UNIVERSITAS INDONESIA

ANALISIS FENOMENA *MEMCAPACITOR* DENGAN MENGGUNAKAN *SIMULINK MATLAB*

SKRIPSI

ARIF RAHMANSYAH

0405030117

FAKULTAS TEKNIK
DEPARTEMEN TEKNIK ELEKTRO
DEPOK
DESEMBER 2009

UNIVERSITAS INDONESIA

ANALISIS FENOMENA *MEMCAPACITOR* DENGAN MENGGUNAKAN *SIMULINK MATLAB*

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik

ARIF RAHMANSYAH

0405030117

FAKULTAS TEKNIK

PROGRAM TEKNIK ELEKTRO

DEPOK

DESEMBER 2009

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri,

dan semua sumber baik yang dikutip maupun dirujuk

telah saya nyatakan dengan benar.

Nama	**: Arif Rahmansyah**
NPM	**: 0405030117**
Tanda Tangan	**:**
Tanggal	**: 15 Desember 2009**

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh	:	
Nama	:	Arif Rahmansyah
NPM	:	0405030117
Program Studi	:	Teknik Elektro
Judul Skripsi	:	Analisis Fenomena *Memcapacitor* dengan Menggunakan *Simulink Matlab*

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Elektro, Fakultas Teknik, Universitas Indonesia

DEWAN PENGUJI

Pembimbing	: Prof. Dr. Ir. Djoko Hartanto, M.Sc	( )
Penguji	: Dr. Ir. Purnomo Sidi Priambodo, Ph.D	( )
Penguji	: Prof. Dr. Ir. Nji Raden Poespawati, MT	( )

Ditetapkan di : Depok

Tanggal : 30 Desember 2009

KATA PENGANTAR

Puji syukur atas kehadirat Allah SWT yang telah memberikan inspirasi dan kemudahan kepada penulis untuk menyelesaikan skripsi ini.

Penulisan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Teknik Departemen Tenik Elektro Fakultas Teknik Universitas Indonesia. Skripsi ini dapat terselesaikan atas bantuan serta dukungan banyak pihak. Penulis mengucapkan terima kasih kepada:

(1) Prof. Dr. Ir. Djoko Hartanto, MSc sebagai dosen pembimbing yang telah menentukan dan menyetujui riset ini sebagai bagian dari riset pada Sensor Device Research Group, bersedia meluangkan waktunya untuk memberikan saran, bimbingan, serta pengarahan dalam menyeselesaikan riset ini,

(2) Citra Purdiaswari selaku rekan satu bimbingan yang telah membantu dalam proses penulisan skripsi dan Franciskus Arthur M. yang telah membantu dalam perancangan program simulasi,

(3) Orang tua dan keluarga serta sahabat yang telah membantu.

Akhir kata, saya berharap Allah SWT berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga skripsi ini membawa manfaat bagi pengembangan ilmu.

Depok, 15 Desember 2009

Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI

TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini :

Nama	:	Arif Rahmansyah
NPM	:	0405030117
Program Studi	:	Teknik Elektro
Fakultas	:	Teknik
Jenis karya	:	Skripsi

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (Non-eksklusif Royalty-Free Right)** atas karya ilmiah saya yang berjudul:

ANALISIS FENOMENA *MEMCAPACITOR* DENGAN MENGGUNAKAN *SIMULINK MATLAB*

Beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/format-kan, mengelola dalam bentuk pangkalan data (database), merawat, dan mempublikasikan tugas akhir saya selama tetap mencantumkan saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada Tanggal : 15 Desember 2009

Yang menyatakan

(Arif Rahmansyah)

ABSTRAK

Nama : Arif Rahmansyah

Program Studi : Teknik Elektro

Judul : Analisis Fenomena *Memcapacitor* dengan Menggunakan *Simulink Matlab*

Perkembangan teknologi yang semakin cepat dari hari ke hari telah melahirkan banyak penemuan baru dalam bidang elektronika. Salah satu penemuan mutakhir dalam bidang elektronika adalah ditemukannya komponen bermemori (*memory elements*), yaitu *memristor*, *memcapacitor*, dan *meminductor*. Dalam riset pada skripsi ini, dilakukan penjelasan tentang fenomena, karakteristik, dan prinsip kerja dari *memcapacitor*, penurunan rumus umum dari memristor, dibuat *memcapacitor emulator* dengan memodifikasi *memristor emulator* untuk mensimulasikan *memcapacitor*. Hasil simulasi menunjukkan bahwa *memcapacitor* bekerja sebagai kapasitor non-linier yang dapat menyimpan energi dan memori pada kisaran frekuensi antara 24 Hz hingga 230 Hz.

Kata kunci :

memori, *memcapacitor*, frekuensi, energi.

ABSTRACT

Name	:	Arif Rahmansyah
Study Program	:	Electrical Engineering
Title	:	Analysis of Memcapacitor's Phenomenon Using Simulink MATLAB

Technological development accelerated from day to day has given birth to many new discoveries in the field of electronics. One of the recent discoveries in the field of electronics is the discovery of memory elements, namely memristor, memcapacitor, and meminductor. In the research on this thesis, described the phenomenon, characteristics, and the principle of memcapacitor, generating the general equation of memristor, built memcapacitor emulator by modifying memristor emulator to simulate the memcapacitor, and analyze the simulation results. Simulation results show that memcapacitor worked as a non-linear capacitor that can store energy and memory in the range of frequencies between 24 Hz to 230 Hz.

Key words:
memory, memcapacitor, frequency, energy.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

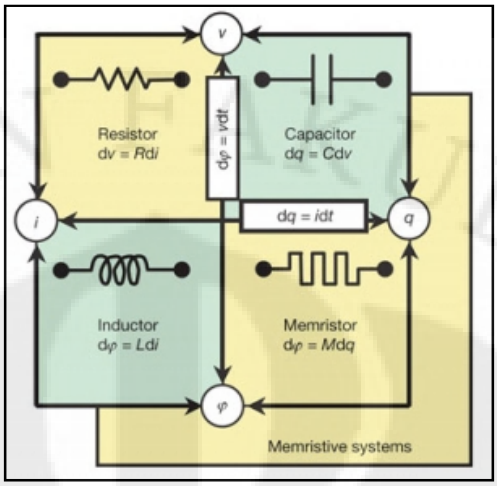
BAB 1

PENDAHULUAN

1.1 Latar Belakang Masalah

Dalam dunia elektronika, ada tiga komponen yang merupakan komponen-komponen dasar pembentuk suatu rangkaian elektronika. Tiga komponen ini adalah resistor, kapasitor, dan induktor. Ketiga komponen ini termasuk jenis komponen pasif, karena komponen akan bekerja pada tegangan atau arus tanpa memerlukan syarat tertentu, tanpa memerlukan pemicu untuk aktif bekerja. Dengan tegangan berapapun dan arus berapapun ketiga komponen ini akan melaksanakan tugasnya sesuai dengan besaran yang dimiliki oleh komponen tersebut. Hingga saat ini, penggunaan ketiga komponen ini sudah sangat luas, dan dapat ditemukan hampir pada setiap rangkaian elektronika, baik yang sederhana, maupun yang kompleks. Pada tahun 1971, Prof. Leon Chua, salah satu professor di *University of California, Berkeley*, mengemukakan hipotesisnya dalam sebuah paper [1] tentang komponen dasar keempat selain resisitor, kapasitor, dan induktor, yaitu memristor. Memristor merupakan singkatan dari *memory-resistor*, yang secara sederhana didefinisikan sebagai resistor yang dapat mengingat muatan yang pernah melewatinya. Jika resistor merepresentasikan hubungan antara tegangan dengan arus, kapasitor merepresentasikan hubungan antara muatan dengan tegangan, induktor merepresentasikan hubungan antara flux listrik dengan arus, maka memristor merepresentasikan hubungan antara muatan dengan flux listrik. Hubungan-hubungan tersebut dapat dilihat pada **Gambar 1.1**.

Pada tahun 2008, *HP Laboratory* berhasil membuat suatu *prototype* divais yang bekerja sesuai prinsip kerja memristor [2]. Prinsip kerja memristor menyerupai resistor biasa. Perbedaannya, memristor dapat menyimpan muatan yang pernah melewatinya atau dengan kata lain, memristor dapat mengingat resistansinya. Dengan ditemukannya *prototype* ini, berbagai kemungkinan mengenai pemanfaatan memristor pun mulai bermunculan. Salah satunya adalah penggunaan memristor sebagai *non-volatile memory*.



Gambar 1. 1. Hubungan antar variabel listrik [3]

Dengan digunakannya memristor sebagai *non-volatile memory*, nantinya komputer-komputer akan dapat dengan cepat dinyalakan tanpa harus menunggu waktu *booting* (pembacaan data dari *harddisk*) yang cukup lama seperti sekarang ini. Hal ini dikarenakan memristor dapat mengingat tegangan yang pernah melewatinya, sehingga ia akan dapat dengan mudah diaktifkan kembali, berbeda dengan *DRAM* yang kehilangan informasinya ketika tegangan dilepaskan (bersifat *volatile memory*).

Pada Januari 2009, dalam suatu jurnal [4] dijelaskan tentang fenomena lain dari divais memori (komponen yang memiliki memory). Ternyata ditemukan bahwa selain memristor (fenomenanya disebut dengan *memristive system*), ada pula memkapasitor dan meminduktor. Untuk dua yang terakhir ini, fenomenanya disebut dengan *memcapacitative system* dan *meminductive system*. Para peneliti yang membuat jurnal tersebut menemukan bahwa fenomena memristive dapat pula diaplikasikan pada kapasitor dan induktor, sehingga lahirlah istilah *memcapacitative system* dan *meminductive system*. Jika kapasitor dan induktor biasa hanya bisa menyimpan energi, maka memkapasitor dan meminduktor ini selain dapat menyimpan energi, komponen ini juga dapat menyimpan kondisi sebelumnya (*past state*).

Pada dasarnya, masing-masing komponen baru ini merepresentasikan hubungan yang sama antara dua variabel seperti pada komponen tanpa memori

(resistor, kapasitor, induktor), yaitu arus-tegangan pada memristor, tegangan-muatan pada memkapasitor, dan muatan-flux pada meminduktor. Perbedaan mendasar dari memristor dengan memkapasitor dan meminduktor adalah bahwa keduanya dapat menyimpan energi, sedangkan memristor tidak. Selain itu, perbedaan ketiganya dengan komponen tanpa memory adalah kemampuan ketiganya dalam mengingat kondisi sebelumnya.

1.2 Tujuan Penulisan

Tujuan penulisan skripsi ini adalah membuktikan fenomena memkapasitor dengan melakukan simulasi dan menganalisis hasil simulasi untuk mengetahui frekuensi kerja memkapasitor.

1.3 Batasan Masalah

Pembahasan pada skripsi ini dibatasi pada penjelasan tentang *memory device*, khususnya memristor dan memkapasitor, fenomena, karaketristik, dan prinsip kerja dari memkapasitor serta menurunkan fungsi persamaan memkapasitor untuk digunakan dalam simulasi kerja memkapasitor. Berdasarkan studi literatur, maka skripsi ini dibatasi untuk nilai w(t) pada kisaran $0 \leq w(t)/D \leq 1$.

1.4 Metode Penulisan

Metode yang digunakan dalam skripsi ini adalah pendekatan tinjauan pustaka, yaitu dengan melakukan studi literatur dari buku-buku pustaka serta buku referensi dan jurnal referensi yang membahas divais memori, pendekatan diskusi dengan pembimbing skripsi dan pihak-pihak yang terkait dengan topik bahasan skripsi, serta membuat simulasi memkapasitor untuk mengetahui frekuensi kerja dari memkapasitor.

1.5 Sistematika Penulisan

Sistematika penulisan skripsi ini adalah sebagai berikut. Bab satu berisi tentang latar belakang masalah, batasan masalah, tujuan penulisan, metode penulisan, dan sistematika penulisan. Bab dua menjelaskan tentang teori

komponen dasar tanpa memori (resistor, kapasitor, dan induktor), teori komponen bermemori (memristor, memkapasitor, dan meminduktor), penjelasan tentang karakteristik memkapasitor, penjelasan konsep TiO_2 memristor, penjelasan *hysteresis loop*, dan implementasi memristor pada memkapasitor. Bab tiga menjelaskan tentang penurunan rumus dari memristor, penurunan rumus dari memkapasitor, simulasi memristor, dan simulasi memkapasitor. Bab empat menjelaskan tentang analisa kurva V-Q Memkapasitor untuk mengetahui frekuensi kerja dari memkapasitor. Bab lima berisi kesimpulan dari simulasi yang dilakukan.

BAB 2
DASAR TEORI

2.1 Komponen Elektronika Dasar

2.1.1 Resistor

Resistor berfungsi sebagai penahan arus listrik, digunakan untuk membatasi jumlah arus yang mengalir dalam rangkaian. Karakteristik utamanya adalah resistansi (R), menunjukkan besar nilai arus tahanannya. Setiap bahan memiliki sifat resistif yang berbeda-beda. Besar nilai resistansi ditentukan oleh faktor hambatan jenis bahan, panjang penampang bahan, dan luas penampang bahan. Secara matematis ditunjukkan pada Persamaan (2.1) [5].

$$= - \quad (2.1)$$

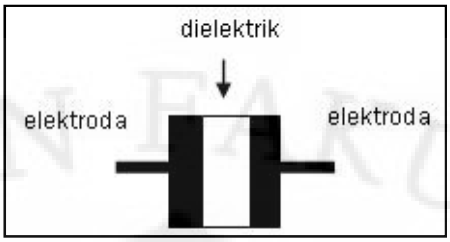
Secara matematis, hubungan antara arus, tegangan, dan nilai hambatan adalah ditunjukkan pada Persamaan (2.2) [5].

$$=- \quad (2.2)$$

2.1.2 Kapasitor

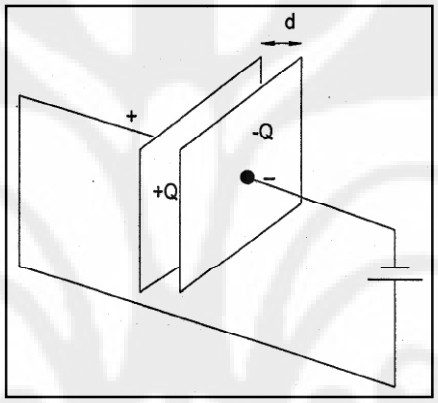
Kapasitor atau juga sering disebut dengan kondensator merupakan komponen elektronika yang dapat menyimpan energi atau muatan listrik dalam kurun waktu tertentu tanpa disertai terjadinya reaksi kimia, yaitu dengan cara mengumpulkan ketidakseimbangan internal dari muatan listrik [6].

Struktur sebuah kapasitor terbuat dari 2 buah keping logam yang dipisahkan oleh suatu bahan dielektrik, seperti diperlihatkan pada **Gambar 2.1.** Bahan-bahan dielektrik yang umum digunakan misalnya hampa udara, keramik, gelas dan lain-lain. Jika kedua ujung keping logam diberi tegangan listrik, maka muatan-muatan positif akan mengumpul pada salah satu kaki (elektroda) logamnya dan pada saat yang sama muatan-muatan negatif terkumpul pada ujung logam yang satu lagi. Muatan positif tidak dapat mengalir menuju ujung kutub negatif dan sebaliknya muatan negatif tidak bisa menuju ke ujung kutub positif, karena terpisah oleh bahan dielektrik yang non-konduktif. Muatan elektrik ini tersimpan selama tidak ada konduksi pada ujung-ujung kakinya.



Gambar 2.1. Elektroda kapasitor [6]

Untuk lebih jelasnya, perhatikan gambar rangkaian dalam kapasitor pada **Gambar 2.2.**



Gambar 2.2. Rangkaian dalam kapasitor [7]

Misalkan tegangan DC dikenakan pada kedua keping seperti ditunjukkan pada **Gambar 2.2.** Kedua keping tersebut dipisahkan oleh suatu isolator, sehingga pada dasarnya tidak ada elektron yang dapat menyeberang celah di antara kedua keping. Pada saat baterai belum terhubung, kedua keping akan bersifat netral (belum temuati). Saat baterai terhubung, titik dimana kawat pada ujung kutub negatif dihubungkan akan menolak elektron, sedangkan titik dimana kutub positif terhubungkan menarik elektron. Elektron-elektron tersebut akan tersebar ke seluruh keping kapasitor. Elektron mengalir ke dalam keping sebelah kanan dan elektron mengalir keluar dari keping sebelah kiri dalam waktu yang relatif singkat. Pada kondisi ini, arus mengalir melalui kapasitor walaupun sebenamya tidak ada elektron yang mengalir melalui celah kedua keping tersebut. Setelah bagian luar dari keping terisi muatan, secara perlahan bagian luar ini akan menolak muatan baru dari sumber tegangan. Oleh karena itu, arus pada keping tersebut akan menurun besarnya terhadap waktu sampai kedua keping tersebut berada pada

tegangan yang dimiliki baterai. Keping sebelah kanan akan memiliki kelebihan elektron yang terukur dengan muatan -*Q* dan pada keping sebelah kiri termuati sebesar +*Q*. Besarnya muatan *Q* ini sebanding dengan *V* atau secara matematis ditulis sesuai Persamaan (2.3) [7].

$$Q \sim V \tag{2.3}$$

Q merupakan kelipatan dari V sehingga secara matematis ditulis sesuai Persamaan (2.4) [7].

$$Q = C\,V \tag{2.4}$$

dimana satuan kapasitansi ini dinyatakan dengan Farad (F).

