G0Q57A: Modellering en simulatie
Samenvatting

Arne Van Den Kerchove - arne@vandenkerchove.com

22 januari 2019

Inhoudsopgave

1 Modellen en simulaties

Antwoorden op examenvragen opnemen

2 Numerieke lineaire algebra en toepassingen

Dit hoofdstuk als er tijd over is

2.1 QR-factorisatie

Definitie 1 *De volle QR-factorisatie van de matrix A wordt gegeven door*

$$A = QR$$

met q een $m \times m$ orthogonale matrix en R een $m \times n$ bovendriehoeksmatrix.

2.1.1 Gram-Schmidt orthogonalisatie

Algorithm 1 Gram-Schmidt-algoritme

```
procedure QRGramSchmidt
    for j = 1 to n do
        v_j = a_j
        for i = 1 to j − 1 do
            r_ij = q_i^T a_j
            v_j = v_j − r_ij q_i
        r_jj = ||v_j||_2
        q_j = v_j / r_jj
```

Complexiteit: $\mathcal{O}(2mn^2)$
Stabiliteit: niet stabiel

2.1.2 QR-factorisatie met Givens-rotaties

Definitie 2 *Een Givens-rotatie is een $m \times m$ orthogonale matrix van de vorm*

$$G_{ij} = \begin{bmatrix} c & -s \\ s & c \end{bmatrix}$$

met

$$c^2 + s^2 = 1$$

Om een Givens-rotatie op te stellen die plaats (j, k) 0 maakt in matrix A, kies dan een element in dezelfde kolom (bv. het element boven (j, k)) op plaats (i, k) en maak G_{ij} met

$$c = \frac{a_{ik}}{\sqrt{a_{ik}^2 + a_{jk}^2}} \text{ en } s = \frac{a_{jk}}{\sqrt{a_{ik}^2 + a_{jk}^2}}$$

Algorithm 2 Givens-rotatie-algoritme

procedure QRGivens
  $Q = \mathbb{1}$
  $R = A$
  for $j = 1$ to n **do**
    for $i = m$ to $j + 1$ **do**
      $c = \frac{r_{i-1,j}}{\sqrt{r_{i-1,j}^2 + r_{i,j}^2}}$
      $s = \frac{r_{i,j}}{\sqrt{r_{i-1,j}^2 + r_{i,j}^2}}$
      $r_{i,j} = 0$
      $r_{i-1,j} = \sqrt{r_{i-1,j}^2 + r_{i,j}^2}$
      for $k = j + 1$ to n **do**

$$\begin{bmatrix} r_{i-1,k} \\ r_{ik} \end{bmatrix} = \begin{bmatrix} c & s \\ -s & s \end{bmatrix} \begin{bmatrix} r_{i-1,k} \\ r_{i,k} \end{bmatrix}$$

      for $k = 1$ to m **do**

$$\begin{bmatrix} q_{k,i-1} & q_{ki} \end{bmatrix} = \begin{bmatrix} q_{k,i-1} & q_{ki} \end{bmatrix} \begin{bmatrix} c & s \\ -s & s \end{bmatrix}$$

Complexiteit: $\mathcal{O}(3mn^2 - n^3)$
Stabiliteit: stabiel

2.1.3 QR-factorisatie met kolompivotering

Indien A niet van volle rang is, is het voor de stabiliteit beter om kolompivotering toe te passen. In stap j van het QR-algoritme met Givens-rotaties verwisselen we kolom j met de kolom p waarvan de 2-norm het grootst is.

Complexiteit: $\mathcal{O}(3mn^2 - n^3)$
Stabiliteit: stabieler voor rang-deficiënte matrices

2.2 Singuliere-waardenontbinding

Definitie 3 *De singuliere-waardenontbinding van matrix A wordt gegeven door*

$$A = \hat{U}\hat{\Sigma}V^T$$

waarbij $\hat{U}$ orthonormale kolommen heeft, $\hat{\Sigma}$ een diagonaalmatrix met de singuliere waarden is en V een orthogonale matrix is.

Eigenschappen van de SVD:

- De rang van A is gelijk aan de rang van Σ is gelijk aan het aantal niet-nul singuliere waarden.

