ .
  Het zal duidelijk zijn dat de berekeningen met de Lennard-Jones potentiaal niet goed kloppen met het experimentele resultaat.

- een beter resultaat kan worden verkregen als we de Lennard-Jones potentiaal vervangen door de Morse potentiaal

$$\boxed{v(x) = \left[1 - \exp\left(-\beta(x - x_{min})\right)\right]^2 - 1} \tag{6.12}$$

  met

$$\boxed{x_{min} = 2^{\frac{1}{6}} \qquad \beta = 1.5} \tag{6.13}$$

- de **opgave** luidt : herhaal het zoeken naar eigenwaarden met de Morse potentiaal, en vergelijk het numerieke resultaat met het experiment.

# 7 Partiële Differentiaal Vergelijkingen (PDE): elliptisch

## 7.1 Inleiding

Wanneer de functies die we willen uitrekenen, afhankelijk zijn van $meer\ dan\ één$ variabele, dan worden de afgeleiden $partiële$ afgeleiden; we spreken dan van $partiële$ differentiaalvergelijkingen. Dit doet zich voor wanneer de functie het gedrag beschrijft van een systeem in een twee- of drie- dimensionale ruimte, of wanneer de functie afhangt van zowel plaats als tijd.
De algemene vorm van een 2e orde partiële differentiaal vergelijking van een functie van twee variabelen is :

$$\boxed{A\frac{\partial^2 u}{\partial x^2} + B\frac{\partial^2 u}{\partial x \partial y} + C\frac{\partial^2 u}{\partial y^2} + D\left(x, y, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}\right) = 0} \tag{7.1}$$

De vergelijking heet $elliptisch$, als $B^2 - 4AC < 0$
De vergelijking heet $parabolisch$, als $B^2 - 4AC = 0$
De vergelijking heet $hyperbolisch$, als $B^2 - 4AC > 0$

## 7.2 Probleemstelling

- We beschouwen een probleem uit de $Electrostatica$ : in een ruimte waarin geen electrische ladingsbronnen aanwezig zijn, moet de electrische potentiaal $V(x, y, z)$ voldoen aan de Laplace- vergelijking :

$$\boxed{\Delta V(x, y, z) = \frac{\partial^2 V(x, y, z)}{\partial x^2} + \frac{\partial^2 V(x, y, z)}{\partial y^2} + \frac{\partial^2 V(x, y, z)}{\partial z^2} = 0} \tag{7.2}$$

- Om wille van de eenvoud beperken we ons tot een $2\text{-}dimensionaal$ probleem; de vergelijking wordt dan :

$$\boxed{\Delta V(x, y) = \frac{\partial^2 V(x, y)}{\partial x^2} + \frac{\partial^2 V(x, y)}{\partial y^2} = 0} \tag{7.3}$$

- De 'ruimte' waarop $V(x, y)$ moet worden berekend is beperkt door 'randen' (in plaats van eindpunten). wij $begrenzen$ de 'ruimte' door een rechthoekig gebied :

$$x_{min} \leq x \leq x_{max} \qquad y_{min} \leq y \leq y_{max}$$

- De benodigde $randcondities$ kunnen twee vormen aannemen :
  - Dirichlet type : legt de waarde van de functie op een rand vast
  - Von Neumann type : legt waarde van de **afgeleide** van de functie (**loodrecht** op een rand) vast.

## 7.3 Numerieke methode : in 1 dimensie, leesstof

De numerieke methode die wordt gebruikt om bovenstaand type vergelijking op te lossen, kan het makkelijkst worden beschreven aan de hand van een $gewone$ differentiaalvergelijking van de 2e orde, van het 'boundary value' type. Aan het 1 dimensionale probleem is overigens geen opdracht verbonden.

- we beschouwen de vergelijking :

$$\boxed{\frac{d^2\phi}{dx^2} = S(x) \quad (a \leq x \leq b)} \tag{7.4}$$

  met randcondities: $\phi(a)$ en $\phi(b)$ gegeven.

- Verdeel interval $(a, b)$ in $N$ intervallen van gelijke lengte $h$ met **roosterpunten**

$$x_i = a + ih \quad (i = 0, 1, 2....., n)$$

- Representeer $\phi(x)$ in de vorm van een **tabel** :

$$\phi_i = \phi(x_i) \quad (i = 0, 1, 2, ....., n)$$

- Randcondities leggen $\phi_0$ en $\phi_n$ vast.

- Representeer $S(x)$ in de vorm van een **tabel** :

$$S_i = S(x_i) \quad (i = 0, 1, 2, ....., n)$$

- Benader de 2e afgeleide :

$$\frac{d^2\phi_i}{dx^2} = \frac{\phi_{i-1} - 2\phi_i + \phi_{i+1}}{h^2}$$

- De vergelijking wordt dan herleid tot een stelsel van $(n-1)$ vergelijkingen

$$\boxed{\phi_{i-1} - 2\phi_i + \phi_{i+1} = h^2 S_i \quad (i = 1, 2, ....., n-1)} \tag{7.5}$$

