Technische Universiteit Delft
Faculteit Elektrotechniek, Wiskunde en Informatica

Bachelor Eind Project

BSc Verslag TECHNISCHE WISKUNDE

Google's PageRank Algoritmes

Auteur:
Alice Gianolio

Onder begeleiding van:
Prof. Dr. Ir. C. Vuik

Delft, 3 juli 2012

Voorwoord

Voor u ligt het resultaat van de scriptie van Alice Gianolio. Deze is uitgevoerd ter afsluiting van de bachelor Technische Wiskunde in de richting Numerieke Wiskunde aan de Technische Universiteit Delft. Dit rapport is bedoeld voor alle lezers die zich geroepen voelen om zich te verdiepen in de wereld van de dagelijks gebruikte zoekmachine Google. Na een half jaar hard aan deze scriptie te hebben gewerkt, is er dit meesterstuk uitgekomen.

In het voorwoord behoort een woord van dank; deze analyse van Google heb ik natuurlijk niet alleen kunnen volbrengen. Bij deze wil ik graag meneer Prof. Dr. Ir. C. Vuik bedanken voor zijn tijd en moeite die hij mij heeft gedoneerd. Hij heeft een bijdrage geleverd aan de totstandkoming van dit verslag.

Verder zou ik meneer Spandaw, mevrouw Keijzer en meneer Dekker willen bedanken die samen met meneer Vuik de examencommissie vormen die deze scriptie zal beoordelen.

Deze scriptie wil ik opdragen aan mijn ouders, Susanna en Alberto Gianolio, die mij altijd hebben aangemoedigd om mijn best te doen en door te blijven zetten wanneer het minder ging. Ik ben hun zeer dankbaar voor wat ze mij hebben geleerd en gegeven.

Ik hoop zeer dat u net zo veel plezier beleeft aan het lezen van dit verslag als ik heb gehad met het maken ervan.

Alice Gianolio
Delft, 3 juli 2012.

Inhoudsopgave

1 Inleiding

Stel dat ergens op de wereld een bibliotheek is bestaande uit 10 miljard boeken. In deze bibliotheek zijn geen medewerkers en de boeken zijn niet gesorteerd. Hier komt nog bij dat iedereen op elk moment ongemerkt een boek kan toevoegen of verwijderen uit de collectie. U bent nu bezig met een onderzoek en weet dat een of meer boeken uit de collectie waardevolle informatie bevatten voor uw analyse. Omdat u zoals de meeste mensen ongeduldig bent, zou u deze boeken zo snel mogelijk willen vinden. U voelt al aan dat dit een (bijna) onmogelijke zaak is.

Deze situatie is te vergelijken met het World Wide Web. Dit is een grote, ongeordende collectie van documenten in veel verschillende soorten en formaten. Twee promovendi aan de Stanford University, Larry Page en Sergey Brin, zagen als eerste in dat het web een interessante structuur vertoonde. Het duurde ook niet lang totdat ze in 1998 de zoekmachine Google oprichtten. Hiermee kon men vanaf toen gemakkelijk webpagina's vinden en sorteren.

De oprichters wilden deze zoekmachine een naam geven die de taak weerspiegelde om de miljarden pagina's te ordenen. Het woord Googol was in 1920 bedacht door de negenjarige Milton Sirotta. Zijn vader vroeg hem om een term te bedenken die het getal 10^{100} weergaf. De heren Page en Brin vonden dit een bijpassend woord voor hun uitvinding. Door een spellingsfout is deze zoekmachine bekend geworden als Google.

Tegenwoordig is Google uitgegroeid tot de meest gebruikte zoekmachine ter wereld. Het is een middel geworden die ondenkbaar is uit deze samenleving. Voornamelijk komt dit door het zoekalgoritme die Page en Brin hebben ontworpen, de PageRank methode. Dit is het onderwerp waar deze scriptie zich op zal concentreren. Het project dat uitgevoerd is, dient als vervolgonderzoek waar Erik Berkhof in 2009 aan begonnen was.

De PageRank methode kan worden gezien als een eigenwaarde probleem. Om dit te begrijpen is kennis uit de Lineaire Algebra nodig. In hoofdstuk 2 is er, om deze kennis op te frissen, aandacht gegeven aan al bekende definities, Markov Keten theorie en gelijksoortige matrices.
In hoofdstuk 3 zal het PageRank concept uitgebreid aan bod komen. Er zal onder andere worden uitgelegd hoe dit een eigenwaarde probleem voorstelt.
Er bestaan verschillende algoritmes om dit probleem op te lossen. Twee van deze algoritmes zijn de Power Methode en de Arnoldi Methode. Hierop zal in hoofdstuk 4 ingezoomd worden waarna er in hoofdstuk 5 numerieke experimenten in Matlab uitgevoerd zullen worden om na te gaan welke van de twee algoritmes het meest efficiënt is. Hierbij zal er naar de tijdsduur worden gekeken en het aantal benodigde iteraties.
In de afgelopen tijd hebben de heren La Cruz, Martìnez en Raydan een nieuw methode ontwikkeld om het eigenwaarde probleem op te lossen, de Rayleigh Quotiënt Methode. Dit algoritme maakt gebruikt van de DF-SANE nulpuntsmethode. Deze theorie zal in hoofdstuk 6 besproken worden. Er zullen in Matlab experimenten worden uitgevoerd waarbij de Rayleigh Quotiënt Methode vergeleken wordt met de eerder besproken Power Methode.
Vervolgens zal men concluderen welke methoden in welke situatie het meest geschikt is om te gebruiken. Tot slot zal in de discussie besproken worden hoe de resultaten verbeterd zouden kunnen worden en zullen er suggesties worden gegeven voor een vervolg onderzoek.
In de bibliografie is een lijst weergegeven van de geraadpleegde literatuur.

2 Voorkennis

In deze paragraaf zal informatie worden uitgelegd die bekend acht te zijn om de PageRank methode te kunnen begrijpen. Eerst zullen definities worden gegeven waarna een gedeelte van de theorie van Markov Ketens voorkomt. Ten slotte zal het begrip van gelijksoortige matrices worden herhaald.

2.1 Definities

Definitie 1 (**Sparse Matrix**). *Een matrix waarvan de elementen voornamelijk gelijk zijn aan nul.*
Definitie 2 (**Stochastiche Matrix**). *Een $n \times n$ matrix met de eigenschap dat zowel alle elementen niet negatief en reeël zijn als dat de som van elke rij gelijk is aan één.*
Definitie 3 (**Substochastiche Matrix**). *Een $n \times n$ matrix met de eigenschap dat zowel alle elementen niet negatief en reeël zijn als dat de som van elke rij minstens één is.*
Definitie 4 (**Eigenwaarden / Eigenvector**). *Zij A een $n \times n$ matrix. Een scalar λ is een eigenwaarde van A als er een niet nul kolom vector $\overline{v}$ in $\mathbb{R}^n$ bestaat zodanig dat $A\overline{v} = \lambda\overline{v}$. De vector $\overline{v}$ heet een eigenvector van A bijbehorend bij eigenwaarde λ.*
Definitie 5 (**Convexe verzameling**). *Een verzameling $A \subseteq B$ heet convex wanneer voor $x, y \in A$ geldt dat $(1-t)\,x + ty \in A$ voor alle $0 \leq t \leq 1$.*
Definitie 6 (**Hessenberg Matrix**). *Een $n \times n$ matrix waarvoor geldt*
- $h_{i,j} = 0$ voor alle $i \geq j + 1$. ((Boven) Hessenberg Matrix)
- $h_{i,j} = 0$ voor alle $j \geq i + 1$ (Beneden Hessenberg Matrix)
Definitie 7 (**Link Matrix**). *Een $n \times n$ matrix waarvoor geldt $B_{ij} = 1$ als er een verwijzing bestaat van B_i naar B_j, anders $B_{ij} = 0$.*
Definitie 8 (**Symmetrische Matrix**). *Een $n \times n$ matrix waarvoor geldt $A^T = A$.*
Definitie 9 (**Positief-Definiet Matrix**). *Zij A een $n \times n$ matrix. A heet positief-definiet als $\overline{x}^T A\overline{x} > 0$ voor alle $\overline{x} \in \mathbb{C}^n$, $\overline{x} \neq \overline{0}$ in het complexe geval of $\overline{x}^T A\overline{x} > 0$ voor alle $\overline{x} \in \mathbb{R}^n$, $\overline{x} \neq \overline{0}$ in het reeële geval.*
Definitie 10 (**Positief Semi Definiet Matrix**). *Zij A een $n \times n$ matrix. A heet positief semi definiet als $\overline{x}^T A\overline{x} \geq 0$ $\overline{x} \in \mathbb{C}^n$, $\overline{x} \neq \overline{0}$ in het complexe geval of $\overline{x}^T A\overline{x} > 0$ voor alle $\overline{x} \in \mathbb{R}^n$, $\overline{x} \neq \overline{0}$ in het reeële geval.*
Definitie 11 (**Symmetrische Positief-Definiet Matrix**). *Een matrix die zowel symmetrisch als positief-definiet is, wordt een symmetrische positief-definiet matrix genoemd (afgekort SPD).*
Definitie 12 (**Hermitische Matrix**). *Zij A een $n \times n$ matrix. A heet Hermitisch wanneer zijn getransponeerde matrix gelijk is aan zijn geconjungeerde matrix, $A^T = \overline{A}$.*

2.2 Markov Keten Theorie

Het gehele web kan worden voorgesteld als een gerichte graaf. Het is een geordend paar $G = (V, E)$ waarbij V de verzameling knopen is en E de verzameling van van geordende paren knopen, ook wel gerichte takken genoemd. In deze webvoorstelling stelt elke knoop i de pagina P_i voor en elke gerichte tak (i, j) een verwijzing van pagina P_i naar pagina P_j.

Een gerichte graaf als hierboven beschreven kan worden voorgesteld als een Markov matrix A. Beschouw een webstructuur met n aantal pagina's, oftewel met n aantal knopen. A is dan een $n \times n$ matrix waarbij $A_{ij} = \frac{1}{|P_i|}$ als er een verwijzing bestaat van P_i naar P_j, anders $A_{ij} = 0$. $|P_i|$ stelt het aantal verwijzingen voor van P_i naar andere pagina's. Zie volgend hoofdstuk voor meer diepgang hierover.

Nu zullen er verschillende matrixeigenschappen worden besproken, die men later nodig zal hebben om de PageRank methode te begrijpen.

- **Stochastische Matrix**
  Wanneer P_i verwijzingen heeft naar andere pagina's P_j bevat de rij i tenminste één element ongelijk aan 0. Wanneer dit voor alle pagina's i geldt, dan is A een stochastische matrix; de som van iedere rij i is gelijk aan 1.
- **Substochastische Matrix**
  Wanneer P_i geen enkele verwijzingen heeft naar een andere pagina, zal rij i uit slechts nullen bestaan. Pagina P_i wordt dan een Dangling node genoemd. Hier zal later worden ingegaan. Wanneer A ten minste een Dangling node bevat, is er sprake van een substochastische matrix.
- **Irreducibele Matrix**
  Een matrix is irreducibel wanneer er voor een reeks elementen van de matrix geldt dat $a_{ik_1}a_{k_1k_2}\cdots a_{k_{t-1}k_t}a_{k_tj} \neq 0$ voor elk paar knopen (i, j). In woorden gezegd; voor elk paar knopen (i, j) bestaat er een pad die ze verbindt.
- **Aperiodieke Matrix**
  Een matrix is aperiodiek wanneer er geen $k \in \mathbb{Z}$ bestaat waarvoor geldt dat $A^k = A$.
- **Primitieve Matrix** Een matrix is primitief wanneer hij zowel irreducibel als aperiodiek is.

Om na te gaan of een matrix primitief is, maakt men gebruik van de onderstaande stelling.

Stelling 1. *Een niet-negatieve $n \times n$ matrix wordt primitief genoemd dan en slechts dan als er een $k > 0$ bestaat zodat voor alle paren (i, j) met $0 < i, j < n$ er geldt dat $A^k_{ij} > 0$.*

De stelling zegt dat er een positieve k bestaat zodat voor alle combinaties (i, j) de elementen van A^k positief zijn.

2.3 Algemene Begrippen

2.3.1 Gelijksoortige Matrices

Gelijksoortige matrices is een begrip dat naar voren zal komen in de theorie van de Arnoldi methode. Men zegt dat een matrix $B \in \mathbb{R}^{n\times n}$ gelijksoortig is met een matrix $A \in \mathbb{R}^{n\times n}$ wanneer er een inverteerbare matrix $T \in \mathbb{R}^{n\times n}$ bestaat waarvoor geldt;

$$B = T^{-1}AT$$

Er geldt dan

$$B\overline{y} = \lambda\overline{y}$$
$$\leftrightarrow$$
$$T^{-1}AT = \lambda\overline{y}$$
$$\leftrightarrow$$
$$A\left(T\overline{y}\right) = \lambda\left(T\overline{y}\right)$$
$$\leftrightarrow$$
$$A\overline{x} = \lambda\overline{x}$$

waarbij λ een eigenwaarde is van B en $\overline{y}$ de bijbehorende eigenvector is.
Hieruit volgt dus dat A en B dezelfde eigenwaarden hebben. De bijbehorende eigenvector van A wordt dan gegeven door $\overline{x} = T\overline{y}$.

3 Het PageRank Model

Google is in de laatste jaren uitgegroeid tot de meest gebruikte zoekmachine ter wereld. Dit is niet alleen te danken aan de gebruiksvriendelijkheid, maar ook aan de kwaliteit van de zoekresultaten vergeleken met de andere zoekmachines. Deze kwaliteit wordt bepaald aan de hand van de PageRank methode, genoemd naar de Google oprichter Lawrence Page.

De kwaliteit van een zoekmachine wordt onder andere vastgesteld aan de hand van de methode die gebruikt wordt bij het ordenen van de websites in een lijst. Wanneer de belangrijke sites bovenaan de lijst staan en de minder belangrijke onderaan, kan er gesproken worden van een hoge kwaliteit.

Vanaf het ontstaan van zoekmachines zijn er verschillende methodes bedacht om websites te ordenen. Sommige methodes zijn gebaseerd op het voorkomen van het gezochte woord of zin. Wanneer het gezochte woord of zin voorkomt op een pagina, wordt deze gewogen met de lengte van het document. Wanneer er sprake is van een hoog gewicht, dan zal de pagina hoog in de lijst staan en anders laag.
Deze methode had echter veel valkuilen. Websites die automatisch gegenereerd werden aan de hand van het gezochte woord of zin kwamen bovenaan de lijst te staan, wat de kwaliteit van deze zoekmachines deed verminderen.

3.1 Het PageRank Concept

Zoals hierboven gezegd, ligt het succes van Google in het gebruik van de PageRank methode. Deze methode wijst aan elke website die bekend is bij Google een belangrijkheidsscore toe. Deze score bepaalt de positie van de website op de resultatenlijst; hogere scores (en dus belangrijke sites) komen bovenaan de lijst, lagere scores (onbelangrijke sites) onderaan.

De belangrijkheid van een pagina A hangt af van twee verschillende factoren;

- De belangrijkheid van een pagina B waarin verwezen wordt naar pagina A.
- Het totaal aantal verwijzingen op pagina B.

Het eerste punt zegt dat een pagina een hoge belangrijkheidsscore krijgt wanneer hij een verwijzing krijgt van een andere belangrijke pagina. Neem bijvoorbeeld het geval dat een site een verwijzing krijgt door www.nu.nl, dit zal meer bijdragen aan zijn belangrijkheidsscore dan een link van www.traffic-builders.com.
Met het tweede punt wordt er bedoeld dat een verwijzing op een pagina met veel verwijzingen minder waarde heeft dan een verwijzing die op een pagina staat met weinig verwijzingen.

Men kan uit het bovenstaande afleiden dat de belangrijkheidsscore van een pagina 'relatief' veel wordt verhoogd wanneer hij een verwijzing krijgt van een belangrijke pagina die weinig verwijzingen bevat. Deze beredenering leidt echter tot een cirkelredenatie; een pagina is belangrijk wanneer hij een verwijzing krijgt van een andere belangrijke pagina, die op zijn beurt belangrijk wordt verklaard omdat hij een verwijzing krijgt van een andere belangrijke pagina. De belangrijkheidsscore van een pagina wordt dus altijd recursief bepaald door de scores van andere pagina's.

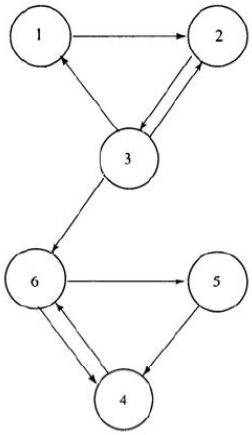
Omdat de scores van elke pagina beinvloed worden door de scores van andere pagina's, kan er worden geconcludeerd dat de PageRank methode gebaseerd is op de gelinkte structuur van het gehele web. De belangrijkheidsscore van een pagina wordt de PageRank waarde genoemd.

Definitie 13 (PageRank waarde). *Veronderstel dat pagina P_i naar pagina P_j verwijst. Definieer $r(P_j)$ als de PageRank waarde van P_j, $|P_j|$ als het aantal verwijzingen vanaf P_j en n voor het totaal aantal pagina's waarbij $1 \leq i, j \leq n$. Er geldt dan*

$$r(P_i) = \sum_{k=1}^{n} c \frac{r(P_k)}{|P_k|}$$

waarbij $c = 1$ wanneer een verwijzing bestaat van P_k naar P_i en $c = 0$ als anders.

De bovenstaande definitie zegt dat de PageRank waarde van P_i wordt bepaald door de PageRank waarden die naar P_i verwijzen te delen door het totaal aantal verwijzingen van de desbetreffende pagina P_k en deze op te sommen.
Men kan afleiden dat wanneer P_k naar P_i verwijst, P_k een fractie $\frac{r(P_k)}{|P_k|}$ bijdraagt aan de PageRank waarde van P_i. Wanneer P_k niet naar P_i verwijst, draagt hij niets toe aan deze waarde.
Om het idee te verduidelijken, zal er een simpel voorbeeld worden gegeven met 6 pagina's. Beschouw de gelinkde structuur uit Figuur 1;



Figuur 1: Simpele voorstelling van web met zes pagina's

Men kan zien dat pagina P_1 een verwijzing krijgt van pagina P_3. Verder krijgt P_2 verwijzingen van P_1 en P_3. Op dezelfde manier kan de gehele graaf worden bekeken.
Door gebruik te maken van Definitie 13 volgt er voor de PageRank waarden;

$$r(P_1) = \frac{r(P_3)}{3}$$

$$r(P_2) = \frac{r(P_1)}{1} + \frac{r(P_3)}{3}$$

$$r(P_3) = \frac{r(P_2)}{1}$$

$$r(P_4) = \frac{r(P_3)}{3} + \frac{r(P_6)}{1}$$

$$r(P_5) = \frac{r(P_4)}{2}$$

$$r(P_6) = \frac{r(P_4)}{2} + \frac{r(P_5)}{1}$$

Het probleem dat hier boven komt drijven is dat de PageRank waarden onbekend zijn. Dit kan worden opgelost door gebruik te maken van een matrix met de eigenschap

$$H\overline{p} = \overline{p}$$

waarbij $\overline{p}$ de vector is die bestaat uit de verschillende PageRank waarden $r(P_j)$. Nu kan er de hyperlink matrix H worden geintroduceerd.

Definitie 14 (Hyperlink matrix H). *Een hyperlink matrix H is een $n \times n$ matrix, met n het totaal aantal pagina's, waarvoor geldt*

$$H_{ij} = \frac{1}{|P_i|}$$

als er een verwijzing bestaat van P_i naar P_j. Als niet, dan geldt er $H_{ij} = 0$.

In woorden stelt H_{ij} de bijdrage van P_i aan P_j voor. Nog steeds geldt dat $|P_i|$ het aantal verwijzingen voorstelt vanaf pagina P_i. Verder is het geoorloofd dat een pagina naar zichzelf verwijst. Hieruit volgt dat H_{ii} niet noodzakelijk gelijk aan 0 hoeft te zijn.
Wanneer er terug wordt gekeken naar het voorbeeld van het web uit Figuur 10, kan men de volgende hyperlink matrix H opstellen;

$$H = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

De niet nul elementen (i, j) corresponderen met de verwijzingen van P_i naar P_j. Deze kunnen worden gezien als kansen. In het bovenstaande voorbeeld kan men bijvoorbeeld zeggen dat er 50% kans is dat men van pagina P_4 naar pagina P_5 zal gaan.

Matrix H heeft verschillende eigenschappen die hieronder zullen worden opgesomd.

- Alle elementen zijn niet negatief of gelijk aan nul.
- H is een (sub)stochastische matrix. De Dangling Nodes (zie Definitie 15 verderop) vormen een rij met nullen. De som van elke overige rij is gelijk aan 1. Wanneer er geen sprake is van een Dangling Node geldt er dus $\sum_{j=1}^{n} H_{ij} = 1$ met $1 \leq i \leq n$ en is matrix H een stochastische matrix. Anders geldt er $\sum_{j=1}^{n} H_{ij} = 0$ en is H een substochastische matrix.
- Wanneer er sprake is van grote n waarden, is H een sparse matrix. Dit is te verklaren omdat elke pagina naar een beperkt aantal andere pagina's verwijst.

Hierboven was vermeld dat matrix H de eigenschap heeft

$$H\overline{p} = \overline{p}$$

Vector $\overline{p}$ is dan de eigenvector bijbehorend bij de eigenwaarde $\lambda = 1$. Men zal $\overline{p}$ de stationaire vector noemen.

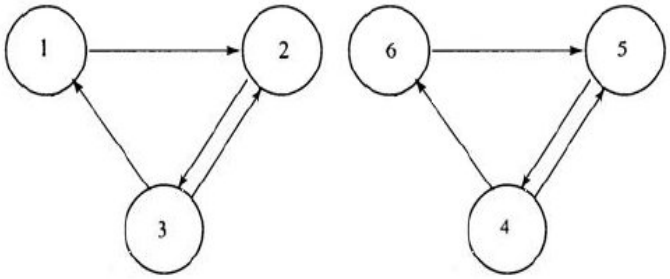
De gelinkte structuur van Google bestaat uit meer dan 25 miljard pagina's. Om de stationaire vector vast te stellen zal er gebruik worden gemaakt van een iteratie methode. In het algemene geval zal er worden uitgegaan van de Power Methode. Deze methode zal uitgebreider worden uitgelegd in Hoofdstuk 4.1

3.2 Tekortkomingen in de PageRank Methode

Wanneer er gebruik wordt gemaakt van de iteratie methode om de eigenvector te bepalen, kan men in verschillende valkuilen komen. Deze worden hieronder opgesomd.

3.2.1 Onsamenhangend web

Het web kan onsamenhangend zijn. H is dan een reducibele matrix en er is dan sprake van twee of meer subwebben. Het hangt dus van de startvector af wat de oplossing $\overline{p}$ zal worden na het uitvoeren

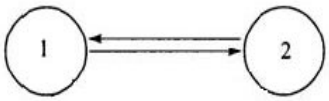


Figuur 2: Webvoorstelling met 6 pagina's

van verschillende iteraties. Een voorbeeld hiervan is het onderstaande web gevormd door 8 pagina's. Dit web kan onderverdeeld worden in twee subwebben bestaande uit de pagina's $\{P_1, P_2, P_3\}$ en $\{P_4, P_5, P_6\}$.

3.2.2 Geen unieke oplossing

De iteratie convergeert niet altijd naar een unieke oplossing. Dit is het geval bij cykels. Neem bijvoorbeeld de volgende simpele webpresentatie met twee pagina's



Figuur 3: Webvoorstelling met 2 pagina's

Pagina 1 verwijst slechts naar pagina 2 en andersom. De vergelijking $H\overline{p} = \overline{p}$ krijgt de vorm van

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \overline{p} = \overline{p}$$

De startvector $\overline{p}$ zal in dit geval altijd de oplossing vormen.

