
# ESCUELA POLITÉCNICA DEL EJÉRCITO

## SEDE LATACUNGA

## CARRERA DE INGENIERÍA ELECTROMECÁNICA

### MODELACIÓN Y SIMULACIÓN DE LAS MÁQUINAS ELÉCTRICAS DE INDUCCIÓN EN CONDICIONES DINÁMICAS UTILIZANDO MATLAB

**PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTROMECÁNICA**

**JOSÉ LUIS GARCÍA MORENO**

**LATACUNGA, JULIO 2008**

# DECLARACIÓN DE AUTENTICIDAD Y RESPONSABILIDAD

Quién suscribe, José Luis García Moreno portador de la cédula de identidad No. 1715867543, libre y voluntariamente declaro que el tema de investigación “Modelación y simulación de las máquinas eléctricas de inducción en condiciones dinámicas utilizando Matlab”, su contendido, ideas, implementación, conclusiones y recomendaciones son originales.

En tal motivo son para efectos legales y académicos que se desprenden de la presente tesis, es y será de mi exclusiva responsabilidad legal y académica como autor de este proyecto de grado.

---

José Luis García Moreno

# DEDICATORIA

*A mis padres en especial a mi negra bella Rosita Piedad por su incansable e invalorable apoyo para la consecución de esta tesis, a mis hermanos Germania, Gabriel y Jorge por jamás dejarme solo, a los demás miembros que conforman mi familia por siempre darme amor para seguir adelante y a ti Natalia por apoyarme y creer en mi cada segundo de tu vida.*

*Joselo*

# AGRADECIMIENTOS

*Al Ingeniero Mario Jiménez Director de tesis, Doctor Marcelo Román Codirector de tesis, al Ingeniero Washington Freire Director de Carrera y de manera especial al Ingeniero Fabricio Perez por sus sabios conocimientos y consejos impartidos hacia mi persona y que ayudaron para la consecución de este proyecto, a la Familia Bustillos Jiménez parte imprescindible de mi vida como también lo son mis amigos en general. Gracias a Dios por permitirme vivir cada día.*

*Joselo*

# ÍNDICE DE CONTENIDOS

# 1. MODELACIÓN DE LA MÁQUINA DE INDUCCIÓN



# 2. MODELACIÓN DE LA MÁQUINA PARA LAS CONDICIONES DE OPERACIÓN

## 3. PROGRAMACIÓN



## 4. ANÁLISIS DE RESULTADOS



## 5. CONCLUSIONES Y RECOMENDACIONES

## BIBLIOGRAFÍA



## ANEXOS

# LISTADO DE FIGURAS

## CAPÍTULO I



## CAPÍTULO II

## CAPÍTULO III



## CAPÍTULO IV

## LISTADO DE TABLAS

## CAPÍTULO III

## RESUMEN

El presente proyecto permite realizar la simulación de los procesos de arranque, frenado y falla y de manera especial el determinar las curvas que se generan en estado transitorio mientras se producen estos procesos a través de

un software desarrollado en Matlab. Para poder determinar la eficacia del programa se diseño un modulo para el Laboratorio de Máquinas Eléctricas que permita la adquisición de los datos que se producen en un motor real y luego graficarlos en Lab View y de esta manera poder comparar las curvas que se dan de la simulación y la adquisición.

El contenido de la tesis “Modelación y simulación de las máquinas eléctricas de inducción en condiciones dinámicas utilizando Matlab” en su parte escrita se encuentra divida en cinco capítulos. El primer capítulo permite modelar la máquina de inducción eléctrica y determinar las ecuaciones diferenciales que rigen el comportamiento del motor eléctrico de inducción jaula de ardilla, para esto se parte de determinar las ecuaciones que rigen a la máquina en los ejes de máquina (abc), luego se determinan las ecuaciones que permiten realizar un cambio de variables abc a qd0 y viceversa, el sistema qd0, permite obtener un sistema de ecuaciones diferenciales con coeficientes constantes lo que no ocurre en el sistema de ejes abc; determinadas estas ecuaciones de transformación se procede a cambiar las ecuaciones de la máquina en ejes abc a ejes qd0. Determinadas las ecuaciones en ejes qd0 se determinan las variables de estado que son las concatenaciones de flujo por segundo y que a la final son las ecuaciones que rigen el comportamiento de la máquina en condiciones dinámicas. Finalmente se determinan las ecuaciones que rigen a la máquina en estado estable.

El segundo capítulo permite determinar todas las ecuaciones de fase que rigen la máquina en las condiciones de arranque directo en estrella y triángulo, arranque estrella – triángulo, frenado por corriente continua del motor con el estator en estrella y triángulo, frenado por inversión de fases del motor con el estator en estrella y triángulo, falla de una fase a tierra con el estator en estrella y triángulo, falla de dos fases a tierra con el estator en estrella y triángulo y falla trifásica del motor con el estator en estrella y triángulo, también se ha incluido las ecuaciones que rigen la variación de carga en función de la velocidad para distintos tipos de carga. Las ecuaciones que rigen todos estos funcionamientos son parte de la programación estructurada en Matlab.

El capítulo tercero explica toda la programación que se realiza en los programas tanto en Matlab como en Lab View, determinándose los pasos más importantes de la programación en Matlab puesto que toda las líneas de programación se encuentran en la parte de anexos, parte de este capítulo es también como se determinan los parámetros de la máquina que son función de las pruebas de vacío y rotor bloqueado.

El capítulo cuarto es muy importante pues en él se determina si el software de simulación es eficaz al comparar las curvas de la simulación del arranque e inversión de fases con las obtenidas de la adquisición de datos, comprobado la eficacia del mencionado software de simulación la restante parte del capítulo será para realizar simulaciones utilizando todas las herramientas que posee el software y determinando con su respectivo análisis lo que ocurre en cada simulación. El quinto capítulo son las conclusiones y recomendaciones desprendidas de la realización de este proyecto de tesis.

# CAPÍTULO I

# 1. MODELACIÓN DE LA MÁQUINA DE INDUCCIÓN

## 1.1.- INTRODUCCIÓN

Se ha comprobado que la máquina eléctrica de inducción (Motor-Generador y viceversa) en su versión como motor es parte esencial de todos los procesos industriales, a tal punto que estas máquinas dentro de la industria consumen más de la mitad de la energía eléctrica que generan las centrales eléctricas.

Por la importancia que representan las máquinas de inducción se necesita del estudio completo de las mismas teniendo en cuenta: principios básicos de operación, condiciones de operación, análisis de estado permanente, selección de protecciones, análisis de estados transitorios, etc. El objetivo de esta tesis es analizar, modelar y simular los periodos transitorios mediante la utilización de una herramienta matemática computacional como la es Matlab.

Para analizar estos periodos transitorios es necesario tener un cálculo aproximado o real de los valores de corriente y duración de estos periodos, por lo que es necesario recurrir a modelos matemáticos de la máquina que permitan evaluar estas respuestas transitorias para los diferentes modos de operación como son: arranque, frenado y falla. Dentro de estos procesos transitorios las inductancias varían considerablemente debido a que las

corrientes en estos procesos aumentan y estas producen grandes concatenaciones de flujos, las cuales influyen en las inductancias. Entonces, si para el estudio de las máquinas de inducción con métodos anteriores se asumía que las inductancias son constantes ahora esto queda descartado ya que con este análisis las inductancias llegan a ser variables en el tiempo y función de las corrientes de la máquina. Al tener esta condición se complica el estudio en el modelo de los ejes de la máquina lo que obliga a recurrir a un sistema de ejes arbitrarios que permitan simplificar y facilitar la resolución de este problema de inductancias variables, todo esto se analizará con detenimiento a continuación.

## 1.2.- ECUACIONES DE LA MÁQUINA EN LOS EJES (abc)[1]

Para este primer análisis se ha tomado como punto de partida el analizar a la máquina de inducción como una máquina ideal lo que implica las siguientes condiciones:[2]

- Entrehierro uniforme

- Circuito magnético lineal

- Bobinados del estator idénticos, desfasados 120˚ eléctricos, distribuidos simétricamente para generar ondas de fuerza magnetomotriz (fmm) y de densidad de flujo totalmente sinusoidales.

- Bobinados del rotor idénticos desfasados 120˚ eléctricos, distribuidos simétricamente para generar ondas de fuerza magnetomotriz (fmm) y de densidad de flujo totalmente sinusoidales.

- Histéresis, corrientes de Eddy y saturación magnética despreciables.
- Las inductancias propias y mutuas entre un par de bobinados ubicados en el mismo lado del entrehierro son constantes.

[1] Analysis of Electric Machinery. Paul C. Krause. Capítulo 4.
[2] Tesis Valdivieso Veintimilla Luis Fernando. Politécnica Nacional. Pág. 4

- Las inductancias mutuas entre un par de bobinados ubicados frente con frente del entrehierro varían con el coseno del ángulo que forman sus respectivos ejes magnéticos.

- Las resistencias de los bobinados son constantes.

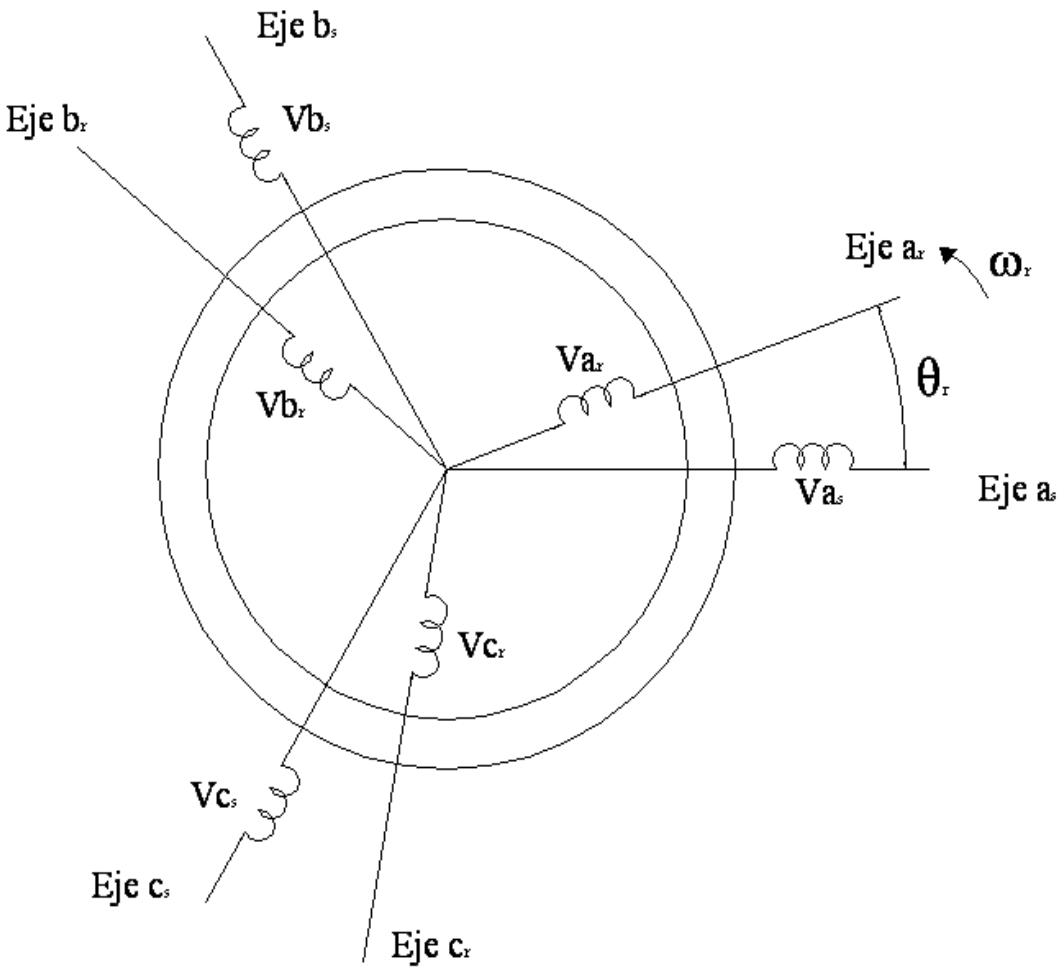



**Figura 1.1 Máquina de inducción simétrica trifásica conectada en Y**

Tomado en cuenta estas condiciones, se representa la máquina de inducción trifásica ideal conectada en Y en la figura 1.1 y su circuito eléctrico equivalente en la figura 1.2.

$$\theta_r = \int_0^t \omega_r \, dt \qquad (1)$$

Donde:

$\theta_r$ : Desplazamiento angular entre los ejes magnéticos del estator y rotor.

$\omega_r$ : Velocidad angular eléctrica con la que gira el rotor.

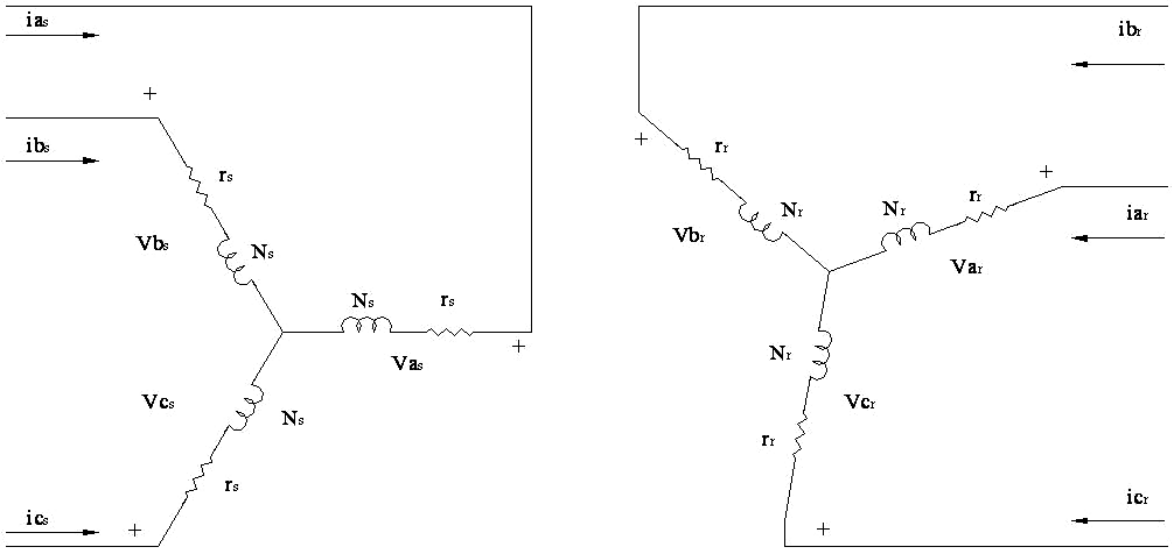



**Figura 1.2 Circuito Equivalente de la máquina de inducción simétrica conectada en Y**

Para formular las ecuaciones de voltaje se aplica la teoría de circuitos eléctricos en donde el voltaje se encuentra en función de las corrientes que la producen así:

$$[Vabc] = [r].[iabc] + \frac{d}{dt}\ [\lambda abc] \qquad (2)$$

$$\lambda = L\ .\ i \qquad (3)$$

Estas ecuaciones se aplicaran para cualquier fase ya sea esta del estator o del rotor.

Donde:

V : Voltaje en cada fase

r : Resistencia total por fase de cada bobinado

i : Corriente por fase

λ : Encadenamiento de flujo de un determinado bobinado.

L : Inductancia de un determinado bobinado.

Para mayor comodidad de ahora en adelante se reemplazara por ρ el valor de $\frac{d}{dt}$, y a todas aquellas cantidades que son del estator se denota con el subíndice “s” y las cantidades del rotor con el subíndice “r”. De esta forma las ecuaciones de voltaje en el estator (s) y en rotor (r), quedan de la siguiente manera:

$$\left.\begin{aligned} V\,abc_s &= r_s \,.\, i\,abc_s + \rho\,\lambda\,abc_s \\ V\,abc_r &= r_r \,.\, i\,abc_r + \rho\,\lambda\,abc_r \end{aligned}\right\} \qquad (4)$$

En forma matricial es:

$$\begin{bmatrix} V abc_s \\ V abc_r \end{bmatrix} = \begin{bmatrix} r_s & 0 \\ 0 & r_r \end{bmatrix} \begin{bmatrix} i abc_s \\ i abc_r \end{bmatrix} + \rho \begin{bmatrix} \lambda abc_s \\ \lambda abc_r \end{bmatrix} \qquad (5)$$

De donde cada una de las sub matrices equivalen a:

$$V abc_s = \begin{bmatrix} Va_s \\ Vb_s \\ Vc_s \end{bmatrix};$$

$$V abc_r = \begin{bmatrix} Va_r \\ Vb_r \\ Vc_r \end{bmatrix};$$

$$i abc_s = \begin{bmatrix} ia_s \\ ib_s \\ ic_s \end{bmatrix};$$

$$[abc_r] = \begin{bmatrix} ia_r \\ ib_r \\ ic_r \end{bmatrix};$$

$$[r_s] = \begin{bmatrix} r_s & 0 & 0 \\ 0 & r_s & 0 \\ 0 & 0 & r_s \end{bmatrix};$$

$$[r_r] = \begin{bmatrix} r_r & 0 & 0 \\ 0 & r_r & 0 \\ 0 & 0 & r_r \end{bmatrix};$$

$$[\lambda abc_s] = \begin{bmatrix} \lambda a_s \\ \lambda b_s \\ \lambda c_s \end{bmatrix};$$

$$[\lambda abc_r] = \begin{bmatrix} \lambda a_r \\ \lambda b_r \\ \lambda c_r \end{bmatrix};$$

Según la ecuación (3) se puede aplicar la relación siguiente para las sub matrices $[\lambda abc_s]$ y $[\lambda abc_r]$.

$$[\lambda] = [L][i] \qquad (6)$$

Donde: $[L]$ es la matriz de inductancias propias y mutuas de la máquina.

$$[L] = \begin{bmatrix} [L_s] & [L_{sr}] \\ [L_{rs}] & [L_r] \end{bmatrix} \qquad (7)$$

Donde: $[L_{rs}] = [L_{sr}]^T$

Todos los bobinados tienen una auto inductancia o inductancia propia ($L_s$) y entre bobinados de estator y rotor una inductancia mutua ($L_{sr}$), aplicando a la sub matriz de concatenaciones de flujo queda:[3]

$$\begin{bmatrix} [\lambda abc_s] \\ [\lambda abc_r] \end{bmatrix} = \begin{bmatrix} [L_s] & [L_{sr}] \\ [L_{rs}] & [L_r] \end{bmatrix} \begin{bmatrix} [abc_s] \\ [abc_r] \end{bmatrix} \qquad (8)$$

Las inductancias propias ($L_s$ *y* $L_r$) están conformadas a su vez por inductancias de dispersión o pérdida (*Ll*) y también por inductancias de magnetización (*Lm*) así:

$$L_s = Ll_s + Lm_s$$
$$L_r = Ll_r + Lm_r$$

Donde cada una de las sub matrices son ahora:

$$[L_s] = \begin{bmatrix} Ll_s + Lm_s & 0 & 0 \\ 0 & Ll_s + Lm_s & 0 \\ 0 & 0 & Ll_s + Lm_s \end{bmatrix} \qquad (9)$$

$$[L_r] = \begin{bmatrix} Ll_r + Lm_r & 0 & 0 \\ 0 & Ll_r + Lm_r & 0 \\ 0 & 0 & Ll_r + Lm_r \end{bmatrix} \qquad (10)$$

[3] Dynamic Simulation of Electric Machinery. Chee-Mun Ong. Pág. 171

Para las inductancias mutuas ($L_{sr}$ y $L_{rs}$) se asume que cualquier cambio en los devanados del rotor influye en los del estator y viceversa, por lo que estas inductancias están en función del ángulo $\theta_r$ que ya se denoto en la ecuación (1):[4]

$$[L_{sr}] = Lsr\begin{bmatrix} \cos(\theta_r) & \cos(\theta_r + 120°) & \cos(\theta_r - 120°) \\ \cos(\theta_r - 120°) & \cos(\theta_r) & \cos(\theta_r + 120°) \\ \cos(\theta_r + 120°) & \cos(\theta_r - 120°) & \cos(\theta_r) \end{bmatrix} \quad (11)$$

$$[L_{rs}] = Lsr\begin{bmatrix} \cos(\theta_r) & \cos(\theta_r - 120°) & \cos(\theta_r + 120°) \\ \cos(\theta_r + 120°) & \cos(\theta_r) & \cos(\theta_r - 120°) \\ \cos(\theta_r - 120°) & \cos(\theta_r + 120°) & \cos(\theta_r) \end{bmatrix} \quad (12)$$

Donde Lsr es el valor pico de la inductancia mutua entre estator y rotor.

Ahora que ya se ha definido todas las matrices es necesario referir todos los valores y parámetros del rotor al estator. Esta operación se la puede hacer como cuando alguna cantidad del secundario del transformador se quería referir al primario, esto se lograba con las relaciones de espira entre primario y secundario, y en el caso de la máquina de inducción es igual relacionando espiras del estator y rotor, encontrándose las siguientes relaciones:

$$V'abc_r = \frac{N_s}{N_r} \; V\,abc_r \quad (13)$$

$$\lambda'abc_r = \frac{N_s}{N_r} \; \lambda\,abc_r \quad (14)$$

$$i'abc_r = \frac{N_r}{N_s} \; i\,abc_r \quad (15)$$

[4] Dynamic Simulation of Electric Machinery. Chee-Mun Ong. Pág. 172

$$Lm_s = \frac{N_s}{N_r} \quad L_{sr} \tag{16}$$

$$Lm_r = \left(\frac{N_r}{N_s}\right)^2 \quad Lm_s \tag{17}$$

$$L'_{sr} = \frac{N_s}{N_r} \quad L_{sr} \tag{18}$$

$$L'_r = \left(\frac{N_s}{N_r}\right)^2 \quad L_r \tag{19}$$

$$L'l_r = \left(\frac{N_s}{N_r}\right)^2 \quad Ll_r \tag{20}$$

$$r'_r = \left(\frac{N_s}{N_r}\right)^2 \quad r_r \tag{21}$$

Se reemplaza la ecuación (18) en $\bar{L}_{sr}$ y luego relacionando con la ecuación (16) se obtiene:

$$\bar{L}'_{sr} = Lm_s \begin{bmatrix} \cos(\theta_r) & \cos(\theta_r + 120°) & \cos(\theta_r - 120°) \\ \cos(\theta_r - 120°) & \cos(\theta_r) & \cos(\theta_r + 120°) \\ \cos(\theta_r + 120°) & \cos(\theta_r - 120°) & \cos(\theta_r) \end{bmatrix} \tag{22}$$

Y como $\bar{L}'_{rs} = \bar{L}'^{T}_{sr}$ se obtiene:

$$\bar{L}'_{rs} = Lm_s \begin{bmatrix} \cos(\theta_r) & \cos(\theta_r - 120°) & \cos(\theta_r + 120°) \\ \cos(\theta_r + 120°) & \cos(\theta_r) & \cos(\theta_r - 120°) \\ \cos(\theta_r - 120°) & \cos(\theta_r + 120°) & \cos(\theta_r) \end{bmatrix} \tag{23}$$

Ahora para referir la sub matriz $\bar{L}_r$ a términos del estator utilizamos la ecuación (19) que se reemplaza en la ecuación (10) y mediante las ecuaciones. (17) y (20) se obtiene los reemplazos quedando:

$$[L'_r] = \begin{bmatrix} L'l_r + Lm_s & 0 & 0 \\ 0 & L'l_r + Lm_s & 0 \\ 0 & 0 & L'l_r + Lm_s \end{bmatrix} \quad (24)$$

De esta forma se ha referido todos los valores del rotor con respecto al estator y la ecuación general de voltajes de la máquina de inducción en los ejes abc es:

$$\begin{bmatrix} Vabc_s \\ Vabc_r \end{bmatrix} = \begin{bmatrix} r_s + \rho L_s & \rho L'_{sr} \\ \rho L'_{rs} & r'_r + \rho L'_r \end{bmatrix} \begin{bmatrix} iabc_s \\ iabc_r \end{bmatrix} \quad (25)$$

Ahora que ya se han encontrado las ecuaciones de voltaje hace falta encontrar la ecuación que rige el torque electromagnético (Te). La ecuación se obtiene a través de la energía instantánea almacenada en el campo magnético que se define por:[5]

$$W = \frac{1}{2} [i]^T [L] [i] \quad (26)$$

El torque electromagnético está definido según la siguiente expresión:

$$Te = \frac{P}{2} \frac{dW}{d\theta_r} \quad (27)$$

Donde P representa el número de polos de la máquina. Se reemplaza la ecuación (26) en la ecuación (27) quedando en forma simplificada:

$$Te = \frac{P}{2} [i]^T \left[ \frac{dL}{d\theta_r} \right] [i] \quad (28)$$

Desarrollando término a término la ecuación anterior se obtiene lo siguiente:

[5] Tesis Mancero Santillán Fernando. Politécnica Nacional. Pág. 10

$$Te = -\left(\frac{P}{2}\right) Lm_s \left\{ \begin{array}{l} \left[ ia_s\left( i'a_r - \frac{1}{2}i'b_r - \frac{1}{2}i'c_r \right) + ib_s\left( i'b_r - \frac{1}{2}i'a_r - \frac{1}{2}i'c_r \right) + ic_s\left( i'c_r - \frac{1}{2}i'b_r - \frac{1}{2}i'a_r \right) \right] \\ sen\ \theta_r + \frac{\sqrt{3}}{2} \left[ a_s(i'b_r - i'c_r) + ib_s(i'c_r - i'a_r) + ic_s(i'a_r - i'b_r) \right] \cos\ \theta_r \end{array} \right\} \quad (29)$$

Este valor es positivo para acción motora (Tercera ley de Newton acción = reacción), también este torque electromagnético (Te) se lo puede relacionar con la velocidad del rotor y se obtiene el torque electromecánico así:

$$Te = J\left(\frac{2}{P}\right)\rho\omega_r + T_L \quad (30)$$

Donde:

J : Momento de inercia del rotor.
P : Numero de polos de la máquina.
$\omega_r$ : Velocidad angular eléctrica del rotor
$T_L$ : Torque de carga.

## 1.3.- ECUACIONES DE TRANSFORMACIÓN PARA EL EJE ARBITRARIO DE REFERENCIA (qd0).

La matriz de inductancias en especial las sub matrices de inductancias mutuas están en función del ángulo $\theta_r$, es decir varían con el coseno de este ángulo, lo que dificulta notablemente la resolución de estas ecuaciones debido a que los coeficientes de las ecuaciones diferenciales de voltaje de la máquina se vuelven variantes en el tiempo. Para facilitar la resolución de estas ecuaciones se recurre a un artificio que es usar un “sistema de referencia arbitrario” que será común para el estator y el rotor. Este artificio o método lo que pretende es referir todos las variables de estator y rotor a un sistema de referencia arbitrario que gira a una velocidad angular (ω), mientras el eje del rotor sigue girando a

una velocidad angular ($\omega_r$) y la velocidad del eje del estator ($\omega_s$) esta fijo es decir $\omega_s = 0$.[6]

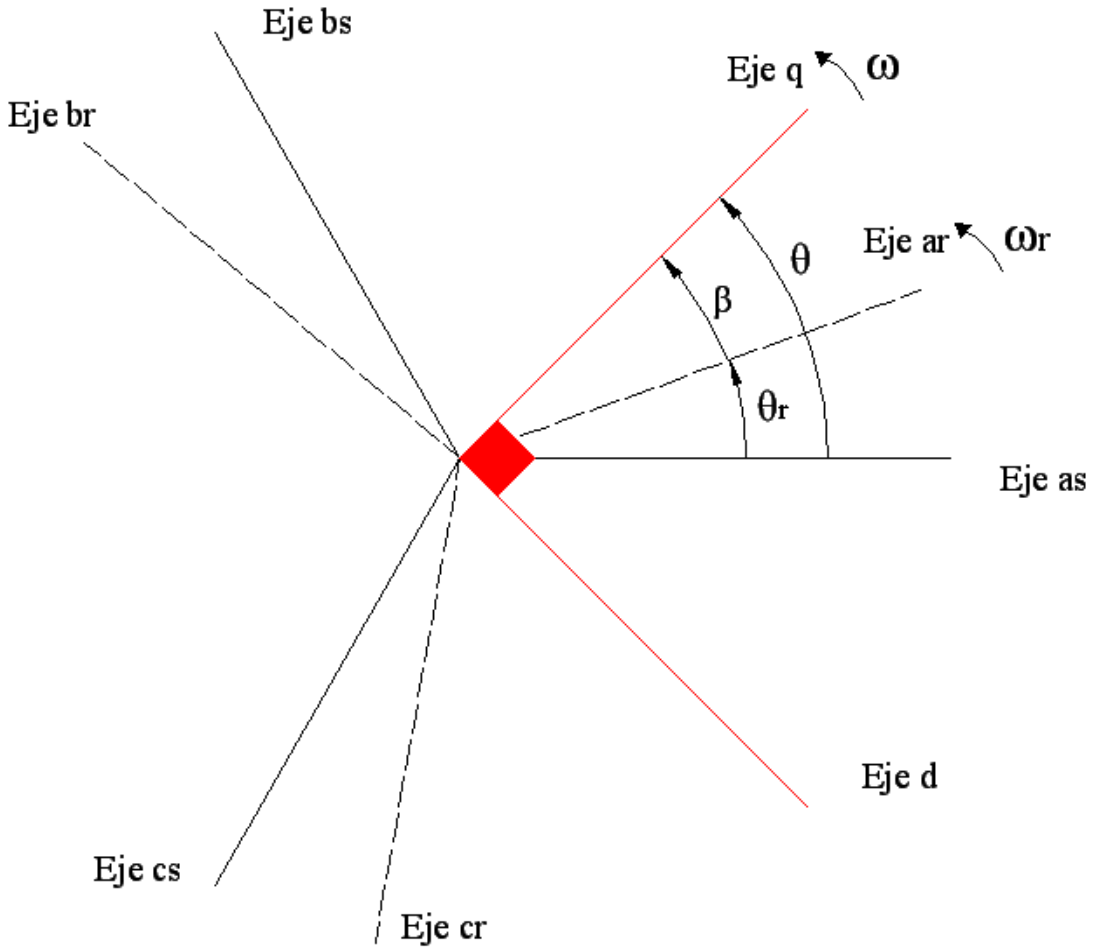



**Figura 1.3 Representación del eje de referencia arbitrario**

Las ecuaciones de transformación que llevan a este sistema de referencia arbitrario como su nombre lo indica son expresiones que permiten realizar un cambio de variables y por lo tanto pueden escribirse sin ninguna interpretación física, sin embargo para su entendimiento es necesario relacionar el cambio de variables con funciones trigonométricas que existen entre los ejes del rotor y estator con el sistema de referencia arbitrario que vale mencionar es un sistema ortogonal con un eje directo y otro en cuadratura como se ve en la figura 1.3:[7]

[6] Dynamic Simulation of Electric Machinery. Che-Mun Ong. Pág. 172
[7] Analysis of Electric Machinery. Paul C. Krause. Pág. 170

$$\theta = \int_0^t \omega \ dt$$

Donde:

θ : Desplazamiento angular entre el sistema de referencia arbitrario y el eje del estator.

ω : Velocidad angular del sistema de referencia arbitrario.

El sistema de referencia en ejes abc es un sistema trifásico mientras que el sistema de referencia en ejes qd0 es un sistema bifásico lo que implica que las tres bobinas en los ejes abc deben ser representadas por solo dos bobinas en los ejes qd. Las condiciones de análisis para encontrar estas ecuaciones de transformación son:[8]

- Las dos bobinas (q,d) deben tener el mismo número de vueltas que las bobinas trifásicas, esto para producir una onda de fmm de amplitud igual al sistema en los ejes abc.

- La corriente que circule por cada bobina (q,d) es 3/2 veces la corriente de cualquiera de las bobinas en el eje abc.

Para la formulación de estas ecuaciones se parte con un concepto general que propone lo siguiente:

$$\text{fmm} = N\, i \qquad (31)$$

A la expresión anterior se denota en forma matricial así:

$$\begin{bmatrix} fmmq_s \\ fmmd_s \end{bmatrix} = N \begin{bmatrix} \cos(\theta) & \cos(\theta - 120°) & \cos(\theta + 120°) \\ sen(\theta) & sen(\theta - 120°) & sen(\theta + 120°) \end{bmatrix} \begin{bmatrix} ia_s \\ ib_s \\ ic_s \end{bmatrix} \qquad (32)$$

[8] Tesis Mancero Santillán Fernando. Politécnica Nacional. Pág. 13.

De las condiciones anteriores se obtiene:

$$\begin{bmatrix} fmmq_s \\ \\ fmmd_s \end{bmatrix} = \frac{3}{2} N \begin{bmatrix} iq_s \\ \\ id_s \end{bmatrix} \tag{33}$$

Si se iguala la ecuación (33) con la ecuación (32), se obtiene lo siguiente:

$$\begin{bmatrix} iq_s \\ \\ id_s \end{bmatrix} = \frac{2}{3} \begin{bmatrix} \cos(\theta) & \cos(\theta - 120°) & \cos(\theta + 120°) \\ \\ sen(\theta) & sen(\theta - 120°) & sen(\theta + 120°) \end{bmatrix} \begin{bmatrix} ia_s \\ ib_s \\ ic_s \end{bmatrix} \tag{34}$$

En la ecuación (34) se ha obtenido la matriz transformación de la corriente, entonces a partir de la misma se puede generalizar para cualquier variable, así:

$$\begin{bmatrix} fq_s \\ \\ fd_s \end{bmatrix} = \frac{2}{3} \begin{bmatrix} \cos(\theta) & \cos(\theta - 120°) & \cos(\theta + 120°) \\ \\ sen(\theta) & sen(\theta - 120°) & sen(\theta + 120°) \end{bmatrix} \begin{bmatrix} fa_s \\ fb_s \\ fc_s \end{bmatrix} \tag{35}$$

Donde: f puede representar voltaje (V), corriente (i) o encadenamiento de flujo (λ).

La transformación es a un sistema de referencia arbitrario qd0 y no solo qd como se ha analizado hasta aquí, por lo que tiene gran importancia el analizar el eje 0. Este eje 0 tiene la particularidad de no estar asociado a la velocidad ω que es a la que gira el sistema arbitrario como se puede ver en la figura 1.3, y esto se debe a que las variables en este eje 0 no están asociadas a la transformación y solo se dan en el sistema donde son originadas. Para entender esto se afirma que el eje 0 únicamente aparece cuando existen condiciones asimétricas ya sea en el estator o en el rotor y de igual forma aparecerá este eje en el estator si la asimetría se produce en el estator y cosa similar si sucede en el rotor es decir una asimetría que se produce en el estator no se reproduce en el rotor y viceversa, dicho esto el eje 0 queda así:

$$f0 = \frac{1}{3}\ (fa + fb + fc)$$

Volviendo a escribir la ecuación (35) tomado en cuenta al eje 0 se tiene lo siguiente:

$$\begin{bmatrix} fq_s \\ fd_s \\ f0_s \end{bmatrix} = \underbrace{\frac{2}{3}\begin{bmatrix} \cos(\theta) & \cos(\theta-120°) & \cos(\theta+120°) \\ sen(\theta) & sen(\theta-120°) & sen(\theta+120°) \\ 1/2 & 1/2 & 1/2 \end{bmatrix}}_{[Ks]} \begin{bmatrix} fa_s \\ fb_s \\ fc_s \end{bmatrix} \qquad (36)$$

[Ks] : Matriz de transformación del estator en ejes abc a ejes qd0.

$$\begin{bmatrix} f'q_r \\ f'd_r \\ f'0_r \end{bmatrix} = \underbrace{\frac{2}{3}\begin{bmatrix} \cos(\beta) & \cos(\beta-120°) & \cos(\beta+120°) \\ sen(\beta) & sen(\beta-120°) & sen(\beta+120°) \\ 1/2 & 1/2 & 1/2 \end{bmatrix}}_{[Kr]} \begin{bmatrix} f'a_r \\ f'b_r \\ f'c_r \end{bmatrix} \qquad (37)$$

[Kr] : Matriz de transformación del rotor referido al estator en ejes abc a ejes qd0. Donde:

$$\beta = \theta - \theta_r \qquad (38)$$

En forma resumida las matrices de transformación quedan:

$$[fqd0_s] = [Ks][fabc_s] \qquad (39)$$

$$[f'qd0_r] = [Kr][f'abc_r] \qquad (40)$$

Entonces la matriz transformación general [K] será representada por:

$$[K] = \begin{bmatrix} [Ks] & [0] \\ [0] & [Kr] \end{bmatrix} \tag{41}$$

Al igual que las anteriores expresiones se puede encontrar las inversas de [Ks] y de [Kr] para encontrar la matriz transformación del estator y rotor de los ejes qd0 a los ejes abc, así:

$$[f abc_s] = [Ks^{-1}][f qd0_s] \tag{42}$$

$$[f' abc_r] = [Kr^{-1}][f' qd0_r] \tag{43}$$

Donde:

[$Ks^{-1}$] : Matriz transformación del estator en ejes qd0 a ejes abc.
[$Kr^{-1}$] : Matriz transformación del rotor referido al estator en ejes qd0 a ejes abc.

Y se expresan de la siguiente forma:

$$[Ks^{-1}] = \begin{bmatrix} \cos(\theta) & sen(\theta) & 1 \\ \cos(\theta - 120°) & sen(\theta - 120°) & 1 \\ \cos(\theta + 120°) & sen(\theta + 120°) & 1 \end{bmatrix} \tag{44}$$

$$[Kr^{-1}] = \begin{bmatrix} \cos(\beta) & sen(\beta) & 1 \\ \cos(\beta - 120°) & sen(\beta - 120°) & 1 \\ \cos(\beta + 120°) & sen(\beta + 120°) & 1 \end{bmatrix} \tag{45}$$

La matriz transformación inversa general [$K^{-1}$] queda definida por:

$$[K^{-1}] = \begin{bmatrix} [Ks^{-1}] & 0 \\ 0 & [Kr^{-1}] \end{bmatrix} \qquad (46)$$

Obtenidas las ecuaciones de transformación se procede a encontrar las ecuaciones de voltaje y torque en el eje de referencia arbitrario qd0.

## 1.4.- ECUACIONES DE LA MÁQUINA EN LOS EJES (qd0)[9, 10]

Volviendo a escribir la ecuación (2) y aplicando la ecuación de transformación general de ejes abc a qd0 que es la matriz [K], se tiene:

$$[Vabc] = [r].[iabc] + \rho[\lambda abc]$$

$$[K][Vabc] = [K][r].[iabc] + [K]\rho[\lambda abc]$$

Si a la anterior igualdad aplicamos la transformada inversa para escribirla de otra forma esta queda:

$$[K][Vabc] = [K][r].[K^{-1}][iqd0] + [K]\rho[K^{-1}][\lambda qd0]$$

$$[Vqd0] = [r].[iqd0] + \rho[\lambda qd0] + [K]\,\rho[K^{-1}]\,[\lambda qd0] \qquad (47)$$

Al resolver las matrices $[K]\,\rho[K^{-1}] = [Z]$ se obtiene:

[9] Dynamic Simulation of Electric Machinery. Che Mun Ong. Capítulo Seis.
[10] Analysis of Electric Machinery. Paul C. Krause. Capítulo Cuatro.

$$[Z] = \begin{bmatrix} 0 & \rho\theta & 0 & 0 & 0 & 0 \\ -\rho\theta & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \rho\beta & 0 & 0 \\ 0 & 0 & -\rho\beta & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (48)$$

Volviendo a la ecuación (47) ya hecho el anterior reemplazo se tiene:

$$[Vqd0] = [r].[iqd0] + \rho[\lambda qd0] + [Z]\,[\lambda qd0]$$

Ahora bien la matriz [λqd0] se obtiene de la siguiente forma:

$$[\lambda abc] = [L]\,[iabc]$$

$$[K]\,[\lambda abc] = [K]\,[L]\,[iabc]$$

$$[K]\,[\lambda abc] = [K]\,[L]\,[K^{-1}]\,[iqd0]$$

$$[\lambda qd0] = [Z']\,[iqd0]$$

Donde : [Z'] = [K] [L] [$K^{-1}$] ,y expresado matricialmente es:

$$[Z'] = \begin{bmatrix} [Ks][L_s][Ks]^{-1} & [Ks][L'_{sr}][Kr^{-1}] \\ [Kr][L'_{rs}][Ks^{-1}] & [Kr][L'_r][Kr]^{-1} \end{bmatrix} \quad (49)$$

Donde las sub matrices [Ls] y [Lr'] ya son conocidas y procedemos a encontrar el producto de estas matrices:

$$
\left.
\begin{aligned}
[Ks][L_s][Ks^{-1}] &= \begin{bmatrix} Ll_s + Lm & 0 & 0 \\ 0 & Ll_s + Lm & 0 \\ 0 & 0 & Ll_s \end{bmatrix} \\
[Kr][L'_r][Kr^{-1}] &= \begin{bmatrix} L'l_r + Lm & 0 & 0 \\ 0 & L'l_r + Lm & 0 \\ 0 & 0 & L'l_r \end{bmatrix} \\
[Ks][L'_{sr}][Kr^{-1}] = [Kr][L'_{rs}][Ks^{-1}] &= \begin{bmatrix} Lm & 0 & 0 \\ 0 & Lm & 0 \\ 0 & 0 & 0 \end{bmatrix}
\end{aligned}
\right\} \quad (50)
$$

Donde:

$$Lm = \frac{3}{2} Lm_s \quad (51)$$

Obtenido esto se regresa a la ecuación (47) continuando con la resolución.

$$[Vqd0] = [r].[iqd0] + \rho[\lambda qd0] + [Z]\,[\lambda qd0]$$

$$[Vqd0] = [r].[iqd0] + \rho[Z']\,[iqd0] + [Z]\,[Z']\,[iqd0]$$

$$[Vqd0] = \{[r] + \rho[Z'] + [Z]\,[Z']\}\,[iqd0] \quad (52)$$

Reemplazando todos los valores de estas matrices que ya se conoce se llega a obtener las ecuaciones de voltaje para cada eje de referencia así:

$$
\left.
\begin{aligned}
Vq_s &= r_s.iq_s + \rho\lambda q_s + \omega\lambda d_s \\
\end{aligned}
\right\}
$$

$$
\begin{aligned}
Vd_s &= r_s.id_s + \rho\lambda d_s - \omega\lambda q_s \\
V0_s &= r_s.i0_s + \rho\lambda 0_s \\
V'q_r &= r_r.i'q_r + \rho\lambda' q_r + (\omega - \omega_r)\lambda' d_r \\
V'd_r &= r_r.i'd_r + \rho\lambda' d_r - (\omega - \omega_r)\lambda' q_r \\
V'0_r &= r_r.i'0_r + \rho\lambda' 0_r
\end{aligned}
\qquad (53)
$$

De donde:

$$
\left.
\begin{aligned}
\lambda q_s &= Ll_s.iq_s + Lm(iq_s + i'q_r) \\
\lambda d_s &= Ll_s.id_s + Lm(id_s + i'd_r) \\
\lambda 0_s &= Ll_s.i0_s \\
\lambda' q_r &= L'l_r.i'q_r + Lm(iq_s + i'q_r) \\
\lambda' d_r &= L'l_r.i'd_r + Lm(id_s + i'd_r) \\
\lambda' 0_r &= L'l_r.i'0_r
\end{aligned}
\right\} \qquad (54)
$$

Para la ecuación del torque electromagnético en ejes qd0 se reemplaza la ecuación de transformación a la ecuación (28) y se obtiene lo siguiente:

$$
Te = \frac{P}{2}\left[ iabc_s \right]^T \left[ \frac{dL'_{sr}}{d\alpha} \right] \left[ qd0_r \right] \qquad (55)
$$

$$
Te = \frac{P}{2}\left[ Ks^{-1} \right][iabc_s]^T \left[ \frac{dL'_{sr}}{d\alpha} \right] [Kr^{-1}] \left[ qd0_r \right] \qquad (56)
$$

Desarrollando esta última ecuación resulta:

$$
\begin{aligned}
Te &= \frac{n}{2}\left( \frac{P}{2} \right) Lm(iq_s.i'd_r - id_s.i'q_r) \\
Te &= \frac{n}{2}\left( \frac{P}{2} \right)(\lambda' q_r.i'd_r - \lambda' d_r.i'q_r)
\end{aligned}
\qquad (57)
$$

n : Número de fases de la máquina de Inducción en este caso es igual a 3. El torque electromecánico esta dado por la siguiente expresión:[11]

$$Te = J\rho\omega_r + D\omega_r + T_L \qquad (58)$$

Donde:

Te : Torque electromagnético

J : Momento de inercia del rotor y en algunos casos de la máquina.

D : Coeficiente angular de viscosidad

$T_L$ : Torque mecánico o de carga

$J\rho\omega_r$ : Torque de aceleración

$D\omega_r$ : Torque de amortiguamiento mecánico

De aquí el valor D resulta ser muy pequeño con relación a los demás valores por lo que se desprecia este valor quedando la ecuación anterior en forma definitiva así:

$$Te = J\rho\omega_r + T_L \qquad (59)$$

## 1.5.- ECUACIONES EN VARIABLES DE ESTADO [12]

Las ecuaciones de voltaje (53) se encuentran en función de corrientes y concatenaciones de flujo, ambas no pueden ser las variables de estado, por lo que se utiliza las concatenaciones de flujo como las variables de estado antes que la corriente debido a que las ecuaciones que rigen el comportamiento de la máquina de inducción se hacen más manejables. Se parte de:

$$X = 2\pi f L \qquad (60)$$

[11] Tesis Mancero Santillán Fernando. Politécnica Nacional. Pág 27.
[12] Analysis of Electric Machinery. Paul C. Krause. Pág. 174.

Donde $2\pi f = \omega_e$ y es la velocidad angular eléctrica base, que es función de la frecuencia nominal, por lo que la ecuación (54) al multiplicar por esta velocidad angular quedará:

$$\left.\begin{aligned}
\psi\ q_s &= Xl_s.iq_s + Xm(iq_s + i'q_r)\\
\psi\ d_s &= Xl_s.id_s + Xm(id_s + i'd_r)\\
\psi\ 0_s &= Xl_s.i0_s\\
\psi' q_r &= X'l_r.i'q_r + Xm(iq_s + i'q_r)\\
\psi' d_r &= X'l_r.i'd_r + Xm(id_s + i'd_r)\\
\psi' 0_r &= X'l_r.i'0_r
\end{aligned}\right\} \quad (61)$$

Y reescribiendo las ecuación (53) de voltaje se obtienen:

$$\left.\begin{aligned}
Vq_s &= r_s.iq_s + \frac{1}{\omega_e}\rho(\psi\ q_s) + \left(\frac{\omega}{\omega_{\grave{e}}}\right)\psi\ d_s\\
Vd_s &= r_s.id_s + \frac{1}{\omega_e}\rho(\psi\ d_s) - \left(\frac{\omega}{\omega_{\grave{e}}}\right)\psi\ q_s\\
V0_s &= r_s.i0_s + \frac{1}{\omega_e}\rho(\psi\ 0_s)\\
V'q_r &= r_r.i'q_r + \frac{1}{\omega_e}\rho(\psi' q_r) + \left(\frac{\omega-\omega_r}{\omega_e}\right)\psi' d_r\\
V'd_r &= r_r.i'd_r + \frac{1}{\omega_e}\rho(\psi' d_r) - \left(\frac{\omega-\omega_r}{\omega_e}\right)\psi' q_r\\
V'0_r &= r_r.i'0_r + \frac{1}{\omega_e}\rho(\psi' 0_r)
\end{aligned}\right\} \quad (62)$$

Se define la siguiente igualdad:

$$\psi mq = Xm(iq_s + i'q_r) \quad (63)$$

$$\psi md = Xm(id_s + i'd_r) \quad (64)$$

Ahora las ecuaciones (63) y (64) se reemplaza en la ecuación (61) quedando:

$$\left.\begin{aligned}
\psi\ q_s &= Xl_s.iq_s + \psi mq \\
\psi\ d_s &= Xl_s.id_s + \psi md \\
\psi\ 0_s &= Xl_s.i0_s \\
\psi' q_r &= X'l_r.i'q_r + \psi mq \\
\psi' d_r &= X'l_r.i'd_r + \psi md \\
\psi' 0_r &= X'l_r.i'0_r
\end{aligned}\right\} \quad (65)$$

Al despejar las corrientes queda:

$$\left.\begin{aligned}
iq_s &= \left(\frac{\psi q_s - \psi mq}{Xl_s}\right) \\
id_s &= \left(\frac{\psi d_s - \psi md}{Xl_s}\right) \\
i0_s &= \left(\frac{\psi 0_s}{Xl_s}\right) \\
i'q_r &= \left(\frac{\psi' q_r - \psi mq}{X'l_r}\right) \\
i'd_r &= \left(\frac{\psi' d_r - \psi md}{X'l_r}\right) \\
i'0_r &= \left(\frac{\psi' 0_s}{X'l_r}\right)
\end{aligned}\right\} \quad (66)$$

Así se tiene las corrientes en los ejes qd0 expresadas en función de las variables de estado que se eligió que son las concatenaciones de flujo por segundo (ψ). Ahora esta ecuación (66) se vuelve a reemplazar en las ecuaciones (63) y (64) obteniéndose lo siguiente después de haber realizado las operaciones necesarias:

$$\psi mq = Xmq\left(\frac{\psi q_s}{Xl_s} + \frac{\psi' q_r}{X' l_r}\right) \tag{67}$$

$$\psi md = Xmq\left(\frac{\psi d_s}{Xl_s} + \frac{\psi' d_r}{X' l_r}\right) \tag{68}$$

De donde:

$$Xmq = Xmd = \frac{1}{\frac{1}{Xm} + \frac{1}{Xl_s} + \frac{1}{X' l_r}} \tag{69}$$

Por último queda expresar los voltajes obtenidos en la ecuación (62) en variables de estado y se hará reemplazando la ecuación (66) en las mismas, resolviendo las ecuaciones de voltaje y despejando los términos (ρψ) se obtiene:

$$\left.\begin{aligned}
\rho\psi q_s &= \omega_e\left[Vq_s - \frac{\omega}{\omega_e}\psi d_s + \frac{r_s}{Xl_s}\left(\psi mq - \psi q_s\right)\right] \\
\rho\psi d_s &= \omega_e\left[Vd_s + \frac{\omega}{\omega_e}\psi q_s + \frac{r_s}{Xl_s}\left(\psi md - \psi d_s\right)\right] \\
\rho\psi 0_s &= \omega_e\left[V0_s - \frac{r_s}{Xl_s}\left(\psi 0_s\right)\right] \\
\rho\psi' q_r &= \omega_e\left[V' q_r + \frac{\omega_r - \omega}{\omega_e}\psi' d_r + \frac{r'_r}{X' l_r}\left(\psi mq - \psi' q_r\right)\right] \\
\rho\psi' d_r &= \omega_e\left[V' d_r + \frac{\omega - \omega_r}{\omega_e}\psi' q_r + \frac{r'_r}{X' l_r}\left(\psi md - \psi' d_r\right)\right] \\
\rho\psi' 0_r &= \omega_e\left[V' 0_r - \frac{r'_r}{X' l_r}\left(\psi' 0_r\right)\right]
\end{aligned}\right\} \tag{70}$$

La ecuación (57) del torque también hay que expresarlo en función de las concatenaciones de flujo por lo se multiplica a dicha ecuación por ωe y se obtiene la siguiente ecuación:

$$Te = \frac{3}{2}\left(\frac{P}{2}\right)\left(\frac{1}{\omega_e}\right)(\psi' q_r . i' d_r - \psi' d_r . i' q_r) \qquad (71)$$

Finalmente la ecuación del torque electromecánico definida ya anteriormente en la ecuación (59) queda exactamente igual, solo que se despeja la variación de la velocidad rotórica ($\rho\omega_r$):

$$\rho\omega_r = \frac{Te - T_L}{J} \qquad (72)$$

Las ecuaciones (63) a (72) representan a todas las ecuaciones que rigen a la máquina en variables de estado en función de las concatenaciones de flujo y que serán las que se introducirán en el software para su respectiva simulación.