Secara umum hubungan antara muatan dan tegangan untuk sebuah kapasitor dapat dituliskan sesuai Persamaan (2.5) [7].

$$q = C\,v \tag{2.5}$$

Dengan demikian arus *i* yang mengalir diberikan oleh I pada Persamaan (2.6) dan Persamaan (2.7) [7].

$$= — = \quad — \tag{2.6}$$

$$= – = – \int \quad + \tag{2.7}$$

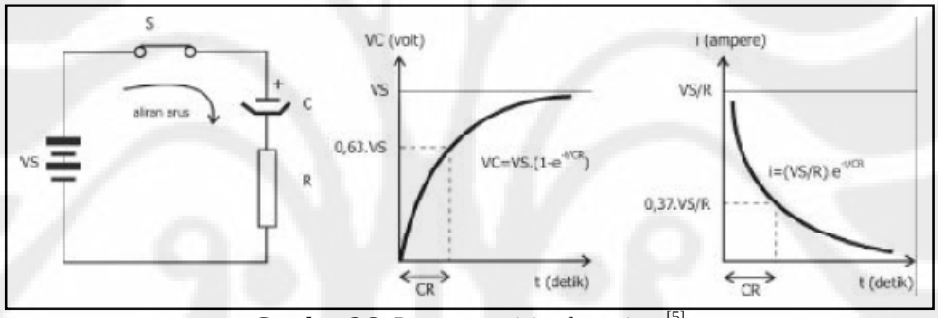
Dalam praktek pembuatan kapasitor, kapasitansi dihitung dengan mengetahui luas area keping metal (A), jarak (d) antara kedua keping metal (tebal dielektrik) dan konstanta (k) bahan dielektrik. Dengan rumus dapat di tulis sebagai Persamaan (2.8) [6].

$$=(8.85 \quad 10 \quad) \; – \tag{2.8}$$

Selain kapasitansi, karakteristik kapasitor lainnya yang tidak kalah penting adalah tegangan kerja dan temperatur kerja. Tegangan kerja adalah tegangan maksimum yang diizinkan agar kapasitor masih dapat bekerja dengan baik. Misal, kapasitor 10uF25V, maka tegangan yang diberikan tidak boleh melebihi 25 volt dc. Umumnya kapasitor-kapasitor polar bekerja pada tegangan DC dan kapasitor non-polar bekerja pada tegangan AC. Sedangkan temperatur kerja yaitu batasan temperatur dimana kapasitor masih bisa bekerja dengan optimal. Misalnya, jika

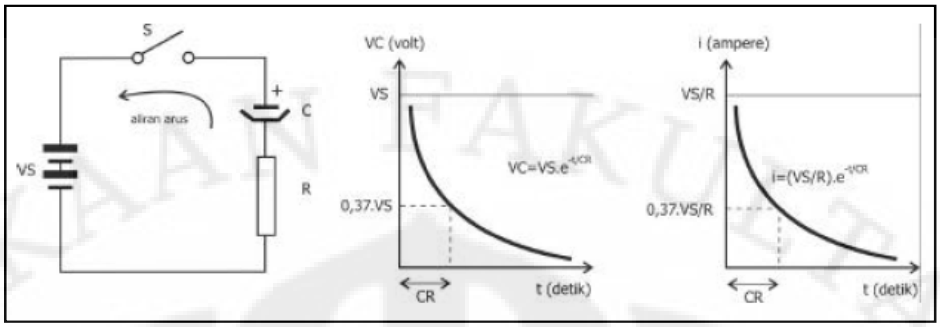
pada kapasitor tertulis X7R, maka kapasitor tersebut mempunyai suhu kerja yang direkomendasikan antara -55C^{o} sampai +125C^{o}. Biasanya spesifikasi karakteristik ini disajikan oleh pabrik pembuat di dalam *datasheet*.

Pada prinsipnya proses pengisian dan pengosongan kapasitor dapat dilihat pada **Gambar 2.3** dan **Gambar 2.4.** Pada proses pengisian kapasitor, ketika saklar S ditutup, tegangan V_S akan menyebabkan arus mengalir ke dalam salah satu sisi kapasitor dan keluar dari sisi yang lainnya, arus ini tidak tetap karena ada penyekat dielektrik sehingga arus menurun ketika muatan pada kapasitor meninggi sampai $V_C = V_S$ ketika i = 0.



Gambar 2.3. Proses pengisian kapasitor [5]

Proses pengosongan kapasitor hampir sama dengan pengisian, dengan dihilangkannya sumber tegangan, rangkaian menjadi terhubung singkat. Ketika saklar S dibuka, arus mengalir dari salah satu sisi kapasitor yang mengandung muatan listrik ke sisi yang lainnya. Ketika V_C menjadi nol maka arus juga menghilang. Kalau dihubungkan dengan sirkuit AC (bolak-balik), kapasitor akan terisi oleh tegangan searah dan kemudian menutup aliran arus selanjutnya, serta kapasitor akan terisi dan kosong secara kontinyu dan arus bolak-balik mengalir dalam rangkaian.



Gambar 2.4. Proses pengosongan kapasitor [5]

2.1.3 Induktor

Induktor adalah komponen elektronika yang dapat menyimpan energi pada medan magnet yang ditimbulkan oleh arus listrik yang melewatinya. Induktor biasanya terbuat dari kawat penghantar yang dililit membentuk kumparan. Lilitan kawat penghantar ini menghasilkan medan magnet dalam kumparan karena adanya induksi magnet. Berikut ini akan dijelaskan secara singkat cara kerja induktor.

Kawat penghantar dialiri arus listrik sehingga akan terbentuk medan magnet, sesuai Persamaan (2.9) [8].

$$= \frac{}{} \qquad (2.9)$$

kawat penghantar memiliki suatu luas area tertentu, sehingga dengan adanya medan magnet ini, akan timbul flux magnet, sesuai Persamaan (2.10) [8].

$$\Phi = BA \qquad (2.10)$$

Besarnya medan magnet akan berubah sesuai dengan perubahan arus. Perubahan medan magnet ini akan menginduksi suatu tegangan pada koil. Hal ini terjadi karena suatu sifat yang disebut dengan induksi diri atau sering disebut dengan induktansi (L). Induktansi diukur berdasarkan jumlah gaya elektromotif yang ditimbulkan untuk setiap perubahan arus terhadap waktu, sesuai Persamaan (2.11) [8].

$$= - \frac{}{} \qquad (2.11)$$

dimana E menyatakan tegangan yang timbul jika induktor di aliri listrik, dan bilangan negatif sesuai dengan hukum **Lenz** yang mengatakan efek induksi cenderung melawan perubahan yang menyebabkannya. Satuan untuk induktansi

adalah Henry (H). Hubungan antara tegangan, arus, dan induktansi dinyatakan oleh Persamaan (2.12) [8].

$$( ) = \quad - \tag{2.12}$$

2.2 Pengenalan Divais Bermemori

Divais bermemori (*memory device*) didefinisikan sebagai suatu divais atau komponen yang memiliki kemampuan untuk mengingat sesuatu (muatan atau tegangan) yang pernah melewatinya atau mengingat kondisi sebelumnya/kondisi yang telah lalu (*past state*). Konsep mengenai divais memori ini dicetuskan pertama kali oleh Prof. Leon Chua, salah satu professor di *University of California, Berkeley*, yang memperkenalkan tentang memristor (*memory-resistor*), pada tahun 1971. Dalam hipotesisnya, ia memperkenalkan memristor sebagai komponen keempat setelah tiga komponen yang telah disebutkan sebelumnya (resistor, kapasitor, dan induktor). Memristor ini didefinisikan sebagai komponen yang mampu mengingat muatan yang pernah melewatinya.

Dalam hipotesanya, Prof Leon Chua menjelaskan bahwa memristor merupakan komponen keempat yang menyatakan hubungan antara muatan dengan tegangan. Jika dilihat dari hubungan antara variabel listrik (tegangan, arus, muatan, flux), maka satu-satunya hubungan variabel yang belum diwakilkan oleh suatu komponen dasar adalah hubungan antara muatan dengan flux, yang menurut Prof. Leon Chua, hubungan ini diwakilkan oleh memristor. Secara matematis, hubungan muatan dengan flux ini didefinisikan sebagai integral dari tegangan terhadap waktu, yang tidak membutuhkan penafsiran dari flux magnet. Hubungan ini dapat digeneralisasi untuk setiap divais yang memiliki dua terminal, yang resistansinya tergantung dari kondisi internal dari sistemnya, dan disebut dengan sistem memristif.

Ada banyak sistem yang menunjukkan sistem memristif [4], seperti *thermistor* (kondisi internalnya tergantung dari temperaturnya), molekul (yang resistansinya berubah menurut konfigurasi atomnya), atau divais *spintronic* yang resistansinya bervariasi menurut perputaran polarisasinya.

Pada tahun 2008, tepatnya pada bulan April 2008, para peneliti di *HP Laboratory* berhasil membuat suatu prototipe dari memristor [2]. Dalam

publikasinya, *HP Lab* melaporkan bahwa mereka berhasil membuat suatu divais yang bekerja sesuai dengan konsep memristor atau disebut juga memiliki sifat memristif (*memristive behavior*). Secara umum, hal ini disebut dengan *memristive system.* Dalam beberapa penelitian [4], sifat memristif dan kemampuan menyimpan memori dilaporkan telah dapat bekerja pada *solid-state* TiO_2 *thin films*, dimana perubahan resistansinya terjadi karena adanya pergerakan ionis kekosongan oksigen yang diaktifkan oleh aliran muatan. Selain itu, sifat memristif juga telah didemonstrasikan pada film tipis VO_2, dimana mekanisme penyimpanan memori berhubungan dengan proses peralihan isolator menjadi metal (*insulator-to-metal transition*) yang terjadi pada struktur VO_2. Terakhir, dari hasil penemuan terkini, sifat memristif ini diperkenalkan sebagai suatu mekanisme yang mungkin terjadi dalam sistem adaptasi pada hewan bersel tunggal seperti *amoeba*.

Semua contoh diatas membuktikan terjadinya sifat memristif yang alami pada beberapa bahan/makhluk. Faktanya, bukan kejutan lagi bahwa dari contoh diatas, semuanya terjadi pada skala nano, yang resistansi sistemnya kemungkinan besar tergantung dari kondisi internal dan sejarah dinamis sistem, setidaknya dalam skala waktu yang diberikan oleh variabel kondisi dasar yang mengontrol kerjanya.

Belakangan, para peneliti kembali menunjukkan kemajuan dalam studi tentang divais memori. Hal ini ditunjukkan dalam salah satu jurnal yang menjelaskan bahwa konsep divais memori tidak terbatas hanya pada resistansi, tetapi juga dapat diterapkan pada sistem kapasitif dan induktif.

2.3 *Memory Device System*

Sistem divais memori, yang meliputi *memristive system*, *memcapacitive system*, dan *meminductive system*. Secara umum, persamaan untuk divais memori dapat ditulis sesuai Persamaan (2.13) dan Persamaan (2.14) [4].

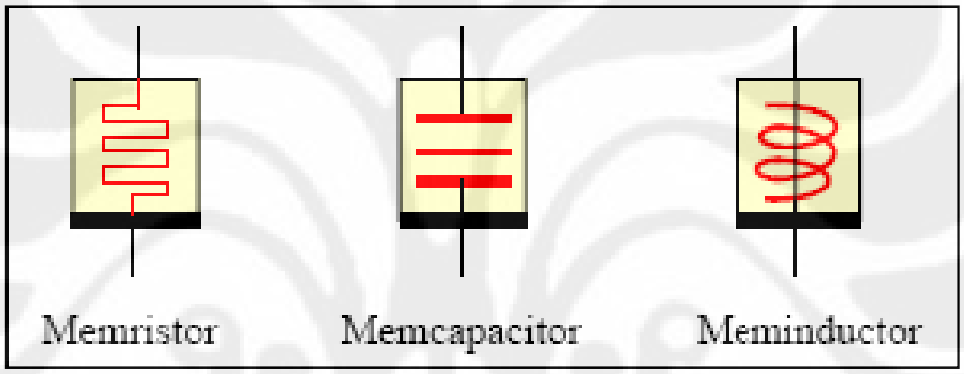
$$y(t) = g(x, u, t)\, u(t) \tag{2.13}$$

$$\dot{x} = f(x, u, t) \tag{2.14}$$

Pada Persamaan (2.13) dan Persamaan (2.14), *x* menyatakan suatu n variabel kondisi yang menggambarkan kondisi internal sistem, u(t) dan y(t) adalah variabel fundamental (yaitu arus, tegangan, muatan, atau flux) yang menyatakan masukan

dan keluaran sistem, *g* menunjukkan fungsi respon, dan f adalah fungsi vektor n-dimensi yang kontinyu, dan diasumsikan jika kondisi awal yang diberikan adalah $u(t = t_0)$ saat t_0, maka Persamaan (2.14) akan memberikan solusi yang unik.

Memcapacitive dan *meminductive* adalah peristiwa khusus dari Persamaan (2.13) dan Persamaan (2.14), dimana dua variabel yang menyatakannya adalah muatan dan tegangan untuk *memcapacitance*, dan arus dan flux untuk *meminductance*. Untuk symbol dari memristor, memkapasitor, dan meminduktor, perhatikan **Gambar 2.5**.



Gambar 2.5. Simbol *Memory device* [4]

2.3.1 *Memristive System*

Dari Persamaan (2.13) dan Persamaan (2.14), suatu sistem memristif arus terkontrol berorde n dinyatakan oleh Persamaan (2.15) dan Persamaan (2.16) [4].

$$V_M(t) = R(x, I, t)\, I(t) \quad (2.15)$$

$$\dot{x} = f(x, I, t) \quad (2.16)$$

dengan x adalah besaran vektor yang mewakili n variabel kondisi internal, $V_M(t)$ dan I(t) menyatakan tegangan dan arus yang melalui divais, dan R adalah besaran skalar, yang disebut sebagai memristansi dengan satuan Ohm. Persamaan untuk memristor arus terkontrol adalah kasus khusus dari Persamaan (2.15) dan Persamaan (2.16), ketika R hanya tergantung pada muatan yang ditujukkan pada Persamaan (2.17) [4].

$$V_M = R(q(t))\, I \quad (2.17)$$

dimana muatan q(t) merupakan fungsi waktu dari I pada Persamaan (2.18) [4].

$$I = dq/dt \quad (2.18)$$

sedangkan untuk sistem memristive tegangan terkontrol memiliki Persamaan yang ditunjukkan pada Persamaan (2.19) dan Persamaan (2.20) [4].

$$I(t) = G(x, V_M, t)\, V_M(t) \quad (2.19)$$

$$\dot{x} = f(x, V_M, t) \quad (2.20)$$

dengan *G* adalah memduktansi (memori-konduktansi)

2.3.2 *Memcapacitive System*

Suatu sistem memkapasitif tegangan terkontrol berorde n dinyatakan oleh : Persamaan (2.21) dan Persamaan (2.22) [4].

$$q(t) = C(x, V_C, t)\, V_C(t) \quad (2.21)$$

$$\dot{x} = f(x, V_C, t) \quad (2.22)$$

Dengan q(t) adalah muatan pada kapasitor saat waktu t, Vc(t) adalah tegangan pada kapasitor, dan C adalah memkapasitansi, yang bergantung pada kondisi sistem. Dengan cara yang sama, suatu sistem memkapasitatif muatan terkontrol berorde n dinyatakan oleh Persamaan (2.23) dan Persamaan (2.24) [4].

$$V_C(t) = C^{-1}(x, q, t)\, q(t) \quad (2.23)$$

$$\dot{x} = f(x, q, t) \quad (2.24)$$

C^{-1} adalah invers dari memkapasitansi.

Untuk memkapasitor tegangan terkontrol didapat dari Persamaan (2.21) dan Persamaan (2.22) , sehingga menghasilkan Persamaan (2.25) [4].

$$( ) = \int ( ) \quad ( ) \quad (2.25)$$

Untuk memkapasitor arus terkontrol didapat dari Persamaan (2.23) dan Persamaan (2.24) akan menghasilkan Persamaan (2.26) [4].

$$( ) = \int ( ) \quad ( ) \quad (2.26)$$

Dari Persamaan (2.25) dan Persamaan (2.26), batas bawah integral dapat dipilih antara -∞, atau 0 jika $\int ( ) = 0$ dan $\int ( ) = 0$.

2.4 Memkapasitor

2.4.1 Fenomena Memkapasitor

Untuk menjelaskan sistem memkapasitif, maka perhatikan kembali Persamaan (2.21) dan Persamaan (2.22). Dari Persamaan (2.21), diketahui bahwa muatan akan benilai nol, jika tegangan bernilai nol, sehingga pada kasus ini, I = 0 tidak menyebabkan q = 0 dan dengan demikian divais ini akan dapat menyimpan energi (dengan kata lain, muatan dalam kapasitor dipengaruhi oleh tegangan, bukan arus). Dalam pembahasan selanjutnya, energi ini dapat ditambahkan dan ditiadakan dari sistem memkapasitif.

Dalam skala mikroskopi, perubahan kapasitansi dari suatu kapasitor dapat disebabkan oleh, dua hal :

(1) perubahan geometri dari kapasitor itu sendiri (misalnya, perubahan pada bentuk strukturnya), atau

(2) perubahan properti kuantum mekanik dari pembawa (*carrier)* dan lompatan muatan pada material yang membentuk kapasitor, atau keduanya.

Efek inelastisitas (disipasi) mungkin saja terlibat dalam terjadinya perubahan kapasitansi sistem terhadap parameter kontrol eksternal (seperti tegangan dan arus). Proses disipasi ini melepaskan energi dalam bentuk panas pada material yang membentuk kapasitor. Bagaimanapun juga, panas ini tidak dapat dianggap sebagai resistansi seperti pada kapasitor.

Dalam suatu kondisi, akan terjadi kondisi dimana energi tidak berasal dari parameter kontrol, tetapi dari sumber yang mengontrol persamaan variabel keadaan, (Persamaan (2.22)), yang dibutuhkan untuk mengubah kapasitansi sistem. Energi ini nantinya akan dapat dilepaskan ke dalam rangkaian sehingga meningkatkan arus. Oleh karena itu, Persamaan (2.21) dan Persamaan (2.22) untuk sistem memkapasitif yang dipaparkan diatas, pada prinsipnya, dapat menggambarkan divais aktif dan pasif sekaligus.