3.2.3 Dangling Nodes en Importance Sinks

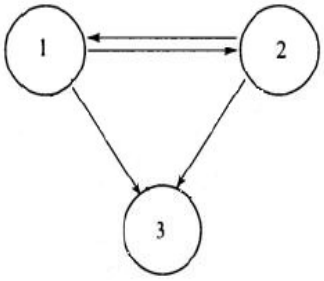
Men is geinteresseerd in de oplossing van de vergelijking $H\overline{p} = \lambda\overline{p}$ waarbij $\lambda = 1$ zodat men $H\overline{p} = \overline{p}$ krijgt. Wanneer er geen eigenwaarde gelijk aan 1 voorkomt, dan zal de enige mogelijke oplossing de triviale oplossing $\overline{p} = \overline{0}$ zijn. Om dit te bedrijpen moet er het begrip Dangling Nodes geintroduceerd worden.

Definitie 15 (Dangling nodes). *Een Dangling node is een pagina die geen verwijzingen heeft naar andere pagina's. De bijdrage van de Dangling node P_i aan de PageRank van de andere pagina's P_j is dan gelijk aan 0.*

80% van de internet pagina's zijn Dangling nodes. Voorbeelden hiervan zijn pdf bestanden en figuren waarnaar er verwezen kan worden maar die niet zelf kunnen verwijzen naar andere pagina's.
Zoals eerder vermeld, is men op zoek naar de oplossing $\overline{p}$ waarvoor geldt $H\overline{p} = \lambda\overline{p}$ waarbij $\lambda = 1$. Wanneer er een Dangling node aanwezig is in de webstructuur, dan zijn de elementen in de bijbehorende rij allen gelijk aan nul. Dit komt omdat een Dangling node naar geen andere pagina kan verwijzen. Matrix H is dan niet meer stochastisch en heeft daarom niet meer vanzelfsprekend een eigenwaarde gelijk aan 1. De unieke oplossing van de vergelijking

$$H\overline{p} = \lambda\overline{p}$$

is daarom de triviale oplossing $\overline{p} = \overline{0}$.
Een voorbeeld van een Dangling node is te zien in Figuur 11. Pagina 1 verwijst naar pagina's 2 en 3,



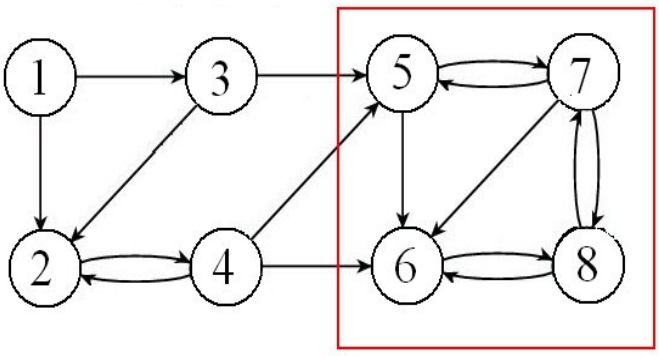
Figuur 4: Webvoorstelling met 3 pagina's

pagina 2 verwijst naar pagina's 1 en 3. De enige pagina zonder verwijzingen is pagina 3. Dit is een Dangling node.
De bijbehorende matrix H is

$$H = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 \end{bmatrix}$$

De enige oplossing van de vergelijking $H\overline{p} = \lambda\overline{p}$ waarbij H als hierboven gegeven is, is de triviale oplossing $\overline{p} = \overline{0}$.

Dangling nodes zijn voorbeelden van importance sinks; het zijn knopen die enkelvoudig voorkomen. Importance sinks kunnen echter ook uit meerdere knopen bestaan. Ze kunnen worden gezien als een groep die bestaat uit meerdere knopen die samen dezelfde functie hebben als een Dangling node, dus die samen geen verwijzingen hebben naar de overige knopen van de webstructuur. De importance sink knopen verzamelen bij elke iteratie meer PageRank waarde en in de tussentijd dragen ze niet bij aan de verhoging van de PageRank waarde van andere pagina's. Ze monopoliseren de belangrijkheidsscore. Een voorbeeld van een importance sink is hieronder weergegeven.



Figuur 5: Webvoorstelling met 8 pagina's

Wanneer men terecht komt op een van de pagina's 5, 6, 7 of 8 dan zijn pagina's 1, 2, 3 of 4 niet meer bereikbaar. De groep van deze eerstgenoemde pagina's heeft geen verwijzingen naar de pagina's 1, 2 3 en 4. Op den duur zal de PageRank waarde van deze laatstgenoemde pagina's gelijk worden aan 0. De pagina's 5, 6, 7 en 8 worden daarom 'importance sinks' genoemd. Men wenst echter om een eigenvector te verkrijgen met waarden die niet negatief en ongelijk aan nul zijn.

3.3 Aanpassingen aan het model

Zoals hierboven opgesomd, kunnen er verschillende valkuilen voorkomen. Door aanpassingen te maken aan de hyperlink matrix H kunnen de bovenstaande problemen worden opgelost zodat er met behulp van de iteratie methode een eigenvector wordt bepaald. Herinner de Markov eigenschap

Er bestaat een unieke positieve eigenvector van matrix A wanneer A stochastisch en irreducibel is. Wanneer A eveneens aperiodiek is, zal de iteratie methode (Power Methode) convergeren naar deze eigenvector, ongeacht de startvector.

Hieruit kan men afleiden dat matrix H zodanig aangepast moet worden dat het een stochastische, irreducibele en aperiodieke matrix wordt.

3.3.1 Stochastische Aanpassing

Beschouw een 'random surfer'. Dit is een persoon die van pagina P_i naar pagina P_j gaat door slechts gebruik te maken van verwijzingen. Wanneer hij op een pagina aankomt met verwijzingen, kiest hij een willekeurige en surft op deze manier door de hyperlink structuur van het web. De kans dat men dan van P_i naar P_j gaat aan de hand van verwijzingen, is gelijk aan $\frac{1}{|P_i|}$ met $|P_i|$ het aantal verwijzingen op P_i. Dit proces kan oneindig lang doorgaan.
Er ontstaat echter een probleem wanneer de 'random surfer' een Dangling node tegenkomt. Omdat in Dangling nodes geen verwijzingen aanwezig zijn naar andere pagina's, wordt de persoon verhindert om verder door het web te 'surfen' door slechts gebruik te maken van verwijzingen. Nu kan het begrip teleportatie worden geintroduceerd.

Definitie 16 (**Teleportatie**). *Er is sprake van teleportatie wanneer men niet via verwijzingen maar via de navigatiebalk op een willekeurige pagina komt.*

Wanneer er n pagina's beschouwd worden, is de kans dat men via teleportatie van de ene pagina naar een andere (of naar dezelfde) gaat gelijk aan $\frac{1}{n}$.
Nu kan H worden aangepast zodanig dat de matrix stochastisch wordt. Deze nieuwe matrix wordt S genoemd. Er wordt vanuit gegaan dat men altijd gebruik maakt van verwijzingen om op andere pagina's te komen en slechts bij Dangling nodes de navigatiebalk gebruikt. De nulrij wordt dan vervangen door een rij bestaande uit de elementen $\frac{1}{n}$.

Definitie 17 (**Matrix S**). *Definieer matrix S als*

$$S = H + \overline{a}\left(\frac{1}{n}\overline{e}^T\right)$$

waarbij $\overline{e}^T = (1, \cdots, 1)$ en a_i gelijk aan 1 is wanneer er bij pagina P_i sprake is van een Dangling node, anders $a_i = 0$ met $1 \leq i \leq n$.

Wanneer men op een Dangling node terecht komt, maakt de navigatiebalk het mogelijk om op elke willekeurige pagina te komen. Alle pagina's hebben nu een gelijke kans $\frac{1}{n}$ om op te komen vanuit een Dangling node. De vector $\overline{a}$ noemt men de Dangling node vector.
Deze theorie kan ter illustratie worden toegepast op matrix H van Figuur 15. De hierbij horende matrix S wordt dan

$$S = \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ 0 & 0 & 0 & 1 \\ \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \end{bmatrix}$$

Samenvattend, matrix S heeft de volgende eigenschappen

- Matrix S is een stochastische matrix.

- De som van iedere rij is gelijk aan 1, dus $\sum_{i=1}^{n} S_{ij} = 1$.
- De grootste eigenwaarde van S is gelijk aan 1.

3.3.2 Primitieve aanpassing

Zoals in het vorige hoofdstuk is vermeld, geldt dat een primitieve matrix irreducibel en aperiodiek is. Men streeft om matrix S aan te passen zodanig dat hij primitief wordt. Dan bestaat een unieke stationaire vector die bepaald wordt door middel van de iteratie methode. Deze methode convergeert dus naar een unieke oplossing.
Tot nu toe is er uit gegaan van de situatie dat een surfer slechts gebruik maakt van de navigatiebalk wanneer hij zich in een Dangling node bevindt. In de realiteit is dit niet zo; elke pagina kan op elk moment via de navigatiebalk bereikt worden (en dus niet alleen in Dangling nodes pagina's). Dit gebeurt bijvoorbeeld wanneer men sneller via de navigatiebalk op de gewenste pagina komt of wanneer de pagina waar men zich op bevindt niet van interesse is. Het gebruik maken van de navigatiebalk op elk willekeurig moment heet α-teleportatie.

Definitie 18 (α **Teleportatie).** *Laat* $0 < \alpha < 1$. *Dit is de kans dat men gebruik maakt van verwijzingen om op andere pagina's te komen. De kans* $(1 - \alpha)$ *is de kans dat men gebruik maakt van de navigatiebalk.*

Het principe van α teleportatie moet worden opgenomen in het model. Om dit te doen, moet eerst teleportatie-matrix T worden gedefinieerd. Bij deze matrix wordt slechts van teleportatie uit gegaan.

Definitie 19 (**Teleportatie-matrix** T)**.** *Matrix* T *is een* $n \times n$ *matrix die als volgt is gedefinieerd*

$$T = \frac{1}{n}\overline{e}\overline{e}^T = \frac{1}{n}\begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix}$$

De teleportatie matrix linkt elke pagina met elkaar. Men kan met gelijke kansen overal naar toe. Een eigenschap van matrix T is dat hij stochastisch is.
Wanneer deze theorie wordt toegepast op het besproken voorbeeld, geldt er voor de teleportatiematrix

$$T = \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{bmatrix}$$

In de realiteit maakt men echter niet alleen gebruik van teleportatie, maar ook van verwijzingen. Door matrix S en T te combineren, volgt Google matrix G.

Definitie 20 (**Google matrix** G)**.** *Matrix* G *is een* $n \times n$ *matrix die als volgt is gedefinieerd*

$$G = \alpha S + (1 - \alpha)\, T = \alpha\left(H + \overline{a}\left(\frac{1}{n}\overline{e}^T\right)\right) + (1 - \alpha)\, T$$

waarbij $0 < \alpha < 1$ *en* $\alpha_i = 1$ *wanneer* P_i *een Dangling node is, anders* $\alpha_i = 0$ *met* $1 \leq i \leq n$.

In woorden betekent dit dat men met kans α naar andere pagina's gaat met behulp van verwijzingen en met kans $(1 - \alpha)$ met behulp van de navigatiebalk.
Google matrix G heeft verschillende eigenschappen die hieronder zullen worden opgesomd.

- G is een stochastische matrix. De som van iedere rij is gelijk aan 1, dus $\sum_{i=1}^{n} G_{ij} = 1$.
- G is een convexe combinatie van de matrices S en T.

- G is een irreducibele matrix.
- G is aperiodiek.
- G is primitief. Dit volgt uit $G^k > 0$ voor een k. Dit geldt voor $k = 1$. Hieruit volgt dat er een unieke positieve vector $\overline{p}$ bestaat. Verder convergeert de Power methode uitgevoerd op G naar deze $\overline{p}$.
- G is een volle matrix.
- De grootste eigenwaarde van G is gelijk aan 1.

3.4 PageRank Vector

Als gevolg van het feit dat matrix G een eigenwaarde gelijk aan 1 heeft, moet er nu een niet-triviale oplossing gevonden worden voor

$$G\overline{p} = \overline{p}$$

Omdat het gehele webstructuur enorm is en miljoenen sites telt, is de bovenstaande vergelijking te groot om op te lossen aan de hand van de basis veeg-methode. Er zal gebruik worden gemaakt van een iteratie methode, namelijk de Power Methode.
Op elk moment bewegen mensen zich over de webstructuur. Zoals eerder gezegd, is de kans dat men van site i naar site j gaat in matrix G weergegeven. Omdat G een primitieve matrix is, zal er na toepassing van een iteratie methode een unieke stabiele oplossing worden gevonden. Deze oplossing vormt de PageRank vector.
De Power Methode bestaat uit de volgende iteratie

$$\overline{p}^{(k+1)} = G\overline{p}^{(k)}$$

waarbij de startvector $\overline{p}^{(0)}$ gegeven is en $k > 0$.
Na het uitvoeren van verschillende iteraties zal de vector $\overline{p}^k$ op den duur verwaarloosbaar weinig veranderen. Men kan nu zeggen dat de vector $\overline{p}$ benaderd wordt door $\overline{p}^k$.

Definitie 21 (PageRank Vector). *Een PageRank vector $\overline{p}$ is een unieke vector in $\mathbb{R}$ die voldoet aan de eigenschappen $G\overline{p} = \overline{p}$ en $e^T\overline{p} = 1$.*

Wanneer men deze vector gevonden heeft, kan de belangrijkheid van de pagina's worden bepaald; een hogere positie in de vector betekent dat de pagina meer belangrijk is.

Nu zullen er een twee belangrijke stellingen uit het BSc verslag van Erik Berkhout geciteerd worden die een rol spelen in het kader van de PageRank methode. Voor de bewijzen, zie het hierboven genoemde verslag.

Stelling 2. *Laat $0 < \alpha < 1$ en $G\overline{p} = \overline{p}$, waar G de Google matrix is. Neem als startvector $\overline{p}^{(0)}$, waarvan alle elementen groter zijn dan 0, waar we de Power methode op toepassen. Dan zijn alle elementen van de Google vector $\overline{p}$ groter dan 0. Met andere woorden, elke bladzijde heeft een positieve PageRank.*
Stelling 3. *Kies de parameter α zodanig dat $0 \leq \alpha \leq 1$. Neem de dekpunt iteratie waarvoor geldt $\overline{p}^{(k+1)} = G\overline{p}^{(k)}$. Laat $\overline{p}^{(0)}$ een willekeurige startvector zijn, waarbij de som van alle elementen 1 moet zijn. Dan geldt*

$$\left|\left|\overline{p} - \overline{p}^{(k)}\right|\right|_1 \leq \alpha^k \left|\left|\overline{p} - \overline{p}^{(0)}\right|\right|_1$$

De waarde van α heeft dus invloed op de convergentie snelheid. Wanneer α dichter bij nul zal liggen, zal de Power Methode sneller convergeren. Hierover moet echter iets worden opgemerkt. Google-Matrix G was gedefinieerd als

$$G = \alpha S + (1 - \alpha)\, T$$

Als er gekozen wordt voor een α dicht bij 0, zal de invloed van matrix S gering zijn. Men maakt dus voornamelijk gebruik van teleportatie via de navigatiebalk dan van verwijzingen. De PageRank methode gaat er echter vanuit dat men zich voornamelijk via verwijzingen door het web verplaatst. Men moet dus een geschikte waarde van α vinden die zowel de Power Methode snel laat convergeren als een goede afspiegeling geeft van de mate waarin men gebruikt maakt van verwijzingen. Google heeft bekend gemaakt dat de waarde van α die gebruikt wordt in de praktijk gelijk is aan 0.85.

4 Methoden

Zoals in het vorige hoofdstuk is uitgelegd, streeft men ernaar de vergelijking $G\overline{p} = \overline{p}$ op te lossen. Hiervoor kan de Power Methode worden toegepast. Deze is al eerder geintroduceerd en zal in dit hoofdstuk uitgebreid aan bod komen.
Echter is dit niet de enige manier om tot een oplossing te komen. Een alternatieve methode die volgens een ander algoritme te werk gaat, is de Arnoldi methode. Deze methode maakt eveneens gebruik van de eigenwaarden van matrix G.

4.1 De Power Methode

De Power Methode werkt volgens een algoritme die gebruik maakt van eigenwaarden; gegeven een matrix A zal het algoritme de grootste eigenwaarde produceren met de bijbehorende (niet gelijk aan nul) eigenvector $\overline{p}$ zodanig dat er geldt $A\overline{p} = \lambda\overline{p}$. Voor deze situatie zoekt men in feite een vector $\overline{p}$ die hoort bij de eigenwaarde 1. Wanneer er voor matrix A de in het vorige hoofdstuk gedefinieerde Google matrix G wordt genomen, dan zal de eigenwaarde 1 voorkomen. De overige eigenwaarden zullen dan een grootte hebben kleiner dan 1 in het spectrum van eigenwaarden. Wanneer λ_i wordt gedefinieerd als de mogelijke eigenwaarden van G, dan geldt er

$$1 = \lambda_1 > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$$

aangezien de grootste eigenwaarde van G gelijk is aan 1.
Om de Power Methode uit te leggen, zal er worden aangenomen dat de eigenvectoren $\overline{p}_i$ bijbehorend bij de eigenwaarden λ_i een basis vormen voor G. Dit is niet altijd noodzakelijk het geval, maar door deze aanname zal het principe van de Power Methode duidelijker zijn.
Zoals eerder gezegd bestaat de Power Methode uit de iteratie

$$\overline{p}^{(k+1)} = G\overline{p}^{(k)}$$

met een gegeven start vector $\overline{p}^{(0)}$ en $k > 0$.
De startvector $\overline{p}^{(0)}$ kan men in de vorm schrijven van

$$\overline{p}^{(0)} = c_1v_1 + c_2v_2 + \cdots + c_nv_n$$

Na het toepassen van k verschillende iteratiestappen volgt;

$$\begin{array}{rclcccl}
\overline{p}^{(1)} & = & G\overline{p}^{(0)} & = & c_1Gv_1 + c_2Gv_2 + \cdots + c_nGv_n & = & c_1\lambda_1v_1 + c_2\lambda_2v_2 + \cdots + c_n\lambda_nv_n \\
\overline{p}^{(2)} & = & G\overline{p}^{(1)} & = & c_1\lambda_1Gv_1 + c_2\lambda_2Gv_2 + \cdots + c_n\lambda_nGv_n & = & c_1\lambda_1^2v_1 + c_2\lambda_2^2v_2 + \cdots + c_n\lambda_n^2v_n \\
\vdots & & \vdots & & \vdots & & \vdots \\
\overline{p}^{(k)} & = & G\overline{p}^{(k-1)} & = & c_1\lambda_1^{k-1}Gv_1 + c_2\lambda_2^{k-1}Gv_2 + \cdots + c_n\lambda_n^{k-1}Gv_n & = & c_1\lambda_1^kv_1 + c_2\lambda_2^kv_2 + \cdots + c_n\lambda_n^kv_n
\end{array}$$

waarbij $\lambda_1 = 1$.
Zoals gezegd, is de grootte van de eigenwaarden λ_i met $i \geq 2$ kleiner dan 1. Als gevolg hiervan zullen na het uitvoeren van verschillende iteratie stappen de waarden van eigenwaarden naar 0 gaan. In formules gezegd, na het uitvoeren van k iteratiestappen volgt er

$$\lambda_i^k \to 0$$

voor $i \geq 2$.
Hieruit volgt dat

$$\overline{p}^{(k)} \to \overline{p} = c_1v_1$$

Dit zegt dat de bijdragen van eigenvectoren v_i met $i \geq 2$ verwaarloosbaar zijn na het uitvoeren van k aantal iteratie stappen. $\overline{p}^{(k)}$ convergeert dan naar de vector $\overline{p} = c_1v_1$. Dit is de vector die correspondeert met eigenwaarde 1.

Nu volgt het algoritme in $\mathbb{R}$ van de Power Methode voor k aantal iteratie stappen met $k = 1, 2, \cdots$. Er wordt niet gewerkt met complexe getallen.

De PageRank Power Methode

Zet stopcriterium = 1
Zet $\lambda^{(0)} = 0$
Kies startvector $\overline{p}^{(0)} \in \mathbb{R}^n$
Kies ϵ

while stopcriterium $\geq \epsilon$
    $\overline{z}^k = G\overline{p}^{(k-1)}$
    $\overline{p}^{(k)} = \overline{z}^{(k)} / \left|\left|\overline{z}^{(k)}\right|\right|_2$
    $\lambda^{(k)} = \overline{p}^{(k-1)T}\overline{z}^{(k)}$
    stopcriterium $= \left|\left|\lambda^{(k)} - \lambda^{(0)}\right|\right|_2$
end

return $\overline{p}^{(k)}$

In de tweede regel van de while loop wordt de vector $\overline{z}^k$ genormaliseerd.

Wanneer er geldt dat het stopcriterium een kleinere waarde heeft dan ϵ, dan wordt de grootste benaderde eigenwaarde gegeven door $\lambda^{(k)}$ en de bijbehorende eigenvector door $\overline{p}^{(k)}$. De gevonden vector heeft een fout van orde

$$\left|\lambda^{(k)} - \lambda_1\right| = \mathcal{O}\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right)$$

De Power Methode convergeert lineair.

4.2 De Arnoldi Methode

Men heeft nu gezien dat de Power Methode gebruikt wordt om de (grootste) eigenwaarde te vinden van een $m \times m$ matrix G. Bij een willekeurig gegeven startvector $\overline{p}^{(0)}$ bepaalt deze methode $\overline{p}^{(1)} = G\overline{p}^{(0)}, \overline{p}^{(2)} = G^2\overline{p}^{(0)}, \overline{p}^{(3)} = G^3\overline{p}^{(0)}, \cdots$ waarbij de vector $\overline{p}^{(i)}$ bij elke iteratie stap wordt bepaald. Deze vector zal op den duur convergeren naar de eigenvector, bijbehorend bij de grootste eigenwaarde λ_1, zoals hierboven is aangetoond.
Bij de Power Methode wordt er slechts gekeken naar het laatste resultaat $G^{k-1}\overline{p}^{(0)}$. De Arnoldi Methode onthoudt echter elk resultaat en vormt hiermee de zogehete Krylov matrix

$$K_n = \begin{bmatrix} \overline{p}^{(0)} & G\overline{p}^{(0)} & G^2\overline{p}^{(0)} & \cdots & G^{n-1}\overline{p}^{(0)} \end{bmatrix}$$

De kolommen van deze matrix zijn niet orthogonaal, maar men kan een orthogonale basis vinden door bijvoorbeeld de Gramm-Schmid methode toe te passen. Deze vectoren die tot de basis behoren zijn een orthogonale basis voor de zo genoemde Krylov subspace.

$$\mathcal{K}_n = \mathcal{K}\left(G, \overline{p}^{(0)}, n\right) = \text{span}\{\overline{p}^{(0)}, \cdots, \overline{p}^{(n)}\} = \text{span}\{\overline{p}^{(0)}, G\overline{p}^{(0)}, \cdots, G^{n-1}\overline{p}^{(0)}\}$$

Vanwege hetzelfde argument dat $G^{n-1}\overline{p}^{(0)}$ goed de grootste eigenvector benaderd (zie vorig hoofdstuk), kan men afleiden dat de vectoren die de basis vormen van de Krylov subspace een goede benadering geven van de eigenvectoren die horen bij de k grootste eigenwaarden.