- De eerste r kolommen van U vormen een basis voor de kolomruimte van A.
- De laatste $n-r$ kolommen van V vormen een basis voor de nulruimte van A.
- $A = U\Sigma V^T = \sum_{i=1}^{r} \sigma_i u_i v_i^T$
- $normA_2 = \sqrt(\sigma_1^2 + \sigma_2^2 + ... + \sigma_r^2)$
- De singuliere waarden zijn de vierkantswortels van de eigenwaarden van $A^T A$. De kolommen van V zijn de bijhorende eigenvectoren.
- De singuliere waarden zijn de vierkantswortels van de n grootste eigenwaarden van AA^T. De eerste n kolommen van U zijn de bijhorende eigenvectoren.
- Als A symmetrisch is, zijn de singuliere waarden de absolute waarden van de eigenwaarden van A.

2.2.1 Lage rangbenadering

Definitie 4 *De ϵ-rang van een matrix A wordt gedefinieerd als*

$$rang(A, \epsilon) = \min_{\|A-B\|_2 \leq \epsilon} rang(B)$$

De matrix B ligt ϵ-dicht bij A als hij een rang heeft die de kleinste is onder alle matrices die ϵ-dicht bij A liggen.

Definitie 5 *Een rang k-benadering A_k met $(k \leq r)$ van A wordt berekend door de singuliere waardenontbinding te vermenigvuldigen, maar Σ te vervangen door een diagonaalmatrix met de k grootste singuliere waarden op de diagonaal.*

Hierdoor geldt de eigenschap

$$\|A - A_k\|_2 = min_{B \in \mathbb{R}^{m\times m} \mathrm{rang}(B) \leq k} \|A - B\|_2 = \sigma_k + 1$$

Lagerangbenadering met QR

2.3 Kleinste-Kwadratenbenadering

Om de coëfficiënten te bepalen wordt een Vandermondematrix A opgesteld. De te minimaliseren fout bij KK-benadering wordt gegeven door

$$\min_{x\in\mathbb{R}^n} \|b - Ax\|_2 = \min_{x\in\mathbb{R}^n} \sqrt{\sum_{i=1}^{m} (b_i - \sum_{j=1}^{n} a_{i,j} x_j)^2}$$

met $r = b - Ax$ de *residuvector.*

Dit probleem kan opgelost worden door x te bepalen in

$$A^T Ax = A^T b$$

De Vandermondematrix A is slecht geconditioneerd. We zoeken dus andere manieren om het KK-probleem op te lossen.

2.3.1 Oplossing met QR-ontbinding

Indien de QR-factorisatie van A bekend is, kan deze gebruikt worden om een oplossing voor het KK-probleem te vinden:

$$\min_{x\in\mathbb{R}^n} \|b - Ax\|_2 = \min_{x\in\mathbb{R}^n} \|b - QRx\|_2 = \min_{x\in\mathbb{R}^n} \left\|Q^T b - RAx\right\|_2$$

Aangezien vermenigvuldiging vooraan met een orthogonale matrix de norm behoudt.
De vector $Q^T b = c$ kan opgesplitst worden in de volgende componenten: $\begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$
met $c_1 \in \mathbb{R}^n$ en $c_2 \in \mathbb{R}^{m-n}$
Hieruit volgt:

$$\min_{x\in\mathbb{R}^n} \|b - Ax\|_2 = \min_{x\in\mathbb{R}^n} \left\| \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} - \begin{bmatrix} \hat{R} \\ 0 \end{bmatrix} x \right\|_2$$

Volgens de stelling van Pythagoras geldt:

$$\min_{x\in\mathbb{R}^n} \|b - Ax\|_2^2 = \min_{x\in\mathbb{R}^n} (\left\|c_1 - \hat{R}x\right\|_2^2 + \|c_2\|_2^2)$$

De vector x met coëfficiënten met minimale fout kan dus ook bekomen worden als x de oplossing van $\hat{R}x = c_1$

Complexiteit: $\mathcal{O}(mn) + \mathcal{O}n^2$ indien QR-factorisatie bekend.
Stabiliteit: stabiel indien A van volle rang.

2.4 Eigenwaardenproblemen

2.4.1 Methode van de machten

Deze methode vindt de eigenvector bij de grootste eigenwaarde en is geschikt voor ijle matrices.
q_k zal convergeren naar $\gamma_1 x_1$
Voordelen:

- Eenvoudige berekeningen
- zeer efficiënt voor ijle matrices

Nadelen:

- Zeer trage convergentie als λ_1 niet sterk dominant is.

