



INTEGRACIÓN NUMÉRICA CON REDES NEURONALES

TESIS PROFESIONAL

Que para obtener el título de:

Ing. en Ciencias de la Computación

Presenta:

Iván Christofer Chaman García

Asesor:

Dra. María de Lourdes Sandoval Solís

H. Puebla de Zaragoza.

Prefacio

En el presente trabajo de tesis se produjo un material para químicos, físicos e investigadores para los cuales sea de utilidad el cálculo de Factores Franck-Condon (FFC) de Morse para moléculas diatómicas con números cuánticos pequeños y grandes.

Este material no incluye el desarrollo completo de los Factores Franck-Condon de Morse, solo se hace una breve mención al respecto, ya que el enfoque adoptado no es hacer un planteamiento especializado en estas funciones de onda, sino implementar el uso de las Redes Neuronales Artificiales Supervisadas (RNAS) para el cálculo de la integral de la función de onda para los Factores Franck-Condon; aspecto que constituye estudio tema central.

Primero se estudia la metodología presentada con funciones oscilantes con y sin singularidades, posteriormente se aplicará la técnica con sus diferentes variantes al cálculo de Factores Franck-Condon para moléculas diatómicas.

Se presentan seis algoritmos de entrenamiento para el ajuste de los pesos sinápticos de las RNAS para el cálculo de la integral de las funciones de onda de los FFC. A estos métodos se le han asignado un nombre dependiendo de la técnica que se utiliza para el entrenamiento de las RNAS. Los algoritmos son los siguientes: RNA_BP (Back-Propagation, usa la técnica clásica de RNAS), RNA_FM (Factor Momentum, usa la técnica clásica de RNAS), RNA_GC (usa la técnica de Gradientes Conjugados), RNA_GCR (usa la técnica de Gradientes Conjugados Residual), RNA_N (usa la técnica de Newton) y RNA_NT (usa la técnica Newton Truncado).

En el Capítulo 1 presenta una breve descripción de los FFC para la evaluación de la función de onda, el uso de las RNAS junto a los métodos numéricos para el cálculo numérico de integrales definidas y el objetivo general y específicos de esta tesis.

En el Capítulo 2 y 3, respectivamente, empiezan con el estudio del estado del arte de las Redes Neuronales y las técnicas clásicas para aproximar numéricamente la integral definida.

En el Capítulo 4 se aplica este nuevo enfoque para aproximar la integral definida, lo cual se hace en dos fases: En la primera fase se utilizan las RNAS (RNA_BP, RNA_FM, RNA_GC, RNA_GCR, RNA_N y RNA_NT) para ajustar los pesos sinápticos de una combinación lineal de funciones coseno, la cual es una aproximación a una función real deseada, en esta fase se detalla las técnicas empleadas en cada algoritmo. En la segunda fase se obtiene la integral definida de la combinación línea.

En el Capítulo 5 se muestran las pruebas y resultados obtenidos para los seis algoritmos de entrenamiento de las RNAS empleados para el ajuste de funciones oscilantes y el cálculo de la integral de la función de onda de FFC para números cuánticos pequeños y grandes.

Contenido

1. INTRODUCCIÓN	1
1.1. LOS FACTORES FRANCK-CONDON.....	1
1.2. LAS REDES NEURONALES Y LOS MÉTODOS NUMÉRICOS.....	2
1.3. OBJETIVOS	2
1.3.1. Objetivo General	2
1.3.2. Objetivos Específicos	2
2. REDES NEURONALES ARTIFICIALES	3
2.1. ANTECEDENTES DE LAS REDES NEURONALES.....	3
2.2. USOS DE LAS REDES NEURONALES	6
2.3. PERCEPTRÓN SIMPLE.....	13
2.4. ADALINE	16
2.5. CARACTERÍSTICAS DE OPERACIÓN.....	17
3. INTEGRACIÓN NUMÉRICA	20
3.1. MÉTODOS DE INTEGRACIÓN NUMÉRICA TRADICIONAL	20
3.2. FÓRMULAS DE CUADRATURA.....	21
3.2.1. Regla del Rectángulo	22
3.2.2. Regla del Punto Medio	22
3.2.3. Regla del Trapecio	22
3.2.4. Regla de Simpson	22
3.2.5. Reglas Compuestas.....	22
3.3. INTEGRACIÓN GAUSSIANA.....	23
3.4. ALGORITMOS ADAPTIVOS	24
4. INTEGRACIÓN NUMÉRICA CON REDES NEURONALES	26
4.1. DEFINICIÓN DE LA RED NEURONAL	26
4.2. AJUSTE DE LA FUNCIÓN.....	27
4.2.1. RNA Back-Propagation.....	28
4.2.2. RNA Factor Momentum.....	32
4.2.3. RNA Gradientes Conjugados.....	34
4.2.4. RNA Gradientes Conjugados Residuales	37
4.2.5. RNA Newton.....	40
4.2.6. RNA Newton Truncado.....	41
4.3. INTEGRACIÓN DEL AJUSTE DE LA FUNCIÓN.....	43
5. PRUEBAS Y RESULTADOS.....	44
5.1. FUNCIONES CONTINUAS.....	45
5.2. FUNCIONES CON SINGULARIDADES.....	48
5.3. FACTORES FRANCK-CONDON	51
5.3.1. Moléculas Diatómicas	51
5.3.2. Moléculas Diatómicas con Números Cuánticos Grandes	61
6. CONCLUSIONES	71
APÉNDICE A: TABLAS	73
APÉNDICE B: FIGURAS	75
APÉNDICE C: PROGRAMAS EN MATLAB	76
APÉNDICE D: INTERFAZ Y MANUAL DE USUARIO EN JAVA	110
REFERENCIAS	127

1. Introducción

En este capítulo introduciremos los conceptos básicos que nos darán todo el soporte teórico básico donde que nuestro trabajo.

1.1. Los Factores Franck-Condon

Una aplicación importante en la astrofísica es detectar las moléculas interestelares, para esto requerimos calcular los “Factores Franck-Condon” (FFC) para moléculas diatómicas, los cuales están definidos de la siguiente forma:

El principio de Franck-Condon nos permite hacer predicciones bastantes exactas de donde son altamente probables las transiciones vibracionales entre un par de estados electrónico. La enunciación de este principio es el siguiente:

El salto de un electrón en una molécula se realiza tan rápidamente en comparación con el movimiento vibracional que inmediatamente después, el núcleo aún tiene la misma posición relativa y velocidad que tenía antes del salto.

Esta idea fue propuesta inicialmente en el año de 1925 por Franck; y un poco más tarde, confirmada por Condon en la mecánica cuántica.

En resumen, la probabilidad de transición entre dos estados, caracterizados por las eigenfunciones $\Psi_{v'}$ y $\Psi_{v''}$, es proporcional al cuadrado de los correspondientes elementos de la matriz del momento electrónico (o momento de transición):

$$R = \left| \int \Psi_{v'}^* M \Psi_{v''} dr \right|^2.$$

A esta integral se le conoce como integral de traslape vibracional; entonces, al cuadrado de la integral sobre el producto de las eigenfunciones vibracionales de los dos estados involucrados (base y excitación), viene dado por:

$$\left| \int \Psi_{v'}^* \Psi_{v''} dr \right|^2.$$

Los cuadrados de las integrales de traslape vibracional son conocidos como los Factores Franck-Condon

Aplicando el potencial de Morse para las eigenfunciones vibracionales; tanto para el estado base como para el estado excitado, que intervienen en los Factores Franck-Condon se resume como:

$$FFC = \left| \int \Psi_{v'}^* \Psi_{v''} dr \right|^2,$$

donde $\Psi_v(r) = N_v e^{-\frac{z}{2}} z^{\frac{b}{2}} L_{k,v}(z)$, $k = \frac{4\pi(2\mu D)^{\frac{1}{2}}}{ah} = \frac{w_e}{w_e x_e}$, $z = k e^{-a(r-r_e)}$, $b = k - 2v - 1$,

N_v es la constante de normalización para el estado vibracional v , y

$L_{k,v} = L_v^k$ son los polinomios generalizados de Laguerre.

Estas integrales para el potencial de Morse no se pueden obtener analíticamente, por lo que se aproximan numéricamente. En 1992 en la Escuela de Ciencias Físico-Matemáticas de la BUAP, se presentó una tesis desarrollada para aproximar los FFC numéricamente usando el Método de Simpson Adaptivo (Abad, 1991), con este software se han aproximado los FFC y ha permitido identificar las componentes de los cometas observados (Churyumov, y otros, 2002) (Churyumov, y otros, 2003) (Churyumov, y otros, 2005) (Churyumov, y otros, 2005) (Churyumov, y otros, 2005) (Churyumov, y otros, 2007).

Conforme se ha aumentado la aplicación del software y mejorado la tecnología en espectrometría se ha requerido aproximar los FFC para números cuánticos vibracionales grandes, esto nos ha llevado a la necesidad de buscar nuevas formas de aproximar las integrales para funciones muy oscilantes.

1.2. Las Redes Neuronales y los Métodos Numéricos

En el 2005 se publicó un artículo (Zeng, y otros, Jul/Aug 2006) donde se proponen cuatro algoritmos basados en Redes Neuronales Artificiales para la solución de integrales definidas, con una mejor aproximación que la regla de Simpson Adaptivo.

Por lo que en este trabajo se aplica para aproximar la integral definida, lo cual se hace en dos fases:

En la primera fase se utilizan redes neuronales con aprendizaje supervisado para ajustar los pesos w_i de la combinación lineal siguiente:

$$f(x) = \sum_{i=0}^N w_i \cos(ix).$$

En la segunda fase se obtiene la integral definida de la siguiente forma:

$$\int_a^b f(x) dx = \int_a^b \sum_{i=0}^N w_i \cos(ix) dx.$$

Además en este trabajo se presenta una aplicación de la metodología desarrollada, la aproximación de los Factores Franck-Condon para moléculas diatómicas.

1.3. Objetivos

1.3.1. Objetivo General

Generar una aplicación que permita aproximar integrales definidas con mayor precisión y exactitud utilizando redes neuronales.

1.3.2. Objetivos Específicos

- Implementar los algoritmos para aproximar integrales definidas usando las seis diferentes algoritmos para ajustar los pesos sinápticos de la Red Neuronal.
- Probar los algoritmos a las integrales de las funciones de Onda del Potencial de Morse (Factores Franck-Condon).

2. Redes Neuronales Artificiales

Actualmente existe una gran tendencia a establecer nuevas maneras de resolver problemas que no pueden ser descritos de una manera fácil por algoritmos tradicionales, estas nuevas maneras de resolverlo tiene su inspiración en la emulación de sistemas biológicos, como son el uso de Redes Neuronales Artificiales (RNA). En este capítulo se describe el inicio de las Redes Neuronales, sus características principales y las diferentes clasificaciones de las mismas.

2.1. Antecedentes de las Redes Neuronales

Las Redes Neuronales Artificiales (RNA) o sistemas conexionistas son sistemas de procesamiento de la información cuya estructura y funcionamiento están inspirados en las redes neuronales biológicas. Consisten en un conjunto de (Montaño Moreno, 2002) elementos simples de procesamiento llamados nodos o neuronas conectadas entre sí por conexiones que tienen un valor numérico modificable llamado peso.

La arquitectura de procesamiento de la información de los sistemas de RNA se distingue de la arquitectura convencional Von Neumann (fundamento de la mayor parte de los ordenadores existentes) en una serie de aspectos fundamentales.

En primer lugar, el procesamiento en un sistema conexionista no es secuencial sino paralelo, esto es, muchas unidades de procesamiento pueden estar funcionando simultáneamente.

En segundo lugar, la información que posee un sistema no está localizada o almacenada en compartimentos discretos, sino que está distribuida a lo largo de los parámetros del sistema. Los parámetros que definen el “conocimiento” que una red neuronal posee en un momento dado son sus conexiones y el estado de activación de sus unidades de procesamiento.

El aprendizaje en una RNA es un proceso de ajuste o modificación de los valores o pesos de las conexiones, “hasta que la conducta del sistema acaba por reproducir las propiedades estadísticas de sus entradas” (Montaño Moreno, 2002).

Las RNA constituyen una línea de investigación en Inteligencia Artificial (IA), la cual tiene como objetivo primario la construcción de máquinas inteligentes. Los orígenes de la IA hay que buscarlos en el movimiento científico de la cibernética de los años cuarenta y cincuenta. Este movimiento científico se articuló en torno a la idea de que el funcionamiento de muchos sistemas, vivos o artificiales, puede ser captado mejor por modelos basados en la transferencia de información que por modelos basados en la transferencia de energía. La cibernética se propuso estudiar los elementos comunes entre el funcionamiento de máquinas automáticas y el del sistema nervioso humano (los procesos de control y comunicación en el animal y en la máquina). Este problema fue abordado en un esfuerzo interdisciplinar, en el que intervinieron investigadores procedentes de áreas como matemáticas, ingeniería electrónica, fisiología y neurociencia, lógica formal, ciencias de la computación, psicología y otras más.

Una importante característica de la cibernética fue la proliferación de distintas perspectivas en torno al problema de las relaciones entre cerebro y máquina. En la segunda mitad de la década de los cincuenta comenzaron a destacar dos de entre estas perspectivas: la IA basada en el procesamiento simbólico, y la investigación en redes neuronales.

Los sistemas de IA simbólica simulan procesos mentales y cognitivos humanos por medio de programas ejecutados por un ordenador del tipo Von Neumann. Entre los investigadores más importantes

de esta primera época de investigación en este paradigma se puede destacar a John McCarthy, Allen Newell, Herbert Simon y Marvin Minsky. Paralelamente, en la segunda mitad de los años 50, algunos investigadores comenzaron a desarrollar una perspectiva diferente en la construcción de máquinas inteligentes: la perspectiva de las RNA o sistemas conexionistas. Esta perspectiva no perseguía la modelación de redes neuronales fisiológicas, sino la construcción de máquinas inteligentes empleando arquitecturas computacionales de cierta semejanza con las redes neuronales del cerebro. Como antecedentes más directos a este grupo de investigadores, cabe destacar las aportaciones, por un lado, de Warren McCulloch y Walter Pitts y, por otro lado, de Donald Hebb (Montaño Moreno, 2002).

McCulloch y Pitts (1943) presentaron la estructura y funcionamiento de la unidad elemental de procesamiento de una red conexionista llamado Perceptron Simple. La neurona de McCulloch-Pitts (ver figura 2-1), como actualmente se conoce, tiene un funcionamiento muy sencillo: si la suma de entradas excitatorias supera el umbral de activación de la unidad, y además no hay una entrada inhibitoria, la neurona se activa y emite respuesta (representada por el valor 1); en caso contrario, la neurona no se activa (valor 0 que indica la ausencia de respuesta).

Combinando varias neuronas de este tipo con los adecuados umbrales de respuesta, se puede construir una red que calcule cualquier función lógica finita: $\{0,1\}^n \rightarrow \{0,1\}$, usando únicamente dos neuronas.

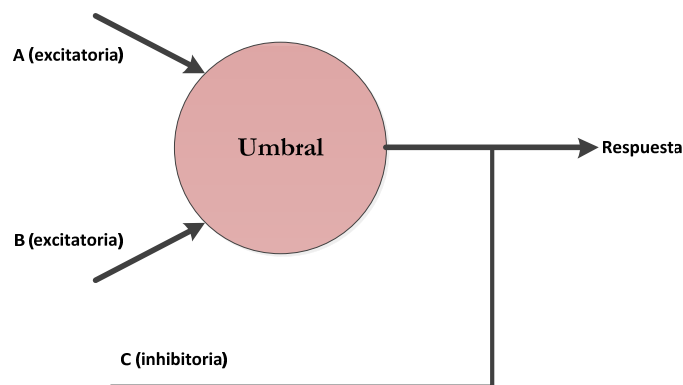


Figura 2-1 Neurona de McCulloch-Pitts.

Hebb (1949) postuló un sencillo pero potente mecanismo de regulación de las conexiones neuronales, que constituyó la base de las reglas de aprendizaje que más tarde se desarrollarían. La regla de Hebb, en su versión más elemental, se expresa como sigue: “*Cuando un axón de una célula A está bastante cerca para excitar a una célula B y repetida o persistentemente dispara, entonces se produce algún proceso de desarrollo o cambio metabólico de tal forma que la eficiencia del disparo de A hacia B aumenta*” (Montaño Moreno, 2002). La propuesta de Hebb es de especial relevancia porque indica que la información necesaria para modificar el valor de una conexión se encuentra localmente disponible a ambos lados de la conexión. En la actualidad existe un gran número de redes neuronales cuyo aprendizaje está basado en la regla de Hebb como las conocidas redes de Hopfield (1982) y algunos modelos de red propuestos por Kohonen (1977).

La evolución de la investigación en redes neuronales desde los años 50 ha estado condicionada a las limitaciones observadas en la red Perceptrón simple y la emergencia del conexionismo en la segunda mitad de los 80 como paradigma aceptado en IA, gracias, entre otros avances, La aparición de un algoritmo, denominado *backpropagation error* (propagación del error hacia atrás) o simplemente *backpropagation*, que permite modificar las conexiones de arquitecturas multiestrato.

En el período de los 50 y 60's se produjeron importantes contribuciones científicas. Una de las más importantes fue el trabajo de los grupos de Rosenblatt y Widrow con sistemas conexionistas de único

estrato o capa (RNA que solo tienen un estrato de conexiones modificables). La red diseñada por Rosenblatt (1958), denominada Perceptrón, es un sistema de este tipo. A pesar de tener dos estratos de conexiones, sólo uno de ellos está compuesto de conexiones modificables. La capa de entrada o retina consiste en un conjunto de unidades de entrada binarias conectadas por conexiones con valor fijo con las unidades de la capa de asociación o de predicados. La última capa es la de respuesta o decisión, cuya única unidad, con salida binaria, es la que tiene conexiones modificables con los predicados de la capa anterior.

El teorema de convergencia de la regla de aprendizaje del Perceptrón desarrollado por Rosenblatt establecía que, si los parámetros o pesos del sistema eran capaces de realizar una determinada clasificación, el sistema acabaría aprendiéndola en un número finito de pasos, si se modificaban las conexiones de acuerdo con dicha regla de aprendizaje (Fausett, 1994). Más concretamente, la regla de aprendizaje del Perceptrón es un algoritmo de los denominados supervisado por corrección de errores y consiste en ir ajustando de forma iterativa los pesos en proporción a la diferencia existente entre la salida actual de la red y la salida deseada, con el objetivo de minimizar el error actual de la red.

El Perceptrón, una máquina conexionista diseñada y estudiada teóricamente por Rosenblatt, construida por un grupo de ingenieros del Laboratorio de Aeronáutica de Cornell (CAL, Ithaca, Nueva York) y financiada por la Oficina de Investigación Naval del Ejército de los Estados Unidos (ONR, *Office of Naval Research*), fue una de las contribuciones científicas y tecnológicas más importantes de la primera fase del conexionismo.

Otra importante contribución científica es la aportada por Widrow y Hoff en 1960. Estos autores propusieron un nuevo tipo de unidad de procesamiento, con estructura similar a la del Perceptrón pero con un mecanismo de aprendizaje diferente que permitía también la entrada de información de tipo continuo: la neurona ADALINE (ADaptative LINear Elements). La innovación de esta tipología de neurona se halla en su mecanismo de aprendizaje denominado regla delta o regla de Widrow- Hoff, que introduce el concepto de reducción del gradiente del error. La deducción de la regla delta se puede expresar de la siguiente forma: teniendo en cuenta que E^p (el error que comete la red para un determinado patrón p), es función de todos los pesos de la red, el gradiente de E^p es un vector igual a la derivada parcial de E^p respecto a cada uno de los pesos. El gradiente toma la dirección del incremento más rápido en E^p ; la dirección opuesta toma el decremento más rápido en el error. Por tanto, el error puede reducirse iterativamente ajustando cada w_i en la dirección $-\frac{\partial E^p}{\partial w_i}$, la regla delta basada en la reducción del gradiente del error es la precursora del algoritmo *backpropagation* aplicado a redes múltiples.

La contribución más importante en el resurgimiento del conexionismo en los años ochenta fue la técnica *backpropagation* desarrollada por Rumelhart, Hinton y Williams, representantes del grupo PDP (*Parallel Distributed Processing*) (Universidad de San Diego, California). Realmente, esta técnica fue desarrollada inicialmente por Paul Werbos (1974) a mediados de los 70, y después independientemente redescubierta por varios grupos de investigadores (Le Cun, 1985; Parker, 1985; Rumelhart, Hinton y Williams, 1986). Es, por tanto, un caso de “descubrimiento múltiple”. Sin embargo, en general se reconoce que fue la versión del grupo PDP la que desató el interés en RNA a mediados de los ochenta y consiguió finalmente forzar la revisión del consenso contrario al conexionismo.

El algoritmo *backpropagation* también recibe el nombre de regla delta generalizada o método de gradiente decreciente, debido a que supone una extensión de la regla propuesta por Widrow y Hoff en 1960 (regla delta) a redes con capas intermedias (ver figura 2-2). Este tipo de arquitectura recibe el nombre genérico de Perceptrón Multicapa o MLP (*Multilayer Perceptron*). Rosenblatt ya tuvo la idea de utilizar una técnica de este tipo a principios de los sesenta (Rosenblatt, 1962), aunque no pudo desarrollarla de un modo satisfactorio.

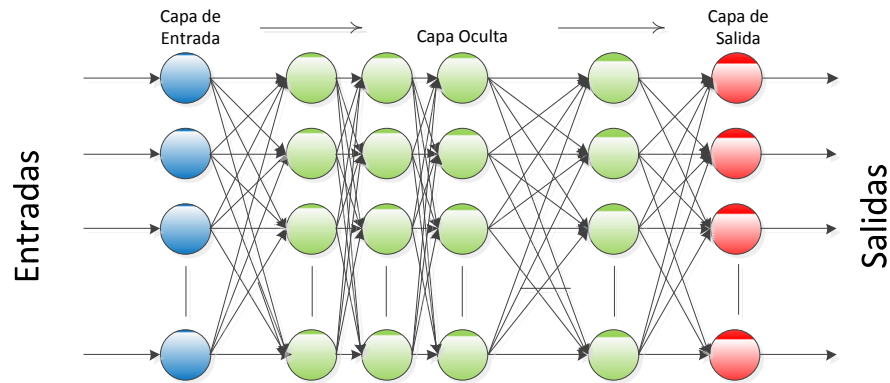


Figura 2-2 Arquitectura de un Perceptrón multicapa.

El objetivo del algoritmo *backpropagation* es propagar los errores cometidos por las unidades de salida hacia atrás, ya que, en un sistema de este tipo, el error cometido por una unidad intermedia depende del error cometido por las unidades de salida a las que dicha unidad intermedia está conectada. Tras conocerse el error cometido por las unidades intermedias, pueden entonces modificarse las conexiones entre unidades de entrada y unidades intermedias. De forma similar a la regla delta, la base matemática del algoritmo *backpropagation* es la técnica de gradiente decreciente, basada en modificar los pesos en la dirección opuesta al gradiente, esto es $-\frac{\partial EP}{\partial w_i}$ determina el decremento más rápido del error. El algoritmo *backpropagation* exige la utilización de funciones de activación continuas para poder realizar el cálculo de la derivada parcial del error con respecto a los pesos del modelo.

2.2. Usos de las Redes Neuronales

El creciente interés despertado por las RNA, que se manifiesta de forma palpable a principios de los 90 a la fecha, está relacionado con un acontecimiento fundamental en la historia de las RNA comentado al inicio, a saber: la publicación de *Parallel Distributed Processing* (Procesamiento Distribuido en Paralelo o PDP) (Montaño Moreno, 2002), obra que se ha llegado a conocer como la “biblia” del nuevo paradigma conexionista donde se describe, entre otras cosas, el algoritmo *backpropagation* aplicado a redes MLP.

El cerebro es uno de las cumbres de la evolución biológica (ver Figura 2-3), ya que es un gran procesador de información. Entre sus características podemos destacar, que es capaz de procesar a gran velocidad grandes cantidades de información procedentes de los sentidos, combinarla o compararla con la información almacenada y dar respuestas adecuadas. Además es destacar su capacidad para aprender a representar la información necesaria para desarrollar tales habilidades, sin instrucciones explícitas para ello.

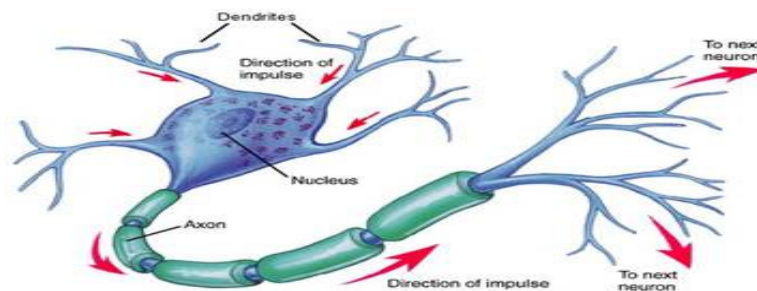


Figura 2-3 Neurona Biológica

Las neuronas artificiales son modelos que tratan de simular el comportamiento de las neuronas biológicas, cada neurona se representa como una unidad de proceso que forma parte de una entidad mayor, esto se conoce como red neuronal.

El proceso consta de una serie de entradas x_i , que equivalen a las dendritas de donde reciben la estimulación, ponderadas por unos pesos w_i que representan los impulsos entrantes, son evaluados y se combinan con la función de red que nos dará el nivel de potencial de la neurona; la salida de la función de red es evaluada en la función de activación que da lugar a la salida de la unidad de proceso (ver Figura 2-4). Una neurona artificial se comporta como la neurona biológica pero de una forma muy simplificada.

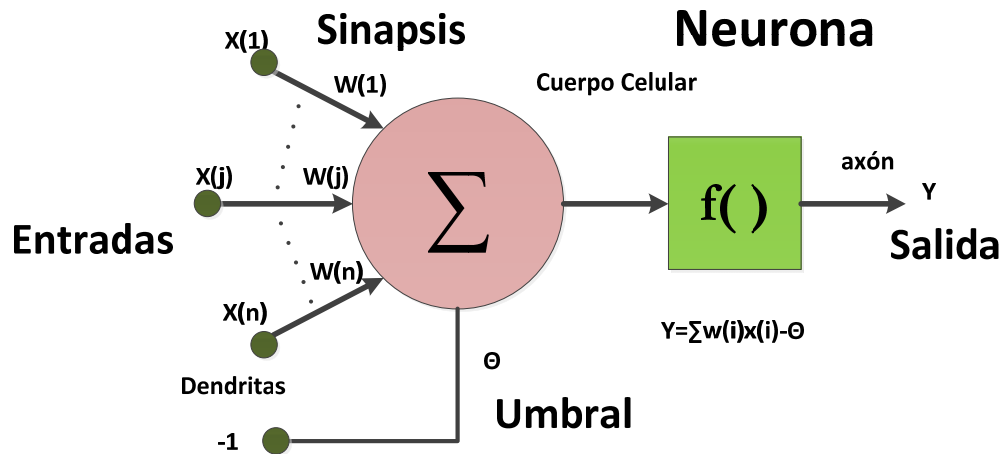


Figura 2-4 Neurona Artificial

En los usos que se le da a las Redes Neuronales Artificiales es muy variado para resolver problemas en diversas disciplinas, por ejemplo, en finanzas: se utilizan para valorar el riesgo de los créditos, identificar falsificaciones, interpretar firmas, entre otras aplicaciones (Cheng, Chen, & Wei, 2010), en Medicina: para la extracción de características de las bandas de frecuencias pertinentes a señal en un Electroencefalograma (Khare, Santhosh, Anand, & Bhatia, 2010), usos militares: para clasificar las señales de radar, creación de armas inteligentes y optimización del uso de recursos (Caliskan, Aykan, & Hajiyeve, 2008), implementaciones físicas en FPGA que existen de las redes neuronales como ejemplo la clasificación de patrones mediante la aplicación de hardware (Polat & Yildirim, 2010).

Existen registros sobre bases de datos (Montaño Moreno, 2002) cuya descripción y resultados generales obtenidos, que para discriminar el tipo de aplicación de las RNA que se realiza mayoritariamente, en el 2002 existía un total los 2.057 registros que se clasificaron en una de 43 áreas o disciplinas de aplicación, siendo las más frecuentes y por este orden: medicina (637 registros), ingeniería (597 registros), biología (362 registros) y psicología (132 registros). Las áreas de aplicación abarcan prácticamente cualquier disciplina de conocimiento (alimentación, aviación, agricultura, arqueología, documentación, hidrología, medio ambiente, música, tráfico, veterinaria, etc.). Entre otras aplicaciones (Unidad de Investigación en Automatización Industrial, Departamento de Sistemas y Automatica, Escuela de Ingenieria Electrica, Universidad de Carabobo, Valencia Venezuela, 2004)

Actualmente existen más de **57604** registros de trabajos sobre redes neuronales en las bases de datos Academic Search Complete, Computers & Applied Sciences Complete, Environment Complete y Web of Science, en las diferentes áreas del conocimiento.

Según Sarle (2002), un Perceptrón simple puede ser considerado como un Modelo Lineal Generalizado (MLG) (Montaño Moreno, 2002), debido a la equivalencia entre el concepto de función de enlace en un MLG y la función de activación de la neurona de salida en un Perceptrón:

$$Y \equiv f(X, W),$$

donde el valor de la variable de respuesta Y (o variable de salida) se obtiene aplicando una función de enlace (o función de activación) sobre una combinación lineal de coeficientes W (o pesos) y variables explicativas X (o variables de entrada).

Una importante diferencia entre RNA y modelos estadísticos consiste en que los parámetros obtenidos por la red neuronal no son susceptibles de una interpretación práctica.

La función de enlace en un MLG no suele estar acotada y, en la mayoría de casos, es necesario que sea monótona como las funciones identidad, recíproca y exponencial. Por su parte, la función de activación en un Perceptrón puede estar acotada, como la función sigmoideal logística, o puede no estarlo, como la función identidad; sin embargo, en general todas ellas son monótonas.

El concepto de discrepancia en un MLG y el concepto de función de error en un Perceptrón también son equivalentes (Montaño Moreno, 2002). En el caso del Perceptrón la función que en general se intenta minimizar es la suma del error cuadrático:

$$E = \sum_{p=1}^P \frac{1}{2} \sum_{k=1}^M (d_{pk} - y_{pk})^2,$$

donde P es el número de patrones, M es el número de neuronas de salida, d_{pk} es la salida deseada para la neurona de salida k para el patrón p e y_{pk} es la salida obtenida por la red para la neurona de salida k para el patrón p .

El Perceptrón normalmente estima los parámetros del modelo mediante el criterio de mínimos cuadrados, es decir, intentando minimizar la función E , el MLG ajusta el modelo mediante el método de máxima verosimilitud para una variedad de distribuciones de la clase exponencial. Sin embargo, Bishop (1995), entre otros, ha apuntado que el criterio de mínimos cuadrados asumiendo un error con distribución normal obtiene estimaciones máximo-verosímiles, tal como ocurre en el MLG. De forma similar, se puede aplicar el método de máxima verosimilitud a un Perceptrón en tareas de clasificación binaria asumiendo un error con distribución de Bernoulli. En este caso, la función de error que se intenta minimizar se denomina *cross entropy* que viene dada por (Montaño Moreno, 2002):

$$E = \sum_{p=1}^P \frac{1}{2} \sum_{k=1}^M [d_{pk} \log(y_{pk}) + (1 - d_{pk})(1 - y_{pk})].$$

Utilizando esta función de error conseguimos que las salidas puedan ser interpretadas como probabilidades *a posteriori*. Sin embargo, en general la obtención de los parámetros de una red se realiza mediante un criterio de optimización sin tener en cuenta el tipo de distribución de los errores, a diferencia de los MLG.

Estableciendo analogías entre RNA y modelos concretos pertenecientes a MLG, tenemos que un Perceptrón simple con función de activación lineal en la neurona de salida y utilizando la suma del error cuadrático equivale a un modelo de regresión lineal (ver figura 2-5).

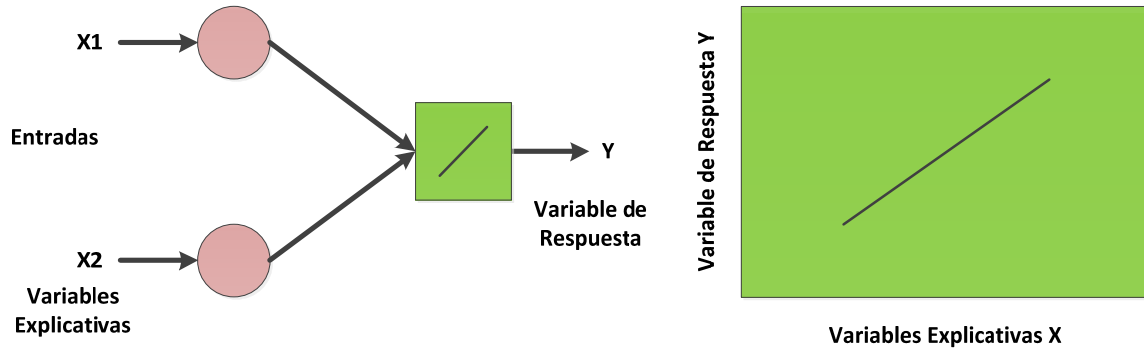


Figura 2-5 Perceptrón simple con función lineal = Modelo de regresión lineal

Por su parte, un Perceptrón simple con función de activación logística en la neurona de salida es similar a un modelo de regresión logística (ver figura 2-6).

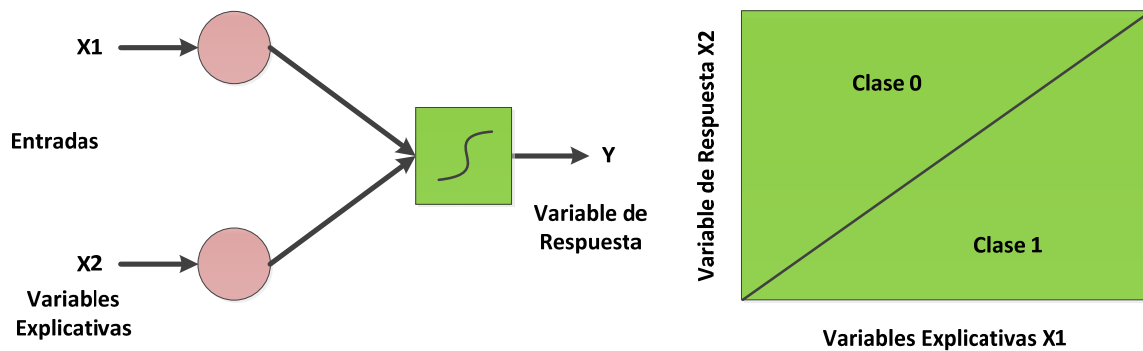


Figura 2-6 Perceptrón simple con función logística = Modelo de regresión logística

Un Perceptrón simple con función de activación umbral en la neurona de salida es similar a la Función Discriminante Lineal de Fisher (ver figura 2-7).

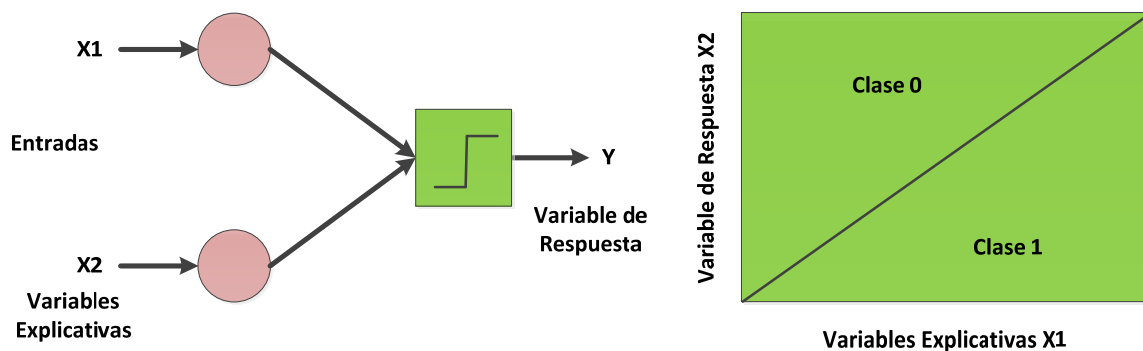


Figura 2-7 Perceptrón simple con función umbral = Análisis discriminante lineal.

Una red MLP compuesta por tres capas cuya capa oculta de neuronas utiliza una función de activación no lineal, puede ser vista como una generalización no lineal de los MLG. La principal virtud de una red MLP que permite explicar su amplio uso en el campo del análisis de datos es que se trata de un *aproximador universal de funciones* (XU, Oja, & Suen, 1992) (Minasny & AB, 2002). La base matemática de esta afirmación se debe a Kolmogorov (1957), quien constató que una función continua de diferentes variables puede ser representada por la concatenación de varias funciones continuas de una misma variable. Esto significa que un Perceptrón conteniendo al menos una capa oculta con suficientes unidades no lineales, tiene la capacidad de aprender virtualmente cualquier tipo de relación siempre que pueda ser aproximada en términos de una función continua. También utilizando más de una capa oculta, la red puede aproximar

relaciones que impliquen funciones discontinuas. Si no se utilizan funciones de activación no lineales en las capa oculta, la red queda limitada a actuar como un discriminador (aproximador lineal).

Las redes MLP son capaces de manejar tareas de elevada dimensionalidad mediante la utilización de arquitecturas relativamente sencillas. Esto explica el hecho de que no es necesario introducir explícitamente en el modelo las interacciones entre las variables explicativas, ya que las posibles interacciones son aprendidas por la red neuronal de forma automática en el proceso de entrenamiento.

Las RNA estiman los pesos en base a algún criterio de optimización sin tener en cuenta supuestos como el tipo de distribución o la dependencia funcional entre las variables. Por este motivo, las RNA han sido consideradas por muchos autores como modelos no paramétricos. Otros autores como Masters (1993) son más flexibles y sostienen que supuestos como normalidad, homogeneidad de variancias y aditividad en las variables de entrada son características recomendables para una red neuronal aunque no son estrictamente necesarias como sucede en los modelos estadísticos. Este conjunto de propiedades convierten las redes MLP en herramientas de propósito general, flexibles y no lineales.

Dependiendo del *tipo de función de activación* utilizado en la capa de salida, el MLP se puede orientar a la predicción o a la clasificación. Así, en caso de utilizar la función identidad en la capa de salida, estaríamos ante un modelo de regresión no lineal (ver figura 2-8).

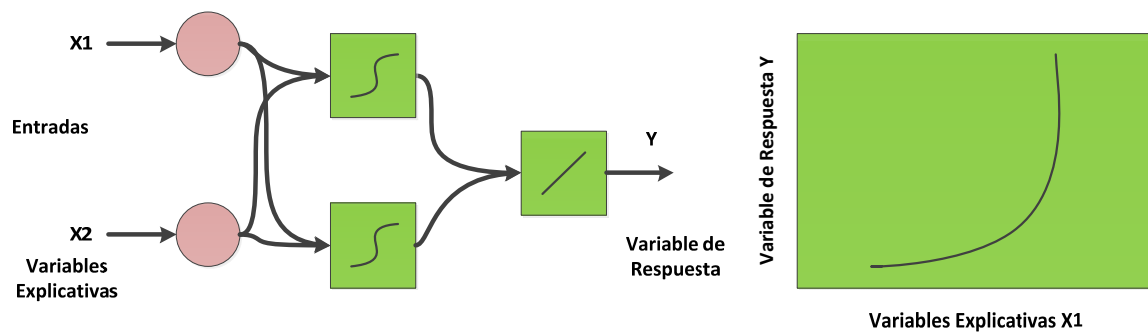


Figura 2-8 Perceptrón multicapa con función lineal en la salida = Modelo de regresión no lineal

Una red MLP con funciones de activación logísticas en las salidas puede ser utilizada como una Función Discriminante no lineal (ver figura 2-9). Como se puede observar en la figura, cada neurona oculta corresponde a un límite no lineal entre la clase 0 y la clase 1. Así, la utilización de un número considerable de neuronas ocultas permite obtener regiones de decisión arbitrariamente complejas.

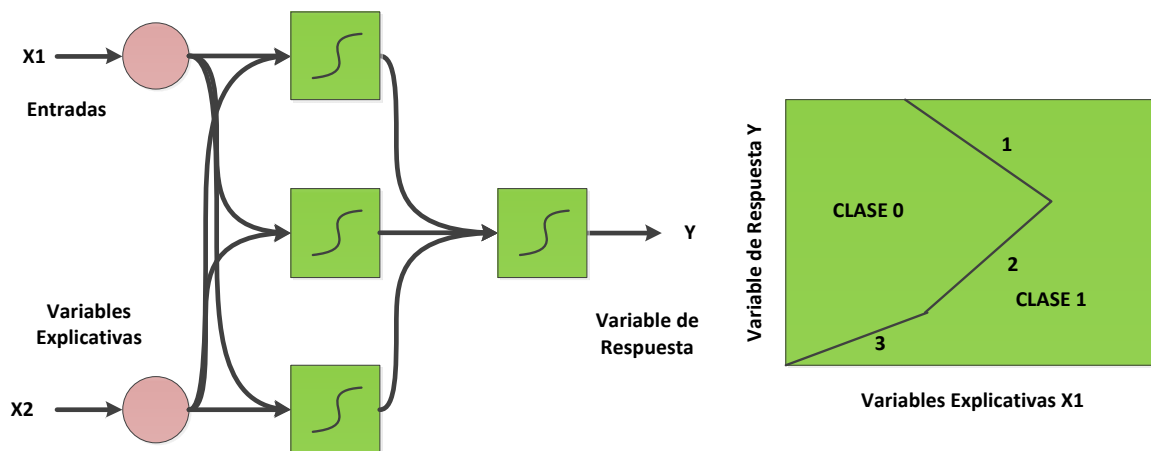


Figura 2-9 Perceptrón multicapa con función logística = Función discriminante no lineal

El método más utilizado para la estimación de los pesos de esta arquitectura MLP es el *backpropagation* o técnica de gradiente decreciente basado en el criterio de mínimos cuadrados. Esta técnica ya fue utilizada en el campo de la estadística antes del desarrollo de las RNA. Sin embargo, el algoritmo *backpropagation* padece dos defectos graves. Por un lado, el gradiente es un indicador extremadamente local de la función de cambio óptima. De forma que para zonas próximas entre sí de la superficie del error, el gradiente puede tomar direcciones opuestas. Estas fluctuaciones provocan que el tiempo de búsqueda del mínimo del error sea considerablemente largo, en lugar de tomar una ruta más directa. Por otro lado, no se sabe *a priori* cuál es el tamaño del cambio de los pesos más adecuado para una tarea dada. Este tamaño o magnitud está determinado por los valores de la *tasa de aprendizaje* y el *factor momento* (Al-Haik, Garmestani, & Navon, 2003). Un ritmo de aprendizaje demasiado pequeño ocasiona una disminución importante en la velocidad de convergencia y un aumento en la probabilidad de acabar atrapado en un mínimo local. En cambio, un ritmo de aprendizaje demasiado grande conduce a inestabilidades en la función de error con el peligro de poder pasar por encima del mínimo global. Por tanto, el valor de estos parámetros de aprendizaje se debe determinar mediante ensayo y error.

Se han propuesto, desde los campos de las RNA y la estadística, diversos métodos alternativos al *backpropagation* dirigidos a estimar los pesos de la red de una forma mucho más rápida y eficaz. Muchos de estos métodos alternativos son extensiones de la propia técnica de gradiente decreciente como la regla *delta-bar-delta* basada en la utilización de tasas de aprendizaje adaptativas aplicadas al valor del gradiente, el algoritmo RPROP (*Resilient propagation* basado en un método de aprendizaje adaptativo parecido a la regla *delta-bar-delta*, donde los pesos se modifican en función del signo del gradiente, no en función de su magnitud y, finalmente, el algoritmo *Quickprop* basado en modificar los pesos en función del valor del gradiente obtenido en la iteración actual y del gradiente obtenido en la iteración anterior

Por otra parte, existe un conjunto de algoritmos de optimización no lineal derivados del campo del análisis numérico y la estadística (Montaño Moreno, 2002), que se caracterizan por hacer uso no sólo de la información proporcionada por la derivada de primer orden del error con respecto a los pesos, como es el caso del *backpropagation* y sus extensiones, sino también de la información proporcionada por la derivada de segundo orden. Mientras la derivada de primer orden informa de la pendiente de la superficie del error, la derivada de segundo orden informa de la curvatura de dicha superficie. Así, si la primera derivada representa la velocidad de decremento, la segunda derivada representa el error.

La información adicional proporcionada por la derivada de segundo orden puede servir para acelerar o decelerar el descenso por la superficie dependiendo de la distancia a la cual se encuentre respecto al mínimo.

Uno de los métodos más conocidos en el campo de las RNA es el algoritmo de gradientes conjugados (Moller, 1993), el cual se basa en dividir la derivada de primer orden por la derivada de segundo orden para determinar el incremento de cada peso

$$\Delta w_i = \frac{\frac{\partial E}{\partial w_i}}{\frac{\partial^2 E}{\partial w_i^2}}$$

Este incremento representa la distancia necesaria para que la deceleración determine una velocidad igual a 0, que es el punto en el que se alcanza un mínimo en la función de error. Esta estrategia permite acelerar el proceso de aprendizaje de forma considerable con respecto a los métodos anteriores, alcanzando la convergencia de los parámetros de una forma más eficaz y directa. Otros métodos de segundo orden son los algoritmos de Newton, cuasi-Newton, Gauss-Newton, Newton-Raphson y Newton Truncado (Al-Haik, Garmestani, & Navon, 2003).

Existen otros modelos de RNA, a partir de los cuales también se puede establecer una analogía con modelos estadísticos clásicos conocidos. Este es el caso de las redes entrenadas mediante la regla de Oja

(1982, 1989), las cuales permiten realizar Análisis de Componentes Principales (PCA). La regla de Oja supone una modificación de la regla no supervisada de Hebb. La arquitectura de este tipo de red está compuesta por una capa de entrada con N neuronas y una capa de salida con M neuronas lineales (ver figura 10).