Es así que estas ecuaciones se pueden expresar en cualquier eje de referencia arbitrario asignando la correspondiente velocidad de estos ejes. Para la simulación los ejes de referencia más usados son:[13]

- Eje de Referencia Estacionario ($\omega = 0$), ($\rho\theta = 0$) y ($\beta = -\theta_r$)

- Eje de Referencia Fijo en el Rotor ($\omega = \omega_r$), ($\rho\ \beta = 0$).

- Eje de Referencia Rotando Sincrónicamente ($\omega = \omega_e$), ($\theta = \theta_e$) y ($\beta = \theta_e - \theta_r$).

La elección del eje de referencia conveniente para el análisis y/o simulación se lo hará en base a las condiciones de operación.

## 1.6.- ECUACIONES EN ESTADO ESTABLE

[13] Analysis of Electric Machinery. Paul C. Krause. Pág. 178.

Las ecuaciones en estado estable son muy importantes en el análisis debido a que en ciertos períodos de asimetría se puede omitir este estado como es el arranque de la máquina pues aquí desde un principio se da el periodo transitorio, lo que no ocurre por ejemplo con un cortocircuito en donde la máquina ya estuvo funcionando en un "estado estable" para que se dé el periodo transitorio, he aquí la importancia de analizar este periodo.

Dicho esto el estado estable opera en condiciones balanceadas, aquí las corrientes del estator producen un campo magnético que gira en el entrehierro a la velocidad sincrónica o velocidad angular eléctrica ($\omega_e$) y debido a que el rotor gira a una velocidad diferente de ($\omega_e$) se inducen corrientes en el rotor que producen un campo magnético que gira a una velocidad ($\omega_e - \omega_r$).

Estas velocidades son en los ejes abc, ahora bien para la transformación al sistema arbitrario de referencia que gira a una ($\omega$) respecto al estator, varía ahora con una velocidad ($\omega_e - \omega$). Lo mismo se hace para el rotor, el cual gira a una velocidad ($\omega_e - \omega_r$) – ($\omega - \omega_r$) lo que resolviendo lo anterior se obtiene que el rotor gira en el sistema arbitrario de referencia a una velocidad ($\omega_e - \omega$). Por lo tanto todas las variables del estator y rotor en el sistema arbitrario de referencia qd0 varían con una velocidad ($\omega_e - \omega$) claro está en estado estable, todo esto para cualquier sistema de referencia excepto para el que gira a la velocidad sincrónica ($\omega = \omega_e$).

Aquí se incluye un concepto que hasta ahora no se ha visto que son los fasores, esto se debe a que solo en estado estable los voltajes y corrientes que son cantidades sinusoidales se pueden representar como cantidades fasoriales.

De las ecuaciones que se obtuvieron en (53) se hará las siguientes consideraciones para encontrar las ecuaciones en estado estable:

- Se reemplaza Fq (s o r) por Fa (s o r).

- Se reemplaza ρ por j(ωe – ω).

- Se fija el sistema de ejes arbitrario en el estator es decir (ω = 0).

Realizando operaciones con estas sustituciones en la ecuación (53) se obtiene lo siguiente en forma fasorial:

$$Va_s = (r_s + jXl_s)Ia_s + jXm(Ia_s + I'a_r) \tag{73}$$

$$\frac{V'a_r}{S} = \left(\frac{r'_r}{S} + jXl_s\right)Ia_r + jXm(Ia_s + I'a_r) \tag{74}$$

Donde:

$$S = \frac{\varpi_e - \omega_r}{\omega_e} \tag{75}$$

*S* es el valor de deslizamiento del motor. $Va_s$, $Vá_r$, $Ia_s$, $Iá_r$ son fasores, las ecuaciones anteriores sugieren el siguiente circuito equivalente:

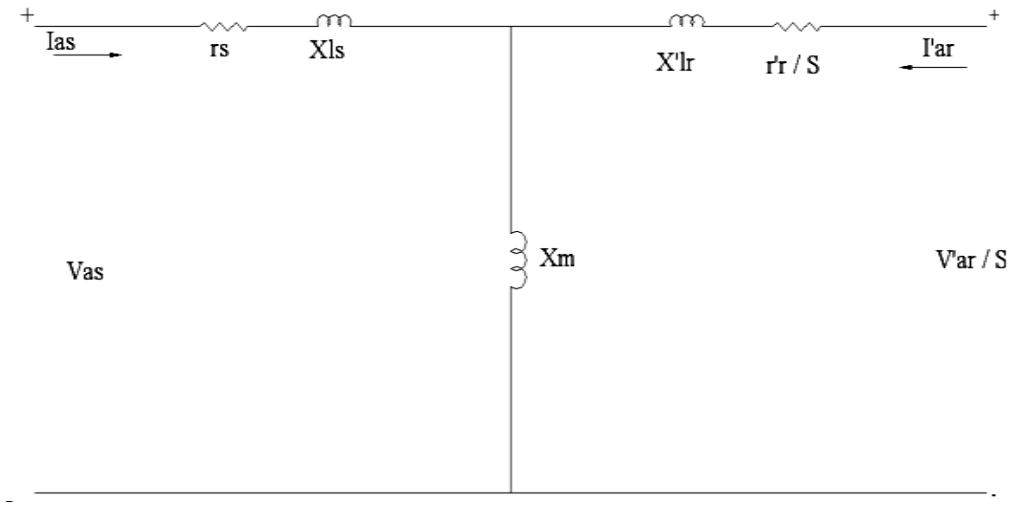



**Figura 1.4 Circuito equivalente para estado estable por fase de la máquina de inducción**.

# CAPÍTULO II

# 2. MODELOS DE LA MÁQUINA PARA LAS CONDICIONES DE OPERACIÓN

## 2.1.- INTRODUCCIÓN

En el capítulo anterior se pudo determinar las ecuaciones diferenciales que permiten la simulación digital de la máquina de inducción. Las ecuaciones que se utilizará en la programación para su posterior simulación son:

- Ecuaciones de transformación de variables abc a variables qd0 (36).
- Ecuaciones diferenciales de voltaje (70).
- Ecuaciones de corriente (66).
- Ecuaciones de concatenaciones de flujo por segundo (65), (67), (68) y (69).
- Ecuación de torque electromagnético (71).
- Ecuación de velocidad (72).

Las ecuaciones anteriores dentro del modelo digital en la simulación realizarán todos los cálculos, siendo necesario establecer condiciones que permitirán simular operaciones de arranque, falla y frenado.

Es necesario establecer que para cualquier modo de operación que se desee simular siempre se partirá del arranque de la máquina claro esta como se lo hace en cualquier aplicación con el motor. Lo que se pretende es que se pueda incluso realizar una operación de falla o frenado cuando la máquina aun no ha entrado a la condición de estado estable, he aquí la razón de siempre partir del arranque de la máquina.

Para la simulación de los distintos modos de operación de la máquina de inducción se consideran ciertas condiciones que se aprecian en los voltajes de fase del estator de la máquina que son las variables de entrada y son los que determinan las características de operación de la máquina, así que se empezara analizando las condiciones para arranque, falla, freno y variación de carga en la máquina.

## 2.2.- ARRANQUE DE LA MÁQUINA

En el arranque la máquina se considera sin alimentación debido a que el mismo es el punto de inicio de la simulación para cualquier modo de operación, en el arranque de la máquina en el instante t = 0 todas las variables de la máquina serán igual a 0, dicho esto el programa de simulación podrá realizar los siguientes arranques:

**2.2.1.- ARRANQUE DIRECTO.-** Aquí el estator de la máquina de inducción se conecta directamente a la fuente de voltaje o red en un solo tiempo (directo). Las características de este arranque son su alta corriente de arranque (4 a 8 veces la corriente nominal). Es ideal (si la red permite el pico de corriente de arranque) para arranques a plena carga de motores de pequeña y mediana potencia. Los voltajes de la red (línea) que se aplicarán al estator para cualquier simulación se han tomado como balanceados y de secuencia positiva teniendo lo siguiente:

$$Vab = \sqrt{2}\, V \cos( 2\pi f t )$$

$$Vbc = \sqrt{2}\, V \cos( 2\pi f t - 120^{\circ})$$

$$Vca = \sqrt{2}\, V \cos(2\pi ft + 120^{\circ})$$

Las conexiones del estator para el arranque directo pueden ser:

**2.2.1.1.- Estator en triángulo (Δ).-** El voltaje del estator corresponde al voltaje de línea. El Voltaje de línea es igual al voltaje de fase (VL = Vf). Esto se lo puede apreciar en el siguiente gráfico:

$$Vas = Vab$$
$$Vbs = Vbc$$
$$Vcs = Vca$$

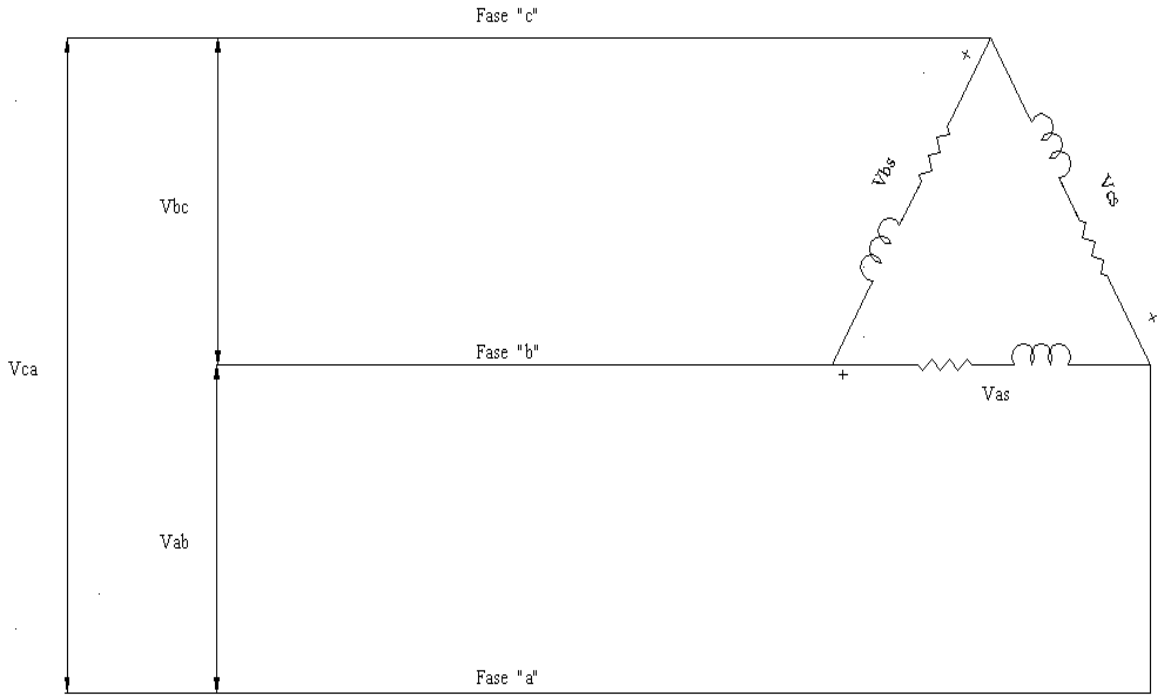



**Figura 2.1 Conexión del estator en triángulo**

**2.2.1.2.- Estator en estrella (Y).-** Conociendo el voltaje de línea, el voltaje aplicado al estator del motor es igual al voltaje de línea dividido para la raíz de tres (Vf = VL/$\sqrt{3}$). Según la figura siguiente será:

$$Vas = \frac{Vab}{\sqrt{3}}$$

$$Vbs = \frac{Vbc}{\sqrt{3}}$$

$$Vcs = \frac{Vca}{\sqrt{3}}$$

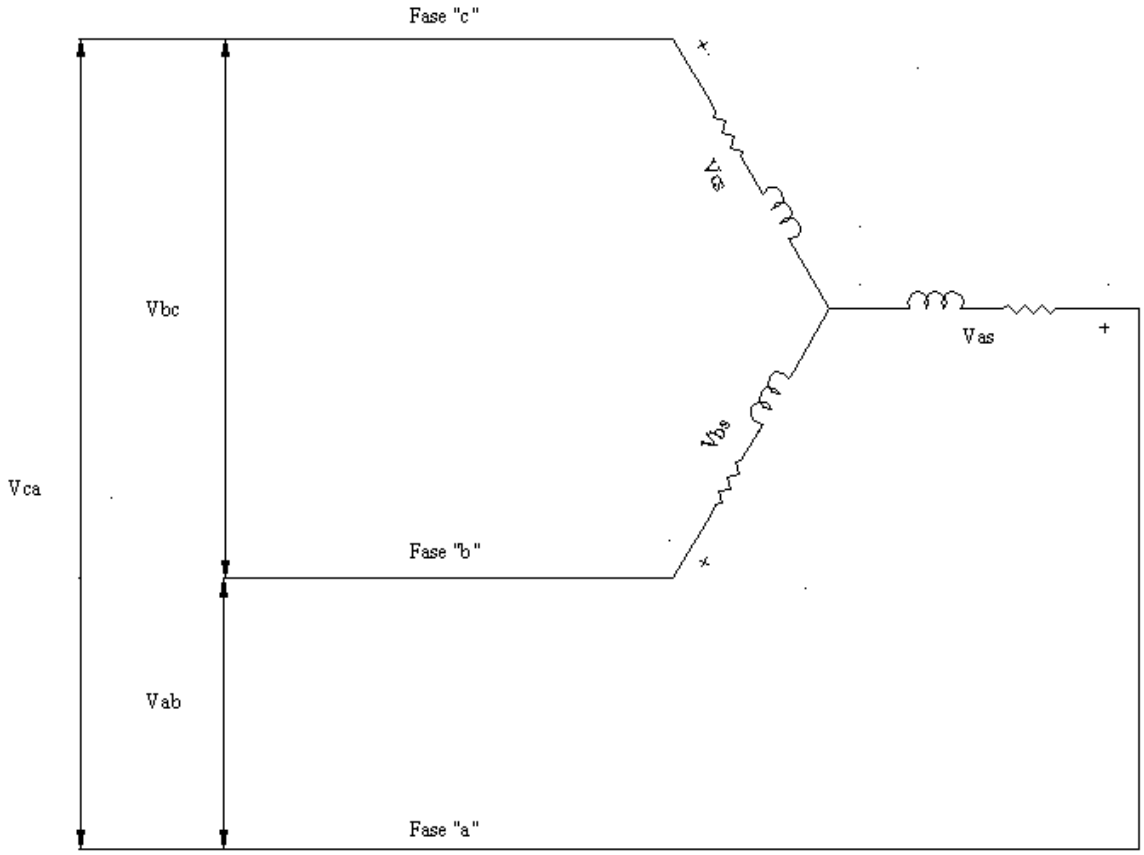



**Figura 2.2 Conexión del estator en estrella**

**2.2.2.- ARRANQUE ESTRELLA – TRIÁNGULO (Y-Δ).-** Ciertos motores poseen los dos extremos de los devanados del estator con salida a la caja de bornes del motor lo que implica tener seis terminales para poder conectar al motor ya sea en estrella o triángulo. Solo los motores con estos seis terminales disponibles y en donde la tensión de línea de la red sea igual al acoplamiento en triángulo se puede realizar este arranque.

El arranque se produce primero al conectar al motor en acoplamiento estrella, lo que en un principio quiere decir que el motor se encuentra alimentado con

una tensión del 57.8 % respecto a la tensión nominal ($1/\sqrt{3}$), a la vez esto produce un tercio del torque a tensión nominal debido a la relación cuadrática que tiene con este. Las ecuaciones que rigen este primer paso se anoto en 2.2.1.2.

El segundo paso será desacoplar el acoplamiento estrella y conectar ahora el acoplamiento triángulo ahora los voltajes de línea pasan a ser los voltajes de fase del estator, aunque no es una regla se recomienda que el paso del acoplamiento estrella a triángulo se dé cuando el motor haya arrancado. Con este paso se logra obtener todas las características a tensión nominal que tiene el motor, por lo que este arranque es más recomendable para máquinas que arranquen en vació (sin carga) o tengan un par resistente pequeño. Las ecuaciones que rigen este segundo paso se enunciaron en el arranque directo con el estator conectado en triángulo.

## 2.3.- FALLAS[14]

Dentro de la operación normal de un motor de inducción cualquiera que fuese este se pueden producir fallas las cuales muchas veces se generan por los circuitos de control, de fuerza e incluso del motor mismo, es de vital importancia por lo tanto que este tipo de anomalías deben ser adecuadamente estudiadas para luego dar una adecuada inserción y calibración de los dispositivos de protección. Las fallas más comunes que se presentan en los motores de inducción y en general son los cortocircuitos, por lo que serán el punto importante las ecuaciones que rigen el comportamiento de las fallas por cortocircuito, dándose los siguientes casos: falla de una fase a tierra, falla de dos fases a tierra y falla de tres fases a tierra, cada una de estas fallas para el estator conectado en estrella y en triángulo.

**2.3.1.- FASE "a" A TIERRA. ESTATOR EN TRIÁNGULO.-** Para explicar esta falla se utiliza el triángulo dado en la siguiente figura.

[14] Tesis Valdivieso Veintimilla Luis Fernando. Politécnica Nacional. Págs. 63-67

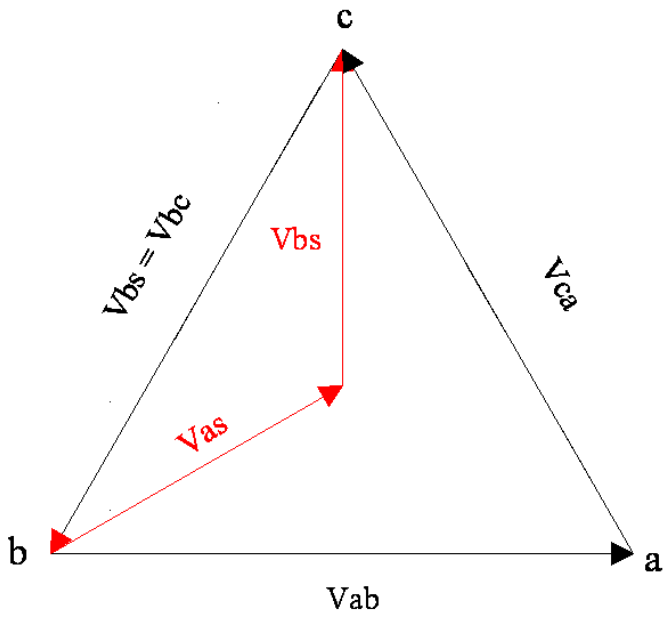



**Figura 2.3 Estator en delta falla en fase "a"**

Aquí el estator está conectado en triángulo y la falla se produce en la fase "a", entonces los voltajes que se producen en las fases "a" y "c" en el momento de la falla se reducen, mientras que el voltaje de la fase "b" se mantiene constante. Dicho de esta forma las ecuaciones de voltaje de fase antes de la falla corresponden a las analizadas en la parte del arranque directo en triángulo, mientras que las ecuaciones de voltaje de fase que se producen al momento de la falla son las siguientes:

$$Vas = \frac{Vab}{\sqrt{3}} \angle 30°$$
$$Vbs = Vbc$$
$$Vcs = \frac{Vca}{\sqrt{3}} \angle -30°$$

**2.3.2.- FASE "a" A TIERRA. ESTATOR EN ESTRELLA.-** Este análisis es más sencillo debido a que solo se hace cero la fase que está en falla y no necesita un gráfico para su explicación. Entonces antes de la falla el motor se encuentra con los voltajes de fase estudiados en arranque directo en estrella para después de la falla se den los siguientes voltajes de fase.

$$Vas = 0$$
$$Vbs = \frac{Vbc}{\sqrt{3}}$$
$$Vcs = \frac{Vca}{\sqrt{3}}$$

**2.3.3.- FASES "a y b" A TIERRA. ESTATOR EN TRIÁNGULO.-** Al producirse la falla en estas fases y según el gráfico el voltaje de fase en "a" llega a ser cero mientras que en las otras dos fases los voltajes se reducen. Al igual que en la falla de una fase a tierra el motor antes de la falla se encuentra con los voltajes estudiados para el arranque con el estator conectado en triángulo para luego después de la falla se produzcan estas ecuaciones de voltaje de fase:

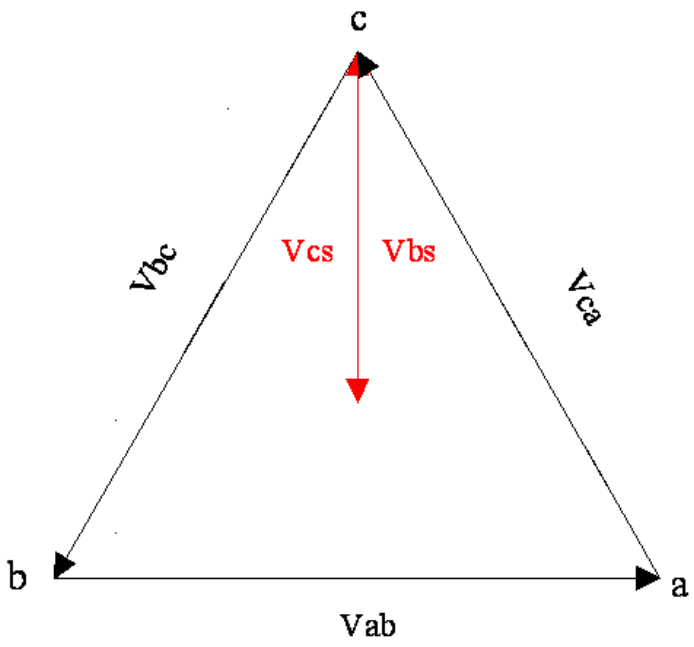



**Figura 2.4 Estator en delta falla en fases "a" y "b"**

$$Vas = 0$$
$$Vbs = \frac{Vbc}{\sqrt{3}} \angle 30°$$
$$Vcs = \frac{Vca}{\sqrt{3}} \angle -30°$$

**2.3.4.- FASES "a y b" A TIERRA. ESTATOR EN ESTRELLA.-** Como ya se dijo anteriormente en esta condición únicamente interesa hacer cero los voltajes

que se encuentra bajo la influencia de la falla, antes de producirse la falla el motor se encuentra alimentado por los voltajes de fase del arranque directo con el estator conectado en estrella para luego de la falla producirse estos voltajes de fase:

$$Vas = 0$$
$$Vbs = 0$$
$$Vcs = \frac{Vca}{\sqrt{3}}$$

**2.3.5.- FASES "a, b y c" A TIERRA. ESTATOR EN TRIÁNGULO Y ESTRELLA.-** En este caso el análisis se facilita incluso no se precisa de un gráfico para las explicación debido que al producirse una falla trifásica sin importar si el estator esta en triángulo o en estrella los voltajes de fase se hacen cero, de igual forma antes de la falla el motor se encuentra con los voltajes de fase correspondientes a cada caso de conexión del estator para luego de la falla darse estos voltajes de fase:

$$Vas = 0$$
$$Vbs = 0$$
$$Vcs = 0$$

## 2.4.- FRENADO[15]

Dentro de la industria resulta ser de gran importancia para las aplicaciones que se requiere el detener con rapidez y en muchos casos con exactitud el motor. Al realizarse el frenado se producen al igual que en los otros casos estados transitorios que llevaran al técnico encargado de la operación a la correcta manipulación del frenado así como también la correcta elección según la

[15] Tesis Valdivieso Veintimilla Luis Fernando. Politécnica Nacional. Págs. 67-70

magnitud y duración de los períodos transitorios de las protecciones y sistemas de control del motor.

Existe una serie de métodos para el frenado de motores siendo en los motores de inducción los más utilizados dos que son: el de inyección de corriente continua y el llamado por contracorriente o inversión de fases[16] que en muchos casos no solo se lo utiliza como freno de motores sino también de inversión de sentido de giro debido a su principio de funcionamiento que se detallará a continuación con sus respectivos voltajes de fase para su simulación en el computador:

**2.4.1.- FRENADO POR CORRIENTE CONTINUA.-** Este procedimiento consiste en inyectar o conectar o una fuente de corriente continua a dos fases del motor previamente de haber desconectado al motor de la fuente de alimentación de corriente alterna. La corriente que circula por las dos fases del motor conectadas a la fuente de CC crea un campo magnético estacionario en el espacio o entrehierro, mientras que en el rotor debido a su rotación se inducen corrientes, estas corrientes rotóricas interactúan con las del estator produciéndose un torque tal que lleva al motor a su detención.

Se recomienda que el voltaje aplicado de CC debe ser de tan solo el 25% del voltaje de fase del motor o que la corriente de CC deba ser 1,3 veces la corriente nominal del motor. A continuación las ecuaciones de frenado por corriente continua según la conexión del estator.

**2.4.1.1.- Estator en triángulo.-** En la siguiente figura se representa como se introduce el voltaje de corriente continua que se aplica a las fases “a y b”, antes de eso el motor se encuentra funcionando con los voltajes de fase que se dieron del arranque directo con conexión en triángulo, luego de desconectar la fuente de corriente alterna y conectar las dos fases a corriente continua se tiene los siguientes voltajes de fase:

[16] Concepto de un Automatismo – Telemecanique. Pág. 92.

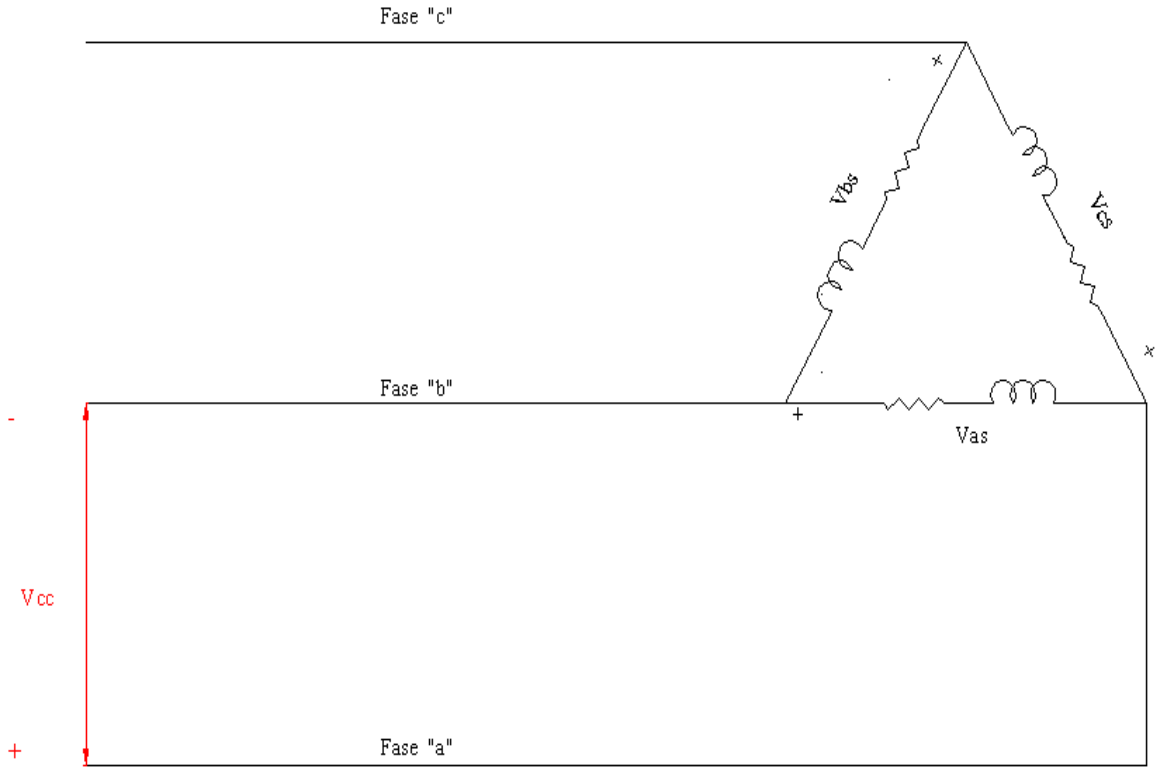



**Figura 2.5 Voltaje de corriente continua a las fases “a” y “b”. Estator en triángulo.**

$$Vas = Vcc$$

$$Vbs = -\frac{Vcc}{2}$$

$$Vcs = -\frac{Vcc}{2}$$

**2.4.1.2.- Estator en estrella.-** De igual forma con la figura 2.6 se muestra la conexión de la fuente de corriente continua a las fases “a y b” del motor, antes de producirse el freno el motor se encuentra funcionando con los voltajes de fase dados en el arranque directo con conexión en estrella y cuando se produce el freno se dan los siguientes voltaje de fase:

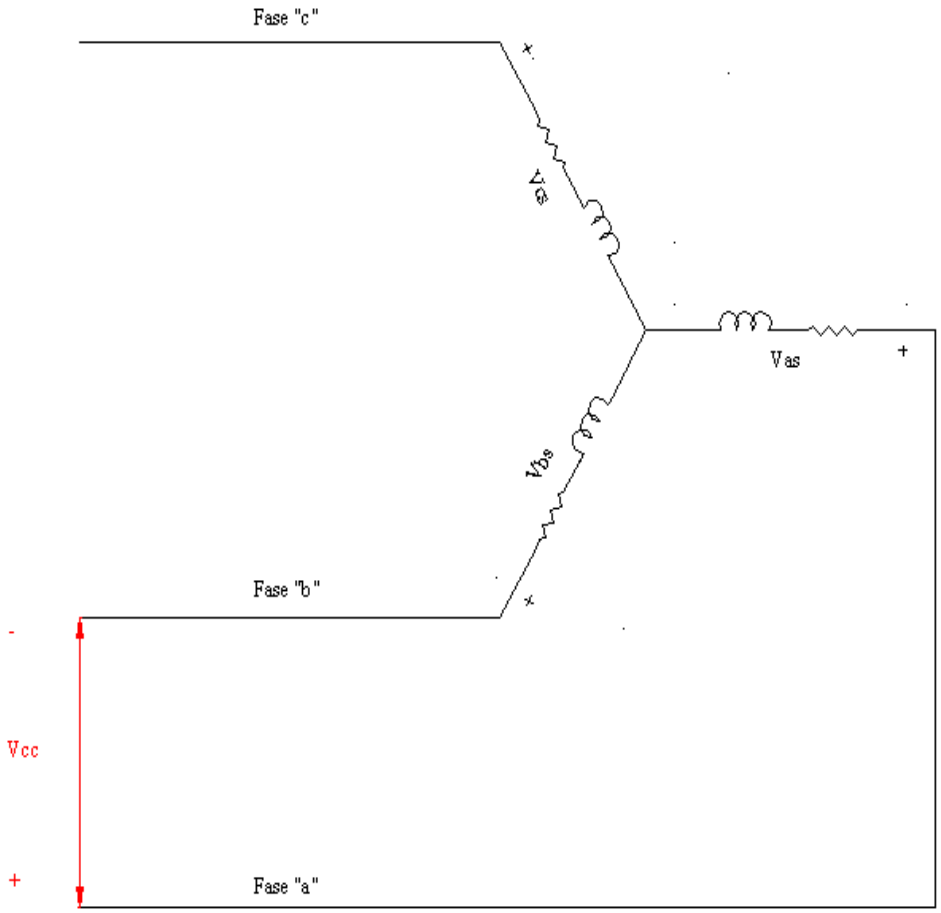



**Figura 2.6 Voltaje de corriente continua a las fases “a” y “b”. Estator en estrella.**

$$Vas = \frac{Vcc}{2}$$

$$Vbs = -\frac{Vcc}{2}$$

$$Vcs = 0$$

**2.4.2.- FRENADO POR CONTRACORRIENTE.-** Este frenado consiste en después de haber desconectado la alimentación de la red y girando aun el motor, inmediatamente invertir dos de sus fases y se reconecta el motor logrando así que las corrientes del rotor no disminuyan en su totalidad logrando con esto que ahora el campo magnético gire en sentido contrario oponiéndose al sentido de giro original logrando así que el motor se frene, cuando el motor este llegando a velocidad igual cero se debe re invertir las fases nuevamente sino el motor ahora girará en sentido contrario, esto ya no es parte del estudio.

Este frenado se utiliza mucho en aplicaciones típicas como máquinas herramientas y en aplicaciones a motores de baja potencia, a continuación se analiza los voltajes de fase producidos en el momento de la inversión para las dos conexiones del estator:

**2.4.2.1.- Estator en triángulo.-** La inversión de fases se las realiza entre las fases "b y c" y antes de que se realice el frenado el motor se encuentra funcionando con los voltajes de fase dados en arranque directo con conexión en triángulo y luego cuando se produce el frenado los voltajes que se generan se pueden apreciar en la figura 2.1 y ya producida la inversión los voltajes de fase llegan a ser los siguientes:

$$Vas = Vca$$
$$Vbs = Vbc$$
$$Vcs = Vab$$

**2.4.2.2.- Estator en estrella.-** De forma más sencilla se puede apreciar en la figura 2.2 que la inversión de fases se da en las fases "b y c" y de forma similar que en los casos anteriores antes de producirse el frenado el motor se encontrara funcionando con los voltajes de fase dados en el arranque directo conexión en estrella para luego producirse los siguientes voltajes de fase:

$$Vas = \frac{Vab}{\sqrt{3}}$$
$$Vbs = \frac{Vca}{\sqrt{3}}$$
$$Vcs = \frac{Vbc}{\sqrt{3}}$$

## 2.5.- VARIACIÓN DE CARGA

La variación de carga en un motor está en función del tipo de carga con la que se accione a esta pudiendo tener tres clases que son:

- Cargas cuyo torque es independiente a la velocidad del motor.

- Cargas cuyo torque varía linealmente con la velocidad del motor

- Cargas cuyo torque varía cuadráticamente con la velocidad del rotor.

- En vacío.

Para que la condición de variación de carga sea introducida a los módelos matemáticos analizados anteriormente se recurre a una ecuación matemática experimental dada por la siguiente expresión:[17]

$$T_L = T_N * \left( \frac{\omega_r}{\omega_N} \right)^X$$

Donde:

$T_L$ = Torque mecánico o de carga
$T_N$ = Torque nominal de la máquina accionada a la velocidad nominal
$\omega_N$ = Velocidad nominal de la máquina
X = Exponente que define el tipo de carga para la variación de carga.

De la expresión anterior se conocen todos los datos incluso el exponente X que según lo que se ve en la figura 2.7 y su explicación se puede elegir según su aplicación:

[17] Tesis Pozo Acosta Mauricio Efraín. Politécnica Nacional. Pág. 53.

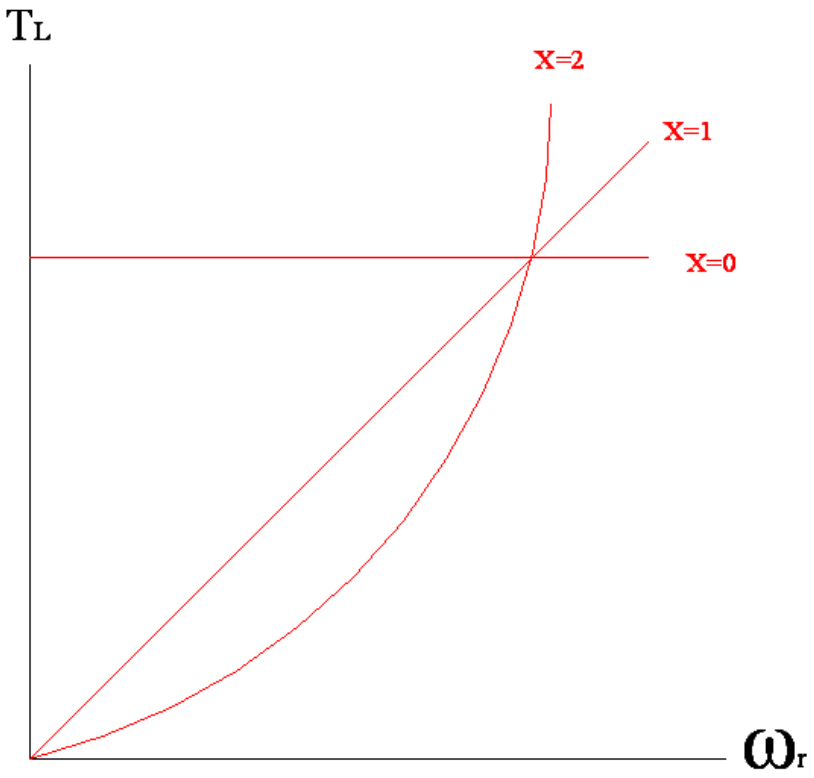



**Figura 2.7. Variación de carga según el tipo de carga**

**2.5.1.- CARGA CUYO TORQUE ES INDEPENDIENTE A LA VELOCIDAD (X=0).-** Este tipo de cargas se caracterizan por ser utilizadas en aplicaciones como bombas de pistón, aparatos de elevación, cintas transportadoras con la masa transportada constante, también en los mecanismos de avance de las máquinas – herramientas, etc.

**2.5.2.- CARGA CUYO TORQUE VARÍA LINEALMENTE CON LA VELOCIDAD (X=1).-** Este tipo de cargas no son muy generales o utilizadas pero como una manera de explicación es cuando se enciende un generador de excitación independiente que actúa sobre una resistencia externa fija.

**2.5.3.- CARGA CUYO TORQUE VARÍA CUADRÁTICAMENTE CON LA VELOCIDAD (X=2).-** Este es el tipo de carga que en la aplicación de motores esta mayormente difundida y se lo puede apreciar según sus aplicaciones como en compresores, hélices de buques, bombas centrifugas, ventiladores, etc.

**2.5.4.- EN VACÍO.-** Esta condición dentro de la industria no es muy aplicada más que en laboratorios para hacer pruebas a los motores como es la llamada prueba en vacío. En forma ideal se puede decir que en esta condición el $T_N = 0$, cosa que no es muy cierta pues siempre se genera un cierto torque mínimo por rozamientos y demás factores mecánicos, aquí no es necesario determinar el exponente X pues para cualquier valor de este el $T_L$ siempre será cero o cercano a este.

La razón de trabajar con voltajes de fase Vas, Vbs, Vcs se debe a que las ecuaciones diferenciales no toman en cuenta las condiciones de la máquina sino únicamente pide que todas las ecuaciones de entrada para el programa sean transformadas a voltajes de fase como se lo ha realizado en el presente capítulo.

Estos voltajes según la condición de funcionamiento serán los que interactúen con los voltajes normales de la fuente dados en un principio, los demás datos de entrada son invariantes tan solo los tiempos para que se den las condiciones serán parte in dispensable en la simulación.

# CAPÍTULO III

# 3. PROGRAMACIÓN

## 3.1.- INTRODUCCIÓN

En este capítulo se tratará exclusivamente lo que tiene que ver con la parte computacional y su programación así como también la parte de obtención de parámetros del motor a utilizar en base a las pruebas de vacío y corto circuito que se verá posteriormente en este capítulo.

Para el desarrollo de la tesis se ha creído necesario utilizar programas computacionales que faciliten ciertas etapas como son la simulación, la resolución de ecuaciones diferenciales y la adquisición de datos por lo que ha sido necesario el uso de programas computacionales que se encuentran en el mercado y que puedan facilitar el análisis, esto se hará empleando los programas Matlab y Lab View.

## 3.2.- SIMULACIÓN EN EL COMPUTADOR

El hecho de modelar la máquina y a través de la utilización de la transformación qd0 para ayudar a obtener ecuaciones diferenciales de coeficientes constantes (70 y 72) obliga a utilizar un método de resolución digital puesto que esta

resolución sería muy difícil quizás imposible de realizarlo sin la ayuda de estos programas.

Por tal razón se ha creído conveniente realizar el desarrollo y resolución numérica de estas ecuaciones utilizando el programa matemático computacional Matlab R14 versión 7.0 debido a que este incorpora dentro de su programación a más de una interface del usuario que es el GUIDE (Graphical User Interface Development Environment) una serie de algoritmos que permiten la resolución de este tipo de ecuaciones como son los ODE (Ordinary Differential Equations):

ode45, ode23, ode113, ode15s, ode23s, ode23t, ode23tb

La elección de uno de los algoritmos escritos anteriormente depende mayormente de la aplicación y el tiempo de computo, pues si bien es cierto la exactitud de los mismos varía y por ende el tiempo de computo puede ser mayor o menor, sin embargo la propia ayuda de Matlab proporciona una tabla de sugerencia para poder guiarse con el tipo de algoritmo a utilizar así:

| Solver | Problem Type | Order of Accuracy | When to Use |
|---|---|---|---|
| ode45 | Nonstiff | Medium | Most of the time. This should be the first solver you try. |
| ode23 | Nonstiff | Low | For problems with crude error tolerances or for solving moderately stiff problems. |
| ode113 | Nonstiff | Low to high | For problems with stringent error tolerances or for solving computationally intensive problems. |
| ode15s | Stiff | Low to medium | If ode45 is slow because the problem is stiff. |
| ode23s | Stiff | Low | If using crude error tolerances to solve stiff systems and the mass matrix is constant. |
| ode23t | Moderately Stiff | Low | For moderately stiff problems if you need a solution without numerical damping. |
| ode23tb | Stiff | Low | If using crude error tolerances to solve stiff systems. |

**Tabla 1. Ayuda de ODE's proporcionada por Matlab**

Ahora bien según la tabla anterior la recomendación es el utilizar el algoritmo ODE 45 que será el primer método a intentar utilizar y según el tiempo de computo y la exactitud es el más conveniente, sin embargo para que no quede duda en la programación y en las pruebas del software se realizaron pruebas con los algoritmos ODE45, ODE23, ODE113 encontrándose efectivamente que

el algoritmo ODE 45 es la mejor opción para la resolución de las ecuaciones diferenciales de coeficientes constantes.

El método que utiliza el algoritmo ODE45 para la resolución de las ecuaciones diferenciales es el de bisección, y la línea de programación que utiliza el mismo es el siguiente:

[t,x] = ode45('ecdif',[0:0.001:t],[0,0,0,0,0,0,0],options)

Ampliando la línea de programación anterior lo que explica es que va a resolver un sistema de variables x (puede ser x o y o cualquier otra variable con la que he identifica el sistema de ecuaciones) en función del tiempo. A continuación la explicación de cada parte de esta línea de programación:

[t,x] : Este es un vector de resultado que me da la resolución del sistema de ecuaciones en función del tiempo.

Ode45 : Es el algoritmo a emplear para la resolución de las ecuaciones.

ecdif : Es el archivo donde se encuentras las ecuaciones diferenciales que se van a resolver.

[0:0,001:t] : Es el vector tiempo para la resolución, el primer valor que es cero siempre será constante pues siempre será cero el inicio para cualquier simulación y el valor de 0,001 es un valor de incremento que se determina al igual que al algoritmo Ode45 en base a pruebas realizadas para tener una buena exactitud con un buen tiempo de computo.

[0,0,0,0,0,0,0] : Este es un vector de condiciones iníciales para las variables, como se precisó en el primer capítulo estas condiciones siempre serán cero pues se empieza desde el arranque del motor.

Options : Esta palabra provee la posibilidad de poder agregar ciertas características como tolerancias y que en la programación tiene la siguiente sintaxis:

options = odeset('RelTol',1e-4,'AbsTol',[1e-4 1e-4 1e-4 1e-4 1e-4 1e-4 1e-4],'MaxStep',0.001 )

Esta última línea de programación no se la explicará a profundidad puesto que más que una característica importante de la programación son arreglos que se introducen al algoritmo de resolución.

Para poder realizar una programación estructurada se tomara en cuenta la siguiente figura con sus respectivos pasos:[18]

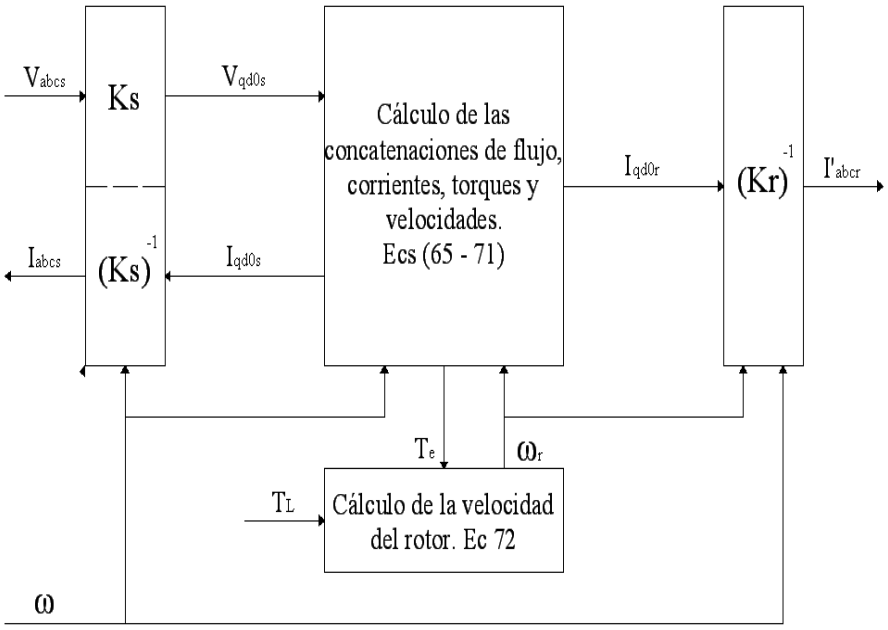



**Figura 3.1 Diagrama de bloques de la simulación de la máquina de inducción eléctrica.**

1.- Elegir un sistema de referencia como ya se vio puede ser el sistema de referencia fijo en el estator (W=0), fijo en el rotor (W=Wr) o la de sincronismo (W=We).

2.- Determinar los voltajes qd0 del estator en base a los voltajes de fase según el caso a simular que se dieron en el capítulo anterior.

[18] Tesis Valdivieso Veintimilla Luis Fernando. Politécnica Nacional. Pág. 62.

3.- Resolución del sistema de ecuaciones según los dos parámetros descritos anteriormente (1 y 2).

4.- Obtención de todas las variables de salida como corrientes en ejes qd0, torques electromagnéticos, velocidad rotórica, etc.

5.- Transformar las corrientes qd0 obtenidas a ejes abc.

6.- Graficar voltajes de fase, corrientes en el estator, corrientes en el rotor, torque electromagnético, velocidad rotórica y variación de carga en función de la velocidad.

Estos pasos hay que traducirlos a lenguaje de programación como ya se verá, pero antes de esto es necesario saber que en la programación el archivo “ecdif” es el que recopila toda la parte de programación de ecuaciones diferenciales, de variables de fase según el caso a simular, obtención de corrientes, torques velocidades, etc. así:

- **Paso Uno.- Elección del sistema de referencia deseado.-** La elección del sistema de referencia más adecuado se lo hará mediante un menú desplegable de selección en la interfaz del usuario en cada caso de simulación (arranque, falla y freno) y tendrá el siguiente código de programación:

```
case 1
    W=0;
    theta=0;
 case 2
    W=We;
    theta=x(7);
 case 3
    W=x(5);
    theta=x(6);
```

En donde cada caso representa lo siguiente: case 1 (referencia estatórica), case 2 (referencia sincrónica) y por último case 3 (referencia rotórica) este paso de programación se insertará como se dijo anteriormente en cada caso a simular. Los valores x(7) que es el ángulo θe, x(5) que es la velocidad angular rotórica y x(6) que es el ángulo θr, son calculados en cada paso de integración y resolución de las ecuaciones diferenciales.

- **Paso Dos.- Determinación de voltajes qd0.-** Para la determinación de estos voltajes se toma en cuenta los valores de fase del estator que según el caso a simular pueden ser los vistos en el capítulo dos, y en lenguaje de programación están dados por lo siguiente:

Vas = V*cos($\omega_e$*t);
Vbs = V*cos($\omega_e$*t-2*pi/3);
Vcs = V*cos($\omega_e$*t+2*pi/3);

Las ecuaciones anteriores son los voltajes de fase de la fuente trifásica y que son aplicados al estator, en este capítulo se está analizando los pasos de programación más no la programación en sí que se la tendrá en el Anexo D, por lo que aquí se dan los preliminares, por ejemplo si la simulación es del freno por corriente continua de un motor con el estator en delta se toman los voltajes dados en el capítulo dos y en lenguaje de programación es:

Vas = k(4,1);
Vbs = -(k(4,1)/2);
Vcs = -(k(4,1)/2);

Donde k(i,j) representa a una posición para definir un valor en este caso k(4,1) representa el valor de voltaje de corriente continua para la maniobra de freno. A continuación se indica la línea de programación que realiza la transformación de variables abc a qd0.

%Voltajes de abc a qd0

Vqs = (2/3)*(Vas*cos(theta)+Vbs*cos(theta-2*pi/3)+Vcs*cos(theta+2*pi/3));

Vds = (2/3)*(Vas*sin(theta)+Vbs*sin(theta-2*pi/3)+Vcs*sin(theta+2*pi/3));

Vqr=0;

Vdr=0;

Como se dijo se simulará máquinas de jaula de ardilla sencilla por lo que aquí la consideración de que Vqr y Vdr = 0, más no Vqs y Vds que siguen las ecuaciones dadas en el capítulo uno de transformación de variables abc a qd0.

- **Paso Tres.- Resolución del sistema de ecuaciones.-** Para este caso lo que se ha hecho es el escribir las ecuaciones en función de (x) por lo que es solo nomenclatura, las ecuaciones a resolver son siete que describen todo el comportamiento de la máquina y en código de programación son:

f(1)= $\omega_e$ *[Vqs+(rs/Xls)*(mq-x(1))-x(2)*( $\omega$/ $\omega_e$)];

f(2)= $\omega_e$ *[Vds+(rs/Xls)*(md-x(2))+x(1)*( $\omega$/ $\omega_e$)];

f(3)= $\omega_e$ *[Vqr+(rr/Xlr)*(mq-x(3))-x(4)*(( $\omega$-x(5))/ $\omega_e$)];

f(4)= $\omega_e$ *[Vdr+(rr/Xlr)*(md-x(4))+x(3)*(( $\omega$-x(5))/ $\omega_e$)];

f(5) = (Te-Tc)/J;

f(6) = x(5);

f(7)= $\omega_e$;

Estas siete ecuaciones no tienen otra función más que obtener los valores anotados anteriormente, los datos de entrada y el cálculo es automático a través del algoritmo Ode45, cabe mencionar que las primeras cuatro ecuaciones son las ecuaciones de voltaje encontradas en la ecuación 70; la quinta representa el cálculo de la velocidad rotórica en rad/seg, la sexta el cálculo de $\theta$r en base al cálculo de la velocidad rotórica y por último la séptima el cálculo del ángulo $\theta$e en función de la velocidad de sincronismo.