Artinya, jika syarat-syarat kondisinya terpenuhi, maka sistem memkapasitif ini akan dapat bekerja baik sebagai komponen aktif maupun komponen pasif.

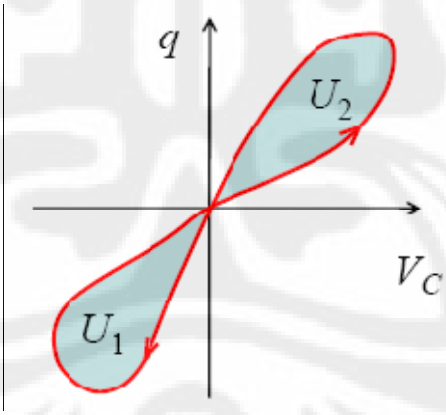
2.4.2 Karakteristik Memkapasitor

Dimulai dari kondisi saat kapasitor kosong, jumlah energi yang dilepaskan dari sistem memkapasitor tidak dapat melebihi jumlah energi yang sebelumnya telah terisi. Secara matematis, dapat dituliskan sebagai Persamaan (2.27) [4].

$$= \int \quad ( ) \, ( ) \qquad \geq 0 \tag{2.27}$$

dengan catatan, saat t = t_0, tidak ada energi yang tersimpan pada sistem. Persamaan (2.27) akan memenuhi untuk bentuk apapun dari parameter kontrol, seperti tegangan $V_C(t)$ yang diberikan pada sistem memkapasitif pasif. Lebih jauh, pada sistem memkapasitif dengan proses disipasi (seperti panas), tanda pertidaksamaan pada Persamaan (2.27) tidak terpenuhi untuk t > t_0 (dengan asumsi $V_C(t) \neq 0$)

Gambar 2.6 menunjukkan grafik *hysteresis loop* yang melalui pusat sumbu (0,0) pada sistem memkapasitif. Daerah yang berbayang menunjukkan energi *Ui* yang ditambahkan ke sistem atau dilepaskan dari sistem. Catat bahwa energi ini berhubungan dengan suatu derajat kebebasan internal, yaitu suatu proses elastis atau nonelastis yang mengiringi perubahan kapasitansi. Sistem akan pasif jika $U_1+U_2 = 0$, disipasi jika $U_1+U_2 > 0$, dan aktif jika $U_1+U_2 < 0$ [4].



Gambar 2.6. Skematik dari sistem memkapasitif [4].

Akhirnya, karena persamaan kondisi pada Persamaan (2.27) hanya memiliki sebuah solusi unik untuk setiap waktu t ≥ t0 yang diberikan, jika $V_C(t)$ adalah periodik dengan frekuensi ω, yaitu $V_C(t) = V_0 \sin(2\pi\omega t)$, maka kurva q – V_C yang terbentuk adalah suatu *loop* sederhana yang melalui pusat sumbu *Cartesian*, sehingga akan ada paling banyak dua nilai muatan q untuk tegangan

V_C yang diberikan, jika divais dianggap sebagai sistem tegangan terkontrol, atau dua nilai tegangan V_C untuk muatan q yang diberikan, jika divais dianggap sebagai sistem muatan terkontrol. *Loop* ini juga antisimetris terhadap pusat sumbu, jika pada Persamaan (2.21) dan Persamaan (2.22), $C(x,V_C,t) = C(x,-V_C,t)$ dan $f(x,V_C,t) = f(x,-V_C,t)$.

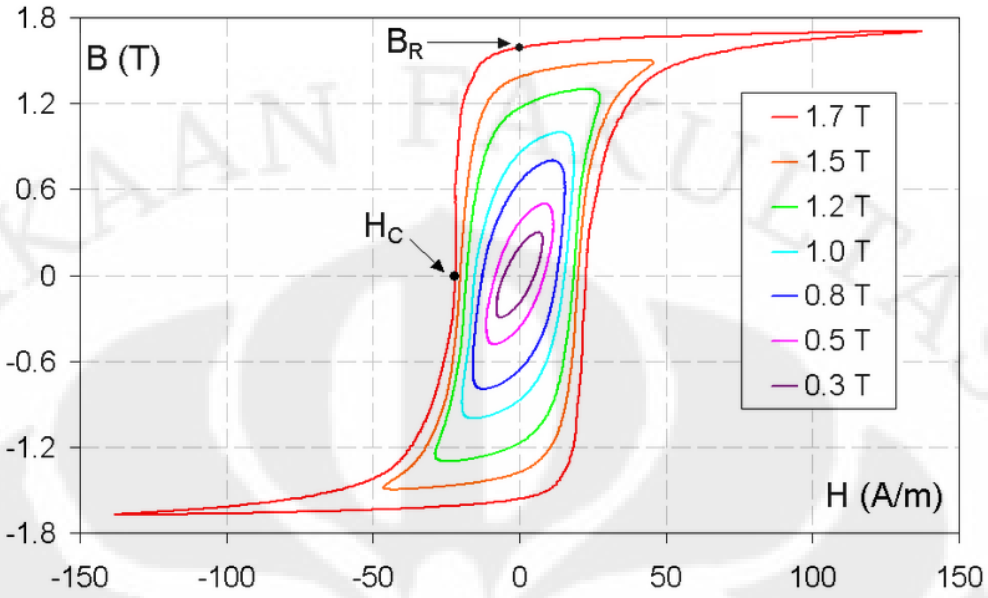
Seperti peristiwa pada sistem memristif, sistem memkapasitif akan bekerja sebagai kapasitor linier pada frekuensi mendekati tak hingga, dan bekerja sebagai kapasitor nonlinier pada frekuensi mendekati nol, dengan mengasumsikan Persamaan (2.22) dan Persamaan (2.24) menghasilkan suatu solusi *steady-state*. Penyebab timbulnya sifat ini berlandaskan pada kemampuan sistem dalam mengatur perubahan bias yang perlahan-lahan (pada frekuensi rendah) dan sebaliknya, ketidakmampuannya untuk menangani osilasi pada frekuensi tinggi [4].

2.5 Penjelasan Mengenai *Hysteresis Loop*

Hysteresis loop dikenal sebagai suatu kurva tertutup yang tidak berujung (*continue*) yang menunjukkan hubungan antara variabel yang mengontrol dengan variabel yang terkontrol. Ciri-ciri dari suatu sistem yang memiliki respons berupa *hysteresis loop* adalah sistem tersebut memiliki suatu kondisi tertentu, suatu histori, dan memori. Ini berarti, agar suatu sistem dapat menunjukkan suatu respons berupa *hysteresis loop*, maka sistem tersebut harus memiliki elemen bermemori dan *initial state*.

Ilustrasinya adalah, jika pada sistem tanpa *hysteresis*, dimungkinkan untuk memperkirakan keluaran sistem hanya dengan memerhatikan masukan dari sistem tersebut. Sedangkan pada suatu sistem yang memiliki respons *hysteresis*, hal ini tidak mungkin. Tidak ada cara untuk memperkirakan keluaran sistem tanpa mengetahui kondisi sistem saat itu, dan tidak ada cara untuk mengetahui kondisi sistem tanpa melihat kembali histori dari input yang dimasukkan. Ini berarti, sistem dengan respons *hysteresis* memiliki kemampuan untuk mengingat kondisi masa lalunya sekaligus menyimpannya. Dengan kata lain, sistem semacam ini memiliki memori untuk menyimpan histori dari sistem.

Berikut ini adalah contoh *hysteresis loop* yang terjadi pada material *ferromagnetic*.



Gambar 2.7. *Hysteresis loop* pada *ferromagnetic material* [9]

Salah satu contoh sistem dengan respons berupa *hysteresis loop* adalah *system memcapacitive*. Pada sistem ini, memkapasitor bertindak sebagai komponen yang dapat menyimpan memory dan mengingat kondisi sistem pada waktu sebelumnya (kondisi saat t = t-1). Pada **Gambar 2.7** ditunjukkan respons dari sistem memkapasitif berupa *hysteresis loop* yang menggambarkan hubungan antara q (muatan) dengan Vc (tegangan yang melalui memkapasitor). Hubungan ini merepresentasikan jumlah energi yang tersimpan dalam memkapasitor, yang secara matematis diberikan pada Persamaan (2.27). Perbedaan dengan *hysteresis loop* pada material *ferromagnetic* adalah, pada *system memcapacitive, loop* yang terjadi melalui pusat sumbu (0,0), sedangkan pada material *ferromagnetic, loop* yang terjadi tidak melalui pusat sumbu.

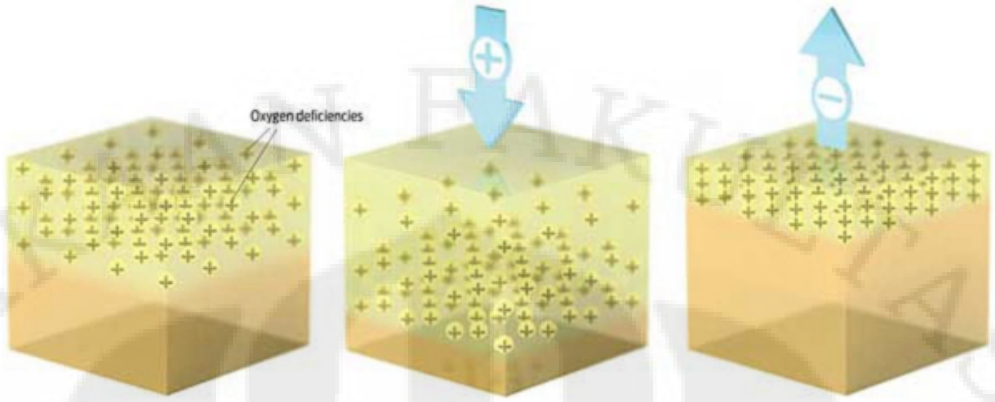
2.6 Mekanisme Kerja dari TiO_2 Memristor

HP Lab telah berhasil membuat suatu memristor dengan material dasar adalah TiO_2. Untuk memudahkan penjelasan mengenai memristor, dapat dianalogikan dengan memerhatikan cara kerja resistor dan dibandingkan dengan cara kerja memristor. Dengan menganggap resistor adalah sebuah pipa untuk mengalirkan air (dalam hal ini arus listrik), dan resistansi dari resistor diwakilkan

oleh lebar pipa tersebut (semakin sempit lebar pipa, maka resistansi semakin tinggi).

Pada resistor normal, ukuran pipa tidak akan berubah berapapun arus listrik yang melewatinya. Sedangkan pada memristor, lebar pipa akan berubah sesuai arus listrik yang melewatinya. Jika arus melewati memristor pada arah tertentu, maka lebar pipa akan membesar (nilai resistansinya mengecil). Namun jika arus mengalir dalam arah sebaliknya, lebar pipa akan mengecil (nilai resistansinya membesar). Saat aliran arus dihentikan, lebar pipa akan sama seperti saat terakhir kali dialiri arus. Dengan kata lain, lebar pipa tidak berubah saat aliran arus dihentikan dan memristor dapat mengingat saat terakhir kali arus mengalir. Ini berarti memristor dapat mengingat sekaligus menyimpan memori masa lalunya.

Pada tahun 2008, HP Lab membuat divais memristor dengan material dasar berupa thin film TiO_2 (titanium dioksdia). Divais ini bekerja berdasarkan mekanisme kimia, yaitu kekosongan atom oksigen pada *layer thin film* TiO_2. Mula-mula dibentuk dua *layer* pada *thin film* TiO_2, dengan salah satu *layer*-nya mengalami sedikit *depletion* (pengosongan) atom oksigen. Kekosongan atom oksigen ini berfungsi sebagai perantara arus, yaitu pada daerah dengan kadar oksigen yang lebih rendah, maka nilai resistansinya jauh lebih kecil dibandingkan dengan daerah yang tidak mengalami pengosongan atom oksigen. Saat medan listrik diberikan, kekosongan atom oksigen ini akan bergeser (ke atas atau ke bawah) sehingga mengubah letak batas yang memisahkan antara daerah yang memiliki resistansi tinggi dengan daerah yang memiliki resistansi rendah. Resistansi sebagai satu kesatuan pada thin film ini tergantung dari berapa banyak arus listrik yang telah melewati divais ini pada arah tertentu, yang arahnya dapat diubah dengan mengubah arah arus listrik yang diberikan. Untuk lebih jelasnya, perhatikan **Gambar 2.8.**



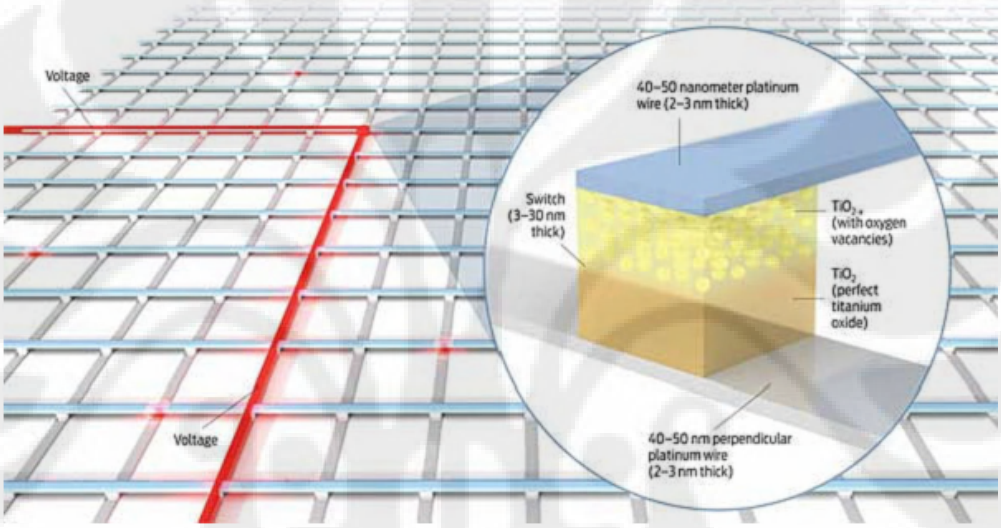
Gambar 2.8. Struktur dalam memristor [2]

Pada **Gambar 2.8.** , lapisan yang berwarna coklat (lapisan TiO_2) merupakan lapisan yang bersifat isolator, dan lapisan diatasnya (lapisan TiO_{2-x}) yang mengandung muatan-muatan positif, merupakan lapisan yang bersifat konduktor. Lapisan yang mengandung muatan positif ini menunjukkan daerah yang mengalami pengosongan atom oksigen (karena kekosongan atom oksigen merupakan pendonor elektron, sehingga menyebabkan kekosongan tersebut bermuatan positif). Karakteristik dari sifat memristansi ini dikontrol dengan mengatur aliran atom-atom oksigen pada lapisan TiO_2 (lapisan yang berwarna coklat).

Jika tegangan positif diberikan pada lapisan bagian atas dari divais ini, akan menyebabkan daerah yang saat ini berpolaritas positif mendorong kekosongan oksigen pada lapisan TiO_{2-x} menuju lapisan TiO_2 murni. Akibatnya, lapisan TiO_{2-x} akan mendorong lapisan TiO_2 yang ada dibawahnya sekaligus mengubahnya menjadi lapisan TiO_{2-x} yang bersifat konduktif, sehingga lapisan TiO_{2-x} akan bertambah tebal, mengkibatkan batas antara TiO_{2-x} dan TiO_2 bergerak ke bawah dan membuat divais berada pada kondisi “on”.

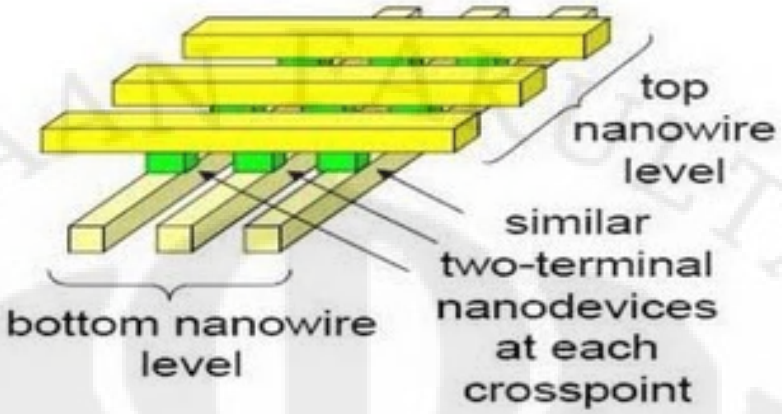
Saat tegangan negatif diberikan pada lapisan bagian atas, efek yang dihasilkan akan berlawanan dengan sebelumnya yaitu kekosongan oksigen pada layer TiO_{2-x} (yang bermuatan positif) menjadi tertarik ke atas. Akibatnya, kekosongan oksigen pada lapisan TiO_{2-x} yang berada pada bagian paling bawah (paling dekat dengan batas) akan terisi kembali oleh atom oksigen sehingga kembali menjadi lapisan TiO_2 yang murni. Dengan begitu, ketebalan lapisan TiO_2 kembali meningkat dan menyebabkan divais berada pada kondisi “off”.

Divais ini menjadi sangat menarik karena ketika tegangan listrik dimatikan, dalam kondisi positif atau negatif, atom-atom oksigennya tidak akan berpindah, sehingga menyebabkan batas antara kedua lapisan TiO_2 tersebut menjadi beku atau tidak bergerak. Fenomena inilah yang membuat memristor dapat mengingat berapa besar dan berapa lama tegangan yang telah diberikan atau mengingat berapa besar total nilai resistansi di dalam memristor pada saat terakhir kali diberikan tegangan. **Gambar 2.9, Gambar 2.10** dan **Gambar 2.11** adalah struktur dari memristor.