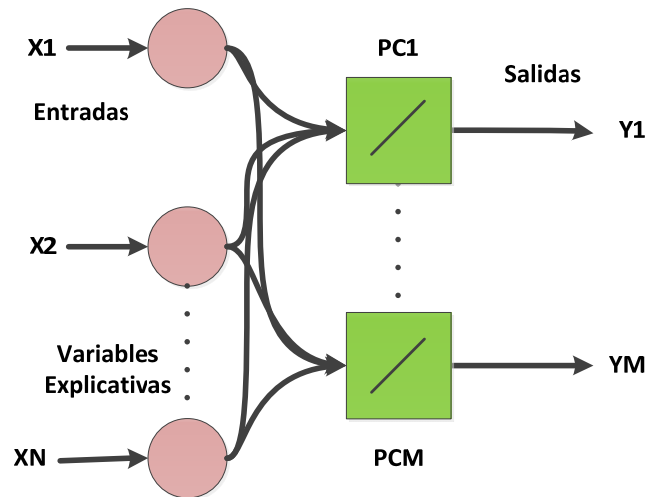


Figura 2-10 Regla de Oja = Análisis de componentes principales.

La red *backpropagation* autosupervisada o MLP autoasociativo es otro modelo de red que también ha sido aplicado al PCA y a la reducción de la dimensionalidad. Esta red fue utilizada inicialmente por Cottrell, Munro y Zipser (1989) para la compresión de imágenes y ha sido aplicada en el campo de la ingeniería. También se pueden establecer semejanzas entre determinados modelos estadísticos y las redes de función de base radial (*Radial Basis Function*: RBF) que constituyen la red más usada en problemas de predicción y clasificación, tras la red MLP, las RBF están compuestas de tres capas al igual que la red MLP (ver Figura 11).

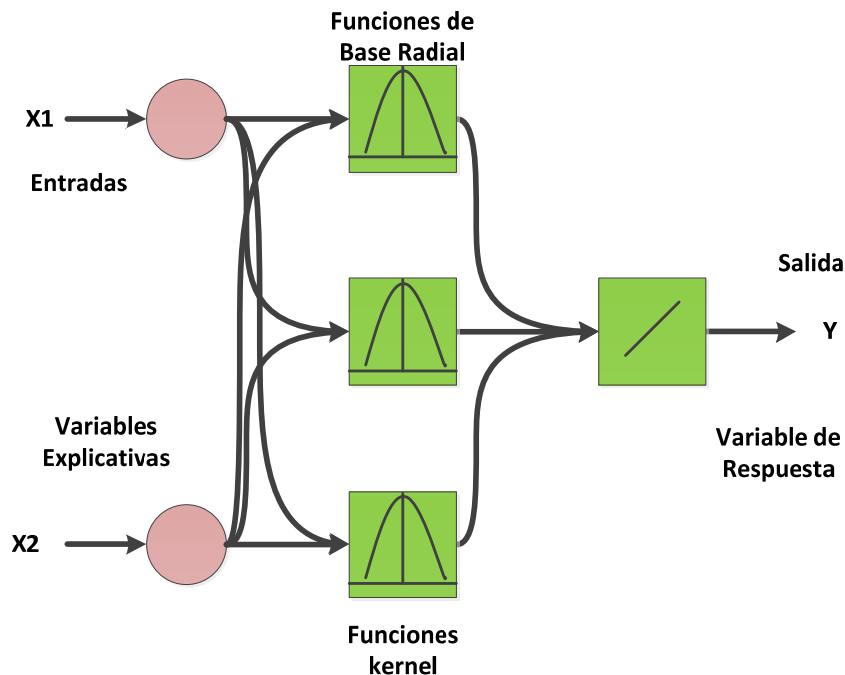


Figura 2-11 Red de función de base radial = Modelo de regresión kernel

Como la red MLP, las RBF permiten realizar con relativa facilidad modelados de sistemas no lineales arbitrarios y también constituyen aproximadores universales de funciones, con la particularidad de que el tiempo requerido para su entrenamiento suele ser mucho más reducido. Esto es debido en gran medida a que las redes RBF dividen el aprendizaje en dos fases. En una primera fase, los vectores de pesos o centroides asociados a las neuronas ocultas se pueden obtener mediante un aprendizaje no supervisado a través de un método estadístico como el algoritmo k-medias o a través de un método propio del campo de las RNA como el algoritmo de Kohonen utilizado en los mapas autoorganizados. En una segunda fase, los pesos de conexión entre las neuronas ocultas y las de salida se obtienen mediante un aprendizaje supervisado a través de la regla delta de Widrow-Hoff. Considerando las redes RBF como modelos de regresión no lineales, los pesos también podrían ser estimados por un método convencional como mínimos cuadrados no lineales o máxima-verosimilitud. A la familia de las redes RBF pertenecen, entre otras, la Red Neuronal Probabilística (PNN, *Probabilistic Neural Network*), la *General Regression Neural Network* (GRNN) y la *Counterpropagation* (Montaño Moreno, 2002).

2.3. Perceptrón Simple

De acuerdo a Frank Rosenblatt se define el perceptrón simple como: Supongamos que tenemos una función f de R^n en $\{-1,1\}$, que aplica un patrón de entrada $x = \{x_1, x_2, \dots, x_n\}^T \in R^n$ en la salida deseada $z \in \{-1,1\}$, es decir, $f(x) = z$. La información de que dispone sobre dicha función viene dada por p pares de patrones de entrenamiento

$$\{x_1, z_1\}, \{x_2, z_2\}, \dots, \{x_p, z_p\},$$

donde $x_i \in R^n$ y $f(x_i) = z_i \in \{-1,1\}$, $i = 1, 2, \dots, p$. Dicha función realiza una partición en el espacio R^n de patrones de entrada; por una parte estarían los patrones con salida $+1$ y por otra parte los patrones con salida -1 , por lo tanto, la función f clasifica a los patrones de entrada en dos clases.

Ejemplos de funciones f de este tipo son la función lógica OR o la función par. Ahora al construir un dispositivo sencillo que aprenda dicha función a partir de un conjunto conocido de patrones de entrenamiento, se utiliza una unidad de proceso bipolar (no lineal, ver Figura 2-12) que es una función matemática con dominio el conjunto n -dimensional $\{-1,1\}^n$ y rango el conjunto $\{-1,1\}$, definida por la siguiente expresión:

$$f(x_1, x_2, \dots, x_n) = \begin{cases} +1 & \text{si } w_1x_1 + w_2x_2 + \dots + w_nx_n \geq \theta \\ -1 & \text{si } w_1x_1 + w_2x_2 + \dots + w_nx_n < \theta \end{cases}$$

donde los parámetros $w_1 + w_2 + \dots + w_n$, se llaman pesos sinápticos y son los pesos con los que se ponderan los valores de entrada $x_1 + x_2 + \dots + x_n$, o argumentos de la función; la suma ponderada $u = w_1x_1 + w_2x_2 + \dots + w_nx_n$ se llama potencial sináptico y el parámetro θ se llama umbral o sesgo. También se puede expresar la función f mediante la función signo (ver figura 12), es decir:

$$f(x_1, x_2, \dots, x_n) = \text{sgn}(u - \theta),$$

siendo la función signo,

$$\text{sgn}(x) = \begin{cases} 1 & \text{si } x \geq 0 \\ -1 & \text{si } x < 0 \end{cases}.$$

Del mismo modo, se define una unidad de proceso binaria como una función matemática con dominio el conjunto n -dimensional $\{0,1\}^n$ y rango el conjunto $\{0,1\}$, definida por la siguiente expresión:

$$y = f(x_1, x_2, \dots, x_n) = \begin{cases} 1 & \text{si } w_1x_1 + w_2x_2 + \dots + w_nx_n \geq \theta \\ 0 & \text{si } w_1x_1 + w_2x_2 + \dots + w_nx_n < \theta \end{cases}$$

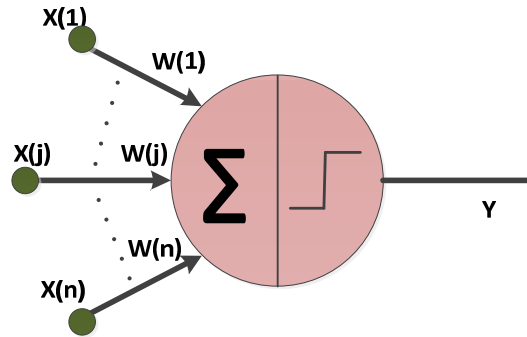


Figura 2-12 Unidad de proceso bipolar.

Cuando la salida de la unidad de proceso es igual a 1, se dice que dicha unidad de proceso está activada o encendida y presenta el estado 1, mientras que si su salida es igual a cero se dice que está desactivada o apagada, presentando el estado 0.

Para la determinación de los pesos sinápticos y del umbral se sigue un proceso adaptativo que consiste en comenzar con unos valores iniciales aleatorios e ir modificándolos iterativamente cuando la salida de la unidad no coincide con la salida deseada. La regla que se sigue para modificar los pesos sinápticos se conoce con el nombre de regla de aprendizaje del Perceptrón simple y viene dada por la expresión:

$$w_j(k+1) = w_j(k) + \Delta w_j(k), k = 1, 2, \dots,$$

siendo

$$\Delta w_j(k) = \eta(k)[z(k) - y(k)]x_j(k),$$

esto nos indica que la variación del peso w_j es proporcional al producto del error $z(k) - y(k)$ por la componente j -ésima del patrón de entrada que hemos introducido en la iteración k , es decir, $x_j(k)$. La constante de proporcionalidad $\eta(k)$ es un parámetro positivo que se llama factor de aprendizaje puesto que cuanto mayor sea, más se modifica el peso sináptico y viceversa, es decir, es el parámetro que controla el proceso de aprendizaje. Cuando es muy pequeño la red aprende poco a poco. Cuando se toma constante en todas las iteraciones, $\eta(k) = \eta > 0$ tendremos la regla de aprendizaje con incremento fijo.

Cuando la función de transferencia usada es la función signo (valores bipolares) la regla de aprendizaje se puede escribir de la forma:

$$w_j(k+1) = \begin{cases} w_j(k) + 2\eta x_j(k) & \text{si } y(k) = -1 \text{ y } z(k) = 1 \\ w_j(k) & \text{si } y(k) = z(k) \\ w_j(k) - 2\eta x_j(k) & \text{si } y(k) = 1 \text{ y } z(k) = -1 \end{cases},$$

por lo tanto, esta regla de aprendizaje es un método de detección del error y corrección. Solo aprende, es decir, modifica los pesos, cuando se equivoca. Cuando tenemos un patrón que pertenece a la primera clase ($z(k) = 1$) y no es asignado a la misma, entonces corrige el valor del peso sináptico añadiéndole una cantidad proporcional al valor de entrada, es decir, lo refuerza, mientras que el patrón de entrada no pertenece a esta clase y el Perceptrón lo asigna a ella, lo que hace es debilitar el peso restándole una

cantidad proporcional al patrón de entrada, y no se modifican los pesos cuando el valor deseado coincide con la salida de la red.

El umbral se modifica de la misma manera, teniendo en cuenta que el umbral se puede considerar como el peso sináptico correspondiente a un nuevo sensor de entrada que tiene siempre una entrada igual a $x_{n+1} = 1$ y como peso sináptico el valor del umbral, entonces tenemos:

$$w_1x_1 + w_2x_2 + \dots + w_nx_n \geq \theta \leftrightarrow w_1x_1 + w_2x_2 + \dots + w_nx_n + w_{n+1}x_{n+1} \geq \theta,$$

cuando $w_{n+1} = \theta$ y $x_{n+1} = -1$, así, la red equivalente tendría $n + 1$ sensores, su umbral sería siempre cero, los patrones de entrada $(x_1, x_2, \dots, x_n)$ serán ahora $(x_1, x_2, \dots, x_n - 1)$, los pesos asociados $(w_1, w_2, \dots, w_n)$ serán $(w_1, w_2, \dots, w_n, w_{n+1})$ con $w_{n+1} = \theta$ y la regla de aprendizaje será:

$$\Delta\theta(k) = -\eta(k)[z(k) - y(k)], \quad k = 1, 2, \dots$$

A partir de ahora vamos a considerar el umbral como un peso sináptico más, entonces el algoritmo que actualiza el vector de pesos sinápticos para cada patrón de entrada es:

Algoritmo de aprendizaje del Perceptrón con entrenamiento individualizado

Un vector $X \in \mathbb{R}^n$, donde $X = \{x_1, x_2, \dots, x_n\}^t$ es el vector de entrada.

Un vector $W \in \mathbb{R}^n$, donde $W = \{w_1, w_2, \dots, w_n\}^t$ es el vector de pesos sinápticos.

Un vector $Z \in \mathbb{R}^n$, donde $Z = \{z_1, z_2, \dots, z_n\}^t$ es la salida deseada.

Paso 1: Inicialización: Inicializar los pesos sinápticos con números aleatorios del intervalo $[-1, 1]$. Ir al paso 2 con $k = 1$.

Paso 2: (k-ésima iteración). Calcular $y(k) = \text{sgn}(\sum_{j=1}^n w_j x_j(k))$.

Paso 3: Corrección de los pesos sinápticos. Si $z(k) \neq y(k)$ modificar los pesos sinápticos según la expresión: $w_j(k+1) = w_j(k) + \eta[z(k) - y(k)]x_j(k)$, $j = 1, 2, \dots, n+1$.

Paso 4: Parada. Si no se han modificado los pesos en las últimas p iteraciones, es decir, $w_j(r) = w_j(k)$, $j = 1, 2, \dots, n+1, r = k+1, \dots, k+p$, parar. La red se ha estabilizado. En otro caso, ir al **Paso 2** con $k = k+1$.

Ahora dado un conjunto de patrones de entrenamiento, el Perceptrón aprende a clasificarlos correctamente solo si los patrones son linealmente separables. Esto reduce considerablemente el campo de aplicaciones del Perceptrón simple puesto que ni siquiera es capaz de implementar la función lógica XOR. En cambio, se pueden clasificar correctamente los patrones cuando se utiliza una función no lineal.

Decimos que dos conjuntos de puntos A y B son linealmente separables en un espacio n -dimensional si existen $n+1$ números reales $w_1, w_2, \dots, w_n, \theta$, de manera que cada punto $(x_1, x_2, \dots, x_n) \in A$ satisface $\sum_{i=1}^n w_i x_i \geq \theta$ y cada punto $(x_1, x_2, \dots, x_n) \in B$ satisface $\sum_{i=1}^n w_i x_i < \theta$.

Teorema de Convergencia del Perceptrón Simple: (Montaño Moreno, 2002) Si el conjunto de patrones de entrenamiento, $\{x_1, z_1\}, \{x_2, z_2\}, \dots, \{x_p, z_p\}$, es linealmente separable entonces el Perceptrón simple encuentra una solución en un número finito de iteraciones, es decir, consigue que la salida de la red coincida con la salida deseada para cada uno de los patrones de entrenamiento.

2.4. ADALINE

Otro modelo clásico de redes neuronales es la ADALINE, pues corresponde al acrónimo de **AD**aptive **L**inear **NE**uron) o neurona con adaptación lineal que fue introducida por Widrow en 1959. Esta neurona es similar al Perceptrón simple pero utiliza como función de transferencia la función

identidad en lugar de la función signo. La salida de la ADALINE es simplemente una función lineal de las entradas (ponderadas con los pesos sinápticos):

$$y = \sum_{i=1}^n w_i x_i - \theta.$$

Ahora la salida de la red es continua en lugar de binaria. Si consideramos una entrada adicional con $x_{n+1} = 1$ valor cuyo peso sináptico $w_{n+1} = \theta$, entonces podemos escribir simplemente

$$y = \sum_{i=1}^{n+1} w_i x_i.$$

Así, de forma general podemos tratar el valor umbral θ como un peso sináptico adicional con entrada igual a -1 .

Con la ADALINE se pretende implementar la correspondencia entre las entradas y las salidas de un sistema utilizando un conjunto finito de relaciones entre entradas y salidas. Supongamos que disponemos de p patrones de entrada $\{x_1, x_2, \dots, x_p\}$ y sus correspondientes salidas deseadas $\{d_1, d_2, \dots, d_p\}$. Se trata de determinar los pesos sinápticos que consiguen que las salidas de la red sean lo más parecidas a las salidas deseadas para el conjunto dado de patrones de entrenamiento. Es decir, se trata de determinar los pesos sinápticos de manera que se minimice la función de error cuadrático siguiente:

$$E = \frac{1}{2} \sum_{k=1}^p (d_k - y(k))^2 = \frac{1}{2} \sum_{k=1}^p \left(d_k - \sum_{j=1}^{n+1} w_j(k) x_j^k \right)^2.$$

Para ello vamos a seguir el método de descenso del gradiente, es decir, en la misma dirección y en sentido opuesto al gradiente. Si en la iteración k hemos introducido el patrón de entrenamiento x_k , cuya salida deseada es d_k , y los pesos sinápticos son $w_j(k)$, $j = 1, 2, \dots, n$, entonces la modificación de los mismos motivada por dicho patrón es:

$$w_r(k+1) = w_r(k) + \Delta w_r(k),$$

donde

$$\Delta w_r(k) = -\eta \frac{\partial E}{\partial w_r(k)} = -\eta (d_k - y(k)) x_r^k, j = r, 2, \dots, n+1.$$

El parámetro η controla la longitud del paso que vamos a dar en la dirección opuesta del gradiente. Conforme mayor sea η mayor será la cantidad por la que se modificarán los pesos sinápticos. Dicho parámetro debe ser un valor pequeño para evitar dar pasos demasiado largos, es decir, que nos lleven a soluciones peores que la que teníamos, puesto que el método del gradiente solamente garantiza el decrecimiento de la función de error si nos desplazamos en la direcciones opuesta del gradiente pero en un entorno suficientemente pequeño. A η lo llamaremos factor de aprendizaje.

En el proceso de entrenamiento hemos introducido un patrón en cada iteración, por ello diremos que hemos realizado un aprendizaje en línea. También podemos introducir los p patrones directamente y comparar las salidas de la red con las salidas deseadas, pasando entonces a actualizar los pesos sinápticos, en cuyo caso diremos que el aprendizaje es por lotes. La modificación de los pesos sinápticos se hace tomando como función de error el error medio (se divide por p el error total), es decir,

$$E = \frac{1}{2p} \sum_{k=1}^p (d_k - y(k))^2 = \frac{1}{2p} \sum_{k=1}^p \left(d_k - \sum_{j=1}^{n+1} w_j(k) x_j^k \right)^2,$$

y así la regla de aprendizaje es:

$$\Delta w_r(k) = -\eta \frac{\partial E}{\partial w_j} = -\eta \frac{1}{p} \sum_{j=1}^{n+1} (d_k - y(k)) x_j^k.$$

Aquí el peso sináptico no depende del patrón introducido, es el mismo para los p patrones puesto que se actualiza conjuntamente según el error medio para los p patrones

2.5. Características de Operación

Las Redes Neuronales Artificiales pueden tener factores de pesos sinápticos fijos o adaptables. Las que tienen pesos adaptables emplean reglas de aprendizaje para ajustar el valor de la fuerza de una interconexión con otras neuronas. Si las neuronas utilizan pesos fijos, entonces su tarea debe estar previamente definida, los pesos serán determinados a partir de una descripción completa del problema. Por otra parte, los pesos adaptables son esenciales si no se conoce previamente; cual deberá de ser su valor correcto

El principal objetivo del entrenamiento de la red es encontrar los pesos óptimos para minimizar el error entre los valores de la entrada y las salidas de respuesta actual, existen dos tipos de aprendizaje, el supervisado y el no supervisado: El entrenamiento supervisado ocurre cuando se le proporciona a la red tanto la entrada como la salida correcta, y la red ajusta sus pesos tratando de minimizar el error de su salida calculada. Este tipo de entrenamiento se aplica por ejemplo, en el reconocimiento de patrones. El entrenamiento no supervisado se presenta cuando a la red se le proporcionan únicamente los estímulos, y la red ajusta sus interconexiones basándose únicamente en sus estímulos y la salida de la propia red.

Las reglas de aprendizaje determinan como la red ajustará sus pesos utilizando una función de error o algún otro criterio. La ley de aprendizaje adecuada se determina en base a la naturaleza del problema que se intenta resolver.

Las Redes Neuronales Artificiales adaptables tienen dos fases en su operación:

Entrenamiento de la red, es cuando el usuario proporciona a la red un número adecuado; de estímulos de entrada, y de salida, la red entonces ajusta su pesos de interconexión o sinapsis hasta que la salida de la red está lo suficientemente cerca de la salida correcta.

Recuperación de lo aprendido, cuando la red se le presenta un conjunto de estímulos de entrada y esta simplemente calcula su salida. Cuando la red emplea entrenamiento no supervisado, algunas veces será necesario que reajuste su sinapsis durante la fase de recuperación.

La gran diferencia del empleo de las redes neuronales en relación con otras aplicaciones de la computación radica en que no son algorítmicas, esto es no se programan haciéndoles seguir una secuencia predefinida de instrucciones. Las redes neuronales artificiales generan ellas mismas sus propias reglas, para asociar la respuesta a su entrada; es decir, aprende por ejemplos y de sus propios errores. El conocimiento de una redes neuronales artificiales se encuentra en la función de activación utilizada y en los valores de sus pesos.

2.6. Clasificación de las Redes Neuronales Artificiales

Respecto a la cuestión de cuántos tipos de redes neuronales existen actualmente, se puede decir que se trata de un número inabarcable. A continuación, se presenta la clasificación de las RNA más conocidas en función del tipo de aprendizaje utilizado: supervisado o no supervisado (Montaño Moreno, 2002):

❖ Supervisado.

➤ Con conexiones feedforward:

- Lineales:
 - Perceptrón (Rosenblatt, 1958).
 - Adaline (Widrow y Hoff, 1960).
- Perceptrón multicapa (Multilayer perceptron) (MLP):
 - Backpropagation (Rumelhart, Hinton y Williams, 1986).
 - Correlación en cascada (Cascade correlation) (Fahlman y Lebiere, 1990).
 - Quickpropagation (Quickprop) (Fahlman, 1988).
 - Delta-bar-delta (Jacobs, 1988).
 - Resilient Propagation (RPROP) (Riedmiller y Braun, 1993).
 - Gradiente conjugado (Battiti, 1992).
- Radial Basis Function (RBF) (Broomhead y Lowe, 1988; Moody y Darken, 1989):
 - Orthogonal Least Squares (OLS) (Chen, Cowan y Grant, 1991).
- Cerebellar Articulation Controller (CMAC) (Albus, 1975).
- Sólo clasificación:
 - Learning Vector Quantization (LVQ) (Kohonen, 1988).
 - Red Neuronal Probabilística (PNN) (Probabilistic Neural Network) (Specht, 1990).
- Sólo regresión:
 - General Regression Neural Network (GRNN) (Specht, 1991).

➤ Con conexiones feedback:

- Bidirectional Associative Memory (BAM) (Kosko, 1992).
- Máquina de Boltzman (Ackley, Hinton y Sejnowski, 1985).
- Series temporales recurrentes:
 - Backpropagation through time (Werbos, 1990).
 - Elman (Elman, 1990).
 - Finite Impulse Response (FIR) (Wan, 1990).
 - Jordan (Jordan, 1986).
 - Real-time recurrent network (Williams y Zipser, 1989).
 - Recurrent backpropagation (Pineda, 1989).
 - Time Delay NN (TDNN) (Lang, Waibel y Hinton, 1990).

➤ Competitivo:

- ARTMAP (Carpenter, Grossberg y Reynolds, 1991).
- Fuzzy ARTMAP (Carpenter, Grossberg, Markuzon, Reynolds y Rosen, 1992).
- Gaussian ARTMAP (Williamson, 1995).
- Counterpropagation (Hecht-Nielsen 1987, 1988, 1990).
- Neocognitrón (Fukushima, Miyake e Ito, 1983; Fukushima, 1988).

❖ No supervisado.

➤ Competitivo:

- Vector Quantization.
 - Grossberg (Grossberg, 1976).
 - Kohonen (Kohonen, 1984).

- Conscience (Desieno, 1988).
- Mapa Auto-Organizado (Self-Organizing Map) (Kohonen, 1982; 1995).
- Teoría de la Resonancia Adaptativa (Adaptive Resonance Theory, ART):
 - ART 1 (Carpenter y Grossberg, 1987a).
 - ART 2 (Carpenter y Grossberg, 1987b).
 - ART 2-A (Carpenter, Grossberg y Rosen, 1991a).
 - ART 3 (Carpenter y Grossberg, 1990).
 - Fuzzy ART (Carpenter, Grossberg y Rosen (1991b).
- Differential Competitive Learning (DCL) (Kosko, 1992).
- Reducción de dimensionalidad:
 - Regla de Oja (Oja, 1989).
 - Sanger (Sanger, 1989).
 - Differential hebbian (Kosko, 1992).
- Autoasociación:
 - Autoasociador lineal (Anderson, Silverstein, Ritz y Jones, 1977).
 - Brain-State-in-a-Box (BSB) (Anderson, Silverstein, Ritz y Jones, 1977).
 - Red de Hopfield (1982).

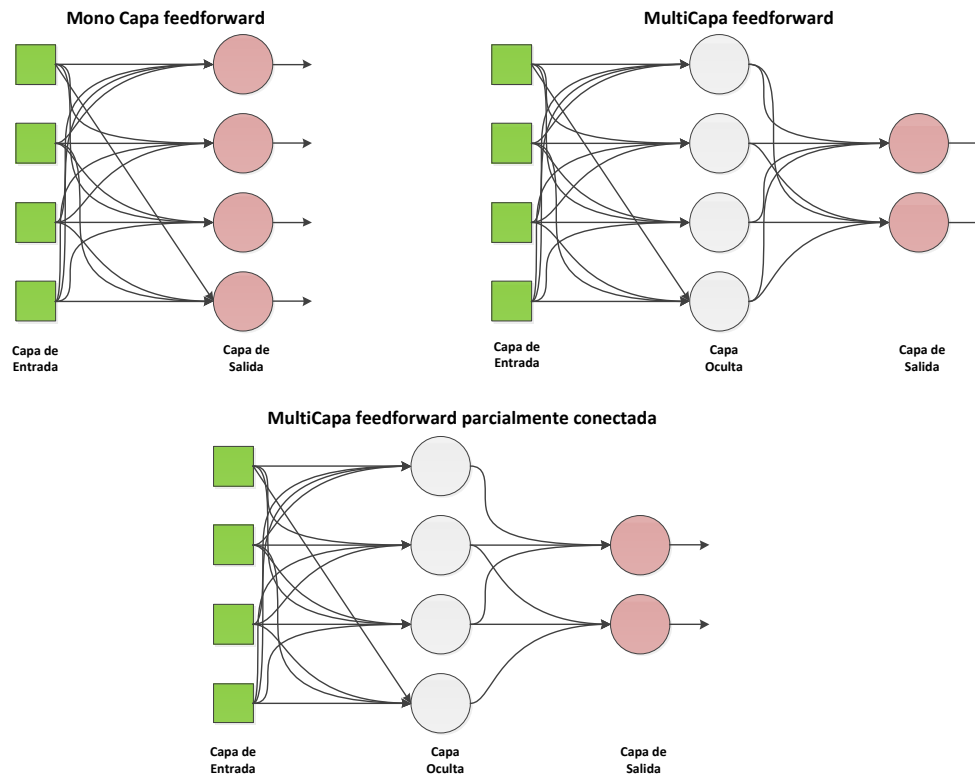


Figura 2-13 Modelado Neuronal

En este trabajo se utilizan las Redes Neuronales Multicapa, en especial el Perceptrón Multicapa (MLP) con Aprendizaje Supervisado.

3. Integración Numérica

A lo largo de este capítulo se mostrarán los diferentes métodos numéricos tradicionales que existen para aproximar numéricamente la integral definida. Diferentes son las razones para llevar a cabo la integración numérica. La principal puede ser la de no poder realizar la integración de forma analítica, es decir, algunas integrales no pueden ser resueltas de manera directa hallando su primitiva, solo mediante métodos numéricos, siendo la integración numérica de vital importancia.

La solución analítica de una integral nos arroja una solución exacta mientras que la solución numérica nos daría una solución aproximada. El error de la aproximación, que depende del método y del número de nodos que se utilicen, puede llegar a ser tan pequeño que es posible obtener un resultado idéntico a la solución analítica.

3.1. Métodos de Integración Numérica Tradicional

Un problema clásico es la aproximación de una integral es ¿Cuánto vale el área de la zona sombreada de la Figura 3?, al preguntamos por el valor del área bajo la curva, una forma de hacerlo es subdividiendo en rectángulos cada vez más pequeños y calcular el límite de la sucesión donde cada termino representa el área del rectángulo (ver Figura 3-1), lo cual se puede representar matemáticamente como:

$$\int_a^b f(x)dx = F(b) - F(a),$$

donde $F(x)$ se le conoce como la primitiva de $f(x)$, es decir, $f(x) = F'(x)$, pero cuyo cálculo no se puede llevar a cabo por medios analíticos. Este tipo de problemas se remontan a las Matemáticas clásicas en el cálculo de áreas y volúmenes de figuras curvas.

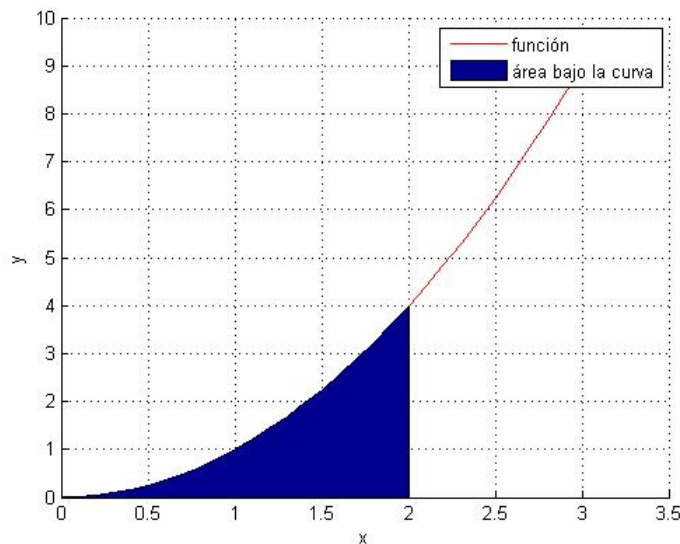


Figura 3-1. Área bajo la curva

Cuando no se conoce la antiderivada, se aplican aproximaciones numéricas. Los métodos numéricos para aproximar la integral definida son principalmente los métodos tipo Newton-Cotes, en el que se

incluye el método de Simpson, y los métodos tipo cuadraturas Gaussianas, en el que se incluye el método de Laguerre.

3.2. Fórmulas de Cuadratura

La propia definición de integral se fórmula actualmente en términos del límite de fórmulas de cuadratura. El Análisis Numérico trata de analizar no sólo la convergencia de estas fórmulas y otras que se puedan plantear, sino especialmente la calidad de estas aproximaciones estimando el error que cometen. En última medida, el uso local de estimaciones del error permiten explicar qué zonas del intervalo de integración concentran el error cometido por la fórmula y por tanto donde se concentran para reducir el error cometido. Obtenemos así un método adaptativo, que reconoce las dificultades del problema y se adapta a él (Burden & Douglas Faires, 2002).

Desde una óptica completamente ingenua, y a la vista de la gráfica de la función por integrar, podemos plantear su aproximación por una suma de áreas de rectángulos que aproximen el área total que define f (ver Figura 3-2). A este tipo de aproximaciones se les conoce como las Reglas del Rectángulo. Queda por elegir el punto que al evaluar define la altura de cada rectángulo. Las elecciones más sencillas son tomar el extremo inferior de cada subintervalo, el extremo superior o, a priori, tomar el punto medio.

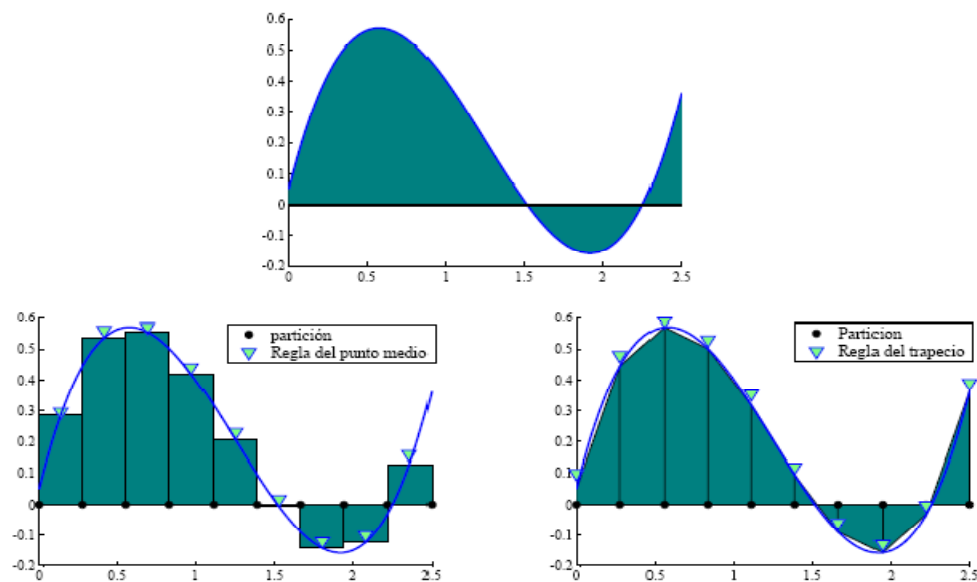


Figura 3-2. Reglas de Cuadratura

La interpolación con polinomios evaluada en puntos igualmente separados en $[a, b]$ da las fórmulas de Newton-Cotes (ver Figura 10), ejemplo de esto son la regla del Rectángulo, la regla del Trapecio y la regla de Simpson.

La deducción normal de las fórmulas de error de las cuadraturas se basa en determinar la clase de polinomios con los cuales las formulas producen resultados exactos. La definición siguiente sirve para facilitar la explicación:

Definición 1: El grado de exactitud o precisión de una fórmula de cuadratura es el entero positivo más grande n , tal que la formula sea exacta para x^p , cuando $p = 1, 2, \dots, n$.

Por definición la regla del Trapecio y de Simpson tienen, respectivamente, un grado de precisión de uno y tres, es decir, para polinomios de grado uno o menor y grado tres o menor, arrojan resultados exactos.

3.2.1. Regla del Rectángulo

El método más simple de este tipo es hacer a la función interpoladora es una función constante (un polinomio de orden cero) que pasa a través del punto $(a, f(a))$. Este método se llama la regla del Rectángulo:

$$\int_a^b f(x) dx \approx (b-a)f(a).$$

3.2.2. Regla del Punto Medio

Si en el método anterior la función pasa a través del punto $\left(\frac{a+b}{2}, f\left(\frac{a+b}{2}\right)\right)$ este método se llama la regla del punto medio (ver Figura 11):

$$\int_a^b f(x) dx \approx (b-a)f\left(\frac{a+b}{2}\right).$$

3.2.3. Regla del Trapecio

La función interpoladora puede ser una función afín (un polinomio de grado uno o sea una recta) que pasa a través de los puntos $(a, f(a))$ y $(b, f(b))$. Este método se llama regla del trapecio (ver Figura 12):

Suponemos que $h = \frac{b-a}{n}$, con n puntos o nodos, existe una $\xi \in [a, b]$ para la cual

$$\int_a^b f(x) dx \approx \frac{h}{2} [f(a) + f(b)] - \frac{h^3}{12} f''(\xi), \quad \text{donde } a < \xi < b.$$

3.2.4. Regla de Simpson

La función interpoladora puede ser un polinomio de grado dos que pasa a través de los puntos $(a, f(a))$, $\left(\frac{a+b}{2}, f\left(\frac{a+b}{2}\right)\right)$ y $(b, f(b))$. Este método se llama regla de Simpson (ver Figura 13):

$$\int_a^b f(x) dx \approx \frac{(b-a)}{6} \left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right] - \frac{h^5}{90} f^4(\xi), \quad \text{donde } a < \xi < b.$$

3.2.5. Reglas Compuestas

Para cualquier regla interpoladora, se puede hacer una aproximación más precisa dividiendo el intervalo $[a, b]$ en algún número n de subintervalos, hallando una aproximación para cada subintervalo, y finalmente sumando todos los resultados. Las reglas que surgen de hacer esto se llaman reglas compuestas, y se caracterizan por perder un orden de precisión global frente a las cuadraturas simples, si bien globalmente dan valores más precisos de la integral, a costa de incrementar significativamente el coste operativo del método. Por ejemplo, la regla compuesta de Simpson para n subintervalos de propósito general puede expresarse con su término error como:

$$\int_a^b f(x) dx \approx \frac{h}{3} \left[f(a) + 2 \sum_{j=1}^{\left(\frac{n}{2}\right)-1} f(x_{2j}) + 4 \sum_{j=1}^{\frac{n}{2}} f(x_{2j-1}) + f(b) \right] - \frac{b-a}{180} h^4 f^{(4)}(\xi).$$

3.3. Integración Gaussiana

En el estudio de fórmulas de Newton-Cotes vimos que los pesos estaban elegidos de manera que la fórmula tuviese grado máximo de precisión aunque la elección de los nodos, es decir, de los puntos en los que se evalúa la función, está fija a priori.

Podemos plantearnos ahora hacer de la posición de los nodos una variable más del problema. Para ello, y con objeto de simplificar el análisis, nos situamos en el intervalo $[-1,1]$ y buscamos la fórmula de mayor precisión de dos puntos:

$$w_1, w_2, \xi_1, \xi_2 \in \mathbb{R}$$

$$w_1 p(\xi_1) + w_2 p(\xi_2) = \int_{-1}^1 f(x) dx, f \in \{1, x, x^2, x^3\},$$

es decir, tenemos cuatro incógnitas ξ_1, ξ_2 (los nodos de la fórmula) y w_1, w_2 (los pesos), y exigimos que la fórmula integre de forma exacta a polinomios de grado 3. En principio este problema está bien planteado.

Supongamos que $f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$ donde a_i con $i = 0, 1, 2, 3$ son constantes, integrando e igualando se obtiene el sistema de ecuaciones no lineales siguiente:

$$\begin{cases} w_1 + w_2 = 2 \\ w_1 \xi_1 + w_2 \xi_2 = 0 \\ w_1 \xi_1^2 + w_2 \xi_2^2 = \frac{2}{3} \\ w_1 \xi_1^3 + w_2 \xi_2^3 = 0 \end{cases}$$

La solución de dicho sistema es $w_1 = w_2 = 1, \xi_1 = -\frac{\sqrt{3}}{3}, \xi_2 = \frac{\sqrt{3}}{3}$.

En resumen, una elección de los nodos nada obvia define una regla de dos puntos con mejores propiedades de convergencia que, por ejemplo, la regla del trapecio.

Podemos proceder de la misma forma y dada una regla de cuadratura de n nodos exigir que éste integre de forma exacta a polinomios de grado $2n - 1$. Dada la dificultad del sistema no lineal resultante, con $n = 2$, se deduce que el problema debe ser atacado desde un punto de vista muy diferente.

Existe una teoría ya clásica que demuestra que:

1. Existe una única regla de cuadratura que cumpla esas condiciones.
2. Los pesos w_i son siempre positivos.

El segundo punto es importante, pues asegura que los pesos no pueden crecer sin control dado que al ser positivos y al integrar de forma exacta a las constantes $w_1 + \dots + w_n = 2$, esta situación que no se

daba en las fórmulas de Newton-Cotes. En concreto, los nodos de la fórmula de cuadratura son las raíces de una familia de polinomios, los polinomios de Legendre, que se encuentra tabulada en multitud de textos científicos.

3.4. Algoritmos Adaptivos

Las formulas anteriores tienen un importante defecto: todas asumen que el comportamiento de la función es más o menos uniforme en todo el intervalo de integración. La situación usual es que la función tenga zonas donde varía de forma brusca y zonas donde su comportamiento sea considerablemente más suave (ver Figura 3-3). Intuitivamente, se entiende que las primeras zonas son las más problemáticas. El siguiente paso en cualquier algoritmo numérico es diseñar métodos adaptativos (Domínguez Báúena & Rapú Banzo, 2006).

Estos esquemas reconocen aquellas zonas que requieren mayor y aquéllas donde basta unas pocas evaluaciones para obtener una aproximación suficientemente buena. Para abordar esta tarea debemos disponer en primer lugar de un buen estimador del error, esto es, de un post-proceso que nos dé información sobre el error que estamos cometiendo y que así permita dilucidar qué partes del intervalo de integración requieren mayor esfuerzo y cuáles no.

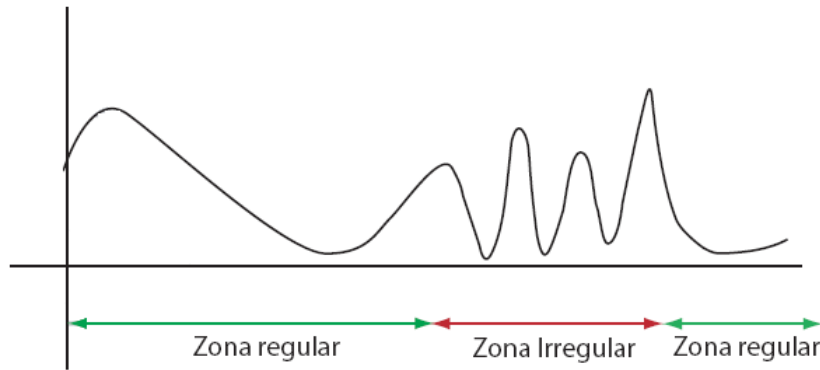


Figura 3-3. Integración Adaptiva

Hay implementaciones sencillas de integración adaptativa basadas en diferentes reglas como la regla de Simpson, se basa en comparar el resultado obtenido, en cada subintervalo, por la regla de Simpson simple, y la compuesta con dos subintervalos.

Aplicando la regla de Simpson al intervalo (a, b) :

$$\int_a^b f(x) dx - \underbrace{\frac{h}{6} (f(a) + 4f(c) + f(b))}_{Q_1(f)} = c_2^{(1)} h^4 + O(h^6).$$

Subdividiendo el intervalo en (a, c) y (c, b) , se tiene:

$$\int_a^b f(x) dx - \underbrace{\frac{h}{12} (f(a) + 4f(d) + 2f(c) + 4f(e) + f(b))}_{Q_2(f)} = \frac{c_2^{(1)}}{16} h^4 + O(h^6),$$

donde $h = (b - a)$, c es el punto medio de (a, b) , d y e , los puntos medios de (a, c) y (c, b) . El termino dominante del error, para h suficientemente pequeño, $c_2^{(1)}h^4$ de forma que:

$$\int_a^b f(x) dx - Q_1(f) \approx c_2^{(1)}h^4.$$

A priori este término no puede ser calculado, pero puede ser despejado de las formulas anteriores, sin más que sustraer a la primera entidad la segunda. Así obtenemos:

$$Q_2(f) - Q_1(f) \approx \left(1 - \frac{1}{16}\right)c_2^{(1)}h^4,$$

y por tanto

$$\frac{16}{15}[Q_2(f) - Q_1(f)] \approx c_2^{(1)}h^4 \approx \underbrace{\int_a^b f(x) dx - Q_1(f)}_{\text{error}}.$$

La idea del esquema adaptivo es la que sigue: dado un intervalo (a, b) y una tolerancia ε :

Calculamos $Q_1(f)$ y $Q_2(f)$ como antes, y

$$est = \left| \frac{16}{15}[Q_2(f) - Q_1(f)] \right|.$$

Si $est < \varepsilon$ el resultado se considera bueno y se devuelve $Q_2(f)$.

Si $est > \varepsilon$ se aplica el argumento anterior en los subintervalos (a, c) y en (c, b) con una tolerancia de $\frac{\varepsilon}{2}$ y se devuelve como integral la que es calculada en (a, c) y en (c, b) .

Hay por tanto un proceso de subdivisión de los intervalos, de manera que la única forma, en este estado inicial, de que un subintervalo no se divida es que la integral se calcule dentro de la tolerancia prefijada. Obviamente el algoritmo anterior se programa de forma recursiva.

4. Integración Numérica con Redes Neuronales

En este capítulo se presenta la aplicación de las Redes Neuronales Artificiales Supervisadas (RNAS) en la Integración Numérica y se proponen para esto seis algoritmos para el entrenamiento de estas Redes Neuronales.

Las dos primeras RNAS Back-Propagation y Factor Momentum son las redes clásicas que presentan un factor de aprendizaje constante, mientras que las otras cuatro Redes Neuronales aprovechan la potencialidad de los métodos de optimización como son Gradientes Conjugados, Gradientes Conjugados Residual, Newton y Newton Truncado para tener un factor de aprendizaje variable y así mismo lograr reducir el número de entrenamientos.

La arquitectura de la red neuronal está basada en funciones de cosenos como base del espacio de funciones. La red consiste en una capa de entrada y una de salida. En la capa de entrada tenemos un solo vector X de entrada con $(n + 1)$ elementos; la parte oculta consiste en un agente inteligente que tiene $(n + 1)$ neuronas; en la capa de salida solo tiene un vector Y con $(n + 1)$ elementos. La función de activación de la capa oculta es una vector CW con $(n + 1)$ elementos, donde W es el vector de pesos de la Red Neuronal con $(n + 1)$ elementos, C es una matriz de activación de la unidad oculta, y F es la salida deseada como vector salida (ver Figura 4-1).