- **Paso Cuatro.- Cálculo de variables en función de las ecuaciones diferenciales.-** Aquí se presentan los cálculos necesarios para obtener corrientes, torque velocidades etc., siendo algunas como por ejemplo la velocidad rotórica un cálculo directo de las ecuaciones diferenciales para los demás valores tenemos los siguientes códigos de programación.

```
%Reactancia Mutua en ejes qd0

    Xlm = 1/(1/Xls + 1/Xlr + 1/Xm);

%Flujo Mutuo

    mq = Xlm*(x(1)/Xls + x(3)/Xlr);
    md = Xlm*(x(2)/Xls + x(4)/Xlr);

%Corrientes en ejes qd0

    iqs = (x(1) - mq)/Xls;
    ids = (x(2) - md)/Xls;
    iqr = (x(3) - mq)/Xlr;
    idr = (x(4) - md)/Xlr;

%Torques y velocidades

    Te = (3/4)*(P/We)*((x(2)*iqs)-(x(1)*ids));
    Wr =((freq /(P*pi))*(x(5)));
    Tc = Tl*((Wr/Wn)^(v));
```

Hasta aquí se ha calculado todos los valores que son necesarios en la simulación, lo único que hay que tomar en cuenta es que las corrientes calculadas están en variables qd0 por lo que será necesario transformarlas a variables abc para su respectiva graficación los demás valores se pueden ya almacenar en archivos que posteriormente se abrirán para obtener sus valores.

- **Paso Cinco.- Transformación de valores de corriente qd0 a variables abc.**- en este punto es necesario utilizar las transformadas inversas de variables qd0 que se presentaron en el primer capítulo para obtener las corrientes del estator y rotor en variables abc, esto en código de programación se lo logra de la siguiente manera:

  ```
  %Corrientes del estator en ejes abc

  ias=iqs*cos(theta)+ids*sin(theta);
  ibs=iqs*cos(theta-2*pi/3)+ids*sin(theta-2*pi/3);
  ics=iqs*cos(theta+2*pi/3)+ids*sin(theta+2*pi/3);
  %Corrientes del rotor en ejes abc
  iar=iqr*cos(theta-x(6))+idr*sin(theta-x(6));
  ibr=iqr*cos(theta-x(6)-2*pi/3)+idr*sin(theta-x(6)-2*pi/3);
  icr=iqr*cos(theta-x(6)+2*pi/3)+idr*sin(theta-x(6)+2*pi/3);
  ```

- **Paso Seis.- Graficación de los valores obtenidos.-** Este último paso de la programación viene a ser el más sencillo pero a la vez el más importante puesto que a través de la graficación de los valores obtenidos en la simulación y grabados en las matrices permiten determinar los análisis de resultados en el siguiente capítulo.

  Cada grafica tiene su código de programación que varía únicamente en la matriz que se tome para la graficación en función del tiempo, por lo que el código básico y que se utiliza en todas las graficas es el de plot (t,A) donde:

  t : Vector tiempo

  A : Matriz que dependiendo de lo que se va graficar se deberá elegir.

De esta forma se explica en forma clara los pasos que se deben seguir para la simulación y que está en función de todos las ecuaciones analizadas y obtenidas en los dos capítulos anteriores. Es de mencionar también que aquí los códigos de programación se han transcrito en forma general pues no se

puede poner toda la programación pues no es necesario en este punto por lo que la totalidad de la misma se encuentra en el anexo D.

Sin embargo en el anexo C se ubica la estructura del programa computacional, que servirá para la simulación del motor de inducción, la mejor forma de hacerlo es mediante la representación en un diagrama de flujo el cual se presenta en tal anexo.

Ahora que se ha analizado los pasos a seguir para la simulación en el computador a través de Matlab es necesario entrar a la segunda parte que es la adquisición de datos.

## 3.3.- ADQUISICIÓN DE DATOS

La segunda parte de este tema de tesis será el de obtener de una forma rápida y confiable los datos correspondientes al motor del laboratorio, construyéndose para esto un módulo que permita una conexión que sin necesidad de mucho cableado se pueda obtener lo mencionado.

En un principio se pensó en la opción de utilizar osciloscopios u otros equipos eléctricos o electrónicos para la obtención de las curvas reales del motor que resulta válido pero no con la precisión que requiere este estudio, además que de esta forma no se tendría un proceso automático lo que repercutía en la perdida de ciertos transitorios, por tal motivo se recurre a hacerlo a través de un software que permite visualizar y tener un control del motor.

El programa más adecuado, más amigable en programación y con múltiples ayudas virtuales y físicas para la adquisición de datos es Lab View que con sus múltiples herramientas hacen más fácil la obtención de estas curvas, no sin antes esta tener a la mano la parte del hardware (Módulo llamado Transitorios Motor) para la obtención que en este caso ha sido a través de una tarjeta de adquisición de datos que es la que distribuye National Instruments y es la USB 6008 cuya hoja técnica esta en el anexo B.

Por las limitaciones del laboratorio a resistir un corto circuito, las pruebas a realizar serán dos: arranque directo y frenado por inversión de fases, lo más importante en el motor es la obtención de la forma de onda de corriente, que es donde más se aprecia los transitorios y es la variable de mayor importancia para el sistema de adquisición. Se diseño un programa en Lab View que permita la captura y visualización de datos del arranque y cualquier otro fenómeno como frenos o fallas con la respectiva conexión de los equipos, en este caso debido a lo mencionado antes los únicos fenómenos eléctricos que se realizarán son el arranque y el freno por inversión de fases donde el programa realizará la adquisición de los datos y las presentara en formas de onda que se verán y analizarán en el siguiente capítulo. A continuación se da una explicación de lo que realiza el programa a través de su programación y además se presenta los circuitos de control que permiten la manipulación o activación de los contactores que a su vez realizaran las maniobras en el motor.

En forma general el programa se diseño para activar dos relés que provocarán la conexión o desconexión de los contactores que realizarán el arranque y posterior a esto el frenado por inversión de fases del motor, antes de entrar al gráfico de la programación primero veremos la parte de visualización, control y activación del programa.

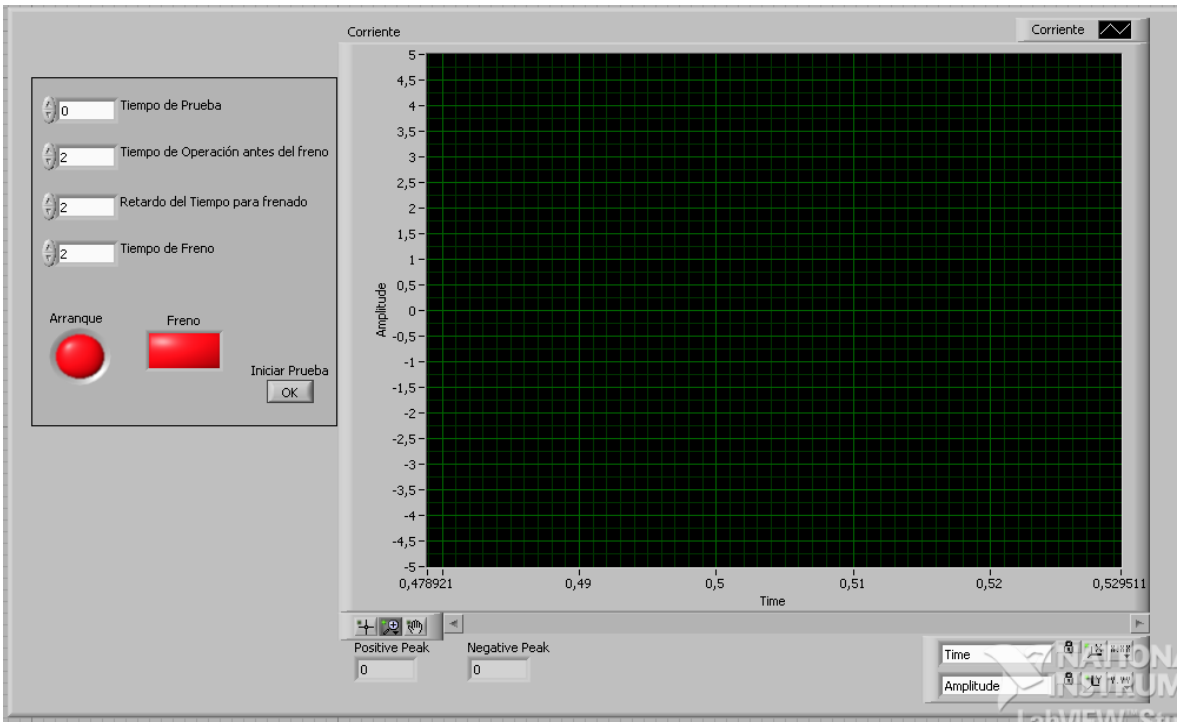



**Figura 3.2 Panel frontal del programa de adquisición de datos**

En la figura 3.2 se observa el panel frontal del programa de adquisición de datos donde los selectores de tiempo indicados significan lo siguiente:

- Tiempo de prueba: es el tiempo de duración total de adquisición y claro esta no podrá ser menor bajo ningún caso a la suma de los tres tiempos restantes ya que no se obtendría la forma de onda deseada. Tiempo recomendado de 2 a 5 seg.

- Tiempo de operación: antes del freno es el tiempo que el motor permanece así para el arranque y estado estable. Tiempo recomendado 1 seg.

- Retardo de tiempo para el frenado: es el tiempo de desconexión entre un contactor y otro. Valor recomendado 0.

- Tiempo de freno: es el tiempo que el motor funcionará con las fases invertidas. Valor recomendado para el motor de prueba 1 seg.

- El botón OK permite realizar la adquisición de los datos y el arranque del programa, se recomienda que el modo de Run sea el normal no el continuo.

La parte derecha de la figura 3.2 que es una pantalla virtual permite la visualización de la forma de onda de la corriente además de herramientas que permitan agrandar, mover, deslizar la grafica obtenida, así como dos indicadores de valores máximo y mínimo.

La programación se encuentra en una sola pantalla pero por cuestiones de espacio y de explicación se encuentra dividida en tres partes que se analizan a continuación:

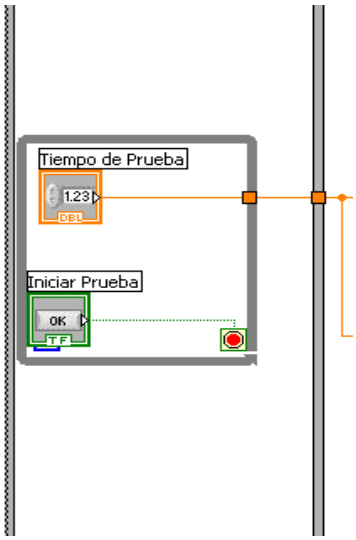



**Figura 3.3 Primera parte del programa**

En esta primera figura se observa la parte de activación del programa, en donde el mando o el botón OK se encuentra en un lazo *while loop* que dará inicio a todo el programa de adquisición, el tiempo de prueba es un valor que también permite realizar el cálculo de muestras por segundo que está en función de este valor, esto se verá con mayor profundidad en las otras figuras. Una parte muy importante del programa debido a que se desea obtener toda la curva desde el instante mismo del arranque sin perder ningún pico de corriente, es la de dar un tiempo para que el programa ya empiece a adquirir datos antes de arrancar al motor y así no perder ningún flanco, esto se logra con las

estructuras de secuencia que obligan a que primero realice una primera recolección de datos con valores cero paro luego pasar a la adquisición como se verá en la figura 3.4.

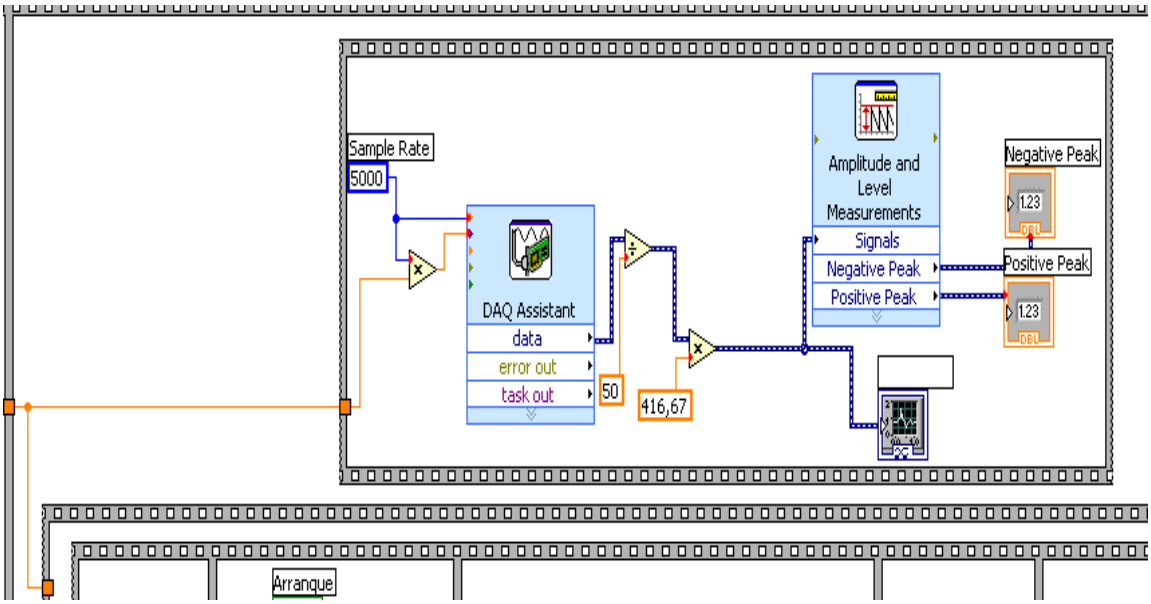



**Figura 3.4 Segunda parte del programa**

Se detalla en la figura 3.4 claramente el valor de tiempo que se mencionó en el texto anterior. El valor que se coloca en el de “tiempo de la prueba” ingresa a este lazo de secuencia, que es la parte de adquisición de datos, donde se ve un *“simple rate”* de 5000 que se lo ha tomado después de hacer pruebas y determinar que era lo mejor, como se explicó el valor de “tiempo de la prueba” permite automáticamente según el tiempo que se ingrese el determinar las muestras por segundo que son muy importantes para la buena adquisición de datos, este valor ingresa a *“DAQ Assistent”* que proporciona Lab View y permite evitar toda una programación pues ya vienen predefinidos en Lab View 8.2 para la adquisición de datos, el valor de 50 es el de las resistencias y el de 416,67 es la relación de transformación del transformador de corriente utilizado para atenuar el valor de corriente convertirlo en voltaje y así poder manipular con la tarjeta DAQ USB 6008, después de multiplicar estos dos valores y como ya se tienen los datos solo queda utilizar la herramienta de graficación y a través de herramientas virtuales obtener el valor máximo positivo y negativo de la forma de onda. De la figura 3.4 se notará que en el lado izquierdo de la

misma, se encuentra una derivación del “tiempo de la prueba” que realmente no tiene otro fin que obligar a que exista dependencia entre el bloque analizado anteriormente y el que se analizará a continuación.

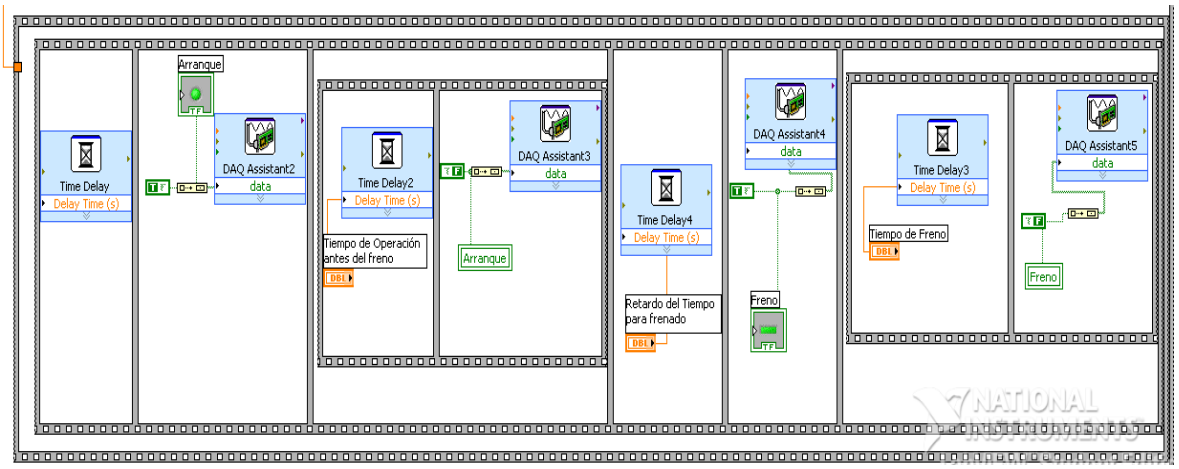



**Figura 3.5 Tercera parte del programa**

La figura 3.5 representa la parte de programación que contiene la parte más importante de todo el programa que es la activación de las salidas digitales y los retardos de tiempo, esta forma de programación, por lazos de secuencia, como se mencionó anteriormente obliga a que se vaya realizando secuencia por secuencia lo que interiormente se encuentra logrando que no se enclaven las salidas. En primer lugar se consigue un retardo automático que no es manipulado y que permite que el programa primero ejecute la parte de la secuencia anterior por un tiempo de 1 seg., para luego inmediatamente activar una salida digital TTL que es la P0.1 en la tarjeta de adquisición de datos, esta realizará el arranque del motor, una vez más se logra un retardo que se ingresa en el panel frontal y permite dar el tiempo de funcionamiento en estado estable antes de desconectar esta salida.

Siguiendo con el análisis se vuelve a dar un retardo que permite la conmutación del un contactor (arranque) al otro (inversión de fases) que incluso puede ser eliminado pues los contactores por su parte mecánica generan este retardo, sin embargo no se lo ha eliminado para poder ver este cambio en el programa, dado este retardo inmediatamente el programa manda a activar la salida P0.3 que realizará el activado del otro contactor que se utilizará para

cualquier maniobra en este caso exclusivo el frenado por inversión de fases, finalmente se manipulara un tiempo que permita decidir qué tiempo permanecerá en esta posición y luego mandará a desactivar el contactor apagando el motor, este tiempo no tiene nada que ver con el tiempo de prueba que según lo que se haya puesto también dejará de adquirir datos.

El programa analizado anteriormente en sus tres partes permite la comunicación con el hardware a través de la tarjeta de adquisición de datos la cual a través de un circuito electrónico permitirá activar los contactores con el circuito de la figura 3.6, cabe mencionar que el mismo circuito se aplica para las dos salidas digitales P0.1 y P0.3

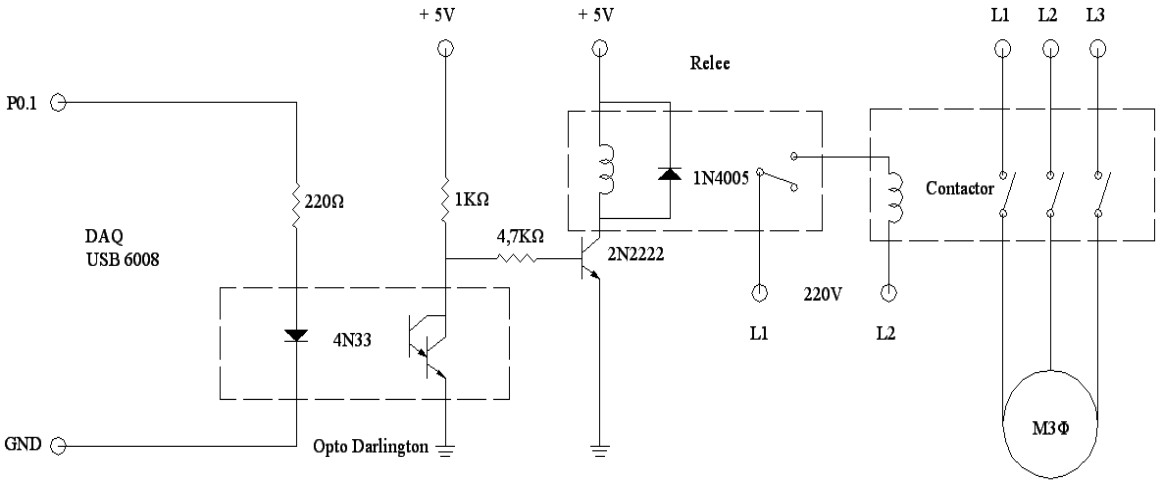



**Figura 3.6 Circuito para salida digital de la DAQ USB 6008**

La entrada análoga para adquirir la señal de corriente que proviene de la línea “L1” del motor también requiere de un circuito que permite acondicionar la señal de corriente, bajarla a un valor aceptable y a la vez convertirla en señal de voltaje que pueda ser aceptada por la entrada análoga AI 0 de la tarjeta de adquisición de datos, el circuito es:

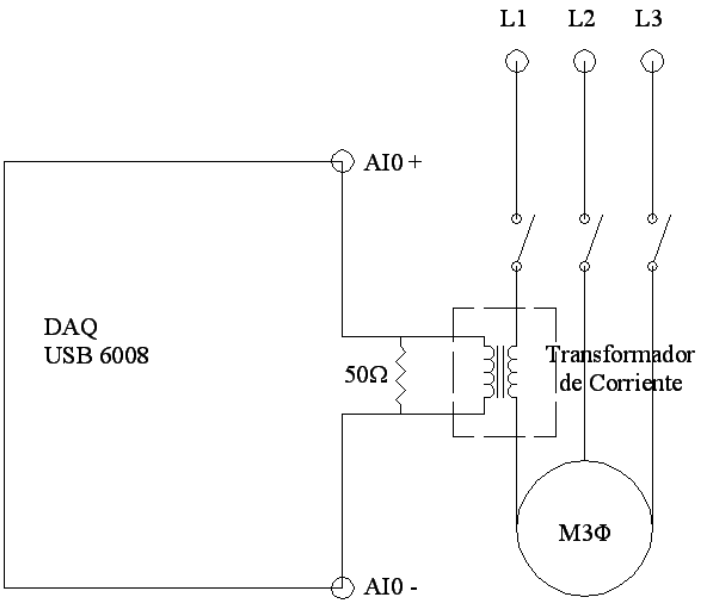



**Figura 3.7 Circuito para entrada análoga de la DAQ USB 6008**

## 3.4.- OBTENCIÓN DE PARÁMETROS

Para poder dejar listo todas aquellas ayudas que permitirán realizar la simulación y correspondiente análisis en el capítulo cuatro es necesario determinar también los parámetros o datos de entrada que se incluirán en la simulación y que podrán ser manipulados por el usuario para cualquier simulación.

Las pruebas que permiten obtener los parámetros eléctricos del motor se basan en las pruebas de vacío y rotor bloqueado que se efectúan en los transformadores para determinar sus parámetros y de forma similar se lo realiza en el motor, debido a la analogía que existe entre transformador y motor.

La información sobre pruebas de vacío y rotor bloqueado son extensas en lo que a procedimiento, ecuaciones y diagramas se refiere, tal es así que la información bibliografía de esta sección provee un pequeño programa para calcular estos valores en base a los resultados obtenidos en estas pruebas este se encuentra en el anexo E. Dicho esto no se ahondara en la consecución

de los valores de pruebas de vacío y rotor bloqueado ni tampoco en las ecuaciones para el cálculo de parámetros por lo que solo se expresara los valores obtenidos en las pruebas y después se presentan los valores obtenidos.

**Valores de Prueba en Vacío**

$V_L$ = 210 V
$I_L$ = 2,4 A
$P_T$ = 90 W
n = 1796 rpm
Rs = 2,6 Ω

**Valores de Prueba a Rotor Bloqueado**

$V_L$ = 56,1 V
$I_L$ = 4,3 A
$P_T$ = 283,5 W
frb = 60 Hz

Determinados estos valores se aplica las ecuaciones que todo texto de máquinas eléctricas posee o se puede aplicar el programa que proporciona la referencia bibliográfica transcrita en el anexo E,[19] cualquiera que fuese el método a utilizar los valores obtenidos para los parámetros eléctricos del motor son:

Rs = 2,6 Ω
R'r = 2,51 Ω
Xls = 2,2133 Ω
Xl'r = 3,3199 Ω
Xm = 48,183 Ω

[19] Máquinas eléctricas. Jimmy Cathey. Pág. 407.

Para que queden definidos totalmente todos los parámetros del motor obtenidos los parámetros eléctricos solo hace falta encontrar el momento de inercia del motor a simular, en la mayoría de los casos se lo encuentra en tablas como en este caso pero en otros será necesario aproximar y en otros calcular, la casa fabricante del motor que es Siemens proporciona este dato que se lo obtuvo de la siguiente tabla en función de la nomenclatura del motor que es 1LA7 080 4YA60 y la potencia del mismo que es 1 HP.[20]

| Velocidad 1800 rpm (4 polos) | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Código Actual | Código Antiguo | Tipo | Frame IEC | HP | kW | F.S. | In a 220 V Amp. | In a 440 V Amp. | rpm | Eficiencia η % | Torque nominal Nm | Momento de inercia Kg m2 | Torque de arranque de Tn | Intensidad como fac. de In | Peso aprox. kg |
| 01108 | 836440 | 1LA7 070-4YC60 | 71 | 0.40 | 0.29 | 1.05 | 1.6 | 0.80 | 1640 | 66.0 | 1.74 | 0.0006 | 1.8 | 2.8 | 4.7 |
| 01109 | 836472 | 1LA7 070-4YA60 | 71 | 0.50 | 0.37 | 1.15 | 1.9 | 0.95 | 1590 | 66.0 | 2.20 | 0.0006 | 1.3 | 2.7 | 5.5 |
| 01110 | 836460 | 1LA7 071-4YA60 | 71 | 0.60 | 0.45 | 1.05 | 2.2 | 1.10 | 1645 | 69.0 | 2.60 | 0.0008 | 1.8 | 3.4 | 6.0 |
| 01111 | 836473 | 1LA7 073-4YA60 | 71 | 0.75 | 0.56 | 1.15 | 2.9 | 1.45 | 1650 | 65.0 | 3.30 | 0.0008 | 1.9 | 3.7 | 6.0 |
| 01112 | 836490 | 1LA7 080-4YC60 | 80 | 0.90 | 0.67 | 1.05 | 3.1 | 1.55 | 1675 | 68.0 | 3.83 | 0.0015 | 2.3 | 4.4 | 8.1 |
| 01113 | 836474 | 1LA7 080-4YA60 | 80 | 1.00 | 0.75 | 1.15 | 3.5 | 1.75 | 1660 | 69.2 | 4.29 | 0.0015 | 1.9 | 3.7 | 8.1 |
| 01114 | 836401 | 1LA7 081-4YA60 | 80 | 1.20 | 0.90 | 1.05 | 4.0 | 2.00 | 1675 | 70.0 | 5.10 | 0.0018 | 2.2 | 3.7 | 9.3 |
| 01115 | 836476 | 1LA7 083-4YA60 | 80 | 1.50 | 1.12 | 1.15 | 5.0 | 2.5 | 1650 | 72.0 | 6.48 | 0.0018 | 1.8 | 3.0 | 9.3 |
| 01116 | 836402 | 1LA7 090-4YC60 | 90 | 1.80 | 1.34 | 1.05 | 6.4 | 3.2 | 1700 | 77.0 | 7.54 | 0.0028 | 2.4 | 5.2 | 11.9 |
| 01117 | 836478 | 1LA7 090-4YA60 | 90 | 2.00 | 1.50 | 1.15 | 7.0 | 3.5 | 1700 | 77.0 | 8.38 | 0.0028 | 2.2 | 4.4 | 12.1 |
| 01118 | 836403 | 1LA7 094-4YA60 | 90 | 2.40 | 1.79 | 1.05 | 7.4 | 3.7 | 1690 | 77.0 | 10.12 | 0.0035 | 2.0 | 4.5 | 14.9 |
| 01119 | 836479 | 1LA7 096-4YA60 | 90 | 3.00 | 2.20 | 1.15 | 9.6 | 4.8 | 1708 | 79.0 | 12.51 | 0.0035 | 1.8 | 3.6 | 14.9 |
| 01120 | 836404 | 1LA7 111-4YA60 | 112 | 4.00 | 3.00 | 1.15 | 13.0 | 6.5 | 1750 | 76.3 | 16.28 | 0.0048 | 2.2 | 5.6 | 27.1 |
| 01121 | 836406 | 1LA7 112-4YA60 | 112 | 5.00 | 3.73 | 1.15 | 15.8 | 7.9 | 1750 | 80.5 | 20.36 | 0.0058 | 2.3 | 6.5 | 28.7 |
| 01122 | 836407 | 1LA7 113-4YA60 | 112 | 6.60 | 4.92 | 1.05 | 19.6 | 9.8 | 1745 | 78.0 | 26.95 | 0.011 | 2.0 | 6.0 | 31.0 |
| 01123 | 836483 | 1LA7 114-4YA60 | 112 | 7.50 | 5.60 | 1.15 | 23.2 | 11.6 | 1740 | 80.0 | 30.71 | 0.011 | 2.2 | 5.6 | 32.7 |
| 01124 | 836484 | 1LA7 131-4YA70 | 132S/M | 10.00 | 7.50 | 1.15 | 28.8 | 14.4 | 1750 | 81.0 | 41.50 | 0.018 | 2.3 | 6.0 | 46.5 |
| 01125 | 836412 | 1LA7 133-4YA70 | 132S/M | 12.00 | 9.00 | 1.05 | 34.0 | 17.0 | 1750 | 81.2 | 49.50 | 0.024 | 2.5 | 6.6 | 49.0 |
| 01126 | 836485 | 1LA7 134-4YA70 | 132S/M | 15.00 | 11.2 | 1.15 | 43.0 | 21.5 | 1750 | 82.5 | 62.00 | 0.024 | 1.8 | 5.0 | 62.0 |
| 01127 | 836486 | 1LA5 164-4YB70 | 160 M/L | 20.00 | 14.9 | 1.15 | 53.0 | 26.5 | 1760 | 85.0 | 80.96 | 0.040 | 1.8 | 6.3 | 77.5 |
| 01128 | 836487 | 1LA5 167-4YC70 | 160 M/L | 25.00 | 18.7 | 1.15 | 64.0 | 32.0 | 1755 | 89.0 | 101.5 | 0.052 | 1.8 | 5.4 | 85.5 |

**Tabla 2. Valores de J para el motor.**

Según los datos del motor en la tabla el código actual a buscar es 01113 a cuyo valor corresponde un momento de inercia de:

$$J = 0{,}0015\ Kg.m^2$$

Con la programación en los dos programas antes mencionados Matlab y Lab View y con la obtención de estos parámetros está ya listo la simulación y la adquisición, y por medio de estos se podrá analizar las curvas y determinar la

[20] http://prof.usb.ve/jaller/Guia_Maq_pdf/cat_motores_ind.pdf

eficacia del programa de simulación. La simulación, la adquisición de datos y el análisis de las mismas se presenta con más profundidad en el capítulo siguiente.

# CAPÍTULO IV

# 4. ANÁLISIS DE RESULTADOS

## 4.1 INTRODUCCIÓN

En este capítulo se analizará las bondades del programa con respecto a los resultados obtenidos del motor Siemens. Cabe mencionar que las pruebas realizadas en el laboratorio fueron dos: la de arranque directo del motor y el de frenado del motor por inversión de fases, las demás pruebas como frenado por inyección de corriente continua y fallas se han omitido, por lo que la comparación solo se lo hará con lo anteriormente mencionado.

Para el motor que se ha hecho las pruebas se simulará todos los casos haciendo variantes de carga para poder demostrar las ventajas del software aunque no se puedan comparar con los datos del motor, en base al arranque directo y frenado por inversión de fases se podrá determinar si las demás simulaciones son o no validas.

## 4.2.- ANÁLISIS DEL ARRANQUE DIRECTO DEL MOTOR

Para el análisis primero se tomará la forma de onda obtenida del motor Siemens de 1 HP, 220 V, conexión en Y, y que para este caso estuvo trabajando en vacío. El procedimiento para la obtención de esta forma de onda es el de conectar el motor a su respectiva alimentación, y para la adquisición el

de conectar el motor al módulo “Transitorios Motor”, en forma general la obtención de la curva de corriente se lo hace a través de un transformador de corriente como se lo explicó en el capítulo anterior. El diagrama de conexión para obtener las formas de onda se encuentran en el anexo F.

El motor para esta prueba se lo conecto en Y, alimentado con un voltaje trifásico balanceado de 220 V. Obteniéndose para el Arranque Directo sin carga la siguiente forma de onda:

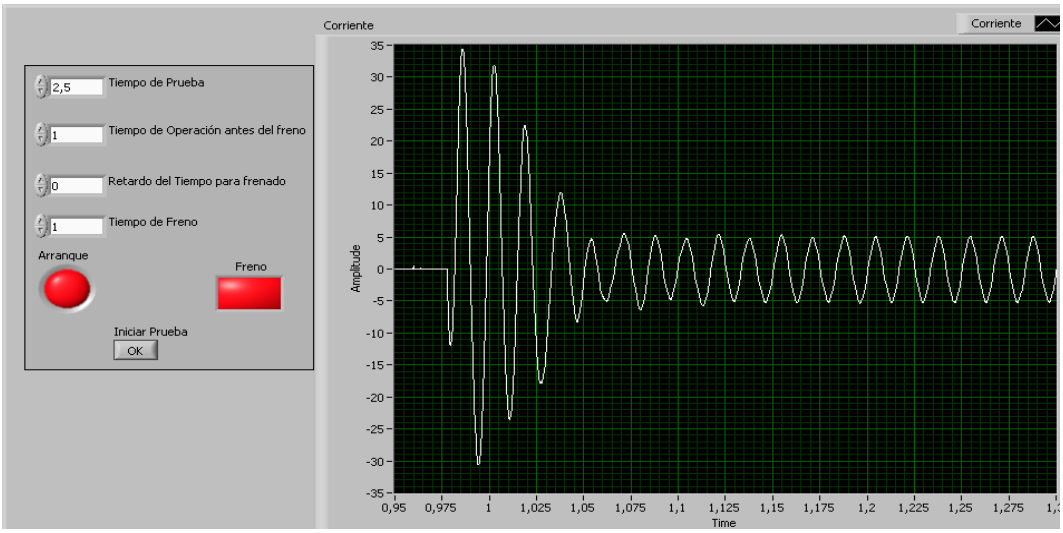



**Figura 4.1 Corriente de arranque en estado transitorio**

Al analizar esta forma de onda se puede apreciar que el estado transitorio es muy corto y se genera con cuatro picos de corriente, cuyo primer pico de corriente alcanza los 35 A y va disminuyendo hasta alcanzar el estado estable en donde se estabiliza en 5.2 A esto quiere decir que en este proceso se produjo que la corriente de arranque sea de 6.73 veces la corriente de régimen. Cabe recalcar que el ángulo de arranque o el desfase que existe entre la forma de onda de voltaje y corriente en el momento mismo del arranque produce ciertos cambios en la magnitud y forma de onda de la corriente, en este caso no se puede saber el ángulo pues como se ve la única variable que se tiene es la corriente.

Después de obtener la forma de onda con sus valores del motor Siemens se procede a realizar la simulación en base al software elaborado en Matlab

tratando en lo posible de simular los mismos valores, las mismas condiciones, y los mismos tiempos, es así que en la siguiente figura se observa la pantalla de ingreso de datos los cuales se encuentran con los valores por defecto que se utilizará para la simulación del motor Siemens así:

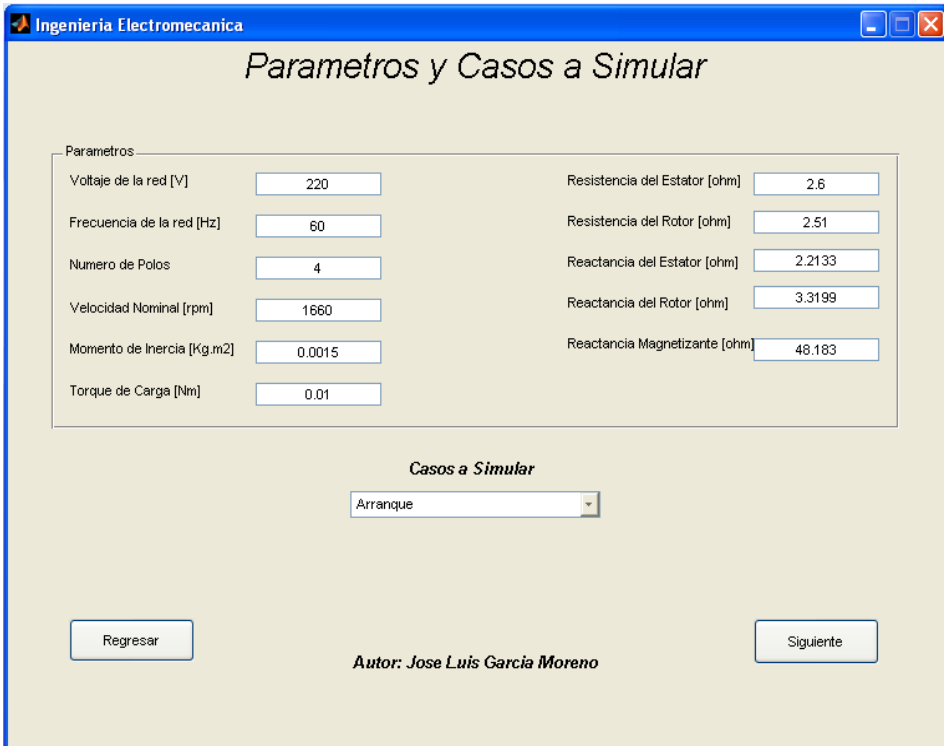



**Figura 4.2 Pantalla de ingreso de datos generales para la simulación del arranque**

En la figura 4.2 se puede apreciar los datos ingresados para la realización de la simulación en donde se encuentran los datos generales que simulan el comportamiento que tuvo el motor para la adquisición de datos analizado.

Es así que el motor bajo estas condiciones siempre se conectará a una red trifásica balanceada y esta es de 220 V según la condición dada en el casillero de Voltaje de la Red, así también se mencionó que el motor se encontraba sin carga en el momento de la adquisición, esto también lo llevamos al software de simulación al ubicar en el casillero de torque de carga el valor de 0.01 que es un valor cercano a cero como ya se dijo anteriormente debido a las condiciones propias de rozamiento entre las partes entonces este en un valor referencial para simular dicho fenómeno.

Los demás valores que se encuentran en esta pantalla únicamente corresponden a las características propias de la red y el motor como son: la frecuencia de la red, número de polos del motor, velocidad nominal del motor, y sus parámetros como: son resistencia y reactancia del estator, resistencia y reactancia del rotor referida al estator, reactancia magnetizante e inercia del motor, los demás valores son condiciones impuestas por el usuario según lo que se desee simular y sus condiciones. La forma de manejar este programa se lo verá con mayor detalle en el Manual del usuario en el anexo G.

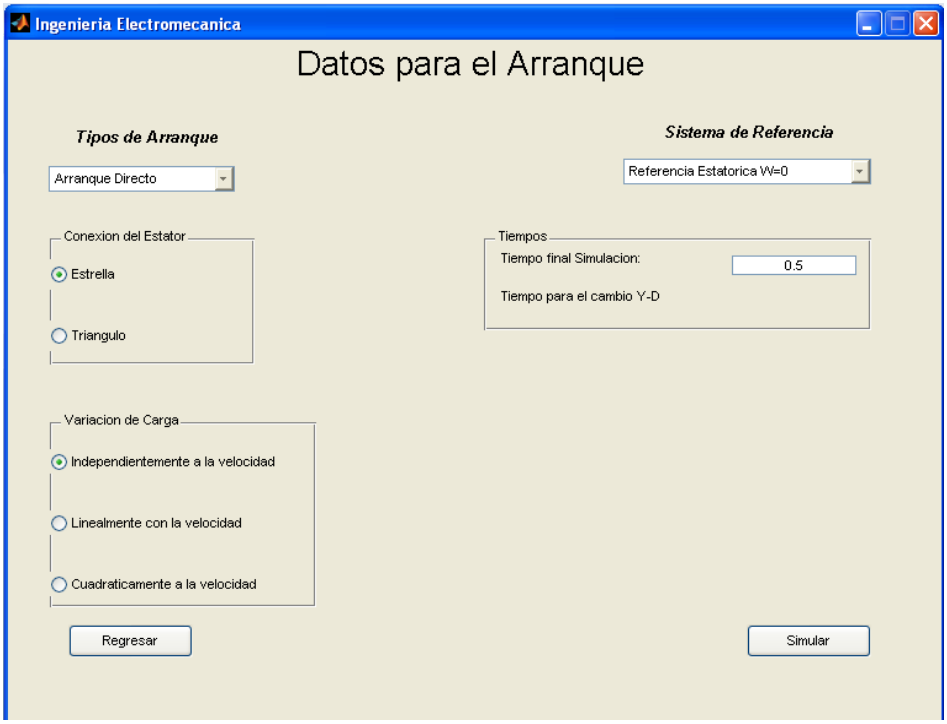



**Figura 4.3 Pantalla para el ingreso de datos correspondiente al arranque.**

En la pantalla anterior se observa las condiciones necesarias para la simulación expresa del arranque y sus tiempos de operación. La pantalla (Datos para el arranque) determina todas las condiciones necesarias para generar un arranque directo, es así que en los casilleros de selección se elegirá en el tipo de arranque el correspondiente a este caso que es el arranque directo, la conexión del estator en este caso según el motor solo permite una conexión en estrella y esa opción se elegirá, en la pantalla anterior se eligió que el torque de carga es nulo razón por lo cual la elección de variación de

carga resulta intrascendente por lo que se dejará en la primera opción que es la que varia independientemente a la velocidad, aquí vienen dos parámetros importantes, el primero es determinar el sistema de referencia en el que se desea simular y por ser un arranque directo se lo hará en el sistema de referencia estacionario (W=0), y por último los tiempos en este caso al no cambiar de condición que es un arranque directo entonces únicamente se plantea el tiempo en el que se termina la simulación que es de 0.5 seg, este valor se toma en base a la adquisición de datos en donde se observa que se estabiliza en 0.15 segundos y al terminar en 0.5 segundos se puede apreciar el transitorio y también su estado estable. De esta manera se simula el arranque directo con alimentación trifásica balanceada y por supuesto sin carga obteniéndose las corrientes en las fases a, b y c como se muestra en la figura 4.4.

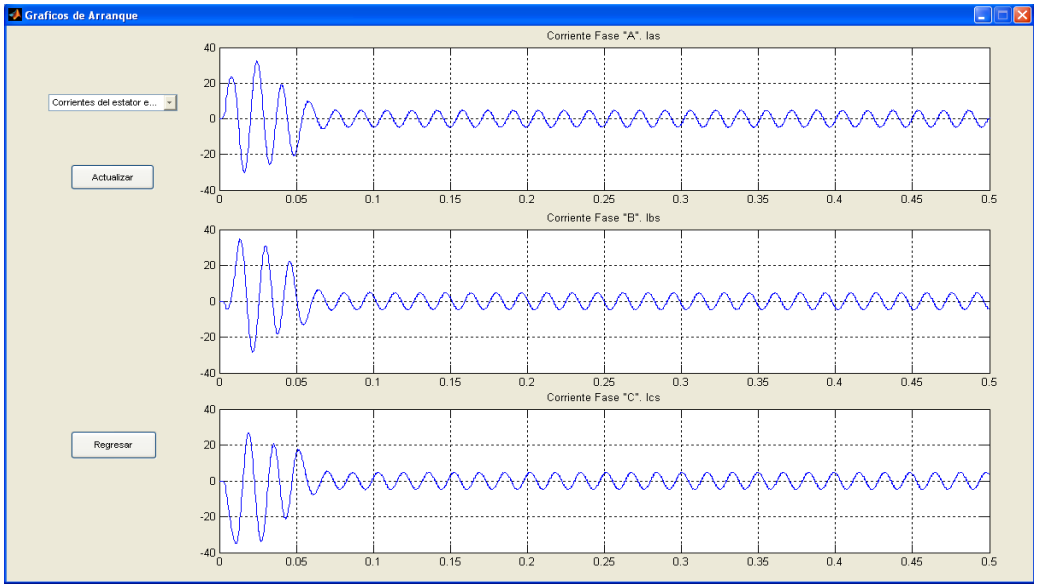



**Figura 4.4 Corrientes de arranque en las fases a, b y c**

En el gráfico anterior se puede apreciar las tres corrientes en las fases a, b y c, sin embargo a pesar de que la adquisición se obtuvo en la fase “a” de la fuente de alimentación aquí la corriente que más se asemeja a la obtenida en la adquisición de datos es la de la fase “b”, esto se debe a que el instante inicial de adquisición no coincide con el cero, razón por lo cual se hará el análisis en base a la fase “b”.

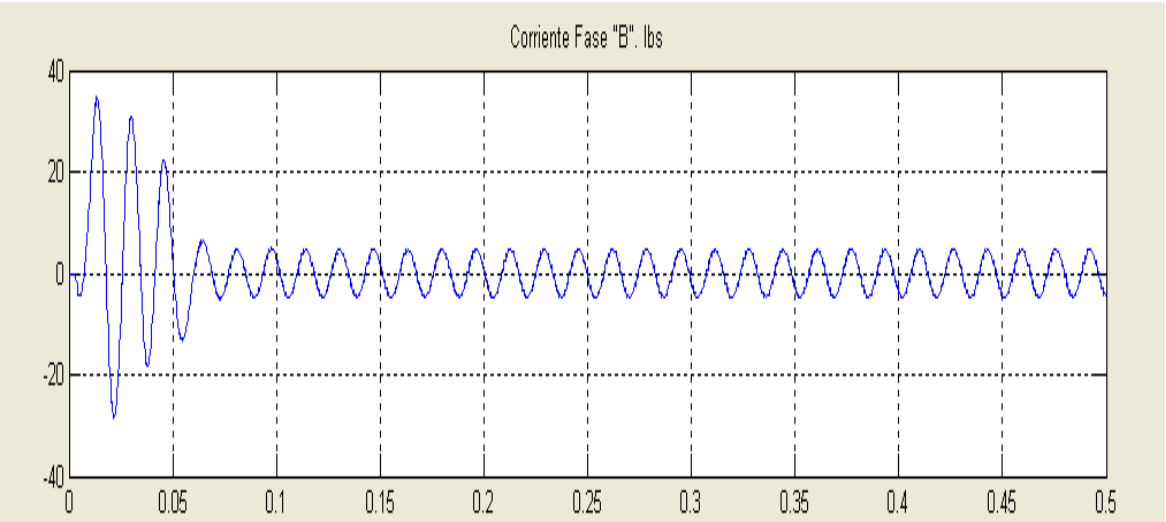



**Figura 4.5 Corriente en la fase “b” de la simulación**

En esta figura se puede apreciar que al igual que en la adquisición el arranque se produce con cuatro picos de corriente y tomando en cuenta que la envolvente es similar a la de la adquisición, ahora queda por determinar si las magnitudes corresponden de igual forma o semejante, en la figura no se puede apreciar exactamente el valor de pico de corriente pero si se puede determinar en el archivo que se genera de realizar la simulación y que almacena todos los datos logrando determinar que el valor pico alcanzado es de 34.93866 A que se asemeja en gran medida a los 35 A que se obtuvo de la adquisición de datos, ahora queda por determinar el valor en estado estable que alcanza un promedio de 4.91605 A de igual forma muy semejante a los 5.2 A y determinándose de esta forma que la corriente de arranque en la simulación es de 7.10 veces la corriente en estado estable lo que se acerca a los 6.73 logrado en la adquisición lo que genera un error de apenas 5.22% esto ya en forma general, pero se observa que esto se debe a la parte de estado estable en donde se genera un error mayor que en el de estado transitorio en donde se puede apreciar que el error es apenas del 0.8%.

Esto determina que para el arranque directo especialmente en estado transitorio la simulación es de muy alta eficacia con los datos obtenidos del motor, razón por lo cual se analizará las otras variables que el software permite

apreciar como son las corrientes del rotor, torques y velocidades en el arranque directo.

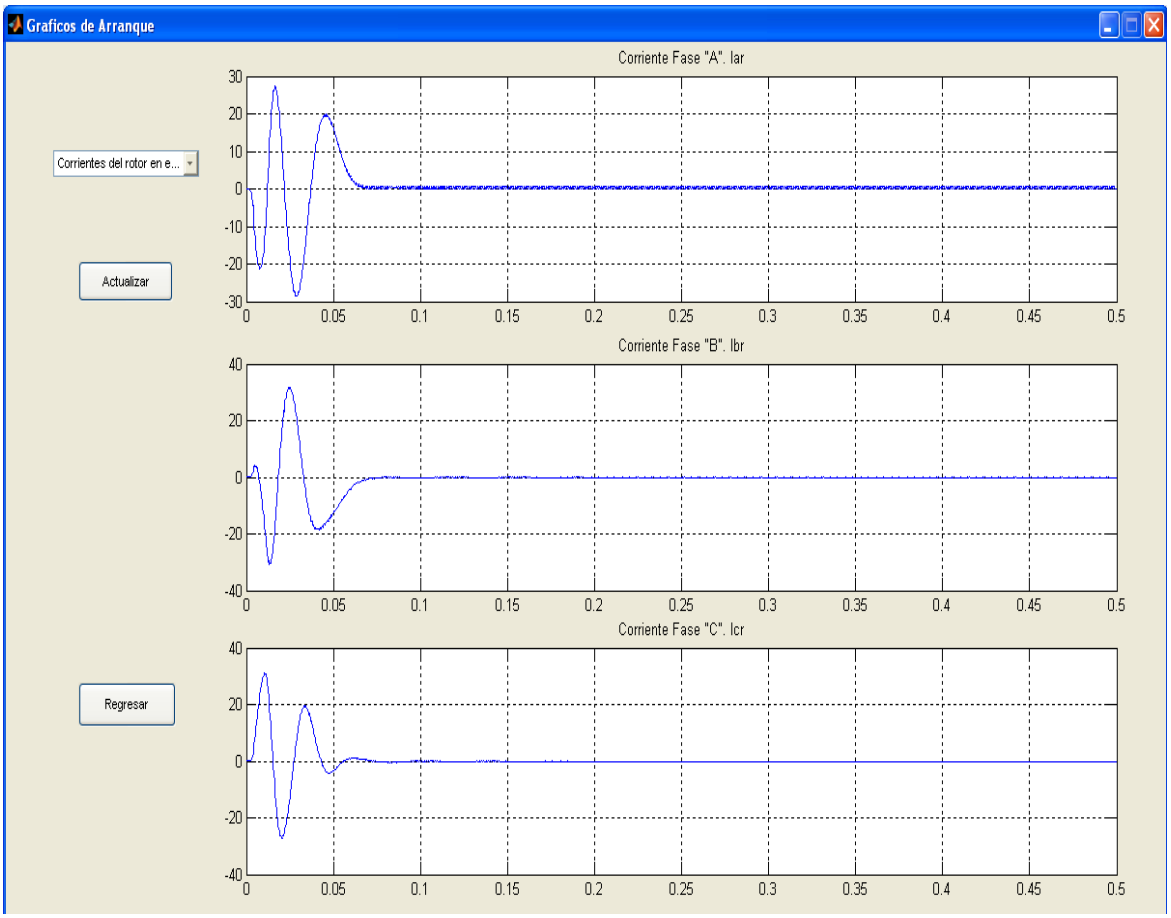



**Figura 4.6 Corrientes rotóricas**

En la figura 4.6 se observa las corrientes que se generan en el rotor, esta es una de las características potenciales que tiene esta simulación ya que se pueden apreciar ciertas variables que en la práctica son difíciles o quizás imposibles de apreciar. Aquí se ve que las corrientes en los tres ejes empiezan con una frecuencia inicial igual a la de la frecuencia de la red y que luego irá disminuyendo conforme se acerca el motor a su estado estable, esto también sucede con el pico de corriente que alcanza casi los 30 A y que va disminuyendo cuando se alcanza el estado estable, además que los picos de corrientes son apenas dos hasta alcanzar el estado estable.