Convergentie: lineair, afhankelijk van afstand tussen grootste eigenwaarden.

Algorithm 3 Methode der machten

procedure EigenPowermethod
  q_1 random
  for $k = 0, 1, 2, ...$ **do**
    take γ_k such that $\|q_{k+1}\|_2 = 1$
    $q_{k+1} = \frac{Aq_k}{\gamma_k}$

2.4.2 Deelruimte-iteratie

We itereren nu niet meer op één vector (methode der machten) maar op de volledige ruimte opgespannen door een orthonormaal stel vectoren.

Voor n eigenwaarden:

Algorithm 4 Deelruimte-iteratie

procedure EigenPartialSpace
  $\hat{Q}_0 = \begin{bmatrix} q_1^0 & q_2^0 & ... & q_n^0 \end{bmatrix}$ random orthonormaal
  for $k = 0, 1, 2, ...$ **do**
    $\hat{P}_k = A\hat{Q}_{k-1}$
    $\hat{Q}_k\hat{R}_k = \hat{P}_k$ door QR-factorisatie

Convergentie: lineair, afhankelijk van afstand tussen eigenwaarden.

2.4.3 QR-algoritme zonder shift

Algorithm 5 QR-algoritme zonder shift

procedure EigenQR
  for $k = 1, 2, 3...$ **do**
    $A_k = Q_k R_k$ door QR-factorisatie
    $A_{k+1} = R_k Q_k$

$\tilde{Q}_k = \begin{bmatrix} \tilde{q}_1^{(k)} & \tilde{q}_2^{(k)} & ... & \tilde{q}_i^{(k)} \end{bmatrix}$ convergeert naar de eigenvctoren van A.

Complexiteit: $\mathcal{O}(km^3)$

2.4.4 Omvorming tot Hessenbergmatrix

Het aantal stappen in het QR-algoritme kan teruggebracht worden door eerst de matrix om te vormen naar een *Hessenbergvorm* m.b.v. Givens-rotaties.

Complexiteit van omvorming: $\mathcal{O}(m^3)$

2.4.5 QR-algoritme met shift

De convergentie van het QR-algoritme kan versneld worden door een *shift* toe te passen.

Algorithm 6 QR-algoritme met shift

procedure EIGENQRSHIFT
  $A = A_0$ Hessenberg
  for $k = 1, 2, 3...$ **do**
    $\kappa = a_{m,m}^{(k)}$
    $A_k - \kappa\mathbb{1} = Q_k R_k$ door QR-factorisatie
    $A_{k+1} = R_k Q_k + \kappa\mathbb{1}$

$\tilde{Q}_k = \begin{bmatrix} \tilde{q}_1^{(k)} & \tilde{q}_2^{(k)} & ... & \tilde{q}_i^{(k)} \end{bmatrix}$ convergeert naar de eigenvectoren van A.

Complexiteit: $\mathcal{O}(km^3)$
Convergentie: kubisch indien symmetrisch, anders lineair, afhankelijk van afstand tussen eigenwaarden.

2.5 Toepassingen in de grafentheorie

2.5.1 PageRank

Stel een grafe op van alle links op webpagina's naar andere webpagina's. Het gewicht van de edges wordt genormaliseerd met het aantal links op de pagina die verwijst. Stel deze grafe voor als matrix A

A geeft ook een Markov-model voor het web.

Gebaseerd op de matrix A wordt aan elke pagina P_i een score $r(P_i) \geq 0$ toegekend via de volgende principes:

1. als veel andere pagina's naar P_i verwijzen, begunstigd dit $r(P_i)$
2. als $r(P_i)$ hoog is, begunstigd dit de score van pagina's waarnaar P_i verwijst.
3. als P_i weinig links heeft is dit begunstigend voor de pagina's waarnaar P_i verwijst.

Vervolgens wordt A omgevormd tot een *irreduceerbare* en *rij-stochastische* matrix $\hat{A}$. Volgens deze eigenschappen heeft $\hat{A}$ 1 als grootste eigenwaarde en kan de *PageRank*-vector met scores dus gevonden worden door het volgende stelsel op te lossen:

$$\hat{A}^T \Pi = 1\Pi$$

wat neer komt op het bepalen van de eigenvector van $\hat{A}$ horende bij de dominante eigenwaarde 1.