Gambar 2.9. *Crossbar memristor* [2]

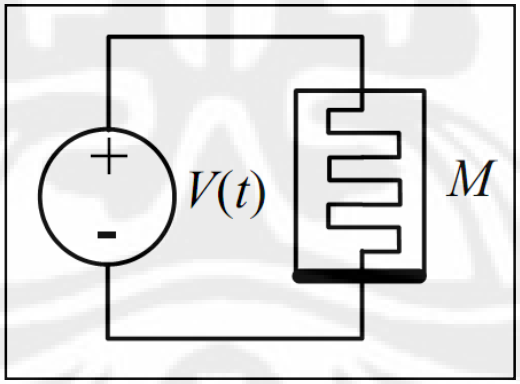
Gambar 2.10. Penampang *crossbar memristor* [10]



Gambar 2.11. Struktur memristor dalam *nanowire* [10]

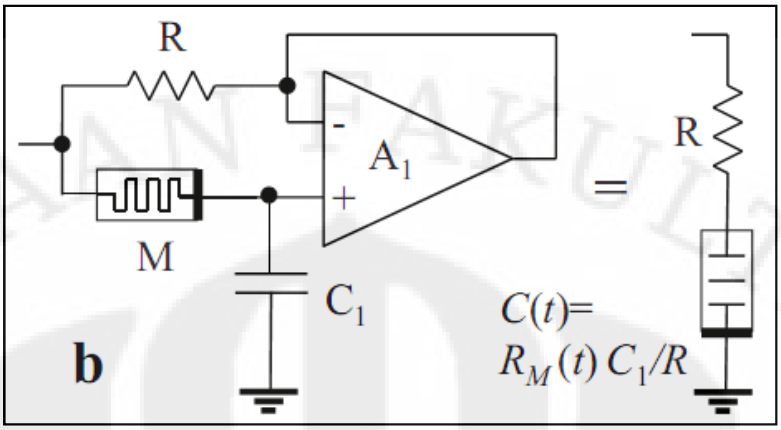
2.7 Implementasi Memristor untuk Mensimulasikan Memkapasitor

Pada Subbab 2.6 telah dijelaskan struktur fisik dari memristor. Dalam suatu publikasi ilmiah [11], dua orang peneliti berhasil mensimulasikan memristor untuk beroperasi sebagai memkapasitor dan meminduktor. Mereka menggunakan suatu rangkaian yang bekerja sebagai memristor dan memodifikasinya agar dapat bekerja sebagai memkapasitor dan meminduktor. Rangkaian memristor ditunjukkan pada **Gambar 2.12**.



Gambar 2.12. Rangkaian memristor [4]

Dari rangkaian pada **Gambar 2.12**, dibuat simulasi memkapasitor dengan memodifikasi rangkaian menjadi seperti ditunjukkan pada **Gambar 2.13**.



Gambar 2.13. Rangkaian pengganti memkapasitor [11]

Dari **Gambar 2.13**, dapat diketahui hubungan antara memristansi [11], $R_M(t)$ dengan kapasitansi, C(t) seperti yang ditulis pada Persamaan (2.28), Persamaan (2.29), dan Persamaan (2.30) [11].

$$RC(t) = R_M(t)C_1 \quad (2.28)$$

$$C(t) = R_M(t)C_1/R = (V_{in} - V_-)/(RdV_-/dt) \quad (2.29)$$

$$R_M(t)= (V_{in} - V_-)/I =(V_{in} - V_-)/ (C_1dV_-/dt) \quad (2.30)$$

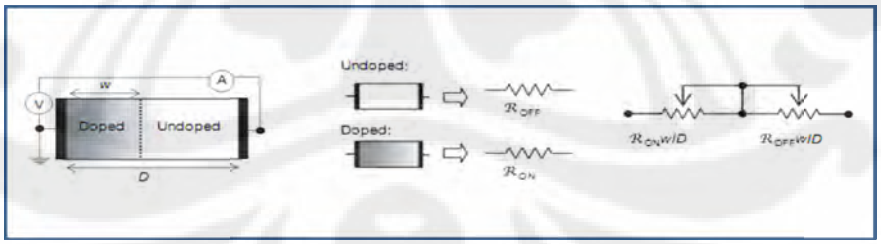
V_{in} dan V_- adalah tegangan pada kedua input *operational amplifier*.

BAB 3
SIMULASI

3.1 Simulasi Memristor

Seperti telah disebutkan pada Bab 2, bahwa simulasi memkapasitor dapat dibuat dengan memodifikasi rangkaian memristor yang sudah ada [11]. Untuk itu, perlu dijelaskan terlebih dahulu rangkaian dari memristor ini sendiri. Sebuah memristor arus terkontrol dinyatakan oleh Persamaan (2.15). Dari Persamaan (2.15), x adalah suatu fungsi *state variable* yang melibatkan arus I dan waktu t, yang didefinisikan oleh : $\dot{x} = f(x, I, t)$. Pada Bab 2, telah dijelaskan tentang cara kerja dari memristor TiO_2, yang bekerja berdasarkan kondisi *on-off* -nya. Secara skematik, ditunjukkan oleh **Gambar 3.1.**

Dari **Gambar 3.1**, dapat dilihat bahwa pada saat memristor diberi tegangan negatif, ia akan ter-*doping* rendah (*undoped*). Saat ketebalan lapisan TiO_2 yang bersifat isolator meningkat, memristor akan berada pada kondisi "*off*". Sebaliknya, saat memristor diberi tegangan positif, ia akan ter-*doping* tinggi (*doped*), yaitu saat ketebalan TiO_{2-x} meningkat, maka memristor berada pada kondisi "*on*". Memristansi (resistansi memristor) dari memristor didapat dari total resistansi saat keadaan "*on*" dan "*off*". Dari **Gambar 3.1**, didapat Persamaan memristansi [12].



Gambar 3.1. Skematik Rangkaian Memristor [2]

$$R_{mem}(x) = R_{on}.x + R_{off}.(1 - x) \quad (3.1)$$

$$x = w(t)/D \quad (3.2)$$

x adalah lebar dari daerah yang terdoping yang juga merupakan fungsi muatan (pada memristor) terhadap waktu, D adalah ketebalan dari dua lapisan TiO_2 dan

TiO_{2-x}, dan w(t) sendiri merupakan fungsi diferensial yang dinyatakan oleh Persamaan (3.3) dan Persamaan (3.4) [13] :

$$\frac{dw(t)}{dt} = \mu_v \frac{Ron}{D} i(t) \tag{3.3}$$

sehingga,

$$w(t) = \mu_v \frac{Ron}{D} q(t) \tag{3.4}$$

Persamaan (3.4) berlaku untuk w(t) pada interval [0,D]. μ_v di sini merupakan rata-rata dari mobilitas ion yang terdapat di dalam memristor. Dari Persamaan (3.1) hingga Persamaan (3.4), telah didapat hubungan antara variabel-variabel x, I, dan t. Sekarang, substitusikan Persamaan (3.1) dan Persamaan (3.2) ke dalam Persamaan (2.15), sehingga menjadi Persamaan (3.5) [13]:

$$v(t) = \left(R_{on} \frac{w(t)}{D} + R_{off} \left(1 - \frac{w(t)}{D} \right) \right) i(t) \tag{3.5}$$

jika Persamaan (4.4) disubstitusikan ke dalamnya, dan arus diintegralkan, maka akan menghasilkan persamaan karakteristik untuk memristansi M(q), saat nilai $R_{on} << R_{off}$. Hasil akhir dari tersebut ditunjukkan pada Persamaan (3.6) [13] :

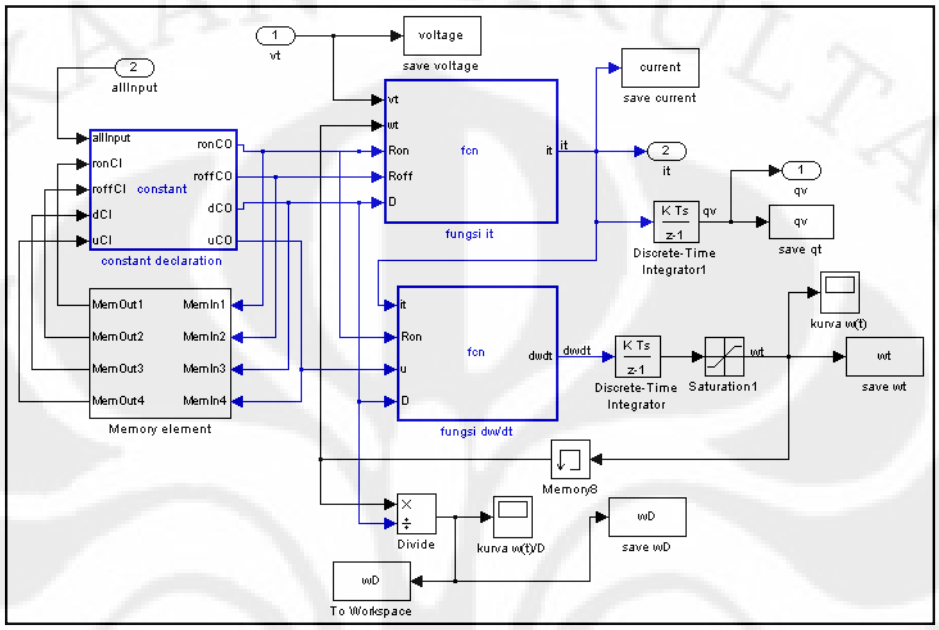
$$M(q) = R_{off} \left(1 - \frac{{}_v R_{on}}{D} q(t) \right) \tag{3.6}$$

untuk w(t) pada interval [a,D], dimana a merupakan lebar minimal dari R_{on}.
Dari Persamaan (3.5), nilai i(t) didapat dengan mengubahnya menjadi
Persamaan (3.7) [13]:

$$i(t) = \frac{v(t)}{\left(R_{on} \frac{w(t)}{D} + R_{off} \left(1 - \frac{w(t)}{D} \right) \right)} \tag{3.7}$$

Persamaan (3.1) sampai Persamaan (3.7) diperlukan untuk membuat suatu *memristor emulator*, yaitu rangkaian elektronika yang bekerja dengan keluaran berupa *memristive behavior* dan diimplementasikan dalam kurva V-I Memristor berupa *pinched hysteresis loop* seperti pada **Gambar 3.3.** Simulasi dibuat dengan menggunakan Simulink dan diintegrasikan dengan GUIDE dari MATLAB. Diagram blok untuk memristor emulator ditunjukkan oleh **Gambar 3.2.** Blok memristor emulator ini nantinya akan digunakan pada simulasi memkapasitor

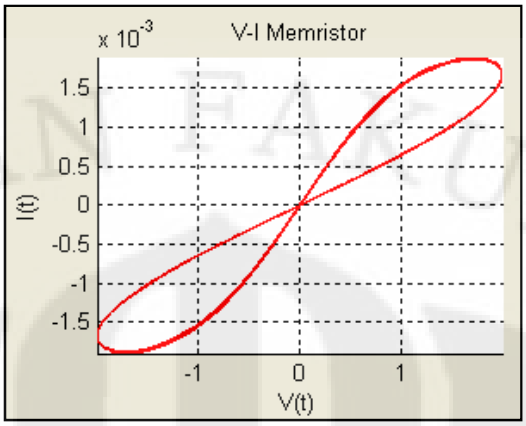
untuk menghasilkan kurva Q-V Memkapasitor yang juga berupa *pinched hysteresis loop.*



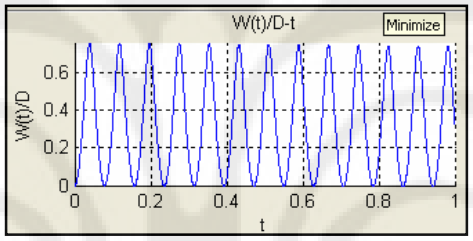
Gambar 3.2. Diagram blok *Memristor emulator*

Gambar 3.3 dan **Gambar 3.4** adalah kurva V-I Memristor dan kurva w(t)/D yang dihasilkan oleh *memristor emulator* dan. Kurva ini terjadi pada saat nilai variabel-variabel yang dimasukkan adalah : Ron/Roff = 100, A = 2, f = 80 Hz, D = 1.10^{-8}, μ_v = 1.10^{-14}, dan t = 1 detik. Dengan diberikannya tegangan, tercipta muatan (q) pada divais, dan menyebabkan nilai w(t)/D berubah-ubah diantara 0 ~ 1 seperti pada **Gambar 3.3**. Karena nilai w(t)/D yang tidak pernah mencapai 1, M(q) tidak pernah mencapai R_{on} atau R_{off}. Untuk mendapatkan bentuk kurva seperti pada **Gambar 3.4**, nilai perbandingan amplitudo (A) dan frekuensi (f) pada sumber tegangan berupa A.sin2πft harus memenuhi syarat [14]:

$$D^2 \frac{\left(2R_{off} + (R_{on} - R_{off})\right)}{2} > \frac{A}{f} \qquad (4.8)$$



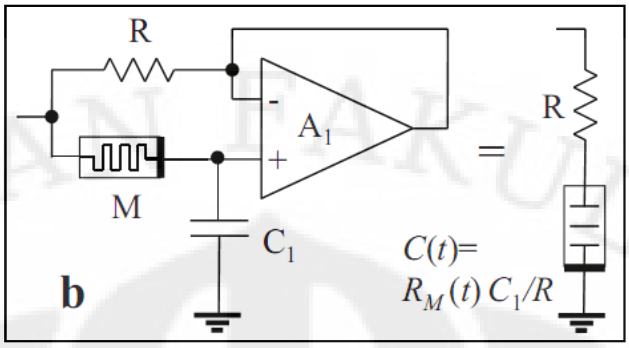
Gambar 3.3. Kurva V-I Memristor



Gambar 3.4. Kurva w(t)/D

3.2 Simulasi Memkapasitor

Untuk membuat simulasi memkapasitor, digunakan *memristor emulator* dan sedikit memodifikasinya sehingga menghasilkan simulasi yang menunjukkan respon sistem memkapasitif. Pada Bab 2, telah dijelaskan bahwa untuk mensimulasikan memkapasitor, digunakan rangkaian dengan menggunakan beberapa komponen, yaitu *op-amp*, sumber tegangan dan resistor. Secara keseluruhan, rangkaian tersebut dapat dilihat pada **Gambar 3.5.** Terminal negatif memristor M dihubungkan dengan input positif dari *op-amp* A_1 dan di-paralel-kan dengan kapasitor C_1 yang terhubung pada *ground*, sedangkan terminal positifnya dihubungkan dengan resistor R yang terhubung pula dengan input negatif dari *op-amp*. Tegangan pada *op-amp* adalah sama untuk input positif dan negatifnya, sehingga tegangan pada kapasitor C_1 akan sama dengan tegangan pada terminal di sebelah kanan dari R.



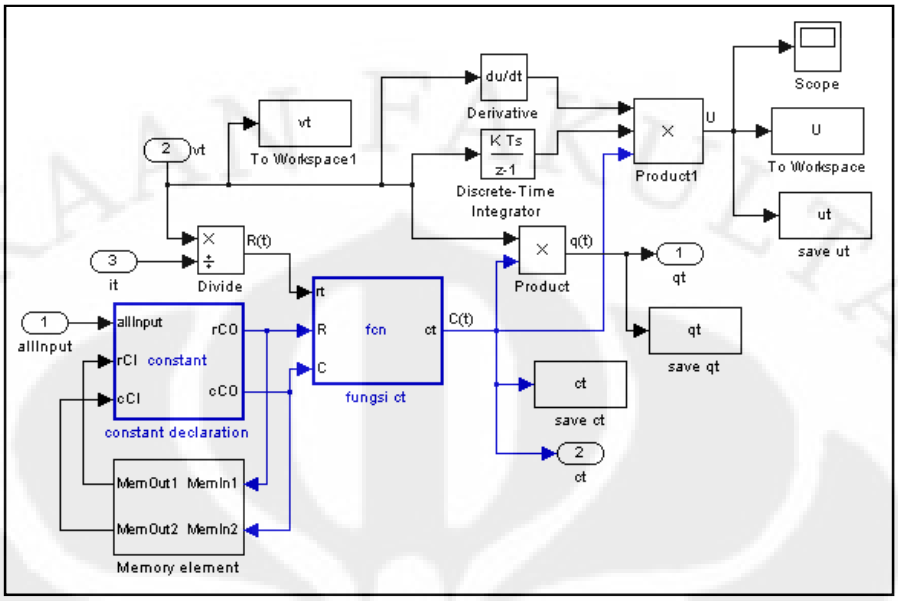
Gambar 3.5. Rangkaian pengganti memkapasitor [11]

Kemudian, anggap bahwa terdapat kapasitor efektif dengan kapasitansi yang bergantung pada waktu, C(t), terhubung dengan terminal kanan dari R, sehingga didapat hubungan $RC(t) = R_M(t)C_1$ atau $C(t) = R_M(t)C_1/R$, dengan batas $R << R_M$. Dengan mensubstitusi persamaan C(t) ke dalam Persamaan (2.21), maka akan didapat Persamaan (3.8) :

$$q(t) = \frac{(R_M(t))C_1}{R} V_C(t) \Rightarrow q(t) = \frac{\left(Ron\frac{w(t)}{D} + Roff\,(1-\frac{w(t)}{D})\right)C_1}{R} V_C(t)$$

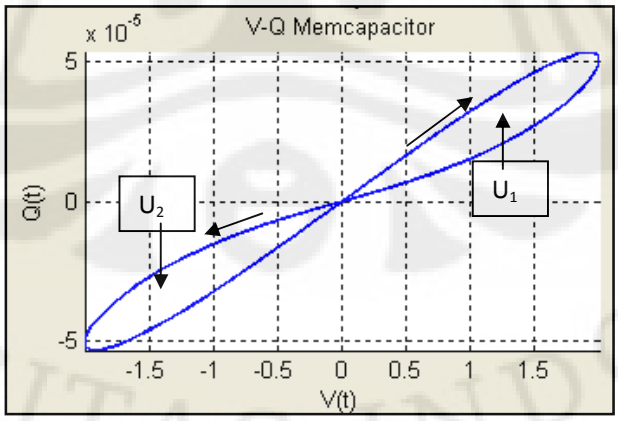
$$q(t) = \frac{\left(Roff + (Ron - Roff)\frac{w(t)}{D})\right)C_1}{R} A\sin 2 \quad ft \qquad (3.8)$$

dimana A adalah amplitudo tegangan sumber, f adalah frekuensi (Hz), dan t adalah waktu (s). Hubungan antara w(t)/D dengan frekuensi adalah berbanding terbalik. Diagram blok *memcapacitor emulator* ditunjukkan oleh **Gambar 3.6**.