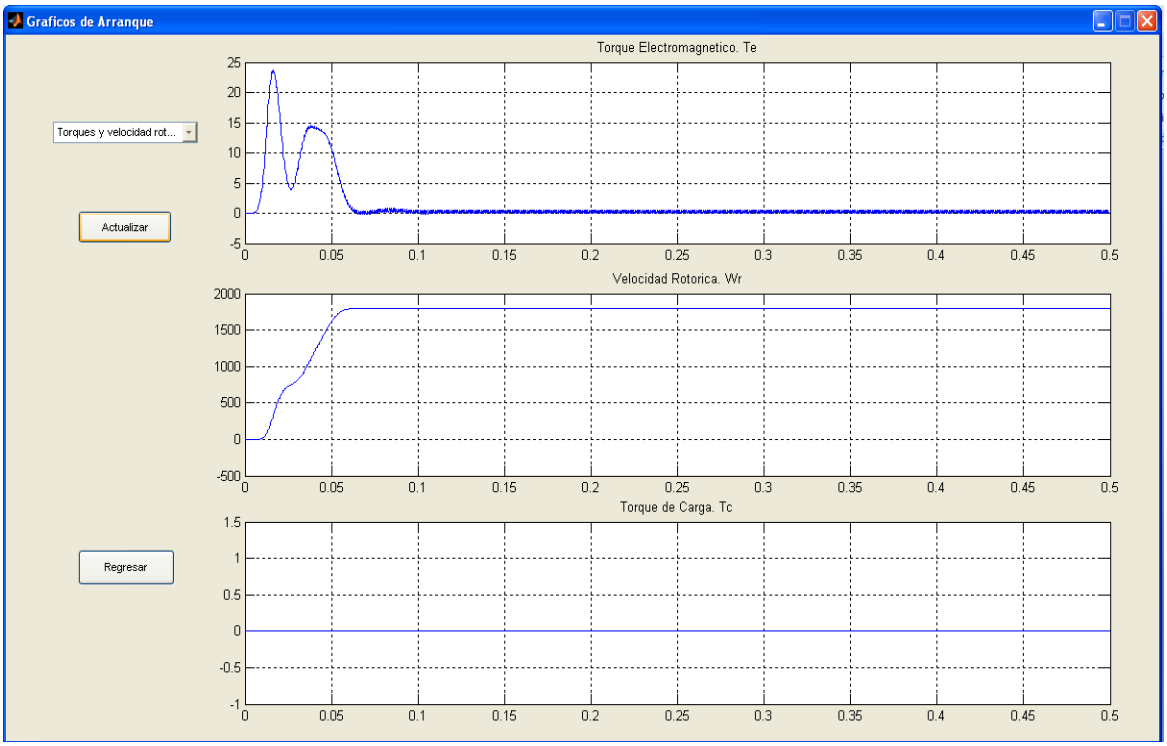



**Figura 4.7 Torque electromagnético, velocidad del motor y torque de carga.**

En la figura 4.7 se puede apreciar en primer lugar el torque electromagnético que se genera en el arranque del motor y que al iniciar en cero no se produce un pico negativo lo que hace deducir que se produce un arranque limpio, suave sin mucha vibración del mismo eso se debe a que el motor se arranco en vacío y que además el motor tiene un momento de inercia bajo lo que quiere decir que no tiene mucha oposición a salir del reposo. El pico de torque se genera cercanamente a los 25 Nm que luego desciende hasta casi 4 Nm para nuevamente subir todo esto en función de las corrientes generan para luego estabilizarse en el valor de 0.01 que es el valor del torque en vacío.

En segundo lugar se observa la velocidad rotórica o velocidad del motor, quizá sea esta variable la que dará una visión más amplia de cómo se genera el arranque en el motor, se nota que el arranque se va generando de una forma casi lineal con pequeñas imperfecciones debido a las corrientes que se generan en ese pequeño espacio de tiempo, realmente el tiempo de estabilización de la velocidad es muy pequeño que es de aproximadamente 0.1 seg, que son los necesarios para alcanzar la velocidad de 1800 rpm.

Por último debido a que el torque es de apenas 0.01 Nm (en vacío), se puede apreciar que es una línea recta constante, esto se debe a que en las condiciones se seleccionó la condición que el torque de carga es independiente a la velocidad lo que ocurre en los mecanismos de avance de las máquinas-herramientas. Estas características de carga se analizarán después con mayor detalle en las simulaciones a realizar.

## 4.3.- ANÁLISIS DEL FRENO POR INVERSIÓN DE FASES DEL MOTOR

Para lograr la adquisición de datos del frenado de un motor por inversión de fases una vez más se hará uso del software Lab View que a través del módulo "Transitorios Motor" permitirá adquirir esta señal transformada en una onda de corriente que permitirá el analizar, comparar y determinar sus bondades como se lo hizo anteriormente en el caso del arranque directo del motor.

Las condiciones a utilizarse será nuevamente que el motor se alimente con un voltaje trifásico balanceado en Y de 220 V, que trabaje sin carga y que el freno se dé cuando ya se encuentre en estado estable, todas estas características luego se tendrán que simular en el software de simulación. El procedimiento para la obtención de esta forma de onda es similar al realizado para la obtención de la forma de onda de corriente en el arranque directo del motor es decir se lo hace al conectar el motor a su respectiva alimentación como también es necesario del módulo "Transitorios Motor" para lograr adquirir la forma de onda de corriente que se genera durante el frenado. De igual forma que en el arranque directo la obtención de la corriente del motor se lo hace a través de un transformador de corriente. El diagrama de conexión se encuentra en el anexo F.

Determinadas todas las características con las que estará expuesto el arranque y posterior frenado del motor se procede a realizar la respectiva adquisición en donde en el software de Lab View se ingresa un tiempo al que se genera el frenado que es de 1 segundo, esto debido a que si se lo hace antes quizá no

logre estabilizarse y pueda surgir datos erróneos, lo que perjudicaría el análisis respectivo, en la simulación este tiempo se puede acortar. Las curvas obtenidas se muestran en la figura siguiente.

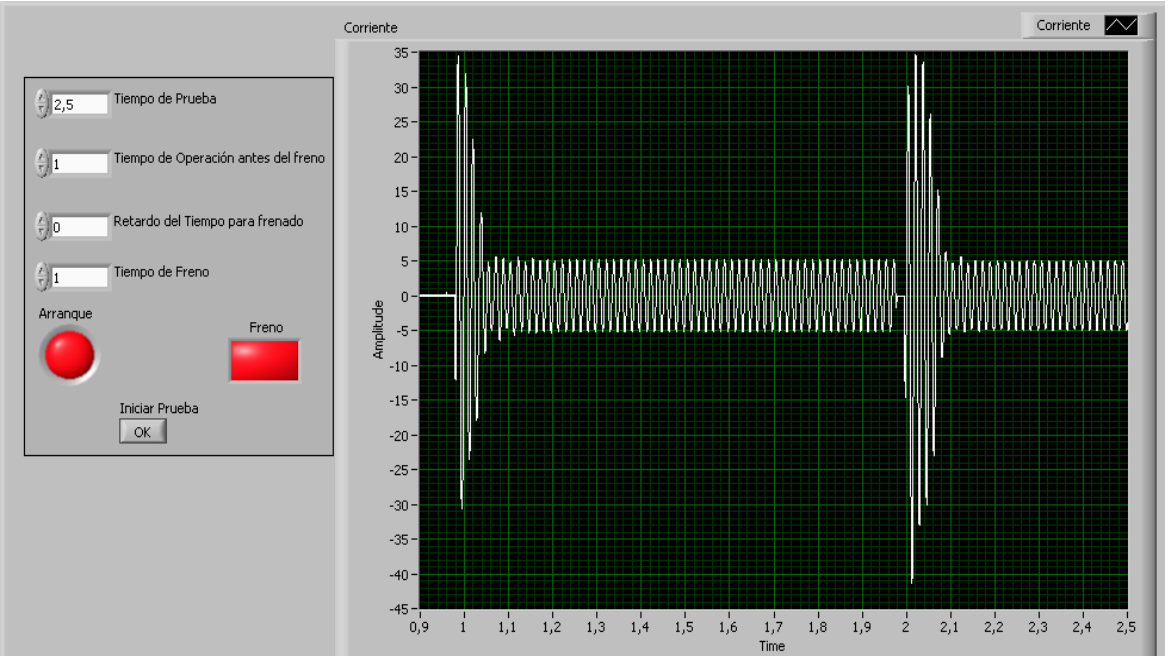
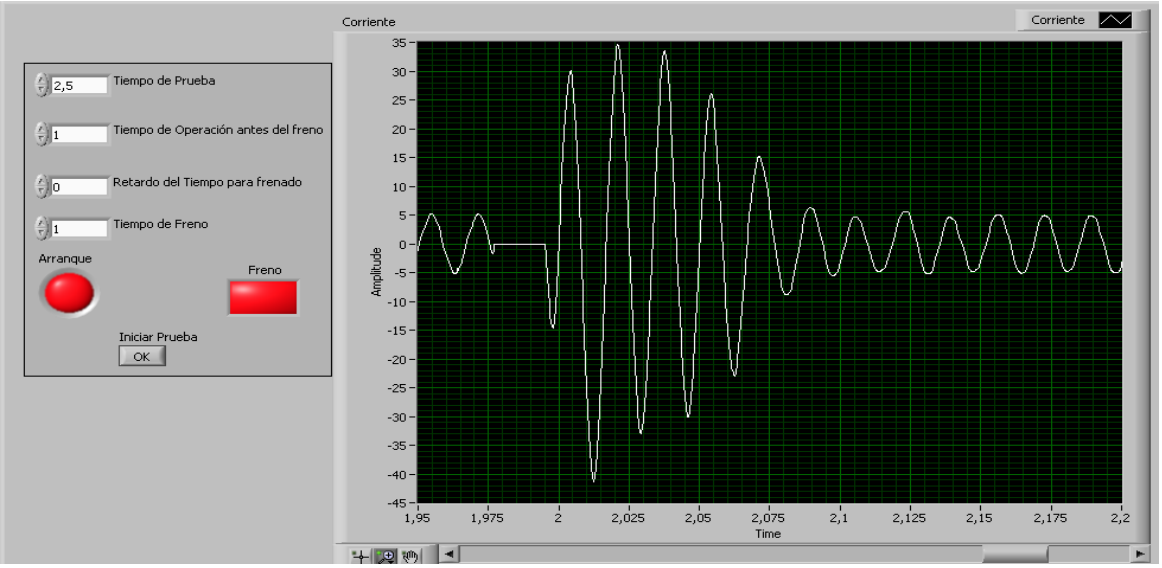



**Figura 4.8 Corriente de arranque y freno por inversión de fases en estado transitorio**

En la figura 4.8 se puede apreciar el arranque y posterior inversión de fases que provocará el frenado del motor, en este caso no se volverá a analizar el arranque del motor debido a que las condiciones son las mismas que para el arranque directo del motor y también que la forma de onda es similar en lo que al arranque se refiere por lo que lo correcto será analizar la parte donde empieza el frenado por inversión de fases.

En la figura 4.9 se puede notar que después del estado estable en el que entra el motor después del arranque se genera la inversión de fases provocándose un pequeño espacio de tiempo en donde la corriente se hace cero, esto se debe a que si bien en el software se pone un tiempo de espera de 0 segundos para el cambio de fases este no se da debido a que en la transición en los contactores, estos tienen partes mecánicas que de por sí ya generan un retardo que es el que se aprecia en la figura siguiente, este es un valor que no se puede omitir como ya se dijo por las características mecánicas de los

contactores que permiten el accionamiento y posterior inversión de fases para el respectivo frenado del motor. A continuación la mencionada figura y el respectivo análisis de la forma de onda y su comparación con la simulación.




**4.9 Corriente en el momento de la inversión de fases para el frenado del motor.**

La forma de onda como se puede notar genera cinco picos de corriente y luego se estabiliza nuevamente, quizá aquí no se pueda apreciar el frenado del motor más bien solamente lo que pasa con la corriente en el instante del frenado, esto si se lograr ver en la simulación cuando se analice la velocidad, sin embargo se analiza esta forma de onda en donde los picos positivos llegan nuevamente a los 35 A mientras que existe un pico negativo que llega a los -42 A , mientras que al estabilizarse la corriente lo hace en la misma magnitud que la del arranque que es de 5.2 A. El hecho de que exista más picos de corriente y que la magnitud haya aumentado en relación al arranque se debe a que el motor se encuentra girando ya en un sentido y al invertir sus fases el motor debe girar para el otro lado pero para eso deberá vencer la inercia que ya se generó en un sentido esto hace que el motor se forcé más para lograr la inversión y esto se visualiza en la corriente que se analiza en la grafica anterior y que posteriormente se verá en las demás variables en la simulación.

Una vez analizada la forma de onda lo que queda es comparar con la simulación que se genera del software elaborado en Matlab, nuevamente como en el arranque directo lo que se debe hacer es  al ingresar los datos del motor tratar en lo posible de generar las mismas condiciones que se dan en el motor real, en este caso la pantalla principal será similar a la dada en el arranque directo con una diferencia que se hará notar en la siguiente figura.

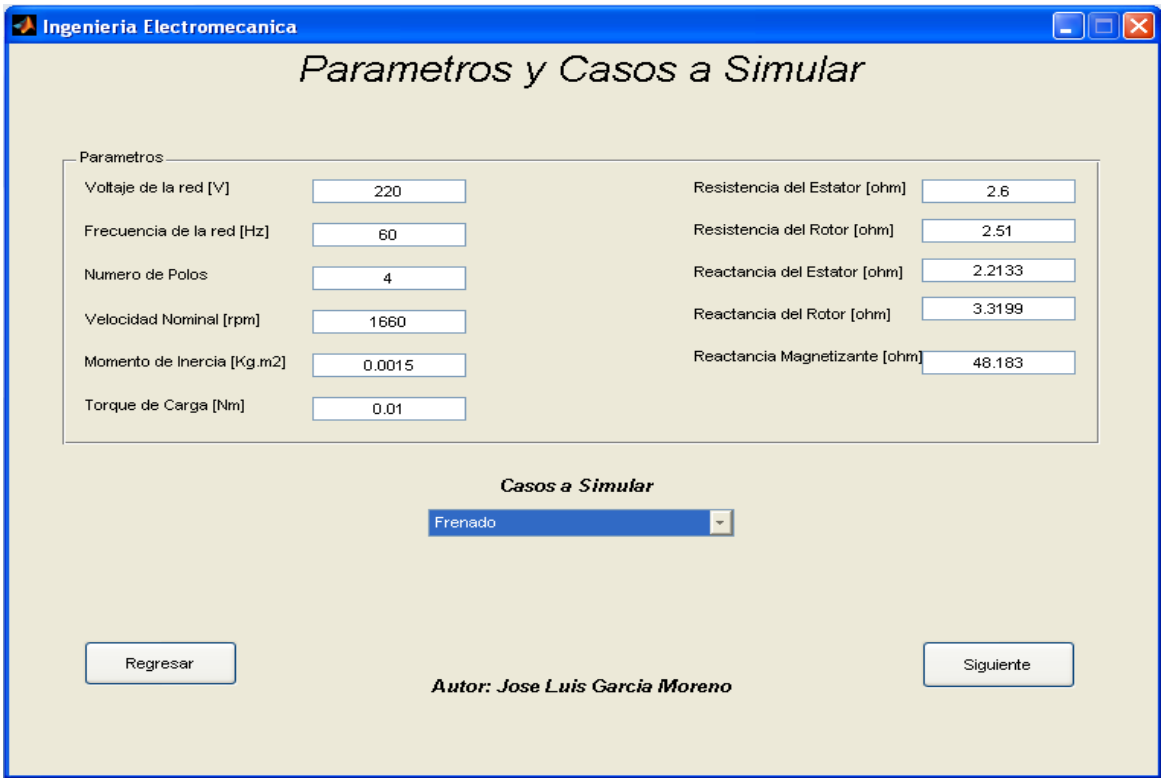



**Figura 4.10 Pantalla de ingreso de datos generales para la simulación del frenado**

En la pantalla anterior se puede ver todos los parámetros que ya se analizó anteriormente y que son exactamente son iguales, lo único que cambia es que en la opción de casos a simular que se encuentra de color azul se ubica la opción de frenado, el cual llevará a la pantalla donde si se cambiará algunas condiciones por lo que será de imperiosa necesidad analizarla.

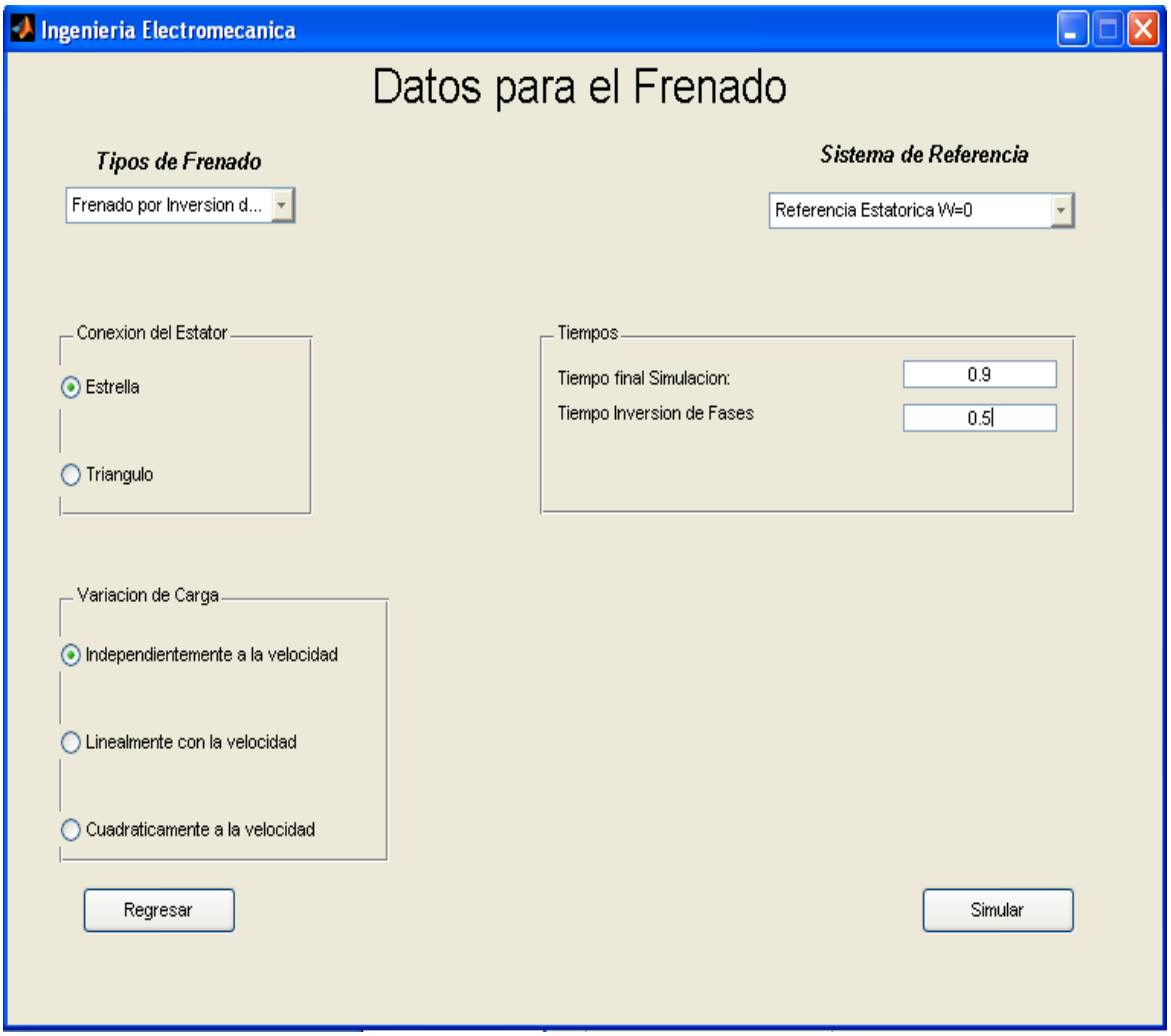



**Figura 4.11 Pantalla para el ingreso de datos correspondiente al frenado**

En el selector del tipo de frenado se elegirá el frenado por inversión de fases que es el caso que se desea simular, cosa similar que en la caso de arranque directo se lo hará con el estator en estrella y la variación de carga será igual, esto no afectara a la condición de trabajar sin carga por lo que se le dejará en la misma opción que es independientemente a la velocidad, en lo que se refiere al sistema de referencia a simular se elegirá el sistema estatórico (W=0) por ser en el estator donde se generan los cambios ósea la inversión de fases. Por último los tiempos aquí son importantes y no será similar a la adquisición de datos puesto que en el motor real se necesita que se estabilice, el programa no porque no analiza por ejemplo la saturación magnética o histéresis del hierro, por lo que únicamente se elige el momento del cambio de fases y lo hace inmediatamente, provocándose desde ya una diferencia puesto que en el motor

real se genera un tiempo muerto lo que no sucede aquí ya que es inmediato lo que puede verse en la simulación como un cambio brusco, al ver que el motor entra en estado estable en 0.15 segundos como máximo es más que conveniente provocar la inversión de fases a los 0.5 segundos como se ve en la figura 4.11. El resultado obtenido de la simulación de arranque y frenado se muestra en las figuras 4.12 y 4.13.

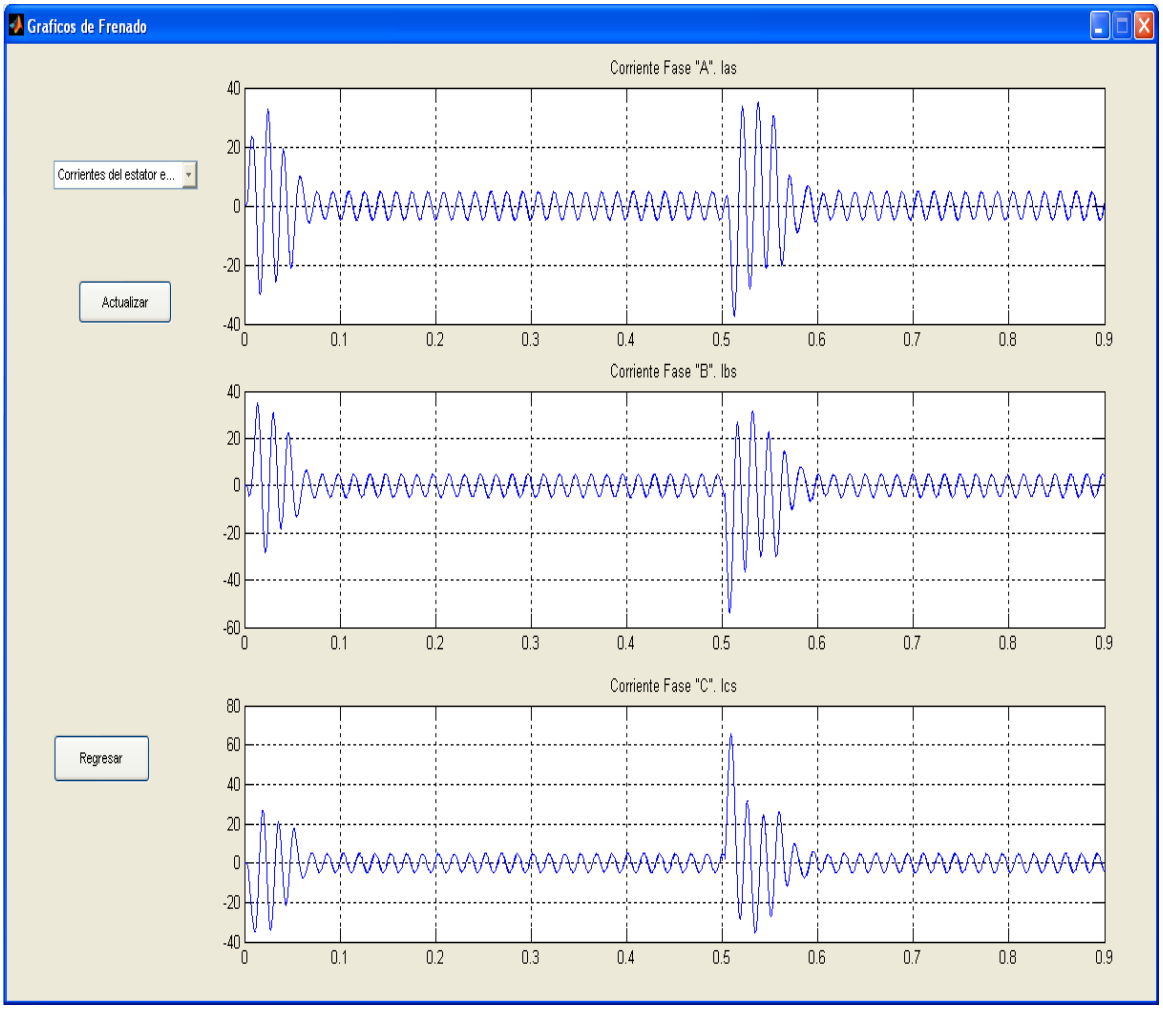



**Figura 4.12 Corrientes de arranque y frenado en las fases a, b y c**

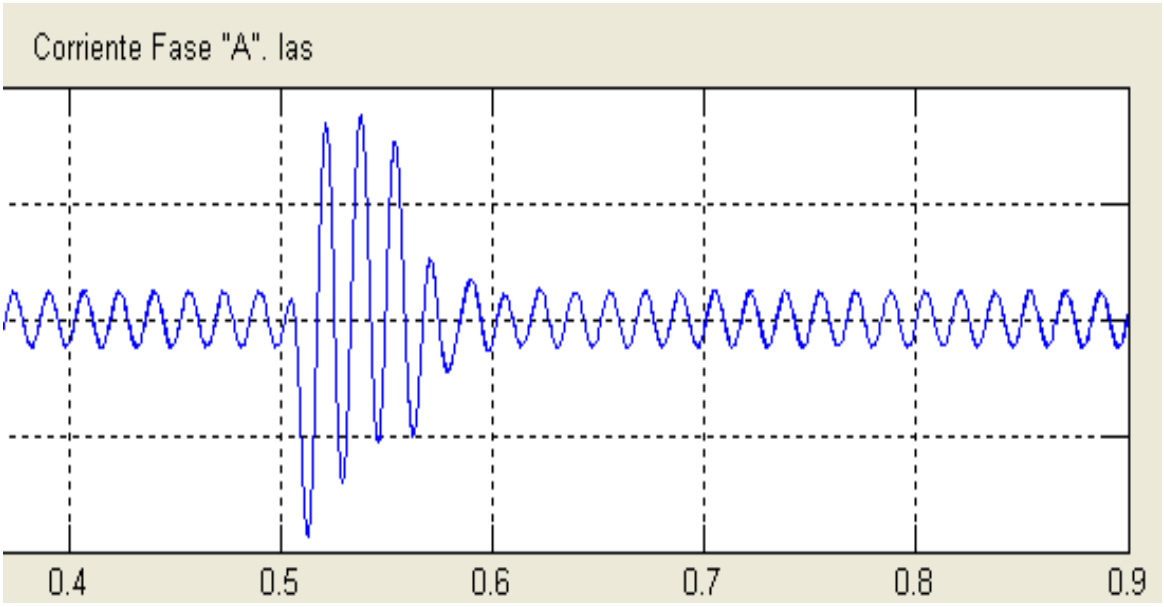



**Figura 4.13 Corriente en la fase "a" del frenado**.

En las figura 4. 12 se observa las corrientes en los ejes a, b y c como se generan en su totalidad en el momento de la inversión de fases y de manera más concreta y para poder determinar la eficacia del programa se analizara la forma de onda de la figura 4.13, en donde se puede ver que visualmente es similar a la generada en la adquisición de datos en donde igual se generan cinco picos de corriente, notándose claramente que aquí no existe ese tiempo muerto que si se puede apreciar en la adquisición de datos, el valor pico positivo de esta onda es de 34.9253 A lo que es muy cercano a los 35 A que se registro en la adquisición y en cuanto a los valores negativos la simulación alcanzo los -37.52325 A. a diferencia de los 42 que se registra en el programa Lab View, en cuanto al estado estable, este generó los mismos valores que se recogió del arranque directo, es así que el porcentaje de error es mayor que en el caso anterior y esto se debe al ángulo de conexión y reconexión que se genera al invertir las fases lo que ha hecho que se dispare el pico negativo pero si se mira las otras fases este pico también se genera en las otras fases pero aunque las formas de onda no necesariamente se asemejen, esto da cuenta que nuevamente que los datos obtenidos del software de simulación es muy similar a los obtenidos en la adquisición de datos, variando siempre los ángulos que no se puede saber cuáles son y también en menor grado de la saturación

magnética que se genera en el motor real y que no se han simulado en este proyecto.

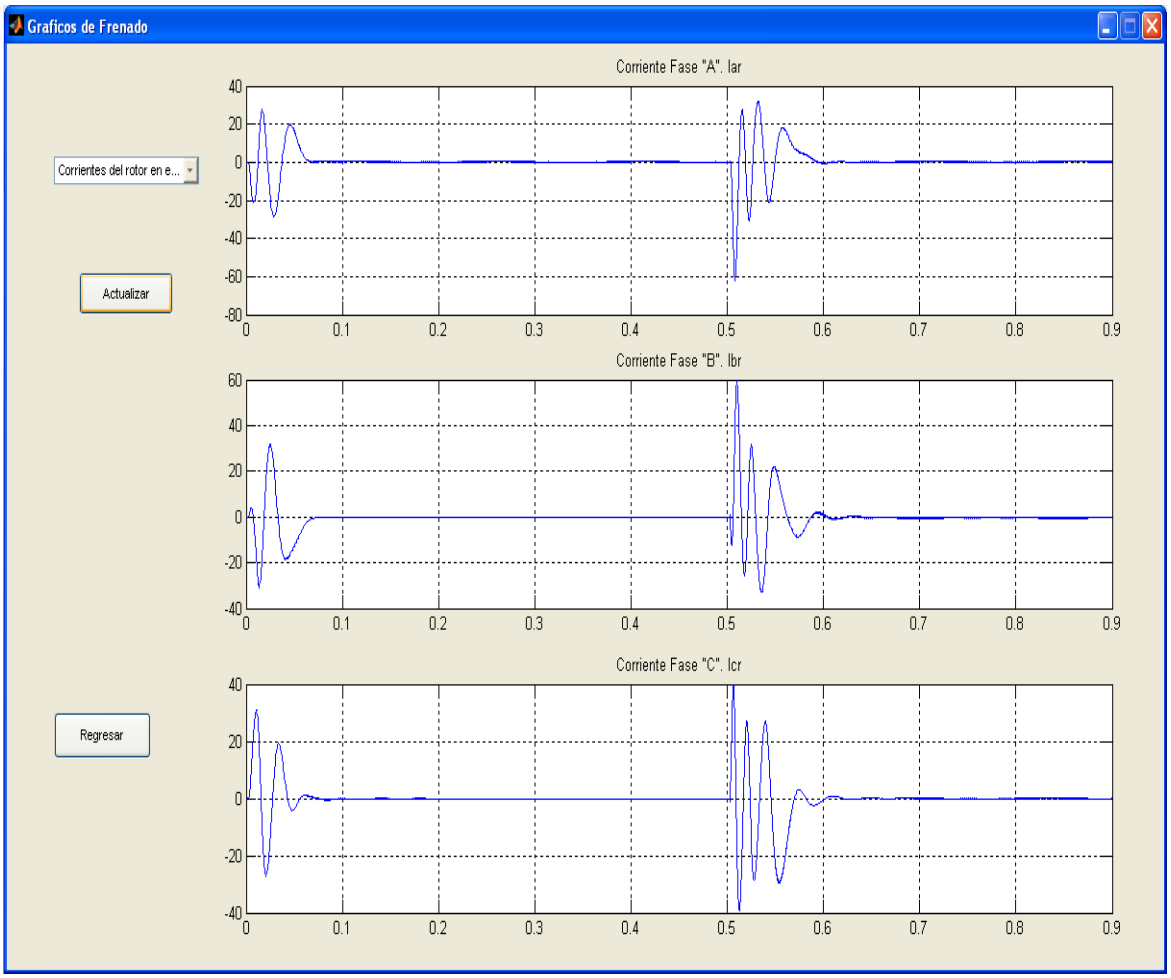



**Figura 4.14 Corrientes rotóricas**

En la figura 4.14 se observa las corrientes en el rotor que tienen una característica similar a las producidas en el estator en donde se ve que el pico de corriente llega a dispararse en el momento de la inversión al igual que también aumenta los picos de corriente en relación a los del arranque, en cuanto a la frecuencia de la forma de onda es igual a la de la red en el momento exacto de inversión de fases para luego ir disminuyendo conforme se vuelve a estabilizar cuando el motor ha invertido su sentido de giro, todos estos valores nos dan una idea de lo que pasa en el frenado pero nada mejor que analizar lo que pasa con el torque y especialmente la velocidad que se analizará en base a la figura siguiente.

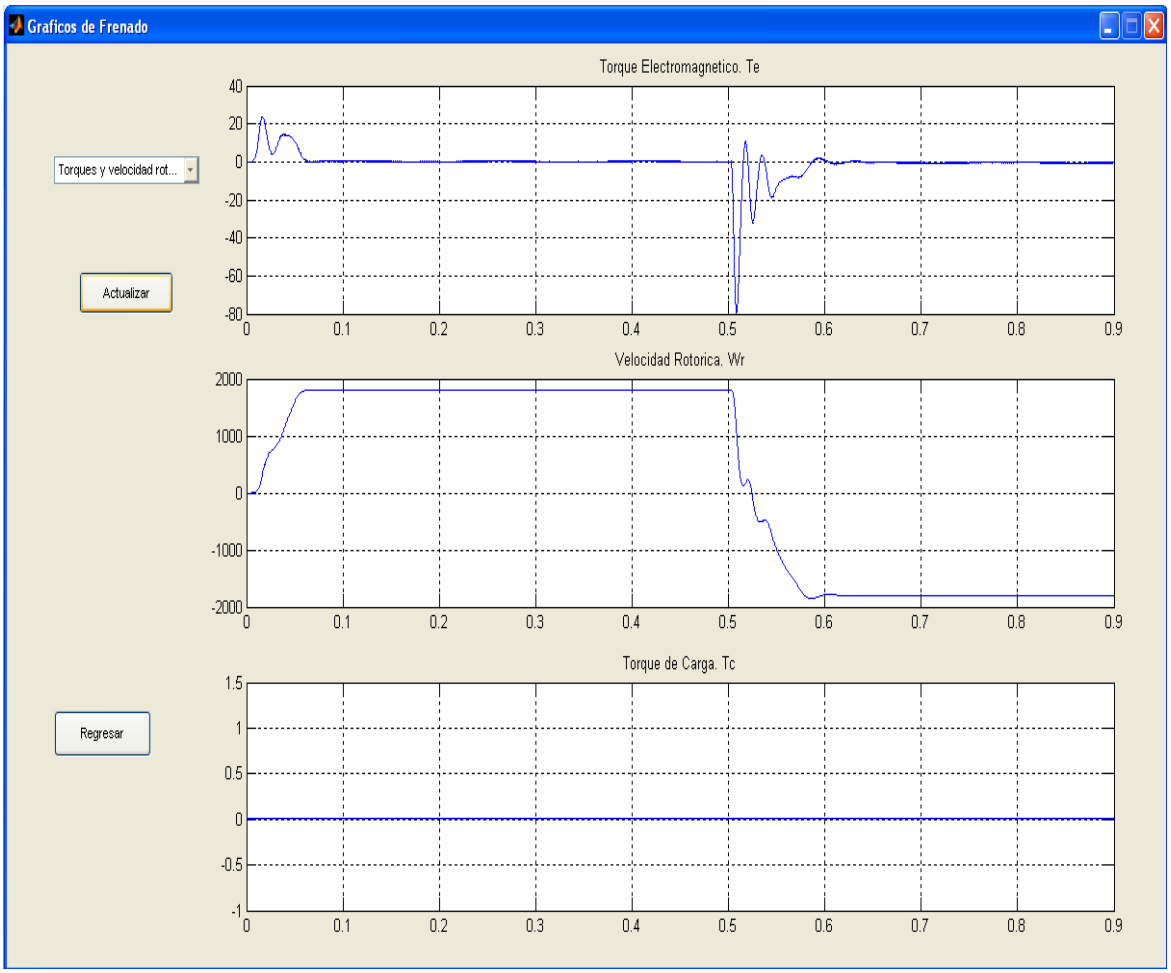



**Figura 4.15 Torque electromagnético, velocidad del motor y torque de carga.**

En el primer gráfico de la figura 4.15 se tiene el torque electromagnético generado por el motor y se puede apreciar que el arranque es un tanto suave y que después este se hace cero o cercano, y donde realmente interesa el análisis que es en 0.5 segundos se ve que el torque se vuelve negativo y de gran valor llegando incluso a ser de tres a cuatro veces el del arranque y de sentido contrario, en la situación real del motor se puede sentir como al invertir el sentido de giro del motor se tiene una contra fuerza, debido al gran esfuerzo que hace para detener el motor e invertir su giro, esto se vera de mejor manera en el segundo gráfico donde se ve la variable de velocidad aquí se ve que después de que se encuentra en estado estable es decir a una velocidad constante de 1800 rpm al momento de producirse la inversión en el tiempo 0.5 segundos se ve que la velocidad decae muy rápidamente incluso más rápido que el tiempo que toma para llegar a su velocidad nominal en el momento del

arranque. También se puede observar que no es un decaimiento de velocidad muy suave más bien se le ve brusco y esto en la práctica resulta ser real debido a su golpeteo que ya se explico anteriormente, cuando cruza por el eje del tiempo empieza a tener valores negativos que alcanzan el valor de -1800 rpm aproximadamente para luego estabilizarse lo que es entendido como que hasta tocar la curva con el eje del tiempo el motor se para pero de ahí en adelante sigue con los voltajes aplicados lo que provoca que el motor gire en sentido contrario al arranque. Como se ve este frenado específicamente en este motor es muy rápido, este tipo de simulaciones permite saber el tiempo que toma solo para detenerse y así poder diseñar el sistema de frenado optimo para que frene y se detenga. Por último se tiene nuevamente el valor de 0.01 que corresponde al valor de carga que no varía en función de la velocidad y que no representa ningún cambio en las variables estudiadas.

Estudiados y analizados estos dos casos que corresponden a lo que se pudo hacer en el Laboratorio de Máquinas Eléctricas se ha simulado y determinado en base a sus valores y formas de onda que existe un alto porcentaje de efectividad y que hace al software conveniente para realizar cualquier simulación, el hecho de que no se analice la saturación del núcleo y demás factores que se desprecian en esta tesis y que si forman parte de un motor real hace que existan estos errores, que como se ha analizado no interfieren en gran magnitud en los resultados obtenidos por lo que se considera aceptable la modelación y simulación que se ha realizado.

## 4.4.- SIMULACIONES

Al haber verificado que el software tiene un alto de grado de confiabilidad con lo obtenido experimentalmente hace que se pueda confiar en él en base al análisis que se realizó de las dos operaciones como son arranque directo y frenado por inversión de fases, lo que en esta parte y debido a que ya no se pueden hacer más pruebas en el motor se limitará a realizar las simulaciones que corresponden al motor que se está analizando, sin cambiar su estructura general únicamente se cambia factores como: carga, la inercia de la máquina,

otras condiciones como frenado por corriente continua y fallas, esto sin cambiar el voltaje aplicado que seguirá siendo de 220 voltios y tampoco la conexión del estator que seguirá siendo en estrella, excepto para la simulación de un arranque estrella - triángulo. Serán una serie de simulaciones que permitirán ver aun más las herramientas que posee el software y que hasta aquí no se ha visto, así que se empezará con la pantalla Parámetros y Casos a Simular del programa diseñado en Matlab.

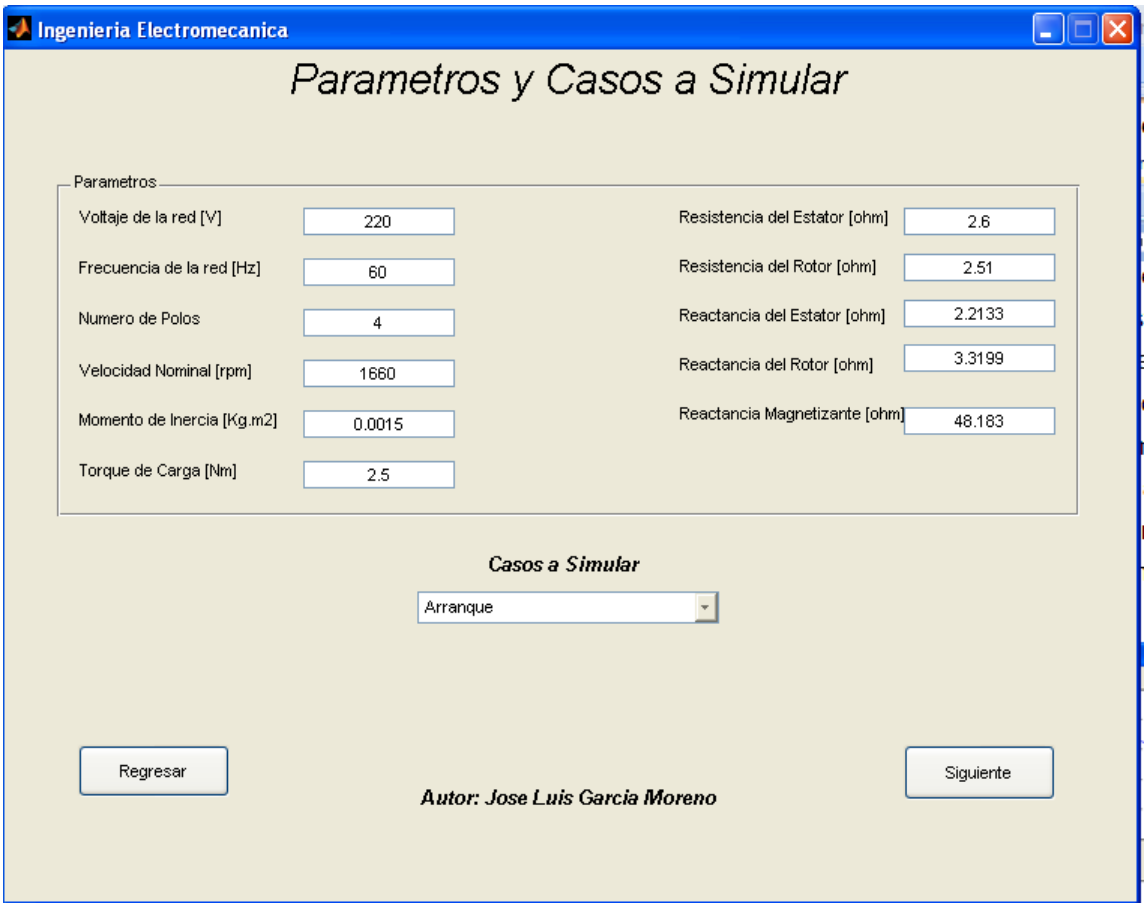



**Figura 4.16 Pantalla de ingreso de datos generales para la simulación del arranque**

En la figura 4.16 se ingresan los datos para la respectiva simulación, que en este caso se ha elegido realizar un arranque estrella – triángulo que si bien es cierto el motor real no permite realizar este arranque pero como esta es una simulación se lo podrá hacer verificando ya en los resultados que es lo que pasa con dicha simulación, también se ha ingresado un valor de 2.5 Nm como torque de carga los demás datos de esta simulación se verán en la siguiente

figura que son los que faltan para determinar la simulación que se desea realizar.

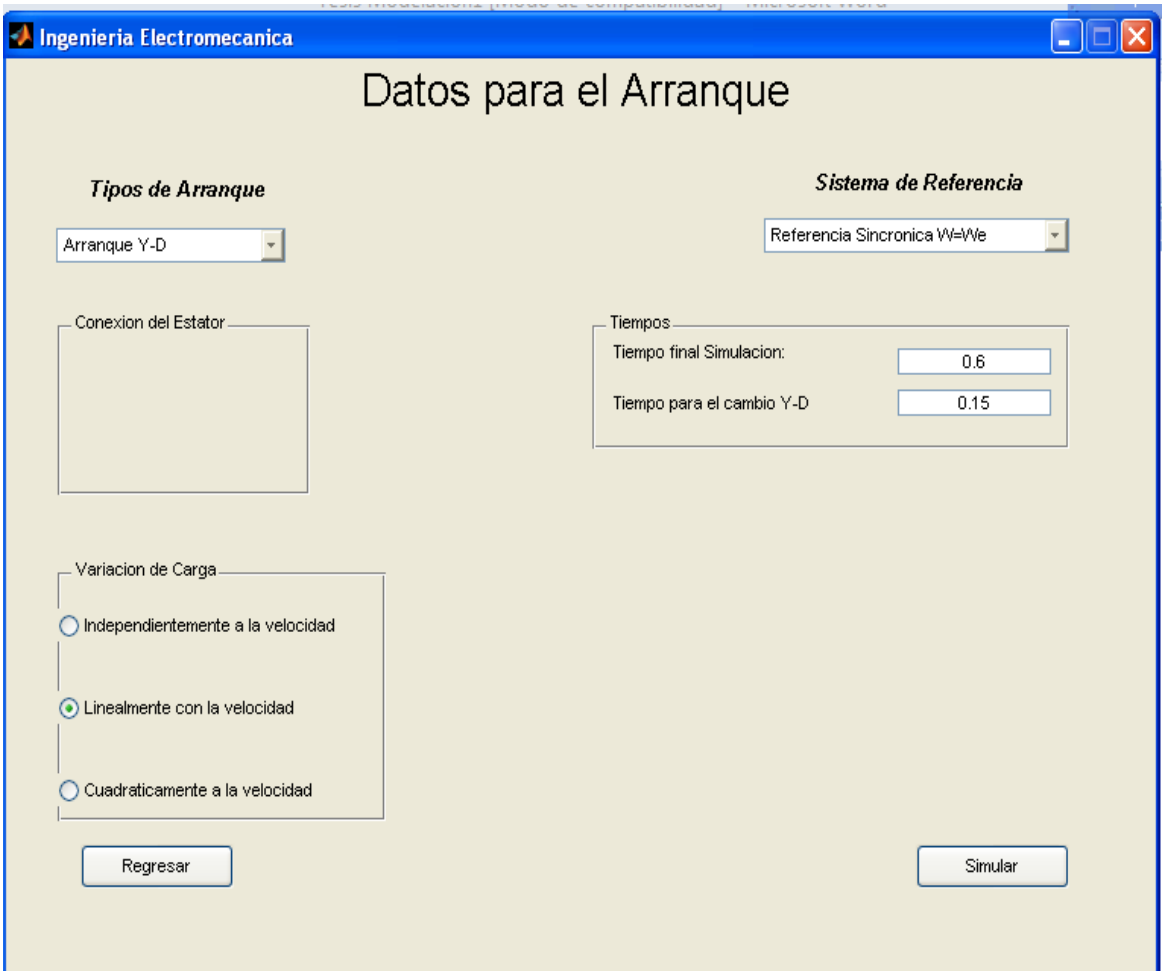



**Figura 4.17 Pantalla para el ingreso de datos correspondiente al Arranque Y-D**

La figura 4.17 representa las condiciones del arranque, se puede observar que el caso a simular es un arranque estrella – triángulo, mientras que el sistema de referencia se ha elegido el cual rota sincrónicamente (W=We), en lo que respecta a la carga en si se ha elegido la opción de variación de carga que sea linealmente con la velocidad que esto en la práctica no se da muy a menudo sin embargo para un entendimiento más amplio esta variación de carga se presenta por ejemplo al encender un generador de excitación independiente que actúa sobre una resistencia externa fija como ya se explico en páginas anteriores.