- Je kunt dit interpreteren als een matrix-vergelijking:

$$\begin{pmatrix} -2 & 1 & 0 & 0 & ... & 0 \\ 1 & -2 & 1 & 0 & ... & 0 \\ 0 & 1 & -2 & 1 & ... & 0 \\ . & . & . & . & ... & . \\ . & . & . & . & ... & . \\ . & . & . & 1 & -2 & 1 \\ . & . & . & 0 & 1 & -2 \end{pmatrix} \begin{pmatrix} \phi_1 \\ \phi_2 \\ ... \\ ... \\ ... \\ \phi_{n-2} \\ \phi_{n-1} \end{pmatrix} = \begin{pmatrix} h^2 S_1 - \phi_0 \\ h^2 S_2 \\ ... \\ ... \\ ... \\ h^2 S_{n-2} \\ h^2 S_{n-1} - \phi_n \end{pmatrix} \tag{7.6}$$

Oplossing van de vergelijking $A\vec{\phi} = \vec{b}$ is

$$\vec{\phi} = A^{-1}\vec{b}$$

Maar: de matrix heeft een bijzondere structuur : $\mathbf{sparse}$ , $\mathbf{tridiagonal}$, $\mathbf{blokken}$. Vandaar dat andere oplossingsrecepten (anders dan matrix-inversie en vermenigvuldiging) efficienter kunnen zijn ( in termen van geheugenruimte en/of rekentijd)

- We behandelen de methode van $\mathbf{Gauss\text{-}Seidel\ iteratie}$.
  Bij deze methode hanteren we een iteratieve procedure die uiteindelijk moet leiden tot een set waarden voor $\phi_i$ die voldoet aan vgl (7.5).
  De methode gaat als volgt :
  - Zet de waarden van $\phi_0$ en $\phi_n$ in de eindpunten vast op de randvoorwaarden. Deze waarden mogen verder $\mathbf{niet}$ veranderd worden.
  - Zet de waarde van $\phi_i$ in de overige punten op een willekeurige startwaarde, bijv: $\phi_i = 0 \quad (i = 1, 2, ...., n-1)$ . Duidelijk is dat deze set waarden voor $\phi_i$ niet aan de voorwaarden voldoet.
  - Loop vervolgens de punten $(i = 1, 2, ...., n-1)$ een voor een af, en vervang de 'oude waarde' van $\phi_i$ door een 'nieuwe waarde'

    $$\boxed{vervang\ \phi_i\ door\ \ \frac{1}{2}\left(\phi_{i+1} + \phi_{i-1} - h^2 S_i\right)} \tag{7.7}$$

    Vervang de 'oude waarde' $\mathbf{onmiddellijk}$ door de 'nieuwe!!
  - De nieuwe set waarden voor $\phi_i$ zal $\mathbf{beter}$ aan de voorwaarden voldoen, maar mogelijkerwijze nog $\mathbf{niet\ goed\ genoeg}$. Vandaar dat we de vervangingsprocedure een aantal malen herhalen. Als alles goed gaat zal het resultaat convergeren naar de juiste oplossing.

- Overgang naar 'bewijs' van convergentie.
  Beschouw de grootheid

$$\boxed{E = \int_a^b dx \left[\frac{1}{2}\left(\frac{d\phi}{dx}\right)^2 + S\phi\right]}$$

$$\boxed{= \frac{1}{2h}\sum_{i=1}^{n}\left(\phi_i - \phi_{i-1}\right)^2 + h\sum_{i=1}^{n-1} S_i\phi_i} \tag{7.8}$$

Voor de 1ste afgeleide van $E$ naar een bepaalde $\phi_j$ geldt :

$$\frac{\partial E}{\partial \phi_j} = \frac{1}{h}\left(2\phi_j - \phi_{j-1} - \phi_{j+1}\right) + hS_j$$

Dus : als $\phi_{j-1}$ , $\phi_j$ en $\phi_{j+1}$ aan de basisvergelijking voldoen, geldt :

$$\frac{\partial E}{\partial \phi_j} = 0$$

Voor de 2e afgeleide van $E$ naar een bepaalde $\phi_j$ geldt :

$$\frac{\partial^2 E}{\partial \phi_j^2} = \frac{2}{h} > 0$$

Dus : voor de 'correcte' set waarden van $\phi_i$ (die aan de basisvergelijking voldoet) geldt dat de grootheid $E$ zijn $\mathbf{minimale}$ waarde heeft.

- Schets 'bewijs' van convergentie.
  In onze procedure hebben we op elk moment een set waarden

  $$\phi_i \quad (i = 0, 1, ..., n)$$

  Daarbij behoort een zekere waarde van $E$.
  Door onze substitutie (7.7) verandert de waarde van $E$ met :

  $$-\frac{1}{h}\left[\frac{1}{2}(\phi_{i+1} + \phi_{i-1} - h^2 S_i) - \phi_i\right]^2 \leq 0$$

  Dus $E$ wordt kleiner en we zijn op de (goede) weg naar het minimum.

- $\mathbf{Relaxatie}$. We kunnen de substitutie (7.7) ook schrijven als :

  $$\boxed{vervang \quad \phi_i \quad door \quad \phi_i + \frac{1}{2}\left(\phi_{i+1} + \phi_{i-1} - 2\phi_i - h^2 S_i\right)} \tag{7.9}$$

  waarbij de 'nieuwe' waarde van $\phi_i$ uit de oude wordt verkregen door er een correctie-term bij op te tellen. Deze correctie-term $= 0$ als de oplossing is bereikt.

