UNIVERSITÀ DEGLI STUDI DI NAPOLI

"FEDERICO II"

FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA ELETTRONICA

TESI DI LAUREA

ANALISI DELL'ONERE COMPUTAZIONALE IN SPICE PER MODELLI ELETTRO-TERMICI DI COMPONENTI E SISTEMI ELETTRONICI

Relatore

Ch. mo Prof.

MASSIMILIANO de MAGISTRIS

Correlatore

Ing. ALESSANDRO MAGNANI

Candidato

ANTONIO POLITO

Matr. 528/1443

ANNO ACCADEMICO 2011/2012

A mio padre e a mia madre
A Carmen

Ringraziamenti

Ringrazio il professore de Magistris, che con la sua professionalità e disponibilià mi ha permesso di svolgere questo lavoro di tesi serenamente e con passione.
Ringrazio l'Ing. Alessandro Magnani, i cui aiuti e consigli sono stati preziosissimi per il raggiungimento di questo obiettivo.
Ringrazio la mia famiglia, che mi ha permesso di intraprendere questo percorso di studio e che ha sempre creduto in me.
Ringrazio tutte le persone che mi sono state sempre vicino, sostenendomi nei momenti più difficili, in particolar modo Marco e Carmen.
Ringrazio tutti i miei amici, quelli di universià con cui ho condiviso i momenti più belli e più difficili di questa carriera,e gli amici di infanzia del Parco, con cui ho condiviso fino ad oggi le esperienze più importanti della mia vita.

Indice

Capitolo 1

Introduzione

L'oggetto di questa tesi è l'analisi dell'onere computazionale in Spice per modelli elettro-termici di componenti e sistemi elettronici.
La gestione delle problematiche termiche sta assumendo un ruolo sempre più di primo piano nella moderna progettazione elettronica, dal momento in cui è sempre più richiesta una elevata integrazione dei componenti e dei sottosistemi. È infatti ben noto che il surriscaldamento influisce negativamente sia sulle prestazioni che sulla affidabilità dei sistemi elettronici. Vari metodi sono stati proposti per derivare efficienti modelli termici dinamici di dispositivi e sistemi. Per una simulazione elettotermica a livello di sistema è utile che i modelli considerati siano ridotti al minimo della complessità (con le classiche tecniche di riduzione d'ordine), tali da essere facilmente integrabili in strumenti standard di simulazione di circuiti.
Una strategia efficace [7] per l'estrazione di equivalenti elettrici di modelli termici dinamici per sistemi elettronici è basata sulle tecniche del macromodeling elettrico: a partire dai dati di ingresso, ovvero le impedenze termiche auto e mutue del sistema in esame, si effettua una identificazione passiva che conduce ad un modello elettrico equivalente con la sintesi di Foster multi-

porta. È stata proposta [6] una topologia generalizzata della rete di Foster, che però ha riportato un calo di prestazioni rispetto alla standard, dovuto all'aumento di generatori controllati, e quindi dell'onere computazionale della rete. In questo lavoro di tesi è definita ed implementata in Matlab la procedura di sintesi circuitale di un sistema multi-porta, ed è proposta una sintesi efficiente della rete di Foster generalizzata in termini di tempi di risoluzione del circuito.

L'elaborato è quindi strutturato come segue: nel capitolo 2 sono introdotte le problematiche termiche on-chip, e sono descritti alcuni effetti da esse derivanti che possono causare una degradazione del circuito in termini di prestazioni ed affidabilità. Vengono poi riportati alcuni benefici che si ottengono con l'applicazione del macromodeling elettrotermico, e cenni sulle tecniche di identificazione di modelli elettrotermici ridotti, quali il Vector Fitting e il Positive Fraction Vector Fitting. Infine è implementato e descritto il codice Matlab per la procedura di sintesi circuitale.

Nel capitolo 3 vengono valutate le prestazioni dei diversi tipi di generatori controllati, al fine di analizzare i problemi di onere computazionale in Spice delle reti di feedback elettrotermico. La strategia adottata è quella di confrontare diverse topologie di rete a parità di numero di nodi: scelta una rete di riferimento di soli resistori lineari, sono valutati i tempi di risoluzione delle diverse topologie in Spice, prima nel caso statico e poi in quello dinamico.

Nel capitolo 4 sono considerati alcuni casi test che consentono di evidenziare dei miglioramenti nei tempi di risoluzione della rete di Foster generalizzata, a fronte dei risultati ottenuti nell'analisi effettuata nel capitolo 3.

Infine nell'appendice sono riportati i listati Matlab implementati nel corso dell'elaborato.

Capitolo 2

Macromodeling per simulazioni elettrotermiche

2.1 Problematiche termiche

Il problema di rimuovere calore da un chip non è nuovo, infatti problemi di natura termica sono stati in parte, se non del tutto, responsabili per la scomparsa di una varietà di tecnologie prima dell'avvento dei CMOS.
Inizialmente ogni chip conteneva un solo transistor, in seguito, con lo sviluppo tecnologico, ne furono integrati sempre di più aumentando la scala di integrazione. Con i miglioramenti tecnologici il numero aumentò vertiginosamente fino alla realizzazione di microprocessori VLSI (*Very large scale integration*), la cui denominazione indica una elevata integrazione di transistor all'interno di un chip. Ma anche se oggi giorno è possibile confezionare più transitor all'interno di un singolo chip, solo una frazione di essi può essere effettivamente utilizzata alla massima potenzialità a causa delle limitazioni imposte dai problemi di dissipazione di potenza termica.
L'avvento dell'integrazione tridimensionale ha reso le problematiche termiche

on-chip più sentite, dato che per i circuiti 2D i transistor vengono immessi tutti in un singolo piano con sopra diversi strati di interconnessione, quelli 3D impilano una sopra l'altra queste strutture 2D. Pertanto, nonostante i numerosi vantaggi della tecnologia 3D che la rendono affidabile anche per applicazioni future, la densità maggiore della integrazione 3D presenta l'inconveniente di esacerbare i problemi termici, motivo per cui è importante prestare attenzione su questo aspetto.

L'elevata temperatura *on-chip* comporta degli effetti sulle prestazioni del circuito [15] [18] , in particolar modo sul tempo di propagazione (per circuiti digitali), sulle prestazioni e sulla affidabilità. Gli incrementi di temperatura influenzano i parametri dei transistori e le interconnessioni provocando ritardi al circuito: ad esempio la mobilità μ dei portatori di carica di un transistore si riduce con l'aumentare della temperatura T secondo l'equazione

$$\mu = \mu\left(T_0\right)\left(\frac{T}{T_0}\right)^{-m} \tag{2.1}$$

dove T_0 è la temperatura ambiente (300K), m>0 è l'esponente di mobilità della temperatura. La riduzione della mobilità comporta una diminuzione della corrente di pilotaggio del transistore che a sua volta implica un aumento dei ritardi al crescere della temperatura.

Anche la resistenza R di un interconnessione aumenta all'aumentare della temperatura, secondo l'equazione

$$R = R_0(1 + \beta(T - T_0)) \tag{2.2}$$

dove R_0 è la resistenza del collegamento a temperatura ambiente T_0, β è una costante positiva ed implica che il ritardo di un collegamento aumenta con la temperatura.

Inoltre gli effetti termici possono innestare fenomeni che comportano l'invecchiamento prematuro del circuito, quali:

- *l'instabilità in temperatura della tensione di polarizzazione*, fenomeno che provoca sbalzi di tensione di soglia per lunghi periodi di tempo portando così il circuito a non rispettare le sue specifiche (si parla di *polarizzazione* poichè questa degradazione è accresciuta con l'applicazione di una polarizzazione sul nodo di gate di un transistore).

- *Rottura del dielettrico tempo-dipendente*, fenomeno che avviene negli ossidi di gate che si traduce in un improvviso aumento discontinuo della conduttanza dell'ossido di gate nel punto di rottura, da cui si verifica un aumento della corrente che attraversa l'isolante di gate. Ad elevate temperature tale aumento porta alla rottura del dispositivo più velocemente.

- *Elettromigrazione*, fenomeno che accade in un metallo quando si ha un trasporto di materia causato dalla presenza di correnti elettriche. Questo trasporto è dovuto dal fatto che gli elettroni trasferiscono quantità di moto agli ioni positivi del metallo. Quando una elevata corrente attraversa le sottili metallizzazioni presenti nei circuiti integrati si provoca un accumulo di ioni positivi in certe regioni e quindi la formazione di cavità in altre. Tali accumuli possono cortocircuitare le linee di interconnessioni adiacenti e la formazione di cavità può trasformare conduttori in circuiti aperti, causando in entrambi i casi la rottura del dispositivo. La dipendenza della temperatura in questo fenomeno si vede nell'equazione del tempo medio prima del guasto (MTTF,*mean time to failure*)

$$MTTF = AJ^{-n}e^{\frac{Q}{KT}} \tag{2.3}$$

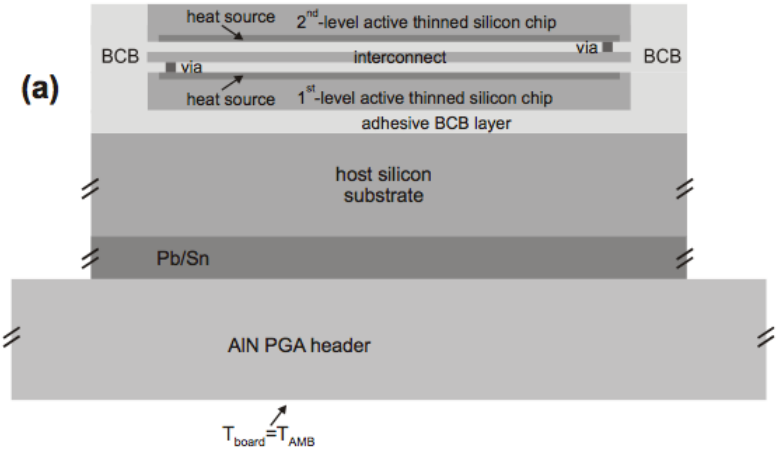
dove A è una costante che dipende dalla geometria della interconnessione , J è la corrente DC (media),n vale due sotto normali condizioni di utilizzo, Q è l'energia di attivazione, K la costante di Boltzman e T la temperatura del metallo. Da questa equazione si vede che una elevate temperatura degrada la durata delle linee di interconnessione e quindi del chip.

2.2 Applicazioni del macromodeling elettrotermico

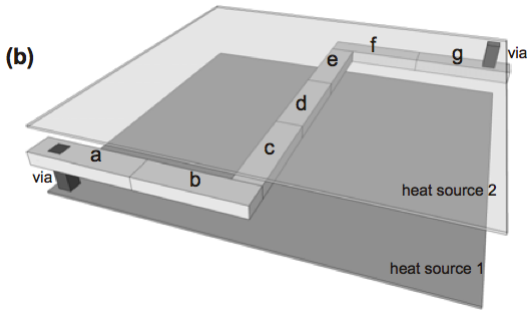
I problemi derivanti dagli equivalenti elettrotermici dinamici possono essere formulati nell'ambito delle tecniche standard del macromodeling elettrico.

Oggigiorno, a fronte di aumentare significativamente la densità di integrazione di sistemi semiconduttori e quindi realizzare prodotti più piccoli, più leggeri e meno costosi, si ricorre alla tecnologia UTCS (ultra-thin chip stacking). Nei sistemi UTCS i chip di silicio, assottigliati fino a 10 μm, sono integrati verticalmente sullo stesso strato di silicio e l'isolamento elettrico è garantito da strati di benzocyclobutene (BCB). I problemi dovuti agli effetti termici sono aggravati a causa della scarsa conducibilità del BCB [5](uguale a 0.18 W/mK, ottocento volte più piccola di quella del silicio) che contrasta la propagazione verso il basso del calore dalle regioni di dissipazione di potenza del circuito alla board, che si assume a temperatura ambiente. È possibile applicare la procedura di sintesi/identificazione tramite lo schema generalizzato multi-porta di Foster ad un modulo UTCS, contente due chip di silicio sottili posti verticalmente e isolati dal BCB.

La struttura è mostrata in figura 2.1(a), dove si può notare che il chip sepolto è attaccato al substrato di silicio grazie ad uno strato adesivo di BCB. In figura 2.1(b) è illustrata una vista 3D delle regioni attive dei chips



(a) Sezione trasversale del modulo 2-chip in tecnologia UTCS



(b) Vista 3D della rappresentazione termica della circuiteria che si trova sui chips e schema della interconnesione

Figura 2.1: Modulo 2-chip in tecnologia UTCS

e dello schema di interconnesioni: la lunga linea in rame è stata divisa in sette elementi individuali (a,b,c,d,e,f,g) , ad ognuno dei quali è associata una sorgente di calore. Il risultato dell'applicazione di questa procedura è un circuito elettro-termico equivalente al modulo UTCS, in grado di stimare con precisione ed in pochi secondi l'impatto del riscaldamento sulla propagazione di un segnale attraverso la linea di interconnessione. La linea di rame è quindi modellata con il circuito standard a celle RC, connesso ad un blocco di feedback termico (figura 2.2).

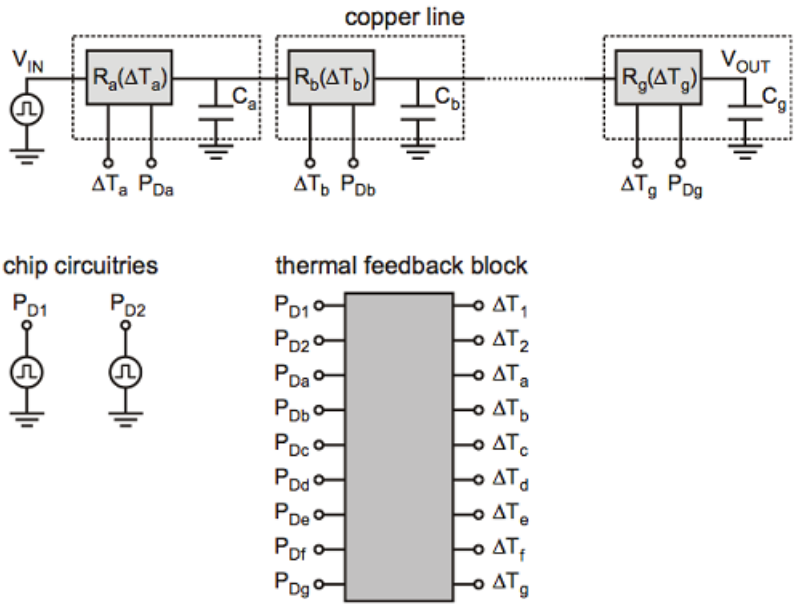


Figura 2.2: Approccio alla simulazione elettrotermica della linea di interconnessione

Questa procedura permette di rappresentare tutti gli effetti termici che influenzano la linea di interconnessione, vale a dire il riscaldamento reciproco dovuto alla dissipazione di potenza dei chips, l'auto-riscaldamento di ogni elemento per effetto Joule e l'accoppiamento termico tra gli elementi.

Questo aspetto risulta molto importante per la progettazione di un sistema di interconnessione dal punto di vista dell'integrità del segnale, poichè un significativo ritardo di propagazione termicamente indotto può comportare violazioni sui tempi o sullo skew del segnale di clock.

Lo stesso approccio può essere impiegato con successo per studiare il comportamento transitorio di amplificatori differenziali soggetti a effetti termici significativi [6][16]. Nelle moderne tecnologie bipolari si possono avere gravi effetti elettrotermici a causa delle grandi resistenze termiche dei singoli transistori bipolari, ad esempio come una radicale distorsione della caratte-

ristiche relative allo stato stazionario. In figura 2.3 è rappresentato lo schema di una coppia differenziale, accoppiata ad un blocco di feedback termico.

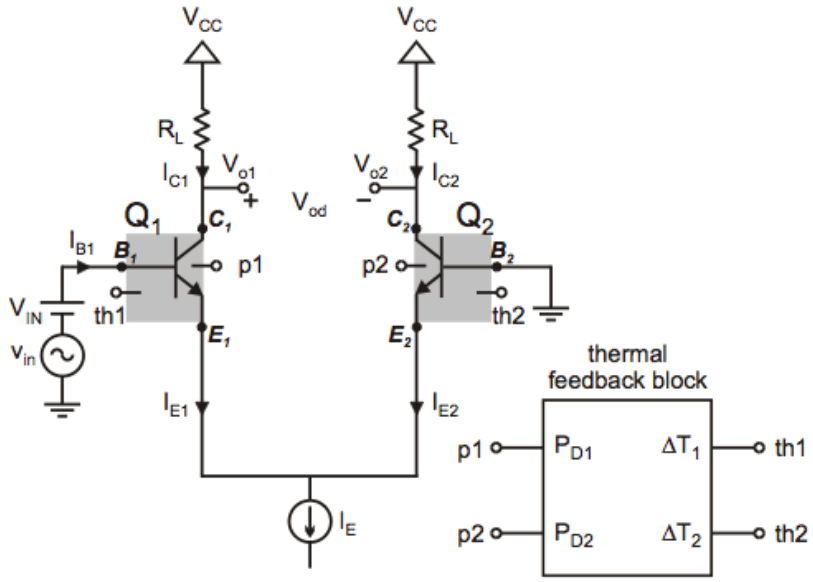


Figura 2.3: Schema di una coppia differenziale accoppiata ad un blocco di feedback termico

In figura 2.4 viene illustrata la tensione di uscita differenziale $V_{od} = V_{o1} - V_{o2}$ in funzione della tensione di ingresso V_{IN}, ottenuta tramite una simulazione elettrotermica statica. Il grafico mostra che la regione di funzionamento lineare della caratteristica di trasferimento, dove la coppia di transistori si comporta come un amplificatore, è sostituita da un ramo di resistenza positiva differenziale. La resistenza termica dei transistori Q_1 e Q_2 deve essere ridotta a valori $\leq 6300K/W$ per ripristinare la regione di funzionamento lineare desiderata. Gli effetti elettrotermici vengono restaurati connettendo la coppia di transistori al blocco di feedback termico, che si basa sulla rete multi-porta di Foster.