2.5.2 Meest centrale knoop

Centraliteit van een knoop kan gemeten worden door het aantal verbindingen of lussen. Het aantal lussen van lengte n vertrekkende uit knoop i is $(A^n)_{i,i}$. Voor de definitie van de meest centrale knoop moeten lussen van elke lengte worden meegeteld, maar hoe groter de lengte, hoe minder gewicht er gegeven moet worden aan die lus.

Definitie 6 *De centraliteit van een knoop i in matrix A wordt gegeven door $(e^A)_{i,i}$ met e^A de matrix-exponentiële van matrix A, die berekend wordt als*

$$\mathbb{1} + \frac{A}{1!} + \frac{A^2}{2!} + ... + \frac{A^n}{n!} + ...$$

Indien de eigenwaardenontbinding van A bekend is als

$$A = X diag(\lambda_1, ..., \lambda_N) X^{-1}$$

kan de matrix-exponentiële berekend worden als

$$e^A = X diag(e^{\lambda_1}, ..., e^{\lambda_N}) X^{-1}$$

3 Optimalisatie

Definitie 7 *De gradiënt van een functie $f : \mathbb{R}^n \to \mathbb{R}$ wordt gedefinieerd als*

$$\nabla f(x) := \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} & ... & \frac{\partial f(x)}{\partial x_n} \end{bmatrix}^T$$

Definitie 8 *De Hessiaan van een functie $f : \mathbb{R}^n \to \mathbb{R}$ wordt gedefinieerd als*

$$\nabla^2 f(x) := \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_i \partial x_j} \end{bmatrix}_{i,j=1}^n \in \mathbb{R}^{n \times n}$$

Definitie 9 *De Jacobiaan van een functie $K : \mathbb{R}^n \to \mathbb{R}^m$ wordt gedefinieerd als*

$$J_K f(x) := \begin{bmatrix} \frac{\partial k_i(x_1 ..., x_n)}{\partial x_j} \end{bmatrix}_{i,j=1}^n \in \mathbb{R}^{n \times n}$$

De Hessiaan van $f(x)$ is de Jacobiaan van haar gradiënt $(r)f(x)$

3.1 Optimalisatieproblemen zonder beperkingen

De helling in de richting van de gradiënt is het grootst.

3.1.1 Algemene afdalingsmethode

Een iteratie van afdalingsmethode om een minimum van een functie f te vinden begint met het vinden van een volgend iteratiepunt $x^(k)$. Dit gebeurt in twee stappen:

1. Bereken een *trial*-stap $h^{(k)}$ waarvoor $[h^{(k)}]^T f(x^{(k)}) < 0$
2. Zoek de stapgrootte $\alpha^{(k)} > 0$ zodat $x^{(k+1)} = x^{(k)} + \alpha^{(k)} h^{(k)}$ een lagere functiewaarde geeft dan $f(x^{(k)})$

algoritme

In veel methodes neemt α af naarmate het minimum nadert.

3.1.2 Methode van de steilste afdaling

De richting waarin f het snelst daalt wordt bepaald door

$$h^{(k)} = -\nabla f(x^{(k)})$$

Dit zorgt voor inefficiëntie door een zig-zagverloop omdat $\alpha^{(k)}$ vaak te groot is.

3.1.3 Methode van Newton

De afdaalrichting $h^{(k)}$ wordt bepaald door $f(x)$ in de omgeving van $x^{(k)}$ door een *kwadriek*:

$$f(x^{(k)} + h) \approx q(h) := f(x^{(k)}) + h^T \nabla f(x^{(k)}) + \frac{1}{2} h^T \nabla^2 f(x^{(k)}) h$$

Deze benaderende functie heeft in punt $x^{(k)}$ dezelfde gradiënt (raakvlak) en dezelfde Hessiaan (kromming) als $f(x)$.
De richting naar het nieuwe iteratiepunt is het minimum van deze kwadriek, berekend door $\nabla q(h) = 0$:

$$h^{(k)} = -[\nabla^2 f(x^{(k)})]^{-1} \nabla f(x^{(k)})$$

Convergentie: Kwadratisch indien Hessiaan niet-singulier in het attractiepunt en startwaarde in attractiegebied van minimum.