- Een iets algemener 'recept' is :

  $$\boxed{vervang \quad \phi_i \quad door \quad (1-\omega)\phi_i + \frac{\omega}{2}\left(\phi_{i+1} + \phi_{i-1} - h^2 S_i\right)} \tag{7.10}$$

  De verandering $\Delta E$ in $E$ ten gevolge van deze substitutie is :

  $$-\frac{\omega(2-\omega)}{2h}\left[\frac{1}{2}(\phi_{i+1} + \phi_{i-1} - h^2 S_i) - \phi_i\right]^2 \leq 0$$

  - Zolang $0 < \omega < 2$ is $\Delta E \leq 0$ , dus is de procedure $\mathbf{convergent}$.
  - De keuze $\omega = 1$ komt overeen met $\mathbf{geen\text{-}relaxatie}$.
    De ervaring heeft uitgewezen dat de convergentie vaak kan worden $\mathbf{versneld}$ door $\omega \neq 1$ te kiezen.
    Als $\omega < 1$ spreken we van $\mathbf{over\text{-}relaxatie}$.
    Als $\omega > 1$ spreken we van $\mathbf{onder\text{-}relaxatie}$.
  - Een $\mathbf{algemeen}$ recept voor de $\mathbf{beste}$ keuze van $\omega$ is niet te geven : probeer het uit !

## 7.4 Numerieke methode in 2 dimensies

De bovenbeschreven methode laat zich makkelijk generaliseren tot 2 dimensies. Lees de onderstaande stof eerst goed door en dan volgt de opdracht om een numerieke oplossing te vinden die je kunt controleren met een kant en klaar programma met de analystische oplossing.

- de vergelijking voor de potentiaal was :

$$\boxed{\Delta V(x,y) = \frac{\partial^2 V(x,y)}{\partial x^2} + \frac{\partial^2 V(x,y)}{\partial y^2} = 0} \tag{7.11}$$

- De 'ruimte' waarop $V(x,y)$ moet worden berekend werd beperkt door de 'randen'

$$\boxed{x_{min} \leq x \leq x_{max} \qquad y_{min} \leq y \leq y_{max}}$$

- We nemen aan dat er Dirichlet randcondities zijn, die de potentiaal op de randen vastleggen.

- Verdeel interval $(x_{min}, x_{max})$ in $N$ intervallen van gelijke lengte $h$ . Coordinaten van de intervalgrenzen zijn :

$$x_i = x_{min} + ih \quad (i = 0, 1, 2....., n)$$

- Verdeel interval $(y_{min}, y_{max})$ in $M$ intervallen van gelijke lengte $h$ . Coordinaten van de intervalgrenzen zijn :

$$y_j = y_{min} + jh \quad (j = 0, 1, 2....., m)$$

- Roosterpunten hebben coordinaten $(x_i, y_j)$

- Representeer $V(x,y)$ in de vorm van een **matrix** :

$$V_i^j = V(x_i, y_j) \quad (i = 0, 1, 2, ....., n)(j = 0, 1, 2, ....., m)$$

- Benader de 2e afgeleiden :

$$\frac{\partial^2 V_i^j}{\partial x^2} = \frac{V_{i-1}^j - 2V_i^j + V_{i+1}^j}{h^2}$$

$$\frac{\partial^2 V_i^j}{\partial y^2} = \frac{V_i^{j-1} - 2V_i^j + V_i^{j+1}}{h^2}$$

- De Laplace vergelijking resulteert dan in een stelsel van $(n-1)(m-1)$ lineaire vergelijkingen :

$$V_{i-1}^j - 2V_i^j + V_{i+1}^j + V_i^{j-1} - 2V_i^j + V_i^{j+1} = 0$$

Hieruit volgt :

$$\boxed{V_i^j = \frac{1}{4}\left(V_{i-1}^j + V_{i+1}^j + V_i^{j-1} + V_i^{j+1}\right)} \tag{7.12}$$

M.a.w. de potentiaal in $elk\ punt$ moet gelijk zijn aan het gemiddelde van de potentialen in zijn $naaste\ buur\text{-}punten$.

- Dit stelsel vergelijkingen kan (weer) worden opgelost met behulp de methode van **Gauss-Seidel iteratie**. De methode gaat nu als volgt :
  - Zet de waarden van $V_0^j$ $V_n^j$ $V_i^0$ $V_i^m$ vast op de randvoorwaarden. Deze waarden mogen verder **niet** veranderd worden.
  - Zet de waarde van $V_i^j$ in de overige punten op een willekeurige startwaarde, bijv: $V_i^j = 0$ .
  - Loop vervolgens de roosterpunten een voor een af,
    $(j = 1; i = 1, 2, ..., n-1)$
    $(j = 2; i = 1, 2, ..., n-1)$
    $(.....)$
    $(j = m-1; i = 1, 2, ..., n-1)$

    en vervang de 'oude waarde' van $V_i^j$ door een 'nieuwe waarde'

$$\boxed{vervang\ V_i^j\ door \quad \frac{1}{4}\left(V_{i-1}^j + V_{i+1}^j + V_i^{j-1} + V_i^{j+1}\right)} \tag{7.13}$$

    eventueel kan ook de relaxatie parameter worden gebruikt :

$$\boxed{vervang\ V_i^j\ door \quad (1-\omega)V_i^j + \frac{\omega}{4}\left(V_{i-1}^j + V_{i+1}^j + V_i^{j-1} + V_i^{j+1}\right)} \tag{7.14}$$