Gambar 3.6. Diagram blok *memcapacitor emulator*

Gambar 3.7 adalah salah satu contoh keluaran *memcapacitor emulator*, berupa kurva *V-Q Memcapacitor*, yang berbentuk *pinched hysteresis loop*. Berbeda dengan kurva V-I Memristor, pada kurva *V-Q Memcapacitor* arah gerak kurva adalah searah jarum jam untuk nilai V-Q positif dan berlawanan arah jarum jam untuk nilai V-Q negatif, sedangkan pada kurva V-I Memristor, terjadi sebaliknya. Perbedaan ini disebabkan oleh penggunaan kapasitor C dalam *memcapacitor emulator*, yang mempengaruhi nilai muatan dalam memkapasitor Q(t), sehingga arah gerak kurva *V-Q Memcapacitor* berlawanan dengan V-I Memristor.

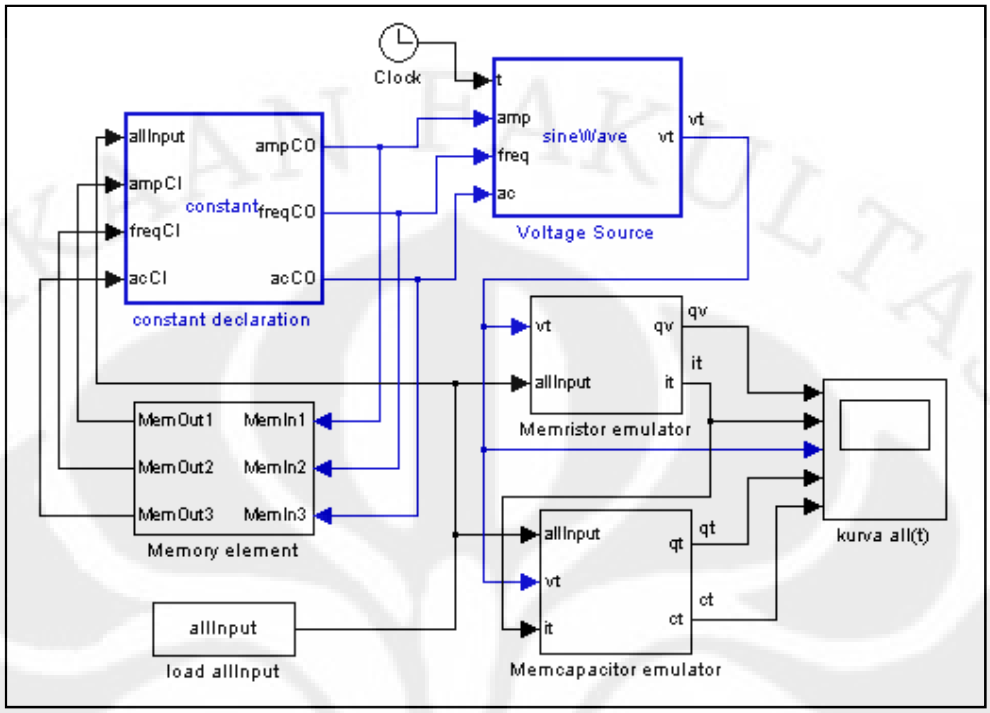


Gambar 3.7. Kurva V-Q Memcapacitor

3.3 Penjelasan Program Simulasi

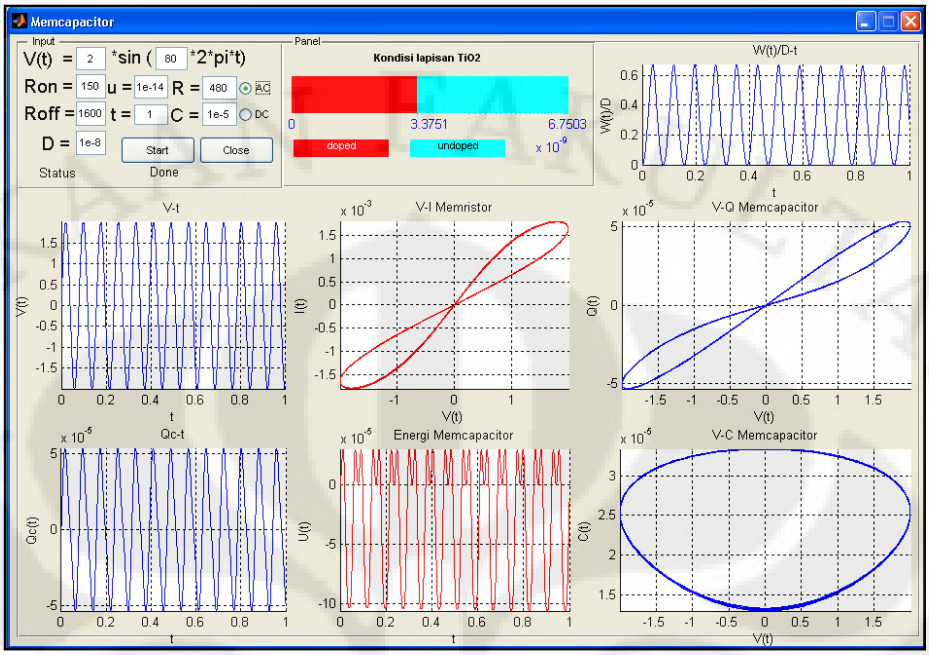
Blok diagram keseluruhan dapat dilihat pada **Gambar 3.8.** Blok diagram ini dibagi menjadi lima bagian, yaitu blok konstanta, blok memori, blok sumber tegangan, blok *memristor emulator*, dan blok *memcapacitor emulator*. Khusus untuk blok konstanta dan blok memori, berjumlah tiga pasang. Pertama pada diagram keseluruhan seperti terlihat pada **Gambar 3.8,** yang menerima masukan berupa amplitudo dan frekuensi, kedua pada blok *memristor emulator*, yang menerima masukan berupa R_{on}, R_{off}, D (ketebalan memristor), dan μ_v (kecepatan rata-rata mobilitas ion dalam memristor), dan ketiga terdapat pada blok *memcapacitor emulator* yang menerima masukan R dan C. Blok konstanta ini menerima masukan variabel dari pengguna dan disimpan dalam dokumen allInput.mat. Sedangkan fungsi blok memori adalah untuk menyimpan sementara semua variabel saat $t = t_{0,}$ untuk kemudian dikeluarkan pada saat $t = t_1$ dan selanjutnya digunakan dalam setiap perhitungan. Penyimpanan ini terus dilakukan hingga keseluruhan waktu t_n selesai dijalankan atau hingga program selesai menjalankan semua perhitungan sampai pada t_n yang ditentukan.

Semua masukan yang disimpan dalam dokumen allInput.mat akan digunakan ketika program GUI dijalankan. Amplitudo dan Frekuensi akan masuk ke blok sumber tegangan dan digunakan untuk menghitung nilai tegangan yang masuk ke dalam divais, v(t). Kemudian, v(t) beserta R_{on}, R_{off}, D, dan μ_v akan masuk ke blok memristor emulator dan digunakan dalam persamaan yang telah dijelaskan sebelumnya untuk menghitung arus yang keluar dari memristor, i(t). Selanjutnya arus i(t) ini bersama dengan v(t) akan masuk ke blok *memcapacitor emulator* dan digunakan untuk menghitung kapasitansi memkapasitor (disebut juga memkapasitansi), C(t), serta muatan dalam memkapasitor, Q(t).



Gambar 3.8. Diagram blok keseluruhan memcapacitor emulator

Gambar 3.9 adalah tampilan dari program GUI berjudul *Memcapacitor* yang bekerja berdasarkan masukan dari pengguna. Saat tombol start ditekan, maka semua variabel masukan akan disimpan dalam file ber-ekstensi .mat (salah satu jenis file yang dapat dibaca oleh MATLAB), yaitu pada allInput.mat, dan sesaat kemudian mengeluarkannya pada *simulink* untuk menghitung semua keluaran yang akan ditampilkan pada tujuh grafik yang ada, yaitu Grafik sumber tegangan v(t), Grafik muatan dalam memkapasitor Q(t), Grafik V-I Memristor, Grafik V-Q Memkapasitor, Grafik Energi Memkapasitor, Grafik Memkapasitansi C(t) dan Grafik w(t)/D. Selain itu, ada satu panel yang terletak di tengah atas, yang menggambarkan kondisi *on-off* pada memristor berdasarkan pada proses pengisian dan pengosongan oksigen pada lapisan TiO_{2-x}. Pada bab selanjutnya, analisa akan difokuskan pada keluaran grafik V-Q Memkapasitor, dengan memerhatikan variabel-variabel yang dimasukkan.



Gambar 3.9. Tampilan program simulasi memkapasitor

BAB 4
ANALISIS FREKUENSI KERJA MEMKAPASITOR

Untuk mengamati frekuensi kerja pada Memkapasitor, dilakukan lima kali simulasi dan menganalisis hasil kurva V-Q Memkapasitor. Agar pengaruh perubahan frekuensi terlihat jelas, maka nilai variabel lain dibuat tetap. Nilai untuk R_{on} dan R_{off} adalah 100 Ω dan 1600 Ω, atau $R_{on}/R_{off} = 160$, karena menurut Persamaan (3.6), untuk mendapatkan karakteristik memristansi, maka $R_{on} << R_{off}$ harus dipenuhi [13]. Selain itu, dalam literatur lain disebutkan bahwa untuk nilai w(t) dalam kisaran [0,D], maka perbandingan antara R_{on} dengan R_{off} adalah $10^2 \sim 10^3$ [12]. Nilai Amplitudo = 1 V, D = 10 nm, $\mu_v = 1x10^{-14}$ m^2/Vs [10], R = 480 Ω, dan $C_1 = 1$ μF [11]. Sedangkan frekuensi yang diamati adalah pada kisaran 12 Hz – 360 Hz. Kisaran ini digunakan karena untuk kondisi variabel-variabel seperti diperlihatkan dalam **Tabel 4**, f = 12 Hz adalah frekuensi terendah yang dibutuhkan untuk membentuk *hysteresis loop* pada kurva V-Q Memkapasitor, dan f = 360 Hz adalah frekuensi saat memkapasitor telah berubah menjadi kapasitor biasa, yang ditunjukkan oleh kurva V-Q Memkapasitor yang berbentuk garis lurus. Variabel-variabel yang dimasukkan terlihat pada **Tabel 4.1**.

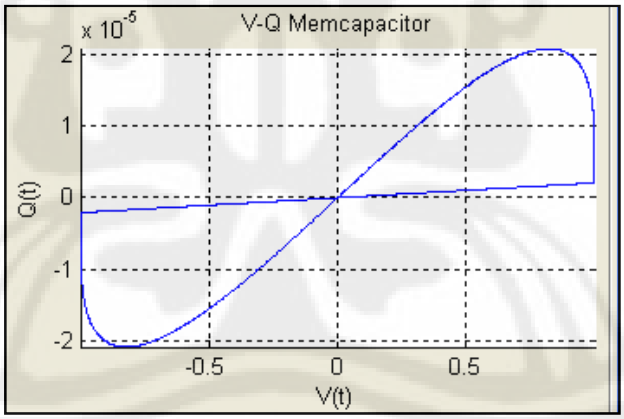
Tabel 4.1. Variabel masukan pada simulasi memkapasitor

No.	Freq	Amp	R_{on}	R_{off}	D	μ_v	R	C_1	t
1.	12	1	100	1600	1e-8	1e-14	480	1e-5	1
2.	23	1	100	1600	1e-8	1e-14	480	1e-5	1
3.	24	1	100	1600	1e-8	1e-14	480	1e-5	1
3.	230	1	100	1600	1e-8	1e-14	480	1e-5	1
4.	240	1	100	1600	1e-8	1e-14	480	1e-5	1

Setelah simulasi dijalankan, maka penjelasan kurva yang terjadi untuk masing-masing frekuensi adalah sebagai berikut :

4.1 Kurva V-Q saat frekuensi = 12 Hz

Pada saat nilai frekuensi sebesar 12 Hz, kurva V-Q Memkapasitor telah menunjukkan respon *hysteresis loop*, yang merupakan ciri dari suatu sistem yang memiliki memori, yaitu dapat menyimpan kondisi masa lalu sistem [9]. Karena nilai frekuensi tegangan input yang sangat rendah, maka sistem juga membutuhkan waktu yang agak lama untuk menyesuaikan dengan perubahan yang sangat lambat pada tegangan input [4]. Hal ini kemudian berimbas pada nilai w(t), yaitu dengan nilai frekuensi yang rendah, maka kisaran nilai w(t)/D akan maksimum, yaitu antara 0 dan 1, disebabkan hubungan keduanya yang berbanding terbalik. Dengan kisaran nilai w(t)/D yang maksimum ini, osilasi yang terjadi pada kurva V-Q juga berlangsung dengan lambat, karena sesuai dengan Persamaan (3.8), satu-satunya variabel yang mempengaruhi nilai Q(t) adalah w(t)/D, saat variabel yang lain konstan. Akibatnya, kurva V-Q Memkapasitor yang dihasilkan berbentuk *hysteresis loop* yang lebar seperti pada **Gambar 4.1**.

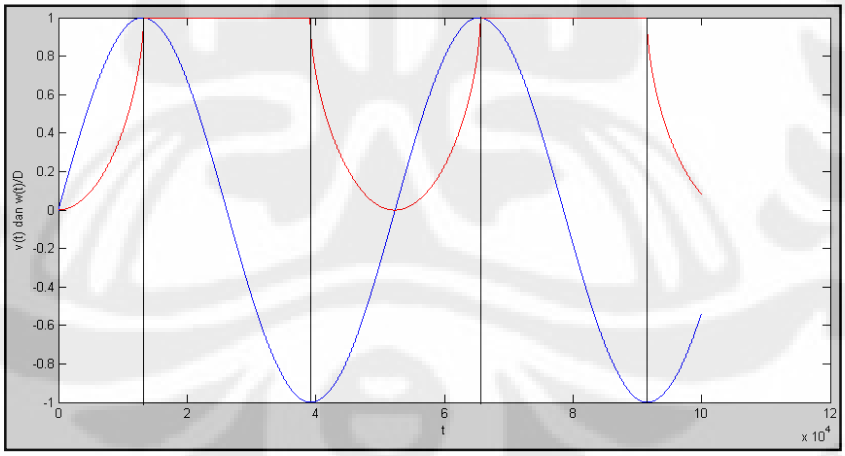


Gambar 4.1. Kurva V-Q Memkapasitor saat f = 12 Hz

Namun kurva pada **Gambar 4.1**, *hysteresis loop* yang terjadi tidak sempurna, terlihat dari masih adanya nilai yang konstan berupa garis lurus di sekitar koordinat nol sumbu Q(t) atau sumbu muatan, sepanjang nilai v(t) maksimum hingga v(t) minimum. Pada koordinat positif (kuadran 1), ketika muatan telah mencapai nilai maksimumnya, tegangan belum mencapai nilai maksimum. Dan terjadi sebaliknya pada koordinat negatif (kuadran 3), saat

muatan belum mencapai nilai minimumnya, tegangan telah mencapai nilai minimum. Jika diperhatikan saat program dijalankan, hal ini terjadi saat w(t)/D bernilai konstan pada batas maksimumnya. Jika dianalisa lebih lanjut, penyebabnya adalah Persamaan (3.4) yang hanya berlaku untuk interval w(t) pada [0,D]. Saat nilai w(t) mencapai nilai D, maka nilai w(t)/D akan terus konstan hingga polaritas tegangan v(t) berubah [14].

Hal ini dapat dilihat pada **Gambar 4.2**. Kurva tegangan v(t) ditunjukkan oleh kurva berwarna biru, kurva w(t)/D ditunjukkan oleh kurva berwarna merah, dan terdapat garis-garis bantu tegak berwarna hitam untuk memperlihatkan bahwa nilai w(t)/D konstan selama tegangan berpolaritas negatif (nilai tegangan menurun), dan segera turun bersamaan dengan polaritas tegangan yang juga berubah menjadi positif (nilai tegangan kembali meningkat). Saat nilai w(t)/D mencapai nilai 0, ia akan kembali naik sampai nilainya mencapai D, dan kembali konstan. Kondisi seperti ini akan terus terjadi selama nilai w(t) berhasil mencapai nilai D, dan menyebabkan terjadinya nilai yang konstan pada kurva w(t)/D saat tegangan berada dalam fase menurun (tegangan berubah dari nilai maksimum ke nilai minimumnya).

