

KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT DER TOEGEPASTE WETENSCHAPPEN
DEPARTEMENT ELEKTROTECHNIEK
Kard. Mercierlaan 94, 3001 Leuven (Heverlee)

**ALGORITHMS AND ARCHITECTURES
FOR ADAPTIVE ARRAY SIGNAL PROCESSING**

Promotor:
Prof. Dr. Ir. J. Vandewalle

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de toegepaste wetenschappen

door

Filiep VANPOUCKE

Februari 1995

KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT DER TOEGEPASTE WETENSCHAPPEN
DEPARTEMENT ELEKTROTECHNIEK
Kard. Mercierlaan 94, 3001 Leuven (Heverlee)

**ALGORITHMS AND ARCHITECTURES
FOR ADAPTIVE ARRAY SIGNAL PROCESSING**

Jury:

Prof. Dr. Ir. W. Dutré, vice-decaan, voorzitter
Prof. Dr. Ir. J. Vandewalle, promotor
Prof. Dr. Ir. A. Bultheel
Prof. Dr. Ir. F. Catthoor
Prof. Dr. Ir. B. De Moor
Prof. Dr. Ir. E. Deprettere (T.U. Delft)
Prof. Dr. Ir. J. McWhirter (DRA, U.K.)
Dr. Ir. M. Moonen

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de toegepaste wetenschappen

door

Filiep VANPOUCKE

U.D.C. 621.391

Februari 1995

Voor Inge

Voorwoord

Bij de voltooiing van mijn doctoraat wil ik met veel genoegen een woord van dank richten tot iedereen die ertoe heeft bijgedragen. SISTA is zonder twijfel een aangename en stimulerende groep met internationale uitstraling. Dit klimaat is niet in het minst het werk van mijn promotor Prof. Vandewalle. Ik dank hem dan ook oprecht voor de vele kansen die ik in de voorbije jaren gekregen heb.

Prof. De Moor stond mee aan de wieg van dit doctoraat. Zijn dynamisme en actieve belangstelling stonden altijd garant voor een stevige zet in de rug.

Mijn speciale dank gaat naar Dr. Marc Moonen. Dit doctoraat sluit aan bij zijn vroegere werk en is gegroeid uit een intense samenwerking. Met zijn rijke ervaring heeft hij mij geïntroduceerd in de wetenschappelijke wereld. Welgemeend mag ik stellen dat ik me geen betere begeleider kon wensen.

Een gewaardeerde wetenschapper die ik via Marc heb leren kennen, is Prof. Deprettere van de Technische Universiteit Delft. De interactie met hem en de onderzoekers in zijn groep is een constante verrijking. Ik ben dan ook verheugd dat hij bereid was om deel te nemen aan het leescomité.

Ik wil Prof. Catthoor danken voor de aangename samenwerking op het vlak van architectuurontwerp. De visie op parallelle architecturen in dit werk is terdege beïnvloed door het NANA project waarvan hij de enthousiaste voortrekker is.

It is also a pleasure to have Prof. McWhirter from DRA Malvern as a member of the jury. He has always shown a sincere interest in my work. Therefore, I am grateful for his willingness to review this text.

Verder bedank ik ook Prof. Bultheel voor zijn wijde wetenschappelijke interesse en onmiddellijke bereidheid om in de jury te zetelen.

Als lid van de facultaire doktoraatscommissie had ik de kans om Prof. Dutré beter te leren kennen. Ik dank hem voor zijn vlotte medewerking als voorzitter van de jury.

During my work I have had the opportunity to pay two visits to Prof. Paulraj and Prof. Kailath at the Information Systems Laboratory of Stanford University. These magnificent stays were very rewarding. Part of this text is a direct outcome of a fruitful collaboration.

Alle huidige en vroegere leden van SISTA met naam vermelden is een delicate aangelegenheid geworden. Laat me enkel de volgende personen expliciet vernoemen: Bart en Lieven, met wie ik het langst aangename uren op één bureau gedeeld heb, en Jeroen voor zijn bereidheid om een eerste versie van dit proefschrift van commentaar te voorzien. Aan allen dank voor de vele ernstige en minder ernstige discussies over onderwerpen allerhande.

Ook gaat mijn dank naar Ingrid, Rita en Ann op het secretariaat en Johan Buelens en de mensen van de systeemmanagementploeg.

Bij het einde van mijn studies wil ik mijn ouders even in de bloemetjes zetten. Samen met mijn schoonouders, broers en zussen hebben zij zich dikwijls afgevraagd waarmee ik op de universiteit zo druk bezig was. Ik hoop dat deze tekst hun nieuwsgierigheid kan stillen.

Ik kan niet onder woorden brengen wat ik verschuldigd ben aan Inge. Ons huwelijk en dit doctoraat zijn ongeveer gelijktijdig gestart. Met warme genegenheid heeft ze me de volledige periode intens gesteund. Onder andere als compensatie voor de slapeloze nachten tijdens mijn talrijke afwezigheden draag ik dit werk aan haar op.

Tenslotte is dit werk enkel tot stand kunnen komen met de financiële steun van het Nationaal Fonds voor Wetenschappelijk Onderzoek en de ESPRIT BRA 3280 en 6632 projecten van de Europese Unie.

Abstract

Antenna arrays sample propagating waves at multiple locations. They are employed *e.g.* in radar, sonar and wireless communication systems because of their capability of spatial selectivity and localization of radiating sources. Current model-based algorithms make use of computationally demanding orthogonal matrix decompositions such as the singular value decomposition (SVD). On the other hand the data rates are often extremely high. Therefore, real-time execution of complex algorithms often requires parallel computing. We study the simultaneous design of new algorithms and parallel architectures for subspace tracking, for robust adaptive beamforming and for direction finding of narrow-band and wide-band sources. By structuring all recursive algorithms in a similar way, they can be mapped efficiently onto the Jacobi architecture, which was originally developed for SVD updating. The numerical and architectural aspects of this algorithm are improved by the use of a minimal parameterization of the orthogonal matrix of short singular vectors. Finally, a new Fourier-based linear model for direction finding in colored ambient noise fields is proposed.

Abstract

Roosterantennes bemonsteren propagerende golven op meerdere plaatsen. Ze worden o.a. in radar-, sonar- en radiocommunicatiesystemen gebruikt omwille van de mogelijkheid tot ruimtelijke selectiviteit en plaatsbepaling van signaalbronnen. De huidige modelgebaseerde algoritmen maken gebruik van rekenintensieve orthogonale matrixdecomposities zoals de singuliere waarden ontbinding (SWO). Anderszijds zijn de datasnelheden vaak heel hoog. Daarom vraagt uitvoering in reële tijd van deze complexe algoritmen vaak het gebruik van parallele computers. We bestuderen het gelijktijdig ontwerp van nieuwe algoritmen en parallele architecturen voor recursieve schatting van deelruimten, voor robuuste adaptieve straalvormers en voor richtingshoekbepaling van zowel smalbandige als breedbandige signalen. Door alle recursieve algoritmen op een gelijkaardige manier te structureren, kunnen ze efficiënt geïmplementeerd worden op de Jacobi architectuur die oorspronkelijk ontwikkeld werd voor recursieve SWO. De numerieke en architecturale aspecten van dit algoritme worden verbeterd door het gebruik van een minimale parameterisatie van de orthogonale matrix die de korte singuliere vectoren bevat. Tot slot wordt een nieuw Fourier-gebaseerd lineair model voorgesteld voor richtingshoekbepaling in gecorreleerde omgevingsruis.

Glossary

Symbols

$\begin{bmatrix} \cdot \\ \cdot \\ \cdot \end{bmatrix}$: general matrix or vector
$\begin{bmatrix} \cdot \\ \cdot \\ \cdot \end{bmatrix}^T$: transpose of a matrix or vector
$\begin{bmatrix} \cdot \\ \cdot \\ \cdot \end{bmatrix}^*$: conjugated matrix or vector
$\begin{bmatrix} \cdot \\ \cdot \\ \cdot \end{bmatrix}^H$: Hermitian transpose of a matrix or vector
$\ \cdot\ $: Frobenius norm of a matrix or vector
x_i	: i th element of vector x
X_{ij}	: element of matrix X on row i and column j
$\text{vec}(X)$: column vector obtained by stacking the columns of X
$\det(X)$: determinant of matrix X
$\text{tr}(X)$: trace of matrix X
$X^{1/2}$: Cholesky factor of square matrix X
Q^{ij}	: matrix embedding of Givens rotation operating on rows (columns) i and j
I_n	: identity matrix of size $n \times n$
$O_{m \times n}$: zero matrix of size $m \times n$
$1_{m \times n}$: constant matrix of ones of size $m \times n$
$\mathcal{O}(x)$: order of x operations
\propto	: proportional to
\otimes	: Kronecker product
\odot	: Khatri-Rao product
\star	: don't care value
j	: $\sqrt{-1}$
$x_{[k]}$: vector x sampled at sampling time k

Acronyms

1-D	:	one dimensional
2-D	:	two dimensional
ACMP	:	algebraically coupled matrix pencils algorithm
AOA	:	angle of arrival
ARMA	:	autoregressive moving average
CORDIC	:	coordinate rotation digital computer
CRB	:	Cramér-Rao bound
DF	:	direction finding
DG	:	dependence graph
DOA	:	direction of arrival
ESPRIT	:	estimation of signal parameters by rotational invariance techniques
EVD	:	eigenvalue decomposition
FDMA	:	frequency division multiple access
FIR	:	finite impulse response
GSC	:	generalized sidelobe canceler
GSM	:	global system for mobile communications
GSD	:	generalized Schur decomposition
LCMV	:	linearly constrained minimum variance beamformer
LMI	:	linear matrix inequality
MEMP	:	matrix enhancement and matrix pencil algorithm
ML	:	maximum likelihood
MSE	:	mean square error
MUSIC	:	multiple signal classification
QRD	:	QR decomposition
RLS	:	recursive least squares
RMSE	:	root mean square error
SFG	:	signal flow graph
SINR	:	signal to interference and noise ratio
SNR	:	signal to noise ratio
SOI	:	signal of interest
SVD	:	singular value decomposition
TDMA	:	time division multiple access
TDOA	:	time difference of arrival
TLS	:	total least squares
ULA	:	uniform linear array
VLSI	:	very large scale integration
WSF	:	weighted subspace fitting

Contents

Voorwoord	iii
Abstract	v
Glossary	vii
Contents	ix
Nederlandse Samenvatting	xiii
1 Introduction	1
1.1 Motivation	1
1.2 General survey	8
1.3 Chapter overview	11
2 Concepts and Tools	17
2.1 Narrow-band data model	17
2.2 Beamforming	22
2.2.1 LCMV beamforming	24
2.2.2 Recursive LCMV beamforming	27
2.2.3 Parallel mapping	31
2.3 Direction finding	36
2.3.1 Subspace algorithms	36
2.3.2 SVD updating	40
2.4 Conclusion	48
3 Factored Jacobi SVD Updating	49
3.1 Error accumulation	50
3.2 Orthogonal matrix - vector product	52
3.3 Updating the angles	56

3.4	A modified array for SVD updating	60
3.5	Conclusion	66
4	Factored Spherical Subspace Tracking	69
4.1	Spherical subspace tracking	70
4.2	Spherical SVD updating algorithm	71
4.3	Factored spherical SVD updating	79
4.4	A linear array	82
4.5	A planar array	86
4.6	Conclusion	96
5	Robust Adaptive LCMV Beamforming	97
5.1	Signal leakage	98
5.2	An adjustable constraint	100
5.3	Simulation results	110
5.4	Mapping onto the Jacobi architecture	112
5.5	Conclusion	113
6	2-D Harmonic Retrieval	115
6.1	Introduction	116
6.2	Data model	118
6.3	The Algebraically Coupled Matrix Pencil algorithm	119
6.3.1	Noiseless data	119
6.3.2	Shared frequencies	122
6.3.3	Noisy data	125
6.4	Simulations	126
6.5	Conclusion	130
	Appendices	132
6.A	Estimation of the amplitudes	132
6.B	Cramér-Rao bound	132
6.C	A modified MEMP algorithm	134
7	State Space Direction Finding for Wide-Band Emitters	137
7.1	Introduction	138
7.2	Data model	140
7.3	State space direction finding	143
7.3.1	Input-output equations	144
7.3.2	An instrumental variable method	145
7.3.3	Estimating the TDOAs	147

7.3.4	Relation doublets versus emitters	148
7.3.5	Basic algorithm	149
7.4	Parallel and adaptive wide-band direction finding	152
7.5	Simulations	157
7.5.1	Non-recursive direction finding	158
7.5.2	Recursive wide-band direction finding	160
7.6	Conclusion	161
8	Direction Finding in Unknown Ambient Noise Fields	163
8.1	Introduction	164
8.2	Modeling the noise field	166
8.3	Identifiability	169
8.4	Maximum likelihood and subspace direction finding	173
8.5	Simulations	181
8.6	Conclusion	184
9	Conclusion and Open Problems	187
9.1	Overview of the contributions	187
9.2	Suggestions for further research	189

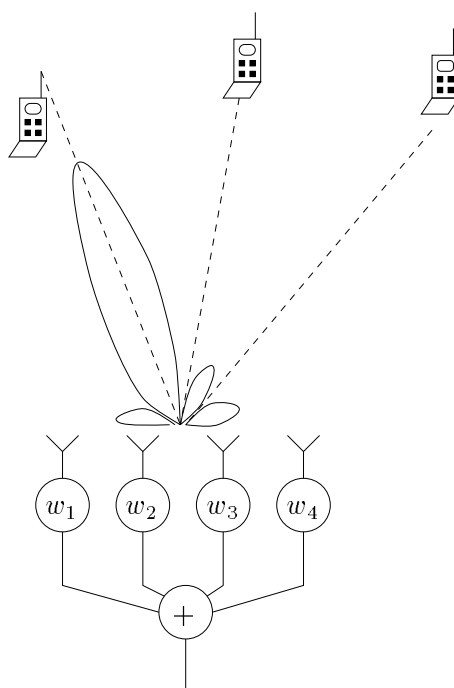
Algoritmen en architecturen voor adaptieve signaalverwerking van roostersensoren

Nederlandse samenvatting

Inleiding

Het toepassingsgebied van dit proefschrift is de verwerking van signalen afkomstig van roosterantennes of roostersensoren. Een roostersensor is een verzameling van sensoren die op meerdere plaatsen een golffront bemonsteren. Het gebruik van roostersensoren biedt tal van nieuwe mogelijkheden ten opzichte van enkelvoudige sensoren. Ze kunnen ruwweg opgedeeld worden in twee categorieën. Een eerste categorie beoogt spatiëel filteren. Door de uitgangen van de sensoren lineair te combineren wordt de ontvangst van de roosterantenne richtingsgevoelig. Deze structuur wordt een straalvormer (*E. beamformer*) genoemd en is afgebeeld in Figuur 0.1. De richtingsgevoeligheid wordt bepaald door de keuze van de filtercoëfficiënten. In vele applicaties, zoals satellietcommunicatie, wordt er gestreefd naar een maximale ontvangst uit de richting van het gewenste signaal terwijl signalen uit andere richtingen zoveel mogelijk onderdrukt worden. In andere toepassingen is de positie van ongewenste interferenties gekend, en dient er een nul geplaatst te worden in het ontvangstopatroon. Het verband met klassieke banddoorlaat FIR filters in het tijdsdomein ligt voor de hand.

De tweede categorie beoogt karakterisatie van de propagerende golven. Iedere puntbron wordt gekenmerkt door een beperkt aantal parameters,



Figuur 0.1: Een straalvormer in een cellulair mobilofoniesysteem. Drie mobiele zenders met verschillende locaties communiceren simultaan op dezelfde frequentie met het basisstation. De coëfficiënten w_i bepalen de richtingsgevoeligheid van de roosterantenne van het basisstation. Hier is de roosterantenne gericht naar de linkse mobiele zender. De twee overige signalen worden volledig onderdrukt.

zoals zijn positie (hoek en afstand) en het uitgezonden signaalvermogen. Om deze signaalparameters te kunnen schatten is het gebruik van roosterantennes noodzakelijk. Men kan geen informatie over de positie van een signaalbron afleiden als het golffront niet op meerdere plaatsen bemonsterd wordt.

Het domein van digitale signaalverwerking voor roosterantennes heeft sinds de jaren tachtig een sterke groei gekend. Veel nieuwe filter- en parameterestimatietechnieken zijn voorgesteld. Alle algoritmen hebben gemeen dat ze sterk steunen op numerieke lineaire algebra. Lineaire algebra vormt het natuurlijke kader om de problemen wiskundig te formuleren. Signalen als functie van de tijd kunnen voorgesteld worden door

vectoren. Ook de uitgangen van de roosterantenne op een bepaald ogenblik vormen een vector. De algoritmen manipuleren dus logischerwijze vectoren en matrices. Uit de onderliggende geometrie van het datamodel volgt dat veel informatie vervat is in bepaalde invariante deelruimtes. Matrixdecomposities spelen dan ook een belangrijke rol. De twee meest courante matrixdecomposities zijn de QR decompositie (QRD) en de singuliere waardenontbinding (SWO). Het domein van de numerieke lineaire algebra biedt bovendien een rijk gamma robuuste algoritmen om onder andere matrixdecomposities te berekenen.

De prijs voor de verhoogde performantie van deze signaalverwerkingsalgoritmen is hun relatief hoge rekenkost. Per tijdsstap bedraagt de rekencomplexiteit typisch $\mathcal{O}(M^2)$ waarbij M het aantal sensoren voorstelt. Daarbij komt dat de datadoorvoersnelheden in typische antennetoepassingen zoals radar-, sonar- of communicatiesystemen heel hoog liggen, in de orde grootte van 10 kbit/s tot 1Mbit/s.

De combinatie van hoge bitsnelheden en een aanzienlijke rekenkost maakt uitvoering in reële tijd problematisch, zelfs met de huidige generatie van snelle processoren. Soms kunnen applicatie-specifieke processoren een oplossing bieden. Als ook dat niet het geval is, kan de benodigde rekenkracht enkel geleverd worden door parallelle computers. Hierbij stellen zich opnieuw een aantal uitdagingen. Uitvoering op parallelle computers versnelt de berekening enkel als er een goede afstemming is tussen het algoritme en de architectuur van de parallelle machine. Er moet een evenwicht heersen tussen berekening, communicatie tussen de verschillende processoren onderling en tussen processoren en geheugen.

Een belangrijk aandachtspunt van deze thesis is de afleiding van optimale architecturen voor uitvoering in reële tijd van de geavanceerde signaalverwerkingsalgoritmen. De motivatie hiervoor is dat simultaan ontwerp van algoritme en parallelle architectuur het optimale compromis tussen de verschillende systeemcomponenten (rekenmodules, geheugen en invoer/uitvoer) het dichtst kan benaderen. Omwille van haar kracht en bevattelijke visualisatie zal de grafische methode gebaseerd op signaalstroomgrafes (*E. signal flow graphs*) hierbij gebruikt worden. Een signaalstroomgrafe is een gerichte grafe waarin de knopen corresponderen met de berekeningen in het algoritme en de pijlen de uitgewisselde signalen voorstellen.

Algemeen overzicht

Twee rode draden lopen door dit proefschrift. Een eerste rode draad is de ontwikkeling van nieuwe algoritmen voor de verwerking van antennesignalen. De nadruk ligt op recursieve algoritmen die geschikt zijn voor uitvoering in reële tijd. Zowel spatiaal filteren als invalshoekschatting worden behandeld. Veel algoritmen in de literatuur zijn gebaseerd op niet-lineaire optimaliseringstechnieken. Deze zijn echter moeilijk implementeerbaar op een parallelle computer. Daarom beperken we ons zoveel mogelijk tot matrixdecomposities in de keuze van bouwblokken voor recursieve algoritmen.