  - De nieuwe set waarden voor $V_i^j$ zal **beter** aan de voorwaarden voldoen, maar mogelijkerwijze nog **niet goed genoeg**. Vandaar dat we de vervangingsprocedure een aantal malen herhalen. Als alles goed gaat zal het resultaat convergeren naar de juiste oplossing.
  - we hebben hier weer te maken met een $iteratieve$ procedure, en hebben derhalve behoefte aan een $convergentie\text{-}monitor$ : een getal dat wordt gebruikt om de iteratie te beeindigen. een geschikte grootheid hiervoor is (naar analogie van het 1-dimensionale geval) :

$$\boxed{E = \sum_{i=1}^{n}\sum_{j=1}^{m}\left[\left(V_i^j - V_{i-1}^j\right)^2 + \left(V_i^j - V_i^{j-1}\right)^2\right]} \tag{7.15}$$

## 7.5 De opgave : electrostatische problemen

### 7.5.1 De Fysische achtergrond

Deze opgave betreft een eenvoudig probleem uit de Electrostatica. Het probleem is zowel analytisch als numeriek oplosbaar. Het ziet er als volgt uit:

- Beschouw een rechthoek in het xy-vlak,
  begrensd door de punten (0,0) (a,0) (0,b) en (a,b) .
- De randcondities voor de potentialen zijn:
  $V = 0$ voor $x = 0$
  $V = 0$ voor $x = a$
  $V = 0$ voor $y = b$
  $V = V_0$ voor $y = 0$, waarin $V_0 > 0$
- Bij deze randcondities ligt de waarde van $V(x, y)$ voor elk punt in de rechthoek vast. Deze wordt bepaald door de Laplace-vergelijking.
- Bij de praktische uitwerking kiezen we :

  $$a = b = 1. \qquad V_0 = 10.$$

  we verdelen het $x$- en het $y$- interval in $100$ subintervallen.

### 7.5.2 De analytische oplossing: formules

Volgens bijv. Morse en Feshbach: Methods of Theoretical Physics, kan dit probleem langs analytische weg worden opgelost. Het resultaat is:

$$\boxed{V(x,y) = \frac{2V_0}{\pi} \sum_{n=1}^{\infty} F_1(n) F_2(n,x,a) \frac{F_3(n,y,a,b)}{F_4(n,b,a)}} \tag{7.16}$$

waarin:

$$\boxed{F_1(n) = \frac{1}{n}[1 - (-1)^n]}$$

$$\boxed{F_2(n,x,a) = \sin[\frac{\pi n x}{a}]}$$

$$\boxed{F_3(n,y,a,b) = \sinh[(\frac{\pi n}{a})(b-y)]}$$

$$\boxed{F_4(n,b,a) = \sinh[\frac{\pi n b}{a}]} \tag{7.17}$$

met

$$\sinh(u) = \frac{\exp(u) - \exp(-u)}{2}$$

### 7.5.3 De analytische oplossing: kant en klaar

Open en executeer $vana.py$. Bekijk hoe en wat er wordt geplot in de code en neem dat over als je aan de numerieke oplossing gaat werken, zodat je het grafische resultaat kunt gebruiken om de numerieke oplossing (zie verder) te controleren. Het programma $vana.py$ plot de waarde van de potentiaal langs de x en y richting, met de andere coordinaat (y of x) in het midden van het bereik.

### 7.5.4 De analytische oplossing: de aanpak voor liefhebbers (leesstof, niet verplicht)

Ga na dat de analytische oplossing voldoet aan de randcondities voor
$x = 0$ , $x = a$ en $y = b$
Bereken de analytische oplossing in alle roosterpunten. Neem daarbij het volgende in acht:

- De 'som' in de analytische oplossing gaat over een oneindig aantal termen. Dit kan uiteraard op de computer niet. Vervang daarom $\sum_{n=1}^{\infty}$ door $\sum_{n=1}^{nmax}$ met $nmax = 101$
- In het algemeen zal de bijdrage van een term aan de totale som kleiner worden, naarmate $n$ groter wordt. Breek de sommatie af zodra de absolute waarde van de relatieve bijdrage kleiner wordt dan $10^{-20}$
- Aan de analytische oplossing is eenvoudig te zien dat voor sommige waarden van $n$ de bijdrage aan de potentiaal **altijd** gelijk aan 0 is. Reken deze termen **niet** uit!
- De waarden van $F_3$ en $F_4$ kunnen zeer groot worden. Behandel daarom $F_1, F_2, F_3, F_4$ met **dubbele precisie**.
- Om het resultaat te controleren heb je minstens twee hulpmiddelen:
  - Uit de randcondities volgt een 'voorspelling' over het verloop van $V(x, y)$ als functie van $y$, bij constante $x$
  - Uit de symmetrie van de randcondities volgt een 'voorspelling' over het verloop van $V(x, y)$ als functie van $x$, bij constante $y$

### 7.5.5 De opdracht: de numerieke oplossing

De numerieke oplossing kan worden verkregen door middel van Gauss-Seidel iteratie, eventueel met gebruik van relaxatie.