3.2 Optimalisatieproblemen met beperkingen

3.2.1 Gelijkheidsbeperkingen

Voor **één gelijkheidsbeperking** geldt de eigenschap dat als x^* een lokale minimizer is en $\nabla g(x^*) \neq 0$ er een $\lambda \in \mathbb{R}$ bestaat zo dat $-\nabla f(x^*) = \lambda \nabla g(x^*)$

Hiermee kunnen kandidaat minima gevonden worden door het volgende stelsel op te lossen met de methode van Newton-Raphson:

$$\begin{cases} \nabla f(x) + \lambda \nabla g() = 0 \\ g(x) = 0 \end{cases}$$

Voor **meerdere gelijkheidsbeperkingen** geldt de eigenschap dat als x^* een lokale minimizer is en de vectoren

$$\nabla g_1(x^*), ...\nabla g_p(x^*)$$

zijn lineiair onafhankelijk, dan bestaan er $\lambda_1, ..., \lambda_p \in \mathbb{R}$ zo dat

$$-\nabla f(x^*) = \sum_{i=1}^{p} \lambda_i \nabla g_i(x^*)$$

Hiermee kunnen kandidaat minima gevonden worden door het volgende stelsel op te lossen met de methode van Newton-Raphson:

$$\begin{cases} \nabla f(x) + \sum_{i=1}^{p} \lambda_i \nabla g_i(x) = 0 \\ g_1(x) = 0 \\ ... \\ g_p(x) = 0 \end{cases}$$

3.2.2 Ongelijkheidsbeperkingen

TODO

4 Trigoniometrische benadering

4.1 Fourier-analyse

Een Fourier-transformatie stelt een functie, die origineel in het tijdsdomein werd voorgesteld, voor in het frequentiedomein. Afhankelijk van de situatie onderscheiden we de volgende Fourier-transformaties:

1. **Continue Fourier-transformatie** wanneer $x(t)$ gedefinieerd voor $t \in (-\infty, \infty)$.

2. **Fourier-reeksontwikkeling** wanneer $x(t)$ gedefinieerd is op een compact interval $[a, b]$.

3. **Discrete Fourier-transformatie** wanneer $x(t)$ bemonsterd op een eindig aantal equidistante tijdstippen.

4. **Z-transformatie** wanneer $x(t)$ bemonsterd op een oneindig aantal equidistante tijdstippen.

4.2 Discrete Fourier-transformatie

4.2.1 DFT en IDFT

Definitie 10 *Voor een eindige rij* $\{x_n\}_{n=0}^{N-1}$ *met* N *willekeurige getallen wordt de discrete Fourier-transformatie (DFT) gegeven door*

$$X_k = \sum_{n=0}^{N-1} x_n W_N^{kn}$$

voor $k = 0, ..., N-1$ *met*

$$W_N = e^{2\pi i/N}$$

Definitie 11 *Voor een eindige rij* $\{X_k\}_{k=0}^{N-1}$ *met* N *willekeurige getallen wordt de inverse discrete Fourier-transformatie (IDFT) gegeven door*

$$x_n = \sum_{k=0}^{N-1} X_k W_N^{-kn}$$

voor $n = 0, ..., N-1$ *met*

$$W_N = e^{2\pi i/N}$$

Uit de eigenschap dat $W_N^{k+N} = W_N^k$ volgt dat er slechts N verschillende W_N^{kn} en W_N^{-kn} zijn.
Complexiteit van directe berekening: $\mathcal{O}(N^2)$

4.2.2 Daniel-Lanczos splitsingsalgoritme

Vanwege de eigenschappen $W_N^{N/2} + k = -W_N^k$, $X_{N/2+k}^{even} = X_k^{even}$ en $X_{N/2+k}^{odd} = X_k^{odd}$ geldt dat

$$X_k = X_k^{even} + W_N^k X_k^{odd}$$

Algorithm 7 Daniel-Lanczos Splitsingsalgoritme

```
1: procedure DFTSplit
2:     {X_k^even}_{k=0}^{N/2-1} = {∑_{n=0}^{N/2-1} x_{2n} W_{N/2}^k}_{k=0}^{N/2-1}
3:     {X_k^even}_{k=0}^{N/2-1} = {∑_{n=0}^{N/2-1} x_{2n} W_{N/2}^k}_{k=0}^{N/2-1}
4:     {X_k}_{k=0}^N = {X_{k/2}^even + W_N^k + X_{k/2}^odd}_{k=0}^N
```