De tweede rode draad is toepassingsgerichte parallelle architecturen. Het is echter niet de bedoeling om per algoritme een totaal verschillende architectuur te poneren. Onze aanpak bestaat erin voor alle toepassingen structureel gelijkaardige algoritmen te ontwikkelen, die dan efficiënt implementeerbaar zijn op eenzelfde architectuur. Deze raamarchitectuur is de Jacobi architectuur die oorspronkelijk ontwikkeld werd voor recursieve singuliere waardenontbinding [66, 67]. De structuur van dit algoritme is een opeenvolging van matrix-vector vermenigvuldigingen, recursieve QRD en tweezijdige orthogonale transformaties. Door de signaalverwerkingsalgoritmen ook op die manier te formuleren, kan dezelfde Jacobi architectuur herbruikt worden. Bovendien wordt deze architectuur aantrekkelijker om een VLSI implementatie te ontwikkelen. Aan dit ontwerp wordt momenteel gewerkt [24, 118].

Hoofdstuk 2. Concepten en bouwblokken

Het proefschrift steunt op drie peilers. Een eerste peiler is digitale signaalverwerking van meervoudige signalen met de nadruk op roosterantennes. Dit is het toepassingsdomein. Een tweede peiler is numerieke lineaire algebra. Dit domein vormt het natuurlijke kader voor de wiskundige probleemstelling en reikt een veelvoud van algoritmische bouwblokken aan. Het derde domein is grafische methodologieën voor simultaan ontwerp van algoritmen en architecturen.

Dit inleidend hoofdstuk bevat het nodige achtergrondmateriaal uit elk van deze drie domeinen. De opbouw van het hoofdstuk volgt de ontwikkeling van de signaalverwerkingsapplicaties. De bespreking van de nodige concepten uit lineaire algebra en grafische ontwerpmethodologieën gebeurt aan de hand van de toepassingen.

Eerst wordt het datamodel afgeleid. We beschouwen een basisband-model voor een scenario waarbij de D signaalbronnen puntbronnen in het verre veld van de roosterantenne zijn. De uitgangen van de M antennes in het rooster op N bemonsteringstijden vormen een matrix $X \in \mathbb{C}^{M \times N}$. Elke (incoherente) puntbron geeft aanleiding tot een rang-1 bijdrage in X . Voor D signalen voldoet de datamatrix aan de vergelijking

$$X = A \cdot S + W,$$

waarbij $A \in \mathbb{C}^{M \times D}$ de roosterwinstmatrix (*E. array gain matrix*) wordt genoemd, $S \in \mathbb{C}^{D \times N}$ de invallende basisbandsignalen bevat en $W \in \mathbb{C}^{M \times N}$ bestaat uit de additieve ruis. De informatie omtrent de invalshoeken van de signalen zit vervat in de kolommen van de roosterwinstmatrix. Uit dit elementair datamodel volgt onmiddellijk het belang dat matrixdecomposities zullen spelen in de opbouw van de algoritmen.

Het is bekend dat de bepaling van de coëfficiënten van een optimale straalformer kan geformuleerd worden als een kleinste kwadraten probleem met lineaire beperkingen.

$$\min_w w^H \cdot (X \cdot X^H) \cdot w \quad \text{waarbij} \quad C^H \cdot w = m.$$

De matrix $C \in \mathbb{C}^{M \times K}$ bevat de beperkingen en $m \in \mathbb{C}^K$ bevat de opgelegde winstvector. Een typische beperking legt de waarde van de ontvangst in de richting van het signaal θ_s vast.

$$a(\theta_s)^H \cdot w = \mu.$$

De vector $a(\theta_s) \in \mathbb{C}^M$ wordt de richtingsvector genoemd en bevat de amplitude en fase van een signaal uit richting θ_s voor elk van de sensoren. Kleinste kwadraten problemen kunnen elegant opgelost worden door gebruik te maken van de QR decompositie van de data matrix. De gewichtsvector w volgt dan uit een bovendriehoekig stelsel lineaire vergelijkingen.

In een tijdsvariante signaal- en ruisomgeving moet de straalvormer zich voortdurend aanpassen aan de veranderingen in zijn omgeving. De QRD oplossingsmethode leent zich uitstekend tot recursieve berekening. De nieuwe gewichtsvector kan berekend worden in $\mathcal{O}(M^2)$ operaties met de Givens rotatiemethode voor recursieve QRD [33]. Deze methode stelt een sequentie van 2×2 Givens rotaties op die de inkomende datavector gradueel nul maken. Deze Givens methode is erg geschikt voor parallelle

implementatie omdat elke rotatie lokaal berekend kan worden uit 2×2 submatrices. Een driehoekige signaalstroomgrafe (SSG) voor recursieve QRD is voorgesteld door Gentleman en Kung [31]. De complete signaalstroomgrafe van de straalvormer met lineaire beperkingen bestaat uit een rechthoekige grafe voor matrix-vector vermenigvuldiging gevolgd door de driehoekige grafe voor recursieve QRD. De topologie van deze grafe is eenvoudig. Alle data-afhankelijkheden zijn lokaal. Een knoop wisselt enkel signalen uit met zijn naaste burens. Bovendien is het afhankelijkheidspatroon identiek van knoop tot knoop. Daardoor is de omvorming van deze signaalstroomgrafe in een planaire architectuur haast triviaal. De omvorming behelst de toewijzing van een processor en een uitvoeringstijd aan elk stuk van de berekening. Omwille van haar uniformiteit leidt deze grafe bij maximale pijplijning tot een systolisch rooster. Een systolisch rooster is een synchrone parallelle architectuur waarin alle processoren lokale en regelmatige interconnecties hebben. Omwille van hun regelmaat zijn ze uitermate geschikt voor VLSI implementatie van applicatie-specifieke processoren.

Naast spatiale filtering is ook scheiding en invalshoekschatting van meerdere signalen die invallen op de roosterantenne een belangrijk probleem. Deelruimte-algoritmen zoals MUSIC [96] en ESPRIT [92] worden hiervoor meest gebruikt. Omdat ze modelgebaseerd zijn, kunnen ze een beduidend hogere resolutie bereiken dan niet-modelgebaseerde technieken zoals de spatiale Fouriertransformatie. Alle algoritmen uit deze klasse berekenen eerst een singuliere waardenontbinding van de datamatrix. Op die manier bepalen ze de signaalruimte $\mathcal{S} = \text{Bereik}\{A\}$ en onderdrukken ze de additieve ruis. De tweede stap bestaat uit het vinden van de invalshoeken die de geschatte deelruimte zo goed mogelijk verklaren. De algoritmen onderscheiden zich in deze stap. Het MUSIC algoritme gebruikt niet-lineaire optimalisering. Het heeft dus een hoge rekenkost en is niet erg geschikt voor parallellisatie. Het ESPRIT algoritme biedt een oplossing voor beide problemen door de invalshoeken te schatten aan de hand van een veralgemeend eigenwaardenprobleem. Dit algoritme is enkel toepasbaar als de roosterantenne kan opgedeeld worden in twee identieke verschoven subroosters. In de praktijk is dit veelal het geval.

Een recursief Jacobi SWO algoritme met bijhorende signaalstroomgrafe is ontwikkeld in [66, 67]. Deze grafe is een combinatie van de grafe voor de adaptieve straalvormer (matrix-vectorvermenigvuldiging en recursieve QRD) met een grafe voor sequenties van tweezijdige Givens rotaties. Deze SSG is moeilijk parallelliseerbaar, omdat ze een lange bidi-

rectionele keten van data-afhankelijkheden bevat. Een efficiënte pijplijning kan enkel door het opbreken van keten. In dit geval zijn standaard transformatietechnieken op grafen ontoereikend. Het algoritme zelf moet gewijzigd worden. De techniek van algoritmische transformaties die ontwikkeld werd voor recursieve SWO, is sindsdien ook succesvol gebleken in de afleiding van efficiënte architecturen voor andere recursieve signaalverwerkingsalgoritmen zoals recursieve kleinste kwadraten [65].

Hoofdstuk 3. Gefactoriseerde Jacobi SVD updating

Dit hoofdstuk bevat de eerste originele bijdrage van dit proefschrift, nl. een variant op het originele recursieve Jacobi SWO algoritme met betere numerieke eigenschappen en VLSI implementeerbaarheid.

Orthogonale matrixdecomposities zijn een belangrijke hoeksteen van moderne signaalverwerking, identificatie- en controletheorie, . . . Voorbeelden zijn de QR decompositie, vaak gebruikt in kleinste kwadratenproblemen, en de singuliere waardenontbinding, vaak gebruikt voor problemen in verband met matrices van lage rang. Bovendien is er een rijk gamma van veralgemeningen en varianten op deze decomposities.

Orthogonale decomposities zijn niet enkel conceptueel belangrijk. Ze zijn ook geliefd omwille van hun goede numerieke eigenschappen. Fouten in de berekening kunnen niet aangroeien. Nochtans zijn orthogonale technieken niet helemaal veilig. Als voorbeeld beschouwen we het Jacobi algoritme voor recursieve SWO [66]. In elke iteratie wordt de matrix van korte singuliere vectoren $V \in \mathbb{R}^{M \times M}$ vermenigvuldigd met een orthogonale updating matrix $\Phi \in \mathbb{R}^{M \times M}$

$$V_{[k+1]} \leftarrow V_{[k]} \cdot \Phi_{[k]}. \quad (0.1)$$

Accumulatie van afrondingsfouten in de opeenvolgende matrixvermenigvuldigingen leidt tot geleidelijk verlies van de orthogonaliteit van $V_{[k]}$.

Haast alle recursieve schema's voor orthogonale decomposities wapen zich hiertegen door periodiek de betrokken matrices te herorthogonaliseren. Courante technieken zijn gebaseerd op een Gram-Schmidt orthogonalisatie (QRD). Belangrijke nadelen zijn de beduidende rekenkost ($\mathcal{O}(M^3)$) en de moeilijke parallellisatie.

In dit hoofdstuk stellen we een alternatief voor. Een willekeurige orthogonale matrix $V \in \mathbb{R}^{M \times M}$ met positieve determinant wordt uniek

geparameteriseerd door een sequentie van Givens rotaties Q^{ij} , elk gekenmerkt door één rotatiehoek α^{ij}

$$V = \prod_{i=1}^{M-1} \prod_{j=i+1}^M Q^{ij}.$$

Zo'n Givens rotatie matrix Q^{ij} is een 2×2 rotatiematrix ingebed in een omvattende identiteitsmatrix. Door impliciet te rekenen met deze parameterisatie $\{Q^{ij}\}$ in plaats van met de matrix zelf, blijft $V_{[k]}$ noodzakelijkerwijze binnen de verzameling orthogonale matrices. Afrondingsfouten manifesteren zich enkel nog op de hoeken. Deze fouten worden teniet gedaan door de terugkoppeling in het recursieve SVD algoritme. Waar reorthogonalisatiemethodes de afwijking van orthogonaliteit enkel binnen de perken houden, garandeert deze minimale parameterisatie de orthogonaliteit in elke iteratie. We bestuderen nu de implementatie van dit idee voor het recursieve Jacobi SWO algoritme.

Er zijn twee bewerkingen waarbij de matrix van korte singuliere vectoren $V_{[k]}$ betrokken is. Een eerste bewerking is een matrix-vector vermenigvuldiging met de inkomende data vector.

$$\tilde{x}_{[k]}^T = x_{[k]}^T \cdot V_{[k-1]}.$$

Deze matrix-vector vermenigvuldiging wordt nu vervangen door een sequentie van $M(M-1)/2$ Givens rotaties. Hierdoor wordt de SSG gewijzigd. De SSG van een matrix-vectorvermenigvuldiging is een rechthoekige grafe waarbij elke knoop een scalaire vermenigvuldiging en een optelling uitvoerde. Dit wordt nu vervangen door een driehoekige grafe waarbij elke knoop een Givens rotatie uitvoert. De impact hiervan op een hardware implementatie is niet onbelangrijk. Het volledige recursieve Jacobi SWO algoritme bestaat nu exclusief uit eenzijdige en tweezijdige 2×2 rotaties. Voor snelle en accurate uitvoering van zulke rotaties is een CORDIC processor [143] de ideale hardware component. De vervanging van vermenigvuldigingen door rotaties maakt een parallelle Jacobi architectuur mogelijk die enkel bestaat uit CORDIC-gebaseerde processoren. Dit vereenvoudigt het ontwerp en de programmering van een toepassingsgerichte parallelle computer aanzienlijk.

De tweede en laatste operatie op de matrix $V_{[k]}$ is gegeven door vergelijking (0.1). De orthogonale matrix $\Phi_{[k]}$ is het product van $M-1$ rotaties op opeenvolgende kolommen

$$\Phi_{[k]} = \prod_{i=1}^{M-1} \Phi_{[k]}^{i|i+1}.$$

De originele bijdrage van dit hoofdstuk is een schema om de rotaties $Q_{[k]}^{i|j}$ rechtstreeks aan te passen zonder expliciete berekening van $V_{[k]}$. De $\Phi_{[k]}^{i|i+1}$ rotaties moeten geleidelijk in de sequentie van $Q_{[k]}^{i|j}$ verwerkt worden. De transformaties op de hoeken hangen af van de kolomindices van de twee interagerende rotaties. Drie types van rotaties zijn noodzakelijk.

1. Als de indexparen totaal verschillen, commuteren de rotaties en is er geen berekening.
2. Als de indexparen identiek zijn, is de samenstelling van de rotaties de rotatie over de som van de hoeken.
3. Als de indexparen één index gemeen hebben, moet een derde rotatie in beschouwing genomen worden om de volgorde van de indices te wijzigen.

De gecombineerde SSG die zowel de matrix-vector vermenigvuldiging als het updatingschema bevat, blijft driehoekig. De grafe bevat net als het oorspronkelijke Jacobi algoritme lange afhankelijkheidsketens. Daarom moet bij het pijplijnen opnieuw gebruik gemaakt worden van dezelfde algoritmische transformaties om tot een efficiënte parallelle (systolische) implementatie te komen.

Hoofdstuk 4. Een gefactoriseerde sferische deelruimtevolger

In dit hoofdstuk passen we de minimale parameterisatie van orthogonale matrices toe op een tweede algoritme voor recursieve deelruimteschatting, nl. de sferische deelruimte volger. Een variant op het originele algoritme wordt voorgesteld en twee systolische architecturen worden afgeleid.

Een belangrijke applicatie van recursieve SWO algoritmen is het volgen van een traag tijdsvariante deelruimte. Een voorbeeld is het schatten van invalshoeken van signalen met behulp van roosterantennes. Als de bronnen bewegen, moet het algoritme in staat zijn deze evolutie te volgen. Het recursieve Jacobi SWO algoritme berekent de SWO op elk ogenblik slechts benaderend. De reden hiervoor is dat per iteratie bij exacte berekening de rekenkost $\mathcal{O}(M^3)$ zou bedragen. In vele applicaties is deze kost te hoog. Ook de $\mathcal{O}(M^2)$ kost van het Jacobi algoritme kan nog problemen stellen. Daarom zijn verscheidene ruwere benaderingen voor recursieve SWO ontwikkeld. Het adaptieve sferische deelruimte

algoritme [22] verlaagt de rekenkost extreem tot $\mathcal{O}(M \cdot D)$. Dit wordt bereikt door twee technieken. Een eerste techniek bestaat erin om op elk ogenblik de singuliere waarden in de signaaldeelruimte en de ruisdeelruimte apart uit te middelen. Dit kan in applicaties waar de exacte kennis van de singuliere waarden zelf niet cruciaal is. Een voorbeeld is invalshoekschatting waarbij enkel de scheiding van signaal- en ruisdeelruimte cruciaal zijn. Een deelruimte waarvan de geassocieerde singuliere waarden uitgemiddeld zijn, wordt een sferische deelruimte genoemd omdat elke orthogonale basis voor deze deelruimte een basis van singuliere vectoren vormt. Een tweede techniek bestaat in het bijhouden van enkel de kleinste deelruimte, bvb. de signaaldeelruimte als $D < M$. Indien kennis van de ruisdeelruimte vereist is, kan die op elk ogenblik berekend worden als het orthogonale complement.

Het originele algoritme is afgeleid als een benaderend recursief eigenwaardenalgoritme. Om de vergelijking met het recursieve Jacobi SWO algoritme expliciet te maken, hebben we ervoor geopteerd om hier het algoritme opnieuw af te leiden als een benaderend SWO algoritme. Uit de afleiding blijkt duidelijk dat het sferische algoritme steunt op hetzelfde werkingsprincipe als het Jacobi algoritme om de deelruimte recursief te schatten. In elke iteratie wordt een orthogonale matrix die de deelruimte omspant, aangepast door 2×2 kolomrotaties. Daarom is er ook hier weer gevaar voor geleidelijk verlies van orthogonaliteit van deze matrix. We stellen daarom voor om dezelfde orthogonale parameterisatie met hoeken te gebruiken. Daardoor wordt het algoritme perfect numeriek stabiel.

Een nieuw aspect is dat nu enkel een deelmatrix van een orthogonale matrix moet geparаметeriseerd worden. Dit stelt geen probleem. Aangezien de berekening van de rotaties Q^{ij} kolomsgewijze gebeurt, volstaat het deze parameterisatie stop te zetten na D kolommen.

Bovendien tonen we aan dat het bewaren van een (parameterisatie van een) orthogonale basis voor de ruisruimte niet hoeft. Het volstaat in elke iteratie de projectie van de inkomende datavector op de ruisruimte te berekenen. In het nieuwe schema met hoeken kan dit zelfs veel eleganter dan in het oude schema waar de matrices expliciet bewaard worden. Daar vergt de berekening van de geprojecteerde vector een sequentie van twee matrix-vectorvermenigvuldigingen en een normalisatie. In de SSG geeft dit aanleiding tot tegengestelde datastromen. Met de parameterisatie is de berekening veel eenvoudiger en vergt geen tweezijdige datastroom. De rotatieknopen in de laatste kolom berekenen hun hoek door één coördinaat nul te maken.

De SSG bestaat opnieuw uit twee delen. Het bovenste trapezoïdaal gedeelte voert een matrix-vector vermenigvuldiging uit. Het onderste gedeelte is niet langer driehoekig. Eén rij van rotatieknopen volstaat. Vertrekkend vanuit deze nieuwe SSG worden twee systolische architecturen afgeleid. De eerste architectuur is een lineaire rij van $D + 1$ processoren en heeft een pijplijningsperiode van $2M - 1$ cycli. Deze architectuur kan eenvoudig afgeleid worden door alle knopen in eenzelfde kolom van de SSG aan dezelfde processor toe te wijzen. De uitvoeringstijd volgt dan uit de data-afhankelijkheden.

De tweede architectuur is planair. Elke knoop in de SSG wordt aan één enkele processor toegewezen. De pijplijning van deze architectuur is meer complex. Algoritmische transformaties zijn opnieuw onontbeerlijk om het kritische pad van lengte $\mathcal{O}(M)$ verder op te delen. Het eindresultaat is minder elegant dan voor het recursieve Jacobi SWO algoritme. De algoritmische transformaties introduceren nieuwe rotaties die in dit geval niet eenvoudig toegewezen kunnen worden aan bestaande knopen zonder de regelmaat en de localiteit van de SSG te verstoren. Dit is de prijs voor de verhoogde doorvoersnelheid (1 cyclus) van de planaire architectuur.

Hoofdstuk 5. Een robuuste adaptieve LCMV straalvormer

In dit hoofdstuk stellen we een robuuste adaptieve straalvormer voor. De parallele implementatie van dit algoritme kan heel efficiënt gebeuren op de Jacobi architectuur voor recursieve SWO.

Een belangrijk praktijkgericht probleem van adaptieve straalvormers is hun hoge gevoeligheid voor perturbaties van de elementen in de gelijkheidsbeperkingen. Vooral wanneer de signaal-ruishouding aan de ingang van de antennes hoog is, is de verslechtering van de signaal-ruisverhouding aan de uitgang aanzienlijk. Eén van deze vectoren – en vaak ook de enige – is noodzakelijkerwijze de richtingsvector van het gewenste signaal. In de praktijk zal de echte richtingsvector altijd wat afwijken van de richtingsvector gebruikt in de beperkingsmatrix door onnauwkeurigheden in de positionering van de sensoren, amplitude- en fazefouten in de antennes,...