De opdracht luidt :

- bereken voor het hierboven gedefinieerde probleem, de numerieke oplossing (en controleer je werk met het gegeven programma met de analytische oplossing). Gebruik de convergentie monitor om de berekening te stoppen.

- geef de gevonden resultaten grafisch weer.

(**Bonus**) Het is interessant om de volgende situaties numeriek door te rekenen (en grafisch weer te geven). Bedenk zelf de (rand)voorwaarden en experimenteer eventueel maar een beetje. Interpreteer het resultaat (dus vergelijk het met de theorie).

- een plaatcondensator.
- een of twee puntladingen.
- een zelfbedachte configuratie.

### 7.5.6 in te leveren

**Lever in** :

- Het programma *electrostat.py* dat bij uitvoering de volgende resultaten produceert:
  - Grafische weergave van de numerieke oplossing van het eerste probleem.
  - Grafische weergave van de convergentie-monitor van het eerste probleem
  - Grafische weergave van een vergelijk tussen de analytische en numerieke oplossing. Hiervoor kan je de geleverde code gebruiken. In de huidige vorm vult dit programma twee lijsten met de waarden van de potentiaal op twee dwarsdoorsneden, ongeveer in het midden van de x en de y as.
  - *Eventueel* Grafische resultaten van de extra opdrachten
- Grafische resultaten als boven beschreven

### 7.5.7 Appendix: twee-dimensionale arrays in Python

Bij deze opgave moet je met *matrices* werken. Een manier om dat in Python te implementeren is via twee-dimensionale *lists* (of eigenlijk een list van lists...). Een twee-dimensionale 5 bij 5 matrix kan als volgt worden geïnitialiseerd:

```
matrix = [[0 for x in xrange(5)] for x in xrange(5)]
```

En de elementen worden bijvoorbeeld als volgt gebruikt:

```
matrix[0][0] = 1
matrix[4][5] = 0

print matrix[0][0]
print matrix[4][5]
```

Er bestaan Python packages die geavanceerde alternatieven kunnen leveren. Het voordeel is dat er functies zijn gedefinieerd die de manipulatie van de matrices makkelijker maken. Het pakket **numpy**, bevat de *array* klasse. Zie bijvoorbeeld `http://docs.scipy.org/doc/numpy/reference/`.

# 8 Monte-Carlo Methoden

Een kernbergrip bij Monte-Carlo methoden is het $toevalsgetal$ (in het Engels: $random\ number$). Een toevalsgetal is een getal waarvan de waarde die het krijgt onvoorspelbaar is, zoals het werpen van een dobbelsteen. We spreken dan ook van het 'werpen' van een toevalsgetal. We kunnen wel spreken van een kansverdeling van de mogelijke waarden. Bij herhaaldelijk werpen zal deze verdeling gereproduceerd worden. Een voorbeeld van een verdeling is de $vlakke\ verdeling$ op een bepaald interval. Elke getal in dit interval heeft dan een gelijke kans om geworpen te worden.
Echte $toevalligheid$ is moeilijk te realiseren in een deterministisch systeem als een computer. De kwaliteit van een generator van toevalsgetallen wordt gemeten met de hoeveelheid worpen die men kan doen voordat de reeks zich gaat herhalen.

Monte-Carlo methoden maken gebruik van toevalsgetallen om uiteenlopende problemen te lossen of te bestuderen. Dit wordt gedaan door de faseruimte dat het probleem beslaat te bemonsteren met gebruik van toevalsgetallen. De techniek wordt vooral gebruikt wanneer het probleem complex is; het heeft bijvoorbeeld te veel vrije parameters, ingewikkelde randvoorwaarden of er is geen analytische beschrijving voorhanden. Aan de basis van Monte-Carlo studies ligt vaak een simulatie van het probleem die het stochastisch gedrag beschrijft.

## 8.1 Hit or Miss

Een universeel toepasbare Monte-Carlo techniek is de zogenaamde $Hit\ or\ Miss$ methode. Deze kan bijvoorbeeld gebruikt worden voor het genereren van toevalsgetallen volgens eens arbitraire verdeling of voor het integreren van een willekeurige functie.
Gegeven een functie $f(x)$ met $a \leq x \leq b$ en $f(x) \geq 0$ in dat domein, is de methode als volgt samen te vatten:

1. Werp een toevalsgetal $x$ uit een vlakke verdeling tussen $a$ en $b$
2. Bereken $f = f(x)$
3. Werp nog een toevalsgetal $r$, vlak, maar dit keer tussen 0 en $m$, waar $m \geq max(f(x))$
4. Als $r > f$, neem een nieuw toevalsgetal $x$
   Als $r < f$, accepteer de waarde van $x$

Op deze manier zal de dichtheid van $x$ evenredig zijn aan $f(x)$

Deze methode heeft zijn beperkingen en kan inefficiënt zijn voor bijvoorbeeld hele steile functies en verdelingen. In dat geval is het beter een andere techniek te gebruiken, vooral als de functie inverteerbaar is.