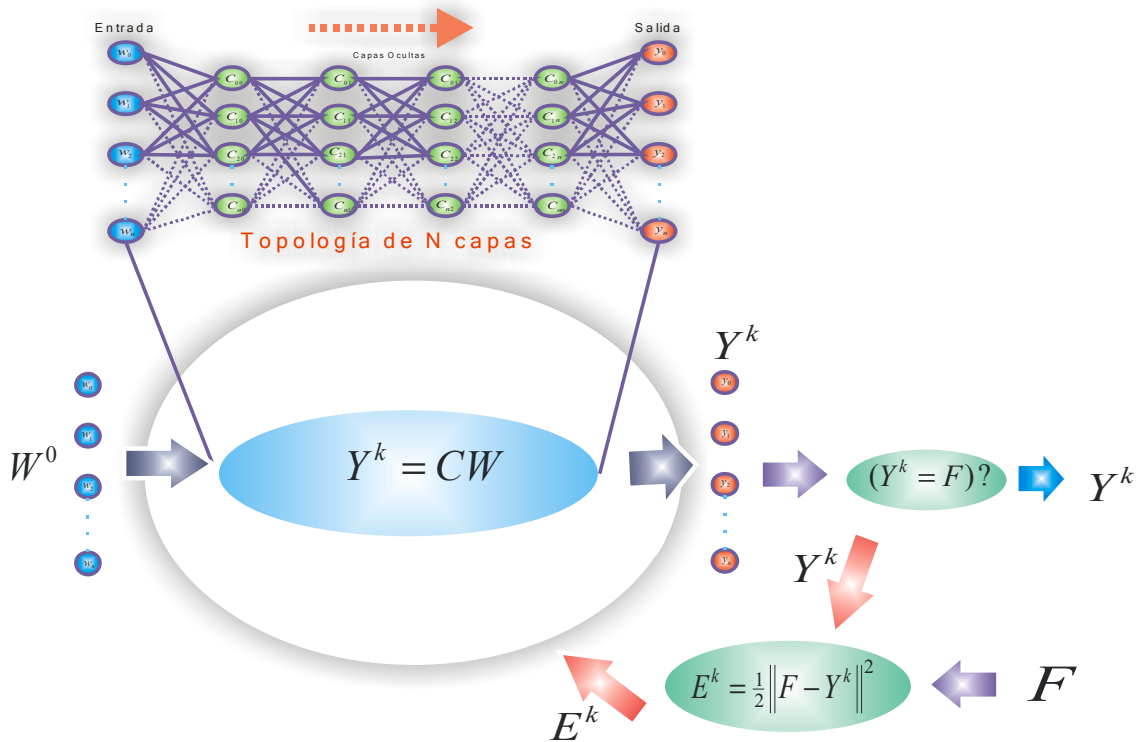


Figura 4-1. Arquitectura de la Red Neuronal

4.1. Definición de la Red Neuronal

Definimos la RNAS como sigue:

Suponemos que tenemos un patrón de entrada inicial $X = \{x_0, x_1, \dots, x_n\}^T \in R^{n+1}$ como muestra y dada una matriz de activación $C \in R^{(n+1) \times (n+1)}$ definida como:

$$C = \begin{bmatrix} 1 & 1 & \dots & 1 \\ \cos(x_0) & \cos(x_1) & \dots & \cos(x_n) \\ \vdots & \vdots & \dots & \vdots \\ \cos(nx_0) & \cos(nx_1) & \dots & \cos(nx_n) \end{bmatrix},$$

dado un vector de pesos sinápticos aleatorios $W = \{w_0, w_1, \dots, w_n\}^T \in \mathbb{R}^{n+1}$ y una salida deseada $F = \{f(x_0), f(x_1), \dots, f(x_n)\}^T$, donde $f(x)$ es la función muestra.

Ahora al construir un dispositivo sencillo que aproveche la función de la salida deseada F , a partir de un conjunto de datos conocidos de entrenamiento X , se utiliza la matriz de activación, que es una matriz construida en base a los patrones de entrada en un dominio $(n+1)$ -dimensional, obtenemos una salida deseada $Y = \{y(x_0), y(x_1), \dots, y(x_n)\}^T$ definida por:

$$Y = \begin{bmatrix} 1 & 1 & \dots & 1 \\ \cos(x_0) & \cos(x_1) & \dots & \cos(x_n) \\ \vdots & \vdots & \dots & \vdots \\ \cos(nx_0) & \cos(nx_1) & \dots & \cos(nx_n) \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix},$$

es decir:

$$Y = CW = F,$$

donde $W = \{w_0, w_1, \dots, w_n\}^T$ son llamados pesos sinápticos y son los pesos ponderados de la matriz de activación C , donde la suma ponderada es:

$$Y = \sum_{i=0}^n w_i \cos(ix),$$

que llamamos potencial sináptico y los parámetros $F = \{f(x_0), f(x_1), \dots, f(x_n)\}^T$ es el umbral o sesgo de la red neuronal.

4.2. Ajuste de la Función

Una función cualquiera se puede aproximar como una combinación de funciones cosenos (Burden & Douglas Faires, 2002), entonces, para aproximar la integral definida usando el enfoque de Redes Neuronales Artificiales, se hace en dos fases:

En la primera de estas fases se utilizan una red neuronal con aprendizaje supervisado para ajustar los pesos w_i de la combinación lineal siguiente:

$$y(x) = \sum_{i=0}^N w_i \cos(ix).$$

Dado que el ajuste de la función se hace en un intervalo $[a, b]$, cuando este intervalo es diferente que el intervalo $[0, \pi]$, se hace un cambio de variable $f(y) = f\left(a + \frac{b-a}{\pi} \cdot x\right)$, para mejorar el ajuste de la función. Este cambio de variable no es necesario si el intervalo $[a, b]$ está contenido en el intervalo $[0, \pi]$ (ver figura 4-2).

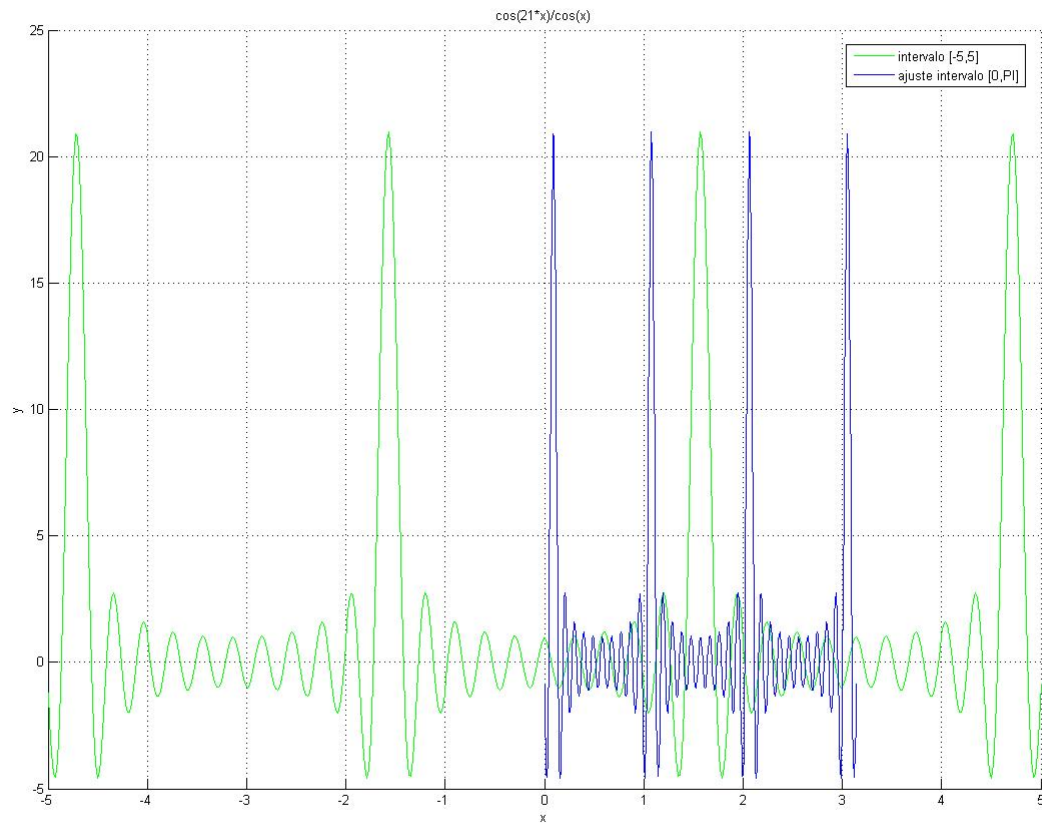


Figura 4-2. Cambio de Variable

A continuación se presentan las seis formas diferentes de entrenar las RNAS.

4.2.1. RNA Back-Propagation

El siguiente algoritmo muestra que todos los datos de la red se entrenan de forma sincronizada en forma de un vector y no entre cada componente del vector (uno por uno), que sigue un proceso adaptivo, que consiste iniciar con valores aleatorios los pesos sinápticos e ir modificando iterativamente cuando la salida no coincida con la salida deseada. (ver Figura 4-3).

Algoritmo RNA Back-Propagation

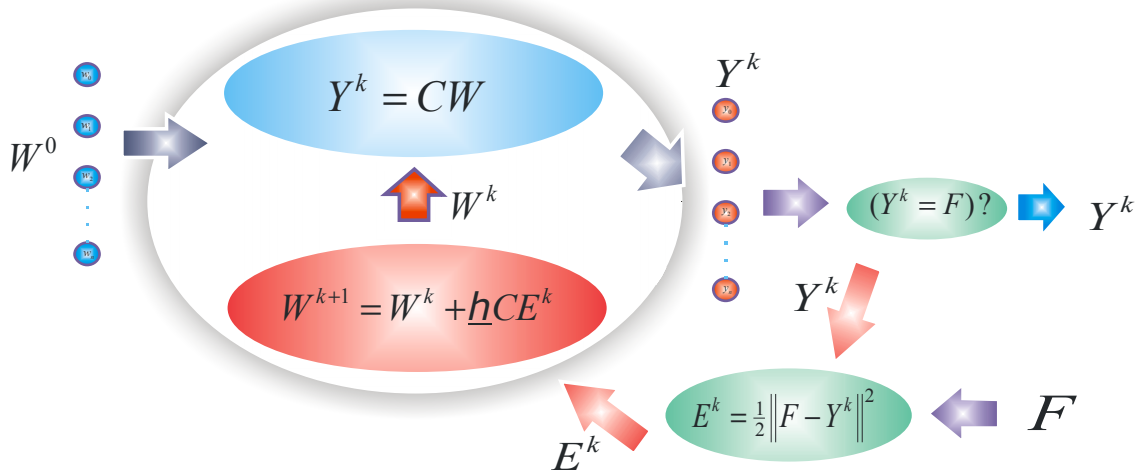


Figura 4-3. Arquitectura de la Red Neuronal Back-Propagation

Entonces como vimos podemos expresar la función $f(x)$ en forma de la siguiente ecuación:

$$y(x) = \sum_{i=0}^N w_i \cos(nx) \quad \text{ó} \quad Y = CW,$$

donde $x \in [0, \pi]$, con la función objetivo ϕ dada por el error cuadrático

$$\phi = \frac{1}{2} \|E^k\|^2, \quad (1)$$

donde

$$E^k = F - Y^k. \quad (2)$$

E^k es el vector error entre la red neuronal y la salida actual, y F es la salida deseada. Para minimizar ϕ , realizamos en forma recursiva para W^k el método de Gradiente con dirección descendente, en la misma dirección y sentido opuesto al gradiente. Tenemos que:

$$\min(\phi(W) = \frac{1}{2} \|E^k\|^2), \quad (3)$$

esto podemos denotarlo por la ecuación 1 y 2 como:

$$\min(\phi(W)), \text{ donde } \phi(W) = \frac{1}{2} \|E^k\|^2 = \frac{1}{2} E^{k^t} E^k.$$

Haciendo su desarrollo de Taylor tenemos que:

$$\phi(W) = \phi(W^k) + \nabla\phi(W^k)(W - W^k) + \frac{1}{2}(W - W^k)^t \nabla^2\phi(W^k)(W - W^k).$$

Por definición de Gradiente y la condición de primer orden $(\nabla\phi) = 0$ tenemos que:

$$\nabla\phi(W) = \nabla\phi(W^k) + \nabla^2\phi(W^k)(W - W^k) = 0, \quad (4)$$

es decir:

$$\begin{aligned}\nabla^2 \phi(W^k)(w - W^k) &= -\nabla \phi(W^k). \\ (W - W^k) &= -[\nabla^2 \phi(W^k)]^{-1} \nabla \phi(W^k). \\ W &= W^k - [\nabla^2 \phi(W^k)]^{-1} \nabla \phi(W^k).\end{aligned}\tag{5}$$

donde $\eta = [\nabla^2 \phi(W^k)]^{-1}$ y $\nabla \phi(W^k) = \frac{\partial \phi}{\partial W^k}$ sustituyendo en la ecuación (5)

$$W = W^k - \eta \frac{\partial \phi}{\partial W^k},$$

por lo tanto

$$W^{k+1} = W^k - \eta \frac{\partial \phi}{\partial W^k},\tag{6}$$

donde $\eta > 0$ es el factor de aprendizaje. La ecuación (6) es nuestra regla de aprendizaje que se sigue para modificar los pesos sinápticos de la RNA Back-Propagation.

Por otro lado, diferenciando el error cuadrático usando la regla de la cadena tenemos que:

$$\frac{\partial \phi}{\partial W^k} = \frac{\partial \phi^k}{\partial E^k} \frac{\partial E^k}{\partial Y^k} \frac{\partial Y^k}{\partial W^k},\tag{7}$$

donde

$$\text{Si } Y^k = C^t W \Rightarrow \frac{\partial Y^k}{\partial W^k} = C^t.$$

$$\text{Si } E^k = F - Y \Rightarrow \frac{\partial E^k}{\partial Y^k} = -1.$$

$$\text{Si } \phi = \frac{1}{2} \|E^k\|^2 \Rightarrow \frac{\partial \phi^k}{\partial E^k} = (E^k)^t.$$

Finalmente sustituyendo en la ecuación (7)

$$\frac{\partial \phi}{\partial W^k} = (E^k)^t (-1) C^t = -(E^k C)^t.$$

Recordando que $(AB)^t = B^t A^t$ y $(A^t)^t = A$ tenemos que

$$\nabla \phi = g^k = \frac{\partial \phi}{\partial W^k} = -C E^k.\tag{8}$$

Haciendo la segunda derivada de ϕ tenemos que la matriz Hessiana Q es:

$$\nabla^2 \phi = \frac{\partial}{\partial W^k} \left(\frac{\partial \phi}{\partial W^k} \right) = -\frac{\partial C E^k}{\partial W^k} = -\frac{\partial C E^k}{\partial E^k} \frac{\partial E^k}{\partial Y^k} \frac{\partial Y^k}{\partial W^k} = -C (-1) C^t = C C^t = Q.\tag{9}$$

Finalmente sustituyendo en la ecuación (6) tenemos que:

$$W^{k+1} = W^k + \eta CE^k. \quad (10)$$

Esto nos indica que la variación del vector de pesos W es proporcional a la suma del vector de pesos W anterior y el producto del error cuadrático con la matriz de activación en la iteración k , con una constante de proporcionalidad η positiva, definida como:

$$0 < \eta < \frac{2}{\|C\|^2}. \quad (11)$$

Teorema 1: (Zeng & Yao-Nan, Jul/Aug 2006) Cuando $0 < \eta < \frac{2}{\|C\|^2}$, el algoritmo de la red neuronal converge, donde η es el factor de aprendizaje y C es una matriz producida por una base de funciones cosenos.

Demostración

De la ecuación (6), tenemos que:

$$W^{k+1} = W^k - \eta \frac{\partial \phi}{\partial W^k} \Rightarrow \eta CE^k = \Delta W^k.$$

Por otra parte, de la ecuación (2), tenemos que:

$$E^k = F - Y^k = F - C^t W^k \Rightarrow \frac{\partial E^k}{\partial W^k} = -C^t.$$

Recordando que por definición:

$$\Delta E^k \triangleq \frac{\partial E^k}{\partial W^k} \Delta W^k = -C^t \eta CE^k = -\eta C^t CE^k = -\eta \|C\|^2 E^k, \quad \text{con } \|C\|^2 = C^t C.$$

Se propone que:

$$E^{k+1} = E^k + \Delta E^k = E^k - \eta \|C\|^2 E^k = (1 - \eta \|C\|^2) E^k.$$

Entonces de la ecuación (1), tenemos que:

$$\Delta \phi^k = \frac{1}{2} \|E^{k+1}\|^2 - \frac{1}{2} \|E^k\|^2 = \frac{1}{2} \|(1 - \eta \|C\|^2) E^k\|^2 - \frac{1}{2} \|E^k\|^2.$$

Usando la propiedad de normas de matrices $\|AB\| \leq \|A\| \|B\|$ tenemos que:

$$\Delta \phi^k \leq \frac{1}{2} \|(1 - \eta \|C\|^2)\|^2 \|E^k\|^2 - \frac{1}{2} \|E^k\|^2 \Rightarrow \Delta \phi^k \leq \frac{1}{2} \{ \|(1 - \eta \|C\|^2)\|^2 - 1 \} \|E^k\|^2.$$

Si aún no tenemos la solución, es decir, $E^k \neq 0$, entonces $\|E^k\|^2 > 0$. Si E^k es convergente tenemos:

$$E^{k+1} < E^k \Rightarrow E^{k+1} - E^k < 0 \Rightarrow \Delta \phi^k < 0,$$

por lo tanto

$$\{\|(1 - \eta\|C\|^2)\|^2 - 1\} < 0 \Rightarrow \|(1 - \eta\|C\|^2)\|^2 < 1.$$

Usando la siguiente propiedad $\|a\|^2 \leq 1$, $(-1 \leq a \leq 1)$, entonces:

$$\begin{aligned} -1 &\leq 1 - \eta\|C\|^2 \leq 1 \\ -2 &\leq -\eta\|C\|^2 \leq 0 \\ 0 &\geq \eta\|C\|^2 \geq 2. \end{aligned}$$

Por lo tanto tenemos que

$$0 \geq \eta \geq \frac{2}{\|C\|^2}.$$

Ahora podemos ajustar la red neuronal con el vector de pesos W , entonces el algoritmo que modifica el vector de pesos sinápticos es el siguiente:

Algoritmo 1 Red Neuronal Back-Propagation

Dados los siguientes parámetros iniciales:

Un $n \geq 5 \in N$ como número de nodos de la función.

Un vector $X \in R^{n+1}$ donde $X = \{x_0, x_1, \dots, x_n\}$ como patrón de entrada.

Un vector de pesos sinápticos aleatorios $W \in R^{n+1}$ donde $W = \{w_0, w_1, \dots, w_n\}$ en el intervalo $[0,1]$.

Se calcula una matriz de activación $C \in R^{(n+1) \times (n+1)}$, donde $C = \sum_{i=0}^n \cos(ix)$.

En el intervalo de integración $[a, b]$ se hace un cambio de variable al intervalo $[0, \pi]$.

Una salida deseada F donde $F = \{f(x_0), f(x_1), \dots, f(x_n)\}$.

Una tolerancia tol cualquiera y sea $k = 0$.

Un factor de aprendizaje $\eta = \frac{1.35}{\|C\|^2}$ (Zeng & Yao-Nan, Jul/Aug 2006)

Paso 1: Calcular el vector de salida dada la ecuación $Y^k = \sum_{i=0}^n w_i \cos(ix)$.

Paso 2: Calculamos el vector error con respecto a los valores reales deseados $E^k = \frac{1}{2} \|F - Y^k\|^2$.

Paso 3: Calculamos el nuevo vector de pesos $W^{k+1} = W^k + \eta C E^k$.

Paso 4: Hacer $k = k + 1$.

Paso 5: Si $tol < E^k$ entonces regresamos al Paso 1, sino ir al Paso 6.

Paso 6: Ir a la segunda fase (cálculo de la integral).

4.2.2. RNA Factor Momentum

Se conoce que el algoritmo anterior conocido también como el método descendente, ya que cada paso, el vector de pesos W es ajustado en la dirección en la cual, la función error decrece más rápidamente. Esta dirección es determinada por el gradiente de la superficie error del punto actual, en el espacio del vector de pesos. Usando el error del gradiente, podemos ajustar el vector de pesos para la conexión en una dirección negativa al gradiente. Lo cual tenemos de la ecuación (10):

$$W^{k+1} = W^k + \eta C E^k.$$

Para reducir la oscilación del método Back-Propagation, se usa el algoritmo de entrenamiento Momentum, donde se modifica la función error de modo que una porción del vector de pesos anterior es modificado completamente por el vector de pesos actual (ver Figura 4-4). Por lo tanto, el vector de pesos es actualizado según:

$$W^{k+1} = W^k + \eta CE^k + \lambda \Delta W^k.$$

Esto podemos reescribirlo como

$$W^{k+1} = W^k + \eta(\lambda + 1)CE^k, \quad (12)$$

donde λ es el factor Momentum ($0 < \lambda < 1$)

Algoritmo RNA Factor Momentum

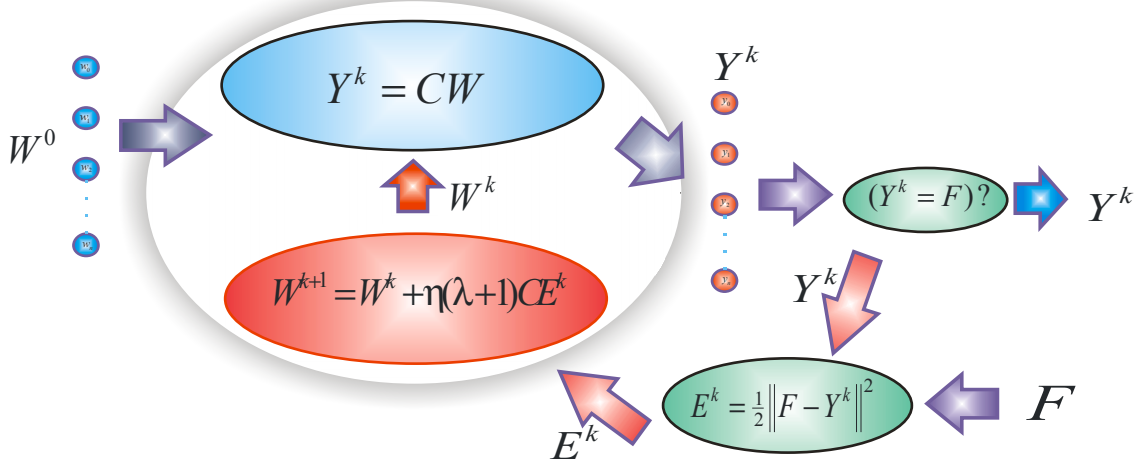


Figura 4-4. Arquitectura de la Red Neuronal Factor Momentum

Ahora podemos ajustar la red neuronal con el vector de pesos W , entonces el algoritmo que modifica el vector de pesos sinápticos es el siguiente:

Algoritmo 2 Red Neuronal Factor Momentum

Dados los siguientes parámetros iniciales:

Un $n \geq 5 \in N$ como número de nodos de la función.

Un vector $X \in R^{n+1}$ donde $X = \{x_0, x_1, \dots, x_n\}$ como patrón de entrada.

Un vector de pesos sinápticos aleatorios $W \in R^{n+1}$ donde $W = \{w_0, w_1, \dots, w_n\}$ en el intervalo $[0, 1]$.

Se calcula una matriz de activación $C \in R^{(n+1) \times (n+1)}$, donde $C = \sum_{i=0}^n \cos(ix)$.

En el intervalo de integración $[a, b]$ se hace un cambio de variable al intervalo $[0, \pi]$.

Una salida deseada F donde $F = \{f(x_0), f(x_1), \dots, f(x_n)\}$.

Una tolerancia tol cualquiera y sea $k = 0$.

Una factor de aprendizaje $\eta = \frac{1.35}{\|C\|^2}$ (Zeng & Yao-Nan, Jul/Aug 2006) y $\lambda = 1$.

Paso 1: Calcular el vector de salida dada la ecuación $Y^k = \sum_{i=0}^n w_i \cos(ix)$.

Paso 2: Calculamos el vector error con respecto a los valores reales deseados $E^k = \frac{1}{2} \|F - Y^k\|^2$.

Paso 3: Calculamos el nuevo vector de pesos $W^{k+1} = W^k + \eta(\lambda + 1)CE^k$.

Paso 4: Hacer $k = k + 1$.

Paso 5: Si $tol < E^k$ entonces regresamos al Paso 1, sino ir al Paso 6.

Paso 6: Ir a la segunda fase (cálculo de la integral).

4.2.3. RNA Gradientes Conjugados

Las aplicaciones de las Redes Neuronales Artificiales Supervisadas a menudo supone ajustar varios cientos de pesos, sólo el enfoque de optimización a problemas a gran escala son relevantes como alternativa de entrenamiento para estos algoritmos. El Gradientes Conjugados es bueno para manejar efectivamente problemas a gran escala, por lo que se usa este método para entrenar la red neuronal.

El algoritmo de Gradientes Conjugados combina las ventajas de la simplicidad del método de menor descenso y una mejor convergencia sin invertir la matriz Hessiana Q , el método de Gradientes Conjugados tal vez es uno de los más fáciles que se aplican a problemas de gran escala. Entonces para minimizar ϕ (ver ecuación 1), actualizamos el vector de peso W según una regla de Gradientes Conjugados (Luenbeger, 1989) (ver Figura 4-5):

$$W^{k+1} = W^k + \alpha^k p^k, \quad (13)$$

donde p^k es una dirección de descenso y α^k es el entrenamiento de aprendizaje adaptable con respecto a k iteraciones. Podemos describir el algoritmo de Gradientes Conjugados como sigue (Zeng & Yao-Nan, Jul/Aug 2006):

Tenemos que $g^k = \nabla \phi(x) = -CE^k$ (ver ecuación (8)):

El vector de aprendizaje adaptable viene dado por:

$$\alpha^k = \frac{\phi^k}{\|g^k\|^2}, \quad (14)$$

$$\beta^k = \frac{g^{k^t} Q g^k}{g^{k-1^t} Q g^{k-1}} = \frac{\|g^k\|^2}{\|g^{k-1}\|^2}, \quad (15)$$

donde $Q = CC^t$ (ver ecuación 9). Demostración de β^k :

De la regla de gradientes conjugados (González Velázquez, 1992):

$$p^{k+1} = -g^{k+1} + \beta^k p^k \quad (15a)$$

$$p^k = -g^k + \beta^{k-1} p^{k-1} \quad (15b)$$

$$g^k = QW^k - b \quad (15c)$$

$$g^{k+1} = QW^{k+1} - b \quad (15d)$$

Restando las ecuaciones (15d) y (15c) tenemos que:

$$g^{k+1} - g^k = Q(W^{k+1} - W^k). \quad (15e)$$

es decir:

$$(Q(W^{k+1} - W^k))^t = (W^{k+1} - W^k)^t Q^t.$$

De la ecuación (13) sumamos $-W^k$ y al multiplicar por la matriz Hessiana Q en ambos lados tenemos que:

$$Q(W^{k+1} - W^k) = \alpha^k Q p^k. \quad (15f)$$

Sustituyendo de la ecuación (15f) en la ecuación (15e) tenemos que:

$$g^{k+1} - g^k = \alpha^k Q p^k. \quad (15g)$$

Ahora en la ecuación (15b) la multiplicamos por $p^{k^t} Q$ tenemos que:

$$p^{k^t} Q p^k = -p^{k^t} Q g^k + \beta^{k-1} p^{k^t} Q p^{k-1}.$$

Siguiendo la propiedad de Q-ortogonalidad que dice $p^{k^t} Q p^{k-1} = 0$, por lo que tenemos:

$$p^{k^t} Q p^k = -p^{k^t} Q g^k. \quad (15h)$$

Ahora en la ecuación (15a) la multiplicamos por $p^{k^t} Q$ tenemos que:

$$0 = p^{k^t} Q p^{k+1} = -p^{k^t} Q g^{k+1} + \beta^k p^{k^t} Q p^k.$$

Entonces

$$\beta^k = \frac{p^{k^t} Q g^{k+1}}{p^{k^t} Q p^k}.$$

Sustituyendo la ecuación (15h), tenemos que

$$\beta^k = \frac{p^{k^t} Q g^{k+1}}{-p^{k^t} Q g^k} = \frac{\alpha^k p^{k^t} Q g^{k+1}}{-\alpha^k p^{k^t} Q g^k}.$$

Sustituyendo las ecuaciones (15g) y (15e) tenemos que:

$$\beta^k = \frac{(W^{k+1} - W^k)^t Q g^{k+1}}{-(W^{k+1} - W^k)^t Q g^k} = \frac{(g^{k+1} - g^k)^t g^{k+1}}{-(g^{k+1} - g^k)^t g^k},$$

usando $g^{k^t} g^{k+1} = 0$, nos queda que:

$$\beta^k = \frac{g^{k+1^t} Q g^{k+1}}{g^{k^t} Q g^k} = \frac{\|g^{k+1}\|^2}{\|g^k\|^2}.$$

Aquí, β^k es el índice de búsqueda conjugado ó tamaño de paso en la dirección p^{k-1} , de la nueva dirección conjugada p^k (ver Ecuación (15b)):

$$p^k = -g^k + \beta^{k-1} p^{k-1}.$$

Algoritmo RNA Gradientes Conjugados

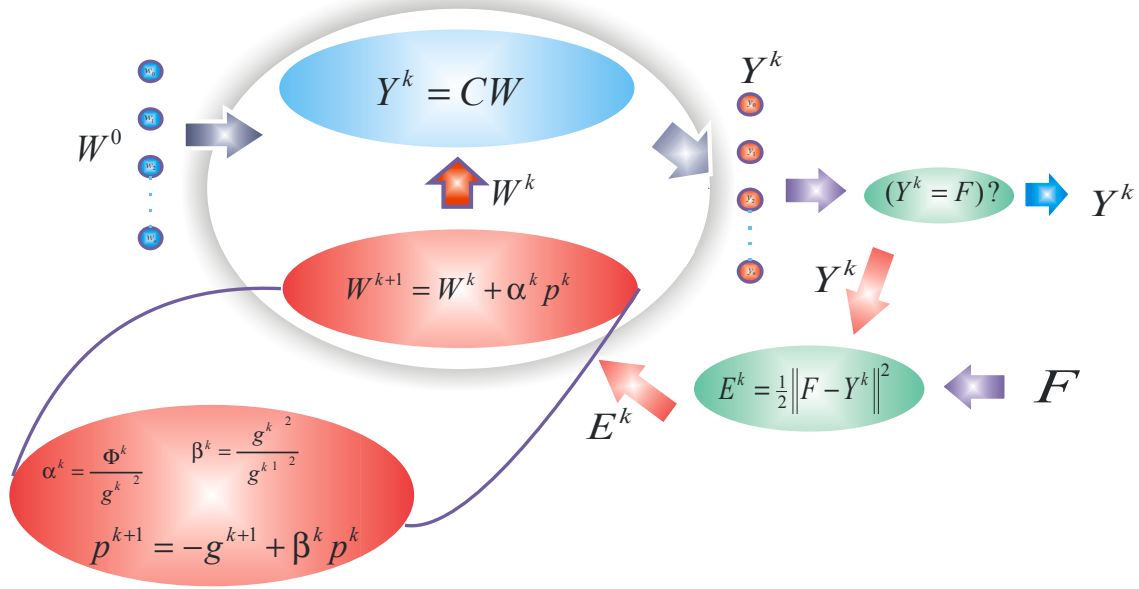


Figura 4-5. Arquitectura de la Red Neuronal Gradientes Conjugados

Ahora podemos ajustar la red neuronal con el vector de pesos W , entonces el algoritmo que modifica el vector de pesos sinápticos es el siguiente:

Algoritmo 3 Red Neuronal Gradientes Conjugados

Dados los siguientes parámetros iniciales:

Un $n \geq 5 \in \mathbb{N}$ como número de nodos de la función.

Un vector $X \in \mathbb{R}^{n+1}$ donde $X = \{x_0, x_1, \dots, x_n\}$ como patrón de entrada.

Un vector de pesos sinápticos aleatorios $W \in \mathbb{R}^{n+1}$ donde $W = \{w_0, w_1, \dots, w_n\}$ en el intervalo $[0, 1]$.

Se calcula una matriz de activación $C \in \mathbb{R}^{(n+1) \times (n+1)}$, donde $C = \sum_{i=0}^n \cos(ix)$.

En el intervalo de integración $[a, b]$ se hace un cambio de variable al intervalo $[0, \pi]$.

Una salida deseada F donde $F = \{f(x_0), f(x_1), \dots, f(x_n)\}$.

Una tolerancia tol cualquiera y sea $k = 0$.

Definimos $g^0 = \nabla \phi(0) = -CE^0$, $p^0 = CE^0$, donde E^0 es el primer vector error.

Paso 1: Calcular el factor de aprendizaje $\alpha^k = \frac{\phi^k}{\|g^k\|^2}$.

Paso 2: Calculamos el nuevo vector de pesos $W^{k+1} = W^k + \alpha^k p^k$.

Paso 3: Calcular el vector de salida dada la ecuación $Y^k = \sum_{i=0}^n w_i \cos(ix)$.

Paso 4: Calculamos el vector error con respecto a los valores reales deseados $E^k = \frac{1}{2} \|F - Y^k\|^2$.

Paso 5: Calcular el índice de búsqueda conjugado: $\beta^k = \frac{\|g^k\|^2}{\|g^{k-1}\|^2}$.

Paso 6: Actualizamos el vector dirección $p^{k+1} = -g^{k+1} + \beta^k p^k$.

Paso 7: Hacer $k = k + 1$.

Paso 8: Si $tol < E$ entonces regresamos al Paso 1, sino ir al Paso 9.

Paso 9: Ir a la segunda fase (cálculo de la integral).

4.2.4. RNA Gradientes Conjugados Residuales

El método de Gradientes Conjugados lo podemos expresar de manera residual (A Series of Books in Mathematics R.A. Rosenbaum G. Philip Johnson, 1963) (ver Figura 4-6). Sea Q una matriz definida positiva y sea $QW = -g$ se tiene un sistema lineal. La solución del sistema es conectado con el problema de hallar el vector que tiene un mínimo para la función.

$$H(X) = W^t QW - 2g^t W. \quad (16)$$

Que difiere solo por un elemento constante $W^{*t} QW^*$ para la función error $\phi(W) = \psi^t Q\psi$, donde W^* es la solución exacta del sistema y coincide con el vector que es el mínimo para $H(W)$; donde $\psi = W^* - W$ es el vector de error.

El problema se resuelve de la siguiente manera. Se escoge un vector arbitrario W^0 , se calcula la dirección opuesta al gradiente de la función $H(X)$ en este punto; esta dirección, la aproximación inicial, coincide con la dirección del residual, $r^0 = g^0 - QW^0$. Este punto W^0 lo movemos en una dirección dirigida al punto W^1 el cual es un mínimo de la función $H(W)$. Entonces tenemos, sustituyendo $W = W^0 + \alpha r^0$ en la Ecuación (16):

$$\begin{aligned} H(W) &= W^t QW - 2g^t W \\ &= (W^0 + \alpha r^0)^t Q(W^0 + \alpha r^0) - 2g^t (W^0 + \alpha r^0) \\ &= W^{0t} QW^0 + W^{0t} Q\alpha r^0 + \alpha r^{0t} QW^0 + (\alpha)^2 r^{0t} Qr^0 - 2g^t W^0 - 2\alpha g^t r^0 \\ &= H(W^0) + (\alpha)^2 r^{0t} Qr^0 + 2\alpha W^{0t} Qr^0 - 2\alpha g^t r^0 \\ &= H(W^0) + 2\alpha (W^{0t} Q - g^t) r^0 + (\alpha)^2 r^{0t} Qr^0 \\ &= H(W^0) - 2\alpha r^{0t} r^0 + (\alpha)^2 r^{0t} Qr^0 \\ &= H(W^0) - 2\alpha r^{0t} r^0 + (\alpha)^2 r^{0t} Qr^0 + \frac{(r^{0t} r^0)^2}{r^{0t} Qr^0} - \frac{(r^{0t} r^0)^2}{r^{0t} Qr^0} \\ &= H(W^0) + r^{0t} Qr^0 \left[\alpha^2 - \frac{2\alpha r^{0t} r^0}{r^{0t} Qr^0} + \left(\frac{r^{0t} r^0}{r^{0t} Qr^0} \right)^2 \right] - \frac{(r^{0t} r^0)^2}{r^{0t} Qr^0} \\ &= H(W^0) - \frac{(r^{0t} r^0)^2}{r^{0t} Qr^0} + r^{0t} Qr^0 \left[\alpha - \frac{r^{0t} r^0}{r^{0t} Qr^0} \right]^2. \end{aligned}$$

Donde esta expresión logra el mínimo para α

$$\alpha = \frac{r^{0t} r^0}{r^{0t} Qr^0}. \quad (17)$$

Así el mínimo es:

$$H(W^0) - \frac{(r^{0t} r^0)^2}{r^{0t} Qr^0}. \quad (18)$$

Luego $W^1 = W^0 + \alpha^0 r^0$ donde $r^0 = g - QW^0$ y $\alpha^0 = \frac{r^{0t} r^0}{r^{0t} Qr^0}$.

Y como este proceso se repite, de manera general se tiene que:

$$W^{k+1} = W^k + \alpha^k r^k. \quad (19)$$

Dado que

$$\begin{aligned} W^{k+1} &= W^k + \alpha^k r^k \\ W^{k+1} - W^k &= \alpha^k r^k \\ Q(W^{k+1} - W^k) &= \alpha^k Qr^k \\ QW^{k+1} - g - QW^k + g &= \alpha^k Qr^k \\ -r^{k+1} + r^k &= \alpha^k Qr^k, \end{aligned}$$

donde $r^k = g - QW^k$ y $r^{k+1} = r^k - \alpha^k Qr^k$, de manera general la ecuación 17 tenemos que:

$$\alpha^k = \frac{r^{k^t} r^k}{r^{k^t} Qr^k}. \quad (20)$$

Por lo tanto para minimizar el mínimo de la función ϕ tenemos que:

$$\phi(W^{k+1}) = \phi(W^k) - \frac{(r^{k^t} r^k)^2}{r^{k^t} Qr^k}. \quad (21)$$

Entonces tenemos que para minimizar ϕ , actualizamos el vector de peso W^k según una regla de Gradientes Conjugados $W^{k+1} = W^k + \alpha(k)p(k)$, donde p^k es una dirección de descenso para W , d^k es la dirección de descenso del residuo y α^k es el entrenamiento adaptable de aprendizaje con respecto a k iteraciones

Según la regla de Gradientes Conjugados tenemos que:

$$g^k = \nabla \phi(k) = -CE^k. \quad (22)$$

$$\alpha^k = \frac{\|r^k\|^2}{d^{k^t} q^k}, q^{k+1} = Qd^k, Q = CC^t. \quad (23)$$

$$p^{k+1} = p^k + \alpha^k d^k. \quad (24)$$

$$r^{k+1} = r^k - \alpha^k q^k. \quad (25)$$

$$\beta^{k+1} = \frac{\|r^{k+1}\|^2}{\|r^k\|^2}. \quad (26)$$

$$d^{k+1} = r^{k+1} + \beta^k d^k. \quad (27)$$

Aquí, β^k es un índice de búsqueda conjugado.

Algoritmo RNA Gradientes Conjugados Residuales

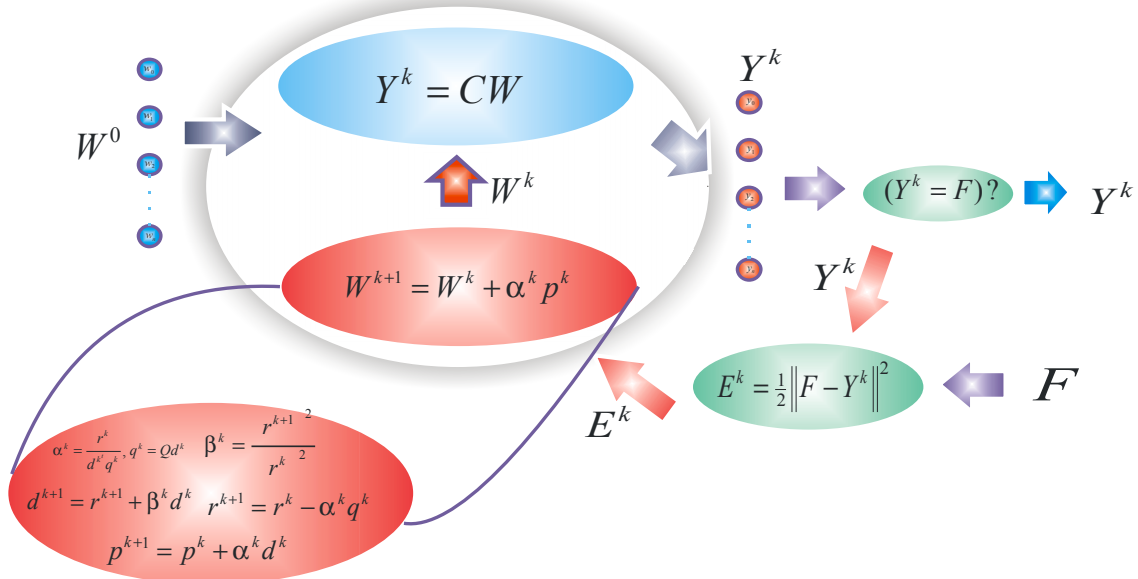


Figura 4-6. Arquitectura de la Red Neuronal Gradientes Conjugados Residuales

Ahora podemos ajustar la red neuronal con el vector de pesos W , entonces el algoritmo que modifica el vector de pesos sinápticos es el siguiente:

Algoritmo 4 Red Neuronal Gradientes Conjugados Residual

Dados los siguientes parámetros iniciales:

Un $n \geq 5 \in N$ como número de nodos de la función.

Un vector $X \in R^{n+1}$ donde $X = \{x_0, x_1, \dots, x_n\}$ como patrón de entrada.

Un vector de pesos sinápticos aleatorios $W \in R^{n+1}$ donde $W = \{w_0, w_1, \dots, w_n\}$ en el intervalo $[0,1]$.

Se calcula una matriz de activación $C \in R^{(n+1) \times (n+1)}$, donde $C = \sum_{i=0}^n \cos(ix)$.

En el intervalo de integración $[a, b]$ se hace un cambio de variable al intervalo $[0, \pi]$.

Una salida deseada F donde $F = \{f(x_0), f(x_1), \dots, f(x_n)\}$.

Una tolerancia tol cualquiera y sea $k = 0$.

Definimos $Q = CC^t$, $g^0 = \nabla \phi(0) = -CE^0$, $p^0 = W^0$, $r^0 = -g^0$, $d^0 = r^0$, donde E^0 es el primer vector error.

Paso 1: Calcular el factor de aprendizaje $\alpha^k = \frac{\|r^k\|^2}{d^k q^k}$, donde $q^k = Q d^k$.

Paso 2: Actualizar el vector dirección $p^{k+1} = p^k + \alpha^k d^k$.

Paso 3: Calcular el siguiente residual $r^{k+1} = r^k - \alpha^k q^k$.

Paso 4: Calcular $\beta^k = \frac{\|r^{k+1}\|^2}{\|r^k\|^2}$.

Paso 5: Calcular $d^{k+1} = r^{k+1} + \beta^k d^k$.

Paso 6: Calculamos el nuevo vector de pesos $W^{k+1} = W^k + \alpha^k p^k$.

Paso 7: Calcular el vector de salida dada la ecuación $Y^k = \sum_{i=0}^n w_i \cos(ix)$.

Paso 8: Calculamos el vector error con respecto a los valores reales deseados $E^k = \frac{1}{2} \|F - Y^k\|^2$.

Paso 9: Hacer $k = k + 1$.

Paso 10: Si $tol < E$ entonces regresamos al Paso 1, sino ir al Paso 11.

Paso 11: Ir a la segunda fase (cálculo de la integral).

4.2.5. RNA Newton

Un problema en programación no lineal sin restricciones consiste en: $\min\{\phi(W), W \in R^{n+1}\}$ tal que $\phi(W)$ es una función lineal con las siguientes propiedades: $\phi(W)$ es continua y, al menos, dos veces diferenciable, el objetivo consiste en obtener un mínimo local W^* ; es decir, un punto para el cual $\exists \delta > 0$ tal que $\phi(W^*) \leq \phi(W) \forall \|x - x^*\| < \delta$. Sea g el gradiente de la función $\phi(W)$ evaluado en W y Q su matriz Hessiana. Una condición suficiente y necesaria para W^* es que $g(W^*) = 0$ y $p^t Q p > 0 \forall d = W - W^*$.

El método clásico para obtener W^* es el método de Newton (Escudero, 1982) que, dada una estimación inicial, sea W^0 , obtiene una secuencia de direcciones de búsqueda $\{p^k\}$ tal que la iteración k resuelve el sistema.

$$Qp^k = -g^k. \quad (28)$$

Se obtiene la estimación $W^{k+1} = W^k + \alpha^k p^k$, donde α^k es el tamaño de paso en la iteración k , tal que minimiza $\phi(W)$ en la dirección p^k , la solución de $\alpha^k = \min\{\phi(W^{k+1} + \alpha p^k): \alpha > 0\}$, denominado búsqueda lineal exacta, no es tan compleja como minimizar $\phi(W)$.

El método de Newton es importante como base para comparar métodos alternativos, una principal característica consiste en que es local y cuadráticamente convergente, unas inconveniencias principales son: no es globalmente convergente, no existe solución en el sistema si Q es singular, Q puede no ser definida positiva en problemas no convexos (y, por tanto, p^k puede no ser descendente) y, es preciso evaluar la matriz Hessiana y resolver el sistema n -dimensional en cada iteración k , se resuelve el sistema de ecuaciones usando la factorización de Cholesky (ver Figura 4-7).

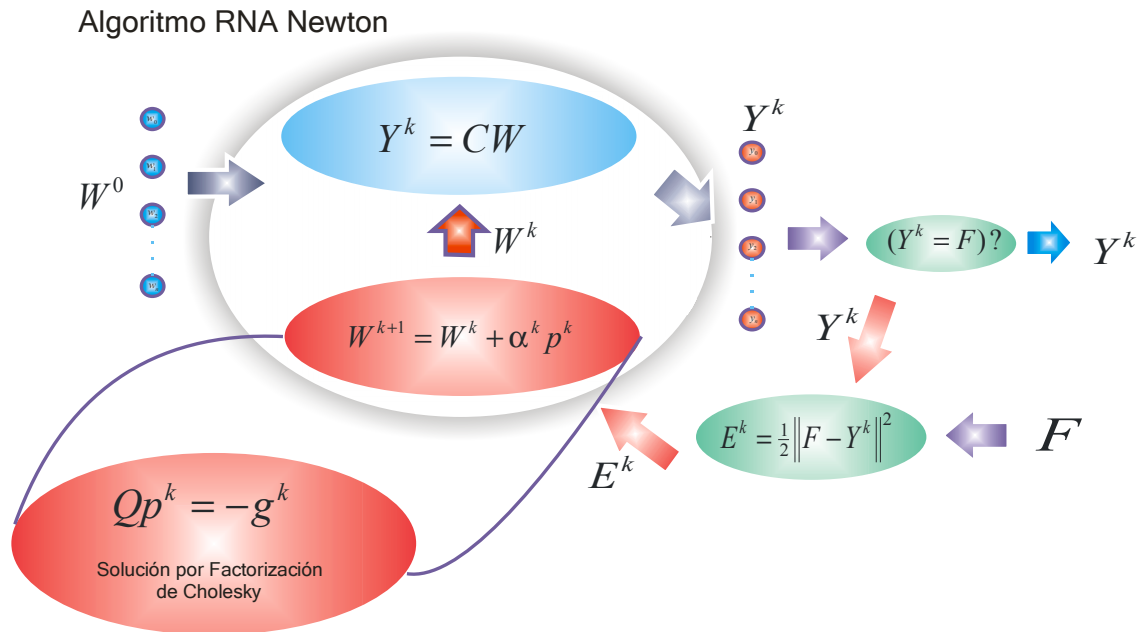


Figura 4-7. Arquitectura de la Red Neuronal Newton

Ahora podemos ajustar la red neuronal con el vector de pesos W , entonces el algoritmo que modifica el vector de pesos sinápticos es el siguiente:

Algoritmo 5 Red Neuronal Newton

Dados los siguientes parámetros iniciales:

Un $n \geq 5 \in N$ como número de nodos de la función.