Una vez que se ha colocado en las pantallas del software los parámetros y condiciones que regirán el arranque estrella - triángulo que se desea simular, lo que ahora viene es simular y analizar las curvas que se generan de esta simulación tomando como particular análisis a aquellas que representan el torque electromagnético, velocidad y la variación de carga en función de la velocidad, este análisis se lo hará a través de las figuras siguientes:

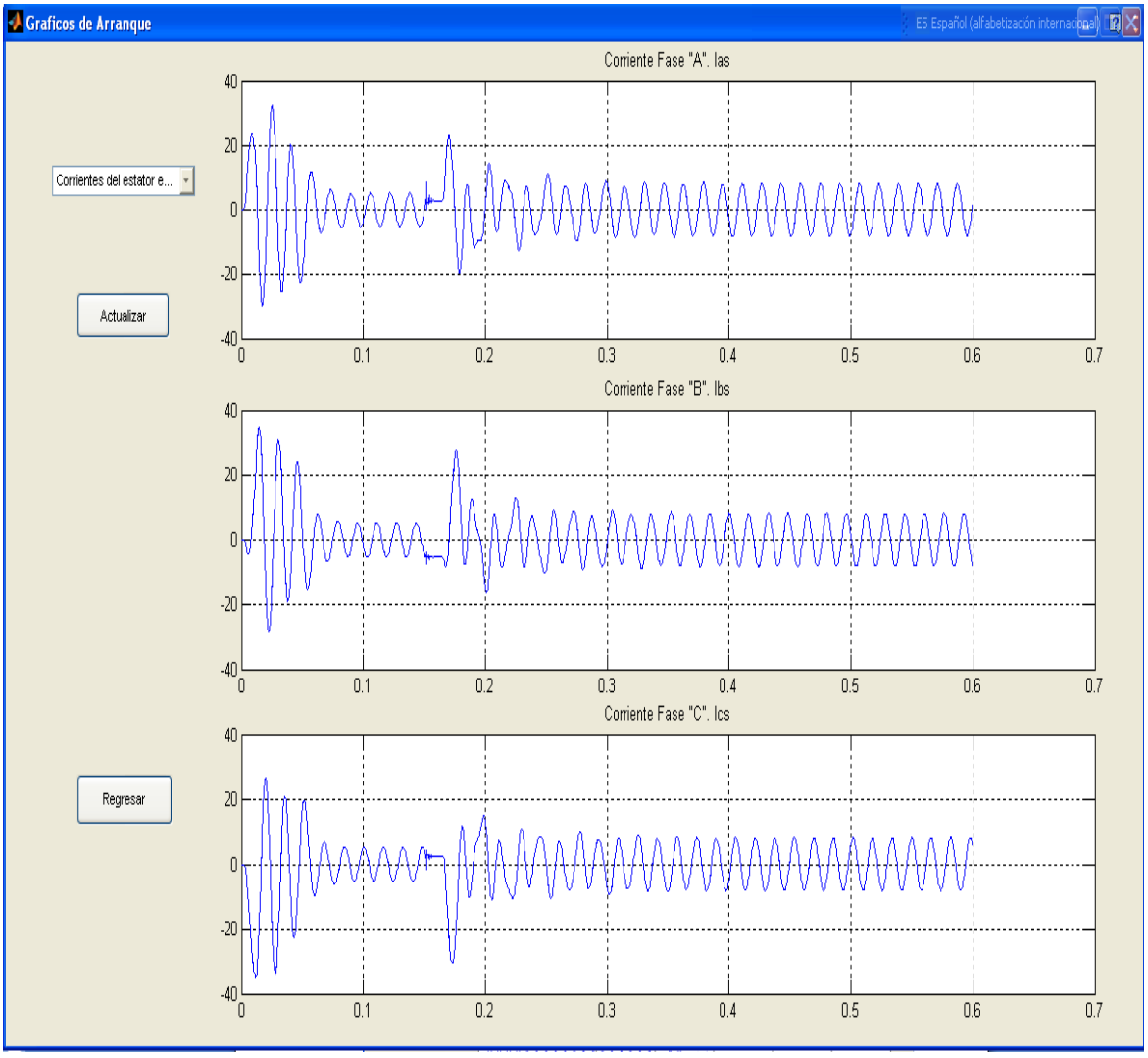



**Figura 4.18 Corrientes en las fases a, b y c**

En la figura 4.18 se observan las corrientes que se generan en el estator al momento del arranque estrella – triángulo que en forma general se ve que en la conexión estrella las corrientes bordean los 35 A que son similares a los que se producen y se analizaron en el arranque directo sin embargo el estado estable

demora más en estabilizarse que en el caso anterior esto se debe al valor de carga que se eligió que es de 2.5 Nm lo que provoca que se tarde más para entrar en régimen estable y con una magnitud mayor de corriente, en el tiempo 0.15 segundos que se eligió para la conmutación a la conexión triángulo se genera un pico alto de corriente que sin embargo es de muy corto tiempo, luego como se ve se estabiliza entrando a funcionar en conexión triángulo, en esta simulación se comprueba que este arranque sirve para dar un arranque suave y progresivo cuando existe un arranque con carga, se ve que si se arrancara directamente en triángulo el pico de corriente sería muy alto lo que no ocurre a primero realizar el arranque en conexión estrella para luego hacerlo en conexión triángulo permitiendo evitar picos muy altos de corriente y quizá lo más importante el evitar que se generen muchos picos de corriente como se ve en la figura anterior.

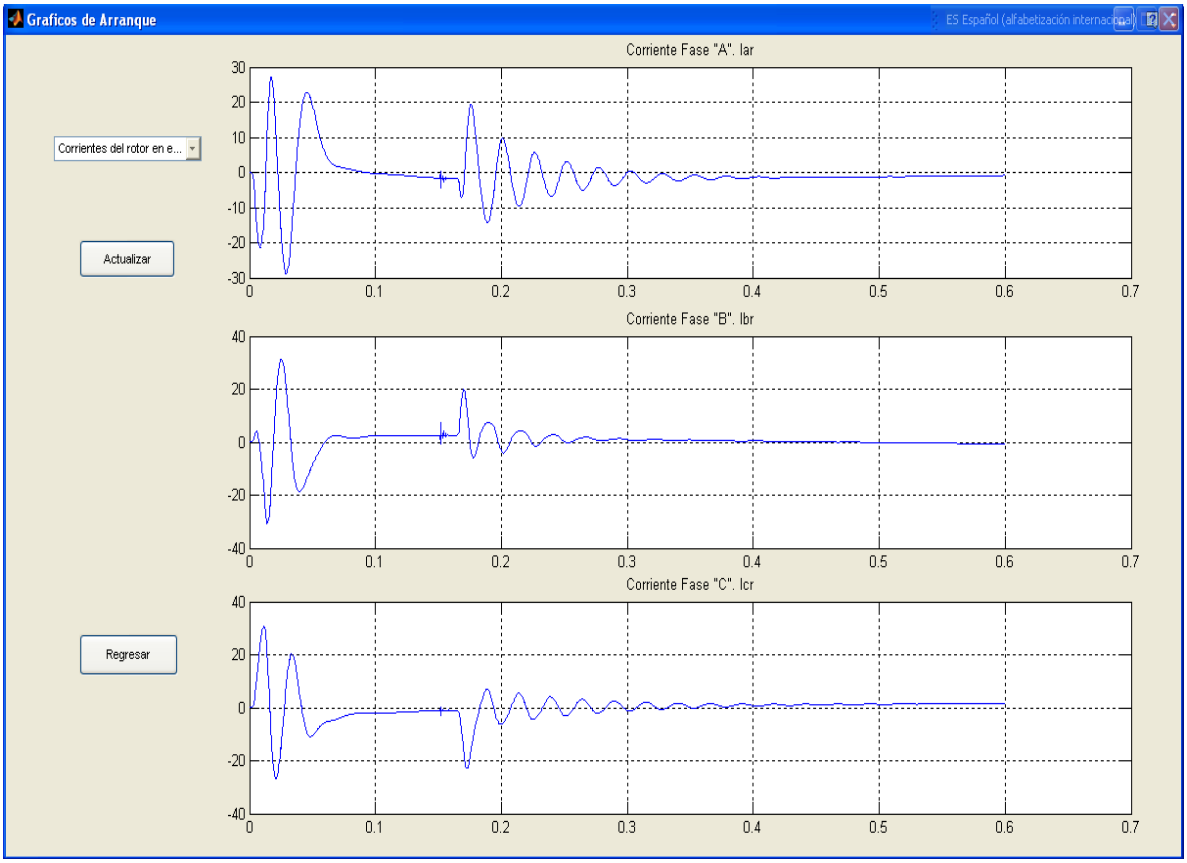



**Figura 4.19 Corrientes rotóricas**

En la figura 4.19 las corrientes rotóricas que se generan del arranque estrella – triángulo se observa que se generan picos de corrientes de similar magnitud que en las corrientes del estator con la diferencia que la frecuencia en un principio es igual a la de la red para luego ir disminuyendo hasta donde se hace el cambio a conexión triángulo aquí se generan también algunos picos de corriente pero de magnitud menor a las que se generan en la conexión estrella y también empieza con una frecuencia de magnitud igual a la de la red y luego irá disminuyendo con mayor rapidez que lo producido en la conexión estrella.

Finalmente se analiza los torques y velocidades que es donde mejor se aprecian los efectos del arranque estrella – triángulo y esto a través de la figura 4.20 en donde en primer lugar se analiza el torque electromagnético producido por la máquina en donde se ve que este alcanza un máximo de 25 Nm no muy brusco ya que los picos van disminuyendo hasta estabilizarse en el valor exacto de 2.5 Nm que es el valor que se eligió de carga para el motor y el mismo produce dicho torque, en el tiempo de 0.15 segundos al realizar la conmutación a conexión triángulo se genera un torque más brusco que incluso llega a tener un valor negativo de -20 Nm aproximadamente y se generan más picos que con el estator en conexión estrella para luego irse estabilizando hasta alcanzar el valor de 2.5 Nm que es el valor de carga colocado en las condiciones de simulación.

En segundo término se analiza la velocidad rotórica aquí se observa que el arranque en conexión estrella se produce de forma suave según la forma de onda de la velocidad se estabiliza en unas 1600 rpm aproximadamente y que luego de conmutar a conexión triángulo en el valor 0.15 segundos empiezan a generarse oscilaciones de velocidad en valores muy pequeños de tiempo que luego se estabilizan y llegan a alcanzar las 1660 rpm, que es el valor al que debe llegar cuando ya se estabiliza completamente el motor.

Finalmente la variación de carga  en función de la velocidad provoca una forma de onda exactamente igual a la de la velocidad esto se debe a que el torque de carga está en función de la velocidad y en este caso tiene una relación lineal en donde el torque de carga en conexión estrella llega a estabilizarse en unos 2.5

Nm para luego cuando se realiza la conmutación a conexión triángulo el valor se hace oscilante sin embargo alcanza finalmente el valor de 2.5 Nm correspondientes al valor de la carga conectada al eje del motor y simulada en la figura.

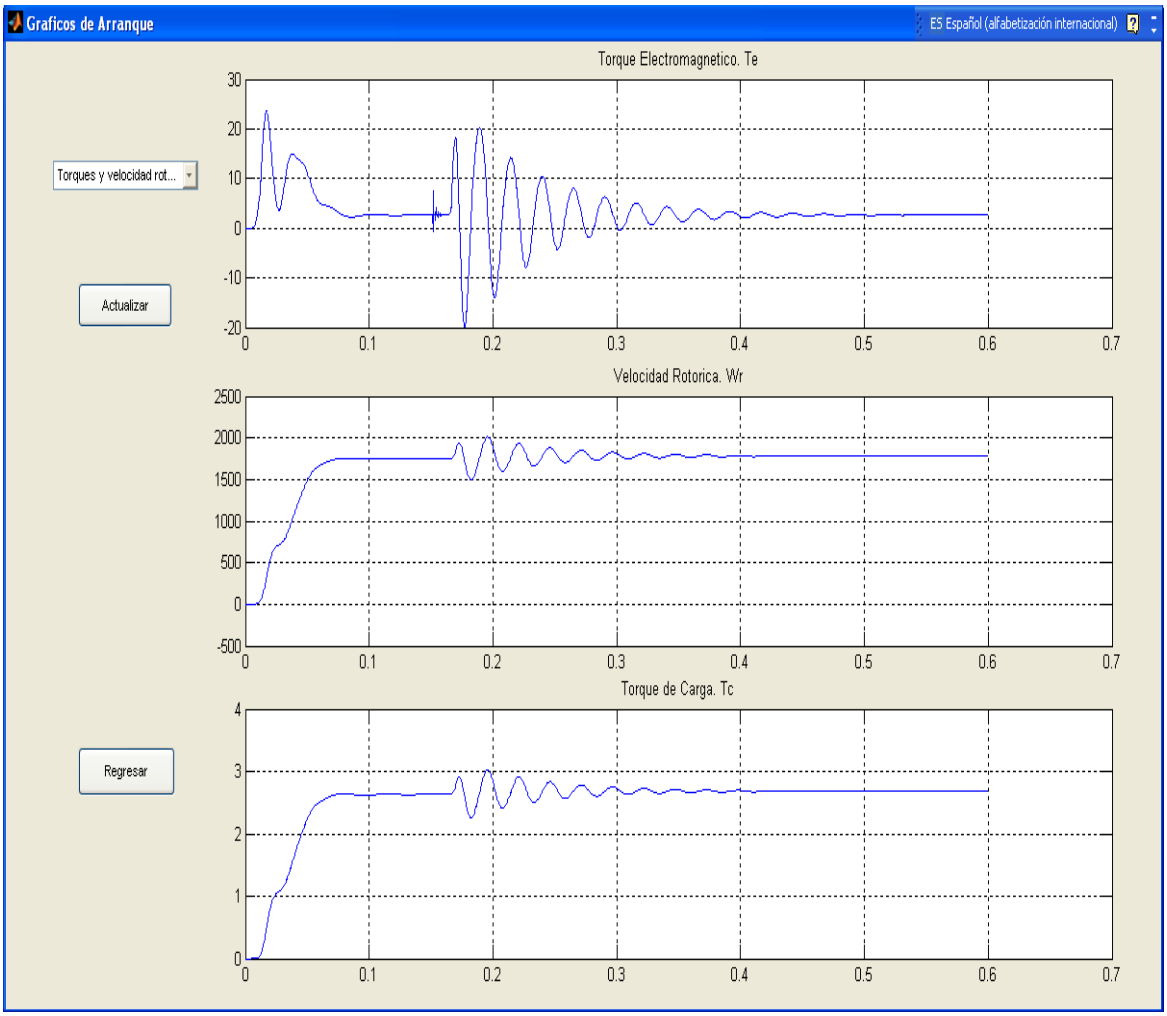



**Figura 4.20 Torque electromagnético, velocidad y torque de carga**

El siguiente caso a simular será una variante de los casos anteriores pues ahora se realizará la simulación del frenado del motor pero con el método de inyección de corriente continua que ya se explico anteriormente, así como también el aumento del momento de inercia del motor y también el cambio del sistema de referencia a utilizar para de esta forma tratar de utilizar todas las herramientas que el programa ofrece.

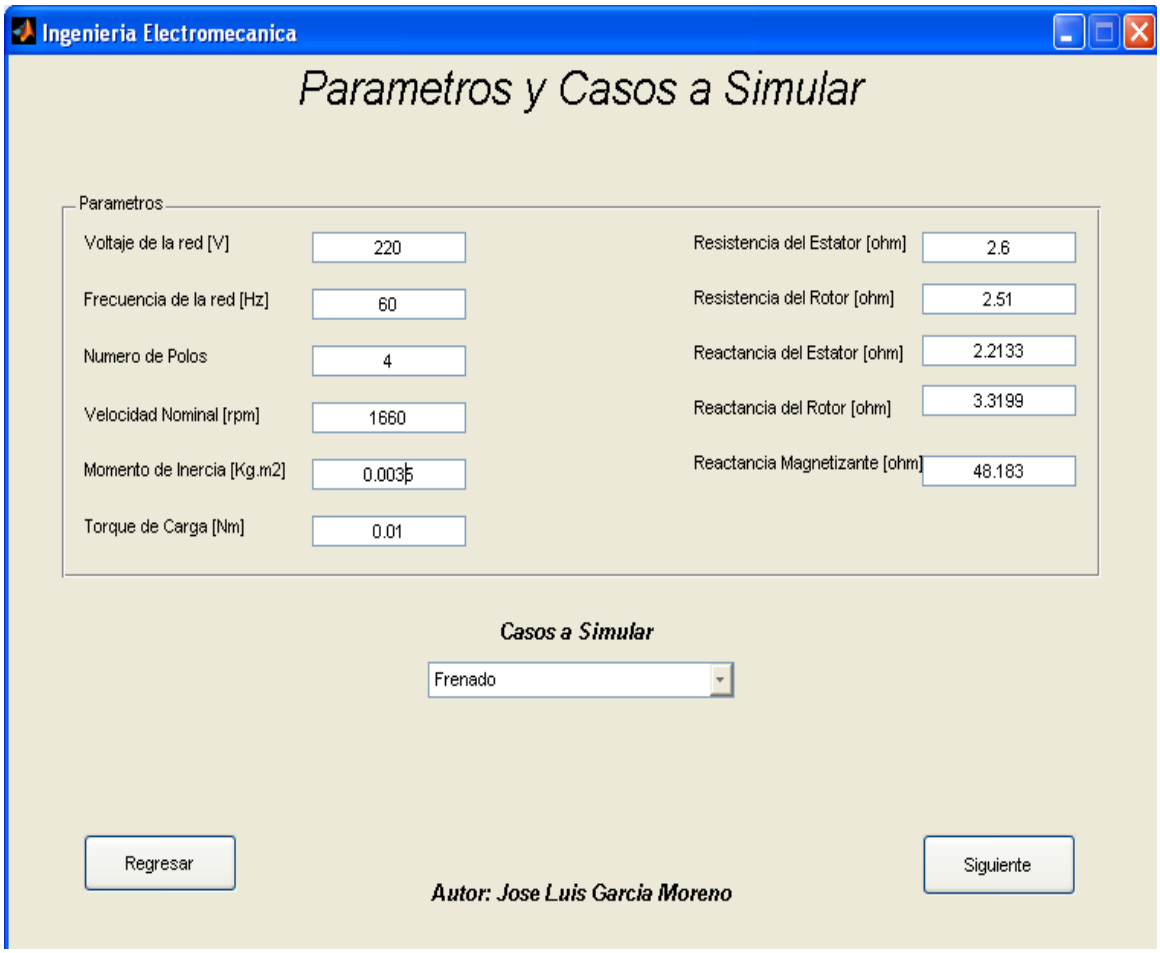



**Figura 4.21 Pantalla de ingreso de datos generales para la simulación del frenado**

De la figura 4.21 se puede notar que el motor arrancara en vacío con la diferencia que ahora el momento de inercia se ha aumentado en un valor de 0.002 Kg.m$^2$ y que sumados al valor de inercia del motor suman un valor de 0.0035 Kg.m$^2$, lo que resulte de este cambio se analizará al tener los datos. En la figura 4.22 se observan los datos correspondientes al frenado por corriente continua que se observa en la pantalla respectiva, otro cambio es la elección del sistema de referencia que en este caso se eligió el de referencia rotórica (W=Wr), al tener un valor de torque de carga de valor cero, la elección de la variación de carga no incide así que lo dejaremos en la primera opción, por último y lo más importante es el elegir el valor de voltaje de corriente continua para el frenado que es de 30 V que proviene de elegir el 25% del voltaje de fase del motor que se analizó anteriormente.

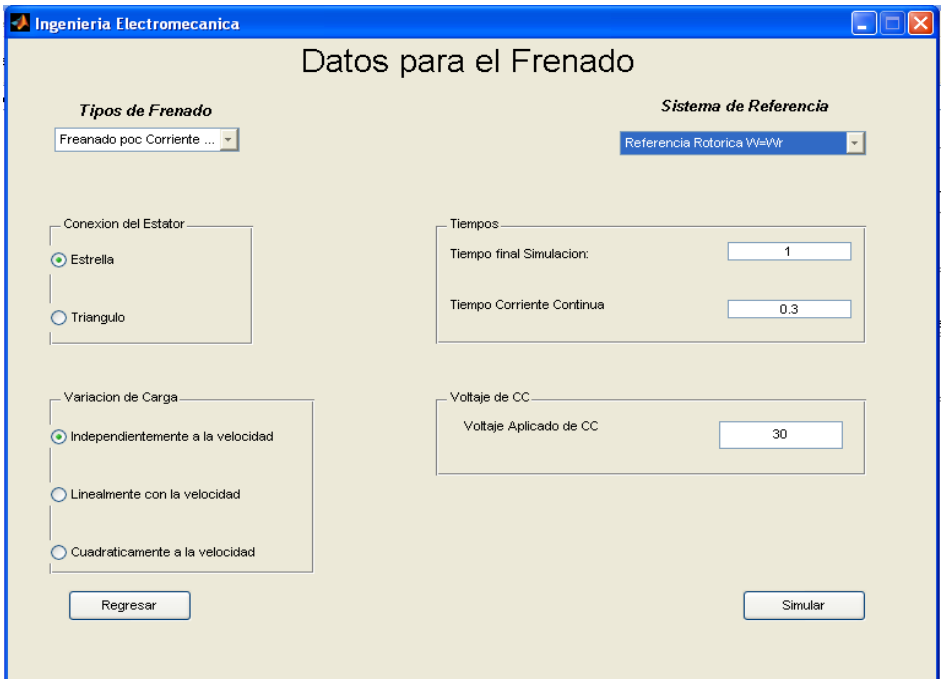



**Figura 4.22 Pantalla para el ingreso de datos correspondiente al frenado por corriente continua**

Determinados los parámetros y condiciones para la simulación se procede a analizar los resultados obtenidos de la simulación así:

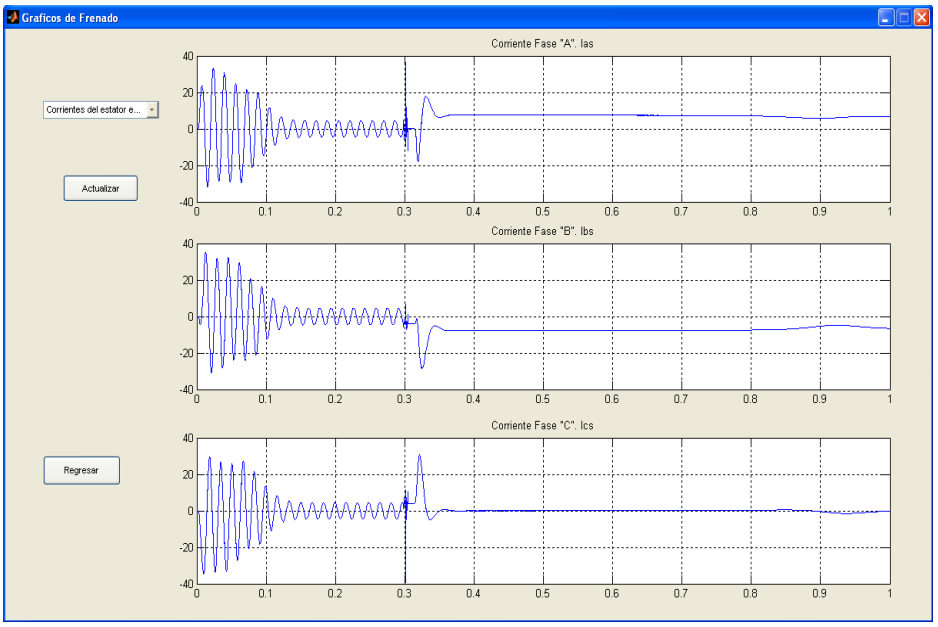



**Figura 4.23 Corrientes en las fases a, b y c**

En la figura 4.23 las corrientes al momento del arranque se generan de manera casi igual que arranque directo pero aquí se ve que existen más picos de corriente aunque la magnitud sea prácticamente la misma de unos 35 A, esta diferencia se debe al momento de inercia que se aumento y provoca que el motor demore más tiempo en vencer el estado de reposo en el que se encuentra antes del arranque y luego llega el estado estable que como se ve es normal, sin embargo en el momento de inyectar corriente continua en el momento 0,3 se generan picos de corriente muy altos en las tres fases pero de corto tiempo y que luego en las fases “a y b” se puede observar valores de 6 y -6 A respectivamente y que son los valores que corresponden a la corriente continua que se inyecta y se ve casi una linealidad que caracteriza este tipo de corriente y en la fase “c” el valor de corriente es de 0 ya que se desconecta de la red.

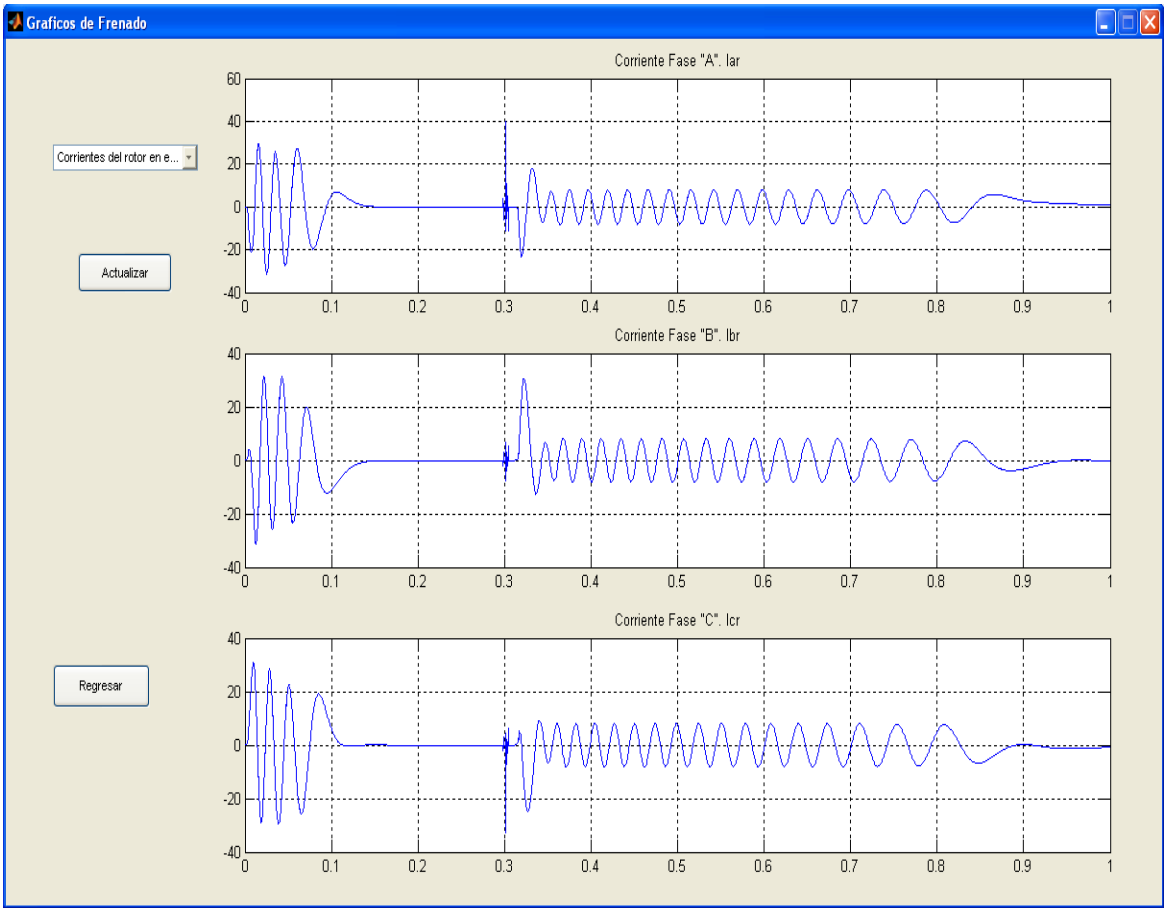



**Figura 4.24 Corrientes rotóricas**

Las corrientes rotóricas que se encuentran representadas en la figura 4.24 dan una visión general de lo que provoca en el rotor al momento del arranque se producen transitorios de similar valor a los que se genero en las fases del estator y que tienen una frecuencia igual a la de la red y que después se va reduciendo en estado estable, ya en el frenado del motor se ve que la inyección de corriente continua genera una forma de onda sinusoidal que tiene el valor de frecuencia de la red y que luego va disminuyendo hasta hacerse cero cuando el motor se ha frenado completamente.

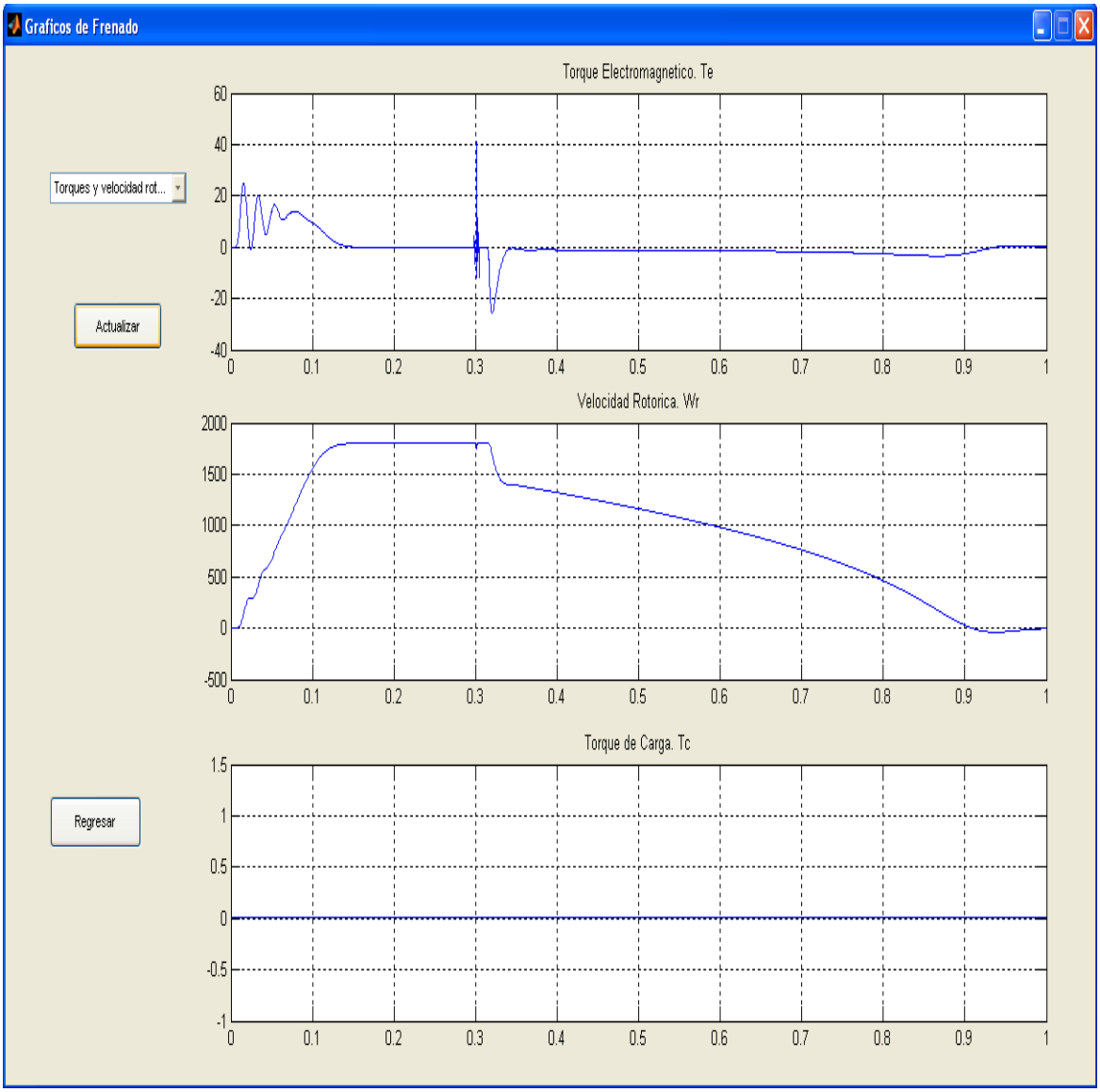



**Figura 4.25 Torque electromagnético, velocidad y torque de carga**

De la figura 4.25 se analizan los torques y velocidades que definitivamente darán una mejor percepción de lo que ocurre en el frenado por inyección de corriente continua es así que en el gráfico de la primera variable que es el torque electromagnético el arranque se produce de manera normal únicamente con la particularidad que al igual que en la corriente del estator los picos transitorios aumentan debido al momento de inercia que se aplicó, pero definitivamente el motor entra en estado estable de manera normal hasta cuando se provoca el frenado y el torque genera un pico muy alto aunque de corto tiempo seguido de un valor de torque negativo que es lo que corresponde al frenado del motor pues se produce una contra fuerza en él para lograr el frenado durante un periodo corto de tiempo, este transitorio llega al valor de cero mientras se va produciendo el frenado.

En el gráfico de velocidad se ve claramente que a diferencia del arranque directo sin aumento del momento de inercia este se demora un poco más en alcanzar la velocidad que se aprecia que es de aproximadamente 1800 rpm y que al momento del frenado se ve que se produce una caída brusca en un principio que corresponde al momento mismo de la inyección de corriente continua y que después la velocidad va disminuyendo linealmente hasta alcanzar el valor de cero que es lo que indica que el freno se ha efectuado completamente y se mantiene bloqueado.

En lo que corresponde al torque de carga no hay mucho que analizar puesto que el valor de carga es casi nulo lo que no influye en los resultados obtenidos y analizados.

Se ha dejado para un último análisis lo correspondiente a las fallas que quizá es lo que más preocupa al momento de realizar un estudio de los estados transitorios y que se verá en las siguientes figuras conforme se vea sus variantes en cada caso para no repetir las condiciones de una y otra, es así que se analizará según la figura siguiente lo que se desea simular.

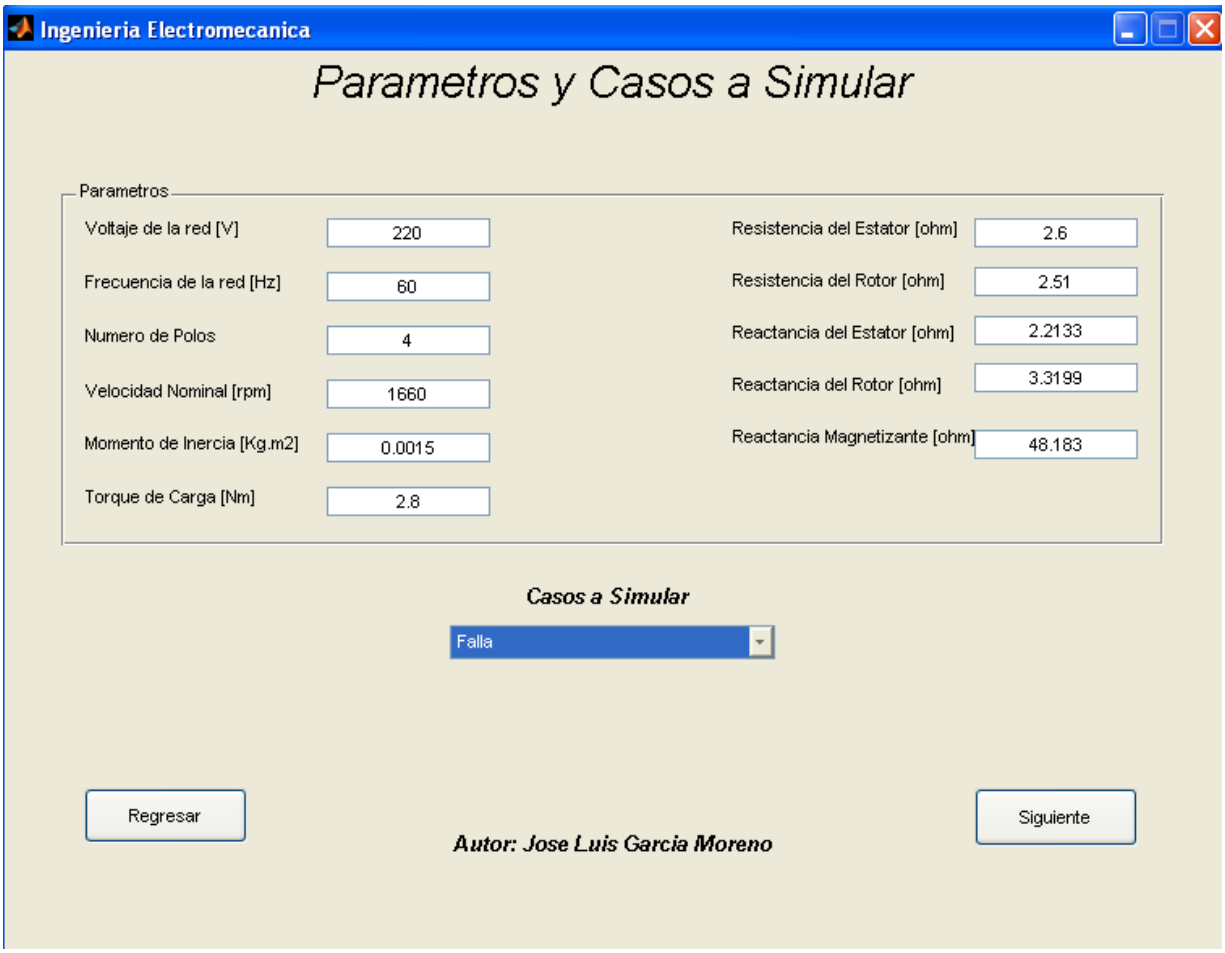



**Figura 4.26 Pantalla de ingreso de datos generales para la simulación de la falla**

De la figura 4.26 la simulación que se realizará es la de fallas tomando en cuenta que el torque de carga que se ha puesto es de 2.8 Nm que es mayor al que se analizo antes y se verá las diferencias por lo demás todo es igual y corresponden a los datos que son normales del motor analizado.

En la figura 4.27 se ingresan todos los datos adicionales que permiten realizar una simulación de falla que se produce únicamente en la fase “a” como se puede ver en los tipos de falla, el sistema de referencia a elegir será el estatórico (W=0), la variante resulta en elegir a la variación de carga como una función cuadrática a la velocidad, colocando también los respectivos tiempos en los que se desee que se produzca la falla que es a 0.4 segundos y la finalización de la simulación a los 0.8 segundos.

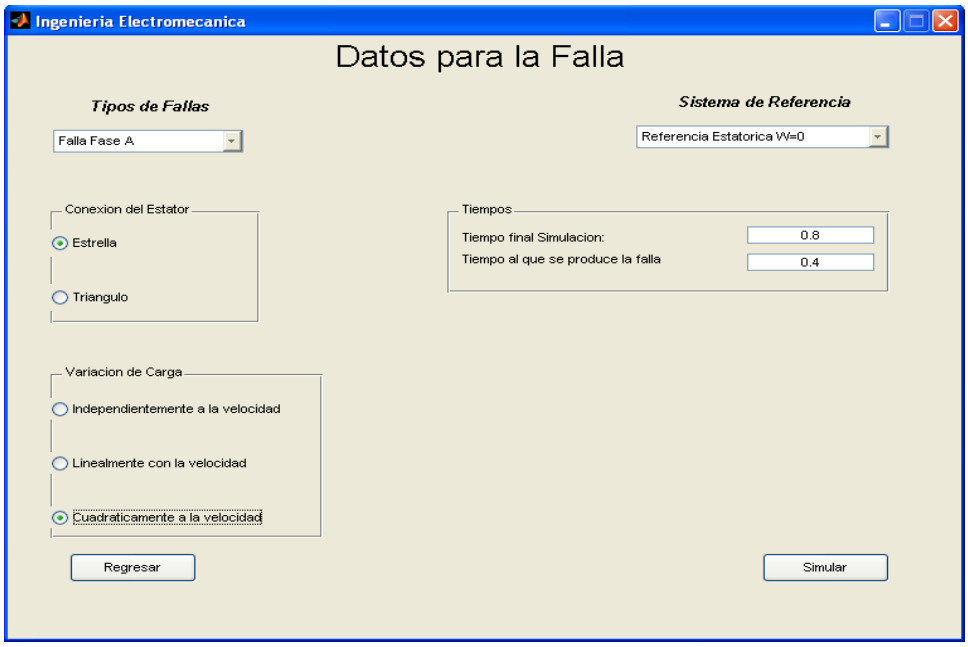



**Figura 4.27 Pantalla para el ingreso de datos correspondiente a la falla**

Realizado el ingreso de todos los parámetros y condiciones para la simulación de la falla en la fase “a” con un valor de torque de carga de 2.8 Nm y que este varié cuadráticamente con la velocidad, se procede entonces al análisis de las curvas.

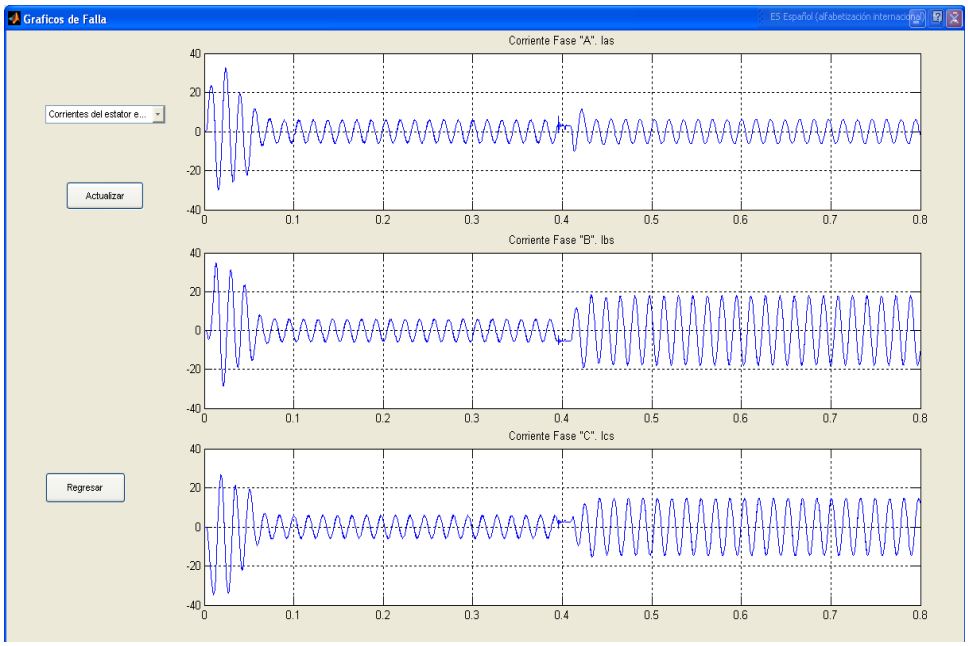



**Figura 4.28 Corrientes en las fases a, b y c**

En la figura 4.28 las corrientes que se producen son de un valor próximo a los 35 A en el arranque y que se generan con unos tres picos de corriente hasta estabilizarse en 7 A de estado estable este aumento de corriente se debe a la carga que se conecto y que tiene un valor de 2.8 Nm, ya producida la falla en el tiempo 0.4 se da un fenómeno transitorio, se observa que en la fase donde se produjo la falla la corriente se mantiene en el valor de 7 A en estado estable, pero esto no ocurre en las fase “b y c” en donde se ve que al contrario de la fase “a” las corrientes aumentan en relación al estado estable de arranque es así que llegan a tener un valor aproximado de 18 A y se mantiene así mientras se produce la falla.

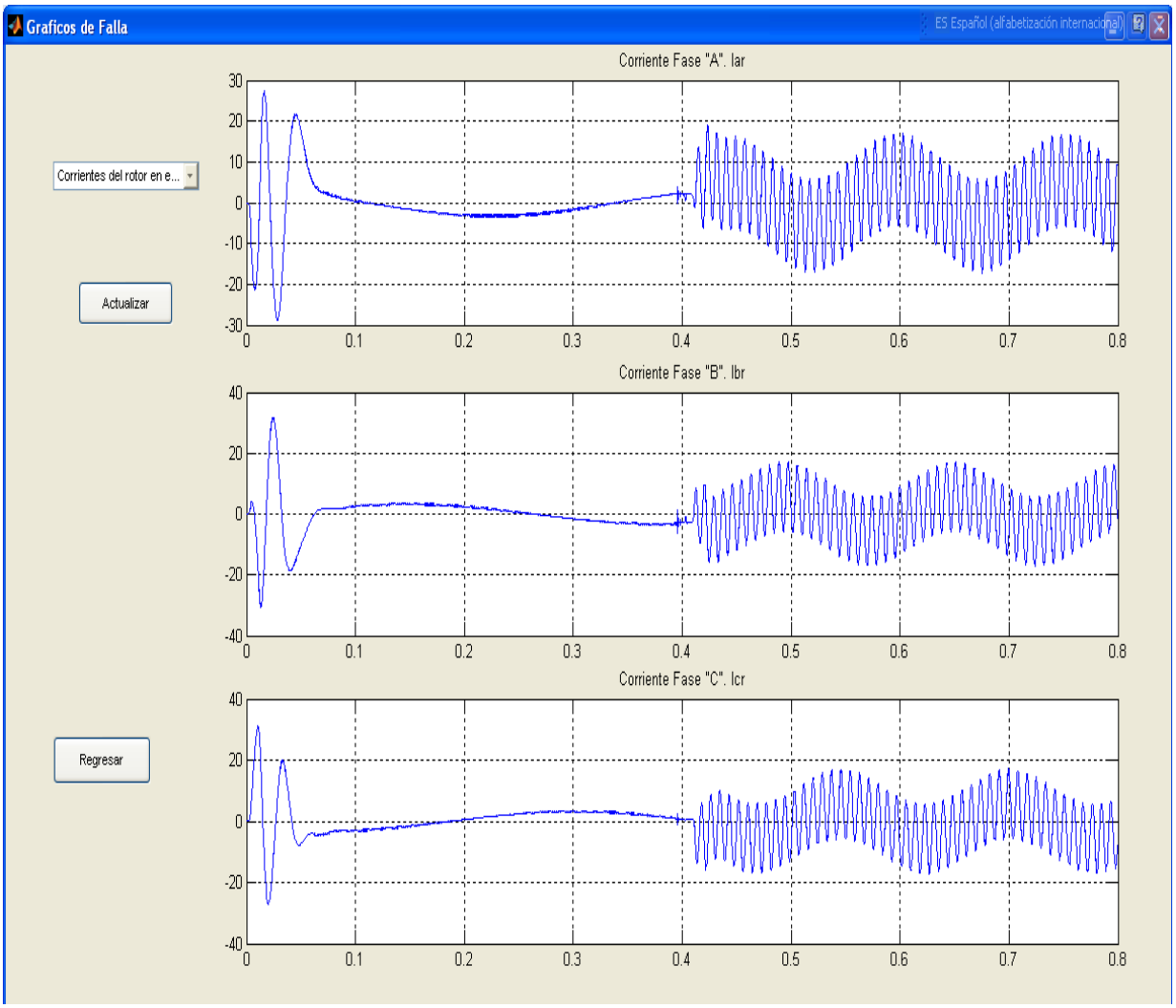



**Figura 4.29 Corrientes rotóricas**

Cosa similar a los expresado anteriormente sucede con las corrientes rotóricas que se encuentran en la figura 4.29 en donde al producirse el arranque se

generan ondas sinusoidales de valor de frecuencia similar al de la red que después disminuye esto siendo una constante en todas las corrientes rotóricas analizadas hasta ahora, sin embargo en el momento de la falla se producen ondas que carecen de uniformidad y que representan las perturbaciones que se están generando en el motor debido a la falla provocada en la fase “a”, las tres formas de onda son similares esto se debe a que a pesar que en las formas de onda de corriente en el estator una era de distinta magnitud la falla se produce en el estator y aunque influye en el rotor lo hace de forma similar en las tres fases.

Por último se analiza  los torques y la velocidad que se producen al momento de la falla de la fase “a” que se puede apreciar en la figura 4.30, de manera particular en el primer gráfico del torque electromagnético que al igual que en casos anteriores de simulación el valor del torque después del estado transitorio se sitúa en los 2.8 Nm que se ingresó como valor del torque de carga, sin embargo en el momento mismo de la falla se genera un valor negativo de -4 Nm que en la práctica simularía ser como un atrancamiento del motor y que provoca que el valor del torque este oscilando a un valor cercano a los 10 Nm provocando que el motor se forcé.

Lo anterior también se observa en la velocidad que después de realizar un arranque común, al momento de la falla se produce una reducción de la velocidad a un valor cercano a las 1600 rpm y en forma oscilante y no constante que difieren de las 1800 rpm sin la presencia de falla y en forma constante, por último la variación de carga en función de la velocidad provoca una curva que se asemeja a la de la velocidad pero con la característica que tienen las ecuaciones cuadráticas, este tipo de carga es el que más se encuentra en la industria y como ejemplo se mencionará que este corresponde al comportamiento de los compresores, ventiladores, etc.

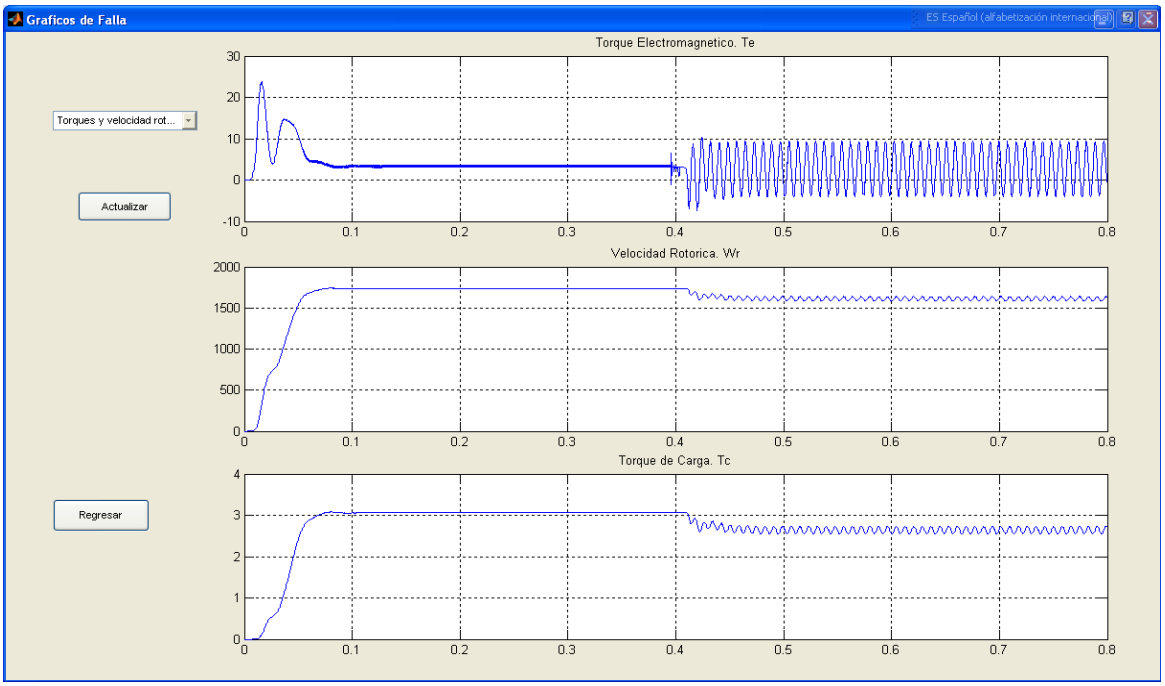



**Figura 4.30 Torque electromagnético, velocidad y torque de carga**

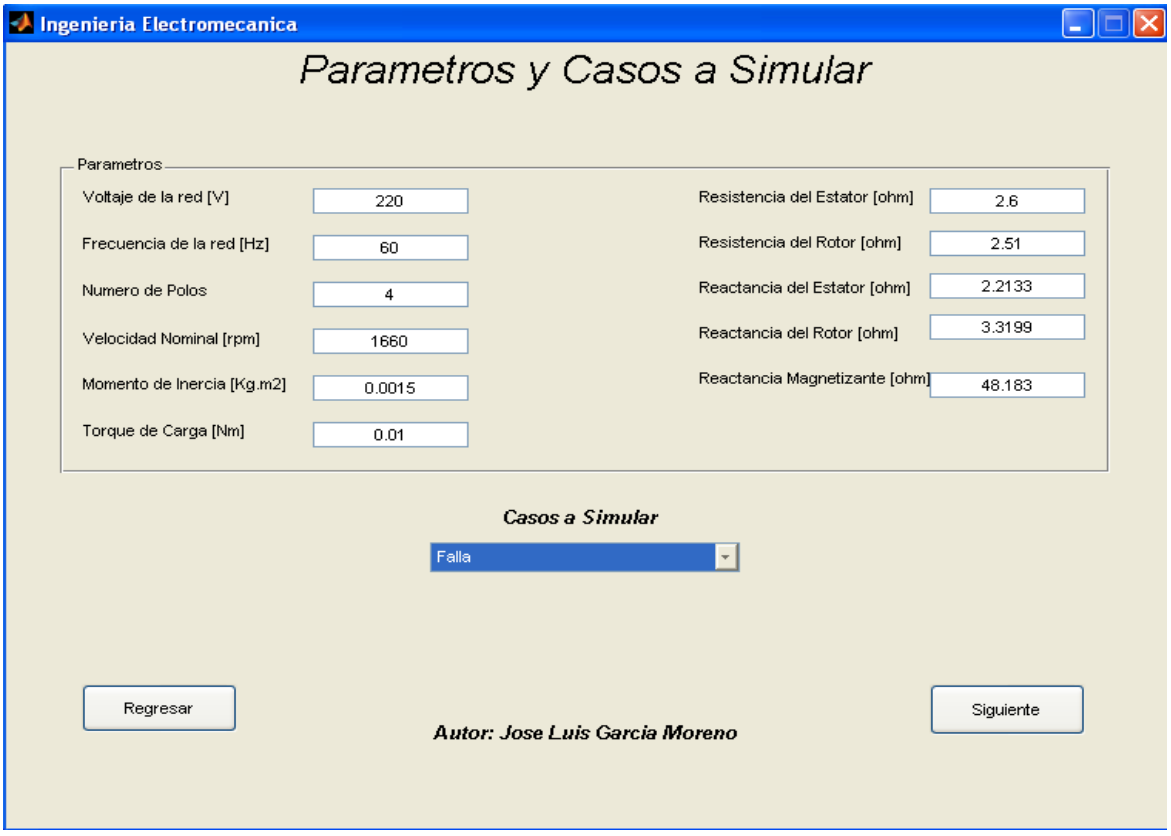



**Figura 4.31 Pantalla de ingreso de datos generales para la simulación de la falla**

La figura 4.31 permite observar que la siguiente simulación se lo hará con el motor trabajando sin carga y sin ninguna alteración de sus datos principales, lo que si se pretende es realizar la falla pero que se produzcan en dos fases como se verá en la siguiente figura que corresponde a los datos para la falla.