## 8.2 Opdracht 1: Werpen van toevalsgetallen

- Schrijf een functie die toevalsgetallen genereert volgens een vlakke verdeling tussen -5 en 5. Doe dit een paar keer en bereken het gemiddelde en de standaard deviatie. Plot een histogram van de toevalsgetallen. Gebruik de functie **random** (zie beneden). **N.B.**: bereken het gemiddelde en de standaard deviatie zelf, dat wil zeggen zonder hulp van bibliotheek-functies.

- Schrijf een functie die toevalsgetallen werpt volgens een normaal-verdeling met een gemiddelde waarde van 0 en een standaard deviatie van 1. Gebruik de $Hit\ or\ Miss$ methode. Controleer de positie van de piek een de breedte van de verdeling. Plot een histogram van de toevalsgetallen. Gebruik de functie **random** (zie beneden).

Hint: de python module **random** bevat de functie **random()** die een toevalsgetal werpt uit een vlakke verdeling op $[0, 1)$. De begin-toestand van deze (pseudo-)randomnumber generator kan gezet worden via de functie **random.seed([x])**. Zonder argument wordt de systeem tijd gebruikt (en dus is het resultaat moeilijker reproduceerbaar). Om een reproduceerbaar resultaat te maken moet een waarde van x gekozen worden. Het volgende stukje code produceert en print een toevalsgetal uit een vlakke verdeling:

```
import random

"""
zet seed op systeemtijd
"""
random.seed()

toevalsgetal = random.random()

print toevalsgetal
```

Dit stuk code zal elke keer dat je het draait een ander getal geven (tenzij het *toevallig* hetzelfde getal geeft ..). Om er voor te zorgen dat je programma elke keer hetzelfde getal geeft - en beter te controleren is- zal je de regel

```
random.seed()
```

kunnen vervangen door

```
random.seed(X)
```

waar X een 'seed' naar jouw keuze is.

### 8.2.1 Lever in

- Programma *randomnumber.py* Dat bij uitvoering het volgende produceert:
  - Gemiddelde en standaard deviatie van de vlakke verdeling

  - Grafische weergave van de vlakke verdeling
  - Gemiddelde en standaard deviatie van de normaal-verdeling
  - Grafische weergave van de normaal-verdeling
- Grafische weergaven als hierboven genoemd

## 8.3 Opdracht 2: numerieke integratie

- Bereken $\pi$ door punten (x,y) te genereren in een vlak met $-1 < x < 1$ en $-1 < y < 1$. Tel het aantal keer dat een punt binnen een cirkel met straal 1 valt. Hoe verhoudt zich dat tot de ratio van de oppervlakken van de cirkel en het vierkant?

- Bereken de integraal van een normaalverdeling met gemiddelde $\mu = 0$ en breedte $\sigma = 1$ tussen de grenzen -1 en 1. Dus:

$$\int_{-1}^{1} Gauss(x|\mu, \sigma)dx \tag{8.1}$$

  Wat zou je verwachten?

- Plot de waarde van de integraal als functie van het aantal punten $N$ dat je gebruikt samen met de onzekerheid - foutenbalken zijn een goede manier. De standaard deviatie van een distributie is een goede maat voor de onzekerheid. Om de onzekerheid numeriek vast te stellen, kan de waarde van de integraal bij dezelfde waarde van $N$ meerdere malen berekend worden en de standaard deviatie van de distributie bepaald worden. Dit zal je dan voor elke waarde van $N$ moeten doen. Zorg dat de punten in de grafieken onafhankelijk zijn van elkaar: dus geen hergebruik van data. Je zal elke keer weer nieuwe getallen moeten werpen. Bevestig dat de onzekerheid kleiner wordt met $1/\sqrt(N)$.

### 8.3.1 Lever in :

- Programma *mcintegrate.py* Dat bij uitvoering het volgende produceert:
  - De waarde van $\pi$ berekend op de bovenstaande wijze
  - De integraal van de normaal verdeling zoals hierboven beschreven
  - Een grafische weergave van de waarde van de integraal en de onzekereid erop berekend voor verschillende waarden van $N$ (x-as: $N$, y-as: waarden van de integraal+foutenbalk)
  - Een grafische weergave van de onzekerheid op de integraal voor verschillende waarden van $N$ (x-as: $N$, y-as : onzekerheid), samen met een lijn die de verwachte $\frac{1}{\sqrt{N}}$ relatie geeft.
- De grafische weergaven als hierboven beschreven. Dit zijn dus 2 grafieken.