Deze hierboven besproken methode is echter instabiel. In het Arnoldi algoritme wordt er daarom echter gewerkt met het gestabiliseerde Gramm-Schmidt proces die orthonormale vectoren bepaalt die

een basis vormen voor Krylov subspace $\mathcal{K}_n$. Deze vectoren, $\overline{q}_1, \overline{q}_2, \cdots, \overline{q}_n$, worden de Arnoldi vectoren genoemd en spannen dus deze gehele deelruimte op.
Definieer nu de matrix Q als de matrix die bestaat uit de Arnoldi vectoren. Gegeven is de Google matrix G. Verder is H_m een upper Hessenberg matrix waarvan de vectoren h_{ij} gegeven worden door het Arnoldi algoritme. Deze laatstgenoemde matrix H_m heeft de vorm van

$$H_m = \begin{bmatrix} h_{1,1} & h_{1,2} & h_{1,3} & \cdots & h_{1,m} \\ h_{2,1} & h_{2,2} & h_{2,3} & \cdots & h_{2,m} \\ 0 & h_{3,2} & h_{3,3} & \cdots & h_{3,m} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & h_{m,m-1} & h_{m,m} \end{bmatrix}$$

Omdat men hier slechts in het reële vlak werkt, geldt er dat $Q, G, H_m \in \mathbb{R}^{m \times m}$.
Men kan nu de Hessenberg herleiding opstellen, namelijk

$$H_m = Q^{-1} G Q$$

met $n \in \mathbb{N}$.
Er geldt dan dat H_m en G dezelfde eigenwaarden hebben.
Omdat Q inverteerbaar is, is deze herleiding equivalent aan

$$GQ = QH_m$$

Als men nu $n \leq m$ neemt, dan kan de vergelijking uitschreven worden als

$$G\left[\overline{q}_1, \overline{q}_2, \cdots, \overline{q}_n, \overline{q}_{n+1}, \cdots, \overline{q}_m\right]$$

$$=$$

$$\left[\overline{q}_1, \overline{q}_2, \cdots, \overline{q}_n, \overline{q}_{n+1}, \cdots, \overline{q}_m\right] \begin{bmatrix} h_{1,1} & h_{1,2} & & \cdots & & h_{1,n} & \cdots & & h_{1,m} \\ h_{2,1} & h_{2,2} & & \cdots & & h_{2,n} & \cdots & & h_{2,m} \\ 0 & h_{3,2} & h_{3,3} & & & h_{3,n} & \cdots & & h_{3,m} \\ & 0 & h_{4,3} & \ddots & & & & & \\ & & 0 & \ddots & h_{n-1,n-2} & \vdots & & & \vdots \\ \vdots & & & \ddots & h_{n,n-1} & h_{n,n} & & & \\ & & & & 0 & h_{n+1,n} & & & \\ & & & & & 0 & \ddots & \ddots & \\ 0 & & & \cdots & & & 0 & h_{m,m-1} & h_{m,m} \end{bmatrix}$$

Men concentreert zich nu slechts op een gedeelte van dit systeem. Definieer

$$Q_n = [\overline{q}_1, \overline{q}_2, \cdots, \overline{q}_n]$$

$$Q_{n+1} = \left[\overline{q}_1, \overline{q}_2, \cdots, \overline{q}_n, \overline{q}_{n+1}\right]$$

$$\widetilde{H}_n = \begin{bmatrix} h_{1,1} & h_{1,2} & & \cdots & & h_{1,n} \\ h_{2,1} & h_{2,2} & & \cdots & & h_{2,n} \\ 0 & h_{3,2} & h_{3,3} & & & h_{3,n} \\ & 0 & h_{4,3} & \ddots & & \\ & & 0 & \ddots & h_{n-1,n-2} & \vdots \\ \vdots & & & \ddots & h_{n,n-1} & h_{n,n} \\ & & & & 0 & h_{n+1,n} \end{bmatrix}$$

Stel nu de volgende vergelijking op

$$GQ_n = Q_{n+1}\widetilde{H}_n$$

Merk op dat er in dit systeem geldt $G \in \mathbb{R}^{m\times m}$, $Q_n \in \mathbb{R}^{m\times n}$, $Q_{n+1} \in \mathbb{R}^{m\times n+1}$ en $\widetilde{H}_n = \mathbb{R}^{n+1\times n}$ waardoor beide kanten van de vergelijking resulteren in een $m \times n$ matrix. Men gaat nu de n-de kolommen vergelijken;

$$G\overline{q}_n = h_{1,n}\overline{q}_1 + h_{2,n}\overline{q}_2 + \cdots + h_{n,n}\overline{q}_n + h_{n+1,n}\overline{q}_{n+1} = \sum_{i=1}^{n+1} h_{i,n}\overline{q}_i$$

waarbij n loopt van 1 tot $m-1$.
Deze vergelijking kan worden herschreven, zodat het resulteert in een recursieve formule voor $\overline{q}_{n+1}$. Haal hiervoor de eerste term uit de sommatie. Hieruit volgt dan;

$$h_{n+1,n}\overline{q}_{n+1} = G\overline{q}_n - \sum_{i=1}^{n} h_{i,n}\overline{q}_i \equiv \overline{r}_n$$

Vanwege $H_n = Q^{-1}GQ$ geldt er dat $h_{i,n} = \overline{q}_i^{-1}G\overline{q}_n$. Wanneer $\overline{r}_n \neq \overline{0}$, dan kan de bovenstaande vergelijking worden herschreven tot de recursieve formule;

$$\overline{q}_{n+1} = \frac{G\overline{q}_n - \sum_{i=1}^{n} h_{i,n}\overline{q}_i}{h_{n+1,n}} = \frac{\overline{r}_n}{h_{n+1,n}}$$

waarbij $h_{n+1,n} = ||\overline{r}_n||_2$ om aan de orthonormale eigenschap te voldoen.
De kolommen van matrix Q worden op deze manier recursief bepaald. Het algoritme waarin dit wordt gedaan, staat bekend als het Arnoldi algoritme.

Het Arnoldi Algoritme

Kies willekeurige startvector $\overline{b} \in \mathbb{R}^m$
Normaliseer door $\overline{r}_0 = \overline{q}_1 = \frac{\overline{b}}{||\overline{b}||_2}$
for $j = 1 : n$
    $\overline{r}_j = G\overline{q}_j$

    for $i = 1 : j$
        $h_{i,j} = \overline{q}_i^T\overline{r}_j$
        $\overline{r}_j = \overline{r}_j - h_{i,j}\overline{q}_i$
    end

    $h_{j+1,j} = ||\overline{r}_j||_2$

    if $h_{j+1,j} = 0$
        quit

    $\overline{q}_{j+1} = \frac{\overline{r}_j}{h_{j+1,j}}$
end

De input van dit algoritme bestaat dus uit de Google matrix G, een willekeurige startvector $\overline{b}$ en het aantal iteraties n. Als output verkrijgt men de matrix $Q_{n+1} \in \mathbb{R}^{m\times n+1}$ met orthonormale kolommen en de upper Hessenberg matrix $\widetilde{H}_n \in \mathbb{R}^{n+1\times n}$ die voldoen aan $GQ_n = Q_{n+1}\widetilde{H}_n$. Hier is Q_n de $m \times n$ matrix die bestaat uit de eerste n kolommen van Q_{n+1}. Zoals eerder gezegd vormen deze kolommen van Q_n een orthogonale basis voor de Krylov subspace $\mathcal{K}_n = \mathcal{K}\left(G, \overline{q}^{(1)}, n\right)$.
De Hessenberg herleiding $GQ_n = Q_{n+1}\widetilde{H}_n$ na n aantal iteraties kan nu worden omgeschreven in de Arnoldi fractorizatie;

$$GQ_n = Q_nH_n + \overline{r}_ne_n^T$$

waarbij

$$H_n = \begin{bmatrix} h_{1,1} & h_{1,2} & & \cdots & & h_{1,n} \\ h_{2,1} & h_{2,2} & & \cdots & & h_{2,n} \\ 0 & h_{3,2} & h_{3,3} & & & h_{3,n} \\ \vdots & 0 & h_{4,3} & \ddots & & \\ & & 0 & \ddots & h_{n-1,n-2} & \vdots \\ 0 & & & \ddots & h_{n,n-1} & h_{n,n} \end{bmatrix}$$

en $e_n^T = I_n(:, n)$.
Bij elke iteratie wordt de dimensie van de Hessenberg matrix groter. Bij een gegeven matrix $G \in \mathbb{R}^{m \times m}$, is $H_n \in \mathbb{R}^{n \times n}$ waar $n \leq m$.
Men is nu geinteresseerd om de eigenvector te bepalen die hoort bij de grootste eigenwaarde van G, namelijk 1. Dit is de gewenste Page Rank vector.
De eigenwaarden van matrix H_n heten de Ritz eigenwaarden. Deze benaderen volgens de theorie van gelijksoortige matrices de eigenwaarden van matrix G. Laat nu $\overline{y} \in \mathbb{R}^n$ een eigenvector zijn van H_n die hoort bij de eigenwaarde λ. Wanneer $H_n\overline{y} = \lambda\overline{y}$, dan geldt er;

$$(G - \lambda I)\,\overline{x} = \left(e_n^T \overline{y}\right)\overline{r}_n$$

waar $\overline{x} = Q_n\overline{y}$.
De vector $Q_n\overline{y}$ waar $\overline{y}$ een eigenvector is van H_n bijbehorend bij een Ritz eigenwaarde van H_n, benadert dus een eigenvector van G.
Zoals eerder gezegd is men op zoek naar de eigenvector die hoort bij de grootste eigenwaarde van G. Hiervoor moet dus eerst de grootste eigenwaarde van H_n bepaald worden en de hierbij horende eigenvector $\overline{y} \in \mathbb{R}^n$. Na vermenigvuldiging met matrix $Q \in \mathbb{R}^{m \times n}$ wordt de eigenvector benaderd die hoort bij de (benaderde) grootste eigenwaarde van G. Deze verkregen vector is de gewenste PageRank vector $\overline{x}$.

Een voordeel van het gebruik van de Arnoldi methode is dat het een stabiele methode is. De kans op een break down is verwaarloosbaar klein. Hiermee wordt er bedoeld dat het niet zal voorkomen dat de methode halverwege niet meer zou kunnen voortzetten.
Verder convergeert de methode snel. De Ritz eigenwaarden en de geproduceerde eigenvectoren geven al na relatief weinig iteraties een goede benadering van de eigenwaarden en eigenvectoren van de matrix G.
Aan de Arnoldi methode zitten echter ook nadelen verbonden. Een daarvan is te wijten aan het gebruik van de (gestabiliseerde) Gram-Schmidt orthogonalisatie tijdens de Arnoldi iteratie. Als gevolg hiervan zal het werk kwadratisch toenemen. Elke nieuwe vector $\overline{q}_n$ moet orthogonaliseerd worden tegen alle kolommen van de matrix Q_n. Bij elke iteratie zal daarom de hoeveelheid werk toenemen. Vooral bij grote matrices zoals de Google matrix G zal deze eigenschap vervelend blijken.

Bij de Arnoldi iteratie methode is er gekozen voor het stopcriterium $h_{n+1,n} = 0$. In sommige gevallen zal dit laatste echter nooit voorkomen, waardoor de iteratie methode in theorie eindeloos door zou kunnen gaan. Men zou over kunnen gaan naar het stopcriterium $\left|\lambda^{(n)} - \lambda_1\right| < \epsilon$ waarbij $\lambda^{(n)}$ de grootste eigenwaarde is van H_n, λ_1 de grootste eigenwaarde van G en ϵ de toegestane fout. Al bekend is dat $\lambda_1 = 1$ en $\epsilon > 0$.

5 Numerieke experimenten

In de vorige hoofdstukken is de theorie behandeld van verschillende methoden die gebruikt kunnen worden om de PageRank vector samen te stellen; er is gekeken naar de Power Methode en de Arnoldi Methode. In dit hoofdstuk zal deze theorie worden toegepast; er zullen numerieke experimenten uitgevoerd worden om na te gaan welke methode het meest efficiënt is. Om dit na te gaan, wordt er gekeken naar de uitvoertijd van het experiment en het aantal iteraties die nodig zijn om tot de PageRank vector te komen.

Om de experimenten uit te voeren is een uitgebreide dataset nodig. In dit onderzoek wordt deze dataset op twee manieren verkregen, namelijk

- via het al bestaande Matlab programma surfer.m
- via de nagebootste Link Matrix

Bij elk experiment zal er van beide datasets gebruik gemaakt worden.

Om een indicatie te krijgen van de resultaten, zullen eerst een paar experimenten worden uitgevoerd met het al bestaande Matlab programma pagerank.m. Uit dit experiment zal naar voren komen welke methode de kortste uitvoertijd en minste iteraties heeft.
Vervolgens zullen experimenten worden uitgevoerd met de zelf geimplementeerde Matlab programma's van de methoden. Men kan de resultaten vergelijken met de verkregen resultaten uit pagerank.m en kijken of deze overeenkomen of verschillen.

5.1 Datasets

5.1.1 Surfer.m

Om te laten zien hoe verschillende pagina's naar elkaar verwijzen in de praktijk, is er gekozen om gebruik te maken van het Matlab programma surfer.m. Dit programma vormt testdata van hoe bestaande webpagina's met elkaar linken. De input is een willekeurige bestaande website. Er wordt zichtbaar gemaakt hoe vanaf deze site de linkende pagina's weer op hun beurt linken naar andere pagina's. Men kan aangeven tot hoeveel bladzijdes men dit proces wil laten doorgaan.
De resultaten worden weergegeven in een matrix G. Omdat dit verwarrend is ten opzichte van de Google Matrix, zal matrix G voortaan in het verslag L worden genoemd van Link Matrix. Er geldt $L_{ij} = 1$ wanneer er een verwijzing bestaat van pagina B_j naar B_i. Wanneer er geen verwijzing bestaat, zal er gelden $L_{ij} = 0$. Vanuit deze matrix kunnen de hyperlink matrix H en de Google Matrix G gevormd worden. Surfer.m vormt eveneens een vector U waarin de namen van de webpagina's worden weergegeven; de naam van pagina B_i staat bij U_i.

Het gebruik van Surfer.m brengt ook nadelen met zich mee. Een van deze is dat er slechts matrices gecreëerd kunnen worden van een beperkte grootte. Het programma blijft goed werken bij een matrix van maximaal 130 sites. Het programma zal stoppen om naar verwijzingen te zoeken wanneer men een groter aantal sites wil bekijken. Dit komt omdat het programma een pagina niet goed kan lezen of er is iets niet juist aan de gelezen pagina.

5.1.2 Zelf Geimplementeerde Matlab Programma's

Omdat matrices met een grotere dimensie een betere weerspiegeling geven van hoe snel een methode convergeert, is het belangrijk om experimenten uit te voeren met grotere matrices. In plaats van een Link Matrix L te laten vormen door het bestaande programma surfer.m, is er een programma geimplementeerd die random Link matrices zal genereren. Voor een uitgebreide code zie 'Bijlagen

Matlab Coden - Nabootsen van Link Matrix'. Om een goed beeld te krijgen van de realiteit, is hierbij rekening gehouden dat 80% van de pagina's uit Dangling Nodes bestaan en dat er slechts een beperkt aantal onderlinge verwijzingen bestaan.
Uit deze random gegenereerde $n \times n$ matrix L kunnen nu weer hyperlink matrix H en de Google matrix G worden gevormd. Verder geeft het geimplementeerde programma ook een vector U waarin de namen van de pagina's worden weergegeven in cijfers van 1 tot en met n.
Merk op dat er nu geldt dat $L_{ij} = 1$ wanneer er een verwijzing bestaat van B_i naar B_j, anders $L_{ij} = 0$.

5.2 Matlab Programma Pagerank.m

5.2.1 Uitleg

Er zullen nu een paar experimenten worden uitgevoerd met een al bestaand Matlab Programma, namelijk pagerank.m. Aan de hand hiervan creëert men een verwachting van de resultaten door de verschillende methoden te gebruiken. Bij het uitvoeren van experimenten met de zelf geimplementeerde programma's kunnen de daar verkregen resultaten worden vergeleken met de pagerank.m resultaten.
De twee datasets die bij deze experimenten zullen worden gebruikt zijn, zoals eerder gezegd, verkregen via surfer.m en de nagebootste link matrix.

Het pagerank.m programma richt zich op vijf verschillende methoden, namelijk

- de Power Methode
- de Arnoldi Methode
- de Gauss-Seidel methode
- BiCGSTAB
- GMRES8

Slechts de theorie van de eerste twee methoden is behandeld in het vorige hoofdstuk. Er zal in dit verslag niet worden ingegaan op de laatste drie methoden. Men zal aan de hand van de resultaten zien dat deze niet als meest efficiënt naar voren zullen komen en dus niet relevant zijn in dit onderzoek.

Men moet opmerken dat het programma gebruik maakt van Arnoldi8. Hiermee wordt bedoeld dat elke iteratie die het programma uitvoert equivalent is aan 8 iteraties van het Arnoldi iteratie algoritme beschreven in het vorige hoofdstuk. Wanneer er uit het programma blijkt dat er 'slechts' 1 iteratie nodig is, moet men rekening houden dat dit in feite 8 iteraties zijn.

5.2.2 Resultaten Pagerank.m

Er zullen verschillende experimenten worden uitgevoerd met matrices van vijf verschillende dimensies. Eerst zal er een 150×150 matrix worden beschouwd die via zowel surfer.m als door middel van de random generator is verkregen. Bij gebruik van surfer.m worden als input zowel de bekende webpagina http://www.nos.nl gebruikt als de minder bekende http://www.lyricsfreak.com/. Men zal dan na kunnen gaan of er een aantoonbaar verschil is in snelheid en aantal iteraties wanneer er gekeken wordt naar pagina's met onderling een andere structuur en pagina's die nagebootst worden in Matlab. Met een andere structuur wordt er bedoeld dat de pagina's op een verschillende manier aan elkaar gelinkt zijn.
Omdat de 150×150 matrix een relatief kleine dimensie heeft en men niet weet of de resultaten representatief zijn, zullen er nog 4 andere matrices worden bekeken met dimensies 1000×1000, 2000×2000, 3000×3000 en 4000×4000 die met de random generator zijn gecreëerd. Als stopcriterium is er

gekozen voor $\epsilon = 10^{-8}$ en als teleportatiecoëfficient 0.85.
De resultaten zijn weergegeven in de onderstaande tabellen.

Methode	Tijd (sec)	Aantal iteraties
Power Methode	0.19	81
Arnoldi8	0.12	4
Gauss-Seidel	0.06	20
BiCGSTAB	0.30	23
GMRES8	0.17	27

Tabel 1: Methoden toegepast op 150×150 matrix genereerd door surfer.m (input: www.nos.nl)

Methode	Tijd (sec)	Aantal iteraties
Power Methode	0.01	84
Arnoldi8	0.01	2
Gauss-Seidel	0.01	45
BiCGSTAB	0.04	13
GMRES8	0.04	11

Tabel 2: Methoden toegepast op 150×150 matrix genereerd door surfer.m (input: http://www.lyricsfreak.com/)

Methode	Tijd (sec)	Aantal iteraties
Power Methode	0.00	6
Arnoldi8	0.01	1
Gauss-Seidel	0.00	9
BiCGSTAB	0.05	7
GMRES8	0.03	7

Tabel 3: Methoden toegepast op 150×150 matrix gegenereerd door random generator

Methode	Tijd (sec)	Aantal iteraties
Power Methode	0.01	5
Arnoldi8	0.01	1
Gauss-Seidel	0.03	8
BiCGSTAB	0.05	6
GMRES8	0.04	6

Tabel 4: Methoden toegepast op 1000×1000 matrix gegenereerd door random generator

Methode	Tijd (sec)	Aantal iteraties
Power Methode	0.01	4
Arnoldi8	0.03	1
Gauss-Seidel	0.05	8
BiCGSTAB	0.07	6
GMRES8	0.05	6

Tabel 5: Methoden toegepast op 2000×2000 matrix gegenereerd door random generator

Methode	Tijd (sec)	Aantal iteraties
Power Methode	0.03	4
Arnoldi8	0.06	1
Gauss-Seidel	0.09	8
BiCGSTAB	0.10	6
GMRES8	0.07	5

Tabel 6: Methoden toegepast op 3000×3000 matrix gegenereerd door random generator

Methode	Tijd (sec)	Aantal iteraties
Power Methode	0.05	4
Arnoldi8	0.08	1
Gauss-Seidel	0.14	8
BiCGSTAB	0.12	5
GMRES8	0.09	5

Tabel 7: Methoden toegepast op 4000×4000 matrix gegenereerd door random generator

In de tabellen hierboven kan men aflezen welke tijdsduur elke methode nodig heeft om tot de gewenste PageRank vector $\overline{p}$ te komen en hoeveel iteraties er plaats vinden. Men is vooral geinteresseerd in de tijdsduur. In de histogrammen in 'Bijlagen - Figuren - Resultaten pagerank.m - Tijdhistogrammen' worden de tijdsduren per methoden per matrix dimensie vergeleken.

Men kan aan de hand van de staafdiagrammen en de resultaten hierboven zien dat er een aanzienlijk verschil bestaat in de tijdsduur en aantal iteraties wanneer er surfer.m of de random generator wordt gebruikt. Het aantal iteraties ligt bij het gebruik van surfer.m aanzienlijk hoger dan wanneer er de random generator wordt gebruikt bij het vormen van de link matrix. Een ander opmerkelijk verschil is dat wanneer er een website als www.nos.nl als input wordt gebruikt bij surfer.m, de tijdsduur van de methoden hoger zal liggen dan wanneer er een website wordt genomen als www.lyricsfreak.nl of wanneer de link matrix door middel van de random generator wordt gevormd.

Om het verschil aan te tonen in de structuur van de pagina's www.nos.nl en www.lyricsfreak.nl, zijn er twee figuren gemaakt. Deze geven weer hoe 150 verschillende pagina's met elkaar verbonden zijn wanneer er als startpagina een van de twee hierboven genoemde pagina's wordt genomen. De plotten zijn te zien in 'Bijlage - Figuren - Visuele Weergaven van Gelinkte Pagina's'. Daar waar een blauwe stip is weergegeven, betekent dat er een link bestaat tussen de bijbehorende twee websites.
Wat opvalt is dat er meer onderlinge verbindingen bestaan tussen de pagina's wanneer er als input www.lyricsfreak.nl wordt genomen dan wanneer www.nos.nl als startpagina wordt beschouwd. Dit is in tegenstelling met wat verwacht werd. Men vermoedde juist dat er meer verbindingen zouden bestaat tussen de 150 pagina's met als input www.nos.nl vanwege de langere tijdsduur die de methoden erover doen om te convergeren. Dit is een punt voor verder onderzoek.

Wanneer men zich beperkt tot de resultaten gegenereerd door de random generator, zal men zien dat het groter worden van de matrixdimensie een toename in de tijdsduur als gevolg zal hebben. Het aantal iteraties bij de verschillende methoden blijft echter ongeveer constant. Zoals eerder gezegd, is men vooral geinteresseerd in een methode die in een zo kort mogelijke tijd convergeert. Bij het gebruik van de random generator komt de Power Methode als beste naar voren; deze heeft, ongeacht de matrixdimensie, altijd de kortste tijdsduur. De tweede plek wordt ingenomen door de Arnoldi methode. Wisselend op de 3^e, 4^e en 5^e plek komen Gauss-Seidel, GMRES8 en de BiCGSTAB. Men heeft nu gezien dat deze laatste drie methoden minder relevant zijn in dit onderzoek omdat ze niet de snelste convergeertijd zullen leveren ten opzichte van de Power- en Arnoldi8 Methode.

Als men echter slechts kijkt naar de resultaten gegenereerd met behulp van surfer.m, kan men tot

andere conclusies komen. Nog steeds is gewenst om een zo kort mogelijke tijdsduur te verkrijgen. Uit de resultaten komt nu naar voren dat de Arnoldi8- en de Power Methode niet altijd de twee snelste methoden zijn. Dit is in tegenspraak met wat men eerder heeft geconcludeerd.

Er is gekozen om de resultaten die gegenereerd zijn aan de hand van de random generator aan te houden. De resultaten die verkregen zijn door het gebruik van surfer.m zijn niet representatief voor de werkelijkheid aangezien er gewerkt wordt met relatief kleine matrices. Aan de andere kant kan men echter niet met zekerheid zeggen dat de resultaten van de random generator volledig betrouwbaar zijn. Het blijft een model en een versimpeling van de werkelijkheid. De conclusies die worden genomen zouden kunnen afwijken wanneer men het gehele web bekijkt.