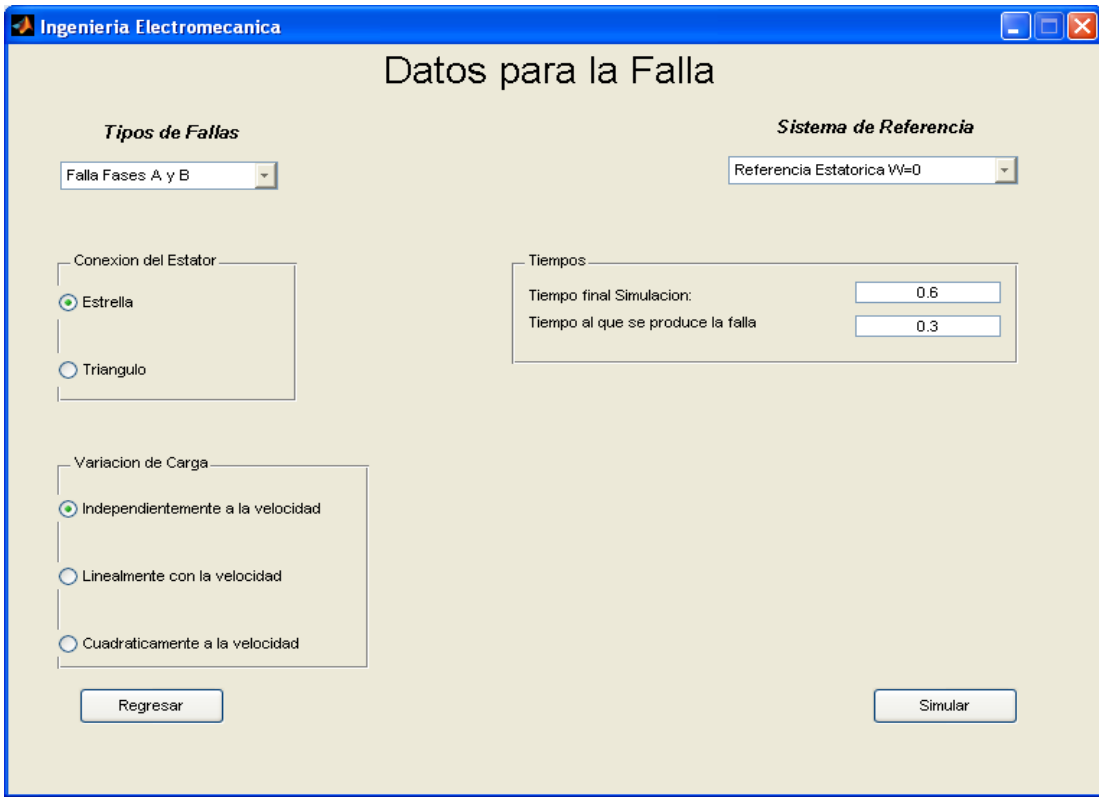



**Figura 4.32 Pantalla para el ingreso de datos correspondiente a la falla**

Los datos que se utilizaran para esta simulación corresponde a los indicados en la figura 4.32 en donde se ha elegido realizar la simulación de falla que se provoca en dos fases que son las fases “a y b” , por lo demás el sistema de referencia se mantiene igual que en el caso anterior con la referencia estatórica (W=0).

La siguiente figura muestra lo obtenido al realizar la simulación y se ve que ahora el arranque produce menos transitorios que si se hubiese puesto un valor de carga o si se aumentaba el momento de inercia lo que no ocurre aquí sin embargo al producirse la falla se observa que se producen transitorios de magnitud más bien baja en relación a los provocados en la anterior simulación,

sin embargo la corriente en las fases “a y b” ya luego de entrar en estado estable se incrementa al producirse la falla de un valor de 5 a casi 10 A que se mantienen durante la falla se produce, sin embargo en la fase “c” el aumento de corrientes aun mayor llegando a los 15 A aproximadamente y al igual que en el caso anterior se mantiene en este valor hasta cuando la falla dure.

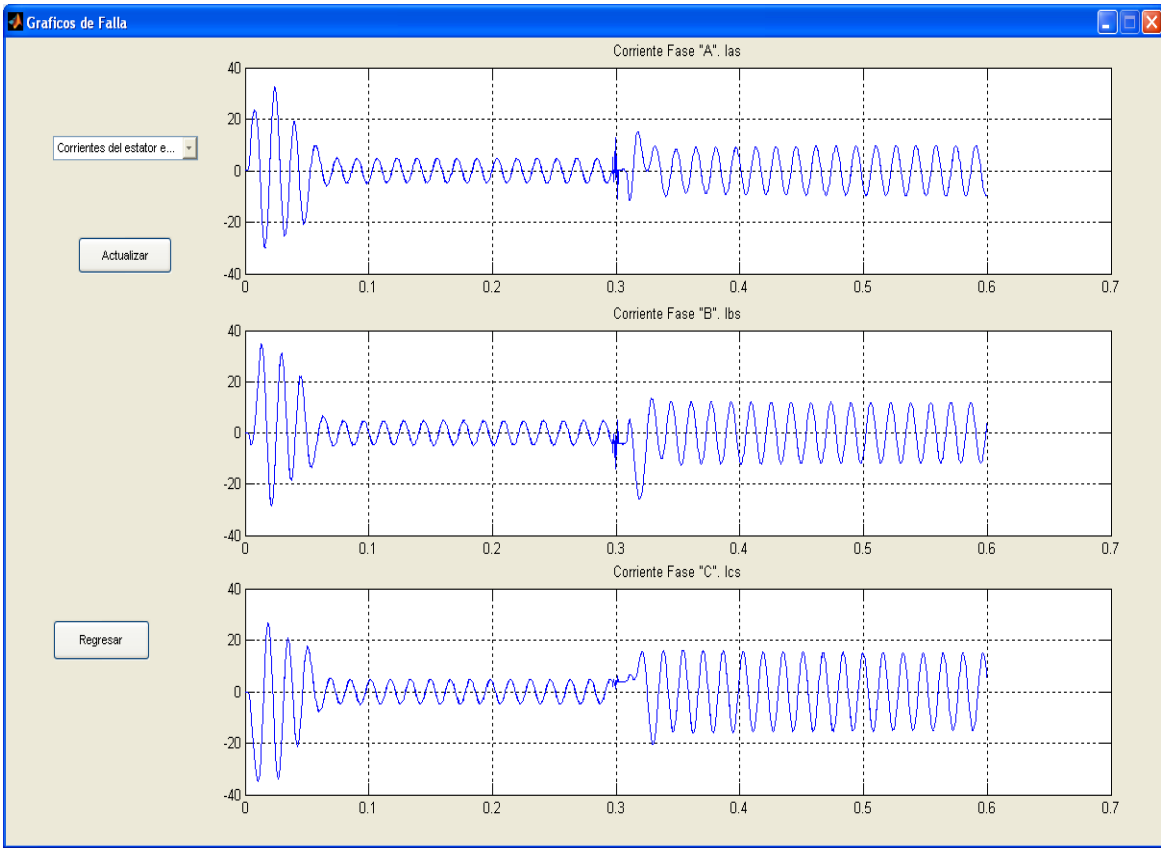



**Figura 4.33 Corrientes en las fases a, b y c**

La siguiente figura a analizar es la correspondiente a las corrientes rotóricas cuyas formas de onda se encuentran en la figura 4.34, se puede observar que en el arranque la corriente tiene un valor pico de 25 A y con dos picos de corriente para luego hacerse cero mientras la frecuencia de la misma va disminuyendo, sin embargo al producirse la falla se genera un pico transitorio de no muy alto valor y de periodo muy corto de tiempo para luego de estabilizarse generar una forma de onda sinusoidal que contiene a ondas sinusoidales y que se mantienen en un valor diferente de cero lo que no es normal hasta cuando la falla se mantenga.

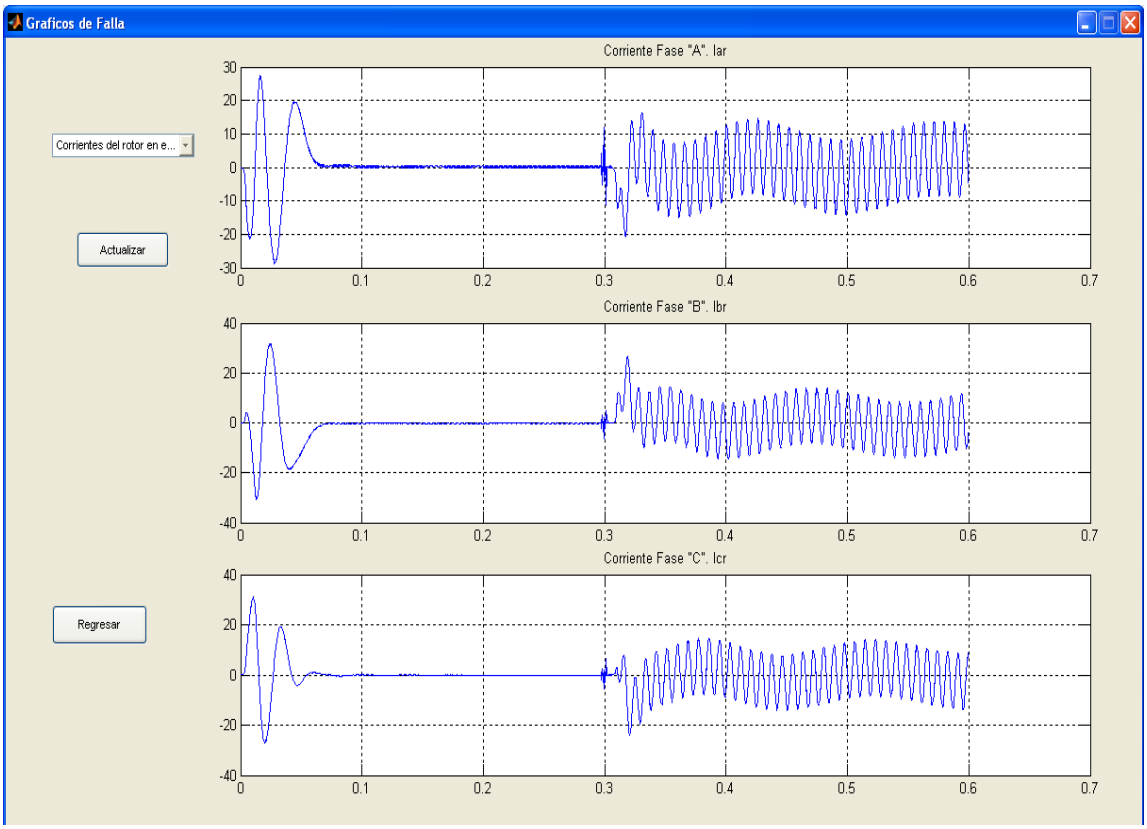



**Figura 4.34 Corrientes rotóricas**

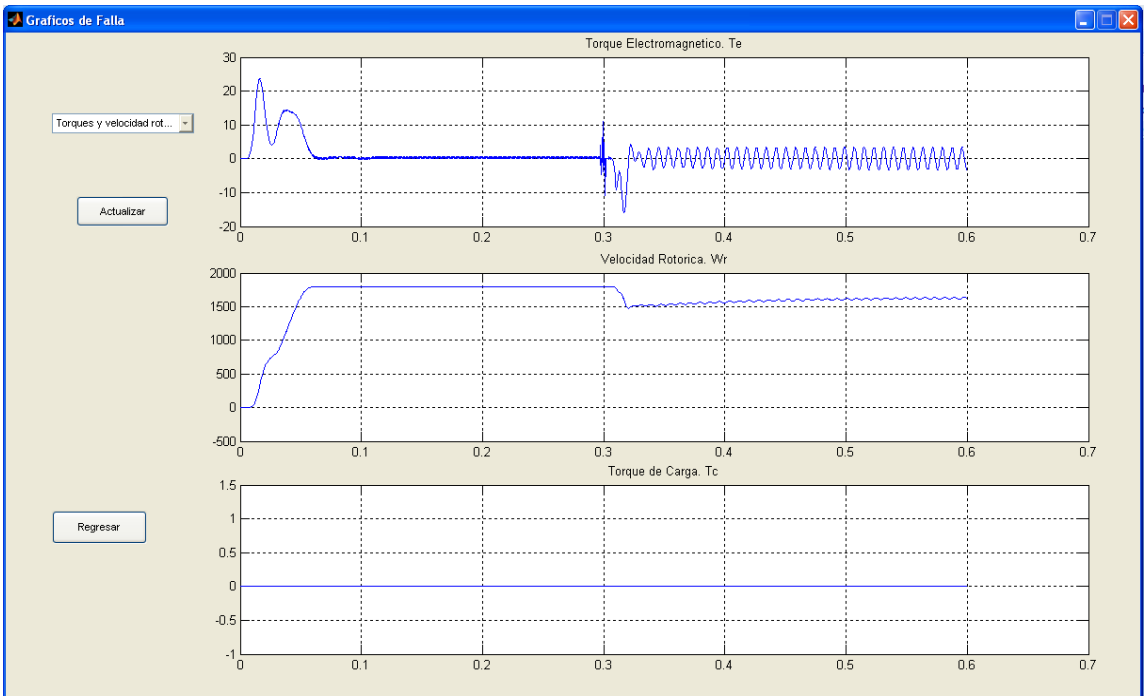



**Figura 4.35 Torque electromagnético, velocidad y torque de carga**

La figura 4.35 representa el torque electromagnético en primer lugar mostrando que durante el arranque se produce un torque máximo de 25 Nm aproximadamente y con una rápida etapa transitoria se pasa al estado estable en donde el valor baja a 0.01 Nm, en la etapa de la falla se presenta un estado transitorio cuyos picos de torque más altos son negativos luego para ingresar a un estado estable que no es normal pues se generan ondas sinusoidales que significan que el motor se está forzando y que llega a ser constante mientras dura el periodo de falla, ahora en lo que se refiere a la velocidad es muy claro que después de un arranque limpio y suave y después de alcanzar el estado estable en donde se alcanza la velocidad de 1800 rpm se produce la falla provocando un decaimiento de la velocidad a 1500 rpm esto en un tiempo muy corto sin embargo a partir de ahí la velocidad empieza a aumentar de forma lenta según dure la falla.

Por último para finalizar esta serie de simulaciones que han permitido ver las bondades del programa se realizará la simulación de la falla trifásica del motor que se encuentra con carga y con un momento de inercia mayor, como se ve en la figura 4.36.

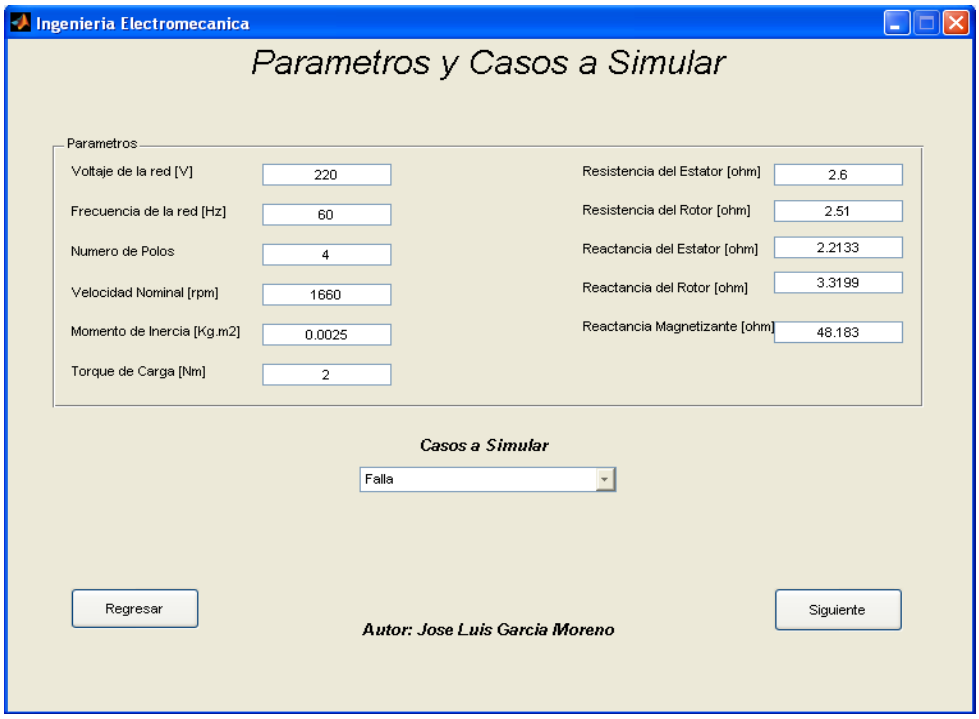



**Figura 4.36 Pantalla de ingreso de datos generales para la simulación de la falla**

En la figura anterior se observa de manera general que se ha variado el momento de inercia del motor aumentándolo en 0.001 Kg.m$^2$ obteniendo el valor total de 0.0025 Kg.m$^2$ y también el torque de carga se ha ingresado con un valor de 2 Nm.


**Figura 4.37 Pantalla para el ingreso de datos correspondiente a la falla**

Los datos que se presentan en la pantalla anterior permiten al programa simular una falla trifásica así como también al tener un valor de torque de carga de 2 Nm como se observó en la figura 4.36 se puede elegir la variación de carga y se eligió una variación de carga cuadrática a la velocidad y el sistema de referencia se ha elegido el sistema estacionario (W=0) y los tiempos se han elegido de 0.5 seg. para el momento de producirse la falla y de 1 seg. para el de finalización de la simulación.

De los datos que se han observado en las figuras anteriores se ha realizado la simulación registrándose las curvas que se presentan en la figuras siguientes, en donde la figura 4.38 muestra las corrientes en el estator que se generan con

un valor pico de 35 A como en casos anteriores con la diferencia de que los picos de corrientes aumentan produciéndose alrededor de seis picos de corriente antes de estabilizarse esto se debe al hecho de incrementar el momento de inercia del motor y también el de ingresar un torque de carga de valor 2 Nm, al producirse la falla en las tres fases “a, b y c” se genera un pico de corriente muy alto pero de muy corto tiempo y que luego se vuelve cero en todas las fases.

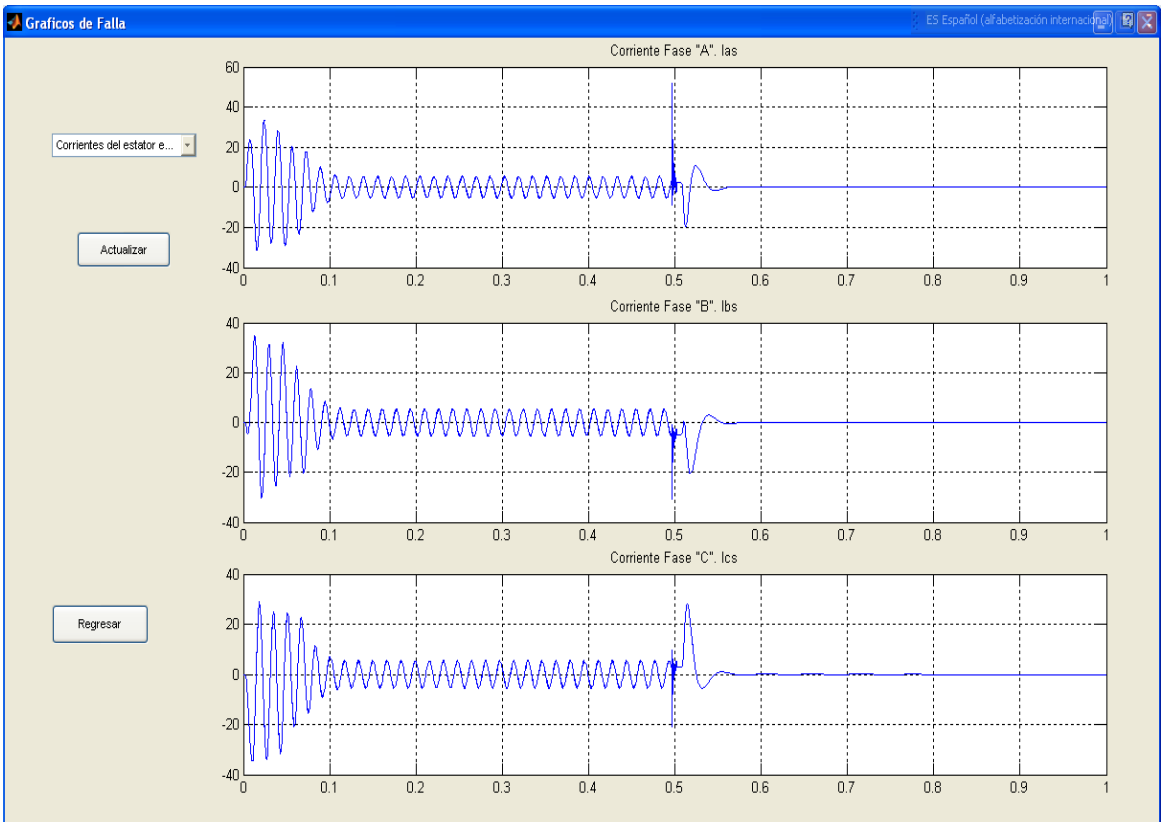



**Figura 4.38 Corrientes en las fases a, b y c**

La siguiente figura a analizar es la de las corrientes rotóricas que como se puede observar tiene las mismas características que las producidas en las corrientes del estator que después de producirse los transitorios de arranque y llegar a estabilizarse las corrientes, al momento de la falla se produce un alto pico de corriente similar al producido en las corrientes del estator y de igual forma después de un breve espacio transitorio se hace cero sin reducción de frecuencia sino de manera instantánea.

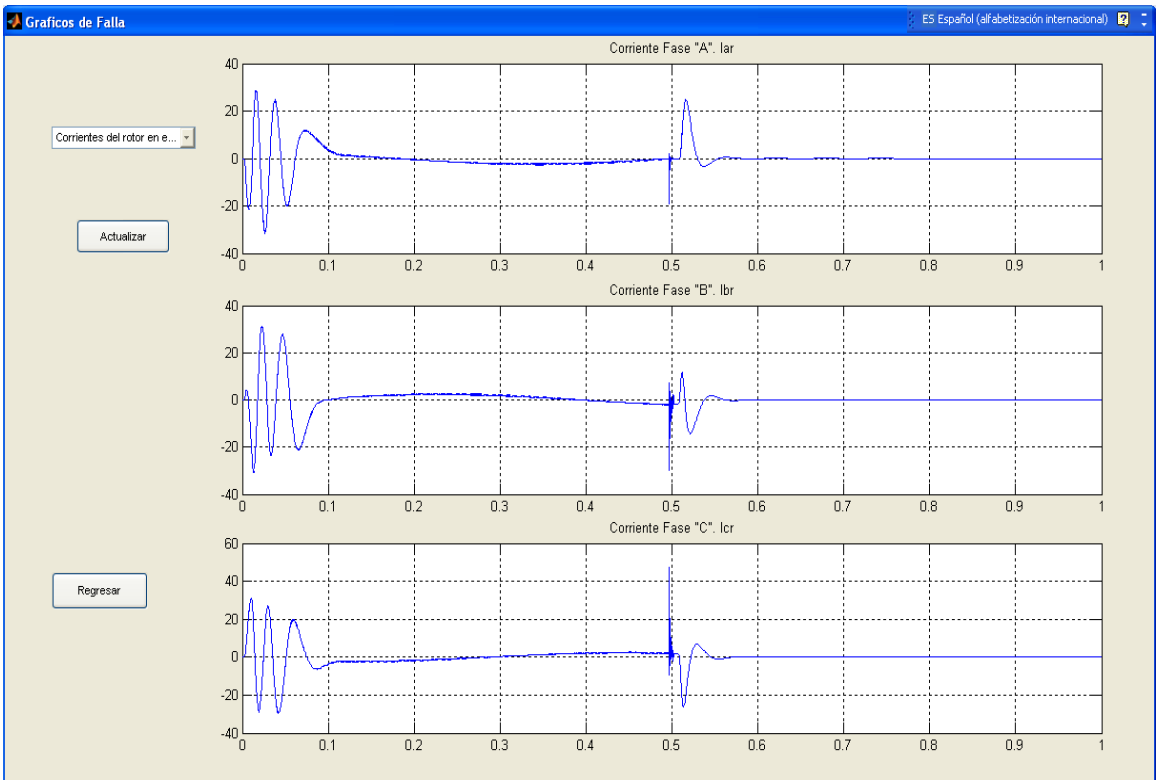



**Figura 4.39 Corrientes rotóricas**

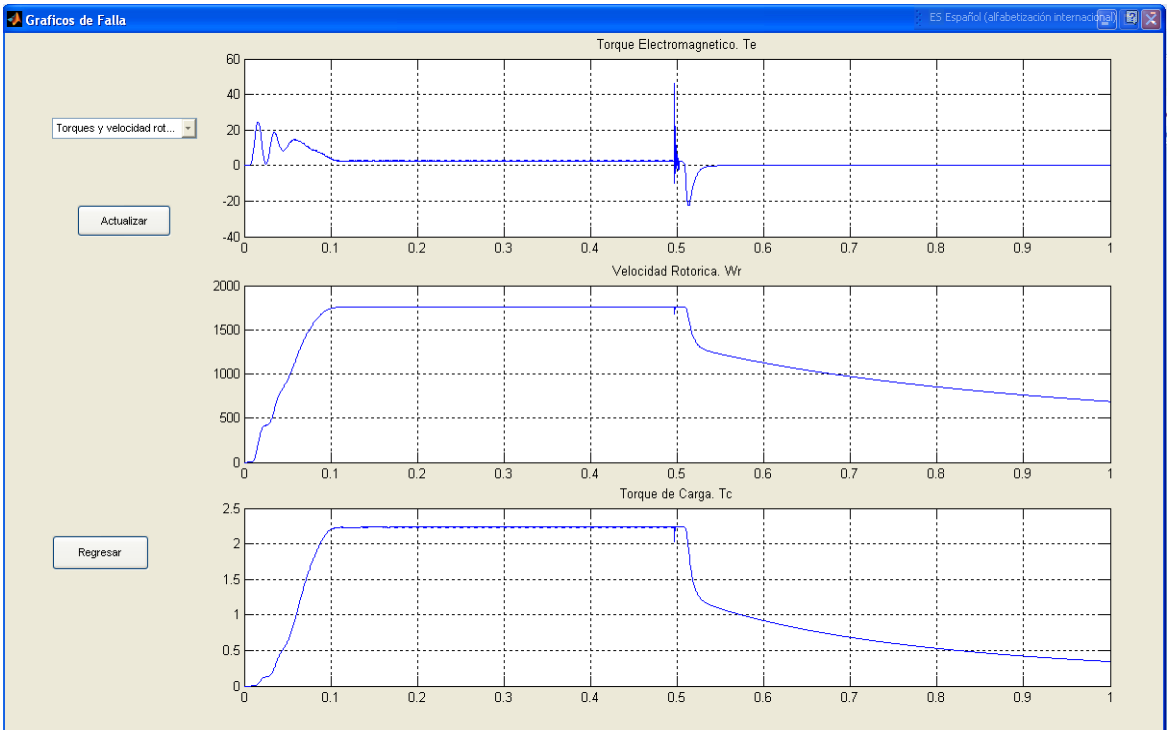



**Figura 4.40 Torque electromagnético, velocidad y torque de carga**

Finalmente se analiza la figura 4.40 en donde en primer lugar se observa el torque electromagnético el cual alcanza un pico en el arranque de 25 Nm y después de unos dos picos de transición se estabiliza en el valor de 2 Nm que es el valor que se observó en la figura 4.36 y corresponde al valor de carga, en el momento de producirse la falla trifásica se confirma lo visto en las corrientes tanto estatóricas como rotóricas en donde se genera un pico alto de torque pero que no dura mucho tiempo más bien es instantáneo y después se genera un torque negativo muy característico de las fallas y los frenados como ya se observó y analizó en anteriores simulaciones hasta llegar al valor de cero. La velocidad después de realizar el arranque y de llegar al estado estable de 1800 rpm se produce la falla trifásica provocando una disminución considerable y muy rápida hasta llegar a valores cercanos a cero.

# CAPÍTULO V

# 5. CONCLUSIONES Y RECOMENDACIONES

## 5.1.- CONCLUSIONES

- La modelación de la máquina eléctrica de inducción y en particular del motor jaula de ardilla es una excelente herramienta que permite representar el funcionamiento en condiciones dinámicas al producirse los estados de arranque, frenado y falla.

- La adquisición de datos por medio del software Lab View y su respectivo hardware como es la tarjeta NI USB 6008 permite obtener datos muy rápidos y precisos de los estados transitorios que son el punto más importante dentro de la adquisición.

- El módulo diseñado para contener la tarjeta y los respectivos circuitos de control han permitido evitar mucho cableado logrando ahorrar tiempo y espacio al momento de realizar las conexiones para realizar la adquisición de datos.

- El software de simulación realizado en el programa computacional Matlab permite la simulación de un sin número de variantes con respecto al arranque, frenado y falla debido a lo fácil y rápido que resulta el ingresar los datos y determinar las condiciones en las cuales trabajará el motor simulado tomando en cuenta siempre que es una modelación y

por lo tanto una aproximación en este caso muy buena de los datos que se obtienen de un motor en operación como el realizado en este tema de tesis.

- El software de simulación se vuelve muy útil al poder simular, observar y analizar variables que en la práctica es muy difícil y quizás imposible de medir, tal es el caso que se puede obtener los datos de lo que ocurre con las corrientes del rotor, torque electromagnético y torque de carga.

- La comparación realizada entre la adquisición de datos por Lab View y la simulación realizada por Matlab tienen ciertas diferencias en los resultados; esto se debe a que a la máquina de inducción para su modelación se la idealizó no tomando en cuenta ciertos fenómenos como la saturación, las armónicas de la onda de fmm; corrientes de Eddy, etc.

## 5.2 RECOMENDACIONES

- A futuro para lograr un completo Laboratorio Virtual de Máquinas Eléctricas será de imperiosa necesidad el desarrollar módelos de máquinas tales como las de corriente continua, las máquinas sincrónicas, las máquinas de inducción de rotor bobinado y máquinas de inducción de doble jaula de ardilla.

- Sera necesario tomar en cuenta para futuras modelaciones y simulaciones, lograr modelar aquellos efectos que aquí no se han analizado e introducirlos en esta modelación o en otras que se realicen de otras máquinas, tales efectos como la saturación, la variabilidad de las inductancias de dispersión, armónicas, corrientes de Eddy se podrían introducir para lograr que el modelo de la máquina de inducción sea más preciso.

- Realizar mediante el uso de equipos adecuados la adquisición de datos de múltiples variables como voltaje y velocidad que permitan el análisis no solo en función de una variable como lo fue la corriente sino de otras más como las citadas.

- En un trabajo posterior se podría analizar la posibilidad de que la comparación no sea de forma visual sino lograr que la comparación sea automática a través de tarjetas o programas que permitan la interacción entre el software de adquisición de datos Lab View y el software matemático de simulación Matlab.

# BIBLIOGRAFÍA

## LIBROS


- Chee-Mun Ong, Dynamic Simulation of Electric Machinery Using Matlab/Simulink, Primera Edición, Prentice - Hall, Estados Unidos, 1997.

- Krause, Paul C, Analysis of Electric Machinery, Primera Edición, Mc Graw Hill, México, 1986.

- Cathey Jimmie J, Máquinas Eléctricas Análisis y Diseño con Matlab, Primera edición, Mc Graw Hill, México, 2002.

- Hancock N. N, Matrix Analysis of electric Machinery, Segunda Edición, Pergamon Press, Estados Unidos, 1974.

- Fitzgerald A. E, Máquinas Eléctricas, Sexta Edición, Mc Graw Hill, México, 2004.

- Stevenson, William D, Análisis de Sistemas Eléctricos de Potencia, Segunda Edición, Mc Graw Hill, México, 1992.

- Telemecanique, Concepción de un Automatismo telesquemario, Telemecánica Eléctrica Española, S.A.

- Valdivieso Veintimilla Luis Fernando, Modelo de la máquina de inducción considerando inductancias variables, Escuela Politécnica Nacional, Facultad de Ingeniería Eléctrica, 1997.

- Pozo Acosta Mauricio Efraín, Modelo digital de la máquina trifásica de inducción con rotor bobinado, jaula y doble jaula de ardilla para estudio transitorio, Escuela Politécnica Nacional, Facultad de Ingeniería Eléctrica, 1989.

- Mancero Santillán Fernando, Comportamiento dinámico de motores de inducción con capacitor serie y paralelo, Escuela Politécnica Nacional, Facultad de Ingeniería Eléctrica, 1983.

- Morales Marchena Herón, Matlab 7 para ciencias e ingeniería con métodos numéricos y visualización gráfica, Primera Edición, Grupo Editorial Megabyte, Perú, 2005.

- Nasar S. A. y Boldea I, Máquinas Eléctricas Dinámica y Control, Primera Edición, Compañía Editorial Continental, México, 1995.

- Rodríguez Soldevilla Federico, Dinámica de Máquinas, Primera Edición, Editorial Limusa, México, 1982.


## ENLACES WEB


- http://www.ecosimpro.com/download/articles/C01_17_es.pdf
- http://www.unizar.es/eees/innovacion06/COMUNIC_PUBLI/BLOQUE_III/CAP_III_8.pdf
- http://prof.usb.ve/jaller/PPI_papers/paper_02_00.pdf
- http://prof.usb.ve/jaller/PPI_papers/saaei2.pdf
- http://prof.usb.ve/jaller/PPI_papers/paper_08_98.pdf
- http://prof.usb.ve/jaller/Ascenso/Capítulo_3.pdf
- http://prof.usb.ve/jaller/Ascenso/Capítulo_4.pdf
- http://prof.usb.ve/jaller/Ascenso/Capítulo_5.pdf
- http://prof.usb.ve/jaller/Guia_Maq_pdf/Capítulo07.pdf
- http://prof.usb.ve/jaller/Guia_Maq_pdf/Capítulo09.pdf
- http://prof.usb.ve/jaller/Guia_Maq_pdf/Capítulo10.pdf
- http://prof.usb.ve/jaller/Guia_Maq_pdf/Capítulo11.pdf
- http://prof.usb.ve/jaller/Guia_Maq_pdf/Capítulo13.pdf
- http://prof.usb.ve/jaller/Guia_Maq_pdf/cat_motores_ind.pdf
- http://prof.usb.ve/jaller/Ascenso/CT-6311-cap_02.pdf
- http://www.giaelec.org/ReportesInv/RPT2007-10.pdf
- http://www.rpi.edu/~vanfrl/pdfs/sp_resumen_tesis_lvanfretti.pdf

- http://mat21.etsii.upm.es/ayudainf/aprendainf/Matlab7/matlab7primero.pdf

# ANEXOS

## Datos de Placa:

SIEMENS 3~ Motor
1LA70804YA60
1HP, 220YY / 440Y V, 3.5 / 1.75 A.
60 Hz, η 64.2, cosφ 0.87, Ia 3.7 In.
Tn/Ta 4.29/8.1 Nm, 1660 rpm, 8.1 Kg.
S1, IP55, IMB3, AISL. F, BG 080, IEC 34.
FS 1.15, Ta -15/40°C, 1000 msnm,

## Valores de Prueba en Vacío

$V_L$ = 210 V
$I_L$ = 2,4 A
$P_T$ = 90 W
n = 1796 rpm
Rs = 2,6 Ω

## Valores de Prueba a Rotor Bloqueado

$V_L$ = 56,1 V
$I_L$ = 4,3 A
$P_T$ = 283,5 W
$f_{rb}$ = 60 Hz.

## Parámetros

Rs = 2,6 Ω

R'r = 2,5109 Ω

Xls = 2,2133 Ω

Xl'r = 3,3199 Ω

Xm = 48,183 Ω

J = 0,0015 $Kg.m^2$

| | 6008 | |
|---|---|---|

• Small and portable
• 12 or 14-bit input resolution,
at up to 48 kS/s
• Built-in, removable connectors
for easier and more
cost-effective connectivity
• 2 true DAC analog outputs
for accurate output signals
• 12 digital I/O lines (TTL/LVTTL/CMOS)
• 32-bit event counter


**Operating Systems**

• Windows 2000/XP
• Mac OS X1
• Linux®1
• Pocket PC
• Win CE

**Recommended Software**

• LabVIEW
• LabWindows/CVI

**Hardware Description**

The National Instruments USB-6008 and USB-6009 multifunction data acquisition (DAQ) modules provide reliable data acquisition at a low price. With plug-and-play

USB connectivity, these modules are simple enough for quick measurements but versatile enough for more complex measurement applications.

**Software Description**

The NI USB-6008 and USB-6009 use NI-DAQmx high-performance, multithreaded driver software for interactive configuration and data acquisition on Windows OSs. All NI data acquisition devices shipped with NI-DAQmx also include VI Logger Lite, a configuration-based data-logging software package.

Mac OS X and Linux users can download NI-DAQmx Base, a multiplatform driver with a limited NI-DAQmx programming interface. You can use NI-DAQmx Base to develop customized data acquisition applications with National Instruments LabVIEW or C-based development environments. NI-DAQmx Base includes a ready - to-run data logger application that acquires and logs up to eight channels of analog data. PDA users can download NI-DAQmx Base for Pocket PC and Win CE to develop customized handheld data acquisition applications.

**Recommended Accessories**

The USB-6008 and USB-6009 have removable screw terminals for easy signal connectivity. For extra flexibility when handling multiple wiring configurations, NI offers the USB-6008/09 Accessory Kit, which includes two extra sets of screw terminals, extra labels, and a screwdriver. In addition, the USB-6008/09 Prototyping Accessory provides space for adding more circuitry to the inputs of the USB-6008 or USB-6009.

**Common Applications**

The USB-6008 and USB-6009 are ideal for a number of applications where economy, small size, and simplicity are essential, such as:

• Data logging – Log environmental or voltage data quickly and easily.

• Academic lab use – The low price facilitates student ownership of DAQ hardware for completely interactive lab-based courses.

• Embedded OEM applications.

The following specifications are typical at 25 °C, unless otherwise noted.

**Analog Input**

| | |
|---|---|
| Converter type..... | Successive approximation |
| Analog inputs ..... | 8 single-ended, 4 differential, |
| Input resolution | |
| USB-6008 ..... | 12 bits differential, 11 bits single-ended |
| Max sampling rate1 | |
| USB-6008 ..... | 10 kS/s |
| AI FIFO..... | 512 bytes |
| Timing resolution ..... | 41.67 ns (24 MHz timebase) |
| Timing accuracy..... | 100 ppm of actual sample rate |
| Input range | |
| Single-ended ..... | ±10 V |

| | |
|---|---|
| Differential..... | ±20 V, ±10 V, ±5 V, ±4 V, ±2.5 V, ±2 V, ±1.25 V, ±1 V |
| Working voltage..... | ±10 V |
| Input impedance..... | 144 kΩ |
| Overvoltage protection..... | ±35 |
| Trigger source ..... | Software or external digital trigger |
| System noise1 | |
| Single-ended | |
| ±10 V range..... | 5 mVrms |
| Differential | |
| ±20 V range ..... | 5 mVrms |
| ±1 V range..... | 0.5 mVrms |

**Analog Output**

| | |
|---|---|
| Converter type ..... | Successive approximation |
| Analog outputs..... | 2 |
| Output resolution ..... | 12 bits |
| Maximum update rate ..... | 150 Hz, software-timed |
| Output range ..... | 0 to +5 V |
| Output impedance ..... | 50 □ |
| Output current drive ..... | 5 mA |
| Power-on state..... | 0 V |
| Slew rate..... | 1 V/□s |

Short circuit current ..... 50 mA

Absolute accuracy (no load) ..... 7 mV typical, 36.4 mV maximum at

full scale

**Digital I/O**

P0.<0..7>..... 8 lines

P1.<0..3>..... 4 lines

Direction control ..... Each channel individually programmable as input or output

Output driver type

USB-6008 ..... Open collector (open-drain)

Compatibility ..... TTL, LVTTL, CMOS

Absolute maximum voltage range ..... –0.5 to 5.8 V with respect to GND

Pull-up resistor ..... 4.7 k□to 5 V

Power-on state..... Input

| Level | Min | Max | Units |
|---|---|---|---|
| Input low voltage | –0.3 | 0.8 | V |
| Input high voltage | 2.0 | 5.8 | V |
| Input leakage current | — | 50 | µA |
| Output low voltage (I = 8.5 mA) | — | 0.8 | V |
| Output high voltage | | | |
| Active drive (push-pull), I = –8.5 mA | 2.0 | 3.5 | V |
| Open collector (open-drain), I = –0.6 mA, nominal | 2.0 | 5.0 | V |
| Open collector (open-drain), I = –8.5 mA, with external pull-up resistor | 2.0 | — | V |

**External Voltage**

| | |
|---|---|
| +5 V output (200 mA maximum)..... | +5 V typical, +4.85 V minimum |
| +2.5 V output (1 mA maximum)..... | +2.5 V typical |
| +2.5 V accuracy..... | 0.25% max |
| Reference temperature drift ..... | 50 ppm/°C max |

**Counter**

| | |
|---|---|
| Number of counters ..... | 1 |
| Resolution..... | 32 bits |
| Counter measurements ..... | Edge counting (falling-edge) |
| Counter direction ..... | Count up |
| Pull-up resistor..... | 4.7 k□to 5 V |
| Maximum input frequency ..... | 5 MHz |
| Minimum high pulse width..... | 100 ns |
| Minimum low pulse width..... | 100 ns |
| Input high voltage ..... | 2.0 V |
| Input low voltage ..... | 0.8 V |

**Bus Interface**

| | |
|---|---|
| USB specification ..... | USB 2.0 full-speed |
| USB bus speed ..... | 12 Mb/s |

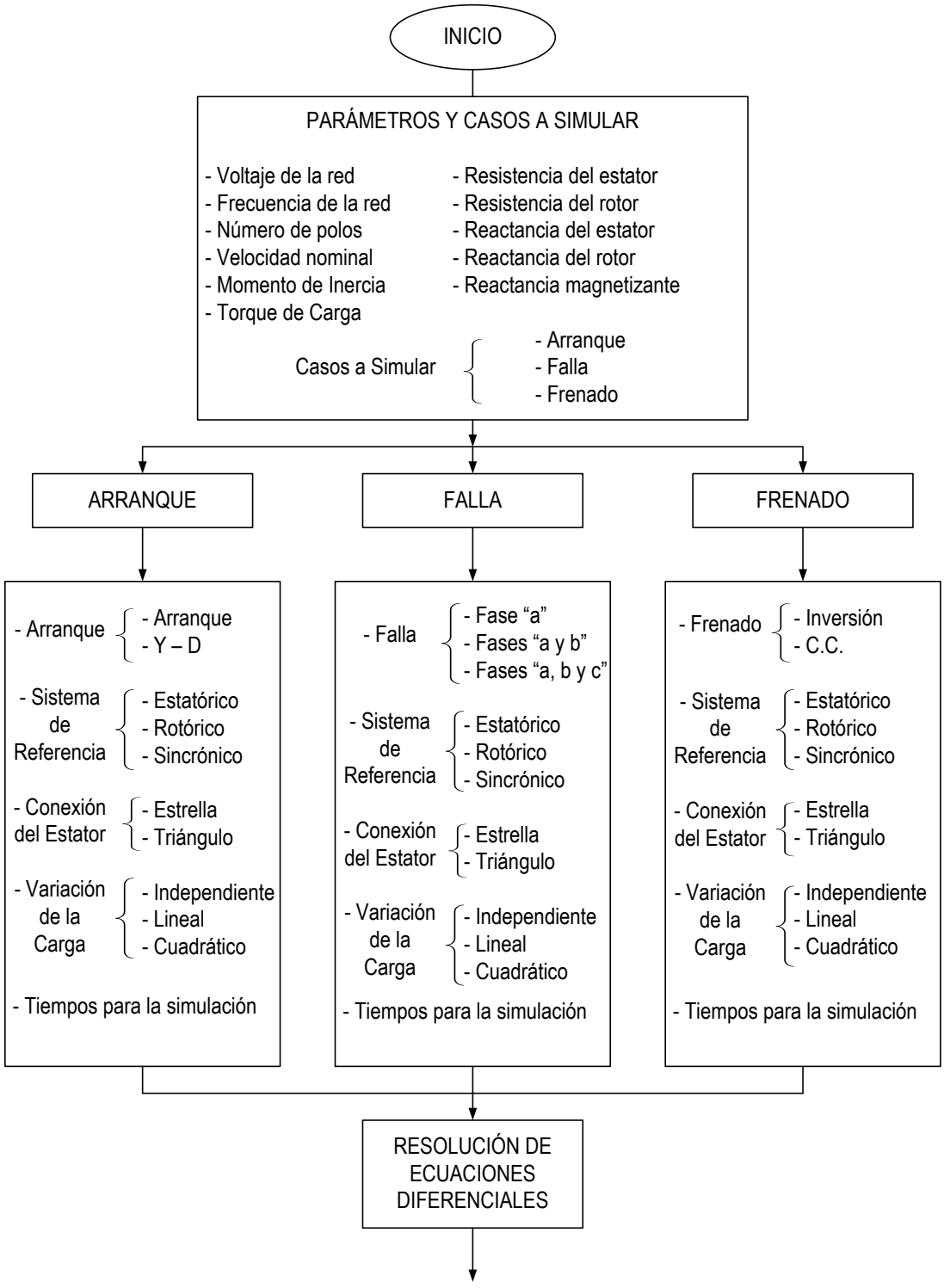
**Power Requirements**

| | |
|---|---|
| USB | |
| 4.10 to 5.25 VDC..... | 80 mA typical, 500 mA max |
| USB suspend..... | 300 mA typical, 500 mA max |

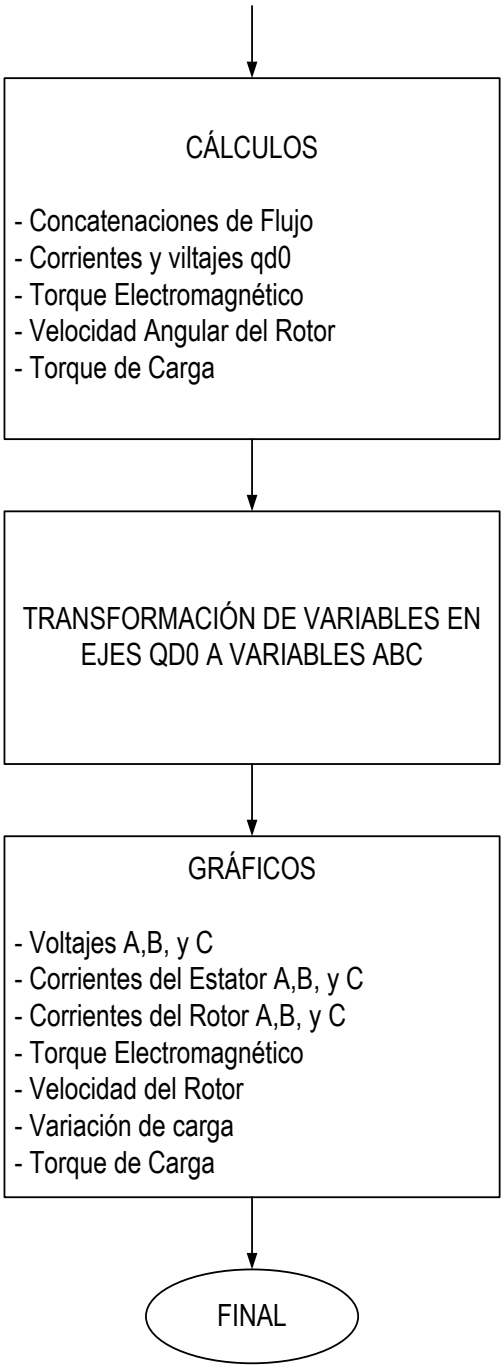
**Physical Characteristics**

If you need to clean the module, wipe it with a dry towel.

| | |
|---|---|
| Dimensions | |
| Without connectors ..... | 6.35 cm □8.51 cm □2.31 cm (2.50 in. □3.35 in. □0.91 in.) |
| With connectors ..... | 8.18 cm □8.51 cm □2.31 cm (3.22 in. □3.35 in. □0.91 in.) |
| I/O connectors ..... | USB series B receptacle, (2) 16 position terminal block plug headers |
| Weight | |
| With connectors ..... | 84 g (3 oz) |
| Without connectors ..... | 54 g (1.9 oz) |
| Screw-terminal wiring..... | 16 to 28 AWG |
| Torque for screw terminals ..... | 0.22–0.25 N · m (2.0–2.2 lb · in.) |




| | | |
|---|---|---|
| **ANEXO C** | **Estructura del Programa** | **2 – 2** |


CÁLCULOS
- Concatenaciones de Flujo
- Corrientes y viltajes qd0
- Torque Electromagnético
- Velocidad Angular del Rotor
- Torque de Carga
TRANSFORMACIÓN DE VARIABLES EN EJES QD0 A VARIABLES ABC
GRÁFICOS
- Voltajes A,B, y C
- Corrientes del Estator A,B, y C
- Corrientes del Rotor A,B, y C
- Torque Electromagnético
- Velocidad del Rotor
- Variación de carga
- Torque de Carga
FINAL

**Principal.m**

```
function varargout = Principal(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Principal_OpeningFcn, ...
                   'gui_OutputFcn',  @Principal_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
function Principal_OpeningFcn(hObject, eventdata, handles, varargin)
%Colocar Imagen de fondo
background = imread('sello.jpeg', 'jpeg'); %Leer imagen
axes(handles.background); %Carga la imagen en background
axis off;
imshow(background); %Presenta la imagen
handles.output = hObject;
guidata(hObject, handles);
```

```
function varargout = Principal_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function Salir_1_Callback(hObject, eventdata, handles)
Final
function Ing_1_Callback(hObject, eventdata, handles)
Simulacion
close(Principal)
```