Un vector $X \in R^{n+1}$ donde $X = \{x_0, x_1, \dots, x_n\}$ como patrón de entrada.

Un vector de pesos sinápticos aleatorios $W \in R^{n+1}$ donde $W = \{w_0, w_1, \dots, w_n\}$ en el intervalo $[0,1]$.

Se calcula una matriz de activación $C \in R^{(n+1) \times (n+1)}$, donde $C = \sum_{i=0}^n \cos(ix)$.

En el intervalo de integración $[a, b]$ se hace un cambio de variable al intervalo $[0, \pi]$.

Una salida deseada F donde $F = \{f(x_0), f(x_1), \dots, f(x_n)\}$.

Una tolerancia tol cualquiera y sea $k = 0$.

Definimos $Q = CC^t$, $g^0 = \nabla \phi(0) = -CE^0$, donde E^0 es el primer vector de error.

Paso 1:

Resolver el sistema $p^k = -Q^{-1}g^k$ usando Factorización de Cholesky para Q , $Q = LL^t$.

Resolviendo $L^t s = -g^k$ para s por sustitución hacia arriba.

Resolviendo $Ld^k = s$ para d^k por sustitución hacia abajo.

Paso 2: Calculamos el nuevo vector de pesos $W^{k+1} = W^k + \alpha^k p^k$, con $\alpha^k = 1$.

Paso 3: Calcular el vector de salida dada la ecuación $Y^k = \sum_{i=0}^n w_i \cos(ix)$.

Paso 4: Calculamos el vector error con respecto a los valores reales deseados $E^k = \frac{1}{2} \|F - Y^k\|^2$.

Paso 5: Actualizar el gradiente $g^{k+1} = -CE^k$.

Paso 6: Hacer $k = k + 1$.

Paso 7: Si $tol < E$ entonces regresamos al Paso 1, sino ir al Paso 8.

Paso 8: Ir a la segunda fase (cálculo de la integral).

4.2.6. RNA Newton Truncado

Considerando el problema de optimización sin restricciones, se quiere minimizar $\phi(W): W \in R^{n+1}$, donde $\phi: R^{n+1} \rightarrow R$ es no lineal, el mismo problema de optimización que los anteriores cumpliendo las mismas condiciones para que converja. El método de newton truncado produce buenos resultados en problemas de gran escala; su convergencia es súper-lineal, elimina inconvenientes teóricos del método de newton y no requiere obtener y almacenar la matriz Hessiana para ciertos casos, en nuestro caso la matriz Hessiana es una constante, este método es una extensión natural de Gradientes Conjugados para resolver sistemas. Al minimizar ϕ , actualizamos el vector pesos conforme a esta regla $W^{k+1} = W^k + \alpha^k p^k$, donde d^k es la dirección de búsqueda y α^k es el tamaño de paso de búsqueda (Al-Haik, Garmestani, & Navon, 2003)(ver Figura 4-8).

Hay 3 condiciones para que el método de Newton Truncado converja:

Caso 1: El vector gradiente g ; puntos en una dirección de curvatura negativa $g^t Q g < 0$: en este caso la iteración interna retorna con $p^k = -g^k$.

Caso 2: una dirección de curvatura negativa es encontrada en la iteración de Gradientes Conjugados $d^t Q d < 0$ antes de satisfacer el criterio de término de Newton Truncado, en este caso la iteración gradientes conjugados es terminada y p^k es tomado como la dirección de descenso.

Caso 3: El algoritmo termina por satisfacer el criterio de Newton Truncado o al término de n iteraciones.

Algoritmo RNA Newton Truncado

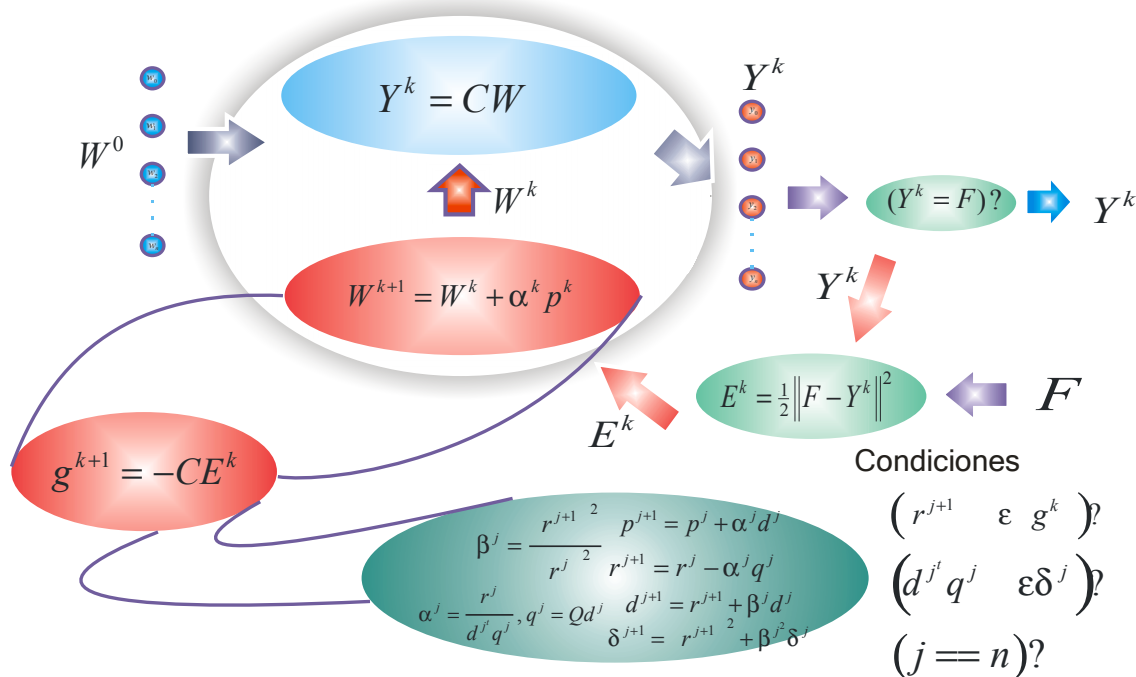


Figura 4-8. Arquitectura de la Red Neuronal Newton Truncado

Ahora podemos ajustar la red neuronal con el vector de pesos W , entonces el algoritmo que modifica el vector de pesos sinápticos es el siguiente:

Algoritmo 6 Red Neuronal Newton Truncado

Dados los siguientes parámetros iniciales:

Un $n \geq 5 \in N$ como número de nodos de la función.

Un vector $X \in R^{n+1}$ donde $X = \{x_0, x_1, \dots, x_n\}$ como patrón de entrada.

Un vector de pesos sinápticos aleatorios $W \in R^{n+1}$ donde $W = \{w_0, w_1, \dots, w_n\}$ en el intervalo $[0,1]$.

Se calcula una matriz de activación $C \in R^{(n+1) \times (n+1)}$, donde $C = \sum_{i=0}^n \cos(ix)$.

En el intervalo de integración $[a, b]$ se hace un cambio de variable al intervalo $[0, \pi]$.

Una salida deseada F donde $F = \{f(x_0), f(x_1), \dots, f(x_n)\}$.

Una tolerancia *tol* cualquiera y sea $k = 0$, $j = 0$, $\varepsilon = 10^{-10}$.

Definimos $Q = CC^t$, $g^0 = \nabla\phi(0) = -\tilde{C}\tilde{E}^0$, $p^0 = W^0$, $r^0 = -g^0$, $d^0 = r^0$, $\delta^0 = \|r^0\|^2$, donde E^0 es el primer vector de error.

Ciclo Externo

Paso 1: Calculamos el nuevo vector de pesos $W^{k+1} = W^k + \alpha^k p^k$, donde $\alpha^k = 1$.

Ciclo Interno

Paso 1: Calcular $q^j = Qd^j$.

Paso 2: Si $d^{jt} q^j \leq \varepsilon \delta^j$, detener y salir con $d^k = \begin{cases} d^0, j = 0 \\ p^j, j > 0. \end{cases}$

Paso 3: Calcular el factor de aprendizaje $\alpha^j = \frac{\|r^j\|^2}{d^j g^j}$.

Paso 4: Actualizar el vector dirección $p^{j+1} = p^j + \alpha^j d^j$.

Paso 5: Calcular el siguiente residual $r^{j+1} = r^j - \alpha^j q^j$.

Paso 6: Si $\|r^{j+1}\| \leq \varepsilon \|g^k\|$, detener y salir con: $d^k = p^{j+1}$.

Paso 4: Calcular $\beta^j = \frac{\|r^{j+1}\|^2}{\|r^j\|^2}$.

Paso 5: Calcular $d^{j+1} = r^{j+1} + \beta^j d^j$.

Paso 6: Calcular $\delta^{j+1} = \|r^{j+1}\|^2 + \beta^{j^2} \delta^j$.

Paso 7: Hacer $j = j + 1$.

Paso 8: Si $j = n$, detener y salir con: $d^k = p^{j+1}$.

Fin del Ciclo Interno

Paso 2: Calcular el vector de salida dada la ecuación $Y^k = \sum_{i=0}^n w_i \cos(ix)$.

Paso 3: Calculamos el vector error con respecto a los valores reales deseados $E^k = \frac{1}{2} \|F - Y^k\|^2$.

Paso 4: Actualizar el gradiente $g^{k+1} = -CE^k$.

Paso 5: Hacer $k = k + 1$.

Paso 6: Si $tol < E$ entonces regresamos al Paso 1, sino ir al Paso 7.

Paso 7: Ir a la segunda fase (cálculo de la integral).

4.3. Integración del Ajuste de la Función

En la segunda fase se obtiene la integral definida de la siguiente forma:

$$\int_a^b f(x) dx = \int_a^b \sum_{i=0}^N w_i \cos(ix) dx.$$

Teorema 2: (Zeng & Yao-Nan, Jul/Aug 2006) Sea $f(x) \in C[a, b]$, $0 \leq a, b \leq \pi$, y $W = [w_0, w_1, \dots, w_N]^t$ el vector pesos de la red neuronal. Si $y(x) = \sum_{i=0}^N w_i \cos(ix)$ es la salida de la red neuronal, entonces:

$$I = \int_a^b f(x) dx \approx (b - a)w_0 + \sum_{i=1}^N \frac{1}{i} w_i [\sin(ib) - \sin(ia)]. \quad (29)$$

De acuerdo con el cambio de variable, nos regresamos del intervalo $[0, \pi]$ al intervalo $[a, b]$, entonces la integral de este ajuste queda como:

$$I = \int_a^b f(x) dx \approx (b - a)w_0 + \sum_{i=1}^N \frac{b - a}{i} w_i [\sin(i\pi)]. \quad (30)$$

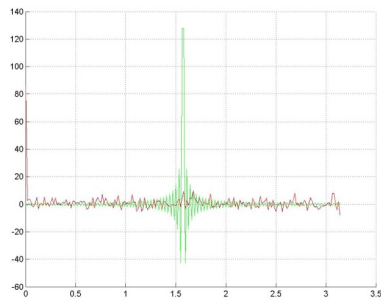
5. Pruebas y Resultados

En este capítulo se presentan los resultados obtenidos por las seis diferentes Redes Neuronales en tres partes, en la primera parte se presentan las pruebas realizadas a funciones continuas, en la segunda parte se presentan pruebas a funciones con singularidades y en la tercera parte se presenta la aplicación de las técnicas desarrolladas en esta tesis a la aproximación de Factores Franck-Condon de Morse para moléculas diatómicas con números cuánticos vibraciones grandes y pequeños, estos se comparan con resultados publicados. En las dos primeras partes se presentan los resultados del ajuste de la función y la aproximación de la integral definida

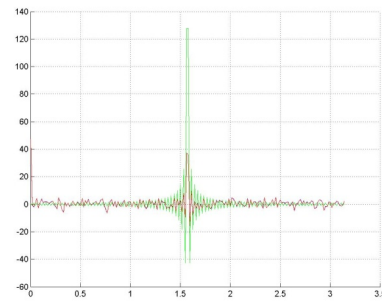
Las Redes Neuronales fueron implementadas en MatLab y en Java como software libre en una *laptop* hp-pavilion dv4-1140go Core2Duo, 2GHz, 4GB RAM, verificando los resultados con la integral real. Las funciones oscilantes de esta prueba son las siguientes (Gradshteyn & M., 2007):

- $\int \sin(mx) dx = -\frac{1}{m} \cos(mx).$
- $\int x^3 \sin(mx) dx = \left(\frac{3x^2}{m^2} - \frac{6}{m^4}\right) \sin(mx) + \left(\frac{6x}{m^3} - \frac{x^3}{m}\right) \cos(mx).$
- $\int_0^{\frac{\pi}{2}} x \tan(x) dx = -\pi \ln(2).$
- $\int_0^{\frac{\pi}{2}} \sin(m \cot(x)) \sin(2x) dx = \frac{m\pi}{2} e^{-m}.$
- $\int e^{ax} \sin(bx) dx = \frac{e^{ax}}{a^2 + b^2} (a \sin(bx) - b \cos(bx)).$
- $\int \sin(ax) \cos(ax) dx = \frac{\sin^2(ax)}{2a}.$
- $\int \sin(ax) \cos(bx) dx = -\frac{\cos(a-b)}{2(a-b)} - \frac{\cos(a+b)}{2(a+b)}.$
- $\int_0^{2\pi} (1 - \cos(x))^m \sin(mx) dx = 0.$
- $\int_0^{\pi} \frac{\cos((2m+1)x)}{\cos(x)} dx = (-1)^m \pi.$
- $\int \frac{\sin(3x)}{\cos(x)} dx = 2\sin^2(x) - 4\ln(\cos(x)),$

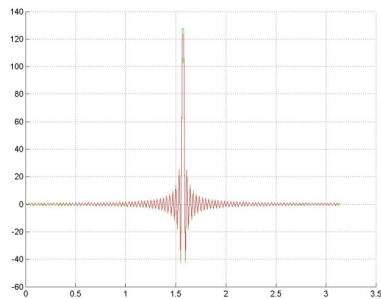
con $m = 5, 10, 50$ y 100 en todas las Redes Neuronales. Los resultados son presentados en las tablas siguientes. A continuación se presentan las gráficas del ajuste de la función $F = \frac{\cos(201x)}{\cos(x)}$ con 201 nodos, una tolerancia de $1e-10$, aumentando el número de entrenamientos.



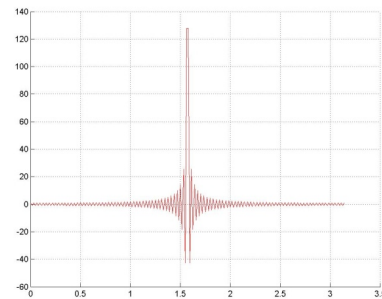
K=5 Entrenamientos



k=50 Entrenamientos



K=500 Entrenamientos



K=1000 Entrenamientos

Figura 5-1. Aproximación de la función $F = \frac{\cos(201x)}{\cos(x)}$ con 201 nodos, una tolerancia de $1e-10$, aumentando el número de entrenamientos

De la Figura 5-1 se observa que al dejar fijo el número de nodos y una cierta tolerancia, la aproximación de la función mejora al aumentar el número de entrenamientos.

5.1. Funciones Continuas

Al hacer las pruebas para el ajuste de las diferentes funciones usando la RNA de Newton dado que solo se requiere un solo entrenamiento, se varía el argumento ($m=5$, $m=10$, $m=50$, $m=100$), es decir, haciendo la función más oscilante y variando el número de nodos desde $n=5$ hasta $n=3000$, de diez en diez, donde no hay diferencia si el número de nodos es par o impar. En la tabla 1 se reporta el mínimo y máximo error en la aproximación de la función.

Función	Aproximación a la Función			
	n	Menor Error	n	Mayor Error
$\sin(5x)$	5	7.087e-031	3000	4.963e-020
$\sin(10x)$	5	7.965e-031	3000	4.820e-020
$\sin(50x)$	5	1.020e-031	3000	5.410e-020
$\sin(100x)$	5	2.196e-031	3000	5.033e-020
$x^3 \sin(5x)$	5	2.096e-027	3000	5.732e-020
$x^3 \sin(10x)$	5	1.131e-026	3000	6.089e-020
$x^3 \sin(50x)$	10	1.082e-026	3000	3.187e-019
$x^3 \sin(100x)$	5	7.964e-027	3000	8.387e-019
$\sin(5 \cot(x)) \sin(2x)$	5	8.891e-032	3000	5.062e-020
$\sin(10 \cot(x)) \sin(2x)$	10	4.548e-031	3000	5.117e-020
$\sin(50 \cot(x)) \sin(2x)$	5	3.830e-031	3000	5.123e-020
$\sin(100 \cot(x)) \sin(2x)$	5	1.782e-031	3000	5.349e-020
$e^x \sin(5x)$	5	5.402e-031	3000	5.114e-020
$e^x \sin(10x)$	5	6.686e-032	3000	5.212e-020
$e^x \sin(50x)$	5	3.789e-031	3000	5.272e-020
$e^x \sin(100x)$	5	1.350e-031	3000	5.368e-020
$\sin(5x) \cos(5x)$	5	1.525e-031	3000	5.3158e-020
$\sin(5x) \cos(10x)$	5	2.666e-031	3000	5.185e-020
$\sin(5x) \cos(50x)$	5	4.723e-031	3000	5.414e-020
$\sin(10x) \cos(50x)$	5	2.877e-031	3000	5.014e-020
$\sin(10x) \cos(100x)$	5	1.956e-031	3000	4.918e-020
$(1 - \cos(x))^5 \sin(5x)$	5	6.793e-031	3000	5.300e-020
$(1 - \cos(x))^{10} \sin(10x)$	5	3.149e-031	3000	5.334e-020

Tabla 5-1. Aproximación de las funciones con el Método de Newton

De la Tabla 5-1, se observa que el error en la aproximación de la función está entre $1e-31$ a $1e-20$, resumiendo para un número de nodos (n) pequeño el error es pequeño y conforme se aumenta el número de nodos (n) el error aumenta en la aproximación de la función.

En la siguiente tabla, se comparan las funciones con todos los algoritmos, con un número de nodos fijos ($n=1000$), donde se varía el argumento ($m=5$, $m=10$, $m=50$, $m=100$), es decir, haciendo las funciones más oscilante, con el número de nodos $n=1000$, y una tolerancia fija de $1e-10$.

Función	Back-Propagation		Factor Momentum		Gradientes Conjugados		Gradientes Conjugados Residual		Newton		Newton Truncado	
	k	Aprox. Función	k	Aprox. Función	k	Aprox. Función	k	Aprox. Función	k	Aprox. Función	k	Aprox. Función
$\sin(5x)$	12268	9.98e-11	4512	9.97e-11	18	7.36e-11	4	6.62e-22	1	4.96e-20	2	7.04e-22
$\sin(10x)$	12266	9.97e-11	4526	9.93e-11	18	7.88e-11	4	6.79e-22	1	4.82e-20	2	6.95e-22
$\sin(50x)$	12266	9.98e-11	4522	9.97e-11	18	7.25e-11	4	6.32e-22	1	5.41e-20	2	7.06e-22
$\sin(100x)$	12262	9.98e-11	4521	9.93e-11	18	7.63e-11	4	6.86e-22	1	5.03e-20	2	7.35e-22
$x^2 \sin(5x)$	15044	9.99e-11	5606	9.92e-11	18	1.30e-11	4	9.73e-21	1	5.73e-20	2	9.89e-21
$x^2 \sin(10x)$	15045	9.98e-11	5606	9.93e-11	18	1.16e-11	4	1.14e-20	1	6.08e-20	2	1.20e-20
$x^2 \sin(50x)$	15033	9.99e-11	5601	9.95e-11	18	9.64e-12	4	7.21e-20	1	3.18e-19	2	7.16e-20
$x^2 \sin(100x)$	15033	9.99e-11	5601	9.96e-11	18	1.50e-11	4	2.55e-19	1	8.38e-19	2	2.53e-19
$\sin(5 \cot(x)) \sin(2x)$	12277	9.99e-11	4529	9.96e-11	18	7.57e-11	4	6.63e-22	1	5.06e-20	2	6.63e-22
$\sin(10 \cot(x)) \sin(2x)$	12267	9.97e-11	4525	9.94e-11	18	7.35e-11	4	6.18e-22	1	5.11e-20	2	6.39e-22
$\sin(50 \cot(x)) \sin(2x)$	12273	9.99e-11	4532	9.96e-11	18	8.25e-11	4	6.62e-22	1	5.12e-20	2	6.77e-22
$\sin(100 \cot(x)) \sin(2x)$	12252	9.98e-11	4522	9.96e-11	18	7.80e-11	4	5.86e-22	1	5.34e-20	2	6.70e-22
$e^x \sin(5x)$	12519	9.987e-11	6325	9.97e-11	18	7.42e-11	4	6.43e-22	1	5.11e-20	2	6.02e-22
$e^x \sin(10x)$	12507	9.99e-11	6259	9.95e-11	18	6.43e-11	4	6.63e-22	1	5.21e-20	2	6.45e-22
$e^x \sin(50x)$	12505	9.99e-11	4626	9.93e-11	18	5.84e-11	4	6.44e-22	1	5.27e-20	2	6.87e-22
$e^x \sin(100x)$	12508	9.98e-11	6249	9.98e-11	18	5.82e-11	4	6.44e-22	1	5.36e-20	2	7.11e-22
$\sin(5x) \cos(5x)$	12261	9.99e-11	6121	9.98e-11	18	7.61e-11	4	6.88e-22	1	5.31e-20	2	6.82e-22
$\sin(5x) \cos(10x)$	12262	9.99e-11	6139	9.97e-11	18	6.48e-11	4	6.619e-22	1	5.18e-20	2	6.64e-22
$\sin(5x) \cos(50x)$	12267	9.98e-11	6131	9.97e-11	18	7.80e-11	4	6.52e-22	1	5.41e-20	2	6.71e-22
$\sin(10x) \cos(50x)$	12261	9.99e-11	6124	9.96e-11	18	8.21e-11	4	7.18e-22	1	5.01e-20	2	6.79e-22
$\sin(10x) \cos(100x)$	12285	9.97e-11	6137	9.97e-11	18	7.54e-11	4	6.04e-22	1	4.91e-20	2	7.22e-22
$(1 - \cos(x))^5 \sin(5x)$	12718	9.98e-11	6337	9.97e-11	18	4.86e-11	4	6.63e-22	1	5.30e-20	2	5.08e-20
$(1 - \cos(x))^{10} \sin(10x)$	15086	9.97e-11	7499	9.94e-11	18	8.03e-11	4	6.58e-22	1	5.33e-20	2	2.92e-21

Tabla 5-2. Comparación de todas las Redes Neuronales para la aproximación de las Funciones

De la Tabla 5-2, se observa que el error en la aproximación de la función con una tolerancia fija y un número de nodos fijo, se tiene que en los primeros 3 algoritmos (RNA Back-Propagation, RNA Factor Momentum y RNA Gradientes Conjugados) se conserva esta tolerancia para el ajuste de la función, mientras que los algoritmos de RNA Gradientes Conjugados Residual, RNA Newton y RNA Newton Truncado se obtiene el mínimo error para las funciones. El número de nodos se fija de este tamaño dado que posteriormente nos da una mejor aproximación en la integral de la función. Se observa que para la función $F = (1 - \cos(x))^m \sin(mx)$ si el argumento m es grande ($m > 50$), existe un overflow y esto hace que no se pueda aproximar la función.

En las siguientes tablas se comparan la aproximación a la integral de las funciones con cada uno de los algoritmos, con un número de nodos $n=5$ hasta $n=3000$, de diez en diez, donde se varía el argumento ($m=5$, $m=10$, $m=50$, $m=100$), es decir, haciendo las funciones más oscilante conservando las diferencias que existen en el número de nodos para los ajustes de las funciones con singularidad y se tiene una tolerancia fija de $1e-10$, solo se reportan las integrales de las funciones donde los métodos con RNA mejoran las aproximaciones obtenidas por el Método de Simpson Adaptivo.

Funciones		$\int_0^{10} x^3 \sin(5x) dx$	$\int_0^{10} x^3 \sin(10x) dx$	$\int_0^{2\pi} (1 - \cos(x))^5 \sin(5x) dx$	$\int_0^{2\pi} (1 - \cos(x))^{10} \sin(10x) dx$
Integral Real		-195.65918	-87.69068	0	0
Error Simpson Adaptivo		1.681e-2	8.260e-3	7.771e-15	6.411e-13
Back-Propagation	N	1000	1000	1001	91
	K	15044	15045	12718	1341
	Tiempo (s)	746.608677	759.895155	645.964104	0.396049
	Error Integral	2.273e-2	6.234e-2	8.859e-14	1.151e-13
Momentum	N	1000	1000	1001	91
	K	7518	7519	12718	1341
	Tiempo (s)	547.404727	552.913711	645.964104	0.396049
	Error Integral	2.273e-2	6.234e-2	8.859e-14	1.151e-13
Gradientes Conjugados	λ	1	1	1000	100
	N	3000	3000	3000	3000
	K	18	18	19	18
	Tiempo (s)	6.827979	6.873922	6.977661	6.990940
Gradiente Conjugado Residual	Error Integral	1.242e-2	4.168e-4	5.761e-07	4.595e-09
	N	3000	3000	3000	3000
	K	4	4	4	4
	Tiempo (s)	7.851679	8.012166	8.152390	7.693463
Newton	Error Integral	1.242e-02	4.169e-4	2.739e-13	2.293e-13
	N	3000	3000	300	30
	K	1	1	1	1
	Tiempo (s)	417.562057	465.044473	2.047893	1.186237
Newton Truncado	Error Integral	1.242e-002	4.169e-4	6.580e-016	5.838e-014
	N	3000	3000	10	90
	K	2	2	2	2
	Tiempo (s)	7.631695	7.742893	0.037233	0.0620132
	Error Integral	1.2428e-02	4.169e-4	1.149e-15	7.790e-15

Tabla 5-3. Aproximación a la Integral Definida de las Funciones con las diferentes RNA

Funciones		$\int_0^{\frac{\pi}{2}} \sin(5 \cot(x)) \sin(2x) dx$	$\int_0^{\frac{\pi}{2}} \sin(10 \cot(x)) \sin(2x) dx$	$\int_0^{\frac{\pi}{2}} \sin(50 \cot(x)) \sin(2x) dx$	$\int_0^{\frac{\pi}{2}} \sin(100 \cot(x)) \sin(2x) dx$
Integral Real		5.29197e-2	7.13140e-4	1.514836e-20	5.843481e-42
Error Simpson Adaptivo		7.177e-5	2.480e-4	5.600e-4	1.998e-3
Back-Propagation	N	3	200	750	750
	K	12277	2248	9042	9060
	Tiempo (s)	604.052392	0.901774	217.652134	216.172727
	Error Integral	2.948e-4	8.942e-6	2.277e-3	8.754e-4
Momentum	N	60	1000	90	1000
	K	325	6129	492	6127
	Tiempo (s)	0.160450	878.928895	3.519536	842.499731
	Error Integral	1.267e-1	2.839e-2	1.423e-2	1.021e-2
Gradientes Conjugados	λ	1	1	1	1
	N	3000	3000	3000	3000
	K	19	19	19	19
	Tiempo (s)	6.979154	7.053597	7.149253	6.822643
Gradiente Conjugado Residual	Error Integral	1.069e-4	9.798e-5	1.619e-4	4.645e-4
	N	3000	3000	3000	3000
	K	4	4	4	4
	Tiempo (s)	7.697458	7.643403	7.844370	8.082599
Newton	Error Integral	1.069e-4	9.791e-5	1.620e-4	4.644e-4
	N	3000	3000	300	
	K	1	1	1	1
	Tiempo (s)	418.732004	422.344307	448.496313	461.996848
Newton Truncado	Error Integral	4.651e-5	5.503e-5	7.997e-6	5.585e-4
	N	3000	3000	3000	3000
	K	2	2	2	2
	Tiempo (s)	7.779573	7.966196	7.695761	7.866569
	Error Integral	1.069e-4	9.791e-5	1.620e-4	4.644e-4

Tabla 5-4. Aproximación a la Integral Definida de las Funciones con las diferentes RNA

En la tabla 5-3, se observa que la aproximación a la integral definida de algunas funciones es mejor que el Método de Simpson Adaptivo. En la tabla 5-4, se observa que la aproximación a la integral definida se mantiene entre $1e-4$ y $1e-5$, para algunas funciones muy oscilantes se mejoran los resultados obtenidos por el Método de Simpson.

5.2. Funciones con Singularidades

Al hacer las pruebas para las diferentes funciones que presentan una singularidad usando la RNA de Newton dado que solo se requiere un solo entrenamiento, donde se varía el argumento ($m=1$, $m=5$, $m=10$, $m=100$), es decir, haciendo la función más oscilante y variando el número nodos desde $n=5$ hasta $n=3001$, de diez en diez, con número de nodos par o impar. En la tabla 2 se reporta el mínimo y máximo error en la aproximación de la función.

Función	Aproximación a la Función			
	n	Menor Error	n	Mayor Error
$x \tan(x)$	5	2.246e-30	3001	1.095e-16
$\frac{\cos(3x)}{\cos(x)}$	5	3.107e-30	3001	6.284e-20
$\frac{\cos(11x)}{\cos(x)}$	5	2.526e-31	3001	6.307e-20
$\frac{\cos(21x)}{\cos(x)}$	5	3.266e-31	3001	6.546e-20
$\frac{\cos(201x)}{\cos(x)}$	5	2.526e-31	3001	6.805e-20
$\frac{\sin(3x)}{\cos(x)}$	5	3.063e-31	3001	6.404e-19

Tabla 5-5. Aproximación de las funciones con el Método de Newton

De la Tabla 5-5, se observa que el error en la aproximación de la función está entre $1e-31$ a $1e-20$, para un número de nodos impar (n), resumiendo para n pequeño el error es pequeño y conforme se aumenta el número de nodos (n) el error aumenta en la aproximación de la función. Estas funciones tienen una singularidad en $\frac{\pi}{2}$, donde si el número de nodos es par los errores a la aproximación de la función son muy grandes ($1e+6$), en este caso la RNA de Newton falla dado que la matriz de activación no es lo suficientemente definida positiva, por lo tanto, no se encuentra la solución del sistema, entonces, el algoritmo se cicla y al probar con el número de nodos impar se obtienen mejores resultados para el ajuste de la función.

En la siguiente tabla, se comparan las funciones con todos los algoritmos, con un número de nodos fijos ($n=1001$ para las funciones con singularidad), donde se varía el argumento para las funciones con singularidad también se varían sus argumentos ($m=1$, $m=5$, $m=10$, $m=100$), es decir, haciendo las funciones más oscilantes, con el número de nodos $n=1000$ y $n=1001$ para funciones con singularidad, y una tolerancia fija de $1e-10$.

Función	Back-Propagation		Factor Momentum		Gradientes Conjugados		Gradientes Conjugados Residual		Newton		Newton Truncado	
	k	Aprox. Función	k	Aprox. Función	k	Aprox. Función	k	Aprox. Función	k	Aprox. Función	k	Aprox. Función
$x \tan(x)$	13804	9.977e-11	6898	9.961e-11	18	7.634e-11	4	2.894e-21	1	1.095e-16	2	8.783e-21
$\frac{\cos(3x)}{\cos(x)}$	12296	9.996e-11	6140	9.982e-11	18	7.325e-11	4	6.361e-22	1	6.284e-20	2	6.796e-22
$\frac{\cos(11x)}{\cos(x)}$	12351	9.992e-11	6180	9.972e-11	18	5.905e-11	4	6.789e-22	1	6.307e-20	2	7.147e-22
$\frac{\cos(21x)}{\cos(x)}$	12422	9.976e-11	6222	9.957e-11	18	6.501e-11	4	6.547e-22	1	6.546e-20	2	6.507e-22
$\frac{\cos(201x)}{\cos(x)}$	12936	9.974e-11	6466	9.982e-11	18	7.359e-11	4	7.379e-22	1	6.805e-20	2	7.690e-22
$\frac{\sin(3x)}{\cos(x)}$	13476	9.984e-11	6732	9.986e-11	18	2.968e-11	4	3.737e-21	1	6.404e-19	2	6.507e-22

Tabla 5-6. Comparación de todas las Redes Neuronales para la aproximación de las Funciones

De la Tabla 5-6, se observa que el error en la aproximación de la función con una tolerancia fija y un número de nodos fijo, se tiene que en los primeros 3 algoritmos (RNA Back-Propagation, RNA Factor Momentum y RNA Gradientes Conjugados) se conserva esta tolerancia para el ajuste de la función, mientras que los algoritmos de RNA Gradientes Conjugados Residual, RNA Newton y RNA Newton Truncado se obtiene el mínimo error para las funciones, a pesar que las funciones presentan una singularidad.

En las siguientes tablas se comparan la aproximación a la integral de las funciones con cada uno de los algoritmos, con un número de nodos $n=5$ hasta $n=3000$, de diez en diez, (nodos impares con $n=5$ hasta $n=3001$ para las funciones con singularidad), donde se varía el argumento ($m=5, m=10, m=50, m=100$) y para las funciones con singularidad también se varían los argumentos ($m=1, m=5, m=10, m=100$), es decir, haciendo las funciones más oscilante conservando las diferencias que existen en el número de nodos para los ajustes de las funciones con singularidad y se tiene una tolerancia fija de $1e-10$, solo se reportan las integrales de las funciones donde los métodos con RNA mejoran las aproximaciones obtenidas por el Método de Simpson Adaptivo.

Funciones		$\int_0^{\pi} x \tan(x) dx$	$\int_0^{\pi} \frac{\sin(3x)}{\cos(x)} dx$
Integral Real		-2.177586090303602	0
Error Simpson Adaptivo		9.663191186362639	6.151779252201354
Back-Propagation	N	1001	1001
	K	13804	3819
	Tiempo (s)	506.358620	5.052069
	Error Integral	2.578e-06	1.837e-14
Momentum	N	1000	301
	K	6898	1906
	Tiempo (s)	329.886692	3.121622
	Error Integral	2.578e-06	1.291e-14
Gradientes Conjugados	λ	1	1
	N	3001	3001
	K	18	19
	Tiempo (s)	6.874513	6.923010
Gradiente Conjugado Residual	Error Integral	3.340e-07	5.981e-08
	N	3001	5
	K	4	4
	Tiempo (s)	7.910823	0.029341
Newton	Error Integral	2.869e-07	1.607e-16
	N	31	5
	K	1	1
	Tiempo (s)	4.293082	4.292185
Newton Truncado	Error Integral	4.44e-016	3.487e-016
	N	3001	11
	K	2	2
	Tiempo (s)	7.872709	0.032787
Newton Truncado	Error Integral	2.869e-07	6.586e-16

Tabla 5-7. Aproximación a la Integral Definida de las Funciones con las diferentes RNA

En la tabla 5-7, las funciones tienen singularidad el Método de Simpson falla y en todos los algoritmos se obtiene un error entre $1e-8$ y $1e-15$.

Funciones		$\int_0^{\pi} \frac{\cos(3x)}{\cos(x)} dx$	$\int_0^{\pi} \frac{\cos(11x)}{\cos(x)} dx$	$\int_0^{\pi} \frac{\cos(21x)}{\cos(x)} dx$	$\int_0^{\pi} \frac{\cos(201x)}{\cos(x)} dx$
Integral Real		-3.1415926535897	-3.141592653589793	3.14159265358979	3.1415926535897
Error Simpson Adaptivo		0	1.7674e-013	1.4805e-012	1.5107e-008
Back-Propagation	N	1001	1001	1001	1001
	K	12296	12351	12422	12936
	Tiempo (s)	457.744996	419.042243	500.849788	510.196471
	Error Integral	1.327e-13	1.301e-13	1.030e-13	1.052e-13
Momentum	N	1001	1001	1001	301
	K	6140	6180	6222	1888
	Tiempo (s)	214.875383	219.364123	295.756086	2.600673
	Error Integral	9.681e-14	8.704e-14	8.482e-14	5.195e-14
Gradientes Conjugados	λ	1	1	1	1
	N	3001	3001	3001	3001
	K	19	19	19	19
	Tiempo (s)	6.954365	7.040782	7.051032	7.111690
Gradiente Conjugado Residual	Error Integral	1.496e-07	1.973e-07	2.34106e-07	5.6713e-08
	N	3001	51	61	5
	K	4	3	4	4
	Tiempo (s)	7.891616	0.032703	0.045400	0.027691
Newton	Error Integral	1.807e-13	8.881e-16	4.440e-16	3.996e-15
	N	51	81	201	3001
	K	1	1	1	1
	Tiempo (s)	12.872087	11.914467	11.650442	448.048384
Newton Truncado	Error Integral	0	0	1.554e-014	2.575e-014
	N	61	11	91	501
	K	2	2	2	2
	Tiempo (s)	0.043965	0.035097	0.052529	0.236734
	Error Integral	0	4.440e-16	8.881e-16	1.332e-15

Tabla 5-8. Aproximación a la Integral Definida de las Funciones con las diferentes RNA

En la tabla 5-8, se observa que la aproximación a la integral definida se mantiene entre un error de $1e-7$ y cero, cuando la función es más oscilante se mejora Método de Simpson Adaptivo, donde este crece su error en la aproximación a la integral.

Las siguientes funciones presentan más de una singularidad y esto hace que los algoritmos fallen al calcular la integral definida y se tenga un error numérico grande.

- $F = x \cot^2(mx)$.
- $F = x \sec^2(mx)$.
- $F = \frac{1}{\cos(mx)}$.

5.3. Factores Franck-Condon

Se presentan los resultados obtenidos por las RNA Newton, debido a que los resultados en el cálculo de la integral de la función de onda de los FFC de Morse en las demás RNAS son equivalentes. Se muestra la precisión de los resultados generados en comparación con los valores experimentales. El intervalo de integración es de $[0.4, 3.5]$, para todas las RNAS.

5.3.1. Moléculas Diatómicas

Se presentan un conjunto de FFC calculados para un rango de números cuánticos del 0 al 9. Así mismo se muestra el algoritmo empleado y el tiempo de cálculo.

Sistema de bandas $B^3\Pi_g \rightarrow A^3\Sigma_u^+$ del $^{15}N_2$

	$B^3\Pi_g$				$A^3\Sigma_u^+$					
w_e	1675.355				1411.210					
$w_e x_e$	13.433				12.921					
r_e	1.21252				1.2864					
μ	7.500053859									
$v_2 v_1$	0	1	2	3	4	5	6	7	8	9
0	0.3892	0.4003	0.1680	0.0373	0.0048	0.0004	0.0000	0.0000	0.0000	0.0000
1	0.3308	0.0012	0.2617	0.2812	0.1043	0.0188	0.0018	0.0001	0.0000	0.0000
2	0.1710	0.1499	0.0776	0.0817	0.2918	0.1718	0.0443	0.0054	0.0003	0.0000
3	0.0704	0.1936	0.0166	0.1563	0.0034	0.2279	0.2366	0.0801	0.0121	0.0009
4	0.0257	0.1339	0.1179	0.0086	0.1460	0.0168	0.1370	0.2663	0.1229	0.0230
5	0.0087	0.0690	0.1427	0.0358	0.0587	0.0848	0.0688	0.0577	0.2636	0.1675
6	0.0028	0.0304	0.1041	0.1035	0.0007	0.0995	0.0266	0.1137	0.0110	0.2328
7	0.0009	0.0123	0.0592	0.1121	0.0491	0.0127	0.1029	0.0007	0.1287	0.0006
8	0.0003	0.0047	0.0292	0.0829	0.0915	0.0103	0.0455	0.0750	0.0092	0.1129
9	0.0001	0.0017	0.0132	0.0499	0.0910	0.0553	0.0003	0.0719	0.0369	0.0374

Tabla 5-9. Datos Experimentales Sistema de bandas $B^3\Pi_g \rightarrow A^3\Sigma_u^+$ del $^{15}N_2$

Los datos obtenidos con las Red Neuronal Newton son los siguientes:

$v_2 v_1$	0	1	2	3	4	5	6	7	8	9
0	3.899e-1	4.004e-1	1.675e-1	3.700e-2	4.674e-3	3.421e-4	1.403e-5	2.950e-7	2.614e-9	5.779e-12
1	3.310e-1	1.290e-3	2.630e-1	2.807e-1	1.034e-1	1.851e-2	1.761e-3	8.900e-5	2.218e-6	2.258e-8
2	1.707e-1	1.510e-1	7.651e-2	8.336e-2	2.923e-1	1.767e-1	4.359e-2	5.267e-3	3.218e-4	9.374e-6
3	7.014e-2	1.960e-1	1.741e-2	1.556e-1	3.878e-3	2.297e-1	2.353e-1	7.901e-2	1.194e-2	8.711e-4
4	2.544e-2	1.335e-1	1.194e-1	7.861e-3	1.469e-1	1.576e-2	1.393e-1	2.655e-1	1.213e-1	2.274e-2
5	8.586e-3	6.844e-2	1.432e-1	3.733e-2	5.712e-2	8.687e-2	6.710e-2	5.973e-2	2.634e-1	1.656e-1
6	2.780e-3	3.001e-2	1.037e-1	1.051e-1	1.055e-3	9.850e-2	2.837e-2	1.125e-1	1.197e-2	2.333e-1
7	8.806e-4	1.200e-2	5.858e-2	1.126e-1	5.128e-2	1.144e-2	1.035e-1	1.126e-3	1.290e-1	3.598e-4
8	2.764e-4	4.535e-3	2.861e-2	8.249e-2	9.328e-2	1.175e-2	4.353e-2	7.718e-2	8.011e-3	1.147e-1
9	8.675e-5	1.657e-3	1.277e-2	4.914e-2	9.145e-2	5.785e-2	9.345e-5	7.064e-2	3.956e-2	3.527e-2

Tabla 5-10. FFC-RNA Newton Sistema de bandas $B^3\Pi_g \rightarrow A^3\Sigma_u^+$ del $^{15}N_2$

Tiempo de ejecución: 39.485115 s

Sistema de bandas $B'^3\Sigma_u^- \rightarrow B^3\Pi_g$ del $^{15}\text{N}_2$

Los datos experimentales con los que se hace la comparación del sistema de bandas de esta molécula (Abad, 1991), son los siguientes:

	$B'^3\Sigma_u^-$	$B^3\Pi_g$
w_e	1465.570	1675.355
$w_e x_e$	11.323	13.433
r_e	1.278387	1.21252
μ	7.500053859	

$v_2 v_1$	0	1	2	3	4	5	6
0	0.4691	0.3225	0.1387	0.0484	0.0151	0.0044	0.0013
1	0.3868	0.285	0.2136	0.1950	0.1057	0.0449	0.1674
2	0.1234	0.3442	0.0225	0.0717	0.1638	0.1966	0.0775
3	0.0192	0.2401	0.1954	0.1043	0.0535	0.0968	0.1317
4	0.0015	0.0582	0.3042	0.00729	0.1530	0.0785	0.0364
5	0.0000	0.0061	0.1098	0.3129	0.0110	0.01506	0.0427
6	0.0000	0.0003	0.0190	0.1650	0.2810	0.0015	0.1150

Tabla 5-11. Datos Experimentales Sistema de bandas $B'^3\Sigma_u^- \rightarrow B^3\Pi_g$ del $^{15}\text{N}_2$

Los datos obtenidos con las Red Neuronal Newton son los siguientes:

$v_2 v_1$	0	1	2	3	4	5	6	7	8	9
0	4.713e-1	3.233e-1	1.378e-1	4.735e-2	1.443e-2	4.099e-3	1.114e-3	2.948e-4	7.686e-5	1.988e-5
1	3.859e-1	3.038e-2	2.176e-1	1.956e-1	1.041e-1	4.322e-2	1.558e-2	5.146e-3	1.606e-3	4.837e-4
2	1.223e-1	3.458e-1	2.017e-2	7.678e-2	1.679e-1	1.367e-1	7.555e-2	3.388e-2	1.337e-2	4.863e-3
3	1.881e-2	2.376e-1	2.005e-1	9.938e-2	7.545e-3	1.033e-1	1.353e-1	9.937e-2	5.491e-2	2.565e-2
4	1.433e-3	5.668e-2	3.024e-1	7.867e-2	1.496e-1	5.260e-3	4.332e-2	1.081e-1	1.075e-1	7.318e-2
5	4.816e-5	5.787e-3	1.066e-1	3.138e-1	1.410e-2	1.512e-1	3.631e-2	8.239e-3	6.992e-2	9.954e-2
6	4.652e-7	2.376e-4	1.404e-2	1.602e-1	2.854e-1	5.077e-4	1.196e-1	7.109e-2	3.007e-4	3.453e-2
7	4.33e-12	2.524e-6	6.843e-4	2.656e-2	2.107e-1	2.345e-1	1.892e-2	7.648e-2	9.298e-2	1.174e-2
8	3.12e-11	4.27e-10	7.765e-6	1.503e-3	4.318e-2	2.531e-1	1.757e-1	5.072e-2	3.772e-2	9.715e-2
9	1.21e-13	2.38e-10	4.995e-9	1.774e-5	2.787e-3	6.332e2	2.850e-1	1.197e-1	8.229e-2	1.181e-2