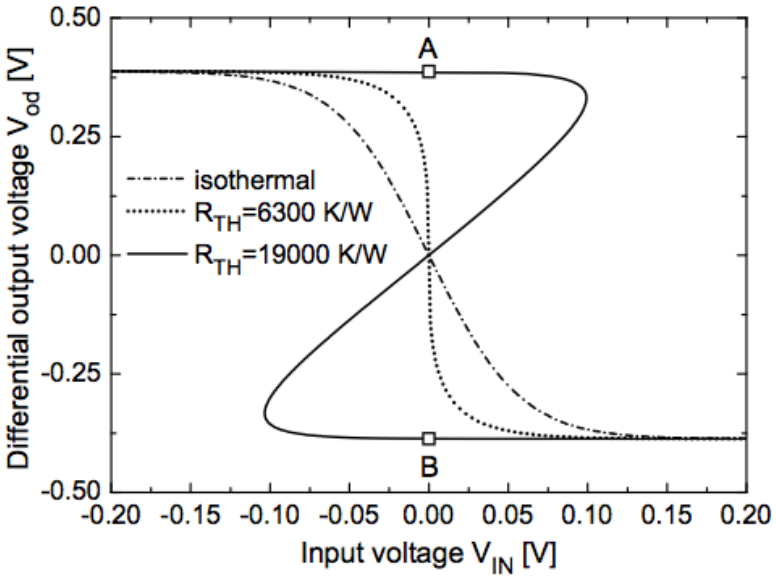


Figura 2.4: Tensione di uscita differenziale $V_{od} = V_{o1} - V_{o2}$ in funzione della tensione di ingresso V_{IN} sotto condizione elettrotermiche (linea continua e linea tratteggiata) ed isotermiche (linea punteggiata) a $T = 300K$

L'auto riscaldamento dei transistori modifica le loro proprietà elettriche, e questo può influenzare il corretto funzionamento e l'affidabilità del circuito. È quindi diventato importante la gestione degli effetti termici per la progettazione dei moderni sistemi elettronici [3], a livello di transistore, di chip, di package, di circuito stampato (PCB) e di sistema; grosse correnti durante scariche elettrostatiche possono aumentare la temperatura locale al punto da distruggere il transistore, così come è importante posizionare gli elementi sul chip in modo tale che gli effetti del riscaldamento siano minimi rispetto al suo funzionamento.

2.3 Resistenze ed impedenze termiche

I problemi termici possono essere descritti attraverso una equivalenza elettrica, in cui ad una dissipazione di potenza si fa corrispondere una corrente e al relativo incremento di temperatura corrisponde una tensione. Per un singolo dispositivo l'incremento di temperatura in DC, dovuto all'auto riscaldamento, può essere espresso in funzione della potenza dissipata e della *resistenza termica* attraverso l'equivalente termico della legge di Ohm [6]

$$\Delta T = T - T_0 = R_{TH} \cdot P_D \tag{2.4}$$

dove ΔT è la differenza di temperatura tra quella del dispositivo, T, e la temperatura di riferimento T_0, P_D è la potenza dissipata ed R_{TH} è la resistenza termica.
La resistenza termica [14] quantifica la capacità di un dato percorso termico di trasferire calore e la definizione generale, che include i tre differenti modi di trasferimento del calore (conduzione,convezione e irraggiamento), è il rapporto tra l'incremento di temperatura rispetto a quella di riferimento e il flusso di calore

$$R_{TH} = \frac{\Delta T}{\Delta P} \tag{2.5}$$

Nel transitorio, il comportamento termico è completamente descritto dall'*impedenza termica*, definita come la risposta termica all'applicazione di un gradino unitario di potenza nell'istante di tempo t

$$Z_{TH}(t) = \frac{T(t) - T_0}{P_D} \rightarrow \Delta T(s) = Z_{TH}(s) \cdot P_D \tag{2.6}$$

Gli effetti delle interazioni termiche tra due o più dispositivi sono tenu-

ti in conto attraverso la mutua resistenza/impedenza termica: ad esempio $Z_{ij}(t)$ è definito come il rapporto tra l'incremento di temperatura del i-esimo dispositivo dovuto all'applicazione di un gradino di potenza unitario del j-esimo dispositivo.

Il comportamento termico complessivo di un dispositivo con più fonti di calore può essere caratterizzato attraverso la matrice delle resistenze/impedenze termiche, quindi considerando due dispositivi in DC abbiamo

$$\begin{aligned} \Delta T_1 &= R_{11}P_{D1} + R_{12}P_{D2} \\ \Delta T_2 &= R_{21}P_{D1} + R_{22}P_{D2} \end{aligned} \tag{2.7}$$

da cui

$$R_{TH} = \begin{pmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{pmatrix} \tag{2.8}$$

Nel transitorio invece

$$\begin{aligned} \Delta T_1(s) &= Z_{11}(s)P_{D1}(s) + Z_{12}(s)P_{D2}(s) \\ \Delta T_2(s) &= Z_{21}(s)P_{D1}(s) + Z_{22}(s)P_{D2}(s) \end{aligned} \tag{2.9}$$

e quindi

$$Z_{TH} = \begin{pmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{pmatrix} \tag{2.10}$$

2.4 Identificazione di modelli elettrotermici ridotti

Data la complessità dei moderni circuiti elettronici, è ormai indispensabile il ricorso all'uso di software per l'analisi e la simulazione. Inoltre prima di iniziare una simulazione è indispensabile, per ragioni di efficienza computazionale, ricorrere a tecniche di riduzione d'ordine dei modelli in maniera tale da ottenere circuiti equivalenti facilmente implementabili in simulatori circuitali standard, garantendo una buona approssimazione nel dominio di interesse e conservando le proprietà principali del sistema.
Il processo di identificazione per sistemi lineari consiste nel cercare una forma approssimata per la matrice o funzione di trasferimento del sistema. L'espressione trovata analiticamente è in termini di rapporto tra polinomi o in termini di poli e residui e deve approssimare in maniera accurata la risposta in frequenza del sistema, e soddisfare le proprietà fisiche e le specifiche del sistema di partenza. Tra gli algoritmi proposti in letteratura per la procedura di identificazione troviamo il *Vector Fitting* [2], il quale è un algoritmo iterativo basato sulla ricollocazione dei poli; ad ogni iterazione viene risolto un problema lineare, fino a quando non si raggiunge la migliore accuratezza possibile. Questo algoritmo suddivide il problema di partenza non lineare in due stadi, ognuno dei quali risolve un problema lineare. Riferendoci all'algoritmo del Vector Fitting per l'espansione approssimata di una funzione, si giunge a questo tipo di approssimazione per la $f(s)$

$$f(s) = \sum_{m=1}^{N} \frac{c_m}{s - p_m} + d + sh \tag{2.11}$$

Le incognite da determinare sono c_m, p_m, d, h e la non linearità del problema sta nella presenza delle incognite p_m al denominatore. Il problema però è suddivisibile in due problemi lineari: in un primo stadio si determinano i

poli della $f(s)$ (Vector Fitting 1° stadio), mentre nel secondo si calcolano i residui della $f(s)$, fissando i poli trovati nello step precedente (Vector Fitting 2° stadio). Questa procedura porta ad una funzione con poli stabili ma non assicura la passività, proprietà indispensabile al successo di una simulazione di una rete di grandi dimensioni, poichè anche se un circuito è stabile ma non passivo, inserito in un macro-modello può generare un comportamento instabile. Per tanto si ricorre alle metodologie di forzamento della passività. Esistono due tipologie di approcci nell'identificazione passiva: un approccio a-posteriori che consiste nel correggere le violazioni di passività del modello identificato con tecniche perturbative [1][13], e un approccio a-priori nel quale si giunge ad un modello passivo imponendo dei vincoli durante la fase di identificazione[4][10].

Un approccio a-priori è il *Positive Fraction Vector Fitting* [11] [10]. Per questa procedura, i poli sono identificati mediante il Vector Fitting [9], formulando il calcolo dei residui come un problema di ottimizzazione convessa, e la matrice che si ottiene è un'espansione in frazione positiva. Lo scopo ultimo di questa procedura è la realizzazione passiva di modelli ridotti associata all'espansione di Foster generalizzata.

2.5 Realizzazione dell'impedenza termica con la rete di Foster

Date in ingresso le caratteristiche ai terminali di un generico sistema multi-porta lineare, si può ottenere una sua rappresentazione matematica in termini di matrici ABCD o di matrice di trasferimento con degli algoritmi che prendono il nome di **procedure di identificazione**. A partire dalla rappresentazione matematica identificata, è possibile disegnarne un modello

circuitale equivalente, utile per l'integrazione in fase di simulazione di porzioni diverse dello stesso sistema. Si parla in questo caso di **sintesi circuitale**. Il PFVF (Positive Fraction Vector Fitting) è una procedura di identificazione, basata sul Vector Fitting con l'aggiunta della imposizione a-priori della passività. Scopo ultimo è quello di approssimare la H(s) con l'espansione in poli e residui

$$H(s) = R_0 + \sum_{n=1}^{N_{cp}} \left[\frac{r_n}{s - p_n} + \frac{r_n^*}{s - p_n^*} \right] A_n + \sum_{n=1}^{Nrp} \frac{r_n}{s - p_n} A_n \tag{2.12}$$

R_0 e A_n sono matrici quadrate M x M (M è il numero di porte del sistema) , reali simmetriche e definite positive; N_{cp} è il numero di coppie di poli complessi coniugati, N_{rp} è il numero di poli reali. Supponendo che siano presenti solo poli reali, come accade per gli equivalenti dei problemi termici, la matrice di trasferimento è della forma

$$H(s) = \sum_{n=1}^{N_p} \frac{A_n}{s - p_n} \tag{2.13}$$

N_p è il numero di poli reali e stabili; p_n è un vettore di N_p poli; A_n sono N_p matrici M x M di residui. La H(s) può essere la matrice delle impedenze Z(s) o la matrice delle ammettenze Y(s) o al più la matrice ibrida. Si può dimostrare [12] che tale espressione la si può far corrispondere, manipolando la matrice di trasferimento e sotto opportune ipotesi, allo schema circuitale di figura 2.5, come spiegato in dettaglio nel corso del paragrafo.

2.5.1 Descrizione del codice in Matlab

Andiamo ora a descrivere passo per passo la procedura di sintesi circuitale che è stata implementata in Matlab. Faremo riferimento in particolare alla

sintesi di una matrice di impedenze termiche. Innanzitutto descriviamo lo schema circuitale ed i passaggi matematici per ottenerlo; in seguito, si mostra la traduzione dello schema circuitale in netlist per uno specifico simulatore circuitale (Orcad PSPICE).

Come primo passo inizializziamo il workspace:

```
clear
close all
clc
format long
warning off
```

La routine di sintesi accetta in ingresso una struttura PFVFId, fornita dalla routine di identificazione, contente N_p matrici M x M di residui (PFVFId.residue) e il vettore degli N_p poli (PFVFId.poles) :

```
M = size(PFVFId.residue,1);
Np = length(PFVFId.poles);
```

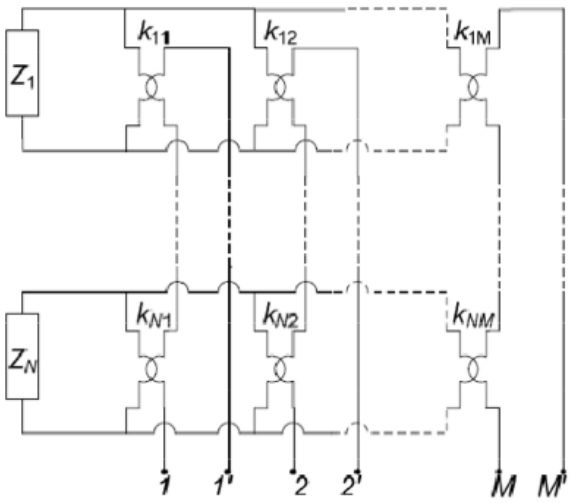


Figura 2.5: Schema di sintesi per l'impedenza relativa a una rete con N blocchi ed M porte

Una volta determinata la matrice delle impedenze termica nella forma (2.13), la procedura di sintesi richiede un opportuna diagonalizzazione di ogni matrice A_n per poi esprimerla in somma di matrici a rango unitario. Supponiamo che la matrice A_n sia diagonalizzabile, ovvero che esista una trasformazione lineare reale T_n tale che

$$A_n = T_n^{-1} A_n^* T_n \tag{2.14}$$

$$A_n^* = T_n^{-1} A_n T_n \tag{2.15}$$

dove A_n^* (indicata nel codice con D) è una matrice diagonale

$$A_n^* = \begin{pmatrix} a_{n,1}^* & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & a_{n,M}^* \end{pmatrix} \tag{2.16}$$

i cui coefficienti non nulli sono gli autovalori della matrice A_n^*

$$(A_n - a_{n,m}^* I) u_{n,m} = 0 \qquad m = 1, ...M \tag{2.17}$$

La matrice di trasformazione T_n è la matrice colonna degli autovettori $u_{n,i}$

$$T_n = [u_{n,1}, u_{n,2}, ..., u_{n,M}] \tag{2.18}$$

```
for i=1:length(PFVFId.poles)
    [T,D]=eig(PFVFId.residue(:,:,i));
```

Esprimiamo ora la matrice A_n in somma di matrici a rango unitario. Essendo la matrice di trasformazione costruita per diagonalizzare una matrice reale e simmetrica, T_n è ortogonale quindi la sua trasposta e la sua inversa coincidono e pertanto nel codice consideriamo la sua trasposta. La matrice A_n^* può essere scomposta in somma di matrici diagonali con un solo termine non nullo

$$A_n^* = \sum_{m=1}^{M} A_{n,m}^* \tag{2.19}$$

dove

$$A_{n,m}^* = \begin{pmatrix} 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & a_{n,m}^* & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 \end{pmatrix} \tag{2.20}$$

In modo analogo si può scomporre anche la matrice A_n in somma di matrici a rango 1

$$A_n = \sum_{m=1}^{M} A_{n,m} \qquad A_{n,m} = T_n^{-1} A_{n,m}^* T_n \tag{2.21}$$

```
for i=1:length(PFVFId.poles)
    [T,D]=eig(PFVFId.residue(:,:,i));
    for k=1:M
        temp = zeros(M,M);
        temp(k,k) = D(k,k);
        PFVFId.residuediag(:,:,i,k) = T*temp*T';
```

```
    end
end
```

È possibile verificare quanto fatto nel passo precedente, definendo una matrice C di ”confronto” M x M che viene utilizzata per controllare che non siano stati fatti errori nel processo di diagonalizzazione:

```
for i=1:Np
    C = zeros(M,M);
    for k=1:M
        C = C + squeeze(PFVFId.residuediag(:,:,i,k));
    end
    isequal(C,squeeze(PFVFId.residue(:,:,i)))
end
```

Per un’impedenza relativa ad un polo reale e al suo corrispondente residuo bisogna considerare una cella costituita dal parallelo di una capacità e una conduttanza, come in figura 2.6.

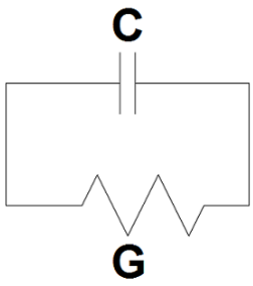


Figura 2.6: Impedenza Z_n per polo reale

Tale cella va poi connessa con un numero M-1 di trasformatori ideali. Poichè la matrice delle impedenze Z(s) è somma dei singoli termini dell’espansione (2.13), dobbiamo considerare tutte queste celle in serie ottenendo così il circuito di sintesi di figura 2.7 nel caso di poli reali.

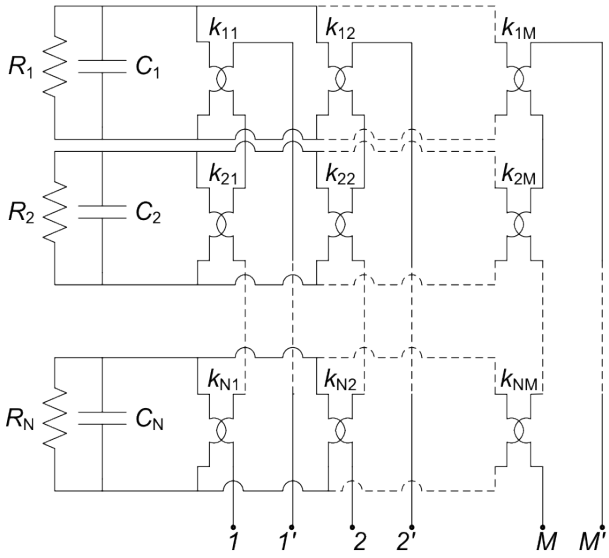


Figura 2.7: Circuito di sintesi per la matrice Z(s) nel caso di poli reali

Quindi il prossimo passo sarà quello di trovare i valori di resistenza e capacità e scrivere una matrice che contenga i vari rapporti di trasformazione. Cominciamo col sintetizzare i cappi RC da mettere in ingresso ai vari trasformatori; costruiamo due vettori colonna R e C di tutti zeri e dimensione M*N_p i quali saranno riempiti con i valori di resistenza e capacità da calcolare. Il polo p_n e il residuo r_n sono legati alla conduttanza G e alla capacità C dalle seguenti relazioni

$$Z_n = \frac{1}{G + sC} = \frac{r_n}{s - p_n} \qquad C = \frac{1}{r_n} \qquad G = -\frac{p_n}{r_n} \rightarrow R = -\frac{r_n}{p_n} \tag{2.22}$$

```
C=zeros(M*Np,1); R=zeros(M*Np,1);
ind=1;
for i=1:length(PFVFId.poles)
    for k=1:M
```

```
        C(ind) = 1/PFVFId.residuediag(1,1,i,k);
        R(ind) = -PFVFId.residuediag(1,1,i,k)/PFVFId.poles(i); ind
            =ind+1;
    end end
tau = R.*C;
```

Successivamente, andiamo a realizzare la matrice che contiene i vari rapporti di trasformazione, raccogliendo il primo elemento della matrice

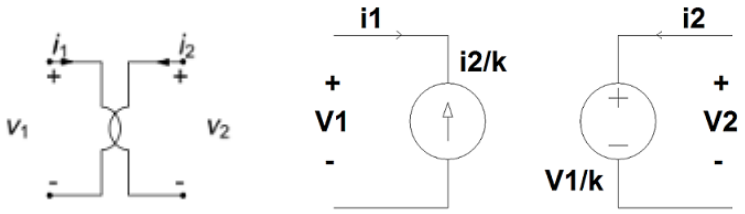
$$\begin{pmatrix} 1 & k_{n,1,1} & \cdots & k_{n,1,M-1} \\ k_{n,1,1} & k_{n,1,1}^2 & \cdots & k_{n,1,1}k_{n,1,M-1} \\ \vdots & \vdots & \ddots & \vdots \\ k_{n,1,M-1} & k_{n,1,1}k_{n,1,M-1} & \cdots & k_{n,1,M-1}^2 \end{pmatrix}_m \tag{2.23}$$

```
kncir = zeros(M*Np,M-1);
indinterno = 1;
for i=1:length(PFVFId.poles)
    for k=1:M
        PFVFId.residuediag(:,:,i,k)=
        PFVFId.residuediag(:,:,i,k)/PFVFId.residuediag(1,1,i,k);
        kncir(indinterno,:) = 1./PFVFId.residuediag(1,2:end,i,k);
        indinterno=indinterno+1;
    end
end
```

Con i valori così calcolati è possibile implementare il circuito in un qualsiasi simulatore circuitale. Come esempio, è stata implementata la generazione automatica della netlist del circuito di figura 2.7 per Orcad PSPICE. In particolare si va a definire un elemento di libreria (file .lib) che può essere facilmente inserito in uno schematic più complesso sotto forma di "blocchetto" gerarchico. Innanzitutto osserviamo che in PSPICE non esiste il componen-

te trasformatore ideale. Riordinando le equazioni del trasformatore ideale di rapporto di trasformazione n, si vede che questo è equivalente ad un GCCC alla porta 1 controllato dalla corrente della porta 2 ed ad un GTCT alla porta 2 controllato dalla tensione alla porta 1 (figura 2.8)

$$\begin{cases} V_1 = kV_2 \\ i_2 = -ki_1 \end{cases} \rightarrow \begin{cases} V_2 = \frac{1}{k}V_1 \\ i_1 = -\frac{1}{k}i_2 \end{cases} \tag{2.24}$$



(a) Trasformatore ideale

(b) Equivalente del trasformatore ideale con generatori controllati

Figura 2.8: Implementazione in PSPICE del trasformatore ideale

I valori per k sono contenuti nella matrice (2.23) trovata in precedenza. Cominciamo la scrittura del file .lib definendo il nome del componente e dei pin di ingresso e uscita: per una rete termica consideriamo come ingressi le potenze (correnti equivalenti) e come uscite le temperature (tensioni equivalenti):

```
M_str = num2str(M);
nomecomponente = 'Multiporta';
netlist = ['* THERMAL Network M = ' M_str ' \n'];
netlist = [netlist '.SUBCKT ' nomecomponente ' DT1 \n'];
for i=2:M
    netlist = [netlist '+ DT' num2str(i) ' \n'];
end
for i=1:M
    netlist = [netlist '+ PDIN' num2str(i) ' \n'];
```

```
end
```

Per la porta 1, i nodi $PDIN1$ e $DT1$ di ingresso e uscita rispettivamente sono in realtà lo stesso nodo. Quindi per distinguerli effettuiamo un processo di *replicazione di corrente* (figura 2.9) per cui alla porta 1 replichiamo la corrente in ingresso ai cappi; a tale scopo inseriamo un generatore di tensione nullo tra il nodo $PDIN1$ e 0, VF_TH_F1, e replichiamo la corrente in ingresso con un generatore di corrente controllato in corrente tra il nodo $DT1$ e 0, la corrente replicata andrà in ingresso ai cappi del lato sinistro dello schema di figura 2.7.