Om deze vector zo accuraat mogelijk te kennen, stellen we voor om deze vector continu te schatten uit de data [10]. Dit kan bijvoorbeeld in communicatietoepassingen waar vaak een referentiesignaal voorhanden is. Op voorwaarde dat ruis- en eventuele interferentiesignalen ongecorrleerd

zijn met het referentiesignaal, is de kruiscorrelatievector van uitgangen en referentiesignaal evenredig met de echte richtingsvector. Mits de schatting voldoende nauwkeurig gebeurt, wordt de robuustheid van de straalvormer sterk verhoogd. Een tweede voordeel is dat de straalvormer in staat is om een bewegende bron te volgen zonder het traject vooraf te kennen.

De wiskundige formulering leidt tot een kleinste kwadratenprobleem met een tijdsafhankelijke beperking en eventueel nog constante beperkingen. In het inleidend hoofdstuk hebben we al een parallelle architectuur besproken voor adaptieve straalvorming. Het bovengedeelte is een matrix die de beperkingen implementeert en het ondergedeelte is een adaptieve driehoekige kleinste-kwadraten-processor. Bovenop deze structuur moet nu een adaptatieschema geïmplementeerd worden. Het adaptatieschema zorgt ervoor dat de geschatte kruiscorrelatievector tussen uitgangen en referentie altijd in de laatste kolom behouden blijft. Dit kan opnieuw gebeuren door tweezijdige rotaties. De kolomrotaties zorgen voor de aligenering van de kolommen. Ze moeten ook toegepast worden op de matrix R van de QR decompositie. Hierdoor verliest deze matrix zijn driehoeksvorm. Rijrotaties zijn noodzakelijk om de ingevulde elementen onder de diagonaal weer weg te werken.

Dit algoritme is structureel haast identiek aan het recursieve Jacobi SWO algoritme. Het bestaat ook uit een sequentie van een matrix-vectorvermenigvuldiging, een recursieve QRD stap en tweezijdige orthogonale rotaties. Daarom kan het onmiddellijk afgebeeld worden op de Jacobi architectuur. Het verdient ook de aandacht dat de factorisatie van orthogonale matrices uit vorige hoofdstukken ook hier haar nut kan bewijzen.

Hoofdstuk 6. Schatting van tweedimensionale spectraallijnen

In de laatste hoofdstukken van dit proefschrift gaat de aandacht naar het schatten van invalshoeken. In dit hoofdstuk stellen we een nieuwe efficiënte methode voor voor het schatten van invalshoeken van smalbandige signalen waarbij hun draagfrequenties niet gekend en verschillend kunnen zijn. Dit is een toepassing van tweedimensionale (2-D) spectraalschatting. De koppels (ϕ_i, ψ_i) zijn gerelateerd tot de invalshoek θ_i en de draagfrequentie van het signaal f_i

$$\begin{aligned}\phi_i &= \exp(j2\pi f_i \Delta \sin(\theta_i)/c) \\ \psi_i &= \exp(j2\pi f_i T_s).\end{aligned}$$

De efficiëntie van het voorgestelde algoritme is te danken aan twee eigenschappen. Een eerste eigenschap is de separabiliteit van het datamodel. Hierdoor kan het 2-D estimatieprobleem opgelost worden als twee 1-D estimatieproblemen. Een complicatie is wel het combineren van de twee verzamelingen van 1-D schattingen. We tonen aan dat extra berekeningen vermeden kunnen worden door een juiste keuze van transformaties. Een tweede eigenschap is de Vandermonde structuur van de datamatrix. Hierdoor kunnen efficiënte matrixdecomposities gebruikt worden om de frequenties te schatten.

De datamatrix $Z \in \mathbb{C}^{N \times M}$ is sterk gestructureerd

$$Z = X_N \cdot A \cdot Y_M^T + W.$$

De matrices $X_N \in \mathbb{C}^{N \times D}$ en $Y_M \in \mathbb{C}^{M \times D}$ zijn Vandermonde matrices in ϕ_i en ψ_i respectievelijk. $A \in \mathbb{C}^{D \times D}$ is een diagonale matrix met de amplitude en fase van elke component. De matrix $W \in \mathbb{C}^{N \times M}$ bevat additieve ruis.

Eerst beschouwen we het geval waarin alle horizontale respectievelijk verticale frequenties verschillen. In het ruisloze geval is de matrix Z van lage rang (D). Met ruis is de eerste stap van het algoritme een SWO om de kolom- en rijruimten te schatten. Omwille van de Vandermondestructuur van X_N en Y_M kan de stap van deelruimte naar frequentieschattingen opnieuw gebeuren door matrixdecomposities. Het ESPRIT algoritme [92] is hiervoor de aangewezen kandidaat. Uitvoering van dit algoritme volgens de horizontale en verticale richting geeft twee verzamelingen schattingen. Het paren van de juiste frequenties vergt in principe $\mathcal{O}(D^2)$ bewerkingen. Deze extra berekeningen kunnen vermeden worden door een algebraïsche koppelingmethode [115]. De eigentransformaties van zowel het horizontale als verticale eigenwaardenprobleem kunnen identiek gemaakt worden door keuze van bepaalde deelmatrices van Z . De winst in berekeningstijd is beduidend. Er moet maar één eigenwaardenprobleem meer opgelost worden. Toepassing van dezelfde eigentransformaties op de tweede matrix geeft de complementaire frequenties in de juiste volgorde.

In het geval dat een bepaalde component in meerdere frequentieparen voorkomt, faalt het beschreven algoritme. In de corresponderende Vandermondematrix zijn twee kolommen identiek. Hierdoor zakt de rang tot $D-1$. Dit probleem kan opgelost worden door uitmiddeling. Dit kan door op basis van de oorspronkelijke data een grote matrix J te construeren die opnieuw van volle rang D is. Bovendien moet J nog altijd separabel zijn en een Vandermonde-achtige structuur behouden. In het hoofdstuk

construeren we een 'dubbele' blokhankelmatrix die aan beide voorwaarden voldoet. Deze techniek heeft de eigenschap dat de dimensies van de matrix enkel vermenigvuldigd worden met de maximale meervoudigheid van een frequentiecomponent. Dit is een verder voordeel ten opzichte van bestaande methodes.

Hoofdstuk 7. Localisatie van breedbandige signalen met toestandsruimtemodellen

In dit hoofdstuk bestuderen we een verdere uitbreiding van het datamodel naar breedbandige stochastische signalen. We stellen een klasse van deelruimte-algoritmen voor die gestoeld zijn op toestandsruimtemodellen van de breedbandige signalen. Deze aanpak laat toe om opnieuw op een vrij eenvoudige manier een efficiënt parallel recursief algoritme te ontwikkelen.

Over het breedbandige probleem is veel minder literatuur verschenen dan over het smalbandige. Nochtans zijn er ook belangrijke toepassingen zoals localisatie van akoestische bronnen. De meeste methodes steunen op de opsplitsing van het breedbandige signaal in meerdere smalbandige signalen. De schattingen van al deze deelproblemen worden dan gecombineerd. Een alternatief is de breedbandige signalen te modelleren met behulp van toestandsmodellen geëxciteerd door onafhankelijke witte ruis-signalen. De signalen worden dan ontbonden in modes in plaats van frequenties. Op voorwaarde dat de bemonstering uniform is en dat de roosterantenne uit twee verschoven deelroosters bestaat, kan voor elk van deze modes de invalshoek geschat worden met behulp van matrixdecomposities.

In een eerste stap van het deelruimte-algoritme worden blokhankelmatrices geconstrueerd met de uitgangen van de twee deelroosters. Elk van deze datamatrices is een som van een term die lineair is in de toestand van het totale systeem, een term die bijdragen van de ingangen bevat en een ruisterm. Enkel de toestandsterm bevat informatie over de ligging van de systeempolen en de locatie van de bronnen. Daarom moeten de ingangs- en ruistermen afgescheiden worden. Uit het stochastisch karakter van de signalen volgt dat dit kan gebeuren door een projectie op verleden uitgangen. Met de geprojecteerde matrices kunnen, precies zoals in het vorige hoofdstuk, twee algebraïsch gekoppelde eigenwaardenproblemen opgesteld worden. Uit de eigenwaardeparen kan tenslotte de invalshoek van elk van de breedbandige bronnen bepaald worden.

Dit deelruimte-algoritme bestaat uit een opeenvolging van een projectiestap en meerdere eigenwaardedecomposities op vier matrices. Omwille van numerieke stabiliteit en precisie is het goed de projectiestap uit te voeren via een QR decompositie. Eigenwaarden kunnen ook berekend worden door een andere orthogonale transformatie, nl. de Schur decompositie. Omdat het hier in feite om veralgemeende eigenwaardenproblemen met matrixparen gaat, is de veralgemeende Schurdecompositie nodig. Uiteindelijk kan het volledige algoritme opgebouwd worden met enkel orthogonale transformaties.

Het belang van deze opbouw schuilt in de afleiding van een recursief algoritme. Het uiteindelijke resultaat is een Jacobi schema dat simultaan rij- en kolomrotaties op de vier matrices uitvoert. De datastroom blijft echter identiek aan het recursieve Jacobi SWO algoritme. Daarom vergt de implementatie op de Jacobi architectuur enkel herprogrammering van de processoren.

Hoofdstuk 8. Een parametrische methode voor localisatie van bronnen in onbekende omgevingsruis

Het laatste hoofdstuk introduceert een nieuwe aanpak voor een belangrijk praktijkgericht probleem. Alle hoge-resolutiemethoden voor het schatten van invalshoeken van smalbandige signalen gaan uit van de veronderstelling dat de ruis van sensor tot sensor ongecorrleerd is. In de praktijk is dat nooit zo. Als alle sensoren dezelfde karakteristieken hebben, is dit een goed model voor ruis intern gegenereerd in de sensoren. Daartegenover staat dat omgevingsruis die invalt op de roosterantenne altijd gecorrleerd is.

Men kan corrigeren voor de ruisrelatie door de signalen eerst te filteren met de inverse van de ruisrelatiematrix. Daarom zijn methodes om goede schattingen van de ruisrelatiematrix te vinden van groot belang. Onze aanpak is modelgebaseerd. We ontwikkelen een model voor de ruisrelatiematrix dat gebaseerd is op een Fourieranalyse van het ruisvermogen in functie van de invalshoek. Volgens dit model is de ruisrelatiematrix een lineaire combinatie van een aantal basismatrices met onbekende coëfficiënten. De basismatrices corresponderen met een term in de Fourierexpansie. Ze kunnen vooraf berekend worden op voorwaarde dat de transfertfunctie van de roosterantenne gekend is. De lineaire parameters zijn de Fouriercoëfficiënten. Dit model is aantrekkelijk omwille van

drie aspecten. Ten eerste is het fysisch betekenisvol. De coëfficiënten zijn niet zomaar fittingparameters, maar bevatten informatie over de plaatsafhankelijkheid van de ruis. Ten tweede is dit model toepasbaar op antennes met een willekeurige geometrie. De meeste ruismodellen in de literatuur zijn enkel geldig voor regelmatige roosterantennes. Tenslotte is er de eenvoud van de parameterisatie. De elementen van de ruiscorrelatiematrix zijn lineaire functies van de parameters. Dit vergemakkelijkt de algoritmen.

Een belangrijk aspect is de uniciteit van de parameterisatie. Het is echter moeilijk onder- of bovengrenzen op het aantal uniek identificeerbare Fouriercoëfficiënten te geven. Uit een studie van de topologie van het probleem volgt dat het aantal oplossingen gelijk is aan het aantal matrices van lage rang die aan twee lineaire matrixongelijkheden voldoen. Het is bekend dat dit probleem in zijn algemeenheid NP-compleet is [16].

Om de parameters te schatten worden twee methodes besproken. De eerste aanpak steunt op de theorie van maximale waarschijnlijkheid (*E. maximum likelihood*). Uit de afleiding volgt dat de optimalisatie van ruis- en signaalparameters niet gescheiden kan worden. De oplossing vergt daarom een niet-lineaire optimalisatie in een hoogdimensionale ruimte. Dit is rekentechnisch niet interessant. Daarom opteren we voor een tweede lichtjes suboptimale aanpak. Eerst worden de ruisparameters via niet-lineaire optimalisatie geschat. Met deze optimale ruiscorrelatiematrix worden de uitgangen gefilterd. Tenslotte kunnen de signaalparameters zoals invalshoek en signaalvermogen geschat worden aan de hand van de klassieke hoge-resolutie-algoritmen.

Het nieuwe optimalisatiecriterium volgt uit de theorie van maximale waarschijnlijkheid voor een vereenvoudigd datamodel waarin de structuur van de richtingsvectoren buiten beschouwing gelaten wordt. Zolang het aantal bemonsteringen groot genoeg is, blijft het verschil met optimale maximale waarschijnlijkheid klein. Het criterium is een maat voor de gelijkheid van de kleinste eigenwaarden van de ruiscorrelatiematrix. Het is de verhouding van het geometrisch tot het arithmetisch gemiddelde van deze eigenwaarden. Deze verhouding komt ook voor in detectiecriteria voor de bepaling van het aantal bronnen zoals MDL en AIC [147]. Aan de hand van simulaties wordt aangetoond dat dit algoritme een goede performantie en robuustheid bezit.

Hoofdstuk 9. Conclusies en open vragen

In dit proefschrift ging de aandacht naar recursieve algoritmen voor digitale signaalverwerking van roostersignalen in reële tijd. Numerieke aspecten waren zeer belangrijk. Omwille van de combinatie van aanzienlijke berekeningskost en hoge datasnelheden hebben we de implementatie van deze algoritmen op parallele architecturen bestudeerd.

De (recursieve) singuliere waardenontbinding vormt de ruggegraat van heel wat voorgestelde algoritmen. In een eerste luik hebben we een numeriek meer betrouwbare variant op het Jacobi algoritme voor recursieve SWO afgeleid. Dit algoritme bood inspiratie voor de afleiding en parallelisatie van volledige signaalverwerkingsalgoritmen zoals robuuste straalvormers en localisatie-algoritmen. Voor invalshoekschatting hebben we algoritmen voor smalbandige signalen uitgebreid naar het geval van breedbandige signalen. Tenslotte is ook een niet-recursief algoritme voorgesteld voor invalshoekschatting in omgevingen met onbekende gekleurde ruis.

Tot slot sommen we enkele grote lijnen op voor verder onderzoek. De grote krachtlijn hierbij is een verdere oriëntering naar de specifieke eigenschappen van de toepassing. Dit zal een belangrijke verdere optimalisering van de performantie van de algoritmen toelaten. De algoritmen zijn nu vooral gegroeid vanuit de theoretische studie van simultaan ontwerp van algoritmen en architecturen en vanuit de digitale signaalverwerkingstheorie. In de praktijk betekent dit dat een aantal veronderstellingen waarop de algoritmen steunen maar gedeeltelijk voldaan zijn. Een voorbeeld zijn de cellulaire mobilofoniesystemen. Het signaal van de mobiele zender bereikt het basisstation via meerdere paden met verschillende sterktes en fazeverschuivingen. Door dit coherente multipad verschijnsel verliest de antenneresponsievector zijn spatiale structuur en kan er geen hoek meer eenduidig mee geassocieerd worden. Het zenden vanuit het basisstation naar de mobiele zender stelt ook specifieke moeilijkheden. Het zend- en ontvangstkanaal verschillen in frequentie (45MHz in GSM). Hierdoor zijn er ook verschillen in de responsievectoren. Daarom kan de ontvangstresponsievector niet zomaar gebruikt worden voor de berekening van de zendgewichtsvectoren.

Op het domein van de mobiele telefonie-applicatie is er nood aan een algemeen aanvaard, realistisch, maar mathematisch hanteerbaar model voor het propagatiekanaal van mobiele zender naar roosterantenne. Voorlopig zijn heel weinig metingen beschreven [76]. Men kan het model wel theoretisch uitbreiden vertrekkend van bestaande modellen voor één

antenne.

Op het domein van de algoritmen zijn er twee grote lijnen. Ten eerste is een statistische gevoeligheidsanalyse van de algoritmen noodzakelijk. Kleine verschillen in de transfertfunctie van de ontvangstapparatuur of onnauwkeurigheden in de plaats van de antenne-elementen zorgen voor afwijkingen tussen het model en het werkelijke systeem. In dit proefschrift lag de nadruk op simultaan ontwerp van algoritmen en architecturen. Testen van de algoritmen gebeurde enkel op basis van simulaties. Een nadeel van deze – zij het algemeen aanvaarde – methode is de moeilijke veralgemeenbaarheid van de resultaten. Een asymptotische analyse in eerste orde heeft een algemenere geldigheid.

Anderzijds kan er nog in belangrijke mate gesleuteld worden aan de algoritmen zelf. In communicatietoepassingen is men niet zozeer geïnteresseerd in de exacte locatie van de zender, maar in de kwaliteit van de signaalreconstructie. Omwille van het stochastisch karakter van het propagatiekanaal is de spatiale structuur van de signalen niet goed gedefinieerd. Maar communicatiesignalen zijn veelal sterk gestructureerd. Vele signalen bezitten een constante modulus of hebben een digitale structuur. Deze signaalstructuur wordt vrij goed behouden na propagatie. Recent is er onderzoek verricht naar algoritmen die op basis van de partiële kennis van de signaalstructuur het propagatiekanaal schatten [109, 117]. Dit is nog maar een aanzet. Verder onderzoek op deze matrices van lage rang met zowel een (niet-lineaire) structuur op kolomruimte als op de rijruimte zal naar alle verwachting een beduidende verbetering van de performantie brengen.

Chapter 1

Introduction

1.1 Motivation

Array signal processing is the branch of digital signal processing which studies propagating wavefronts sampled in time and space by sensor arrays. The most common sensor types are antennas, listening to electromagnetic radiation, and microphones, registering acoustic pressure waves. The use of arrays of sensors has a long history. Important applications include radar and sonar systems, radio astronomy and satellite communication [44]. However, nowadays they attract a lot of attention for terrestrial communication [2, 106]. The main reason is the expansion of the market for wireless communication systems, such as cellular mobile telephony. The success of the pan-European digital GSM system (Global System for Mobile communications [72]) operational in Belgium since January 1994 is illustrative. Also wireless data networks gain importance [4]. The huge demand for user capacity requires economical use of the scarce spectral resources. Several techniques are in use to maximize the capacity.

First, GSM is a cellular system as illustrated in Figure 1.1. The propagation environment for mobile communication is such that electromagnetic power evolves inversely proportional to $r^{3\cdots 4}$ [52], where r is the distance to the emitter. Because of this fast decay, frequencies can be reused as soon as a distant signal has become too weak to disturb the communication with a nearby user. Therefore, the area to be serviced can be divided into non-overlapping cells. Inside each cell mobile users, such as phones installed in cars or hand sets held by pedestrians, communicate with a central base station. When a user leaves a cell, the communication is switched to a new base station by a hand-over procedure.

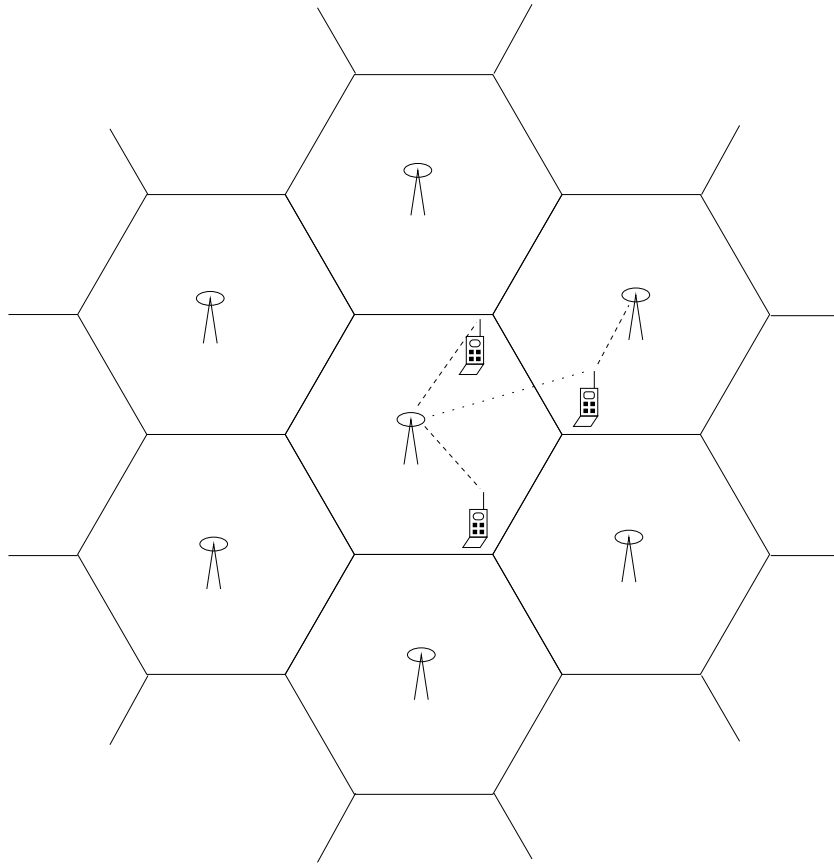


Figure 1.1: A cellular communication system. The service area is covered with cells. All mobile users inside a cell communicate with a central omnidirectional base station on different channels (dashed lines). Users from neighboring cells on a same channel are perceived as interferences (dotted line).