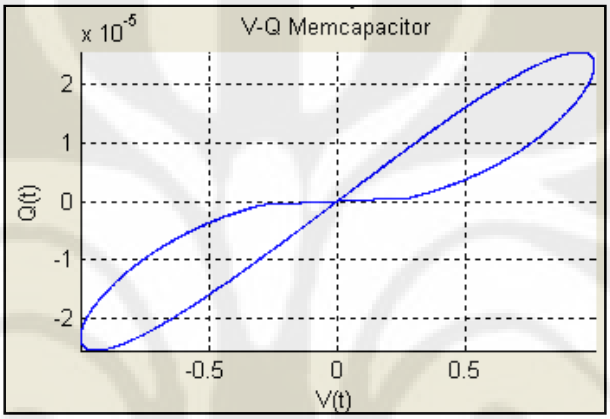


Gambar 4.2. Kurva v(t) dan w(t)/D saat f = 12 Hz

4.2 Kurva V-Q saat frekuensi = 23 Hz

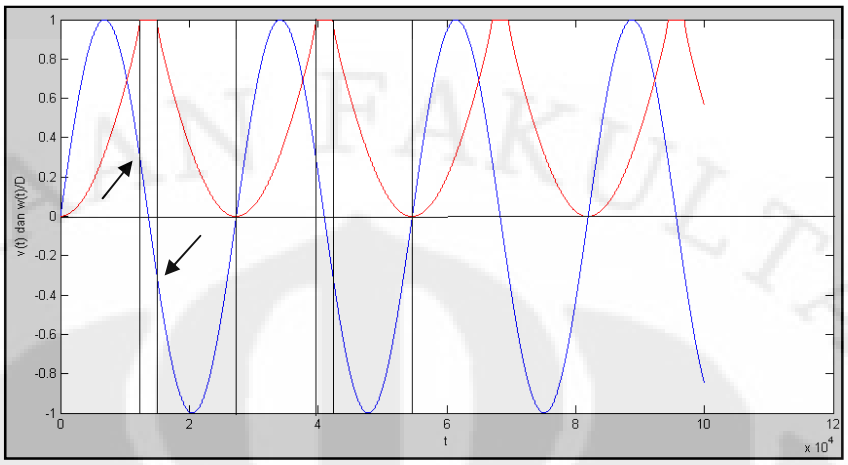
Pada saat frekuensi bernilai 23 Hz, nilai konstan pada kurva V-Q Memkapasitor yang terjadi tidak lagi sepanjang v(t) maksimum hingga v(t)

minimum, disebabkan oleh frekuensi tegangan yang diperbesar, mengakibatkan waktu saat w(t)/D bernilai konstan menjadi semakin singkat, sehingga garis lurus pada kurva V-Q Memkapasitor menjadi semakin pendek jika dibandingkan dengan kurva V-Q saat frekuensinya sebesar 12 Hz. Hal ini dapat dilihat pada **Gambar 4.3**. *Hysteresis loop* yang dihasilkan pun menjadi lebih sempit karena osilasi yang meningkat dari w(t)/D akibat diperbesarnya frekuensi tegangan.



Gambar 4.3. Kurva V-Q Memkapasitor saat f = 23 Hz

Namun jika melihat kurva v(t) dan w(t)/D terhadap waktu, ada hal yang berbeda. Nilai w(t)/D yang konstan memang terjadi saat tegangan berada pada fase menurun, namun tidak terjadi sepanjang penurunan nilai tegangan. Nilai w(t)/D mulai konstan saat tegangan akan mendekati nilai nol ketika bergerak turun dari nilai maksimumnya, dan tidak lagi konstan sekaligus menurun setelah tegangan memasuki nilai negatifnya. Hal ini ditunjukkan oleh tanda panah pada **Gambar 4.4**. Saat tegangan berubah dari nol menjadi tegangan positif, saat itu pula kurva w(t)/D bergerak naik kembali hingga mencapai nilai maksimumnya, yaitu 1, saat w(t) = D.



Gambar 4.4. Kurva v(t) dan w(t)/D saat f = 23 Hz

4.3 Kurva V-Q saat frekuensi = 24 Hz

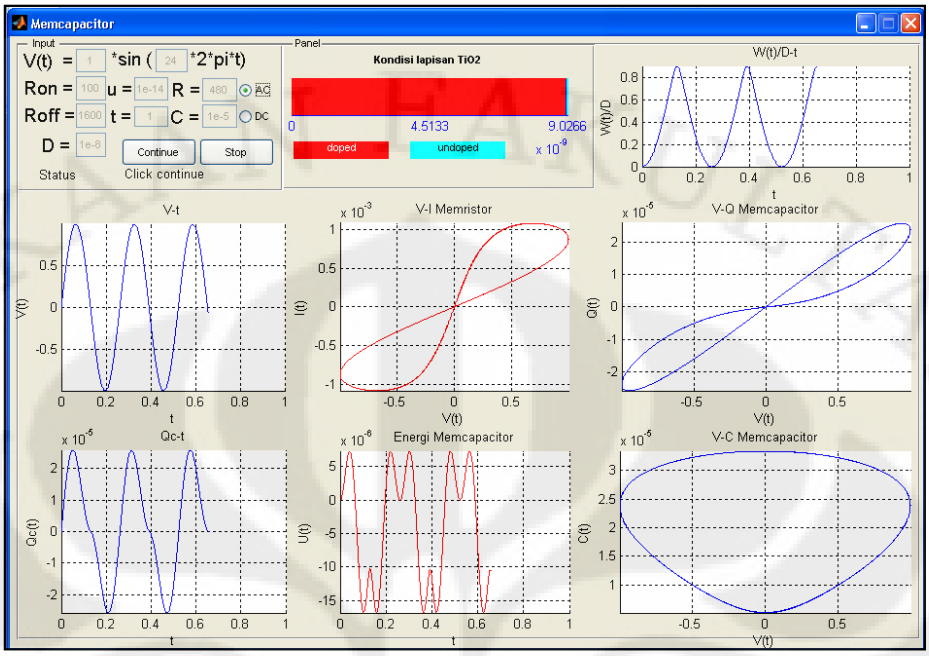
Saat frekuensi bernilai 24 Hz, kurva berbentuk menyerupai baling-baling pesawat, dan tidak ada lagi garis lurus atau nilai konstan yang terjadi di sekitar koordinat nol sumbu Q(t). Hal ini disebabkan oleh frekuensi yang sudah cukup tinggi, mengakibatkan nilai w(t)/D rendah, yaitu kurang dari 1, dan semakin cepat berosilasi. Keadaan ini merupakan keadaan ideal dari sebuah memkapasitor, karena masih berfungsi sebagai kapasitor non linier yang tidak hanya dapat menyimpan memori masa lalunya, tetapi juga energi.

Memkapasitor dikatakan dapat menyimpan memori masa lalunya karena memiliki initial state, yang dinyatakan oleh hubungan antara Q(t), C(t) dan $V_C(t)$ sesuai Persamaan (2.21). C(t) adalah *state variabel* yang didalamnya terdapat hubungan antara tegangan memkapasitor $V_C(t)$ dengan waktu, dinyatakan oleh Persamaan (2.22). Karena simulasi ini dibuat dengan menggunakan *memristor emulator*, maka tegangan yang melalui memkapasitor sama dengan tegangan yang melalui memristor, yang merupakan fungsi dari memristansi $R_M(w(t)/D)$ dan arus I(t). $R_M(w(t)/D)$ ini selalu berubah-ubah terhadap waktu sesuai dengan muatan q(t) yang melalui memristor karena w(t)/D merupakan fungsi dari muatan q(t), dan nilainya mempengaruhi tegangan yang melalui memkapasitor. Saat tegangan memristor bernilai nol, tegangan memkapasitor juga nol, akibatnya muatan dalam memkapasitor bernilai nol, seperti yang diperlihatkan pada **Gambar 4.5**.

Namun, muatan dalam memristor tidak ikut bernilai nol. Hal ini disebabkan oleh karakteristik elemen memori yang dapat mengingat kondisi masa lalunya. Artinya, saat tegangan dilepaskan dari rangkaian, muatan masih tersimpan dalam memristor, karena kondisi terakhir dari memristor dalam bentuk muatan tidak tergantung pada tegangan bias [14]. Masih adanya muatan dalam memristor diperlihatkan oleh Panel berjudul **Kondisi Lapisan TiO_2** pada **Gambar 4.5** bagian tengah atas. Panel **Kondisi Lapisan TiO_2** menunjukkan bahwa saat tegangan bernilai nol, lapisan TiO_2 masih dalam kondisi terdoping tinggi, sehingga masih menyimpan muatan.

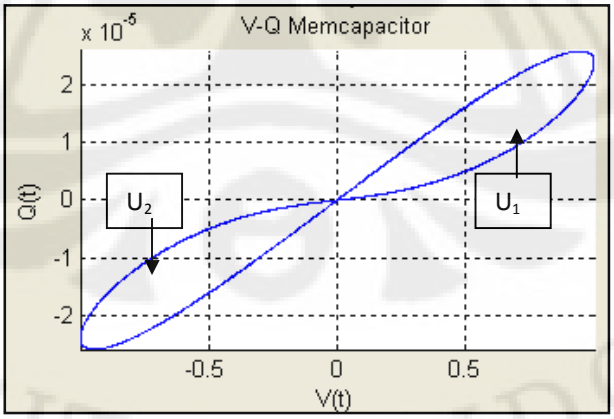
Sedangkan yang tersimpan dalam memkapasitor adalah energi. Hal ini dapat dilihat pada Grafik Energi Memkapasitor yang terletak pada **Gambar 4.5** bagian tengah bawah, energi yang tersimpan dalam memkapasitor masih ada, dan nilainya negatif, yang berarti divais sedang dalam keadaan aktif [4]. Energi yang tersimpan inilah yang membedakan memristor dan memkapasitor, yaitu memkapasitor dapat menyimpan energi, sesuai dengan Persamaan (2.27).

Muatan yang masih tersimpan dalam memristor, dan energi yang tersimpan dalam memkapasitor selain ia dapat mengingat masa lalunya, merupakan representasi nyata dari pernyataan yang menyatakan bahwa memristor dan memkapasitor merupakan elemen bermemori. Jumlah energi yang tersimpan dalam memkapasitor adalah daerah yang dibatasi oleh kurva dan sumbu Q(t). Pada **Gambar 4.6**, ditunjukkan oleh U_1 dan U_2.



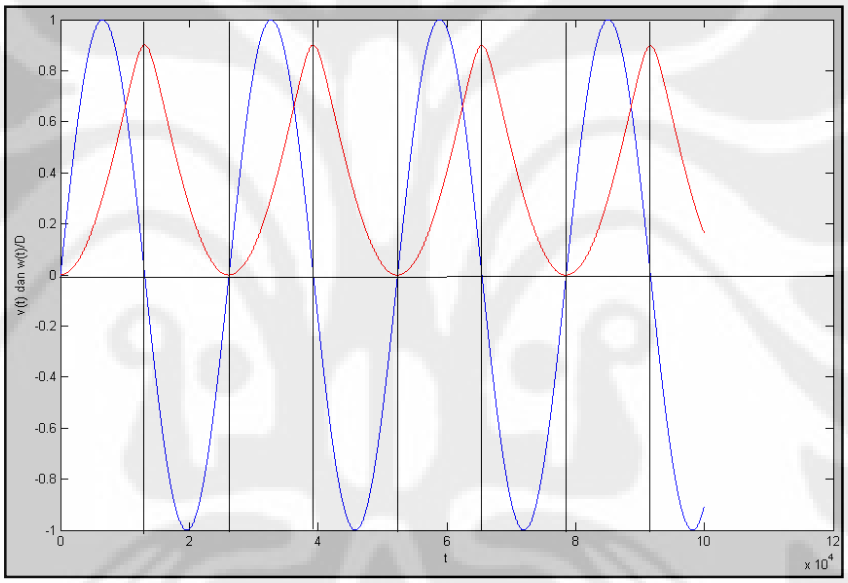
Gambar 4.5. Hasil simulasi memkapasitor saat f = 24 Hz

Untuk kurva saat f = 24 Hz, nilai w(t)/D semakin kecil, karena w(t) yang dihasilkan tidak lagi mencapai nilai D, sehingga w(t)/D juga tidak mencapai nilai maksimum 1. Karena w(t)/D tidak mencapai nilai maksimumnya, maka nilai w(t)/D yang konstan tidak lagi terjadi. Akibatnya, osilasi yang terjadi pada kurva V-Q Memkapasitor juga terjadi dengan cepat seiring dengan tingginya frekuensi tegangan input.



Gambar 4.6. Kurva V-Q memkapasitor saat f = 24 Hz

Jika diperhatikan kurva v(t) dan w(t)/D pada **Gambar 4.7**, maka terlihat bahwa saat tegangan bernilai positif, maka nilai w(t)/D berada dalam fase naik, yaitu nilai w(t)/D meningkat dari nol hingga mencapai nilai maksimumnya. Sedangkan saat tegangan bernilai negatif, nilai w(t)/D berada dalam fase turun, yaitu nilai w(t)/D berkurang dari nilai maksimumnya hingga bernilai nol. Hal ini menyebabkan memkapasitor akan berada pada kondisi *on* selama tegangan bernilai positif, dan akan kembali *off* hanya dengan tegangan negatif yang kecil [10]. Keadaan seperti ini akan terjadi selama nilai w(t)/D tidak mencapai nilai D atau w(t) < D.

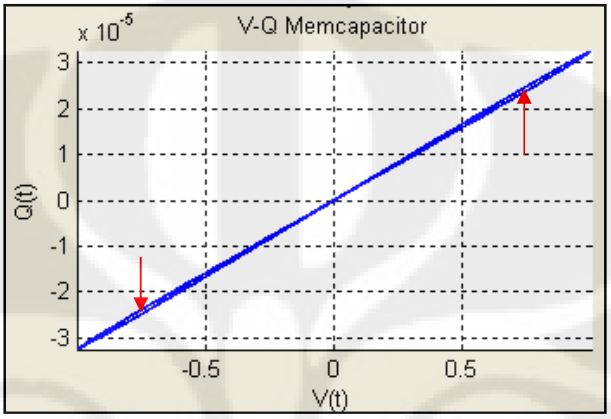


Gambar 4.7. Kurva v(t) dan w(t)/D saat f = 24 Hz

4.4 Kurva V-Q saat frekuensi = 230 Hz

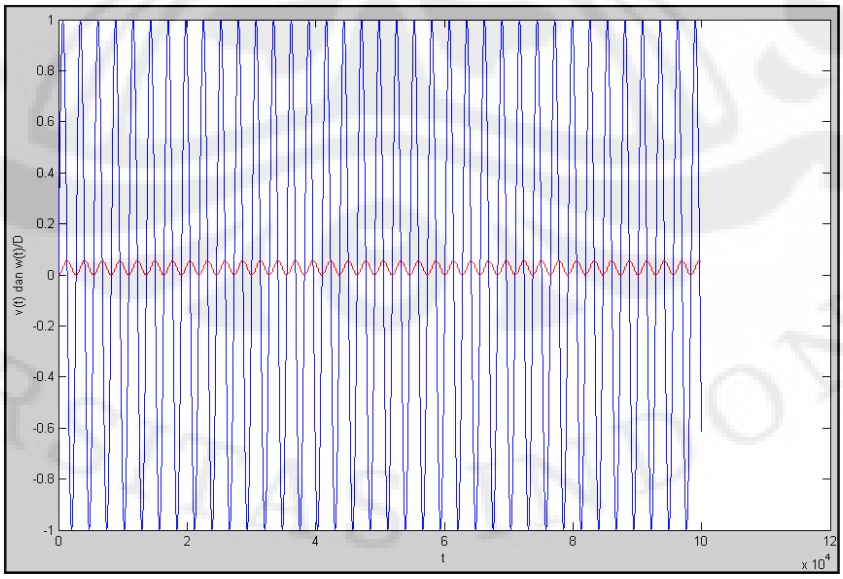
Untuk mengetahui pada batas frekuensi berapa memkapasitor masih menunjukkan *hysteresis loop*, maka frekuensi tegangan diperbesar hingga 10 kali lipat dari batas frekuensi terakhir sebelum kurva menunjukkan respon *hysteresis loop* sempurna, yaitu menjadi sebesar 230 Hz. Hasilnya, Kurva V-Q Memkapasitor semakin menyempit, karena nilai w(t)/D yang begitu cepat berosilasi, mengakibatkan nilai Q(t) juga berubah dengan cepat seiring dengan perubahan sinusoidal yang cepat pada tegangan input. Namun, kurva masih belum rapat membentuk garis lurus sempurna, melainkan masih ada celah tipis

seperti ditunjukkan oleh tanda panah merah pada **Gambar 4.8**. Pada kondisi ini, divais belum berfungsi sebagai kapasitor linier karena masih menunjukkan kurva *hysteresis loop* walaupun kurva yang terjadi sangat sempit hampir menyerupai garis lurus.



Gambar 4.8. Kurva V-Q Memcapacitor saat f = 230 Hz

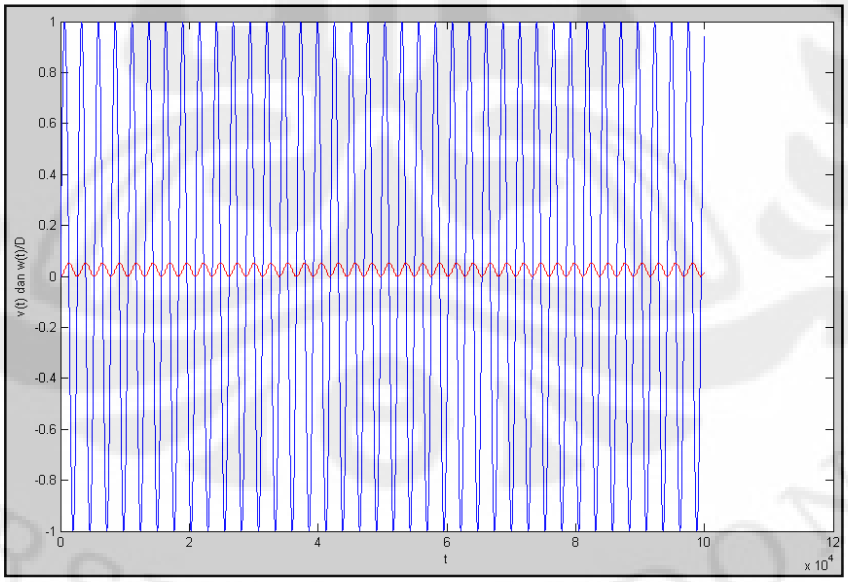
Jika diperhatikan pada kurva v(t) dan w(t)/D pada **Gambar 4.9**, keadaannya mirip dengan **Gambar 4.7**, hanya saja nilai w(t)/D menjadi jauh lebih kecil. Hal ini disebabkan oleh pemberian frekuensi tegangan yang tinggi sehingga membuat nilai w(t)/D menjadi sangat kecil dan mengakibatkan kurva V-Q Memkapasitor terlihat seperti pada **Gambar 4.8**.