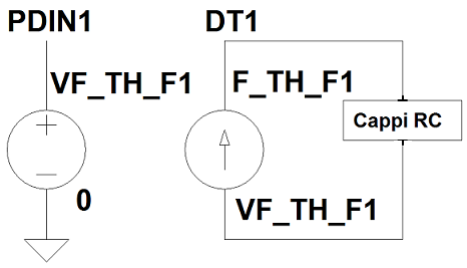


Figura 2.9: Replicazione di corrente

```
nodointerno='TH';
netlist = [netlist '* INGRESSO PORTA 1 - Replicazione di corrente
    \n' ];
netlist = [netlist 'F_' nodointerno '_F1 DT1 0 ' ...
    'VF_' nodointerno '_F1 -1 \n'];
netlist = [netlist 'VF_' nodointerno '_F1 ' ...
    ' PDIN1 0 0V \n'];
```

A questo punto passiamo a definire gli MxN_p cappi RC. Prima di tutto scriviamo i nodi del lato sinistro dello schema di figura 2.7 : indichiamo questi ultimi con nodi A e sono $DT1$, $A2$, $A3$, ... ,$AMxN_p$, 0.
Quindi definiamo le resistenze e le capacità dei vari cappi le quali saranno

compresi, a partire dal primo cappio, tra $DT1$ e il nodo TH_A2, per il secondo tra TH_A2 e TH_A3 e così via e assegniamo a queste i valori trovati nei passaggi precedenti (figura 2.10):

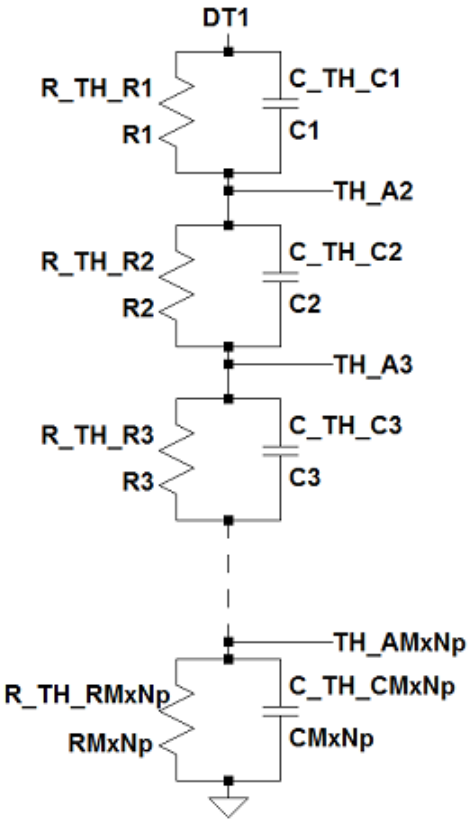


Figura 2.10: Schema dei cappi RC

```
netlist = [netlist '* CAPPI ALLA PORTA 1 \n']; nodi_A = {'DT1'};
for i=2:(M*Np)
    nodi_A{i}= [nodointerno '_A' num2str(i)];
end
nodi_A{M*Np+1}=num2str(0);
for k=1:(M*Np)
    k_str = num2str(k);
```

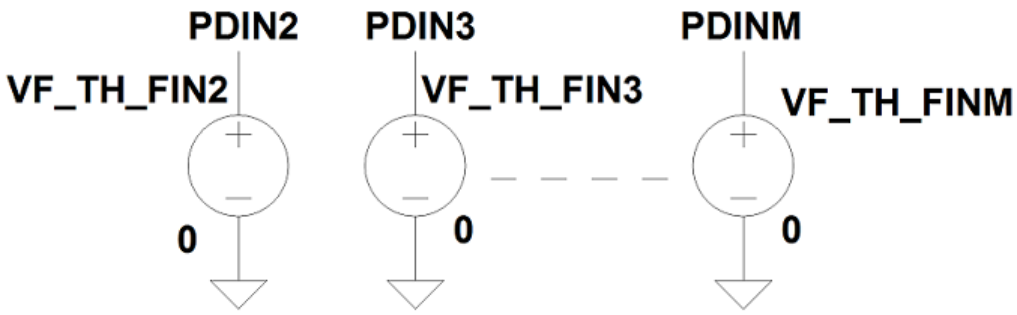


Figura 2.11: Generatori di tensione nulla alle porte 2 ... M

```
    netlist = [netlist 'R_' nodointerno '_R' k_str ' ' nodi_A{k} '
        ' nodi_A{k+1} ' ' num2str(R(k),'%0.16e') ' \n'];
    netlist = [netlist 'C_' nodointerno '_C' k_str ' ' nodi_A{k} '
        ' nodi_A{k+1} ' ' num2str(C(k),'%0.16e') ' \n'];
end
```

Adesso ci resta da realizzare i trasformatori utilizzando i generatori controllati secondo la (2.24). Aggiungiamo i generatori di tensione nulla per leggere le altre correnti in ingresso alle porte 2 ... M (figura 2.11):

```
netlist = [netlist '* Aggiunte alla porta 1 dovute alle porte 2...
    M \n'];
for indm = 2:M
    indm_str = num2str(indm);
    netlist = [netlist '* -> dovuta alla PORTA ' indm_str ' \n'];
    netlist = [netlist 'VF_' nodointerno '_Fin_' indm_str ' ' ...
        'PDIN' num2str(indm) ' 0 0V \n'];
```

Realizziamo quindi la "porzione" di trasformatori alla porta 1 inserendo in parallelo ai cappi RC i generatori di corrente controllati dalle correnti in ingresso (figura **??**):

```
for k=1:(M*Np)
    k_str = num2str(k);
```

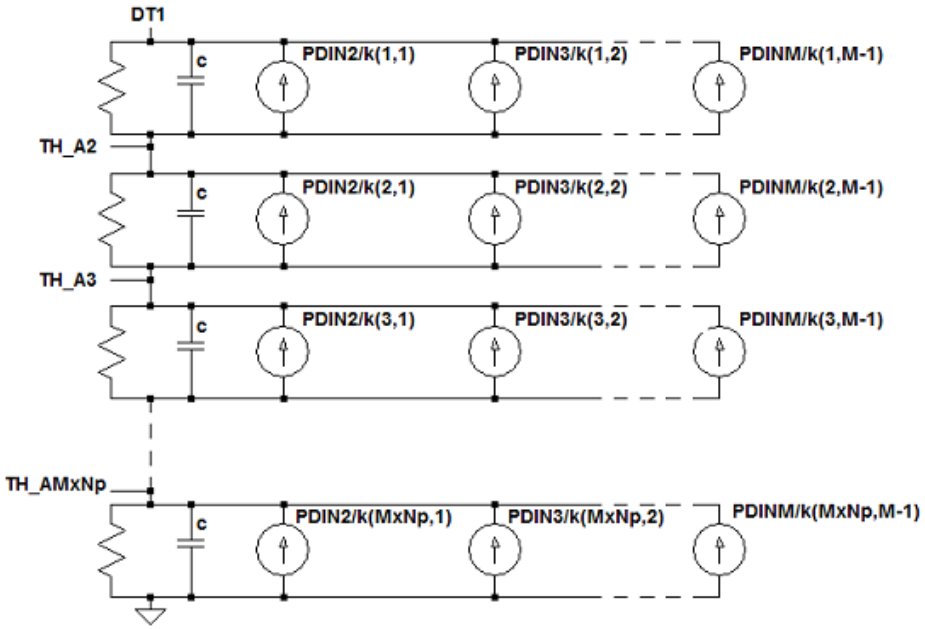


Figura 2.12: Realizzazione del lato sinistro del trasformatore

```
    netlist = [netlist 'F_' nodointerno '_F' k_str '_' indm_str '
        ' ...
        nodi_A{k} ' ' nodi_A{k+1} ' ' ...
        'VF_' nodointerno '_Fin_' indm_str ' ' ...
        num2str(-1/kncir(k,indm-1),'%0.16e') ' \n'];
end
end
```

Dobbiamo ora realizzare la seconda ”porzione” dei trasformatori, ovvero i generatori di tensione controllati in tensione, con i trasformatori della porta i-sima tutti in serie. Per prima cosa definiamo i nodi interni tra i quali bisogna inserire i generatori: per la porta DT2 saranno NintDT12 , NintDT22, ... , NintDT(MxNp)2, 0 e così via per le restanti porte:

```
netlist = [netlist '* Le altre 2...M porte \n'];
for indm = 2:M
```

```
indm_str = num2str(indm);
netlist = [netlist '* -> PORTA ' indm_str ' \n'];
nodi_N = {['DT' indm_str]};
for in=2:(M*Np)
    nodi_N{in}= ['NintDT_' num2str(in-1) '_' indm_str];
end
nodi_N{M*Np+1}=num2str(0);
```

Inseriamo, quindi, i generatori di tensione che saranno compresi tra i nodi interni definiti immediatamente prima e controllati dai generatori di corrente alla porta 1 (figura 2.13):

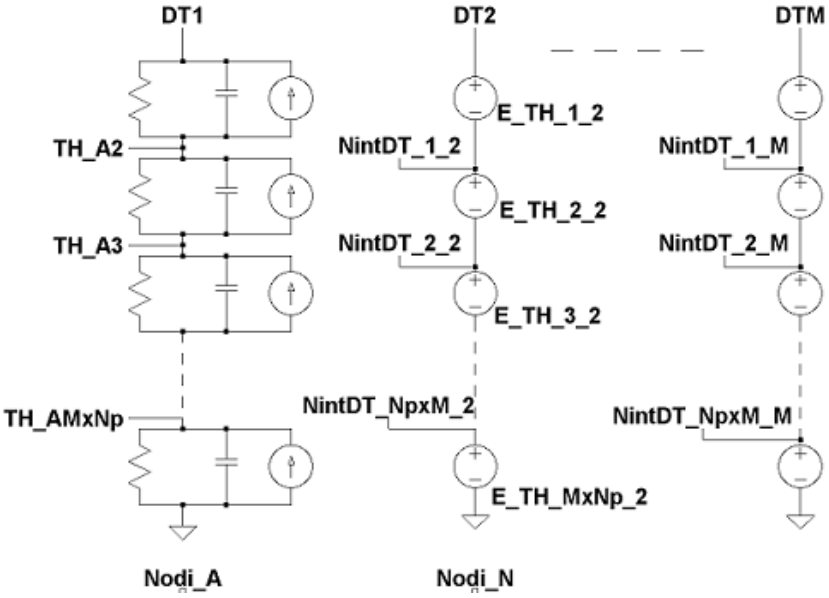


Figura 2.13: Realizzazione del lato destro del trasformatore

```
netlist = [netlist '* Le altre 2...M porte \n'];
for indm = 2:M
```

```
    for k=1:(M*Np)
        k_str = num2str(k);
        netlist = [netlist 'E_' nodointerno '_E' k_str '_'
            indm_str ' ' ...
            nodi_N{k} ' ' nodi_N{k+1} ' ' ...
            nodi_A{k} ' ' nodi_A{k+1} ' ' ...
            num2str(1/kncir(k,indm-1),'%0.16e') ' \n'];
    end
end
```

Infine terminiamo la netlist e salviamo il file:

```
netlist = [netlist '* TERMINAZIONE \n'];
netlist = [netlist ' .ENDS \n'];
fid = fopen ([nomecomponente '.lib'],'wt');
fprintf(fid, netlist);
fclose(fid);
```

Capitolo 3

Problemi di onere computazionale in SPICE delle reti di feedback elettrotermico

In questo capitolo sono valutate le prestazioni di Spice nell'utilizzo di diversi tipi di generatori controllati, confrontando differenti topologie di rete a parità di numero di nodi. Il confronto viene effettuato rispetto ad una rete di riferimento di soli resistori lineari, dopodichè sono valutati i tempi di risoluzione delle diverse topologie in Spice, prima nel caso statico e poi in quello dinamico; affinchè il confronto tra i due casi sia omogeneo , questo è effettuato a **parità di nodi** e a **parità di numero di time steps** (scegliendo opportunamente le costanti di tempo nel caso dinamico).

3.1 Confronto a parità di numero di nodi

Consideriamo innanzitutto il caso più semplice per la rete di riferimento: una rete di soli resistori lineari e con 2 nodi (figura 3.1).

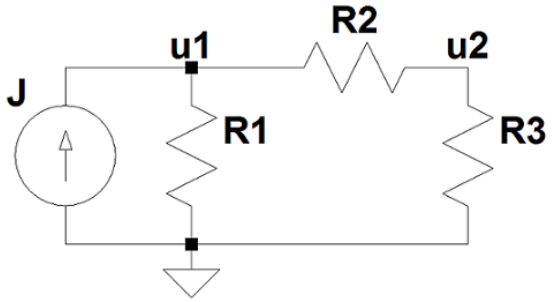


Figura 3.1: Rete di riferimento con soli resistori

Applicando il metodo dei potenziali di nodo, cioè scrivendo le KCL ai nodi e sostituendo le caratteristiche dei resistori, si ottiene un sistema lineare nelle due incognite u_1 e u_2

$$\begin{aligned} \left(\frac{1}{R_1}+\frac{1}{R_2}\right)u_1 - \frac{1}{R_2}u_2 &= J \\ -\frac{1}{R_2}u_1 + \left(\frac{1}{R_2}+\frac{1}{R_3}\right)u_2 &= 0 \end{aligned} \tag{3.1}$$

Da cui, posto per semplicità $R_1 = R_2 = R_3 = 1\Omega$, $J = 1A$, si ottiene

$$\begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{3.2}$$

È possibile estendere la rete precedentemente considerata ad un numero di nodi maggiore, ad esempio 4 (figura 3.2).

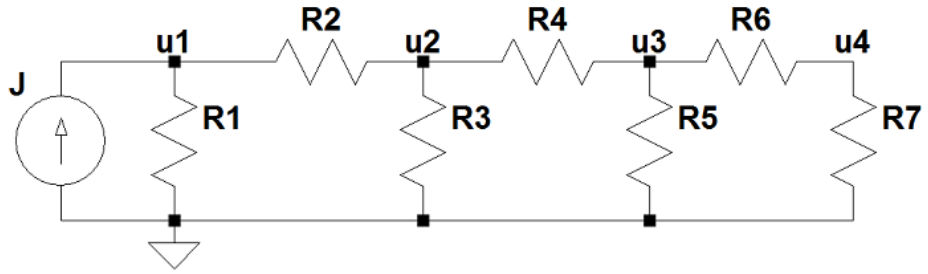


Figura 3.2: Rete di riferimento con 4 nodi

In questo modo il sistema diventa a quattro equazioni e quattro incognite

$$\begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 3 & -1 & 0 \\ 0 & -1 & 3 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{3.3}$$

È semplice generalizzare ad un numero n arbitrario di nodi

$$\begin{pmatrix} 2 & -1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ -1 & 3 & -1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & -1 & 3 & -1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & -1 & 3 & -1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & -1 & 3 & -1 & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & \cdots & 0 & 0 & -1 & 2 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_n \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ \vdots \\ 0 \end{pmatrix} \tag{3.4}$$

Definiamo la densità della matrice come il rapporto tra il numero di elementi non nulli e il numero totale di elementi. La sparsità di una matrice è invece il rapporto tra il numero di elementi nulli e il numero totale di elementi, ed è pari ad uno meno la densità. Quindi per un numero n di nodi, la sparsità della matrice per la rete di riferimento si scrive come

$$sparsita' = 1 - \frac{3n-2}{n^2} \tag{3.5}$$

Il confronto delle prestazioni tra i generatori controllati è fatto, assumendo come riferimento la rete di resistori lineari, costruendo delle reti che abbiano lo stesso numero di nodi ma nelle quali siano presenti i quattro tipi di generatori controllati. Si va ad osservare che, in funzione del tipo di generatori controllati, a parità di nodi si ottiene un numero diverso di incognite.

Generatore di corrente controllato in tensione

Il primo caso da analizzare è il generatore di corrente controllato in tensione, la rete da considerare è quella in figura 3.3.