Secondly, GSM makes use of time and frequency division multiple access (TDMA/FDMA). Two frequency bands of 25 MHz, one for the uplink and one for the downlink, have been allocated around 900 MHz. Inside a band there are 8 time slots and 124 frequency slots of 200 kHz available. In each cycle of 8 slots, a typical channel carrying digitized voice occupies one time slot on one frequency slot. Therefore, in principle 992 users could be serviced simultaneously inside a cell. The actual number is much lower. First, in all countries there are (or will be) several system

operators who run competing GSM systems in parallel. Therefore, an operator only gets part of the frequency slots. Also, frequencies cannot be reused in neighboring cells. As seen in Figure 1.1 a user acts as an interference source in surrounding cells. To keep the interference level sufficiently low, frequencies are only reallocated to the cells of the second tier (layer). This further divides the capacity with a factor 7. In an area where 2 operators coexist, the resulting maximal capacity per cell is roughly 70 users.

It is clear that this upper limit can often be reached in busy areas. A long-term solution is the installation of a new GSM-like system (DCS1800) in the 1800 MHz band where two 75 MHz band will be available. However, this requires a complete redesign of the system, whereas congestion often occurs in a limited number of busy cells. These local saturation problems can currently only be handled by subdividing a cell into smaller cells, in the limit leading to microcells only covering a few street blocks. This approach has important disadvantages. The investment in expensive base stations becomes prohibitive and frequent hand-overs have to be performed.

Recently, antenna arrays have been recommended to provide a cheap local solution to such a local capacity problem [2, 73, 106]. Currently, the omnidirectional base station has no knowledge of the position of the mobile user. The communication messages are broadcast omnidirectionally over the cell. They are received by all mobile users and each user selects its own messages. Broadcasting requires substantial power and possibly creates severe interference in the neighboring cells. In fact, it is more optimal to establish several point to point communication links. This can be achieved by directional transmitting and receiving by means of antenna arrays at the base station. This has several advantages. For the same reception level at the mobile user, the power emitted by the base station can go down. This in turn lowers also co-channel interference. And last but not least, array signal processing adds the possibility of spatial division multiple access (SDMA). Several users can be put onto the same frequency and time slots and still be discriminated according to their location. If three users with different positions are multiplexed onto the same channel, then the capacity of a cell is tripled. Only in cells suffering from saturation a more complex base station benefiting from antenna array technology has to be installed. Cells where no congestion occurs, remain unaffected. Also, the antenna array and the additional signal processing equipment are entirely part of the base station hardware.

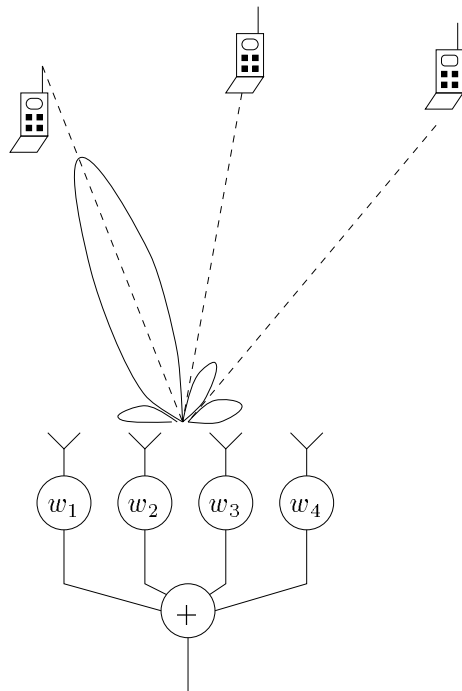


Figure 1.2: Spatial division multiple access. For each co-channel user a weighted linear combination of the antennas is computed. The associated directivity pattern of the weight vector w , indicating the directional array gain, exhibits a beam (maximal reception) pointing towards the desired user and a null (zero gain) along the directions of co-channel users.

The mobile transceivers are not altered.

Two operations are needed for directional air links. First combinations of the antenna elements have to be computed such that a focusing in space is attained. As shown in Figure 1.2 this is achieved by linearly combining the antenna outputs. The weights are determined such that the array is maximally receptive to the signal from the desired user, while all signals impinging from other directions are suppressed or at least sufficiently attenuated. This spatial filtering structure is often called a beamformer. If there is only one mobile user with a known position, then the weight vector can be computed by least squares estimation techniques. If there are multiple users with unknown locations, then eigendecomposition based techniques have to be used to determine the number of users

and estimate their location. This position information is impossible to obtain with a single omnidirectional antenna.

As an alternative a similar spatial selectivity can be obtained by a single sensor with a large continuous aperture. Examples of such sensors are the parabola antennas used in radio astronomy [44]. However, this approach is often undesirable. The technical difficulties and the equipment cost increase more than proportional with the aperture of such a large sensor. Moreover, array processing offers the user important flexibility. It is well known that a single antenna with a continuous aperture is only optimal in a spatially white noise environment [153]. Digital processing of array signals allows to take into account arbitrary noise correlation. It also adds the capability to steer notches (zeros) to cancel known interferences and jamming signals. And the adaptation of the weights is done electronically, without physically moving the sensors. Therefore, the array can continuously adapt to changes in its environment, such as moving users or short time interferences. Finally, arrays allow to attain a resolution which is higher than the one of continuous aperture antennas of comparable size.

The last decade has witnessed a large activity in the field of digital array signal processing. New concepts have been brought forward and many advanced algorithms have been developed. The main breakthrough is the fact that recent algorithms heavily rely on a mathematical model for the antenna outputs. The geometry of the data model is expressed in terms of concepts from *linear algebra*. The outputs of the array at a certain sampling time form a vector of observations called a snapshot. Similarly the signals from a single sensor observed over time are collected in vectors. The observations of the sensor array over a time interval are stored in matrices. As mentioned earlier, optimal weight vectors can be computed based on solving a set of linear equations (least squares estimation) and the position of the users is hidden in certain invariant subspaces of the data matrix. Therefore, matrix decompositions, such as the QR decomposition (QRD) and the singular value decomposition (SVD), play an important role.

In addition to being the natural framework for modeling the data, linear algebra also provides the algorithm designer with a whole set of robust algorithmic building blocks. Advanced algorithms offering improved performance can often be built up as a sequence of matrix decompositions. The trade off is the increase in computation with respect to simpler, not model-based algorithms. Therefore, execution of these algorithms in real

time necessitates the use of powerful computers.

Moreover, throughput rates in the front end processing of antenna array systems are often extremely high. In the GSM system the bit duration is only $3.7 \mu\text{sec}$, which yields a bit rate of 270.3 kbit/s. At this speed the signal reconstruction has to be performed. It consists of filtering the antenna outputs and performing equalization in time. The adaptation of the filter coefficients can be done at a lower speed. The position of users only changes very slowly compared to the bit rate. However, the propagation environment changes rapidly. The mobile user evolves at low height – his antenna is typically at 2 to 3 m – in an environment full of large structures, such as tall buildings or hills. Due to reflection and diffraction, a signal from a mobile reaches the antenna array via various paths with different gains and delays. In a mountainous area these multipath components may have delays up to $15 \mu\text{sec}$ (± 4 bit periods) which corresponds to a difference in path length of 4500 m. In urban areas maximum delays of $10 \mu\text{sec}$ are observed [15, 97]. Therefore, equalization is needed. But the interference of the multipath components also causes fast fading of the signal power. The fast fading may vary substantially in as little as half a wavelength (16.6 cm at 900 MHz) [5]. For a car driving at 120 km/h, it takes 5 ms to travel this distance. This number can be taken as a measure for the rate of change of the mobile channel. A cycle (sequence of 8 time slots) in GSM lasts 4.65 ms. Therefore, the GSM system recomputes the ± 5 equalization filter taps in each cycle, based on a sequence of 26 training bits in the middle of a burst.

At these speeds, even today's fast digital processors have a hard time performing complicated digital signal processing tasks on multiple channels in real time. Single dedicated or application specific processors may be able to provide the required computing power. If this is not the case, parallel computing is the natural way to further increase the execution speed. This is not an easy issue. In order to obtain a good speed up, one has to carefully balance computation, communication between processors and memory access. Of course, the constraints imposed by the architecture of the particular parallel machine under consideration have to be taken into account. Important distinctions can be drawn between synchronous and asynchronous machines and between global memory and distributed memory machines [26]. This double distinction leads to four classes of architectures with a different style of programming. Networks of workstations belong to the class of asynchronous distributed memory machines. Application specific systolic arrays are clearly a member of the

class of synchronous architectures with distributed memory. However, these features are only important at a later stage of the mapping from algorithm to architecture.

Whatever the target architecture may be, modern parallel design methodologies begin with a data flow analysis of the algorithm [50]. The flavor of these methodologies is mainly graphical. First a dependence graph (DG) of the algorithm is constructed. This is an acyclic graph uncovering the order in which signals are generated and consumed by operations. An alternative, more compact representation is the signal flow graph (SFG) which contains loops, *i.e.*, memory. Various transformations can already be applied to these graphs to make them more suitable for mapping onto a parallel machine. In a second phase a processor and an execution time have to be associated with each node (operation) in the graph. Assigning nodes to a physical processor is called the placement step. This step depends of course on the architecture of the particular machine under consideration. Determining an execution time for each node is called the scheduling step. Here the important issue is to respect the ordering imposed by the dependencies. A signal cannot be used in the computation before it has been calculated.

A large class of matrix algorithms is well suited for parallel execution because of their intrinsic regularity. Vectors and matrices are regular data objects. Also the processing is often shift invariant, *i.e.*, the same operations are to be performed to neighboring entries of the matrix. Moreover, sometimes the operations on a certain entry only require knowledge of the previous value of the entry and its neighbors. This property gives rise to local dependencies in the DG. Such algorithms are ideal candidates for implementation on systolic arrays. A systolic array is a special purpose synchronous architecture in which the processors are interconnected to nearest neighbors in a regular pattern. This type of architecture is attractive because of the extreme degree of pipelining and its regularity which eases a VLSI implementation. However, it should be noted that many implementations of algorithms for which a systolic architecture exists will not be systolic, *e.g.*, asynchronous communication or global memory may be used. A first reason is that it is unlikely that an algorithm has to be executed at such high data rates that the ultimate pipelining is needed. In general, the throughput requirements can already be met at a lower cost by allocating large parts of the computation to a few processors. A second reason is that a systolic array is parameterized by the problem size, *e.g.*, the size of the antenna array. Often problems

with different sizes have to be solved on the same architecture. The implemented architecture with a fixed size must then have an additional control to allocate computation over time and processors. Therefore, the attribute 'systolic' is more a property of the algorithm, indicating that its SFG is homogeneous and regular with local dependencies such that a fully systolic architecture is feasible. In this sense, one should realize that the (systolic) architectures which will be presented later on, are not really a specification of a physical parallel machine, but a virtual parameterized architecture from which an optimal physical parallel machine can be derived.

Currently formal methodologies and interactive software tools are under development to assist the designer in developing a special purpose array processor for his algorithm [23, 62, 84]. This simultaneous design of the signal processing algorithm and the parallel architecture on which it is to be executed, offers the largest freedom to obtain an optimal fit between algorithm and architecture. The corresponding discipline of deriving stable versions of algorithms which are well suited for parallel implementation by transformations on the SFG is often labeled *algorithmic engineering*¹ [57, 58, 83].

Economic considerations determine whether the development cost of an application specific parallel architecture is affordable or not for a particular application. In the near future the advent of semi-automated parallelization software may cut this cost considerably. In any case, whether an algorithm is to be executed on an application specific or on a general purpose parallel machine, reconsidering existing algorithms to increase their parallelism and regularity is an important issue. In a design it is generally the rule that the largest optimizations can be obtained at the highest level, *i.e.*, the algorithmic level.

1.2 General survey

As shown in Figure 1.3 two themes recur throughout this text. The first theme is the development of new algorithms for array processing applications. The second one is parallel architectures. Chapters proposing algorithms for similar applications are ordered sequentially. First chapter 2 introduces the necessary background material. Then chapter 3 and 4 pro-

¹McWhirter and co-workers use the term 'algorithmic engineering' for transformations on the SFG on a high level. Here the term denotes the complete field of simultaneous design of algorithms and architectures.

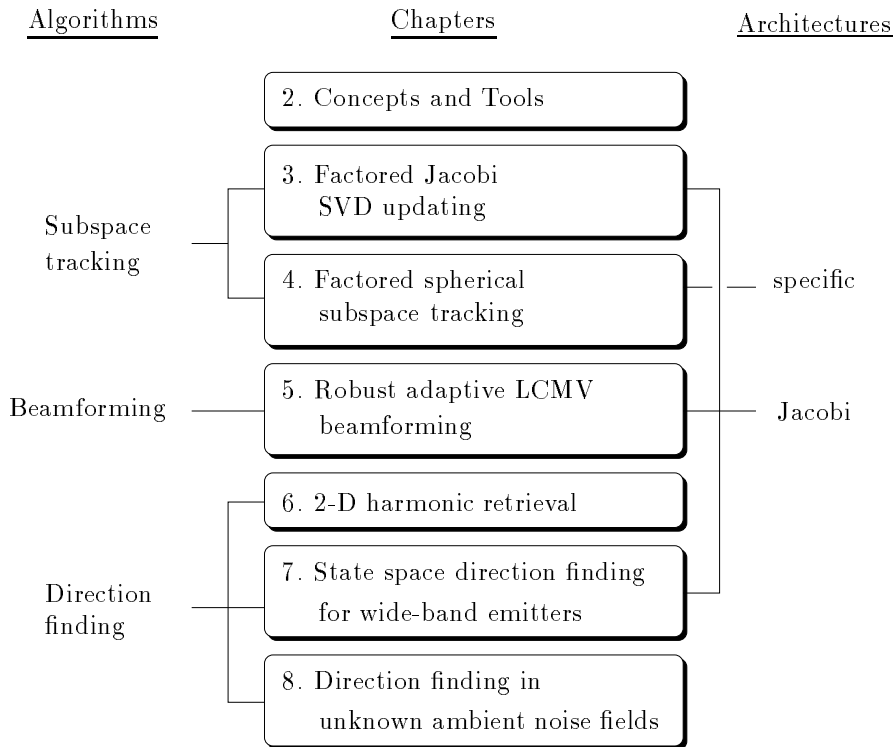


Figure 1.3: Relations between chapters 2 to 8

pose algorithms for *subspace tracking*. Tracking the dominant subspace of the data matrix will be a key component of recursive direction finding algorithms. The difference between the algorithms in chapter 3 and 4 is their computational cost and the accuracy with which they track the subspace. Chapter 5 introduces an adaptive beamformer with increased robustness to errors in the constraints. These constraints influence for instance the position of the main lobe of the beamformer. An example is a mobile user whose position estimate is subject to estimation errors. Insensitivity of the beamformer to this type of errors is an important asset. Finally, chapters 6 to 8 treat the direction finding problem under various assumptions. In the standard narrow-band data model there are multiple signals modulated onto the same known carrier frequency. In chapter 6 a subspace algorithm is proposed for the situation in which the carrier frequencies are not known. Chapter 7 presents a further generalization of the data model. Now the signals are assumed to be wide-band signals.

Such a model is needed for the processing of acoustic signals, such as speech or audio. Finally, chapter 8 returns to the standard narrow-band model for the signals, but proposes an extension of the noise model. The often unrealistic assumption that the sensor noise is spatially white, is relaxed to the assumption that the noise correlation matrix belongs to a well chosen linear model class. We also discuss an algorithm to estimate the additional noise parameters.

We will be especially interested in recursive algorithms for real time execution. Because of our concern for parallel implementation, the algorithms are composed of matrix decompositions. More irregular operations, such as nonlinear optimization methods, are avoided as much as possible. An exception is the colored noise algorithm of chapter 8 for which nonlinear optimization is necessary. But there, we will not be looking for a recursive algorithm either.

In the text four recursive algorithms will be mapped onto a parallel architecture. Our approach is not to come up with a totally different architecture for each new algorithm. Instead three of the four algorithms will be mapped efficiently onto the same architecture, *i.e.*, the Jacobi array which was originally developed for SVD updating [66, 67]. The fourth algorithm is mapped onto a related but simplified architecture. The wide applicability of the Jacobi architecture is at first sight surprising. However, several recursive signal processing algorithms for seemingly widely different applications can be formulated as Jacobi algorithms. Here we take the freedom to call an algorithm a Jacobi algorithm² if its structure is identical to the structure of the Jacobi SVD updating algorithm to be discussed in chapter 2, *i.e.*, it consists of a sequence of matrix-vector multiplication, QRD updating and a series of two-sided Givens rotations. Their data flow is identical. Only the node descriptions specifying the input output mapping, may differ. Examples are recursive algorithms for SVD and its generalizations, such as quotient SVD, product SVD [60] and the URV decomposition [68, 102]. Also other signal processing algorithms such as inverse recursive least squares (RLS) [61] and narrow-band direction finding [71] have been formulated as Jacobi algorithms. Here we extend this class by three more algorithms: a stable Jacobi SVD updating algorithm without reorthogonalization, a beamforming algorithm and a direction finding algorithm for wide-band sources. The fact that a rela-

²Originally the term Jacobi algorithm refers to a symmetric eigenvalue solver using two sided rotations [42]. Later the term was also adopted to denote the Kogbetliantz algorithm which uses two sided rotations to compute the SVD [48].

tively large class of relevant signal processing algorithms can be mapped onto the same Jacobi architecture, makes it much more attractive for implementation in hardware. The only requirement is that this architecture is not fully hard wired, but instead slightly programmable. It provides a limited set of operations, mainly rotations, and the user has to program certain actions, such as the way the rotation angles are determined. The design of such an array is currently taking place [24, 118].

1.3 Chapter overview

Below we give a detailed overview of the text. The main contributions of each chapter are summarized.

Chapter 2. Concepts and tools

In this introductory chapter we give an outline of the background material from the fields of *array processing*, *linear algebra* and *algorithmic engineering*. These are the three major areas on which this text is based. The chapter gradually introduces the field of array processing. First the data model for narrow-band sources impinging on an array is developed. Next some algorithms for beamforming and direction-of-arrival estimation are presented. In later chapters we will propose modifications and extensions to these algorithms. The concepts of linear algebra and algorithmic engineering are introduced as they are needed.

Chapter 3. Factored Jacobi SVD updating

This chapter presents a first algorithm for stable subspace tracking. As will be explained in chapter 2, the dominant subspace of the data matrix is determined by the position of the mobile users. When the mobiles are moving, this subspace changes. These variations have to be captured in order to track the position of the mobiles. SVD updating is well suited for subspace tracking. We propose a modification to the Jacobi algorithm for SVD updating. The original version of this algorithm requires periodic reorthogonalization in order to remain numerically stable.

The contribution of this chapter is the introduction of a minimal factorization for orthogonal matrices. An arbitrary orthogonal matrix $\mathbb{R}^{M \times M}$ can be factored as a sequence of $M(M - 1)/2$ Givens rotations. This factorization is applied to the matrix of short singular vectors in the Jacobi SVD updating algorithm. This ensures the numerical stability of

the algorithm, by excluding accumulation of round off errors in finite precision arithmetic. Moreover, this approach leads to a new Jacobi systolic architecture which solely consists of rotation nodes. This resemblance of the functionality of all nodes simplifies the hardware design of such a processor node.