Complexiteit: $\mathcal{O}(N^2/2)$

4.2.3 Snelle Fourier-transformatie

Met de combinatiestap van het Daniel-Lanczos-splitsingsalgoritme kunnen we een recursief algoritme opstellen

Algorithm 8 Snelle Fourier-transformatie

```
1: procedure FFT
2:     if N = 1 then
3:         X_k = x_k
4:     else
5:         reorder {x_k}
6:         {X_k} = {FFT(x_{k/2}^even) + W_N^k + FFT(x_{k/2}^odd)}
```

Complexiteit: $\mathcal{O}(N log_2(N))$

4.3 Symmetrische discrete Fourier-transformaties

4.3.1 Discrete cosinustransformatie

wat is een oneven uitbreiding

De DFT van een rij met even symmetrie wordt gegeven door

$$X_k = 2\sum_{n=0}^{N}{}'' x_n cos(\pi kn/N)$$

Dit is een transformatie van de symmetrische rij en is verwant aan de discrete cosinustransformatie.

Definitie 12 *De discrete cosinustransformatie (DCT) wordt gedefinieerd als*

$$X_k = \sum_{n=0}^{N}{}'' x_n cos(\pi kn/N)$$

voor $k = 0, ..., N$

Definitie 13 *De inverse discrete cosinustransformatie (IDCT) wordt gedefinieerd als*

$$x_n = \frac{2}{N}\sum_{k=0}^{N}{}'' X_k cos(\pi kn/N)$$

voor $n = 0, ..., N$

4.3.2 Snelle cosinustransformatie

Om de cosinustransformatie efficiënt te berkenen, kan een algoritme gelijkaardig aan FFT worden.

FCT

Algorithm 9 Snelle cosinustransformatie

```
procedure FCT

```

4.3.3 Discrete sinustransformatie

De DFT van een rij met oneven symmetrie wordt gegeven door

$$X_k = 2i\sum_{n=1}^{N-1} x_n sin(\pi kn/N)$$

Dit is een transformatie van de symmetrische rij en is verwant aan de discrete sinustransformatie.

Definitie 14 *De discrete sinustransformatie (DST) wordt gedefinieerd als*

$$X_k = \sum_{n=1}^{N-1} x_n sin(\pi kn/N)$$

voor $k = 1, ..., N-1$

Definitie 15 *De inverse discrete sinustransformatie (IDST) wordt gedefinieerd als*

$$x_n = \frac{2}{N} \sum_{k=1}^{N-1} X_k sin(\pi kn/N)$$

voor $n = 1, ..., N-1$

4.4 Meerdimensionele Fourier-transformatie

Deze sectie als er tijd over is

5 Toevalsgeneratoren

5.1 Genereren van willekeurige getallen

Een software-generator moet voldoen aan de volgende eigenschappen:

- snel en weinig geheugengebruik
- draagbaar over verschillende platformen
- voldoende lange periode
- reproduceerbare gegenereerde rij (bv. via seed)
- gegenereerde getallen voldoen aan eigenschappen van willekeurige getallen:

5.1.1 Lineaire congruentiële generator

Definitie 16 *Een lineaire congruentiële generator (LCG) met parameters (a, c, m) genereert getallen volgens de recursiebetrekking*

$$X_i + 1 = (aX_i + c) \ mod \ m$$

waarin a de multiplier, c het increment en m de modulus.

Enkele keuzes voor (a, c, m) die een grote periode opleveren zijn de volgende (met b het aantal bits in een geheugenplaats en $k \in \mathbb{Z}$):

- Voor $m = 2^b$, $c \neq 0$ is de maximale periode $P = m = 2^b$, die kan bereikt worden wanneer c relatief priem met m en $a = 1 + 4k$.
- Voor $m = 2^b$, $c = 0$ is de maximale periode $P = m/4 = 2^b - 2$, die kan bereikt worden wanneer X_0 oneven is en als $a = 3 + 8k$ of $a = 5 + 8k$.