Tabla 5-12. FFC- RNA Newton Sistema de bandas $B'^3\Sigma_u^- \rightarrow B^3\Pi_g$ del $^{15}\text{N}_2$

Tiempo de ejecución: 28.379652 s

Sistema de bandas $A^1\Sigma_u^+ \rightarrow X^1\Sigma_g^+$ del ${}^7\text{Li}_2$

Los datos experimentales con los que se hace la comparación del sistema de bandas de esta molécula (Abad, 1991), son los siguientes:

	$A^1\Sigma_u^+$	$X^1\Sigma_g^+$
w_e	255.45	351.43
$w_e x_e$	1.574	2.592
r_e	3.107	2.672
μ	3.50908	

$v_1 v_2$	A	B	C	D	E
0 0	0.0527	0.0527	0.052	0.053	0.053
0 1	0.1789	0.1789	0.176	0.131	0.180
0 2	0.276	0.2762	0.270	0.182	0.278
0 3	0.2543	0.2535	0.250	0.187	0.254
0 4	0.1541	0.1543	0.156	0.158	0.153
1 0	0.130	0.130	0.134	0.18	0.131
1 1	0.1915	0.1917	0.197	0.191	0.191
1 2	0.055	0.055	0.058	0.078	0.054
1 3	0.0114	0.0115	0.009	0.004	0.012
1 4	0.144	0.1506	0.134	0.025	0.145
2 0	0.1816	0.1818	0.187	0.277	0.182
2 1	0.0792	0.0791	0.079	0.054	0.078
2 2	0.0128	0.0128	0.015	0.013	0.013
2 3	0.1194	0.1255	0.127	0.009	0.120
2 4	0.0522	0.0541	0.056	0.088	0.051
3 0	0.1874	0.187	0.190	0.254	0.188
3 1	0.0047	0.0048	0.003	0.012	0.004
3 2	0.0895	0.0941	0.098	0.120	0.090
3 3	0.0465	0.0472	0.045	0.046	0.046
3 4	0.0191	0.0093	0.025	0.002	0.020
4 0	0.1586	0.1612	0.157	0.153	0.158
4 1	0.0147	0.0153	0.018	0.145	0.015
4 2	0.0891	0.0924	0.090	0.051	0.089
4 3	0.0015	0.0013	0.004	0.0220	0.002
4 4	0.0842	0.747	0.092	0.084	0.084

Tabla 5-13. Datos Experimentales Sistema de bandas $A^1\Sigma_u^+ \rightarrow X^1\Sigma_g^+$ del ${}^7\text{Li}_2$

Los datos obtenidos con la Red Neuronal Newton son los siguientes:

$v_2 v_1$	0	1	2	3	4	5	6	7	8	9
0	5.419e-2	1.400e-1	1.959e-1	1.970e-1	1.595e-1	1.104e-1	6.791e-2	3.799e-2	1.968e-2	9.568e-3
1	1.795e-1	1.976e-1	7.332e-2	1.337e-3	2.656e-2	8.271e-2	1.137e-1	1.097e-1	8.552e-2	5.753e-2
2	2.724e-1	5.387e-2	1.965e-2	1.039e-1	8.475e-2	1.831e-2	2.110e-3	3.589e-2	7.448e-2	9.040e-2
3	2.493e-1	1.186e-2	1.306e-1	3.640e-2	8.364e-3	6.999e-2	7.367e-2	2.494e-2	5.535e-6	1.818e-2
4	1.522e-1	1.382e-1	5.158e-2	3.461e-2	9.074e-2	1.953e-2	8.041e-3	5.654e-2	6.315e-2	2.454e-2
5	6.439e-2	2.032e-1	8.426e-3	1.238e-1	2.673e-3	5.405e-2	6.089e-2	7.987e-3	9.847e-3	5.026e-2
6	1.835e-2	1.361e-1	1.075e-1	3.324e-2	8.549e-2	5.891e-2	7.702e-3	6.043e-2	3.458e-2	1.757e-3
7	2.779e-3	4.484e-2	1.200e-1	8.338e-3	1.130e-1	7.280e-3	1.138e-1	1.040e-2	3.565e-2	5.441e-2
8	7.964e-6	3.293e-3	3.590e-2	4.281e-2	1.138e-2	8.008e-2	2.315e-2	6.118e-2	7.932e-2	7.188e-4
9	4.612e-4	2.178e-3	8.098e-5	1.289e-2	1.123e-2	2.119e-2	1.939e-2	5.259e-2	2.531e-3	8.148e-2

Tabla 5-14. FFC-RNA Newton Sistema de bandas $A^1\Sigma_u^+ \rightarrow X^1\Sigma_g^+$ del ${}^7\text{Li}_2$

Tiempo de ejecución: 26.357455 s

Sistema de bandas $B^1\Pi_u \rightarrow X^1\Sigma_g^+$ del ${}^7Li_2$

Los datos experimentales con los que se hace la comparación del sistema de bandas de esta molécula (Abad, 1991), son los siguientes:

	$B^1\Pi_u$	$X^1\Sigma_g^+$
w_e	269.69	351.43
$w_e x_e$	2.744	2.592
r_e	2.936	2.672
μ	3.50908	

$v_1 v_2$	A	B	C	D	E
0 0	0.3256	0.3256	0.3188	0.3267	0.3267
0 1	0.4102	0.410	0.3827	0.3149	0.4104
0 2	0.2073	0.2076	0.2103	0.1961	0.2065
0 3	0.0512	0.0514	0.0698	0.0969	0.0507
0 4	0.0056	0.0057	0.0156	0.0413	0.0055
1 0	0.3149	0.3151	0.03340	0.4104	0.3149
1 1	0.0041	0.0041	0.0077	0.0039	0.0039
1 2	0.2026	0.2036	0.1511	0.0844	0.2042
1 3	3.1274	0.3135	0.2711	0.1692	0.3127
1 4	0.1404	0.1409	0.1657	0.1516	0.1395
2 0	0.1965	0.1967	0.2008	0.2065	0.1961
2 1	0.0836	0.0836	0.0942	0.2042	0.0844
2 2	0.1118	0.1121	0.1345	0.1110	0.1110
2 3	0.0319	0.0320	0.0063	0.00005	0.0329
2 4	0.2813	0.2826	0.1834	0.0622	0.2826
3 0	0.0973	0.0975	0.0918	0.0507	0.0976
3 1	0.1688	0.1693	0.1884	0.3127	0.1692
3 2	0.0000183	0.0000181	0.0001	0.0329	0.00005
3 3	0.1390	0.1396	0.1508	0.1391	0.1391
3 4	0.0032	0.0032	0.0303	0.0423	0.0028
4 0	0.0407	0.0417	0.0358	0.0055	0.0413
4 1	0.1537	0.1523	0.1585	0.1395	0.1516
4 2	0.0614	0.0615	0.0711	0.2826	0.0622
4 3	0.0430	0.0430	0.550	0.0028	0.0423
4 4	0.0792	0.0803	0.0661	0.0800	0.0800

Tabla 5-15. Datos Experimentales Sistema de bandas $B^1\Pi_u \rightarrow X^1\Sigma_g^+$ del ${}^7Li_2$

Los datos obtenidos con la Red Neuronal Newton son los siguientes:

$v_2 v_1$	0	1	2	3	4	5	6	7	8	9
0	3.182e-1	3.345e-1	2.014e-1	9.194e-2	3.559e-2	1.239e-2	4.025e-3	1.247e-3	3.756e-4	1.111e-4
1	3.834e-1	7.715e-3	9.525e-2	1.902e-1	1.591e-1	9.215e-2	4.344e-2	1.800e-2	6.842e-3	2.456e-3
2	2.105e-1	1.524e-1	1.341e-1	1.425e-5	7.480e-2	1.416e-1	1.274e-1	8.165e-2	4.310e-2	2.004e-2
3	6.956e-2	2.719e-1	7.142e-3	1.541e-1	5.134e-2	3.915e-3	7.095e-2	1.164e-1	1.050e-1	7.106e-2
4	1.538e-2	1.641e-1	1.859e-1	2.877e-2	7.559e-2	1.046e-1	1.550e-2	1.153e-2	6.920e-2	1.001e-1
5	2.379e-3	5.389e-2	2.182e-1	6.260e-2	1.107e-1	1.004e-2	9.734e-2	5.662e-2	2.890e-3	1.768e-2
6	2.455e-4	1.064e-2	1.014e-1	1.866e-1	3.024e-4	1.567e-1	8.209e-3	5.810e-2	7.660e-2	2.500e-2
7	8.527e-6	9.744e-4	2.131e-2	1.134e-1	8.148e-2	4.373e-2	1.030e-1	8.048e-2	1.180e-2	8.519e-2
8	3.574e-6	1.167e-5	4.677e-4	1.667e-2	6.567e-2	8.350e-3	9.268e-2	1.091e-2	1.348e-1	1.392e-2
9	1.268e-5	3.038e-4	1.944e-3	1.960e-3	2.988e-3	2.715e-2	1.620e-5	5.998e-2	5.422e-3	6.654e-2

Tabla 5-16. FFC-RNA Newton Sistema de bandas $B^1\Pi_u \rightarrow X^1\Sigma_g^+$ del ${}^7Li_2$

Tiempo de ejecución: 25.711498 s

Sistema de bandas $A^2\Pi \rightarrow X^2\Sigma^+$ del CN

Los datos experimentales con los que se hace la comparación del sistema de bandas de esta molécula (Abad, 1991), son los siguientes:

	$A^2\Pi$	$X^2\Sigma^+$
w_e	1814.43	2068.705
$w_e x_e$	12.883	13.144
r_e	1.2327	1.1718
μ	6.46427	

En la tabla de datos experimentales que se presenta a continuación, v'' se refiere al número cuántico vibracional del estado base y v' al estado excitado.

$v_2 v_1$	A	B	C	D	E
0 0	0.4953	0.4953	0.94	0.496	0.499
1 0	0.3241	0.3244	0.360	0.347	0.371
2 0	0.1285	0.1287	0.119	0.122	0.111
3 0	0.0392	0.0393	0.0234	0.0280	0.0174
0 1	0.3698	0.370	0.335	0.347	0.320
1 1	0.0442	0.0443	0.0426	0.0450	0.0456
2 1	0.2437	0.244	0.318	0.294	0.350
3 1	0.201	0.2018	0.222	0.215	0.223
0 2	0.1148	0.1150	0.126	0.122	0.126
1 2	0.3411	0.3422	0.264	0.294	0.240
2 2	0.0118	0.0118	0.0137	0.0120	0.0122
3 2	0.1025	0.1029	0.174	0.151	0.210
0 3	0.0186	0.0186	0.0349	0.0280	0.0399
1 3	0.2259	0.2267	0.207	0.215	0.195
2 3	0.2008	0.202	0.120	0.151	0.0989
3 3	0.0852	0.0857	0.0925	0.0887	0.0905

Tabla 5-17. Datos Experimentales Sistema de bandas $A^2\Pi \rightarrow X^2\Sigma^+$ del CN

Los datos obtenidos con la Red Neuronal Newton son los siguientes:

$v_2 v_1$	0	1	2	3	4	5	6	7	8	9
0	4.994e-1	3.207e-1	1.257e-1	3.937e-2	1.092e-2	2.827e-3	7.039e-4	1.717e-4	4.157e-5	1.007e-5
1	3.702e-1	4.628e-2	2.424e-1	1.951e-1	9.397e-2	3.540e-2	1.162e-2	3.513e-3	1.010e-3	2.820e-4
2	1.111e-1	3.492e-1	1.165e-2	1.013e-1	1.840e-1	1.337e-1	6.684e-2	2.727e-2	9.850e-3	3.295e-3
3	1.749e-2	2.221e-1	2.107e-1	8.882e-2	1.655e-2	1.259e-1	1.435e-1	9.474e-2	4.756e-2	2.032e-2
4	1.564e-3	5.447e-2	2.863e-1	8.419e-2	1.489e-1	1.651e-3	6.109e-2	1.243e-1	1.102e-1	6.801e-2
5	8.060e-5	6.663e-3	1.047e-1	2.952e-1	1.431e-2	1.584e-1	3.010e-2	1.605e-2	8.760e-2	1.092e-1
6	2.313e-6	4.367e-4	1.693e-2	1.591e-1	2.599e-1	1.151e-3	1.274e-1	6.985e-2	8.140e-5	4.791e-2
7	3.388e-8	1.523e-5	1.377e-3	3.328e-2	2.084e-1	1.990e-1	2.584e-2	7.910e-2	9.819e-2	9.032e-3
8	2.096e-10	2.626e-7	5.728e-5	3.299e-3	5.575e-2	2.455e-1	1.309e-1	6.578e-2	3.472e-2	1.054e-1
9	3.241e-13	1.862e-9	1.144e-6	1.612e-4	6.650e-3	8.351e-2	2.659e-1	7.054e-2	1.025e-1	7.149e-3

Tabla 5-18. FFC-RNA Newton Sistema de bandas $A^2\Pi \rightarrow X^2\Sigma^+$ del CN

Tiempo de ejecución: 26.298744 s

Sistema de bandas $B^1\Sigma^+ \rightarrow X^1\Sigma^+$ del $B^{11}H^1$

Los datos experimentales con los que se hace la comparación del sistema de bandas de esta molécula (Abad, 1991), son los siguientes:

	$B^1\Sigma^+$	$X^1\Sigma^+$
w_e	2400.0	2366.0
$w_e x_e$	65.0	49.0
r_e	1.2149	1.2325
μ	0.923585	

Los datos de la esta tabla, corresponden a datos calculados usando un programa llamado “TRAPRB”.

$v_2 v_1$	0	1	2	3
0	0.9936	0.0061	0.0002	0.0000
1	0.0059	0.9873	0.0058	0.0009
2	0.0004	0.0053	0.9899	0.0021
3	0.0000	0.0011		0.9936
4	0.0000			

$v_2 v_1$	0	1	2	3
0	0.9933	0.0061	0.0002	0.0000
1	0.0059	0.9869	0.0058	0.0009
2	0.0004	0.0053	0.9899	0.0021
3	0.0000	0.0011		0.9936
4	0.0000			

Tabla 5-19. Datos Experimentales Sistema de bandas $B^1\Sigma^+ \rightarrow X^1\Sigma^+$ del $B^{11}H^1$

Los datos obtenidos con la Red Neuronal Newton son los siguientes:

$v_2 v_1$	0	1	2	3	4	5	6	7	8	9
0	9.929e-1	6.826e-3	1.934e-4	9.823e-7	1.869e-6	4.596e-7	9.455e-8	2.181e-8	6.120e-9	2.081e-9
1	6.570e-3	9.847e-1	7.931e-3	7.604e-4	1.404e-7	6.340e-6	2.697e-6	7.595e-7	2.131e-7	6.726e-8
2	4.436e-4	7.289e-3	9.854e-1	5.065e-3	1.754e-3	2.144e-5	8.757e-6	7.900e-6	3.181e-6	1.121e-6
3	7.894e-6	1.126e-3	4.457e-3	9.901e-1	1.105e-3	2.927e-3	1.499e-4	2.761e-6	1.344e-5	8.546e-6
4	1.143e-7	2.784e-5	1.947e-3	9.430e-4	9.923e-1	5.613e-4	3.634e-3	5.175e-4	6.224e-6	1.053e-5
5	4.154e-7	1.455e-7	5.773e-5	2.932e-3	4.385e-4	9.824e-1	9.782e-3	3.048e-3	1.159e-3	1.036e-4
6	2.180e-7	1.489e-6	3.602e-8	8.908e-5	4.253e-3	7.619e-3	9.484e-1	3.625e-2	1.074e-3	1.739e-3
7	8.095e-8	9.988e-7	3.009e-6	4.376e-8	1.084e-4	6.312e-3	2.797e-2	8.765e-1	8.605e-2	1.704e-4
8	2.676e-8	4.457e-7	2.546e-6	4.463e-6	5.751e-7	1.005e-4	9.917e-3	6.686e-2	7.548e-1	1.585e-1
9	8.514e-9	1.732e-7	1.344e-6	4.736e-6	5.350e-6	1.960e-6	5.686e-5	1.664e-2	1.272e-1	5.798e-1

Tabla 5-20. FFC-RNA Newton Sistema de bandas $B^1\Sigma^+ \rightarrow X^1\Sigma^+$ del $B^{11}H^1$

Tiempo de ejecución: 27.411383 s

Sistema de bandas $A^1\Sigma^+ \rightarrow X^1\Sigma^+$ del AgH^1

Los datos experimentales con los que se hace la comparación del sistema de bandas de esta molécula (Abad, 1991), son los siguientes:

	$A^1\Sigma^+$	$X^1\Sigma^+$
w_e	1663.6	1760.0
$w_e x_e$	87.0	34.05
r_e	1.641	1.617
μ	0.998800	

Datos Experimentales obtenidos con el Método de Simpson

$v_2 v_1$	0	1	2	3	4	5	6	7	8	9
0	0.9633	0.0343	0.0016	0.0003	0.0002	0.0001	0.0000	0.0000	0.0000	0.0000
1	0.0358	0.8260	0.1203	0.0131	0.0023	0.0008	0.0004	0.0002	0.0001	0.0000
2	0.0005	0.1303	0.5536	0.2354	0.0550	0.0134	0.0042	0.0016	0.0006	0.0000
3	0.0003	0.0074	0.2739	0.2094	0.2710	0.1273	0.0499	0.0196	0.0071	0.0004
4	0.0000	0.0020	0.0379	0.3693	0.0051	0.1323	0.1314	0.0793	0.0354	0.0019
5	0.0000	0.0000	0.0101	0.1227	0.2675	0.0830	0.0005	0.0215	0.0206	0.0014
6	0.0000	0.0000	0.0004	0.0406	0.2082	0.0335	0.1440	0.0752	0.0215	0.0009
7	0.0000	0.0000	0.0001	0.0041	0.0952	0.1090	0.0481	0.0005	0.0112	0.0010
8	0.0000	0.0000	0.0000	0.0001	0.0076	0.0630	0.0002	0.0208	0.0188	0.0011
9	0.0000	0.0000	0.0000	0.0004	0.0028	0.0000	0.0102	0.0063	0.0010	0.0000

Tabla 5-21. Datos Experimentales Sistema de bandas $A^1\Sigma^+ \rightarrow X^1\Sigma^+$ del AgH^1

Los datos obtenidos con la Red Neuronal Newton son los siguientes:

$v_2 v_1$	0	1	2	3	4	5	6	7	8	9
0	9.633e-1	3.428e-2	1.636e-3	3.182e-4	1.535e-4	8.689e-5	4.866e-5	2.592e-5	1.147e-5	6.232e-7
1	3.581e-2	8.259e-1	1.202e-1	1.313e-2	2.290e-3	7.718e-4	3.800e-4	2.039e-4	9.291e-5	5.113e-6
2	5.327e-4	1.303e-1	5.535e-1	2.353e-1	5.495e-2	1.341e-2	4.184e-3	1.621e-3	6.235e-4	3.252e-5
3	3.008e-4	7.377e-3	2.739e-1	2.094e-1	2.708e-1	1.275e-1	4.985e-2	1.951e-2	7.117e-3	3.607e-4
4	5.375e-6	1.973e-3	3.989e-2	3.695e-1	5.262e-3	1.309e-1	1.319e-1	7.983e-2	3.559e-2	1.918e-3
5	2.344e-6	3.318e-6	1.013e-2	1.234e-1	2.720e-1	7.794e-2	3.554e-4	1.999e-2	1.985e-2	1.327e-3
6	6.115e-8	1.318e-5	4.486e-4	4.215e-2	2.235e-1	4.724e-2	1.321e-1	6.506e-2	1.892e-2	8.064e-4
7	2.471e-9	1.359e-8	9.782e-5	5.558e-3	1.265e-1	1.862e-1	2.737e-2	5.945e-3	1.664e-2	1.278e-3
8	1.710e-8	7.741e-9	2.148e-6	9.492e-4	3.475e-2	2.265e-1	1.691e-2	6.680e-2	3.062e-2	1.469e-3
9	1.440e-8	7.703e-8	1.503e-8	7.750e-5	8.150e-3	1.244e-1	1.516e-1	5.276e-2	2.727e-3	6.194e-4

Tabla 5-22. FFC-RNA Newton Sistema de bandas $A^1\Sigma^+ \rightarrow X^1\Sigma^+$ del AgH^1

Tiempo de ejecución: 28.233771 s

Sistema de bandas $B^1\Sigma_u^+ \rightarrow X^1\Sigma_g^+$ del H^1H^2

Los datos experimentales con los que se hace la comparación del sistema de bandas de esta molécula (Abad, 1991), son los siguientes:

	$B^1\Sigma_u^+$	$X^1\Sigma_g^+$
w_e	1179.2	3817.09
$w_e x_e$	16.091	94.958
r_e	1.2910	0.7413
μ	0.671917	

Datos Experimentales obtenidos con el Método de Simpson

$v_2 v_1$	0	1	2	3	4	5	6	7	8	9
0	0.0032	0.0162	0.0432	0.0800	0.1157	0.1389	0.1441	0.1322	0.1092	0.0822
1	0.0192	0.0653	0.1094	0.1155	0.0800	0.0317	0.0026	0.0053	0.0313	0.0628
2	0.0616	0.1280	0.1118	0.0417	0.00110	0.0146	0.01449	0.0530	0.0346	0.0103
3	0.1324	0.1355	0.0304	0.0039	0.0479	0.0614	0.0296	0.0021	0.0056	0.0241
4	0.2040	0.0604	0.0074	0.0696	0.0515	0.0041	0.0110	0.0399	0.0402	0.0171
5	0.2303	0.0001	0.0850	0.0498	0.0003	0.0384	0.0472	0.0129	0.0009	0.0201
6	0.1889	0.0641	0.0828	0.0018	0.0614	0.0364	0.0000	0.0245	0.0385	0.0168
7	0.1089	0.1853	0.0023	0.0817	0.0306	0.0089	0.0502	0.0247	0.0000	0.0179
8	0.0414	0.2032	0.0775	0.0564	0.0209	0.0638	0.0054	0.0183	0.0406	0.0151
9	0.0092	0.1109	0.2214	0.0116	0.0909	0.0001	0.0496	0.0305	0.0003	0.0254

Tabla 5-23. Datos Experimentales Sistema de bandas $B^1\Sigma_u^+ \rightarrow X^1\Sigma_g^+$ del H^1H^2

Los datos obtenidos con la Red Neuronal Newton son los siguientes:

$v_2 v_1$	0	1	2	3	4	5	6	7	8	9
0	3.191e-3	1.623e-2	4.320e-2	7.999e-2	1.156e-1	1.389e-1	1.440e-1	1.321e-1	1.092e-1	8.218e-2
1	1.915e-2	6.530e-2	1.093e-1	1.154e-1	8.002e-2	3.170e-2	2.624e-3	5.340e-3	3.130e-2	6.283e-2
2	6.163e-2	1.279e-1	1.117e-1	4.170e-2	9.784e-4	1.460e-2	4.491e-2	5.297e-2	3.459e-2	1.029e-2
3	1.323e-1	1.355e-1	3.043e-2	3.937e-3	4.785e-2	6.137e-2	2.958e-2	2.146e-3	5.595e-3	2.412e-2
4	2.040e-1	6.043e-2	7.372e-3	6.961e-2	5.154e-2	4.062e-3	1.102e-2	3.985e-2	4.019e-2	1.713e-2
5	2.302e-1	9.974e-5	8.500e-2	4.982e-2	2.884e-4	3.840e-2	4.724e-2	1.286e-2	9.462e-4	2.009e-2
6	1.888e-1	6.407e-2	8.275e-2	1.801e-3	6.141e-2	3.638e-2	6.434e-9	2.452e-2	3.851e-2	1.676e-2
7	1.089e-1	1.852e-1	2.307e-3	8.170e-2	3.060e-2	8.890e-3	5.018e-2	2.465e-2	7.078e-6	1.787e-2
8	4.143e-2	2.031e-1	7.749e-2	5.644e-2	2.093e-2	6.380e-2	5.357e-3	1.830e-2	4.059e-2	1.508e-2
9	9.216e-3	1.109e-1	2.214e-1	1.157e-2	9.090e-2	5.305e-5	4.961e-2	3.051e-2	2.632e-4	2.545e-2

Tabla 5-24. FFC-RNA Newton Sistema de bandas $B^1\Sigma_u^+ \rightarrow X^1\Sigma_g^+$ del H^1H^2

Tiempo de ejecución: 27.759538 s

Sistema de bandas $B^3\Pi_g \rightarrow A^3\Sigma_u^+$ del $^{14}N_2$

Los datos experimentales con los que se hace la comparación del sistema de bandas de esta molécula (Abad, 1991), son los siguientes:

		$B^3\Pi_g$	$A^3\Sigma_u^+$
w_e		1734.11	1460.370
$w_e x_e$		14.47	13.891
r_e		1.2123	1.293
μ		7.0037700	

$v_2 v_1$	0	1	2	3	4	5	6	7	8	9
0	0.338	0.406	0.197	0.050	0.007					
1	0.324	0.002	0.212	0.298	0.131	0.027	0.003			
2	0.190	0.103	0.113	0.039	0.273	0.210	0.061	0.008		
3	0.088	0.178	0.001	0.162	0.002	0.180	0.260	0.106	0.019	0.002
4	0.036	0.145	0.077	0.032	0.114	0.048	0.083	0.270	0.156	0.034
5	0.014	0.086	0.127	0.009	0.088	0.043	0.104	0.019	0.243	0.202
6	0.005	0.044	0.113	0.069	0.005	0.106	0.003	0.129	0.000	0.191
7	0.002	0.020	0.075	0.101	0.018	0.038	0.081	0.007	0.116	0.017
8	0.001	0.009	0.042	0.091	0.064	0.000	0.070	0.039	0.036	0.078
9		0.004	0.022	0.064	0.083	0.025	0.013	0.077	0.008	0.068

Tabla 5-25 Datos Experimentales Sistema de bandas $B^3\Pi_g \rightarrow A^3\Sigma_u^+$ del $^{14}N_2$

Los datos obtenidos con la Red Neuronal Newton son los siguientes:

$v_2 v_1$	0	1	2	3	4	5	6	7	8	9
0	3.380e-1	4.064e-1	1.974e-1	5.015e-2	7.193e-3	5.875e-4	2.617e-5	5.703e-7	4.724e-9	6.732e-12
1	3.247e-1	2.319e-3	2.119e-1	2.982e-1	1.318e-1	2.729e-2	2.926e-3	1.614e-4	4.177e-6	3.966e-8
2	1.899e-1	1.031e-1	1.132e-1	3.863e-2	2.737e-1	2.106e-1	6.150e-2	8.466e-3	5.679e-4	1.720e-5
3	8.859e-2	1.781e-1	1.196e-3	1.622e-1	1.817e-3	1.807e-1	2.605e-1	1.065e-1	1.857e-2	1.495e-3
4	3.650e-2	1.450e-1	7.719e-2	3.230e-2	1.138e-1	4.784e-2	8.297e-2	2.705e-1	1.561e-1	3.421e-2
5	1.399e-2	8.649e-2	1.274e-1	9.027e-3	8.825e-2	4.256e-2	1.040e-1	1.911e-2	2.437e-1	2.029e-1
6	5.150e-3	4.368e-2	1.127e-1	6.905e-2	5.244e-3	1.056e-1	3.154e-3	1.290e-1	2.934e-5	1.917e-1
7	1.852e-3	2.000e-2	7.497e-2	1.009e-1	1.794e-2	3.832e-2	8.074e-2	6.771e-3	1.158e-1	1.697e-2
8	6.601e-4	8.632e-3	4.249e-2	9.079e-2	6.357e-2	9.665e-7	6.969e-2	3.941e-2	3.637e-2	7.856e-2
9	2.350e-4	3.592e-3	2.182e-2	6.407e-2	8.347e-2	2.450e-2	1.291e-2	7.732e-2	8.333e-3	6.776e-2

Tabla 5-26. FFC-RNA Newton Sistema de bandas $B^3\Pi_g \rightarrow A^3\Sigma_u^+$ del $^{14}N_2$

Tiempo de ejecución: 28.354280 s

Sistema de bandas $b^1\Sigma_g^+ \rightarrow X^3\Sigma_g^-$ del $^{16}\text{O}_2$

Los datos experimentales con los que se hace la comparación del sistema de bandas de esta molécula (Abad, 1991), son los siguientes:

	$b^1\Sigma_g^+$	$X^3\Sigma_g^-$
w_e	1432.687	1580.361
$w_e x_e$	13.95	12.073
r_e	1.22675	1.20740
μ	8.0	

$v_2 v_1$	0	1	2	3	4
0	0.93033	6.6975e-2	2.6244e-3	6.6902e-5	1.0151e-6
1	0.66967	0.79072			
2	2.6344e-3				
3	6.3536e-5				

Tabla 5-27. Datos Experimentales Sistema de bandas $b^1\Sigma_g^+ \rightarrow X^3\Sigma_g^-$ del $^{16}\text{O}_2$

Los datos obtenidos con la Red Neuronal Newton son los siguientes:

$v_2 v_1$	0	1	2	3	4	5	6	7	8	9
0	9.299e-1	6.727e-2	2.656e-3	6.873e-5	1.081e-6	7.161e-9	2.77e-11	3.51e-11	2.89e-11	1.06e-11
1	6.728e-2	7.897e-1	1.336e-1	8.950e-3	3.566e-4	8.706e-6	1.064e-7	5.974e-10	1.755e-10	1.909e-10
2	2.650e-3	1.339e-1	6.471e-1	1.953e-1	1.975e-2	1.128e-3	3.997e-5	7.937e-7	7.657e-9	5.817e-10
3	6.406e-5	8.671e-3	1.969e-1	5.072e-1	2.485e-1	3.564e-2	2.787e-3	1.367e-4	4.061e-6	6.5073e-8
4	9.380e-7	3.015e-4	1.873e-2	2.529e-1	3.754e-	2.895e-1	5.675e-2	5.883e-3	3.859e-4	1.618e-5
5	1.033e-8	5.961e-6	8.806e-4	3.337e-2	2.987e-1	2.572e-1	3.150e-1	8.256e-2	1.109e-2	9.470e-4
6	3.73e-11	8.246e-8	2.258e-5	2.042e-3	5.289e-2	3.312e-1	1.577e-1	3.228e-1	1.117e-1	1.912e-2
7	7.64e-13	4.25e-10	3.835e-7	6.611e-5	4.111e-3	7.727e-2	3.480e-1	8.102e-2	3.121e-1	1.420e-1
8	1.25e-14	7.22e-12	2.675e-9	1.356e-6	1.647e-4	7.501e-3	1.060e-1	3.476e-1	2.950e-2	2.837e-1
9	3.79e-15	7.65e-14	3.97e-11	1.229e-8	4.039e-6	3.666e-4	1.270e-2	1.380e-1	3.298e-1	3.746e-3

Tabla 5-28. FFC-RNA Newton Sistema de bandas $b^1\Sigma_g^+ \rightarrow X^3\Sigma_g^-$ del $^{16}\text{O}_2$

Tiempo de ejecución: 37.231213 s

Los resultados que proporcionan el uso de los algoritmos de RNAS son bastante precisos para la generalidad de las moléculas con números cuánticos pequeños. En cuanto al menor tiempo de ejecución de todas las RNAS el algoritmo de gradientes conjugados es el mejor.

5.3.2. Moléculas Diatómicas con Números Cuánticos Grandes

Se presentan un conjunto de FFC calculados para un rango de números cuánticos grandes. Así mismo se muestra el algoritmo empleado y el tiempo de cálculo.

Sistema de bandas $B'^3\Sigma_u^- \rightarrow B^3\Pi_g$ del $^{15}\text{N}_2$

Los datos experimentales con los que se hace la comparación del sistema de bandas de esta molécula (NIST Standard Reference Data is governed by the Standard Reference Data Act. , 2008), son los siguientes:

	$B'^3\Sigma_u^-$	$B^3\Pi_g$
w_e	1465.570	1675.355
$w_e x_e$	11.323	13.433
r_e	1.278387	1.21252
μ	7.500053859	

$v_1 v_2$	9	10	11	12	13	14	15	16	17
6	0.2848	0.7811e-1	0.5032e-2	0.1348e-4	0.4285e-5	0.6743e-8	0.1245e-8	0.019e-9	0.1885e-7
7	0.1377	0.3086	0.1019	0.6926e-2	0.9455e-5	0.7067e-5	0.7702e-8	0.364e-7	0.6824e-9
8	0.7564e-1	0.9007e-1	0.3273	0.1227	0.878e-2	0.1999e-5	0.1759e-4	0.739e-8	0.3835e-7
9	0.145e-1	0.9953e-1	0.5334e-1	0.3332	0.1463	0.1054e-1	0.1963e-5	0.2811e-4	0.7299e-7
10	0.8757e-1	0.1563e-2	0.1135	0.2758e-1	0.3349	0.1697	0.1193e-1	0.2873e-4	0.4194e-4
11	0.5063e-1	0.6831e-1	0.1077e-2	0.1166	0.1155e-1	0.3364	0.1914	0.1288e-1	0.1321e-3
12	0.3073e-2	0.6118e-1	0.4646e-1	0.9171e-2	0.1117	0.3103e-2	0.3313	0.2101	0.1307e-1
13	0.1053e-1	0.1364e-1	0.6816e-1	0.2678e-1	0.2164e-1	0.1010	0.1477e-3	0.3277	0.2288
14	0.4033e-1	0.1523e-2	0.2606e-1	0.6451e-1	0.1206e-1	0.3455e-1	0.8700e-1	0.5492e-3	0.3273

Tabla 5-29. Datos Experimentales Sistema de bandas $B'^3\Sigma_u^- \rightarrow B^3\Pi_g$ del $^{15}\text{N}_2$ (Kuzmenko, Kuznetsova, & Kuziakov, 1984)

Los datos obtenidos con la Red Neuronal Newton son los siguientes:

$v_1 v_2$	9	10	11	12	13	14	15	16	17
6	2.8508e-1	8.6246e-2	6.9564e-3	7.9797e-5	9.4203e-7	3.5871e-8	8.5361e-9	1.3086e-6	1.1508e-6
7	1.1979e-1	3.0580e-1	1.1109e-1	9.8462e-3	1.0613e-4	2.1140e-6	3.3743e-8	8.2673e-9	5.8818e-5
8	8.2296e-2	7.2902e-2	3.1589e-1	1.3706e-1	1.3214e-2	1.2991e-4	4.6969e-6	1.2728e-6	4.4266e-5
9	1.1817e-2	1.0579e-1	3.7892e-2	3.1675e-1	1.6342e-1	1.6967e-2	1.4716e-4	1.5940e-5	4.3506e-7
10	8.6338e-2	7.0916e-4	1.1813e-1	1.5053e-2	3.1017e-1	1.8964e-1	2.1031e-2	1.4771e-4	6.3121e-5
11	5.1382e-2	6.6561e-2	2.2710e-3	1.1950e-1	3.1192e-3	2.9799e-1	2.1508e-1	2.5577e-2	2.7000e-4
12	2.7217e-3	6.4448e-2	4.4187e-2	1.2455e-2	1.1194e-1	2.3029e-5	2.8216e-1	2.3877e-1	2.8576e-2
13	1.2112e-2	1.3039e-2	6.8736e-2	2.4230e-2	2.6882e-2	9.8224e-2	3.4355e-3	2.6418e-1	2.6372e-1
14	4.4401e-2	2.0952e-3	2.6523e-2	6.4710e-2	9.7078e-3	4.1767e-2	8.1269e-2	1.0991e-2	2.4611e-1

Tabla 5-30. FFC-RNA Newton Sistema de bandas $B'^3\Sigma_u^- \rightarrow B^3\Pi_g$ del $^{15}\text{N}_2$

Tiempo de ejecución: 53.022433 segundos.

En la tabla 5-30, se observa en color rojo los resultados que proporcionan el uso de los algoritmos de RNAS son bastante precisos para esta molécula para el rango de números cuánticos que se encuentra entre $v_2 \in [9, 17]$ y $v_1 \in [6, 14]$, con respecto a los valores experimentales.

Sistema de bandas $a^1\Pi_g \rightarrow a'^1\Sigma_u^-$ del $^{15}N_2$

Los datos experimentales con los que se hace la comparación del sistema de bandas de esta molécula (NIST Standard Reference Data is governed by the Standard Reference Data Act. , 2008), son los siguientes:

	$a^1\Pi_g$	$a'^1\Sigma_u^-$
w_e	1694.208	1530.254
$w_e x_e$	13.9491	12.0747
r_e	1.2203	1.2755
μ	7.500053859	

$v_1 v_2$	7	8	9	10	11	12	13
10	0.1913	0.3208	0.2208e-2	0.9465e-1	0.7899e-1	0.7146e-2	0.1072e-1
11	0.1922e-1	0.2240	0.2976	0.1217e-1	0.6869e-1	0.6660e-1	0.01826e-1
12	0.1250e-3	0.2415e-1	0.2538	0.2689	0.2622e-1	0.4562e-1	0.8797e-1
13	0.1277e-4	0.1402e-3	0.2973e-1	0.2806	0.2414	0.4153e-1	0.2704e-1
14	0.1503e-7	0.1262e-4	0.1045e-3	0.3539e-1	0.3107	0.2150	0.5600e-1
15	0.1455e-6	0.1564e-6	0.3862e-4	0.8340e-4	0.4063e-1	0.3318	0.1901

Tabla 5-31. Datos Experimentales Sistema de bandas $a^1\Pi_g \rightarrow a'^1\Sigma_u^-$ del $^{15}N_2$ (Kuzmenko, Kuznetsova, & Kuziakov, 1984)

Los datos obtenidos con la Red Neuronal Newton son los siguientes:

$v_1 v_2$	7	8	9	10	11	12	13
10	2.0647e-1	3.0645e-1	6.4179e-3	8.1876e-2	8.2771e-2	1.1258e-2	6.7294e-3
11	2.2192e-2	2.3924e-1	2.7733e-1	1.9956e-2	5.6010e-2	8.7229e-2	2.3659e-2
12	1.7141e-4	2.8314e-2	2.6997e-1	2.4627e-1	3.6589e-2	3.4201e-2	8.4626e-2
13	1.0398e-5	1.8890e-4	3.4805e-2	2.9841e-1	2.1539e-1	5.3370e-2	1.7786e-2
14	7.6275e-9	1.8684e-5	1.8964e-4	4.1467e-2	3.2450e-1	1.8617e-1	6.8395e-2
15	4.5916e-8	5.3756e-10	3.2772e-5	1.7303e-4	4.8123e-2	3.4813e-1	1.5966e-1

Tabla 5-32. FFC-RNA Newton Sistema de bandas $a^1\Pi_g \rightarrow a'^1\Sigma_u^-$ del $^{15}N_2$

Tiempo de ejecución: 30.505142 segundos.

En la tabla 5-32, se observa en color rojo los resultados que proporcionan el uso de los algoritmos de RNAS son bastante precisos para esta molécula para el rango de números cuánticos que se encuentra entre $v_2 \in [7,13]$ y $v_1 \in [10,15]$, con respecto a los valores experimentales.

Sistema de bandas $B^3\Pi_g \rightarrow A^3\Sigma_u^+$ del $^{15}N_2$

Los datos experimentales con los que se hace la comparación del sistema de bandas de esta molécula (NIST Standard Reference Data is governed by the Standard Reference Data Act. , 2008), son los siguientes:

	$B^3\Pi_g$	$A^3\Sigma_u^+$
w_e	1675.355	1411.210
$w_e x_e$	13.433	12.921
r_e	1.21252	1.2864
μ	7.500053859	

$v_1 v_2$	7	8	9	10	11	12	13
11	0.2378	0.1516	0.3931e-1	0.5171e-1	0.7956e-1	0.6061e-2	0.1769e-1
12	0.7324e-1	0.2595	0.9598e-1	0.7216e-1	0.1992e-1	0.8327e-1	0.2420e-1
13	0.8555e-2	0.9955e-1	0.2722	0.4969e-1	0.9926e-1	0.2437e-2	0.7088e-1
14	0.3660e-3	0.1343e-1	0.1302	0.2728	0.1739e-1	0.1135	0.1597e-2
15	0.3998e-5	0.6280e-3	0.2011e-1	0.1610	0.2571	0.1725e-2	0.1116
16	0.2266e-8	0.6741e-5	0.1039e-2	0.2882e-1	0.1930	0.2305	0.2003e-2
17	0.1955e-7	0.3053e-8	0.1367e-4	0.1367e-2	0.3993e-1	0.2241	0.1963

Tabla 5-33. Datos Experimentales Sistema de bandas $B^3\Pi_g \rightarrow A^3\Sigma_u^+$ del $^{15}N_2$ (Kuzmenko, Kuznetsova, & Kuziakov, 1984)

Los datos obtenidos con la Red Neuronal Newton son los siguientes:

$v_1 v_2$	7	8	9	10	11	12	13
11	2.3872e-1	1.2790e-1	4.9495e-2	4.2753e-2	8.2359e-2	8.3682e-3	1.5783e-2
12	8.3794e-2	2.5868e-1	7.4706e-2	8.3008e-2	1.3570e-2	8.2452e-2	2.7752e-2
13	1.1522e-2	1.1253e-1	2.6424e-1	3.3048e-2	1.0713e-1	5.2460e-4	6.7057e-2
14	6.4639e-4	1.7969e-2	1.4362e-1	2.5516e-1	7.7461e-3	1.1599e-1	4.3463e-3
15	1.2143e-5	1.1287e-3	2.6647e-2	1.7535e-1	2.3278e-1	1.0724e-5	1.0897e-1
16	9.9314e-7	2.8952e-5	1.7634e-3	3.8146e-2	2.0593e-1	1.9932e-1	7.6394e-3
17	1.2410e-5	6.2348e-5	1.1534e-6	3.0219e-3	5.3038e-2	2.2929e-1	1.6340e-1

Tabla 5-34. FFC-RNA Newton Sistema de bandas $B^3\Pi_g \rightarrow A^3\Sigma_u^+$ del $^{15}N_2$

Tiempo de ejecución: 30.505142segundos.

En la tabla 5-34, se observa en color rojo los resultados que proporcionan el uso de los algoritmos de RNAS son bastante precisos para esta molécula para el rango de números cuánticos que se encuentra entre $v_2 \in [7,13]$ y $v_1 \in [11,17]$, con respecto a los valores experimentales.

Sistema de bandas $B^3\Pi_g \rightarrow X^1\Sigma_g^+$ del $^{15}N_2$

Los datos experimentales con los que se hace la comparación del sistema de bandas de esta molécula (NIST Standard Reference Data is governed by the Standard Reference Data Act. , 2008), son los siguientes:

	$B^3\Pi_g$	$X^1\Sigma_g^+$
w_e	1675.355	2358.57
$w_e x_e$	13.433	14.324
r_e	1.21252	1.097685
μ	7.500053859	

$v_1 v_2$	11	12	13	14	15	16	17	18
4	0.1489	0.7565e-1	0.2516e-1	0.5741e-2	0.9421e-3	0.1078e-3	0.9893e-5	0.7493e-6
5	0.1121	0.1673	0.1144	0.4661e-1	0.1268e-1	0.2407e-2	0.3192e-3	0.2901e-4
6	0.6829e-2	0.4851e-1	0.1544	0.1477	0.758e-1	0.2778e-1	0.2478e-1	0.5476e-2
7	0.8227e-1	0.4476e-1	0.6578e-2	0.1101	0.1628	0.1084	0.4256e-1	0.1087e-1
8	0.1234e-1	0.4116e-1	0.8045e-1	0.3535e-2	0.5637e-1	0.1547	0.1377	0.6576e-1
9	0.36e-1	0.4759e-1	0.5618e-2	0.8338e-1	0.3204e-1	0.1474e-1	0.1245	0.1578
10	0.4793e-1	0.4232e-2	0.6555e-1	0.3824e-2	0.5426e-1	0.6713e-1	0.1339e-4	0.7848e-1

Tabla 5-35. Datos Experimentales Sistema de bandas $B^3\Pi_g \rightarrow X^1\Sigma_g^+$ del $^{15}N_2$ (Kuzmenko, Kuznetsova, & Kuziakov, 1984)

Los datos obtenidos con la Red Neuronal Newton son los siguientes:

$v_1 v_2$	11	12	13	14	15	16	17	18
4	1.5902e-1	1.0320e-1	4.2737e-2	1.2220e-2	1.9165e-3	6.8813e-5	5.6876e-3	6.4307e-3
5	6.1484e-2	1.5139e-1	1.3884e-1	7.2842e-2	2.6089e-2	1.1383e-2	1.5489e-4	4.2311e-2
6	3.2303e-2	1.0777e-2	1.0986e-1	1.5639e-1	1.0706e-1	3.7599e-2	6.6807e-3	4.4303e-3
7	5.2752e-2	7.3170e-2	2.0745e-3	5.4637e-2	1.4515e-1	1.3560e-1	1.0260e-1	4.6377e-3
8	3.6419e-2	1.0690e-2	8.2579e-2	2.9700e-2	1.2943e-2	1.2489e-1	1.1569e-1	3.0351e-1
9	1.1874e-2	6.3076e-2	1.6436e-3	5.5380e-2	6.6083e-2	1.5770e-3	8.5165e-2	2.7682e-2
10	5.4867e-2	1.1224e-3	5.3078e-2	2.6313e-2	1.7937e-2	7.4151e-2	2.2992e-2	5.2012e-2

Tabla 5-36. FFC-RNA Newton Sistema de bandas $B^3\Pi_g \rightarrow X^1\Sigma_g^+$ del $^{15}N_2$

Tiempo de ejecución: 47.263279 segundos.

En la tabla 5-36, se observa en color rojo los resultados que proporcionan el uso de los algoritmos de RNAS son bastante precisos para esta molécula para el rango de números cuánticos que se encuentra entre $v_2 \in [11, 17]$ y $v_1 \in [4, 10]$, con respecto a los valores experimentales. El modelo de Morse no es adecuado a partir de los números cuánticos mayores a 17.