The emphasis in this chapter is mainly on numerical linear algebra and architectures. However, updating the SVD is a key to the array processing algorithms to follow.

The material from this chapter is published in [135]. More concise descriptions are given in [136, 137].

Chapter 4. Parallel spherical subspace tracking

The spherical SVD algorithm is a second subspace tracker. It differs from the Jacobi algorithm in that it requires substantially less computation. This is due to the fact that it only keeps track of the dominant subspace of the SVD and that the exact signal and noise singular values are averaged. In contrast to the Jacobi SVD algorithm, the spherical subspace tracker is non-iterative. The trade off is a slower convergence speed.

The original algorithm suffers from the same error accumulation as the Jacobi SVD updating algorithm. Therefore, we propose to apply the same factorization idea as in the previous chapter. This leads to a novel factored spherical subspace tracker. Based on its SFG, we obtain by a linear placement and a piecewise linear schedule a new linear architecture. Finally, a planar architecture is developed using algorithmic transformations. Unfortunately, due to some irregularities this planar array is less efficient than its counterpart for the Jacobi algorithm.

The derivation of the algorithm and the mapping onto a linear array are described in [128].

Chapter 5. Robust adaptive LCMV beamforming

In chapter 3 we already proposed a modified Jacobi algorithm for SVD updating. This chapter introduces a second Jacobi algorithm for robust beamforming. The adaptive linearly constrained minimum variance (LCMV) beamformer is a popular choice for updating the weight vector of an antenna array, when a single signal-of-interest (SOI) with a known position is received in an unknown, possibly time-varying, noise and interference environment.

An application is a cellular communication system in which the base stations are equipped with antenna arrays, but no spatial multiplexing is applied. Once the position of the user is estimated, a beamformer can be computed such that interferences coming from neighboring cells are attenuated. Ideally in such a scheme the capacity could be multiplied by a factor close to 7 by reusing all frequencies in all cells [85]. The base station then has to cancel the co-channel interference from neighboring cells.

However, the LCMV beamformer is known to be very sensitive to errors in the constraint matrix caused for instance by inaccuracies in the location estimate. We propose to increase the robustness of the beamformer by recursively estimating the array response vector of the SOI. This is possible in applications such as GSM where a periodic training signal is available. The mathematical formulation leads to a recursive least squares problem with a time-varying constraint. Based on a known structure for LCMV beamforming with fixed constraints, *i.e.*, the generalized sidelobe canceler (GSC), a Jacobi-type algorithm can be formulated. Again, it can be mapped efficiently onto the parallel Jacobi architecture.

This chapter is based on [129]. In this reference also a second application specific systolic architecture is derived. Parts of the work are reported in [124, 125, 138]. An alternative approach to increase the robustness, based on SVD regularization, is given in [70].

Chapter 6. 2-D Harmonic retrieval

The thesis is concluded with three chapters on direction-of-arrival (DOA) estimation of multiple signals. Each chapter makes different assumptions on the data or noise model. In the standard narrow-band model the sources have a known carrier frequency in common. In this chapter we assume that the carrier frequencies are not longer known. They may or may not differ and have to be estimated simultaneously with the DOAs.

An example is a broadband antenna array system for controlling the regulations on spectral allocation. In every nation there is a governmental institute distributing licenses for radio transmission. Each licensee has to comply with the regulations on power levels, bandwidth, . . . As shown in Figure 1.4 airborne array systems can be used to check if the regulations are observed in the field. Such a system has to scan the frequency band of interest, determine the location of all emitters and measure their spectral characteristics.

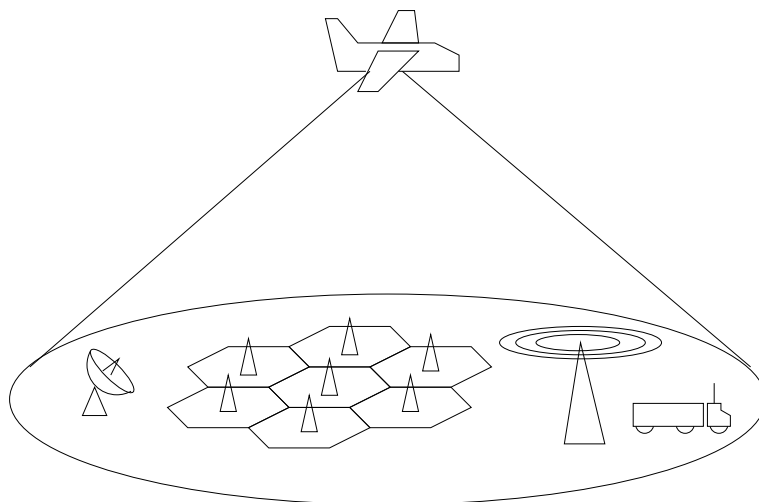


Figure 1.4: Airborne surveillance system monitoring all radio sources in a given spectral band, such as radio and TV broadcasting and wireless communication systems.

In this chapter we restrict our attention to a non-recursive algorithm for off-line processing. No architectural issues are discussed. The new 2-D harmonic retrieval algorithm is cheaper than existing algorithms. This is due to the fact that the 2-D problem is separated into two related 1-D estimation problems. For each 1-D estimation problem a matrix pencil has to be diagonalized. In fact, by careful selection of the matrices in the two pencils both 1-D problems are solved by the same transformations. Also an efficient rank restoration method is proposed to cope with situations where multiple signals have a common carrier frequency.

Part of this material can be found in [130].

Chapter 7. State space direction finding for wide-band emitters

In this chapter the assumption that the signals are narrow-band is relaxed. Instead the signals are modeled by a (low order) time-invariant linear system driven by white noise. Such a model is applicable to microphone arrays for processing acoustic signals. An interesting application is the development of a hands-free mobile telephony set [14, 75]. By use of a small microphone array a car driver could make a call without having to hold a handset. The microphone array serves to enhance the speech

signal in the background noise.

Existing approaches to wide-band direction finding are mainly based on reducing the wide-band problem to a set of narrow-band problems with different center frequencies [104, 145]. We do not decompose the wide-band signals into different frequency bins. Instead the novelty of our approach lies in the use of state space descriptions for the sensor outputs. We combine the conceptually new subspace algorithms for identification of linear systems [63, 121] with array signal processing to simultaneously identify the system poles and the locations of the sources.

Two algorithms are introduced. The first is a non-recursive algorithm which is related to the 2-D harmonic retrieval subspace algorithm of chapter 6. Again the estimates of the system poles and the locations are computed as the rank reducing numbers of two related matrix pencils. The second algorithm is a recursive wide-band direction finding algorithm, which is structured as a Jacobi algorithm such that the Jacobi architecture can be used for parallel implementation.

This chapter is largely based on [134]. The relation between the number of antennas and the number of signals is reported in [127]. An alternative fast algorithm based on displacement structures is reported in [126]. Early summaries of the chapter are [132, 133].

Chapter 8. A parametric approach to direction finding in unknown ambient noise fields

The flavor of this last chapter is different from the previous ones. It is written from the viewpoint of estimation theory and the emphasis is not on recursive algorithms for real time operation on parallel machines.

We consider again communication systems in which the signals are narrow-band waves. The standard data model assumes that the sensor noise is spatially white. This assumption is realistic for noise which is generated internally in the sensors. However, it is hard to defend for ambient noise impinging on the antenna array.

It is well known that the performance of subspace algorithms for direction finding is sensitive to unknown noise correlation [141]. Therefore, good methods to model and estimate the noise correlation matrix are crucial.

The main contribution of this chapter is a simple linear model for the ambient noise. If the response of the antenna array as a function of the direction-of-arrival is known, then the noise correlation matrix is a linear combination of a set of basis matrices. The noise parameters in the

model are the Fourier coefficients of the unknown noise field. In contrast to existing approaches this model is applicable to antenna arrays with an arbitrary geometry.

Secondly, we discuss the identifiability of the data model in the framework of linear matrix inequalities (LMIs) [7]. We also study maximum likelihood approaches for the estimation of the noise parameters. Unfortunately, this requires nonlinear optimization. In order to study the convergence, a concise analysis of the object function is presented. Finally, the performance of a gradient algorithm is illustrated by simulations.

This work has been initiated and elaborated during two visits to Prof. A. Paulraj at the Information Systems Laboratory of Stanford University. The noise model and the identifiability are discussed in [139].

Chapter 2

Concepts and Tools

In this introductory chapter we give an overview of the necessary background material from the three major areas on which the thesis is based, *i.e.*, array signal processing, linear algebra and algorithmic engineering. The first section gives a derivation of the data model for narrow-band point sources impinging on a sensor array. Specific geometric properties of this data model are crucial for the development of algorithms. The next section describes the beamforming task and introduces the QR decomposition. The generalized sidelobe canceler for recursive beamforming is discussed. This algorithm and its associated parallel architecture will form the basis for the robust LCMV algorithm in chapter 5. In the last section various direction finding (DF) algorithms are reviewed. Here, the appropriate mathematical tool is the singular value decomposition. We introduce the Jacobi algorithm for SVD updating. This algorithm and its parallel Jacobi architecture are the backbone of the recursive array processing algorithms of later chapters.

2.1 Narrow-band data model

Array signal processing is the branch of signal processing aiming at extracting information from propagating waves using sensor arrays. For an excellent introduction to the field, the reader is referred to [44]. Although array processing has its own peculiarities, there are close links to the more familiar theory of time series processing. Time and space are often interchangeable. A sensor array samples a wave at different (equidistant) points in space, whereas a tapped delay line samples a continuous-time signal at different points in time. Below we develop a mathematical model

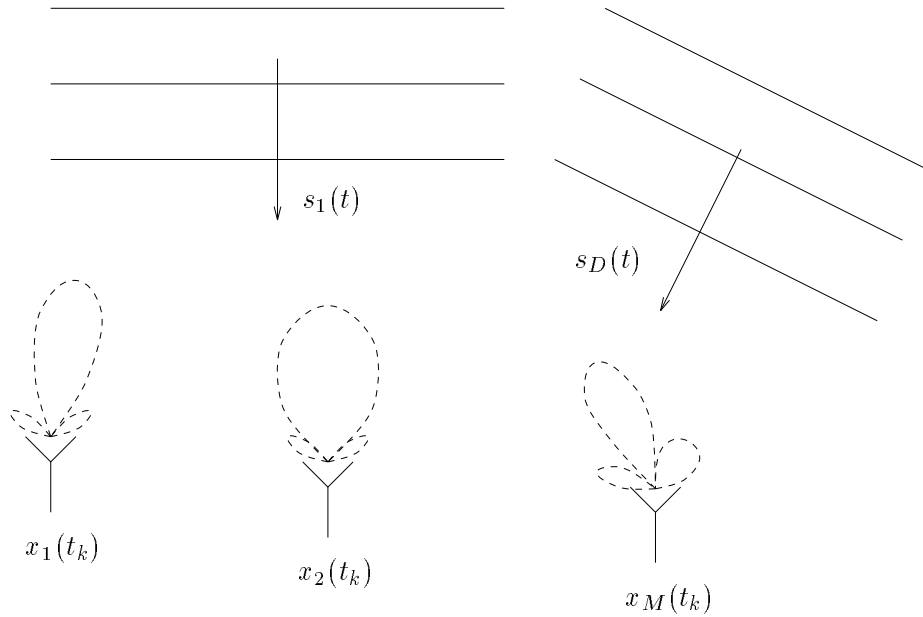


Figure 2.1: An antenna array with M elements receiving D incident wavefronts (full lines). The sources are in the far field of the array since the wavefronts are planar. The array elements do not have to be equal. This is indicated by the directivity patterns (dashed lines), which show the input-output gain of the element as a function of the direction-of-arrival.

for the sensor array output.

A sensor array is a group of M sensors placed at different spatial locations $\{r_m\}_{m=0}^{M-1}$ ($r_m \in \mathbb{R}^3$), sampling an electro-magnetic or acoustic propagating field at a set of (regularly spaced) time instants $\{t_k\}$ (Figure 2.1). Under the assumption that the sensor is a linear device of non-negligible size (*e.g.*, a parabola antenna) with impulse response $h_m(r, t)$, the output $\tilde{x}_m(t_k)$ of sensor m is related to the wave field $f(r, t)$ through a spatiotemporal filtering operation

$$\tilde{x}_m(t_k) = \iiint_{\rho} \int_{\tau} h_m(\rho, \tau) \cdot f(r_m - \rho, t_k - \tau) d\rho d\tau$$

where the vector $\rho \in \mathbb{R}^3$ is a relative coordinate with respect to the midpoint r_m of the sensor. If the size of the sensor (*i.e.*, the aperture) is small by comparison with the variations in the wave field, $f(r, t)$ can be

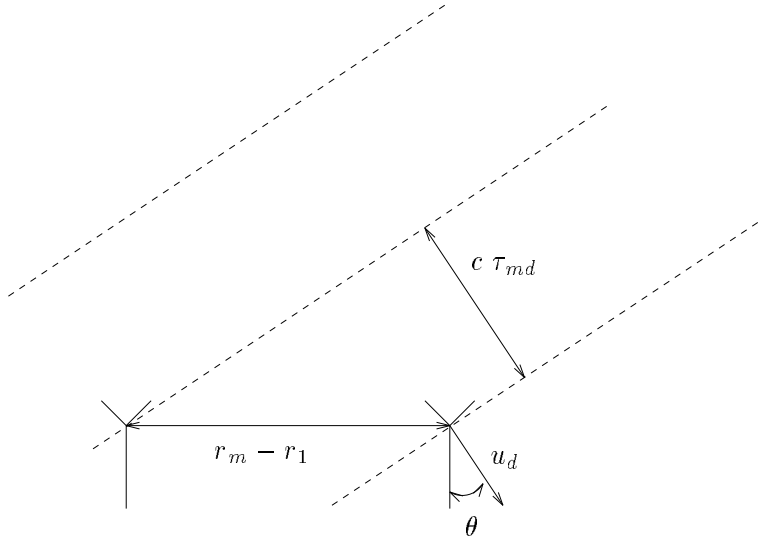


Figure 2.2: The time-difference-of-arrival τ_{md} between the reference sensor and sensor m for a planar wave with propagation vector u_d is given by the ratio of the projected distance between the two sensors onto u_d and the propagation speed c .

considered constant as a function of r and the 4-dimensional convolution reduces to an integral over time.

In many applications the data model can be further simplified. Consider a radio communications application. Here the amplitude-and-phase modulated information signal has a complex envelope

$$s_d(t) = \alpha_d(t) e^{j\phi_d(t)}$$

and the input signal to the reference sensor (sensor 1) is the information signal modulated onto a carrier wave with frequency f_c

$$\tilde{s}_d(t) = \alpha_d(t) \cdot \cos(2\pi f_c \cdot t + \phi_d(t)).$$

The output of sensor m is a scaled and delayed sinusoidal signal of the same frequency f_c

$$\tilde{x}_m(t) = g_m(\theta_d) \cdot \alpha_d(t - \tau_{md}) \cdot \cos(2\pi f_c \cdot (t - \tau_{md}) + \phi_d(t - \tau_{md}))$$

where $g_m(\theta_d)$ is the gain of sensor m for a signal impinging from angle-

of-arrival¹ (AOA) θ_d , and τ_{md} is the time-difference-of-arrival (TDOA) between the reference sensor and sensor m for signal d . For an emitter in the far field and a homogeneous non-dispersive medium (*i.e.*, constant propagation speed c), this TDOA is related to the propagation direction by

$$\tau_{md} = u_d^T \cdot \frac{(r_m - r_1)}{c}$$

where the vector $u_d \in \mathbb{R}^3$ denotes the normalized propagation vector (Figure 2.2). As an example, consider a typical array for a GSM base station consisting of $M = 10$ elements spaced half a wavelength apart at 900 MHz. The maximal TDOA between two consecutive elements is then of the order of 0.5 nsec. The maximal traveling time across the array τ_{\max} is of the order of 5 nsec. A signal $s_d(t)$ is called narrow-band, if its bandwidth B is smaller than the reciprocal of the maximal TDOA ($B \ll 1/\tau_{\max}$) [91]. For GSM, $B = 100$ MHz and $1/\tau_{\max} = \pm 200$ MHz such that the narrow-band condition is fulfilled.

The following approximation then holds for all m and d

$$\begin{aligned} \alpha_d(t - \tau_{md}) &\approx \alpha_d(t) \\ \phi_d(t - \tau_{md}) &\approx \phi_d(t). \end{aligned}$$

Therefore, the complex envelope $x_m(t)$ of the sensor output $\tilde{x}_m(t)$ equals the complex input signal up to a complex scalar $a_m(\theta_d)$

$$\begin{aligned} x_m(t) &= g_m(\theta_d) e^{-j2\pi f_c \tau_{md}} \cdot s_d(t) \\ &= a_m(\theta_d) \cdot s_d(t). \end{aligned}$$

This simple narrow-band relation leads to an elegant fundamental matrix expression for the sensor array output. Let the vector $x_{[k]} = x(t_k) \in \mathbb{C}^M$ be

$$x_{[k]} = \begin{bmatrix} x_1(t_k) & \cdots & x_M(t_k) \end{bmatrix}^T$$

and define the data matrix $X \in \mathbb{C}^{M \times N}$ as

$$X = \begin{bmatrix} x_{[1]} & \cdots & x_{[N]} \end{bmatrix}.$$

Column k of X contains the array output at time t_k , whereas row l of X contains the time series observed by sensor l . The data matrix can now

¹We assume that the sensor gain can be fully parameterized by the azimuth angle. In general, other parameters, *e.g.*, the elevation angle, also affect the sensor gain.

be written as the outer product $X = a(\theta_d) \cdot s_d$ of the input signal vector $s_d \in \mathbb{C}^N$

$$s_d = \begin{bmatrix} s_d(t_1) & \cdots & s_d(t_N) \end{bmatrix}$$

and the array response vector² $a(\theta_d) \in \mathbb{C}^M$

$$a(\theta_d) = \begin{bmatrix} a_1(\theta_d) & \cdots & a_M(\theta_d) \end{bmatrix}^T.$$

The set of array response vectors for all angles θ is called the array manifold \mathcal{A}

$$\mathcal{A} = \{a(\theta) \mid 0 \leq \theta < 2\pi\}.$$

If the sensor characteristics vary smoothly, then the array manifold draws a closed continuous curve in \mathbb{C}^M (Figure 2.3). An array manifold \mathcal{A} is called unambiguous when any set of M array response vectors is linearly independent. This property is crucial for the uniqueness of the direction finding solution [91].

In case of noisy observations of multiple input signals $\{s_d(t)\}_{d=1}^D$, all sharing the same carrier frequency but impinging from distinct AOAs $\{\theta_d\}_{d=1}^D$, the data model is extended to

$$X = A \cdot S + W \tag{2.1}$$

where the input signal matrix $S \in \mathbb{C}^{D \times N}$ and the array gain matrix $A \in \mathbb{C}^{M \times D}$ are given by

$$\begin{aligned} S &= \begin{bmatrix} s_1^T & \cdots & s_D^T \end{bmatrix}^T \\ A &= \begin{bmatrix} a(\theta_1) & \cdots & a(\theta_D) \end{bmatrix}. \end{aligned}$$

The matrix $W \in \mathbb{C}^{M \times N}$ contains both noise contributions which are generated internally in the sensors and unwanted observed signals (*e.g.*, ambient noise and interference signals).

Without noise on the data, the column vectors of X are contained in the column space of A . The span of the array gain matrix A is commonly called the signal subspace \mathcal{S}

$$\mathcal{S} = \{x \in \mathbb{C}^M \mid \exists y \in \mathbb{C}^D, x = A \cdot y\}.$$

²Note that the array response vector - also called direction-of-arrival vector or steering vector - is only dependent on the array characteristics, *i.e.*, directivity patterns and location of the sensors.