**Simulacion.m**

```
function varargout = Simulacion(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Simulacion_OpeningFcn, ...
                   'gui_OutputFcn',  @Simulacion_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 Simulacion is made visible.
function Simulacion_OpeningFcn(hObject, eventdata, handles, varargin)
% Choose default command line output for Simulacion
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
initialize_gui(hObject, handles, false);
% --- Outputs from this function are returned to the command line.
function varargout = Simulacion_OutputFcn(hObject, eventdata, handles)
% Get default command line output from handles structure
varargout{1} = handles.output;
function V_red_Callback(hObject, eventdata, handles)
V_red= str2double(get(hObject, 'String'));
if isnan(V_red)
    set(hObject, 'String', 220);
    errordlg('Dato debe ser un numero','Error');
end
% Graba el nuevo valor
handles.datosmaquina.V_red = V_red;
guidata(hObject,handles)
% --- Executes during object creation, after setting all properties.
function V_red_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function X_mag_Callback(hObject, eventdata, handles)
% hObject    handle to X_mag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
X_mag = str2double(get(hObject, 'String'));
if isnan(X_mag)
   set(hObject, 'String', 48.183);
   errordlg('Dato debe ser un numero','Error');
end
% Graba el nuevo valor
handles.datosmaquina.X_mag = X_mag;
guidata(hObject,handles)
% Hints: get(hObject,'String') returns contents of X_mag as text
%        str2double(get(hObject,'String')) returns contents of X_mag as a double
% --- Executes during object creation, after setting all properties.
function X_mag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to X_mag (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

   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function X_rot_Callback(hObject, eventdata, handles)
% hObject    handle to X_rot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
X_rot = str2double(get(hObject, 'String'));
if isnan(X_rot)
   set(hObject, 'String', 3.3199);
   errordlg('Dato debe ser un numero','Error');
end
% Graba el nuevo valor
handles.datosmaquina.X_rot = X_rot;
guidata(hObject,handles)
% Hints: get(hObject,'String') returns contents of X_rot as text
%        str2double(get(hObject,'String')) returns contents of X_rot as a double
% --- Executes during object creation, after setting all properties.
function X_rot_CreateFcn(hObject, eventdata, handles)
% hObject    handle to X_rot (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
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function X_est_Callback(hObject, eventdata, handles)
% hObject    handle to X_est (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
X_est = str2double(get(hObject, 'String'));
if isnan(X_est)
   set(hObject, 'String', 2.2133);
   errordlg('Dato debe ser un numero','Error');
end
% Graba el nuevo valor
handles.datosmaquina.X_est = X_est;
guidata(hObject,handles)
% Hints: get(hObject,'String') returns contents of X_est as text
%        str2double(get(hObject,'String')) returns contents of X_est as a double
% --- Executes during object creation, after setting all properties.
function X_est_CreateFcn(hObject, eventdata, handles)
% hObject    handle to X_est (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
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function R_rot_Callback(hObject, eventdata, handles)
% hObject    handle to R_rot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
R_rot = str2double(get(hObject, 'String'));
if isnan(R_rot)
   set(hObject, 'String', 2.51);
   errordlg('Dato debe ser un numero','Error');
end
% Graba el nuevo valor
handles.datosmaquina.R_rot = R_rot;
guidata(hObject,handles)
% Hints: get(hObject,'String') returns contents of R_rot as text
%        str2double(get(hObject,'String')) returns contents of R_rot as a double
% --- Executes during object creation, after setting all properties.
function R_rot_CreateFcn(hObject, eventdata, handles)
% hObject    handle to R_rot (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
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function R_est_Callback(hObject, eventdata, handles)
% hObject    handle to R_est (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
R_est = str2double(get(hObject, 'String'));
if isnan(R_est)
   set(hObject, 'String', 2.6);
   errordlg('Dato debe ser un numero','Error');
end
% Graba el nuevo valor
handles.datosmaquina.R_est = R_est;
guidata(hObject,handles)
% Hints: get(hObject,'String') returns contents of R_est as text
%        str2double(get(hObject,'String')) returns contents of R_est as a double
% --- Executes during object creation, after setting all properties.
function R_est_CreateFcn(hObject, eventdata, handles)
% hObject    handle to R_est (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
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function T_carga_Callback(hObject, eventdata, handles)
% hObject    handle to T_carga (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
T_carga = str2double(get(hObject, 'String'));
if isnan(T_carga)
   set(hObject, 'String', 0.01);
   errordlg('Dato debe ser un numero','Error');
end
% Graba el nuevo valor
handles.datosmaquina.T_carga = T_carga;
guidata(hObject,handles)
% Hints: get(hObject,'String') returns contents of T_carga as text
%        str2double(get(hObject,'String')) returns contents of T_carga as a double
% --- Executes during object creation, after setting all properties.
function T_carga_CreateFcn(hObject, eventdata, handles)
% hObject    handle to T_carga (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
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function M_ine_Callback(hObject, eventdata, handles)
% hObject    handle to M_ine (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
M_ine = str2double(get(hObject, 'String'));
if isnan(M_ine)
   set(hObject, 'String', 0.0015);
   errordlg('Dato debe ser un numero','Error');
end
% Graba el nuevo valor
handles.datosmaquina.M_ine = M_ine;
guidata(hObject,handles)
% Hints: get(hObject,'String') returns contents of M_ine as text
%        str2double(get(hObject,'String')) returns contents of M_ine as a double
% --- Executes during object creation, after setting all properties.
function M_ine_CreateFcn(hObject, eventdata, handles)
% hObject    handle to M_ine (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
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function Vel_nom_Callback(hObject, eventdata, handles)
% hObject    handle to Vel_nom (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Vel_nom = str2double(get(hObject, 'String'));
if isnan(Vel_nom)
   set(hObject, 'String', 1660);
   errordlg('Dato debe ser un numero','Error');
end
% Graba el nuevo valor
handles.datosmaquina.Vel_nom = Vel_nom;
guidata(hObject,handles)
% Hints: get(hObject,'String') returns contents of Vel_nom as text
%        str2double(get(hObject,'String')) returns contents of Vel_nom as a
double
% --- Executes during object creation, after setting all properties.
function Vel_nom_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Vel_nom (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
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function N_polos_Callback(hObject, eventdata, handles)
% hObject    handle to N_polos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
N_polos = str2double(get(hObject, 'String'));
if isnan(N_polos)
   set(hObject, 'String', 4);
   errordlg('Dato debe ser un numero','Error');
end
% Graba el nuevo valor
handles.datosmaquina.N_polos = N_polos;
guidata(hObject,handles)
% Hints: get(hObject,'String') returns contents of N_polos as text
%        str2double(get(hObject,'String')) returns contents of N_polos as a double
% --- Executes during object creation, after setting all properties.
function N_polos_CreateFcn(hObject, eventdata, handles)
% hObject    handle to N_polos (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
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function f_red_Callback(hObject, eventdata, handles)
% hObject    handle to f_red (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
f_red = str2double(get(hObject, 'String'));
if isnan(f_red)
   set(hObject, 'String', 60);
   errordlg('Dato debe ser un numero','Error');
end
% Graba el nuevo valor
handles.datosmaquina.f_red = f_red;
guidata(hObject,handles)
% Hints: get(hObject,'String') returns contents of f_red as text
%        str2double(get(hObject,'String')) returns contents of f_red as a double
% --- Executes during object creation, after setting all properties.
function f_red_CreateFcn(hObject, eventdata, handles)
% hObject    handle to f_red (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
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function Arr_Callback(hObject, eventdata, handles)
% hObject    handle to Arr (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 Arr
% --- Executes on button press in Fre.
function Fre_Callback(hObject, eventdata, handles)
% hObject    handle to Fre (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 Fre
% --- Executes on button press in Fal.
function Fal_Callback(hObject, eventdata, handles)
% hObject    handle to Fal (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 Fal
% --- Executes on button press in Ing.
function Sig1_Callback(hObject, eventdata, handles)
% hObject    handle to Sig1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
V_red=handles.datosmaquina.V_red ;
f_red=handles.datosmaquina.f_red ;
N_polos=handles.datosmaquina.N_polos ;
Vel_nom=handles.datosmaquina.Vel_nom ;
M_ine=handles.datosmaquina.M_ine;
```

```
T_carga=handles.datosmaquina.T_carga ;
R_est=handles.datosmaquina.R_est ;
R_rot=handles.datosmaquina.R_rot ;
X_est=handles.datosmaquina.X_est ;
X_rot=handles.datosmaquina.X_rot ;
X_mag=handles.datosmaquina.X_mag ;
cs = get(handles.Casos_Simu, 'Value');
dat=fopen('dato_inicial.dat','w');
h=[V_red; f_red; N_polos; Vel_nom; M_ine; T_carga; R_est; R_rot; X_est; X_rot; X_mag; cs];
fprintf(dat,'%5.5f %5.5f %5.5f %5.5f %5.5f %5.5f %5.5f %5.5f %5.5f %5.5f %5.5f %5.5f \n',h);
fclose(dat);
switch cs
   case 1
      Arranques;
   case 2
      Fallas;
   case 3
      Frenados;
end
close(Simulacion);

function Regresar_1_Callback(hObject, eventdata, handles)

Principal
close(Simulacion)
```

```
function Casos_Simu_Callback(hObject, eventdata, handles)
% --- Executes during object creation, after setting all properties.
function Casos_Simu_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Casos_Simu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
% --- Executes on button press in Salir_general.
% --- Executes on selection change in Tip_Ref.
function Tip_Ref_Callback(hObject, eventdata, handles)
% hObject    handle to Tip_Ref (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = get(hObject,'String') returns Tip_Ref contents as cell array
%        contents{get(hObject,'Value')} returns selected item from Tip_Ref
% --- Executes during object creation, after setting all properties.
function Tip_Ref_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Tip_Ref (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: popupmenu controls usually have a white background on Windows.
```

```
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
% --------------------------------------------------------------------
function initialize_gui(fig_handle, handles, isreset)
% If the metricdata field is present and the reset flag is false, it means
% we are we are just re-initializing a GUI by calling it from the cmd line
% while it is up. So, bail out as we dont want to reset the data.
if isfield(handles, 'datosmaquina') && ~isreset
   return;
end
handles.datosmaquina.V_red = 220;
handles.datosmaquina.f_red = 60;
handles.datosmaquina.N_polos = 4;
handles.datosmaquina.Vel_nom = 1660;
handles.datosmaquina.M_ine = 0.0015;
handles.datosmaquina.T_carga = 0.01;
handles.datosmaquina.R_est = 2.6;
handles.datosmaquina.R_rot = 2.51;
handles.datosmaquina.X_est = 2.2133;
handles.datosmaquina.X_rot = 3.3199;
handles.datosmaquina.X_mag = 48.183;
 set(handles.V_red, 'String', handles.datosmaquina.V_red);
set(handles.f_red, 'String', handles.datosmaquina.f_red);
```

```
set(handles.N_polos, 'String', handles.datosmaquina.N_polos);
set(handles.Vel_nom, 'String', handles.datosmaquina.Vel_nom);
set(handles.M_ine, 'String', handles.datosmaquina.M_ine);
set(handles.T_carga, 'String', handles.datosmaquina.T_carga);
set(handles.R_est, 'String', handles.datosmaquina.R_est);
set(handles.R_rot, 'String', handles.datosmaquina.R_rot);
set(handles.X_est, 'String', handles.datosmaquina.X_est);
set(handles.X_rot, 'String', handles.datosmaquina.X_rot);
set(handles.X_mag, 'String', handles.datosmaquina.X_mag);
% Update handles structure
guidata(handles.figure1, handles);
```

**Arranque.m**

```
function varargout = Arranques(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Arranques_OpeningFcn, ...
'gui_OutputFcn',  @Arranques_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 Arranques is made visible.
function Arranques_OpeningFcn(hObject, eventdata, handles, varargin)
% Choose default command line output for Arranques
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
initialize_gui(hObject, handles, false);
function varargout = Arranques_OutputFcn(hObject, eventdata, handles)
% Get default command line output from handles structure
varargout{1} = handles.output;
% --- Executes on selection change in Arr_dir.
function Arr_dir_Callback(hObject, eventdata, handles)
ta = get(handles.Arr_dir, 'Value');
switch ta
    case 1
%     set(handles.uipanel6,'visible','off')
        set(handles.Con_estator,'visible','on')
        set(handles.Con_estrella,'visible','on')
        set(handles.Con_triangulo,'visible','on')
        set(handles.tyad,'visible','off')
        set(handles.Tiempofalla,'visible','off')
    case 2
```

```
        set(handles.Con_estator,'visible','on')
        set(handles.Con_estrella,'visible','off')
        set(handles.Con_triangulo,'visible','off')
        set(handles.tyad,'visible','on')
        set(handles.Tiempofalla,'visible','on')
end
% --- Executes during object creation, after setting all properties.
function Arr_dir_CreateFcn(hObject, eventdata, handles)
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
% --- Executes on button press in Sim_1.
function Sim_1_Callback(hObject, eventdata, handles)
Tfinal=handles.datosarranque.Tfinal ;
tyad=handles.datosarranque.tyad ;
ta = get(handles.Arr_dir, 'Value');
sr = get(handles.Elec_SR, 'Value');
ee = get(handles.Con_estrella, 'Value');
et = get(handles.Con_triangulo, 'Value');
iv = get(handles.Ind_vel, 'Value');
cv = get(handles.Cuad_velocidad, 'Value');
lv = get(handles.Linel_velocidad, 'Value');
dat=fopen('dato_arranque.dat','w');
r=[Tfinal; tyad; ta; sr; ee; et; iv; lv; cv ];
```

```
fprintf(dat,'%5.5f %5.5f %5.5f %5.5f %5.5f %5.5f %5.5f %5.5f %5.5f \n',r);
fclose(dat);
dat=fopen('dato_arranque.dat','r');
r=fscanf(dat,'%f',inf);
fclose(dat);
options = odeset('RelTol',1e-4,'AbsTol',[1e-4 1e-4 1e-4 1e-4 1e-4 1e-4 1e-4],'MaxStep',0.001 );
[t,x] = ode45('ecdif',[0:0.001:r(1,1)],[0,0,0,0,0,0,0],options);
GraficosArranque
close(Arranques)
function Con_estrella_Callback(hObject, eventdata, handles)
set(handles.Con_estrella,'value',1)
set(handles.Con_triangulo,'value',0)
% --- Executes on button press in Con_triangulo.
function Con_triangulo_Callback(hObject, eventdata, handles)
set(handles.Con_estrella,'value',0)
set(handles.Con_triangulo,'value',1)
% --- Executes on button press in Ind_vel.
function Ind_vel_Callback(hObject, eventdata, handles)
set(handles.Ind_vel,'value',1)
set(handles.Cuad_velocidad,'value',0)
set(handles.Linel_velocidad,'value',0)
% --- Executes on button press in Cuad_velocidad.
function Cuad_velocidad_Callback(hObject, eventdata, handles)
set(handles.Ind_vel,'value',0)
set(handles.Cuad_velocidad,'value',1)
                    set(handles.Linel_velocidad,'value',0)
```

```
% --- Executes on button press in Linel_velocidad.
function Linel_velocidad_Callback(hObject, eventdata, handles)
set(handles.Ind_vel,'value',0)
set(handles.Cuad_velocidad,'value',0)
set(handles.Linel_velocidad,'value',1)
function Tfinal_Callback(hObject, eventdata, handles)
Tfinal = str2double(get(hObject, 'String'));
if isnan(Tfinal)
   set(hObject, 'String', 1);
   errordlg('Dato debe ser un numero','Error');
end
% if handles.tyad >= handles.Tfinal
%     errordlg('Dato debe ser mayor al tiempo de cambio Y-Delta','Error');
% end
% if handles.Tfinal==0
%       errordlg('Dato debe ser mayor a cero y mayor al tiempo de cambio Y-Delta','Error');
% end
% Graba el nuevo valor
handles.datosarranque.Tfinal = Tfinal;
guidata(hObject,handles)
% --- Executes during object creation, after setting all properties.
function Tfinal_CreateFcn(hObject, eventdata, handles)
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
```

```
end
function tyad_Callback(hObject, eventdata, handles)
tyad = str2double(get(hObject, 'String'));
if isnan(tyad)
   set(hObject, 'String', 0.5);
   errordlg('Dato debe ser un numero','Error');
end
% if handles.tyad >= handles.Tfinal || handles.tyad == 0
%     errordlg('Dato debe ser menor al tiempo final de simulacion','Error');
% % elseif handles.tyad == 0
% %     errordlg('Dato debe ser mayor a cero','Error');
% end
% Graba el nuevo valor
handles.datosarranque.tyad = tyad;
guidata(hObject,handles)
% --- Executes during object creation, after setting all properties.
function tyad_CreateFcn(hObject, eventdata, handles)
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
function Reg_1_Callback(hObject, eventdata, handles)
Simulacion
close(Arranques)
```

```
% --- Executes during object creation, after setting all properties.
function figure1_CreateFcn(hObject, eventdata, handles)
% --- Executes on selection change in Elec_SR.
function Elec_SR_Callback(hObject, eventdata, handles)
% Hints: contents = get(hObject,'String') returns Elec_SR contents as cell array
%        contents{get(hObject,'Value')} returns selected item from Elec_SR
% --- Executes during object creation, after setting all properties.
function Elec_SR_CreateFcn(hObject, eventdata, handles)
% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
function initialize_gui(fig_handle, handles, isreset)
% If the metricdata field is present and the reset flag is false, it means
% we are we are just re-initializing a GUI by calling it from the cmd line
% while it is up. So, bail out as we dont want to reset the data.
if isfield(handles, 'datosarranque') && ~isreset
    return;
end
handles.datosarranque.Tfinal = 1;
handles.datosarranque.tyad = 0.4;
set(handles.Tfinal, 'String', handles.datosarranque.Tfinal);
set(handles.tyad, 'String', handles.datosarranque.tyad);
guidata(handles.figure1, handles);
```

**Fallas.m**

```
function varargout = Fallas(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Fallas_OpeningFcn, ...
                   'gui_OutputFcn',  @Fallas_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 Fallas is made visible.
function Fallas_OpeningFcn(hObject, eventdata, handles, varargin)
% Choose default command line output for Fallas
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
initialize_gui(hObject, handles, false);
% --- Outputs from this function are returned to the command line.
```

```
function varargout = Fallas_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;
% --- Executes on selection change in Casos_falla.
function Casos_falla_Callback(hObject, eventdata, handles)
% hObject    handle to Casos_falla (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
cf = get(handles.Casos_falla, 'Value');
switch cf
   case 1
   case 2
   case 3
end
% Hints: contents = get(hObject,'String') returns Casos_falla contents as cell array
%        contents{get(hObject,'Value')} returns selected item from Casos_falla
% --- Executes during object creation, after setting all properties.
function Casos_falla_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Casos_falla (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: popupmenu controls usually have a white background on Windows.
```

```
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
% --- Executes on button press in Sim_1.
function Sim_1_Callback(hObject, eventdata, handles)
% hObject    handle to Sim_1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Tfinal = handles.datosfalla.Tfinal ;
tyad = handles.datosfalla.tyad ;
cf = get(handles.Casos_falla, 'Value');
sr1 = get(handles.Eleccion_SR, 'Value');
ee1 = get(handles.Con_estrella, 'Value');
et1 = get(handles.Con_triangulo, 'Value');
iv1 = get(handles.Ind_vel, 'Value');
lv1 = get(handles.Linel_velocidad, 'Value');
cv1 = get(handles.Cuad_velocidad, 'Value');
dat=fopen('dato_falla.dat','w');
g=[Tfinal; tyad; cf; sr1; ee1; et1; iv1; lv1; cv1];
fprintf(dat,'%5.5f %5.5f %5.5f %5.5f %5.5f %5.5f %5.5f %5.5f %5.5f \n',g);
fclose(dat);
dat=fopen('dato_falla.dat','r');
g=fscanf(dat,'%f',inf);
fclose(dat);
options = odeset('RelTol',1e-4,'AbsTol',[1e-4 1e-4 1e-4 1e-4 1e-4 1e-4 1e-
```

```
4],'MaxStep',0.001 );
[t,x] = ode45('ecdif',[0,g(1,1)],[0,0,0,0,0,0,0],options);
GraficosFalla
close(Fallas)
% --- Executes on button press in Con_estrella.
function Con_estrella_Callback(hObject, eventdata, handles)
% hObject    handle to Con_estrella (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.Con_estrella,'value',1)
set(handles.Con_triangulo,'value',0)
% Hint: get(hObject,'Value') returns toggle state of Con_estrella
% --- Executes on button press in Con_triangulo.
function Con_triangulo_Callback(hObject, eventdata, handles)
% hObject    handle to Con_triangulo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.Con_estrella,'value',0)
set(handles.Con_triangulo,'value',1)
% Hint: get(hObject,'Value') returns toggle state of Con_triangulo
% --- Executes on button press in Ind_vel.
function Ind_vel_Callback(hObject, eventdata, handles)
% hObject    handle to Ind_vel (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.Ind_vel,'value',1)
set(handles.Cuad_velocidad,'value',0)
```

```
set(handles.Linel_velocidad,'value',0)
% Hint: get(hObject,'Value') returns toggle state of Ind_vel
% --- Executes on button press in Cuad_velocidad.
function Cuad_velocidad_Callback(hObject, eventdata, handles)
% hObject    handle to Cuad_velocidad (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.Ind_vel,'value',0)
set(handles.Cuad_velocidad,'value',1)
set(handles.Linel_velocidad,'value',0)
% Hint: get(hObject,'Value') returns toggle state of Cuad_velocidad
% --- Executes on button press in Linel_velocidad.
function Linel_velocidad_Callback(hObject, eventdata, handles)
% hObject    handle to Linel_velocidad (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.Ind_vel,'value',0)
set(handles.Cuad_velocidad,'value',0)
set(handles.Linel_velocidad,'value',1)
% Hint: get(hObject,'Value') returns toggle state of Linel_velocidad
function Tfinal_Callback(hObject, eventdata, handles)
% hObject    handle to Tfinal (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Tfinal = str2double(get(hObject, 'String'));
if isnan(Tfinal)
   set(hObject, 'String', 1);
```

```
errordlg('Dato debe ser un numero','Error');
end
% Graba el nuevo valor
handles.datosfalla.Tfinal = Tfinal;
guidata(hObject,handles)
% Hints: get(hObject,'String') returns contents of Tfinal as text
%        str2double(get(hObject,'String')) returns contents of Tfinal as a double
% --- Executes during object creation, after setting all properties.
function Tfinal_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Tfinal (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
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
function tyad_Callback(hObject, eventdata, handles)
% hObject    handle to tyad (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
tyad = str2double(get(hObject, 'String'));
if isnan(tyad)
   set(hObject, 'String', 0.5);
   errordlg('Dato debe ser un numero','Error');
```

```
end
% Graba el nuevo valor
handles.datosfalla.tyad = tyad;
guidata(hObject,handles)
% Hints: get(hObject,'String') returns contents of tyad as text
%        str2double(get(hObject,'String')) returns contents of tyad as a double
% --- Executes during object creation, after setting all properties.
function tyad_CreateFcn(hObject, eventdata, handles)
% hObject    handle to tyad (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
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
% --- Executes on button press in Reg_1.
function Reg_1_Callback(hObject, eventdata, handles)
% hObject    handle to Reg_1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Simulacion
close(Fallas)
% --- Executes during object creation, after setting all properties.
function figure1_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% --- Executes on selection change in Eleccion_SR.
function Eleccion_SR_Callback(hObject, eventdata, handles)
% hObject    handle to Eleccion_SR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = get(hObject,'String') returns Eleccion_SR contents as cell
array
%        contents{get(hObject,'Value')} returns selected item from Eleccion_SR
% --- Executes during object creation, after setting all properties.
function Eleccion_SR_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Eleccion_SR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
function initialize_gui(fig_handle, handles, isreset)
% If the metricdata field is present and the reset flag is false, it means
% we are we are just re-initializing a GUI by calling it from the cmd line
% while it is up. So, bail out as we dont want to reset the data.
```

```
if isfield(handles, 'datosfalla') && ~isreset
    return;
end
handles.datosfalla.Tfinal = 1;
handles.datosfalla.tyad = 0.6;
 set(handles.Tfinal, 'String', handles.datosfalla.Tfinal);
set(handles.tyad, 'String', handles.datosfalla.tyad);
% Update handles structure
guidata(handles.figure1, handles);
```

**Frenado.m**

```
function varargout = Frenados(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Frenados_OpeningFcn, ...
                   'gui_OutputFcn',  @Frenados_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 Frenados is made visible.
function Frenados_OpeningFcn(hObject, eventdata, handles, varargin)
% Choose default command line output for Frenados
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
initialize_gui(hObject, handles, false);
% UIWAIT makes Arranques wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = Frenados_OutputFcn(hObject, eventdata, handles)
% Get default command line output from handles structure
varargout{1} = handles.output;
% --- Executes on selection change in Tip_fre.
function Tip_fre_Callback(hObject, eventdata, handles)
tfr = get(handles.Tip_fre, 'Value');
switch tfr
   case 1
      set(handles.V_corrientecontinua,'visible','off')
      set(handles.Vcc,'visible','off')
      set(handles.V,'visible','off')
      set(handles.Con_estator,'visible','on')
      set(handles.Con_estrella,'visible','on')
      set(handles.Con_triangulo,'visible','on')
      set(handles.Tcc,'visible','off')
```

| | | |
|---|---|---|
| **ANEXO D** | **Líneas de Programación** | **35 – 71** |

```
set(handles.Tiemp_CC,'visible','off')
        set(handles.tyad,'visible','on')
        set(handles.Tiemp_inv,'visible','on')
    case 2
        set(handles.Con_estator,'visible','on')
        set(handles.Con_estrella,'visible','on')
        set(handles.Con_triangulo,'visible','on')
        set(handles.tyad,'visible','off')
        set(handles.Tiemp_inv,'visible','off')
        set(handles.Tcc,'visible','on')
        set(handles.Tiemp_CC,'visible','on')
        set(handles.V_corrientecontinua,'visible','on')
        set(handles.Vcc,'visible','on')
        set(handles.V,'visible','on')
end
% --- Executes during object creation, after setting all properties.
function Tip_fre_CreateFcn(hObject, eventdata, handles)
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
% --- Executes on button press in Sim_1.
function Sim_1_Callback(hObject, eventdata, handles)
Tfinal=handles.datosfrenado.Tfinal ;
tyad=handles.datosfrenado.tyad ;
```

```
Tcc=handles.datosfrenado.Tcc ;
Vcc=handles.datosfrenado.Vcc ;
tfr=get(handles.Tip_fre, 'Value');
sr2=get(handles.Eleccion_SR, 'Value');
ee2=get(handles.Con_estrella, 'Value');
et2=get(handles.Con_triangulo, 'Value');
iv2=get(handles.Ind_vel, 'Value');
lv2=get(handles.Linel_velocidad, 'Value');
cv2=get(handles.Cuad_velocidad, 'Value');
dat=fopen('dato_frenado.dat','w');
k=[Tfinal; tyad; Tcc; Vcc; tfr; sr2; ee2; et2; iv2; lv2; cv2 ];
fprintf(dat,'%5.5f %5.5f %5.5f %5.5f %5.5f %5.5f %5.5f %5.5f %5.5f %5.5f \n',k);
fclose(dat);
dat=fopen('dato_frenado.dat','r');
k=fscanf(dat,'%f',inf);
fclose(dat);
options = odeset('RelTol',1e-4,'AbsTol',[1e-4 1e-4 1e-4 1e-4 1e-4 1e-4 1e-4],'MaxStep',0.001 );
[t,x] = ode45('ecdif',[0,k(1,1)],[0,0,0,0,0,0,0],options);
GraficosFrenado
close(Frenados)
% --- Executes on button press in Con_estrella.
function Con_estrella_Callback(hObject, eventdata, handles)
set(handles.Con_estrella,'value',1)
set(handles.Con_triangulo,'value',0)
% --- Executes on button press in Con_triangulo.
```

| | | |
|---|---|---|
| **ANEXO D** | **Líneas de Programación** | **37 – 71** |

```
function Con_triangulo_Callback(hObject, eventdata, handles)
set(handles.Con_estrella,'value',0)
set(handles.Con_triangulo,'value',1)
% --- Executes on button press in Ind_vel.
function Ind_vel_Callback(hObject, eventdata, handles)
set(handles.Ind_vel,'value',1)
set(handles.Cuad_velocidad,'value',0)
set(handles.Linel_velocidad,'value',0)
% --- Executes on button press in Cuad_velocidad.
function Cuad_velocidad_Callback(hObject, eventdata, handles)
set(handles.Ind_vel,'value',0)
set(handles.Cuad_velocidad,'value',1)
set(handles.Linel_velocidad,'value',0)
% --- Executes on button press in Linel_velocidad.
function Linel_velocidad_Callback(hObject, eventdata, handles)
set(handles.Ind_vel,'value',0)
set(handles.Cuad_velocidad,'value',0)
set(handles.Linel_velocidad,'value',1)
function Tfinal_Callback(hObject, eventdata, handles)
Tfinal = str2double(get(hObject, 'String'));
if isnan(Tfinal)
    set(hObject, 'String', 1);
    errordlg('Dato debe ser un numero','Error');
end
% Graba el nuevo valor
handles.datosfrenado.Tfinal = Tfinal;
guidata(hObject,handles)
```

| | | |
|---|---|---|
| **ANEXO D** | **Líneas de Programación** | **38 – 71** |

```
function Tfinal_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
function tyad_Callback(hObject, eventdata, handles)
tyad = str2double(get(hObject, 'String'));
if isnan(tyad)
    set(hObject, 'String', 0.5);
    errordlg('Dato debe ser un numero','Error');
end
% Graba el nuevo valor
handles.datosfrenado.tyad = tyad;
guidata(hObject,handles)
function tyad_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
function Tcc_Callback(hObject, eventdata, handles)
Tcc = str2double(get(hObject, 'String'));
if isnan(Tcc)
    set(hObject, 'String', 0.5);
    errordlg('Dato debe ser un numero','Error');
end
```

| | | |
|---|---|---|
| **ANEXO D** | **Líneas de Programación** | **39 – 71** |

```
% Graba el nuevo valor
handles.datosfrenado.Tcc = Tcc;
guidata(hObject,handles)
% --- Executes during object creation, after setting all properties.
function Tcc_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
% --- Executes on button press in Reg_1.
function Reg_1_Callback(hObject, eventdata, handles)
Simulacion
close(Frenados)
% --- Executes during object creation, after setting all properties.
function figure1_CreateFcn(hObject, eventdata, handles)
function Vcc_Callback(hObject, eventdata, handles)
Vcc = str2double(get(hObject, 'String'));
if isnan(Vcc)
    set(hObject, 'String', 1);
    errordlg('Dato debe ser un numero','Error');
end
% Graba el nuevo valor
handles.datosfrenado.Vcc = Vcc;
guidata(hObject,handles)
% --- Executes during object creation, after setting all properties.
```

```
function Vcc_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
% --- Executes on selection change in Eleccion_SR.
function Eleccion_SR_Callback(hObject, eventdata, handles)
function Eleccion_SR_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
function initialize_gui(fig_handle, handles, isreset)
if isfield(handles, 'datosfrenado') && ~isreset
    return;
end
handles.datosfrenado.Tfinal = 0.9;
handles.datosfrenado.tyad = 0.6;
handles.datosfrenado.Tcc = 0.3;
handles.datosfrenado.Vcc = 0;
 set(handles.Tfinal, 'String', handles.datosfrenado.Tfinal);
set(handles.tyad, 'String', handles.datosfrenado.tyad);
set(handles.Tcc, 'String', handles.datosfrenado.Tcc);
set(handles.Vcc, 'String', handles.datosfrenado.Vcc);
guidata(handles.figure1, handles);
```

| | | |
|---|---|---|
| **ANEXO D** | **Líneas de Programación** | **41 – 71** |

**ecdif.m**

```
function  [f]= ecdif(t,x);
%Abre el Archivo dato_inicial.dat
dat=fopen('dato_inicial.dat','r');
h=fscanf(dat,'%f',inf);
fclose(dat);
% Datos
V = h(1,1);
freq = h(2,1);
P = h(3,1);
Wn = h(4,1);
J = h(5,1);
Tl = h(6,1);
rs = h(7,1);
rr = h(8,1);
Xls = h(9,1);
Xlr = h(10,1);
Xm = h(11,1);
We = 2*pi*freq;
%Voltajes de entrada
switch h(12,1)
   case 1
      %Abre el Archivo dato_arranque.dat
      dat1=fopen('dato_arranque.dat','r');
      r=fscanf(dat1,'%f',inf);
      fclose(dat1);
            switch r(4,1);
```

```
    case 1
        W=0;
        theta=0;
    case 2
        W=We;
        theta=x(7);
    case 3
        W=x(5);
        theta=x(6);
end
    if r(7,1)==1;
    v=0;
else if r(8,1)==1;
        v=1;
    elseif r(9,1)==1;
        v=2;
    end
end
switch r(3,1)
    case 1
        if r(5,1)==1;
            Vas = (sqrt(2)*V/sqrt(3))*cos(We*t);
            Vbs = (sqrt(2)*V/sqrt(3))*cos(We*t-2*pi/3);
            Vcs = (sqrt(2)*V/sqrt(3))*cos(We*t+2*pi/3);
        elseif r(6,1)==1;
            Vas = sqrt(2)*V*cos(We*t);
            Vbs = sqrt(2)*V*cos(We*t-2*pi/3);
```

```
            Vcs = sqrt(2)*V*cos(We*t+2*pi/3);
        end
    case 2
        if t>=0 & t<= r(2,1);
            Vas = (sqrt(2)*V/sqrt(3))*cos(We*t);
            Vbs = (sqrt(2)*V/sqrt(3))*cos(We*t-2*pi/3);
            Vcs = (sqrt(2)*V/sqrt(3))*cos(We*t+2*pi/3);
        elseif t >= r(2,1) ;
            Vas = sqrt(2)*V*cos(We*t);
            Vbs = sqrt(2)*V*cos(We*t-2*pi/3);
            Vcs = sqrt(2)*V*cos(We*t+2*pi/3);
        end
  end
case 2
  %Abre el Archivo dato_falla.dat
  dat2=fopen('dato_falla.dat','r');
  g=fscanf(dat2,'%f',inf);
  fclose(dat2);
  switch g(4,1);
    case 1
      W=0;
      theta=0;
    case 2
      W=We;
      theta=x(7);
    case 3
      W=x(5);
```

```
        theta=x(6);
end
if g(7,1)==1;
   v=0;
else if g(8,1)==1
        v=1;
   elseif g(9,1)==1;
        v=2;
   end
end
switch g(3,1)
   case 1
      if g(5,1) == 1;
         if t >= 0 & t <= g(2,1);
            Vas = (sqrt(2)*V/sqrt(3))*cos(We*t);
            Vbs = (sqrt(2)*V/sqrt(3))*cos(We*t-2*pi/3);
            Vcs = (sqrt(2)*V/sqrt(3))*cos(We*t+2*pi/3);
         elseif t >= g(2,1);
            Vas = 0;
            Vbs = (sqrt(2)*V/sqrt(3))*cos(We*t-2*pi/3);
            Vcs = (sqrt(2)*V/sqrt(3))*cos(We*t+2*pi/3);
         end
      elseif g(6,1) == 1;
         if t>=0 & t<= g(2,1);
            Vas = sqrt(2)*V*cos(We*t);
            Vbs = sqrt(2)*V*cos(We*t-2*pi/3);
            Vcs = sqrt(2)*V*cos(We*t+2*pi/3);
```

```
            elseif t >= g(2,1);
                Vas = (sqrt(2)*V/sqrt(3))*cos(We*t+pi/6);
                Vbs = (sqrt(2)*V/sqrt(3))*cos(We*t-2*pi/3);
                Vcs = (sqrt(2)*V/sqrt(3))*cos(We*t+2*pi/3-pi/6);
            end
        end
case 2
    if g(5,1) == 1;
        if t>=0 & t <= g(2,1);
         Vas = (sqrt(2)*V/sqrt(3))*cos(We*t);
            Vbs = (sqrt(2)*V/sqrt(3))*cos(We*t-2*pi/3);
            Vcs = (sqrt(2)*V/sqrt(3))*cos(We*t+2*pi/3);
        elseif t>= g(2,1);
            Vas = 0;
            Vbs = 0;
            Vcs = (sqrt(2)*V/sqrt(3))*cos(We*t+2*pi/3);
        end
    elseif g(6,1) == 1;
        if t>=0 & t <= g(2,1);
            Vas = sqrt(2)*V*cos(We*t);
            Vbs = sqrt(2)*V*cos(We*t-2*pi/3);
            Vcs = sqrt(2)*V*cos(We*t+2*pi/3);
        elseif t>=g(2,1);
            Vas = 0;
            Vbs = (sqrt(2)*V/sqrt(3))*cos(We*t-2*pi/3+pi/6);
            Vcs = (sqrt(2)*V/sqrt(3))*cos(We*t+2*pi/3-pi/6);
        end
```

```
    end
case 3
    if g(5,1) == 1;
        if t>=0 & t<= g(2,1);
            Vas = (sqrt(2)*V/sqrt(3))*cos(We*t);
            Vbs = (sqrt(2)*V/sqrt(3))*cos(We*t-2*pi/3);
            Vcs = (sqrt(2)*V/sqrt(3))*cos(We*t+2*pi/3);
        elseif t>=g(2,1);
            Vas=0;
            Vbs=0;
            Vcs=0;
        end
    elseif g(6,1) ==1;
        if t>= 0 & t<= g(2,1);
            Vas = sqrt(2)*V*cos(We*t);
            Vbs = sqrt(2)*V*cos(We*t-2*pi/3);
            Vcs = sqrt(2)*V*cos(We*t+2*pi/3);
        elseif t>= g(2,1);
            Vas=0;
            Vbs=0;
            Vcs=0;
        end
    end
end
    case 3
%Abre el Archivo dato_frenado.dat
dat3=fopen('dato_frenado.dat','r');
```

```
k=fscanf(dat3,'%f',inf);
      fclose(dat3);
      switch k(6,1);
         case 1
            W=0;
            theta=0;
         case 2
            W=We;
            theta=x(7);
         case 3
            W=x(5);
            theta=x(6);
      end
      if k(9,1)==1;
         v=0;
      else if k(10,1)==1;
            v=1;
         elseif k(11,1)==1;
            v=2;
         end
      end
   switch k(5,1)
         case 1
            if k(7,1) == 1;
               if t >= 0 & t <= k(2,1);
                  Vas = (sqrt(2)*V/sqrt(3))*cos(We*t);
                  Vbs = (sqrt(2)*V/sqrt(3))*cos(We*t-2*pi/3);
```

```
            Vcs = (sqrt(2)*V/sqrt(3))*cos(We*t+2*pi/3);
        elseif t >= k(2,1);
            Vas = (sqrt(2)*V/sqrt(3))*cos(We*t);
            Vbs = (sqrt(2)*V/sqrt(3))*cos(We*t+2*pi/3);
            Vcs = (sqrt(2)*V/sqrt(3))*cos(We*t-2*pi/3);
        end
    elseif k(8,1) == 1;
        if t>=0 & t<= k(2,1);
            Vas = sqrt(2)*V*cos(We*t);
            Vbs = sqrt(2)*V*cos(We*t-2*pi/3);
            Vcs = sqrt(2)*V*cos(We*t+2*pi/3);
        elseif t >= k(2,1);
            Vas = -sqrt(2)*V*cos(We*t+2*pi/3);
            Vbs = -sqrt(2)*V*cos(We*t-2*pi/3);
            Vcs = -sqrt(2)*V*cos(We*t);
        end
    end
case 2
    if k(7,1) == 1;
        if t>=0 & t <= k(3,1);
            Vas = (sqrt(2)*V/sqrt(3))*cos(We*t);
            Vbs = (sqrt(2)*V/sqrt(3))*cos(We*t-2*pi/3);
            Vcs = (sqrt(2)*V/sqrt(3))*cos(We*t+2*pi/3);
        elseif t>= k(3,1);
            Vas = k(4,1)/2;
            Vbs = -(k(4,1)/2);
            Vcs = 0;
```

```
            end
        elseif k(8,1) == 1;
            if t>=0 & t <= k(3,1);
                Vas = sqrt(2)*V*cos(We*t);
                Vbs = sqrt(2)*V*cos(We*t-2*pi/3);
                Vcs = sqrt(2)*V*cos(We*t+2*pi/3);
            elseif t>=k(3,1);
                Vas = k(4,1);
                Vbs = -(k(4,1)/2);
                Vcs = -(k(4,1)/2);
            end
        end
    end
end
%Voltajes de abc a qd0
Vqs = (2/3)*(Vas*cos(theta)+Vbs*cos(theta-2*pi/3)+Vcs*cos(theta+2*pi/3));
Vds = (2/3)*(Vas*sin(theta)+Vbs*sin(theta-2*pi/3)+Vcs*sin(theta+2*pi/3));
Vqr=0;
Vdr=0;
Xs = Xls + Xm;
Xr = Xlr + Xm;
D = Xs*Xr - Xm*Xm;
%Corrientes
iqs = (Xr*x(1) - Xm*x(3))/D;
ids = (Xr*x(2) - Xm*x(4))/D;
iqr = (Xs*x(3) - Xm*x(1))/D;
idr = (Xs*x(4) - Xm*x(2))/D;
```

| | | |
|---|---|---|
| **ANEXO D** | **Líneas de Programación** | **50 – 71** |

```
%Torques y velocidades
Te = (3/4)*(P/We)*((x(2)*iqs)-(x(1)*ids));
Wr =((freq /(P*pi))*(x(5)));
Tc = Tl*((Wr/Wn)^(v));
f(5) = (Te-Tc)/J;%Wr en rad/seg
f(6) = x(5);%Thetar
f(7)=We;%Thetae
% Ecuaciones Diferenciales
f(1) = We*[Vqs-(rs/D)*(Xr*x(1)-Xm*x(3))-(W/We)*x(2)];
f(2) = We*[Vds-(rs/D)*(Xr*x(2)-Xm*x(4))+(W/We)*x(1)];
f(3) = We*[Vqr-(rr/D)*(Xs*x(3)-Xm*x(1))-((W-x(5))/We)*x(4)];
f(4) = We*[Vdr-(rr/D)*(Xs*x(4)-Xm*x(2))+((W-x(5))/We)*x(3)];
f = f';
%Corrientes del estator en ejes abc
ias=(iqs*cos(theta)+ids*sin(theta));
ibs=(iqs*cos(theta-2*pi/3)+ids*sin(theta-2*pi/3));
ics=(iqs*cos(theta+2*pi/3)+ids*sin(theta+2*pi/3));
%Corrientes del rotor en ejes abc
iar=(iqr*cos(theta-x(6))+idr*sin(theta-x(6)));
ibr=(iqr*cos(theta-x(6)-2*pi/3)+idr*sin(theta-x(6)-2*pi/3));
icr=(iqr*cos(theta-x(6)+2*pi/3)+idr*sin(theta-x(6)+2*pi/3));
datos=fopen('vol.dat','a');
z=[Vas; Vbs; Vcs; ias; ibs; ics; iar; ibr; icr; Te; Wr; Tc];
fprintf(datos,'%5.5f \t %5.5f \t %5.5f \t %5.5f \t %5.5f \t %5.5f \t %5.5f \t %5.5f \t %5.5f \t %5.5f \t %5.5f \t \n',z);
fclose(datos);
end
```

| | | |
|---|---|---|
| **ANEXO D** | **Líneas de Programación** | **51 – 71** |

**GraficosArranque.m**

```
function varargout = GraficosArranque(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GraficosArranque_OpeningFcn, ...
                   'gui_OutputFcn',  @GraficosArranque_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 GraficosArranque is made visible.
function GraficosArranque_OpeningFcn(hObject, eventdata, handles, varargin)
% Choose default command line output for GraficosArranque
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
function varargout = GraficosArranque_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
```

```
% --- Executes on selection change in Tipo_grafico.
function Tipo_grafico_Callback(hObject, eventdata, handles)
% --- Executes during object creation, after setting all properties.
function Tipo_grafico_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Tipo_grafico (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
     % --- Executes on button press in Reg_3.
function Reg_3_Callback(hObject, eventdata, handles)
% hObject    handle to Reg_3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Arranques
close(GraficosArranque)

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

```
datos = fopen('vol.dat','r');
z = fscanf(datos,'%f',[12 inf]);
fclose(datos);
dat=fopen('dato_arranque.dat','r');
r=fscanf(dat,'%f',inf);
fclose(dat);
axes(handles.Eje1);
tg = get(handles.Tipo_grafico, 'Value');
switch tg
    case 1
    axes(handles.Eje1);
    z1=z(1,:);
    za=z1;
    s=0:(r(1,1)/(length(za)-1)):r(1,1);
    plot(s,za);
    grid;
    title('Voltaje Fase "A". Vas');
    axes(handles.Eje2);
    z2=z(2,:);
    zb=z2;
    s=0:(r(1,1)/(length(zb)-1)):r(1,1);
    plot(s,zb);
    grid;
    title('Voltaje Fase "B". Vbs');
    axes(handles.Eje3);
    z3=z(3,:);
    zc=z3;
```

| | | |
|---|---|---|
| **ANEXO D** | **Líneas de Programación** | **54 – 71** |

```
s=0:(r(1,1)/(length(zc)-1)):r(1,1);
        plot(s,zc);
        grid;
        title('Voltaje Fase "C". Vcs');
    case 2
        axes(handles.Eje1);
        z4=z(4,:);
        zd=z4;
        s=0:(r(1,1)/(length(zd)-1)):r(1,1);
        plot(s,zd);
        grid;
        title('Corriente Fase "A". Ias');
        axes(handles.Eje2);
        z5=z(5,:);
        ze=z5;
        s=0:(r(1,1)/(length(ze)-1)):r(1,1);
        plot(s,ze);
        grid;
        title('Corriente Fase "B". Ibs');
        axes(handles.Eje3);
        z6=z(6,:);
        zf=z6;
        s=0:(r(1,1)/(length(zf)-1)):r(1,1);
        plot(s,zf);
        grid;
        title('Corriente Fase "C". Ics');
```

```
case 3
    axes(handles.Eje1);
    z7=z(7,:);
    zg=z7;
    s=0:(r(1,1)/(length(zg)-1)):r(1,1);
    plot(s,zg);
    grid;
    title('Corriente Fase "A". Iar');
    axes(handles.Eje2);
    z8=z(8,:);
    zh=z8;
    s=0:(r(1,1)/(length(zh)-1)):r(1,1);
    plot(s,zh);
    grid;
    title('Corriente Fase "B". Ibr');
    axes(handles.Eje3);
    z9=z(9,:);
    zi=z9;
    s=0:(r(1,1)/(length(zi)-1)):r(1,1);
    plot(s,zi);
    grid;
    title('Corriente Fase "C". Icr');
case 4
    axes(handles.Eje1);
    z10=z(10,:);
    zj=z10;
    s=0:(r(1,1)/(length(zj)-1)):r(1,1);
```

```
        plot(s,zj);
        grid;
        title('Torque Electromagnetico. Te');
        axes(handles.Eje2);
        z11=z(11,:);
        zk=z11;
        s=0:(r(1,1)/(length(zk)-1)):r(1,1);
        plot(s,zk);
        grid;
        title('Velocidad Rotorica. Wr');
        axes(handles.Eje3);
        z12=z(12,:);
       zl=z12;
        s=0:(r(1,1)/(length(zl)-1)):r(1,1);
        plot(s,zl);
        grid;
        title('Torque de Carga. Tc');
end
```

**GraficosFalla.m**

```
function varargout = GraficosFalla(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GraficosFalla_OpeningFcn, ...
                   'gui_OutputFcn',  @GraficosFalla_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 GraficosFalla is made visible.
function GraficosFalla_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
function varargout = GraficosFalla_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function Tipo_grafico_Callback(hObject, eventdata, handles)
function Tipo_grafico_CreateFcn(hObject, eventdata, handles)
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
function Reg_3_Callback(hObject, eventdata, handles)
Fallas
close(GraficosFalla)
```

| | | |
|---|---|---|
| **ANEXO D** | **Líneas de Programación** | **58 – 71** |

```
% --- Executes on button press in Act_graf.
function Act_graf_Callback(hObject, eventdata, handles)
datos = fopen('vol.dat','r');
z = fscanf(datos,'%f',[12 inf]);
fclose(datos);
dat=fopen('dato_falla.dat','r');
g=fscanf(dat,'%f',inf);
fclose(dat);
axes(handles.Eje1);
tg2 = get(handles.Tipo_grafico, 'Value');
switch tg2
   case 1
      axes(handles.Eje1);
      z1=z(1,:);
      za=z1;
      s=0:(g(1,1)/(length(za)-1)):g(1,1);
      plot(s,za);
      grid;
      title('Voltaje Fase "A". Vas');
      axes(handles.Eje2);
      z2=z(2,:);
      zb=z2;
      s=0:(g(1,1)/(length(zb)-1)):g(1,1);
      plot(s,zb);
      grid;
      title('Voltaje Fase "B". Vbs');
      axes(handles.Eje3);
```

| | | |
|---|---|---|
| **ANEXO D** | **Líneas de Programación** | **59 – 71** |

```
        z3=z(3,:);
        zc=z3;
        s=0:(g(1,1)/(length(zc)-1)):g(1,1);
        plot(s,zc);
        grid;
        title('Voltaje Fase "C". Vcs');
case 2
        axes(handles.Eje1);        z4=z(4,:);
        zd=z4;
        s=0:(g(1,1)/(length(zd)-1)):g(1,1);
        plot(s,zd);
        grid;
        title('Corriente Fase "A". Ias');
        axes(handles.Eje2);
        z5=z(5,:);
        ze=z5;
        s=0:(g(1,1)/(length(ze)-1)):g(1,1);
        plot(s,ze);
        grid;
        title('Corriente Fase "B". Ibs');
        axes(handles.Eje3);
        z6=z(6,:);
        zf=z6;
        s=0:(g(1,1)/(length(zf)-1)):g(1,1);
        plot(s,zf);
        grid;
        title('Corriente Fase "C". Ics');
```

```
case 3
    axes(handles.Eje1);
    z7=z(7,:);
    zg=z7;
    s=0:(g(1,1)/(length(zg)-1)):g(1,1);
    plot(s,zg);
    grid;
    title('Corriente Fase "A". Iar');
    axes(handles.Eje2);
    z8=z(8,:);
    zh=z8;
    s=0:(g(1,1)/(length(zh)-1)):g(1,1);
    plot(s,zh);
    grid;
     title('Corriente Fase "B". Ibr');
    axes(handles.Eje3);
    z9=z(9,:);
    zi=z9;
    s=0:(g(1,1)/(length(zi)-1)):g(1,1);
    plot(s,zi);
    grid;
    title('Corriente Fase "C". Icr');
case 4
    axes(handles.Eje1);
    z10=z(10,:);
    zj=z10;
    s=0:(g(1,1)/(length(zj)-1)):g(1,1);
```