- Als m een priemgetal en $c = 0$ is de maximale periode $P = m - 1$, die kan bereikt worden als a de eigenschap heeft dat de kleinste k waarvoor geldt dat $a^k - 1$ deelbaar is door m gelijk is aan $m - 1$

5.2 Testen voor toevalsgeneratoren

5.2.1 Chi-kwadraattest

De chi-kwadraattest is een frequentietest om na te gaan of de gegenereerde getallen voldoen aan de uniforme verdeling.

De chi-kwadraattest maakt gebruik van de toetsingsgrootheid

$$\chi^2_{n-1} = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

met O_i het aantal observaties in klasse i, E_i het verwachte aantal punten in klasse i en n het aantal klassen.

5.2.2 Kolmogorov-Smirnov-test

De Kolmogorov-Smirnov-test is een frequentietest om na te gaan of de gegenereerde getallen voldoen aan de uniforme verdeling.

Algorithm 10 Kolmogorov-Smirnovtest

```
procedure DFTSplit(KSTest)
    R_(1) ≤ ... ≤ R_(N) = sort(R_1, ..., R_N)
    D+ = max_{1≤i≤N} { i/N − R_(i) }
    D− = max_{1≤i≤N} { R_(i) − (i−1)/N }
    D = max(D+, D−)
    if D > D_α then
        reject H_0
    else
        accept H_0
```

5.2.3 Autocorrelatietest

Een autocorrelatietest vindt het onderling verband in een rij getallen. Gekke wiskunde

5.3 Toevalsgeneratoren voor discrete verdelingen

Definitie 17 *Stel dat X een discrete willekeurige variabele is met cdf F. De inverse distributiefunctie (idf) van X is dan*

$$F^* : [0,1] \to \chi : u \mapsto F^*(u) = \min_x \{x : u < F(x)\}$$

Als U een uniform verdeelde willekeurige variabele is en $F^*(x)$ de idf van $F(x)$, dan is een willekeurige variabele $Z = F^*(U)$ verdeeld volgens F

5.4 Toevalsgeneratoren voor continue verdelingen

Definitie 18 *Stel dat X een continue willekeurige variabele is met cdf F. De inverse distributiefunctie (idf) van X is dan*

$$F^{-1} : [0,1] \to \chi : u \mapsto F^{-1}(u) = \min_x \{x : u < F(x)\}$$

Het is niet altijd eenvoudig om een uitdrukking voor F^{-1} op te stellen. Daarom zijn er enkele methodes.

5.4.1 Transformatiemethode

Als U een uniform verdeelde willekeurige variabele is en $F^{-1}(x)$ de idf van $F(x)$, dan is een willekeurige variabele $Z = F^{-1}(U)$ verdeeld volgens F

5.4.2 Box-Muller-algoritme

Het Box-Muller-algoritme wordt gebruikt om getallen volgens $N(0,1)$ te genereren.

Algorithm 11 Box-Muller-algoritme

```
procedure BoxMullerNormal
    U_1, U_2 ~ U(0,1) random
    Z_1 = sqrt(-2 log(U_1)) cos(2πU_2)
    Z_2 = sqrt(-2 log(U_1)) sin(2πU_2)
```

Z_1 en Z_2 zijn onafhankelijke willekeurige variabelen verdeeld volgens $N(0,1)$

5.4.3 Som van Twaalf

Het Box-Muller-algoritme wordt gebruikt om getallen volgens $N(0,1)$ te genereren.

Algorithm 12 Som van twaalf

```
procedure SumOf12Normal
    U_i ~ U(0,1) for i = 1, 2, ..., 12
    Z = Σ_{12}^{i=1} U_i − 6
```

Z is bij benadering verdeeld volgens $N(0,1)$

5.4.4 Acceptance-rejection methode

De accpetance-rejection methode kan gebruikt worden om getallen volgens een willekeurige verdelingsfunctie $p(x)$ te genereren, gegeven een andere verdelingsfunctie $q(x)$ waar een generator voor bestaat.

In de praktijk wordt voor $q(x)$ vaak $U(0,1)$ genomen.

Algorithm 13 Acceptance-rejection methode

```
procedure AcceptRejectRandom
    Y ~ q(x) random
    x̂ ∈ dom(p(x))
    if 0 < Y < p(x̂)/q(x̂) then
        accept
    else if p(x̂)/q(x̂) < Y < 1 then
        reject
```