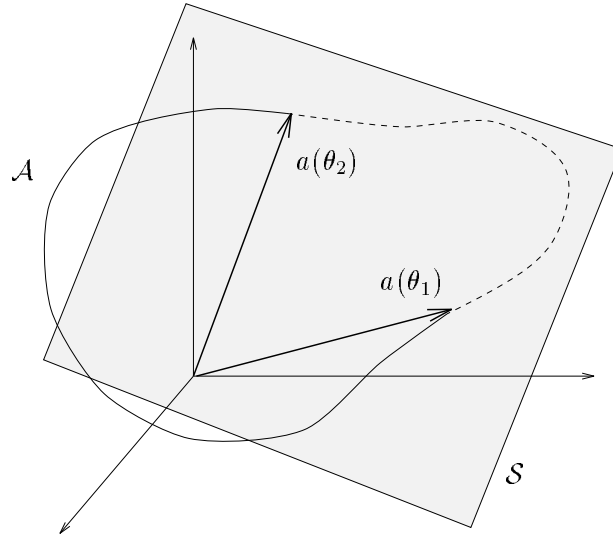


Figure 2.3: Geometry of the narrow-band data model ($M = 3, D = 2$). The array response vectors, corresponding to the AOAs are found in the intersection of the array manifold \mathcal{A} and the signal subspace \mathcal{S} .

The orthogonal complement of the signal subspace is known as the noise subspace \mathcal{N} . The geometry of the narrow-band data model is shown in Figure 2.3. In the absence of noise each snapshot $x[k]$ is a linear combination of the array response vectors $\{a(\theta_d)\}_{d=1}^D$ with varying amplitudes $\{s_d(t_k)\}_{d=1}^D$ and is therefore confined to the signal subspace \mathcal{S} . The location of the signal subspace is determined by the array response vectors and thus by the location of the sources.

2.2 Beamforming

A first important motivation for using sensor arrays is to enhance the signal-to-noise ratio (SNR) beyond that of a single sensor. This objective is the spatial analogue of FIR filtering in the time domain. Such a spatial linear filter is called a beamformer (Figure 2.4). Its output $y[k]$ is a weighted combination of the sensor outputs

$$y[k] = x[k]^H \cdot w.$$

The structure of the beamformer reminds of a transversal finite im-

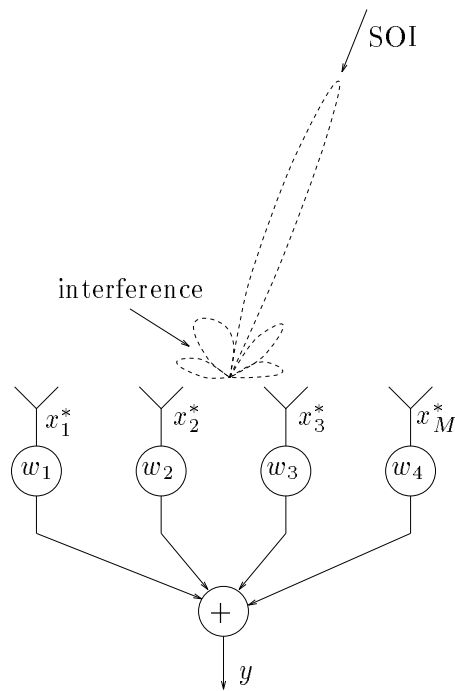


Figure 2.4: A beamformer linearly combines the antenna outputs with a weight vector w . The weights determine the directional gain of the array (dashed curve). The weights are such that the signal of interest (SOI) is enhanced while noise and interferences are suppressed.

pulse response (FIR) filter, in which the outputs of a tapped delay line are weighted and summed. The choice of the weight vector (*i.e.*, the filter coefficients), determines the spatial sensitivity of the sensor array. This spatial sensitivity is plotted as a directivity pattern, which is defined as

$$d(\theta) = |a(\theta)^H \cdot w|$$

for the set of angles θ under consideration. The directivity pattern shows the input-output gain as a function of the angle-of-arrival for a sinusoidal signal of a given frequency. Therefore, it is the analogue of a Bode plot of the FIR filter which shows the input-output gain as a function of frequency. Often the beamformer acts as a spatial bandpass filter, passing the signal-of-interest (SOI) in its main-lobe and rejecting signals arriving from other directions. Ideally the SNR at the output can be increased

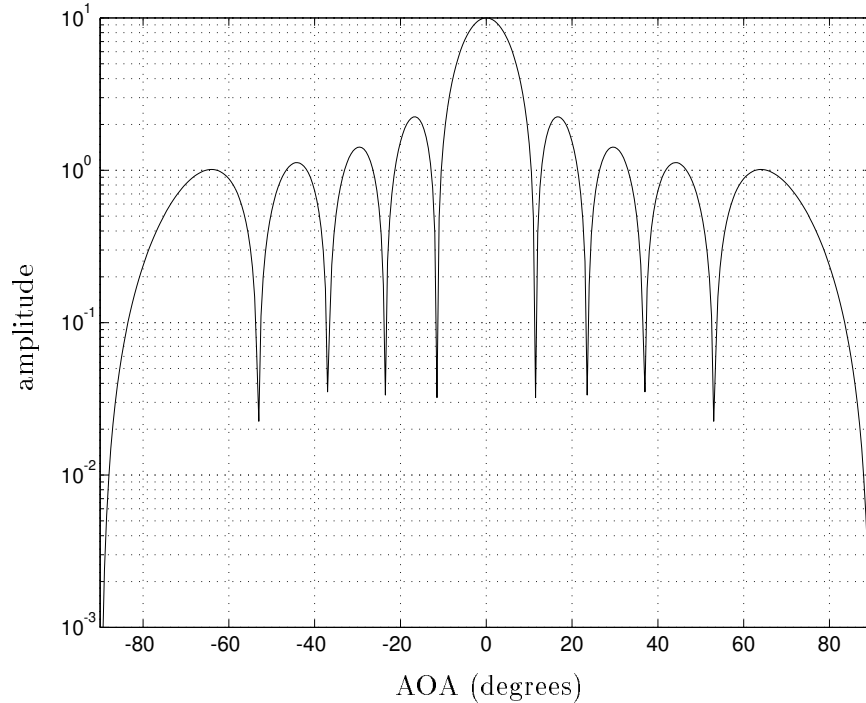


Figure 2.5: Directivity pattern (amplitude) for a 10-element ULA with spacing $\Delta = \lambda/2$ and all weights taken to be unity. The directivity pattern shows the input-output gain of the array in function of the angle-of-arrival for a signal of a given frequency.

by a factor of M , *i.e.*, the number of sensors. This is illustrated by the directivity pattern of a Uniform Linear Array (ULA) in Figure 2.5. Such a ULA is a common sensor array structure, composed of M identical omni-directional sensors, placed regularly on a straight line.

2.2.1 LCMV beamforming

In many applications the beamformer needs to reconstruct an SOI coming from a known AOA θ_s , maximally rejecting interferences and noise. The optimal weight vector can be determined by solving a constrained least squares problem. The total output power is the sum of the output power, due to the SOI, and the interference and noise power. Optimal

signal reconstruction is obtained if the output power is minimized subject to the constraint that the signal power is kept constant. This can be accomplished by a priori imposing a non-zero gain-and-phase μ in the direction of the SOI

$$a(\theta_s)^H \cdot w = \mu. \quad (2.2)$$

Additional constraints can be added, *e.g.*, to force zero gain along known interference directions, or to shape the directivity pattern. A frequently used constraint imposes an extremum of the directivity pattern at the SOI direction

$$\frac{\partial a}{\partial \theta}(\theta_s)^H \cdot w = 0.$$

The time-averaged output power is $p = \frac{1}{N} y^H \cdot y = w^H \cdot R_x \cdot w$ where $y = X^H \cdot w$ and $R_x = \frac{1}{N} X \cdot X^H$ is the sample correlation matrix. The optimal weight is then the unique solution of the following minimization problem

$$\min_w w^H \cdot R_x \cdot w \quad \text{subject to} \quad C^H \cdot w = m. \quad (2.3)$$

The matrix $C \in \mathbb{C}^{M \times K}$ is the constraint matrix and $m \in \mathbb{C}^K$ is the gain vector. Hence, this approach is known as linearly constrained minimum variance (LCMV) beamforming. In order for the optimization to have a solution, the number of constraints K is assumed to be smaller than the number of sensors M . The optimal weight vector \bar{w} is readily obtained as

$$\bar{w} = R_x^{-1} \cdot C \cdot (C^H \cdot R_x^{-1} \cdot C)^{-1} \cdot m.$$

An equivalent expression for \bar{w} can be derived as follows. The constraints form an underdetermined set of linear equations. All solutions are characterized by

$$\mathcal{W} = \{w | w = w_q + B \cdot w_b\}.$$

The designer is free to choose the so-called ‘quiescent weight vector’ $w_q \in \mathbb{C}^M$ as long as it satisfies the constraints

$$C^H \cdot w_q = m.$$

The ‘blocking matrix’ $B \in \mathbb{C}^{M \times (M-K)}$ is a preferably unitary matrix, whose columns span the null space of C^H

$$C^H \cdot B = O_{K \times (M-K)}.$$

Its name is explained by the observation that the columns of B are orthogonal to $a(\theta_s)$. Therefore, they act as notch filters, blocking the SOI and passing only noise and interferences. The SOI is only present in the output of the filter w_q . The new unknown $w_b \in \mathbb{C}^{M-K}$ is a weight vector of reduced dimension. Minimizing the output power with respect to w_b then results in

$$\bar{w}_b = -(B^H \cdot R_x \cdot B)^{-1} \cdot B^H \cdot R_x \cdot w_q. \quad (2.4)$$

This expression is implemented by the generalized sidelobe canceler (GSC) [37], whose high-level signal flow graph (SFG) is shown in Figure 2.6. We call a SFG a high-level graph when it consists of block operators transforming vectors or matrices, and the exact internal structure is not detailed. The upper rectangular operator is called the beamforming network. It contains the compound matrix $\left[B \mid w_q \right]$ and converts the sensor outputs from ‘data space’ into ‘beam space’ by matrix multiplication

$$\left[\tilde{x}_{[k]}^H \mid \tilde{y}_{[k]} \right] = x_{[k]}^H \cdot \left[B \mid w_q \right].$$

The lower operator represents a linear filter of size $M-K$. The expression for the optimal weight vector can be highly simplified by the use of the QR decomposition.

- Given a matrix $Z \in \mathbb{C}^{N \times M}$, $N \geq M$, then its QR decomposition is defined by

$$Z = Q \cdot R$$

where $Q \in \mathbb{C}^{N \times M}$ is a unitary matrix $Q^H \cdot Q = I_M$ and the matrix $R \in \mathbb{C}^{M \times M}$ is upper triangular.

The matrix $I_M \in \mathbb{R}^{M \times M}$ is the identity matrix. On condition that Z has full rank M , this decomposition is unique up to a diagonal matrix D with elements of unit-modulus. Its relevance lies in the following properties.

- The columns for Q constitute an orthonormal basis of the column space of Z .
- The matrix R is a square-root (even a Cholesky factor) of $Z^H \cdot Z$. The use of the QR decomposition avoids explicit computation of the latter matrix product. Therefore, so-called square-root algorithms have better numerical properties.

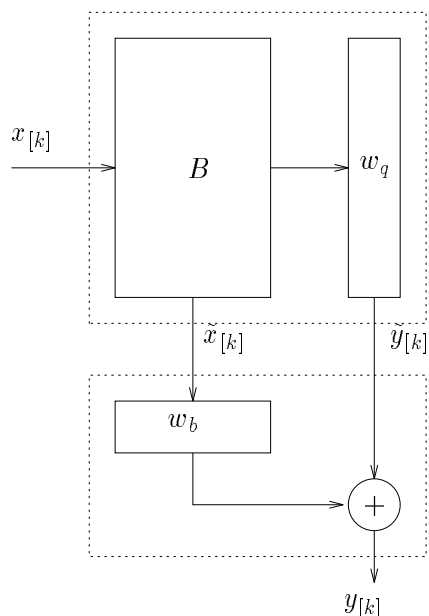


Figure 2.6: High-level signal flow graph of the generalized sidelobe canceler.

- The triangularity of R allows for a cheap solution of the associated linear system by back substitution. It is also important for fast updating.

Let the transformed data matrix $\tilde{X}^H \in \mathbb{C}^{N \times (M-K)}$ and the vector $\tilde{y} \in \mathbb{C}^N$ be defined as

$$\left[\tilde{X}^H \mid \tilde{y} \right] = X^H \cdot \left[B \mid w_q \right].$$

Substituting \tilde{X}^H in (2.4) for its QR decomposition $\tilde{X}^H = Q \cdot R$, we find an alternative, much simpler expression for the optimal weight,

$$\bar{w}_b = -R^{-1} \cdot u \quad (2.5)$$

where the vector $u \in \mathbb{C}^{M-K}$ is defined by $u = Q^H \cdot \tilde{y}$.

2.2.2 Recursive LCMV beamforming

In a time-varying environment, the optimal weight vector has to be adapted continuously. The sample correlation matrix is time-dependent and the

weight optimization becomes a recursive least squares problem. Each data vector is processed as soon as it becomes available. The data matrix $X_{[k]}$ now continuously grows as new observations are appended

$$\underbrace{X_{[k]}}_{k \times M} = \begin{bmatrix} \alpha \cdot X_{[k-1]} \\ x_{[k]}^H \end{bmatrix}.$$

The real scalar $\alpha < 1$ is an exponential weighting factor which is incorporated to de-emphasize older data.

One possible implementation, well-known for its numerical stability and intrinsic parallelism, is based on QRD updating. At each time t_k , the optimal weight is still given by Eq. (2.5)

$$\bar{w}_{b,[k]} = -R_{[k]}^{-1} \cdot u_{[k]}.$$

In order to track the weight vector, the QR decomposition of the growing matrix $\tilde{X}_{[k]}^H$ has to be updated. Assume that the QRD of $\tilde{X}_{[k-1]}^H = Q_{[k-1]} \cdot R_{[k-1]}$ is given. The updated matrix $\tilde{X}_{[k]}^H$ can be decomposed as

$$\begin{aligned} \tilde{X}_{[k]}^H &= \begin{bmatrix} \alpha \tilde{X}_{[k-1]}^H \\ \tilde{x}_{[k]}^H \end{bmatrix} \\ &= \begin{bmatrix} Q_{[k-1]} & O \\ O & 1 \end{bmatrix} \cdot \begin{bmatrix} \alpha R_{[k-1]} \\ \tilde{x}_{[k]}^H \end{bmatrix} \\ &= \underbrace{\begin{bmatrix} Q_{[k-1]} & O \\ O & 1 \end{bmatrix}}_{\left[Q_{[k]} \mid \star \right]} \cdot \underbrace{G_{[k]} \cdot G_{[k]}^H}_{\left[\begin{array}{c} R_{[k]} \\ 0 \cdots 0 \end{array} \right]} \cdot \begin{bmatrix} \alpha R_{[k-1]} \\ \tilde{x}_{[k]}^H \end{bmatrix} \end{aligned}$$

The submatrices O are zero vectors of appropriate dimension and the symbol \star denotes a quantity of no further interest. The transformation $G_{[k]}$ is unitary and restores the upper triangular structure of the R -matrix. Several choices are available for the computation of $G_{[k]}$. In view of a parallel implementation, the best choice is to construct $G_{[k]}$ as a sequence of $M - K$ Givens rotations $G_{[k]}^{i|M-K+1} \in \mathbb{C}^{(M-K+1) \times (M-K+1)}$. A Givens rotation $G^{i|j} \in \mathbb{C}^{n \times n}$ is a 2×2 rotation matrix embedded in the identity

matrix of size n . It only affects rows/columns (i, j)

$$G^{ij} = \begin{bmatrix} I_{i-1} & \vdots & & \vdots & & \\ \cdots & c & & s & \cdots & \\ & \vdots & I_{j-i-1} & \vdots & & \\ \cdots & -s^* & \cdots & c & \cdots & \\ & \vdots & & \vdots & & I_{N-j} \end{bmatrix}$$

where c is real, and $c^2 + |s|^2 = 1$. With each Givens rotation one can associate an angle $\alpha = \tan^{-1} \frac{|s|}{c}$ and a phase factor $e^{j\phi} = \frac{s}{|s|}$. A VLSI hardware component which is especially designed to perform 2×2 rotations accurately and fast is the CORDIC (coordinate rotation digital computer) processor [35, 38, 143]. It decomposes the rotation in a sequence of so-called micro-rotations. Each micro-rotation is such that it can be realized by a single (or a few) shift-and-add operations. The CORDIC can operate in two modes. In vector rotation mode, it rotates its input vector over a given angle. In angle accumulation mode, it computes an angle such that the second component of the input is zeroed.

In the Givens QRD updating algorithm [33], the i th Givens rotation is computed such that it annihilates the i th entry of $\tilde{x}_{[k]}^H$ by combining it with the i th diagonal entry of $\alpha R_{[k-1]}$ (angle accumulation mode). It is then applied to row i and $M - K + 1$ (vector rotation mode). A 3×2 example is given below.

$$\begin{bmatrix} \times & \times \\ & \times \\ \times & \times \end{bmatrix} \xrightarrow{G^{1|3H}} \begin{bmatrix} \times' & \times' \\ & \times \\ 0 & \times' \end{bmatrix} \xrightarrow{G^{2|3H}} \begin{bmatrix} \times' & \times' \\ & \times' \\ 0 & 0 \end{bmatrix}$$

The explicit computation of the weight vector $\bar{w}_{b,[k]}$ requires the solution of a triangular linear system. This requires a back-substitution step, which has an $\mathcal{O}(M - K)^2$ complexity. Moreover, during back-substitution the data flow evolves in the opposite direction of the data flow during the QRD updating. This complicates efficient pipelining of both operations. In array processing one is usually not interested in knowing the weight vector explicitly³. The ultimate goal is knowledge of the reconstructed signal

$$y[k] = \tilde{y}[k] + \tilde{x}_{[k]}^H \cdot \bar{w}_{b,[k]}.$$

³If knowledge of the weight vector is desired, it can be obtained without back substitution by 'freezing' the adaptation and inputting an identity matrix [99].

This so-called residual signal $y_{[k]}$ can be obtained as a side product of the QRD updating. Consider the QRD update applied to the compound matrix

$$\left[\begin{array}{c|c} \alpha R_{[k-1]} & \alpha u_{[k-1]} \\ \hline \tilde{x}_{[k]}^H & \tilde{y}_{[k]} \end{array} \right] = \left[\begin{array}{c|c} G_{[k],11} & G_{[k],12} \\ \hline G_{[k],21} & G_{[k],22} \end{array} \right] \cdot \left[\begin{array}{c|c} R_{[k]} & u_{[k]} \\ \hline O & z_{[k]} \end{array} \right]$$

where the matrix $G_{[k]}$ is partitioned accordingly. It is an easy exercise to check that the following equality is satisfied

$$G_{[k],22} \cdot z_{[k]} = \tilde{y}_{[k]} + \underbrace{\tilde{x}_{[k]}^H \cdot (-R_{[k]}^{-1} \cdot u_{[k]})}_{\tilde{w}_{b,[k]}}$$

Therefore, the optimal output of the beamformer is simply given by

$$y_{[k]} = G_{[k],22} \cdot z_{[k]}.$$

It follows from the construction of $G_{[k]}$ as a sequence of Givens rotations that the scalar $G_{[k],22}$ is the product of all cosines of the rotation angles

$$G_{[k],22} = \prod_{i=1}^{M-K} c_{[k]}^i.$$

Instead of solving a linear system, the computation of $y_{[k]}$ is reduced to a simple multiplication. This result was originally exposed by Shepherd and McWhirter [100].

The recursive generalized sidelobe canceler algorithm is summarized in Algorithm 1. The parameter $\gamma_{[k]}$ accumulates the product of the cosines of all Givens transformations.