```
        plot(s,zj);
        grid;
        title('Torque Electromagnetico. Te');
        axes(handles.Eje2);
        z11=z(11,:);
        zk=z11;
        s=0:(g(1,1)/(length(zk)-1)):g(1,1);
        plot(s,zk);
        grid;
        title('Velocidad Rotorica. Wr');
        axes(handles.Eje3);
        z12=z(12,:);
        zl=z12;
        s=0:(g(1,1)/(length(zl)-1)):g(1,1);
        plot(s,zl);
        grid;
        title('Torque de Carga. Tc');
end
```

**GraficosFrenado.m**

```
function varargout = GraficosFrenado(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
                ‘gui_State = struct('gui_Name',       mfilename, ...
                'gui_Singleton',  gui_Singleton, ...
                'gui_OpeningFcn', @GraficosFrenado_OpeningFcn, ...
```

```
                'gui_OutputFcn',  @GraficosFrenado_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
function GraficosFrenado_OpeningFcn(hObject, eventdata, handles, varargin)
% Choose default command line output for GraficosFrenado
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
function varargout = GraficosFrenado_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
% --- Executes on selection change in Tipo_grafico.
function Tipo_grafico_Callback(hObject, eventdata, handles)
function Tipo_grafico_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

| | | |
|---|---|---|
| **ANEXO D** | **Líneas de Programación** | **63 - 71** |

```
% --- Executes on button press in Reg_3.
function Reg_3_Callback(hObject, eventdata, handles)
Frenados
close(GraficosFrenado)
% --- Executes on button press in Act_graf.
function Act_graf_Callback(hObject, eventdata, handles)
datos = fopen('vol.dat','r');
z = fscanf(datos,'%f',[12 inf]);
fclose(datos);
        dat=fopen('dato_frenado.dat','r');
         k=fscanf(dat,'%f',inf);
fclose(dat);
axes(handles.Eje1);
tg1 = get(handles.Tipo_grafico, 'Value');
switch tg1
    case 1
        axes(handles.Eje1);
        z1=z(1,:);
        za=z1;
        s=0:(k(1,1)/(length(za)-1)):k(1,1);
        plot(s,za);
        grid;
        title('Voltaje Fase "A". Vas');
        axes(handles.Eje2);
        z2=z(2,:);
        zb=z2;
```

| | | |
|---|---|---|
| **ANEXO D** | **Líneas de Programación** | **64 - 71** |

```
s=0:(k(1,1)/(length(zb)-1)):k(1,1);
        plot(s,zb);
        grid;
        title('Voltaje Fase "B". Vbs');
        axes(handles.Eje3);
        z3=z(3,:);
        zc=z3;
        s=0:(k(1,1)/(length(zc)-1)):k(1,1);
        plot(s,zc);
        grid;
        title('Voltaje Fase "C". Vcs');
    case 2
        axes(handles.Eje1);
        z4=z(4,:);
        zd=z4;
        s=0:(k(1,1)/(length(zd)-1)):k(1,1);
        plot(s,zd);
        grid;
        title('Corriente Fase "A". Ias');
        axes(handles.Eje2);
        z5=z(5,:);
        ze=z5;
        s=0:(k(1,1)/(length(ze)-1)):k(1,1);
        plot(s,ze);
        grid;
        title('Corriente Fase "B". Ibs');
```

```
    axes(handles.Eje3);
    z6=z(6,:);
    zf=z6;
    s=0:(k(1,1)/(length(zf)-1)):k(1,1);
    plot(s,zf);
    grid;
    title('Corriente Fase "C". Ics');
case 3
    axes(handles.Eje1);
    z7=z(7,:);
    zg=z7;
    s=0:(k(1,1)/(length(zg)-1)):k(1,1);
    plot(s,zg);
    grid;
    title('Corriente Fase "A". Iar');
    axes(handles.Eje2);
    z8=z(8,:);
    zh=z8;
    s=0:(k(1,1)/(length(zh)-1)):k(1,1);
    plot(s,zh);
    grid;
    title('Corriente Fase "B". Ibr');
    axes(handles.Eje3);
    z9=z(9,:);
    zi=z9;
    s=0:(k(1,1)/(length(zi)-1)):k(1,1);
```

```
        plot(s,zi);
        grid;
        title('Corriente Fase "C". Icr');
    case 4
        axes(handles.Eje1);
        z10=z(10,:);
        zj=z10;
        s=0:(k(1,1)/(length(zj)-1)):k(1,1);
        plot(s,zj);
        grid;
        title('Torque Electromagnetico. Te');
        axes(handles.Eje2);
        z11=z(11,:);
        zk=z11;
        s=0:(k(1,1)/(length(zk)-1)):k(1,1);
        plot(s,zk);
        grid;
        title('Velocidad Rotorica. Wr');
        axes(handles.Eje3);
        z12=z(12,:);
        zl=z12;
        s=0:(k(1,1)/(length(zl)-1)):k(1,1);
        plot(s,zl);
        grid;
        title('Torque de Carga. Tc');
end
```

| | | |
|---|---|---|
| **ANEXO D** | **Líneas de Programación** | **67 - 71** |

**Final.m**

```
function varargout = Final(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Final_OpeningFcn, ...
                   'gui_OutputFcn',  @Final_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 Final is made visible.
function Final_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = 'Yes';
% Update handles structure
guidata(hObject, handles);

if(nargin > 3)
    for index = 1:2:(nargin-3),
```

```
        if nargin-3==index, break, end
        switch lower(varargin{index})
         case 'title'
          set(hObject, 'Name', varargin{index+1});
         case 'string'
          set(handles.text1, 'String', varargin{index+1});
        end
    end
end
% Determine the position of the dialog - centered on the callback figure
% if available, else, centered on the screen
FigPos=get(0,'DefaultFigurePosition');
OldUnits = get(hObject, 'Units');
set(hObject, 'Units', 'pixels');
OldPos = get(hObject,'Position');
FigWidth = OldPos(3);
FigHeight = OldPos(4);
if isempty(gcbf)
    ScreenUnits=get(0,'Units');
    set(0,'Units','pixels');
    ScreenSize=get(0,'ScreenSize');
    set(0,'Units',ScreenUnits);
    FigPos(1)=1/2*(ScreenSize(3)-FigWidth);
    FigPos(2)=2/3*(ScreenSize(4)-FigHeight);
else
    GCBFOldUnits = get(gcbf,'Units');
  set(gcbf,'Units','pixels');
```

| | | |
|---|---|---|
| **ANEXO D** | **Líneas de Programación** | **69 - 71** |

```
   GCBFPos = get(gcbf,'Position');
   set(gcbf,'Units',GCBFOldUnits);
   FigPos(1:2) = [(GCBFPos(1) + GCBFPos(3) / 2) - FigWidth / 2, ...
                  (GCBFPos(2) + GCBFPos(4) / 2) - FigHeight / 2];
end
FigPos(3:4)=[FigWidth FigHeight];
set(hObject, 'Position', FigPos);
set(hObject, 'Units', OldUnits);
% Show a question icon from dialogicons.mat - variables questIconData
% and questIconMap
load dialogicons.mat
IconData=questIconData;
questIconMap(256,:) = get(handles.figure1, 'Color');
IconCMap=questIconMap;
Img=image(IconData, 'Parent', handles.axes1);
set(handles.figure1, 'Colormap', IconCMap);
set(handles.axes1, ...
   'Visible', 'off', ...
   'YDir'   , 'reverse'       , ...
   'XLim'   , get(Img,'XData'), ...
   'YLim'   , get(Img,'YData')  ...
   );
% Make the GUI modal
set(handles.figure1,'WindowStyle','modal')
% UIWAIT makes Final wait for user response (see UIRESUME)
uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
```

```
function varargout = Final_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
% The figure can be deleted now
delete(handles.figure1);
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
close(Principal)
handles.output = get(hObject,'String');
% Update handles structure
guidata(hObject, handles);
% Use UIRESUME instead of delete because the OutputFcn needs
% to get the updated handles structure.
uiresume(handles.figure1);
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
handles.output = get(hObject,'String');
% Update handles structure
guidata(hObject, handles);
% Use UIRESUME instead of delete because the OutputFcn needs
% to get the updated handles structure.
uiresume(handles.figure1);
% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
if isequal(get(handles.figure1, 'waitstatus'), 'waiting')
    % The GUI is still in UIWAIT, us UIRESUME
    uiresume(handles.figure1);
else
```

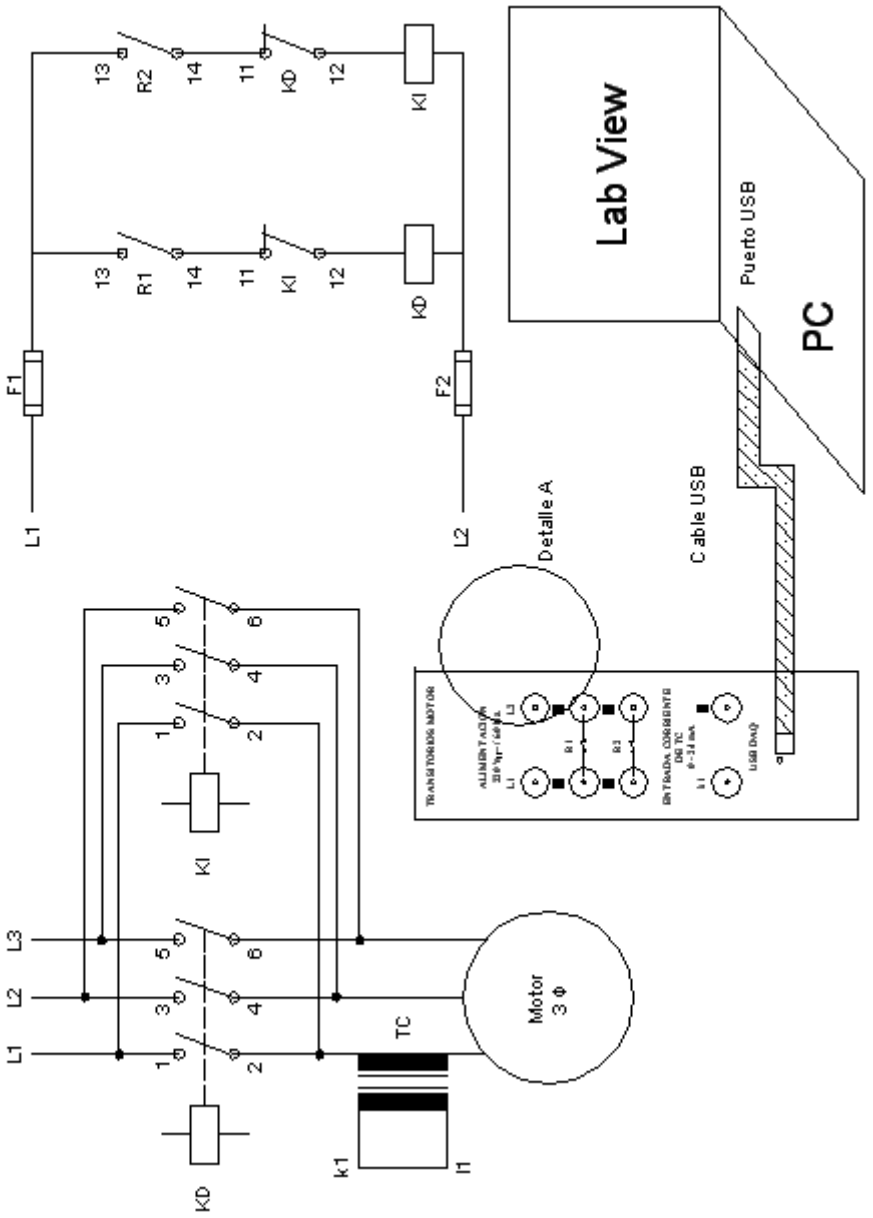
```
    % The GUI is no longer waiting, just close it
    delete(handles.figure1);
end
% --- Executes on key press over figure1 with no controls selected.
function figure1_KeyPressFcn(hObject, eventdata, handles)
if isequal(get(hObject,'CurrentKey'),'escape')
    % User said no by hitting escape
    handles.output = 'No';
    % Update handles structure
    guidata(hObject, handles);
    uiresume(handles.figure1);
end
if isequal(get(hObject,'CurrentKey'),'return')
    uiresume(handles.figure1);
end
```

**Parametros.m**

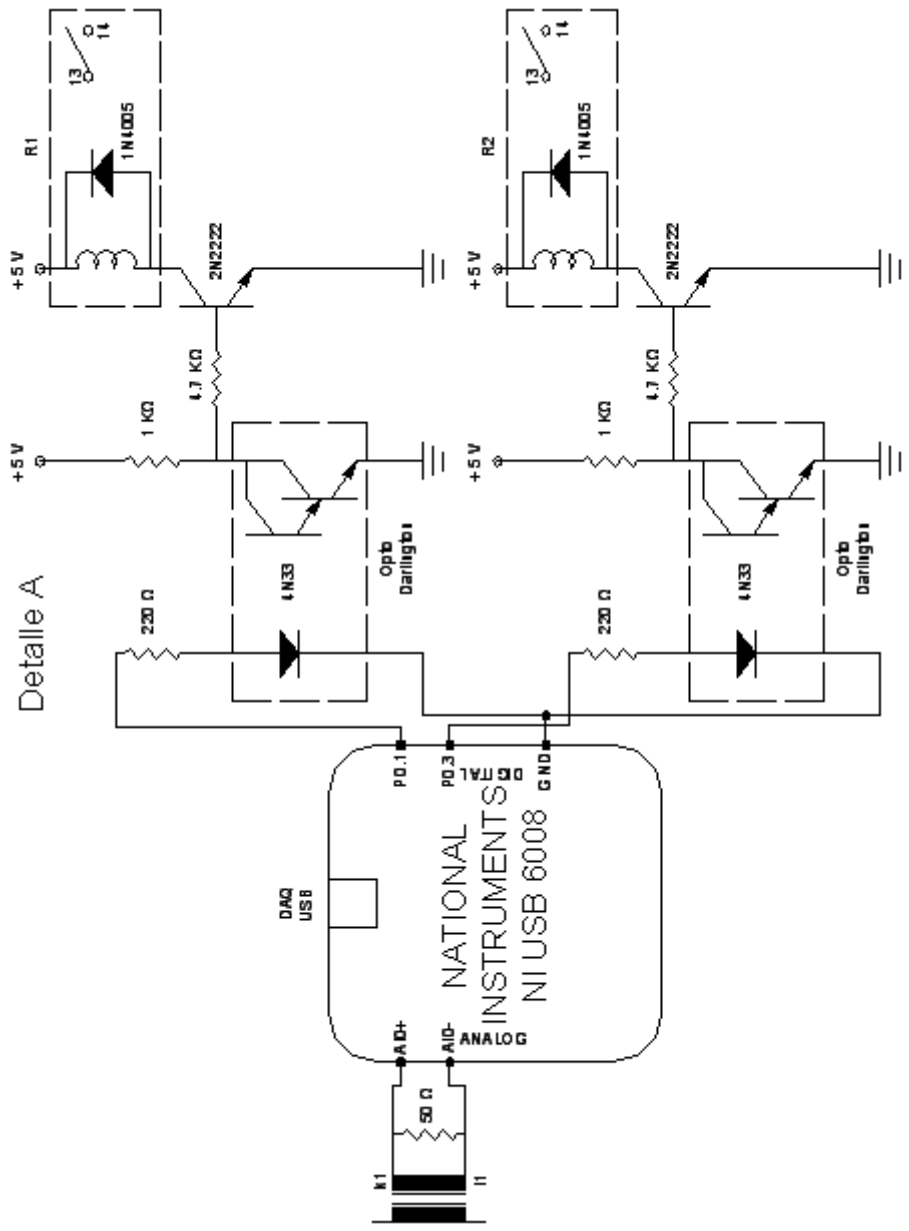
```
clear;
p=4;
f=60;
design = 'B';
R1= 2.6;

%Datos de la prueba a rotor bloqueado
Vbr=56.1/sqrt(3);
Ibr=4.3;
Pbr=283.5/3;
fbr=60;

%Valores de prueba de vacio
Vnl=210/sqrt(3);
Inl=2.4;
Pnl=90/3;
nnl=1796;
%Calculos
Req=Pbr/Ibr^2;
Zbr=Vbr/Ibr;
Xeq=f/fbr*sqrt(Zbr^2-Req^2);
R2pr=Req-R1;

if design=='B';
        X1=0.4*Xeq;
        X2pr=0.6*Xeq;
```

```
elseif design=='C';
        X1=0.3*Xeq;
        X2pr=0.7*Xeq;
else;
        X1=0.5*Xeq;
        X2pr=0.5*Xeq;
end
thetan1=acos(Pnl/Vnl/Inl);
ns=120*60/p;
s=(ns-nnl)/ns;
E1=Vnl-Inl*exp(-j*thetan1)*(R1+j*X1);
I2pr=E1/(R2pr/s+j*X2pr);
E1=abs(E1);
I2pr=abs(I2pr);
Pc=Pnl-Inl^2*R1-I2pr^2*R2pr/s;
Rc=E1^2/Pc;
if Pc<0;
        disp('Algo esta mal');
end;
Qm=Vnl*Inl*sin(thetan1)-Inl^2*X1-I2pr^2*X2pr;
Xm=E1^2/Qm;
ParametrosMotor=['Rs      "       Rr      "       Xs      "       Xr      "       Xm
        ']
format short e
[       R1      R2pr   X1      X2pr   Xm     ]
```

| | | |
|---|---|---|
| **ANEXO F** | **Diagrama de conexiones** | **1 - 2** |




| | | |
|---|---|---|
| **ANEXO F** | **Diagrama de conexiones** | **2- 2** |


R1
R2
1N4005
2N2222
+5 V
4.7 KΩ
1 KΩ
4N33
Opto Darlington
220 Ω
Detalle A
NATIONAL INSTRUMENTS
NI USB 6008
DAQ USB
P0.1
P0.3
DIGITAL GND
GND
AI0+
AI0-
ANALOG
50 Ω
K1

El software de simulación fue desarrollado en el programa computacional Matlab 7.0, por lo que previamente se debe tener instalado Matlab 7.0 o superior en la máquina en donde se instalara el software de simulación. Por lo que se seguirá los siguientes pasos para instalar el software y su utilización.

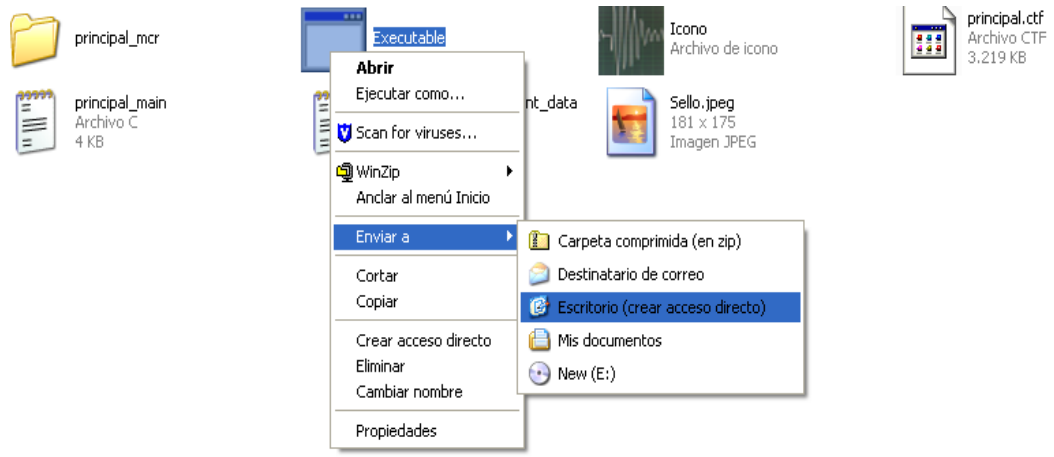
1.- Instalar Matlab 7.0 o superior.

2.- Copiar la carpeta “Modelación y Simulación de las Máquinas de Inducción” que se encuentra en el CD proporcionado y se pega en alguna ubicación de la computadora a utilizar de recomendación hacerlo en el escritorio de la computadora.


3.- Reiniciar la computadora.

4.- Dentro de la carpeta “Modelación y Simulación de las Máquinas de Inducción” se encuentran una serie de archivos, en los cuales hay uno de nombre “Executable”.

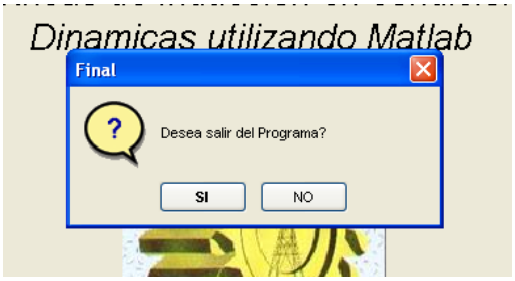
Este archivo al hacer doble clic ya se puede realizar la simulación, sin embargo se recomienda que se realice un acceso directo al escritorio, al hacerlo dentro de la carpeta “Modelación y Simulación de las Máquinas de Inducción” existe también un archivo llamado Icono el cual puede servir una vez que ya se ha realizado el acceso directo al escritorio para poder colocarlo como el icono del acceso directo.



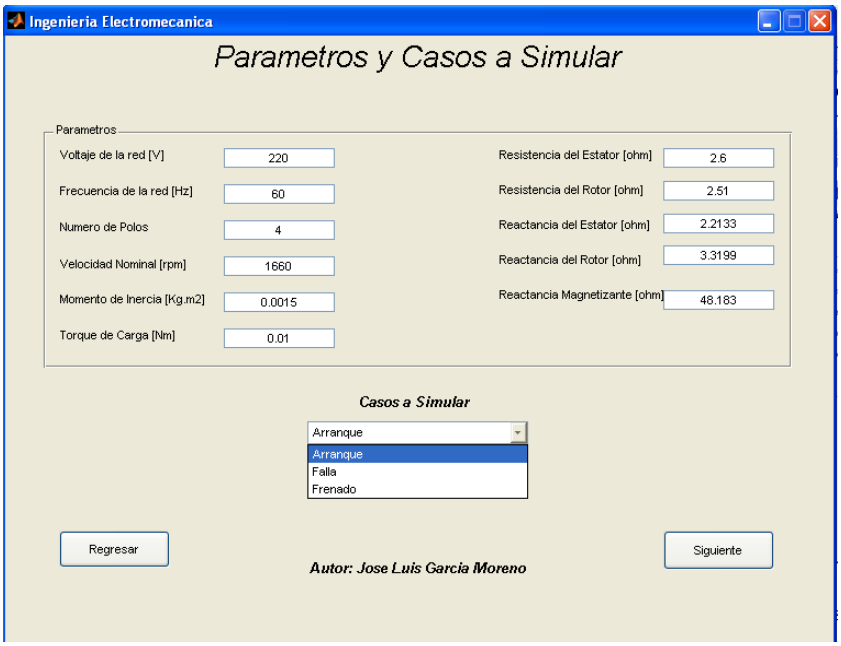

Terminado la instalación del software de simulación cabe decir que cualquiera que fuese el procedimiento para correr el programa sea directamente desde la carpeta “Modelación y Simulación de las Máquinas de Inducción” a través del archivo Executable o a través del acceso directo en el escritorio, cualquier cambio o incremento de archivos como se vera más adelante se lo hará siempre en la carpeta raíz.

Para manipular en si el programa se empezará desde el escritorio en este caso el procedimiento es a través del acceso directo. Entonces en el Acceso directo se hace doble clic y la primera pantalla en aparecer es la siguiente:


La pantalla anterior es la de ingreso en donde lo único que permite es ingresar o salir del programa a través de los botones que se encuentran en el lado derecho e izquierdo respectivamente. Para seguir con la explicación de cómo utilizar el software se elegirá en primer lugar la opción salir para luego seguir con el Ingreso. Al hacer clic en la opción salir la siguiente pantalla aparece en donde al poner NO el software permanece en la pantalla de ingreso y si se coloca SI se sale completamente del software.



La opción anterior es únicamente para salir del programa al volver a la pantalla de ingreso ahora se hará clic en el botón de Ingresar y a continuación se presente la siguiente pantalla:




La pantalla “Parámetros y Casos a Simular” es general para todas las simulaciones pues aquí se ingresan las características eléctricas y mecánicas del motor así como también la elección del caso a simular como son arranque, frenado o falla. Cada uno de los casilleros tienen la siguiente explicación:

Parámetros:

- Voltaje de la red [V]: Es el voltaje que se aplicará al motor para la simulación.
- Frecuencia de la red [Hz]: Es el valor de frecuencia a la que funciona la red, a menos que se desee cambiar este valor siempre será 60.

- Número de Polos: Es el valor del número de polos del motor.
- Velocidad Nominal [rpm]: Es el valor de la placa de datos del motor correspondiente a la velocidad nominal del mismo.
- Torque de Carga [Nm]: Este valor es una condición de simulación que se elegirá para simular carga.
- Resistencia del estator y rotor, reactancia de estator y rotor y reactancia magnetizante todos en [ohmios], son los valores eléctricos que determinan el tipo de motor a simular y que se obtienen de las pruebas en vacío y rotor bloqueado realizado al motor a simular.

Casos a Simular:

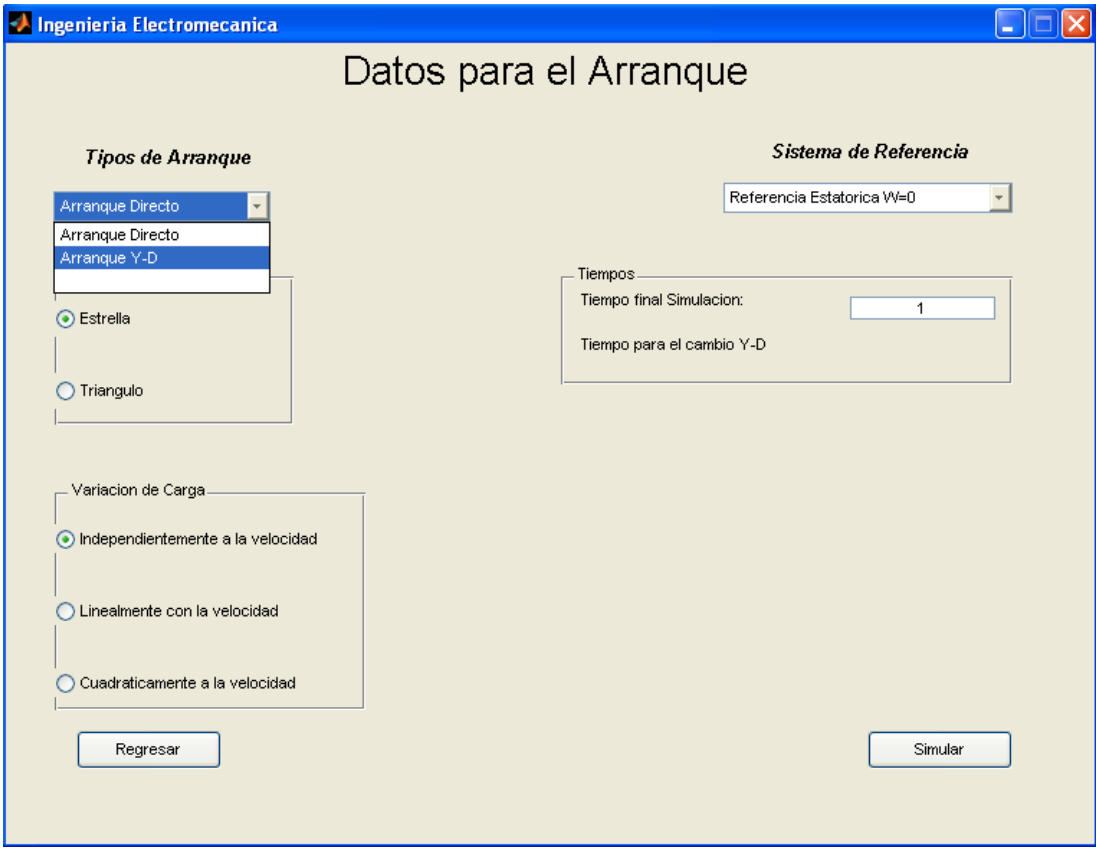
Esta ventana de selección permite elegir el caso a simular pudiéndose elegir entre:

- Arranque
- Falla
- Frenado

Una vez ingresado todos los datos en los casillero se elegirá en primer lugar el Arranque para la correspondiente explicación:

**1.- Arranque**

En la ventana de selección “Casos a Simular” de la figura anterior se elige Arranque y se hace clic en el icono Siguiente y esto permite ingresar a la siguiente pantalla:




La pantalla “Datos para el Arranque” es exclusiva para el arranque en la misma se determinan todas las características del tipo de arranque, condiciones de carga, conexión del estator, sistema de referencia a utilizar y los tiempos de simulación, es así que en la ventana de selección del Tipo de Arranque se podrá elegir entre:

- Arranque Directo
- Arranque Y – D.

Al elegir Arranque Directo se activa todas las características mencionadas aunque

en lo que respecto a los tiempos únicamente se activa la casilla tiempo final de simulación puesto que no tiene ningún cambio de condición, en este casilla se deberá ingresar el tiempo en el que se desea que termine la simulación. En cambio si se elige en la ventana Tipo de Arranque el correspondiente a Arranque Y - D se desactivan las casillas de conexión del estator y se activa en la ventana Tiempo un casillero más que corresponde al Tiempo para el cambio Y – D correspondiente al tiempo en el que se desea que se realice la conmutación a conexión D (Triángulo). La ventana Conexión del Estator permite elegir la conexión del estator ya sea este en estrella o en triángulo, excepto en la simulación del arranque Y – D en donde no se puede elegir la conexión en todas las demás simulaciones se elegirá la conexión del estator.

La ventana Variación de Carga permite elegir como la carga variará en función de la velocidad pudiéndose elegir según la característica de carga lo siguiente:

- Independiente a la velocidad: Bombas de pistón, aparatos de elevación, cintas transportadoras con la masa transportada constante, también en los mecanismos de avance de las máquinas – herramientas, etc.

- Linealmente con la velocidad: Este tipo de cargas no son muy utilizadas pero como a manera de explicación es cuando se enciende un generador de excitación independiente que actúa sobre una resistencia externa fija.

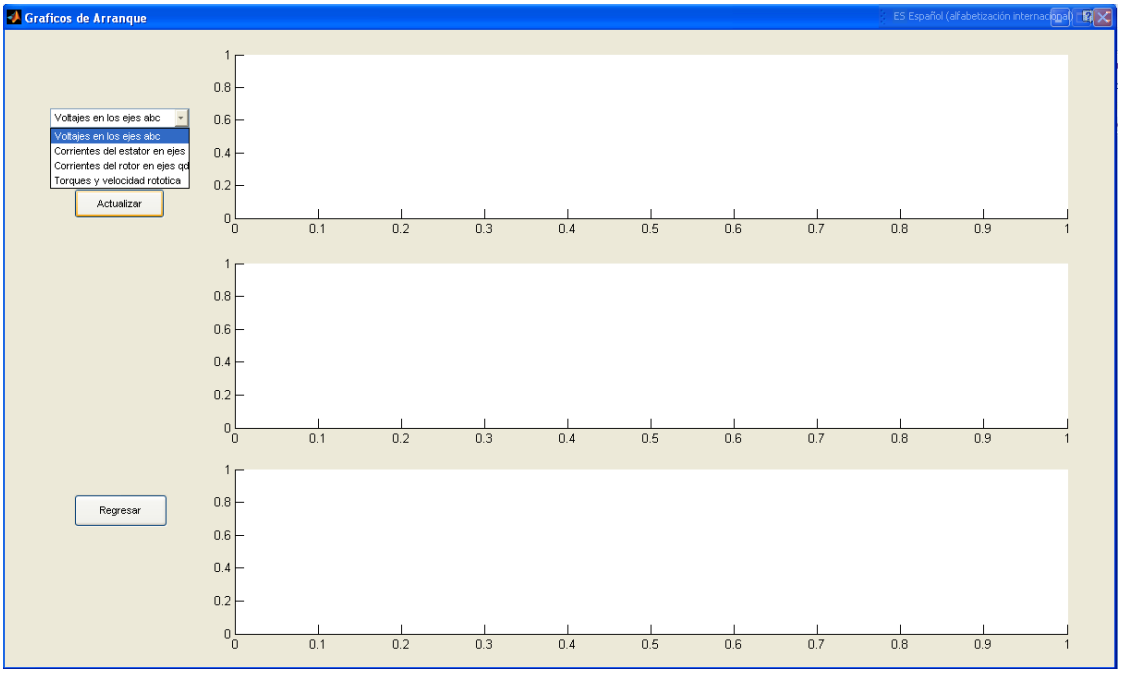
- Cuadráticamente a la velocidad: En la aplicación de motores esta mayormente difundida según sus aplicaciones como en compresores, hélices de buques, bombas centrifugas, ventiladores, etc.

En la ventana de selección Sistema de Referencia se elegirá según se desee puesto que las tres opciones realizan lo mismo, generalmente se elige la referencia estatórica debido a que el cálculo se hace más fácilmente y debido también a que todas los cambios en la simulación se realizan en el estator, las opciones son:

- Referencia Estatórica (W = 0).
- Referencia Sincrónica ($W = W_e$).
- Referencia Rotórica ($W = W_r$).

Por último la ventana de tiempos permite ingresar los tiempos necesarios para la simulación, donde el tiempo final de simulación será el tiempo en donde termina la simulación, mientras que el tiempo para el cambio Y – D permite elegir el tiempo de cambio de conexión del estator de estrella triángulo, tomando siempre en cuenta que el tiempo para el cambio Y – D, jamás será igual o mayor al tiempo final de simulación.

En lo que se refiere a la pantalla analizada las ventanas de conexión del estator, variación de carga, sistema de referencia son generales para simular arranque, frenado y falla por lo que en las demás simulaciones ya no se las tomará en cuenta. Una vez que se ha ingresado y seleccionado todos los valores correspondientes a la pantalla analizada teniendo en cuenta las respectivas condiciones y recomendaciones anteriormente enunciadas lo que resta es realizar la simulación, que se lo realiza dando un clic en el botón Simular, que pasado un determinado tiempo según el caso, los tiempos ingresados para la simulación y la velocidad de la computadora aparecerá la siguiente figura:




La pantalla anterior permite observar las forma de onda generadas durante la simulación, pudiéndose elegir en la ventana de selección:

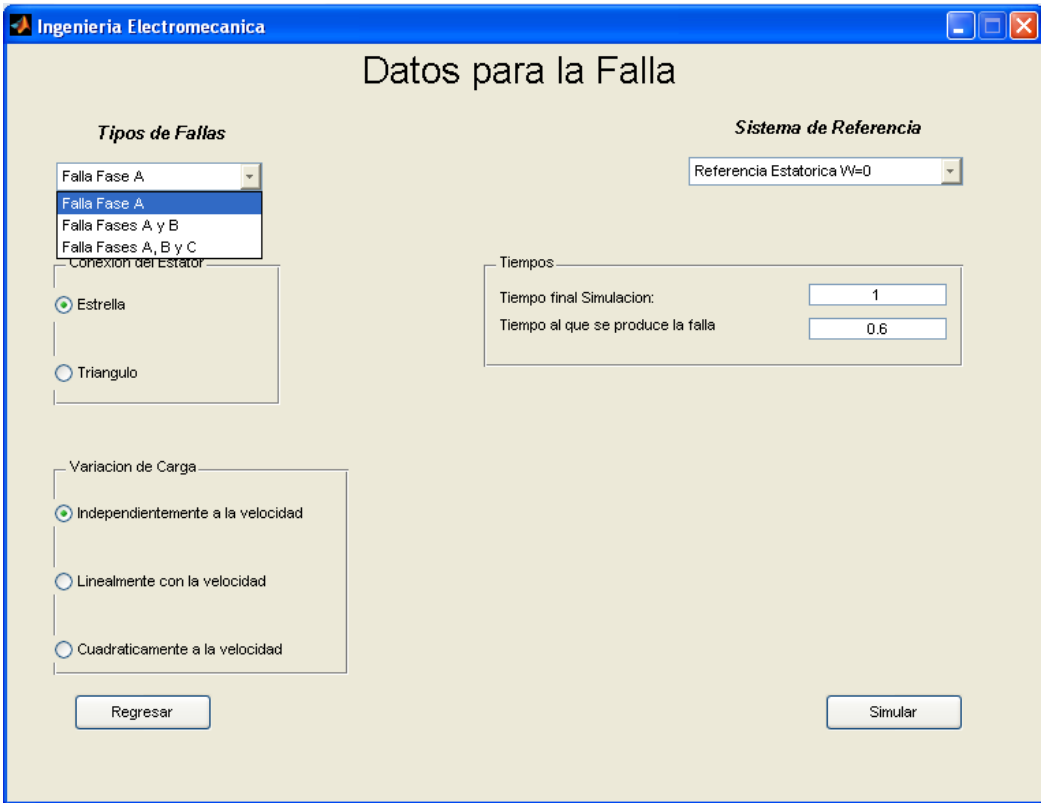
- Voltajes en los ejes abc: Permite observar los voltajes que se aplican al estator según sea el caso simulado.
- Corrientes del estator en ejes abc: Permite observar las corrientes del estator en cada fase a, b y c.
- Corrientes del estator en ejes abc: Permite observar las corrientes del rotor en cada fase a, b y c.
- Torque y velocidad rotórica: Permite observar el Torque electromagnético, la velocidad rotórica y el torque de carga en función de velocidad.

Para poder ver las curvas bastará con elegir una de las opciones y a continuación dar clic en el botón Actualizar.

Así se observarán las formas de onda, esta pantalla y el procedimiento será igual para la simulación del frenado y falla por lo que no se volverá a explicar este paso, también es importante decir que en cualquier momento se puede salir del simulador al hacer clic en la X que se encuentra en la parte superior derecha del software.

**2.- Falla**

En la pantalla de “Parámetros y casos a simular” que se lo vio al principio ahora la elección es de simular fallas y al dar clic en el botón Siguiente la pantalla que aparecerá es la siguiente:



En la pantalla “Datos de falla” se encuentran las opciones enumeradas anteriormente, sin embargo varían dos ventanas: la primera Tipo de Fallas en donde se puede elegir:

- Falla Fase A.
- Falla Fase A, B.
- Falla Fases A, B, C.

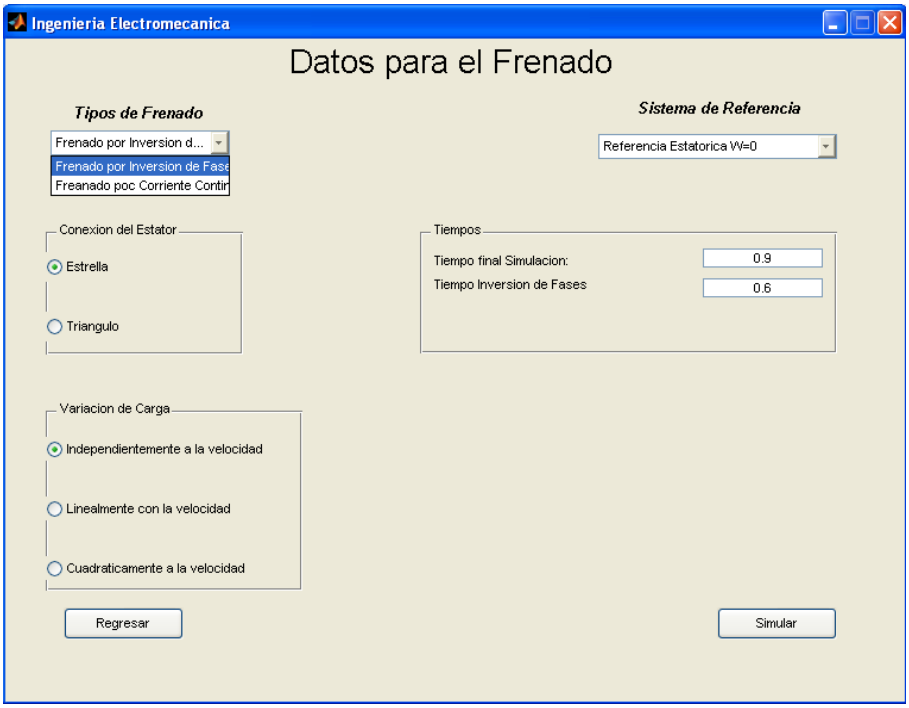
La otra ventana que varia es la correspondiente a los tiempos en donde aparece el Tiempo final de simulación que corresponde al tiempo que finalizará la simulación y el Tiempo al que se produce la falla que es como su nombre lo dice el momento en el que se produce la falla, de igual forma que en el caso anterior el Tiempo al que se produce la falla siempre será menor que el Tiempo final se simulación.

Una vez ingresado y seleccionado todos los datos según las indicaciones dadas en las pantallas anterior lo que queda es simular dando un clic en el botón Simular. Una vez realizada la simulación pasado un determinado tiempo aparecerá la pantalla de gráficos que ya se analizo anteriormente y que no conviene analizarla nuevamente puesto que el procedimiento para observar las curvas es el mismo aparte que también las variables a observar son las mismas en todas las simulaciones.

**3.- Frenado**

Por último se explica la simulación de frenado que como en los dos casos anteriores al elegir en la ventana casos a simular la opción Frenado al dar clic

en el botón Siguiente la pantalla que aparecerá es:

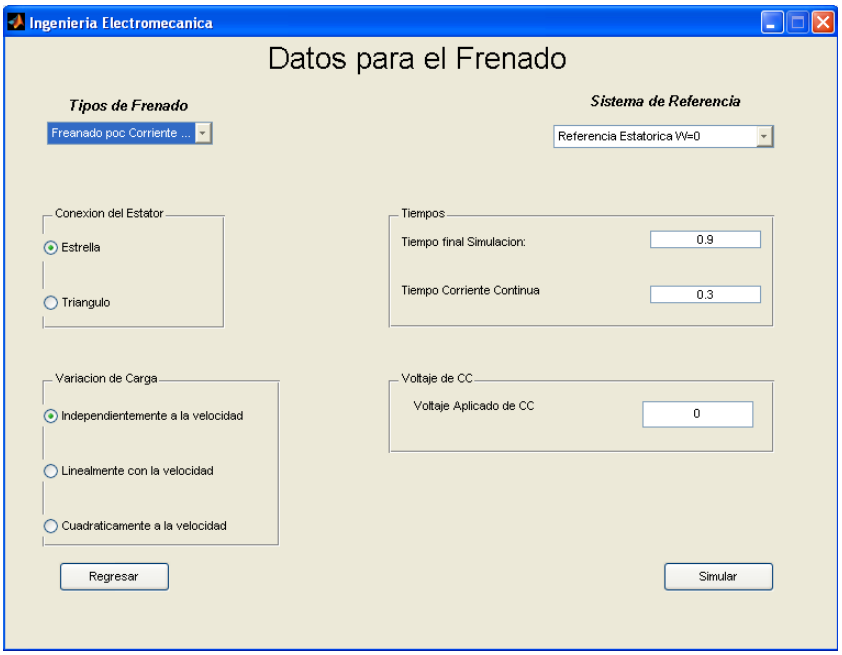



Al igual que en los casos anteriores todo se repito solo cambia la ventana de frenado en donde se puede elegir entre:

- Frenado por inversión de Fases
- Frenado por Corriente Continua.

Al elegir Frenado por Inversión de Fases en la figura anterior, en los tiempos hay dos: el primero el tiempo final de simulación y el otro el tiempo de inversión de fases que es el tiempo en el que se invertirán las fases para lograr el frenado del motor. El tiempo de inversión de fases será igual o mayor al tiempo final de simulación.

En cambio si se elige frenado por corriente continua aparece la siguiente ventana:

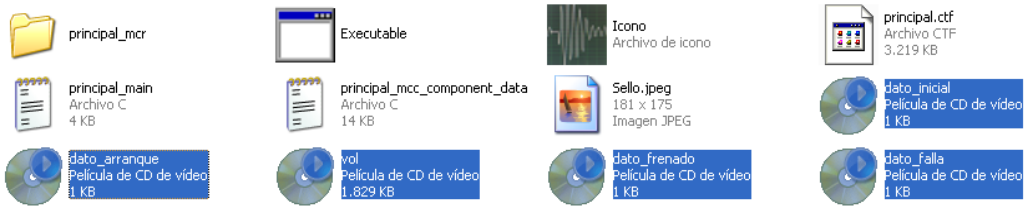



En donde aparte de los tiempos anotados surge una sub ventana que permite elegir el voltaje aplicado de Corriente Continua que por lo general será el 30 % del voltaje aplicado al motor. El tiempo Corriente Continua jamás será mayor o igual al tiempo final de simulación.

Una vez ingresados y elegidos todos los datos se procede a la simulación dando clic en el botón Simular, la pantalla de gráficos es similar a las anotadas anteriormente por lo que se evitará su explicación.

Por último al realizar las simulaciones se generan archivos que permiten guardar o realizar un historial de las simulaciones y se guardan en la carpeta raíz que es

la llamada “Modelación y Simulación de las Máquinas de Inducción” así:




Los archivos con fondo azul corresponden a los datos ingresados de parámetros, arranque, frenado y falla así como también el archivo *vol* contiene todos los datos que se realizan de las simulaciones. Se recomienda que antes de realizar una nueva simulación se borren estos archivos para tener la pantalla de graficación limpia a menos que se desee guardar un historial de las simulaciones realizadas.

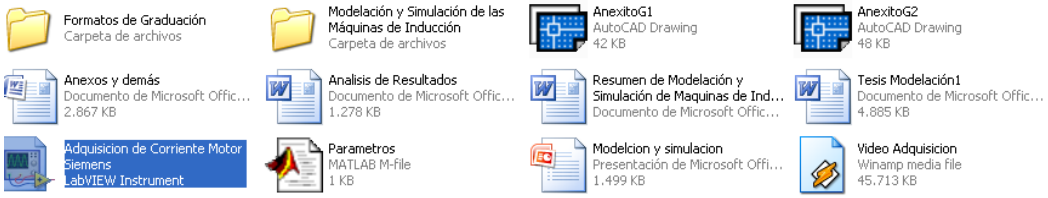
El Manual del Usuario anterior corresponde al software de simulación, a continuación se hará la explicación de la utilización del software de adquisición de datos como también del modulo llamado “Transitorios Motor”.

Para esto es necesario que en la computadora que se realizará la obtención de los datos del motor se instale el software Lab View versión 8.2 en adelante puesto que en esta versión se realizó la programación, así como también el software NI-DAQmx que es el que controla las funciones de la tarjeta USB 6008.

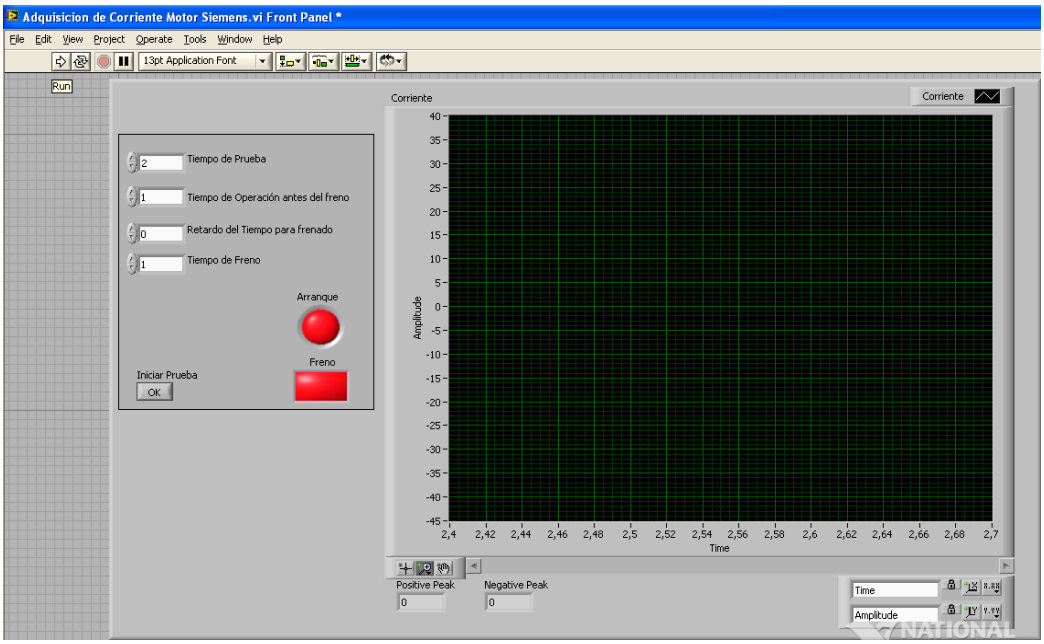
Una vez instalado estos programas lo que primero se hará es colocar el modulo “Transitorios Motor” que se ve en la siguiente figura en alguna de las mesas para las respectivas conexiones.


De la figura anterior los terminales L1 y L2 corresponden a la alimentación con la que funcionará el modulo y es de 220 V, las dos siguientes corresponden a las entradas para la activación de los contactores en donde por 13 entrará la alimentación L1 y de 14 irá a la entrada de la bobina del contactor para el arranque, cosa similar ocurre con las dos siguientes entradas con la diferencia que ahora la salida 14 se conectará a la bobina del contactor que realizará la inversión del motor. Por último los terminales k1 y l1 que provienen del transformador de corriente 010.421 permite la adquisición de datos del motor. Y por último se tiene la USB DAQ que corresponde al enlace entre la PC y el modulo a través del cable USB proporcionado con la tarjeta, esta última conexión se lo hará al final cuando ya se haya hecho todas las conexiones que se explican en el anexo F.

Una vez realizada todas las conexiones y también conectado el cable USB a la PC y al modulo se hará doble clic en el archivo de Lab View “Adquisición de Corriente Motor Siemens” que se encuentra en la carpeta del CD proporcionado.




Al abrir este archivo aparecerá la siguiente pantalla:




Lo primero que hay que hacer es correr el programa a través del icono Run que se señala en la figura anterior. Los datos que se ingresarán únicamente

corresponden a los tiempos, en donde el primer valor que es el Tiempo de Prueba para el motor Siemens analizado se recomienda que sea de un tiempo de 2 seg tomando en cuenta que este valor siempre será mayor o igual a la suma de los tres tiempos restantes. El segundo tiempo que es el Tiempo de Operación antes del Freno corresponde al tiempo que estará funcionando el motor antes de producirse el freno y se recomienda un valor de uno. El tercer tiempo corresponde al valor de Retardo de tiempo para el Frenado, este tiempo corresponde a dar un retardo antes de producirse el frenado que debe ser de corto tiempo en este caso se recomienda cero ya que las partes mecánicas de los contactores ya producen el retardo y no es necesario dar algún valor a este casillero sin embargo se lo puede poner si es el caso, y por último el valor Tiempo de Freno es el valor de tiempo en el que el motor estará bajo la acción del freno antes de terminarse la adquisición y se recomienda que sea de valor de 1 seg. Ingresados los valores conforme a las recomendaciones dadas bastará con hacer clic en el botón OK para iniciar la prueba, manipular el motor y obtener las formas de onda deseadas. Los indicadores arranque y freno únicamente permiten conocer al ponerse de color verde cuando esta en arranque y cuando en freno.