5.3 Resultaten Zelf Geimplementeerde Matlab Programma's

Na het bespreken van de theorie over de Power Methode en Arnoldi methode, kunnen nu de experimenten plaatsvinden met behulp van de zelf geimplementeerde Matlab programma's. Bij elke methode zullen matrices van verschillende dimensies worden beschouwd. Bij elk van deze matrices zullen de iteratieve methoden worden toegepast waarbij de fout ϵ gevarieerd zal worden. Er zal worden gekeken naar de duur die elke methode erover doet om tot de gewenste PageRank vector $\overline{p}$ te komen en naar het aantal iteraties die nodig zijn. Uit de voorgaande theorie volgt dat de Power Methode na convergentie de gewenste vector zal leveren. Bij de Arnoldi methode daarentegen zal de geleverde vector na de iteraties nog vermenigvuldigd moeten worden met matrix Q om tot de PageRank vector te komen. Deze vermenigvuldigingsstap zal eveneens tijd kosten.
Als dataset bij matrices tot een dimensie van 150 wordt gebruik gemaakt van zowel data gegenereerd door surfer.m als door de random generator. Bij grotere dimensies wordt er gebruik gemaakt van slechts data gegenereerd door de random generator.

5.3.1 Resultaten Power Methode

Als eerste zal er worden gekeken naar de Power Methode. Als stopcriterium is er genomen

$$\left|\lambda^{(k)} - \lambda_1\right| < \epsilon$$

met $\epsilon > 0$, λ_1 de grootste eigenwaarde en $\lambda^{(k)}$ de eigenwaarde bij de k-de iteratie. In dit geval is de grootste eigenwaarde gelijk aan 1. De fout ϵ laat men variëren tussen 10^{-2} en 10^{-12}. De resultaten zijn in tabellen 8, 9 en 10 weergegeven. Het aantal iteraties is in deze tabellen afgekort door middel van de letters it.

Matrix Dimensie	$\epsilon = 10^{-2}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-6}$	$\epsilon = 10^{-8}$	$\epsilon = 10^{-10}$	$\epsilon = 10^{-12}$
5	0.0040 sec 2 it.	0.0031 sec 4 it.	0.0029 sec 6 it.	0.0027 sec 9 it.	0.0029 sec 12 it.	0.0029 sec 14 it.
30	0.0042 sec 10 it.	0.0035 sec 23 it.	0.0033 sec 37 it.	0.0036 sec 51 it.	0.0042 sec 65 it.	0.0029 sec 79 it.
50	0.0029 sec 12 it.	0.0033 sec 30 it.	0.0048 sec 48 it.	0.0046 sec 66 it.	0.0046 sec 84 it.	0.0057 sec 102 it.
100	0.0033 sec 12 it.	0.0045 sec 33 it.	0.0049 sec 54 it.	0.0062 sec 74 it.	0.0071 sec 95 it.	0.0085 sec 116 it.
150	0.0022 sec 10 it.	0.0052 sec 35 it.	0.0054 sec 60 it.	0.0094 sec 84 it.	0.0132 sec 109 it.	0.0208 sec 133 it.

Tabel 8: Resultaten Power Methode door gebruik van Surfer.m (input: www.nos.nl)

Matrix Dimensie	$\epsilon = 10^{-2}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-6}$	$\epsilon = 10^{-8}$	$\epsilon = 10^{-10}$	$\epsilon = 10^{-12}$
5	0.0020 sec 2 it.	0.0019 sec 4 it.	0.0021 sec 6 it.	0.0019 sec 9 it.	0.0018 sec 12 it.	0.0019 sec 14 it.
30	0.0018 sec 2 it.	0.0033 sec 2 it.	0.0018 sec 3 it.	0.0017 sec 4 it.	0.0019 sec 5 it.	0.0020 sec 6 it.
50	0.0018 sec 2 it.	0.0017 sec 3 it.	0.0017 sec 4 it.	0.0018 sec 5 it.	0.0019 sec 6 it.	0.0018 sec 7 it.
100	0.0018 sec 3 it.	0.0018 sec 6 it.	0.0019 sec 8 it.	0.0020 sec 10 it.	0.0022 sec 13 it.	0.0021 sec 15 it.
150	0.0025 sec 11 it.	0.0033 sec 37 it.	0.0046 sec 62 it.	0.0055 sec 88 it.	0.0085 sec 144 it.	0.0077 sec 139 it.

Tabel 9: Resultaten Power Methode door gebruik van Surfer.m (input: www.lyricsfreak.com)

Matrix Dimensie	$\epsilon = 10^{-2}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-6}$	$\epsilon = 10^{-8}$	$\epsilon = 10^{-10}$	$\epsilon = 10^{-12}$
5	0.0066 sec 5 it.	0.0061 sec 9 it.	0.0019 sec 14 it.	0.0070 sec 18 it.	0.0073 sec 22 it.	0.0070 sec 26 it.
30	0.0059 sec 4 it.	0.0061 sec 8 it.	0.0082 sec 13 it.	0.0065 sec 18 it.	0.0065 sec 23 it.	0.0083 sec 27 it.
50	0.0062 sec 4 it.	0.0102 sec 10 it.	0.0065 sec 15 it.	0.0080 sec 20 it.	0.0080 sec 25 it.	0.0074 sec 30 it.
100	0.0063 sec 3 it.	0.0082 sec 4 it.	0.0076 sec 8 it.	0.0072 sec 13 it.	0.0075 sec 17 it.	0.0079 sec 20 it.
150	0.0068 sec 3 it.	0.0066 sec 5 it.	0.0086 sec 8 it.	0.0169 sec 10 it.	0.0090 sec 13 it.	0.0074 sec 15 it.
200	0.0065 sec 2 it.	0.005771 sec 5 it.	0.0077 sec 7 it.	0.0130 sec 9 it.	0.0114 sec 13 it.	0.0087 sec 15 it.
1000	0.0066 sec 2 it.	0.0081 sec 3 it.	0.0101 sec 4 it.	0.0128 sec 6 it.	0.0220 sec 8 it.	0.0276 sec 9 it.
2000	0.0164 sec 2 it.	0.0257 sec 3 it.	0.0312 sec 4 it.	0.0425 sec 5 it.	0.0592 sec 7 it.	0.0714 sec 9 it.
3000	0.0350 sec 2 it.	0.0527 sec 3 it.	0.0759 sec 4 it.	0.0923 sec 5 it.	0.1256 sec 7 it.	0.1345 sec 8 it.
4000	0.0687 sec 2 it.	0.0970 sec 3 it.	0.0951 sec 3 it.	0.1477 sec 5 it.	0.2088 sec 6 it.	0.3005 sec 7 it.

Tabel 10: Resultaten Power Methode door gebruik van random generator

Aan de hand van deze resultaten kunnen verschillende verschijnselen worden waargenomen.

Er zal eerst worden gekeken naar het aantal iteraties dat de Power Methode nodig heeft om te convergeren. Bij zowel Tabel 8, 9 en 10 neemt het aantal iteraties toe bij het verkleinen van de fout ϵ (bij enkele gevallen komen uitzonderingen voor). Bij een kleinere fout heeft men dus meer iteraties nodig om tot een oplossing te klomen.

Wat verder opvalt is dat het aantal iteraties in Tabel 8 en 9 toeneemt bij het vergroten van de matrix dimensie, terwijl in Tabel 10 dit aantal juist afneemt. Dit zou tot de vraag kunnen leiden in hoeverre de nabootsing van de link matrix door middel van de random generator de werkelijkheid correct modelleert. Al eerder is gezegd dat men zich aan de resultaten van de random generator zal houden vanwege het feit dat het hiermee mogelijk is om matrices met grotere dimensies te modelleren die in grootte dichter bij de werkelijkheid komen dan bij het gebruik van surfer.m. In beide gevallen zou dit wel betekenen dat de convergentiesnelheid afhankelijk is van de matrixgrootte.

Men kan verder zien dat de spreiding in het aantal iteraties in Tabel 8 en 9 groter is dan bij Tabel 10. Een grotere spreiding zou dus het gevolg zijn van het gebruik van surfer.m voor het vormen van de link matrix.

Een ander opmerkelijk verschijnsel in Tabel 8 is dat het aantal iteraties relatief veel toeneemt bij een matrix dimensie van 30. In Tabel 9 gebeurt dit pas bij een matrix dimensie van 150. Bij matrixdimensies kleiner dan 150 blijft het aantal iteraties redelijk dicht bij elkaar liggen terwijl bij $n = 150$ dit aantal opmerkelijk hoger ligt. Men zou hieruit kunnen concluderen dat bij sommige pagina's er al vele verwijzingen voorkomen wanneer er naar een webstructuur wordt gekeken die bestaat uit een kleiner aantal websites. Bij sites met een andere structuur komen pas veel verwijzingen voor bij een web met een groter aantal websites.

Wanneer er naar de tijdsduur wordt gekeken, ziet men duidelijk in Tabel 10 dat deze toeneemt bij het verkleinen van de fout ϵ. Bij een kleinere fout heeft het programma dus meer tijd nodig om te convergeren. In Tabel 8 en 9 geldt dit over het algemeen ook, maar vele resultaten zijn ook afwijkend van wat er verwacht wordt.

In Tabel 10 kan men zien dat het groter worden van de matrixdimensie, een langere tijdsduur als gevolg heeft. Dit kan men terugzien bij iedere fout ϵ. De tijdsduren nemen geleidelijk toe bij het vergroten van matrixdimensie. Bij Tabel 8 en 9 kan men dit eveneens globaal terugzien, maar in de tijdsduren zit een minder duidelijk toenemend patroon in. Vele tijdsduren wijken af van de verwachte tijdsduren.

Verder hebben het aantal iteraties en de tijdsduren een verband met elkaar. Wanneer de Power Methode meer iteraties nodig heeft, zal deze methode een langere tijdsduur eisen.

Een verder onderzoekspunt zou de convergentie snelheid kunnen zijn. Deze kan men bepalen aan de hand van $\left|\frac{\lambda_2}{\lambda_1}\right|$. Dit wordt echter in dit verslag buiten beschouwing gelaten. De geinteresseerden kunnen dit nalezen in het verslag van Erik Berkhof.

5.3.2 Resultaten Arnoldi Methode

Nu zal er worden gekeken naar de resultaten van de Arnoldi Methode. Het stopcriterium is hetzelfde als die van de Power Methode, namelijk

$$\left|\lambda^{(k)} - \lambda_1\right| < \epsilon$$

met $\epsilon > 0$, λ_1 de grootste eigenwaarde van de Google matrix en $\lambda^{(k)}$ de grootste eigenwaarde van de Hessenberg matrix H_k. Gezien was dat de grootste eigenwaarde in dit geval gelijk was aan 1. De fout laat men weer variëren tussen 10^{-2} en 10^{-12}.

De tijdsduur die verkregen is, is de tijd die de methode nodig heeft om tot de PageRank vector te komen. Zoals in de theorie is uitgelegd, werd de verkregen eigenvector na de Arnoldi iteratie vermenigvuldigd met een matrix om tot deze PageRank vector te komen. Dit kost eveneens tijd. De in de tabellen hieronder genoteerde tijdsduur is de tijdsduur die de Arnoldi iteratie erover doet en de vermenigvuldigingstijd.

Verder wordt het aantal iteraties die de methode nodig heeft om te convergeren weergegeven.

De resultaten zijn weergegeven in Tabel 11, 12 en 13. Het aantal iteraties is afgekort met it.

Matrix Dimensie	$\epsilon = 10^{-2}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-6}$	$\epsilon = 10^{-8}$	$\epsilon = 10^{-10}$	$\epsilon = 10^{-12}$
5	0.0026 sec 2 it.	0.0104 sec 3 it.	0.0120 sec 3 it.	0.0090 sec 3 it.	0.0089 sec 3 it.	0.0239 sec 3 it.
30	0.0024 sec 2 it.	0.0029 sec 4 it.	0.0027 sec 6 it.	0.0025 sec 7 it.	0.0031 sec 7 it.	0.0028 sec 7 it.
50	0.0083 sec 2 it.	0.0104 sec 5 it.	0.0019 sec 7 it.	0.0031 sec 9 it.	0.0040 sec 10 it.	0.0046 sec 11 it.
100	0.0024 sec 2 it.	0.0104 sec 5 it.	0.0131 sec 9 it.	0.0150 sec 11 it.	0.0124 sec 13 it.	0.0168 sec 15 it.
150	0.0008 sec 2 it.	0.0018 sec 6 it.	0.0037 sec 10 it.	0.0057 sec 12 it.	0.0053 sec 16 it.	0.0073 sec 17 it.

Tabel 11: Resultaten Arnoldi Methode door gebruik van Surfer.m (input: www.nos.nl)

Matrix Dimensie	$\epsilon = 10^{-2}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-6}$	$\epsilon = 10^{-8}$	$\epsilon = 10^{-10}$	$\epsilon = 10^{-12}$
5	0.0053 sec 3 it.	0.0090 sec 3 it.	0.0083 sec 3 it.	0.0061 sec 3 it.	0.0128 sec 3 it.	0.0154 sec 3 it.
30	0.0053 sec 2 it.	0.0086 sec 2 it.	0.0026 sec 3 it.	0.0099 sec 3 it.	0.0112 sec 3 it.	0.014 sec 3 it.
50	0.0024 sec 2 it.	0.0061 sec 2 it.	0.0030 sec 3 it.	0.0057 sec 4 it.	0.0196 sec 4 it.	0.0075 sec 4 it.
100	0.0080 sec 2 it.	0.0137 sec 3 it.	0.0058 sec 4 it.	0.0108 sec 6 it.	0.0146 sec 7 it.	0.0026 sec 9 it.
150	0.0079 sec 2 it.	0.0058 sec 4 it.	0.0109 sec 7 it.	0.0107 sec 7 it.	0.0116 sec 9 it.	0.0172 sec 11 it.

Tabel 12: Resultaten Arnoldi Methode door gebruik van Surfer.m (input: www.lyricsfreak.com)

Matrix Dimensie	$\epsilon = 10^{-2}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-6}$	$\epsilon = 10^{-8}$	$\epsilon = 10^{-10}$	$\epsilon = 10^{-12}$
5	0.0037 sec 2 it.	0.0115 sec 2 it.	0.0040 sec 2 it.	0.0035 sec 2 it.	0.0115 sec 2 it.	0.0131 sec 2 it.
30	0.0002 sec 3 it.	0.0037 sec 6 it.	0.0144 sec 6 it.	0.0133 sec 6 it.	0.0133 sec 6 it.	0.0134 sec 6 it.
50	0.0128 sec 2 it.	0.0151 sec 6 it.	0.0155 sec 9 it.	0.0160 sec 10 it.	0.0160 sec 10 it.	0.0162 sec 10 it.
100	0.0150 sec 2 it.	0.0146 sec 3 it.	0.0142 sec 7 it.	0.0156 sec 9 it.	0.0194 sec 12 it.	0.0204 sec 14 it.
150	0.0037 sec 2 it.	0.0134 sec 3 it.	0.0142 sec 4 it.	0.0137 sec 7 it.	0.0152 sec 10 it.	0.0234 sec 13 it.
200	0.0153 sec 2 it.	0.0144 sec 2 it.	0.0144 sec 4 it.	0.0142 sec 6 it.	0.0160 sec 9 it.	0.0174 sec 12 it.
1000	0.0093 sec 2 it.	0.0094 sec 2 it.	0.0134 sec 3 it.	0.0133 sec 4 it.	0.0270 sec 6 it.	0.0225 sec 8 it.
2000	0.0302 sec 2 it.	0.0251 sec 2 it.	0.0355 sec 2 it.	0.0396 sec 3 it.	0.0541 sec 6 it.	0.3296 sec 38 it.
3000	0.0433 sec 2 it.	0.0426 sec 2 it.	0.0753 sec 2 it.	0.0764 sec 4 it.	0.1077 sec 5 it.	0.2469 sec 14 it.
4000	0.0809 sec 2 it.	0.0711 sec 2 it.	0.0816 sec 2 it.	0.1035 sec 3 it.	0.1549 sec 4 it.	0.6839 sec 19 it.

Tabel 13: Resultaten Arnoldi Methode door gebruik van random generator

Wanneer men naar de resultaten kijkt, valt het meteen op dat het verkleinen van de fout een toename in het aantal iteraties als gevolg heeft. Dit geldt bij elk van de drie bovenstaande tabellen. Bij kleinere matrixdimensies blijft het aantal iteraties bij de verschillende fouten vrijwel constant. Pas vanaf $n = 30/n = 50$ ziet men een geleidelijke toename komen in het aantal iteraties bij het verkleinen van de fout. Wat opvalt in Tabel 13 is dat bij de overgang tussen de fout 10^{-10} en 10^{-12} het aantal iteraties sterk toeneemt bij de matrixdimensies 2000, 3000 en 4000. Waarom dit zo is, is een punt voor verder onderzoek.

Bij Tabel 11 en 12 neemt het aantal iteraties in het algemeen toe bij het vergroten van de matrix dimensie. Bij Tabel 11 geldt dit vanaf een fout van ongeveer grootte 10^{-4}. Bij een grotere fout blijft het aantal iteraties vrijwel constant bij het toenemen van n. In Tabel 12 is dit verschijnsel vanaf een fout van 10^{-6} te zien. Bij grotere fouten zijn er wat schommelingen in het aantal iteraties en is er geen vast patroon te vinden waaruit conclusies getrokken kunnen worden.

Er is echter geen vast patroon te vinden in het aantal iteraties bij het vergroten van de matrixdimensies bij de verschillende fouten ϵ in Tabel 13. De aantallen nemen soms toe, dan weer af waardoor er geen duidelijke structuur te herkennen is.

Wanneer er wordt gekeken naar de tijdsduren, ziet men dat deze over het algemeen toenemen bij

- Het verkleinen van de fout ϵ
- Het vergroten van de matrix dimensie n

In sommige gevallen zijn er echter afwijkingen in het verwachte patroon. Het toenemen van het aantal iteraties gaat gepaard met het toenemen van de tijdsduur. Een globale waarneming is dat de tijdsduren die gevonden zijn met behulp van surfer.m relatief kleiner zijn dan degene verkregen door gebruik van de random generator.

Wanneer men de resultaten vergelijkt die verkregen zijn na het uitvoeren van de Power Methode en de Arnoldi Methode, kan er geconcludeerd worden dat de Power Methode sneller is. De Power Methode is in zowel de gevallen waarbij er gebruik is gemaakt van surfer.m als in die waarbij de random generator te hulp is geweest, sneller. Het vermoeden dat verkregen was aan de hand van pagerank.m is dus juist.

6 Onderzoek - De Rayleigh Quotiënt Methode

Tot nu toe heeft men twee verschillende iteratieve methoden onderzocht waarmee de PageRank vector $\overline{p}$ kon worden bepaald. Deze twee methoden zijn relatief bekend en al toepasbaar in de praktijk.
In dit hoofdstuk zal men zich echter concentreren op een nieuwere, minder bekende methode genaamd de Rayleigh Quotiënt Methode.

6.1 Theorie

Men streeft bij de Rayleigh Quotiënt Methode om de vector $\overline{v} \in \mathbb{R}^n$ en een $\mu \in \mathbb{R}$ te vinden zodanig dat er wordtvoldaan aan

$$A\overline{v} = \mu B\overline{v}$$

Hier herkent men een eigenwaarde probleem in. De gezochte μ stelt de eigenwaarde voor en $\overline{v}$ de bijbehorende eigenvector. In feite komt de methode neer om op een andere, derde manier de PageRank vector $\overline{v}$ vast te stellen.

Men zal nu $B = I$ nemen voor het gemak. Verder wordt A als een symmetrische, positief definiete, $n \times n$ matrix genomen. Omdat de dimensie van A gelijk is aan n, volgt dat er n eigenwaarden zijn. Deze worden weergegeven als μ_i met $i \in [1, n]$ waarbij geldt

$$0 < \mu_1 \leq \mu_2 \leq \cdots \leq \mu_n$$

Omdat A symmetrisch positief definiet is, geldt er dat $\mu_i \geq 0$ voor alle i.

Als men per eigenwaarde het eigenwaarde probleem beschouwt, krijgt men;

$$A\overline{v}_i = \mu_i\overline{v}_i \tag{1}$$

waarbij $\overline{v}_i \neq 0$.

Om $\overline{v}$ vast te stellen bij de bijbehorende μ wordt er gebruik gemaakt van het Rayleigh quotient

$$r\left(\overline{x}\right) = \frac{\overline{x}^T A\overline{x}}{\overline{x}^T\overline{x}}$$

waarbij $\overline{x}$ een willekeurige vector is in $\mathbb{R}^n$ en ongelijk aan $\overline{0}$. Wanneer als vector de eigenvector $\overline{x} = \overline{v_i}$ wordt genomen, geldt er vanwege (1);

$$r\left(\overline{v}_i\right) = \frac{\overline{v}_i^T A\overline{v}_i}{\overline{v}_i^T\overline{v}_i} = \frac{\overline{v}_i^T\mu_i\overline{v}_i}{\overline{v}_i^T\overline{v}_i} = \mu_i\frac{\overline{v}_i^T\overline{v}_i}{\overline{v}_i^T\overline{v}_i} = \mu_i \tag{2}$$

Wat hieruit volgt is dat men de eigenwaarde μ_i verkrijgt wanneer de eigenvector $\overline{v}_i$ als variabele aan de functie $r\left(\overline{x}\right)$ wordt meegegeven.

Gebruikmakend van eigenschap (2) kan er worden afgeleid;

$$\mu_1 \leq r\left(\overline{x}\right) \leq \mu_n$$

$$r\left(\overline{v}_1\right) = \mu_1$$

$$r\left(\overline{v}_n\right) = \mu_n$$

waarbij $\overline{v}_1$ en $\overline{v}_n$ de bijbehorende eigenvectoren zijn bij respectievelijk μ_1 en μ_n en $\overline{x}$ een willekeurige vector voorstelt. Voor het bewijs zie de wikipedia pagina over 'Min-Max Theorem'.

Omdat er gestreefd wordt om de eigenvector te bepalen die hoort bij de grootste eigenwaarde, wordt er gezocht naar een vector $\overline{x}$ waarvoor geldt

$$\max_{\overline{x}\in\mathbb{R}^n} r\left(\overline{x}\right) = \mu_n$$

Men wil in feite het maximum bepalen van de functie $r\left(\overline{x}\right)$. Deze vormt dan de grootste eigenwaarde.