Sistema de bandas $B'^3\Sigma_u^- \rightarrow X^1\Sigma_g^+$ del $^{15}\text{N}_2$

Los datos experimentales con los que se hace la comparación del sistema de bandas de esta molécula (NIST Standard Reference Data is governed by the Standard Reference Data Act. , 2008), son los siguientes:

	$B'^3\Sigma_u^-$	$X^1\Sigma_g^+$
w_e	1465.570	2358.57
$w_e x_e$	11.323	14.324
r_e	1.278387	1.097685
μ	7.500053859	

$v_1 v_2$	11	12	13	14	15	16	17	18
2	0.1470	0.1180	0.669e-1	0.2807e-1	0.9052e-2	0.2303e-2	0.4548e-3	0.722e-4
3	0.2564e-1	0.1039	0.1428	0.1132	0.6152e-1	0.2439e-1	0.7247e-2	0.1623e-2
4	0.5947e-1	0.3041e-2	0.3108e-1	0.114	0.1391	0.1020	0.5033e-1	0.1778e-1
5	0.1356e-1	0.7285e-1	0.4657e-1	0.2173e-4	0.4955e-1	0.1261	0.1325	0.8351e-1
6	0.5747e-1	0.6231e-2	0.2758e-2	0.7451e-1	0.2655e-1	0.5251e-1	0.7962e-1	0.1377
7	0.1284e-2	0.3818e-1	0.4506e-1	0.1279e-5	0.492e-1	0.6504e-1	0.6375e-2	0.2779e-1

Tabla 5-37 .Datos Experimentales Sistema de bandas $B'^3\Sigma_u^- \rightarrow X^1\Sigma_g^+$ del $^{15}\text{N}_2$ (Kuzmenko, Kuznetsova, & Kuziakov, 1984)

Los datos obtenidos con la Red Neuronal Newton son los siguientes:

$v_1 v_2$	11	12	13	14	15	16	17	18
2	1.3311e-1	1.3292e-1	9.0712e-2	4.5661e-2	1.7429e-2	6.1214e-3	3.9417e-3	1.6796e-1
3	2.8549e-3	6.1929e-2	1.2618e-1	1.3014e-1	8.9204e-2	4.0922e-2	1.1339e-2	1.0029e-1
4	7.4656e-2	2.3516e-2	4.0851e-3	6.7199e-2	1.2492e-1	1.2490e-1	8.8108e-2	3.0231e-1
5	2.2874e-4	4.8843e-2	6.8597e-2	1.3897e-2	1.2986e-2	8.4908e-2	1.0868e-1	1.0071e-4
6	5.4347e-2	2.7574e-2	3.7669e-3	6.0016e-2	6.0206e-2	3.7478e-3	5.2431e-2	3.7941e-1
7	1.4904e-2	1.3818e-2	5.5139e-2	1.2443e-2	1.4367e-2	6.3844e-2	6.7351e-2	2.0756e-2

Tabla 5-38. FFC-RNA Newton Sistema de bandas $B'^3\Sigma_u^- \rightarrow X^1\Sigma_g^+$ del $^{15}\text{N}_2$

Tiempo de ejecución: 28.745154 segundos.

En la tabla 5-38, se observa en color rojo los resultados que proporcionan el uso de los algoritmos de RNAS son bastante precisos para esta molécula para el rango de números cuánticos que se encuentra entre $v_2 \in [11, 17]$ y $v_1 \in [4, 10]$, con respecto a los valores experimentales. El modelo de Morse no es adecuado a partir de los números cuánticos mayores a 17.

Sistema de bandas $A^3\Sigma_u^+ \rightarrow X^1\Sigma_g^+$ del $^{15}N_2$

Los datos experimentales con los que se hace la comparación del sistema de bandas de esta molécula (NIST Standard Reference Data is governed by the Standard Reference Data Act. , 2008), son los siguientes:

	$A^3\Sigma_u^+$	$X^1\Sigma_g^+$
w_e	1411.210	2358.57
$w_e x_e$	12.921	14.324
r_e	1.2864	1.097685
μ	7.500053859	

$v_1 v_2$	11	12	13	14	15	16	17	18
1	0.8884e-1	0.4555e-1	0.1879e-1	0.6230e-2	0.1699e-2	0.3773e-3	0.6878e-4	0.9530e-5
2	0.1276	0.1368	0.9951e-1	0.5318e-1	0.2203e-1	0.7302e-2	0.1932e-2	0.4171e-3
3	0.1670e-3	0.4670e-1	0.1170	0.1338	0.9864e-1	0.5246e-1	0.2110e-1	0.6561e-2
4	0.7624e-1	0.3702e-1	0.9779e-4	0.4705e-1	0.1165	0.1307	0.9232e-1	0.4619e-1
5	0.1822e-2	0.3246e-1	0.7336e-1	0.2989e-1	0.1469e-2	0.5811e-1	0.1229	0.1239
6	0.4341e-1	0.4304e-1	0.1158e-3	0.4127e-1	0.6905e-1	0.1804e-1	0.7823e-1	0.7681e-1

Tabla 5-39. Datos Experimentales Sistema de bandas $A^3\Sigma_u^+ \rightarrow X^1\Sigma_g^+$ del $^{15}N_2$ (Kuzmenko, Kuznetsova, & Kuziakov, 1984)

Los datos obtenidos con la Red Neuronal Newton son los siguientes:

$v_1 v_2$	11	12	13	14	15	16	17	18
1	1.0940e-1	6.5032e-2	3.0903e-2	1.2001e-2	3.8277e-3	7.6105e-4	1.9153e-4	9.0601e-2
2	9.6325e-2	1.3422e-1	1.1884e-1	7.6125e-2	3.7316e-2	1.5846e-2	9.1519e-3	1.6114e-1
3	7.7369e-3	1.3748e-2	7.9110e-2	1.2747e-1	1.2041e-1	7.5241e-2	3.0711e-2	6.2017e-2
4	6.2192e-2	6.2223e-2	1.0005e-2	1.1701e-2	7.5294e-2	1.2764e-1	1.2560e-1	3.8978e-1
5	1.8450e-2	6.9470e-3	6.0251e-2	5.7655e-2	6.0073e-3	1.6390e-2	7.2919e-2	9.8933e-5
6	1.9454e-2	5.4894e-2	1.3578e-2	1.1335e-2	6.6966e-2	4.9052e-2	8.3787e-6	2.4352e-1

Tabla 5-40. FFC-RNA Newton Sistema de bandas $A^3\Sigma_u^+ \rightarrow X^1\Sigma_g^+$ del $^{15}N_2$

Tiempo de ejecución: 30.590602 segundos.

En la tabla 5-40, se observa en color rojo los resultados que proporcionan el uso de los algoritmos de RNAS son bastante precisos para esta molécula para el rango de números cuánticos que se encuentra entre $v_2 \in [12, 17]$ y $v_1 \in [1, 6]$, con respecto a los valores experimentales. El modelo de Morse no es adecuado para el número cuántico $v_2 = 11$ y a partir de los números cuánticos mayores a 17.

Sistema de bandas $a'^1\Sigma_u^- \rightarrow X^1\Sigma_g^+$ del $^{15}N_2$

Los datos experimentales con los que se hace la comparación del sistema de bandas de esta molécula (NIST Standard Reference Data is governed by the Standard Reference Data Act. , 2008), son los siguientes:

	$a'^1\Sigma_u^-$	$X^1\Sigma_g^+$
w_e	1530.254	2358.57
$w_e x_e$	12.0747	14.324
r_e	1.2755	1.097685
μ	7.500053859	

$v_1 v_2$	11	12	13	14	15	16	17	18
2	0.1470	0.1089	0.5740e-1	0.2246e-1	0.6740e-2	0.1592e-2	0.2901e-3	0.4264e-4
3	0.3912e-1	0.1187	0.1428	0.1029	0.5138e-1	0.1877e-1	0.5133e-2	0.1052e-2
4	0.4763e-1	0.4801e-4	0.4852e-1	0.1259	0.1361	0.8973e-1	0.4028e-1	0.1297e-1
5	0.2556e-1	0.7708e-1	0.3264e-1	0.2786e-2	0.7057e-1	0.1368	0.1244	0.7003e-1
6	0.5149e-1	0.8223e-3	0.4279e-1	0.7162e-1	0.1314e-1	0.1724e-1	0.1020	0.1408
7	0.2096e-3	0.4923e-1	0.3331e-1	0.3157e-2	0.6323e-1	0.5269e-1	0.3302e-3	0.5007e-1

Tabla 5-41. Datos Experimentales Sistema de bandas $a'^1\Sigma_u^- \rightarrow X^1\Sigma_g^+$ del $^{15}N_2$ (Kuzmenko, Kuznetsova, & Kuziakov, 1984)

Los datos obtenidos con la Red Neuronal Newton son los siguientes:

$v_1 v_2$	11	12	13	14	15	16	17	18
2	1.4488e-1	1.2753e-1	7.7325e-2	3.4436e-2	1.1481e-2	3.6445e-3	2.5641e-3	1.6106e-1
3	1.2321e-2	8.7562e-2	1.4033e-1	1.2338e-1	7.3433e-2	2.8888e-2	6.5610e-3	1.0936e-1
4	6.9596e-2	1.0405e-2	1.7237e-2	9.6215e-2	1.3671e-1	1.1391e-1	6.8874e-2	2.6522e-1
5	4.5319e-3	6.4146e-2	5.8742e-2	2.7825e-3	3.5615e-2	1.1479e-1	1.1253e-1	1.7214e-3
6	5.8625e-2	1.5585e-2	1.4665e-2	7.2369e-2	4.2707e-2	4.0293e-4	9.3890e-2	4.3175e-1
7	7.1421e-3	2.5646e-2	5.2739e-2	2.7956e-3	3.2515e-2	6.5986e-2	3.8360e-2	2.6543e-3

Tabla 5-42. FFC-RNA Newton Sistema de bandas $a'^1\Sigma_u^- \rightarrow X^1\Sigma_g^+$ del $^{15}N_2$

Tiempo de ejecución: 29.708305 segundos.

En la tabla 5-42, se observa en color rojo los resultados que proporcionan el uso de los algoritmos de RNAS son bastante precisos para esta molécula para el rango de números cuánticos que se encuentra entre $v_2 \in [11, 17]$ y $v_1 \in [2, 7]$, con respecto a los valores experimentales. El modelo de Morse no es adecuado a partir de los números cuánticos mayores a 17.

Sistema de bandas $a^1\Pi_g \rightarrow X^1\Sigma_g^+$ del $^{15}N_2$

Los datos experimentales con los que se hace la comparación del sistema de bandas de esta molécula (NIST Standard Reference Data is governed by the Standard Reference Data Act. , 2008), son los siguientes:

	$a^1\Pi_g$	$X^1\Sigma_g^+$
w_e	1694.208	2358.57
$w_e x_e$	13.9491	14.324
r_e	1.2203	1.097685
μ	7.500053859	

$v_1 v_2$	11	12	13	14	15	16	17	18
4	0.1644	0.1136	0.4912e-1	0.1433e-1	0.2991e-2	0.4451e-3	0.4841e-4	0.3776e-5
5	0.4687e-1	0.1472	0.1481	0.8134e-1	0.2880e-1	0.7055e-2	0.1210e-2	0.1451e-3
6	0.4631e-1	0.3962e-2	0.9858e-1	0.1602	0.1165	0.5072e-1	0.1454e-1	0.2881e-2
7	0.3804e-1	0.8192e-1	0.7572e-2	0.7170e-1	0.1431	0.1453	0.7830	0.2636e-1
8	0.5004e-1	0.2884e-2	0.7844e-1	0.4214e-1	0.5531e-2	0.1027	0.1575	0.1077

Tabla 5-43. Datos Experimentales Sistema de bandas $a^1\Pi_g \rightarrow X^1\Sigma_g^+$ del $^{15}N_2$ (Kuzmenko, Kuznetsova, & Kuziakov, 1984)

Los datos obtenidos con la Red Neuronal Newton son los siguientes:

$v_1 v_2$	11	12	13	14	15	16	17	18
4	1.5983e-1	1.3435e-1	6.7283e-2	2.2446e-2	4.2356e-3	4.7637e-5	8.3206e-3	3.9937e-4
5	1.7134e-2	1.1970e-1	1.5798e-1	1.0474e-1	4.5417e-2	1.9643e-2	1.0945e-6	3.1329e-2
6	7.1296e-2	1.1905e-3	5.8901e-2	1.5029e-1	1.3666e-1	5.7638e-2	2.4424e-2	1.3112e-2
7	1.1849e-2	8.4123e-2	3.0591e-2	1.1770e-2	1.1198e-1	1.6669e-1	1.2724e-1	6.4338e-2
8	6.4413e-2	2.3587e-3	5.3367e-2	6.9116e-2	8.9870e-4	6.3530e-2	1.1493e-1	2.5601e-1

Tabla 5-44. FFC-RNA Newton Sistema de bandas $a^1\Pi_g \rightarrow X^1\Sigma_g^+$ del $^{15}N_2$

Tiempo de ejecución: 24.951172 segundos.

En la tabla 5-44, se observa en color rojo los resultados que proporcionan el uso de los algoritmos de RNAS son bastante precisos para esta molécula para el rango de numero cuánticos que se encuentra entre $v_2 \in [11, 17]$ y $v_1 \in [4, 8]$, con respecto a los valores experimentales. El modelo de Morse no es adecuado a partir del número cuántico 17.

Sistema de bandas $\omega^1\Delta_u \rightarrow B^3\Pi_g$ del $^{15}N_2$

Los datos experimentales con los que se hace la comparación del sistema de bandas de esta molécula (NIST Standard Reference Data is governed by the Standard Reference Data Act. , 2008), son los siguientes:

	$\omega^1\Delta_u$	$B^3\Pi_g$
w_e	1465.570	1675.355
$w_e x_e$	11.323	13.433
r_e	1.278387	1.21252
μ	7.500053859	

$v_1 v_2$	11	12	13	14	15	16	17	18
2	0.1355	0.8202e-1	0.3603e-1	0.1182e-1	0.2968e-2	0.5865e-3	0.8824e-4	0.1108e-1
3	0.8430e-1	0.1441	0.1288	0.7388e-1	0.3012e-1	0.9033e-2	0.2023e-2	0.3364e-3
4	0.1567e-1	0.1425e-1	0.9773e-1	0.1452	0.1149	0.5935e-1	0.2136e-1	0.2136e-1
5	0.6087e-1	0.6556e-1	0.4365e-2	0.3284e-1	0.1191	0.1417	0.9466e-1	0.4134e-1
6	0.2388e-1	0.1030e-1	0.7361e-1	0.4362e-1	0.5058e-3	0.6630e-1	0.1400	0.1267

Tabla 5-45. Datos Experimentales Sistema de bandas $\omega^1\Delta_u \rightarrow B^3\Pi_g$ del $^{15}N_2$ (Kuzmenko, Kuznetsova, & Kuziakov, 1984)

Los datos obtenidos con la Red Neuronal Newton son los siguientes:

$v_1 v_2$	11	12	13	14	15	16	17	18
2	1.4589e-1	1.0168e-1	5.0182e-2	1.8256e-2	4.8115e-3	1.4026e-3	1.3984e-3	1.7392e-1
3	5.2080e-2	1.3004e-1	1.4046e-1	9.3684e-2	4.4463e-2	1.3483e-2	1.7372e-3	1.3138e-1
4	3.7015e-2	1.2688e-3	6.5993e-2	1.3668e-1	1.2652e-1	7.7594e-2	4.1416e-2	2.1255e-1
5	3.7324e-2	7.6433e-2	2.0062e-2	1.0790e-2	9.7545e-2	1.4570e-1	8.1722e-2	7.6453e-3
6	4.2720e-2	3.9823e-4	5.6579e-2	6.3281e-2	4.3912e-3	3.3606e-2	1.7846e-1	3.4988e-1

Tabla 5-46. FFC-RNA Newton Sistema de bandas $\omega^1\Delta_u \rightarrow B^3\Pi_g$ del $^{15}N_2$

Tiempo de ejecución: 24.951172 segundos.

En la tabla 5-46, se observa en color rojo los resultados que proporcionan el uso de los algoritmos de RNAS son bastante precisos para esta molécula para el rango de números cuánticos que se encuentra entre $v_2 \in [11, 17]$ y $v_1 \in [2, 6]$, con respecto a los valores experimentales. El modelo de Morse no es adecuado a partir del número cuántico 17.

Sistema de bandas $B^1\Pi_u \rightarrow X^1\Sigma_g^+$ del ${}^7\text{Li}_2$

Los datos experimentales con los que se hace la comparación del sistema de bandas de esta molécula (NIST Standard Reference Data is governed by the Standard Reference Data Act. , 2008), son los siguientes:

	$B^1\Pi_u$	$X^1\Sigma_g^+$
w_e	269.69	351.43
$w_e x_e$	2.744	2.592
r_e	2.936	2.672
μ	3.50908	

$v_1 v_2$	11	12	13	14	15	16
6	0.1892	0.1052	0.0339	0.0070	0.0010	0.0001
7	0.0713	0.1871	0.1470	0.0602	0.0152	0.0026
8	0.0495	0.0150	0.1485	0.1779	0.0960	0.0303
9	0.0372	0.0905	0.0013	0.0845	0.1833	0.1368
10	0.0679	0.0017	0.0913	0.0328	0.0237	0.1523

Tabla 5-47. Datos Experimentales Sistema de bandas $B^1\Pi_u \rightarrow X^1\Sigma_g^+$ del ${}^7\text{Li}_2$ (Kuzmenko, Kuznetsova, & Kuziakov, 1984)

Los datos obtenidos con la Red Neuronal Newton son los siguientes:

$v_1 v_2$	11	12	13	14	15	16
6	8.2079e-3	2.4123e-3	5.9047e-3	3.2505e-6	2.8066e-3	1.0729e-4
7	1.8822e-2	5.2084e-3	4.0369e-3	4.3390e-3	8.9259e-4	2.7685e-3
8	1.2559e-2	6.4151e-3	7.4962e-3	7.1303e-4	3.4467e-3	8.0371e-5
9	2.1175e-2	1.4639e-2	4.8212e-3	8.0044e-3	1.2211e-3	4.0859e-3
10	4.4541e-2	2.7004e-3	1.8028e-2	2.4343e-4	6.8571e-3	3.2841e-5

Tabla 5-48. FFC-RNA Newton Sistema de bandas $B^1\Pi_u \rightarrow X^1\Sigma_g^+$ del ${}^7\text{Li}_2$

Tiempo de ejecución: 21.578273 segundos.

En la tabla 5-48, se observa que el modelo de Morse para esta molécula no es adecuado para obtener buenos resultados con el uso de los algoritmos de RNAS para el rango de números cuánticos que se encuentra entre $v_2 \in [11, 16]$ y $v_1 \in [6, 10]$.

6. Conclusiones

A lo largo de éste trabajo se han introducido los conceptos necesarios para la comprensión de las Redes Neuronales Artificiales Supervisadas y para aproximar numéricamente integrales definidas. Aplicando todos estos conceptos, se realizaron seis técnicas para aproximar numéricamente integrales definidas usando un enfoque de Redes Neuronales Artificiales Supervisadas; el uso de dos RNAS clásicas con un factor de aprendizaje constante y cuatro RNAS con un factor variable de aprendizaje, donde estos aprovechan la potencialidad de los métodos de optimización, como son los Gradientes Conjugados, Gradientes Conjugado Residuales, Newton y Newton Truncado.

Para aproximar la integral de la función $f(x)$ en el intervalo $[a, b]$ se aplica el método que consiste en dos etapas, en la primera etapa se ajusta a la función como una combinación lineal de las funciones coseno, para ello se utilizó las RNAS mencionadas anteriormente:

$$f(x) = \sum_{i=0}^N w_i \cos(ix).$$

Para varias funciones muy oscilantes con y sin singularidad, se compararon los resultados de las aproximaciones del ajuste de la función, donde no se observaron problemas. El método de gradientes conjugados presenta el menor tiempo de ejecución para aproximar la función y el método de Newton de un solo entrenamiento, en general, los métodos con las RNAS con factor de aprendizaje variable reducen el número de entrenamientos y el tiempo de cálculo.

En la segunda etapa se integra la combinación lineal de las funciones coseno, que es inmediata.

$$\int_a^b f(x) dx = \int_a^b \sum_{i=0}^N w_i \cos(ix) dx.$$

Al comparar los resultados para varias funciones muy oscilantes sin singularidad, los resultados de la aproximación de la integral fueron equivalentes para la mayoría de las RNAS. Para las funciones con singularidad se mejora el Método de Simpson Adaptivo para aproximar numéricamente la integral definida para algunas funciones.

Una aplicación importante en la astrofísica es detectar las moléculas interestelares, para esto requerimos calcular los FFC para moléculas diatómicas

Definida como la integral de traslape vibracional; es decir, el cuadrado de la integral sobre el producto de las eigenfunciones vibracionales de los dos estados involucrados (base y excitación), viene dado por:

$$\left| \int \Psi_v^* \Psi_{v'} dr \right|^2.$$

Estas integrales para el potencial de Morse no se pueden obtener analíticamente, por lo que se aproximan numéricamente. Conforme se ha mejorado la tecnología en espectrometría se ha requerido aproximar los FFC para números cuánticos vibracionales grandes, esto nos ha llevado a la necesidad de buscar nuevas formas de aproximar las integrales para funciones muy oscilantes, por lo que se implementaron seis algoritmos para RNAS para aproximar estas integrales. El uso de las seis RNAS es equivalente, lo único que se gana es tiempo en el número de entrenamientos necesarios para ajustar a función de onda y el cálculo de la integral.

El uso de estos algoritmos son equivalentes al método de Simpson Adaptivo (Abad, 1991) para números cuánticos pequeños, donde se corrigen los intervalos de integración a [0.4,3.5] para mejorar la aproximación de estas integrales. Se incremento el rango de los números cuánticos vibracionales con una mejor eficiencia en los resultados producidos por estas RNAS.

Finalmente, se creó una aplicación en java que permite la aproximación de funciones de manera general y un apartado para el cálculo de los FFC para el potencial de Morse. Se tiene programas en MatLab para comparación de resultados.

Apéndice A: Tablas

Tabla 5-1. Aproximación de las funciones con el Método de Newton.....	45
Tabla 5-2. Comparación de todas las Redes Neuronales para la aproximación de las Funciones.....	46
Tabla 5-3. Aproximación a la Integral Definida de las Funciones con las diferentes RNA.....	47
Tabla 5-4. Aproximación a la Integral Definida de las Funciones con las diferentes RNA.....	47
Tabla 5-5. Aproximación de las funciones con el Método de Newton.....	48
Tabla 5-6. Comparación de todas las Redes Neuronales para la aproximación de las Funciones.....	48
Tabla 5-7. Aproximación a la Integral Definida de las Funciones con las diferentes RNA.....	49
Tabla 5-8. Aproximación a la Integral Definida de las Funciones con las diferentes RNA.....	50
Tabla 5-9. Datos Experimentales Sistema de bandas $B3\Pi g \rightarrow A3\Sigma u$ + del $^{15}N2$	51
Tabla 5-10. FFC-RNA Newton Sistema de bandas $B3\Pi g \rightarrow A3\Sigma u$ + del $^{15}N2$	51
Tabla 5-11. Datos Experimentales Sistema de bandas $B'3\Sigma u \rightarrow B3\Pi g$ del $^{15}N2$	52
Tabla 5-12. FFC- RNA Newton Sistema de bandas $B'3\Sigma u \rightarrow B3\Pi g$ del $^{15}N2$	52
Tabla 5-13. Datos Experimentales Sistema de bandas $A1\Sigma u \rightarrow X1\Sigma g$ + del $^7Li2$	53
Tabla 5-14. FFC-RNA Newton Sistema de bandas $A1\Sigma u \rightarrow X1\Sigma g$ + del $^7Li2$	53
Tabla 5-15. Datos Experimentales Sistema de bandas $B1\Pi u \rightarrow X1\Sigma g$ + del $^7Li2$	54
Tabla 5-16. FFC-RNA Newton Sistema de bandas $B1\Pi u \rightarrow X1\Sigma g$ + del $^7Li2$	54
Tabla 5-17. Datos Experimentales Sistema de bandas $A2\Pi \rightarrow X2\Sigma$ + del CN	55
Tabla 5-18. FFC-RNA Newton Sistema de bandas $A2\Pi \rightarrow X2\Sigma$ + del CN	55
Tabla 5-19. Datos Experimentales Sistema de bandas $B1\Sigma \rightarrow X1\Sigma$ + del $B11H1$	56
Tabla 5-20. FFC-RNA Newton Sistema de bandas $B1\Sigma \rightarrow X1\Sigma$ + del $B11H1$	56
Tabla 5-21. Datos Experimentales Sistema de bandas $A1\Sigma \rightarrow X1\Sigma$ + del $AgH1$	57
Tabla 5-22. FFC-RNA Newton Sistema de bandas $A1\Sigma \rightarrow X1\Sigma$ + del $AgH1$	57
Tabla 5-23. Datos Experimentales Sistema de bandas $B1\Sigma u \rightarrow X1\Sigma g$ + del $H1H2$	58
Tabla 5-24. FFC-RNA Newton Sistema de bandas $B1\Sigma u \rightarrow X1\Sigma g$ + del $H1H2$	58
Tabla 5-25 Datos Experimentales Sistema de bandas $B3\Pi g \rightarrow A3\Sigma u$ + del $^{14}N2$	59
Tabla 5-26. FFC-RNA Newton Sistema de bandas $B3\Pi g \rightarrow A3\Sigma u$ + del $^{14}N2$	59
Tabla 5-27. Datos Experimentales Sistema de bandas $b1\Sigma g \rightarrow X3\Sigma g$ - del $^{16}O2$	60
Tabla 5-28. FFC-RNA Newton Sistema de bandas $b1\Sigma g \rightarrow X3\Sigma g$ - del $^{16}O2$	60
Tabla 5-29. Datos Experimentales Sistema de bandas $B'3\Sigma u \rightarrow B3\Pi g$ del $^{15}N2$ (Kuzmenko, Kuznetsova, & Kuziakov, 1984)	61

Tabla 5-30. FFC-RNA Newton Sistema de bandas $B'3\Sigma u \rightarrow B3\Pi g$ del $^{15}N2$	61
Tabla 5-31. Datos Experimentales Sistema de bandas $a1\Pi g \rightarrow a'1\Sigma u$ – del $^{15}N2$ (Kuzmenko, Kuznetsova, & Kuziakov, 1984)	62
Tabla 5-32. FFC-RNA Newton Sistema de bandas $a1\Pi g \rightarrow a'1\Sigma u$ – del $^{15}N2$	62
Tabla 5-33. Datos Experimentales Sistema de bandas $B3\Pi g \rightarrow A3\Sigma u$ + del $^{15}N2$ (Kuzmenko, Kuznetsova, & Kuziakov, 1984)	63
Tabla 5-34. FFC-RNA Newton Sistema de bandas $B3\Pi g \rightarrow A3\Sigma u$ + del $^{15}N2$	63
Tabla 5-35. Datos Experimentales Sistema de bandas $B3\Pi g \rightarrow X1\Sigma g$ + del $^{15}N2$ (Kuzmenko, Kuznetsova, & Kuziakov, 1984)	64
Tabla 5-36. FFC-RNA Newton Sistema de bandas $B3\Pi g \rightarrow X1\Sigma g$ + del $^{15}N2$	64
Tabla 5-37. Datos Experimentales Sistema de bandas $B'3\Sigma u \rightarrow X1\Sigma g$ + del $^{15}N2$ (Kuzmenko, Kuznetsova, & Kuziakov, 1984)	65
Tabla 5-38. FFC-RNA Newton Sistema de bandas $B'3\Sigma u \rightarrow X1\Sigma g$ + del $^{15}N2$	65
Tabla 5-39. Datos Experimentales Sistema de bandas $A3\Sigma u \rightarrow X1\Sigma g$ + del $^{15}N2$ (Kuzmenko, Kuznetsova, & Kuziakov, 1984)	66
Tabla 5-40. FFC-RNA Newton Sistema de bandas $A3\Sigma u \rightarrow X1\Sigma g$ + del $^{15}N2$	66
Tabla 5-41. Datos Experimentales Sistema de bandas $a'1\Sigma u \rightarrow X1\Sigma g$ + del $^{15}N2$ (Kuzmenko, Kuznetsova, & Kuziakov, 1984)	67
Tabla 5-42. FFC-RNA Newton Sistema de bandas $a'1\Sigma u \rightarrow X1\Sigma g$ + del $^{15}N2$	67
Tabla 5-43. Datos Experimentales Sistema de bandas $a1\Pi g \rightarrow X1\Sigma g$ + del $^{15}N2$ (Kuzmenko, Kuznetsova, & Kuziakov, 1984)	68
Tabla 5-44. FFC-RNA Newton Sistema de bandas $a1\Pi g \rightarrow X1\Sigma g$ + del $^{15}N2$	68
Tabla 5-45. Datos Experimentales Sistema de bandas $\omega1\Delta u \rightarrow B3\Pi g$ del $^{15}N2$ (Kuzmenko, Kuznetsova, & Kuziakov, 1984)	69
Tabla 5-46. FFC-RNA Newton Sistema de bandas $\omega1\Delta u \rightarrow B3\Pi g$ del $^{15}N2$	69
Tabla 5-47. Datos Experimentales Sistema de bandas $B1\Pi u \rightarrow X1\Sigma g$ + del $^7Li2$ (Kuzmenko, Kuznetsova, & Kuziakov, 1984)	70
Tabla 5-48. FFC-RNA Newton Sistema de bandas $B1\Pi u \rightarrow X1\Sigma g$ + del $^7Li2$	70

Apéndice B: Figuras

Figura 2-1 Neurona de McCulloch-Pitts	4
Figura 2-2 Arquitectura de un Perceptrón multicapa	6
Figura 2-3 Neurona Biológica	6
Figura 2-4 Neurona Artificial	7
Figura 2-5 Perceptrón simple con función lineal = Modelo de regresión lineal	9
Figura 2-6 Perceptrón simple con función logística = Modelo de regresión logística	9
Figura 2-7 Perceptrón simple con función umbral = Análisis discriminante lineal	9
Figura 2-8 Perceptrón multicapa con función lineal en la salida = Modelo de regresión no lineal	10
Figura 2-9 Perceptrón multicapa con función logística = Función discriminante no lineal	11
Figura 2-10 Regla de Oja = Análisis de componentes principales	12
Figura 2-11 Red de función de base radial = Modelo de regresión kernel	12
Figura 2-12 Unidad de proceso bipolar	14
Figura 2-13 Modelado Neuronal.....	19
Figura 3-1. Área bajo la curva.....	20
Figura 3-2. Reglas de Cuadratura	21
Figura 3-3. Integración Adaptiva	24
Figura 4-1. Arquitectura de la Red Neuronal.....	26
Figura 4-2. Cambio de Variable.....	28
Figura 4-3. Arquitectura de la Red Neuronal Back-Propagation.....	29
Figura 4-4. Arquitectura de la Red Neuronal Factor Momentum.....	33
Figura 4-5. Arquitectura de la Red Neuronal Gradientes Conjugados	36
Figura 4-6. Arquitectura de la Red Neuronal Gradientes Conjugados Residuales	39
Figura 4-7. Arquitectura de la Red Neuronal Newton	40
Figura 4-8. Arquitectura de la Red Neuronal Newton Truncado.....	42
Figura 5-1. Aproximación de la función $F = \cos 201x \cos x$ con 201 nodos, una tolerancia de $1e-10$, aumentando el número de entrenamientos	43

Apéndice C: Programas en MatLab

En este apéndice se muestra el código fuente de todos los algoritmos que se programaron en MatLab. Los algoritmos para las seis Redes Neuronales Artificiales para cualquier función, el algoritmo para el cálculo de la función de onda para los FFC de Morse y los algoritmos de las RNAS y el cálculo de la función de onda para los FFC de Morse. Se describen a continuación:

Redes Neuronales Artificiales para cualquier función.

Algoritmo de la Red Neuronal Back-Propagation:

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA
%FACULTAD DE CIENCIAS DE LA COMPUTACIÓN
%Titulo: Integración Numérica con Redes Neuronales
%Función: RNA_BP (Red Neuronal Artificial Back-Propagation)
%Autor: Iván Christhofer Chaman García
%Fecha: Abril 2010
%Archivo: RNA_BP.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function RNA_BP()
tic;clc; clear all; format long e;
%entradas del algoritmo
tol=1E-10 ; efe=1; k=0; n=201;
%Intervalos para el ajuste de la funcion
a=0; b=pi; h=pi/n;x=a:h:b;
c=zeros();
for j=1:n+1
    for i=1:n+1
        c(j,i)=cos((j-1)*x(i));
    end
end
w=rand(n+1,1);
r1=(sum(c.^2));
r=sqrt(sum(r1));
eta=1.35/r^2;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%intervalos de integracion
a1=0; b1=pi; h1=(b1-a1)/n;
%Funciones Prueba Cambio de Variable Y=(a1+(b1-a1)/pi*x)
F=sin(5*(a1+(b1-a1)/pi*x));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Algoritmo RNA Back-Propagation
while efe > tol
    Y=c*w;
    error=F'-Y;
    w=w+eta*c'*error;
    efe=sqrt(sum(error.^2));
    efe=.5*efe^2;
    k=k+1;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Funcion Real
z=a1:h1:b1;

```

----continuación----

77

Algoritmo de la Red Neuronal Factor Momentum:

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA                                     %
%FACULTAD DE CIENCIAS DE LA COMPUTACIÓN                                       %
%Titulo: Integración Numérica con Redes Neuronales                           %
%Función: RNA_BP (Red Neuronal Artificial Factor Momentum)                   %
%Autor: Iván Christofer Chaman García                                         %
%Fecha: Abril 2010                                                            %
%Archivo: RNA_FM.m                                                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function RNA_FM()
    tic;clc; clear all; format long e;
%entradas del algoritmo
    tol=1E-10 ; efe=1; k=0; n=201;
%Intervalos para el ajuste de la funcion
    a=0; b=pi; h=pi/n;x=a:h:b;
    c=zeros();
    for j=1:n+1
        for i=1:n+1
            c(j,i)=cos((j-1)*x(i));
        end
    end
    w=rand(n+1,1);
    r1=(sum(c.^2));
    r=sqrt(sum(r1));
    eta=1.35/r^2;
    lamda=1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%intervalos de integracion
    a1=0; b1=pi; h1=(b1-a1)/n;
%Funciones Prueba Cambio de Variable Y=(a1+(b1-a1)/pi*x)
    F=sin(5*(a1+(b1-a1)/pi*x));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Algoritmo RNA Factor Momentum
    while efe > tol
        Y=c*w;
        error=F'-Y;
        w= w + eta*(lamda+1)*c'*error;
        efe=sqrt(sum(error.^2));
        efe=.5*efe^2;
        k=k+1;
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Funcion Real
    z=a1:h1:b1;
    Fz=sin(5*z);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Integral de la Funcion
    k1=(b1-a1);
    I=k1*w(1);
    for j=2:n+1
        c1=k1/((j-1)*pi);
        c2=sin((j-1)*pi);
        c3=c1*c2;
        c4=w(j)*c3;
    end
end

```

-----continuación-----

Algoritmo de la Red Neuronal Gradientes Conjugados:

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA                                     %
%FACULTAD DE CIENCIAS DE LA COMPUTACIÓN                                         %
%Titulo: Integración Numérica con Redes Neuronales                             %
%Función: RNA_BP (Red Neuronal Artificial Gradientes Conjugados)                %
%Autor: Iván Christhofer Chaman García                                          %
%Fecha: Abril 2010                                                              %
%Archivo: RNA_GC.m                                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function RNA_GC()
    tic;clc; clear all; format long e;
%entradas del algoritmo
    tol=1E-10 ; k=0; n=201;
%Intervalos para el ajuste de la funcion
    a=0; b=pi; h=pi/n;x=a:h:b;
    c=zeros();
    for j=1:n+1
        for i=1:n+1
            c(j,i)=cos((j-1)*x(i));
        end
    end
    w=rand(n+1,1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%intervalos de integracion
    a1=0; b1=pi; h1=(b1-a1)/n;
%Funciones Prueba Cambio de Variable Y=(a1+(b1-a1)/pi*x)
    F=sin(5*(a1+(b1-a1)/pi*x));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Algoritmo RNA Gradientes Conjugados
    Y=c*w;
    error=F'-Y;
    efe=0.5*(sum(error.^2));
    pk=c*error;
    gk=-pk;
    while efe > tol
        e1=sum(gk.^2);
        alfak=efe/e1 ;
        w=w+alfak*pk;
        Y=c*w;
        error=F'-Y;
        efe=0.5*(sum(error.^2));
        gk1=gk;
        gk=-c*error;
        c1=sum(gk.^2);
        c2=sum(gk1.^2);
        betak=c1/c2;
        pk=-gk+betak*pk;
        k=k+1;
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Funcion Real
    z=a1:h1:b1;
    Fz=sin(5*z);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

-----continuación-----

Algoritmo de la Red Neuronal Gradientes Conjugados Residuales:

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA                                     %
%FACULTAD DE CIENCIAS DE LA COMPUTACIÓN                                       %
%Titulo: Integración Numérica con Redes Neuronales                           %
%Función: RNA_BP (Red Neuronal Artificial Gradientes Conjugados               %
%          Residuales)                                                         %
%Autor: Iván Christofer Chaman García                                         %
%Fecha: Abril 2010                                                            %
%Archivo: RNA_GCR.m                                                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function RNA_GCR()
    tic;clc; clear all; format long e;
%entradas del algoritmo
    tol=1E-10 ; efe=1; k=0; n=201;
%Intervalos para el ajuste de la funcion
    a=0; b=pi; h=pi/n;x=a:h:b;
    c=zeros();
    for j=1:n+1
        for i=1:n+1
            c(j,i)=cos((j-1)*x(i));
        end
    end
    w=rand(n+1,1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%intervalos de integracion
    a1=0; b1=pi; h1=(b1-a1)/n;
%Funciones Prueba Cambio de Variable Y=(a1+(b1-a1)/pi*x)
    F=sin(5*(a1+(b1-a1)/pi*x));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Algoritmo RNA Gradientes Conjugados Residual
    Y=c*w;
    error=F'-Y;
    gk=-c*error;      %Gradiente gk
    hk=c*c';          %Hessiano
    pk=w;
    rk=-gk;            %r(0)=-g(k)
    dk=rk;             %d(0)=r(0)
    while efe > tol
        qk=hk*dk; e1=sum(rk.^2); d2=dk'*qk;
        alfak=e1/d2; %alfak=sqrt(sum(rk.^2))^2/d'*qk
        pk=pk+alfak*dk; %pk+1=pk+alfaj*dk
        rk1=rk;
        rk=rk-alfaj*qk; %rk+1=rk-alfak*qk
        c1=sum(rk.^2); c2=sum(rk1.^2);
        betak=c1/c2; %betak=sqrt(sum(rk+1.^2))^2/sqrt(sum(rk.^2))^2
        dk=rk+betak*dj; %dk+1=rk+1+betak*dk
        w=pk;
        Y=c*w;
        error=F'-Y;
        efe=.5*sum(error.^2) ;
        k=k+1;
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Funcion Real

```

-----continuación-----

Algoritmo de la Red Neuronal Newton:

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA                                %
%FACULTAD DE CIENCIAS DE LA COMPUTACIÓN                                  %
%Titulo: Integración Numérica con Redes Neuronales                      %
%Función: RNA_BP (Red Neuronal Artificial Newton)                       %
%Autor: Iván Christofer Chaman García                                   %
%Fecha: Abril 2010                                                       %
%Archivo: RNA_N.m                                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function RNA_N()
    tic; clc; clear all; format long e;
%entradas del algoritmo
    tol=1E-10; k=0; n=201;
%Intervalos para el ajuste de la funcion
    a=0; b=pi; h=pi/n;x=a:h:b;
    c=zeros();
    for j=1:n+1
        for i=1:n+1
            c(j,i)=cos((j-1)*x(i));
        end
    end
    w=rand(n+1,1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%intervalos de integracion
    a1=0; b1=pi; h1=(b1-a1)/n;
%Funciones Prueba Cambio de Variable Y=(a1+(b1-a1)/pi*x)
    F=sin(5*(a1+(b1-a1)/pi*x));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Algoritmo RNA Newton
    Y=c*w;
    error=F'-Y;
    efe=.5*sum(error.^2);
    gk=-c*error;
    hk=c*c' ;
    while efe > tol
        alfak=1;
        L=chol(hk,'lower');
        s=-L\gk;
        dk=L'\s;
        w=w+alfak*dk;
        Y=c*w;
        error=F'-Y;
        efe=.5*sum(error.^2) ;
        gk=-c*error;
        k=k+1;
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Funcion Real
    z=a1:h1:b1;
    Fz=sin(5*z);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Integral de la Funcion
    k1=(b1-a1);
    I=k1*w(1);

```

```

-----continuación-----

    for j=2:n+1
        c1=k1/((j-1)*pi);
        c2=sin((j-1)*pi);
        c3=c1*c2;
        c4=w(j)*c3;
        I=I+c4;
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Funcion aproximada en el intervalo
%Yx(Cambio de VVariable)  Yxx(sin Cambio de Variable)
    Yx=zeros();
    Yxx=zeros();
    for i=1:n+1
        Yx(i)=w(1);
        Yxx(i)=w(1);
        for j=2:n+1
            Yx(i)=Yx(i)+w(j)*cos((j-1)*pi/k1*(z(i)-a1));
            Yxx(i)=Yx(i)+w(j)*cos((j-1)*z(i));
        end
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Integral Real de las funciones
    real= (1-cos(50))/5;
%Simpson Adaptivo
    a_quad=quad('sin(5*z)',a1,b1);
%Error en las aproximaciones
%Error simpson adaptivo
    e_SA=abs(real-a_quad);
%Error 1
    e_I=abs(real-I);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Resultados obtenidos
    fprintf('k\tError_Funcion\tIntegral\tE_SA\tte_I\n');
    fprintf('%d\t%e\t%e\t%e\t%e\n',k,efe,I,e_SA,e_I);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Graficas
    hold on
    grid on
    plot(z,Fz,'g'); %Funcion Real
    plot(z,Yx,'b'); %Funcion Aproximada Cambio de Variable
    plot(z,Yxx,'r'); %Funciion Aproximada sin Cambio de VVariable
    toc;

```

Algoritmo de la Red Neuronal Newton Truncado:

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA                                     %
%FACULTAD DE CIENCIAS DE LA COMPUTACIÓN                                       %
%Titulo: Integración Numérica con Redes Neuronales                           %
%Función: RNA_BP (Red Neuronal Artificial Newton Truncado)                     %
%Autor: Iván Christofer Chaman García                                         %
%Fecha: Abril 2010                                                            %
%Archivo: RNA_NT.m                                                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function RNA_NT()
    tic;clc; clear all; format long e;
%entradas del algoritmo
    tol=1E-10 ; efe=1; k=0; n=201;
%Intervalos para el ajuste de la funcion
    a=0; b=pi; h=pi/n;x=a:h:b;c=zeros();
    for j=1:n+1
        for i=1:n+1
            c(j,i)=cos((j-1)*x(i));
        end
    end
    w=rand(n+1,1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%intervalos de integracion
    a1=0; b1=pi; h1=(b1-a1)/n;
%Funciones Prueba Cambio de Variable Y=(a1+(b1-a1)/pi*x)
    F=sin(5*(a1+(b1-a1)/pi*x));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Algoritmo RNA Newton Truncado
    Y=c*w; error=F'-Y;
    gk=-c*error; %Gradiente gk
    hk=c*c'; %Hessiano
    pj=seros();
    pj(n+1,1)=0; %p(0)=0
    dk=pj; rj=-gk; %r(0)=-g(k)
    dj=rj; %d(0)=r(0)
    deltax=sum(rj.^2); %delta(0)=sqrt(sum(r(0).^2))^2
    epsi=1e-10; k=0; j=0;
    while efe > tol %Ciclo Externo
        alfak=1; w=w+alfak*dk;
        while true %Cicli Interno
            qj=hk*dj; e1=sum(rj.^2); d2=dj'*qj; d1=epsi*deltaj ;
            if d2<=d1 %dj'*qj < e*deltaj
                if j==0; dk=-gk;
                else dk=pj;
                end
                break;
            end
            alfaj=e1/d2 ; %alfaj=sqrt(sum(rj.^2))^2/dj'*qj
            pj=pj+alfaj*dj; %pj+1=pj+alfaj*dj
            rj1=rj; rj=rj-alfaj*qj; %rj+1=rj-alfaj*qj
            a1=sqrt(sum(rj.^2)); a2=sqrt(sum(gk.^2));
            if a1<=epsi*a2 || j>n+1
                %sqrt(sum(rj.^2))<niu*sqrt(sum(gk.^2)) || j>max
                dk=pj;
            end
        end
    end
end

```