Algorithm 1

Compute w_q and B such that

$$\begin{aligned} C^H \cdot w_q &= m \\ C^H \cdot B &= O_{K \times (M-K)} \end{aligned}$$

$$\begin{aligned} R_{[0]} &\leftarrow O_{(M-K) \times (M-K)} \\ u_{[0]} &\leftarrow O_{(M-K) \times 1} \end{aligned}$$

for $k = 1, \dots, \infty$

1. Matrix-vector multiplication

$$\left[\begin{array}{c|c} \tilde{x}_{[k]}^H & \tilde{y}_{[k]} \end{array} \right] \leftarrow x_{[k]}^H \cdot \left[\begin{array}{c|c} B & w_q \end{array} \right]$$

2. QRD updating

$$\left[\begin{array}{c|c} R_{[k]} & u_{[k]} \\ \hline O_{1 \times (M-K)} & z_{[k]} \end{array} \right] \leftarrow \dots$$

$$\left(\prod_{i=1}^{M-K} G_{[k]}^{i|M-K+1} \right)^H \cdot \left[\begin{array}{c|c} \alpha R_{[k-1]} & \alpha u_{[k-1]} \\ \hline \tilde{x}_{[k]}^H & \tilde{y}_{[k]} \end{array} \right]$$

$$\gamma_{[k]} \leftarrow \prod_{i=1}^{M-K} G_{[k],ii}^{i|M-K+1}$$

3. beamformer output computation

$$y_{[k]} \leftarrow \gamma_{[k]} \cdot z_{[k]}$$

endfor

2.2.3 Parallel mapping

Algorithm 1 is very well suited for parallel implementation. For the mapping from algorithm to parallel architecture we will adhere to the canonical mapping methodology described in [50]. This methodology can be exposed in a graphical way, manipulating graphical representations of algorithms. The advantage is that the parallelism is much more apparent in a graph than in a textual description. It also allows to visualize the transformations by software tools to assist in mapping new algorithms onto application specific or general purpose parallel architectures.

The starting point is a dependence graph (DG) of the algorithm. Usually the algorithm is specified as a nested loop program in a standard computer language. This textual sequential description is converted into a DG by associating with each operation a node indexed by its loop variables. The data flow is represented by directed arcs. An output argument from one operation which is used as an input argument of another operation defines an arc d_i between the corresponding nodes. The resulting DG is an acyclic directed graph showing how signals are produced and consumed by source and sink nodes.

The topology of the DG is crucial for the parallel mapping. For algorithms which are originally described as nested loops, there is a natural index space defined by the loop indices. The DG is normally represented in this index space. Important properties are *e.g.*, locality and homogeneity of the dependencies. A graph is said to have local dependencies

if it exchanges signals only with its neighboring nodes in the index space. It is homogeneous if the dependency pattern is shift-invariant. All nodes then exhibit the same regular connections.

Executing the algorithm involves allocating each operation in the DG to a certain processor (assignment or placement) at a certain time (scheduling). When mapping to a given parallel machine, the processor space is fixed, whereas for an application specific parallel processor one still has the freedom to select an optimal processor configuration. When the DG is homogeneous, a linear placement and schedule is often a good approach. This corresponds to finding a placement vector p and a scheduling vector s . All operations on straight lines parallel to the placement vector are allocated to the same processor. The execution time t_i of a node with coordinate r_i is given by the inner product

$$t_i = s^T \cdot r_i.$$

All operations on hyperplanes orthogonal to the scheduling vector s are executed simultaneously. In order to be permissible the scheduling vector has to satisfy two conditions

$$\begin{aligned} s^T \cdot d_i &\geq 0, & \forall i \\ |s^T \cdot p| &\neq 0. \end{aligned}$$

The first condition states that an operation can only be executed if all its inputs have been computed. The second condition implies that equitemporal operations should not be allocated to the same processor.

A projected scheduled graph is called a signal flow graph (SFG). In addition to nodes and arcs, it also has delays associated with each arc. A possible SFG of the generalized sidelobe canceler is depicted in Figure 2.7. Delays are indicated by the heavy dots. The loops in the upper rectangular part store the matrix $\begin{bmatrix} B & | & w_q \end{bmatrix}$, whereas the loops in the lower part store the matrix $\begin{bmatrix} R_{[k]} & | & u_{[k]} \end{bmatrix}$. The upper rectangular array is a multiplication array. The lower triangular array performs QRD updating [31]. The processors on the diagonal compute the Givens rotations, and pass them on to the right. In addition, they accumulate the product of the cosines. The original DG is obtained by repeating this two-dimensional graph over and over again for each time iteration. This is shown in the high-level DG of Figure 2.8.

The SFG level still allows a lot of freedom. Various parallel architec-

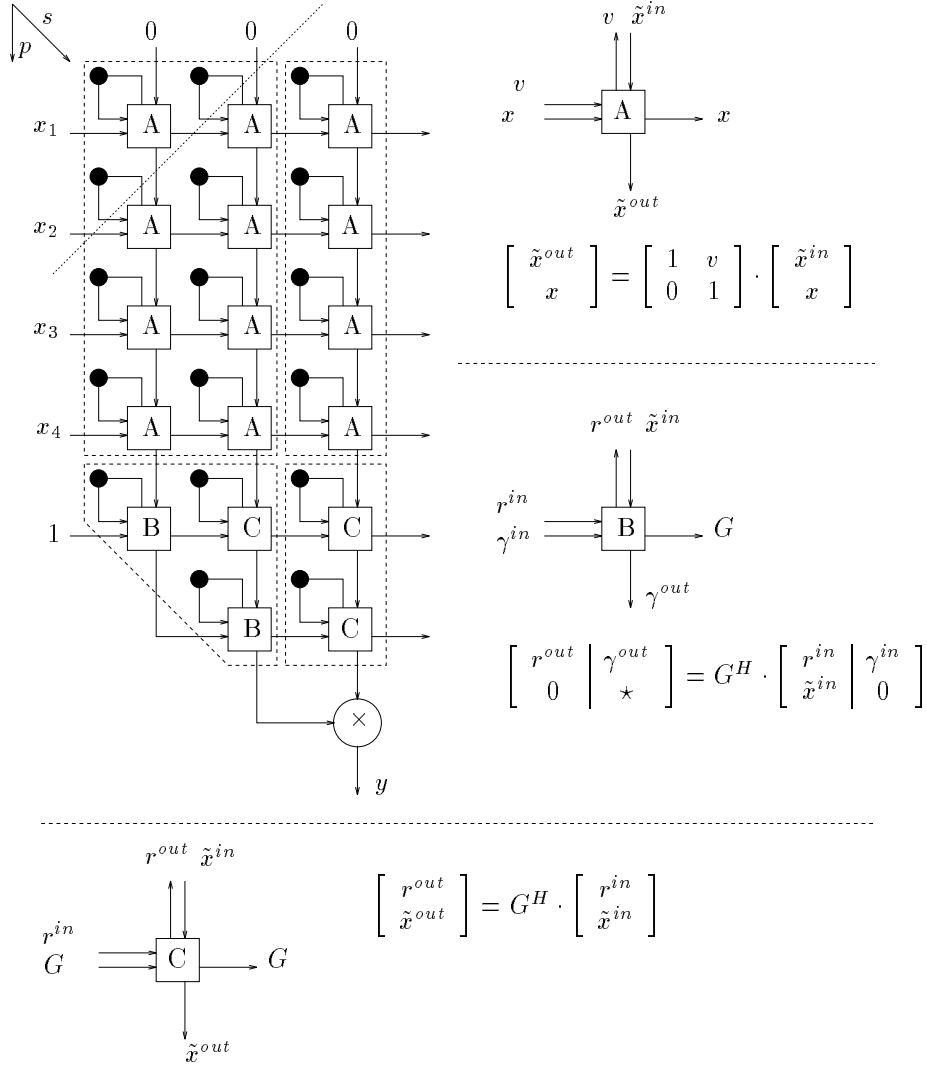


Figure 2.7: SFG for the generalized sidelobe canceler ($M = 4, K = 2$). The delay loops in the upper part store the matrix $[B | w_q]$, whereas the delay loops in the lower part store the matrix $[R_{[k]} | u_{[k]}]$. The vectors p and s indicate the placement and scheduling vector for a linear systolic architecture. The dotted line is a cut set.

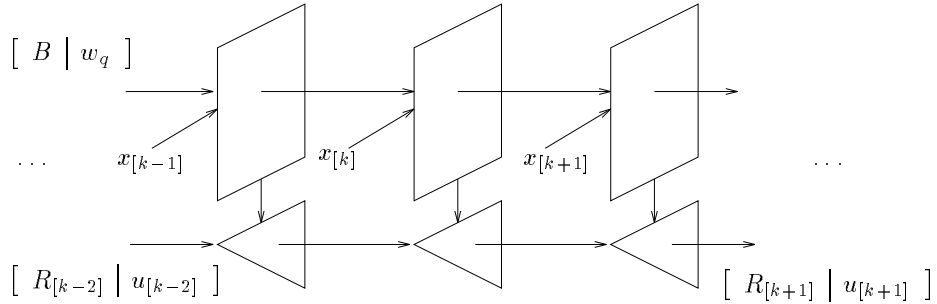


Figure 2.8: High level DG for the recursive generalized sidelobe canceler. The graph extends infinitely over the time dimension. The SFG of the previous figure is obtained by a projection parallel to the time axis.

tures can be derived. If one is interested in a linear array, an additional⁴ placement and scheduling (multi-projection) can be performed, *e.g.*, with the vectors p and s shown in the top-left corner of Figure 2.7. If a two-dimensional array is preferred, then only the scheduling step is performed. The schedule s indicated in the figure is actually an example of a systolic schedule. A way to represent the schedule is to associate $s^T \cdot d_i$ delays with each dependency d_i . In this representation a systolic schedule has the property that at least one delay is present on each dependency arc. In addition to being local in space, the computation is also localized in time. The resulting systolic array has the advantage of highly pipelined operation and consequently a high throughput rate. After a very short cycle new data vectors are fed in, while the others are pushed one stage further in the pipeline.

An alternative way to convert one SFG into another SFG are the following two retiming rules [50].

- *Time-scaling*: All delays in the graph may be multiplied by a factor $n > 1$, on condition that the input and output rates are slowed down accordingly.
- *Delay-transfer*: Given any cut-set of the SFG, which partitions the graph into two components connected by a number of arcs. Delays may be added on each of the outbound arcs on condition that the same number of delays is removed on each of the inbound arcs.

⁴It is clear that the placement and scheduling vectors could have been defined directly on the DG of Figure 2.8.

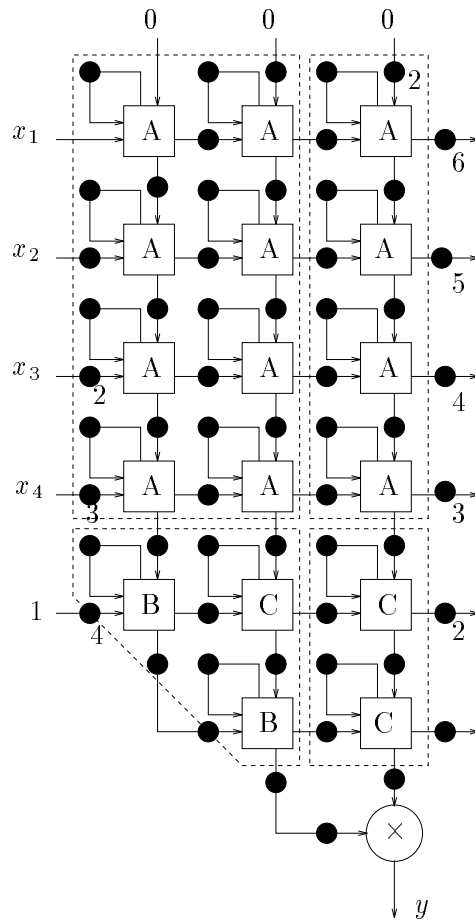


Figure 2.9: Final systolic architecture for the generalized sidelobe canceler ($M = 4, K = 2$). Multiple delays are indicated by a number.

Such a cut-set is indicated by the dotted line in Figure 2.7. It is a hyperplane orthogonal to the scheduling vector s . All arcs cross the cut-set in the same direction. Therefore, it is permitted to add one delay on all these arcs. By repeating this delay insertion on parallel cut-sets, the systolic array of Figure 2.9 is obtained. Due to the multiple cuts, the components of the input vector are skewed before being fed into the array. Similarly, in order to obtain all output components synchronously, the cuts introduce delays at the outputs. The pipelining period of the systolic array is only one cycle.

2.3 Direction finding

In addition to spatial filtering, sensor arrays are often employed for characterization of the wave field. A model for the field is postulated, *e.g.*, one assumes that the field consists of a sum of narrow-band waves, and some unknown parameters have to be estimated, *e.g.*, the number of narrow-band signals. Again one can set up a parallel with time series processing. The corresponding estimation problem is to determine the frequencies of a finite sum of sinusoidal time series drowned in noise [86].

Linear algebra is the natural tool for mathematically formulating the information extraction objective and for deriving algorithms to estimate the unknowns.

2.3.1 Subspace algorithms

Since 1980 high resolution methods for direction finding of narrow-band emitters have been proposed. Their resolution exceeds the Rayleigh resolution limit (*i.e.*, the limit for Fourier based techniques) by full exploitation of the narrow-band data model. An important class is formed by the so-called subspace algorithms, such as MUSIC [96], ESPRIT [92] and WSF [79, 142]. They perform the mapping from observations into a set of AOA's in two steps.

First the signal subspace \mathcal{S} is determined as the column space of the data matrix X . It follows from the data model that in the noiseless case the rank of X equals the number of impinging narrow-band (non-coherent) waves. In the case of noisy observations the data matrix will have full rank. The signal subspace then has to be estimated. For white noise on the data the appropriate tool is the singular value decomposition (SVD).

The singular value decomposition of a matrix $Z \in \mathbb{C}^{N \times M}$, $N \geq M$ is defined by

$$Z = U \cdot \Sigma \cdot V^H$$

where $U \in \mathbb{C}^{N \times M}$ and $V \in \mathbb{C}^{M \times M}$ are unitary matrices and $\Sigma \in \mathbb{R}^{M \times M}$ is a diagonal matrix containing the singular values $\sigma_i \geq 0$ in non-increasing order.

This singular value decomposition is of fundamental importance in modern signal processing, statistics, systems identification and control [93, 114, 123]. Its relevance is due to important properties, which are conceptual as well as numerical.

- From a conceptual point of view there is a relation between the SVD of Z , and the eigenvalue decompositions of $Z^H \cdot Z$ and $Z \cdot Z^H$. The singular values σ_i are the square-roots of the eigenvalues. The singular vectors u_i and v_i are equal to the eigenvectors of the 'squared' matrices. The following approximation property provides the optimal answer for the subspace estimation problem.

Given a matrix $Z \in \mathbb{C}^{N \times M}$, then the matrix $Y \in \mathbb{C}^{N \times M}$, nearest in Frobenius norm or 2-norm to Z , and of rank $D < \min(M, N)$ is given by the truncated SVD of Z

$$Y = \sum_{i=1}^D u_i \cdot \sigma_i \cdot v_i^H.$$

In the case of independent, identically distributed (i.i.d) white Gaussian noise on the data, the truncation of the SVD yields also the maximum likelihood estimator for the signal subspace. Moreover, the smaller singular values can be used for estimation of the noise level γ .

- From a numerical point of view excellent algorithms exist for computing the SVD [33]. They exploit the orthogonality of the decomposition. The theory of many applications is developed based on the eigenvalue decomposition (EVD) of the matrix $Z^H \cdot Z$. However, in actual computation the SVD is preferred over the EVD because of the potential loss of numerical accuracy associated with the explicit calculation of the matrix-matrix product.

The second mapping from estimated signal subspace to AOAs is performed via (partial) knowledge of the array manifold \mathcal{A} . Basically, one has to look for the D array response vectors $\{a(\theta_d), d = 1, \dots, D\}$ ⁵ which are comprised in \mathcal{S} . Here several alternatives exist. We only mention two of them.

The MUSIC algorithm [96] was the first algorithm to fully exploit the geometry of the data model. It determines the array response vectors which are nearest to the estimated signal subspace - or most orthogonal to the estimated noise space - by a one-dimensional maximization of the object function

$$d_{MUSIC}(\theta) = \frac{1}{\|a(\theta)^H \cdot V_n\|}$$

⁵We assume an unambiguous array manifold. If this condition is not met, the mapping from subspace to AOAs may not be unique.

where the matrix $V_n \in \mathbb{C}^{M \times M-D}$ contains the singular vectors corresponding to the smallest singular values. This object function is the reciprocal of the norm of the projection of the array response vector onto the noise subspace. If the array response vector lies in the signal subspace, this projection is zero. The D dominant peaks of $d_{MUSIC}(\theta)$ are selected as the AOA estimates. The performance of the MUSIC algorithm is very good [107]. However, it has a few disadvantages. First the array manifold \mathcal{A} has to be known a priori. If no accurate analytic model of the array response vectors is available, then the array has to be calibrated. This is a difficult and expensive operation. Furthermore, a non-linear optimization is required. In addition to the convergence problems due to local minima, this may require a lot of computation.

The ESPRIT algorithm [92] avoids the knowledge of the array manifold and the non-linear optimization. Instead it assumes a translation structure in the array configuration (Figure 2.10). The sensor array should consist of M doublets. Such a doublet is a pair of identical sensors displaced over a known constant vector. No information on the directivity patterns of the sensors is required. They may be arbitrary, as long as they are pair-wise identical in a doublet. The advantage of using this structure is that the need for non-linear optimization is circumvented. Convergence problems are avoided. Instead the AOAs can be estimated by a sequence of matrix decompositions. Below we give an outline of the method.

The array consists of two identical, but translated subarrays. Under the same assumptions as in section 2.1 the output of the first sub-array is given by

$$X = A \cdot S + W_x. \quad (2.6)$$

The output of the second sub-array obeys a similar equation

$$Y = A \cdot \Phi \cdot S + W_y \quad (2.7)$$

where the diagonal unitary matrix $\Phi \in \mathbb{C}^{D \times D}$ holds the extra phase shifts of each emitter d

$$\Phi_{dd} = \exp(-j2\pi \Delta \sin(\theta_d)/\lambda)$$

due to the translation of the subarrays. Now consider the noiseless matrix pencil

$$Y - \lambda \cdot X = A \cdot (\Phi - \lambda \cdot I_D) \cdot S.$$

Generically this matrix pencil has full rank D . However, if $\lambda = \Phi_{dd}$, the rank of the matrix pencil is reduced. Therefore, the phase shifts Φ_{dd} can

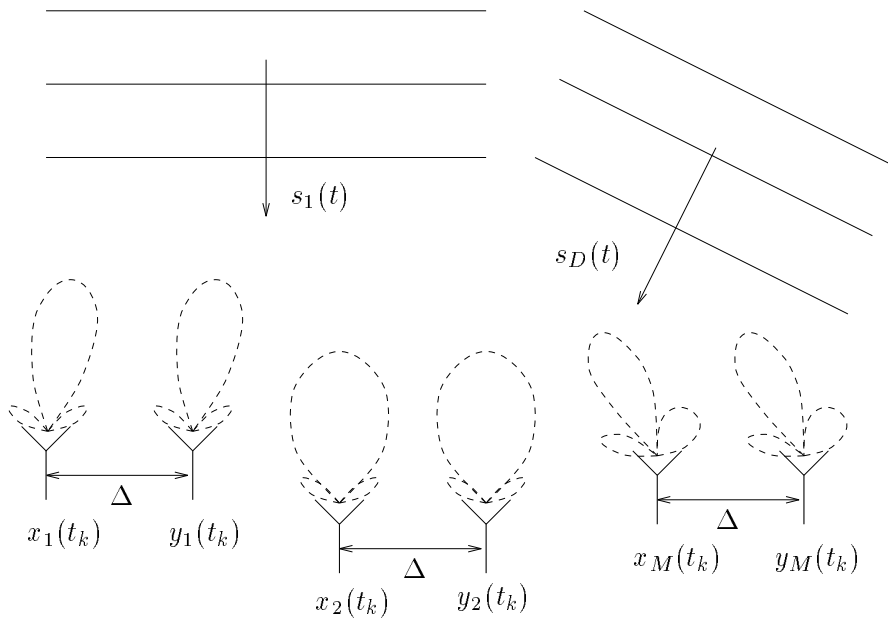


Figure 2.10: Sensor array configuration for the ESPRIT algorithm. The array consists of two identical subarrays, displaced by a vector Δ .

be computed as the rank-reducing numbers of the above matrix pencil. After compression of the column and row dimensions from $N \times M$ to $D \times D$ by pre- and post-multiplying with well-chosen $D \times N$ and $M \times D$ matrices, this problem becomes a generalized eigenvalue problem. In order to compute the rank-reducing numbers with orthogonal decompositions only, the generalized Schur decomposition can be used [11]. In the noisy case, various selections of compression matrices may yield slightly different estimates. The Total Least Squares [119] version of the ESPRIT algorithm [92] is renowned for its excellent noise suppression. However, it is intrinsically sequential. For an algorithm to be amenable to easy parallel implementation, the left and right compression matrices have to be independent of each other. In [113] the compression matrices are computed based on two independent SVDs. In [71] the noise is first suppressed by an instrumental variable method. The choice of the compression matrices then does not matter anymore. A simple choice suffices.