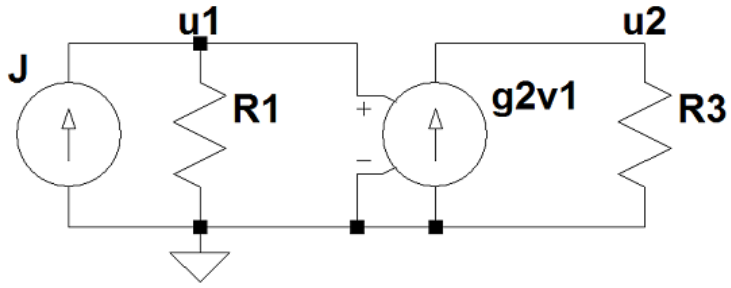


Figura 3.3: Rete con generatore di corrente controllato in tensione

Applicando il metodo dei potenziali di nodo si ottiene

$$\begin{aligned} \frac{1}{R_1}u_1 + 0u_2 &= J \\ g_2u_1 + \frac{1}{R_3}u_2 &= 0 \end{aligned} \tag{3.6}$$

poniamo , come nel caso dei resistori lineari, $R_1 = R_3 = 1\Omega$, $g = 1\Omega^{-1}$, $J = 1A$

e quindi

$$\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{3.7}$$

Confrontandola con la (3.2), il numero di equazioni ed incognite è lo stesso.

Anche in questo caso è possibile estendere la rete ad un numero di nodi maggiori, ad esempio 4 (figura 3.4).

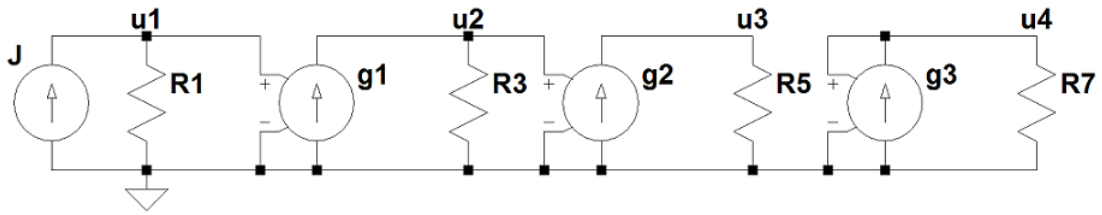


Figura 3.4: Rete con GCCT e 4 nodi

E quindi si ottiene sempre un sistema di quattro equazioni e quattro

incognite

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{3.8}$$

Generalizzando ad un numero n arbitrario di nodi

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_n \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ \vdots \\ 0 \end{pmatrix} \tag{3.9}$$

Il numero di elementi non nulli è inferiore rispetto a quello delle matrici per la rete di riferimento, pertanto la sparsità è minore. Infatti per un numero n arbitrario di nodi, si scrive come segue

$$sparsita' = 1 - \frac{2n-1}{n^2} \tag{3.10}$$

Generatore di tensione controllato in tensione

Consideriamo la stessa rete del caso precedente con un generatore di tensione controllato in tensione, come in figura 3.5.

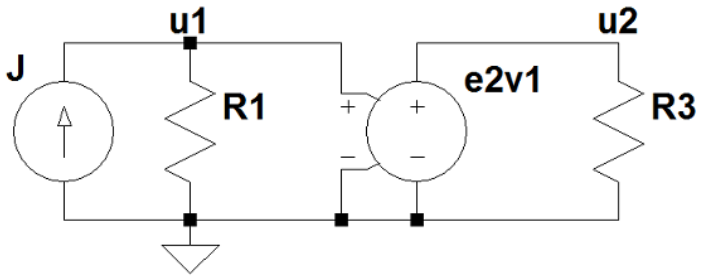


Figura 3.5: Rete con generatore di tensione controllato in tensione

Applicando il metodo dei potenziali di nodo e ponendo poi tutte le grandezze in gioco pari ad uno, si ottiene

$$\begin{aligned} \frac{1}{R_1}u_1 + 0u_2 &= J \\ 0u_1 + \frac{1}{R_3}u_2 + i_{e_2} &= 0 \\ -e_2u_1 + u_2 &= 0 \end{aligned} \tag{3.11}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ -1 & 1 & 0 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ i_{e_2} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \tag{3.12}$$

Rispetto al caso di riferimento, la rete con il generatore di tensione controllato in tensione aggiunge un'incognita al sistema, ovvero la corrente nel generatore di tensione. Estendendo la rete a quattro nodi, e quindi con tre

GTCT, otteniamo un sistema di sette equazioni e sette incognite

$$\begin{aligned}
\frac{1}{R_1}u_1 + 0u_2 + 0u_3 + 0u_4 &= J \\
0u_1 + \frac{1}{R_3}u_2 + 0u_3 + 0u_4 + i_{e_2} &= 0 \\
0u_1 + 0u_2 + \frac{1}{R_5}u_3 + 0u_4 + 0 + i_{e_3} &= 0 \\
0u_1 + 0u_2 + 0u_3 + \frac{1}{R_7}u_4 + 0 + 0 + i_{e_4} &= 0 \\
-e_2u_1 + u_2 &= 0 \\
-e_3u_2 + u_3 &= 0 \\
-e_4u_2 + u_4 &= 0
\end{aligned} \tag{3.13}$$

il quale, tradotto in forma matriciale, è nella forma

$$\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 \\
-1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 1 & 0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
u_1 \\ u_2 \\ u_3 \\ u_4 \\ i_{e_2} \\ i_{e_3} \\ i_{e_4}
\end{pmatrix} =
\begin{pmatrix}
1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0
\end{pmatrix} \tag{3.14}$$

Rispetto al caso di riferimento, si aggiungono tre incognite dovute alle

correnti che scorrono nei generatori di tensione. Per n nodi, il sistema diventa

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \vdots & \cdots & 0 \\ 0 & 0 & 0 & \cdots & -1 & 1 & 0 & \cdots & 0 \end{pmatrix} \begin{pmatrix} u_1 \\ \vdots \\ \vdots \\ u_n \\ i_{e_2} \\ \vdots \\ i_{e_n} \end{pmatrix} = \begin{pmatrix} 1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ 0 \end{pmatrix} \tag{3.15}$$

La sparsità per questa matrice risulta essere

$$sparsita' = 1 - \frac{4n-3}{(2n-1)^2} \tag{3.16}$$

Generatore di tensione controllato in corrente

Ripetiamo gli stessi procedimenti nel caso di un generatore di tensione controllato in corrente, come in figura 3.6.

$$\begin{aligned} \frac{1}{R_1}u_1 + 0u_2 &= J \\ 0u_1 + \frac{1}{R_3}u_2 + i_{r_2} &= 0 \\ -r_2\frac{u_1}{R_1} + u_2 &= 0 \end{aligned} \tag{3.17}$$

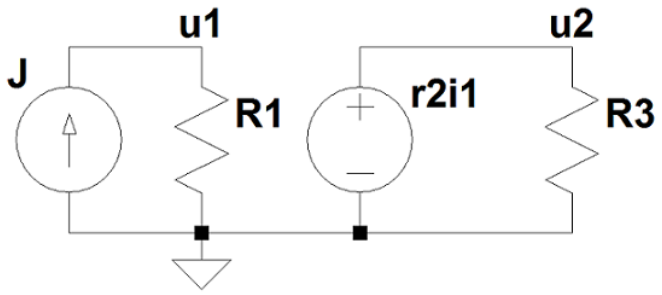


Figura 3.6: Rete con generatore di tensione controllato in corrente

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ -1 & 1 & 0 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ i_{r_2} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \tag{3.18}$$

Anche in questo caso si aggiunge un'incognita al sistema, la corrente che scorre nel generatore di tensione. Nel caso di quattro nodi, e quindi tre GTCC, si ha un sistema di sette equazioni e sette incognite

$$\begin{aligned} \frac{1}{R_1}u_1 + 0u_2 + 0u_3 + 0u_4 &= J \\ 0u_1 + \frac{1}{R_3}u_2 + 0u_3 + 0u_4 + i_{r_2} &= 0 \\ 0u_1 + 0u_2 + \frac{1}{R_5}u_3 + 0u_4 + 0 + i_{r_3} &= 0 \\ 0u_1 + 0u_2 + 0u_3 + \frac{1}{R_7}u_4 + 0 + 0 + i_{r_4} &= 0 \\ -r_2\frac{u_1}{R_1} + u_2 &= 0 \\ -r_3\frac{u_2}{R_3} + u_3 &= 0 \\ -r_4\frac{u_2}{R_5} + u_4 &= 0 \end{aligned} \tag{3.19}$$

e quindi

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_6 \\ i_{r_2} \\ i_{r_3} \\ i_{r_4} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{3.20}$$

La matrice ottenuta è identica al caso del GTCT, di conseguenza per un numero n di nodi arbitrario la matrice è la stessa della (3.15), e considerazioni analoghe si possono fare sulla sparsità della matrice.

Generatore di corrente controllato in corrente

Come ultimo caso dobbiamo considerare l'utilizzo di un generatore di corrente controllato in corrente, come in figura 3.7.

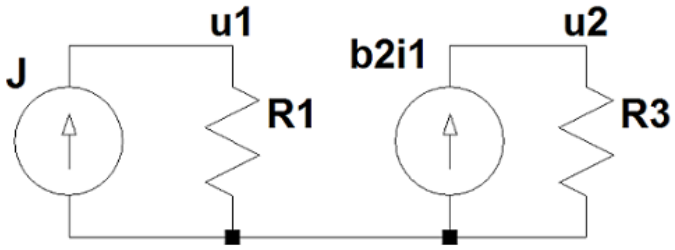


Figura 3.7: Rete con generatore di corrente controllato in corrente

Applicando il metodo dei potenziali di nodo e facendo le semplificazioni

fatte in tutti i casi precedenti, si ottiene

$$\begin{aligned} \frac{1}{R_1}u_1 + 0u_2 &= 0 \\ 0u_1 + \frac{1}{R_3}u_2 + i_{b_2} &= 0 \\ -b_2\frac{u_1}{R_1} + u_2 &= 0 \end{aligned} \tag{3.21}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ -1 & 1 & 0 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ i_{b_2} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \tag{3.22}$$

estendendo la rete a 4 nodi il sistema diventa

$$\begin{aligned} \frac{1}{R_1}u_1 + 0u_2 + 0u_3 + 0u_4 &= J \\ 0u_1 + \frac{1}{R_3}u_2 + 0u_3 + 0u_4 + i_{b_2} &= 0 \\ 0u_1 + 0u_2 + \frac{1}{R_5}u_3 + 0u_4 + 0 + i_{b_3} &= 0 \\ 0u_1 + 0u_2 + 0u_3 + \frac{1}{R_7}u_4 + 0 + 0 + i_{b_4} &= 0 \\ -b_2\frac{u_1}{R_1} + u_2 &= 0 \\ -b_3\frac{u_2}{R_3} + u_3 &= 0 \\ -b_4\frac{u_2}{R_5} + u_4 &= 0 \end{aligned} \tag{3.23}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_6 \\ i_{b_2} \\ i_{b_3} \\ i_{b_4} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{3.24}$$

Il risultato è anche in questo caso, identico a quello ottenuto con i GTCT e i GTCC.

3.2 Simulazione statica

A questo punto bisogna valutare e confrontare i tempi di risoluzione in Spice tra la rete di riferimento e quella con i generatori controllati: per fare questo utilizziamo una funzione Matlab che generi automaticamente le netlist dei circuiti per un numero di nodi arbitrario.

Partiamo con la rete di riferimento e quindi consideriamo la netlist della rete di figura 3.8:

```
I1 N001 0 1
R1 N001 0 1
R3 N002 0 1
R2 N002 N001 1
R5 N003 0 1
R4 N003 N002 1
R7 N004 0 1
R6 N004 N003 1
.dc I1 0 1 1e-4
```

```
.backanno
.end
```

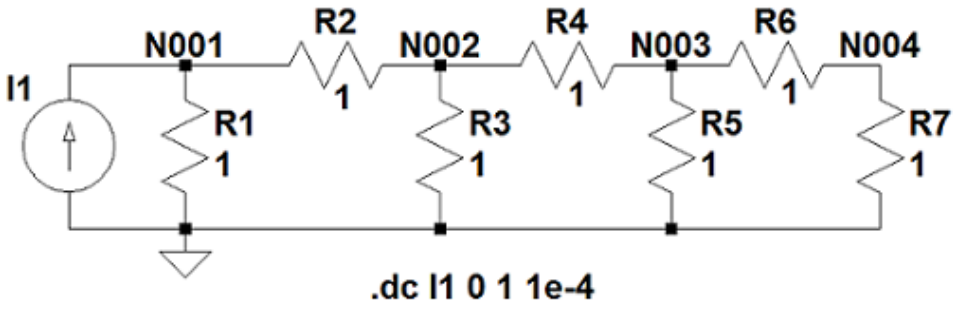


Figura 3.8: Rete di riferimento

Per la generazione della netlist per un numero arbitrario di nodi utilizziamo la seguente funzione Matlab:

```
function netliststring = netlist_Resistori(numeronodi)
netliststring = ['*numeronodi = ' num2str(numeronodi) '\n'];
%Il primo nodo sara' sempre lo stesso
netliststring = [netliststring 'I1 N1 0 1 \n'];
netliststring = [netliststring 'R1 N1 0 1 \n'];
%per i restanti nodi
numeroresistenza=2;
for k=2:numeronodi
    netliststring = [netliststring 'R' num2str(numeroresistenza)
        ...
        ' N' num2str(k) ...
        ' 0 1 \n'];
    netliststring = [netliststring 'R' num2str(numeroresistenza+1)
         ...
        ' N' num2str(k) ' N' num2str(k-1) ' 1 \n'];
    numeroresistenza=numeroresistenza+2;
end
%Comandi di simulazione LTSPICE
```

```
netliststring = [netliststring ...
                                    '.dc I1 0 1 1e-4 \n' ...
                                    '.backanno \n' ...
                                    '.end'];
end
```

Per lanciare la simulazione in Spice ,utilizziamo uno script Matlab esterno che:

- crea la netlist della rete considerata;
- esegue la simulazione in LTSPICE;
- legge il tempo di simulazione dal file .out;

Facciamo variare il numero di nodi da 5000 a 25000 in 6 step. I valori sono stati scelti per ottenere dei tempi significativi, dato che per un numero di nodi troppo piccolo avremmo dei tempi molto bassi, e inoltre si possono avere fluttuazioni dipendenti dal calcolatore; mentre per un numero di nodi eccessivamente alto si hanno dei tempi di simulazioni troppo elevati.

```
numeronodi=linspace(5000,25000,6);
%definiamo un vettore in cui salvare i tempi di simulazione
    trovati
tempo_Resistori = zeros(1,length(numeronodi));
indrisultato=1;
for numeronodi
    %creiamo la netlist usando la funzione di appoggio
    netlist = netlist_Resistori(numeronodi);
    %salva la netlist in un file .cir che eseguiamo con LTSPICE
    nomefile = ['Resistori_cappi_' num2str(numeronodi) '.cir'];
    %file da leggere per salvare i risultati
    nomefilelog = ['Resistori_cappi_' num2str(numeronodi) '.log'];
    %apri l'id del file e scrivi sul file
    fid = fopen(nomefile,'wt');
```

```
    fprintf(fid, netlist);
    fclose(fid);
    comando = ['scad3.exe -b ' nomefile];
    dos(comando);
    %apri il file di uscita in sola lettura
    fidlog=fopen(nomefilelog,'r');
    %leggilo per righe
    txt=textscan(fidlog,'%s','delimiter','\n');
    txt=txt{1};
    %prendi la riga che interessa per il tempo
    stringa_tempo = txt{5};
    %Leggi il valore e taglia gli ultimi
    stringa_tempo = stringa_tempo(1:end-8);
    %taglia i primi
    stringa_tempo = stringa_tempo(20:end);
    tempo_Resistori(indrisultato) = str2double(stringa_tempo);
    indrisultato = indrisultato+1;
end
```

Consideriamo adesso la netlist della rete con i GCCT e 4 nodi (figura 3.4):

```
I1 N001 0 1
R1 N001 0 1
G1 0 N002 N001 0 1
R2 N002 0 1
G2 0 N003 N002 0 1
R3 N003 0 1
G3 0 N004 N003 0 1
R4 N004 0 1
.dc I1 0 1 1e-4
.backanno
.end
```

Per la generazione della netlist per un numero arbitrario di nodi utilizziamo la seguente funzione Matlab:

```
function netliststring = netlist_GCCT(numeronodi)
netliststring = ['*numeronodi = ' num2str(numeronodi) '\n'];
%Il primo nodo sara' sempre lo stesso
netliststring = [netliststring 'I1 N1 0 1 \n'];
netliststring = [netliststring 'R1 N1 0 1 \n'];
%per gli altri nodi
for k=1:numeronodi-1
    netliststring = [netliststring 'G' num2str(k) ...
        ' 0 N' num2str(k+1) ...
        ' N' num2str(k) ' 0 1 \n'];
    netliststring = [netliststring 'R' num2str(k+1) ...
        ' N' num2str(k+1) ' 0 1 \n'];
end
%Comandi di simulazione LTSPICE
netliststring = [netliststring ...
                                '.dc I1 0 1 1e-4 \n' ...
                                '.backanno \n' ...
                                '.end'];
end
```

Lo script per lanciare la simulazione in Spice sarà lo stesso per la rete di riferimento facendo le opportune modifiche: il numero di nodi rimane lo stesso, la funzione da richiamare è la netlist_GCCT e il vettore in cui salvare i risultati è tempo_GCCT (lo stesso vale per i restanti casi).
Per la rete con i GTCT, la netlist è simile a quella con i GCCT sostituendo opportunamente i nomi dei componenti. La funzione Matlab per ottenere la netlist per un numero arbitrario di nodi è la seguente:

```
function netliststring = netlist_GTCT(numeronodi)
netliststring = ['*numeronodi = ' num2str(numeronodi) '\n'];
%Primo nodo
```

```
netliststring = [netliststring 'I1 N1 0 1 \n'];
netliststring = [netliststring 'R1 N1 0 1 \n'];
%Altri nodi
for k=1:numeronodi-1
    netliststring = [netliststring 'E' num2str(k) ...
        ' N' num2str(k+1) ...
        ' 0 N' num2str(k) ' 0 1 \n'];
    netliststring = [netliststring 'R' num2str(k+1) ...
        ' N' num2str(k+1) ' 0 1 \n'];
end
%Comandi di simulazione LTSPICE
netliststring = [netliststring ...
                                '.dc I1 0 1 1e-4 \n' ...
                                '.backanno \n' ...
                                '.end'];
end
```

In generale per i generatori controllati in corrente, per effetto della loro implementazione in Spice, bisogna aggiungere in serie alla corrente da monitorare dei generatori di tensione nulla *di sense*. Quindi per i GTCC è necessario includere questo ulteriore componente alla rete di figura 3.6, come in figura 3.9.