## 8.4 Opdracht 3: dronkenmans wandeling

1. Schrijf een functie dat een deeltje verplaatst van een plek op een vlak naar de ander, waarbij de nieuwe positie door het toeval bepaald wordt. Maak twee versies, met

   - Een vaste stapgrootte $r = 1$, of
   - Een toevallige stapgrootte tussen 0 en 1

   Maak een grafische weergave van het pad dat een deeltje aflegt voor vaste stapgrootte. Kies een redelijk aantal stappen. **Produceer dus een figuur.**

2. Laat $n = 100$ deeltjes bewegen gedurende $m = 10$ stappen (dus 100 simulaties waarin een deeltje 10 stappen doet). Bereken de gemiddelde afstand tot het beginpunt $(0, 0)$ na 10 stappen. Bereken ook de standaard deviatie op deze afstand. Vergroot het aantal stappen tot $m = 1000$. Doe dit voor vaste en variable stapgrootte. **Geen grafische output, produceer alleen getallen (8 stuks).**

3. Maak nu een simulatie voor deeltjes in een doos. Pas de functie uit 1 aan zodat de deeltjes binnen de doos blijven en aan de randen worden gereflecteerd. Kies een toevallige startpositie binnen de doos. Maak een grafische weergave van het pad van een enkel deeltje voor vaste stapgrootte en herhaal oefening 2 voor een doos van 20 bij 20. (dus, bereken de gemiddelde afstand en standaard deviatie van 100 deeltjes na 10 en 1000 stappen **Dus een figuur en 8 getallen.**).

4. Deel de doos uit 3 in twee helften (links en rechts). Begin met alle deeltjes in het linker deel. Bereken het aantal deeltjes in het rechter deel als functie van iteratie stap. Bereken de relaxatie tijd van het systeem (hier gedefinieerd als een evenwicht tussen het aantal deeltjes links en rechts.) Doe dit alleen voor vaste stapgrootte. Houdt rekening met de statistische fluctuaties door bijvoorbeeld een gemiddelde van verschillende simulaties te nemen. **Dus, produceer een plaatje met het aantal deeltjes in het rechter deel als functie van stap (*tijd*) en een getal (relaxatietijd.**

5. (BONUS) Herhaal de simulatie en bereken hoe lang het duurt voordat het eerste deeltje in het rechter deel van de doos komt.

6. (BONUS) Bedenk een strategie om interacties tussen de deeltjes in rekening te nemen.

### 8.4.1 Lever in:

- Programma *randomwalk.ph* dat bij uitvoering het volgende produceert:
  - Grafische weergave van het pad van een deeltje voor vaste stapgrootte als beschreven hierboven (1.).
  - Alle 8 getallen als beschreven in (2.) hierboven.
  - Grafische weergave van het pad van een deeltje in een doos met vaste stapgrootte en de 8 getallen als beschreven in (3.) hierboven.

  - Grafische weergave van het aantal deeltjes aan de rechterkant van de doos als functie van de iteratie stappen als beschreven in (4.)
  - De relaxatie tijd als beschreven in (4.)
  - *Eventueel* de resultaten van de BONUS vragen.

- De grafische weergaven als hierboven beschreven (3 stuks).

## 8.5 Opdracht 4: Het file-probleem

Het Nagel-Shreckenberg model is een simpele Monte-Carlo simulatie die een aantal belangrijke eigenschappen van wegverkeer kan tonen. In het model bewegen autos op een ronde baan. De baan is opgedeeld in $M$ zones en er zijn $N$ wagens. De tijd verloopt in discrete eenheden. Per tijdseenheid zal een auto met snelheid $v$ zich $v$ zones verplaatsen. Er geldt een maximumsnelheid $v_{max}$. De snelheid wordt gemeten in het aantal zones dat per tijdsstap wordt afgelegd.
In elke tijdsstap gaat elke auto door de volgende fases:

1. Als de snelheid $v$ beneden $v_{max}$ is, versnelt de bestuurder met 1 eenheid.
2. De bestuurder bepaalt de afstand $d$ tot de volgende auto. Als $v \geq d$ dan verlaagt de bestuurder de snelheid tot $d - 1$, teneinde niet te botsen.
3. Als $v > 0$ dan verlaagt de bestuurder met waarschijnlijkheid $p$ de snelheid met een eenheid. Deze stap introduceert het toevallige gedrag van de bestuurders.
4. De autos nemen hun nieuwe posities in.

Deze stappen gebeuren tegelijkertijd voor alle bestuurders.
Opdrachten:

1. Implementeer het Nagel-Schreckenberg model voor een ronde baan. Vind een passende visualisatie van het resultaat.
2. Varieer $N$,$M$ en $v_{max}$. Bereken de gemiddelde snelheid als functie van de dichtheid van autos voor verschillende waarden van $v_{max}$ (minstens 3)
3. **Bonus:** Bedenk een passende definitie van een file en bereken de posities van eventuele files voor elke tijdsstap. Bereken de gemiddelde snelheid van files. Hoe varieert deze als functie van de variabelen?

### 8.5.1 Lever in:

- Programma *traffic.py* dat bij uitvoering produceert :
  - Grafische weergave van je verkeers simulatie. Dit kan een plot zijn of een animatie. De tijdsevolutie van de verkeerssituatie (posities van de autos met de tijd) moet te zien zijn.

  - Grafische weergave van de gemiddelde snelheid als functie van de dichtheid van autos voor verschillende waarden van $v_{max}$ (minstens 3)
  - *Eventueel* Een grafische weergave van de gemiddelde snelheid *van files* als functie van gekozen variabelen.

- Grafische weergaven zoals boven genoemd (minstens 2 figuren).

# 9 En nu zelf een probleem oplossen: Rutherford scattering

Deze opdracht heeft een basiscomponent die uitgebreid kan worden voor bonuspunten. Bij deze opdracht dient extra aandacht geschonken te worden aan de visualisatie van de resultaten en de mogelijkheid tot invoeren van keuzes voor de begincondities. Echter, het programma moet in eerste instantie correct uitgevoerd worden met door jou gekozen *standaard* parameters.