2.3.2 SVD updating

The first and most time-consuming step of subspace algorithms is the computation of the SVD of the data matrix X . In real-time applications which require tracking of the location estimates, the signal subspace has to be updated recursively. Optimal tracking is obtained when the SVD of the data matrix is updated exactly.

Unfortunately, this is a computationally expensive operation. It requires $\mathcal{O}(M^3)$ operations per update. For many real-time applications this computational cost is a serious impediment. Therefore, approximate algorithms for SVD updating have been developed which trade off accuracy for computational complexity.

Here we concentrate on an $\mathcal{O}(M^2)$ algorithm, developed by Moonen *et al.* [66]. It is reprinted below as Algorithm 2 and computes a unitary decomposition

$$X_{[k]}^H = U_{[k]} \cdot R_{[k]} \cdot V_{[k]}^H$$

where $U_{[k]} \in \mathbb{C}^{k \times M}$ and $V_{[k]} \in \mathbb{C}^{M \times M}$ are unitary matrices, but now $R_{[k]} \in \mathbb{C}^{M \times M}$ is an upper triangular matrix which is nearly diagonal (*i.e.*, close to the true $\Sigma_{[k]}$). The algorithm only keeps track of $R_{[k]}$ and $V_{[k]}$, which is sufficient for most signal processing applications.

After appending a new observation, the augmented data matrix $X_{[k]}^H$ can be written as

$$\begin{aligned} X_{[k]}^H &= \begin{bmatrix} \alpha X_{[k-1]}^H \\ x_{[k]}^H \end{bmatrix} \\ &= \left[\begin{array}{c|c} U_{[k-1]} & \mathbf{0} \\ \hline \mathbf{0} & 1 \end{array} \right] \cdot \left[\begin{array}{c} \alpha R_{[k-1]} \\ x_{[k]}^H \cdot V_{[k-1]} \end{array} \right] \cdot V_{[k-1]}^H. \end{aligned} \quad (2.8)$$

This decomposition has to be turned into a nearly diagonal upper triangular form again. This is accomplished by a sequence of three operations. The first operation is the matrix-vector multiplication

$$\tilde{x}_{[k]}^H = x_{[k]}^H \cdot V_{[k-1]}.$$

The second operation is a QRD updating, which works $\tilde{x}_{[k]}$ into the weighted triangular matrix $\alpha \cdot R_{[k-1]}$

$$\begin{bmatrix} \tilde{R}_{[k]} \\ \mathbf{0} \cdots \mathbf{0} \end{bmatrix} = \left(G_{[k]}^{1|M+1} \cdots G_{[k]}^{M|M+1} \right)^H \cdot \begin{bmatrix} \alpha R_{[k-1]} \\ \tilde{x}_{[k]}^H \end{bmatrix}$$

where each $G_{[k]}^{i|M+1} \in \mathbb{C}^{M+1 \times M+1}$ is a Givens rotation matrix. In order to compensate for this QRD updating in the decomposition of (2.8), the Givens rotations should be applied to the columns of the augmented $U_{[k-1]}$ matrix. However, since this growing matrix is not tracked by the algorithm, this part of the computation is dropped.

The QRD updating step degrades the nearly diagonal structure of the R -matrix. Therefore, the third operation aims at reducing the size of the off-diagonal elements again. This is done by applying a sequence of $M - 1$ Jacobi rotations, *i.e.*, double-sided (row and column) rotations. This sequence zeroes all entries in the first super-diagonal once. The i th Jacobi rotation follows from the SVD of the i th triangular 2×2 block on the diagonal of $\tilde{R}_{[k]}$

$$\begin{bmatrix} \times & 0 \\ 0 & \times \end{bmatrix} = \Theta_{[k]}^{i|i+1H} \cdot \begin{bmatrix} \tilde{R}_{[k],i,i} & \tilde{R}_{[k],i,i+1} \\ 0 & \tilde{R}_{[k],i+1,i+1} \end{bmatrix} \cdot \Phi_{[k]}^{i|i+1}.$$

For details on how to compute the rotation angles, we refer to the original paper [66]. Again, applying the row rotations $\Theta_{[k]}^{i|i+1}$ to the columns of the U -matrix may be omitted. On the other side, compensation for the column rotations $\Phi_{[k]}^{i|i+1}$ is mandatory. They need to be applied to the columns of $V_{[k-1]}$. The operation of a single sequence is illustrated below on a 4×4 example.

$$\begin{array}{ccc} \begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \\ & & & \times \end{bmatrix} & \xrightarrow{\Theta^{1|2H}, \Phi^{1|2}} & \begin{bmatrix} \times' & 0 & \times' & \times' \\ & \times' & \times' & \times' \\ & & \times & \times \\ & & & \times \end{bmatrix} \\ \begin{bmatrix} \times' & e & \times'' & \times' \\ & \times'' & 0 & \times'' \\ & & \times' & \times' \\ & & & \times \end{bmatrix} & \xrightarrow{\Theta^{3|4H}, \Phi^{3|4}} & \begin{bmatrix} \times' & e & \times''' & \times'' \\ & \times'' & e & \times''' \\ & & \times'' & 0 \\ & & & \times' \end{bmatrix} \end{array} \xrightarrow{\Theta^{2|3H}, \Phi^{2|3}}$$

The zero entry created by the i th Jacobi rotation is filled in again by the next Jacobi rotation. These fill-in elements are denoted by the *es*. Zeroing the complete strictly upper triangular part thus requires an iterative algorithm. Each Jacobi rotation decreases the norm of the off-diagonal elements with $|\tilde{R}_{[k],i,i+1}|^2$. However, the convergence could stall if a first super-diagonal of almost zero-entries were created, while other entries are still significant. Therefore, care has to be taken in the selection of the Jacobi rotations such that all entries are circulated towards and away from

the first super-diagonal. Two solutions exist for the angles of the Jacobi rotations. Consequently two circulation strategies may be used. A first strategy always selects the rotations over the smaller angles followed by a two-sided 2×2 permutation. The second strategy always selects the larger rotation angles. Both strategies tend to move entries around such that the convergence is not stalled. More details can be found in [66]. The final description of the Jacobi SVD updating is given in Algorithm 2.

Algorithm 2

$$\begin{aligned} V_{[0]} &\leftarrow I_M \\ R_{[0]} &\leftarrow O_{M \times M} \end{aligned}$$

for $k = 1, \dots, \infty$

1. Orthogonal matrix-vector multiplication

$$\tilde{x}_{[k]}^H \leftarrow x_{[k]}^H \cdot V_{[k-1]}$$

2. QRD updating

$$\begin{bmatrix} \tilde{R}_{[k]} \\ 0 \end{bmatrix} \leftarrow \prod_{i=1}^M G_{[k]}^{M+1-i|M+1^H} \cdot \begin{bmatrix} \alpha \cdot R_{[k-1]} \\ \tilde{x}_{[k]}^H \end{bmatrix}$$

3. Jacobi rotations

$$\begin{aligned} R_{[k]} &\leftarrow \prod_{i=1}^{M-1} \Theta_{[k]}^{M-i|M+1-i^H} \cdot \tilde{R}_{[k]} \cdot \prod_{i=1}^{M-1} \Phi_{[k]}^{i|i+1} \\ V_{[k]} &\leftarrow V_{[k-1]} \cdot \prod_{i=1}^{M-1} \Phi_{[k]}^{i|i+1} \end{aligned}$$

endfor

The algorithm has several desirable properties. A first point is its moderate computational complexity, only $\mathcal{O}(M^2)$ per update. Moreover, the approximation error with respect to the exact SVD is in many cases acceptable. In [66] an analysis for the subspace tracking application (*e.g.*, for high-resolution parameter estimation) shows that the tracking error, defined as the distance in terms of canonical angles, between the true and the estimated signal subspace, both at time t_k , is bounded by the time variation of the true signal subspace in M steps, provided that the signal-to-noise ratio is moderately high. A numerical example is given in



Figure 2.11: (a) A ULA with $M = 5$ antennas and $\Delta = \lambda/2$ tracks a mobile traveling with constant angular speed ω from -20 deg to 80 deg. The signal is a constant modulus signal with random phase. The SNR at the antenna outputs is 20 dB.

(b) The time variation in M samples (TV) is the angle between the array manifold vectors at time k and $k - M$. The tracking error of the Jacobi SVD updating algorithm is the angle between the array manifold vector and the estimated dominant singular vector both at time k . TV is an approximate upper bound on TE. The forgetting factor is low $\alpha = 0.8$, such that the algorithm can follow the fast variation of the system.

Figure 2.11. The relation between the non-stationarity of the data and the number of rotation sequences required for good tracking performance is studied in [54].

Secondly, the algorithm consists of orthogonal (unitary) transformations. These transformations are renowned for their good numerical behavior. There is a good reason to prefer Givens and Jacobi rotations over other types of orthogonal transformations, such as Householder transformations. Givens and Jacobi rotations are computed based on local information (2×2 or 2×1 sub-matrices), and therefore attract much attention in the field of parallel computing.

A high-level SFG of Algorithm 2 is shown in Figure 2.12. The upper square operator takes in the vector $x_{[k]}$ and the matrix $V_{[k-1]}$. It computes the product vector $\tilde{x}_{[k]}$, which is propagated to the lower triangular operator. This operator takes care of the QRD updating and the generation of the Jacobi rotations and their application to the matrix $R_{[k-1]}$. The column transformation $\Phi_{[k]}$ is propagated upwards into the square operator, where the matrix-matrix product $V_{[k]} = V_{[k-1]} \cdot \Phi_{[k]}$ is finally computed. Details on the internal structure of the operators are given in the next signal flow graph of Figure 2.13. This SFG is an intertwining of two SFGs.

The first SFG consists of the rectangular nodes and the black arcs. The nodes execute the matrix-vector multiplication (upper square array) and QRD updating (lower triangular array). The black arcs indicate the flow of the row transformation parameters. The remarkable resemblance of the data flow of matrix-vector product and the QRD updating, is due to the fact that the matrix-vector product is considered as a row transformation

$$\begin{bmatrix} V_{[k-1]} \\ \tilde{x}_{[k]}^H \end{bmatrix} = \begin{bmatrix} I_M & O_{M \times 1} \\ x_{[k]}^H & 1 \end{bmatrix} \cdot \begin{bmatrix} V_{[k-1]} \\ O_{1 \times M} \end{bmatrix}.$$

The row transformation matrix can easily be decomposed as a sequence of M elementary Gauss transformations acting on 2 rows

$$G_{[k]}^{i|M+1} = \begin{bmatrix} 1 & 0 \\ x_{i,[k]}^* & 1 \end{bmatrix}.$$

The zero row $O_{1 \times M}$ is put in at the top side. This first part of the SFG is actually identical to the SFG of the generalized sidelobe canceler (Figure 2.7).

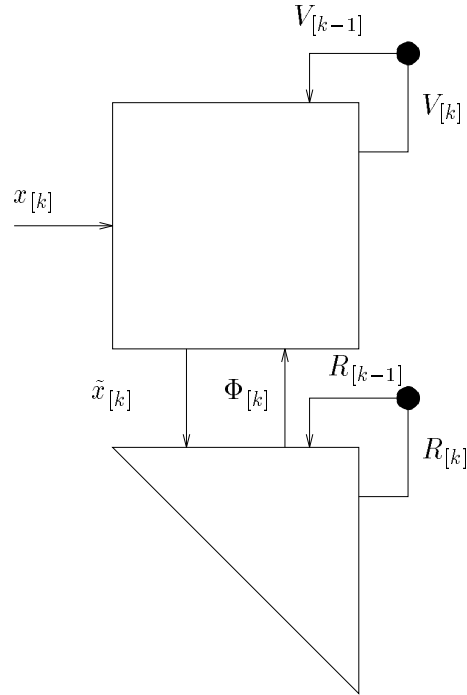


Figure 2.12: High level signal flow graph of the Jacobi-type SVD updating algorithm. The square part performs matrix-vector multiplication and column rotations. The triangular part performs QRD updating and computes the two-sided SVD rotations.

The second part of the SFG is dedicated to the Jacobi row and column rotations. They are executed in the hexagonal nodes. The flow of the rotation parameters is indicated by the grey shaded arcs. The row and column transformations are computed simultaneously on the diagonal and therefore both nodes are linked. The row rotations are then propagated to the right, whereas the column rotations are propagated upwards.

This SVD updating algorithm is essentially sequential. It contains long bidirectional dependency paths of length $\mathcal{O}(4M)$ (small arrows in Figure 2.13). Since there are no delays in one of the directions, this path cannot be pipelined using the standard retiming rules. In a synchronized system, the clock speed is upper bounded by the execution time of the longest dependency path in the algorithm. Implementing Algorithm 2 as such on a planar parallel architecture would result in a very inefficient

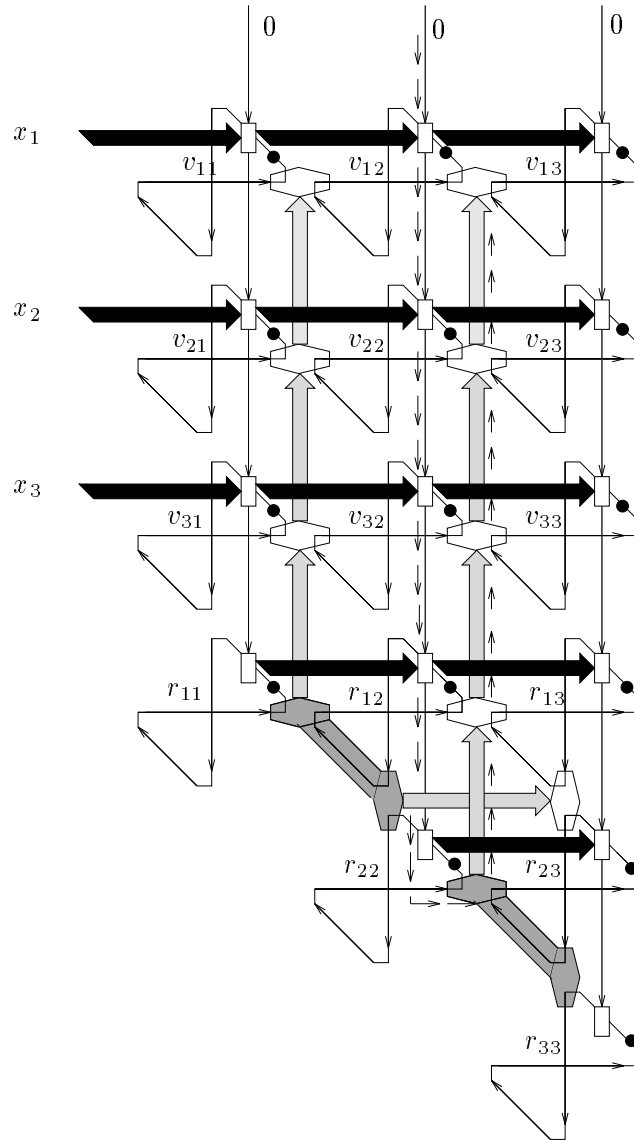


Figure 2.13: Detailed SFG of the sequential Jacobi SVD updating algorithm. The arrows indicate the long bidirectional vertical dependency path.

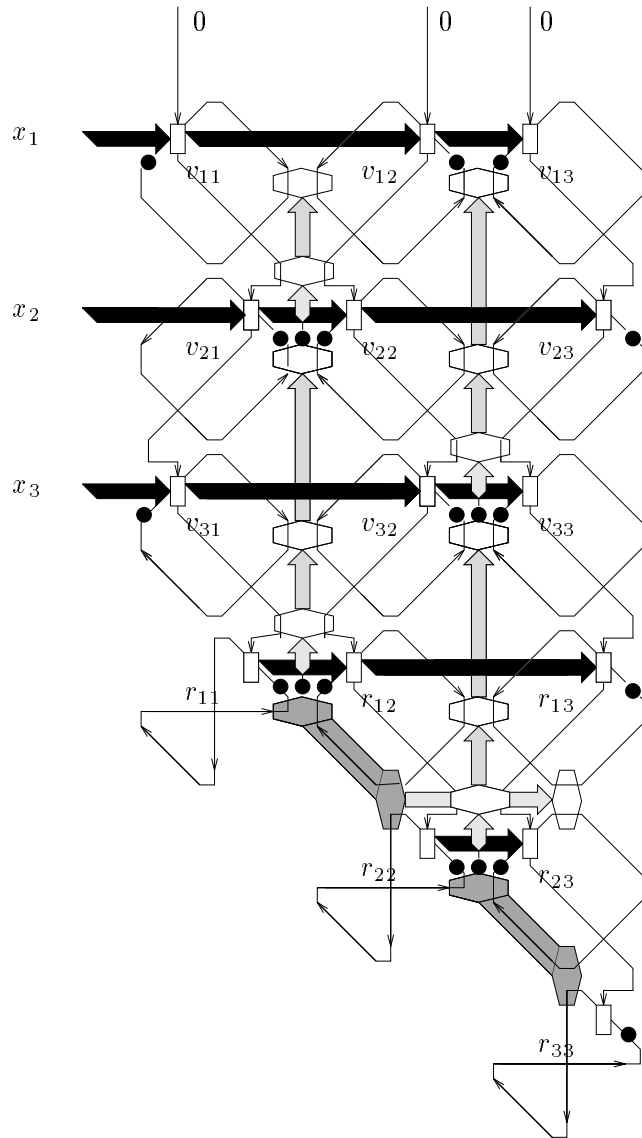


Figure 2.14: Detailed SFG of the parallel Jacobi SVD updating algorithm. The long bidirectional vertical dependency path has been removed.

array with throughput $\mathcal{O}(M^{-1})$. Therefore, the long vertical data dependency loop has to be broken. The only way is to modify the algorithm itself. The re-engineering of the Jacobi SVD algorithm in function of an efficient systolic architecture is described in [64]. We do not elaborate on this technique here. Examples of algorithmic transformations on a signal flow graph will be treated in later chapters. The final result is the SFG of Figure 2.14. This SFG corresponds to a second parallelized version of the SVD updating algorithm. Delays have been introduced on the upward vertical dependencies. Therefore this SFG can be pipelined using the time-scaling and delay-transfer retiming rules. The full size systolic architecture has $\mathcal{O}(M^2)$ nodes, and a pipelining period of 4 cycles, independent of M .

2.4 Conclusion

In this introductory chapter, we have presented the necessary background material. First, a mathematical model for the array data was derived. The concepts of signal subspaces and array manifolds were introduced. Next two common array processing problems, *i.e.*, beamforming and direction finding, were treated. The generalized sidelobe canceler algorithm for LCMV beamforming and the ESPRIT algorithm for direction finding were introduced. These algorithms are the starting point for some of the array processing algorithms in later chapters.

In practice, the signal constellation can often only be considered stationary for a limited time interval. Therefore, special attention was paid to recursive algorithms. The combination of high data rates and heavy computation of matrix decomposition algorithms motivated the study of parallel architectures. The paradigm of systolic arrays turned out to be well-matched for the type of matrix problems we address.

The generalized sidelobe canceler array presented a completed solution for adaptive and recursive parallel LCMV beamforming. The recursive direction finding problem was only partially solved here. We only focused on an efficient systolic architecture for the most burdensome part of the computation, which is the SVD updating. A recursive algorithm and systolic architecture for narrow-band direction finding, based on the Jacobi SVD updating method and the ESPRIT algorithm, is described in [71].