Il nodo $NR1$ viene aggiunto dal simulatore a causa dell'introduzione del generatore di tensione nulla. Quanto fatto non ha alcuna influenza dal punto di vista teorico per le equazioni trovate nel paragrafo 3.1, ma ha un effetto nell'ordine del sistema in Spice poichè la corrente nei generatori *di sense* aggiunge un'incognita. La funzione Matlab che genera la netlist della rete di figura 3.9 per un numero di nodi n arbitrario è la seguente:

```
function netliststring = netlist_GTCC(numeronodi)
netliststring = ['*numeronodi = ' num2str(numeronodi) '\n'];
%primo nodo
```

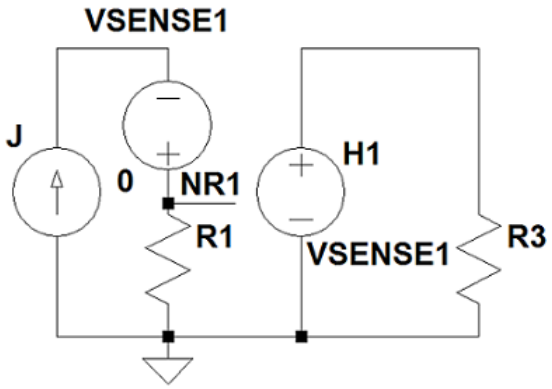


Figura 3.9: Rete con GTCC e l'aggiunta dei generatori di tensione nulla *di sense*

```
netliststring = [netliststring 'I1 N1 0 1 \n'];
netliststring = [netliststring 'VSENSE1 NR1 N1 0 \n'];
netliststring = [netliststring 'R1 NR1 0 1 \n'];
%altri nodi
numeroresistenza=2;numerogeneratore=1;
for k=2:numeronodi
    netliststring = [netliststring 'H' num2str(numerogeneratore)
        ...
        ' N' num2str(k) ...
        ' 0 VSENSE' num2str(numerogeneratore) ' 1 \n'];
    numerogeneratore=numerogeneratore+1;

    netliststring = [netliststring 'VSENSE' num2str(
        numerogeneratore) ...
        ' NR' num2str(k) ' N' num2str(k) ' 0 \n'];

    netliststring = [netliststring 'R' num2str(numeroresistenza)
        ...
        ' NR' num2str(k) ' 0 1 \n'];
    numeroresistenza=numeroresistenza+1;
end
```

```
%comandi di simulazione LTSPICE
netliststring = [netliststring ...
                                    '.dc I1 0 1 1e-4 \n' ...
                                    '.backanno \n' ...
                                    '.end'];

end
```

Considerazioni analoghe vanno fatte per i GCCC, per tanto la funzione Matlab che genera la netlist della rete per un numero di nodi n arbitrario è molto simile a quanto visto prima:

```
function netliststring = netlist_GCCC(numeronodi)
netliststring = ['*numeronodi = ' num2str(numeronodi) '\n'];
%primo nodo
netliststring = [netliststring 'I1 N1 0 1 \n'];
netliststring = [netliststring 'VSENSE1 NR1 N1 0 \n'];
netliststring = [netliststring 'R1 NR1 0 1 \n'];
%altri nodi
numeroresistenza=2;numerogeneratore=1;
for k=2:numeronodi
    netliststring = [netliststring 'F' num2str(numerogeneratore)
        ...
        ' N' num2str(k) ...
        ' 0 VSENSE' num2str(numerogeneratore) ' 1 \n'];
    numerogeneratore=numerogeneratore+1;

    netliststring = [netliststring 'VSENSE' num2str(
        numerogeneratore) ...
        ' NR' num2str(k) ' N' num2str(k) ' 0 \n'];

    netliststring = [netliststring 'R' num2str(numeroresistenza)
        ...
        ' NR' num2str(k) ' 0 1 \n'];
```

```
    numeroresistenza=numeroresistenza+1;
end
%comandi di simulazione LTSPICE
netliststring = [netliststring ...
                                    '.dc I1 0 1 1e-4 \n' ...
                                    '.backanno \n' ...
                                    '.end'];
end
```

Rete	Incognite teoria	Sparsità	Incognite Spice
Resistori	n	$1-\frac{3n-2}{n^2}$	n
GCCT	n	$1-\frac{2n-1}{n^2}$	n
GTCT	2n-1	$1-\frac{4n-3}{(2n-1)^2}$	2n-1
GCCC	2n-1	$1-\frac{4n-3}{(2n-1)^2}$	3n
GTCC	2n-1	$1-\frac{4n-3}{(2n-1)^2}$	4n-1

Tabella 3.1: Legge di variazione del numero di incognite in Spice

Anche in questo caso l'inserimento dei generatori di tensione nulla *di sense* comporta un aumento dell'ordine del sistema. Per una maggiore chiarezza sugli effetti che hanno l'inserimento di questi generatori di tensione sull'ordine del sistema, riportiamo nella tabella 3.1 le leggi con cui variano il numero di incognite in Spice.
Nel grafico di figura 3.10 sono visualizzati gli andamenti dell'ordine del sistema delle diverse reti al cresce del numero di nodi secondo le leggi di variazione del numero di incognite riportate nella tabella mostrata precedentemente.

L'andamento della rete con i resistori lineari e quello con i GCCT sono sovrapposti poichè per entrambe l'ordine del sistema coincide con il numero di nodi; per i generatori controllati in corrente (GTCC,GCCC) e il GTCT l'ordine aumenta rispetto al caso di riferimento per le ulteriori incognite aggiunte al sistema a causa delle correnti che scorrono nei generatori di tensione (per i primi due in quelli *di sense*, per l'altro nel generatore controllato).

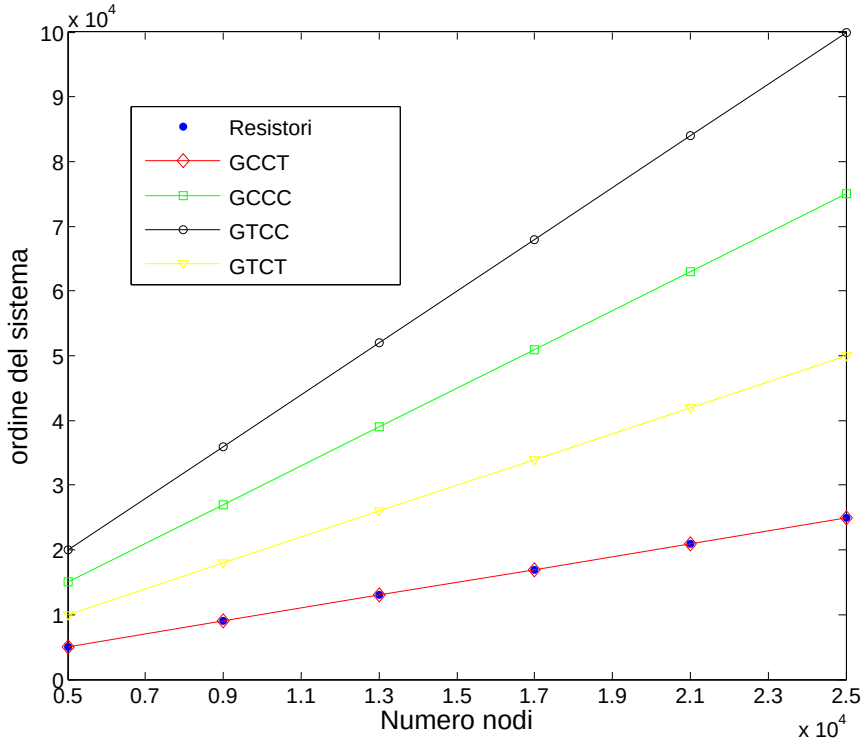


Figura 3.10: Ordine del sistema in funzione del numero di nodi

L'aumento di ordine del sistema ha un impatto significativo sul tempo di simulazione. Per il caso statico, si ottengono i tempi di simulazioni di figura 3.11.

Il grafico mostra che i GCCT hanno un comportamento simile a quello dei resistori lineari, infatti le due linee sono quasi sovrapposte. Per gli altri casi i tempi di simulazione risultano più alti, principalmente a causa del numero di incognite e nodi aggiuntivi.

3.3 Simulazione dinamica

Il prossimo obiettivo è quello di verificare che il comportamento dei generatori nel caso statico sia lo stesso di quello dinamico. Per le simulazioni

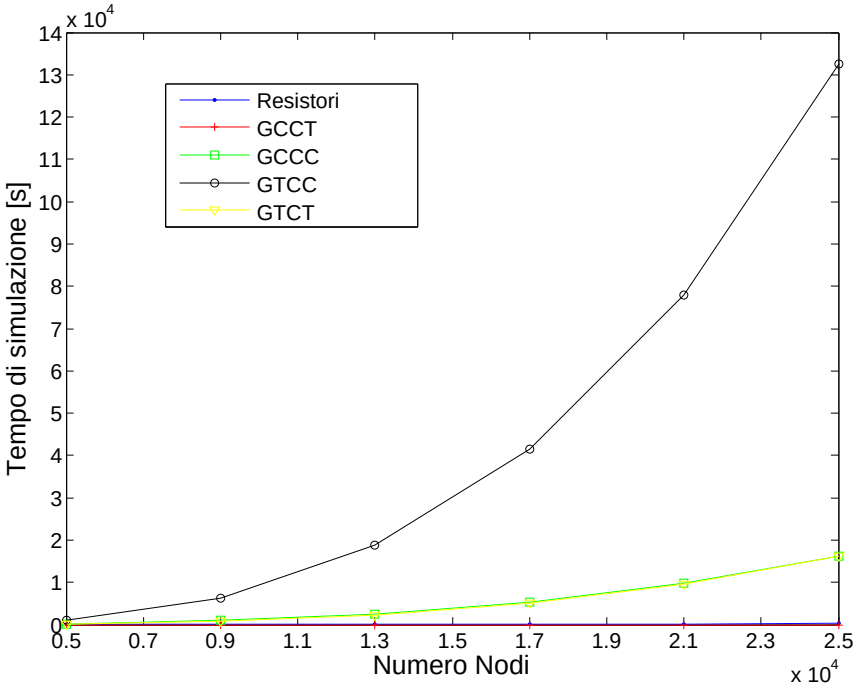
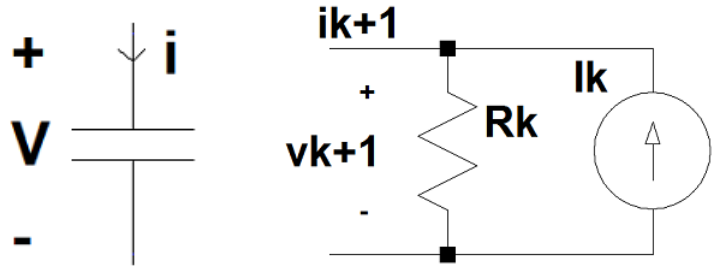


Figura 3.11: Tempo di simulazione in funzione del numero di nodi per il caso statico

dinamiche aggiungiamo alle reti di prova un condensatore: per quelle con i generatori di corrente lo inseriamo in parallelo all'ultima resistenza, in serie invece per quelle con i generatori di tensione. Prima di vedere come vanno modificate le funzioni Matlab che generano le netlist per ciascuna rete, dobbiamo capire come vengono risolti i circuiti dinamici in Spice. Quest'ultimo tratta la rete come una successione di circuiti statici in cui al posto del condensatore sostituisce il *companion circuit*.

Per renderci meglio conto di come Spice realizzi l'equivalente, consideriamo l'equazione costitutiva (BCE) di un condensatore (figura 3.12(a))

$$i = C\frac{\mathrm{d}v}{\mathrm{d}t} \tag{3.25}$$



(a) Simbolo elettrico del condensatore

(b) Companion circuit

Figura 3.12: Il condensatore e il suo equivalente nella dinamica in Spice

dove C è la capacità, misurata in farad, che rappresenta la costante di propozionalità tra la corrente che attraversa il condensatore e la derivata della tensione ai suoi capi. Per modellare il funzionamento del condensatore, Spice usa la forma integrale della equazione (3.25)

$$v = \frac{1}{C}\int_o^t idt + v_{co} \tag{3.26}$$

dove v_{co} è la tensione iniziale , al tempo t=0, sul condensatore.
Consideriamo il valore della tensione v ad un istante t_k ed ad un istante t_{k+1}

$$\dot{v} = \frac{1}{C}i_k; \quad \dot{v}_{k+1} = \frac{1}{C}i_{k+1} \tag{3.27}$$

Applicando l'algoritmo dei trapezi [8] [17], si ottiene

$$\begin{aligned} v_{k+1} &= v_k + \frac{\Delta_{tk}}{2C}(i_k + i_{k+1}) \Rightarrow v_{k+1} = v_k + R_k(i_k + i_{k+1}) \\ i_{k+1} &= \frac{2C}{\Delta_{tk}}v_{k+1} - (\frac{2C}{\Delta_{tk}}v_k + i_k) \Rightarrow i_{k+1} = \frac{1}{R_k}v_{k+1} - I_k \end{aligned} \tag{3.28}$$

dove Δ_{tk} è la differenza tra gli istanti di tempo considerati. Il *companion circuit* del condensatore è quello di figura 3.12(b) e consiste di una

resistenza $R_k = \frac{\Delta_{tk}}{2C}$ in parallelo ad un generatore equivalente di corrente $I_k = \frac{2C}{\Delta_{tk}} v_k + i_k$.

L'equazione (3.28) viene valutata in Spice ad ogni istante di tempo per tenere in conto dei contributi del condensatore.

Come già anticipato, l'introduzione del condensatore comporta delle modifiche alle funzioni Matlab che generano le netlist delle diverse reti. Cominciamo con la rete di riferimento; il primo cambiamento consiste nel fatto che il generatore di corrente è un gradino unitario, quindi per il primo nodo le due righe di codice diventano:

```
netliststring = [netliststring 'I1 N1 0 PULSE(0 1 0 1n 1n 10 10 1)
    \n'];
netliststring = [netliststring 'R1 N1 0 1 \n'];
```

Questo cambiamento va effettuato anche per le funzioni delle altre reti, dato che il primo nodo è uguale per tutte. Le righe di codice che generano i successivi cappi rimangono le stesse, va aggiunto alla fine del ciclo for un'ulteriore riga per includere il condensatore in parallelo all'ultima resistenza:

```
netliststring = [netliststring 'C1 N' num2str(numeronodi) ' 0 1 \n
    '];
```

Affinchè il confronto sia effettuato a parità di istanti temporali, forziamo Spice ad eseguire lo stesso numero di passi per tutte le reti. Per fare questo valutiamo la costante di tempo $\tau = R_{eq}C$ per il circuito, consideriamo per la simulazione un tempo T pari a 10 costanti di tempo ed usiamo come stepsize un valore abbastanza piccolo, $\frac{1}{100}T$:

```
Rk=1;
niter=numeronodi;
%calcolo della Req con tutte le resistenze scelte di valore
    unitario
for k=2:niter
```

```
    Rk=(Rk+1)/(Rk+2);
%C=1
tau=Rk*1;
T=10*tau;
stepsize=T/10e3;
%comandi di simulazione LTSPICE
netliststring = [netliststring ...
        .tran 0 ' num2str(T) ' 0 ' num2str(stepsize) ' \n' ...
        '.backanno \n' ...
        '.end'];
end
```

Anche per la rete con i GCCT la parte sulla generazione dei cappi non varia e bisogna inserire il condensatore in parallelo all'ultima resistenza, come fatto per la rete di riferimento. La R_{eq} risulta sempre pari ad 1 per le scelte fatte sui valori dei resistori, pertanto τ è sempre uguale ad 1, quindi $T = 10$ e lo $stepsize = T/10e3$ (e lo stesso vale per gli altri tre casi).
Per il caso dei GCCC le modifiche da fare sono esattamente le stesse di quelle viste per i GCCT.
Per i GTCT e GTCC il condensatore va inserito in serie all'ultima resistenza, quindi nel ciclo for per la generazione dei cappi non includiamo l'ultimo, il quale va scritto al di fuori del ciclo includendo il condensatore. Quindi per i GTCT abbiamo:

```
for k=1:numeronodi-2
...
...
end
 netliststring = [netliststring 'E' num2str(numeronodi-1) ...
        ' N' num2str(numeronodi) ' 0 N' num2str(numeronodi-1) ...
        ' 0 1 \n'];
    netliststring = [netliststring 'R' num2str(numeronodi) ...
```

```
        ' N' num2str(numeronodi) ' N' num2str(numeronodi+1) ' 1 \n
            '];
    netliststring = [netliststring 'C1 N' num2str(numeronodi+1) '
        0 1 \n'];
```

per i GTCC:

```
for k=2:numerocappi-1
...
...
end
netliststring = [netliststring 'H' num2str(numeronodi-1) ...
        ' N' num2str(numeronodi) ' 0 VSENSE' num2str(numeronodi-1)
            ' 1 \n'];
    netliststring = [netliststring 'VSENSE' num2str(numeronodi)
        ...
        ' NR' num2str(numeronodi) ' N' num2str(numeronodi) ' 0 \n'
            ];
    netliststring = [netliststring 'R' num2str(numeronodi) ...
        ' NR' num2str(numeronodi) ' N' num2str(numeronodi+1) ' 1 \
            n'];
    netliststring = [netliststring 'C1 N' num2str(numeronodi+1) '
        0 1 \n'];
```

I risultati delle simulazioni dinamiche, mostrate nel grafico di figura 3.13, sono molto simili a quelli delle simulazioni statiche, e il comportamento del GCCT si riconferma essere il più simile a quello dei resistori lineari.

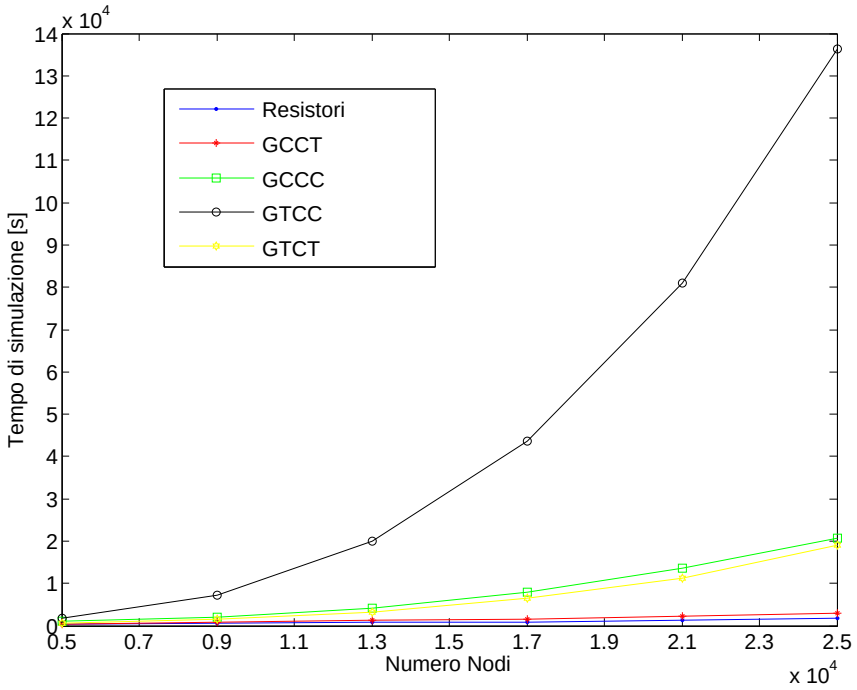


Figura 3.13: Tempo di simulazione in funzione del numero di nodi per il caso dinamico

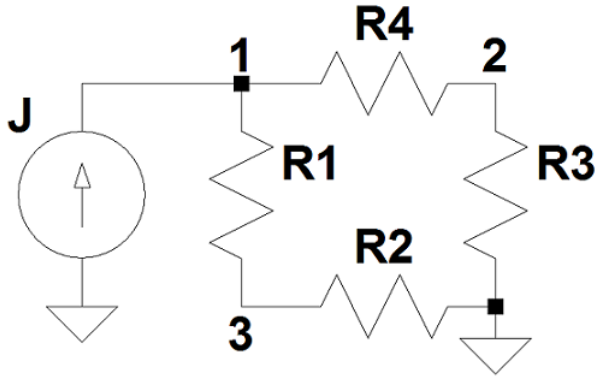
3.4 Prove bidimensionali

Consideriamo uno schema 2D delle reti di prova. Quest'ultime vengono quindi fatte estendere allo stesso modo sia orizzontalmente che verticalmente, e il numero di nodi n è funzione del numero di quadrati Q per riga, ovvero $n = (Q + 1)^2$.