## 9.1 Basisopdracht

We schieten een geladen deeltje op een electrische potentiaal, bijvoorbeeld die van een ander geladen deeltje met *dezelfde* lading. Kies zelf redelijke en leuke begincondities (plaats en snelheid) en bereken de twee-dimensionale baan van het testdeeltje en laat dit langs grafische weg duidelijk zien. **Vergelijk je resultaat met de analytische oplossing.**
Wees er bewust van dat er in deze standaard situatie van uitgegaan wordt dat de electrische potentiaal statisch is, afkomstig van een onbeweeglijk deeltje.
Bedenk eerst zelf hoe je dit probleem numeriek gaat oplossen en ga dan pas aan de slag met de code.
Literatuur: zoek het zelf maar op. Bijvoorbeeld 'Analytical mechanics van Fowles en Cassiday' blz 266.
Zorg ervoor dat je de lading en massa van je testdeeltje kunt instellen. Wat gebeurt er allemaal als je de lading van het deeltje omkeert, zodat het deeltje een *tegenovergestelde* lading ten opzichte van de potentiaal heeft? Tegen welke numerieke problemen loop je dan op?

## 9.2 Uitgebreiding van de opdracht

Er zijn verschillende mogelijkheden om deze opdracht uitdagender te maken en daarmee bonuspunten te verdienen.

- Laat alle (2?) deeltjes bewegen.
- Het probleem uitbreiden naar 3 dimensies en 3 (of meer!) deeltjes. Bedenk interessante begincondities voor het systeem.
- Een gravitationeel systeem kan beschouwd worden, bijvoorbeeld de aarde, maan en een satelliet.
- De paden van de deeltjes kunnen middels matplotlib geanimeerd worden.

## 9.3 Aanwijzing - Iets om op te letten

De keus van je numerieke oplossingsmethode en parameters zoals stapgroottes hebben invloed op de nauwkeurigheid (en dus correctheid) van je oplossing. In opdracht 4 heb

je bijvoorbeeld gezien hoe de nauwkeurigheid kan afhangen van een oplossings methode. Ook wil je in de gaten houden of bijvoorbeeld je tijdsstappen niet te groot zijn vergeleken bij de veranderings-tijdschaal van het systeem (let op, het deeltje kan behoorlijk versnellen!).
Hoe houdt je in de gaten of het resultaat van je programma plausibel is? Als er een analytische oplossing voor handen is, of een benadering voor (rand)gevallen, ligt het voor de hand deze te gebruiken. Als dit niet voorhanden is, kan je controleren of er geen fundamentele fysische wetten worden geschonden. Wanneer je een systeem bestudeert, kan je niet anders verwachten dan dat een correcte oplossing geen behoudswetten schendt (maar het is niet per se zo dat als je geen behoudswetten schendt, je oplossing correct is!). Je kan dus bij het ontwikkelen van je programma controleren of dat ook zo is. Een goede keus is het controleren van het behoud van energie. *Laat bij het maken van je programma zien dat je aandacht besteedt aan dit soort controles van (delen van) je programma.*

## 9.4 In te leveren

**Lever in :**

- Programma code (scatter.py) dat bij uitvoering produceert:
  - Een grafische weergave van het pad van een geladen deeltje in een electrische potentiaal voor minstens de volgende situaties
    - *Standaard* Door jou gekozen standaard waarde(n) voor massa(s), beginpositie(s) en snelhe(i)d(en) en gelijke lading (afstotend) in geval van 2 ladingen.
    - Zelfde situatie als boven, met tegengestelde ladingen
    - Door de gebruiker in te stellen massa(s), lading(en), positie(s) en snelhe(i)d(en).
  - Grafische weergave van een controle van de correctheid of plausibiliteit van de oplossing. In geval dat er een analytische oplossing voorhanden is, moet deze gevisualiseerd worden. Controle van schenden van behoudswetten is alleen verplicht als er geen analytische oplossing voorhanden is.
  - Kort (!) Commentaar op eventuele afwijkingen van de correcte oplossing, of commentaar op beperkingen van het programma.
  - *Eventueel* Grafische weergave(n) horende bij de uitgebreide opdracht *en daarbij behorende controles*.
- Grafische resultaten zoals door het programma gemaakt voor de eerste twee gevallen en de bijbehorende controles.
- *Eventueel* Grafische resultaten behordende bij de uitgebreide opdracht (inclusief controles).

# 10 Vrije keuze opdracht

**Let op, bij deze vrije-keuze opdracht is het de bedoeling dat je veel meer zelfstandig werkt, dus met beduidend minder hulp van de assistenten.**
Je hebt nu de mogelijkheid zelf met een voorstel voor een opdracht te komen. Doe dit in overleg met de docent. Bedenk een fysisch vraagstuk dat je wil bestuderen en kom met een voorstel voor de numerieke aanpak.
Als alternatief zal er later op de site een lijst van gesuggereerde opdrachten komen. Voor elk van de opdrachten zal op de website later in de cursus wat materiaal beschikbaar gemaakt worden.