Uit de analyse volgt dat voor het maximum geldt dat de gradient gelijk is aan nul. Dit is de aanpak die men zal gebruiken bij deze methode.
Het zoeken van het maximum of minimum van de functie $r\left(\overline{x}\right)$ is equivalent met het bepalen van de vector $\overline{x}$ waarvoor geldt

$$\nabla r\left(\overline{x}\right) = 0$$

Men zal nu de gradiënt van $r\left(\overline{x}\right)$ afleiden. Herinner dat

$$\nabla r\left(\overline{x}\right) = \left[\frac{\partial r\left(\overline{x}\right)}{\partial x_1}, \frac{\partial r\left(\overline{x}\right)}{\partial x_2}, \cdots, \frac{\partial r\left(\overline{x}\right)}{\partial x_n}\right]$$

Iedere van deze afgeleiden kan worden bepaald aan de hand van de quotiënt regel;

$$\frac{\partial r\left(\overline{x}\right)}{\partial x_j} = \frac{\partial}{\partial x_j}\left(\frac{\overline{x}^T A\overline{x}}{\overline{x}^T\overline{x}}\right) = \frac{\frac{\partial}{\partial x_j}\left(\overline{x}^T A\overline{x}\right)\overline{x}^T\overline{x} - \overline{x}^T A\overline{x}\frac{\partial}{\partial x_j}\overline{x}^T\overline{x}}{\left(\overline{x}^T\overline{x}\right)^2}$$

Hier wordt $\frac{\partial}{\partial x_j}\left(\overline{x}^T A\overline{x}\right)$ gegeven door;

$$\frac{\partial}{\partial x_j}\left(\overline{x}^T A\overline{x}\right) = \frac{\partial}{\partial x_j}\left(\overline{x}^T\right)A\overline{x} + \overline{x}^T\frac{\partial}{\partial x_j}\left(A\overline{x}\right)$$

$$= [0\ \cdots\ 0\ 1\ 0\ \cdots\ 0]\,A\overline{x} + \overline{x}^T A\begin{bmatrix}0\\ \vdots\\ 0\\ 1\\ 0\\ \vdots\\ 0\end{bmatrix}$$

Slechts op de j-de positie staat het getal 1, de overige posities worden gegeven door het getal 0. Dit heeft als gevolg dat de hierboven gegeven vergelijking gereduceerd wordt tot een scalair probleem. Er geldt dan

$$\begin{aligned}\frac{\partial}{\partial x_j}\left(\overline{x}^T A\overline{x}\right) &= \frac{\partial}{\partial x_j}\left(\overline{x}^T\right)A\overline{x} + \overline{x}^T\frac{\partial}{\partial x_j}\left(A\overline{x}\right)\\ &= \frac{\partial}{\partial x_j}\left(\overline{x}^T\right)A\overline{x} + \overline{x}^T A\frac{\partial}{\partial x_j}\left(\overline{x}\right)\\ &= \frac{\partial}{\partial x_j}\left(\overline{x}^T\right)A\overline{x} + \left(\overline{x}^T A\frac{\partial}{\partial x_j}\overline{x}\right)^T\\ &= \frac{\partial}{\partial x_j}\left(\overline{x}^T\right)A\overline{x} + \frac{\partial}{\partial x_j}\overline{x}^T A^T\overline{x}\end{aligned}$$

Vanwege de symmetrie van A, dus $A^T = A$, volgt er;

$$\begin{aligned}\frac{\partial}{\partial x_j}\left(\overline{x}^T A\overline{x}\right) &= \frac{\partial}{\partial x_j}\left(\overline{x}^T\right)A\overline{x} + \frac{\partial}{\partial x_j}\overline{x}^T A\overline{x}\\ &= 2\frac{\partial}{\partial x_j}\left(\overline{x}\right)^T A\overline{x}\end{aligned}$$

Hieruit kan worden afgeleid;

$$\frac{\partial}{\partial x_j}\left(\overline{x}^T A\overline{x}\right) = 2\left(A\overline{x}\right)_j$$

Op een soortgelijke manier kan er worden beredeneerd dat

$$\frac{\partial}{\partial x_j}\left(\overline{x}^T\overline{x}\right) = [0\ \cdots\ 0\,1\,0\ \cdots\ 0]\,\overline{x} + \overline{x}^T \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = 2x_j$$

Als men deze twee partiële afgeleiden van x_j samenvoegd, wordt er verkregen;

$$\frac{\partial r\left(\overline{x}\right)}{\partial x_j} = \frac{2\left(A\overline{x}\right)_j}{\overline{x}^T\overline{x}} - \frac{\left(\overline{x}^T A\overline{x}\right)2x_j}{\left(\overline{x}^T\overline{x}\right)^2}$$

$$= \frac{2}{\overline{x}^T\overline{x}}\left(\left(A\overline{x}\right)_j - r\left(\overline{x}\right)x_j\right)$$

De gradient van $r\left(\overline{x}\right)$ wordt vervolgens gegeven door

$$\nabla r\left(\overline{x}\right) = \frac{2}{\overline{x}^T\overline{x}}\left(A\overline{x} - r\left(\overline{x}\right)\overline{x}\right)$$

Door de bovenstaande gradient gelijk te stellen aan 0, volgt er;

$$\nabla r\left(\overline{x}\right) = A\overline{x} - r\left(\overline{x}\right)\overline{x} = 0$$

Samengevat, het doel van de methode is het vinden van het nulpunt van de gradient $\nabla r\left(\overline{x}\right)$. Wanneer dit is gevonden, geldt er $r\left(\overline{x}\right) = \mu_n$ of $r\left(\overline{x}\right) = \mu_1$. Omdat $\overline{x}$ de eigenvector voorstelt die hoort bij de grootste eigenwaarde van A, vormt deze de PageRank vector.

Definieer nu voor het gemak de functie $F\left(\overline{x}\right) = A\overline{x} - r\left(\overline{x}\right)\overline{x}$. Men streeft om het nulpunt de vinden van $F\left(\overline{x}\right)$. Hierbij kunnen verschillende nulpuntsmethodes worden gebruikt. Degene die in dit onderzoek zal worden bekeken is de 'Derivative-Free Spectral Algorithm for Nonlinear Equations', afgekort de DF-SANE nulpuntsmethode.

6.2 De DF-SANE nulpuntsmethode

Men introduceert nu de DF-SANE nulpuntsmethode om de vergelijking

$$F\left(x\right) = 0$$

op te lossen waarbij $F : \mathbb{R}^n \to \mathbb{R}^n$ een continue differentieerbare vectorfunctie is. Deze methode is ontworpen door de heren La Cruz, Martínez en Raydan.

6.2.1 Het DF-SANE Algoritme

Men neemt aan dat de functie $F : \mathbb{R}^n \to \mathbb{R}^n$ continue partiële afgeleiden heeft. Er wordt gedefinieerd

$$f\left(\overline{x}\right) = ||F\left(x\right)||_2 \,\forall\, x \in \mathbb{R}^n$$

Nu worden meerdere aannamens over verschillende parameters gedaan om het algorimte goed te laten lopen.

- $\{\eta_k\}$ is een reeks waarbij $\eta_k > 0$ is voor alle $k \in \mathbb{N}$ en

$$\sum_{k=0}^{\infty} \eta_k = \eta < \infty$$

- De parameter γ wordt zo gekozen dat $0 < \gamma < 1$
- $\sigma_{\min}$ en $\sigma_{\max}$ worden zo gekozen dat $0 < \sigma_{\min} < \sigma_{\max} < \infty$
- M wordt zo gekozen dat $M \in \mathbb{N}_{>0}$
- $\tau_{\min}$ en $\tau_{\max}$ worden zo gekozen dat $0 < \tau_{\min} < \tau_{\max} < 1$

De parameters η_0, γ, $\sigma_{\min}$, $\sigma_{\max}$, $\tau_{\min}$, $\tau_{\max}$ en M worden vooraf zo gedefinieerd zodat ze voldoen aan de bovenstaande aannamens. De reeks η_k wordt in het algoritme verder bepaald.
Ten slotte wordt er een willekeurige startvector $\overline{x}_0 \in \mathbb{R}^n$ gekozen.

Nu kan het DF-SANE Algoritme geintroduceerd worden;

Het DF-SANE Algoritme

Stap 0

Zet $k \leftarrow 0$

Stap 1

Kies σ_k zo dat $|\sigma_k| \in [\sigma_{\min}, \sigma_{\max}]$
Bereken $\overline{f}_k = \max\{f(x_k), \cdots, f\left(x_{\max\{0,k-M+1\}}\right)\}$
Zet $d \leftarrow -\sigma_k F(x_k)$
Zet $\alpha_+ \leftarrow 1$, $\alpha_- \leftarrow 1$

Stap 2

If $f(x_k + \alpha_+ d) \leq \overline{f}_k + \eta_k - \gamma \alpha_+^2 f(x_k)$
Definieer $d_k = d$
Definieer $\alpha_k = \alpha_+$
Definieer $x_{k+1} = x_k + \alpha_k d_k$

Else if $f(x_k - \alpha_- d) \leq \overline{f}_k + \eta_k - \gamma \alpha_-^2 f(x_k)$
Definieer $d_k = -d$
Definieer $\alpha_k = \alpha_-$
Definieer $x_{k+1} = x_k + \alpha_k d_k$

Else
Kies $\alpha_{+,\text{nieuw}} \in [\tau_{\min}\alpha_+, \tau_{\max}\alpha_+]$
Kies $\alpha_{-,\text{nieuw}} \in [\tau_{\min}\alpha_-, \tau_{\max}\alpha_-]$
Vervang $\alpha_+ \leftarrow \alpha_{+,\text{nieuw}}$
Vervang $\alpha_- \leftarrow \alpha_{-,\text{nieuw}}$
Herhaal Stap 2

Stap 3

If $F(x_{k+1}) = 0$
quit

Else
Zet $k \leftarrow k + 1$
Ga naar Stap 1

Men noemt de parameters σ_k de spectrale coëfficiënten.

Wanneer de code aandachtig wordt bekeken, kan men zien dat hij is opgebouwd uit verschillende iteraties k. Bovendien vinden in stap 2 aparte iteraties plaats. Deze zullen worden aangeduid als de binnen-iteraties.

Nu zal het algemene idee van dit iteratieve algoritme worden uitgelegd.

Merk op dat wanneer de waarde van $F(\overline{x}_k)$ groot is, men zich ver van het nulpunt zal bevinden. Een kleine waarde voor de lengte van $F(\overline{x}_k)$ betekent echter dat de afstand tot het nulpunt klein is. Het doel van het algoritme is om een vector $\overline{x}$ zo te benaderen zodat er geldt

$$F(\overline{x}) = 0$$

De afstand tussen het nulpunt en de vector $\overline{x}$ is dan verwaarloosbaar klein.

Men begint door als startvector een willekeurige vector $\overline{x}_0$ te kiezen. Vervolgens zullen een of meerdere iteraties plaats vinden. Het eerste schattingspunt bij iedere iteratie wordt volgens dit algoritme gegeven door $\overline{x}_k - \sigma_k F(\overline{x}_k)$ waarbij k de k-de iteratie weergeeft.

Er wordt gestreefd om de afstand bij opeenvolgende iteraties kleiner te maken totdat deze gelijk aan 0 wordt. Men zal nu de eerste iteratie in beschouwing nemen om het algoritme te illustreren.

Wanneer er naar de eerste iteratie wordt gekeken, wil men

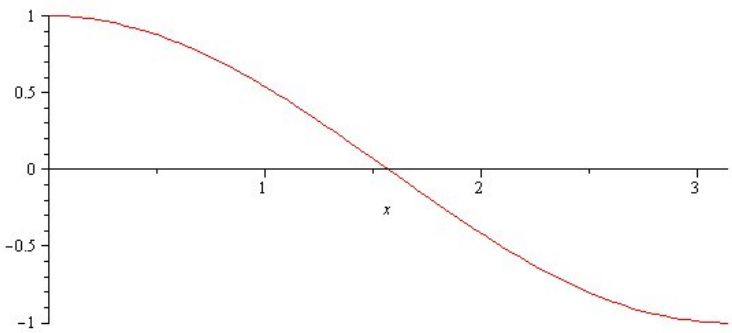
$$||F(\overline{x}_1)||_2 = ||F(\overline{x}_0 - \sigma_0 F(\overline{x}_0))||_2 < ||F(\overline{x}_0)||_2$$

De vraagt die rijst is dan hoe groot de spectrale coëfficiënt σ_0 gekozen moet worden. Deze kan zodanig worden aangepast door hem te vermenigvuldigen met de factor α.

In het algoritme wordt er onderscheid gemaakt of men zich links of rechts van het nulpunt bevindt. Dit komt terug in stap 2. Het ligt aan het teken van $F(x)$ welke van de drie gevallen van stap 2 van toepassing is.

6.2.2 Simpel Voorbeeld

In deze deelparagraaf zal aan de hand van een simpel voorbeeld het idee achter het DF-SANE algoritme geillustreerd worden.
Beschouw de onderstaande grafiek.



Figuur 6: Grafiek van $F(x) = \cos(x)$ met $x \in [0, \pi]$

Deze stelt de cosinus functie voor die loopt van $x = 0$ tot aan $x = \pi$. Men wil nu aan de hand van het DF-SANE algoritme het nulpunt bepalen. In andere woorden, men is geinteresseerd naar de waarde van x waarvoor geldt

$$F(x) = \cos(x) = 0$$

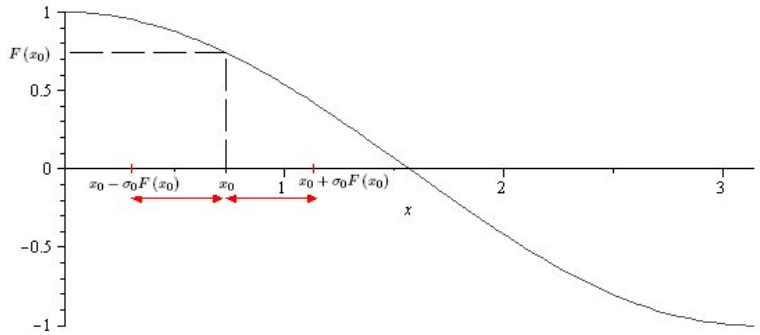
De situatie die hier beschouwd wordt, wordt versimpeld vanwege het feit dat men met slechts 1 variabele werkt.

Het algoritme zegt dat er een willekeurige waarde voor x_0 gekozen wordt. Deze wordt in dit geval aan de linkerkant van het nulpunt genomen. Men evalueert nu de functie in het punt x_0, dus $F(x_0)$ wordt bepaald. Vervolgens wordt de lengte $||F(x_0)||$ berekend.

Nu wordt de waarde van $F(x_0)$ vermenigvuldigd met de spectrale coëfficiënt σ_0. Deze is zo gekozen dat er geldt $\sigma_0 \in [\sigma_{\min}, \sigma_{\max}]$. Er worden vervolgens twee nieuwe punten beschouwd, namelijk

- het punt dat een afstand $-\sigma_0 F(x_0)$ heeft ten opzichte van x_0. Dit is het punt $x_0 - \sigma_0 F(x_0)$.
- het punt dat een afstand $+\sigma_0 F(x_0)$ heeft ten opzichte van x_0. Dit is het punt $x_0 + \sigma_0 F(x_0)$.

Een visuele voorstelling van deze situatie is te zien in de onderstaande figuur.



Figuur 7: Visuele voorstelling van de afstanden $x_0 + \sigma_0 F(x_0)$ en $x_0 - \sigma_0 F(x_0)$

Vervolgens wordt de functie geevalueerd in deze twee nieuwe punten, $F(x_0 + \sigma_0 F(x_0))$ en $F(x_0 - \sigma_0 F(x_0))$ worden dus bepaald. De bijbehorende lengtes $||F(x_0 + \sigma_0 F(x_0))||$ en $||F(x_0 - \sigma_0 F(x_0))||$ worden dan berekend. Merk op dat in dit voorbeeld geldt

$$||F(x_0 + \sigma_0 F(x_0))|| < ||F(x_0)|| < ||F(x_0 - \sigma_0 F(x_0))||$$

Omdat in deze situatie $F(x_0 + \sigma_0 F(x_0))$ de kleinste lengte heeft van de drie bepaalde punten, zal dit punt gekozen worden als startpunt van een nieuwe iteratie. Men zet dus $x_1 = F(x_0 + \sigma_0 F(x_0))$.

Door dit proces te herhalen en meerdere soortgelijke iteraties uit te voeren, zal op den duur het nulpunt worden gevonden.

Het algoritme zoals hierboven beschreven bevat echter nog een valkuil. Men zal dit laten zien aan de hand van de iteratie met als startpunt $x_1 = F(x_0 + \sigma_0 F(x_0))$.

Zoals bij iteratie 0 wil men twee nieuwe punten bepalen. Bij iteratie 1 zijn dat $x_1 - \sigma_1 F(x_1)$ en $x_1 + \sigma_1 F(x_1)$. σ_1 wordt zo gekozen dat er wordt voldaan aan $\sigma_1 \in [\sigma_{\min}, \sigma_{\max}]$. Het zou kunnen voorkomen dat de gekozen waarde van σ_1 ervoor zorgt dat

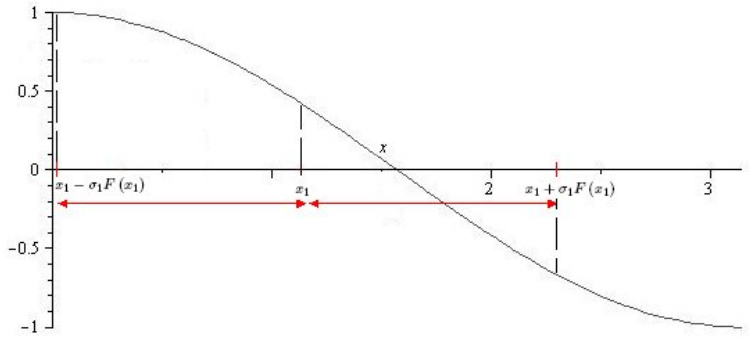
$$||F(x_1)|| < ||F(x_1 - \sigma_1 F(x_1))|| < ||F(x_1 + \sigma_1 F(x_1))||$$

of

$$||F(x_1)|| < ||F(x_1 + \sigma_1 F(x_1))|| < ||F(x_1 - \sigma_1 F(x_1))||$$

In dit geval heeft blijft $||F(x_1)||$ de kleinste lengte hebben. Als gevolg hiervan wordt niet een van de nieuwe punten $F(x_1 - \sigma_1 F(x_1))$ of $F(x_1 + \sigma_1 F(x_1))$ gekozen als startpunt van iteratie 2.

Een visuele voorstelling van deze situatie wordt in de onderstaande Figuur weergegeven.



Figuur 8: Visuele voorstelling van de afstanden $x_1 + \sigma_1 F(x_1)$ en $x_1 - \sigma_1 F(x_1)$

Hier geldt er dat

$$||F(x_1)|| < ||F(x_1 + \sigma_1 F(x_1))|| < ||F(x_1 - \sigma_1 F(x_1))||$$

Men streeft echter om de lengte van $F(x_1 - \sigma_1 F(x_1))$ of $F(x_1 + \sigma_1 F(x_1))$ kleiner te maken dan de lengte van $F(x_1)$. Hiervoor zal de waarde van σ_1 aangepast moeten worden. Dit wordt gedaan door de spectrale coëfficiënt te vermenigvuldigen met een factor α. Wanneer men te maken heeft met het nieuwe punt rechts ten opzichte van het referentiepunt x_1, dan zal σ_1 vermenigvuldigd worden met α_+ waarbij $\alpha_+ \in [\tau_{\min}\alpha_+, \tau_{\max}\alpha_+]$. Bij het punt links van x_1 zal de vermenigvuldiging van σ_1 plaatsvinden met α_- waarbij $\alpha_- \in [\tau_{\min}\alpha_-, \tau_{\max}\alpha_-]$. Vervolgens worden er twee nieuwe punten beschouwd, namelijk

- het punt dat een afstand $-\alpha_-\sigma_1 F(x_1)$ heeft ten opzichte van x_1. Dit is het punt $x_1 - \alpha_-\sigma_1 F(x_1)$.
- het punt dat een afstand $\alpha_+\sigma_1 F(x_1)$ heeft ten opzichte van x_0. Dit is het punt $x_1 + \alpha_+\sigma_1 F(x_1)$.

Vervolgens wordt de functie geëvalueerd in deze twee nieuwe punten. Dit houdt in dat $F(x_1 - \alpha_-\sigma_1 F(x_1))$ en $F(x_1 + \alpha_+\sigma_1 F(x_1))$ berekend worden. De bijbehorende lengtes $||F(x_1 - \alpha_-\sigma_1 F(x_1))||$ en $||F(x_1 + \alpha_+\sigma_1 F(x_1))||$ worden dan vastgesteld. Hierna zal men kijken welke van de drie punten de kleinste lengte heeft.

Wanneer $||F(x_1 - \alpha_-\sigma_1 F(x_1))||$ (of $||F(x_1 + \alpha_+\sigma_1 F(x_1))||$) de kleinste lengte geeft, dan zal het punt $x_1 - \alpha_-\sigma_1 F(x_1)$ (of $x_1 + \alpha_+\sigma_1 F(x_1)$) als startpunt van een nieuwe iteratie functioneren. Men zet dan $x_2 = x_1 - \alpha_-\sigma_1 F(x_1)$ (of $x_2 = x_1 + \alpha_+\sigma_1 F(x_1)$).

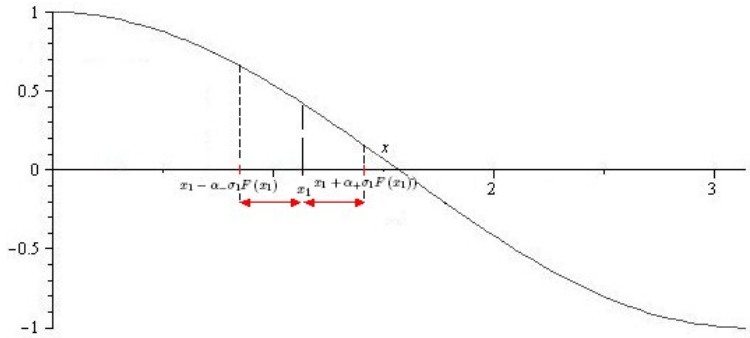
Wanneer de kleinste lengte opnieuw gegeven wordt door $||F(x_1)||$, dan wordt de laatste stap herhaald. Er zullen nieuwe waarden voor α_- en α_+ gekozen moeten worden. Definieer deze als $\alpha_{-,\text{nieuw}}$ en $\alpha_{+,\text{nieuw}}$. Vervolgens wordt er gekeken naar twee nieuwe punten die een afstand $-\alpha_{-,\text{nieuw}}\alpha_-\sigma_1$ en $+\alpha_{+,\text{nieuw}}\alpha_+\sigma_1$ hebben ten opzichte van x_1. $||F(x_1 - \alpha_{-,\text{nieuw}}\alpha_-\sigma_1 F(x_1))||$ en $||F(x_1 + \alpha_{+,\text{nieuw}}\alpha_+\sigma_1 F(x_1))||$ worden bepaald en men zal nagaan welk van de drie punten $x_1 - \alpha_{-,\text{nieuw}}\alpha_-\sigma_1 F(x_1)$, $x_1 + \alpha_{+,\text{nieuw}}\alpha_+\sigma_1 F(x_1)$ of x_1 de kleinste lengte heeft.

Men kiest nieuwe waarden voor α_- en α_+ en vermenigvuldigd deze zoals hierboven totdat de kleinste lengte niet meer gegeven wordt door $||F(x_1)||$. Wanneer dit het geval is zal men overgaan naar een nieuwe iteratie met als startpunt die x-coördinaat die de kleinste lengte levert.

De werking van α zal geillustreerd worden aan de hand van het behandelde voorbeeld. Men had gezien dat

$$||F(x_1)|| < ||F(x_1 + \sigma_1 F(x_1))|| < ||F(x_1 - \sigma_1 F(x_1))||$$

Omdat de kleinste lengte gegeven wordt door $||F(x_1)||$ kiest men bepaalde waarden voor α_- en α_+. Door de hierboven beschreven vermenigvuldiging toe te passen, worden nu twee nieuwe punten gevormd. Deze zijn te zien in de onderstaande Figuur.



Figuur 9: Visuele voorstelling van de afstanden $x_1 + \alpha_+\sigma_1 F(x_1)$ en $x_1 - \alpha_-\sigma_1 F(x_1)$

Nu geldt er

$$||F(x_1 + \alpha_+\sigma_1 F(x_1))|| < ||F(x_1)|| < ||F(x_1 - \alpha_-\sigma_1 F(x_1))||$$

$||F(x_1 + \alpha_+\sigma_1 F(x_1))||$ stelt nu de kleinste lengte voor. Men zet daarom $x_2 = x_1 + \alpha_+\sigma_1 F(x_1)$. Dit punt functioneert als startpunt voor iteratie 2.

Door bij iedere iteratie na te gaan welk 'nieuw' punt (eventueel aangepast door vermenigvuldiging met α) de kleinste lengte geeft en deze te kiezen als startpunt voor de volgende iteratie, zal er op den duur het nulpunt gevonden worden.

6.2.3 Geimplementeerd DF-SANE Algoritme in Matlab

De heer Raydan heeft deze nulpuntsmethode onderzocht en geimplementeerd in Matlab. Voor de code, zie achterin in 'Bijlagen Matlab Codes'. Hij heeft een keuze gemaakt van de waarden van de te vooraf te definiëren parameters. Deze waarden zullen ook in dit onderzoek aangehouden worden en zijn hieronder gerapporteerd;

- $\sigma_{\min} = 1.0 \cdot 10^{-5}$
- $\sigma_{\max} = 1.0$
- $\tau_{\min} = 0.1$
- $\tau_{\max} = 0.5$
- $\gamma = 1.0 \cdot 10^{-4}$
- $M = 10$
- $\eta_0 = ||F(x_0)||_2$

De resterende elementen van de reeks η_k bij de k-de iteratie worden bepaald aan de hand van

$$\eta_k = \frac{||F(x_0)||_2}{(1+k)^2}$$

De geimplementeerde Matlab code is geschreven als een Function File. Er worden verschillende input parameters vereist. Deze zijn;

- x0; Deze parameter stelt de willekeurige startvector $\overline{x}_0$ die gekozen is voor het algoritme.
- Fun; Dit is de functie waarvan het nulpunt bepaald moet worden. Men wil dus een vector $\overline{x}$ zo bepalen zodat er wordt voldaan aan
$$\text{Fun}(\overline{x}) = 0$$
- nmax; Dit is het maximum aantal iteraties die men wil uitvoeren. Deze parameter is ingevoerd zodat de code niet oneindig lang kan runnen en het aantal iteraties oneindig groot wordt.