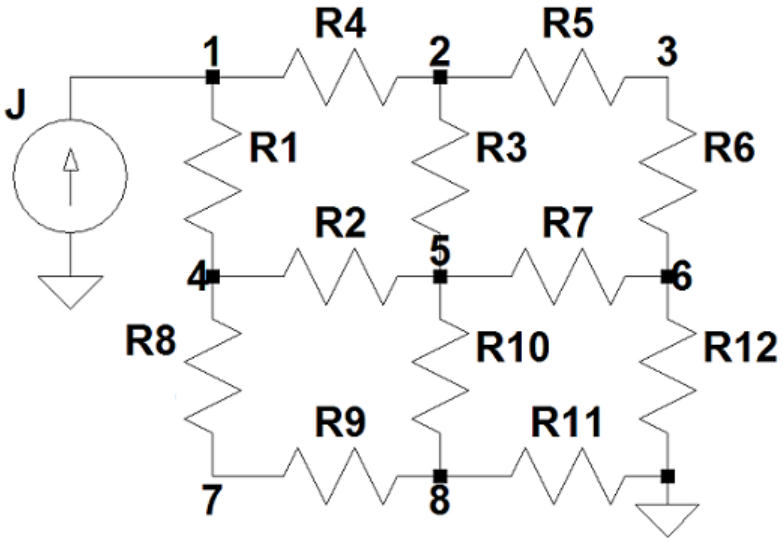
Ad esempio, per la rete di riferimento per $Q = 1 \rightarrow n = 4$, per $Q = 2 \rightarrow n = 9$, eccetera (figura 3.14).

Analizziamo la rete in figura 3.14(a), per cui

$$\underline{\underline{G}} \cdot \underline{u} = \underline{J} \tag{3.29}$$



(a) $Q = 1, n = 4$



(b) $Q = 2, n = 9$

Figura 3.14: Schema 2D della rete di riferimento

dove $\underline{\underline{G}}$ è la matrice delle conduttanze, i cui elementi, per ispezione diretta, sono

- $G_{ii} = \sum$ delle conduttanze al nodo i-esimo;
- $G_{ij,i\neq j} = -\sum$ delle conduttanze tra il nodo i-esimo e il j-esimo;

$\underline{u}$ è il vettore dei potenziali di nodo; $\underline{J}$ è il vettore somma delle correnti entranti al nodo i-esimo. Quindi, per Q=1 e considerando tutte le resistenze

uguali, la (3.29) diventa

$$\begin{pmatrix} \frac{1}{R}+\frac{1}{R} & -\frac{1}{R} & -\frac{1}{R} \\ -\frac{1}{R} & \frac{1}{R}+\frac{1}{R} & -\frac{1}{R}-\frac{1}{R} \\ -\frac{1}{R} & -\frac{1}{R}-\frac{1}{R} & \frac{1}{R}+\frac{1}{R} \end{pmatrix} \cdot \underline{u} = \begin{pmatrix} J \\ 0 \\ 0 \end{pmatrix} \tag{3.30}$$

La matrice $\underline{\underline{G}}$ è anche ricavabile automaticamente dalla matrice di incidenza ridotta A, secondo l'equazione

$$\underline{\underline{G}} = Adiag(y)A^T \tag{3.31}$$

dove

$$y_i = \begin{cases} \frac{1}{R_i} \textit{ se sul lato i c'e' una resistenza} \\ 0 \textit{ se sul lato i c'e' un generatore di corrente} \end{cases} \tag{3.32}$$

La matrice A si ricava dalla matrice di incidenza A_a eliminando da questa una riga (la matrice A ha sempre rango pieno per un grafo connesso, quindi le sue righe risultano sicuramente indipendenti). Per la rete in figura 3.14(a), che presenta quattro nodi e quattro lati, la matrice di incidenza corrispondente è una matrice $4x4$, i cui elementi a_{ij} sono

- 1 se il lato j esce dal nodi i;
- -1 se il lato j entra nel nodo i;
- 0 se il lato j non incide nel nodo;

Pertanto

$$A_a = \begin{pmatrix} -1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \tag{3.33}$$

Eliminiamo da questa l'ultima riga, corrispondente al nodo di massa, ottenendo così la matrice A, e applichiamo la (3.31) per ottenere la matrice $\underline{\underline{G}}$.

Al cresce di Q il sistema si complica ulteriormente, ed è possibile adottare una strategia per calcolare facilmente la matrice A, e quindi la matrice $\underline{\underline{G}}$. Prendiamo come esempio la rete di figura 3.14(b). La matrice di incidenza è costituita da 9 righe (numero nodi) ed 12 colonne (numero lati), come segue

$$A_a = \begin{pmatrix} -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & -1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix} \tag{3.34}$$

All'interno della stessa è possibile individuare $Q+1$ blocchi Q_a,che sono

matrici $Q+1$ x Q, e Q blocchi Q_b, che sono matrici $2(Q+1)$ x $(Q+1)$, ossia

$$Q_a = \begin{pmatrix} -1 & 0 & \cdots & \cdots \\ 1 & -1 & 0 & \cdots \\ 0 & 1 & \ddots & \ddots \\ \vdots & \ddots & \ddots & \ddots \end{pmatrix} \quad Q_b = \begin{pmatrix} -1 & 0 & \cdots & \cdots \\ 0 & -1 & 0 & \cdots \\ \vdots & 0 & \ddots & \ddots \\ 1 & \vdots & \ddots & \ddots \\ 0 & 1 & 0 & \ddots \\ \vdots & \ddots & \ddots & \ddots \end{pmatrix} \tag{3.35}$$

Dopodichè bisogna individuare la posizione del primo elemento delle matrici Q_a e Q_b in A_a, in modo da individuare l'inizio dei blocchi.
Per la matrice Q_a sono

- $1, 1$;
- $1 + Q + 1, 1 + (2Q + 1)$;
- $1 + 2(Q + 1), 1 + 2(2Q + 1)$;
- $1 + 3(Q + 1), 1 + 3(2Q + 1)$;
- ... ;

Per la matrice Q_b sono

- $1, Q + 1$;
- $1 + Q + 1, (2Q + Q + 1) + 1$;
- $1 + 2(Q + 1), [3Q + 2(Q + 1)] + 1$;
- ... ;

In questo modo è possibile ricavare direttamente la matrice di incidenza A_a e quindi, cancellando l'ultima riga, la matrice di incidenza ridotta A, ed infine applicare la (3.31). Questo metodo è stato implementato in Matlab ed riportato nell'appendice 5.2.

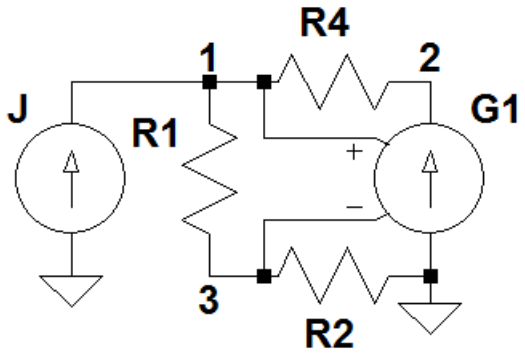
Per effettuare il confronto con la rete di riferimento, bisogna inserire in maniera opportuna negli schemi di figura 3.14 i generatori controllati. Ad esempio, per i generatori di corrente controllati in tensione, gli schemi sono quelli illustrati in figura 3.15.

Analizziamo la rete in figura 3.15(a). Le LKC ai nodi sono

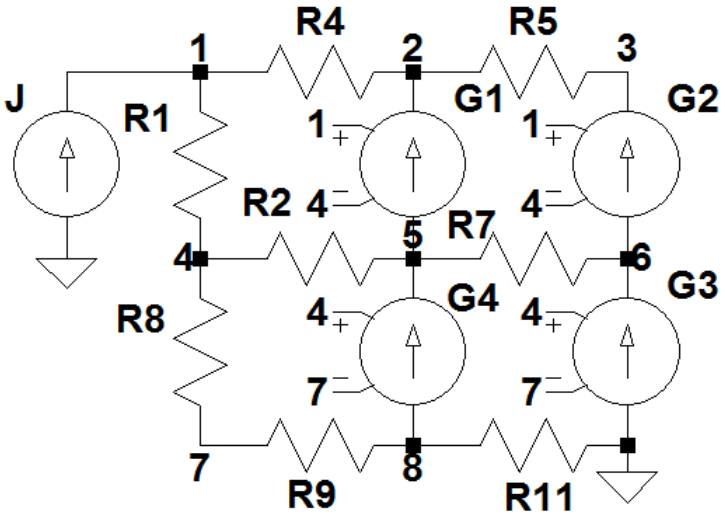
$$\begin{cases} J = i_1 + i_2 \\ i_1 = i_3 \\ i_2 = i_4 \end{cases} \tag{3.36}$$

quindi

$$\begin{cases} \frac{u_1 - u_2}{R} + \frac{u_1 - u_3}{R} = J \\ \frac{u_1 - u_2}{R} = (u_1 - u_3)\,\alpha \\ \frac{u_1 - u_3}{R} = \frac{u_3}{R} \end{cases} \tag{3.37}$$



(a) Rete con GCCT, $Q = 1$



(b) Rete con GCCT, $Q = 2$

Figura 3.15: Schema 2D della rete con GCCT

riordinando le equazioni, si ottiene

$$\begin{pmatrix} \frac{1}{R} + \frac{1}{R} & -\frac{1}{R} & -\frac{1}{R} \\ \alpha - \frac{1}{R} & \frac{1}{R} & 0 \\ -\frac{1}{R} & 0 & \frac{1}{R} + \frac{1}{R} \end{pmatrix} \cdot \underline{u} = \begin{pmatrix} J \\ 0 \\ 0 \end{pmatrix} \tag{3.38}$$

3.4.1 Simulazione statica

Di seguito vengono riportare le funzioni Matlab per la generazione delle netlist delle reti di prova.

Resistori

```
function netlist = netlistQuadratiR(Q)
%Nome della netlist
netlist = ['* Caso Bidimensionale - Resistenza Q=' num2str(Q) ' \n
    '];
%Alimentazione in corrente
netlist = [netlist 'I1 0 1 1 \n'];
%Costruiamo il vettore dei nodi
nodi = 1:((Q+1)^2);
nodi(end) = 0;
%Indice per il numero di resistenza
indiceR = 1;
%Righe: sono Q+1 righe contenenti Q resistenze
indiceriga=1;
for ind=1:(Q+1)
    for indrow = 1:Q
        netlist = [netlist 'R' num2str(indiceR) ' ' num2str(nodi(
            indiceriga+1)) ' ' num2str(nodi(indiceriga)) ' 1 \n'];
        indiceriga=indiceriga+1;
        indiceR=indiceR+1;
    end
    indiceriga=indiceriga+1;
end
%Colonne: sono Q+1 colonne contenenti Q resistenze
for ind=1:(Q+1)
    for indcol = 1:Q
        netlist = [netlist 'R' num2str(indiceR) ' ' num2str(nodi(
            ind+(Q+1)*indcol)) ' ' num2str(nodi(ind+(Q+1)*(indcol
```

```
            -1))) ' 1 \n'];
        indiceR=indiceR+1;
    end
end
%Comandi di simulazione
netlist = [netlist '.dc I1 0 1 1e-4 \n.backanno \n.end \n'];
end
```

GCCT

Per la rete con i GCCT, le colonne da 2 a Q+1 contengono i generatori, pertanto bisogna modificare solo le righe di codice che generano le colonne.

```
% La prima colonna e' identica al caso delle resistenze
ind=1;
for indcol = 1:Q
    netlist = [netlist 'R' num2str(indiceR) ' ' num2str(nodi(ind+(
        Q+1)*indcol)) ' ' num2str(nodi(ind+(Q+1)*(indcol-1))) ' 1
        \n'];
    indiceR=indiceR+1;
end
%Le colonne da 2 a Q+1 contengono i GCCT
indiceG=1;
indiceprimacol=1;
for ind=2:(Q+1);
    for indcol = 1:Q
        netlist = [netlist 'G' num2str(indiceG) ' ' num2str(nodi(
            ind+(Q+1)*indcol)) ' ' num2str(nodi(ind+(Q+1)*(indcol
            -1))) ...
            ' ' num2str(nodi(1+(Q+1)*indcol)) ' ' num2str(nodi(1+(
                Q+1)*(indcol-1))) ' 1 \n'];
        indiceG=indiceG+1;
    end
    indiceprimacol=1;
end
```

GTCT

Per la rete con i GTCT la funzione è la stessa di quella del GCCT, con l'unica differenza che va cambiando il nome dei componenti (ovvero dei generatori).

GCCC

Per la rete con i GCCC vanno inseriti, come nel caso monodimensionale, i generatori di *di sense* per la lettura delle correnti nei generatori. Quindi la parte relativa alla scrittura delle righe rimane invariata, mentre bisogna cambiare quella delle colonne.

```
% Colonne: sono Q+1 colonne contenenti Q resistenze e Q generatore
     di
% VSENSE
ind=1; numerogeneratore=1;
for indcol = 1:Q
    netlist = [netlist 'R' num2str(indiceR) ' ' num2str(nodi(ind+(
        Q+1)*indcol)) ' NR' num2str(indcol) ' 1 \n'];
    netlist = [netlist 'VSENSE' num2str(numerogeneratore) ...
        ' NR' num2str(indcol) ' ' num2str(nodi(ind+(Q+1)*(indcol
            -1))) ' 0 \n'];
    indiceR=indiceR+1;
    numerogeneratore=numerogeneratore+1;
end
%Le colonne da 2 a Q+1 contengono i GCCC
numerogeneratore=1;
indiceG=1;
indiceprimacol=1;
for ind=2:(Q+1);
    for indcol = 1:Q
```

```
        netlist = [netlist 'F' num2str(indiceG) ' ' num2str(nodi(
            ind+(Q+1)*indcol)) ' ' num2str(nodi(ind+(Q+1)*(indcol
            -1))) ...
            ' VSENSE' num2str(indcol) ' 1 \n'];
        indiceG=indiceG+1;
    end
    indiceprimacol=1;
end
```

GTCC

Per la rete con i GTCC la funzione è la stessa di quella del GCCC, con l'unica differenza che va cambiando il nome dei componenti (ovvero dei generatori).

I risultati della simulazione statica sono mostrati nel grafico di figura 3.16.
Le dipendenze del tempo di simulazione sono più complesse che nel caso 1D. Le discrepanze dei risultati nel caso 2D rispetto al caso monodimensionale sono da attribuirsi alla sparsità della matrice, ovvero alla forma particolare del sistema in esame. La rete di riferimento risulta essere più complicata rispetto a quelle di prova con i generatori controllati, in particolar modo rispetto a quella con i GCCT.

Nel grafico di figura 3.17 è visualizzato l'ordine del sistema in funzione del numero di nodi. La linea corrispondente alla rete con le sole resistenze è sovrapposta a quella relativa alla rete con i GCCT, a conferma del fatto che gli ordini non cambiano.

Nella tabella 3.2 è mostrata la legge di variazione dell'ordine del sistema delle reti rispetto al numero di quadrati per riga.