```

        break;
    end
    c1=sum(rj.^2); c2=sum(rj1.^2);
    betaj=c1/c2 ; %betaj=sqrt(sum(rj+1.^2))^2/sqrt(sum(rj.^2))^2
    dj=rj+betaj*dj; %dj+1=rj+1+betaj*dj
    j=j+1;
    if j==n
        dk=pj;
        break;
    end
end
Y=c*w; error=F'-Y; efe=.5*sum(error.^2); k=k+1;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Funcion Real
z=a1:h1:b1;Fz=sin(5*z);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Integral de la Funcion
k1=(b1-a1);I=k1*w(1);
for j=2:n+1
    c1=k1/((j-1)*pi);c2=sin((j-1)*pi);c3=c1*c2;c4=w(j)*c3; I=I+c4;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Funcion aproximada en el intervalo
%Yx(Cambio de VArivable) Yxx(sin Cambio de Variable)
Yx=zeros();Yxx=zeros();
for i=1:n+1
    Yx(i)=w(1);Yxx(i)=w(1);
    for j=2:n+1
        Yx(i)=Yx(i)+w(j)*cos((j-1)*pi/k1*(z(i)-a1));
        Yxx(i)=Yx(i)+w(j)*cos((j-1)*z(i));
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Integral Real de las funciones
real= (1-cos(50))/5;
%Simpson Adaptivo
a_quad=quad('sin(5*z)',a1,b1);
>Error en las aproximaciones
>Error simpson adaptivo
e_SA=abs(real-a_quad);
>Error 1
e_I=abs(real-I);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Resultados obtenidos
fprintf('k\tError_Funcion\tIntegral\tE_SA\tI\tI_SA\n');
fprintf('%d\t%e\t%e\t%e\t%e\n',k,efe,I,e_SA,e_I);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Graficas
hold on
grid on
plot(z,Fz,'g'); %Funcion Real
plot(z,Yx,'b'); %Funcion Aproximada Cambio de Variable
plot(z,Yxx,'r'); %Funcion Aproximada sin Cambio de VArivable
toc;

```

A continuación se muestra el código para el cálculo de la función de onda para los FFC de Morse:

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA                                     %
%FACULTAD DE CIENCIAS DE LA COMPUTACIÓN                                       %
%Titulo: Integración Numérica con Redes Neuronales                           %
%Función: FFC (Factores Franck-Condon)                                       %
%Autor: Iván Christofer Chaman García                                         %
%Fecha: Abril 2010                                                            %
%Archivo: FFC.m                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function FFC()
    tic; clc; clear all; format long e;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Inicio del cálculo de la funcion FFC
%Primer estado
    we(1)=887.2;
    wexe(1)=33.4;
    re(1)=1.864;
%Segundo estado
    we(2)=1262.5;
    wexe(2)=19.01;
    re(2)=1.664;
    mu=1.979260;
    v1=0; v2=0;
%Inicializacion de variables
    ka(1)=we(1)/wexe(1); ka(2)=we(2)/wexe(2);
    cte=1.21777513710683E-01;
    b(1)=cte*sqrt(4*wexe(1)*mu); b(2)=cte*sqrt(4*wexe(2)*mu);
    g(1)=gamma(ka(1)-1); g(2)=gamma(ka(2)-1);
%Generacion de Funcion de Onda para cada estado
%Funonda(1) del primer estado; constante de normalizacion
    cte_normalizacion=sqrt(b(1)/g(1));
    for i=0:v1-1,
        num=(ka(1)-2*i-3.)*(ka(1)-i-1); den=(i+1)*(ka(1)-2*i-1);
        cte_normalizacion=cte_normalizacion*sqrt(num/den);
    end
    Nv1=cte_normalizacion;
%funonda(2) del segundo estado; constante de normalizacion
    cte_normalizacion=sqrt(b(2)/g(2));
    for i=0:v2-1,
        num=(ka(2)-2*i-3.)*(ka(2)-i-1); den=(i+1)*(ka(2)-2*i-1);
        cte_normalizacion=cte_normalizacion*sqrt(num/den);
    end
    Nv2=cte_normalizacion;
%Generacion de la funcion a integrar que es funonda(v1)*funonda(2)
    r=0.4; n=201; lim_a=0.4; lim_b=2.5; h=(lim_b-lim_a)/n;
    y=zeros(); ri=zeros();
    for i=1:n+1, %Numero de Nodos
        ri(i)=r;
        %generacion de la funcion de onda del primer estado
        p=r-re(1); x=ka(1)*exp((-1)*b(1)*p);
        c2 = exp(((1)*x)/2); c3=(ka(1)-2*v1-1)/2.0;
        c3= x^c3;
        if v1 ~= 0,
            sum1 = 0;

```

```

-----continuación-----
for j = 1:v1,
%calculo del coeficiente binomial
    if ( v1 == j),
        coef_binomial = 1;
    else
        %(v1 >= j) & (j >= 0),
        coef_superior=gamma(v1+1);
        coef_inferior=gamma(j+1)*gamma((v1-j)+1);
        coef_binomial = coef_superior / coef_inferior;
    end
    aux1 = (-1)^j*coef_binomial * x^(v1 - j);
    aux = 1;
    for p = 1:j,
        aux = aux * (ka(1) - v1 - p);
    end;
    aux2=aux;
    aux1 = aux1 * aux2;
    sum1 = sum1 + aux1;
end
laguerre = x^v1 + sum1;
else
    laguerre = 1;
end
c4=laguerre;
%generacion de la funcion de onda del segundo estado
fun_onda1=Nv1*c2*c3*c4;
p=r-re(2); x=ka(2)*exp((-1)*b(2)*p);
c2=exp(((1)*x)/2); c3=x^((ka(2)-2*v2-1)/2.0);
if v2 ~= 0,
    sum1 = 0;
    for j = 1:v2,
        %calculo del coeficiente binomial
        if ( v2 == j),
            coef_binomial = 1;
        else
            %(v2 >= j) & (j >= 0),
            coef_superior = gamma(v2 + 1);
            coef_inferior=gamma(j+1)*gamma((v2-j)+1);
            coef_binomial = coef_superior / coef_inferior;
        end
        aux1 = (-1)^j*coef_binomial * x^(v2 - j);
        aux = 1;
        for p = 1:j,
            aux = aux * (ka(2) - v2 - p);
        end;
        aux2=aux;
        aux1 = aux1 * aux2;
        sum1 = sum1 + aux1;
    end
    laguerre = x^v2 + sum1;
else
    laguerre = 1;
end
c4=laguerre;
fun_onda2 = Nv2 * c2 * c3 * c4;
y(i)=fun_onda1*fun_onda2;
r=r+h;
end

```

```

-----continuación-----
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Grafica de la Funcion de Onda
    grid on
    hold on
    plot(ri,y,'g');
    toc;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function gamma=gamma(x)
%{Funcion que calcula la funcion gamma de x.
% El algoritmo usado para calcular la funci n gamma tiene una
% exactitud de hasta 10 lugares decimales. Se usa la f rmula de
% Stirlings para calcular el logartimo natural de la funcion gamma
% a la cual se le saca el exponencial para obtener el valor real
% de la funci n gamma.
% Nota : n! = gamma(n + 1)}
    if x < 7,
        fs = 1.0;
        z = x;
        while z < 7.0
            x=z ;
            fs = fs * z;
            z = z + 1.0;
        end
        x = x + 1.0;
        fs = -log(fs);
    else
        fs = 0;
    end
    z = (1.0/x)^2;
%{Uso de la formula de Stirlings}
    lga = fs;
    lga = lga+ (x - 0.5) * log(x) - x + 0.918938533204673;
    lgb=((-0.000595238095238*z + 0.000793650793651)*z -
0.002777777777778)*z + 0.083333333333333)/x;
    lga=lga+lgb;
    gamma =exp(lga);

```

A continuación se muestra el código para calcular los FFC para el Potencial de Morse usando las RNAS:

Algoritmo de la Red Neuronal Back-Propagation:

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA                                     %
%FACULTAD DE CIENCIAS DE LA COMPUTACIÓN                                       %
%Titulo: Integración Numérica con Redes Neuronales                           %
%Función: FFC (Factores Franck-Condon RNA Back-Propagation)                   %
%Autor: Iván Christhofer Chaman García                                         %
%Fecha: Abril 2010                                                             %
%Archivo: FFC_RNA_BP.m                                                         %
%Programa para calcular los Factores Franck-Condon de Morse                   %
%Usando RNA BackPropagation                                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function FFC_RNA_BP()
tic
clc; clear all
format long e
%entradas del algoritmo
tol=1E-6 ; efe=1; k=0; n=201;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Inicio del cálculo de la funcion FFC
for v1=0:10
    for v2=0:10
        a=0; b=pi; h=pi/n;x=a:h:b;
        c=zeros(n+1,n+1);
        for j=1:n+1
            for i=1:n+1
                c(j,i)=cos((j-1)*x(i));
            end
        end
        w=rand(n+1,1);
        r1=(sum(c.^2));
        r=sqrt(sum(r1));
        eta=1.35/r^2;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %Primer estado
        we(1)=1675.355; wexe(1)=13.433; re(1)=1.21252;
        %Segundo estado
        we(2)=1411.210; wexe(2)=12.921; re(2)=1.2864;
        mu=7.500053859;
        %Inicializacion de variables
        ka(1)=we(1)/wexe(1); ka(2)=we(2)/wexe(2);
        cte=1.21777513710683E-01;
        b(1)=cte*sqrt(4*wexe(1)*mu); b(2)=cte*sqrt(4*wexe(2)*mu);
        g(1)=gamma(ka(1)-1); g(2)=gamma(ka(2)-1);
        %Intervalos de la Funcion de Onda
        lim_a=0.4; lim_b=3.5; h=(lim_b-lim_a)/n;
        %Generacion de Funcion de Onda para cada estado
        %Funonda(1) del primer estado
        %constante de normalizacion
        cte_normalizacion=sqrt(b(1)/g(1));
        for i=0:v1-1,
            num=(ka(1)-2*i-3.)*(ka(1)-i-1);
            den=(i+1)*(ka(1)-2*i-1);

```

```

-----continuación-----
    cte_normalizacion=cte_normalizacion*sqrt(num/den);
end
Nv1=cte_normalizacion;
%funonda(2) del segundo estado
%constante de normalizacion
cte_normalizacion=sqrt(b(2)/g(2));
for i=0:v2-1,
    num=(ka(2)-2*i-3.)*(ka(2)-i-1);
    den=(i+1)*(ka(2)-2*i-1);
    cte_normalizacion=cte_normalizacion*sqrt(num/den);
end
Nv2=cte_normalizacion;
%Generacion de la funcion a integrar que es
%funonda(v1)*funonda(2)
r=lim_a; y=zeros(); ri=zeros();
for i=1:n+1, %Numero de Nodos
    ri(i)=r;
    p=r-re(1);
    x=ka(1) * exp( (-1) * b(1) * p);
    c2=exp((( -1)*x)/2);
    c3=(ka(1)-2*v1-1)/2; c3=x^c3;
    if v1 ~= 0,
        sum1 = 0;
        for j = 1:v1,
            %calculo del coeficiente binomial
            if ( v1 == j),
                coef_binomial = 1;
            else
                %(v1 >= j) & (j >= 0),
                coef_superior=gamma(v1+1);
                coef_inferior=gamma(j+1)*gamma((v1-j)+1);
                coef_binomial=coef_superior/coef_inferior;
            end
            aux1 = (-1)^j*coef_binomial * x^(v1 - j);
            aux = 1;
            for p = 1:j,
                aux = aux * (ka(1) - v1 - p);
            end;
            aux2=aux;
            aux1 = aux1 * aux2;
            sum1 = sum1 + aux1;
        end
        laguerre = x^v1 + sum1;
    else
        laguerre = 1;
    end
    c4=laguerre;
    fun_onda1 = Nv1 * c2 * c3 * c4;
    %generacion de la funcion de onda del segundo estado
    p=r-re(2);
    x=ka(2)*exp((-1)*b(2) * p);
    c2=exp((( -1)*x)/2);
    c3 = x^((ka(2)-2*v2-1)/2.0);
    if v2 ~= 0,
        sum1 = 0;
        for j = 1:v2,
            %calculo del coeficiente binomial

```

```

        ----continuación-----
        if ( v2 == j),
            coef_binomial = 1;
        else
            %(v2 >= j) & (j >= 0),
            coef_superior=gamma(v2+1);
            coef_inferior=gamma(j+1)*gamma((v2-j)+1);
            coef_binomial=coef_superior/coef_inferior;
        end
        aux1 = (-1)^j*coef_binomial * x^(v2 - j);
        aux = 1;
        for p = 1:j,
            aux=aux*(ka(2)-v2-p);
        end;
        aux2=aux;
        aux1 = aux1 * aux2;
        sum1 = sum1 + aux1;
    end
    laguerre = x^v2 + sum1;
else
    laguerre = 1;
end
c4=laguerre;
fun_onda2 = Nv2 * c2 * c3 * c4;
y(i)=fun_onda1*fun_onda2;
r=r+h;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%intervalos de integracion
a1=lim a; b1=lim b; F=y;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Algoritmo RNA Back-Propagation
while efe > tol
    Y=c*w; error=F'-Y;
    w=w+eta*c'*error;
    efe=sqrt(sum(error.^2)); efe=.5*efe^2;
    k=k+1;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Integral de la Funcion
k1=(b1-a1);
I=k1*w(1);
for j=2:n+1
    c1=k1/((j-1)*pi); c2=sin((j-1)*pi);
    c3=c1*c2; c4=w(j)*c3;
    I=I+c4;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Resultados obtenidos
fprintf('%1.4e\t',I^2);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end
fprintf('\n');
end
toc;

```

Algoritmo de la Red Neuronal Factor Momentum:

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA                                     %
%FACULTAD DE CIENCIAS DE LA COMPUTACIÓN                                       %
%Titulo: Integración Numérica con Redes Neuronales                           %
%Función: FFC (Factores Franck-Condon RNA Factor Momentum)                   %
%Autor: Iván Christhofer Chaman García                                       %
%Fecha: Abril 2010                                                            %
%Archivo: FFC_RNA_FM.m                                                        %
%Programa para calcular los Factores Franck-Condon de Morse                  %
%Usando RNA Factor Momentum                                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function FFC_RNA_FM()
tic
clc; clear all
format long e
%entradas del algoritmo
tol=1E-6 ; efe=1; k=0; n=201;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Incio del cálculo de la funcion FFC
for v1=0:10
    for v2=0:10
        a=0; b=pi; h=pi/n;x=a:h:b;
        c=zeros(n+1,n+1);
        for j=1:n+1
            for i=1:n+1
                c(j,i)=cos((j-1)*x(i));
            end
        end
        w=rand(n+1,1);
        r1=(sum(c.^2));
        r=sqrt(sum(r1));
        eta=1.35/r^2;
        lamda=1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %Primer estado
        we(1)=1675.355; wexe(1)=13.433; re(1)=1.21252;
        %Segundo estado
        we(2)=1411.210; wexe(2)=12.921; re(2)=1.2864;
        mu=7.500053859;
        %Inicializacion de variables
        ka(1)=we(1)/wexe(1); ka(2)=we(2)/wexe(2);
        cte=1.21777513710683E-01;
        b(1)=cte*sqrt(4*wexe(1)*mu); b(2)=cte*sqrt(4*wexe(2)*mu);
        g(1)=gamma(ka(1)-1); g(2)=gamma(ka(2)-1);
        %Intervalos de la Funcion de Onda
        lim_a=0.4; lim_b=3.5; h=(lim_b-lim_a)/n;
        %Generacion de Funcion de Onda para cada estado
        %Funonda(1) del primer estado
        %constante de normalizacion
        cte_normalizacion=sqrt(b(1)/g(1));
        for i=0:v1-1,
            num=(ka(1)-2*i-3.)*(ka(1)-i-1);
            den=(i+1)*(ka(1)-2*i-1);
            cte_normalizacion=cte_normalizacion*sqrt(num/den);

```

```

-----continuación-----
end
Nv1=cte_normalizacion;
%funonda(2) del segundo estado
%constante de normalizacion
cte_normalizacion=sqrt(b(2)/g(2));
for i=0:v2-1,
    num=(ka(2)-2*i-3.)*(ka(2)-i-1);
    den=(i+1)*(ka(2)-2*i-1);
    cte_normalizacion=cte_normalizacion*sqrt(num/den);
end
Nv2=cte_normalizacion;
%Generacion de la funcion a integrar que es
%funonda(v1)*funonda(2)
r=lim_a; y=zeros(); ri=zeros();
for i=1:n+1, %Numero de Nodos
    ri(i)=r;
    p=r-re(1);
    x=ka(1)*exp((-1)*b(1)*p);
    c2=exp((-1)*x)/2;
    c3=(ka(1)-2*v1-1)/2; c3=x^c3;
    if v1 ~= 0,
        sum1 = 0;
        for j = 1:v1,
            %calculo del coeficiente binomial
            if (v1 == j),
                coef_binomial = 1;
            else
                %(v1 >= j) & (j >= 0),
                coef_superior=gamma(v1+1);
                coef_inferior=gamma(j+1)*gamma((v1-j)+1);
                coef_binomial=coef_superior/coef_inferior;
            end
            aux1 = (-1)^j*coef_binomial*x^(v1-j);
            aux = 1;
            for p = 1:j,
                aux=aux*(ka(1)-v1-p);
            end;
            aux2=aux;
            aux1=aux1*aux2;
            sum1=sum1+aux1;
        end
        laguerre=x^v1+sum1;
    else
        laguerre=1;
    end
    c4=laguerre;
    fun_onda1=Nv1*c2*c3*c4;
    %generacion de la funcion de onda del segundo estado
    p=r-re(2);
    x=ka(2)*exp((-1)*b(2)*p);
    c2=exp((-1)*x)/2;
    c3 = x^((ka(2)-2*v2-1)/2.0);
    % c4 = laguerre(x,ka(1),v1);
    if v2 ~= 0,
        sum1 = 0;
        for j = 1:v2,
            %calculo del coeficiente binomial

```

```

-----continuación-----
if ( v2 == j),
    coef_binomial = 1;
else
    % (v2 >= j) & (j >= 0),
    coef_superior=gamma(v2+1);
    coef_inferior=gamma(j+1)*gamma((v2-j)+1);
    coef_binomial=coef_superior/coef_inferior;
end
aux1=(-1)^j*coef_binomial*x^(v2-j);
aux = 1;
for p = 1:j,
    aux = aux * (ka(2) - v2 - p);
end;
aux2=aux;
aux1 = aux1 * aux2;
sum1 = sum1 + aux1;
end
laguerre = x^v2 + sum1;
else
    laguerre = 1;
end
c4=laguerre;
fun_onda2 = Nv2 * c2 * c3 * c4;
y(i)=fun_onda1*fun_onda2;
r=r+h;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%intervalos de integracion
a1=lim a; b1=lim b; F=y;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Algoritmo RNA Factor Momentum
while efe > tol
    Y=c*w; error=F'-Y;
    w= w + eta*(lamda+1)*c'*error;
    efe=sqrt(sum(error.^2));    efe=.5*efe^2;
    k=k+1;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Integral de la Funcion
k1=(b1-a1);
I=k1*w(1);
for j=2:n+1
    c1=k1/((j-1)*pi);c2=sin((j-1)*pi);
    c3=c1*c2; c4=w(j)*c3;
    I=I+c4;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Resultados obtenidos
fprintf('%1.4e\t',I^2);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end
fprintf('\n');
end
toc;

```

Algoritmo de la Red Neuronal Gradientes Conjugados:

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA                                     %
%FACULTAD DE CIENCIAS DE LA COMPUTACIÓN                                       %
%Titulo: Integración Numérica con Redes Neuronales                           %
%Función: FFC (Factores Franck-Condon RNA Gradientes Conjugados)             %
%Autor: Iván Christhofer Chaman García                                         %
%Fecha: Abril 2010                                                             %
%Archivo: FFC_RNA_GC.m                                                         %
%Programa para calcular los Factores Franck-Condon de Morse                   %
%Usando RNA Gradientes Conjugados                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function FFC_RNA_GC()
tic
clc; clear all
format long e
%entradas del algoritmo
tol=1E-6 ; efe=1; k=0; n=201;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Incio del cálculo de la funcion FFC
for v1=0:10
    for v2=0:10
        a=0; b=pi; h=pi/n;x=a:h:b;
        c=zeros(n+1,n+1);
        for j=1:n+1
            for i=1:n+1
                c(j,i)=cos((j-1)*x(i));
            end
        end
        w=rand(n+1,1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %Primer estado
        we(1)=1675.355; wexe(1)=13.433; re(1)=1.21252;
        %Segundo estado
        we(2)=1411.210; wexe(2)=12.921; re(2)=1.2864;
        mu=7.500053859;
        %Inicializacion de variables
        ka(1)=we(1)/wexe(1); ka(2)=we(2)/wexe(2);
        cte=1.21777513710683E-01;
        b(1)=cte*sqrt(4*wexe(1)*mu); b(2)=cte*sqrt(4*wexe(2)*mu);
        g(1)=gamma(ka(1)-1); g(2)=gamma(ka(2)-1);
        %Intervalos de la Funcion de Onda
        lim_a=0.4; lim_b=3.5; h=(lim_b-lim_a)/n;
        %Generacion de Funcion de Onda para cada estado
        %Funonda(1) del primer estado
        %constante de normalizacion
        cte_normalizacion=sqrt(b(1)/g(1));
        for i=0:v1-1,
            num=(ka(1)-2*i-3.)*(ka(1)-i-1);
            den=(i+1)*(ka(1)-2*i-1);
            cte_normalizacion=cte_normalizacion*sqrt(num/den);
        end
        Nv1=cte_normalizacion;
        %funonda(2) del segundo estado
        %constante de normalizacion
    end
end

```

```

-----continuación-----
cte_normalizacion=sqrt(b(2)/g(2));
for i=0:v2-1,
    num=(ka(2)-2*i-3.)*(ka(2)-i-1);
    den=(i+1)*(ka(2)-2*i-1);
    cte_normalizacion=cte_normalizacion*sqrt(num/den);
end
Nv2=cte_normalizacion;
%Generacion de la funcion a integrar que es
%funonda(v1)*funonda(2)
r=lim_a; y=zeros(); ri=zeros();
for i=1:n+1, %Numero de Nodos
    ri(i)=r;
    p=r-re(1);
    x=ka(1) * exp( (-1) * b(1) * p);
    c2=exp((( -1)*x)/2);
    c3=(ka(1)-2*v1-1)/2; c3=x^c3;
    if v1 ~= 0,
        sum1 = 0;
        for j = 1:v1,
            %calculo del coeficiente binomial
            if ( v1 == j),
                coef_binomial = 1;
            else
                %(v1 >= j) & (j >= 0),
                coef_superior=gamma(v1+1);
                coef_inferior=gamma(j+1)*gamma((v1-j)+1);
                coef_binomial=coef_superior/coef_inferior;
            end
            aux1=(-1)^j*coef_binomial*x^(v1-j);
            aux = 1;
            for p = 1:j,
                aux = aux * (ka(1) - v1 - p);
            end;
            aux2=aux; aux1=aux1*aux2; sum1=sum1+aux1;
        end
        laguerre=x^v1+sum1;
    else
        laguerre=1;
    end
    c4=laguerre;
    fun_onda1=Nv1*c2*c3*c4;
    %generacion de la funcion de onda del segundo estado
    p=r-re(2);
    x=ka(2)*exp((-1)*b(2)*p);
    c2=exp((( -1)*x)/2);
    c3 = x^((ka(2)-2*v2-1)/2.0);
    if v2 ~= 0,
        sum1 = 0;
        for j = 1:v2,
            %calculo del coeficiente binomial
            if ( v2 == j),
                coef_binomial=1;
            else
                %(v2 >= j) & (j >= 0),
                coef_superior=gamma(v2 + 1);
                coef_inferior=gamma(j+1)*gamma((v2-j)+1);
                coef_binomial=coef_superior/coef_inferior;
            end

```

```

        -----continuación-----
        aux1=(-1)^j*coef_binomial*x^(v2-j);
        aux = 1;
        for p = 1:j,
            aux=aux*(ka(2)-v2-p);
        end;
        aux2=aux;aux1=aux1*aux2; sum1=sum1+aux1;
    end
    laguerre=x^v2+sum1;
else
    laguerre=1;
end
c4=laguerre;
fun_onda2 = Nv2 * c2 * c3 * c4;
y(i)=fun_onda1*fun_onda2;
r=r+h;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%intervalos de integracion
a1=lim a; b1=lim b; F=y;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Algoritmo RNA Gradientes Conjugados
Y=c*w;
error=F'-Y; efe=0.5*(sum(error.^2));
pk=c*error; gk=-pk;
while efe > tol
    e1=sum(gk.^2); alfak=efe/e1 ;
    w=w+alfak*pk;
    Y=c*w;
    error=F'-Y;
    efe=0.5*(sum(error.^2));
    gk1=gk; gk=-c*error;
    c1=sum(gk.^2); c2=sum(gk1.^2);
    betak=c1/c2;
    pk=-gk+betak*pk;
    k=k+1;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Integral de la Funcion
k1=(b1-a1);
I=k1*w(1);
for j=2:n+1
    c1=k1/((j-1)*pi); c2=sin((j-1)*pi);
    c3=c1*c2; c4=w(j)*c3;
    I=I+c4;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Resultados obtenidos
fprintf('%1.4e\t',I^2);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end
fprintf('\n');
end
toc;

```

Algoritmo de la Red Neuronal Gradientes Conjugados Residuales:

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA                                     %
%FACULTAD DE CIENCIAS DE LA COMPUTACIÓN                                       %
%Titulo: Integración Numérica con Redes Neuronales                           %
%Función: FFC (Factores Franck-Condon RNA Gradientes Conjugados Residuales) %
%                                           %
%Autor: Iván Christofer Chaman García                                         %
%Fecha: Abril 2010                                                            %
%Archivo: FFC_RNA_GCR.m                                                       %
%Programa para calcular los Factores Franck-Condon de Morse                   %
%Usando RNA Gradientes Conjugados Residuales                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function FFC_RNA_GCR()
tic
clc; clear all
format long e
%entradas del algoritmo
tol=1E-6 ; efe=1; k=0; n=201;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Inicio del cálculo de la funcion FFC
for v1=0:10
    for v2=0:10
        a=0; b=pi; h=pi/n;x=a:h:b;
        c=zeros(n+1,n+1);
        for j=1:n+1
            for i=1:n+1
                c(j,i)=cos((j-1)*x(i));
            end
        end
        w=rand(n+1,1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %Primer estado
        we(1)=1675.355; wexe(1)=13.433; re(1)=1.21252;
        %Segundo estado
        we(2)=1411.210; wexe(2)=12.921; re(2)=1.2864;
        mu=7.500053859;
        %Inicializacion de variables
        ka(1)=we(1)/wexe(1); ka(2)=we(2)/wexe(2);
        cte=1.21777513710683E-01;
        b(1)=cte*sqrt(4*wexe(1)*mu); b(2)=cte*sqrt(4*wexe(2)*mu);
        g(1)=gamma(ka(1)-1); g(2)=gamma(ka(2)-1);
        %Intervalos de la Funcion de Onda
        lim_a=0.4; lim_b=3.5; h=(lim_b-lim_a)/n;
        %Generacion de Funcion de Onda para cada estado
        %Funonda(1) del primer estado constante de normalizacion
        cte_normalizacion=sqrt(b(1)/g(1));
        for i=0:v1-1,
            num=(ka(1)-2*i-3.)*(ka(1)-i-1);
            den=(i+1)*(ka(1)-2*i-1);
            cte_normalizacion=cte_normalizacion*sqrt(num/den);
        end
        Nv1=cte_normalizacion;
        %funonda(2) del segundo estado constante de normalizacion
        cte_normalizacion=sqrt(b(2)/g(2));
    end
end

```

```

-----continuación-----
for i=0:v2-1,
    num=(ka(2)-2*i-3.)*(ka(2)-i-1);
    den=(i+1)*(ka(2)-2*i-1);
    cte_normalizacion=cte_normalizacion*sqrt(num/den);
end

-----continuación-----
Nv2=cte_normalizacion;
%Generacion de la funcion a integrar que es
%funonda(v1)*funonda(2)
r=lim_a; y=zeros(); ri=zeros();
for i=1:n+1, %Numero de Nodos
    ri(i)=r; p=r-re(1);
    x=ka(1)*exp((-1)*b(1)*p);
    c2=exp((-1)*x)/2; c3=(ka(1)-2*v1-1)/2; c3=x^c3;
    if v1 ~= 0,
        sum1 = 0;
        for j = 1:v1,
            %calculo del coeficiente binomial
            if ( v1 == j),
                coef_binomial = 1;
            else
                %(v1 >= j) & (j >= 0),
                coef_superior=gamma(v1+1);
                coef_inferior=gamma(j+1)*gamma((v1-j)+1);
                coef_binomial=coef_superior/coef_inferior;
            end
            aux1 = (-1)^j*coef_binomial * x^(v1 - j);
            aux = 1;
            for p = 1:j,
                aux=aux*(ka(1)-v1-p);
            end;
            aux2=aux;aux1=aux1*aux2;
            sum1=sum1+aux1;
        end
        laguerre=x^v1+sum1;
    else
        laguerre=1;
    end
    c4=laguerre;
    fun_onda1 = Nv1 * c2 * c3 * c4;
    %generacion de la funcion de onda del segundo estado
    p=r-re(2); x=ka(2)*exp((-1)*b(2) * p);
    c2=exp((-1)*x)/2; c3 = x^((ka(2)-2*v2-1)/2.0);
    % c4 = laguerre(x,ka(1),v1);
    if v2 ~= 0,
        sum1 = 0;
        for j = 1:v2,
            %calculo del coeficiente binomial
            if ( v2 == j),
                coef_binomial = 1;
            else
                %(v2 >= j) & (j >= 0),
                coef_superior=gamma(v2+1);
                coef_inferior=gamma(j+1)*gamma((v2-j)+1);
                coef_binomial=coef_superior/coef_inferior;
            end
            aux1=(-1)^j*coef_binomial*x^(v2-j);
            aux=1;

```

```

        ----continuación-----
        for p=1:j,
            aux=aux*(ka(2)-v2-p);
        end;
        aux2=aux; aux1=aux1*aux2;
        sum1 = sum1 + aux1;
    end
    laguerre = x^v2 + sum1;
else
    laguerre = 1;
end
c4=laguerre;
fun_onda2 = Nv2 * c2 * c3 * c4;
y(i)=fun_onda1*fun_onda2;
r=r+h;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%intervalos de integracion
a1=lim_a; b1=lim_b; F=y;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Algoritmo RNA Gradientes Conjugados Residuales
Y=c*w; error=F'-Y;
gk=-c*error; hk=c*c'; pk=w; rk=-gk; dk=rk;
while efe > tol
    qk=hk*dk;
    e1=sum(rk.^2); d2=dk'*qk;
    alfabak=e1/d2; %alfaj=sqrt(sum(rj.^2))^2/dj'*qj
    pk=pk+alfabak*dk; %pj+1=pj+alfaj*dj
    rk1=rk; rk=rk-alfabak*qk; %rj+1=rj-alfaj*qj
    c1=sum(rk.^2); c2=sum(rk1.^2);
    betak=c1/c2;
    %betaj=sqrt(sum(rj+1.^2))^2/sqrt(sum(rj.^2))^2
    dk=rk+betak*dk; %dj+1=rj+1+betaj*dj
    w=pk;
    Y=c*w; error=F'-Y;
    efe=.5*sum(error.^2);
    k=k+1;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Integral de la Funcion
k1=(b1-a1);
I=k1*w(1);
for j=2:n+1
    c1=k1/((j-1)*pi); c2=sin((j-1)*pi);
    c3=c1*c2; c4=w(j)*c3;
    I=I+c4;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Resultados obtenidos
fprintf('%1.4e\t',I^2);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end
fprintf('\n');
end
toc;

```

Algoritmo de la Red Neuronal Newton:

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA                                     %
%FACULTAD DE CIENCIAS DE LA COMPUTACIÓN                                       %
%Titulo: Integración Numérica con Redes Neuronales                           %
%Función: FFC (Factores Franck-Condon Newton)                                %
%Autor: Iván Christhofer Chaman García                                       %
%Fecha: Abril 2010                                                            %
%Archivo: FFC_RNA_N.m                                                         %
%Programa para calcular los Factores Franck-Condon de Morse                  %
%Usando RNA Gradientes Newton                                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function FFC_RNA_N()
tic; clc; clear all; format long e;
%entradas del algoritmo
tol=1E-10; k=0; n=201;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Inicio del cálculo de la funcion FFC
for v1=0:10
    for v2=0:10
        a=0; b=pi; h=pi/n; x=a:h:b;
        c=zeros(n+1,n+1);
        for j=1:n+1
            for i=1:n+1
                c(j,i)=cos((j-1)*x(i));
            end
        end
        w=rand(n+1,1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %Primer estado
        we(1)=1432.687; wexe(1)=13.95; re(1)=1.22675;
        %Segundo estado
        we(2)=1580.361; wexe(2)=12.073; re(2)=1.20740;
        mu=8.0;
        ka(1)=we(1)/wexe(1); ka(2)=we(2)/wexe(2);
        cte=1.21777513710683E-01;
        b(1)=cte*sqrt(4*wexe(1)*mu); b(2)=cte*sqrt(4*wexe(2)*mu);
        g(1)=gamma(ka(1)-1); g(2)=gamma(ka(2)-1);
        %Intervalos de la Funcion de Onda
        lim_a=0.4; lim_b=3.5; h=(lim_b-lim_a)/n;
        %Generacion de Funcion de Onda para cada estado
        %Funonda(1) del primer estado
        %constante de normalizacion
        cte_normalizacion=sqrt(b(1)/g(1));
        for i=0:v1-1,
            num=(ka(1)-2*i-3.)*(ka(1)-i-1);
            den=(i+1)*(ka(1)-2*i-1);
            cte_normalizacion=cte_normalizacion*sqrt(num/den);
        end
        Nv1=cte_normalizacion;
        %funonda(2) del segundo estado
        %constante de normalizacion
        cte_normalizacion=sqrt(b(2)/g(2));
        for i=0:v2-1,
            num=(ka(2)-2*i-3.)*(ka(2)-i-1);

```

```

-----continuación-----
den=(i+1)*(ka(2)-2*i-1);
cte_normalizacion=cte_normalizacion*sqrt(num/den);
end
Nv2=cte_normalizacion;
%Generacion de la funcion a integrar que es
%funonda(v1)*funonda(2)
r=lim_a; y=zeros(); ri=zeros();
for i=1:n+1, %Numero de Nodos
    ri(i)=r;
    p=r-re(1);
    x=ka(1)*exp((-1)*b(1)*p);
    c2=exp((-1)*x)/2;
    c3=(ka(1)-2*v1-1)/2;
    c3=x^c3;
    if v1 ~= 0,
        sum1 = 0;
        for j = 1:v1,
            %calculo del coeficiente binomial
            if ( v1 == j),
                coef_binomial = 1;
            else
                %(v1 >= j) & (j >= 0),
                coef_superior=gamma(v1+1);
                coef_inferior=gamma(j+1)*gamma((v1-j)+1);
                coef_binomial=coef_superior/coef_inferior;
            end
            aux1=(-1)^j*coef_binomial*x^(v1-j);
            aux = 1;
            for p = 1:j,
                aux = aux * (ka(1) - v1 - p);
            end;
            aux2=aux;
            aux1 = aux1 * aux2;
            sum1 = sum1 + aux1;
        end
        laguerre = x^v1 + sum1;
    else
        laguerre = 1;
    end
    c4=laguerre;
    fun_onda1 = Nv1 * c2 * c3 * c4;
    %generacion de la funcion de onda del segundo estado
    p=r-re(2);
    x=ka(2)*exp((-1)*b(2)*p);
    c2=exp((-1)*x)/2;
    c3 = x^((ka(2)-2*v2-1)/2.0);
    if v2 ~= 0,
        sum1 = 0;
        for j = 1:v2,
            %calculo del coeficiente binomial
            if ( v2 == j),
                coef_binomial = 1;
            else
                %(v2 >= j) & (j >= 0),
                coef_superior=gamma(v2+1);
                coef_inferior=gamma(j+1)*gamma((v2-j)+1);
                coef_binomial=coef_superior/coef_inferior;
            end

```

```

        -----continuación-----
        aux1 = (-1)^j*coef_binomial * x^(v2 - j);
        aux = 1;
        for p = 1:j,
            aux = aux * (ka(2) - v2 - p);
        end;
        aux2=aux;
        aux1 = aux1 * aux2;
        sum1 = sum1 + aux1;
    end
    laguerre = x^v2 + sum1;
else
    laguerre = 1;
end
c4=laguerre;
fun_onda2 = Nv2 * c2 * c3 * c4;
y(i)=fun_onda1*fun_onda2;
r=r+h;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%intervalos de integracion
a1=lim_a; b1=lim_b; F=y;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Algoritmo RNA Newton
Y=c*w;
error=F'-Y; efe=.5*sum(error.^2);
gk=-c*error; hk=c*c' ;
while efe > tol
    alfak=1;
    L=chol(hk, 'lower');
    s=-L\gk; dk=L'\s;
    w=w+alfak*dk;
    Y=c*w;
    error=F'-Y; efe=.5*sum(error.^2) ;
    gk=-c*error;
    k=k+1;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Integral de la Funcion
k1=(b1-a1);
I=k1*w(1);
for j=2:n+1
    c1=k1/((j-1)*pi); c2=sin((j-1)*pi);
    c3=c1*c2; c4=w(j)*c3;
    I=I+c4;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Resultados obtenidos
fprintf('%1.4e\t', I^2);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end
fprintf('\n');
end
toc;

```

Algoritmo de la Red Neuronal Newton Truncado:

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA                                     %
%FACULTAD DE CIENCIAS DE LA COMPUTACIÓN                                       %
%Titulo: Integración Numérica con Redes Neuronales                           %
%Función: FFC (Factores Franck-Condon Newton Truncado)                       %
%Autor: Iván Christhofer Chaman García                                         %
%Fecha: Abril 2010                                                             %
%Archivo: FFC_RNA_NT.m                                                         %
%Programa para calcular los Factores Franck-Condon de Morse                   %
%Usando RNA Gradientes Newton Truncado                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function FFC_RNA_FM()
tic
clc; clear all
format long e
%entradas del algoritmo
tol=1E-6 ; efe=1; k=0; n=201;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Incio del cálculo de la funcion FFC
for v1=0:10
    for v2=0:10
        a=0; b=pi; h=pi/n;x=a:h:b;
        c=zeros(n+1,n+1);
        for j=1:n+1
            for i=1:n+1
                c(j,i)=cos((j-1)*x(i));
            end
        end
        w=rand(n+1,1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %Primer estado
        we(1)=1675.355; wexe(1)=13.433; re(1)=1.21252;
        %Segundo estado
        we(2)=1411.210; wexe(2)=12.921; re(2)=1.2864;
        mu=7.500053859;
        %Inicializacion de variables
        ka(1)=we(1)/wexe(1); ka(2)=we(2)/wexe(2);
        cte=1.21777513710683E-01;
        b(1)=cte*sqrt(4*wexe(1)*mu); b(2)=cte*sqrt(4*wexe(2)*mu);
        g(1)=gamma(ka(1)-1); g(2)=gamma(ka(2)-1);
        %Intervalos de la Funcion de Onda
        lim_a=0.4; lim_b=3.5; h=(lim_b-lim_a)/n;
        %Generacion de Funcion de Onda para cada estado
        %Funonda(1) del primer estado constante de normalizacion
        cte_normalizacion=sqrt(b(1)/g(1));
        for i=0:v1-1,
            num=(ka(1)-2*i-3.)*(ka(1)-i-1);
            den=(i+1)*(ka(1)-2*i-1);
            cte_normalizacion=cte_normalizacion*sqrt(num/den);
        end
        Nv1=cte_normalizacion;
        %funonda(2) del segundo estado constante de normalizacion
        cte_normalizacion=sqrt(b(2)/g(2));
        for i=0:v2-1,

```

```

-----continuación-----
num=(ka(2)-2*i-3.)*(ka(2)-i-1);
den=(i+1)*(ka(2)-2*i-1);
cte_normalizacion=cte_normalizacion*sqrt(num/den);
end
Nv2=cte_normalizacion;
%Generacion de la funcion a integrar que es
%funonda(v1)*funonda(2)
r=lim_a; y=zeros(); ri=zeros();
for i=1:n+1, %Numero de Nodos
    ri(i)=r;
    p=r-re(1);
    x=ka(1) * exp( (-1) * b(1) * p);
    c2=exp((( -1)*x)/2);
    c3=(ka(1)-2*v1-1)/2; c3=x^c3;
    if v1 ~= 0,
        sum1=0;
        for j = 1:v1,
            %calculo del coeficiente binomial
            if (v1==j),
                coef_binomial=1;
            else
                %(v1 >= j) & (j >= 0),
                coef_superior=gamma(v1+1);
                coef_inferior=gamma(j+1)*gamma((v1-j)+1);
                coef_binomial=coef_superior/coef_inferior;
            end
            aux1=(-1)^j*coef_binomial*x^(v1-j);
            aux=1;
            for p = 1:j,
                aux=aux*(ka(1)-v1-p);
            end;
            aux2=aux;
            aux1=aux1*aux2;
            sum1=sum1+aux1;
        end
        laguerre=x^v1+sum1;
    else
        laguerre=1;
    end
    c4=laguerre;
    fun_onda1 = Nv1 * c2 * c3 * c4;
    %generacion de la funcion de onda del segundo estado
    p=r-re(2);
    x=ka(2)*exp((-1)*b(2) * p);
    c2=exp((( -1)*x)/2);
    c3=x^((ka(2)-2*v2-1)/2.0);
    if v2 ~= 0,
        sum1 = 0;
        for j = 1:v2,
            %calculo del coeficiente binomial
            if ( v2 == j),
                coef_binomial = 1;
            else
                %(v2 >= j) & (j >= 0),
                coef_superior=gamma(v2+1);
                coef_inferior=gamma(j+1)*gamma((v2-j)+1);
                coef_binomial=coef_superior/coef_inferior;
            end

```

```

        -----continuación-----
        aux1=(-1)^j*coef_binomial*x^(v2-j);
        aux = 1;
        for p = 1:j,
            aux=aux*(ka(2)-v2-p);
        end;
        aux2=aux;
        aux1=aux1*aux2;
        sum1=sum1+aux1;
    end
    laguerre=x^v2+sum1;
else
    laguerre=1;
end
c4=laguerre;
fun_onda2=Nv2*c2*c3*c4;
y(i)=fun_onda1*fun_onda2;
r=r+h;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%intervalos de integracion
a1=lim_a; b1=lim_b; F=y; pj=zeros();
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Algoritmo RNA Newton Truncado
Y=c*w;
error=F'-Y;
gk=-c*error;      %Gradiente gk
hk=c*c';          %Hessiano
pj(n+1,1)=0 ;     %p(0)=0
dk=pj;
rj=-gk;            %r(0)=-g(k)
dj=rj;             %d(0)=r(0)
deltaj=sum(rj.^2); %delta(0)=sqrt(sum(r(0).^2))^2
epsi=1e-10;
k=0;
j=0;
while efe > tol
    alfak=1 ;
    w=w+alfak*dk;
    while true
        qj=hk*dj;
        e1=sum(rj.^2);
        d2=dj'*qj ;
        d1=epsi*deltaj ;
        if d2<=d1      %dj'*qj < e*deltaj
            if j==0      %j=0
                dk=-gk;
            else          %j>0
                dk=pj;
            end
            break;
        end
        alfaj=e1/d2 ;      %alfaj=sqrt(sum(rj.^2))^2/dj'*qj
        pj=pj+alfaj*dj; %pj+1=pj+alfaj*dj
        rj1=rj;
        rj=rj-alfaj*qj; %rj+1=rj-alfaj*qj
        all=sqrt(sum(rj.^2));
    end
end

```

```

        -----continuación-----
        al2=sqrt(sum(gk.^2)) ;
        %if j>n+1
        if all1<=epsi*al2 || j>n+1
        %sqrt(sum(rj.^2))<niu*sqrt(sum(gk.^2)) || j>max
            dk=pj;
            break;
        end
        c1=sum(rj.^2);
        c2=sum(rj1.^2);
        betaj=c1/c2 ;
        %betaj=sqrt(sum(rj+1.^2))^2/sqrt(sum(rj.^2))^2
        dj=rj+betaj*dj; %dj+1=rj+1+betaj*dj
        j=j+1;
        if j==n
            dk=pj;
            break;
        end
    end
    Y=c*w;
    error=F'-Y;
    efe=.5*sum(error.^2);
    k=k+1;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Integral de la Funcion
    k1=(b1-a1);
    I=k1*w(1);
    for j=2:n+1
        c1=k1/((j-1)*pi);
        c2=sin((j-1)*pi) ;
        c3=c1*c2;
        c4=w(j)*c3;
        I=I+c4;
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Resultados obtenidos
    fprintf('%1.4e\t',I^2);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end
fprintf('\n');

end
toc;

```

Apéndice D: Interfaz y Manual de Usuario en Java

Introducción

La aplicación RNAS permite la aproximación numérica de la integral definida de cualquier función y el cálculo de los Factores Franck-Condon (FFC) utilizando el potencial de Morse; ofrece seis algoritmos de Redes Neuronales Artificiales Supervisadas para este cálculo.

Esta aplicación muestra los resultados en pantalla de las operaciones realizadas para cualquier función, para el cálculo de los FFC se muestran los resultados de dos formas, la primera se muestra en pantalla y la segunda en un archivo de texto.

Requerimientos Básicos

Sistema Operativo

Tipo de ordenador	Monoprocesador ACPI de PC x86
Sistema operativo	Windows 7, Vista, Windows XP, Windows 2003, Windows 2000, Linux

Versión de Java

Java versión "1.6.0_16"
Java(TM) SE Runtime Environment (build 1.6.0_16-b01)
Java HotSpot(TM) Client VM (build 14.2-b01, mixed mode)

Equipo Personal

Tipo de procesador	Intel Pentium IV, 2000 MHz (20 x 100)
Memoria del Sistema	1024 MB (PC3200 DDR SDRAM)
Tarjeta gráfica	NVIDIA GeForce4 MX 420 (Microsoft Corporation) (64 MB)
Monitor	HP Pavilion MX70 [17" CRT] (THLFQ02497)
Disco duro	Maxtor 6L160P0 (80 GB, 7200 RPM, Ultra-ATA/133)
Teclado	Dispositivo de teclado HID
Ratón	LuxeMate 600 Laser

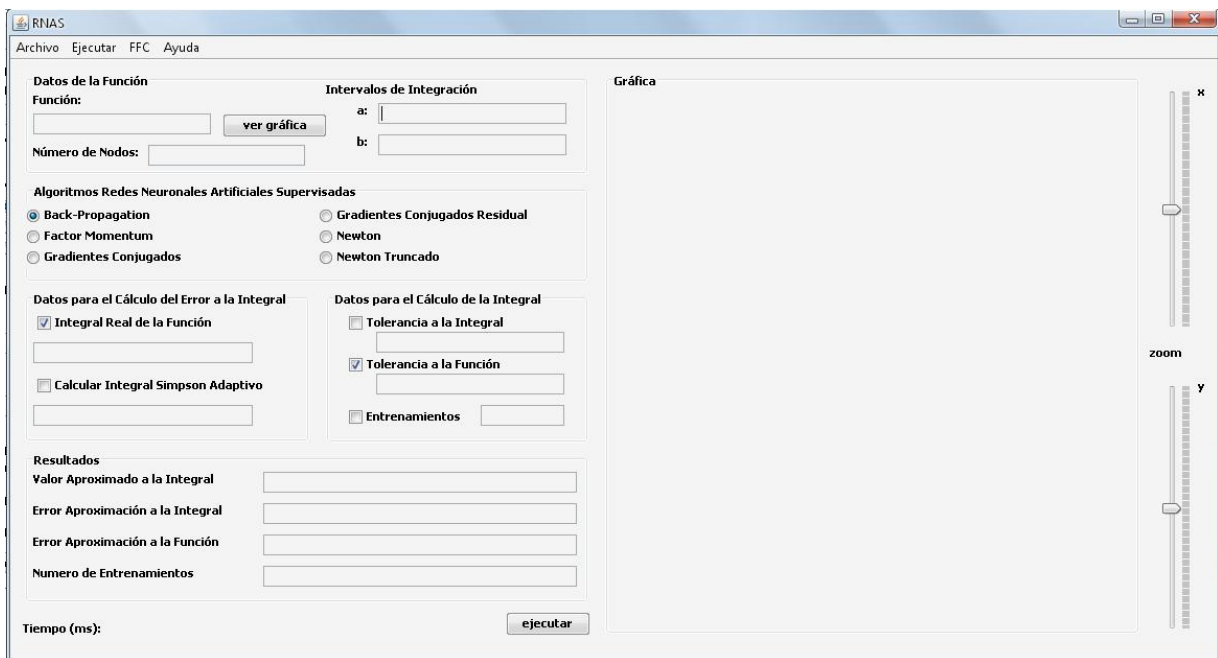
Los archivos necesarios para el funcionamiento del Sistema son: el archivo ejecutable RNAS.jar en lenguaje java, y los archivos appframework-1.0.3.jar, beansbinding-1.2.1.jar, swing-worker-1.1.jar que se encuentran dentro de la carpeta lib.