- ea; Deze parameter stelt de absolute fout voor. Gekozen is om deze waarde gelijk te stellen aan $1.0 \cdot 10^{-4}$.
- er; Deze parameter stelt de relatieve fout voor. Gekozen is om deze waarde gelijk te stellen aan $1.0 \cdot 10^{-4}$.

De code geeft eveneens verschillende output parameters. Deze zullen hieronder worden opgesomt;

- xk; Deze parameter stelt de oplossingsvector $\overline{x}_k$ die het nulpunt geeft, dus waarvoor geldt

$$F\left(\overline{x}_k\right) = 0$$

- IT; Het aantal iteraties weer die de code nodig heeft om tot een oplossingsvector $\overline{x}_k$ te komen wordt door deze parameter voorgesteld.
- EF; Deze parameter stelt het totaal aantal functie-evaluaties weer.
- BL; Deze parameter geeft het aantal keer weer dat de hoeveelheid binneniteraties in een iteratie groter of gelijk aan 1 is geweest.
- t; Deze parameter geeft de rekentijd die de code over doet om tot $\overline{x}_k$ te komen.
- fa; In de code is een fout controle parameter ingevoerd. Deze kan twee verschillende waarden aannemen, 0 en 2. Wanneer fa gelijk is aan de waarde 0 bij het beëindigen van de code, dan betekent dat dat er geen fouten zijn opgetreden en de oplossing $\overline{x}_k$ als een betrouwbare oplossing kan worden gezien. Bij fa = 2 geldt dat het maximum aantal binnen-iteraties is overschreden. Meneer Raydan heeft voor dit maximum de waarde 100 gekozen.
- norma; Dit is een parameter in de vorm van een vector waarbij het i-de element de norm $||F\left(\overline{x}_i\right)||_2$ bij de i-de iteratie weergeeft.

6.3 Rayleigh Quotiënt Methode toegepast op Testmatrices

6.3.1 Experimenten

Men verwacht dat er na toepassing van de Rayleigh Quotiënt Methode een eigenvector wordt verkregen behorend bij de grootste of kleinste eigenwaarde van de bekeken matrix. Dit zou moeten gelden ongeacht welke startvector x_0 er gekozen wordt. Als de methode op een juiste manier in Matlab geimplementeerd is, geldt er voor deze eigenvector $\overline{v}$;

$$\nabla r\left(\overline{v}\right) = F\left(\overline{v}\right) = A\overline{v} - r\left(\overline{v}\right)\overline{v} = 0$$

waar $r\left(\overline{v}\right)$ het Raileigh quotiënt voorstelt.
Er wordt echter gestreefd om de eigenvector te vinden behorend bij de grootste eigenwaarde.

Men zal nu een aantal experimenten uitvoeren om na te gaan of de Rayleigh Quotiënt een geschikte methode is om de eigenvector te verkrijgen. Hierbij zullen testmatrices van verschillende dimensies worden beschouwd. De dimensie zal gevarieerd worden van 5 tot 2000.

Uit de theorie volgt dat de bekeken matrix A symmetrisch en positief (semi)definiet moet zijn om

de DF-SANE methode goed te laten verlopen. Er is daarom gekozen voor testmatrices in de vorm van

$$A = \begin{bmatrix} 2 & -1 & 0 & & \cdots & & \cdots & & 0 \\ -1 & 2 & -1 & 0 & & & & & \vdots \\ 0 & -1 & 2 & -1 & 0 & & & & \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & & & \\ & & & & & & & & \\ & & & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ & & & & 0 & -1 & 2 & -1 & 0 \\ \vdots & & & & & 0 & -1 & 2 & -1 \\ 0 & & \cdots & & \cdots & & 0 & -1 & 2 \end{bmatrix}$$

De startvector x_0 wordt gevarieerd om na te gaan of de methode werkt in verschillende situaties en of hij altijd convergeert naar de eigenvector behorend bij de grootste eigenwaarde. De waarden van x_0 worden random gekozen door de Matlab random generator te gebruiken met als seed 'state, 100'.

Na het uitvoeren van de Rayleigh Quotiënt Methode moet men nagaan of de verkregen eigenvector ook daadwerkelijk correspondeert met de grootste eigenwaarde. Men bepaalt hiervoor de eigenwaarde behorend bij de verkregen eigenvector aan de hand van

$$\frac{||A\overline{v}||}{||\overline{v}||} = \lambda$$

waarbij $\overline{v}$ de verkregen eigenvector voorstelt.
Wanneer λ dezelfde waarde heeft als de grootste eigenwaarde van A, dan kan men concluderen dat $\overline{v}$ de PageRank vector voorstelt. De grootste eigenwaarde van A kan in Matlab worden bepaald aan de hand van het commando `eig(A)`.

Men constateert bij alle matrixdimensies dat de Rayleigh Quotiënt Methode niet altijd convergeert naar de gewenste eigenvector. Het ligt aan de keuze van de startvector x_0 of de eigenvector behorende bij de grootste of kleinste eigenwaarde wordt gevonden. Omdat men streeft om $\overline{v}$ te bepalen corresponderend bij de grootste eigenwaarde, is er besloten om eerst een paar iteraties Power Methode uit te voeren. De output vector van de Power Methode zal vervolgens als startvector x_0 dienen voor de Rayleigh Quotiënt Methode.

Men maakt gebruik van de Power Methode omdat dit een snel convergerende methode is en het al na een aantal iteraties een goede indicatie geeft van de gewenste eigenvector. Er moet echter wel voorkomen worden dat de Power Methode al naar de oplossing convergeert aangezien de Rayleigh Quotiënt Methode dan geen invloed meer zal hebben in het vinden van het PageRank vector. Met grote waarschijnlijkheid zal de Rayleigh Quotiënt Methode met deze input de eigenvector geven die hoort bij de grootste eigenwaarde. Wanneer dit niet zo is, moet men het aantal iteraties Power Methode opgehogen. Verder zal het aan de matrixdimensie afhangen hoeveel iteraties Power Methode er ongeveer uitgevoerd zullen moeten worden.

De Arnoldi Methode wordt hier buiten beschouwing gelaten omdat in eerdere experimenten geconstateerd was dat deze aanzienlijk minder snel was en meer iteraties nodig had vergeleken de Power Methode.

6.3.2 Resultaten van Experimenten

Nu zal er nagegaan worden of het gebruik van het Rayleigh Quotiënt Methode na de Power Methode efficiënt is. Hierbij zal er worden gekeken naar het aantal iteraties en de totale rekentijd. Deze laatstgenoemde wordt gegeven door de tijd die de Power Methode erover doet om het aantal gegeven

iteraties uit te voeren en de duur tot het convergeren van de Rayleigh Quotiënt Methode.

Vervolgens zullen dezelfde testmatrices worden gebruikt bij het uitvoeren van enkel de Power Methode. Opnieuw zullen het aantal iteraties en de tijdsduur tot convergeren worden bijgehouden.

De resultaten verkregen van de twee bovenstaande beschreven experimenten zullen met elkaar vergeleken worden. Aan de hand hiervan zal er geconcludeerd kunnen worden of het geschikt is om de Rayleigh Quotiënt Methode in acht te nemen na het gebruik van de Power Methode. Gekozen is om bij beide experimenten de fout ϵ van de Power Methode gelijk te nemen aan 10^{-8}. Om eerlijk te kunnen vergelijken, wordt de startvector van de Power Methode bij eenzelfde matrixdimensies in beide experimenten gelijk genomen. Deze vector is random gekozen met de Matlab random generator met seed 'state,100'.

De vergelijking met de Arnoldi Methode wordt hier buiten beschouwing gelaten aangezien men al eerder geconcludeerd had dat de Power Methode meer geschikt zou zijn voor het vinden van de PageRank vector wanneer er gekeken wordt naar de duur en het aantal iteraties.

De resultaten zijn weergegeven in Tabellen 14 en 15. Het aantal iteraties is afgekort met it., de Power Methode met PM en de Rayleigh Quotiënt Methode met RQ.

Matrix Dimensie	Duur	Aantal Iteraties
5	0.00327 sec	44 it.
30	0.02125 sec	675 it.
50	0.03382 sec	1005 it.
100	0.07352 sec	4784 it.
150	0.38977 sec	17357 it.
200	0.35997 sec	9676 it.
1000	18.76146 sec	9548 it.
2000	65.03704 sec	8099 it.

Tabel 14: Resultaten Power Methode met Testmatrices

Matrix Dimensie	Duur PM	It. PM	Duur RQ	It. RQ	Totale Duur	Totaal It.
5	0.00008 sec	2 it.	0.03000 sec	17 it.	0.03008 sec	19 it.
30	0.00432 sec	2 it.	0.14000 sec	57 it.	0.14432 sec	59 it.
50	0.00277 sec	2 it.	0.14000 sec	74 it.	0.14277 sec	76 it.
100	0.00330 sec	2 it.	0.73100 sec	610 it.	0.7343 sec	612 it.
150	0.00296 sec	2 it.	0.37100 sec	131 it.	0.37396 sec	133 it.
200	0.00032 sec	2 it.	0.24000 sec	76 it.	0.24032 sec	78 it.
1000	0.06410 sec	10 it.	5.30700 sec	106 it.	5.3711 sec	116 it.
2000	1.64301 sec	200 it.	0.47000 sec	6 it.	2.11301 sec	206 it.

Tabel 15: Resultaten Power Methode en Rayleigh Quotiënt Methode met Testmatrices

Zoals hierboven gezegd, streeft men om het aantal iteraties Power Methode zo laag mogelijk te houden op voorwaarde dat de methode een geschikte vector geeft die als input vector zal dienen voor de Rayleigh Quotiënt Methode. Bij matrixdimensies tot 200 heeft men ervoor gekozen om 2 iteraties Power Methode uit te voeren. Dit was voldoende om een startvector vast te stellen die een geschikte input zou zijn voor de Rayleigh Quotiënt Methode. Bij matrixdimensie 1000 waren 10 iteraties nodig en bij een dimensie van 2000, 200 iteraties. Dit hoge aantal iteraties bij de laatstgenoemde dimensie kan worden verklaard aan de hand van de random gekozen startvector voor de Power Methode. Deze zal hoogst waarschijnlijk ver liggen van de gewilde eigenvector waardoor meer iteraties Power Methode uitgevoerd zullen moeten worden om een geschikte input vector te bepalen voor de Rayleigh Quotiënt Methode.

Aan de hand van de bovenstaande resultaten kan men afleiden dat voor kleine matrixdimensies tot 100 de Power Methode het meest geschikt is wanneer men geinteresseerd is naar de methode met de kortste duur. In dit experiment bij deze matrixdimensies ligt het aantal iteraties echter wel hoger dan bij het experiment met zowel de Power Methode als de Rayleigh Quotiënt Methode. Wanneer men een zo laag mogelijk aantal iteraties wil uitvoeren, zal zijn voorkeur uitgaan naar deze laatst genoemde methode.

Bij matrixdimensies vanaf 150 is de situatie anders. Zowel de totale tijdsduur als het aantal iteraties liggen aanzienlijk lager bij de Power Methode gecombineerd met de Rayleigh Quotiënt Methode. Men zal nu zijn voorkeur uitspreken voor deze methode.

6.4 Rayleigh Quotiënt Methode toegepast op Google Matrix

Men heeft tot nu toe de Rayleigh Quotiënt Methode toegepast op testmatrices die symmetrisch, positief defiet waren genomen. Dit is gedaan om een idee te krijgen of de toepassing van de DF-SANE Methode na de Power Methode een positief effect zou hebben op het aantal iteraties en de tijdsduur van convergeren vergeleken met het gebruik van slechts de Power Methode. Aan de hand van de twee experimenten uit de vorige paragraaf heeft men geconcludeerd dat het geschikt is om vanaf een bepaalde matrixdimensie de Rayleigh Quotiënt Methode te gebruiken.

Omdat er hier gewerkt wordt met SPD matrices, kan men niet met volledige zekerheid zeggen dat de Rayleigh Quotiënt Methode ook in het kader van Google een positieve invloed zal hebben op het aantal iteraties en de tijdsduur. Dit omdat de Google matrix G geen SPD matrix is. Om hardere conclusies te kunnen trekken moet men de SPD matrices vervangen door de Google matrix G die een voorstelling geeft van hoe men zich door het web beweegt. Er zullen twee gevallen onderscheiden worden, de situatie waarbij de Google matrix met behulp van de random generator is gevormd en die waar de Google matrix met surfer.m is bepaald.

6.4.1 Symmetrische, Positief (semi)Definiete Aanpassing

Herinner dat de Rayleigh Quotiënt Methode een symmetrische, positief (semi)definiete matrix vereist. Omdat Google matrix G dat niet is, moet hij zodanig aangepast worden dat hij dat wordt. Om een symmetrische matrix te verkrijgen, wordt matrix F gedefinieerd als

$$F = G + G^T$$

Men zal nu kijken of F een positief (semi)definiete matrix is.

Een voorwaarde voor een positief (semi)definiete Hermitische matrix A is dat de determinanten van A positief moeten zijn, dus $\det(A) > 0$. Dit geldt wanneer alle eigenwaarden van A groter of gelijk zijn aan 0.

Omdat een reële Hermitische matrix een symmetrische matrix is, is het bovenstaande van toepassing op matrix F. Om na te gaan of F positief (semi)definiet is, kan er worden gekeken naar twee verschillende instanties;

- Alle determinanten van F zijn positief of gelijk aan 0.
- Alle eigenwaarden van F zijn groter of gelijk aan 0.

Deze twee uitspraken zijn equivalen.

Men constateert dat matrix F nog negatieve eigenwaarden bevat. Volgens de tweede uitspraak is

F dus geen positief (semi)definiete matrix. F moet nu zodanig aangepast worden zodat het een positief (semi)definiete matrix wordt. Deze verkregen SP(semi)D matrix zal men F_{nieuw} noemen. Men streeft om F_{nieuw} zo 'dicht' mogelijk bij F te laten liggen.

Beschouw nu de volgende stelling

Stelling 4. *Zij A een $n \times n$ diagonaliseerbare matrix met n lineair onafhankelijke eigenvector $\overline{q}_i$ waarbij $i = 1, \cdots, n$. A kan dan gefactoriseerd worden in de vorm van*

$$A = QDQ^{-1}$$

waar Q een $n \times n$ matrix is wiens i-de kolom wordt gegeven door $\overline{q}_i$ en D een diagonaal matrix is wiens diagonaal elementen gegeven worden door de corresponderende eigenwaarden λ_i.

Herinner dat elke symmetrische matrix diagonaliseerbaar is. Hieruit volgt dat de bovenstaande stelling van toepassing is op matrix F. Men kan F dan omschrijven tot

$$F = QDQ^{-1}$$

waarbij Q de matrix is bestaande uit de eigenvectoren van F en D de diagonaalmatrix voorstelt met de corresponderende eigenwaarden op de diagonaal.

Omdat F negatieve eigenwaarden kan bevatten, zullen die elementen op de diagonaal van D negatief zijn. Men definieert nu een aan D identieke matrix D^* met uitzondering van de negatieve eigenwaarden. Deze eigenwaarden worden vervangen door relatief kleine positieve eigenwaarden. Men krijgt nu

$$F_{\text{nieuw}} = QD^*Q^{-1}$$

Deze verkregen matrix heeft slechts gelijk aan 0 en/of positieve eigenwaarden en is daarom een symmetrische, positief (semi)definiete matrix.

Men heeft geprobeerd om F_{nieuw} zo 'dicht' mogelijk bij F te houden. Hiermee wordt er bedoeld dat alle positieve en gelijk aan 0 eigenwaarden hetzelfde zijn gehouden.

6.4.2 Experimenten

Men gaat nu dezelfde experimenten uitvoeren als de experimenten met de testmatrices uit de vorige paragraaf. Zoals eerder verwacht men dat de Rayleigh Quotiënt Methode een eigenvector zal opleveren behorend bij de grootste of kleinste eigenwaarde. Men streeft echter weer om de eigenvector bij de grootste eigenwaarde te bepalen.

In de situatie waarbij de Google matrix G in beschouwing wordt genomen, constateert men dat de Rayleigh Quotiënt Methode niet altijd naar de gewenste eigenvector convergeert. In sommige gevallen convergeert de methode naar de eigenvector behorend bij de kleinste eigenwaarde. Men besluit daarom ook hier om vooraf een paar iteraties Power Methode uit te voeren met als matrix F_{nieuw}. De output vector van de Power Methode zal als x_0 dienen voor de Rayleigh Quotiënt Methode. Men heeft de Power Methode gekozen boven de Arnoldi Methode om dezelfde redenen als eerder.

De dimensies van de matrices gevormd aan de hand van de random generator en surfer.m worden opnieuw gevarieerd. Bij iedere situatie zal de tijdsduur tot convergeren en het aantal iteraties bijgehouden worden. De fout ϵ bij beide experimenten is weer gelijk gesteld aan 10^{-8}.

Vervolgens zal dezelfde F_{nieuw} worden gebruikt bij het uitvoeren van enkel de Power methode. Men is opnieuw geinteresseerd naar de tijdsduur tot convergeren en het aantal iteraties.

De startvector van de Power Methode bij eenzelfde matrixdimensies is in beide experimenten weer gelijk genomen. Dit is gedaan om eerlijk te kunnen vergelijken. Deze vector is random gekozen met de Matlab random generator met seed 'state,100'.

6.4.3 Resultaten van Experimenten

Eerst zal men de situatie bekijken waarbij de Google matrix G aan de hand van de random generator wordt gevormd. Men constateert bij iedere dimensie dat de verkregen eigenvector na toepassing van de Power Methode en de Rayleigh Quotiënt Methode behoort bij de grootste eigenwaarde van F_{nieuw}. De matrixdimensies worden opnieuw gevarieerd van 5 tot 2000.

De resultaten zijn weergegeven in Tabellen 16 en 17. Het aantal iteraties is afgekort met it., de Power Methode met PM. en de Rayleigh Quotiënt Methode met RQ.

Matrix Dimensie	Duur	Aantal Iteraties
5	0.00136 sec	7 it.
30	0.00145 sec	11 it.
50	0.00138 sec	11 it.
100	0.00020 sec	8 it.
150	0.00165 sec	8 it.
200	0.00167 sec	7 it.
1000	0.01475 sec	5 it.
2000	0.03781 sec	5 it.

Tabel 16: Resultaten Power Methode bij gebruik van random generator

Matrix Dimensie	Duur PM	It. PM	Duur RQ	It. RQ	Totale Duur	Totaal It.
5	0.00134 sec	2 it.	0.04000 sec	3 it.	0.04134 sec	5 it.
30	0.00133 sec	2 it.	0.05000 sec	4 it.	0.05133 sec	6 it.
50	0.00160 sec	2 it.	0.04000 sec	4 it.	0.0416 sec	6 it.
100	0.00005 sec	2 it.	0.02000 sec	3 it.	0.0205 sec	5 it.
150	0.00148 sec	2 it.	0.0610 sec	3 it.	0.06248 sec	5 it.
200	0.00012 sec	2 it.	0.04000 sec	3 it.	0.04012 sec	5 it.
1000	0.00385 sec	1 it.	0.44000 sec	3 it.	0.44385 sec	4 it.
2000	0.00803 sec	1 it.	0.66100 sec	2 it.	0.66903 sec	3 it.

Tabel 17: Resultaten Power Methode en Rayleigh Quotiënt Methode bij gebruik van random generator

Men zal zijn voorkeur aan de Power Methode geven wanneer hij geinteresseerd is naar de methode met de zo kort mogelijke tijdsduur tot convergeren. Deze is bij de laatstgenoemde methode aanzienlijk minder dan wanneer de combinatie van de Power Methode met de Rayleigh Quotiënt Methode wordt beschouwd. Dit geldt voor iedere matrixdimensie in tegenstelling tot bij de experimenten met de SPD testmatrices.

Wanneer er echter naar het aantal iteraties wordt gekeken, dan komt de combinatie van de Power Methode met de Rayleigh Quotiënt Methode als meest geschikte methode uit het experiment. Wat opvalt is dat het verschil in iteraties bij eenzelfde matrixdimensies in deze situatie klein is in tegenstelling tot het verschil in iteraties bij het experiment met de testmatrices.

Hieruit kan er geconcludeerd worden dat de methode die men zal kiezen afhangt van de gewensde eigenschap. Wanneer men streeft naar een zo kort mogelijke convergeertijd, dan zal hij kiezen voor enkel de Power Methode. Op het moment dat hij geinteresseerd is om het aantal iteraties zo laag

mogelijk te houden, zal zijn voorkeur uitgaan naar de combinatie van de twee methoden.

De bovenstaande conclusie geldt voor het geval waarbij de Google matrix G gevormd is aan de hand van de random generator. Nu zal de situatie worden beschouwd waarbij er gebruik wordt gemaakt van surfer.m om G te bepalen. Weer geldt er dat het gebruik van slechts de Rayleigh Quotiënt Methode in de meeste gevallen de eigenvector oplevert behorende bij de kleinste eigenwaarde. Er zullen daarom ook hier een paar iteraties Power Methode vooraf plaats vinden. Opnieuw zullen de resultaten van enkel de Power Methode worden vergeleken met die van de gecombineerde Power Methode met Rayleigh Quotiënt Methode. Om te voorkomen dat het programma een extreem lange rekentijd nodig heeft of vastloopt, zullen de matrixdimensies gevarieerd worden van 5 tot 150. Als beginpagina is er gekozen voor www.lyricsfreak.com. De resultaten staan weergegeven in Tabellen 18 en 19.

Matrix Dimensie	Duur	Aantal Iteraties
5	0.00126 sec	3 it.
30	0.00009 sec	4 it.
50	0.00141 sec	11 it.
100	0.00172 sec	10 it.
150	0.00186 sec	10 it.

Tabel 18: Resultaten Power Methode bij gebruik van surfer.m

Matrix Dimensie	Duur PM	It. PM	Duur RQ	It. RQ	Totale Duur	Totaal It.
5	0.00140 sec	1 it.	0 sec	0 it.	0.00140 sec	1 it.
30	0.00003 sec	1 it.	0.03000 sec	2 it.	0.03003 sec	3 it.
50	0.00138 sec	1 it.	0.04000 sec	5 it.	0.04138 sec	6 it.
100	0.00136 sec	1 it.	0.05000 sec	5 it.	0.05136 sec	6 it.
150	0.00163 sec	1 it.	0.03000 sec	5 it.	0.03163 sec	6 it.

Tabel 19: Resultaten Power Methode en Rayleigh Quotiënt Methode bij gebruik van surfer.m

Bij matrixdimensie 5 in Tabel 19 kan er iets worden opgemerkt. Wanneer er 1 iteratie Power Methode wordt uitgevoerd, verkrijgt men al de PageRank vector. Het is dus overbodig om erna nog de Rayleigh Quotiënt Methode uit te voeren. Wanneer men echter slechts deze laatst genoemde methode uitvoert, wordt er niet altijd de gewenste eigenvector verkregen behorende bij de grootste eigenwaarde. Dit zou een uitzondering kunnen zijn. Men laat dit punt over voor een mogelijk verder onderzoek.

Opnieuw ziet men dat de tijdsduur tot convergeren bij de Power Methode minder is vergeleken bij de combinatie van de twee methoden. Er worden wel meer iteraties vereist bij de Power Methode. Het zal dus weer aan de gewenste factor liggen welke van de twee methoden (de Power Methode of de combinatie van de twee methoden) men zal kiezen.