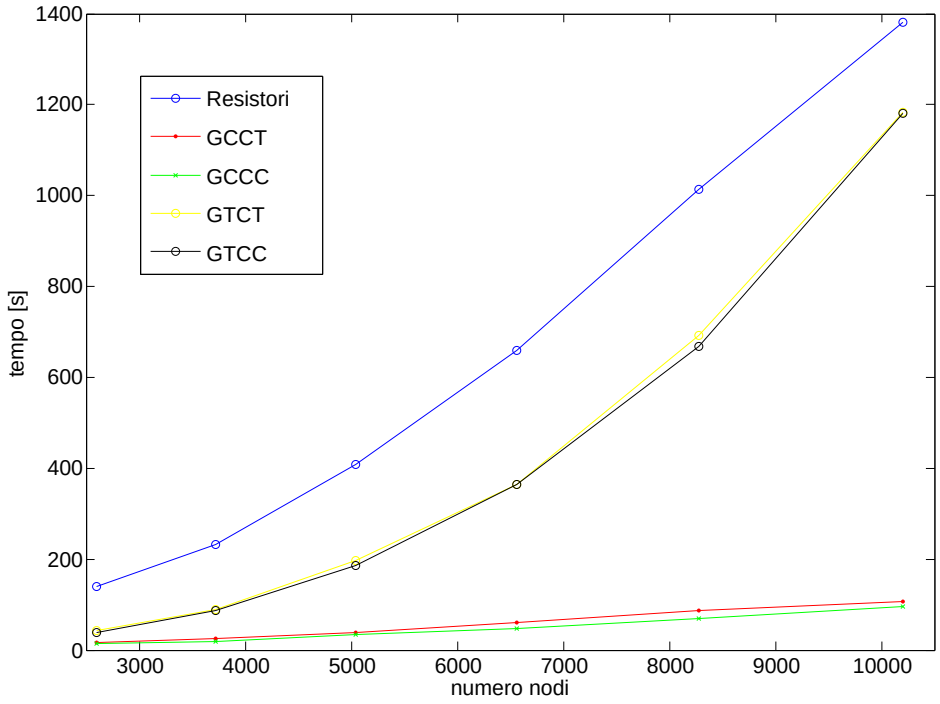


Figura 3.16: Tempo di simulazione in funzione del numero di nodi

Rete	Ordine Sistema
Resistori	$(Q+1)^2 - 1$
GCCT	$(Q+1)^2 - 1$
GTCT	$(Q+1)^2 + Q^2$ - 1
GCCC	$(Q+1)^2 + 2Q$ - 1
GTCC	$(Q+1)^2 + Q^2 + 2Q$ - 1

Tabella 3.2: Legge di variazione dell' ordine del sistema delle reti rispetto al numero di quadrati per riga

Essendo l'ordine del sistema per i GCCT identico a quello della rete di riferimento, la differenza nei tempi di simulazione è dovuta a qualche altro fattore, come, ad esempio, la sparsità della matrice e come essa è strutturata. È allora utile analizzare due casi in cui andiamo a complicare artificialmente la rete per i GCCT:

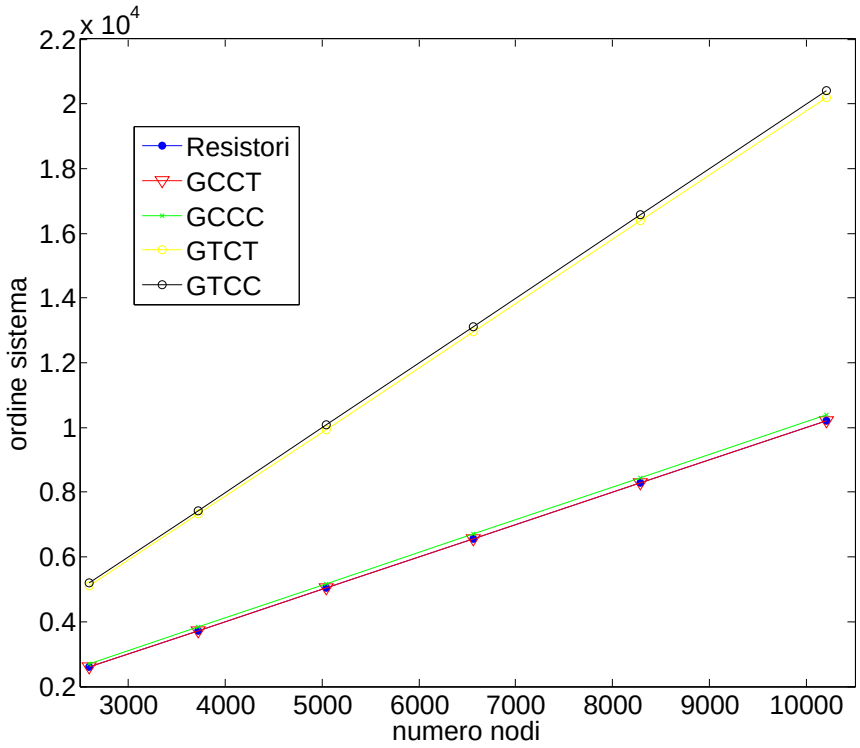


Figura 3.17: Ordine del sistema in funzione del numero di nodi

- cambiamo i nodi che pilotano i generatori controllati; questi non sono più gli stessi, ma quelli corrispondenti al generatore precedente, come in figura 3.18; (1)

- aggiungiamo delle resistenze in parallelo ai generatori di corrente, come in figura 3.19; (2)

I risultati delle due prove, mostrati in figura 3.20, evidenziano dei drastici aumenti dei tempi di simulazione a seguito delle modifiche effettuate alla rete (ovvero si è cambiato l'accoppiamento tra i vari termini del sistema), e quindi è più forte l'ipotesi che sono dovuti alla sparsità della matrice.

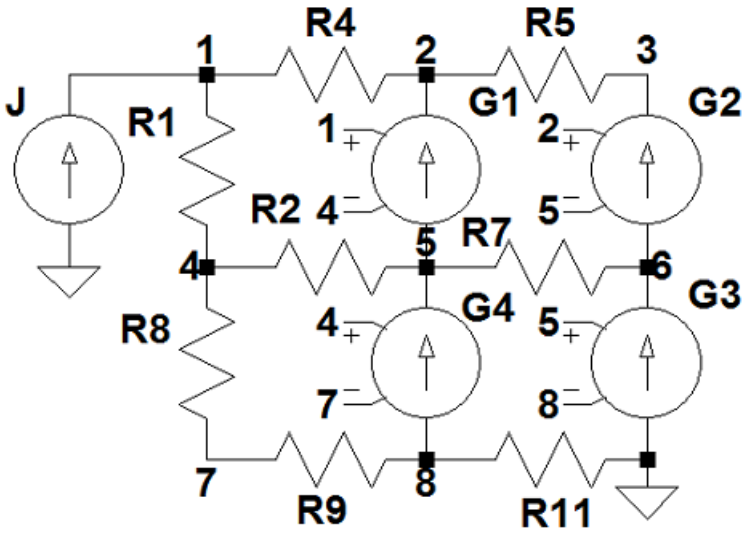


Figura 3.18: Prova (1) - cambio nodi di controllo

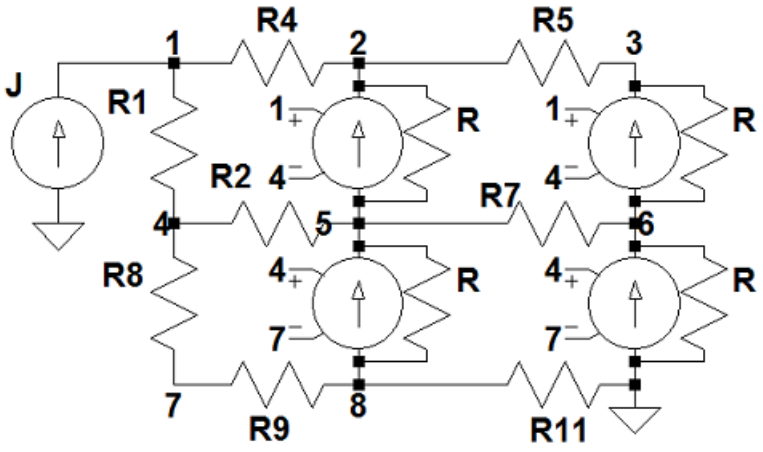


Figura 3.19: Prova (2) - aggiunta delle resistenze in parallelo ai generatori

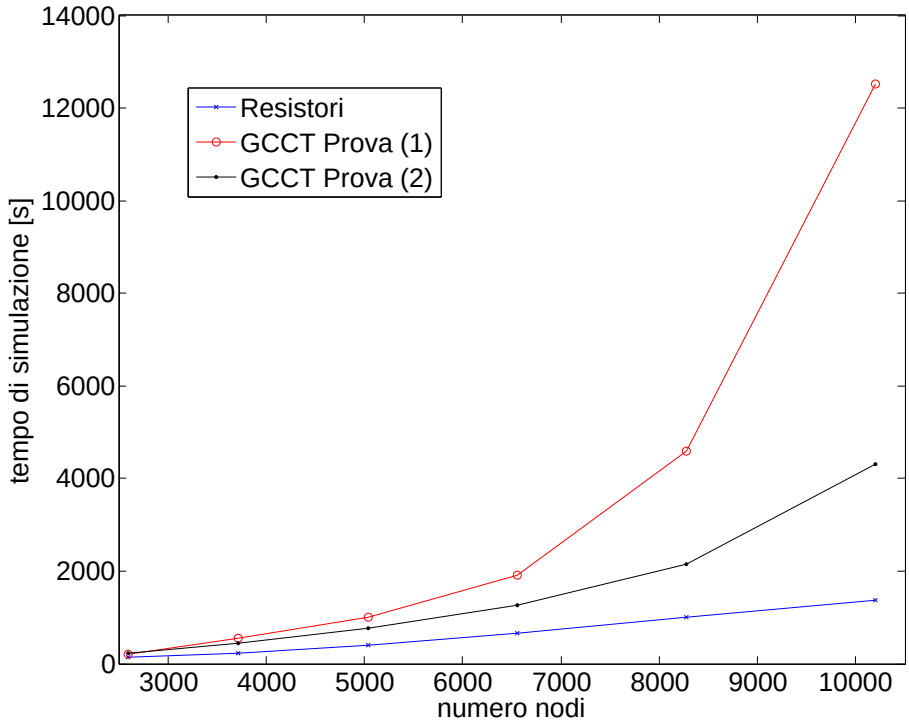


Figura 3.20: Tempi di simulazione per le Prove (1) e (2)

3.4.2 Simulazione dinamica

Per effettuare una simulazione dinamica bisogna aggiungere, in maniera opportuna, dei condensatori all'interno delle reti di prova. Li inseriamo in serie o in parallelo alle resistenza della prima colonna (per Q=2, alle resistenze R1 ed R8 di figura 3.15), a seconda se abbiamo dei generatori di tensione o di corrente rispettivamente. Affinchè il confronto sia effettuato a parità di istanti temporali, bisognerebbe conoscere le costanti di tempo per tutti i casi.

Consideriamo un caso più semplice: mettiamo un cappio RC unitario all'inizio, considerando separato il resto della rete, come in figura 3.21.

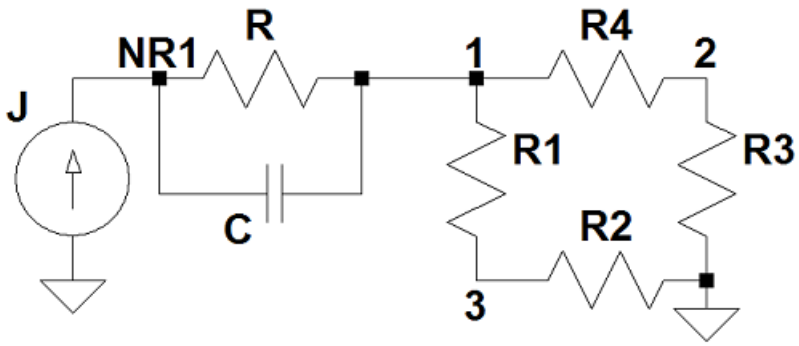


Figura 3.21: Aggiunta della dinamica per il caso bidimensionale

Di conseguenza, le funzioni Matlab per la generazione delle netlist vanno cambiate tutte allo stesso modo, ovvero bisogna

- inserire le righe di codice per il cappio RC e modificare il generatore di corrente che è un gradino unitario:

```
%Alimentazione in corrente e cappio RC
netlist = [netlist 'I1 NR1 0 PULSE(0 1 0 1n 1n 10 10 1) \n'];
netlist = [netlist 'R NR1 1 1 \n'];
netlist = [netlist 'C NR1 1 1 \n'];
```

- cambiare i comandi di simulazione:

```
%Comandi di simulazione
netlist = [netlist '.tran 0 10 0 1e-3 \n.backanno \n.end \n'
    ];
```

I risultati della simulazione dinamica sono mostrati in figura 3.22.

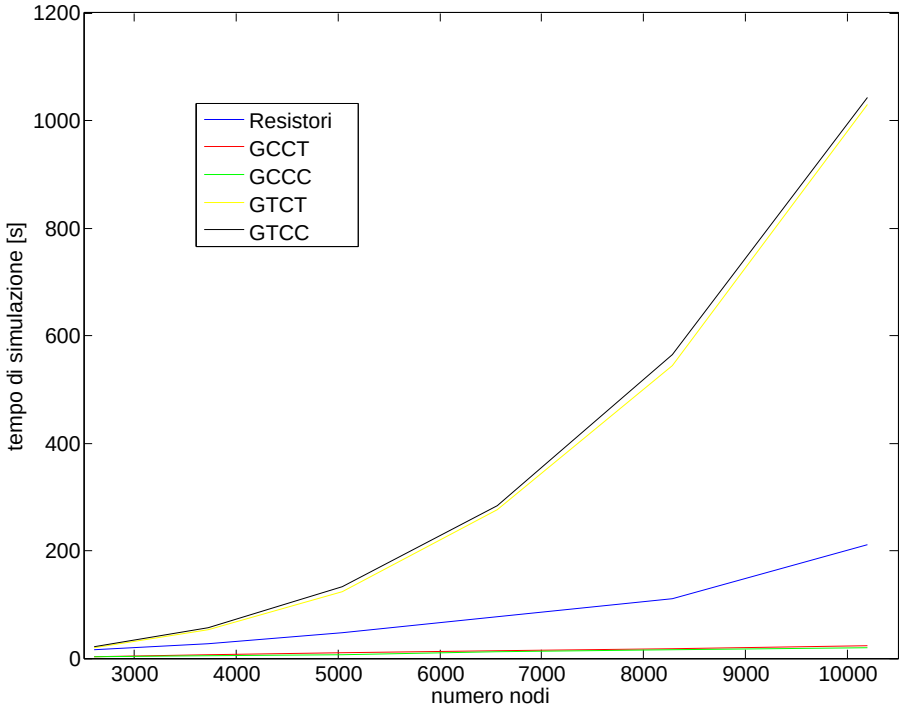


Figura 3.22: Tempo di simulazione in funzione del numero di nodi

Capitolo 4

Sintesi efficiente della rete di Foster multi-porta

Le prove effettuate nel capitolo 3 hanno dimostrato che usare un tipo di generatore o un altro influisce sulle prestazioni in Spice, e che il metodo di risoluzione per i GCCT risulta essere più semplice e più veloce rispetto agli altri tipi di generatori controllati. Come già visto, il circuito di sintesi di Foster generalizzato per un sistema multi-porta lineare, nel caso di soli poli reali, è rappresentabile come una serie di cappi RC ai quali sono collegati dei trasformatori ideali, che, nell'ambiente di simulazione Spice, sono rappresentabili come un GCCC alla porta 1 controllato dalla corrente della porta 2, ed ad un GTCT alla porta 2 controllato dalla tensione alla porta 1. Effettuiamo quindi delle sostituzioni per equivalenza con i GCCT, verificando se effettivamente la loro presenza comporta dei miglioramenti sui tempi di simulazione. Consideriamo come caso test la propagazione di un segnale lungo una linea, utilizzando la rete di Foster generalizzata per tenere in conto degli effetti elettrotermici (un esempio è stato riportato nel paragrafo 2.2). Il circuito di sintesi è realizzato con M=9 e $N_p = 7$ (sette poli risultano

sufficienti per ottenere risultati accurati). Consideriamo i seguenti casi di studio:

1. Prima di tutto valutiamo i tempi di simulazione per la rete di Foster generalizzata, il cui schema è mostrato in figura 2.13 per un numero M arbitrario di porte, e che riportiamo per comodità in figura 4.1;

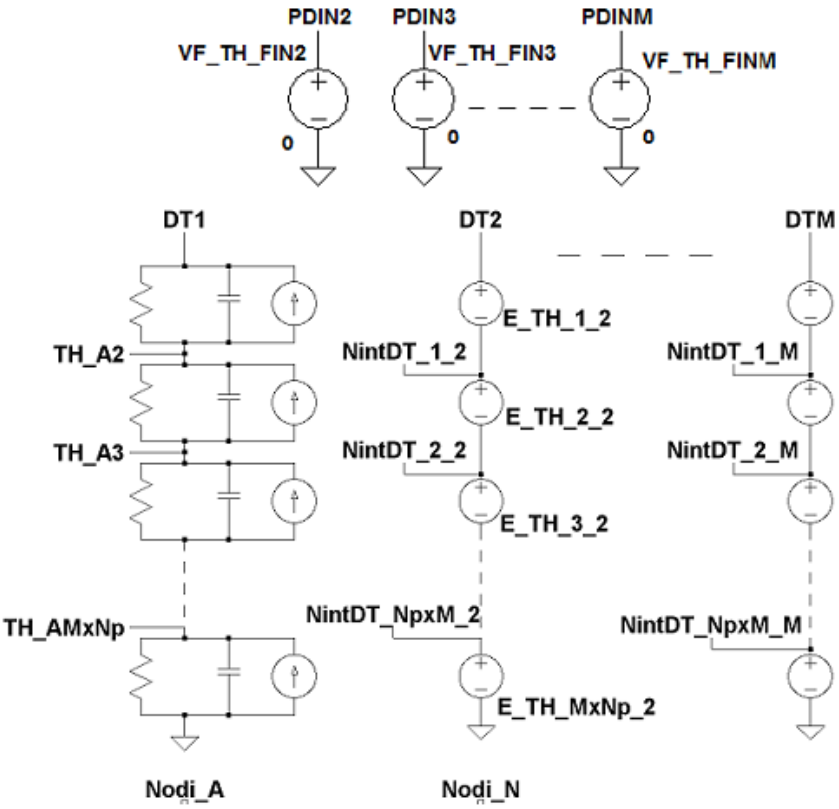


Figura 4.1: Rete di Foster generalizzata

2. Sostituiamo ai GCCC i GCCT. Per realizzare l'equivalenza bisogna sostituire ai generatori di tensione nulla, in cui scorrono le correnti in ingresso alle M porte, dei resistori di valore unitario, come mostrato in figura 4.2; Preso come esempio il primo cappio RC ed un singolo

trasformatore, la sostituzione del generatore avviene come in figura 4.3;

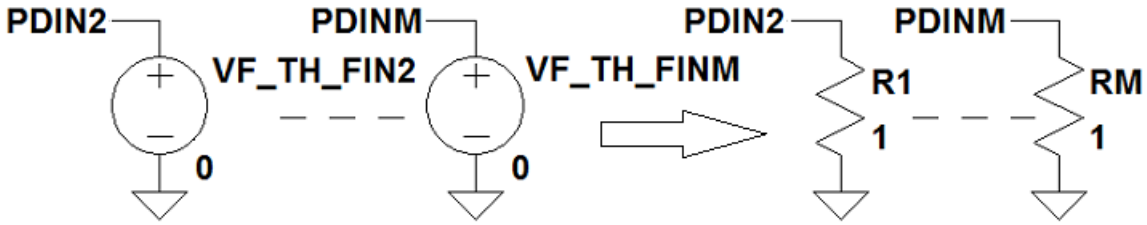


Figura 4.2: Resistenze unitarie in ingresso per la conversione in tensione

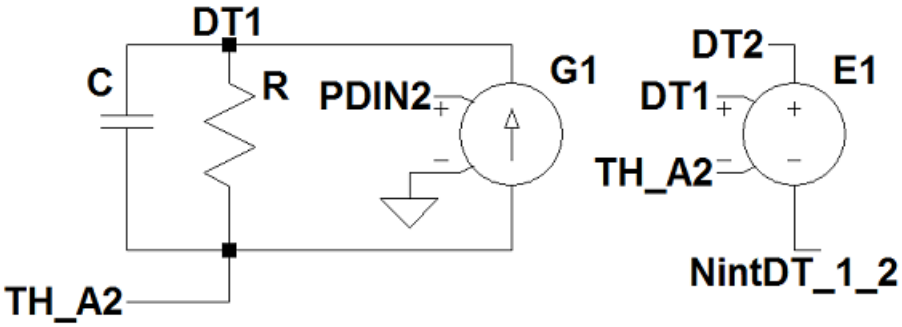


Figura 4.3: caso (2)