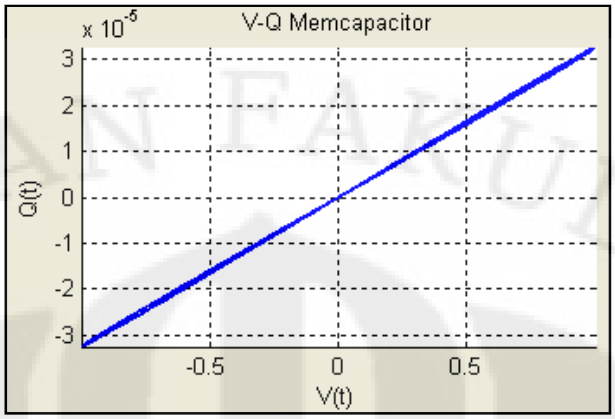
Gambar 4.9. Kurva v(t) dan w(t)/D saat f = 230 Hz

4.5 Kurva V-Q saat f = 240 Hz

Pada saat diberikan frekuensi sebesar 240 Hz, sistem tidak mampu lagi merespon osilasi yang terjadi dengan sangat cepat. Karena nilai w(t)/D mengikuti nilai tegangan, maka saat tegangan yang diberikan positif, kurva w(t)/D naik, dan saat tegangan yang diberikan berubah menjadi negatif, kurva w(t)/D juga turun. Saat frekuensi tegangan yang diberikan tinggi, maka perubahan tegangan secara sinusoidal juga terjadi dengan cepat. Itulah sebabnya nilai w(t)/D juga semakin kecil, karena tidak mampu lagi mencapai nilai maksimumnya, seperti yang terlihat pada **Gambar 4.10**. Akibatnya, kurva Q-V Memkapasitor menjadi semakin sempit serta membentuk suatu garis lurus seperti diperlihatkan pada **Gambar 4.11**. Ini berarti, divais sudah berubah menjadi kapasitor linier, karena respon yang ditunjukkan sama dengan respon Q-V pada kapasitor biasa. Jika diberikan frekuensi yang jauh lebih tinggi lagi, kurva yang terjadi akan semakin rapat dan menyerupai garis lurus sempurna.



Gambar 4.10. Kurva v(t) dan w(t)/D saat f = 240 Hz



Gambar 4.11. Kurva V-Q Memkapasitor saat f = 240 Hz

Dari penjelasan kelima simulasi diatas, untuk kondisi variabel seperti yang terdapat pada **Tabel 4**, dapat disimpulkan bahwa :

1. Saat frekuensi tegangan diantara 12 Hz hingga 23 Hz, divais dapat berfungsi sebagai *switch on-off*, karena adanya nilai yang konstan pada w(t)/D.
2. Saat frekuensi tegangan diantara 24 Hz hingga 230 Hz, divais bekerja sebagai kapasitor non linier dan berfungsi sebagai memkapasitor yang dapat menyimpan memori dan energi.
3. Saat frekuensi diatas 230 Hz, divais berubah menjadi kapasitor linier atau kapasitor biasa.

BAB 5

KESIMPULAN

1. Muatan yang tersimpan dalam memkapasitor dapat dirumuskan dengan persamaan :

$$q(t) = \frac{\left(R_{off} + \left(R_{on} - R_{off} \right) \frac{w(t)}{D} \right) C_1}{R} A \sin 2 \quad ft$$

yang bekerja selama w(t) berada dalam interval $a \leq w(t) \leq D$.

2. Saat frekuensi tegangan diantara 12 Hz hingga 23 Hz, memkapasitor dapat berfungsi sebagai *switch on-off*, karena adanya nilai yang konstan pada w(t)/D.
3. Saat frekuensi tegangan diantara 24 Hz hingga 230 Hz, memkapasitor bekerja sebagai kapasitor non linier dan dapat menyimpan memori dan energi.
4. Saat frekuensi diatas 230 Hz, memkapasitor berubah menjadi kapasitor linier atau kapasitor biasa.

DAFTAR REFERENSI

[1] L. O. Chua, "Memristor - the missing circuit element," *IEEE Trans. Circuit Theory*, vol. 18, no. 5, pp. 507–519, 1971.

[2] R.S. Williams, "How we found the missing memristor," IEEE Spectrum, 2008, December 1, pp. 1-11.

[3] Sally Adee, "*The Mysterious Memristor*", Spectrum News (2008).

[4] M. Di Ventra, Y. V. Pershin, and L. O. Chua, "Circuit elements with memory: memristors, memcapacitors and meminductors," *Proc. IEEE*, vol. 97, pp. 1717–1724, 2009.

[5] www.sukarata.com/files/komp_elka.pdf

[6] http://cnt121.com/2007/11/03/kapasitor-2/

[7] http://yb1zdx.arc.itb.ac.id/data/orari-diklat/teknik/elektronika/elektronika-dasar-I-univ-negeri-jember/bab04-kapasitor-induktor-dan-rangkaian-ac.pdf

[8]http://www.electroniclab.com/index.php?option=com_content&view=article&id=10:induktor&catid=6:elkadasar&Itemid=7

[9] http://en.wikipedia.org/wiki/Hysteresis

[10] http://electronicerror.blogspot.com/2008/11/memristor-was-first-proposed-by.html

[11] Y. V. Pershin and M. Di Ventra, "Memristive circuits simulate memcapacitors and meminductors," arXiv:0910.1583v1 [physics.ins-det] 8 Oct 2009.

[12] Dalibor Biolek, Zdenek Biolek, and Viera Biolkova, "SPICE Modeling of Memristive, Memcapacitative and Meminductive Systems," 978-1-4244-3896-9/09/$25.00 ©2009 IEEE.

[13] D.B. Strukov, G.S. Snider, D.R. Stewart and R.S. Williams, "The missing memristor found," Nature, 2008, vol. 453, pp. 80 – 83, 1 May 2008.

[14] F. Arthur M., "Konsep Dasar Memristor : Analisis Kurva I-V Titanium Dioxide (TiO_2) Memristor", Universitas Indonesia (2009).

[15] Y. V. Pershin and M. Di Ventra, "Practical approach to programmable analog circuits with memristors," arXiv:0908.3162v1 [physics.ins-det] 21 Aug 2009.