Guía rápida para Iniciar

Se creó una aplicación para la aproximación numérica de la integral definida de cualquier función y el cálculo de los FFC utilizando el potencial de Morse.

Pantalla Principal

Esta es la ventana principal para el ingreso de los datos necesarios para el cálculo de la integral de definida y la visualización de los resultados obtenidos.



Pantalla Inicial

La ventana principal se divide en 5 partes las cuales son:

Datos de la Función:

Se requieren los siguientes datos:

- Función.
- Intervalos de Integración de la función
- Número de nodos para la función

This is a close-up of the top-left portion of the application window. It shows the 'Datos de la Función' section with a 'Función:' label and an empty text input field. Below it is the 'Número de Nodos:' label and another empty text input field. To the right of the 'Función' input is a 'ver gráfica' button. The 'Intervalos de Integración' section is to the right, with 'a:' and 'b:' labels and empty text input fields.

Datos de la Función

Para leer la función se utiliza un 'Analizador Sintáctico', es decir una clase que nos permita leer y evaluar o analizar la función como una expresión matemática. El programa se encarga de traducir la expresión matemática que digita el usuario (en notación infija) en una expresión más simple, esta segunda expresión se dice que está en notación postfija; esta forma de escribir una expresión matemática tiene la ventaja que se evalúa de forma lineal por lo que es más sencillo evaluarlo.

Operador	Prioridad
$+, -$	0
$*, /, \%$	1
$^$	2

Para evaluar una expresión matemática se deben seguir las siguientes reglas:

1. Primero se evalúan las potencias.
2. A continuación siguen las multiplicaciones, las divisiones y el resto de la división entera (%).
3. Por último se realizan las sumas y las restas.
4. Si hay paréntesis, se hace primero la expresión que está dentro del paréntesis interno.

Una expresión simple: como primer ejemplo se traduce una expresión simple como $3 * x + 4$, la prioridad indica que primero se tiene que hacer la multiplicación y luego la suma; para esto, a función le asigna prioridad 0 a la suma y 1 a la multiplicación y se dará la regla que una prioridad inferior saca cualquier operación con prioridad igual o superior a ella.

Dado esto, se pueden notar que:

1. Los operadores "+" y "-" son los que tienen menor prioridad (0), por lo que siempre pierden prioridad con todos los operadores precedentes.
2. Los operadores "*", "/" y "%" pierden prioridad con la potencia "^" y a ellos mismos.
3. El operador "^" es el que tiene mayor prioridad, no pierde prioridad frente a otro operador, ni siquiera a sí mismo ya que la prioridad para realizar las potencias es de derecha a izquierda (contrario a todas las demás operaciones matemáticas), es decir: $2^3^4 = 2^(3^4)$.

Una expresión con funciones: Para una expresión con más elementos, por ejemplo: $1 + 3 * \tan(2 * (1 + x) - 1)$, aquí no hay problema con la función prioridad porque a tan le asignaría un -1, donde se agrega el paréntesis de apertura hasta un cierre de paréntesis.

En el programa se admite lo siguiente: La variable permitida es x . La expresión puede contener las constantes π y e . Los operadores válidos de la expresión son:

OPERACIÓN	OPERADOR	OPERACIÓN	OPERADOR
Suma	+	Arcocoseno	<i>acos()</i>
Resta	−	Arcotangente	<i>atan()</i>
Multiplicación	*	Arcosecante	<i>asec()</i>
División	/	Arcocosecante	<i>acsc()</i>
Potencia	^	Arcocotangente	<i>acot()</i>
Módulo	%	Seno hiperbólico	<i>sinh()</i>
Paréntesis	()	Coseno hiperbólico	<i>cosh()</i>
Logaritmo (base natural)	<i>ln()</i>	Tangente hiperbólica	<i>tanh()</i>
Logaritmo (base 10)	<i>log()</i>	Secante hiperbólica	<i>sech()</i>
Valor Absoluto	<i>abs()</i>	Cosecante hiperbólica	<i>csch()</i>
Número Aleatorio	<i>rnd()</i>	Cotangente hiperbólica	<i>coth()</i>
Seno	<i>sin()</i>	Raices cuadradas	<i>sqr()</i>
Coseno	<i>cos()</i>	Arcoseno hiperbólico	<i>asinh()</i>
Tangente	<i>tan()</i>	Arcocoseno hiperbólico	<i>acosh()</i>
Secante	<i>sec()</i>	Arcotangente hiperbólica	<i>atanh()</i>
Cosecante	<i>csc()</i>	Arcosecante hiperbólica	<i>asech()</i>
Cotangente	<i>cot()</i>	Arcocosecante hiperbólica	<i>acsch()</i>
Signo	<i>sgn()</i>	Arcocotangente hiperbólica	<i>acoth()</i>
Arcoseno	<i>asin()</i>	Redondeo	<i>round()</i>

De esta forma se deben cumplir las siguientes reglas:

- Si no se había traducido nada, entonces se puede admitir cualquier cosa menos (+ * / ^ %).
- Si lo anterior fue un número puede seguir cualquier cosa.
- Si lo anterior fue un operador puede seguir cualquier cosa menos otro operador (excepción de −).
- Si lo anterior fue un paréntesis de apertura puede seguir cualquier cosa menos (+ * / ^ %)
- Si lo anterior fue un cierre de paréntesis debe seguir un operador, otro paréntesis de cierre..

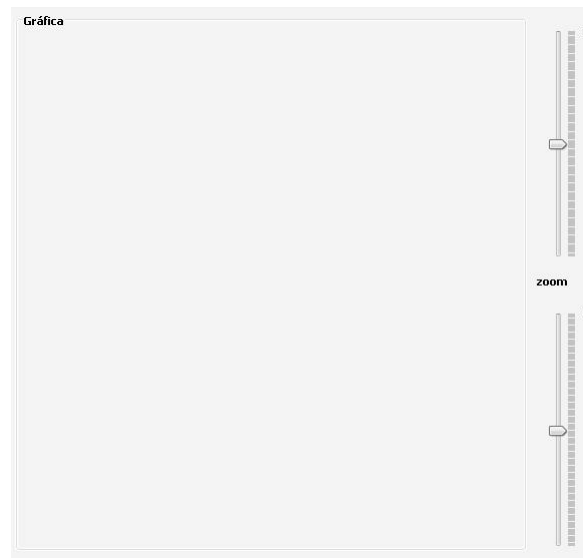
Algunos ejemplos de expresiones válidas son:

- $x + \cos(3) * \tan(x^{(2 * \pi * x - 1)}) / \cos(1/2)$
- $\cosh(x) + \text{abs}(1 - x^2) \% 3$
- $((1 - \cos(x))^5) * \sin(5 * x)$
- $\sin(5 * \cot(x)) * \sin(2 * x)$

En caso de que haya un error en la definición de la función se manda un mensaje de error. Los intervalos de integración son cualquier número real, es decir $(a, b) \in \mathbb{R}$, donde b no debe ser menor que a ($b > a$). Los números de nodos son cualquier número natural mayor o igual a 5 ($n \geq 5 \in \mathbb{N}$).

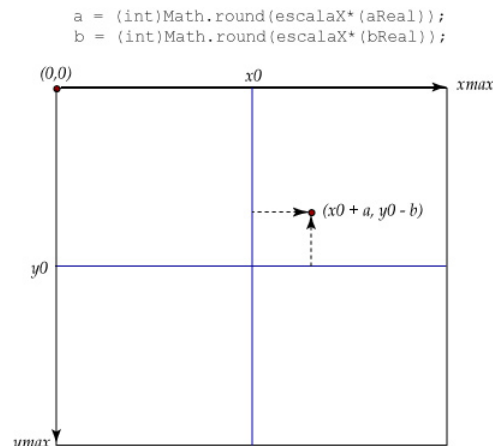
Gráfica

La implementación de un graficador de funciones de una variable real dentro de la interfaz gráfica. Se tiene un método de graficación y se utiliza el ratón para interactuar de manera adecuada con el usuario.



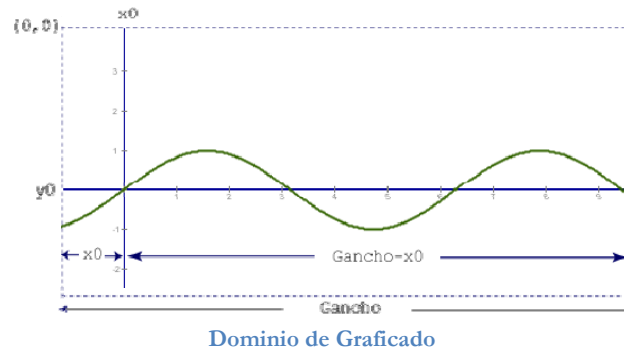
Área de Graficado

Ahora que podemos leer funciones y evaluarlas, se implementó el método que construye el gráfico de la función. Se tiene un factor de escala para el eje $X = 100$ pixeles y el eje $Y = 100$ pixeles. Para manejar la conversión entre coordenadas en números reales y coordenadas de pantalla, se maneja un sistema de coordenadas real (a_{Real}, b_{Real}) con su representación en coordenadas de pantalla (a, b) respecto al origen (x_0, y_0) .

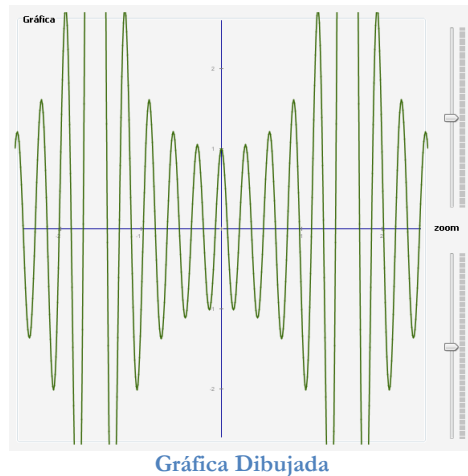


Escala de Graficado

Para dibujar el gráfico de la función necesitamos un dominio $[xmin, xmax]$ (definidos dentro de la aplicación) y un conjunto de pares ordenados $\{(x, f(x))\}$ (dados por la función) en coordenadas de pantalla. En el dominio de graficación, se pensó en solo una parte de la aplicación como todo el dominio de pantalla, tenemos una región de graficación.

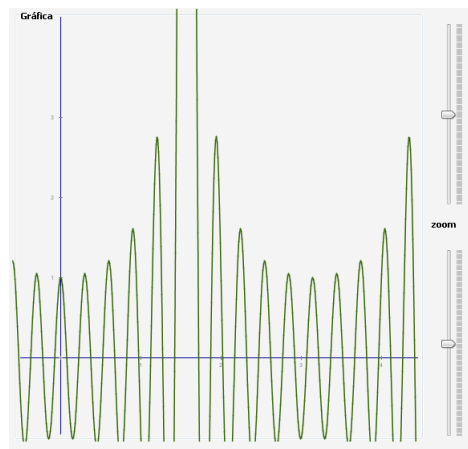


El usuario presiona el botón ver grafica o da clic en la región de graficación, se debe dibujar el gráfico.



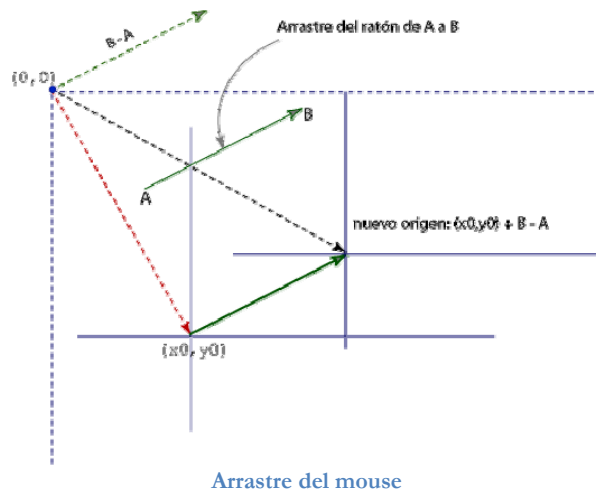
Gráfica Dibujada

Si el usuario arrastra el mouse sobre la zona gráfica, se debe desplazar el gráfico.

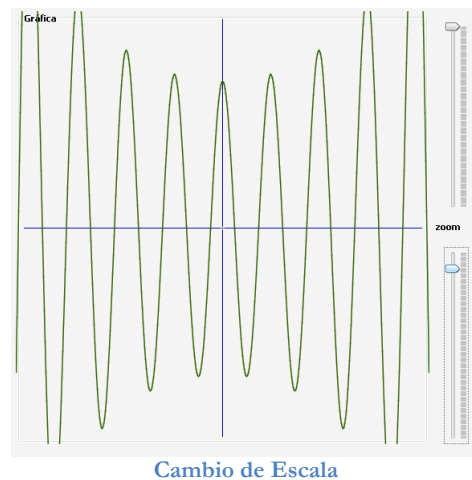


Gráfica Desplazada.

Del punto inicial de arrastre A y el punto final B se redefine el origen (x_0, y_0) de acuerdo al vector $B-A$.



Si el usuario usa los deslizadores (sliders) se ejecuta el cambio de escala en el eje respectivo, (max=200% y min =1 %.)



Cambio de Escala

Algoritmos Redes Neuronales Artificiales Supervisadas

Se escoge cualquiera de las RNAS que se quiera utilizar. La RNAS RNA_BP y RNA_FM tienen un mayor tiempo de ejecución, La RNA_GC tiene el menor tiempo de ejecución y RNA_N solo requiere un entrenamiento.

Se requieren escoger uno de los siguientes datos:

- Back-Propagation.
- Factor Momentum
- Gradientes Conjugados
- Gradientes Conjugados Residuales

- Newton
- Newton Truncado



Algoritmos Redes Neuronales Artificiales Supervisadas

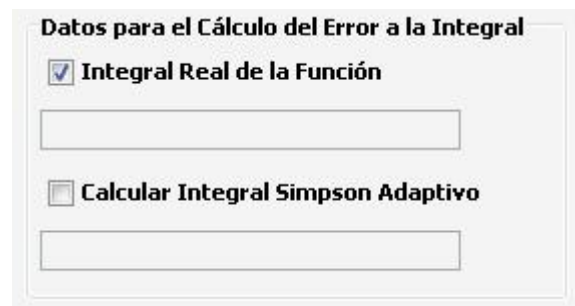
Para mayor información sobres las RNAS ir al capítulo 4 de esta tesis.

Datos para el Cálculo del error de la Integral

Se tienen dos formas de proporcionar una integral real si se conoce, la primera se puede introducir el valor real manualmente y la segunda se usa una aproximación utilizando el método de Simpson Adaptivo. Esto nos sirve para verificar que tan precisos son los resultados con respecto a la integral real o una aproximación con un método de integración clásico.

Se requieren los siguientes datos:

- Integral Real de la Función
- Calcular Integral Simpson Adaptivo



Datos para el Cálculo del Error de la Integral

Tolerancia para el Cálculo de la Integral

Se tienen tres formas de calcular la integral definida, la primera se puede introducir tolerancia a la integral¹, la segunda se usa una tolerancia para la aproximación a la función y la tercera es proporcionando el número máximo de entrenamientos para los algoritmos de las seis RNAS.

Se requieren escoger solo uno de los siguientes datos:

- Tolerancia a la Integral

¹ Si no se proporciona la integral real, esta opción falla.

- Tolerancia a la Función
- Entrenamientos

Datos para el Cálculo de la Integral

☐ Tolerancia a la Integral

☒ Tolerancia a la Función

☐ Entrenamientos

Datos para el cálculo de la Integral

Resultados

Para poder obtener los resultados correctos es necesario ingresar los datos como se indico anteriormente. Los resultados mostrados son Valor aproximado a la integral, Error a la aproximación a la integral, error a la aproximación a la función y número de entrenamientos. Hay tres formas de ejecutar, una dando clic en el botón ejecutar, en el menú Evaluar->Ejecutar o la tecla F5. Mostrando al final el tiempo de ejecución del algoritmo seleccionado.

Resultados

Valor Aproximado a la Integral

Error Aproximación a la Integral

Error Aproximación a la Función

Numero de Entrenamientos

Resultados Obtenidos

Ejemplos

Hay ejemplos dentro de la aplicación, solo cargan los datos para poder ejecutar los algoritmos de las RNAS, los ejemplos cuentan con la integral real en los intervalos dados de cada función. Si se modifican los datos del intervalo y la función es necesario proporcionar la integral real o utilizar el método de Simpson Adaptivo para obtener una aproximación si se desconoce dicha integral.

Ejemplos de Funciones Sencillas

Ejecutar

Evaluar F5

Ejemplos Simples ▶

Ejemplos Funciones Oscilantes ▶

Ejemplos Funciones con Singularidad ▶

x^2

x^4

$\sin(x)$

e^x

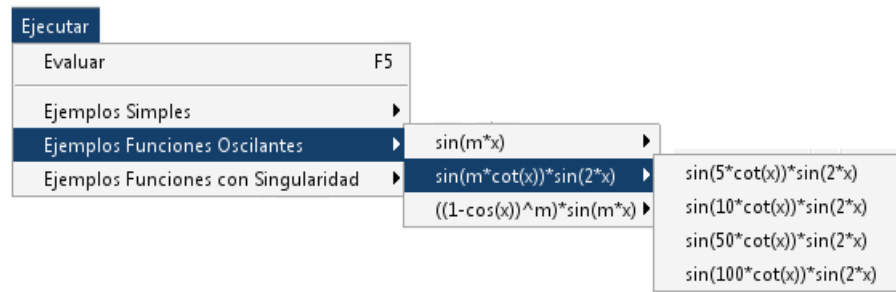
$1/(x+1)$

$\sqrt{1+x^2}$

$\sqrt{1-\cos(x)^2}$

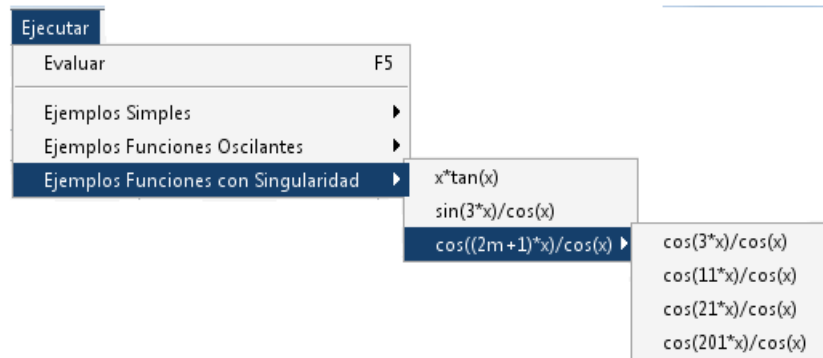
Funciones Sencillas

Ejemplo de Funciones Oscilantes

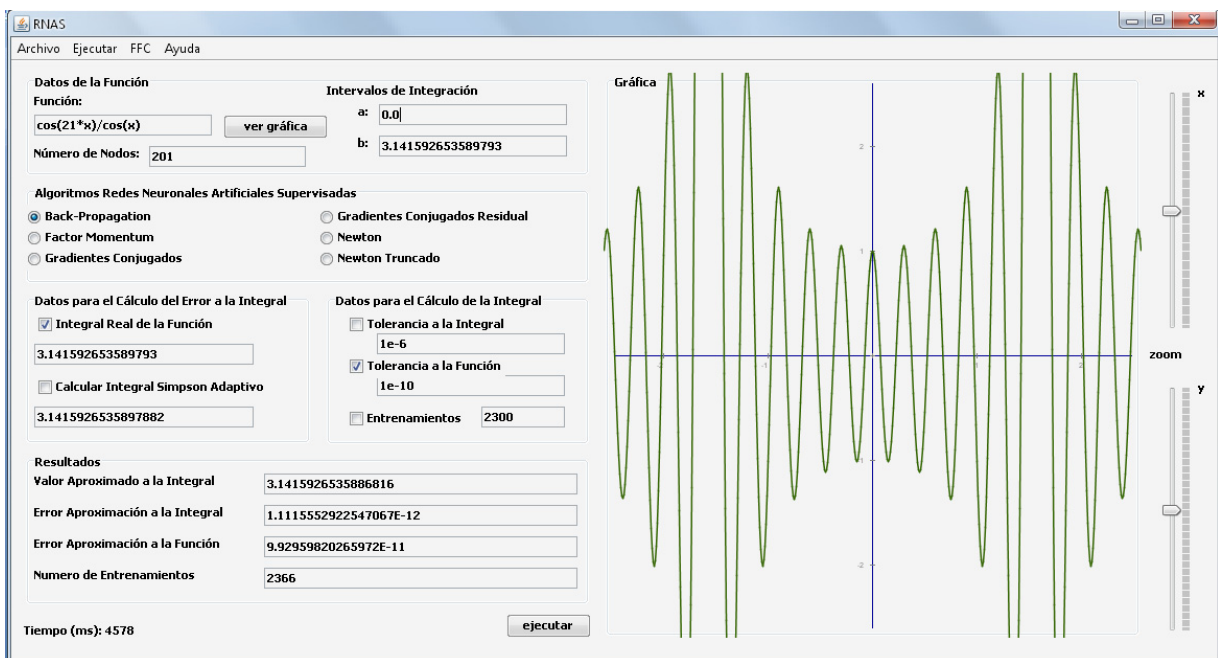


Funciones Oscilantes

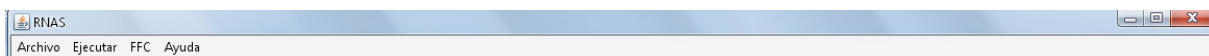
Ejemplo de Funciones Oscilantes con Singularidad



Funciones Oscilantes con Singularidad

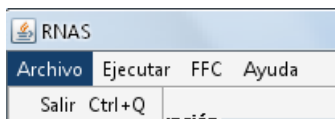
Ejemplo $F = \cos(21 * x)/\cos(x)$ Funciones Oscilantes con Singularidad $F = \cos(21 * x)/\cos(x)$

Dentro de la aplicación se tiene una barra de menús para acceder a las demás aplicaciones.



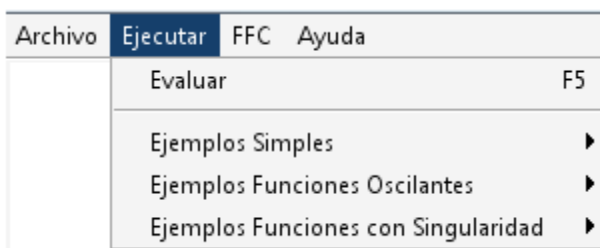
Barra de menús

Para salir de la aplicación se tiene el menú Archivo->Salir o haciendo la combinación Ctrl-Q.



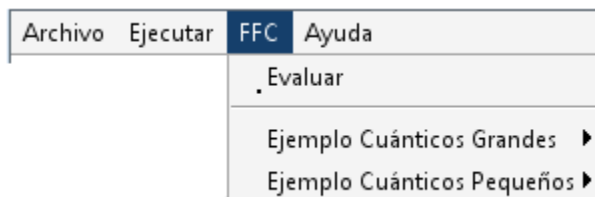
Salir

Para ejecutar y mostrar los ejemplos de la Aplicación se tiene el menú Ejecutar



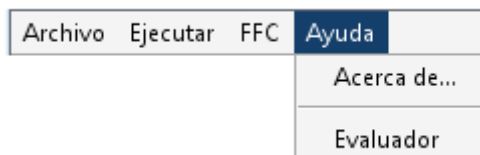
Ejecutar

Para entrar al formulario para el cálculo de los FFC se tiene el menú FFC, para ejecutarlo es con la opción FFC->Ejecutar o pulsando la tecla F6.



FFC

Se tiene un formulario para mostrar la forma de escribir una función y un formulario acerca de quién realizó el trabajo.



Ayuda

Formulario Factores Franck-Condon

Este formulario presenta el cálculo para los FFC con las seis diferentes RNAs. La ventana principal de FFC se divide en 5 partes las cuales son:

Se requieren escoger alguno de los siguientes datos:

- Nombre del Sistema de Bandas
- Números Cuánticos
- Constantes
- Algoritmos Redes Neuronales Supervisadas
- Datos Generados

FFC: RNAS v.1.0

Sistema de Bandas:

Números Cuánticos

	inicio	fin
v1		
v2		

Constantes

	Estado 1	Estado 2
we		
we_xe		
re		

miu

Algoritmos Redes Neuronales Artificiales Supervisadas

☒ Back-Propagation

☐ Factor Momentum

☐ Gradientes Conjugados

☐ Gradientes Conjugados Residual

☐ Newton

☐ Newton Truncado

guardar evaluar

Datos Generados

Tiempo (ms):

FFC

Nombre

Se requiere un nombre para saber con qué molécula se trabaja, que posteriormente se requiere si se guardan los resultados generados.

Sistema de Bandas:

Nombre

Limites

Se requieren los límites inferior y superior de los números cuánticos vibracionales de los dos estados de la molécula para los cuales se calcularán los factores Franck-Condon, donde deben ser mayores a cero y los límites superiores no deben ser menores que los límites inferiores.

Números Cuánticos		
	inicio	fin
v1		
v2		

Límites números cuánticos

Constantes

Se requieren los datos para los estados de la molécula para la cual se está trabajando, donde deben ser mayores a cero y caracteres numéricos.

Constantes		
	Estado 1	Estado 2
we		
we_xe		
re		
miu		

Constantes

donde:

we es la frecuencia armónica en cm^{-1}

we_{xe} es la frecuencia anarmónica en cm^{-1}

re es la distancia de equilibrio en angstroms

$miu(\mu)$ es la masa reducida $m = \frac{m_1 m_2}{m_1 + m_2}$ en u.m.a.

m_1 es la masa atómica del átomo 1 en u.m.a.

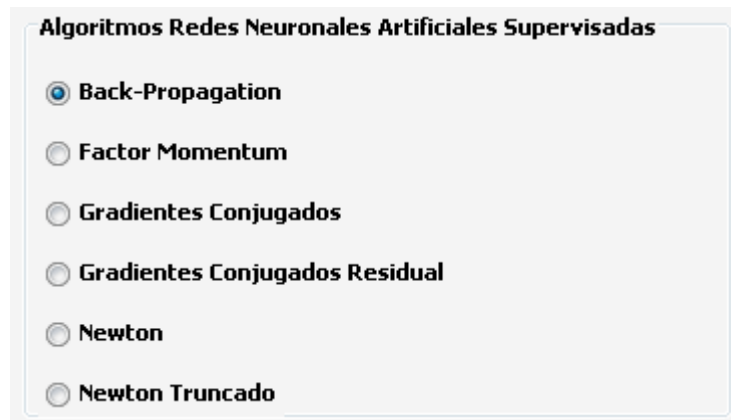
m_2 es la masa atómica del átomo 2 en u.m.a.

Algoritmos

Se escoge cualquiera de las RNAS que se quiera utilizar. La RNA BP y RNA FM tienen un mayor tiempo de ejecución, La RNA GC tiene el menor tiempo de ejecución y RNA N solo requiere un entrenamiento.

Se requieren escoger solo uno de los siguientes algoritmos:

- Back-Propagation.
- Factor Momentum
- Gradientes Conjugados
- Gradientes Conjugados Residuales
- Newton
- Newton Truncado



Algoritmos Redes Neuronales Artificiales Supervisadas

☒ Back-Propagation

☐ Factor Momentum

☐ Gradientes Conjugados

☐ Gradientes Conjugados Residual

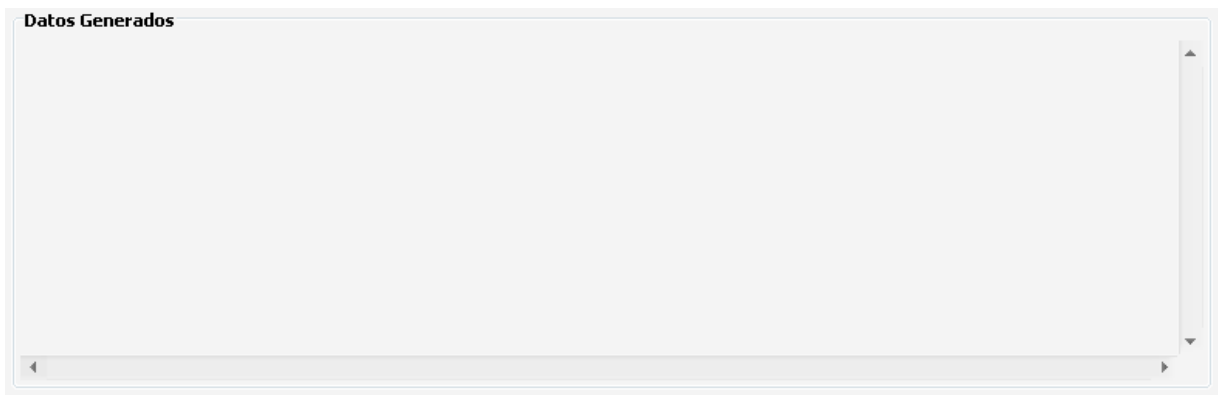
☐ Newton

☐ Newton Truncado

Algoritmos Redes Neuronales Artificiales Supervisadas

Datos

Para poder obtener los resultados correctos es necesario insertar los datos como se menciona antes. Al dar clic el botón ejecutar se generara una tabla que mostrara los resultados obtenidos y el tiempo de cálculo. Al dar clic al botón guardar se genera un archivo con todos los datos anteriores.



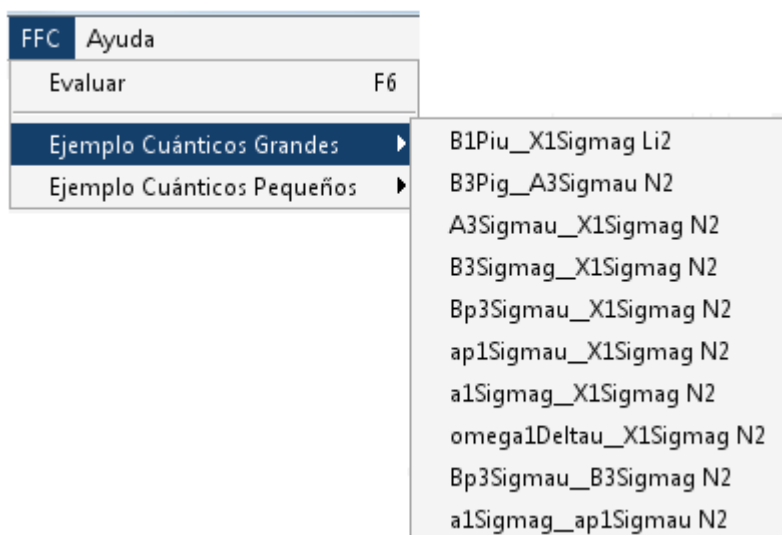
Datos Generados

Tabla de Resultados

Ejemplos

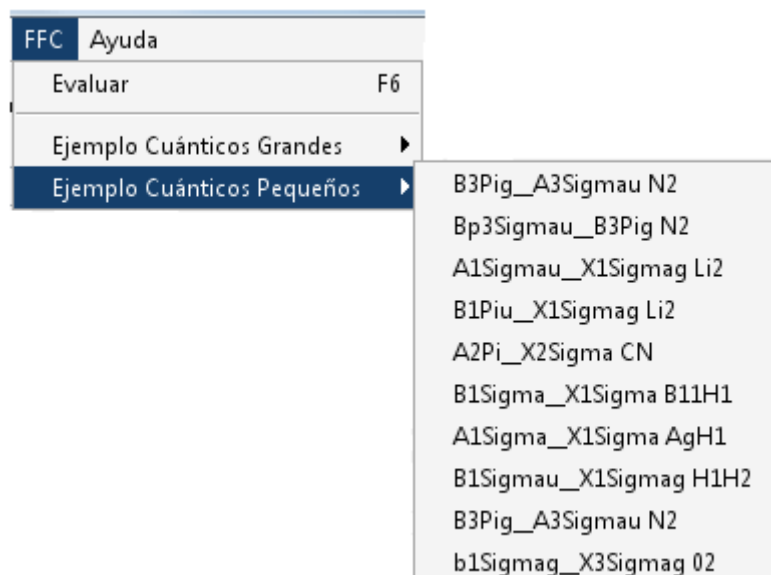
Hay ejemplos dentro de la aplicación, solo cargan los datos y generan la tabla de resultados usando el algoritmo de Gradientes Conjugados, los ejemplos cuentan con datos experimentales, si se modifican los datos será necesario volver a calcular la tabla de resultados.

Se tiene un conjunto de ejemplos con números cuánticos grandes que se accede desde la ventana principal del sistema, la cual carga con los datos experimentales y genera la tabla de resultados.



Ejemplo Números Cuánticos Grandes

Se tiene un conjunto de ejemplos con números cuánticos pequeños que se accede desde la ventana principal del sistema, la cual cargar con los datos experimentales y genera la tabla de resultados.



Ejemplo Números Cuánticos Pequeños

Como ejemplo se desea calcular a los FFC para el Sistema de bandas $B^1\Pi_u \rightarrow X^1\Sigma_g^+$ del ${}^7Li_2$

Los datos experimentales con los que se hace la comparación del sistema de bandas de esta molécula (Abad, 1991), son los siguientes:

	$B^1\Pi_u$	$X^1\Sigma_g^+$
w_e	269.69	351.43
$w_e x_e$	2.744	2.592
r_e	2.936	2.672
μ	3.50908	

Utilizando la aplicación se tienen los siguientes resultados (ver Pruebas y Resultados Tabla 5-15)

FFC-RNA Gradientes Conjugados Sistema de bandas $B^1\Pi_u \rightarrow X^1\Sigma_g^+$ del ${}^7Li_2$

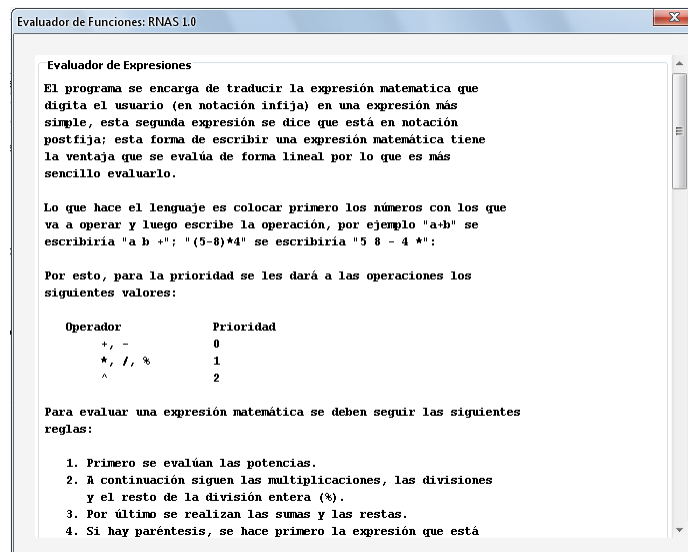
Al generar el archivo de texto se tiene el siguiente resultado, el nombre del archivo generado es B1Piu_X1Sigmag Li2, el cual se toma de lo que se coloca en el Sistema de Bandas que se está trabajando, y este se guarda en el directorio donde se ejecuta la aplicación (..\RNAS\).

Sistema de Bandas B1Piu__X1Sigmag Li2		
+++++		
+++++		
Costantes		
	Estado 1	Estado 2
we	269.69	351.43
we_xe	2.744	2.592
re	2.936	2.672
miu	3.50908	
+++++		
Factores Franck-Condon con Integración con Redes Neuronales Artificiales Supervisadas		

v2 v1	0	1	2	3	4	5	6	7	8	9
0	3,183E-01	3,346E-01	2,014E-01	9,195E-02	3,559E-02	1,240E-02	4,025E-03	1,248E-03	3,763E-04	1,092E-04
1	3,835E-01	7,716E-03	9,526E-02	1,902E-01	1,592E-01	9,215E-02	4,344E-02	1,800E-02	6,839E-03	2,465E-03
2	2,105E-01	1,525E-01	1,342E-01	1,396E-05	7,478E-02	1,417E-01	1,275E-01	8,164E-02	4,311E-02	2,004E-02
3	6,957E-02	2,720E-01	7,127E-03	1,543E-01	5,125E-02	3,905E-03	7,089E-02	1,166E-01	1,051E-01	7,098E-02
4	1,538E-02	1,641E-01	1,856E-01	2,899E-02	7,600E-02	1,044E-01	1,541E-02	1,142E-02	6,913E-02	1,005E-01
5	2,373E-03	5,374E-02	2,173E-01	6,166E-02	1,122E-01	1,023E-02	9,804E-02	5,592E-02	2,840E-03	1,730E-02
6	2,404E-04	1,048E-02	1,000E-01	1,828E-01	1,523E-04	1,584E-01	8,687E-03	5,977E-02	7,722E-02	2,397E-02
7	6,842E-06	8,844E-04	2,008E-02	1,081E-01	7,625E-02	4,542E-02	1,000E-01	8,411E-02	1,137E-02	8,877E-02
8	4,750E-06	3,106E-05	2,457E-04	1,387E-02	5,916E-02	7,363E-03	8,864E-02	9,144E-03	1,328E-01	1,599E-02
9	1,716E-05	3,897E-04	2,437E-03	3,062E-03	1,805E-03	2,541E-02	5,681E-06	5,596E-02	5,031E-03	6,248E-02

Formulario de Ayuda

La ventana de ayuda es una Área de Texto con algunas indicaciones acerca de la sintaxis de las funciones en el Analizador Sintáctico.



Ventana de Ayuda

Formulario Acerca de...

Se tiene un formulario para mostrar la forma de escribir una función y un formulario acerca de quién realizó el trabajo.



Acerca de..

Clases Usadas y Archivos Generados

Package rnas

Class Summary	
<u>Archivo</u>	Clase que guardar el archivo de texto.
<u>AyudaEvaluador</u>	Clase que contiene la ventana de Ayuda para el Analizador Sintáctico.
<u>CambioVariable</u>	Clase que hace el cambio de Variable de las funciones para su ajuste.
<u>FFC_RNAS</u>	Clase principal para el cálculo de los FFC.
<u>FuncionOnda</u>	Clase para el cálculo de la Función de Onda para el Potencial de Morse.
<u>Globales</u>	Clase que maneja los datos generales de la Aplicación.
<u>Parseador</u>	Clase que contiene el Analizador Sintáctico.
<u>PrintfFormat</u>	Clase que escribe con formato para números.
<u>RNA_BP</u>	Clase que contiene la RNA Back-Propagation.
<u>RNA_FM</u>	Clase que contiene la RNA Factor Momentum.
<u>RNA_GC</u>	Clase que contiene la RNA Gradientes Conjugados.
<u>RNA_GCR</u>	Clase que contiene la RNA Gradientes Conjugados Residuales.
<u>RNA_N</u>	Clase que contiene la RNA Newton.
<u>RNA_NT</u>	Clase que contiene la RNA Newton Truncado.
<u>RNASAboutBox</u>	Clase que contiene los datos de Presentación.
<u>RNASApp</u>	Clase Principal de la Aplicación.
<u>RNASView</u>	Clase que contiene la ventana principal de la Aplicación.
<u>Simpson</u>	Clase que calcula el método de Simpson Adaptivo.

Referencias

- Abad, I. L.** (1991). Factores Franck-Condon para el Potencial de Morse. Puebla, Puebla: Benemerita Universidad Autonoma de Puebla.
- Churyumov, K. I., Bereznoi, A., Chavushyan, V. H., Sandoval, L., & Palma, A.** (2007). Franck-Condon Factors for Molecules Observed in Comets. 2650-2653.
- Churyumov, K. I., Luk'yanyk, I. V., Bereznoi, A., Chavushyan, V. H., Sandoval, L., & Palma, A.** (2002). Optical spectroscopy of comet C/2000 WM1 (LINEAR) at the Guillermo Haro Astrophysical Observatory in México. 90, 361-368.
- Churyumov, K. I., Luk'yanyk, I. V., Bereznoi, A., Chavushyan, V. H., Sandoval, L., Palma, A., y otros.** (2005). Exploration of Spectra of Periodic Comet 153P/IKEYA-ZHANG. Kinematics and Physics of Celestial Bodies Suppl. 472-476.
- Churyumov, K. I., Luk'yanyk, I. V., Bereznoi, A., Chavushyan, V. H., Sandoval, L., Palma, A., y otros.** (2005). Identification of Emission Lines in the spectrum of comet C/2002 V1 (NEAT). Astronomical Schools's Report , 111-114.
- Churyumov, K. I., Luk'yanyk, I. V., Bereznoi, A., Chavushyan, V. H., Sandoval, L., Palma, A., y otros.** (2005). Peculiarities of Spectra of Comet C/2002 T7 (LINEAR) in January 2004. 2, 104-106.
- Churyumov, K. I., Luk'yanyk, I. V., Bereznoi, A., Chavushyan, V. H., Sandoval, L., Palma, A., y otros.** (2005). Physical Parameters of the comet C/2002 C1 (Ikeya-Zhang) Atmosphere. 95-98.
- Churyumov, K. I., Luk'yanyk, I. V., Bereznoi, A., Chavushyan, V. H., Sandoval, L., Palma, A., y otros.** (2003). Spectral Observations of comet C/2000 WM1 (LINEAR) in Mexico. 22, 1-5.
- Zeng, Z.-Z. W., & Yao-Nan, W. H.** (Jul/Aug 2006). Numerical Integration Based on a Neural Network Algorithm. 8 (pp. 42-48 7).
- Al-Haik, M., Garmestani, H., & Navon, I.** (2003). Truncated-Newton Training Algorithm for Neurocomputational Viscoplastic Model. Computer Methods in Applied Mechanics and Eng , 2249-2267.
- Caliskan, F., Aykan, R., & Hajiye, C.** (2008). Aircraft Icing Detection, Identification, and Reconfigurable Control Based on Kalman Filtering and Neural Networks. Journal of Aerospace Engineering, 51-60.
- Cheng, C.-H., Chen, T.-L., & Wei, L.-Y.** (2010). A hybrid model based on rough sets theory and genetic algorithms for stock price forecasting. Information Sciences , 1610-1629.
- Khare, V., Santhosh, J., Anand, S., & Bhatia, M.** (2010). Performance comparison of three artificial neural network methods for classification of electroencephalograph signals of five mental tasks. Journal of Biomedical Science & Engineering , 200-205.
- Minasny, B., & AB, M.** (2002). The nuro-m method for fitting neural network parametric pedotransfer functions. Web of Science , 352-361.
- Moller, M.** (1993). A Scaled Conjugate-Gradient Algorithm for Fast Supervised Learning. Neural Networks , 525-533.
- Montaño Moreno, J. J.** (2002). Redes Neuronales Artificiales aplicadas al Análisis de Datos. Universitat De Les Illes Balers, Facultad de Psicología.

Polat, Ö., & Yildirim, T. s. (2010). FPGA implementation of a General Regression Neural Network: An embedded pattern classification. *Digital Signal Processing* , 881-886.

Unidad de Investigación en Automatización Industrial, Departamento de Sistemas y Automatica, Escuela de Ingeniería Eléctrica, Universidad de Carabobo, Valencia Venezuela. (2004). Aplicación de Redes Neuronales artificiales en la predicción de la curva de destilación ASTM D86 en gasolinas automotrices. *Revista de Ingeniería* , 27-34.

XU, L., Oja, E., & Suen, C. (1992). Modified Hebbian Learning for Curve and Surface Fitting. *Web of Science* , 441-457.

Burden, R. L., & Douglas Faires, J. (2002). *Numerical Analysis* (Vol. 7). Thomson Learning.

Domínguez Báuená, V., & Rapú Banzo, M. L. (2006). *Matlab en cinco lecciones de Numérico*. Universidad Pública de Navarra.

A Series of Books in Mathematics R.A. Rosenbaum G. Philip Johnson. (1963). *Computational Methods of Linear Algebra*. London: W.H. Freeman and Company San Francisco London.

Al-Haik, M., Garmestani, H., & Navon, I. (2003). Truncated-Newton Training Algorithm for Neurocomputational Viscoplastic Model. *Computer Methods in Applied Mechanics and Eng* , 2249–2267.

Escudero, I. F. (1982). On Diagonally-Preconditioning the Truncated-Newton Method for Super-Scale Linearly Constrained Nonlinear Programming. *Qtiesttió* , 261-281.

González Velázquez, J. A. (1992). *Sistema Integral del Gradiente Conjugado*. Puebla: Tesis Profesional, Benemérita Universidad Autónoma de Puebla.

Luenbeger, D. E. (1989). *Linear and nonlinear programming*. Addison-wesleyIberoamericana .

Gradshteyn, I. S., & M., R. I. (2007). *Table of integrals, series and products* (Vol. 7). Elsevier.

NIST Standard Reference Data is governed by the Standard Reference Data Act. . (2008). Libro del Web de Química del NIST. Recuperado el Marzo de 2010, de Base de Datos de Referencia Estándar del NIST Número 69: <http://webbook.nist.gov/chemistry/form-ser.html>

Kuzmenko, N. E., Kuznetsova, L. A., & Kuziakov, I. I. (1984). Franck-Condon factors of diatomic molecules. Moscow, Izdatel'stvo Moskovskogo Universiteta, In Russian , 344 p.