7 Conclusie

Nu men alle benodigde experimenten in Matlab heeft uitgevoerd, kunnen er conclusies worden getrokken. Om een idee te krijgen over welke methode het meest geschikt zou zijn, zijn er verschillende situaties gesimuleerd aan de hand van het programma pagerank.m. Als input matrix wordt de Google matrix G gebruikt. Deze kan op twee manieren gegenereerd worden, aan de hand van de random generator en surfer.m. De dimensie van de input Google matrix wordt telkens gevarieerd. In het geval dat G door de random generator wordt bepaald, komt de Power Methode als beste naar voren wanneer men geinteresseerd is in een zo kort mogelijke convergentietijd. Op de tweede plek komt de Arnoldi Methode. Op het moment dat G door het gebruik van surfer.m is vastgesteld, zal men zijn voorkeur uitspreken voor de Arnoldi Methode boven de Power Methode. Deze heeft bij alle matrixdimensies een kortere tijdsduur tot convergentie vergeleken de Power Methode.
Omdat de Gauss-Seidel, BiCGSTAB en GMRES8 methoden in bijna geen van de gevallen als beste uit de test komen, heeft men besloten om ze verder niet in beschouwing te nemen.

Nu de experimenten van pagerank.m uitgevoerd zijn, heeft men een vermoeden opgewekt over welke methode de zo kortst mogelijke tijdsduur tot convergeren zal geven. Hierbij worden slechts de Power Methode en de Arnoldi Methode in beschouwing genomen. Dit vermoeden wordt deels bevestigd bij het experiment waarbij de twee methoden worden vergeleken. In het geval waarbij G door de random generator is bepaald, is de Power Methode aanzienlijk sneller dan de Arnoldi Methoden. Men concludeert dat het vermoeden opgewekt door het gebruik van pagerank.m inderdaad klopte. Dit geldt niet voor het geval waarbij G door surfer.m is bepaald. Uit het experiment is gebleken dat de Power Methode een kortere convergentietijd nodig heeft in tegenstelling tot de resultaten van pagerank.m. Over het algemeen kan men dus concluderen dat de Power Methode het meest geschikt is wanneer men een zo kort mogelijke tijd tot convergentie wil bereiken.

Hierna wordt er een nieuwe methode geintroduceerd, de Rayleigh Quotiënt Methode. Deze vereist een symmetrische, positief (semi)definiete matrix als input. Wanneer men slechts de Rayleigh Quotiënt Methode toepast, wordt niet altijd de gewenste eigenvector bepaald. Er is daarom gekozen om eerst een paar iteraties Power Methode uit te voeren waarna de Rayleigh Quotiënt Methode zal volgen. Vervolgens zullen deze resultaten vergeleken worden met de resultaten verkregen door alleen de Power Methode uit te voeren.

In het eerste experiment zullen de SPD matrices gesimuleerd worden in Matlab. Dit zijn de zogehete testmatrices. De matrix dimensies worden gevarieerd van 5 tot 2000. Wanneer matrixdimensies tot 100 beschouwd worden en er gekeken wordt naar de kortste duur tot convergeren, dan zal men kiezen voor de Power Methode. Het aantal iteraties ligt echter bij het gebruik van alleen de Power Methode wel hoger dan bij het gebruik van de combinatie methoden. Men zal daarom zijn voorkeur uitspreken voor de combinatie methoden wanneer het om het aantal iteraties gaat. Bij matrix dimensies vanaf 150 concludeert men echter dat de combinatie methoden zowel een zo kort mogelijke tijdsduur geeft als een zo laag mogelijk aantal iteraties. Voor matrix dimensies vanaf 150 komt de combinatie van Power Methode en Rayleigh Quotiënt Methode dan als beste uit de test.

In het tweede experiment zullen de SP(semi)D matrices gebaseerd zijn op de Google matrix G. Deze is op een zodanige manier aangepast dat het de eigenschappen vertoont van een SP(semi)D matrix. Er worden opnieuw twee gevallen onderscheiden voor het vormen van G; via surfer.m en door gebruik van de random generator in Matlab. Men kijkt eerst naar het geval waarbij G door de random generator is gevormd. Bij iedere matrixdimensie zal het gebruik van enkel de Power Methode als beste naar voren komen wanneer men geinteresseerd is naar een zo kort mogelijke tijdsduur tot convergeren. Wanneer er echter naar het aantal iteraties wordt gekeken, dan komt de combinatie van de Power Methode met de Rayleigh Quotiënt Methode als meest geschikte methode uit het experiment.

Nu zal men zich concentreren op het geval dat de Google matrix G via surfer.m is gegenereerd. De tijdsduur tot convergeren is bij de Power Methode minder vergeleken bij de combinatie van de twee methoden. De Power Methode vereist echter wel een hoger aantal iteraties in deze situatie. Het zal dus aan de gewenste factor liggen voor welke van de twee methoden (de Power Methode of de combinatie van de twee methoden) er gekozen zal worden.

8 Discussie

In dit onderzoek zijn er verschillende scenario's beschouwd en daarover heeft men conclusies getrokken. Omdat het hier gaat om modellen die de werkelijkheid simuleren, bestaan er een aantal beperkingen die de juistheid van de resultaten onzeker maken.

Tegenwoordig betalen de meerderheid van de belangrijke webpagina's Google om een hogere PageRank waarde te krijgen. Op die manier komen ze hoger in de zoeklijst terecht en komen mensen vaker op die pagina. In dit onderzoek is hier echter geen rekening mee gehouden. Dit verschijnsel is niet in het model ingebouwd.

Men zou kunnen twijfelen aan de resultaten verkregen aan de hand van surfer.m. Omdat het programma niet vast te laten lopen, kunnen matrices tot een dimensie van 150 als input worden gebruikt. Het gehele webstructuur bevat echter meer dan 10 miljoen pagina's. Het webstructuur gevormd aan de hand van surfer.m is daarom niet geheel representatief voor de werkelijkheid.

Om toch een idee te krijgen over het gedrag van iets grotere matrices, heeft men matrices gegenereerd aan de hand van de random generator. Om de werkelijkheid op een zo goed mogelijke manier te simuleren, is er rekening gehouden dat 80% van de pagina's Dangling Nodes zijn en slechts een beperkt aantal pagina's naar elkaar verwijst. Ook in dit geval geldt dat de random gegenereerde matrices niet volledig op een realistische manier de werkelijkheid benaderen. De matrices nemen een dimensie aan tot 4000, wat niet dicht in de buurt van 10 miljoen ligt. Het zou kunnen voorkomen dat er enkele pagina's bestaan die een verwijzing bevatten naar zeer veel sites. Deze uitzonderingen zijn niet meegenomen in het model. Men kan niet met zekerheid zeggen dat de resultaten van de random generator volledig betrouwbaar zijn. Het blijft een model en een versimpeling van de werkelijkheid. De conclusies die hieruit worden getrokken zouden kunnen afwijken wanneer men het gehele web bekijkt.

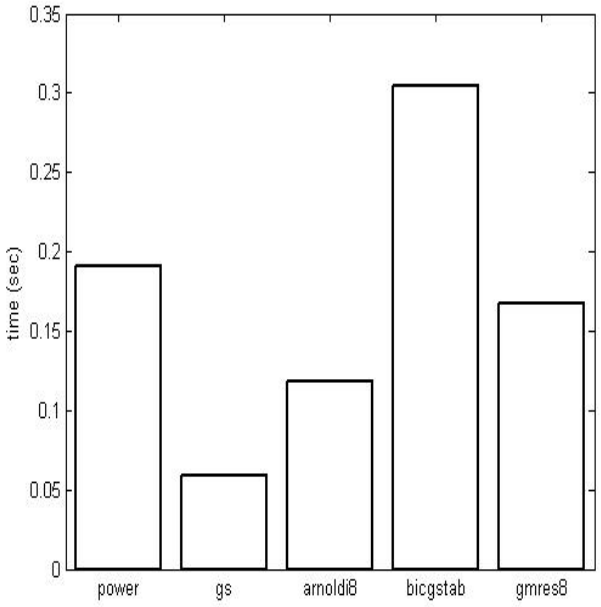
Om de tijdsduur tot convergentie te meten is telkens gebruik gemaakt van het commando 'tic toc'. De hierbij verkregen tijd is echter niet nauwkeurig vanwege de kleine tijdsintervallen waarmee er gewerkt wordt. De werkelijke tijdsduren zouden kunnen afwijken van degene die verkregen zijn. Om de tijdsduren nauwkeuriger te vergelijken, had er gekeken kunnen worden naar het aantal flops van een methode. Dit is het aantal floating point operaties.

Bij de experimenten waarbij de Power Methode vergeleken werd met de Arnoldi Methode, is er gebruik gemaakt van de startvector $\overline{p}_0 = 0.5 * \text{ones}\,(n, 1)$. Wanneer de Rayleigh Quotiënt Methode is geintroduceerd, wordt de startvector willekeurig bepaald met de random generator. Het zou kunnen zijn dat het nemen van een andere startvector bij deze experimenten zou leiden tot verschillende resultaten.

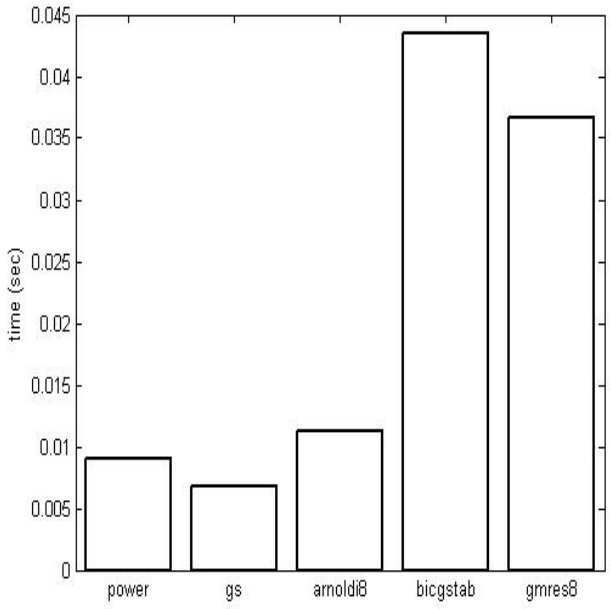
Tot slot zou men de manier waarop Google matrix G aangepast wordt tot de SP(semi)D matrix F_{nieuw} aan kunnen merken. Door deze aanpassingen zou G zijn oorspronkelijke eigenschappen kunnen verliezen. Slechts de eigenwaarden groter en gelijk aan 0 blijven hetzelfde. De resultaten die verkregen worden zijn van toepassing op F_{nieuw}. Men kan niet met zekerheid zeggen dat ze evenzo geldig zijn voor G. Een punt voor vervolgonderzoek zou kunnen zijn hoe men op een zo nauwkeurig mogelijke manier G zou kunnen aanpassen zodat het een symmetrische, positief (semi)definiete matrix wordt opdat het niet zijn oorspronkelijke eigenschappen verliest.

9 Bijlagen Figuren

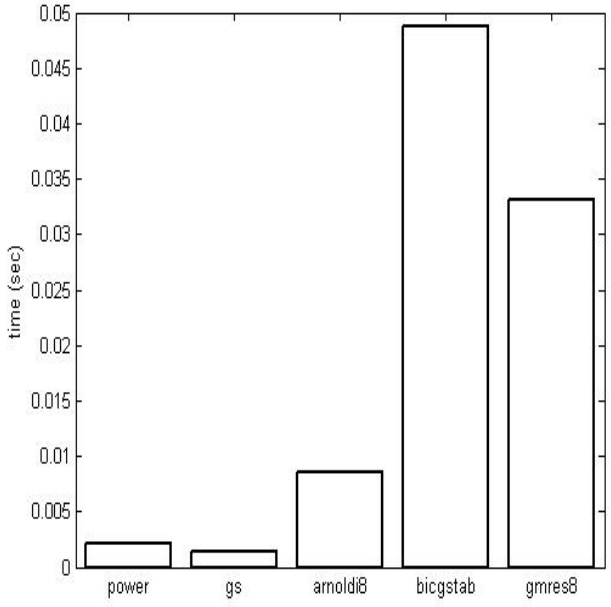
9.1 Tijdhistogrammen pagerank.m



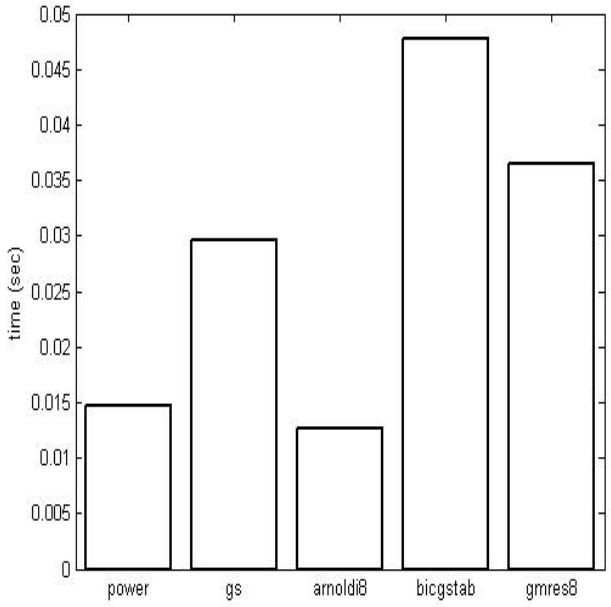
Figuur 10: Tijdhistogram van 150x150 matrix genereerd door surfer.m (input: www.nos.nl)



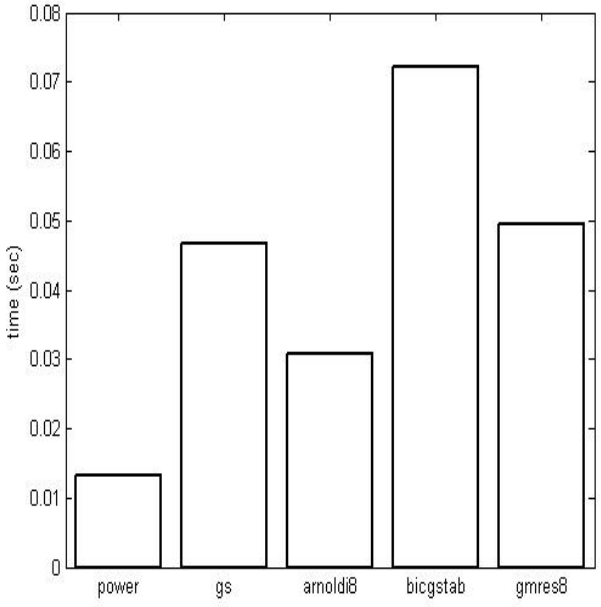
Figuur 11: Tijdhistogram van 150 × 150 matrix genereerd door surfer.m (input: http://www.lyricsfreak.com/)



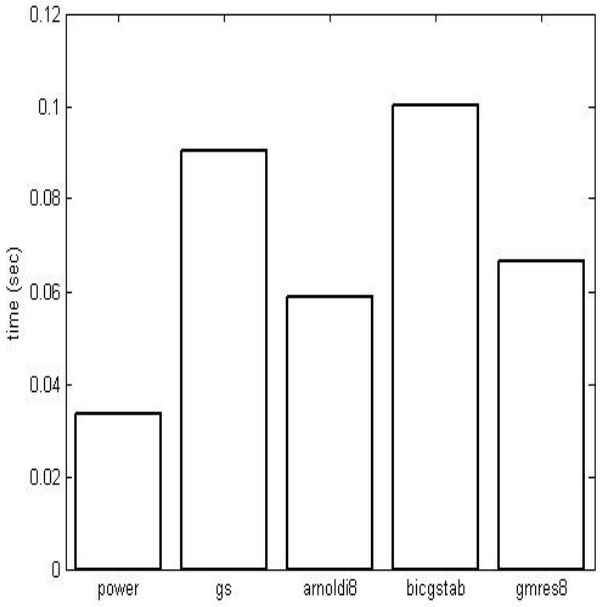
Figuur 12: Tijdhistogram van 150×150 matrix genereerd door random generator



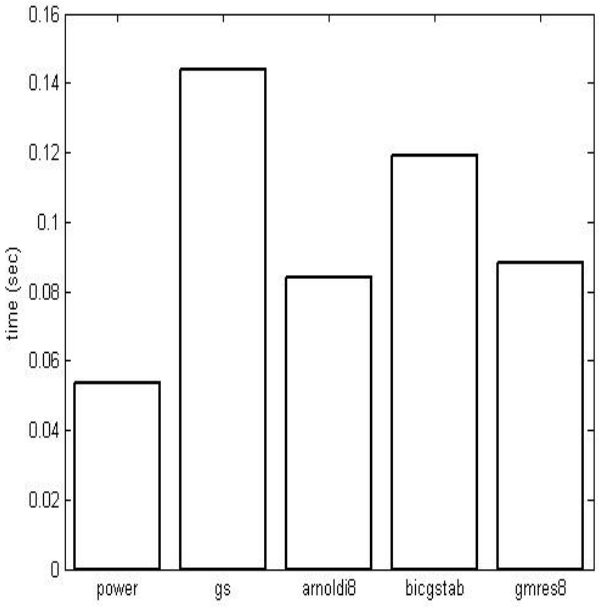
Figuur 13: Tijdhistogram van 1000×1000 matrix genereerd door random generator



Figuur 14: Tijdhistogram van 2000×2000 matrix genereerd door random generator

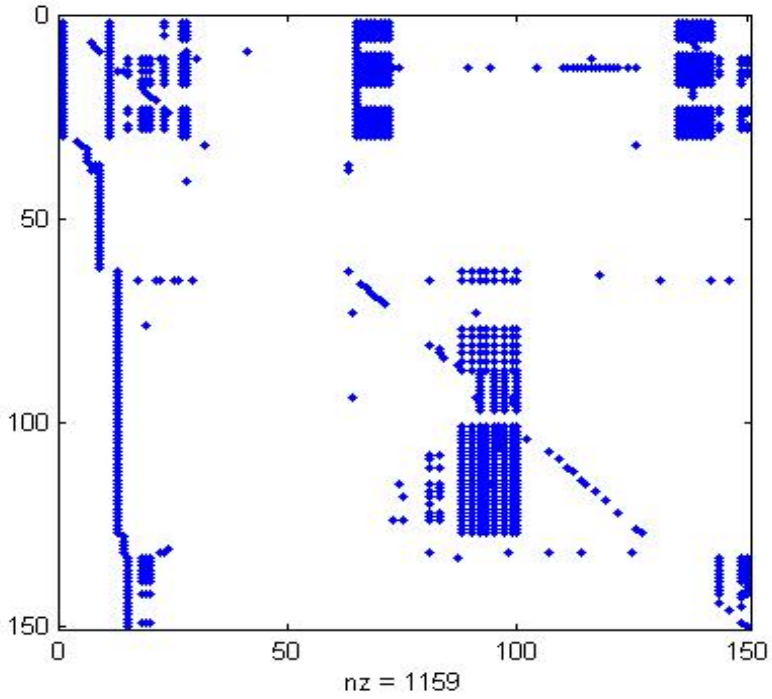


Figuur 15: Tijdhistogram van 3000×3000 matrix genereerd door random generator

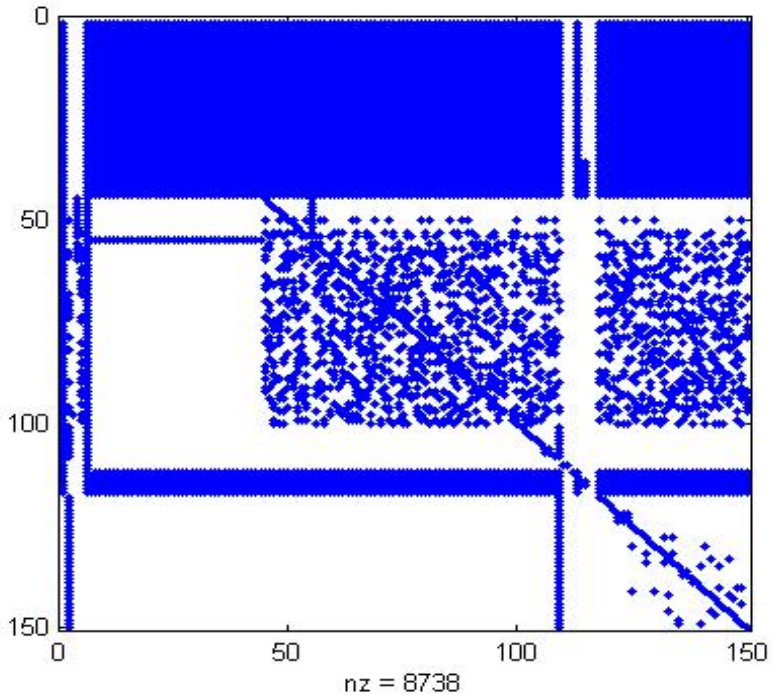


Figuur 16: Tijdhistogram van 4000×4000 matrix genereerd door random generator

9.2 Visuele Weergaven van Gelinkte Pagina's



Figuur 17: Visuele Weergave van 150 Gelinkte Pagina's bij input www.nos.nl



Figuur 18: Visuele Weergave van 150 Gelinkte Pagina's bij input www.lyricsfreak.nl

10 Bijlagen Matlab Codes

10.1 Codes gebruikmakend van surfer.m

10.1.1 Surfer.m

```
function [U,G] = surfer(root,n)
% SURFER  Create the adjacency graph of a portion of the Web.
%    [U,G] = surfer(root,n) starts at the URL root and follows
%    Web links until it forms an adjacency graph with n nodes.
%    U = a cell array of n strings, the URLs of the nodes.
%    G = an n-by-n sparse matrix with G(i,j)=1 if node j is linked to node i.
%
%    Example:  [U,G] = surfer('http://www.harvard.edu',500);
%    See also PAGERANK.
%
%    This function currently has two defects.  (1) The algorithm for
%    finding links is naive.  We just look for the string 'http:'.
%    (2) An attempt to read from a URL that is accessible, but very slow,
%    might take an unacceptably long time to complete.  In some cases,
%    it may be necessary to have the operating system terminate MATLAB.
%    Key words from such URLs can be added to the skip list in surfer.m.

% Initialize

U = cell(n,1);
hash = zeros(n,1);
G = logical(sparse(n,n));
m = 1;
U{m} = root;
hash(m) = hashfun(root);

for j = 1:n

   % Try to open a page.

   try
      disp(['open ' num2str(j) ' ' U{j}])
      page = urlread(U{j});
   catch
      disp(['fail ' num2str(j) ' ' U{j}])
      continue
   end

   % Follow the links from the open page.

   for f = findstr('http:',page);

      % A link starts with 'http:' and ends with the next double quote.

      e = min(findstr('"',page(f:end)));
      if isempty(e), continue, end
      url = deblank(page(f:f+e-2));
```

```
        url(url<' ') = '!';   % Nonprintable characters
        if url(end) == '/', url(end) = []; end

        % Look for links that should be skipped.

        skips = {'.gif','.jpg','.pdf','.css','lmscadsi','cybernet', ...
                 'search.cgi','.ram','www.w3.org', ...
                 'scripts','netscape','shockwave','webex','fansonly'};
        skip = any(url=='!') | any(url=='?');
        k = 0;
        while ~skip & (k < length(skips))
           k = k+1;
           skip = ~isempty(findstr(url,skips{k}));
        end
        if skip
           if isempty(findstr(url,'.gif')) & isempty(findstr(url,'.jpg'))
              disp(['      skip ' url])
           end
           continue
        end

        % Check if page is already in url list.

        i = 0;
        for k = find(hash(1:m) == hashfun(url))';
           if isequal(U{k},url)
              i = k;
              break
           end
        end

        % Add a new url to the graph there if are fewer than n.

        if (i == 0) & (m < n)
           m = m+1;
           U{m} = url;
           hash(m) = hashfun(url);
           i = m;
        end

        % Add a new link.

        if i > 0
           disp(['      link ' int2str(i) ' ' url])
           G(i,j) = 1;
        end
     end
  end

  save('Surfer','n','U','G')

  end
```

```
%-------------------------

function h = hashfun(url)
 %Almost unique numeric hash code for pages already visited.
h = length(url) + 1024*sum(url);
end
```