LAMPIRAN

```
function varargout = Memcapacitor(varargin)
% MEMCAPACITOR M-file for Memcapacitor.fig
%      MEMCAPACITOR, by itself, creates a new MEMCAPACITOR or
raises the existing
%      singleton*.
%
%      H = MEMCAPACITOR returns the handle to a new MEMCAPACITOR
or the handle to
%      the existing singleton*.
%
%      MEMCAPACITOR('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in MEMCAPACITOR.M with the given
input arguments.
%
%      MEMCAPACITOR('Property','Value',...) creates a new
MEMCAPACITOR or raises the
%      existing singleton*.  Starting from the left, property
value pairs are
%      applied to the GUI before Memcapacitor_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes property
application
%      stopbutton.  All inputs are passed to
Memcapacitor_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Memcapacitor

% Last Modified by GUIDE v2.5 10-Dec-2009 13:10:02

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Memcapacitor_OpeningFcn, ...
                   'gui_OutputFcn',  @Memcapacitor_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Memcapacitor is made visible.
```

```
function Memcapacitor_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Memcapacitor (see VARARGIN)

% Choose default command line output for Memcapacitor
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
% UIWAIT makes Memcapacitor wait for user response (see UIRESUME)

set(handles.pauseButton,'Visible','off','Enable','off');
set(handles.stopButton,'Visible','off','Enable','off');
set(handles.continueButton,'Visible','off','Enable','off');
set(handles.status,'String','Click start to begin');

% ============================================================
global pauseStatus axesVI lineVI axesWt lineWt axesVt lineVt
axesQt lineQt axesVC lineVC axesVq lineVq axesU lineU axesAnimWt
lineAnimWtF lineAnimWtB lineAnimWtG animWt;
%cadangan axesQV lineQV

%hold on;
pauseStatus='false';

background=get(handles.background,'Position');
animPanel=get(handles.animPanel,'Position');
animWt=get(handles.animWt,'Position');
graphPanel=get(handles.graphPanel,'Position');
%graphWtPanel=get(handles.graphWtPanel,'Position');
graphVI=get(handles.graphVI,'Position');
graphWt=get(handles.graphWt,'Position');
graphVt=get(handles.graphVt,'Position');
graphQt=get(handles.graphQt,'Position');
graphVC=get(handles.graphVC,'Position');
graphVq=get(handles.graphVq,'Position');
graphU=get(handles.graphU,'Position');
% graphQV=get(handles.graphQV,'Position');
% ============================================================

% ====================== V-I graphic =======================
graphVI([1,2])=graphVI([1,2])+graphPanel([1,2]);
gVIPosition([1,3])=graphVI([1,3])/background(3);
gVIPosition([2,4])=graphVI([2,4])/background(4);

set(handles.graphVI,'Visible','off');

axesVI=axes('Position',[ ...
    gVIPosition(1) ...
    gVIPosition(2) ...
    gVIPosition(3) ...
    gVIPosition(4) ]);
```

```
grid(axesVI);

title('V-I Memristor');xlabel('V(t)');ylabel('I(t)');
%set(gca,'Title','V-I');
lineVI=line( ...
        'color','r', ...
        'LineStyle','-', ...
        'erase','none', ...
        'xdata',0, ...
        'ydata',0);
% ========================================================

% ====================== V-C graphic =====================
graphVC([1,2])=graphVC([1,2])+graphPanel([1,2]);
gVCPosition([1,3])=graphVC([1,3])/background(3);
gVCPosition([2,4])=graphVC([2,4])/background(4);

set(handles.graphVC,'Visible','off');

axesVC=axes('Position',[ ...
    gVCPosition(1) ...
    gVCPosition(2) ...
    gVCPosition(3) ...
    gVCPosition(4) ]);

grid(axesVC);

title('V-C Memcapacitor');xlabel('V(t)');ylabel('C(t)');
%set(gca,'Title','V-C');
lineVC=line( ...
        'color','b', ...
        'LineStyle','-', ...
        'erase','none', ...
        'xdata',0, ...
        'ydata',0);
% ========================================================

% ====================== V-q graphic =====================
graphVq([1,2])=graphVq([1,2])+graphPanel([1,2]);
gVqPosition([1,3])=graphVq([1,3])/background(3);
gVqPosition([2,4])=graphVq([2,4])/background(4);

set(handles.graphVq,'Visible','off');

axesVq=axes('Position',[ ...
    gVqPosition(1) ...
    gVqPosition(2) ...
    gVqPosition(3) ...
    gVqPosition(4) ]);

grid(axesVq);

title('V-Q Memcapacitor');xlabel('V(t)');ylabel('Q(t)');
%set(gca,'Title','V-q');
lineVq=line( ...
```

```
        'color','b', ...
        'LineStyle','-', ...
        'erase','none', ...
        'xdata',0, ...
        'ydata',0);
% ========================================================

% ====================== U(t) graphic ======================
graphU([1,2])=graphU([1,2])+graphPanel([1,2]);
gUPosition([1,3])=graphU([1,3])/background(3);
gUPosition([2,4])=graphU([2,4])/background(4);

set(handles.graphU,'Visible','off');

axesU=axes('Position',[ ...
    gUPosition(1) ...
    gUPosition(2) ...
    gUPosition(3) ...
    gUPosition(4) ]);

grid(axesU);

title('Energi Memcapacitor');xlabel('t');ylabel('U(t)');
%set(gca,'Title','U');
lineU=line( ...
        'color','r', ...
        'LineStyle','-', ...
        'erase','none', ...
        'xdata',0, ...
        'ydata',0);
% ========================================================

% ====================== Q-V graphic ======================
%graphQV([1,2])=graphQV([1,2])+graphPanel([1,2]);
%gQVPosition([1,3])=graphQV([1,3])/background(3);
%gQVPosition([2,4])=graphQV([2,4])/background(4);

%set(handles.graphU,'Visible','off');

%axesQV=axes('Position',[ ...
    %gQVPosition(1) ...
    %gQVPosition(2) ...
    %gQVPosition(3) ...
    %gQVPosition(4) ]);

%grid(axesQV);

%title('V-q Memristor');xlabel('V(t)');ylabel('q(t)');
%set(gca,'Title','Q-V');
%lineQV=line( ...
        %'color','r', ...
        %'LineStyle','-', ...
        %'erase','none', ...
        %'xdata',0, ...
        %'ydata',0);
% ========================================================
```

```
% ====================== Wt/D graphic ========================
graphWt([1,2])=graphWt([1,2])+graphPanel([1,2]);
gWtPosition([1,3])=graphWt([1,3])/background(3);
gWtPosition([2,4])=graphWt([2,4])/background(4);
set(handles.graphWt,'Visible','off');
axesWt=axes('Position',[ ...
    gWtPosition(1) ...
    gWtPosition(2) ...
    gWtPosition(3) ...
    gWtPosition(4) ]);
grid(axesWt);
title('W(t)/D-t');xlabel('t');ylabel('W(t)/D');
lineWt=line( ...
        'color','b', ...
        'LineStyle','-', ...
        'erase','none', ...
        'xdata',0, ...
        'ydata',0);
% ============================================================

% ====================== Vt graphic ========================
graphVt([1,2])=graphVt([1,2])+graphPanel([1,2]);
gVtPosition([1,3])=graphVt([1,3])/background(3);
gVtPosition([2,4])=graphVt([2,4])/background(4);
set(handles.graphVt,'Visible','off');
axesVt=axes('Position',[ ...
    gVtPosition(1) ...
    gVtPosition(2) ...
    gVtPosition(3) ...
    gVtPosition(4) ]);
grid(axesVt);
title('V-t');xlabel('t');ylabel('V(t)');
lineVt=line( ...
        'color','b', ...
        'LineStyle','-', ...
        'erase','none', ...
        'xdata',0, ...
        'ydata',0);
% ============================================================

% ====================== Qct graphic ========================
graphQt([1,2])=graphQt([1,2])+graphPanel([1,2]);
gQtPosition([1,3])=graphQt([1,3])/background(3);
gQtPosition([2,4])=graphQt([2,4])/background(4);
set(handles.graphQt,'Visible','off');
axesQt=axes('Position',[ ...
    gQtPosition(1) ...
    gQtPosition(2) ...
    gQtPosition(3) ...
    gQtPosition(4) ]);
grid(axesQt);
title('Qc-t');xlabel('t');ylabel('Qc(t)');
lineQt=line( ...
        'color','b', ...
        'LineStyle','-', ...
        'erase','none', ...
        'xdata',0, ...
        'ydata',0);
```

```
% ========================================================

% ==================== Wt Animation ====================

animWt([1,2])=animWt([1,2])+animPanel([1,2]);
animWtPosition([1,3])=animWt([1,3])/background(3);
animWtPosition([2,4])=animWt([2,4])/background(4);
set(handles.animWt,'Visible','off');
axesAnimWt=axes('XColor','w','YColor','w','XTick',[],'YTick',[],'P
osition',[ ...
    animWtPosition(1) ...
    animWtPosition(2) ...
    animWtPosition(3) ...
    animWtPosition(4) ]);
% axesAnimWt2=axes('Color','b','XTick',[],'YTick',[],'Position',[
...
%     animWtPosition(1) ...
%     animWtPosition(2) ...
%     1e-100 ...
%     animWtPosition(4) ]);
lineAnimWtF=line( ...
        'color','r', ...
        'LineStyle','-', ...
        'erase','none', ...
        'xdata',0, ...
        'ydata',0);
lineAnimWtB=line( ...
        'color','b', ...
        'LineStyle','-', ...
        'erase','background', ...
        'xdata',0, ...
        'ydata',0);
lineAnimWtG=line( ...
        'color','c', ...
        'LineStyle','-', ...
        'erase','none', ...
        'xdata',0, ...
        'ydata',0);
% ========================================================

% --- Outputs from this function are returned to the command line.
function varargout = Memcapacitor_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function inputAmp_Callback(hObject, eventdata, handles)
% hObject    handle to inputAmp (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of inputAmp as
text
%        str2double(get(hObject,'String')) returns contents of
inputAmp as a double


% --- Executes during object creation, after setting all
properties.
function inputAmp_CreateFcn(hObject, eventdata, handles)
% hObject    handle to inputAmp (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function inputFreq_Callback(hObject, eventdata, handles)
% hObject    handle to inputFreq (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of inputFreq as
text
%        str2double(get(hObject,'String')) returns contents of
inputFreq as a double


% --- Executes during object creation, after setting all
properties.
function inputFreq_CreateFcn(hObject, eventdata, handles)
% hObject    handle to inputFreq (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function inputRon_Callback(hObject, eventdata, handles)
% hObject    handle to inputRon (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of inputRon as
text
%        str2double(get(hObject,'String')) returns contents of
inputRon as a double


% --- Executes during object creation, after setting all
properties.
function inputRon_CreateFcn(hObject, eventdata, handles)
% hObject    handle to inputRon (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function inputU_Callback(hObject, eventdata, handles)
% hObject    handle to inputU (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of inputU as text
%        str2double(get(hObject,'String')) returns contents of
inputU as a double


% --- Executes during object creation, after setting all
properties.
function inputU_CreateFcn(hObject, eventdata, handles)
% hObject    handle to inputU (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end



function inputT_Callback(hObject, eventdata, handles)
% hObject    handle to inputT (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of inputT as text
%        str2double(get(hObject,'String')) returns contents of
inputT as a double


% --- Executes during object creation, after setting all
properties.
function inputT_CreateFcn(hObject, eventdata, handles)
% hObject    handle to inputT (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function inputD_Callback(hObject, eventdata, handles)
% hObject    handle to inputD (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of inputD as text
%        str2double(get(hObject,'String')) returns contents of
inputD as a double


% --- Executes during object creation, after setting all
properties.
function inputD_CreateFcn(hObject, eventdata, handles)
% hObject    handle to inputD (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function inputRoff_Callback(hObject, eventdata, handles)
% hObject    handle to inputRoff (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of inputRoff as
text
%        str2double(get(hObject,'String')) returns contents of
inputRoff as a double


% --- Executes during object creation, after setting all
properties.
function inputRoff_CreateFcn(hObject, eventdata, handles)
% hObject    handle to inputRoff (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function inputR_Callback(hObject, eventdata, handles)
% hObject    handle to inputRoff (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of inputRoff as
text
%        str2double(get(hObject,'String')) returns contents of
inputRoff as a double


% --- Executes during object creation, after setting all
properties.
function inputR_CreateFcn(hObject, eventdata, handles)
% hObject    handle to inputRoff (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function inputC_Callback(hObject, eventdata, handles)
% hObject    handle to inputRoff (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of inputRoff as
text
%        str2double(get(hObject,'String')) returns contents of
inputRoff as a double


% --- Executes during object creation, after setting all
properties.
function inputC_CreateFcn(hObject, eventdata, handles)
% hObject    handle to inputRoff (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in startButton.
function startButton_Callback(hObject, eventdata, handles)
% hObject    handle to startButton (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
%clc,clear;
clc;clear amp freq Ron Roff D u r c X;

set(handles.startButton,'Visible','off','Enable','off');
set(handles.pauseButton,'Visible','on','Enable','on');
set(handles.closeButton,'Visible','off','Enable','off');
set(handles.stopButton,'Visible','on','Enable','on');

global div i wt wD vt it ct qt ut pauseStatus axesVI lineVI axesWt
lineWt axesVt lineVt axesQt lineQt axesVC lineVC axesVq lineVq
axesU lineU axesAnimWt lineAnimWtF lineAnimWtB lineAnimWtG animWt;
%axesQV lineQV
%qv

% ================== Variable declaration ====================
```

```
simModel='Memcapacitormodel';

%sampleTime=1e-5;
%timeRange=1;
%currTime=0;
% length=timeRange/sampleTime+1;

set(lineAnimWtF,'EraseMode','background');
set(lineAnimWtG,'EraseMode','background');
set(lineWt,'EraseMode','background');
set(lineVt,'EraseMode','background');
set(lineQt,'EraseMode','background');
set(lineVI,'EraseMode','background');
set(lineVC,'EraseMode','background');
set(lineVq,'EraseMode','background');
set(lineU,'EraseMode','background');
%set(lineQV,'EraseMode','background');

% amp=[0 str2num(get(handles.inputAmp,'String'))];
% freq=[0 str2num(get(handles.inputFreq,'String'))];
% Ron=[0 str2num(get(handles.inputRon,'String'))];
% Roff=[0 str2num(get(handles.inputRoff,'String'))];
% D=[0 str2num(get(handles.inputD,'String'))];
% u=[0 str2num(get(handles.inputU,'String'))];
% R=[0 str2num(get(handles.inputR,'String'))];
% C=[0 str2num(get(handles.inputC,'String'))];
% X=[0 str2num(get(handles.inputT,'String'))];

allInput=[0; ...
    str2num(get(handles.inputAmp,'String')); ...
    str2num(get(handles.inputFreq,'String')); ...
    str2num(get(handles.inputRon,'String')); ...
    str2num(get(handles.inputRoff,'String')); ...
    str2num(get(handles.inputD,'String')); ...
    str2num(get(handles.inputR,'String')); ...
    str2num(get(handles.inputC,'String')); ...
    str2num(get(handles.inputU,'String'))
    get(handles.acButton,'Value')+1];

set(handles.inputAmp,'Enable','off');
set(handles.inputFreq,'Enable','off');
set(handles.inputRon,'Enable','off');
set(handles.inputRoff,'Enable','off');
set(handles.inputD,'Enable','off');
set(handles.inputR,'Enable','off');
set(handles.inputC,'Enable','off');
set(handles.inputU,'Enable','off');
set(handles.inputT,'Enable','off');
% ======================================================



% ================================
% amp=amp';
% freq=freq';
% Ron=Ron';
% Roff=Roff';
```

```
% D=D';
% u=u';
% r=r';
% c=c';
% X=X';

% save amplitude amp;
% save frequency freq;
% save ron Ron;
% save roff Roff;
% save diameter D;
% save u u;
% save r r;
% save c c;
% save x X;

save allInput allInput;
% ================================

% ================================
set(handles.status,'String','Simulink');
sim(simModel,str2num(get(handles.inputT,'String')));

load current;
load voltage;
load wt;
load wD;
load ct;
load qt;
load ut;
%load qv;

% ================================

% ============================================
%plot(axesWt,wt(1,:),wt(2,:),'m');
%plot(axesVt,vt(1,:),vt(2,:),'r');
%plot(axesIt,it(1,:),it(2,:),'c');

%plot(axesVI,it(2,:),vt(2,:));
%plot(axesVC,ct(2,:),vt(2,:));
%plot(axesVq,qt(2,:),vt(2,:));
%plot(axesU,ut(2,:),ut(2,:));
%plot(axesQV,qv(2,:),vt(2,:));
% ============================================

% ============================================
mmW=minmax(wt);
mmD=minmax(wD);
mmV=minmax(vt);
if mmV(2,1)==mmV(2,2)
    mmV(2,1)=0;
end
mmI=minmax(it);
mmC=minmax(ct);
mmQ=minmax(qt);
mmX=minmax(ut);
```

```
%mmZ=minmax(qv);


set(axesWt,'XLim',[mmD(1,1) mmD(1,2)],'YLim',[mmD(2,1) mmD(2,2)]);
set(axesAnimWt,'XColor','b','XTick',[mmW(2,1) mmW(2,2)/2
mmW(2,2)],'XLim',[mmW(2,1) mmW(2,2)],'YLim',[0 1]);
%hold(axesAnimWt);
set(axesVt,'XLim',[mmV(1,1) mmV(1,2)],'YLim',[mmV(2,1) mmV(2,2)]);
set(axesQt,'XLim',[mmQ(1,1) mmQ(1,2)],'YLim',[mmQ(2,1) mmQ(2,2)]);

set(axesVI,'XLim',[mmV(2,1) mmV(2,2)],'YLim',[mmI(2,1) mmI(2,2)]);
set(axesVC,'XLim',[mmV(2,1) mmV(2,2)],'YLim',[mmC(2,1) mmC(2,2)]);
set(axesVq,'XLim',[mmV(2,1) mmV(2,2)],'YLim',[mmQ(2,1) mmQ(2,2)]);
set(axesU,'XLim',[mmX(1,1) mmX(1,2)],'YLim',[mmX(2,1) mmX(2,2)]);
%set(axesQV,'XLim',[mmV(2,1) mmV(2,2)],'YLim',[mmZ(2,1)
mmZ(2,2)]);

div=floor(length(wt)/2000);
set(handles.status,'String','Print graphics');
%back='false';
%first='true';
%pos=get(axesAnimWt,'position');
%pos2=get(axesAnimWt2,'position');
%hold on;
%72
%set(lineAnimWt,'LineWidth',370);
%set(lineAnimWt,'xdata',[0 0],'ydata',[1e-10 1]);
set(lineAnimWtF,'xdata',[mmW(2,1) mmW(2,1)],'ydata',[0 1]);
a=7.53;
%widthB=animWt(3)*a;
%set(lineAnimWtB,'LineWidth',widthB);
set(lineAnimWtG,'xdata',[mmW(2,2) mmW(2,2)],'ydata',[0 1]);
%set(lineAnimWtB,'Color','b');
%set(lineAnimWtF,'Clipping','off');
%reset(axesVt);
%set(lineVt,'Visible','off');

for i=1:div:length(wt)
%     xLong(1:2)=wt(2,i);
%     %xLong(1:2)=wt(2,i);

    pause(1e-5);
    %wt(1,i:i+floor(length(wt(2,:))/2000))
    %wt(1,i:i+floor(length(wt(2,:))/2000))

%set(lineWt,'xdata',wt(1,i:i+floor(length(wt(2,:))/2000)),'ydata',
wt(2,i:i+floor(length(wt(2,:))/2000)));
    if i+div<=length(wt)
        %if strcmp(back,'false')==1
        %pos(3)
        widthF=wt(2,i)*animWt(3)*a/mmW(2,2);
        widthG=(mmW(2,2)-wt(2,i))*animWt(3)*a/mmW(2,2);
        if widthF==0
            widthF=widthF+1e-5;
            %back='false';
            %set(lineAnimWtB,'Color','r');
        elseif wt(2,i)==mmW(2,2)
            %back='true';
```

```
            %set(lineAnimWtB,'Color','w');
            %set(axesAnimWt,'Color','r');
        end
        if widthG==0
            widthG=widthG+1e-5;
        end

        %if strcmp(back,'false')==1

            %set(lineAnimWtG,'xdata',[wt(2,i) wt(2,i)],'ydata',[0
1]);
            %set(lineAnimWt,'Color','r');
        %else
            %set(lineAnimWtB,'LineWidth',widthB);

            set(lineAnimWtB,'xdata',[wt(2,i) wt(2,i)],'ydata',[0
1]);
            set(lineAnimWtF,'LineWidth',widthF);
            set(lineAnimWtG,'LineWidth',widthG);
            %set(lineAnimWt,'Color','w');
        %end
%         if wt(2,i)~=mmW(2,2)
%
%         end
%
         %   set(axesAnimWt2,'position',[pos2(1) pos2(2)
(pos2(3)+wt(2,i)*pos(3)/(mmW(2,2)-mmW(2,1))) pos2(4)]);
        %else
        %    set(axesAnimWt2,'position',[pos(1) pos(2)
pos(3)+wt(2,i) pos(4)]);
        %end
        %if wt(2,i)==mmW(2,2)
        %    set(lineAnimWt,'color','w');
        %    back='true';
        %elseif wt(2,i)==mmW(2,1)
        %    set(lineAnimWt,'color','r');
        %    back='false';
        %end
%         if strcmp(pauseStatus,'true')==1
%
set(lineWt,'xdata',wt(1,1:i+div),'ydata',wt(2,1:i+div));
%             pauseStatus='false';
%         else
        set(lineWt,'xdata',[wD(1,i) wD(1,i+div)],'ydata',[wD(2,i)
wD(2,i+div)]);
        %end
        set(lineVt,'xdata',[vt(1,i) vt(1,i+div)],'ydata',[vt(2,i)
vt(2,i+div)]);
        set(lineQt,'xdata',[qt(1,i) qt(1,i+div)],'ydata',[qt(2,i)
qt(2,i+div)]);
        set(lineVI,'xdata',[vt(2,i) vt(2,i+div)],'ydata',[it(2,i)
it(2,i+div)]);
        set(lineVC,'xdata',[vt(2,i) vt(2,i+div)],'ydata',[ct(2,i)
ct(2,i+div)]);
        set(lineVq,'xdata',[vt(2,i) vt(2,i+div)],'ydata',[qt(2,i)
qt(2,i+div)]);
        set(lineU,'xdata',[ut(1,i) ut(1,i+div)],'ydata',[ut(2,i)
ut(2,i+div)]);
```

```
        %set(lineQV,'xdata',[vt(2,i) vt(2,i+div)],'ydata',[qv(2,i)
qv(2,i+div)]);
        if wt(1,i)==mmW(1)
            %set(lineAnimWtB,'EraseMode','none');
            set(lineAnimWtF,'EraseMode','none');

            set(lineAnimWtG,'EraseMode','none');
            set(lineWt,'EraseMode','none');
            set(lineVt,'EraseMode','none');
            set(lineQt,'EraseMode','none');
            set(lineVI,'EraseMode','none');
            set(lineVC,'EraseMode','none');
            set(lineVq,'EraseMode','none');
            set(lineU,'EraseMode','none');
            %set(lineQV,'EraseMode','none');
        end
%set(lineVt,'Visible','on');
        %set(lineVI,'xdata',it(2,i:i+div),'ydata',vt(2,i:i+div));
    end
end
% ================================================
set(handles.status,'String','Done');
%status='true';
set(handles.startButton,'Visible','on','Enable','on');
set(handles.pauseButton,'Visible','off','Enable','off');
set(handles.stopButton,'Visible','off','Enable','off');
set(handles.closeButton,'Visible','on','Enable','on');
set(handles.inputAmp,'Enable','on');
set(handles.inputFreq,'Enable','on');
set(handles.inputRon,'Enable','on');
set(handles.inputRoff,'Enable','on');
set(handles.inputD,'Enable','on');
set(handles.inputR,'Enable','on');
set(handles.inputC,'Enable','on');
set(handles.inputU,'Enable','on');
set(handles.inputT,'Enable','on');

% --- Executes on button press in pauseButton.
function pauseButton_Callback(hObject, eventdata, handles)
% hObject    handle to pauseButton (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
    global div i wt wD vt it ct qt ut pauseStatus axesVI lineVI
axesWt lineWt axesVt lineVt axesQt lineQt axesVC lineVC axesVq
lineVq axesU lineU axesAnimWt lineAnimWtF lineAnimWtB lineAnimWtG
animWt;
    %axesQV lineQV
    %qv
    %pauseStatus='true';
    set(handles.status,'String','Click continue');
    set(handles.pauseButton,'Visible','off','Enable','off');
    set(handles.continueButton,'Visible','on','Enable','on');

    set(lineWt,'xdata',wD(1,1:i+div),'ydata',wD(2,1:i+div));
    set(lineVt,'xdata',vt(1,1:i+div),'ydata',vt(2,1:i+div));
    set(lineQt,'xdata',qt(1,1:i+div),'ydata',qt(2,1:i+div));
    set(lineVI,'xdata',vt(2,1:i+div),'ydata',it(2,1:i+div));
```

```
    set(lineVC,'xdata',vt(2,1:i+div),'ydata',ct(2,1:i+div));
    set(lineVq,'xdata',vt(2,1:i+div),'ydata',qt(2,1:i+div));
    set(lineU,'xdata',ut(1,1:i+div),'ydata',ut(2,1:i+div));
    %set(lineQV,'xdata',vt(2,1:i+div),'ydata',qv(2,1:i+div));

    uiwait(gcf);

    set(lineWt,'xdata',wD(1,1:i+div),'ydata',wD(2,1:i+div));
    set(lineVt,'xdata',vt(1,1:i+div),'ydata',vt(2,1:i+div));
    set(lineQt,'xdata',qt(1,1:i+div),'ydata',qt(2,1:i+div));
    set(lineVI,'xdata',vt(2,1:i+div),'ydata',it(2,1:i+div));
    set(lineVC,'xdata',vt(2,1:i+div),'ydata',ct(2,1:i+div));
    set(lineVq,'xdata',vt(2,1:i+div),'ydata',qt(2,1:i+div));
    set(lineU,'xdata',ut(1,1:i+div),'ydata',ut(2,1:i+div));
    %set(lineQV,'xdata',vt(2,1:i+div),'ydata',qv(2,1:i+div));
    set(handles.pauseButton,'Visible','on','Enable','on');
    set(handles.continueButton,'Visible','off','Enable','off');
    set(handles.status,'String','Print graphics');


% --- Executes on button press in continueButton.
function continueButton_Callback(hObject, eventdata, handles)
% hObject    handle to continueButton (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
%pause off;
%pause on;
uiresume(gcf);

% --- Executes on button press in stopButton.
function stopButton_Callback(hObject, eventdata, handles)
% hObject    handle to stopButton (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
global div i wt wD vt it ct qt ut pauseStatus axesVI lineVI axesWt
lineWt axesVt lineVt axesQt lineQt axesVC lineVC axesVq lineVq
axesU lineU axesAnimWt lineAnimWtF lineAnimWtB lineAnimWtG animWt;
%axesQV lineQV
%qv

if strcmp(get(handles.pauseButton,'Visible'),'on')==1
    set(handles.pauseButton,'Visible','off','Enable','off');
else
    set(handles.continueButton,'Visible','off','Enable','off');
end
set(handles.startButton,'Visible','on','Enable','on');
set(handles.stopButton,'Visible','off','Enable','off');
set(handles.closeButton,'Visible','on','Enable','on');
    set(lineWt,'xdata',wD(1,1:i+div),'ydata',wD(2,1:i+div));
    set(lineVt,'xdata',vt(1,1:i+div),'ydata',vt(2,1:i+div));
    set(lineQt,'xdata',qt(1,1:i+div),'ydata',qt(2,1:i+div));
    set(lineVI,'xdata',vt(2,1:i+div),'ydata',it(2,1:i+div));
    set(lineVC,'xdata',vt(2,1:i+div),'ydata',ct(2,1:i+div));
    set(lineVq,'xdata',vt(2,1:i+div),'ydata',qt(2,1:i+div));
    set(lineU,'xdata',ut(1,1:i+div),'ydata',ut(2,1:i+div));
    %set(lineQV,'xdata',vt(2,1:i+div),'ydata',qv(2,1:i+div));
```

```
    set(handles.inputAmp,'Enable','on');
set(handles.inputFreq,'Enable','on');
set(handles.inputRon,'Enable','on');
set(handles.inputRoff,'Enable','on');
set(handles.inputD,'Enable','on');
set(handles.inputR,'Enable','on');
set(handles.inputC,'Enable','on');
set(handles.inputU,'Enable','on');
set(handles.inputT,'Enable','on');
uiwait(gcf);

% --- Executes on button press in closeButton.
function closeButton_Callback(hObject, eventdata, handles)
% hObject    handle to closeButton (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
close all;


% --- Executes on button press in acButton.
function acButton_Callback(hObject, eventdata, handles)
% hObject    handle to acButton (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of acButton
set(handles.dcButton,'Value',0);
set(handles.sinText,'Visible','on');
set(handles.piText,'Visible','on');
set(handles.inputFreq,'Visible','on');

% --- Executes on button press in dcButton.
function dcButton_Callback(hObject, eventdata, handles)
% hObject    handle to dcButton (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of dcButton
set(handles.acButton,'Value',0);
set(handles.sinText,'Visible','off');
set(handles.piText,'Visible','off');
set(handles.inputFreq,'Visible','off');
```