3. sostituiamo ai GTCT i GCCT. L'equivalenza è realizzata sostituendo alla serie degli MxN_p GTCT un parallelo di MxN_p GCCT che termina su un resistore di valore unitario, come in figura 4.4;

4. sostituiamo ad entrambe le porte i GCCT, realizzando entrambe le equivalenze appena descritte, come in figura 4.5;

5. Sostituiamo ai GCCC il componente Analog Behavioral Modeling (ABM), che ingloba tutti i generatori di corrente controllati in corrente in parallelo alla i-esima porta in un unico generatore, la cui corrente si esprime

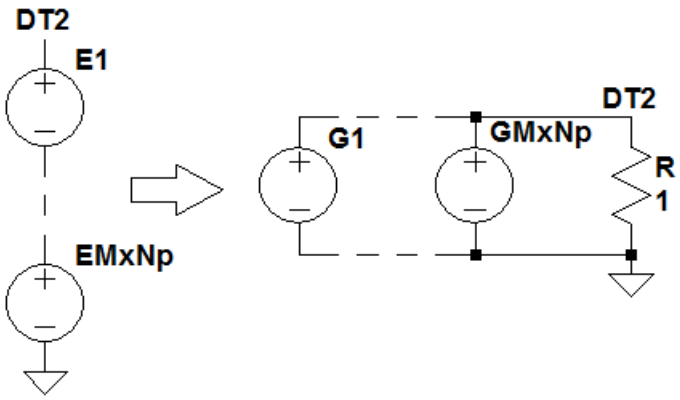


Figura 4.4: caso (3)

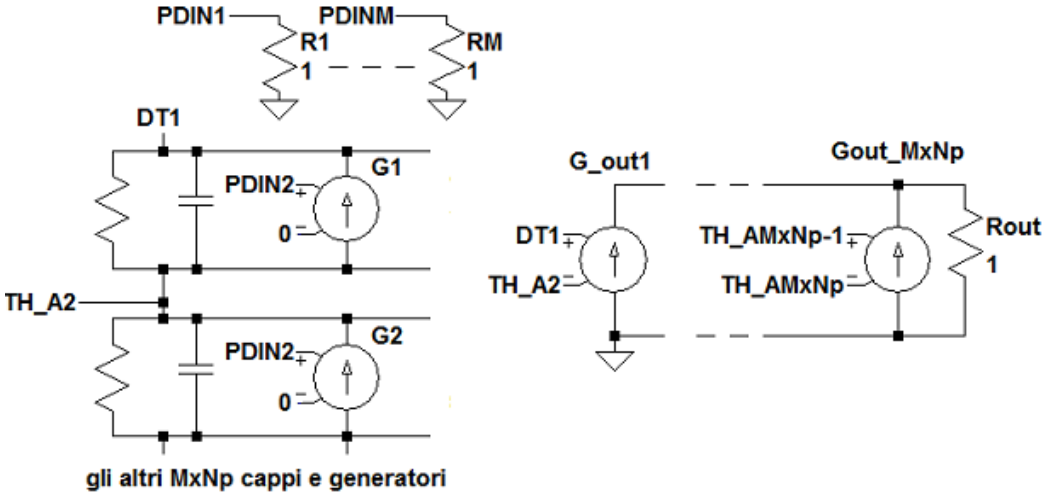


Figura 4.5: caso (4)

come la somma delle correnti di tutti i generatori (figura 4.6);

$$I = \sum_{n=1}^{M-1} \frac{PDINi}{k(1,i)} \tag{4.1}$$

Le netlist sono generate tramite il codice illustrato nel paragrafo 2.5, al quale vanno aggiunte le righe relative ai comandi per la simulazione. Consideriamo in ingresso alle M=9 porte dei generatori a gradino, in modo da ottenere dei tempi di simulazione significativi, e quindi delle differenze più

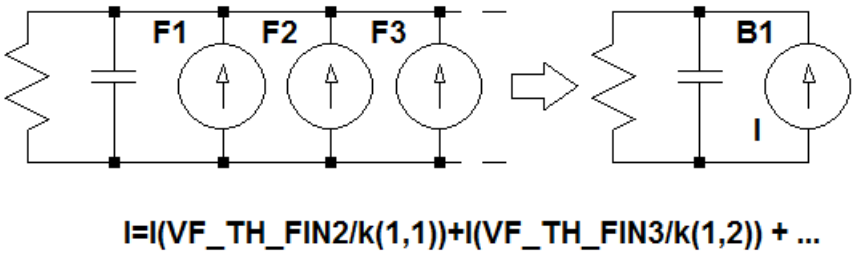


Figura 4.6: caso (5)

marcate tra le varie topologie.

```
%Per la prova specifica UTCS LINEA che prendiamo come test
netlist = [netlist 'I_I1 0 PDIN1 PULSE(0 10 0 1f 1f 1 2 1) \n' ...
   'I_I2 0 PDIN2 PULSE(0 10 0.5 1f 1f 1 2 1) \n' ...
   'I_I3 0 PDIN3 PULSE(0 10 1 1f 1f 1 2 1) \n' ...
   'I_I4 0 PDIN4 PULSE(0 10 1.5 1f 1f 1 2 1) \n' ...
   'I_I5 0 PDIN5 PULSE(0 10 2 1f 1f 1 2 1) \n' ...
   'I_I6 0 PDIN6 PULSE(0 10 2.5 1f 1f 1 2 1) \n' ...
   'I_I7 0 PDIN7 PULSE(0 10 3 1f 1f 1 2 1) \n' ...
   'I_I8 0 PDIN8 PULSE(0 10 3.5 1f 1f 1 2 1) \n' ...
   'I_I9 0 PDIN9 PULSE(0 10 4 1f 1f 1 2 1) \n'];
%Comandi di simulazione
netlist = [netlist '.tran 0 6 0 0.1e-3  \n'];
%Terminazione della netlist
netlist = [netlist '* TERMINAZIONE \n'];
netlist = [netlist ' .ENDS \n'];
```

Topologia	Tempo di simulazione [s]	Ordine sistema
(1)	64.333	1089
(2)	64.960	1080
(3)	46.879	89
(4)	48.202	80
(5)	120.398	1089

Tabella 4.1: Tempi di simulazione per la rete di sintesi

I risultati delle simulazioni sono mostrati nella tabella 4.1. Un effettivo miglioramento sui tempi di simulazione rispetto alla topologia generalizzata si ha quando sostituiamo i GCCT ai GTCT, ottenendo una diminuzione dei tempi di quasi il 30%. A questo corrisponde anche una diminuzione radicale dell'ordine del sistema, dato che inserendo i generatori di corrente controllati in tensione solo in uscita eliminiamo le incognite dovute alle correnti che scorrono nei GTCT e agli altrettanti nodi artificiali introdotti per il posizionamento degli stessi.

Utilizzando gli ABM si ha invece una degradazione delle prestazioni, nonostante la soluzione realizzi una versione della rete più compatta. Probabilmente il cambiamento è solamente esterno e nel metodo di risoluzione Spice implementa tutto come era prima, e non lo fa nemmeno nella maniera più ottimale dato che si è registrato un raddoppio dei tempi.

Capitolo 5

Appendice

5.1 Appendice A : Sintesi

```
%inizializzazione del workspace
clear
close all
clc
format long
warning off
%M è il numero di porte
M = size(PFVFId.residue,1);
%Np è il numero di poli
Np = length(PFVFId.poles);
%diagonalizzazione delle matrici dei residui e decomposizione in
    somma di matrici a rango 1
 %calcolo degli autovalori
%per ogni matrice dei residui...
for i=1:length(PFVFId.poles)
    [T,D]=eig(PFVFId.residue(:,:,i));
    for k=1:M
        temp = zeros(M,M);
        temp(k,k) = D(k,k);
```

```
        PFVFId.residuediag(:,:,i,k) = T*temp*T';
    end
end
%Verifica per la decomposizione
for i=1:Np
    C = zeros(M,M);
    for k=1:M
        C = C + squeeze(PFVFId.residuediag(:,:,i,k));
    end
    isequal(C,squeeze(PFVFId.residue(:,:,i)))
end
%calcolo dei valori di resistenza e capacità dei vari cappi
C=zeros(M*Np,1);
R=zeros(M*Np,1);
ind=1;
for i=1:length(PFVFId.poles)
    for k=1:M
        C(ind) = 1/PFVFId.residuediag(1,1,i,k);
        R(ind) = -PFVFId.residuediag(1,1,i,k)/PFVFId.poles(i);
        ind=ind+1;
    end
end
tau = R.*C;
%calcolo dei rapporti di trasformazione
kncir = zeros(M*Np,M-1);
indinterno = 1;
for i=1:length(PFVFId.poles)
    for k=1:M
        PFVFId.residuediag(:,:,i,k)= PFVFId.residuediag(:,:,i,k)/
            PFVFId.residuediag(1,1,i,k);
        kncir(indinterno,:) = 1./PFVFId.residuediag(1,2:end,i,k);
        indinterno=indinterno+1;
    end
```

```
end

%creazione della netlist del circuito di sintesi

%assegniamo un nome al componente e definiamo i pin di ingresso e
    uscita
%per una rete termica le potenze sono degli ingressi (in corrente)
     e le temperature sono
%delle uscite (in tensione)
M_str = num2str(M);
nomecomponente = 'Multiporta';
netlist = ['* THERMAL Network M = ' M_str ' \n'];
netlist = [netlist '.SUBCKT ' nomecomponente ' DT1 \n'];
for i=2:M
    netlist = [netlist  '+ DT' num2str(i) ' \n'];
end
for i=1:M
    netlist = [netlist '+ PDIN' num2str(i) ' \n'];
end
nodointerno='TH';
%processo di replicazione della corrente alla porta 1
%i nodi PDIN1 e DT1 di ingresso ed uscita alla porta 1 sono in
    realtà lo stesso nodo.
%per distinguere i due nodi replichiamo la corrente in ingresso
    verso i cappi alla porta 1
netlist = [netlist '* INGRESSO PORTA 1 - Replicazione di corrente
    \n' ];
netlist = [netlist 'F_' nodointerno '_F1 DT1 0 ' ...
    'VF_' nodointerno '_F1 -1 \n'];
netlist = [netlist 'VF_' nodointerno '_F1 ' ...
    ' PDIN1 0 0V \n'];
%costruamo i cappi RC alla porta 1
netlist = [netlist '* CAPPI ALLA PORTA 1 \n'];
```

```
%i nodi del lato sinistro dello schema di definiamo come segue
%nodi A - DT1 A2 A3 ... AM*Np 0
nodi_A = {'DT1'};
for i=2:(M*Np)
    nodi_A{i}= [nodointerno '_A' num2str(i)];
end
nodi_A{M*Np+1}=num2str(0);
for k=1:(M*Np)
    k_str = num2str(k);
    netlist = [netlist 'R_' nodointerno '_R' k_str ' ' nodi_A{k} '
        ' nodi_A{k+1} ' ' num2str(R(k),'%0.16e') ' \n'];
    netlist = [netlist 'C_' nodointerno '_C' k_str ' ' nodi_A{k} '
        ' nodi_A{k+1} ' ' num2str(C(k),'%0.16e') ' \n'];
end
%aggiungiamo i generatori di tensione nulli per leggere le
    correnti in
%ingresso alle porte 2 ... M
netlist = [netlist '* Aggiunte alla porta 1 dovute alle porte 2...
    M \n'];
for indm = 2:M
    indm_str = num2str(indm);
    netlist = [netlist '* -> dovuta alla PORTA ' indm_str ' \n'];
    netlist = [netlist 'VF_' nodointerno '_Fin_' indm_str ' ' ...
        'PDIN' num2str(indm) ' 0 0V \n'];
%realizziamo i trasformatori ideali
%per la Şporzione Ť di trasformatori alla porta 1 inseriamo in
    parallelo ai cappi RC i generatori di corrente controllati
    dalle correnti in ingresso
    for k=1:(M*Np)
        k_str = num2str(k);
        netlist = [netlist 'F_' nodointerno '_F' k_str '_'
            indm_str ' ' ...
            nodi_A{k} ' ' nodi_A{k+1} ' ' ...
```

```
            'VF_' nodointerno '_Fin_' indm_str ' ' ...
            num2str(-1/kncir(k,indm-1),'%0.16e') ' \n'];
    end
end
%definiamo i nodi interni tra cui piazzare i generatori di
    tensione
%controllati in tensione che costutuiscono la seconda "porzione"
    dei
%trasformatori..
netlist = [netlist '* Le altre 2...M porte \n'];
for indm = 2:M
    indm_str = num2str(indm);
    netlist = [netlist '* -> PORTA ' indm_str ' \n'];

    nodi_N = {['DT' indm_str]};
    for in=2:(M*Np)
        nodi_N{in}= ['NintDT_' num2str(in-1) '_' indm_str];
    end
    nodi_N{M*Np+1}=num2str(0);
%piazzamento dei generatori di tensione controllati in tensione
    for k=1:(M*Np)
        k_str = num2str(k);
        netlist = [netlist 'E_' nodointerno '_E' k_str '_'
            indm_str ' ' ...
            nodi_N{k} ' ' nodi_N{k+1} ' ' ...
            nodi_A{k} ' ' nodi_A{k+1} ' ' ...
            num2str(1/kncir(k,indm-1),'%0.16e') ' \n'];
    end
end
%terminazione della netlist
netlist = [netlist '* TERMINAZIONE \n'];
netlist = [netlist ' .ENDS \n'];
%salvataggio del file .lib
```

```
fid = fopen ([nomecomponente '.lib'],'wt');
fprintf(fid, netlist);
fclose(fid);
```

5.2 Appendice B: Implementazione Matlab per la risoluzione del sistema della rete di riferimento per il caso bidimensionale

```
format long
 %Numero di quadrati per riga, ad esempio 3
Q=3;
%Costruisci le matrici Qa e Qb
Qa = zeros(Q+1,Q);
for ind=1:Q
   Qa(ind,ind)=-1;
end
for ind=2:Q+1
    Qa(ind,ind-1)=1;
end
Qa;

Qb = zeros(2*(Q+1),Q+1);
for ind=1:Q+1
    Qb(ind,ind)=-1;
    Qb(ind+Q+1,ind)=1;
end
Qb;
%Costruisci la matrice A a partire da Qa e Qb
numerolati = Q*(Q+1) + (Q+1)*Q; %orizzontali + verticali
Aa = zeros((Q+1)^2, numerolati);
%Aggiungi le Qa
for ind=1:Q+1
```

```
    startrowQa = 1+(ind-1)*(Q+1);
    endrowQa = startrowQa + size(Qa,1)-1;
    startcolumnQa = 1+(ind-1)*(2*Q+1);
    endcolumnQa = startcolumnQa +size(Qa,2)-1;
    Aa(startrowQa:endrowQa,startcolumnQa:endcolumnQa)=Qa;
end
%Aggiungi le Qb
for ind=1:Q
    startrowQb = 1+(ind-1)*(Q+1);
    endrowQb = startrowQb + size(Qb,1)-1;
    startcolumnQb = 1+ind*Q+(ind-1)*(Q+1);
    endcolumnQb = startcolumnQb +size(Qb,2)-1;
    Aa(startrowQb:endrowQb,startcolumnQb:endcolumnQb)=Qb;
end
Aa;
%Togli l'ultima riga
A = Aa(1:end-1,1:end);
%Corrente entrante al solo nodo 1
J = zeros(1,(Q+1)^2-1)';
J(1) = 1;
G= A*diag(ones(1,numerolati))*A';
u = G\J;
```

Bibliografia

[1] Gustavsen B. Computer code for passivity enforcement of rational macromodels by residue perturbation. *IEEE Transactions and Advanced Packaging*, 30(2):209–215, 2007.

[2] Gustavsen B. and Semlyen A. Rational approximation of frequency domain responses by vector fitting. *IEEE Transactions on Power Delivery*, 14(3):1052– 1061, 1999.

[3] T. Bechtold, B. Evgenii, and Jan G Korvink. Dynamic electrothermal simulation of microsystems-a review. *Micromechanics and Microengineering*, 17 October 2005.

[4] CP. Coelho, J. Phillips, and Silveira LM. A convex programming approach for generating guaranteed passive approximations to tabulated frequency-data. *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems*, 23(2):293–301, 2004.

[5] V. d'Alessandro, M. de Magistris, A. Magnani, N. Rinaldi, and Russo S. An application to signal integrity analysis in highly integrated electronic systems. *Electrothermal Dynamical Macromodeling.*

[6] V. d'Alessandro, M. de Magistris, A. Magnani, N. Rinaldi, and Russo S. Dynamic electrothermal analysys of bipolar devices and circuits rely-

ing on multi-port positive fraction foster rapresentation. *IEEE BCTM, Electrothermal effects*, October 1 2012.

[7] V. d'Alessandro, M. de Magistris, A. Magnani, N. Rinaldi, and Russo S. Electrothermal reduced equivalents of highly integrated electronic systems with multi-port positive fraction foster expansion. *IEEE*, 2012.

[8] M. de Magistris. *Appunti di modelli numerici per circuiti*, volume II: Algoritmi per la simulazione circuitale. A.A. 2008-2009.

[9] M. de Magistris and Nicolazzo M. On the concretely passive realization of reduced circuits models based on convex constrained positive real fraction identification. *Fiftheenth IEEE Workshop on Signal Porpagation on Interconnects*, 2011.

[10] L. De Tommasi, M. de Magistris, D. Deschrijver, and Dhaene T. An algorithm for direct identification of passive transfer matrices with positive real fractions via convex programming. *Numerical Modelling: Electronic Networks, Devices and Fields*, 24:375–386, 2010.

[11] L. De Tommasi, D. Deschrijver, and Dhaene T. Single-input-single-output passive macromodeling via positive fractions vector fitting. *Twelfth IEEE Workshop on Signal Propagation on Interconnects*, 2008.

[12] Guillemin EA. *Synthesis of Passive Networks*. Wiley, Chapman and Hall, 1957.

[13] S. Grivet-Talocia and Ubolli A. Passivity enforcment with relative error control. *IEEE Transactions on Microwave Theory and Techniques*, 55(11):2374–2383, 2007.

[14] T. Hopkins, C. Cognetti, and Tiziani R. Designing with thermal impedance. *Semitherm procedings*, 1988.

[15] Massoud Pedram and Shahin Nazarian. Thermal modeling,analysis, and management in vlsi circuits: Principles and methods. *Proceedings of the IEEE*, 94(8), August 2006.

[16] N. Rinaldi. Thermal issues in nanoscale vlsi devices and circuits. *IDESA*.

[17] Andrei Vladimirescu. *The Spice book*. John Wiley and Sons,Inc., 1994.

[18] Y. Zhan, S. V. Kumar, and Sapatnekar S. S. Thermally aware design. *Foundations and Trends in Electronic Design Automation*, 2(3):255–370, 2008.