10.1.2 Aanmaken van Google Matrix

```
%Google Matrix implementeren door Link Matrix te gebruiken

load Surfer

alpha = 0.85;
A = full(G);
H = zeros(n);

%Hyperlink matrix H implementeren
for j = 1:n
    som = 0;

    for i = 1:n
        som = som + A(i,j);
    end

    if som > 0
        a = 1/som;
    end

    for i = 1:n

        if A(i,j) == 1
            H(i,j) = a;
        end

    end

end

%Matrix S implementeren
S = H;
b = 1/n;

for j = 1:n
    som = 0;

    for i = 1:n
        som = som + H(i,j);
    end

    if som == 0

        for k = 1:n
            S(k,j) = b;
        end

    end

end

%Transport matrix T implementeren
```

```
T = b*ones(n);

%Google Matrix G implementeren
Google = alpha*S + (1 - alpha)*T;

save('Google_Matrix_Surfer','alpha','H','S','T','Google','n')
```

10.1.3 Power Methode

```
%Power Method toegepast op Google Matrix G
%Matrix G is bepaald aan de hand van surfer.m

load Google_Matrix_Surfer

epsilon = 10^-12;

stop = 1; %stopcriterium
p_k = 0.5*ones(n,1);
lambda_k = 0;
iteraties = 0;

tic
while(stop >= epsilon)

    z_k = Google*p_k;
    p_k = z_k / norm(z_k);
    lambda_k = (p_k)'*z_k;
    p_oud = p_k;

    stop = abs(1 - lambda_k); %stopcriterium voor volgende iteratie

    iteraties = iteraties + 1;

end
toc

positie_lijst = zeros(n,1);
sites_lijst =  cell(n,1);

zeros(n,1);

for i = 1:n

    m = max(p_k);

    for k = 1:n

        if p_k(k,1) == m
            positie_lijst(i,1) = p_k(k,1);

            sites_lijst(i,1) = U(k,1);
            p_k(k,1) = 0;
            break
        end

    end

end

positie_lijst;
sites_lijst;
```

iteraties

10.1.4 Arnoldi Methode

```
%Arnoldi Methode toegepast op Google Matrix G
%Matrix G is bepaald aan de hand van surfer.m

load Google_Matrix_Surfer

epsilon = 10^-2; %toegestane tolerantie

q = 0.5*ones(n,1); %willekeurige startvector
%q = q_0 / norm(q_0); %normaliseren van de startvector
Q_0 = zeros(n);
Q_0(1:n,1) = q;
nrm = 1;
j = 0;

r = q;
nrm = 1;
iteraties = 0;
sc = 3000; %stopcriterium

tic
while sc > epsilon
    iteraties = iteraties + 1;

    q = r/nrm;
    Q_0(1:n,j+1)= q;
    j = j+1;
    r = Google*q;

    for i = 1:j
        h(i,j) = Q_0(1:n,i)'*r;
        r = r - h(i,j)*Q_0(1:n,i);
    end

    nrm = norm(r);
    M = max(eig(h));

    sc = abs(1-M); %stopcriterium opnieuw definieren mbv eigenwaarden

    if j < n
        h(j+1,j) = nrm;
    end

    if j == n
        break
    end
end

H = h(1:iteraties, 1:iteraties); %Hessenberg Matrix
Q = Q_0(1:n, 1:iteraties);

[V,D] = eig(H); %Geeft eigenwaarden in diagonaalmatrix D en
                %bijbehorende eigenvectoren in matrix V
```

```
eig_vec = Q*V(:,1); %Eigenvector van G bepalen bij Eigenwaarden 1


PRV = abs(eig_vec); %PageRank Vector bepalen
toc

positie_lijst = zeros(n,1);
sites_lijst = cell(n,1);

for i = 1:n

    m = max(PRV);

    for k = 1:n

        if PRV(k,1) == m
            positie_lijst(i,1) = PRV(k,1);
            sites_lijst(i,1) = U(k,1);
            PRV(k,1) = 0;
            break
        end

    end

end

 positie_lijst;
 sites_lijst;
 iteraties
```

10.2 Codes gebruikmakend van Link Matrix

10.2.1 Nabootsen van Link Matrix

```
%Link Matrix Q implementeren

%creeren van een n x n matrix waarbij B_ij=1 als B_i een verwijzing
%naar B_j heeft, anders B_ij=0.
% 80% van de gevallen zijn Dangling nodes

 n = input ('Dimensie Matrix = ');
 Q = zeros(n);
 U = zeros(n,1);
 nul = zeros(1,n);

 for i = 1:n
     for j = 1:n
         if rand > 0.6
             Q(i,j) = 1;
         end
     end
 end

 for i = 1:n
     if rand < 0.8
         Q(i,1:n) = nul;
     end
 end

for i = 1:n
    U(i,1) = i;
end

save('linkmatrix_nxn','U','Q','n')
```

10.2.2 Aanmaken van Google Matrix

```
%Google Matrix implementeren door Link Matrix te gebruiken

load linkmatrix_nxn

alpha = 0.85;
A = full(Q);
H = zeros(n);

%Hyperlink matrix H implementeren
for j = 1:n
    som = 0;

    for i = 1:n
        som = som + A(i,j);
    end

    if som > 0
        a = 1/som;
    end

    for i = 1:n

        if A(i,j) == 1
            H(i,j) = a;
        end

    end

end

%Matrix S implementeren
S = H;
b = 1/n;

for j = 1:n
    som = 0;

    for i = 1:n
        som = som + H(i,j);
    end

    if som == 0

        for k = 1:n
            S(k,j) = b;
        end

    end

end

%Transport matrix T implementeren
```

```
T = b*ones(n);

%Google Matrix G implementeren
G = alpha*S + (1 - alpha)*T;

save('Google_Matrix_Link','alpha','H','S','T','G')
```

10.2.3 Power Methode

```
%Power Methode toegepast op Google Matrix G
%Matrix G is bepaald aan de hand van Link Matrix

load linkmatrix_nxn
load Google_Matrix_Link

epsilon = 10^-12;

stop = 1; %stopcriterium
p_k = 0.5*ones(n,1);
lambda_k = 0;
iteraties = 0;

tic
while(stop >= epsilon)
    z_k = G*p_k;
    p_k = z_k / norm(z_k);
    lambda_k = (p_k)'*z_k;
    p_oud = p_k;

    stop = abs(1 - lambda_k); %stopcriterium voor volgende iteratie

    iteraties = iteraties + 1;
end
toc

positie_lijst = zeros(n,1);
sites_lijst = zeros(n,1);

for i = 1:n

    m = max(p_k);

    for k = 1:n

        if p_k(k,1) == m
            positie_lijst(i,1) = p_k(k,1);
            sites_lijst(i,1) = U(k,1);
            p_k(k,1) = 0;
            break
        end

    end

end

positie_lijst;
sites_lijst;
iteraties
```

10.2.4 Arnoldi Methode

```
%Arnoldi Methode toegepast op Google Matrix G
%Matrix G is bepaald aan de hand van Link Matrix

load linkmatrix_nxn
load Google_Matrix_Link

epsilon = 10^-12; %toegestane tolerantie

q = 0.5*ones(n,1); %willekeurige startvector
%q = q_0 / norm(q_0); %normaliseren van de startvector
Q_0 = zeros(n);
Q_0(1:n,1) = q;
nrm = 1;
j = 0;

r = q;
nrm = 1;
iteraties = 0;
sc = 3000; %stopcriterium

tic
while sc > epsilon
    iteraties = iteraties + 1;

    q = r/nrm;
    Q_0(1:n,j+1)= q;
    j = j+1;
    r = G*q;

    for i = 1:j
        h(i,j) = Q_0(1:n,i)'*r;
        r = r - h(i,j)*Q_0(1:n,i);
    end

    nrm = norm(r);
    M = max(eig(h));

    sc = abs(1-M); %stopcriterium opnieuw definieren mbv eigenwaarden

    if j < n
        h(j+1,j) = nrm;
    end

    if j == n
        break
    end
end

H = h(1:iteraties, 1:iteraties); %Hessenberg Matrix
Q = Q_0(1:n, 1:iteraties);

[V,D] = eig(H); %Geeft eigenwaarden in diagonaalmatrix D en
```

```
%bijbehorende eigenvectoren in matrix V


eig_vec = Q*V(:,1); %Eigenvector van G bepalen bij Eigenwaarden 1

PRV = abs(eig_vec); %PageRank Vector bepalen
toc

positie_lijst = zeros(n,1);
sites_lijst = zeros(n,1);

for i = 1:n

    m = max(PRV);

    for k = 1:n

        if PRV(k,1) == m
            positie_lijst(i,1) = PRV(k,1);
            sites_lijst(i,1) = U(k,1);
            PRV(k,1) = 0;
            break
        end

    end

end

positie_lijst;
sites_lijst;
iteraties
```

10.3 Codes Raileigh Quotiënt Methode

10.3.1 Aanmaken van SPD Testmatrices

```
%SPD Testmatrices Implementeren

n = input ('Dimension A = ');

v_1 = [1:n-1];
v_2 = [2:n];
v = v_1 - v_2;

w_1 = [4:n+3];
w_2 = [2:n+1];
w = w_1 - w_2;

A = diag(w) + diag(v,-1) + diag(v,1);


save('matrix','A')
```

10.3.2 Power Methode voor Los Gebruik voor Testmatrices

```
%Power Methode Experiment Raileigh Quotient

A = input ('Matrix A = ');
rand('state',100)

n = size(A);

epsilon = 10^-8;

stop = 1; %stopcriterium
z_0 = rand(n(1),1);
%z_0 = 0.5*ones(n(1),1);
p_0 = z_0 / norm(z_0);
p_oud = p_0;
lambda_oud = 0;
iteraties = 0;

tic
while(stop >= epsilon)
    z_k = A*p_oud;
    p_k = z_k / norm(z_k);
    lambda_k = (p_oud)'*z_k;
    p_oud = p_k;

    stop = norm(lambda_oud - lambda_k); %stopcriterium voor volgende iteratie
    lambda_oud = lambda_k;

    iteraties = iteraties + 1;
end
toc
iteraties
save('startvector','z_0')
```

10.3.3 Power Methode vooraf Rayleigh Quotiënt Methode voor Testmatrices

```
%Power Methode voor Experiment Raileigh Quotient Methode

A = input ('Matrix A = ');
rand('state',100)

load startvector

n = size(A);

epsilon = 10^-8;

stop = 1; %stopcriterium

%z_0 = 0.5*ones(n(1),1);
%z_0 = rand(n(1),1);

p_0 = z_0 / norm(z_0);
p_oud = p_0;
lambda_oud = 0;
iteraties = 0;

tic
while(stop >= epsilon & iteraties < 350)
    z_k = A*p_oud;
    p_k = z_k / norm(z_k);
    lambda_k = (p_oud)'*z_k;
    p_oud = p_k;

    stop = norm(lambda_oud - lambda_k); %stopcriterium voor volgende iteratie
    lambda_oud = lambda_k;

    iteraties = iteraties + 1;
end
toc

p_k
iteraties

save('power','A')
```

10.3.4 DF-SANE Algoritme

```
function [xk,IT,EF,BL,t,fa,norma]=DFSANE(x0,Fun,nmax,ea,er)
%************************ PARAMETERS ********************************
ep = 1.0d-10;
M = 10;
gamma = 1.0d-4;
norma = [];
%*********************************************************************
EV = zeros(M,1);
xk = x0;
n = length(xk);
rn = sqrt(n);
EF = 0;
BL = 0;
fa=0;
alfa = 1;
k = 0;
Fk = feval(Fun,xk);
fmk = Fk'*Fk;
NF = sqrt(fmk);
NF0 = NF;
eta = NF0;
%*************************************************
norma = [norma,NF];
parada = ea + er*(NF/rn);
%parada = er;
%parada = er*(NF/rn);
%*************************************************
EV(1) = fmk;
t1 = clock;
while NF/rn > parada & k<nmax,
%while NF > parada & k<nmax,
    if (abs(alfa) < ep) | (abs(alfa)>(1/ep))
        if (NF>1)
            alfa = 1;
        elseif(NF>=1.0e-5 & NF<=1)
            alfa = NF;
        elseif(NF < 1.0e-5)
            alfa = 1.0e-5;
        end
    end
    d = -(1/alfa).*Fk;
    mf = max(EV);
    bl1 = 0;
    lam1 = 1;
    lam2 = 1;
    while 1,
        if (bl1==100),
            fa=2;
            break;
        end
        d1 = lam1.*d;
        x1 = xk + d1;
```

```
        F1 = feval(Fun,x1);
        fmk1 = F1'*F1;
        NF1 = sqrt(fmk1);
        EF = EF+1;
        if(fmk1 <= mf + eta - (lam1^2*gamma)*fmk)
            sk = d1;
            yk = F1-Fk;
            xk = x1;
            Fk = F1;
            NF = NF1;
            fmk = fmk1;
            break;
        end
        d2 = -lam2.*d;
        x2 = xk + d2;
        F2 = feval(Fun,x2);
        fmk2 = F2'*F2;
        NF2 = sqrt(fmk2);
        EF = EF+1;
        if(fmk2 <= mf + eta - (lam2^2*gamma)*fmk)
            sk = d2;
            yk = F2-Fk;
            xk = x2;
            Fk = F2;
            NF = NF2;
            fmk = fmk2;
            break;
        end
        bl1 = bl1 + 1;
        lam1 = parab2p(lam1, fmk1, fmk, -2*fmk);
        lam2 = parab2p(lam2, fmk2, fmk, -2*fmk);
    end
    if (fa),
            break;
    end
    if bl1 >= 1,
        BL = BL + 1;
    end
    alfa = (sk'*yk)/(sk'*sk);
    k = k+1;
    eta = NF0/((1+k)^2);
    %eta = NF0/(2^k);
    %eta = NF0/(k^1.1);
    EV(mod(k,M+1)+1) = fmk;
    %************************************************
    norma=[norma,NF];
    %************************************************
end
if (fa==1),
    disp('Error ?weird point!');
elseif (fa==2)
        disp('max number of backtrackings exceeded');
else
```

```
    t = etime(clock,t1);
    IT = k;
end;
```

10.3.5 Benodigde Function File voor DF-SANE Methode

```
function lam = parab2p(lam, FF1, FF, g0)
l = 0.1;
u = 0.5;

l2 = lam^2;
lam1 = (-g0*l2)./(2*(FF1-FF-lam*g0));
c1 = l*lam;
c2 = u*lam;
if lam1 < c1,
   lam = c1;
elseif lam1 > c2,
   lam = c2;
else
   lam = lam1;
end
```

10.3.6 Rayleigh Quotiënt voor Testmatrices

```
function [y] = raileigh(x)

load power
r = (x'*A*x)/(x'*x);
y = A*x - r*x;

end
```

10.3.7 Aanmaken van Google Matrix adhv Link Matrix / Google Matrix omzetten naar SPD Matrix F_{nieuw}

```
%Google Matrix implementeren door Link Matrix te gebruiken
%Google Matrix SPD maken

load linkmatrix_nxn

alpha = 0.85;
A = full(Q);
H = zeros(n);

%Hyperlink matrix H implementeren
for j = 1:n
    som = 0;

    for i = 1:n
        som = som + A(i,j);
    end

    if som > 0
        a = 1/som;
    end

    for i = 1:n

        if A(i,j) == 1
            H(i,j) = a;
        end

    end

end

%Matrix S implementeren
S = H;
b = 1/n;

for j = 1:n
    som = 0;

    for i = 1:n
        som = som + H(i,j);
    end

    if som == 0

        for k = 1:n
            S(k,j) = b;
        end

    end

end
```

```
%Transport matrix T implementeren
T = b*ones(n);

%Google Matrix G implementeren
G = alpha*S + (1 - alpha)*T;

%Google Matrix symmetrisch maken
F = G + G';

%Matrix F positief definiet maken
[vec,val]=eig(F);
val(val<0)=eps;
F_nieuw=vec*val*vec';

%save('Google_Matrix_Link_2','alpha','H','S','T','F_nieuw','n')
save('Google','F_nieuw')
```

10.3.8 Aanmaken van Google Matrix adhv surfer.m / Google Matrix omzetten naar SPD Matrix F_{nieuw}

```
%Google Matrix implementeren door Link Matrix te gebruiken
%Google Matrix SPD maken

load Surfer

alpha = 0.85;
A = full(G);
H = zeros(n);

%Hyperlink matrix H implementeren
for j = 1:n
    som = 0;

    for i = 1:n
        som = som + A(i,j);
    end

    if som > 0
        a = 1/som;
    end

    for i = 1:n

        if A(i,j) == 1
            H(i,j) = a;
        end

    end

end

%Matrix S implementeren
S = H;
b = 1/n;

for j = 1:n
    som = 0;

    for i = 1:n
        som = som + H(i,j);
    end

    if som == 0

        for k = 1:n
            S(k,j) = b;
        end

    end

end
```

```
%Transport matrix T implementeren
T = b*ones(n);

%Google Matrix implementeren
Google = alpha*S + (1 - alpha)*T;

%Google Matrix symmetrisch maken
F = Google + Google';

%Matrix F positief definiet maken
[vec,val]=eig(F);
val(val<0)=eps;
F_nieuw=vec*val*vec';

save('Google','F_nieuw')
```

10.3.9 Power Methode voor Los Gebruik voor SPD Matrix F_{nieuw}

```
%Power Methode Experiment Raileigh Quotient

load Google

%F_nieuw = input ('Matrix F_nieuw = ');
rand('state',100)

n = size(F_nieuw);

epsilon = 10^-8;

stop = 1; %stopcriterium
z_0 = rand(n(1),1);
%z_0 = 0.5*ones(n(1),1);
p_0 = z_0 / norm(z_0);
p_oud = p_0;
lambda_oud = 0;
iteraties = 0;

tic
while(stop >= epsilon)
    z_k = F_nieuw*p_oud;
    p_k = z_k / norm(z_k);
    lambda_k = (p_oud)'*z_k;
    p_oud = p_k;

    stop = norm(lambda_oud - lambda_k); %stopcriterium voor volgende iteratie
    lambda_oud = lambda_k;

    iteraties = iteraties + 1;
end
toc
iteraties
save('startvector','z_0','F_nieuw')
```

10.3.10 Power Methode vooraf Rayleigh Quotiënt Methode voor SPD Matrix F_{nieuw}

```
%Power Methode voor Experiment Raileigh Quotient Methode

%F_nieuw = input ('Matrix F_nieuw = ');
rand('state',100)

load startvector

n = size(F_nieuw);

epsilon = 10^-8;

stop = 1; %stopcriterium

%z_0 = 0.5*ones(n(1),1);
%z_0 = rand(n(1),1);

p_0 = z_0 / norm(z_0);
p_oud = p_0;
lambda_oud = 0;
iteraties = 0;

tic
while(stop >= epsilon & iteraties < 1)
    z_k = F_nieuw*p_oud;
    p_k = z_k / norm(z_k);
    lambda_k = (p_oud)'*z_k;
    p_oud = p_k;

    stop = norm(lambda_oud - lambda_k); %stopcriterium voor volgende iteratie
    lambda_oud = lambda_k;

    iteraties = iteraties + 1;
end
toc

p_k
iteraties

save('power','F_nieuw')
```

10.3.11 Rayleigh Quotiënt voor SPD Matrix F_{nieuw}

```
function [y] = rq(x)

load power

r = (x'*F_nieuw*x)/(x'*x);
y = F_nieuw*x - r*x;

end
```

Bibliografie

[1] http://en.wikipedia.org/wiki/Min-max_theorem/.

[2] E. Andersson and P. A. Ekström. *Investigating Google's PageRank Algorithm*. Uppsala University, Information Technology, Dept. of Scientific Computing, Spring 2004. Report in Scientific Computing, Advanced Course.

[3] E. Berkhof. Efficient Approximation of Google's PageRank. repository.tudelft.nl/assets/uuid/3A935f65ff-42bc-43dd-9e79-ec5fee6f48f3/BSc_verslag_Erik_Berkhof.pdf&ei=XuPxT9PrBIn98gO9nKWEDQ&usg=AFQjCNHtGK-ju-ZFA1Y5ZknIVTQlHfU6sg&sig2=sQcOgFZTz4OAC3vpE7L9oA/, July 2009. Delft Institute of Applied Mathematics.

[4] J. Brandts. De PageRank van Google: De grootste matrixberekening ooit. *De Nieuwe Wiskrant*, pages 8–12, December 2007.

[5] J. B. Fraleigh and R. A. Beaugard. *Linear Algebra*. Addison-Wesley Publishing Company, Massachusetts, third edition, 1995.

[6] G. H. Golub and C. Greif. An Arnoldi-Type Algorithm for Computing PageRank. *BIT Numerical Mathematics*, 46(4), December 2006. Department of Computer Science, The University of British Columbia, Vancouver, Canada and SCCM Program, Gates 2B, Stanford University, Stanford CA, USA.

[7] M. T. Heath. Parallel Numerical Algorithms, Chapter 12 Eigenvalue Problems. http://www.scribd.com/doc/76149091/1/Parallel-Numerical-Algorithms/. Department of Computer Science University of Illinois at Urbana-Champaign.

[8] I. C. Ipsen and R. S. Wills. *Mathematical Properties and Analysis of Google's PageRank*. Department of Mathematics, North Carolina State University, Raleigh, USA, 2006.

[9] I. C. Ipsen and R. S. Wills. Ordinal Ranking for Google's PageRank. *SIAM J. Matrix Analysis and Applications*, 30(4):1677–1696, 2009.

[10] X. Jiao. Numerical Analysis I (Numerical Linear Algebra). http://www.ams.sunysb.edu/~jiao/teaching/ams526_fall11/index.html/. SUNY Stony Brook.

[11] W. La Cruz, J. M. Martínez, and M. Raydan. Spectral residual method without gradient information for solving large-scale nonlinear systems of equations. *Mathematics of Computation*, 75(255):1429–1448, April 2006.

[12] A. N. Langville and C. D. Meyer. Deeper Inside PageRank. http://langvillea.people.cofc.edu/, October 2004. Department of Mathematics, N. Carolina State University, Raleigh, USA.

[13] A. N. Langville and C. D. Meyer. *Google's PageRank and Beyond: The Science of Search Engine Rankings*. Princeton University Press, Princeton, New Jersey, 2006.

[14] R. Larson. *Elementary Linear Algebra*. Richard Stratton, Boston, MA, seventh edition, 2012. Chapter 10.3: Power Method for Approximating Eigenvalues.

[15] D. Löchel. Numerical Methods for Eigenvalue Problems. http://na.math.kit.edu/loechel/research/?lang=en/, February 2008. Mathematisches Institut Heinrich-Heine-Universität Düsseldorf.

[16] C. Moler. *Experiments with Matlab*. Electronic Edition: MathWorks, Inc., October 2011. Chapter 7: Google's PageRank.

[17] T. S. Nguyên. Arnoldi Algorithm. https://www.math.uni-bremen.de/zetem/cms/media.php/255/arnoldi.pdf, March 2009. Universitaet Bremen Zentrum fuer Technomathematik.

[18] P. O. Persson. Arnoldi/Lanczos Iterations. http://persson.berkeley.edu/18.335/, December 2007. Introduction to Numerical Methods Lecture 15.

[19] P. O. Persson. The QR Algorithm I. http://persson.berkeley.edu/18.335/, October 2007. Introduction to Numerical Methods Lecture 15.

[20] R. J. Radke. A Matlab Implementation of the Implicity Restarted Arnoldi Method for Solving Large-Scale Eigenvalue Problems. www.ecse.rpi.edu/~rjradke/papers/radkemathesis.pdf&ei=t97xT-uFCsHW0QXW0YH8DQ&usg=AFQjCNFbYVJMFfYgZ4FVXtN0zHH4mdpkmw&sig2=o3u33rtE4sy5i2LVh38O9w/, April 1996. Rice University, Houston, Texas, USA.

[21] S. van Veldhuizen. Efficient Methods for Solving Advection Diffusion Reaction Equations. http://www.ewi.tudelft.nl/index.php?id=23105&L=1/, August 2005. Delft Institute of Applied Mathematics.

[22] M. H. Yang. Matrix Computation. http://faculty.ucmerced.edu/mhyang/course/eecs275/index.htm/. Electrical Engineering and Computer Science University of California, Merced, CA.

[23] J. Yin. On Generalized Arnoldi Methods for the Computation of PageRank. www.tongji.edu.cn/~yin/cn/pr-handout.pdf/, March 2010. Lecture 16: Rayleigh Quotient Iteration.

[24] A. J. Yzelman, T. Hartskeerl, and K. van Vliet. The Google PageRank. http://langvillea.people.cofc.edu/.